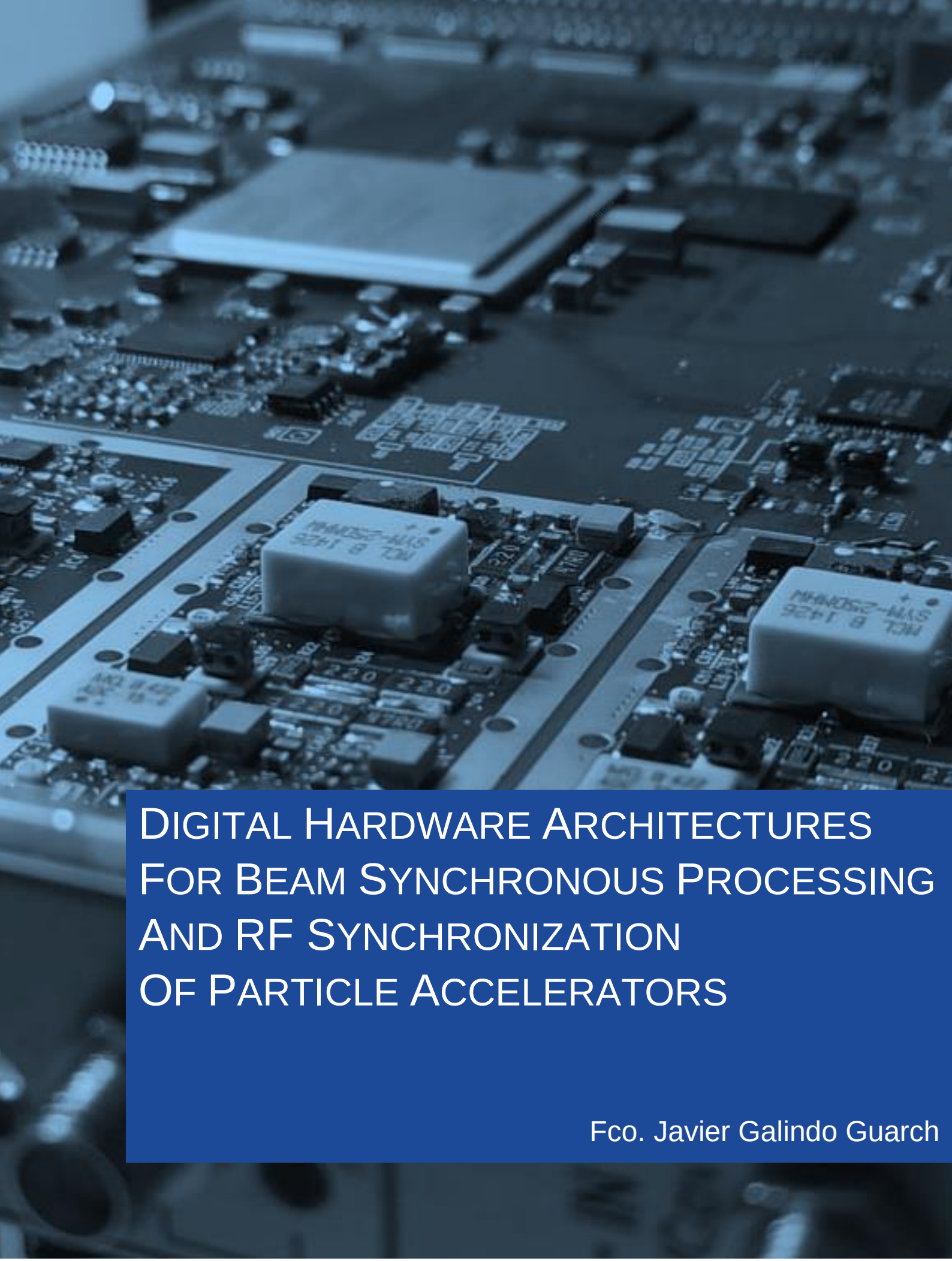


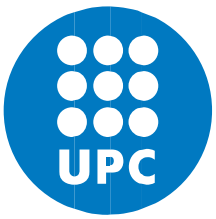
Digital Hardware Architectures For Beam Synchronous Processing And RF Synchronization Of Particle Accelerators

Fco. Javier Galindo Guarch

PhD in Electronic Engineering
Barcelona, 2021

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

CERN



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Departament d'Enginyeria Electrònica

DIGITAL HARDWARE ARCHITECTURES FOR BEAM SYNCHRONOUS PROCESSING AND RF SYNCHRONIZATION OF PARTICLE ACCELERATORS

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE PHD DEGREE ISSUED BY THE UNIVERSITAT POLITECNICA DE CATALUNYA, IN ITS ELECTRONIC ENGINEERING PROGRAM.

FRANCISCO JAVIER GALINDO GUARCH

DIRECTORS:
JUAN MANUEL MORENO AROSTEGUI, UPC
PHILIPPE BAUDRENGHIEN, CERN

Barcelona, Spain February, 2021

A mis Padres,
a mis Abuelos.

Agradecimientos

Tengo que agradecer, en primer lugar, a mis directores J. Manuel Moreno y Philippe Baudrenghien su apoyo y orientación durante los años en los que este trabajo ha tenido lugar. Manuel, gracias por tu infinita paciencia conmigo y tus acertados consejos a los que tendría que haber hecho más caso. Et Philippe, ton soutien, implication au-delà du devoir, exemple… merci!

Quiero a continuación dedicar esta Tesis, que culmina muchos años de trabajo y esfuerzo de mi vida, a mis abuelos, Mercedes, Joaquina, Francisco y Francisco, y a mis padres, Paquita y Vicente, gracias por estar siempre a mi lado, aun cuando a veces nos separen cientos de kilómetros de distancia... también a Diana y Maya, por vuestro inestimable apoyo, escucha y cariño durante estos años.

Deseo expresar mi gratitud a todos aquellos que, de una manera u otra, han contribuido también a que este trabajo haya alcanzado su fin, mis familiares, mis compañeros en Suiza, Holanda, España… En el CERN, en el grupo de RF... Jorge Sánchez, Juan Carlos Allica, Natalia Galindo, Silvia Aguilera, David Cabrerizo, Jordi Ustrell, Endre Bjorsvik, Javier Llorente, Ricardo Hernández, Carolina Belver, Rubén Lorenzo, Miguel Ojeda, Diego Barrientos, Alejandro Díaz, Sergio Calvente, Jorge Flores, Rubén García, Enrique de Nicolás, Nuria Ayala, Alicia del Barrio, Iñaki Ortega, Luis E. Fernández, Alvaro Ferrero, Pablo Prieto, Jesús Cortés, Lorena Vega, Juan Esteban, Gregoire Hagmann, Jose Noirjean, Predrag Kuzmanovic, Tom Levens, Tomasz Wlostowski, Javier Serrano, Michael Jaussi, Robert Borner, Heiko Damerau, Vebjorn Myklebust, Michal Husejko, Eleanor Davies, Fathia Saidi, Themis Mastoridis… Me gustaría expresar mi gratitud a Wolfgang Hofle y Erk Jensen, por apoyar y albergar esta Tesis dentro del grupo de RF.

Sería imperdonable no demostrar mi agradecimiento a todos mis amigos dispersos por el mundo, entre ellos a Pablo T. Chinea, Jaime Crespo, Jorge Solana, Oriol Lluch, Idoia Palicio, Manuel Blanco… a los del pueblo, La Fresneda, y a los de Zaragoza, Santi, Alberto, Alberto, Alex, David, Javi, Kiko, J. Miguel… y especialmente a Ismael Bel quien tanto me ha escuchado. Por supuesto también a Andrea, Fernando, Yolanda y Fernando.

Mi agradecimiento para Herbert Shea, Arturo Mediano, Alfonso Muñoz y Antonio Agudo por descubrirme que es la Investigación y el rigor académico. Antes de terminar, no quiero olvidar a mis compañeros de la universidad donde todo empezó mucho tiempo atrás, J. Ignacio Gimeno, Jesús Gonzalvo, Andrés Grande, Antonio Oteo, Diego Pérez, Sergio Sanz y Jesús Velázquez.

¡A todos, Gracias!

Acknowledgements

I would like to express my gratitude in in first place to my Thesis directors, J. Manuel Moreno and Philippe Baudrenghien for their support and guidance during these years. Thankyou Manuel for your endless patience and wise advice that I should follow more often. Et Philippe, ton soutien, implication au-delà du devoir, exemple… merci!

I would like to dedicate this Thesis, that is the result of many years of devoted work in my life, to my grandparents, Mercedes, Joaquina, Francisco and Francisco, and to my parents, Paquita and Vicente, thank you for being always available and close to me, even when sometimes we are separated by a long distance... and to Diana and Maya, for your invaluable support, for listening to me, and your affection during these years.

I would like to seize this opportunity to thank all of you that, in one way or another, have contributed to the fruitful end of this work, my family, my colleagues in Switzerland, Holland, Spain… at CERN, the RF group… Jorge Sánchez, Juan Carlos Allica, Natalia Galindo, Silvia Aguilera, David Cabrerizo, Jordi Ustrell, Endre Bjorsvik, Javier Llorente, Ricardo Hernández, Carolina Belver, Rubén Lorenzo, Miguel Ojeda, Diego Barrientos, Alejandro Díaz, Sergio Calvente, Jorge Flores, Rubén García, Enrique de Nicolás, Nuria Ayala, Alicia del Barrio, Iñaki Ortega, Luis E. Fernández, Alvaro Ferrero, Pablo Prieto, Jesús Cortés, Lorena Vega, Juan Esteban, Gregoire Hagmann, Jose Noirjean, Predrag Kuzmanovic, Tom Levens, Tomasz Wlostowski, Javier Serrano, Michael Jaussi, Robert Borner, Heiko Damerau, Vebjorn Myklebust, Michal Husejko, Eleanor Davies, Fathia Saidi, Themis Mastoridis… I would like also to thank Wolfgang Hofle and Erk Jensen, for supporting this work within the RF group.

To all my friends spread around the world, among them Pablo T. Chinea, Jaime Crespo, Jorge Solana, Oriol Lluch, Idoia Palicio, Manuel Blanco… to the ones in La Fresneda, and the ones in Zaragoza, Santi, Alberto, Alberto, Alex, David, Javi, Kiko, J. Miguel … especially to Ismael Bel, who has always had a minute for me. And of course, to Andrea, Fernando, Yolanda and Fernando.

Special thanks also to Herbert Shea, Arturo Mediano, Alfonso Muñoz and Antonio Agudo for discovering me what Research is and the academic rigor. I do not want to forget my folks in the university where all this started year ago, J. Ignacio Gimeno, Jesús Gonzalvo, Andrés Grande, Antonio Oteo, Diego Pérez, Sergio Sanz and Jesús Velázquez.

Thanks to each one of you!

Resumen

En un Acelerador de Partículas, el *Low Level RF* (LLRF) es el sistema encargado del control de la Radio Frecuencia, e implícitamente, de la transferencia de energía y aceleración de las partículas, objetivo último de la máquina. El LLRF implementa algoritmos que sincronizan la transferencia de energía de la RF hacia el haz, así como también la configuración de sus parámetros longitudinales. Para ello, usa señales derivadas del haz, cuyo contenido espectral se ve modificado con la aceleración. El incremento en energía implica un incremento en la velocidad del haz que, en el caso de aceleradores circulares (Sincrotrones) se traduce en un decremento del periodo de revolución. Esto es especialmente relevante para los aceleradores de Hadrones, para los cuales la energía de inyección es baja, lo que resulta en grandes incrementos en su velocidad antes de alcanzar niveles relativistas. El LLRF necesita por tanto sintonizar continuamente el procesado y el haz; hemos llamado a esta técnica *Beam Synchronous Processing*.

Una importante misión del LLRF es la compensación de la tensión inducida por el haz en las cavidades de aceleración (*Beam Loading*). En el sincrotrón SPS del CERN, la regulación tiene especificado un ancho de banda de 5 MHz a cada lado de la RF (200 MHz). Dado que el periodo de revolución es de aproximadamente 23 μs, más de cien harmónicos de la frecuencia de revolución están presentes en cada una de las bandas alrededor de la RF. La variación en velocidad del haz altera la posición y el espaciado de estos harmónicos en el espectro. Su gran número y posición cambiante hace de la reconfiguración de los algoritmos de control una opción poco deseable. Dicha problemática ha sido abordada clásicamente en el mundo de los aceleradores mediante un reloj de sistema derivado de la RF, y por tanto variable, que liga por diseño los procesos de muestreo y procesado al haz. Esta solución histórica, todavía en uso en varias máquinas, es ahora un factor limitante para el uso de nuevas y modernas tecnologías.

Esta Tesis presenta una nueva Arquitectura para procesado síncrono de señales derivadas del haz, mediante un reloj de sistema con frecuencia fija, que hace posible el tratamiento de señales periódicas en las que el harmónico fundamental tiene una frecuencia variable y conocida. La Arquitectura es una alternativa válida al complejo problema de reconfiguración de algoritmos de procesado; esta sintoniza el espectro al procesado mediante el re-muestreo de los datos. Dos Re-muestreadores (Resampler en inglés) son combinados originando el denominada *sándwich* de re-muestreo. El algoritmo de aplicación, el cual requiere sincronismo con la señal de entrada, se sitúa en medio de este sándwich.

El elemento clave que hace esto posible es un novedoso Resampler completamente digital y basado en una arquitectura Farrow, que acepta además relaciones de re-muestreo arbitrarias siendo estas modificables en tiempo real. El hardware usa un reloj de sistema único de frecuencia fija, lo cual facilita su

implementación en FPGAs, ASICs y sistemas de última generación, como los nuevos controladores uTCA que se están implantando en los sistemas LLRF del SPS. Los puertos de entrada y salida del Resampler, y en general el data-path en toda la Arquitectura propuesta, son síncronos a este reloj, y además aceptan una frecuencia de muestreo variable y modificable en tiempo real.

La Arquitectura ha sido implementada y puesta en marcha en uno de estos controladores uTCA para LLRF, albergando el algoritmo *One Turn FeedBack* para control de una cavidad del SPS en el CERN. El algoritmo compensa el *Beam Loading*. La Arquitectura ha demostrado ser viable operando sintonizada en todo momento a una rampa de aceleración de energía del haz, con una RF variable que sigue un patrón en diente de sierra con una tasa de cambio de la frecuencia lineal de 2.4 MHz por segundo. La implementación de la Arquitectura en el controlador uTCA ha pasado toda la validación funcional y los test cualitativos.

La Arquitectura se adapta de manera sin igual a los dos cambios de paradigma tecnológico adoptados por el SPS para sus nuevos sistemas LLRF; primero, la distribución del valor instantáneo de la frecuencia de la RF es ahora hecho mediante una palabra digital (usada para el cálculo de la relación de re-muestreo), empleando una red determinista, en este caso White Rabbit. Y segundo, la señal de referencia es ahora un reloj con frecuencia fija extraído de esta red determinista. La adopción de ambos paradigmas se ve beneficiada por el uso de la Arquitectura Beam Synchronous Processing y del nuevo Resampler compleménteme digital, que satisfacen los requerimientos técnicos y tecnológicos para la implementación de nuevos algoritmos y soluciones en el campo del LLRF.

Palabras clave; *Arquitectura Hardware, FPGA, RF de Baja Señal, Conversión de Tasa de Muestreo, Procesado Digital de Señal, Procesado Adaptivo de Señal, Acelerador de Partículas, Sincrotrón, Beam Loading, One Turn Delay Feedback.*

Abstract

In Particle Accelerators, the *Low-Level RF* (LLRF) is the control system of the RF, and in the end, of the purpose of the machine, that is the energy transfer and acceleration of particles. It implements algorithms synchronizing the RF conveying the energy to the beam and tailoring its longitudinal parameters. For this, the LLRF uses beam-related signals whose spectral content changes during the acceleration. The increase in energy results in an increase of the beam velocity, and for circular accelerators (Synchrotrons) a decrease in revolution period. This is especially relevant for Hadron machines whose injection energy is low resulting in a significant increase of their velocity before reaching relativistic speeds. Hence, the LLRF needs to continuously tune its processing to the beam; we call this technique *Beam Synchronous Processing*.

One important task of the LLRF is the compensation of the beam-induced voltage in the accelerating cavities (*Beam Loading*). In the CERN SPS the regulation bandwidth must cover 5 MHz on each side of the 200 MHz RF. With a beam revolution period around 23 μs more than a hundred revolution frequency harmonics, present in the beam signal, fall in the RF sidebands. The variation in beam velocity changes the position and spacing of the harmonics in the spectrum. The large number of harmonics and their varying positions make the algorithm reconfiguration an undesirable option. To cope with this, the early digital implementations used a system clock derived from the sweeping RF. This locks the sampling and the processing to the beam, by design. This historical solution, that is still in use in several machines, is now a limiting factor for the use of modern technologies.

The Thesis presents a novel Beam Synchronous Processing Architecture, using a fixed frequency clocking, and capable of treating periodic signals with known and varying fundamental frequency. The Architecture is an alternative to the burden of reconfiguration in processing algorithms; it tunes the spectrum to the processing by resampling the input data. Two Resamplers are combined in the so-called resampling *sandwich*. The application algorithm requiring synchronism with the input signal is placed in the middle.

The key element is a novel All-Digital Farrow-based Resampler, that accepts arbitrary resampling ratios that can be modified in real-time. The hardware uses a single fixed frequency system clock, making its implementation feasible in State-Of-the-Art FPGAs, ASICs and systems such as the new uTCA platform currently being deployed in the CERN SPS LLRF system. The input and output ports of the Resampler, and all the processing within the Architecture, are synchronous to this fixed frequency clock and accept data streams whose sampling rate can be variable and modified in real time.

The Architecture has been commissioned in a LLRF uTCA crate hosting the One Turn FeedBack algorithm to control a real SPS cavity. The algorithm compensates the Beam Loading. The Architecture has demonstrated its capability to track in real-time an energy ramp with an RF frequency following a linear sawtooth pattern ramped at 2.4 MHz per second. The complete uTCA implementation has successfully passed all the functional validation and qualitative tests.

The Architecture suits seamless the two technological paradigm changes adopted for the new CERN SPS LLRF system; first, the instantaneous value of the RF frequency is transmitted as a numerical word (used to set the resampling ratio), via a deterministic network, the White Rabbit. And second, the reference signal is now the fixed frequency clock recovered from this network. Both paradigms benefit from the all-digital Resampler and the Beam Synchronous Architecture that fulfil the techniques and technological needs for its implementation enabling novel LLRF algorithms and solutions.

Keywords; *Hardware Architecture, FPGA, Low Level RF, Sampling Rate Conversion, Digital Signal Processing, Adaptive Signal Processing, Particle Accelerator, Synchrotron, Beam Loading, One Turn Delay Feedback.*

Résumé

Dans le monde des Accélérateurs de Particules, le *Low-Level RF* (LLRF) est le système de contrôle de la RF et, in-fine, du transfert d'énergie et de l'accélération des particules. Il met en œuvre des algorithmes synchronisant la RF transférant l'énergie au faisceau et adaptant ses paramètres longitudinaux. Pour cela, le LLRF utilise des signaux liés au faisceau dont le contenu spectral est modifié par l'accélération. L'augmentation d'énergie se traduit par une augmentation de la vitesse du faisceau, et pour les accélérateurs circulaires (Synchrotrons), une diminution de la période de révolution. Cela est particulièrement pertinent pour les machines à Hadrons dont l'énergie d'injection est faible, avec la conséquence d'une augmentation significative de leur vitesse durant l'accélération. Le LLRF doit donc ajuster en permanence son traitement au faisceau ; nous appelons cette exigence *Beam Synchronous Processing*.

Une tâche importante du LLRF est la compensation de la tension induite par le faisceau (*Beam Loading*). Dans le SPS au CERN, la régulation couvre 5 MHz de chaque côté de la RF (200 MHz). Avec une période de révolution autour de 23 µs, plus d'une centaine d'harmoniques de fréquence de révolution, présentes dans le spectre du faisceau, tombent dans la bande +- 5 MHz. La variation de vitesse du faisceau modifie la position et l'espacement des harmoniques dans le spectre. Le grand nombre de raies spectrales et leur position variable font de la reconfiguration de l'algorithme une option indésirable. Les solutions digitales existantes ont donc préféré changer l'horloge d'échantillonnage : Celle-ci est verrouillée sur la RF, ce qui synchronise par conception l'échantillonnage et le traitement du faisceau. Cette solution historique, toujours en usage dans plusieurs machines, est aujourd'hui un facteur limitant pour les technologies modernes.

La Thèse présente une nouvelle Architecture de traitement synchrone de faisceau, utilisant une horloge fixe, et capable de traiter des signaux périodiques de fréquence fondamentale connue et possiblement variable. L'Architecture apporte une alternative au fardeau de la reconfiguration dans les algorithmes ; il ajuste le spectre au traitement en rééchantillonnant les données d'entrée. Deux Ré-échantillonneurs ont été combinés dans le *sandwich* de rééchantillonnage. L'algorithme d'application nécessitant un synchronisme avec le signal d'entrée est placé au milieu.

L'élément clé est un nouveau Ré-échantillonneur entièrement numérique basé sur une architecture Farrow, qui accepte des taux de rééchantillonnage arbitraires pouvant également être modifiés en temps réel. L'implémentation utilise une seule horloge système à fréquence fixe, ce qui rend sa mise en œuvre possible dans les FPGA, ASIC et systèmes de pointe comme la nouvelle plate-forme uTCA actuellement déployée dans le SPS du CERN. L'entrée et la sortie du Ré-échantillonneur, et tout le traitement dans

l'Architecture, sont synchrones avec cette horloge et acceptent un taux d'échantillonnage variable que peut être modifiée en temps réel.

L'Architecture a été déployée dans un châssis uTCA hébergeant l'algorithme *One Turn FeedBack* pour contrôler une véritable cavité SPS. L'algorithme compense le *Beam Loading*. L'Architecture a démontré sa capacité à suivre en temps réel une rampe d'énergie avec une fréquence RF suivant une modulation en dent de scie, à 2.4 MHz par seconde. L'implémentation complète sur uTCA a passé avec succès les tests de validation fonctionnelle et qualitative.

L'Architecture convient parfaitement aux deux paradigmes technologiques adoptés pour le nouveau système LLRF du SPS ; premièrement, la valeur instantanée de la fréquence RF est transmise sous forme de mot numérique (qui donnera le taux de rééchantillonnage), via un réseau déterministe, le White Rabbit. Et deuxièmement, le signal de référence est maintenant l'horloge à fréquence fixe récupérée de ce réseau. La solution présentée respecte ces deux paradigmes grâce au Ré-échantillonneur entièrement numérique et à l'horloge fixe.

Mots-clés ; *Architecture Hardware, FPGA, RF à Faible Signal, Conversion de Taux de Echantillonnage, Traitement Numérique du Signal, Traitement Adaptatif du Signal, Accélérateur de Particules, Synchrotron, Beam Loading, One Turn Delay Feedback.*

Resum

En un Accelerador de Partícules, el *Low Level RF* (LLRF) és el sistema encarregat de controlar la Radiofreqüència, i implícitament, de la transferència d'energia i acceleració de les partícules, l'objectiu final de la màquina. El LLRF implementa algoritmes que sincronitzen la transferència d'energia RF al feix, així com la configuració dels seus paràmetres longitudinals. Per a això, utilitza senyals derivats del feix, el contingut espectral del qual es modifica amb acceleració. L'augment de l'energia implica un augment de la velocitat del feix que, en el cas dels acceleradors circulars (Sincrotró) es tradueix en una disminució del període de revolució. Això és especialment rellevant per als acceleradors d'Hadrons, per als quals l'energia d'injecció és baixa resultant en grans augments de velocitat abans d'arribar a nivells relativistes. Per tant, el LLRF necessita sintonitzar contínuament el processament amb l'espectre del feix; hem anomenat aquesta tècnica *Beam Synchronous Processing*.

Una missió important del LLRF és la compensació de la tensió induïda pel feix a les cavitats d'acceleració (*Beam Loading*). En el sincrotró SPS del CERN, la regulació té un ample de banda de 5 MHz especificat a cada costat de la RF (200 MHz). Atès que el període de revolució és d'aproximadament 23 μs, més d'un centenar d'harmònics de la freqüència de revolució estan presents en cadascuna de les bandes al voltant de la RF. La variació en la velocitat del feix canvia la posició i l'espaiat d'aquests harmònics en l'espectre. El seu gran nombre i posició canviant fa que la reconfiguració dels algoritmes de control sigui una opció indesitjable. Aquest problema ha estat abordat històricament amb un rellotge de sistema derivat del RF i, per tant, variable, que lliga per disseny els processos de mostreig i processament al feix. Aquesta solució històrica, encara en ús en diverses màquines, és ara un factor limitant per a l'ús de les noves i modernes tecnologies.

Aquesta Tesi presenta una nova Arquitectura per al tractament síncron dels senyals derivats del feix, utilitzant un rellotge de sistema amb freqüència fixa, el que fa possible el tractament de senyals periòdics en què els quals l'harmònic fonamental té una freqüència variable i coneguda. L'Arquitectura és una alternativa vàlida al complex problema de reconfiguració d'algoritmes de processament; sintonitza l'espectre al processament mitjançant el re-mostreig de les dades. Dos re-mostrajadors (Resampler en anglès) es combinen originant l'anomenat *sandvitx* de re-mostreig. L'algoritme d'aplicació, que requereix sincronització amb el senyal d'entrada, es troba al mig d'aquest sandvitx.

L'element clau que ho fa possible és un nou Resampler totalment digital basat en una arquitectura Farrow, que també accepta relacions arbitràries de re-mostreig que són modificables en temps real. El hardware utilitza un únic rellotge de sistema de freqüència fixa, que fa possible la implementació en FPGAs

d'última generació, ASICs i sistemes tals que els nous controladors uTCA que s'estan desplegant en els sistemes LLRF del SPS. Tan els ports d'entrada i sortida del Resampler, com tot el processat dins aquesta Arquitectura son síncrons amb aquest rellotge de sistema de freqüència fixa i accepten senyals amb freqüència de mostreig que pot ser variable i es pot modificar en temps real.

L'Arquitectura s'ha implementat en un controlador uTCA per LLRF, el qual conté l'algoritme *One Turn FeedBack* per controlar una cavitat SPS del CERN. L'algoritme compensa el *Beam Loading*. L'Arquitectura ha demostrat ser viable operant sintonitzada en tot moment a una rampa d'acceleració d'energia de feix, amb una RF variable seguint un patró de serra amb una velocitat de canvi de freqüència de 2,4 MHz per segon. La implementació de l'Arquitectura en el controlador uTCA ha superat totes les proves de validació funcional i qualitativa.

L'Arquitectura s'adapta d'una manera com cap altra als dos canvis de paradigma tecnològic adoptats pel SPS per als seus nous sistemes LLRF; en primer lloc, la distribució del valor instantani de la freqüència RF es realitza ara mitjançant una paraula digital (utilitzada per al càlcul de la relació de re-mostreig), utilitzant una xarxa determinista, en aquest cas White Rabbit. I en segon lloc, el senyal de referència és ara un rellotge de freqüència fixa extret d'aquesta xarxa determinista. L'adopció d'ambdós paradigmes es beneficia de l'ús de l'Arquitectura Beam Synchronous Processing i del nou Resampler digital, que compleixen els requisits tècnics i tecnològics per a la implementació de nous algoritmes i solucions en l'àmbit de la LLF.

Paraules clau; *Arquitectura Hardware, FPGA, RF de Senyal Feble, Conversió de Taxa de Mostreig, Processat Digital de Senyal, Processat Adaptatiu de Senyal, Accelerador de Partícules, Sincrotró, Beam Loading, One Turn Delay Feedback.*

List of Figures

List of Tables

List of Equations

List of Abbreviations

ADC	Analog to Digital Converter
AMC	Advanced Mezzanine Card
ANC	Adaptive Noise Cancelling
ASIC	Application Specific Integrated Circuit
ASRC	Asynchronous Sampling Rate Conversion
BAP	Beam Asynchronous Processing
BNL	Brookhaven National Laboratory
BSP	Beam Synchronous Processing
BW	BandWidth
CERN	Conseil Européen pour la Recherche Nucléaire
COTS	Commercial-Off-The-Shelf
DAC	Digital to Analog Converter
dB	DeciBel
DC	Direct Current
DDS	Direct Digital Synthesis
DIANA	DIstAnce iN time Algorithm
DSP	Digital (or Discrete) Signal Processing (or Processor)
EM	Electro-Magnetic
FCC	Future Circular Collider
FD	Fractional Delay
FFA	Fixed Frequency Acceleration
FFT	Fast Fourier Transform
FIFO	First-In-First-Out
FIR	Finite Impulse Response
FNAL	Fermi National Accelerator Laboratory
FPGA	Field Programmable Gate Array
FRANCISCO	FabRic with Adaptive aNd deCoupled clockIng for SynChronous prOcessing
FT	Fourier Transform
FTW	Frequency Tunning Word
GSI	Helmholtzzentrum für Schwerionenforschung
HL-LHC	High Luminosity LHC
I/Q	In-phase and in-Quadrature
IF	Intermediate Frequency
IIR	Infinite Impulse Response
IOT	Inductive Output Tube
JOAQUINA	JOintly Averaged and QUaNtized rAtio
JPARC	Japan Proton Accelerator Research Complex
LHC	Large Hadron Collider

LINAC	LINear ACcelerator
LIU	LHC Injectors Upgrade
LLRF	Low Level Radio Frequency
LMS	Least-Mean-Square
LO	Local Oscillator
LS2	Long Shutdown 2
LSB	Least Significative Bit
MERCEDES	MultiplE Rate and Clocking interfacE for Data procEssing and Sampling
NCO	Numerically Controlled Oscillator
NIM	Nuclear Instrumentation Module
OTFB	One Turn FeedBack
PAR	Place And Route
PLL	Phase-Locked Loop
PS	Proton Synchrotron
PSB	Proton Synchrotron Booster
RF	Radio Frequency
RML	Recursive Maximum Likelihood
RPE	Recursive Prediction Error
RS register	Reset Set register
RTM	Rear Transition Module
SA	Spectrum Analyzer
SNR	Signal to Noise Ratio
SOA	State-Of-the-Art
SPS	Super Proton Synchrotron
SQNR	Signal to Quantization Noise Ratio
SRC	Sampling Rate Conversion
SSPA	Solid-State Power Amplifiers
SSRC	Synchronous Sampling Rate Conversion
TBLC	Transient Beam Loading Compensation
TRL	Technology Readiness Level
TWC	Travelling Wave Cavity
uTCA	Micro Telecommunications Computing Architecture
VFD	Variable Fractional Delay
VHDL	Very high-speed integrated circuit Hardware Description Language
VME	Versa Module Eurocard
VNA	Vector Network Analyzer
WLS	Weighted Least Squares
WR	White Rabbit

Contents

Chapter 1

Motivation and Introduction

Abstract: *This chapter introduces the Beam Synchronous Processing topic and motivates the research and the objectives of the problem solved by the Thesis. It presents the CERN laboratory and LLRF systems. Then we overview the Beam-Cavity interaction originating the Transient Beam Loading perturbation, and the current solutions to the problem.*

1.1. Motivation

Signal Processing is a wide field of Science covered by many disciplines of engineering, physics, mathematics, etc. It is a powerful tool that addresses the representation and manipulation of the information contained in signals [1]. Many of its underlying concepts have been known for many years, but the applications born out of the field in the last centuries have boosted its development with the discovery of electricity and the development of micro-electronics providing more advanced and complex solutions and devices. It played a crucial role in control systems in the early 1900s with continuous-time analog systems, and since then, with the advent of digital systems, we can find processing systems in almost any electronic device on the market.

These processing systems implement algorithms that are "*a set of mathematical instructions or rules that, especially if given to a computer, will help to calculate an answer to a problem*" [2]. In general, these instructions and the algorithm need to be aware of the characteristics and properties of the underlying signal or information that they are processing. When you listen to music, for instance on your radio set, you used to move the dial (nowadays you enter some numbers in a screen) to tune the radio station playing your favourite song. The dial instructs the receiver to tune to the frequency in which the radio station is broadcasting its signal. After some treatment in the radio receiver, you listen to your song [3]. Behind the

scenes, the receiver has processed a small band of the radio spectrum around the station frequency; the relevant signal information in your case was the frequency of the carrier broadcasting your song.

There is a sub-field in signal processing, Adaptive Signal Processing, in which the algorithms and/or systems adapt its behaviour to improve its performance [4]; the treatment is adapted to the varying properties of the signal. It finds application and is very common in audio or video processing systems where signals are filtered to cancel echo or to enhance its contents from additive noise [5] or in bioengineering, for instance, with the filtering of electrocardiograms using the Least Mean Square (LMS) algorithm [6]. Usually, in these applications, the complex task is to guess how the property of the signal changes. For instance, if another frequency is assigned to your radio station you will need to re-tune your receiver.

In modern Particle Accelerators radio signals are also used intensively. These machines, that find application nowadays in many fields, increase the energy of charged particle beams [7]: The industry uses them for food sterilization, X-ray lithography, ion implantation, material testing and modification. Synchrotron radiation is used in chemistry, material sciences, molecular and cell biology. Coherent radiation is used in free-electron lasers and holography, for instance. The medical sector is also a well-known user, it makes intensive use with radiotherapy, radiographies, sterilization, etc. In Particle Physics these machines are used to accelerate all species of beams with electrons, protons, ions… to investigate the inside structure of matter; two very energetic beams circulating in opposite directions are guided to collision against each other (or sometimes against a fixed target). In these collisions, the energy is transformed into matter, new particles, and physicists monitor these collisions with detectors to study the behaviour of particles in collisions and probe new theories.

The Radio Frequency (RF) system is responsible for the acceleration (that is the increase of the energy) of the particles. An RF signal induces an Electro-Magnetic (EM) field in an RF structure (usually a resonant cavity). The longitudinal component of the EM field gives a momentum kick to the charged particle when it crosses the cavity [7], [8]. In circular accelerators, called Synchrotrons, the increase in particle speed results in a decrease of the beam revolution period. The RF control system of the synchrotron, called the Low Level RF (LLRF), uses beam signals whose properties (the spectral content) are modified as a result of the change in the revolution period. The LLRF needs hence to tune its processing to the beam [9]; we call this *Beam* (*or Signal*) *Synchronous Processing* (BSP).

This Thesis presents an Architecture for the processing of such a pseudo-periodic signal, whose period changes in a known manner, by tuning the sampling rate of the processed signal. It uses a single fixed frequency hardware clock for the entire data-path and avoids the real-time reconfiguration of the processing elements or the algorithm. The treatment of frequency-variant signals in real-time (adaptive processing) is hence made possible for any static algorithm. The solution is based on a synchronous arbitrary ratio resampler for real-time applications. The Architecture is targeted to a Field Programmable Gate Array (FPGA) implementation. We validate and demonstrate its feasibility by implementing a LLRF

solution, the One Turn FeedBack (OTFB) [10] algorithm for Transient Beam Loading Compensation (TBLC) [9].

This chapter provides an overview of a Particle Accelerator RF system. The European Organization for Nuclear Research (CERN) is introduced, and the context in which the work was born is presented. We outline the CERN future requirements which triggered the use of a fixed frequency clock. The chapter presents in a very simplistic form the LLRF essentials of particle acceleration and beam related signals. The problematic of TBLC is introduced, and the current solutions are presented with their Signal Processing foundations. We introduce the problematics to be solved in the current solution.

1.2. The CERN accelerator complex

The CERN accelerator complex is an international facility established in 1954 and now the largest laboratory for fundamental physics research. It has made many discoveries in the field of experimental particle physics, such as the observation of the Higgs boson in 2012, in the Large Hadron Collider (LHC), completing the Standard Model [11]. It is at the border between France and Switzerland, and the facilities are located on the surface of both countries, but also in underground tunnels. It hosts a group of interconnected accelerators to serve beam for experiments in colliders and fixed targets, aiming at fundamental particle physics research: The structure of matter, the interaction between forces and matter, etc. The technological requirements of the laboratory are usually far beyond State-Of-the-Art (SOA) Commercial-Off-The-Shelf (COTS) technologies. CERN plays therefore also an important role in developing new technologies which can later be exported to other fields not related to physics. Electronics, being a key element of any accelerator, also benefits from this research and technology advance; the Thesis focuses on this field.

The CERN LHC Injectors Upgrade (LIU) project plans to double the intensity extracted from the Super Proton Synchrotron (SPS) for injection into the LHC, therefore requiring a major upgrade of the SPS 200 MHz RF system, including the power plant, layout and cavities [12]. This system is responsible for the acceleration of all beams by means of Travelling Wave Cavities (TWC) that consist of a periodic arrangement of drift-tube cells with a $\pi/2$ phase advance between cells at the centre frequency (200.222 MHz) [13]. The cavity configuration is being modified during the Long Shutdown 2 (LS2, 2019-2020); as the Beam Loading effects scale with the beam current, the new layout will provide higher voltage and reduce the longitudinal impedance at the fundamental harmonic to prevent longitudinal coupled-bunch instabilities [12], [14]. This improves the longitudinal stability required by the planned doubling of the beam intensity for the High Luminosity LHC (HL-LHC) [12]. The LLRF is also replaced. The old system was using beam synchronous clocks (locked to a harmonic of the revolution frequency) for the electronics dealing with longitudinal instabilities (OTFB [15]), resulting in clock frequency sweeping during the acceleration ramp. The electronics was implemented in Nuclear Instrumentation Modules (NIM) and custom-designed Versa Module Eurocard (VME) cards, similar to the ones used in LHC or LINear

ACcelerator 4 (LINAC) [16], [17]. It is entirely replaced with SOA technology during LS2, implemented on the micro Telecom Computing Architecture (uTCA) platform [18], [19].

A Future Circular Collider, (FCC) reaching collision energies of 100 TeV is also being studied at CERN [20]. This project will require a new accelerator with a one-hundred km circumference. This brings many technological endeavours such as new more powerful superconducting bending magnets or a high scale cryogenic plant. On the RF side, with cavities placed in two opposite locations, new techniques and technologies for synchronization in distributed architectures are required [20], [21].

The introduction of the new uTCA standard brings some architectural paradigm changes. The classic master-slave architecture used in timing/synchronization and RF reference clock/phase distribution is now replaced by a distributed network topology [22]–[24], such as the one depicted in Fig. 1.1. The reference clock and instantaneous value of the RF frequency are transmitted as a numerical word, using a deterministic network (blue links in the figure), the White Rabbit (WR) [25], and no longer as point-to-point analog or optical signals. The WR project is a collaboration between several laboratories and universities, including CERN, where many groups are active in its development [26] and applications [18]. The presented work does not include research in the WR but uses the results of many others who are actively working in that field. The main characteristic of this Ethernet-based network is its full determinism, which enables general purpose data transfer and sub-nanosecond precision. Similar architectures are already in use in the accelerator world, for instance, the *Brookhaven National Laboratory* (BNL), in Upton, NY, USA

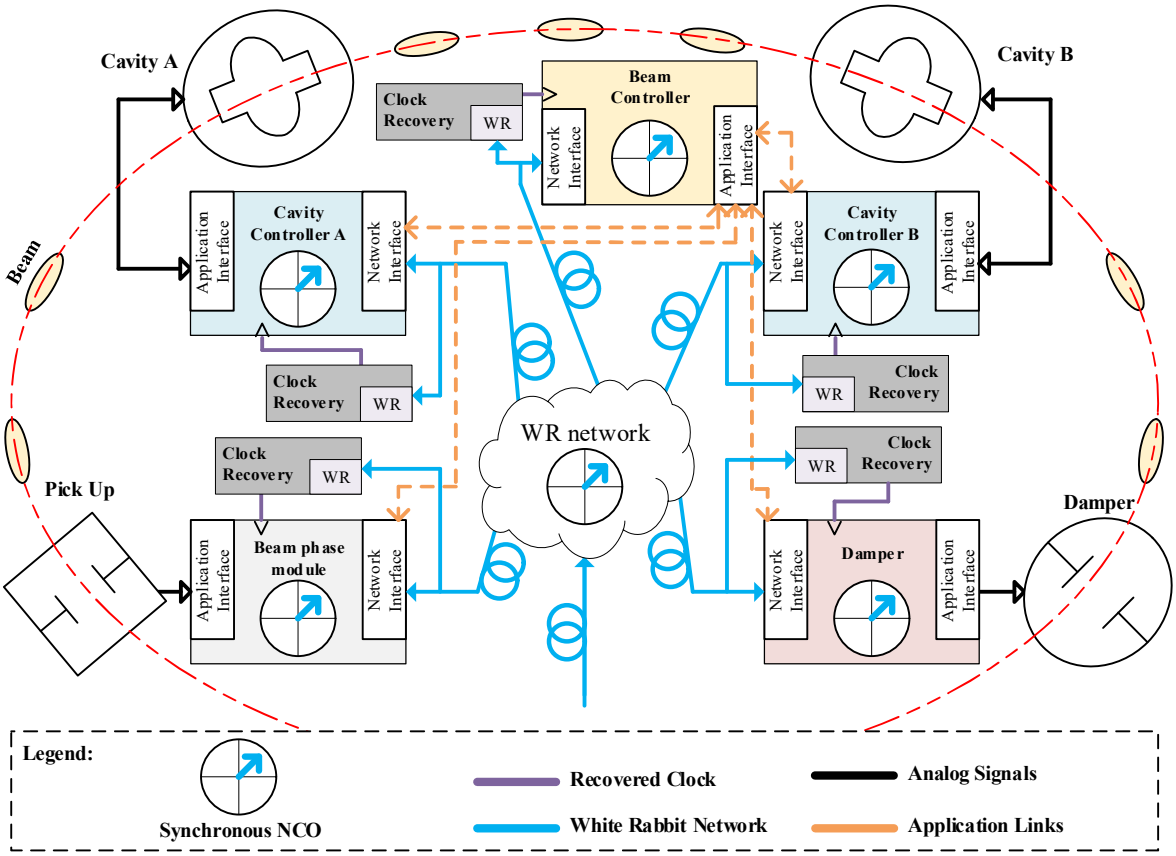


Fig. 1.1. Schematic representation of the LLRF network architecture in a synchrotron that uses White Rabbit for synchronization and recovers the hardware clock from the data stream. Further details are presented in Chapter 6.

[27], uses its own deterministic protocol called the Update Link [28], and the *Helmholtzzentrum für Schwerionenforschung* (GSI), in Darmstadt, Germany [29], uses WR [30]. In our architecture, the receiving slave nodes (cavities, injectors, beam instrumentation, kickers and dampers) get the beam information (RF frequency, cavity amplitude and phase, etc.) from this digital network and extract a 125 MHz reference clock signal from the data stream, clock recovery units in Fig. 1.1. The reference clock is now decoupled from the RF and is fixed in frequency.

1.3. Beam-Cavity interaction

The RF system of a particle accelerator is composed of several subsystems. The cavity, in which a component of the electrical field is aligned to the particle velocity, transfers energy to the beam. The power levels required for this RF signal are in the order of KW to MW. An amplifier is therefore used to increase the level of the RF signal. These high-power amplifiers are normally klystrons, tetrodes, Inductive Output Tube (IOT) or Solid-State Power Amplifiers (SSPA). A distribution system composed of waveguides and coaxial conductors transports the RF power from the amplifier to the cavity. Circulators and dummy loads are commonly used to absorb power reflected from the cavity. Finally, the LLRF is the low power electronics generating the RF signal and keeping the synchronism between beam and RF phase.

The beam is a collection of charged particles travelling in the accelerator vacuum chamber, and crossing the accelerating cavities [31]. These particles are grouped in bunches spaced by a multiple of the RF wavelength [7], [32]. With electromagnetic pick-up sensors, the LLRF system acquires signals related to the azimuth and transverse position of the beam [9]; particle bunches passing through the pick-up generate a short pulse. In circular accelerators this passage is periodic turn after turn; the pick-up signal in the time domain is hence a train of short pulses spaced by the revolution period. The signal spectrum is also a train of pulses at multiples of the revolution frequency [9], [33]. When the beam crosses the cavity, it modifies the electromagnetic field. This effect is called Beam Loading [34], [35]. The cavity impedance is excited by the beam current generating a voltage in addition to the one induced by the generator; the effective accelerating voltage seen by the beam is therefore perturbed. These beam-induced perturbations such as Transient Beam Loading (at the injection or when the beam pattern is not uniform) will appear in the spectrum at the revolution frequency and its multiples [9]. Due to the non-zero bunch length, these revolution frequency harmonics, nF_{rev} (Eq.(1.1)) are not all equal in amplitude, A_n and phase, φ_n (the envelope of the revolution frequency harmonics is the Fourier Transform (FT) of the longitudinal bunch profile [36]).

For leptons (electrons and positrons) synchrotrons the particles are already very relativistic (velocity close to the speed of light) because their rest mass is small. Therefore, the RF frequency can be fixed and the BSP algorithms processing these revolution harmonics do not need reconfiguration during the acceleration ramp. Hadrons (protons and heavy ions) are much heavier. In the CERN SPS, the speed of hadron particles changes significantly during the acceleration ramp, and so does the RF frequency. In that

case, the BSP algorithms treating beam-induced perturbations need to tune their processing to the signal spectrum; the position of the harmonics is displaced as the spacing among them is increased. As the revolution frequency varies very slowly compared to the beam dynamics, the signal from a given bunch sampled by a LLRF pick-up is

$$x(t) = \sum_{n=0}^{N} A_n \cdot e^{i(2\pi \cdot nF_{rev}t + \varphi_n)} \qquad \text{Eq.(1.1)}$$

where N, the number of harmonics to be considered, depends on the intended application. For compensation of the beam-induced transient in a given cavity we must only consider the harmonics falling in the cavity BandWidth (BW) (in the case of the SPS the regulation bandwidth covers 5 MHz on each sideband around the RF). In all cases, the signal bandwidth must be limited before sampling to avoid aliasing. Refer to [7], [9], [32], [33] for more information.

1.4. State-Of-the-Art in BSP algorithms

Let us consider a simple process trying to enhance the pick-up signal out of white additive measurement noise. The required comb filtering contains harmonically related pass-bands centred on the revolution frequency harmonics. Its frequency response needs to follow the signal spectrum at these multiples of the revolution frequency.

Such a problem is commonly found in adaptive signal processing, for instance, a classic method is Adaptive Noise Cancelling (ANC) [5]. The filtering of electrocardiogram using the LMS algorithm was an early example. More sophisticated algorithms such as the Recursive Prediction Error (RPE) and Recursive Maximum Likelihood (RML) have been proposed for comb filtering [37], [38]. The latter did not require knowledge of the instantaneous fundamental frequency. It can be estimated by the algorithm. The computational load of these algorithms scales linearly with the number of harmonics when using the simple LMS, and quadratic for the more sophisticated RPE and RML algorithms. Their performances degrade if the number of harmonics present in the signal is not modelled correctly [37]. We require however a much lighter solution for LLRF algorithms in terms of calculations. Variable Fractional Delay (VFD) filters have been proposed to implement a comb adapted to any fundamental frequency (not necessarily a sub-multiple of the sampling frequency) [39]. The coefficients of the filter are then changed to track the fundamental frequency. A side effect is the displacement of the poles with the delay that can possibly make the Infinite Impulse Response filter (IIR) unstable. We want to avoid reconfiguration of the filter as it might be a cumbersome approach for our real-time system.

The accelerator world employs several BSP approaches in this adaptive filtering to cope with beam-related perturbations such as Transient Beam Loading. In the early digital systems introduced in the mid-1980s, the sampling clock was swept proportionally to the revolution frequency [10]. This locks the processing on the spectral content of the beam signal and the processing algorithm needs not be changed

during the acceleration. This swept clock philosophy has been extensively used in the accelerator world, at CERN and other labs [40]. This easy solution is not optimal, however. In case the processing is not related to the beam energy ramp, for example, to compensate an amplifier frequency response, the processing should not change with beam energy. In that case, *Beam Asynchronous Processing* (BAP) would be preferred. If the system uses such a swept clock, this requires complicated implementations with limitations [41]. Furthermore, modern FPGAs are intended for use with a fixed clock, as swept clocks pose problems in FPGA clocking logic and Phase-Locked Loops (PLLs). This also limits the use of its serial interfaces. The old swept clock scheme would therefore limit the exploitation of the new CERN distributed LLRF architecture, and the SOA uTCA based processing systems.

Alternative approaches have been implemented: In small machines (high revolution frequency and a small number of bunches) with the spectral content of the beam signal limited to a small number of revolution frequency lines, a common solution is to decompose the problem in several parallel processing systems, one per revolution line, where a fixed sampling clock can be used. This requires multiple demodulators (one per revolution harmonic) for base-band down-conversion and several processing systems [42]. The amount of resources grows linearly with the number of revolution lines to be treated. This solution solves the constraint of the swept clock, as it can be implemented with a fixed frequency clock. However, the resources required when extending the regulation bandwidth limits its applicability. Examples of this strategy are found at CERN [43]–[45] and the *Japan Proton Accelerator Research Complex* (JPARC) [46], in Tokai, Japan [47]. But it does not apply to larger machines where many revolution lines are to be covered.

Another approach used is feedforward. Based on the reproducibility of the machine cycles and the slow variation of the beam pattern in each cavity passage, the feedforward employs signal tables precomputed from past observations, which are added to the set-point to compensate the beam loading effect. In this case, the processing itself is not tuned to the spectral content of the signal, but the set-point is adapted to mitigate the predicted effect of the beam passage. These algorithms require a deep knowledge of the machine, including the non-linearity of the amplifier. When many types of cycles and different particles and users are needed, this might require information not always available. Such a system can be found at BNL [28] for instance. A similar approach is used also at the *Fermi National Accelerator Laboratory* (FNAL) [48], in Batavia, IL, USA [49], and CERN SPS [50] where corrections are applied to the amplifier drive based on the beam signal acquired with a pick-up. These feedforward approaches, as open-loop systems, lack adaptive capabilities and are sensitive to variations of the amplifier response for instance.

If we want to use a fixed frequency clock there is still the need for a solution to avoid the reconfiguration of the processing elements (filtering for instance) to tune to the beam revolution frequency (and to the spectrum of the sampled signal). The real-time reconfiguration in complex processing schemes or algorithms can require plenty of parameters to change and becomes cumbersome. A generic BSP solution

is therefore desirable which can be extended to small or larger machines with different regulation bandwidths. The Thesis focus on this BSP Architecture applied to TBLC, but at the same time being generic enough to host other BSP related algorithms such as longitudinal and transverse dampers [51], [52]. The solution needs to be compatible with modern electronics making use of a fixed frequency clock.

1.5. Document organization

This dissertation, that presents a new Architecture for BSP and demonstrates its use for implementing the OTFB algorithm, is organized to guide the reader through the problematics of the actual systems, the decisions taken in the research process, the developments performed and the final application of the ideas in the implementation of the OTFB algorithm.

The introduction in Chapter 1 has presented the *circumstances* and *ideas* which triggered the research work. It has also settled basic *foundations* related to Accelerators and LLRF systems. Then it presented the *SOA in electronics* and the *principal solutions* used for TBLC in LLRF systems.

Chapter 2 presents the *objectives* and *contributions* of the Thesis to the SOA grouped in three main fields: *BSP* solutions for Particle Accelerators, *Resampling* of frequency varying signals in FPGA with variable resampling ratios, and *TBLC* by means of the new OTFB algorithm implementation.

Chapter 3 presents the proposed *Architecture for BSP*; it introduces the so-called *resampling sandwich* and elaborates the ancillary hardware and abstractions needed for its implementation. These are our virtual *FPGA FabRic with Adaptive aNd deCoupled clockIng for SynChronous prOcessing* (*FRANCISCO*), and the *MultiplE Rate and Clocking interfacE for Data procEssing and Sampling* (*MERCEDES*) *interfaces*. It presents also the *JOintly Averaged and QUaNtized rAtio* (*JOAQUINA*) *Frequency-Locked Loop* that solves implementation problems related to quantization of the ratio signals.

Chapter 4 is focused on the core element that performs resampling, our novel *Sampling Rate Conversion (SRC) architecture, the resampler,* that lies at the input and output of the *resampling sandwich*. It elaborates and presents the implementation details for its three functional units; the *DIstAnce iN time Algorithm* (*DIANA*), the *VFD filter* and the *synchronization logic*.

Chapter 5 presents the *verification* and *validation* results of the BSP Architecture and the developed hardware and units of Chapters 3 and 4; first the simulations of all the elements and then the hardware test with the implementation of the Architecture in a uTCA crate for measuring its performance.

Chapter 6 *demonstrates* the use of the BSP Architecture implementing the *new OTFB algorithm for TBLC*. Then it presents the *verification* and *validation* results of the simulations and the hardware tests in a *real CERN SPS cavity*.

Finally, Chapter 7 presents the *conclusions* and suggests some *future work* to be conducted in the field.

Chapter 2

Contributions of the Thesis

Abstract: *This chapter presents the contributions of the Thesis to the State-Of-the-Art in the different fields in which the work has elaborated. First, a high-level overview of the achievements is presented, and some tangible improvements stated. Then the contributions for the main related domains are detailed. These contributions encompass technical (signal processing - LLRF Beam Synchronous Processing), technological (FPGA resampling architectures), and application (particle accelerators – Transient Beam Loading Compensation) aspects.*

2.1. Introduction

The previous chapter has presented the more relevant LLRF Beam Synchronous Processing solutions for Transient Beam Loading Compensation in particle accelerators. The swept clock architecture is the key element of the current CERN solution [10], [15], [22]. It nevertheless poses some problems (that we present in this chapter) and it is a bottleneck for implementation in modern digital technologies. The contributions of the Thesis in this aspect are therefore twofold:

- To solve and avoid the present problems in new LLRF architectures.
- To develop techniques and technologies ensuring the efficient and feasible implementation of new architectural paradigms.

The new projects being planned and implemented at CERN, namely LIU SPS [12], [14], HL-LHC [53] and FCC [20], [54], motivate the change of the synchronization and RF distribution architecture. A distributed approach is envisaged more reliable and offering better scalability for these "new-sized" projects (FCC plans a new accelerator with a 100 km circumference ring). This introduces the use of a deterministic protocol, the White Rabbit [25]. With this new protocol, the distribution of a swept clock in a dedicated

fibre has no sense, since this would duplicate the distribution infrastructure. The clock is therefore now extracted from the data stream, locally in each node (Fig. 1.1), being this a fixed frequency clock [18]. The revolution frequency and RF frequency information in each RF station are distributed via the WR network in a digital format [55]. For Beam Synchronous Processing, this makes it more appropriate to use the fixed frequency WR clock and this digital information, instead of regenerating a swept clock to feed the processing FPGA. A new strategy or algorithm is therefore needed for Transient Beam Loading Compensation, being this the contribution of the Thesis in the application field:

- A new OTFB implementation with fixed frequency processing clock.

2.2. Tangible improvements with the new Architecture

This section presents some tangible contributions to current problems in the LLRF systems object of the Thesis. These problems result from the present implementation of the LLRF architecture at CERN, and the new presented Architecture aims at solving them.

The distributed and fixed clock architecture proposed in this Thesis benefits to RF gymnastics [56], [57]. This term refers to manipulations in the beam tailoring its longitudinal characteristics. These manipulations are done by modulating the RF parameters to achieve the desired beam, including bunch length, energy spread, distance between bunches or number of bunches among others. More complex operations as slip stacking [58] can be now implemented, and machine synchronization schemes [59] simplified. A common clock between machines and an absolute time reference facilitate the computation of synchronization events and the computation of phase advance of signals among others [60], [61].

Another consequence of using the new Architecture with a fixed clock is the improvement on spectral purity of the signals. PLL based architectures are used to clean the clocks. These have a certain operational frequency range based on loop filters which cover a certain bandwidth. The new schema using fixed frequency clocks makes it possible to optimize the cleaning architecture for a given fixed frequency.

In line with these technical aspects, modern electronics, namely FPGAs, will be using this fixed frequency Architecture in uTCA platforms [19]. Digital clock managers in FPGAs, among others, make use of PLLs [62]. Eventual problems associated with these subsystems, as for instance potential unlock of the PLL, are avoided thanks to the new fixed clock. This permits the exploitation of all the features of SOA FPGAs. This can be extended to modern Analog to Digital Converters (ADCs) and Digital to Analog Converters (DACs) using differential serial interfaces. These usually include PLLs and complex logic for clock synchronization for chip-to-chip communications. A swept clock can make its use unfeasible, forcing the designer to rely on old parallel interfaces. On the processing side, these digital systems make intensive use of Discrete Signal Processing (DSP) architectures based on synchronous digital logic design. This philosophy is based on the use of combinatorial logic elements and registers. The combinatorial elements are pipelined between the registers to increase the operation frequency. A variable clock forces the FPGA

Place and Route (PAR) process to use the highest frequency value in the slack estimation. This complicates the re-use of implemented FPGA designs that use clocks which are extracted from the RF and hence swept. A design running in an operational machine can require optimizations of the hardware architecture and a new synthesis and PAR, this limits the re-use in a different accelerator or system where the RF is different. In line with this, multiple clock domains are usually present in FPGA designs, and a swept clock implies a huge complication for the synchronization and communication between clock domains. The fixed frequency clock greatly simplifies these issues.

Another consequence of the proposed solution is to avoid interruptions of the processing clock between cycles; in the old system the clock comes from the RF [10], any interruption on that signal causes also the interruption of the clock. This is the case when doing resynchronization between machines; the RF is interrupted or abruptly modified, thus originating clock problems [59]. The new fixed frequency clock is not any longer extracted from the RF, now it is independent and regenerated from the WR network. Since the WR is always in operation, no more interruptions when resynchronizing machines will be present. A similar phenomenon happens in small synchrotrons as for instance the CERN *Proton Synchrotron Booster* (PSB) [63], where the revolution frequency span is wide. This implies a complex clocking scheme for FPGAs: To avoid such a big span in the clock, several harmonics multiplexed in real-time are used as clock source. This produces phase discontinuities and jumps. Again, this is avoided in the new Architecture proposed in this Thesis.

2.3. Contributions to the State-Of-the-Art

The previous chapter has motivated the Thesis work by presenting some future projects which are being planned at CERN. Such projects require paradigm changes which are not compatible with some implementations and LLRF architectures currently used at CERN. Advances in the Technology Readiness Level (TRL) of the LLRF systems and new techniques have been developed to cope with the new paradigms and to solve the specific requirements of their implementation and application. The three main domains in which contributions have been made are presented now.

2.3.1. Signal/Beam Synchronous Processing

The Thesis proposes a new solution for Signal/Beam Synchronous Processing making use of a fixed frequency processing clock. This Architecture makes it possible the processing of signals with known but possibly varying frequency with algorithms dependent on the spectral content of the signal. The common approach for that is the reconfiguration of the algorithm parameters [39] or the use of adaptive processing (filtering) [5], [37], [38]; an alternative approach is presented here. The presented innovation is especially suited for periodic signals with a varying fundamental frequency. These signals present a Homothetic spectrum (we call it Homothetic because the change in spectrum is equivalent to a dilation of the frequency axis), like the one depicted in Fig. 2.1. The plots depict the spectrum of a beam signal

acquired with a pick-up in different time instants of an accelerating ramp. The blue plot shows the revolution frequency peaks at the beginning of the ramp, while the purple and red plots show the peak position and spacing at later instants during the momentum ramp. The red plot containing the harmonic peaks at higher frequencies and with the widest spacing between them corresponds to the last instants of the accelerating ramp. The purple one shows the spectrum in the middle of the ramp. The processing needs to adapt to the constant change of position and spacing of the peaks. When this signal-synchronous technique is applied in the Accelerator field, we call this *Beam Synchronous Processing*.

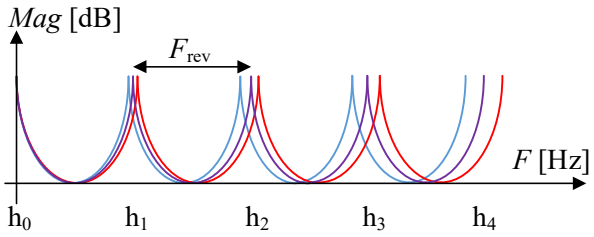


Fig. 2.1. The simplified spectrum of a beam signal acquired with a pick-up; the position and the spacing of the harmonics change during acceleration ramp proportionally to the revolution frequency increase (Homothety).

The presented solution tunes the signal to the processing algorithm by varying the sampling period T_s of the digitized signal with fundamental frequency F while keeping the algorithm parameters fixed (no reconfiguration). If this sampling frequency variation $f'_s = f_s \cdot (1 + \Delta)$ is proportional to the variation of the signal frequency $F' = F \cdot (1 + \Delta)$, the representation of the signal frequency in the discrete normalized spectrum ω remains constant

$$\omega = 2\pi \cdot f = 2\pi \frac{F'}{f'_s} = 2\pi \frac{F \cdot (1+\Delta)}{f_s \cdot (1+\Delta)} = 2\pi \frac{F}{f_s} = \text{constant} \qquad \text{Eq.(2.1)}$$

This resampling operation brings the discrete representation of the signal frequency ω to a predefined digital frequency ω_{proc} where the processing has been defined ($\omega = \omega_{proc}$). The solution is generic and supports the implementation of any processing or filtering algorithms.

2.3.1.1. FRANCISCO fabric

The solution is based on an adaptation fabric, called *FabRic with Adaptive aNd deCoupled clockIng for SynChronous prOcessing* (*FRANCISCO*), built on top of the real FPGA hardware fabric. In the hardware fabric, the frequency of the processing f_p and sampling f_s clocks are identical. In the adaptation fabric, the clocks are decoupled and need not to operate at the same frequency. This makes it possible to use a hardware clock with fixed frequency to operate the hardware fabric, and on top of that to implement the adaptation fabric with an average variable sampling rate. In the adaptation fabric, the sampling period of the data is modified according to our needs, and this can be done in real-time.

The solution is well suited to FPGA technology and applications, where the hardware processing clock is preferably fixed in frequency and stable. This permits to use all the hardware and clocking resources of the FPGA, something that might not be feasible with a swept clock. The solution can also be migrated

to Application Specific Integrated Circuits (ASIC) technologies, however, the target within the Thesis is FPGA.

The adaptation fabric *FRANCISCO* interfaces the real hardware fabric by means of dedicated interfacing entities called *MultiplE Rate and Clocking interfacE for Data procEssing and Sampling* (*MERCEDES*). These interfaces perform processing and sampling clock coupling (*MERCEDES Couple*) and decoupling (*MERCEDES Decouple*). They generate also a valid signal when decoupling, and merge this signal with the processing clock when coupling the data-path. The valid signal, depicted in the chronograms of Fig. 2.2, accompanies each processing slot in the adaptation fabric. The signal is introduced to indicate which processing slots contain valid data in the data-path (populated processing slots). The chronograms present also the clocks in the *MERCEDES Decouple* interface. The data-path arriving at its input is coupled, and processing and sampling clocks operate at the same rate. In the output, the sampling clock remains at the same frequency, but the processing clock operates at the double frequency, $M = 2$. The *MERCEDES Couple* interface performs the complementary operation, it merges the sampling and processing clocks with the frequency of the latter reduced by M. Only the slots flagged valid are passed out of the interface within the now coupled data-path.

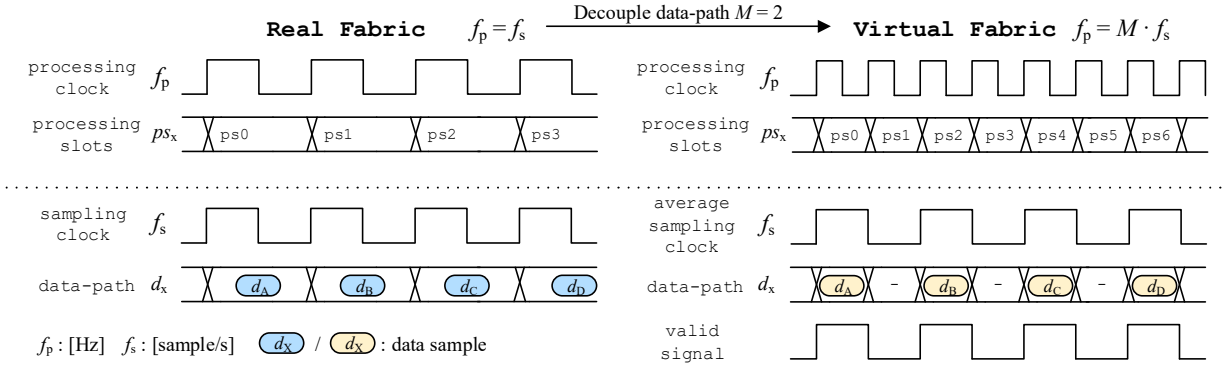


Fig. 2.2. Signals at the input and output ports of a *MERCEDES Decouple* interface. The input port interfaces a coupled data-path with sampling and processing clocks operating at the same frequency. The output port interfaces a decoupled data-path with a processing clock operating at a frequency double with respect to the sampling clock, $M = 2$.

The hardware and *FRANCISCO* fabrics are depicted in Fig. 2.3, the hardware in white while the adaptation fabric is coloured in grey. The hardware processing clocks are also depicted in Fig. 2.3, blue lines are used for the real hardware fabric, and yellow lines for the adaptation fabric. These clocks define the processing cycles or processing slots (ps_x) at which the FPGA can operate on data in both fabrics. This frequency is the maximum sampling rate that the sampling clock can achieve (when $f_s = f_p$). The adaptation sampling clock frequency is thus a slower fraction (or equal in the limiting case) of the hardware clock frequency.

The adaptation sampling clock does not exist as a physical signal, it is an abstraction which defines the average sampling rate of the data in the data-path. It abstracts the samples from the processing slots of the real hardware clock. The decoupling requires the use of a hardware processing clock f_p at higher frequency (or equal in the limiting case) than the *virtual* sampling clock f_s. When this is satisfied, the number of available processing slots is larger than (or equal to) the number of available samples.

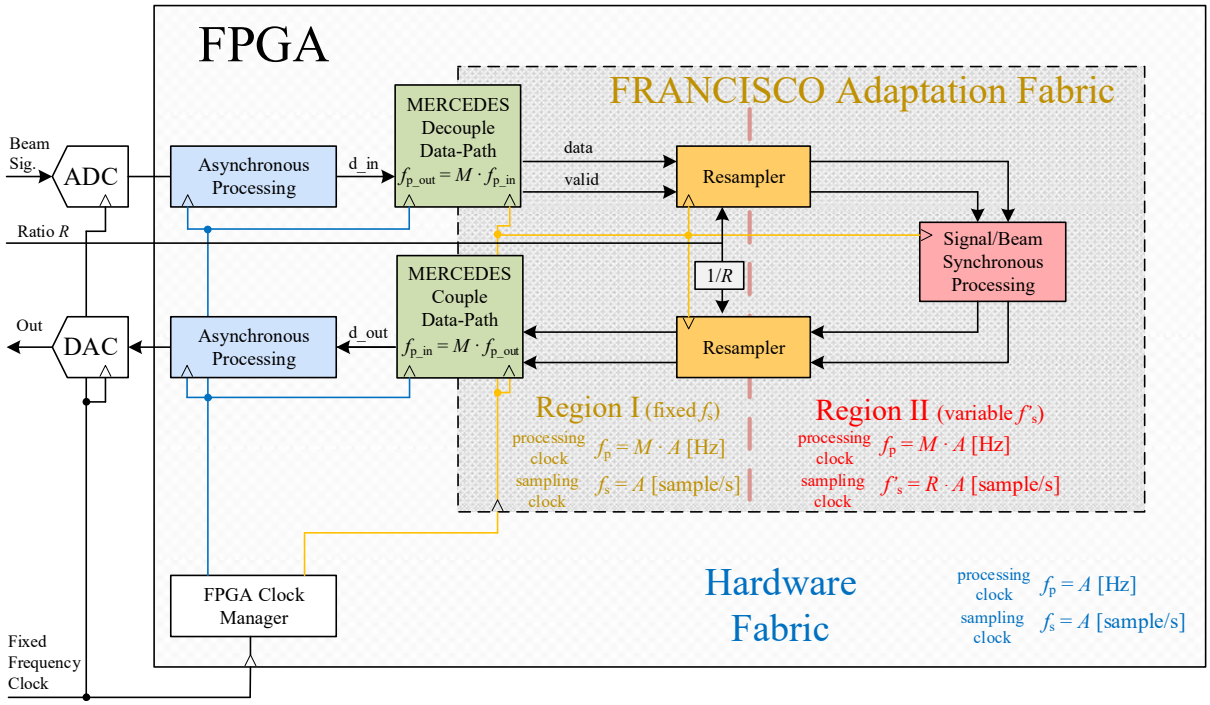


Fig. 2.3. Schematic representation of the fabrics and clocking architecture; sampling f_s and processing f_p clocks for the hardware fabric (white fabric with blue clocks), and the *FRANCISCO* adaptation fabric (grey fabric with yellow clocks). In the figure, A is an arbitrary value, and M is the relation between processing clocks in the *MERCEDES* interfaces.

The hardware fabric in Fig. 2.3 contains the processing f_p and sampling f_s clocks operating at the same arbitrary frequency, $f_p = A$ and $f_s = A$, respectively. The adaptation fabric can host multiple arbitrary sampling rates defined according to the limits specified before. Two different sampling clock regions (region I and region II) are depicted in the adaptation fabric in the figure as an example. In this case, both regions use a processing clock with frequency $f_p = M \cdot A$; for instance, with $M = 2$ the frequency of the hardware clock is doubled. The sampling rate is however different in the two regions; while the region I uses an adaptation sampling rate of $f_s = A$, region II uses a rate of $f'_s = R \cdot A$.

Input and output resamplers in the adaptation fabric perform the conversion of the sampling rate between these two regions. The input resampler translates the input data with a fixed sampling period $f_s = A$, to a data stream with a variable sampling period $f'_s = R \cdot A$. The ratio R in the sampling rate conversion is the relation between the frequencies of input and output sampling clocks of the resampler, and it can vary in real-time. The output resampler performs the inverse operation using a resampling ratio inverse of the input one. Since the processing clock of the adaptation fabric is $f_p = M \cdot A$, the maximum adaptation sampling rate which can be achieved in region II is $f'_s = M \cdot A$, when $R = M$.

In the adaptation fabric, the faster processing clock makes it possible to operate and process any data in the data-path in a bounded time, without data overrun regardless of the sampling rate. The fabric tracks the valid flag (valid line in Fig. 2.3) and uses only data results marked as valid. Contrary to this, the hardware fabric "operates" on any processing slot (sampling and processing clocks are coupled). The *FRANCISCO* fabric and the *MERCEDES* interfaces are described in detail in Chapter 3.

2.3.2. Resampling architecture with arbitrary and real-time variable ratio

The main technical contribution of this work is a new resampling architecture with a real-time variable and arbitrary resampling ratio [36]. The architecture is intended for implementation in an FPGA where the processing clock is preferably fixed, as presented in the previous point. The architecture can easily be ported to ASIC technology. It is based on the *FRANCISCO* fabric with decoupled clocks. Such a resampling architecture, accepting the arbitrary modification in real-time of the resampling ratio is not common in an FPGA. The available solutions accepting variable rates usually support only a predefined set of values [64], [65]. When any other ratio needs to be used a re-synthesis of the design is required [66]–[71]. This is motivated by the clocking schema used in the output interface of the resampler; these architectures do not decouple the sampling and processing clocks. In that case, when the resampling ratio is modified, the frequency of the processing clock in the output port is changed, being this not always acceptable for an FPGA. To cope with this limitation, the presented resampler uses the decoupled data-path. This allows for using any average sampling rate at the input and/or output ports.

The interpolation architecture estimating output samples and the timing generation mechanism [67], [70]–[73] for the output port, are also usually linked to the sampling clock of the output port (Asynchronous Sampling Rate Conversion, ASRC [74]). The existing resampling architectures for ASIC technology [73], [75], [76] supporting a variable resampling ratio require this output clock to be an input signal; the output clock fed to the device dictates the time instant in which the output data is computed and latched in the port. The presented architecture avoids the use of such external output clock by computing the output sampling period based on the input port (or system) clock (Synchronous Sampling Rate Conversion, SSRC). For this purpose, a new algorithm called *DIstAnce iN time Algorithm* (*DIANA*) has been developed. This algorithm is described in detail in Chapter 4. The first asynchronous resampler presented is depicted in Fig. 2.4(a); the input and output ports are fed with their own clocks. The synchronous resampler proposed in this work is depicted on Fig. 2.4(b).

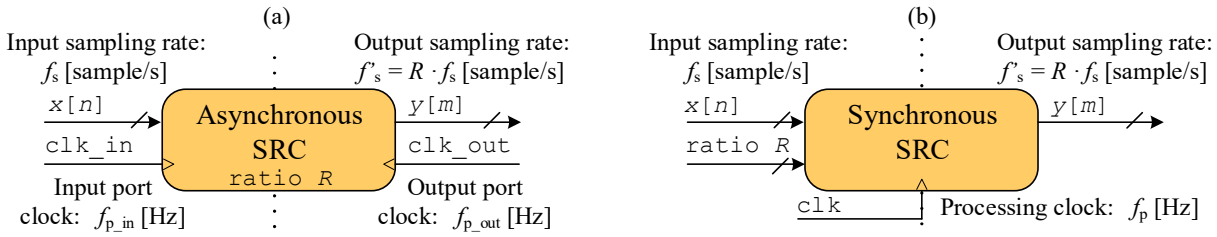


Fig. 2.4. (a) ASIC style asynchronous arbitrary ratio resampler. (b) FPGA synchronous arbitrary ratio resampler.

2.3.2.1. Farrow-based resampler with decoupled clocks

The resampling architecture, depicted in Fig. 2.5, is inspired on a Farrow-based [77] VFD filter [78] for interpolation, and a computing engine implementing the *DIANA* algorithm for timing. The Farrow-based VFD is an efficient hardware architecture to estimate the value of a signal in a time instant different from the available data by less than a sampling period. Such architecture employs a Finite Impulse Response (FIR) filter bank with static coefficients. It is efficient as there is no need to recompute these coefficients

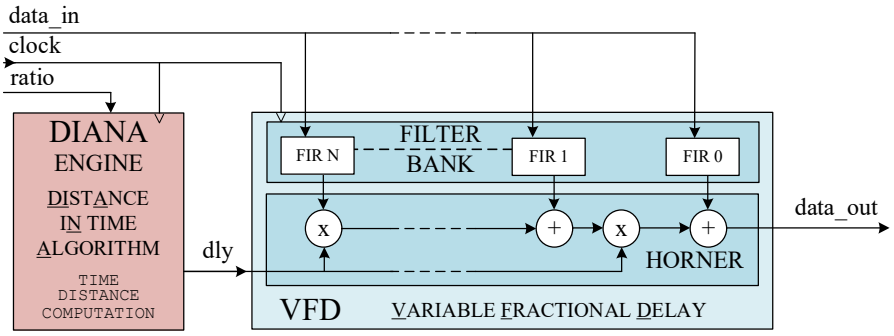


Fig. 2.5. Resampler architecture based on a Farrow Variable Fractional Delay Filter, VFD, and the *DIANA* algorithm.

when the time distance (delay) between available samples and desired new sample varies. The outputs of the filters are combined with the delay value in a Horner structure [79]. The *DIANA* algorithm computes the time distance based on the received samples at the input port of the resampler, the resampling ratio, and the history of processed samples. It generates the delay value signal fed to the VFD when a new output sample can be computed. It also generates a control signal indicating when this output sample can be processed, or conversely when the VFD and the resampler output data-path are void. This is the case when the time distance is bigger than plus or minus half an input sampling period, maximum delay accepted by our VFD implementation.

The resampler relies on the decoupled data-path to implement the real-time and arbitrary variable resampling ratio. The average sampling rate of the samples in the data-path for a given number of clock cycles varies according to the ratio. This is depicted in Fig. 2.6. The left chronogram presents the resampler input where the average frequency of the sampling clock in the adaptation fabric has a value of $f_s = A / 2$ with a processing clock with frequency f_p = A, and $M = 2$. The right chronogram shows the output signals, where the average frequency of the sampling clock in the adaptation fabric, within six processing clocks, is $f'_s = A \cdot (4 / 6)$ with a processing clock with frequency f_p = A, and $M = 2$. Note that the relation M is handled by the *MERCEDES Decouple* interface. The resampler has an up-sampling ratio of $R = 4 / 3$ with

$$R = \frac{f'_s}{f_s} \qquad \text{Eq.(2.2)}$$

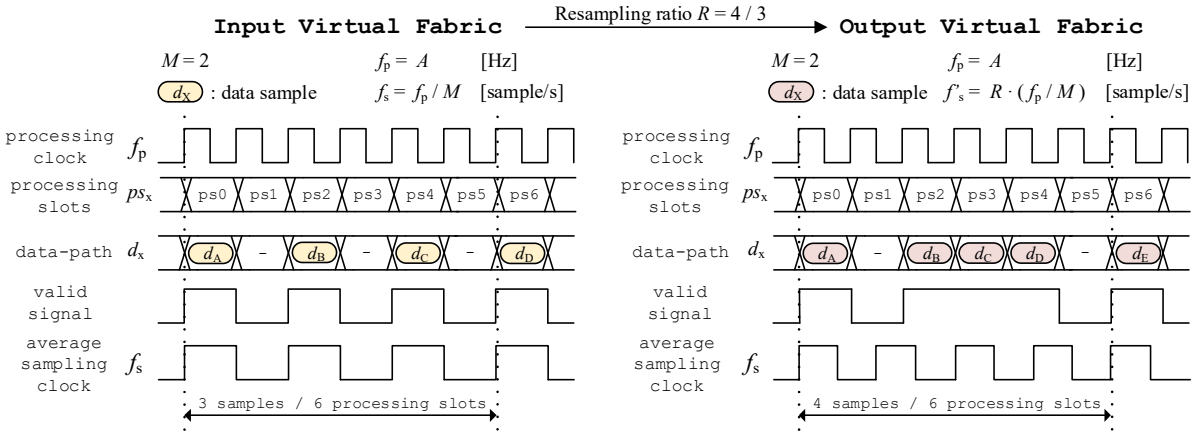


Fig. 2.6. Signals at the input (left) and output (right) port of a resampler configured with an up-sampling ratio $R = 4 / 3$ and implemented in the *FRANCISCO* adaptation fabric.

2.3.3. New Transient Beam Loading Compensation schema

The contribution of the Thesis, as application in the field of Low Level RF for Particle Accelerators, is a new wideband implementation of the One Turn FeedBack algorithm [10] for Transient Beam Loading Compensation [80] implemented in an FPGA using a fixed frequency processing clock. The implementation is based on the original solution from Boussard [10] but employs the Beam Synchronous Processing Architecture presented in section 2.3.1. It solves efficiently the constraints that new hardware technologies and systems impose in terms of clocking schemes. Nowadays it is not feasible to keep using an RF derived clock as in the original solution. The presented approach respects the original idea, as it does not limit or poses constraints on the regulation bandwidth (it can handle an arbitrary number of revolution harmonics). This is not the case with some other alternatives which imply limitations, depending on the bandwidth to be covered; some solutions decompose the problem in multiple instances of processing/filters or use filter banks, each one addressing a single revolution harmonic [43], [44], [46], [81]–[83], or even time multiplexing of hardware [84]. This increases drastically the hardware resources needed and/or reduces the performance of the solution. Reconfiguration of the filter architecture might seem also feasible, however, the rate at which the reconfiguration needs to be done, or the volume of data to do it, limits its applicability.

The present work is a competitive alternative to the approaches indicated previously since it does not imply overhead in the hardware resources when the bandwidth (or the number of harmonics) to be covered increases. Instead of adapting the processing to the spectral content of the signal, this method modifies the sampling rate to tune the spectral representation of the signal to a predefined normalized frequency in which the processing is performed with a fixed filter. The hardware resources to be used are almost the same as with a swept clock architecture. The solution is based on the *FRANCISCO* fabric presented before, which is used for BSP, depicted in Fig. 2.7. The only difference in hardware resources needed, with respect to the original solution, consists in the two extra resamplers and *MERCEDES* interfaces, and some minimal signalling logic in the decoupled fabric.

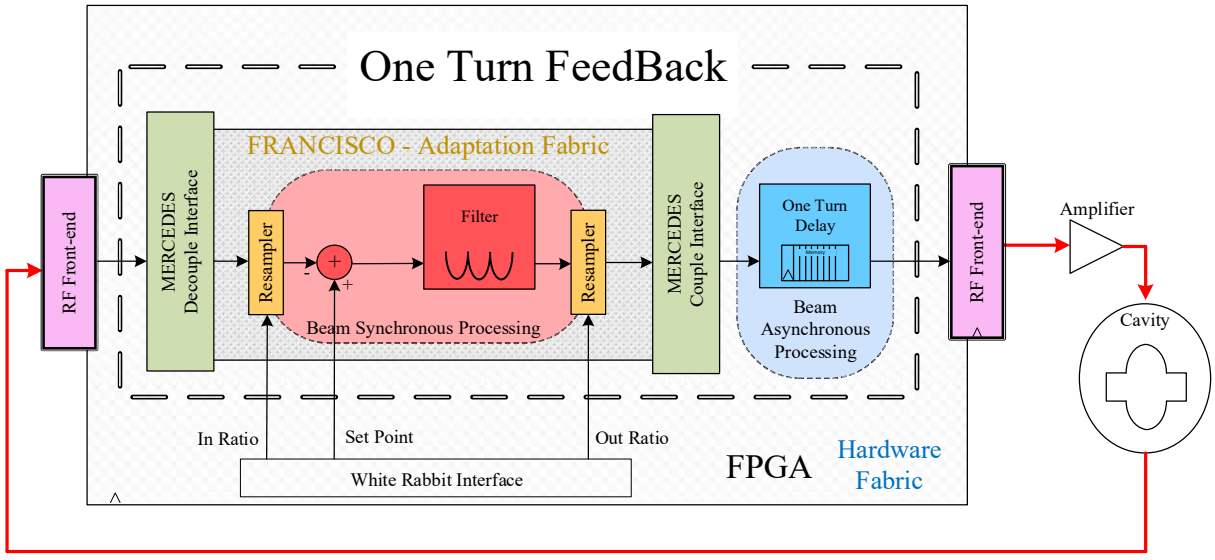


Fig. 2.7. New One Turn FeedBack architecture based on the *FRANCISCO* fabric for the BSP filter.

The communication requirements (bandwidth required in case of reconfiguration of the processing) are also negligible as the only information required is the resampling ratio. It also simplifies the use of more complex filtering in the OTFB, for instance addressing synchrotron sidebands to increase longitudinal stability. This is thanks to the fact that no modification or hardware limitation is imposed to these more complex filtering architectures, thus it is easy to migrate and combine different filters addressing different spectral components.

2.3.3.1. New One Turn FeedBack with the fixed clock

The OTFB has been introduced in Chapter 1, it contains a filter tuned to the revolution frequency harmonics of the processed signal. This filter is the processing that needs to track the momentum ramp of the synchrotron, i.e., the sweeping revolution frequency harmonics. In the presented work, it is implemented in the *FRANCISCO* fabric. Since the revolution frequency is known (it is also the LLRF that controls it), it is possible to tune the discrete representation of the sampled data to the filter response using the presented fabric. This avoids alternatives requiring estimation and tracking of the revolution frequency, based on adaptive filters, or the real-time reconfiguration of the filter parameters.

The filter in the *FRANCISCO* fabric is a static IIR comb as depicted in Fig. 2.8. The first peak of the comb above *Direct Current* (DC) is used as the reference to define the filter response. This peak is computed to be at a normalized frequency ω_{proc1} that matches, at the beginning of the momentum ramp, the first revolution harmonic ω_{h1} of the filtered signal (h_1 in Fig. 2.1), $\omega_{proc1} = \omega_{h1}$. The rest of the filter and the signal peaks follow due to its periodicity.

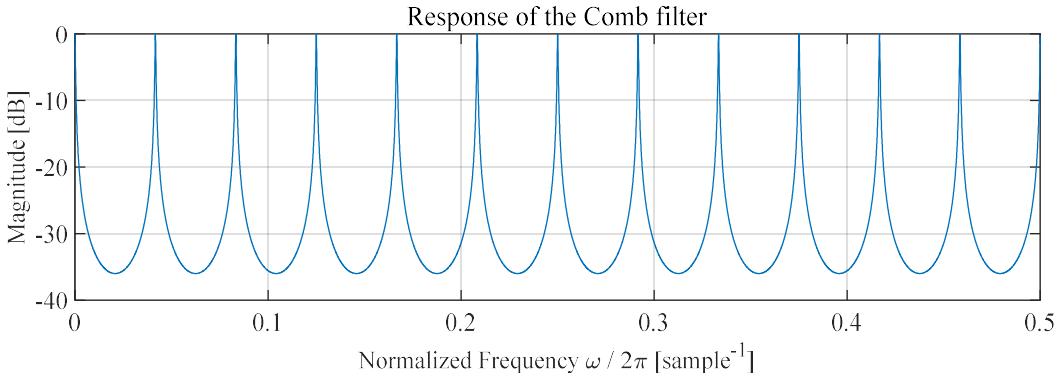


Fig. 2.8. The response of a Comb filter with 12 resonances in the first Nyquist zone.

The filter is placed between two resamplers in the *FRANCISCO* fabric. The input resampler adapts continuously the sampling frequency f'_s in the adaptation fabric (Fig. 2.3). This keeps the discrete representation of the signal tuned to the filter, as in Eq.(2.1). After processing, the output resampler recovers the original sampling period. This Architecture is depicted in Fig. 2.7, the grey region represents the *FRANCISCO* fabric, the dotted red box in the adaptation fabric depicts the region where BSP is performed, and the red filter within it performs the harmonic filtering between the two resamplers.

The one turn matching delay element is implemented in the BAP region, either in the hardware fabric of the FPGA or the region I of the *FRANCISCO* fabric. Any revolution period can be decomposed

into an integer delay, plus a fractional part. The integer delay can be synthetized efficiently with a dual-port memory, adapting the write and read pointers of the memory in real-time. The fractional part can be synthetized with a VFD as the one used in the resampler.

2.4. Conclusions

The chapter has presented the improvements that the proposed solution brings to the State-Of-the-Art. We have shown first the *tangible contributions* and the benefits that the new Architecture introduces for daily LLRF problems. Then we have addressed the changes and implications that these contributions bring, from a scientific and technical perspective. We want that the new projects planned at CERN benefit from all these improvements. This results in advances for the *Technology Readiness Level* of the LLRF systems and techniques. These advances are grouped around three technical fields where new paradigms are introduced.

We have first presented a new *Signal/Beam Synchronous Processing* Architecture for FPGA. The Architecture is based on a fixed frequency processing clock paradigm to ease implementation in the innovative hardware platforms and devices where the old swept clock poses constraints to the clocking architecture. The solution facilitates the migration of any existing algorithm and the development of new ones.

The second paradigm consists in the *Resampling* in the data-path to tune the processed signals to the processing algorithms. We have developed a new resampler with an arbitrary and real-time variable ratio that is used in the proposed Architecture with fixed processing clock.

Finally, a new *Transient Beam Loading Compensation* scheme that exploits the Architecture and the resampler has been presented. This schema makes it feasible its implementation in the proposed new uTCA hardware for the CERN SPS LLRF systems.

The following chapters are dedicated to each of these three main contributions; Chapter 3 presents the BSP Architecture and Chapter 4 the resampler. The validation of these enabling concepts and resulting performances are presented in Chapter 5. Finally, Chapter 6 shows the new One Turn FeedBack control system and algorithm implementation for Transient Beam Loading Compensation.

Chapter 3

Beam Synchronous Processing Architecture

Abstract: *This chapter presents the Beam Synchronous Processing Architecture proposed in this Thesis. It first depicts the Architecture at functional level. Then, the foundations in which the solution is built are inspected. The chapter continues presenting the key elements used in the Architecture at logic and physical level. It concludes with a feasible implementation for each of these elements.*

3.1. Introduction

This chapter presents the Signal/Beam Synchronous Processing architectural solution developed in the Thesis. We start depicting the proposed Architecture from a functional level, abstracted from any hardware aspects at physical or logical level. No implementation details are given and only the functional behaviour is stated. Then the foundations and concepts supporting the proposed functional Architecture are presented, and the characteristics achieved with this solution are stated. Finally, we present the principal elements in the Architecture and a proposed digital implementation for them, valid for both ASIC and FPGA.

3.2. Proposed processing Architecture

This section presents at functional level the proposed Architecture that makes Beam Synchronous Processing feasible in a digital system with a fixed frequency for either the processing or system clock. No hardware details besides the clock are given. The same data-path supports Beam Asynchronous Processing. We first depict the BSP and BAP units and the element interfacing them within the data-path. Then in the following subsections, we review and develop the theoretical concepts on which the BSP solution is based. We start by justifying the use of a technique based on sampling rate variation for tuning of the signal to the

processing. We continue with a description of how we map the resampling operation within the data-path of the Architecture. Finally, we present the implications of the resampling for the modulation architecture.

3.2.1. High-level functional sketch

The Architecture presents a solution for the processing of periodic signals with known and varying fundamental frequency, the so-called Signal/Beam Synchronous Processing introduced in Chapter 2. From a functional point of view, it solves the need for a data-path that supports such a processing with a fixed system clock in control systems for LLRF applications [18], especially for beam based measurements and controls loops.

Several processing algorithms can be used in these LLRF systems depending on the required functionality; our solution is independent of the spectral bandwidth covered by the processing. It supports algorithms performing narrow bandwidth processing, where only one or a few harmonics of the fundamental frequency are treated [45]. It is also compatible with wide bandwidth processing, where many harmonics are processed [10]. The Architecture is also meant for systems where algorithms not dependent on the spectral content of the signal, Beam Asynchronous Processing, are used [9], [85]. It is generic in the sense that the BSP and BAP can coexist in the same platform and data-path, and can be linked to perform more complex processing on the same signal.

The high-level functional sketch of the proposed architectural solution is depicted in Fig. 3.1. No implementation details are given at this point, but the fixed frequency system clock is depicted, as it is a key element in the solution. The figure sketches the data-path inside the device, an FPGA in this case, the input and output RF front-ends and the RF plant. The data-path contains a blue processing unit that performs the BAP and a red processing unit that performs BSP. Two resamplers, in orange, encapsulate the BSP unit. The input resampler converts the fixed sampling rate at which the data arrives into a variable sampling rate sequence. The output resampler uses an inverse resampling ratio to the input one. This results in the complementary operation so that it recovers the original fixed sampling rate. The BSP is performed between the complementary resamplers. The information about the fundamental frequency of the signal is provided externally to the device. This information is known and used for the BSP tuning. Other parameters linked to the algorithm are also provided to the device.

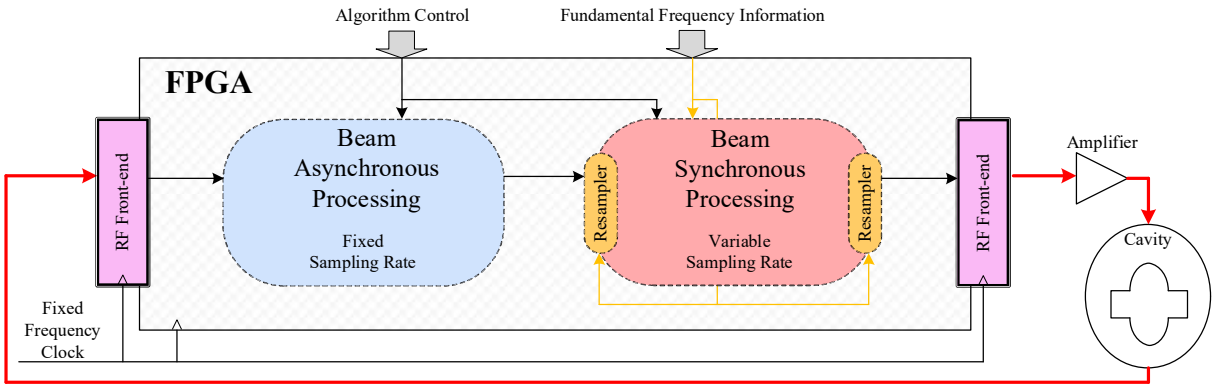


Fig. 3.1. Functional sketch of the Architecture. The BSP unit is surrounded by resamplers performing sampling rate adaptation.

The prototype solution depicted in Fig. 3.1 implements processing and/or up(down)-conversion of signals; LLRF control algorithms commonly perform the processing within the data-path after down-conversion, either to base-band or any other Intermediate Frequency (IF) [86]. As an example, the application of the presented Architecture that will be demonstrated in Chapter 6, down-converts the RF signal to base-band/IF by an analog RF input front-end. Then the signals are sampled, and the algorithms perform either BSP and/or BAP. The processing supports also digital up or down-conversion of the signal frequency from/to base-band to other IFs. Finally, an output stage with a complementary RF front-end brings the signal back from base-band or the IF to the required RF frequency.

It is an all-digital solution; there is no need, when resampling, for analog reconstruction of the processed signal before the DAC output stage in the output front-end, nor any other auxiliary signal or clock. The frequency information of the signal being processed is passed to the Architecture in digital format.

3.2.2. Sampling rate variation

The State-Of-the-Art in Chapter 1 has introduced solutions used for tuning between the response of the processing algorithms and the spectral content of the processed signal. These solutions were based either on functional approaches, as the real-time reconfiguration of the processing architecture [37]–[39], hardware approaches with a sampling of the signal by a variable clock in the ADCs while clocking the processing device with the same swept clock [10], or decomposition of the signal in its multiple spectral harmonics (multiples of the fundamental frequency) using a parallel dedicated system for processing per harmonic [87].

The presented Architecture tunes the digital representation of the spectral content of the signal to the BSP processing by dynamically resampling the digitized signal to a variable rate. It uses only fixed frequency clocks in the sampling and processing sections. To better understand the mechanism, the following subsections elaborate the sampling and resampling concepts and illustrate how the latter performs the tuning.

3.2.2.1. Conventions

The Thesis uses intensively some related concepts such as sampling, processing, clock, and rate. We follow the de-facto conventions used to refer to them in the many signal processing text available in the literature [1], [88], [89]. When referring to the frequency of a real signal we will use capital F as symbol in this document. A real signal $x(t)$ with a tone at a frequency of 100 Hz will be denoted as $F_x = 100$ Hz. When referring to the associated angular frequency, this is represented as $\Omega_x = 2\pi \cdot F_x$ radian/s.

When referring a clock clk_x we refer to a hardware signal at a given real frequency F_x measured in Hz. When referring to a rate r_x we refer to the speed at which something happens.

The sampling clock clk_s is the clock driving the data acquisition; a sampling clock clk_s at a frequency F_s = 100 Hz acquires data at a sampling rate of r_s = 100 sample/s. That clock acquires a hundred data samples, N_{spl} = 100 sample, in one second, and it has a sampling period of T_s = 1/100 s/sample. For this special and specific case, we will refer indistinctly in the Thesis to a sampling clock clk_s by its sampling rate or sampling frequency. We will denote indistinctly the sampling rate as $r_s = f_s$ or the sampling frequency as $F_s = f_s$ as the value is the same and only the units change depending on the context.

The processing or system clock clk_p is the clock driving a processing system that operates the data-path; a processing clock clk_p at a frequency F_p = 100 Hz performs operations at a rate of r_p = 100 operation/s. That clock performs a number of operations N_{op} = 100, with a period of time per operation T_p = 1/100 s. Equivalently the data-path clocked by that processing clock contains a number of processing slots N_{ps} = 100 per second to perform the hundred operations. Again, we will refer indistinctly in the Thesis to a processing clock clk_p by its processing rate or processing frequency. We will denote indistinctly the processing rate as $r_p = f_p$ or processing frequency as $F_p = f_p$ depending on the context.

When referring to a normalized frequency in the discrete representation $x[n]$ of a real signal $x(t)$ having a tone at a frequency F_x that is sampled with a sampling clock at f_s, we will use f_x as symbol. The discrete frequency f_x results from the quotient between the signal frequency and the sampling frequency $f_x = F_x / f_s$ with units of sample^{-1}. We call it normalized frequency as it normalizes the real frequency F_x of the tone in the analog signal to the sampling frequency f_s. When referring to angular normalized frequencies, these are represented as $\omega_x = 2\pi \cdot f_x$ and the units are radian/sample.

3.2.2.2. Sampling

Digital or Discrete Signal Processing systems operate on data samples, $x[n]$, that represent a real continuous-time signal $x(t)$, in the discrete-time domain [1], [90]. The conversion of the real signal to these discrete-time quantized samples is generally performed by Analog to Digital Converters. The process involves two steps. The first stage is sampling, which evaluates the continuous-time signal at discrete-time instants t_n spaced by the sampling period T_s, with $t_n = n \cdot T_s$. Then, quantization translates the signal amplitude in these discrete-time instants to a digital word [91]. The samples $x[n]$ are the digital (or discrete) representation of the $x(t)$ signal value at specific time instants t_n.

The signals at the input and output of the sampling process are depicted in Fig. 3.2. The left plot of the figure depicts in grey the analog signal $x(t)$ sampled by an ADC. This originates the discrete sequence of samples $x[n]$ in beige acquired at f_s.

In this document, sampling is used indistinctly to name the conversion from the real-time continuous signal to its quantized discrete-time samples, including quantization. Sampling therefore maps the real signal $x(t)$ to a discrete representation $x[n]$ sampled at a rate $f_s = 1 / T_s$. This discrete sequence $x[n]$ is later processed by a digital system.

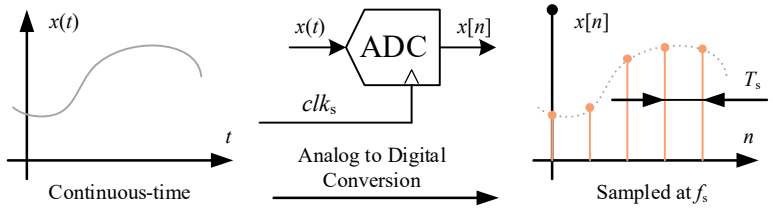


Fig. 3.2. High-level representation of the sampling process. On the left, the real signal $x(t)$ to be acquired by an ADC. In the middle the ADC interfacing the real signal and the discrete representation $x[n]$. On the right, the sequence of discrete samples, spaced by the sampling period T_s.

3.2.2.3. Resampling

Resampling is an operation on a sequence of discrete samples $x[n]$ acquired at a rate f_s that generates a new sequence of discrete samples $y[m]$. The values of this new sequence approximate the values of the real signal $x(t)$ when acquired at a different rate f'_s [72], [92]. This is depicted in Fig. 3.3 where the beige discrete samples of the grey waveform sampled at $f_s = 10 \cdot 10^3$ sample/s, are resampled to a rate of $f'_s = 7.5 \cdot 10^3$ sample/s. The new sequence approximating the real signal is depicted in pink in the right-hand side of the figure.

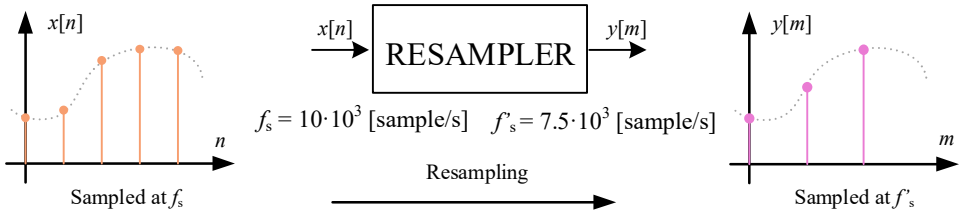


Fig. 3.3. High-level representation of the resampling process. On the left, the input sequence $x[n]$ sampled at a rate f_s. On the right the resulting sequence $y[m]$ after resampling to a rate f'_s.

3.2.2.4. Frequency-domain interpretation of sampling and resampling

Let's now have a look at these operations in the frequency domain. When a real sinewave signal $x(t)$ with fundamental frequency F_0, is sampled at a rate f_s, its representation is mapped to a point in the discrete normalized spectrum [1], [90] at

$$\omega_0 = 2\pi \cdot f_0 = 2\pi \frac{F_0}{f_s} = \frac{\Omega_0}{f_s} \qquad \text{Eq.(3.1)}$$

This is depicted in Fig. 3.4 where a tone at $F_0 = 2$ kHz is sampled with a clock clk_s at $F_s = 10$ kHz. The figure depicts in the left side the tone in the analog spectrum in beige and the sampling clock in purple. The sampling maps the tone in the discrete spectrum $X(e^{j\omega})$ to an angular discrete frequency at $\omega_0 = 2\pi \cdot 0.2$ radian/sample. The right half of the figure depicts the mapping with the tone again in beige and the sampling rate in purple. The figure spans through the first two Nyquist zones [93]; the first ranging from DC to half the sampling rate, and the second Nyquist zone starting at half the sampling rate up to the sampling rate as the upper limit.

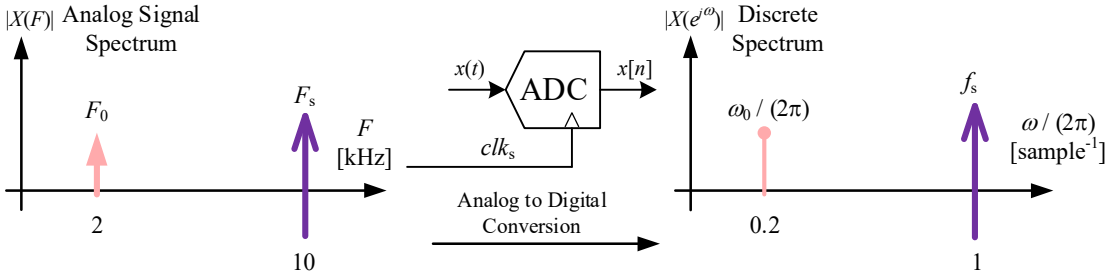


Fig. 3.4. Frequency-domain representation of the sampling process; mapping of F_0 to ω_0 in the discrete normalized spectrum.

When this discrete signal is resampled, this results in a change of the mapping of its discrete spectrum. This is depicted in Fig. 3.5; the left side shows the original digitized tone at a rate of $f_s = 10 \cdot 10^3$ sample/s (purple arrow) mapped to $\omega_0 = 2\pi \cdot 0.2$ radian/sample (beige line). The right-hand side depicts the ideal result after resampling to a new rate of $f'_s = 7.5 \cdot 10^3$ sample/s (brown arrow); the tone is shifted to $\omega'_0 = 2\pi \cdot 0.26$ radian/sample (pink line).

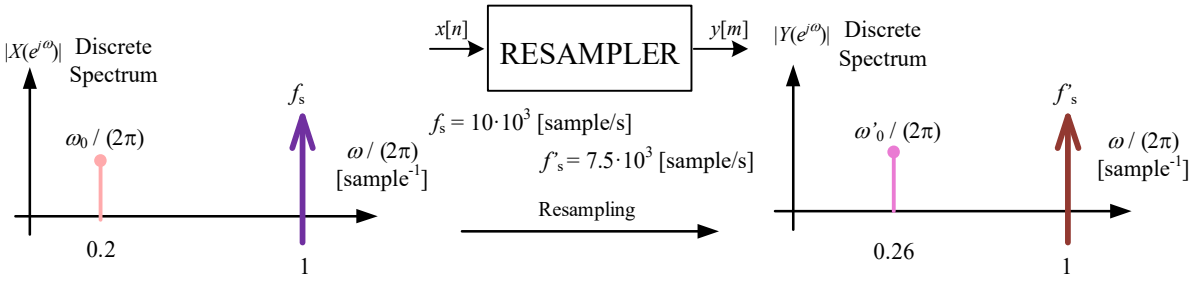


Fig. 3.5. Frequency-domain representation of the resampling process; the discrete normalized spectrum of F_0 is re-mapped from ω_0 to ω'_0.

The resampling of a discrete signal modifies and changes the mapping of frequencies in the discrete frequency domain. The new discrete representation $Y(e^{j\omega})$ confines the spectrum within frequencies spanning up to the new sampling rate. It is hence possible to shift the mapping of a point in the discrete spectrum $X(e^{j\omega})$ (sampled signal) by changing the sampling rate. This makes it possible to tune that discrete spectrum to the desired point at the discrete normalized frequency ω_{proc} where the processing can be defined. This is, therefore, a suitable alternative for tuning the signal to the processing.

This approach is the inverse of the classic reconfiguration of the processing elements; instead of adapting the frequency ω_{proc} in the response of the processing to the spectral content of the signal $X(e^{j\omega})$, the resampling tunes the signal representation $Y(e^{j\omega})$ to a fixed processing frequency ω_{proc} in the discrete normalized spectrum. This is depicted in Fig. 3.6; the processing (red band-pass filter) is centred at a fixed discrete frequency of $\omega_{proc} = 2\pi \cdot 0.26$ radian/sample. In the left side of the figure, before resampling, the discrete representation of the tone lies at $\omega_0 = 2\pi \cdot 0.2$ radian/sample using a sampling rate of $f_s = 10 \cdot 10^3$ sample/s. We then use resampling of $x[n]$ to tune the discrete spectrum $Y(e^{j\omega})$ of the signal $y[m]$ to the fixed processing frequency ω_{proc} according to Eq.(3.1). This is depicted in the right side of the figure: The resampled tone in pink now matches the filter in red. For this purpose, the sampling rate is modified to $f'_s = 7.5 \cdot 10^3$ sample/s, depicted in brown in the figure.

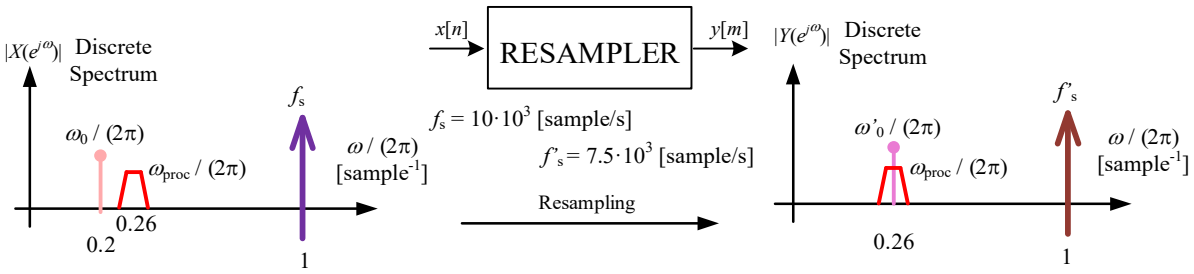
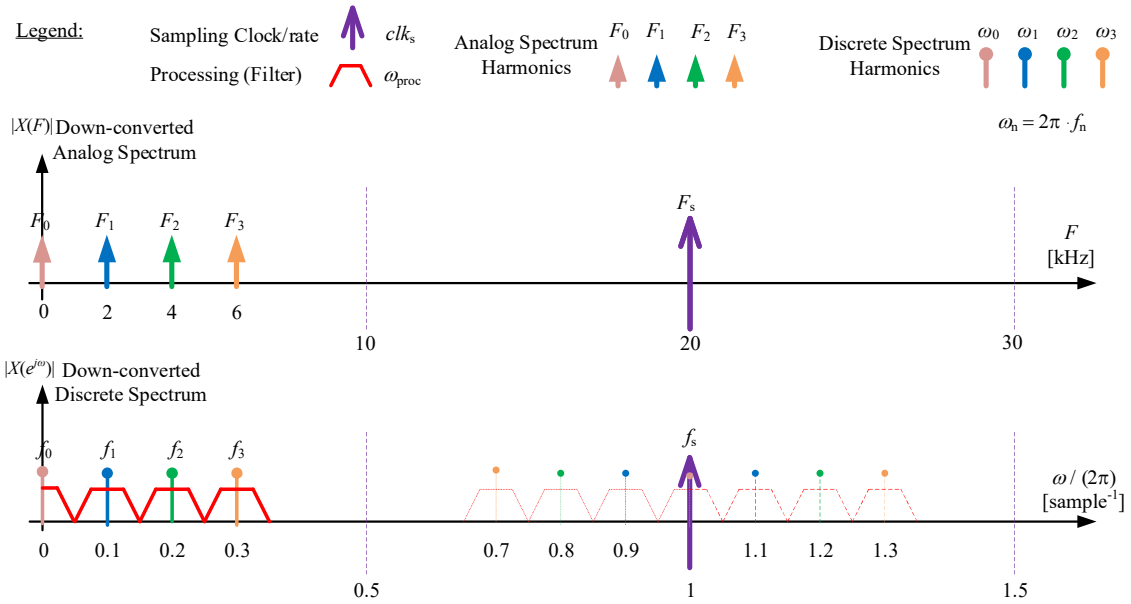


Fig. 3.6. Representation of the resampling process as element to tune the discrete representation of the signal ω_0 to a predefined fixed processing ω_{proc}. The fixed processing ω_{proc} (red band-pass filter) remains constant defined at $\omega_{proc} = 2\pi \cdot 0.26$ radian/sample.

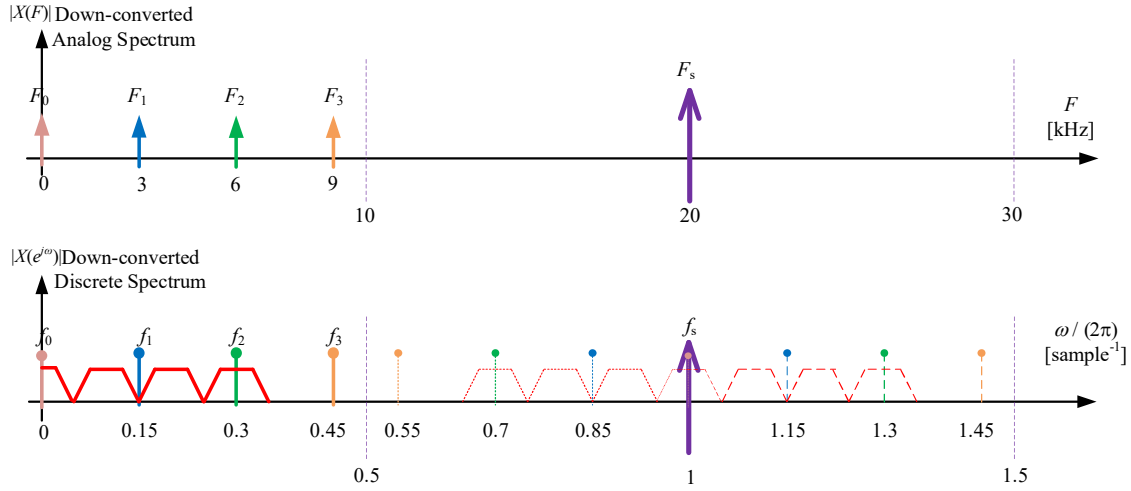
Let us now analyse how we can use this in our BSP application. Consider the case of an accelerator at the beginning of the momentum ramp; a pick-up signal $x_{rev}(t)$ contains the revolution frequency F_{rev} (fundamental harmonic) and harmonics at integer multiples of the fundamental. Assume that the fundamental is at F_{rev} = 2 kHz, and consider three harmonics at $F_{rev\_1} = 2 \cdot F_{rev} = 4$ kHz, $F_{rev\_2} = 3 \cdot F_{rev} = 6$ kHz and $F_{rev\_3} = 4 \cdot F_{rev} = 8$ kHz. This signal $x_{rev}(t)$ is down-converted to base-band generating $x(t)$. During the ramp the fundamental harmonic is kept constant at DC by adjusting the Local Oscillator (LO) of the mixer. The signal $x(t)$ is then sampled generating $x[n]$.

Fig. 3.7 depicts two instants of the acceleration process. In Fig. 3.7(a), which corresponds to the beginning of the ramp, the analog and discrete spectrums of $x(t)$ are presented. The sampling rate at that time is $f_s = 20 \cdot 10^3$ sample/s, depicted with a purple arrow in the figure. The down-converted fundamental harmonic F_{rev} is now at $F_0 = 0$ kHz (in beige) and the harmonics F_{rev_1}, F_{rev_2} and F_{rev_3} at $F_1 = 2$ kHz, $F_2 = 4$ kHz and $F_3 = 6$ kHz (in blue, green and orange) respectively. The discrete spectrum of the sampled signal follows the same colour convention. In that situation, the fundamental lies at DC and the three harmonics, at $\omega_1 = 2\pi \cdot 0.1$ radian/sample, $\omega_2 = 2\pi \cdot 0.2$ radian/sample and $\omega_3 = 2\pi \cdot 0.3$ radian/sample. The discrete processing of the LLRF, the bank of pass-band filters in red in the figure, is at injection centred in the discrete frequency of each harmonic (beginning of the accelerating ramp). This processing will remain fixed (without any reconfiguration) at these discrete spectral positions during all the acceleration.

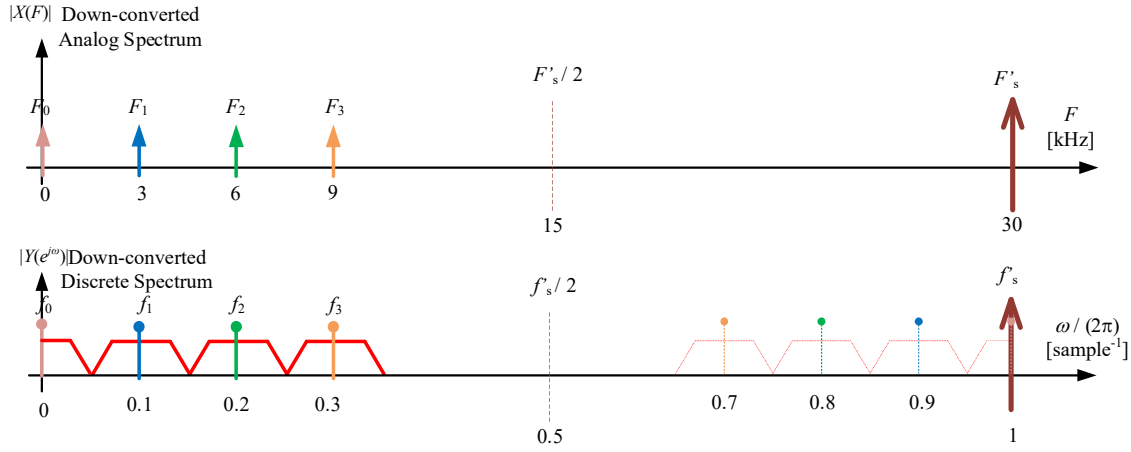
Then the momentum ramp of the accelerator accelerates the beam; the spectrum of the signal $x_{rev}(t)$ sweeps the fundamental from F_{rev} = 2 kHz at injection to F_{rev} = 3 kHz at extraction. The front-end adjusts the LO continuously and keeps the fundamental at DC in the down-converted signal $x(t)$. However, the position of the other harmonics is not constant in base-band. They suffer a homothetic transformation as the momentum ramp advances. They change in position and spacing in the spectrum of the down-converted signal. Fig. 3.7(b) depicts the situation at the end of the ramp (extraction) when the signal is sampled using the same fixed frequency sampling clock clk_s at a rate $f_s = 20 \cdot 10^3$ sample/s. The analog spectrum remains with the fundamental at DC $F_0 = 0$ kHz but the harmonics are now at $F_1 = 3$ kHz, $F_2 = 6$ kHz and $F_3 = 9$ kHz. The sampling maps the harmonics to the discrete spectrum at $\omega_1 = 2\pi \cdot 0.15$ radian/sample, $\omega_2 = 2\pi \cdot 0.3$ radian/sample and $\omega_3 = 2\pi \cdot 0.45$ radian/sample. In this situation, the bank of filters is completely misaligned, as no reconfiguration has been done.



(a) Analog and Discrete Spectrum at the beginning of the accelerating ramp, using a fixed frequency sampling clock



(b) Analog and Discrete Spectrum at the end of the accelerating ramp, using a fixed frequency sampling clock



(c) Analog and Discrete Spectrum at the end of the accelerating ramp, emulating a variable frequency sampling clock with resampling of the discrete samples

Fig. 3.7. Representation of the beam signal in the acceleration process of the example in section 3.2.2.4: (a) depicts the spectrum at the beginning of the ramp, (b) at the end of the ramp when the sampling clock is a fixed frequency one, and (c) at the end of the ramp with a swept clock (or resampling).

Consider now that the signal $x[n]$ is resampled to $y[m]$. The sampling rate f'_s of the new signal $y[m]$ is dynamically modified during the ramp. The resampling ratio R is proportional to the variation of frequency of the fundamental harmonic F_{rev} (increase in revolution frequency).

Fig. 3.7(c) depicts the end of the accelerating ramp but now following this technique. At the end of the ramp, the variable sampling rate in the new signal $y[m]$ reaches $f'_s = 30 \cdot 10^3$ sample/s. In this case, the homothety is compensated and the mapping of the harmonics in the discrete spectrum remains at a constant position, as a result of Eq.(3.1). The figure depicts first the analog spectrum of the signal that remains the same (only the new sampling clock in brown moves). The discrete spectrum maps the harmonics to the same position as at injection; $\omega_0 = 2\pi \cdot 0$ radian/sample, $\omega_1 = 2\pi \cdot 0.1$ radian/sample, $\omega_2 = 2\pi \cdot 0.2$ radian/sample and $\omega_3 = 2\pi \cdot 0.3$ radian/sample. The resampling of the sweeping signal results thus in the fixed discrete position of the harmonics. As a result of this, the filter bank, whose frequency response has not been changed, remains also in tune with the harmonics.

The proposed Architecture is based on this resampling example. Instead of sampling the down-converted signal with a swept clock, we use resampling. We resample the signal acquired with a fixed frequency clock to obtain the same result as when using the swept clock. This tunes the discrete representation ω_x of the signal frequency to the frequency ω_{proc} at which the processing algorithm has been defined and that can remain fixed.

3.2.3. Resampling sandwich

The previous point has presented sampling rate conversion as the enabling solution for tuning between signal and processing. Other elements or algorithms which operate at a fixed sampling rate in the same data-path need, however, to interface the processing at a variable sampling rate, the BSP unit. This is the case for instance of ADCs, DACs and the BAP unit that use fixed frequency clocks for the acquisition and the processing.

The Architecture is capable of supporting both BSP and BAP by proposing a solution based on the encapsulation of the BSP unit within two resamplers, depicted in orange in the functional sketch of the Architecture in Fig. 3.1. This makes it possible to interface the processing at a variable sampling rate to the rest of the system. The BSP is hence performed and encapsulated between the two resamplers, within the so-called "*resampler sandwich*". In that unit of the Architecture, the sampling rate is adapted synchronously to the spectral content of the signal. The "*adaptation to the content*" follows the mechanism presented in section 3.2.2.4. The resamplers perform the conversion between sampling rates, obfuscating the fixed sampling elements in the data-path from the BSP unit. This makes the BSP resampling based solution compatible with any data-path that in some stage uses a fixed sampling rate. Section 3.3 will show the implementation details. It presents how a fixed processing clock can host a data-path with variable sampling rate.

3.2.4. Modulation architecture

Control algorithms for LLRF systems normally perform the processing in base-band. This concept has been presented in section 3.2.2.4 where the signal $x_{rev}(t)$ was down-converted to base-band, $x(t)$, before processing. In systems where the RF is fixed, the down-conversion of the signals to base-band can be done with a static LO. In the SPS, the swept RF [50] poses a further requirement for the presented BSP Architecture. We need a varying LO to down-convert the sweeping RF to base-band during the entire accelerating ramp.

In small machines, the RF signals can be sampled directly and the down-conversion can be done digitally by means of a digital mixer driven from a Numerically Controlled Oscillator (NCO) [71]. Larger machines rely on RF demodulation [85]; a narrowband RF signal is down-converted to an IF with an analog RF front-end using a signal coming from a LO. This IF signal is sampled and a second digital complex In-phase and in-Quadrature (I/Q) mixer down-converts it to base-band, resulting in two components I/Q whose bandwidth is much smaller than the Nyquist rate of the digital processing. An output stage with a complementary up-converter brings the signal back to the required RF frequency.

The presented Architecture supports all these options. Fig. 3.8 presents an example incorporating all of them. In the figure, the RF is first mixed with an analog LO resulting in an intermediate frequency IF1. The signal at this IF1 is sampled by an ADC. The RF component of the discrete signal is then digitally down-converted to base-band within the FPGA using a second swept LO2. This LO2 is synthetized by an NCO. The resulting IF2 (with the RF component translated to DC) is processed by a BSP algorithm. At the output of the BSP, a second digital mixer performs up-conversion of the processed signal to an IF3. In the figure, this LO3 is also swept and translates the RF from DC to a sweeping IF3. There the processing can perform for instance BAP (this is the case in the figure). BAP algorithms are in general not dependant on the instantaneous frequency of the RF, for instance, compensation of the reconstructing response of the DAC. Finally, a fourth mixer recovers the RF signal sent to the amplifier after mixing with a fixed analog LO. The digital LOs are reconstructed locally in the FPGA with the digital information received, the RF frequency and revolution frequency.

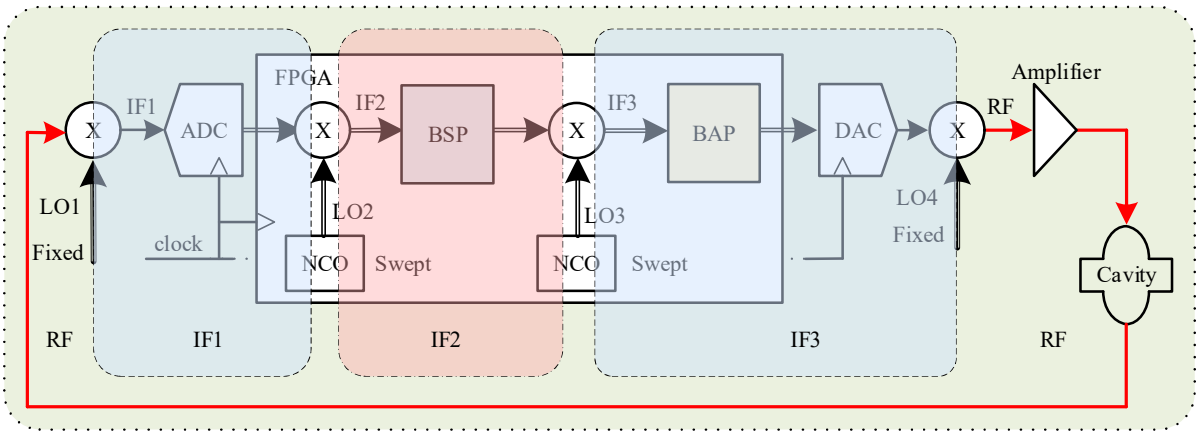


Fig. 3.8. High-level representation of the different zones performing processing at different intermediate frequencies (IFs). In the figure, the RF green zone is the region where the RF is at its nominal value. The IF1, IF2 and IF3 depict different regions in which the RF is down-converted to other intermediate frequencies.

3.3. Implementation of the Processing Architecture

This section presents the implementation details of the proposed Architecture, the data-path and other supporting elements. We first introduce the high-level sketch of the Architecture at logical and physical level including these supporting elements needed to make the functional model presented in section 3.2 feasible. We introduce the concept of decoupled data-path. Then in the following subsections, we present how to implement such a data-path within an adaptation fabric in a digital device, for instance, an FPGA. We show the required interfaces at logic and physical level performing adaptation of the data-path signals between coupled and decoupled regions. Recall that we want to use the BSP Architecture inside a feedback loop, as the example in Fig. 3.8, this is the reason why we place the second resampler as output stage recovering the original sampling rate. We conclude presenting the functional model of the resampler, that is developed in detail in Chapter 4, and the operational configuration of the Architecture.

3.3.1. Conventions

The following sections and chapters intensively elaborate on hardware signals and variables. A convention has been stablished in the Thesis to denote signals and relates them to the variables that provide its value. Hardware signals are represented with Courier font and no italics, `x`. When such a signal is the physical implementation of a variable, the name used in the variable is the same when possible and is represented in Times New Roman font and italics, *x*. A subscript is added when needed to variables and signals to univocally identify them among related ones, for instance, $\mathtt{x}_{\mathtt{in}}$, $\mathtt{x}_{\mathtt{out}}$, x_{in} and x_{out}. In the case of clocks, $\mathtt{clk\_proc}_{\mathtt{in}}$ is the signal, at the input port of an entity, mapping the processing clock clk_{p} with frequency F_{p}.

3.3.2. High-level implementation sketch

The high-level implementation model of the proposed architectural solution supporting BAP and BSP is depicted in Fig. 3.9. It maps the functional model of Fig. 3.1 depicting, in addition, the hardware resources and blocks needed to make the implementation feasible. The mixing stages of the RF front-ends are not included as we only focus on the data-path, however, the ADC and DAC are included in Fig. 3.9 to denote the fixed frequency clocking scheme used within the Architecture.

The implementation model contains two fabrics, the "*Hardware fabric*" in white and blue, and the "*FabRic with Adaptive aNd deCoupled clockIng for SynChronous prOcessing (FRANCISCO) adaptation fabric*" in grey, yellow and red. The hardware fabric (the device hosting the Architecture, for instance, an ASIC or FPGA) is used for implementation of the BAP where a standard data-path (*coupled*) is used. The adaptation fabric is used for implementation of the BSP where a *decoupled* data-path is used. In a decoupled data-path the processing and sampling clocks operate at different frequencies, and the number of processing slots and samples needs not to be the same. Such a data-path requires an extra hardware signal, `valid`, to flag the processing slots containing valid data. In a coupled data-path these two clocks operate at the same

frequency, all the processing slots are hence populated, and this line is not necessary. The hardware and *FRANCISCO* fabrics coexist in the same device and need to communicate, note that the *FRANCISCO* fabric is built on top of the hardware fabric. Two interfaces, the "*MultiplE Rate and Clocking interfacE for Data procEssing and Sampling (MERCEDES) Decouple and Couple interfaces*" in green in the figure are these unique data-path points linking the two regions. A Frequency-Locked Loop, the "*JOintly Averaged and QUaNtized rAtio (JOAQUINA) Frequency-Locked Loop*" is set around the *MERCEDES Couple* interface and the output resampler to correct for the truncation errors in the ratio signals of the resamplers, in red in Fig. 3.9. These ratio signals are `r_in` for the input resampler and `r_out` for the output resampler. The correction signal `corr_R` is added to `r_out` before the output resampler.

In the *FRANCISCO* adaptation fabric, there are two sub-regions. The first, between the *MERCEDES* interfaces and the resamplers, is a sub-region where the sampling rate is fixed and equal to the hardware fabric. In the figure, that sub-region contains only the signals `d_dcpl` and `valid` connecting the *MERCEDES* interfaces and the resampler. These signals compose a segment of the decoupled data-path in which no processing is performed. That segment can nevertheless be used to implement BAP. The second sub-region is encapsulated within two resamplers, in orange in the figure, and contains the red BSP processing unit of the data-path (where the sampling rate varies). The data-path remains decoupled. This second sub-region is hence defined as the "*resampling sandwich*".

The resamplers are responsible for the sampling rate conversion in the data-path, and the resampling ratio of the input resampler dictates the sampling rate within the sandwich. They operate with the fixed frequency processing clock of the *FRANCISCO* fabric, the signal `clk_dcpl` in the figure. These

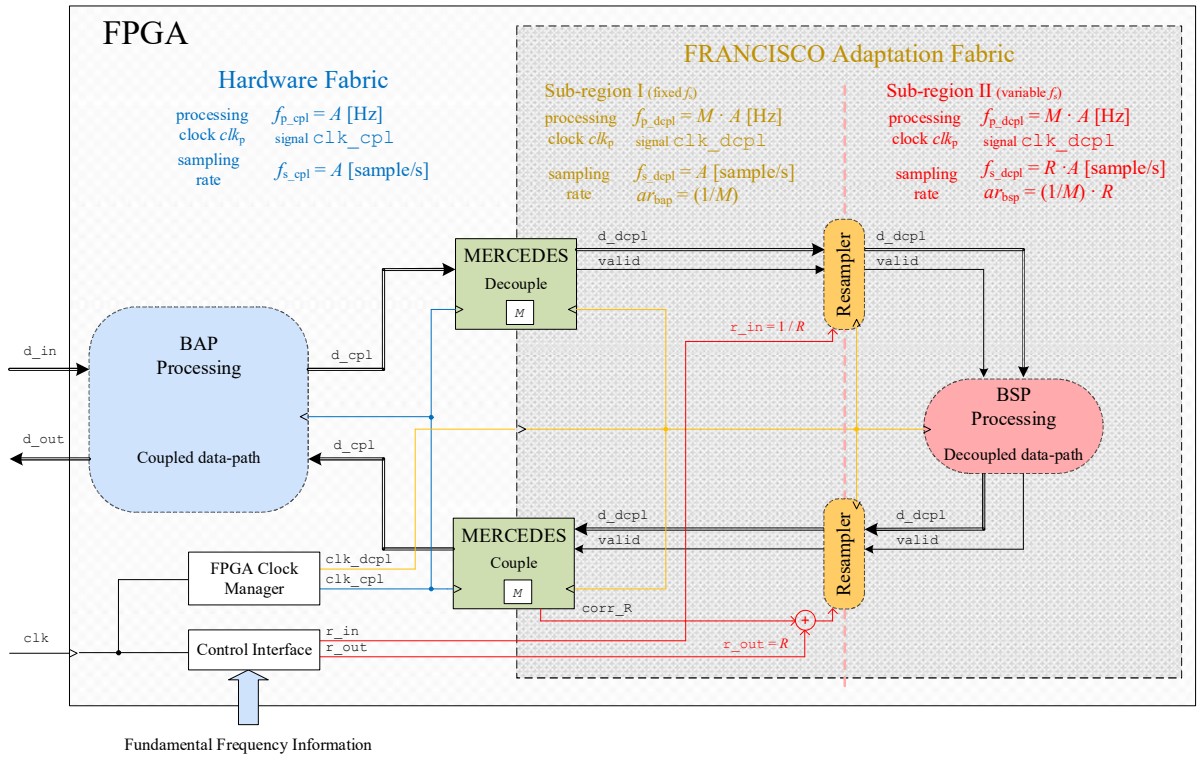


Fig. 3.9. High-level sketch with the implementation of the proposed Architecture in an FPGA.

different processing clocks and the ratio signals for resamplers are depicted with different colours in Fig. 3.9. The clocks related to the coupled data-path in the hardware fabric use the blue. The ones related to the decoupled data-path in the adaptation fabric use yellow. The regions of the adaptation fabric having a sampling rate equal to the hardware fabric will be depicted in yellow, the ones with a sampling rate that is variable use the red colour. A dashed vertical line in pink denotes the partition of the variable sampling rate (right) and fixed sampling rate (left) sub-regions in the *FRANCISCO* fabric.

At the functional level, the resamplers are hence the interfaces between the different sampling rates and processing units in the data-path. At the logic and physical level, the *MERCEDES* are the interfaces between the fabrics in the device (and the coupled and decoupled segments of the data-path), regardless of the sampling rate that crosses these interfaces unchanged. We call "*BAP region*" the hardware fabric where the data-path has a fixed sampling rate, and the sub-region in the *FRANCISCO* fabric where the data-path has a fixed and identical sampling rate (sub-region I in Fig. 3.9). We call "*BSP region*" the sub-region in the *FRANCISCO* fabric where the data-path has a variable sampling rate (sub-region II in Fig. 3.9).

The solution focuses on the use of a fixed processing clock. This is to be compatible with the use of all hardware features of new hardware devices, such as FPGAs, that might not be feasible with a swept processing clock [62], [94]. These devices are substituting Digital Signal Processors in LLRF systems. They offer much higher processing rates, customization of the hardware, integration of resources and parallelism. These devices are best suited to a fixed system clock. The Architecture replaces the swept clock of the old solutions with the fixed frequency clock; this makes the implementation in uTCA systems suitable [19].

3.3.3. Conceptual decoupled data-path

Section 3.2 has introduced resampling as the functional solution to tune the spectral content of the signal to the discrete processing with fixed frequency response in the BSP region. This operation is used to modify the data sampling rate as it is acquired with a processing clock having a fixed frequency at the beginning of the data-path, the ADC in Fig. 3.9. This data leaves the processing device at the end of the data-path towards a DAC that also uses a fixed frequency processing clock. The Architecture and the data-path need therefore to support and accommodate fixed and variable sampling rates, the BAP and BSP regions. As both regions are implemented in the same processing device, the resampling operation needs to be as well.

Since we want to follow the momentum ramp of the accelerator, this sampling rate conversion approach must implement a variable ratio, which should be feasible from a functional point of view. We use the input resampler to define the sampling rate in the BSP region; the data arrives at the resampler at a constant sampling rate, while the output rate is variable. This has, however, implications at hardware level for both the resampler and the data-path; the clocking architecture and hardware need to support this variable sampling rate.

Take the case of the hardware at the output port of the input resampler; when it operates with a clock that has the same frequency as the sampling rate, this clock needs to vary and adapt its frequency (we say that the frequency of the clock and the sampling rate are coupled). This situation is similar to the first solution used for the implementation of the OTFB; the entire system clock was swept [10]. In our case, instead of the entire system, only a part of the data-path (between the resamplers) sweeps the clock. We want to avoid this, otherwise many of the problems presented in the previous chapters would be reproduced.

We hence need a feasible solution supporting the implementation in a digital device of a data-path with a variable sampling rate but using a fixed clock for the whole hardware. Again, from a functional point of view, when we study or develop an algorithm, we think of the data-path regardless of its implementation (the hardware clock level); in that situation, the relevant information for us is the sampling rate of the data. We abstract the sampling operation from the rate at which the data processing is performed. This approach can be extrapolated to the implementation; the processing or system clock clk_p and the sampling rate f_s of the data-path (equivalently the clock clk_s with frequency f_s used for sampling of the data samples) do not need to operate at the same frequency.

For simple systems the frequency of these two clocks is identical, the data-path is coupled; the same clock or one with the same frequency is used in the ADCs, DACs and FPGA. If this is the case, all the processing cycles in the data-path operate on valid data; all the processing slots contain data samples. However, when the processing requirements are more demanding this paradigm limits the exploitation of hardware capabilities, the use of new devices, and the implementation of more complex algorithms.

To overcome this limitation, the processing clock can operate the data-path at a different frequency than the one used for the sampling. In the case of the resampler, the sampling rate at the input and output will be different from the frequency of the hardware clock operating the data-path. The frequency of these two clocks, sampling clock f_s and processing/system clock f_p, is hence different and the data-path is decoupled. The paradigm abstracts the data-path samples from the hardware; the sampling frequency of the data is decoupled from the hardware processing clock.

This paradigm requires only the processing rate to be higher than the sampling rate. This requirement ensures that the data-path can absorb all valid data samples not overflowing the hardware; the data-path operates at a higher frequency and can thus perform more operations than samples are available. We say that there are more processing slots available in the hardware than samples in the data-path. The decoupling idea is not new [70], [95]. It is widely used to implement time multiplexing of hardware resources within digital systems, or for interleaving or serializing data.

Fig. 3.10 depicts the interleaving of three data channels in a single data-path. The three channels are merged in the data-path by populating iteratively one processing slot per channel. Offline data processing in a computer is also a similar concept. In that case, the processor operates the data at an arbitrary

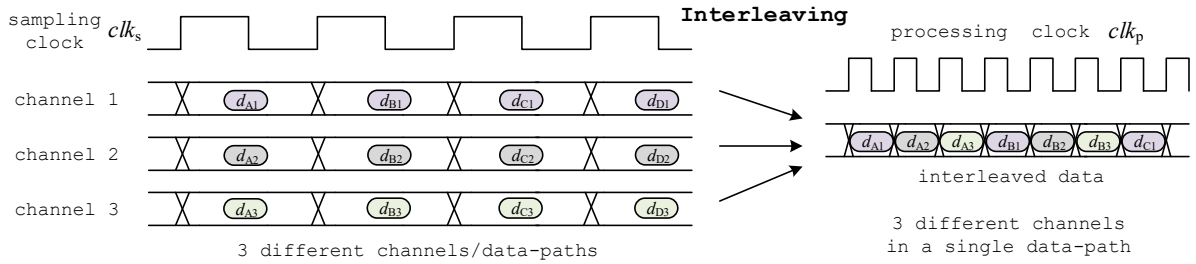


Fig. 3.10. Representation of the interleaving process of three channels. The resulting data-path operates at a clock three times faster than the sampling clock. The samples of the different channels are interleaved within the data-path.

system clock and later, it is the user who interprets and relates the samples to the sampling clock used in the acquisition.

The motivation for this paradigm is in our case to make feasible a variable sampling rate in the data-path of the BSP region between resamplers. Thus, we avoid the variable frequency processing clock by decoupling it from the sampling rate; we use fixed-frequency clocks driving the hardware, both for ADC and FPGA, and we decouple the sampling rate f_s from the processing clock clk_p.

We want to use this paradigm in a feedback system, as in the example in Fig. 2.7, that requires processing in real-time. This type of systems is implemented normally with a fixed sampling rate in the data-path as the variable sampling rate varies the number of samples in the data-path. We are hence more interested in computing the sampling rate in the decoupled data-path based on the average number of samples within it; the sampling rate is a function of the number of samples present in the available processing slots for a given period of time τ (this period of time is not the sampling period T_s but the amount of time in which the average sampling rate is evaluated). We call the relation between samples populating the processing slots and the available processing slots the activation rate *ar* of the data-path. The number of processing slots in a given period of time τ is $N_{ps} = f_p \cdot \tau$. In that same period of time, an ADC at a sampling rate f_s acquires a number of samples $N_{spl} = f_s \cdot \tau$ sample. By populating the data-path with these samples, the activation rate becomes

$$ar = N_{spl} / N_{ps} = (f_s \cdot \tau) / (f_p \cdot \tau) = f_s / f_p \qquad \text{Eq.(3.2)}$$

Recall that there is no physical sampling clock as such in a decoupled data-path; the sampling rate f_s is hence an abstraction computed based on the activation rate and the operation rate of the processing clock

$$f_s = ar \cdot f_p \qquad \text{Eq.(3.3)}$$

When all the processing slots are populated, we reach the maximum number of samples in the data-path. In this case, the sampling rate f_s equals the frequency f_p of the processing clock clk_p; the data-path operates at its maximum sampling rate. When not all the processing slots are populated, the sampling frequency is lower than the processing clock. The distribution of the populated processing slots does not need to be periodic. This concept is depicted in Fig. 3.11.

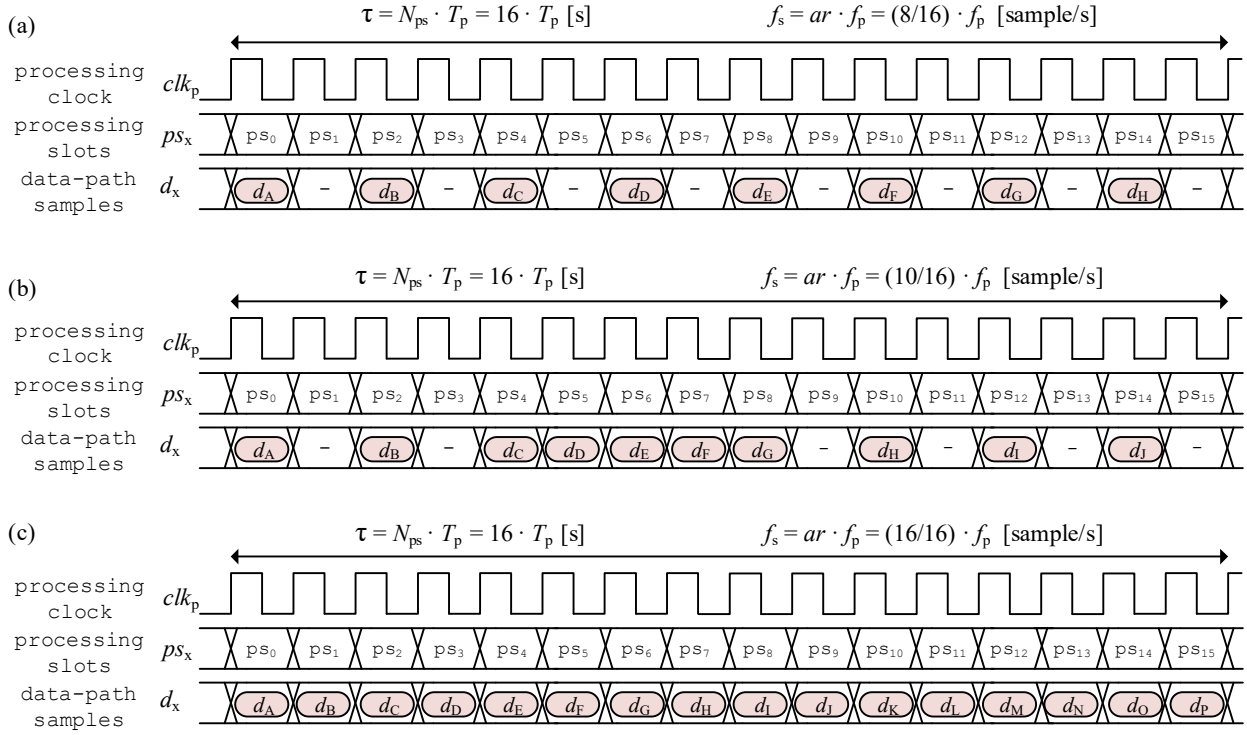


Fig. 3.11. Representation of the distribution of processing slots, the processing clock and samples in the data-path for different sampling rates. The activation rate *ar* dictates the number of occupied processing slots for a given period of time τ. In (a) the activation rate is *ar* = 8/16, in (b) the activation rate is *ar* = 10/16, and in (c) the activation rate is *ar* = 16/16.

In Fig. 3.11(a) the activation rate is *ar* = 8/16, this results in one out of two processing slots populated. This gives a sampling rate in the data-path for a time τ of $f_s = (8/16) \cdot f_p$ sample/s. In Fig. 3.11(b) the activation rate is *ar* = 10/16, ten out of sixteen processing slots are populated, with a burst in the middle. This gives a sampling rate in the data-path for a time τ of $f_s = (10/16) \cdot f_p$ sample/s. In Fig. 3.11(c) the activation rate is *ar* = 16/16, all processing slots are populated. This gives a sampling rate in the data-path for a time τ of $f_s = f_p$ sample/s, the maximum sampling rate supported by the data-path. Such a decoupled data-path solves our problem for a variable sampling rate operated with a fixed frequency processing clock.

3.3.4. Beam Asynchronous Processing fabric

The proposed Architecture supports both BAP and BSP. In the BAP region, we do not vary the data-path sampling rate; the processing does not need to tune the spectral content of the signal. In this case, the hardware fabric, with a data-path that has the sampling clock clk_s and processing clock clk_p coupled, is more suitable. Even though possible, the implementation of BAP in the decoupled data-path in sub-region I of the *FRANCISCO* fabric results in a more complicated design. The preferred option is thus to use the hardware fabric where the processing clock is a fixed frequency clock with the same frequency as the sampling clock used to acquire the data, $f_p = f_s$.

As in this case there is no abstraction of the hardware in the data-path, the FPGA hardware fabric is used as it is. The same clock, or a clock at the same frequency, drives the ADCs and FPGA, and the pipeline in the data-path does not need any special clocking architecture.

A schematic representation of this standard data-path is depicted in Fig. 3.12. Two registers encapsulate a "*processing cloud*" within the data-path of the signal `d_in`. In this case, the "*cloud*" contains only combinatorial logic independent of the clocking signal. The fixed frequency clock drives only the registers pipelining the data-path. Nonetheless, the processing can contain more complex elements, such as DSP slices [96]. In that case, the clock is also fed to these elements. The simple model of the figure remains valid as these more complex elements can be modelled and reduced to simple combinatorial logic pipelined between registers.

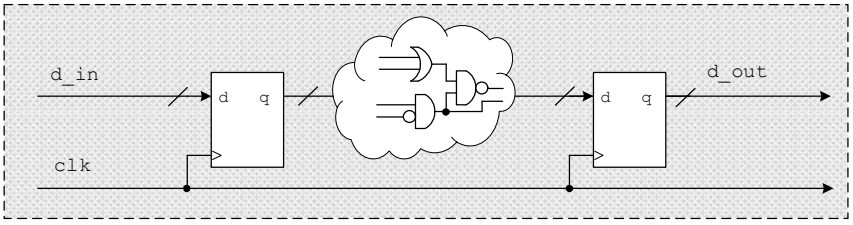


Fig. 3.12. Representation of a coupled data-path with a *cloud* of logic encapsulated within two pipeline registers.

Fig. 3.13 depicts the relation between clocks, processing slots and data samples in this data-path. In this case, the processing and the sampling are performed at the same rate; the clocks, clk_p and clk_s have the same frequency, $f_p = f_s$, and all the processing slots are populated with data. Note that no extra logic elements or signals are needed in the data-path. The processing clock (signal `clk` in Fig. 3.12) is enough to drive and operate the pipeline registers.

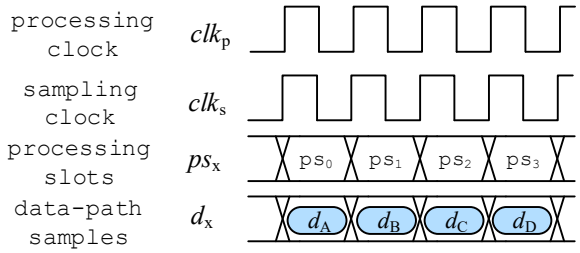


Fig. 3.13. Representation of the relation between the processing clock, the sampling clock, processing slots and data samples in a coupled data-path.

3.3.5. Beam Synchronous Processing *FRANCISCO* fabric

The decoupled data-path implemented in the BSP region is based on the adaptation fabric *FRANCISCO*, introduced in Chapter 2 and section 3.3.2. This adaptation fabric is built on top of the FPGA hardware fabric (BAP processing). It combines logic elements and the FPGA clocking architecture to support the decoupled data-path concept presented in section 3.3.3. It adds very little extra complexity to the system; the processing algorithms implemented in the fabric require an extra hardware line, *`valid`*, one bit wide in the data-path. It is used to indicate which processing slots contain valid data, and which are void slots.

A schematic representation of a segment of such a data-path implementing BSP based on the *FRANCISCO* fabric is depicted in Fig. 3.14. The figure shows an implementation in which the `valid` line is propagated in parallel to the data-bus signal of the data-path. At the output of the data-path segment, the

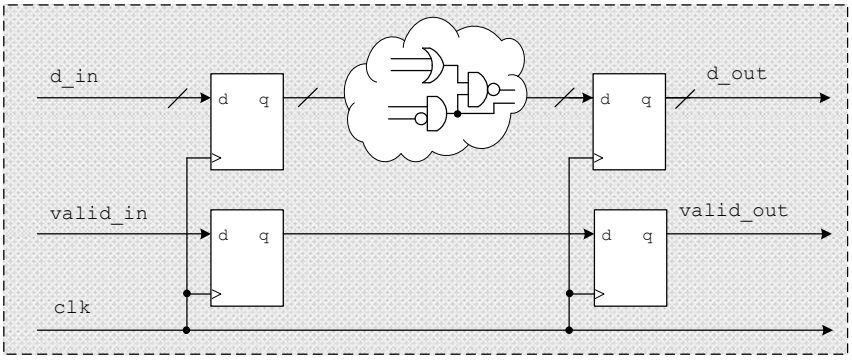


Fig. 3.14. Representation of a decoupled data-path with a *cloud* of logic encapsulated within two pipeline enabled registers.

qualification signal `valid` is used to distinguish samples with valid data. Fig. 3.15 depicts the relation between clocks, processing slots and data samples in such a data-path implemented in the *FRANCISCO* fabric. The decoupling makes the processing clock clk_p and sampling clock clk_s operate at different frequencies. Recall that the sampling clock does not exist as a physical signal in the decoupled data-path. It is just an abstraction indicating the average sampling rate of the data in the data-path. Not all processing slots ps_x are populated, and the `valid` signal flags which ones contain valid data d_x and which ones are void or invalid (denoted with –).

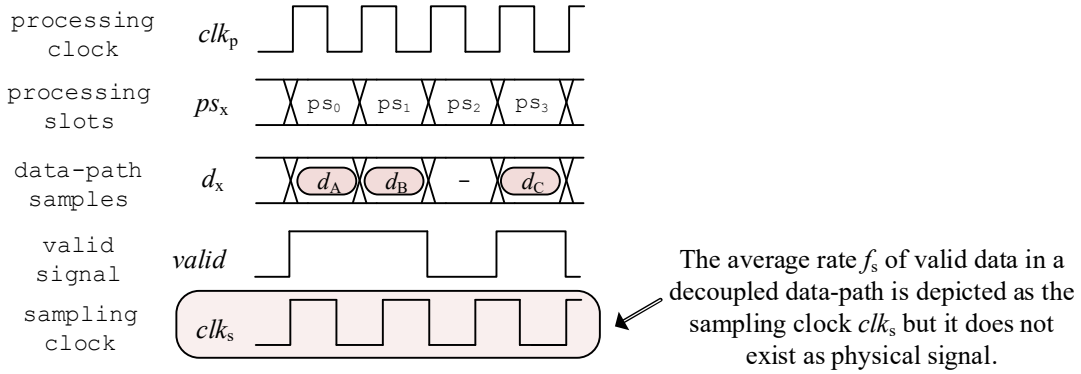


Fig. 3.15. Representation of the relation between the processing clock, the average sampling clock, processing slots, the data samples and the valid signal in a decoupled data-path.

The decoupling paradigm makes the use of the clocking architecture of the FPGA feasible with a variable sampling rate. It solves the need for a fixed frequency clock that does not interfere with PLLs as the swept clock would do. The Architecture has been developed with FPGA as target technology, however, it can be migrated to ASIC technologies easily.

3.3.6. The ratio truncation and inversion

The high-level functional Architecture presented in section 3.2 does not consider the quantization effect on real-valued variables, that are mapped to signals when discretized and implemented in a digital processing system. This is the case for instance of our Architecture being implemented in an FPGA or an ASIC. These quantization effects are especially relevant for the signals containing the resampling ratios used in the resamplers.

Our BSP region is implemented within two of these resamplers. They modify the sampling rate of their input data-path according to their resampling ratio R. In Fig. 3.9 the fundamental frequency

information used to compute that value is externally provided by the control system via a control interface; there the ratio value R is used to compute the ratio signals `r_in` for the input resampler and `r_out` for the output one. As it will be presented in Chapter 4, the input resampler maps R to its ratio signal `r_in` in the form of the value $T\_out\_n = 1/R$. For proper operation of the sandwich, the output resampler needs the ratio value to be the inverse of the ratio of the input resampler; the `r_out` signal needs hence to adopt the exact inverse value of the `r_in` signal, this is R. This ensures that the sampling rate present in the data-path before the input resampler, $f_{s\_dcpl} = A$ sample/s, is again perfectly reconstructed after the output resampler, $f_{s\_dcpl} = A$ sample/s. Recall that we intend to use this Architecture in a feedback loop. The input and output resampling ratio signals need therefore to be exactly inverse for proper operation.

This is in practice impossible for any real-valued resampling ratio R after quantization. Any discrete representation, as for instance fixed-point arithmetic, offers support for only a set of discrete values. Other real values lying between two of the set of discrete values are truncated to either its upper or lower closest neighbour. It is hence impossible to represent a real varying value without error. At some point, as the value varies with infinite precision, the quantized signal will adapt a value which is not exactly in the discrete set.

Our ratios vary continuously with time and are therefore affected by this error; the signals `r_in` and `r_out` that contain these discrete representations will not always be exactly inverse. They will not always feed the perfect inverse ratios valued $1/R$ and R to the resampler. The engineering solution to deal with this problem is to increase the number of bits in the digital word representing the real values. The truncation error between the discrete representation and the real value can become negligible when, depending on the application, a sufficient number of bits is used. This is effective for instance for the processed signal in the data-path, where the precision is in any case limited by the resolution of the ADC and DAC. The dimensioning of the digital representation of the data-path is hence based on that. This solution is, however, not feasible for us in the control signals hosting the ratio values. We need perfect inverse ratios and the error, even when very small using a huge number of bits, is not negligible for us.

Furthermore, the problem becomes more relevant in our Architecture; we need to compute the inverse of R to obtain $1/R$. The inversion is performed in a digital system with finite precision and is again only accurate within an error range. Such operation adds and/or increases the error between the resulting pair of discrete ratios. First, we quantize R when mapped to the output resampler signal `r_out`, this introduces a first truncation error. This quantized value is subsequently inverted in the system to compute $1/R$ for the input resampler signal `r_in`. Note that the inversion is based on a value that already contains an error. The resulting inversion hence accumulates the initial truncation error plus the error resulting from the inversion.

An example of the resulting situation is depicted in Fig. 3.16 where two ratios are swept with time. It shows both the contribution of the truncation error and the contribution of the computation of the inverse ratio in a discrete system. The figure depicts a variable representing a ratio R and its inverse $1/R$; the left

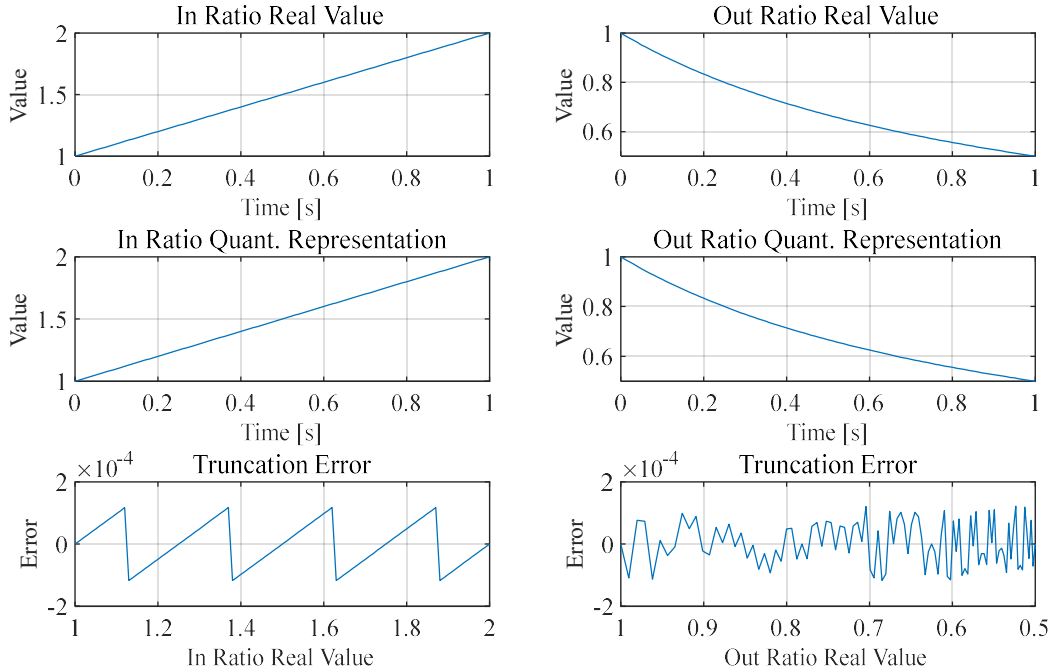


Fig. 3.16. Simulation depicting the truncation error for the up-sampling (left) and down-sampling (right) ratio signals.

column contains the plots of the *R*, and the right column contains the plots of the inverse ratio. The first row depicts the real values of the ratios. The second row depicts the quantized representations of the real values. Finally, the third row depicts the error, i.e., the difference between the quantized and real representation. The inversion is computed in fixed-point arithmetic represented with sixteen bits: A sign bit, three integer bits and twelve fractional bits. The input resampling ratio is swept between an initial value of 1 at the beginning of the ramp, and an end value of 2. The sweep time is 1 s. The inverse output value sweeps from 1 at the beginning reaching a value of 0.5 at the end of the simulation. The truncation error of the input ratio follows a periodic pattern resulting from the truncation to the nearest neighbour, adopting a magnitude in the worst case of half a Least Significative Bit (LSB), $1.22 \cdot 10^{-4}$. The output ratio in the example incorporates to this error the inversion error, making the total error to adopt a complex pattern within the same range.

This results in practice in the impossibility to have two exactly inverse ratio values in the discrete system. That is only achieved for a very limited set of simple cases, but never if the real-valued ratio varies. For instance, the inverse of 1 is also 1. In that case, both same values can univocally be represented in fixed-point arithmetic.

In our case, the vast majority of pair values will not be exact inverse ratios, and thus the product of the pair will not result in the unit value. Fig. 3.17 depicts the product of the ratios of Fig. 3.16 during the sweep; the left column depicts the real values, the right one the discrete representation. The first row contains the input ratio, the second row the output ratio, and the third row the product of both input and output ratios. The product of the real values gives as result 1 during the whole ramp. There is no error as this product has infinite precision. The product of the discrete representations oscillates around 1 reaching an error of magnitude of $2.5 \cdot 10^{-4}$, above one LSB, due to the limited precision and the inversion.

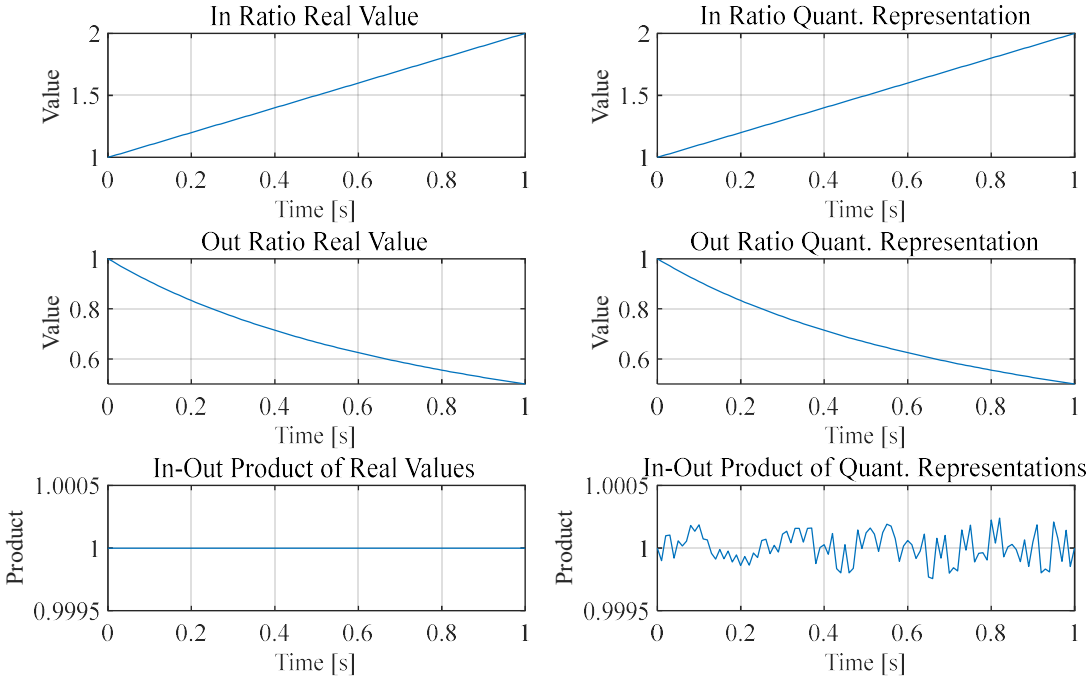


Fig. 3.17. Simulation depicting the truncation and inversion errors; the ratio product results in a value different from 1.

The error in the product between two truncated inverse input and output ratios cannot hence be solved by increasing the number of bits in the discrete word. This result for our BSP Architecture in a discrepancy between the ratio values arriving at the resamplers. The discrepancy generates a difference between the expected sampling rate and the real sampling rate achieved in the output port of the output resampler of the sandwich. The difference in sampling rates hence changes the volume of valid samples in the decoupled data-path, that can lead to a desynchronization between resamplers and fabrics. The solution to the problem is introduced in section 3.3.7 where we present the *MERCEDES* interfaces and the *JOAQUINA* loop. The *MERCEDES Couple* interface is first presented as an ideal interface regardless of this issue and then upgraded to solve the problem.

3.3.7. *MERCEDES* Interfaces

The data-path uses interface entities in the input and output ports of the *FRANCISCO* fabric to communicate with the FPGA fabric. Two entities have been developed for that, the *MERCEDES Decouple* and *MERCEDES Couple* interfaces introduced in Chapter 2. The two interfaces are depicted in Fig. 3.9, in the high-level sketch of the FPGA implementation. From a functional point of view, at application level, they are transparent as the data-path sampling rate remains the same at those points. Note that they are not depicted in Fig. 3.1, where we show the functional view of the Architecture. In these interfaces adaptation is therefore only performed at logic and physical level in the signals of the data-path; they control the `valid` signal and synchronize the different clock domains. The interfaces ensure the activation of the `valid` signal in the correct processing slot at the input and the proper reconstruction of the coupled data-path at the output. The first one, *MERCEDES Decouple,* is used at the input of the *FRANCISCO* fabric to adapt the coupled data-path to a decoupled one. The *MERCEDES Couple* interface performs the

complementary operation, it is used at the output of the *FRANCISCO* fabric to adapt the decoupled data-path to a coupled one. A detailed description is presented in the following subsections.

3.3.7.1. MERCEDES Decouple interface

The functional representation of the *MERCEDES Decouple* interface of Fig. 3.9 is depicted in Fig. 3.18. The input port of the interface receives the signals of the coupled data-path; these signals are composed of the data-bus signal `d_cpl` and the clock signal `clk_cpl`. The output port receives the signal `clk_dcpl` (that clocks also the output decoupled data-path) and provides the data-bus signal `d_dcpl` and the qualification signal `valid` for the processing slots.

In the input port, the processing clock and the data sampling clock have identical frequencies, `clk_cpl` at $f_{p\_cpl} = A$ Hz and `d_cpl` sampled at $f_{s\_cpl} = A$ sample/s. This port uses hence a coupled data-path and it makes no sense to talk about activation rate (expressed in Fig. 3.18 with $ar = -$) as there is no `valid` signal. The transitions in the data bus are synchronized with the rising edges of the clock.

In the output port, the frequency f_{p_dcpl} of the processing clock `clk_dcpl` is an integer multiple M of the frequency f_{p_cpl} of the input processing clock `clk_cpl`; $f_{p\_dcpl} = M \cdot f_{p\_cpl} = M \cdot A$ Hz. The transitions in the data-bus signal, are also synchronous with this clock signal. The average sampling rate of the output data-path remains the same $f_{s\_dcpl} = A$ sample/s; this is achieved by qualifying only the first processing slot out of M with the `valid` signal active.

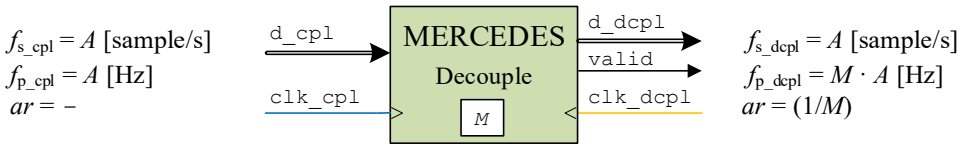


Fig. 3.18. Functional representation of the *MERCEDES Decouple* interface. The input port interfaces a coupled data-path. The output port interfaces the decoupled data-path.

A possible implementation of the *MERCEDES Decouple* interface is depicted in Fig. 3.19. In the implementation, the entity receives both the input and output clocks provided by the FPGA clock manager. The clock domain in the coupled data-path port (input) requires synchronization to the clock domain in the decoupled data-path port (output). This is done with a series of registers that are clocked by the output clock `clk_dcpl`. The output port samples the synchronized data bus at the rate of the output port clock. To keep the sampling rate constant in both ports, only one out of M clock cycles (the relation between clocks) is qualified as valid at the output. For this, an edge detector monitors the input port clock `clk_cpl`. When a rising edge is detected, a qualification signal enables the register driving the data-bus signal `d_dcpl` in the output port during one clock cycle. This signal is also provided to the output port as the `valid` flag signal. This architecture supports any integer relation between input and output clocks, as long as the frequency of the `clk_dcpl` clock is at least the double of the frequency of the `clk_cpl` clock.

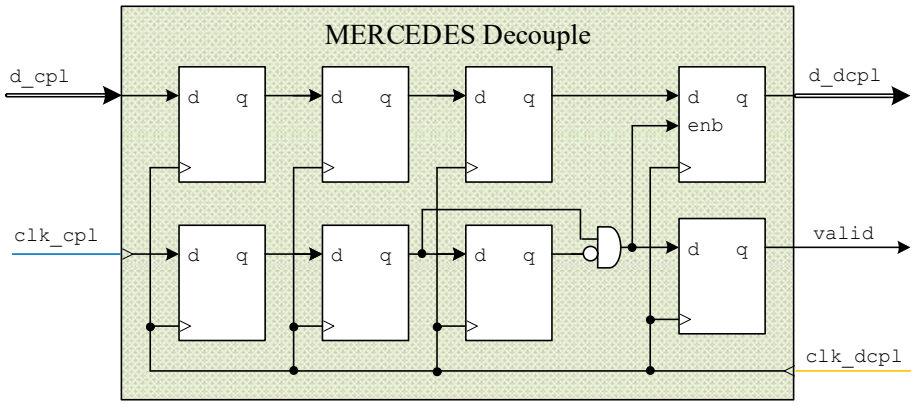


Fig. 3.19. Schematic representation of a possible *MERCEDES Decouple* interface implementation.

Fig. 3.20 depicts a chronogram with the relation between signals at the input and output port of the *MERCEDES Decouple* interface. The relation between clocks is $M = 3$. The latency through the interfaces is omitted for simplicity. The chronogram depicts the data-path processing slots ps_X in `d_cpl` and `d_dcpl`, and the value d_X (− when void slot) of the data sample in the bus signals. The qualified processing slots in the output are signalled by the `valid` signal.

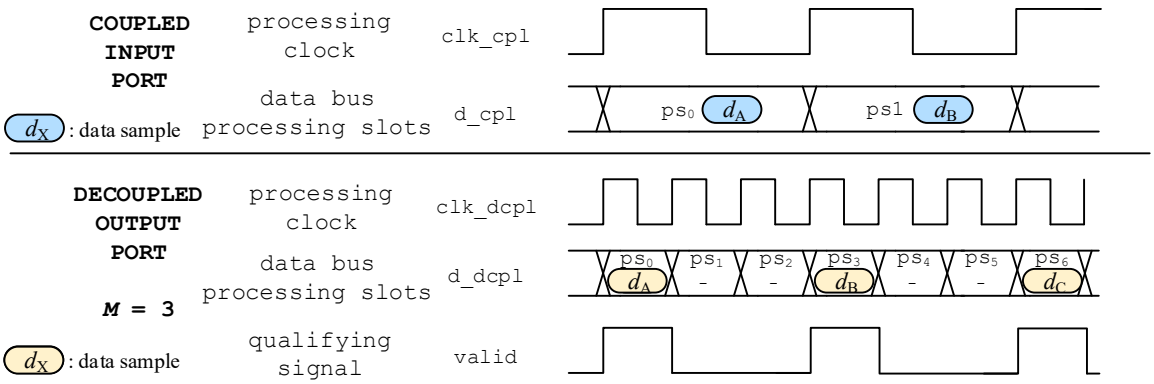


Fig. 3.20. Chronogram with the signals at the input and output ports of the *MERCEDES Decouple* interface.

3.3.7.2. MERCEDES Couple interface

The following section presents implementation details for the *MERCEDES Couple* interface. Two solutions are presented; first, an initial idea where the inversion and truncation errors were not considered. This solution serves to understand the interface from a functional point of view. A feasible solution was found after analysis of the problems noticed in the verification. That solution is presented in the second point of this section and shows the final implementation of the interface solving the truncation problems.

3.3.7.2.1. Initial implementation

The functional representation of the initial *MERCEDES Couple* interface is depicted in Fig. 3.21. The input port of the interface receives the signals of the decoupled data-path; these signals are composed of the data-bus signal `d_dcpl`, and the qualification flag signal `valid`. The clock signal `clk_dcpl` is also provided to this input port. The output port receives the clock signal `clk_cpl` (that clocks also the output coupled data-path) and provides the data-bus signal `d_cpl`. The frequency f_{p_dcpl} of the processing

clock `clk_dcpl` in the input port is a multiple M of the data-path average sampling rate $f_{s\_dcpl} = A$ sample/s, $f_{p\_dcpl} = M \cdot A$ Hz. The activation rate of the `valid` signal is hence $(1/M)$.

In the output port, the frequency $f_{p\_cpl} = A$ Hz of the processing clock `clk_cpl` has the same value as the sampling rate $f_{s\_cpl} = A$ sample/s of the data-bus signal `d_cpl`. Again, in this coupled port there is no `valid` signal, it makes hence no sense to talk about activation rate (expressed in Fig. 3.21 as $ar = -$).

The relation between the frequencies of the input clock `clk_dcpl` and the output clock `clk_cpl` in the interface is hence M, $f_{p\_cpl} = f_{p\_dcpl} / M = A$ Hz. The sampling rate in the data-path remains the same at both the input and output ports $f_{s\_dcpl} = f_{s\_cpl} = A$ sample/s.

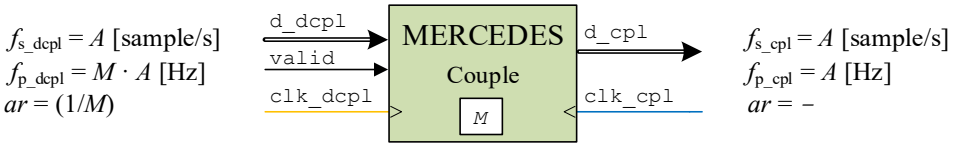


Fig. 3.21. Functional representation of the *MERCEDES Couple* interface. The input port interfaces a decoupled data-path. The output port interfaces the coupled data-path.

The initial implementation of the *MERCEDES Couple* interface is depicted in Fig. 3.22. The synchronization between decoupled (input) and coupled (output) clock domains is done with a First-In-First-Out (FIFO) memory. The interface stores the output samples of the sandwich in the FIFO. This is a technique used for clock domain crossing and synchronization. This makes it possible to recover a uniform input sample pattern with activation ratio $ar = 1/M$, with M the relation between the input and output frequencies of the processing clocks, at the output of the memory. The idea is based on the reading of the FIFO at the same effective rate at which it is written, in that case, the filling level remains constant. If we analyse this again from a functional point of view, this results in the sampling rate passing unaltered through the FIFO and the interface.

Looking at the inside, the entity receives both the input `clk_dcpl`, and output `clk_cpl` clocks provided by the device clock manager, either ASIC or FPGA. The clock of the decoupled data-path is fed to the write port of the memory. The input signal `valid` in the decoupled port is used to enable the write port of the memory, `WE`. Only the samples marked as valid in the input data-bus signal `d_dcpl` (one out of M on average) are written into the FIFO. The output port of the entity is controlled by glue logic that

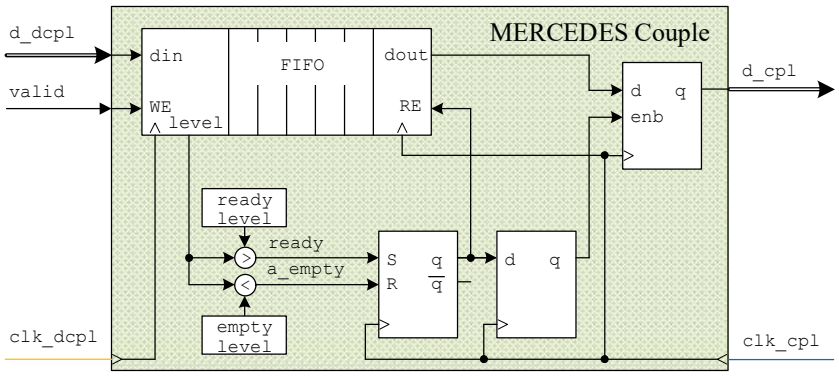


Fig. 3.22. Schematic representation of a possible *MERCEDES Couple* interface implementation.

monitors the filling level of the FIFO. FIFO memories provide embedded functionalities and signals for this; however, we explicitly describe here a feasible implementation of such a logic based on a signal `level` provided by the FIFO that shows the number of samples available within it.

We base on that signal our final implementation (section 3.3.7.2.2) that solves the truncation and ratio inversion problems. The `level` port of the FIFO is compared against two reference levels. A `ready` signal is set in the comparison logic when the number of samples in the memory is above an operation threshold, the `ready level` threshold. This threshold dictates when the FIFO is populated with enough samples for the safe operation of the interface. A second threshold signal `a_empty` is set when the level goes below a safety threshold, the `empty level` threshold. This second level is lower than the `ready level` threshold, and halts the read port of the FIFO when it is "*almost empty*".

The `ready` signal drives the set port of a Reset-Set (RS) register. The `a_empty` signal drives the reset port of the RS register. The output of this register is hence asserted when the number of samples in the FIFO achieves the operational level. The register is de-asserted when the level goes below the safety level. The output of the RS register drives the read enable port, `RE`, of the FIFO to control the synchronous extraction of data. This data is latched in a register driving the data bus of the output port (coupled port) of the entity. This register is also controlled by the RS register; the enable control port is driven by a synchronized version of the RS output signal.

The resulting interface architecture supports any relation between input and output clocks. For proper operation, it requires an average sampling rate in the decoupled input port, the same as the sampling rate of the coupled output port. The operation and safety levels of the FIFO allow for variation of this input rate, however, in the long-term the average needs to be stable. Fig. 3.23 depicts a chronogram example of the relation between signals at the input and output port of the *MERCEDES Couple* interface. The relation between clocks is $M = 3$. The chronogram depicts the data-path processing slots ps_X in `d_dcpl` and `d_cpl`, and the value d_X (− when void slot) of the data sample in the bus signals. The qualified processing slots in the input are signalled by the `valid` signal. The latency through the interface has been omitted for simplicity.

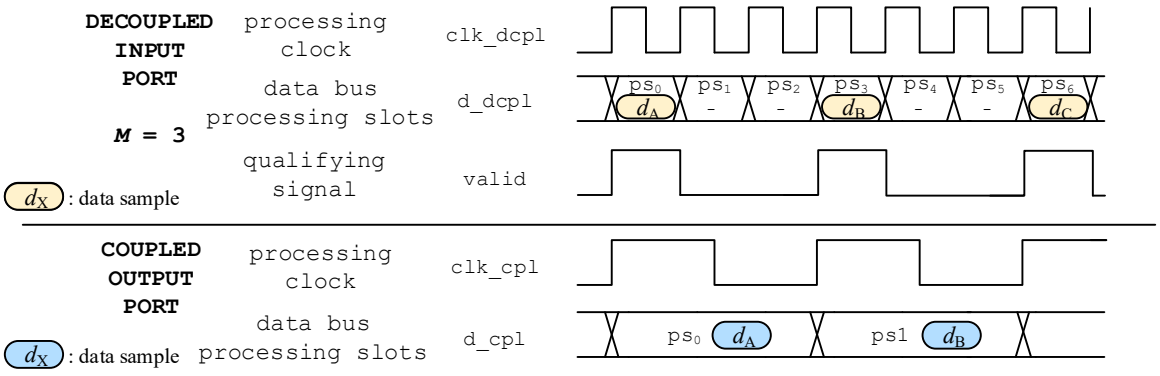


Fig. 3.23. Chronogram with the signals at the input and output ports of the *MERCEDES Couple* interface.

3.3.7.2.2. Final implementation

The *MERCEDES Couple* interface has been modified to cope with the inversion and truncation problem presented in section 3.3.6. These problems result in a difference between the expected sampling rate in the input and output ports of the interface.

Analysing in detail Fig. 3.22, we can see that a clock generated by the device clock manager is used to read the output port of the FIFO memory. This clock `clk_cpl` has a "perfect" fixed frequency $f_{p\_cpl} = A$ Hz. This frequency has the same value as the sampling rate $f_{s\_cpl} = A$ sample/s of the coupled data-path of the interface. This value is also the expected sampling rate $f_{s\_dcpl} = A$ sample/s of the decoupled data-path port of the interface, the same at the output port of the output resampler. This is the case when the values R and $(1 / R)$ of the ratio signals `r_out` and `r_in` of the resampling sandwich are perfectly inverse, and thus $f_{s\_cpl} = f_{s\_dcpl} = A$ sample/s. In this case, the level of the FIFO remains constant; the writing rate is the same as the reading rate.

As we saw in section 3.3.6, the value of the ratio signal in the input resampler can have some error. This error makes the sampling rate in the output port of that resampler not exactly $f_{s\_dcpl} = R \cdot A$ sample/s but $f_{s\_dcpl} = \check{R} \cdot A$ sample/s, with $\check{R} = (1 + \epsilon_1) \cdot R$ the actual value of the truncated resampling ratio and ϵ_1 the magnitude of the truncation error. If the value of the ratio signal in the output resampler compensates that error (its value $\check{R}$ becomes perfect inverses of the input, $\check{R} = (1 / \check{R})$), the error would cancel in the sandwich. This would make the rate in the output port of the output resampler be again $f_{s\_dcpl} = \check{R} \cdot \check{R} \cdot A = A$ sample/s.

But this is not the case, and due to the error between ratios, the rate in the output port of the output resampler becomes $f_{s\_dcpl} = (1 + \epsilon_2) \cdot A$ sample/s, with $\check{R} \cdot \check{R} = (1 + \epsilon_2)$. The variable ϵ_2 is the resulting magnitude of the error from the ratio truncation in the input resampler and the inversion; it makes the ratios diverge. In Fig. 3.17 ϵ_2 has a worst-case magnitude of $2.5 \cdot 10^{-4}$ for fixed-point arithmetic represented with sixteen bits: A sign bit, three integer bits and twelve fractional bits. This is in the order of one LSB. When this is the case, the frequency of the clock used to read the memory and the data rate at the read port, is different from the rate at which the data arrives at the memory, $f_{s\_cpl} = A$ sample/s vs $f_{s\_dcpl} = (1 + \epsilon_2) \cdot A$ sample/s respectively. This results in fluctuations in the FIFO level, that can lead to underflow or overflow in the long term.

The discrepancy between sampling rates that produces the fluctuations results from the truncation and inversion errors in the ratios. If the ratios are kept constant the fluctuation of the FIFO level is monotonic for a given pair of ratios. However, our ratios change with time, hence the variation in the level of the FIFO becomes dependent on the different pair of imperfect inverse ratios. The error accumulates thus through the FIFO level resulting in the known and studied problem of the Random Walk [97]. The computer simulations of the interface were not able to accumulate enough simulation cycles to make the sandwich

lose the synchronization between resamplers due to overflow or underflow. However, the verification in the laboratory quickly revealed the problem. Further results are presented in Chapter 5.

The solution to cope with the problem is to keep, on average, a perfect inverse ratio between resamplers. For this, we exploit the fact that the FIFO can absorb small instantaneous fluctuations deviating from the average. The *JOAQUINA* Frequency-Locked Loop is included around the *MERCEDES Couple* interface and the output resampler. It acts by providing corrections to the output resampling ratio, that keep the sampling rates equal in average at the input and output ports of the *MERCEDES Couple* interface, and thus $f_{s\_cpl} = f_{s\_dcpl} = A$ sample/s. This makes inverse ratios possible, on average, between resamplers. As the divergence and the corrections have very small values and keep the average constant, the modulation of the resulting signal due to the variations in resampling ratio is negligible.

This mechanism is based on the comparison of the FIFO level against a reference value. When the sampling rates in the interface are equal, the FIFO level does not fluctuate. When the difference in the product of the ratio signals is larger than one LSB, the level starts to diverge. The resulting error signal from the comparison of levels is used as a correction for the ratio in the output resampler.

The modified *MERCEDES Couple* interface is depicted in Fig. 3.24. The level of the FIFO memory that should be stable in operation is compared against the ready level. The resulting error signal `e_R` is scaled by a gain factor `K` and propagated out of the resampler as the `corr_R` signal. This signal is used in the output ratio *JOAQUINA* Frequency-Locked Loop. The gain `K` acts as a scaling of the level error to make the loop more reactive or damp its response. We have experimentally verified that a scaling factor, that results in a value of one LSB for the `corr_R` signal, keeps the Architecture stable and operational when the difference between levels is one memory position of the FIFO.

The high-level Architecture presented in Fig. 3.9 implements this modified *MERCEDES Couple* interface. The ratio Frequency-Locked Loop is fully depicted around the output resampler and the *MERCEDES Couple* interface. This interface outputs the correction error signal `corr_R` which is added to the output ratio `r_out`. This solution keeps the sampling rate at the output port of the output resampler, on average, at the same fixed sampling rate present before the input resampler, $f_{s\_dcpl} = A$ sample/s. Thanks

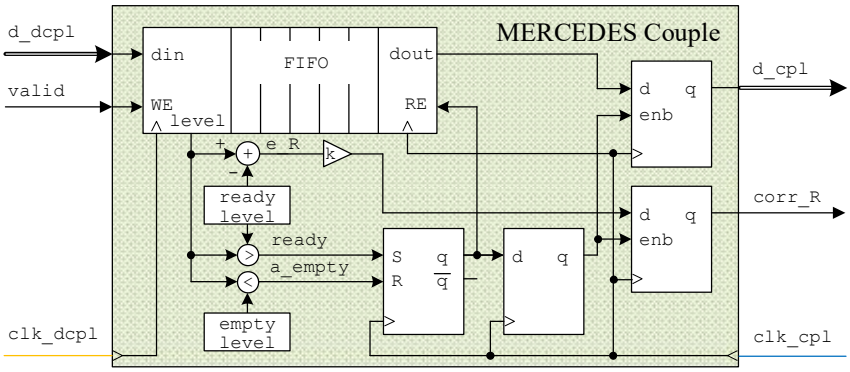


Fig. 3.24. Schematic representation of a possible *MERCEDES Couple* interface implementation with the correction signal `corr_R` used to create the *JOAQUINA* Frequency-Locked Loop to cope with the truncation error in the output resampler ratio.

to this loop, the interfaces are transparent from a functional point of view, they keep the sampling rate constant at its inputs and outputs, performing only adaptation at logical and physical level.

3.3.8. Real-time variable ratio resampler with decoupled data-path

Beam Synchronous Processing is performed within the BSP region after adaptation of the sampling rate in the data-path. We have already introduced the two resamplers, depicted in the functional and implementation sketches of the Architecture. Traditional arbitrary ratio resamplers use the ASRC approach introduced in Chapter 2; the input port uses a hardware processing clock with the same frequency as the input sampling rate, and the output port uses a second hardware clock at the same frequency as the new output sampling rate in the data-path [74]. We wanted to avoid implementation problems in our SSRC derived from this output clock (swept in our case); our resampler is implemented within the *FRANCISCO* fabric to decouple the clocking architecture and the sampling rate.

As resamplers based on a decoupled architecture are not common, we have developed a new one exploiting such a paradigm. It uses the same hardware processing clock in both input and output ports, and at the same time makes it possible to have a variable sampling rate in the data-path. This new resampler is hence the key element in the proposed Architecture that enables the change of the data-path sampling rate with a fixed frequency processing clock. Chapter 4 is dedicated entirely to the resampling architecture; it presents in detail the concept and implementation, but a brief introduction is anticipated here to provide the reader with its foundations. Fig. 3.25 depicts schematically the signals of its physical interfaces.

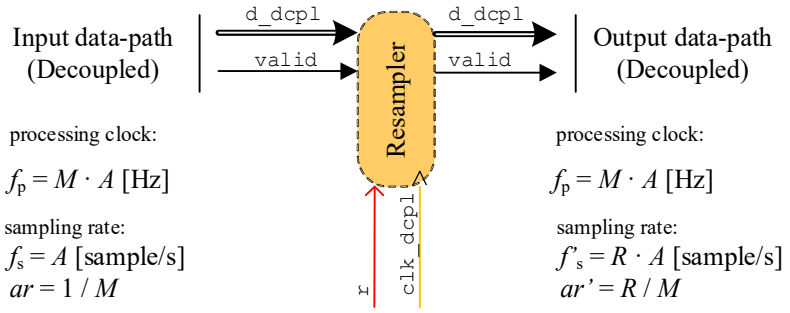


Fig. 3.25. Schematic representation of the developed resampling architecture with decoupled data-path (Chapter 4).

Both, the input and output ports interface the decoupled data-path signals; the data bus `d_bus`, and the qualification signal `valid`. In addition, the decoupled processing clock `clk_dcpl` operates the hardware and the resampling ratio R is dictated by the value of the signal `r`.

The resampler operates by modifying the number of samples in the data-path at its output. For this to happen in the case of up-sampling, it interpolates a new sequence that populates some void processing slots. In the case of down-sampling, the interpolation reduces the number of samples and some previously populated processing slots become now empty. The value of the resampling ratio R, that relates the input and output sampling frequencies, dictates the case. The `valid` signal indicates the populated processing

slots. The activation rate *ar* of this signal is hence also modified by the resampling ratio, and its relation between the input and output ports of the resampler becomes

$$ar' = ar \cdot R \qquad \text{Eq.(3.4)}$$

The resampler of Fig. 3.25 uses a processing clock `clk_dcpl` at a frequency $f_p = M \cdot A$ Hz. The sampling rate at the input is $f_s = A$ sample/s. So that only one out of M processing slots is occupied. The activation rate at the input is hence $ar = 1/M$. According to Eq.(3.4), the activation ratio becomes at the output of the resampler $ar' = R/M$. By using Eq.(3.3) we can relate the sampling rate of the data-path at the output port of a resampler with ratio R to the processing clock resulting in

$$f'_s = R \cdot f_s = R \cdot ar \cdot f_p = ar' \cdot f_p \qquad \text{Eq.(3.5)}$$

3.3.9. Resampling ratio and BSP processing relation

The proposed BSP Architecture modifies the sampling rate of the processed signal to tune its resulting discrete representation $Y(e^{j\omega})$ to the frequency where processing is defined ω_{proc} as depicted in Fig. 3.6. The tuning element is the input resampler and the parameter used in the tuning is hence its resampling ratio R. We need to know how to compute this ratio based on the frequency information of the fundamental tone of the processed signal.

Recall that the processing is performed after resampling to a rate f'_s. The signal $y[m]$ at the output of the resampler is thus the discrete sequence tuned to the processing. It approximates a signal acquired with sampling rate $f'_s = R \cdot f_s$.

The response of the processing algorithm in the frequency domain, for instance a filter, is defined at a fixed normalized frequency ω_{proc} in the resampled domain. When the ratio R is properly selected, the normalized frequency $\omega_x = 2\pi \cdot f_x$ radian/sample of the fundamental tone F_x Hz in the signal $x[n]$ matches the response of the processing defined in the resampled domain at $\omega_{proc} = 2\pi \cdot f_{proc}$ radian/sample. We need thus to compute R to map ω_x to ω_{proc} in the resampled domain.

That is equivalent to say that we want to match the absolute analog frequency of the processing F_{proc} with the frequency of the tone F_x

$$F_{proc} = F_x \qquad \text{Eq.(3.6)}$$

Looking at the problem from the input of the resampler, that absolute analog frequency becomes

$$F_{proc} = f_{proc} \cdot f'_s = f_{proc} \cdot R \cdot f_s \qquad \text{Eq.(3.7)}$$

By inserting Eq.(3.6) in Eq.(3.7) and reordering, the resampling ratio results in

$$R = F_x / (f_{proc} \cdot f_s) \qquad \text{Eq.(3.8)}$$

This resampling ratio R is the ratio needed in the input resampler of the sandwich. It needs to be updated in real-time to track the changes in the fundamental frequency of the signal.

When operating two resamplers in a "sandwich" configuration, we have seen that the output resampler requires the inverse value of the input resampling ratio to properly recover the original sampling rate of the signal prior to the BSP region. This requires from the Architecture not only the *JOAQUINA* Frequency-Locked Loop but also the output ratio to be synchronized with the samples in the data-path; the instantaneous value of the resampling ratio needs hence to match the instantaneous sampling rate of the data arriving at the resampler.

As the resampled data-path signal is being processed in the BSP region after the input resampler, the output ratio needs hence to be synchronized with the arrival of that data to the output resampler. It needs to mimic a latency equal to the latency of the processing in the BSP. This is depicted in Fig. 3.26; a register chain mimicking this BSP processing latency is placed in the ratio signal `r_out` before the output resampler. The synchronized signal is then fed to the output resampler as `r_out_s`.

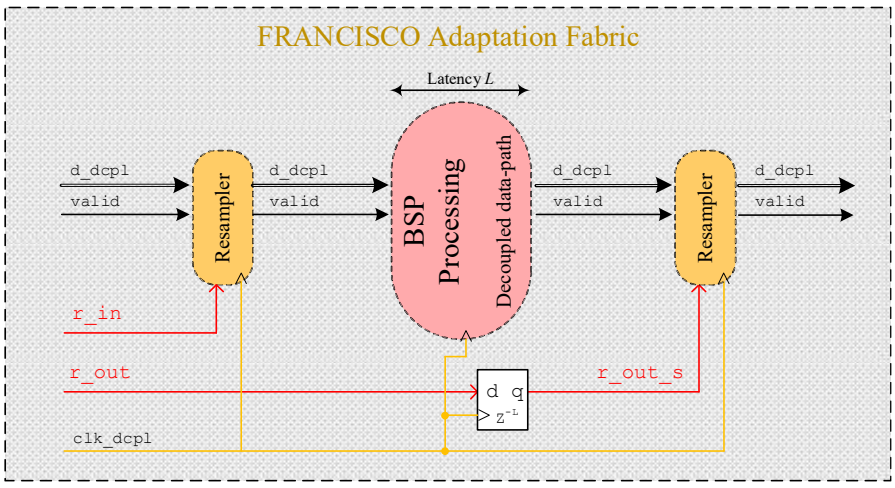


Fig. 3.26. Schematic representation of a processing segment in a decoupled data-path between two resamplers. The ratio signal fed to the output resampler `r_out_s` mimics the latency through the processing.

3.3.10. Input signal bandwidth limit

The proposed BSP Architecture performs processing of a sampled signal and modifies its sampling rate. The resampling operation (we consider an ideal resampling without quantization of signals) imposes some constraints on the maximum absolute bandwidth, BW_{SIGNAL}, that the treated signal can contain at the input of the BSP Architecture. These constraints come from two sources; first the absolute Nyquist frequency F_{NY} of the signal in the data-path that is the highest frequency that can be coded at a sampling rate f_s. The second is the input bandwidth of the resampler, $bw_{RSP} = \alpha \cdot 0.5$ with $\alpha \leq 1$, that is the range of normalized frequencies for which the resampling architecture has been optimized and can hence be resampled (further information will be presented in Chapter 4).

Fig. 3.27 depicts the sampling rates, Nyquist frequencies and normalized resampler bandwidths present in the data-path segment of Fig. 3.26 that models the BSP Architecture. We will consider as starting point the input resampler, disregarding the output one, to derive the maximum normalized bandwidth, bw_{SIGNAL} in Eq.(3.9), that the input signal can contain.

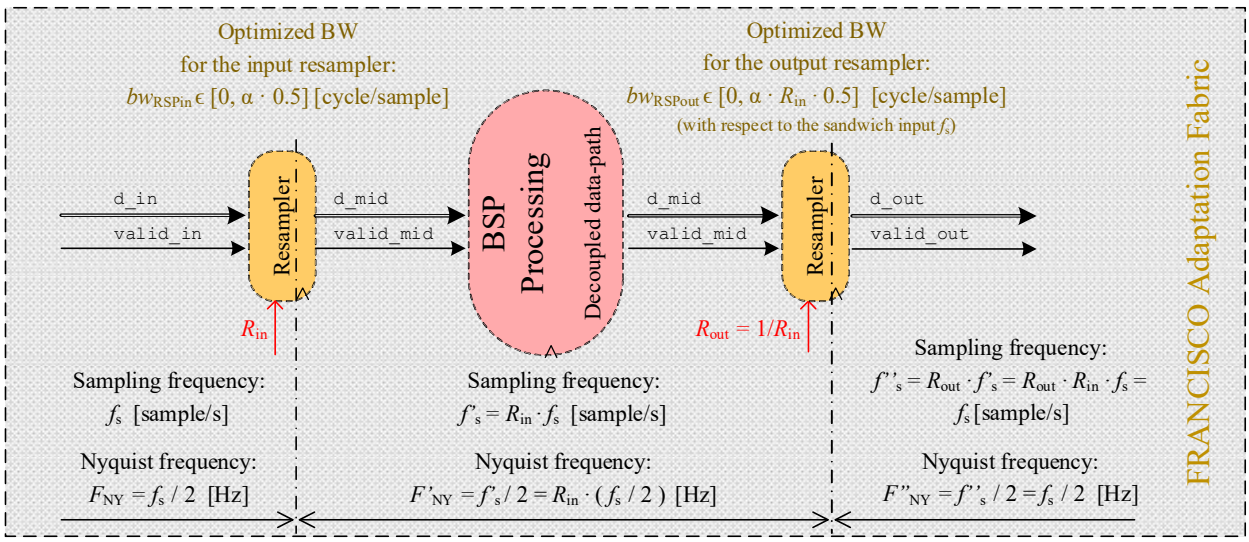


Fig. 3.27. Schematic representation of the relations between sampling frequencies, Nyquist frequencies and resampler bandwidths in the BSP Architecture.

$$bw_{SIGNAL} = BW_{SIGNAL} / f_s \qquad \text{Eq.(3.9)}$$

First, note that a resampler has two ports with different sampling rates. The Nyquist frequency for the treated signal in the resampler is hence defined as the most restrictive one of the Nyquist frequencies between the input port $F_{NY} = f_s / 2$ and the output port $F'_{NY} = f'_s / 2 = R_{in} \cdot (f_s / 2)$. We can normalize these Nyquist frequencies to the input sampling rate as $f_{NY} = F_{NY} / f_s$ and $f'_{NY} = F'_{NY} / f_s$ respectively. In the case of up-sampling, the most restrictive one is found in the input as the resampling ratio is larger than one, $R_{in} > 1$. In the case of down-sampling it is the output who imposes the Nyquist frequency of the resampler data-path, as the resampling ratio is smaller than one, $R_{in} < 1$.

These boundaries are depicted in Fig. 3.28, where we present the configuration found in the input resampler. The resampling ratios accepted in this case are R_{in} ϵ [0.5, 2], and its interpolator has been optimized to operate in a normalized bandwidth $bw_{RSPin} = \alpha \cdot 0.5$ with $\alpha = 0.6$ (we anticipate here some of the characteristics of the resampling architecture that will be presented in Chapter 4). The magenta trace

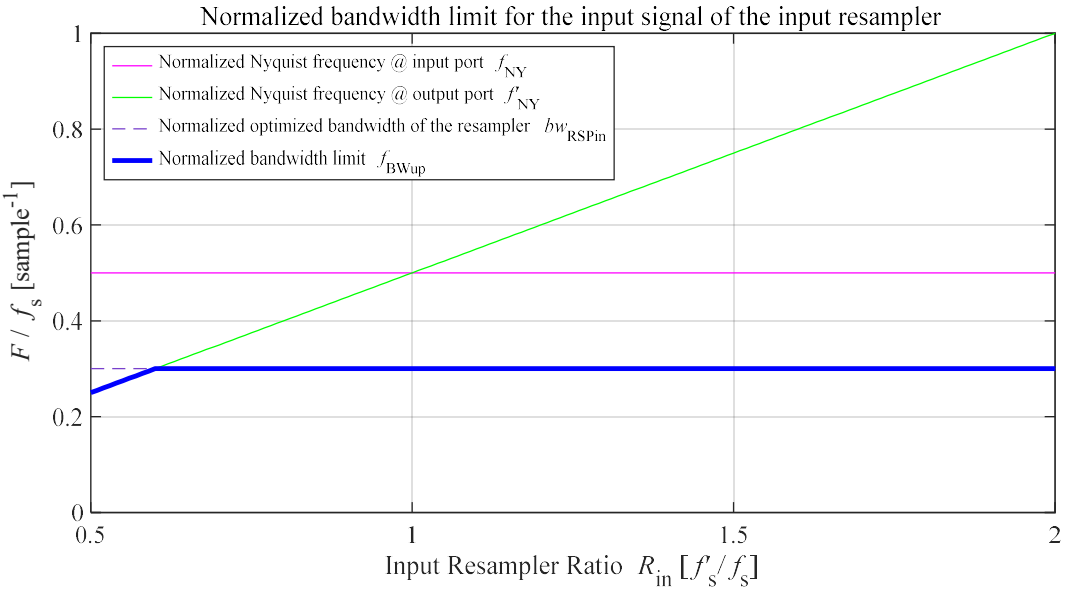


Fig. 3.28. Derivation of the bandwidth limit for the input resampled signal for a single resampler.

depicts the normalized Nyquist frequency of the input port of the resampler, in this case, it adopts a constant value $f_{NY} = 0.5$. The green trace depicts the normalized Nyquist frequency of the output port $f'_{NY}(R_{in}) = R_{in} \cdot 0.5$. This frequency varies as a function of the input resampling ratio R_{in} and spans between 0.25 for $R_{in} = 0.5$, and 1 for $R_{in} = 2$. The dashed line presents the upper frequency for the normalized optimized bandwidth of the resampler that spans up to $bw_{RSPin} = 0.3$. The blue trace depicts hence the upper limit frequency $f_{BWup} = F_{BWup} / f_s$ for any input signal in the normalized bandwidth that the resampler can treat (the absolute bandwidth BW_{SIGNAL} is defined as the range of accepted frequencies for the signal $F_{SIGNAL} \in [0, F_{BWup}]$). The reader can notice that only for resampling ratios $R_{in} \leq 0.6$ does the output data-path limit the maximum frequency of the treated signal, besides that it is the interpolator that sets the limits.

We extend the analysis now to the output resampler of Fig. 3.27. In this case, the input sampling rate f'_s is variable and governed by the configured ratio R_{in} in the input resampler, $f'_s = R_{in} \cdot f_s$. The output sampling rate f''_s however, is fixed; the ratio R_{out} is computed to be the inverse $R_{out} = 1 / R_{in}$, depicted with an orange line in Fig. 3.29. The output sampling frequency is hence $f''_s = R_{out} \cdot f'_s = R_{out} \cdot R_{in} \cdot f_s = f_s$.

We also depict in Fig. 3.29, the boundaries, as we did for the input resampler, normalized to the sampling frequency f_s in the signal at the input of the sandwich. This resampler is identical to the input one, $R_{out} \in [0.5, 2]$ and $bw_{RSPout} = \alpha \cdot (R_{in} \cdot 0.5)$ with $\alpha = 0.6$. The green trace depicts the normalized Nyquist frequency of the input port, that it is the same as the one of the output port in the input resampler $f'_{NY}(R_{in}) = 0.5 \cdot R_{in}$. Note that this is normal as both resamplers share that segment of the data-path. The cyan trace depicts the normalized Nyquist frequency of the output port $f''_{NY} = F''_{NY} / f_s = 0.5$. As expected, it adopts the same value as the input of the sandwich. The dashed line presents the upper frequency for the normalized optimized bandwidth of the resampler. In that case, as the input sampling rate varies, the upper limit does it as well resulting in $bw_{RSPout} = 0.3 \cdot R_{in}$. The red trace finally depicts the upper limit frequency $f'_{BWup} = F'_{BWup} / f_s$ in the normalized bandwidth for the signal present at the input port of the output resampler. In this case, it is the optimized bandwidth of the resampler that sets the limit for ratios below

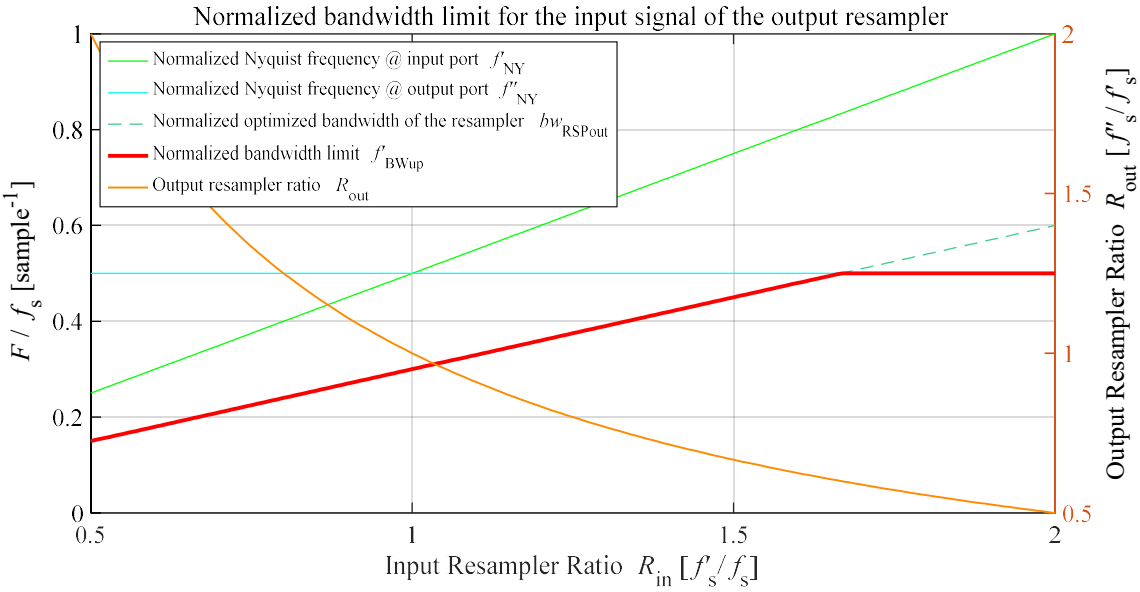


Fig. 3.29. Derivation of the bandwidth limit for the input signal of the output resampler in a sandwich configuration.

$R_{in} \leq 1.66$ (or equivalently $R_{out} \leq 0.6024$). For larger ratios, with the second resampler configured as a down-sampler, the output Nyquist frequency sets the limit.

Now we have the normalized bandwidth limits for the input signals in both resamplers, f_{BWup} and f'_{BWup}. Both of them are referred to the sampling rate f_s at the input of the sandwich; we can hence compute the maximum normalized bandwidth bw_{SIGNAL}, that the treated signal in the BSP Architecture can contain, as the set of the most restrictive limit frequencies of the two resamplers. This is depicted in Fig. 3.30, where the blue trace depicts the upper limit frequency f_{BWup} in the normalized bandwidth of the input resampler, the red trace depicts the upper limit frequency f'_{BWup} in the normalized bandwidth of the output resampler and the black trace depicts the resulting upper limit frequency $f_{BWup\text{-}sandwich}$ combining both resamplers. The normalized bandwidth bw_{SIGNAL} for any signal at the input of the BSP Architecture is hence limited to

$$bw_{SIGNAL} : X(f) \text{ with } f \in [0, f_{BWup\text{-}sandwich}] \qquad \text{Eq.(3.10)}$$

with the absolute upper limit for the real signal defined as

$$F_{BWup\text{-}sandwich} = f_s \cdot f_{BWup\text{-}sandwich} \qquad \text{Eq.(3.11)}$$

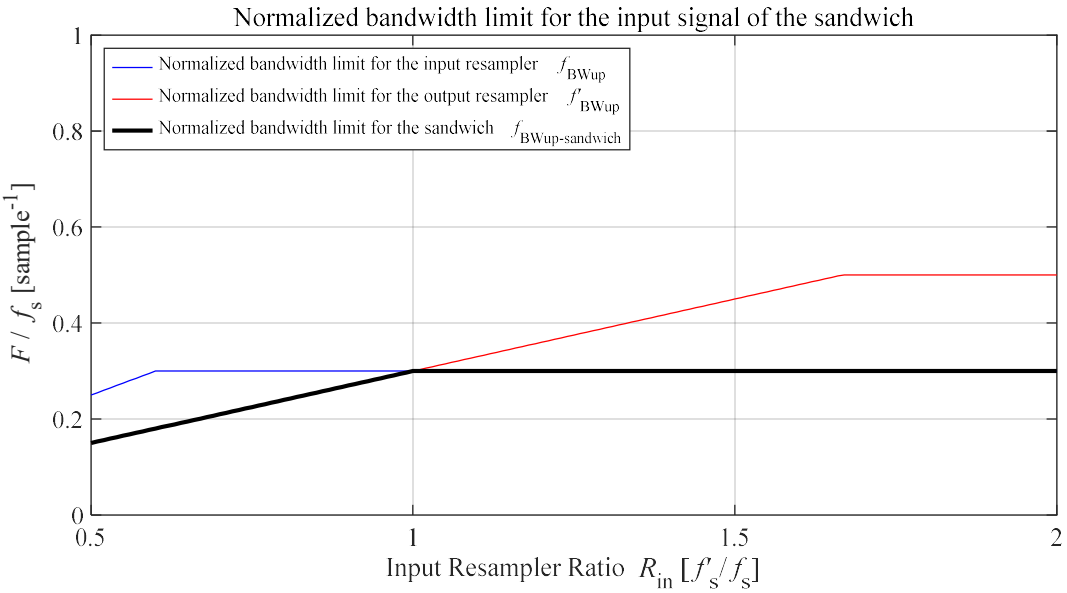


Fig. 3.30. Derivation of the bandwidth limit for the input signal of the sandwich based on the input and output resampler limits.

3.4. Conclusions

The chapter has presented a new architectural solution for Beam Synchronous Processing in a digital device, either FPGA or ASIC, with a fixed frequency system/processing clock. This new Architecture contains both BSP and BAP regions. The BAP contains processing whose functionalities are not dependent on the parameters of the signal. The BSP automatically tunes the signal to the algorithms. The BSP and BAP regions support the porting of any new or existing algorithm, requiring no specific modification of such algorithm.

The only implementation constraint is to use a decoupled data-path in the BSP region. This BSP region is built on top of the FPGA fabric in an adaptation fabric, the *FRANCISCO* fabric that adapts the data sampling rate to the signal spectral properties. This avoids the reconfiguration of BSP algorithms in real-time. The data-path interfaces the *FRANCISCO* fabric by means of dedicated points, the *MERCEDES* interfaces. These interfaces perform coupling and decoupling of the data-path and synchronization of its signals; they act at logical and physical level.

Within the *FRANCISCO* fabric, two resamplers encapsulate the BSP region. The input one performs the conversion of the fixed sampling rate, at which the data arrives at this BSP region, to a new rate proportional to the signal spectral content. At the output port, a second resampler brings the signal back to the original fixed rate. The resampling ratio values of the resamplers are reciprocal (inverse) and vary dynamically. The resamplers interface the different processing regions at functional level.

The implementation problems, truncation of values in digital signals and synchronization, have been presented and solved. The main concern has been to obtain a perfect pair of resampling ratios with inverse values between resamplers. The problem is solved with the *JOAQUINA* Frequency-Locked Loop.

Chapter 4

Arbitrary and Real-Time Variable Ratio Resampling Architecture

Abstract: *This chapter presents a new solution for Sampling Rate Conversion in which the ratio can take any value and can be modified continuously enabling Beam Synchronous Processing. The architecture is based on a Farrow-based Variable Fractional Delay filter and a timing unit element. The resampler architecture is optimized for modern FPGA devices. It decouples the processing and sampling clocks, and uses a single processing (hardware) clock whose frequency remains fixed. First, a high-level overview of the architecture is presented. Then the implementation details of the different resampler blocks are elaborated.*

4.1. Introduction

The previous chapter has presented an Architecture that makes Beam Synchronous Processing possible by resampling the treated signal. The Architecture uses two resamplers that modify the sampling rate of the signal to tune the representation of the spectral content to the processing algorithm. These resamplers are the key element of the BSP Architecture, but in Chapter 3 only the functional model has been introduced. This chapter presents the internal architectural and implementation details.

4.2. Proposed Synchronous Sampling Rate Conversion architecture

Any digital resampling architecture contains two functional elements; an interpolator and a timing unit. These elements are common in up-sampling and down-sampling architectures. The resampling operation first requires the determination of the time instants (or a derived parameter of these) in which the output

sequence needs to be estimated. Then a second process interpolates the output value at those instants using the available samples in the input sequence [78], [98]. The first process requires a timing reference and the resampling ratio *R*, Eq.(2.2). The second uses the input samples $x[n]$ and the output result of the timing process, m. Sampling rate conversion is therefore a twofold process.

The Thesis proposes a resampling architecture with these two functional units fulfilling the requirements presented to the Architecture in Chapters 1, 2 and 3. Fig. 4.1 depicts our resampler model, and illustrates the relations between the two functional elements implementing the interpolation process and the timing unit. The blue block performs the mathematical operation of interpolation, and the red block computes the sampling instant dictated by the resampling ratio *R* and the reference n. The figure presents the flow of signals and relations between the main blocks. Our resampling architecture is generic accepting up-sampling or down-sampling ratios. It addresses SSRC; for that it uses the input sequence as timing reference and computes the output time instants based on the resampling ratio. The resampling ratio, that can adopt any arbitrary and variable value, is made available to the resampler via an external signal, *R*.

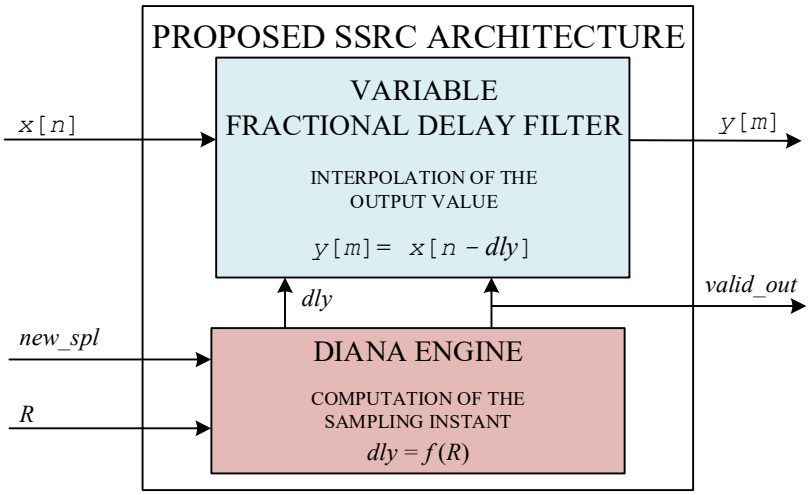


Fig. 4.1. Proposed synchronous sampling rate conversion architecture, the resampler.

The remaining of the section presents the mathematical process behind the interpolating block from a functional point of view, and regardless of any implementation detail. It shows that the two families of resamplers, classified based on the resampling ratio value, can use the same interpolating unit. It then elaborates a second classification according to the timing reference block introduced in Chapter 2.

4.2.1. Interpolation between available samples

The interpolator block performs interpolation; this is a mathematical process used for estimation of the values of an unknown function at points where no information is available. The unknown function is approximated by a second function, for instance a polynomial, that fits a discrete set of known values. Then, when an unknown value is required it suffices to evaluate the polynomial at the required point [99]. Sampling rate conversion is an analogue process in which the input samples are the available information, the input signal is the approximated function, and the output sampling instants are the points where no information is available.

It is common to distinguish between sampling rate conversion structures for up-sampling and down-sampling [72], [100]. We can classify the resamplers in two families according to the number of output samples computed by the interpolator, with respect to the number of samples in the input sequence. That relation is the resampling ratio R, Eq.(2.2). Fig. 4.2(a) depicts an example in case of down-sampling, sampling rate conversion with a ratio R smaller than one. In that case the number of samples at the output is smaller. This makes the spacing between output samples larger; the output sampling period is larger. In case of up-sampling, depicted in Fig. 4.2(b), there are more samples at the output and the spacing between them is smaller than at the input; the output sampling period is smaller.

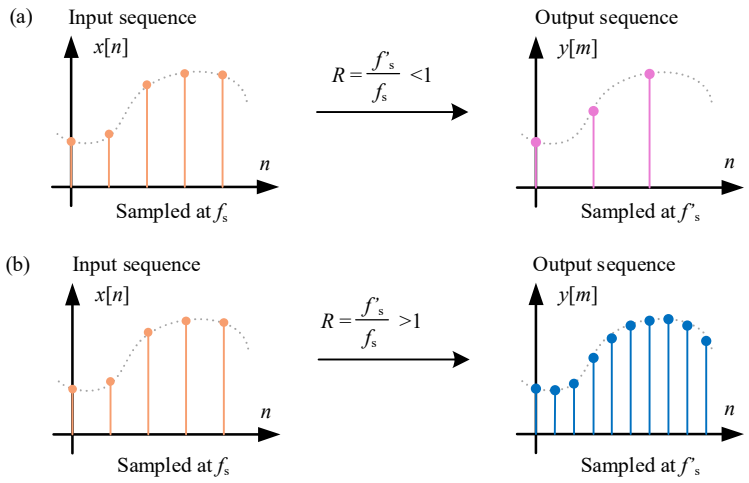


Fig. 4.2. (a) Sampling rate conversion for $R < 1$. (b) Sampling rate conversion for $R > 1$.

In any case, regardless of the resampling ratio, each output sample is computed based only on a given set of neighbour input samples. This dependence abstracts the interpolator from the ratio; the two families of resamplers operate only with input samples and a desired time instant. From a mathematical point of view, there is thus no difference between up-sampling and down-sampling. This implies that the same interpolating unit can be used for both families. We illustrate this in Fig. 4.3, where only a single output sample (in green in the figure) is depicted. The output sample is obfuscated from other samples in the output sequence. It can result from either up-sampling or down-sampling. In any case, the only information the interpolator uses are the five available neighbour samples, in beige.

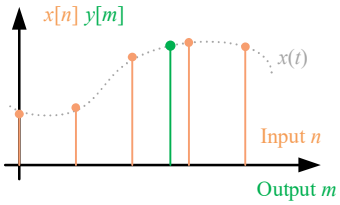


Fig. 4.3. Interpolation between available samples regardless of the resampling ratio R.

The interpolator first fits the available samples to the subjacent input signal (in grey), and then evaluates the resulting function at the desired time instant. It does not observe the resampling ratio, but the desired point where the output is required. Multiple interpolation algorithms exist based on the shape of the

fitting function or polynomial [72], [99]; linear, cubic, splines…but research in the topic is out of the scope of the Thesis.

4.2.2. Timing reference and synchronization

The computation of the sampling instant is based on the resampling ratio *R* and a timing reference. Fig. 4.1 has depicted a generic timing unit; it does not provide information on how the timing reference is made available to the resampler. The ratio *R* is not constrained within any set of values. We have reviewed in the previous section the classification of resamplers based on the resampling ratio *R*. For the timing unit it is more relevant to look instead at the timing reference. Two main families of sampling rate conversion architectures can be identified: Asynchronous SRC (ASRC) and Synchronous SRC (SSRC) [74]. They were introduced in section 2.3.2 and two of these resamplers were depicted in Fig. 2.4. In the first family the timing references for the input and output sequences are different. In that case, the resampling ratio *R* is inferred from the relation between the references. It is called asynchronous as it is in practice impossible to synchronize two sequences with different references. SSRC addresses a different philosophy; a general reference (normally the input sequence or system clock) is common for all the resampler elements. The resampling ratio is provided externally and the output sampling instants are computed based on the two parameters: The ratio and the common reference. In this case it is possible to synchronize the input and output sequences. The timing unit cannot thus be generic and support the two families. It needs to be customized to one of the two approaches. Our proposed architecture uses this second philosophy, SSRC.

4.2.3. Proposed interpolator and timing units

The interpolation block is based on a discrete-time Variable Fractional Delay (VFD) filter, depicted in Fig. 4.1. These filters are Fractional Delay (FD) filters accepting variable delays and are used in discrete interpolation of bandlimited signals [78]. They generate an output sequence $y[m]$ that approximates the value of a real signal $x(t)$. The output sampling instants are arbitrary points lying between samples of the discrete input sequence $x[n]$ that represents $x(t)$ at the input sampling rate. Each output sample of $y[m]$ is estimated by "filtering" the neighbour input samples with a given amount of time, the delay *dly*. The filter synthesizes a phase shift of the input signal with an all-pass filter; it transforms the interpolation operation into filtering. The VFD receives three parameters: The input sequence of samples $x[n]$, the delay value *dly*, and a qualifying signal *valid_out*. The input sequence contains the discrete samples of the real valued signal $x(t)$ uniformly sampled at the input sampling rate, f_s. The output sequence $y[m]$ is composed of the filtered samples. The delay value is different for each pair reference-output sample, and it specifies the amount of time that the input reference sample need to be shifted. The qualifying signal triggers the VFD to estimate the output sample when the delay value can be handled by the filter.

The input and output sampling instants are managed by the timing block; it implements an algorithmic engine. This block uses the input sequence as timing reference and receives the resampling ratio *R* as parameter. The engine, depicted also in Fig. 4.1, hosts the *DIANA* (DIstAnce iN time Algorithm)

algorithm that computes the different delay values *dly* for each desired output sample. This delay is computed based on the sampling instant of the sample used as reference, and the desired output. It monitors the valid incoming input samples, flagged by the signal *new_spl*, to track the current time instant of the input sample used as reference. It concurrently tracks the required time instant for the output sample, that is computed based on the resampling ratio. The difference between these two times dictates the delay to be fed to the VFD filter. When that amount of delay can be handled by the VFD, the triggering signal *valid_out* is raised. That signal is also made available at the output of the resampler as a qualifying signal.

4.3. Application of the architecture to arbitrary SRC

This section presents the functional details of the architecture presented in the previous point. It details how the functional units have been tailored for use in the BSP application of the Thesis. These customizations result from the special need of an arbitrary and real-time variable ratio. It elaborates the algorithmic engine and the principles of the VFD.

4.3.1. The *DIANA* engine

4.3.1.1. Delay computation procedure

The architecture is based on a VFD whose control parameter is the delay value. This delay is computed by the timing engine with the *DIANA* algorithm. For this, the timing unit tracks the input and output sampling instants, that are times measured in seconds. The delay results from subtracting the sampling instant $t_{x[n]}$ of the input reference sample to the sampling instant $t_{y[m]}$ of the desired output. The sampling instant of the reference input can be computed as

$$t_{x[n]} = n \cdot T_s \qquad \text{Eq.(4.1)}$$

T_s is the input sampling period and n the index of the reference input sample in the input sequence. The sampling instant of the desired output can be computed as

$$t_{y[m]} = m \cdot T'_s \qquad \text{Eq.(4.2)}$$

T'_s is the output sampling period and m the index of the desired output sample in the output sequence. The time distance τ is thus a physical time difference in seconds and can be computed as

$$\tau = t_{y[m]} - t_{x[n]} \qquad \text{Eq.(4.3)}$$

In Fig. 4.4(a) we depict several delay cases for different output samples (green) based on the same reference (beige): The diamond marks the input reference sample $x[n]$. The circles represent desired output samples $y[m]$. The reference sample in the input sequence of the figure is $x[3]$, while the desired outputs are $y[3]$ and $y[4]$.

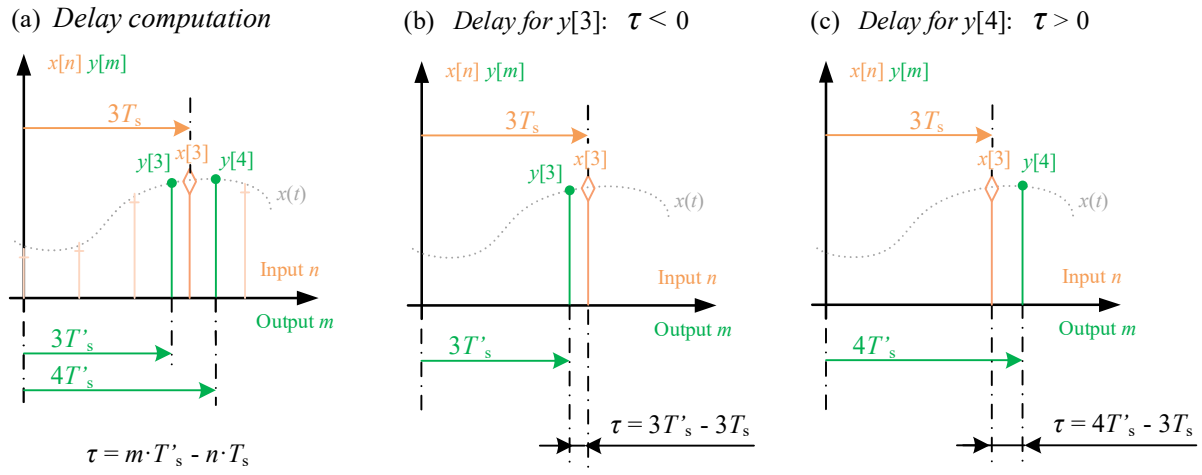


Fig. 4.4. (a) Absolute time position for input sample $x[3]$, and output samples $y[3]$ and $y[4]$. (b) Delay computation for output $y[3]$. (c) Delay computation for output $y[4]$.

Fig. 4.4(b) depicts the delay computation for the output sample $y[3]$, considering $x[3]$ as the input sample. The delay value is the time difference between the sampling instant of the two discrete samples

$$\tau = t_{y[3]} - t_{x[3]} < 0 \qquad \text{Eq.(4.4)}$$

In this case, looking "backwards", the delay value is a negative number.

Fig. 4.4(c) depicts the delay computation for the output sample $y[4]$. In this case, the delay value is positive, looking "forward"

$$\tau = t_{y[4]} - t_{x[3]} > 0 \qquad \text{Eq.(4.5)}$$

4.3.1.2. *DIANA algorithm*

There are applications of FD filters that require the same fixed delay amount for all the output samples. This is the case for instance of echo cancellation, phase array antennas or speech synthesis [78]. In sampling rate conversion this is not the case, and each sample requires a different delay value, thus VFDs are used instead.

When the resampling ratio is fixed, the different delay values follow a periodic pattern within a discrete set of values [72]. If the ratio is known beforehand, the delay values can be computed and the interpolating coefficients of the FD filter stored in a table. It is also possible to customized the FD filter architecture based on this set of values with polyphase architectures [101], [102]. In any case, all the optimizations exploit the fact that the delay values are constrained within a set of discrete values.

Our resampler deals with an arbitrary and variable resampling ratio R. This characteristic translates into different sets of delay values for different ratios. Furthermore, our resampling ratio varies in real-time, this makes that the different sets of associated discrete delays merge: The delay can adopt any value within a certain pair of thresholds. This greatly influences the architecture of the timing unit and the *DIANA* algorithm. To deal with that, the unit computes the delay associated with each input sample based on the

current value of the resampling ratio signal. This delay τ was presented in the previous point, Eq.(4.3). It was presented as a difference between the output samples and the time instant of the input reference . The *DIANA* algorithm needs thus first to compute the output time instant based on the resampling ratio. Then, when this value is known, the algorithm obtains the time difference and finally translates it to a delay value.

Some further considerations are needed for the *DIANA* algorithm. The word *fractional* in a VFD comes from the fact that the amount of delay used by the filter "*to shift*" the output is a fraction of the input sampling period. The timing unit needs thus to feed the VFD with a delay value adopting such a fractional format, *dly* in Fig. 4.1.

The *DIANA* algorithm has been developed satisfying these requirements, and focusing on the adaptive requirements of a variable resampling ratio. The algorithm normalizes any time information it processes to the input sampling period T_s. With this, the delay τ is not anymore expressed in seconds but in number of sampling periods at the input rate as

$$dly = \frac{\tau}{T_s} \qquad \text{Eq.(4.6)}$$

This provides a direct interface with the VFD. The resampling ratio R, Eq.(2.2), can be also expressed in terms of sampling periods instead of sampling rates

$$R = \frac{f'_s}{f_s} = \frac{T_s}{T'_s} \qquad \text{Eq.(4.7)}$$

Note that in this case the ratio is normalized to the output sampling rate, T'_s instead of to the input. It is thus more convenient to feed the resampler and the *DIANA* algorithm with the inverse of the resampling ratio, $1/R$, instead of R

$$T\_out\_n = \frac{1}{R} = \frac{T'_s}{T_s} \qquad \text{Eq.(4.8)}$$

We refer to this inverse resampling ratio as the value or the variable *T_out_n*. With this all the processing within the algorithm is normalized to the input sampling period. That makes it possible to easily compute the sampling instant of the input reference sample. We just need to increment by one a counter (or accumulator) each time that a new sample arrives. Similarly, the output sampling instant can be computed incrementing by $1/R$ a second counter (or accumulator) accumulating the normalized output time (output sampling instant). This output counter is incremented each time that a new sample is computed to prepare the algorithm for the next iteration. The delay value results from the difference between these two counters, for instance a delay equal to one output sample will be $1/R$.

The algorithm can still benefit from some optimization anticipating its implementation and mapping to a given technology, for instance, an FPGA. Recall that as the operation of the resampler advances and new samples are processed, the two time counters grow indefinitely. This is not efficient as

the discrete word containing a count cannot grow indefinitely in the same manner. To solve this problem the algorithm operates and stores only the instantaneous delay value needed to compute a new output sample. The *DIANA* algorithm references this delay to the time instant of last input sample received. In other words, our timing reference is the sampling instant of the last input sample. Our stored delay is the time difference between the current reference and the sampling instant of the desired output. Thanks to that, we store only a delay value that does not grow indefinitely, and not two growing times associated to the input and output sampling instants. The *DIANA* algorithm implementing this idea is presented in Fig. 4.5.

It evaluates two input control variables: *new_spl* that specifies if a new input sample arrives at the resampler, and *valid_prev* that keeps track of whether a valid output sample was calculated during the previous iteration of the algorithm. Another data input variable is *dly_incr* that corresponds to the delay increment for each new output sample as presented in Eq.(4.7) and Eq.(4.8). This value is a function of the current resampling ratio. The last input variable is *vfd_range* that specifies the maximum magnitude in fractions of the input sampling period accepted by the VFD, in our case *vfd_range* = 0.5 sample specifies a delay range of plus or minus half a sampling period.

The algorithm controls two output variables: *dly*, that stores the delay value needed to compute the next output sample based on the input reference of the resampler, and *valid_out*, that flags when the delay value is within the range of delay values accepted by the VFD, *vfd_range*.

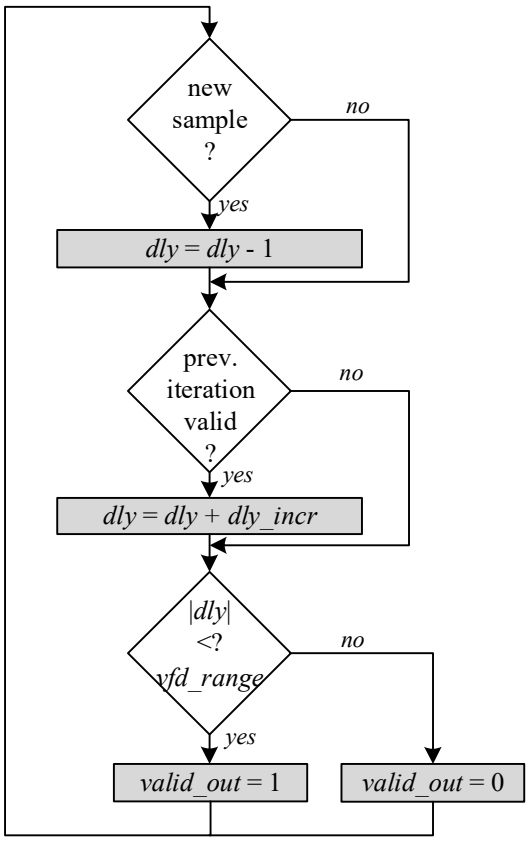


Fig. 4.5. *DIstAnce iN time Algorithm* (*DIANA*).

In each iteration the algorithm evaluates the control variables. In case a new input sample is received, the delay variable *dly* is decremented by one unit. This value results from decrementing the distance between input and output sampling instants by one input sampling period. Then, if the previous iteration has produced a valid output sample, the delay must be incremented by *dly_incr*. This updates the delay value to reflect that the sampling instant for the next output sample has moved further by one output sampling period. Finally, the algorithm evaluates the updated delay; when the delay magnitude is less than or equal to *vfd_range*, a new output sample can be computed in the iteration. This is indicated by asserting the output signal *valid_out*, that is forwarded to the VFD together with the delay value.

The algorithm is initialized with *dly* = 1. This ensures that no output sample will be processed until a first input sample arrives at the resampler; this value is above any fraction of the input sampling period accepted by our VFD, and aligns the first output sample with the reference based on the first input sample.

The variables of the algorithm can directly be mapped to the ports of the timing unit. The only exception is the *valid_prev* variable with the *valid_out* port. In this case the *valid_prev* variable evaluates the *valid_out* state in the previous iteration of the algorithm. This requires a memory element, a register, that stores *valid_out* between iterations.

The algorithm assumes that each new sample arriving at the resampler is directly inserted into the VFD filter. It handles changes in the resampling ratio, in real-time, by just updating the *dly_incr* variable (the ratio signal in the implementation). This is elaborated in more detail in Chapter 5.

4.3.2. The VFD filter

The VFD filters used in resampling applications are responsible for the interpolating process. The operation of such a filter is also a twofold problem. It first requires the computation of an impulse response (the set filter coefficients) that approximates a time shifting operation (delay of the input discrete sequence). Then this impulse response, that is different for different delays, has to be made available to the filtering architecture; it needs to update its coefficients to reflect the new delay. To better understand the problem, let's first have a look at the first problem, synthesis of a filter that approximates a generic and fixed delay value. Then, in the successive sections this filter will be upgraded to a fractional delay filter that accepts variable delays.

4.3.2.1. Discrete filters for delay synthesis

The filter has to compute a good approximate $y[m]$ of a signal $x(t)$ at the new desired sampling point $t_{y[m]}$. The motivation for the use of delay filters is that they transform the interpolation operation into the problem of synthetizing a discrete filter $h[n]$, that approximates a time shift; the shift in time instant, in which the continuous-time signal is sampled, is equivalent to the sampling of a shifted version $y(t)$ of the signal $x(t)$ by τ s. When τ is negative we call the shift operation "*a delay by* τ *s*", otherwise "*an advance by* τ *s*". When referring "*a delay by* τ *s*", we hence normally provide the magnitude $|\tau|$ of the negative valued shift variable $\tau < 0$ fed to the delay filter. This delay operation can be expressed as

$$y(t) = x(t + (-\tau)) = x(t - \tau) \qquad \text{Eq.(4.9)}$$

with $\tau \in \mathbb{R}$ for the shift operation. This continuous-time shifting is a well-known operation that results from convolving the input signal $x(t)$ with a Dirac delta with ideal impulse response $h_{id}(t)$

$$h_{id}(t) = \delta(t - \tau) \qquad \text{Eq.(4.10)}$$

By taking the Fourier Transform of the impulse response we obtain the continuous-time frequency response of the ideal delay operation

$$H_{id}(\Omega) = e^{-j\Omega\tau} \qquad \text{Eq.(4.11)}$$

This filter $H_{id}(\Omega)$ corresponds to an all-pass filter with unitary magnitude. The linear phase is the term that contributes to the delay operation. We want to find an equivalent discrete-time version $h_{id}[n]$ for our discrete fractional delay filter $h[n]$. Recall that when sampling any real signal, as the ideal $x(t)$ or $y(t)$, we first bandlimit its spectral content with respect to the sampling frequency f_s. We use for that antialiasing filters that have an ideal low-pass frequency response $H_{LP}(\Omega)$. We can plug the anti-aliasing filter into our delay operation by setting a cut-off frequency $\Omega_c = 2\pi \cdot (f_s / 2)$ for our ideal filter $H_{id}(\Omega)$. This transforms the desired frequency response to

$$H_{id}(\Omega) \rightarrow H_{LP}(\Omega) = 1 \cdot e^{-j\Omega\tau} \quad \text{for} |\Omega| \leq \Omega_c \qquad \text{Eq.(4.12)}$$

This prototype low-pass filter removes the frequency components above the cut-off frequency Ω_c without affecting the rest of the spectrum. It still observes group $gd_{id}(\Omega)$ and phase $pd_{id}(\Omega)$ delays that are constant in all the pass-band with value $gd_{id}(\Omega) = pd_{id}(\Omega) = \tau$ s. Note that as the cut-off frequency equals half of the sampling rate, the filter corresponds to the ideal filter used in reconstruction of discrete to continuous-time signals [89].

For the time being let's ignore the linear phase term $e^{-j\Omega\tau}$. We develop only the magnitude and the frequency bands. We can compute the impulse response $h_{LP}(t)$ of the filter as the inverse Fourier Transform of the frequency response $H_{LP}(\Omega)$, it becomes

$$h_{LP}(t) = \frac{\Omega_c}{\pi} \operatorname{sinc}\left(\frac{\Omega_c}{\pi} t\right) \qquad \text{Eq.(4.13)}$$

with the sinc(x) function defined as

$$\operatorname{sinc}(x) = \frac{\sin(x\pi)}{x\pi} \qquad \text{Eq.(4.14)}$$

The function equals zero for integer multiples of the variable x, and in the limit for $x = 0$ its value is one. Fig. 4.6 depicts the frequency response of Eq.(4.12) and the impulse response of Eq.(4.13) for a

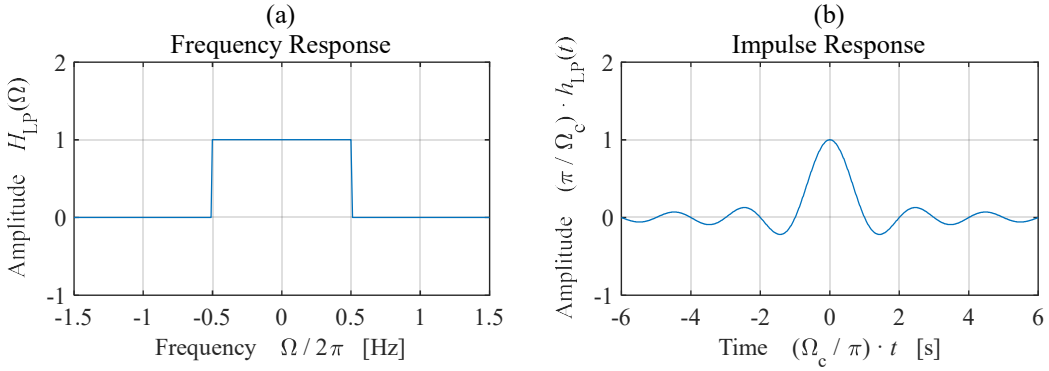


Fig. 4.6. (a) Frequency response $H_{LP}(\Omega)$ of Eq.(4.12). (b) Impulse response $(\pi / \Omega_c) \cdot h_{LP}(t)$ of Eq.(4.13).

$f_s = 1$ Hz. This continuous-time impulse response is also known as the sinus cardinal [98]. It corresponds to an ideal bandlimited interpolator [98] that makes it a perfect reconstruction of bandlimited signals $x_r(t)$ possible from a sampled sequence $x[n]$

$$x_r(t) = \sum_{k=-\infty}^{\infty} x[k] \cdot h_{LP}(t - k \cdot T_s) = \frac{\Omega_c}{\pi} \sum_{k=-\infty}^{\infty} x[k] \cdot \mathrm{sinc}\left(\frac{\Omega_c}{\pi}(t - k \cdot T_s)\right) \qquad \text{Eq.(4.15)}$$

This continuous-time impulse response is exact as the sinc(x) function is infinite and noncausal. We want to compute a discrete counterpart for our filter, but the sinc(x) cannot be made causal by shifting as it would require infinite shifting. It is only possible to obtain a finite-length approximation $h[n]$ of the ideal filter $h_{LP}[n]$ by sampling a shifted and truncated version of the continuous-time impulse response $h_{LP}(t)$ [98]. For this, the intermediate approximation $h_s(t)$ first shifts the impulse response $h_{LP}(t)$ by ζ s with $\zeta \in \mathbb{R}$

$$h_s(t) = h_{LP}(t - \zeta) = \frac{\Omega_c}{\pi} \mathrm{sinc}\left(\frac{\Omega_c}{\pi}(t - \zeta)\right) \qquad \text{Eq.(4.16)}$$

Then this shifted impulse response $h_s(t)$ is made finite by truncating it to a sufficient duration segment of L_{seg} s. This can be done by multiplying in the time domain $h_s(t)$ with a window function $w_{Lseg}(t)$ that spans L_{seg}

$$h(t) = h_s(t) \cdot w_{Lseg}(t) = \frac{\Omega_c}{\pi} \mathrm{sinc}\left(\frac{\Omega_c}{\pi}(t - \zeta)\right) \quad \text{for } t \in [0, L_{seg}] \qquad \text{Eq.(4.17)}$$

This results in a realizable impulse response $h(t)$ that approximates the ideal response of the pure delay. The resulting impulse response can then be sampled to obtain the discrete impulse response counterpart

$$h[n] = \sum_{b=0}^{B-1} q_b \cdot \delta[n - b] = \frac{\Omega_c}{\pi} \mathrm{sinc}\left(\frac{\Omega_c}{\pi}[n - D]\right) \text{ for } n \in [0, B-1] \qquad \text{Eq.(4.18)}$$

In Eq.(4.18) $D = \zeta / T_s$ samples is the shifting term, B the number of coefficients in the impulse response, q_b the value of the b^{th} coefficient of the impulse response and $t = n \cdot T_s$, with D and $q_b \in \mathbb{R}$, and B

and $n \in \mathbb{Z}$. We just need to incorporate the linear phase term $e^{-j\Omega\tau}$ of Eq.(4.12) and make $\zeta = \tau$ to obtain the discrete approximation of our required delay filter.

When processing a discrete sequence $x[n]$ with the computed impulse response $h[n]$ we will obtain the sequence $y[m]$ that approximates the input sequence shifted by τ:

$$y[m] = x[n-D] = x[n] * h[n] = \sum_{b=0}^{B-1} h[b] \cdot x[n-b] \text{ with } D = \tau / T_s \qquad \text{Eq.(4.19)}$$

To conclude, recall that we have used as prototype filter an ideal low-pass with cut-off frequency equal to one half of the sampling frequency, and we have approximated the resulting impulse response by sampling a shifted and truncated segment of its infinite sinc(x) function. We will see in the next section that this method best reflects the required delay when the shifting term D results an integer multiple of the sampling period. The obtained discrete filter works by "implicitly" reconstructing the original bandlimited signal, shifting it in time and finally resampling the resulting continuous-time signal [98]. When the frequency response constraints of the filter are stringent, or D is not an integer, different methods [98] exist to obtain prototype impulse responses that better approximate the required filter response; Least Squares phase and/or delay approximation, Maximally Flat approximations, Weighted Least Squares methods…, however, that topic is out of scope of the Thesis. In section 4.4.2.1 we briefly present the method that we have used to compute the used prototype impulse response in the Thesis. In any case the presented process clearly illustrates the implications that the delay operation poses to achieve a realizable discrete impulse response for a delay filter; the need for truncation and shifting of an ideal prototype impulse response.

4.3.2.2. Fractional Delay discrete filters

We have just presented an ideal discrete filter synthetizing a delay ζ . We are, however, interested in using this delay filter as the interpolating element for the digital architecture; as a building block of the VFD filter. A realizable version of the VFD filter generates the samples $y[m]$, depicted in Fig. 4.3 at arbitrary instants in time $t_{y[m]}$, Eq.(4.2). Any of these arbitrary instants lie between the sampling instants $t_{x[n]}$, Eq.(4.1), of the input sequence $x[n]$. Both originate the time difference τ of Eq.(4.3). This time was translated to a delay term, *dly* in Eq.(4.6), that is measured as a fraction of the input sampling period. We need thus to find the relation between this required fractional delay *dly* computed by the *DIANA* algorithm, the delay ζ (or D) synthetized by the filter in the previous section and a feasible architecture supporting the mapping of the filter.

Note that the filter delay in the previous section can adopt any real value ζ s. This delay when divided by the sampling period T_s results in an integer part D_i sample and a fractional part d sample:

$$D = \zeta / T_s = D_i + d \qquad \text{Eq.(4.20)}$$

We use the low-pass prototype filter, Eq.(4.18), of the previous section within an example to analyse the relations between the fractional delay *dly* and the integer and fractional parts of the term D for the two cases. For simplicity the sampling clock has a period $T_s = 1$ s. In the first case the required delay is

a pure integer multiple of the sampling clock, for instance $D = D_i = 7$ sample. The second case requires a delay of $\zeta = 0.4$ s that results in a D with only a fractional part $d = 0.4$ sample. We define our cut-off frequency again as $\Omega_c = 2\pi \cdot (f_s / 2)$ radian/s.

We start revisiting the ideal continuous-time impulse response of the prototype filter depicted in Fig. 4.7(a). This response is centred in the origin and thus not yet realizable, it corresponds with Eq.(4.13). The sinc(x) in the time domain expands the width of the lobes when the cut-off frequency Ω_c of the filter is reduced. In our case the cut-off frequency $\Omega_c = 2\pi \cdot (f_s / 2)$ makes the impulse response to equal zero in time values t that are integer multiples of the sampling period

$$\frac{\Omega_c}{\pi} \cdot t = \frac{2\pi \cdot F_c}{\pi} \cdot t = f_s \cdot t = \frac{t}{T_s} = \text{integer} \rightarrow t = n \cdot T_s \quad \text{with } n \in \mathbb{Z} \text{ and } n \neq 0 \qquad \text{Eq.(4.21)}$$

We now analyse the resulting impulse response when made it realizable for the first case, $D = 7$ sample. We use an odd B tap FIR filter architecture to map the coefficients of the filter $h[n]$. In these FIR filters the latency L sample accounts for the number of clock cycles that a sample takes to reach the central tap [71] and it becomes

$$L = \frac{B-1}{2} \qquad \text{Eq.(4.22)}$$

By making $L = D$ we obtain a digital filter that exactly reproduces the required delay, and has $B = 15$ taps. The continuous-time impulse response is shifted by $\zeta = D \cdot T_s = 7$ s for a sampling period $T_s = 1$ s, depicted in Fig. 4.7(b) by the blue trace. We have enough information to define the window that we use to make the impulse response realizable; the duration of the window for a symmetric filter becomes $L_{seg} = 2 \cdot L \cdot T_s = 14$ s. The resulting discrete impulse response, Eq.(4.18), that results from the sampling of the shifted continuous-time impulse response is depicted in the same figure by the red circles.

In this case, the shifting operation preserves the zeros of the response matched with the sampling instants, and thus the discrete impulse response become a delta at $n = D$. Furthermore, note that when the shift operation uses as delay an integer multiple of the sampling period, only the coefficient corresponding to $t = 0$ s in the prototype filter is non zero. This results in a discrete filter that rather than approximating reproduces with no error the continuous-time prototype; the sampling of the sinc(x) does not need to extend to the infinite. The Z transform of this perfect delay operation can be expressed as

$$H(z) = z^{-D} \qquad \text{Eq.(4.23)}$$

And the discrete version of Eq.(4.9) becomes

$$y[m] = x[n - D] \qquad \text{Eq.(4.24)}$$

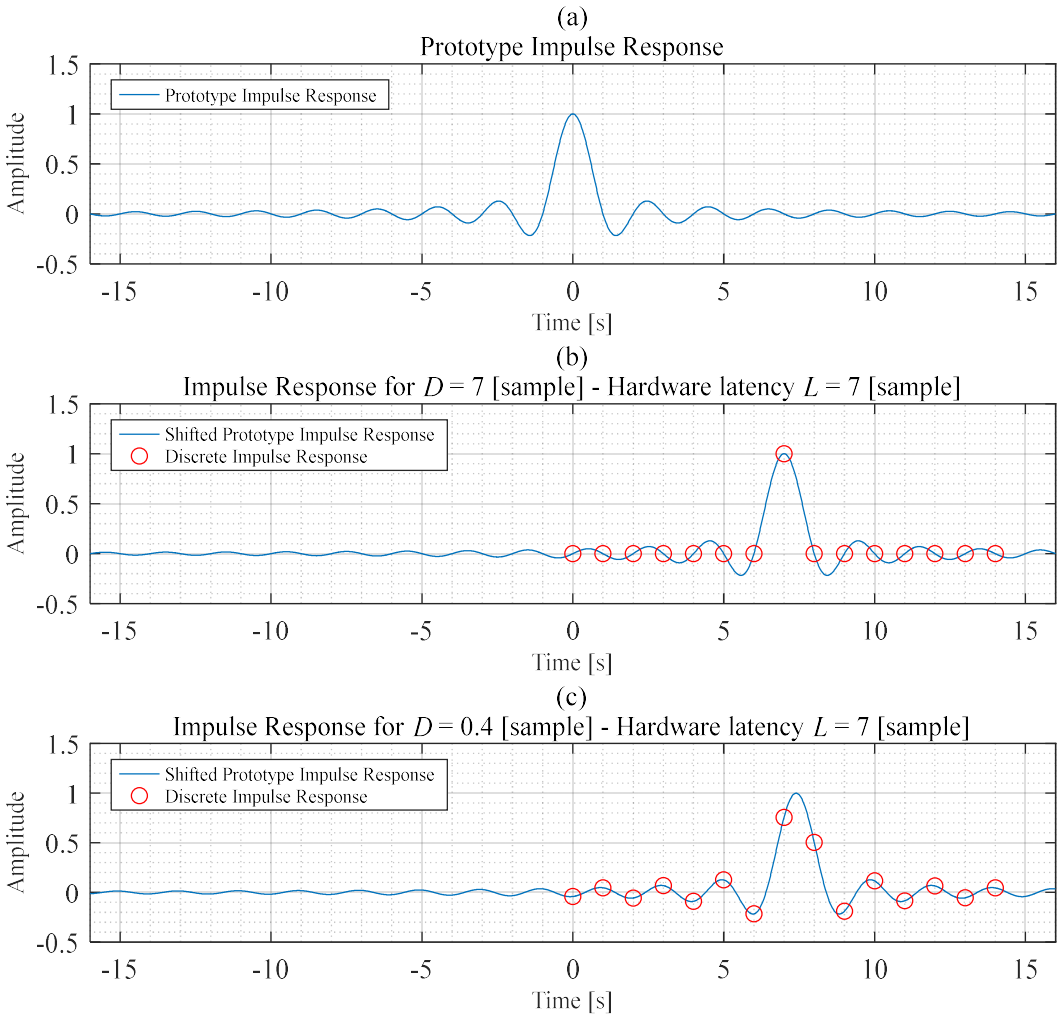


Fig. 4.7. Ideal impulse responses; (a) prototype filter, (b) shifted ideal response and sampled coefficients when the delay $D = 7$ sample, and (c) shifted ideal response and sampled coefficients when the delay $d = 0.4$ sample.

We are, however, interested in VFD filters where the delay *dly*, Eq.(4.6), contains only a fractional part, the term d in Eq.(4.20). For these VFD filters D_i is zero, but this is not a problem. We can use the same odd B tap FIR architecture as before to map the coefficients, setting $D_i = L$ to account for the filter latency. This will result in the required fractional shifting operation present at the output of the filter after a time $t = L \cdot T_s$ s, the latency of the filter.

Take our second case in the example with $dly = d = 0.4$ sample; we will solve it with a VFD filter in which the desired fractional delay $d = 0.4$ sample appears at the output after a latency of 7 samples as introduced above. We compute an impulse response with $D_i = 7$ samples and $d = 0.4$ sample, that corresponds to a shift of the prototype impulse response by $\zeta = 7.4$ s, depicted by the blue trace in Fig. 4.7(c). The duration of the window is again $L_{seg} = 2 \cdot L \cdot T_s = 14$ s. The resulting discrete impulse response is depicted superimposed in the same figure with the red circles at the sampling instants.

By inspecting the resulting impulse response in Fig. 4.7(c) we can realize that now the set of coefficients for the FIR filter (the sampled values of the sinc(x) in Eq.(4.17)) are not anymore matching the zeros of the sinc(x). The shifting operation does not preserve the correspondence between the zeros and

the sampling instants, that have an offset that equals the fractional part d of the delay. This brings as a consequence that now all the coefficients in the filter are different from zero. We would need an infinite number of samples to exactly reproduce the sinc(x) within a discrete impulse response, but we use only the subset lying within our window. This is acceptable for the implementation, but it results in a realizable discrete filter that now only *approximates* the continuous-time prototype. This approximation introduces an error in the frequency response $H(e^{j\omega})$ that it also only approximates the ideal frequency response $H_{LP}(\Omega)$ of Eq.(4.12) within a certain error margin.

4.3.2.3. Cut-off frequency relation to the resampling ratio

We have used an ideal reconstruction filter as prototype whose frequency response is that of an ideal low-pass filter having a cut-off frequency equal to half the sampling rate. This is not normally the case, as the prototype filter (that sets the cut-off frequency) is defined based on the resampling ratio used [89]. We can understand better the characteristics of the reconstruction filter by examining the sampling rate conversion operation from an analog perspective [89]. This analog equivalent of the discrete operation involves a two-step process depicted in Fig. 4.8. First the original discrete sequence sampled at f_s is reconstructed as an analog bandlimited signal $x_{rc}(t)$. This process includes pre-filtering of an intermediate reconstructed signal $x_0(t)$ to obtain $x_{rc}(t)$. This reconstruction filter $h_{rc}(t)$ is a low-pass filter that limits the spectrum of the analog signal $x_0(t)$ to the Nyquist frequency. It removes folded replications (images) of the sampled signal at multiples of the sampling frequency. Then, a second process samples the analog signal $x_{rc}(t)$ at the new output rate f'_s to obtain the output sequence $y[m]$. This second process includes also filtering of the intermediate analog signal $x_{rc}(t)$. It uses an anti-aliasing low-pass filter $h_{aa}(t)$, that limits the bandwidth of the signal according to the Nyquist rate.

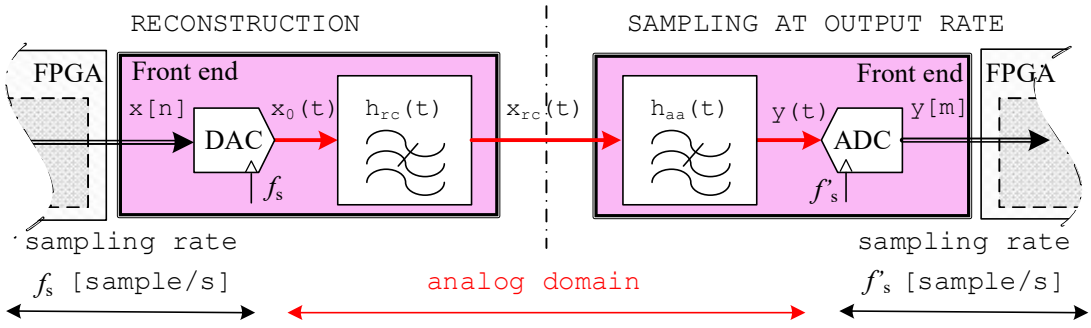


Fig. 4.8. Schematic representation of sampling rate conversion with analog reconstruction.

The Nyquist frequency in the reconstruction filter $h_{rc}(t)$ dictates the lower boundary at which images of the discrete spectrum appear folded in the reconstructed signal $x_{rc}(t)$. This is the highest frequency that can be coded at the sampling rate f_s making full reconstruction of the signal possible. The Nyquist rate is the minimum sampling rate satisfying the sampling theorem (twice the bandwidth of the bandlimited signal) [103]. This rate, given the output sampling frequency f'_s, sets an upper boundary in the spectral contents of the bandlimited signal $x_{rc}(t)$ for sampling without loss of information. This boundary dictates the cut-off frequency of the anti-aliasing filter $h_{aa}(t)$. As the reader can observe in Fig. 4.8, these two filters are in series. They can thus be combined by using the most restrictive characteristics originating a new single

filter $h_c(t)$ that reduces the number of required operations and resources [89]. The frequency response of the new low-pass filter $h_c(t)$ combining both responses inherits thus the most restrictive cut-off frequency Ω_c radian/s according to

$$H_c(\Omega)=\begin{cases}1 \text{ for } |\Omega|\le\Omega_c\\ 0 \text{ for } |\Omega|>\Omega_c\end{cases} \quad \text{with } \Omega_c = \min\left\{\frac{2\pi\cdot f_s}{2},\frac{2\pi\cdot f'_s}{2}\right\} \text{ radian/s} \qquad \text{Eq.(4.25)}$$

The resulting combined analog equivalent is depicted in Fig. 4.9, where $h_c(t)$ denotes the impulse response of the combined filter $H_c(\Omega)$.

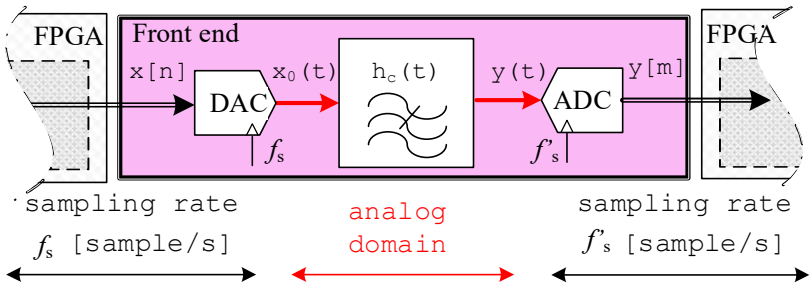


Fig. 4.9. Schematic representation of sampling rate conversion with analog reconstruction merging the two analog filters.

We can think of this procedure as the method to define the cut-off frequency of our fractional delay filter. It makes it possible to tailor the filter response based on the most restrictive sampling rate at the input and output ports of the resampler, provided that the input signal is bandlimited below any of these two rates. In any case, the resulting continuous-time filter using these constraints or any other method needs to be made realizable as presented in the previous sections.

4.3.2.4. Variable Fractional Delay discrete filters

We have presented so far how to estimate the coefficients of our reconstruction filter based on a prototype frequency response. We know how to design a FD filter that has a fixed delay value, and we have developed the procedure to map it to an FIR architecture. We are, however, interested for our interpolator in a VFD with a variable delay. This implies a different impulse response dependent on each required delay value. We can think of our VFD as a reconfigurable FIR filter, in which we update the coefficient set based on the required delay. We, however, need to analyse the consequence of this coefficient update requirement, and sketch a strategy to obtain and update the coefficients of the FD filter based on the delay.

Revisiting one more time the procedure to obtain the set of B coefficients, one realizes that it is always the same no matter what is the delay; we shift (based on the required delay) and sample the associated continuous-time impulse response within a window. One feasible option could be thus to store different sets of coefficients based on the different delay values that we will use. Unfortunately, as we introduced in section 4.3.1.2, our variable resampling ratio makes the possible values for the delay infinite. This, in practice, makes storing multiple sets of coefficients an inefficient approach. We would require infinite sets for an infinite delay precision.

This unfeasible approach is depicted in Fig. 4.10; in the example we present a VFD filter $h[n]$ whose B coefficients are read from a table and updated, based on the required delay. The delay range spans between plus and minus half of an input sampling period, -0.5 sample ≤ dly ≤ 0.5 sample, and we want a delay resolution of 0.1 sample. This results in eleven possible configurable delays, dly_A = {-0.5, -0.4, …, 0.4, 0.5} with $A \in [0, 10]$, and a coefficient set $h_A[n]$ per delay value. When implementing this filter, if we store the coefficient sets in a memory, we need as many rows as taps in the filter, B, and as many columns as delay slices, A; ($B \times A$) table. We read the memory based on a column index computed from the desired delay, extracting the rows in parallel. In the figure, following the examples of 4.3.2.2 we match the latency of filtering architecture $D_i = L$, and we do $d = dly$.

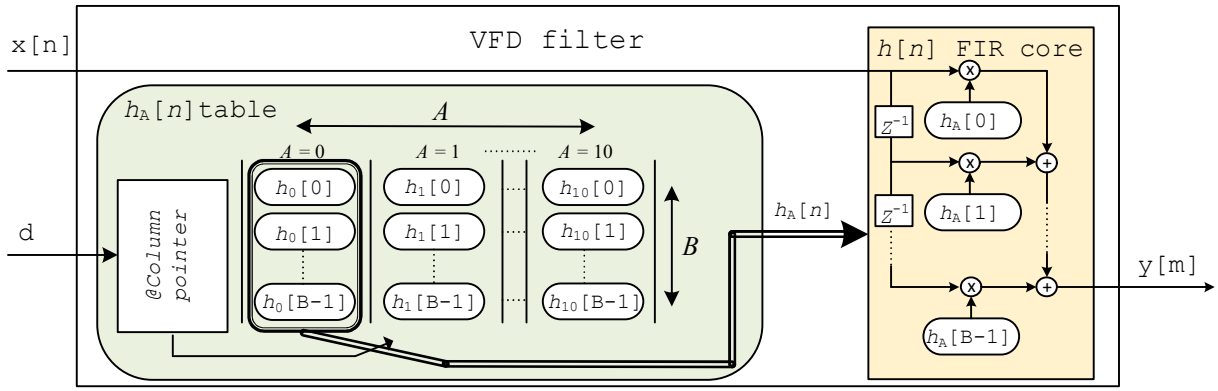


Fig. 4.10. Filtering architecture with the filter coefficients stored in a table accessed based on the delay value.

Instead of storing multiple sets of coefficients ($A \rightarrow \infty$), we can develop a method that lets the hardware implementing the VFD to compute the set based on the required delay. This replaces the infinite memory constraint with a requirement for an efficient and accurate coefficient calculator.

The idea behind this calculator is to divide the continuous-time impulse response of the prototype filter in B segments (one per coefficient) that span for one sampling period, and are centred around the zero crossing of the sinc(x) response, as depicted in Fig. 4.11. Each segment is then approximated by a low order piecewise polynomial that is evaluated with the desired delay d as parameter. These polynomials are thus generators (the calculators based on the required delay) for each one of the B coefficients of the prototype filter of Eq.(4.18) and Eq.(4.19). The resulting coefficient set is passed to the filter core each time the delay is updated. For the sake of simplicity, in the following we disregard the latency of discrete-

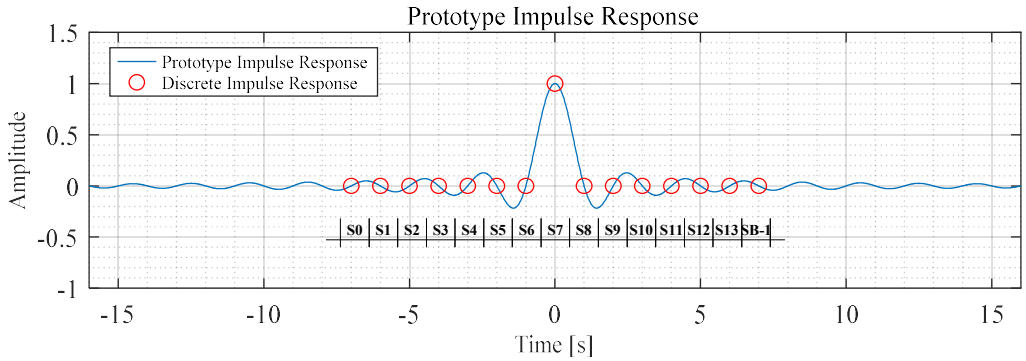


Fig. 4.11. Prototype impulse response divided in B segments.

time implementation, we make $D_i = L = 0$. The polynomials mapping the segments of the impulse response can be expressed as Taylor series [72] becoming

$$h[n=b] = P_b(d) = \sum_{c=0}^{C} g(b,c) \cdot d^c \qquad \text{Eq.(4.26)}$$

with b the index of the coefficient in the FD filter $h[n]$ (Eq.(4.19)), C the order of the polynomial, d the fractional delay value and $g(b,c)$ the c^{th} order coefficient of the polynomial originating the b^{th} coefficient of the filter. The matrix $g(b,c)$ corresponds to a matrix of B rows with C+1 columns. Each one of the B coefficients of the filter $h[n]$ has hence a different set of C+1 polynomial coefficients (calculators). The first coefficient b=0 of the prototype filter $h[n]$ (obtained from the polynomial approximation of the segment "S 0" in Fig. 4.11) is for instance computed by plugging the C+1 elements of the row b=0 in $g(b,c)$ (Eq.(4.26)) with the given delay d.

Fig. 4.12 depicts the architecture implementing this approach. It contains the hardware core implementing the FIR reconstruction filter $h[n]$, and for the coefficients the B polynomial generators of order C controlled by the delay d.

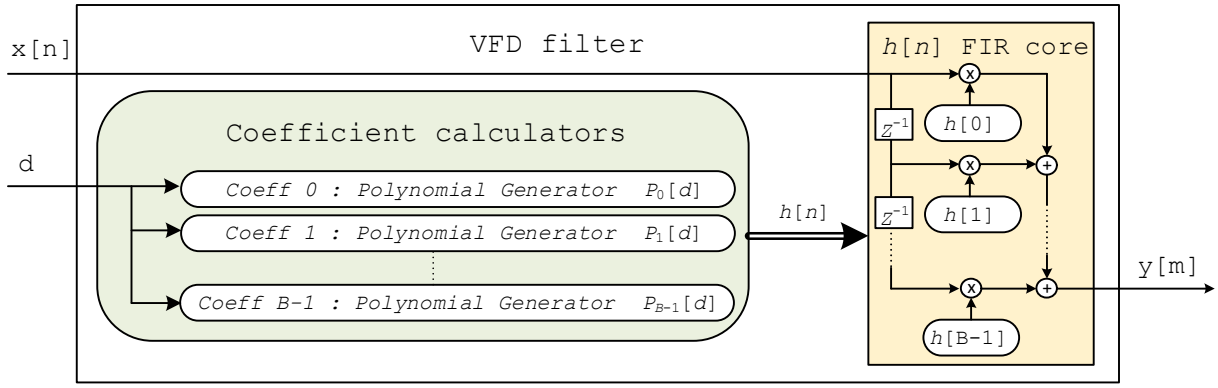


Fig. 4.12. Filtering architecture with the filter coefficients approximated by piecewise polynomial.

4.3.2.5. The Farrow Architecture

The architecture presented in the previous point is a feasible solution for the VFD. However, it requires the computation of the reconstruction filter coefficients before the filtering operation. This requires some control and synchronization mechanism to properly synchronize the filtering stages. An alternative architecture was proposed in [77]. The architecture is known as the Farrow architecture. It combines the polynomials of the coefficients, Eq.(4.26), within the convolution operation in the filtering architecture, Eq.(4.19). This generates an efficient architectural alternative that does not require any computation of coefficients beforehand. Instead, the input samples are pre-filtered with a bank of filters resulting from the arithmetic manipulations on the convolution and the polynomials. The outputs of the bank are combined with the delay value in a Horner structure [79], that efficiently solves the evaluation of polynomials reducing the hardware resources needed and enables the real-time delay update.

The Farrow architecture inserts Eq.(4.26) in Eq.(4.19) to obtain

$$\begin{aligned} y[n] &= x[n-D] = \sum_{b=0}^{B-1} h[b] \cdot x[n-b] \\ &= \sum_{b=0}^{B-1} P_b(d) \cdot x[n-b] \\ &= \sum_{b=0}^{B-1} \sum_{c=0}^{C} g(b,c) \cdot d^c \cdot x[n-b] \end{aligned} \quad \text{Eq.(4.27)}$$

Reordering the summations, we can obtain

$$\begin{aligned} y[n] &= \\ &= \sum_{b=0}^{B-1} \sum_{c=0}^{C} g(b,c) \cdot d^c \cdot x[n-b] \\ &= \sum_{c=0}^{C} d^c \sum_{b=0}^{B-1} g(b,c) \cdot x[n-b] \end{aligned} \quad \text{Eq.(4.28)}$$

We can define a new set of data $u(n,c)$ based on the input sample $x[n]$ and the matrix $g(b,c)$ as

$$u(n,c) = \sum_{b=0}^{B-1} g(b,c) \cdot x[n-b] \quad \text{Eq.(4.29)}$$

By examining Eq.(4.29) we can realize that the data $u(n,c)$ is a vector of C+1 elements linearly dependent on the input data $x[n]$. This input dependent vector results from C+1 parallel convolutions with the sequence $x[n]$. The B coefficients of each parallel convolution correspond to the columns of $g(b,c)$ in Eq.(4.26). Note that now the matrix $g(b,c)$ is read by columns instead of rows. We thus use the same coefficients used in the polynomial generators but we read them by columns instead of by rows (Eq.(4.26) and Fig. 4.12), and we use the input samples $x[n]$ instead of the delay d. We can rewrite $u(n,c)$ as vector of C+1 samples dependent on the input data $x[n]$

$$\bar{u}_c[n] = u(n,c) = \sum_{b=0}^{B-1} g_c[b] \cdot x[n-b] = x[n] * g_c[n] \quad \text{Eq.(4.30)}$$

This vector $\bar{u}_c[n]$ can be seen as the outputs of a bank of C+1 filters $g_c[n]$ pre-processing the input samples $x[n]$. By inserting Eq.(4.30) in Eq.(4.28) we obtain

$$y[n] = x[n-D] = \sum_{c=0}^{C} d^c \cdot \bar{u}_c[n] \quad \text{Eq.(4.31)}$$

This last equation is a Taylor series representation of the filtered output sequence $y[n]$, that uses the vector $\bar{u}_c[n]$ as coefficients and evaluates the series with the delay value d [72].

The Eq.(4.31) can be expanded and results

$$y[n] = x[n-D] = \sum_{c=0}^{C} d^c \cdot \bar{u}_c[n] = u_0[n] + d \cdot u_1[n] + d^2 \cdot u_2[n] + \ldots + d^C \cdot u_C[n] \quad \text{Eq.(4.32)}$$

Finally evaluating this series with the Horner [79] rule we obtain

$$y[n] = x[n-D] = u_0[n] + d \cdot (u_1[n] + d \cdot (u_2[n] + d \cdot (\ldots + d \cdot u_C[n]))) \quad \text{Eq.(4.33)}$$

The resulting VFD architecture, that is depicted in Fig. 4.13, does not need polynomial re-computation of the interpolating coefficients for each output sample. Instead, it exploits the reordering of the filtering arithmetic operations within the Farrow architecture. It is composed of a bank of FIR sub-filters $g_c[n]$, which pre-processes the available input samples, $x[n]$ in Eq.(4.30), to generate intermediate filtered data $\bar{u}_c[n]$. The filter bank coefficients $g_c[n]$ are static and pre-computed offline resulting from the matrix $g(b,c)$, that approximates segments of the prototype impulse response with piecewise polynomials. A new output $y[m]$ is computed combining the intermediate data $\bar{u}_c[n]$ with the delay value $d = dly$ sample using the Horner rule. The output data is obtained by feeding the filter just with only the delay `d` and data `x[n]`.

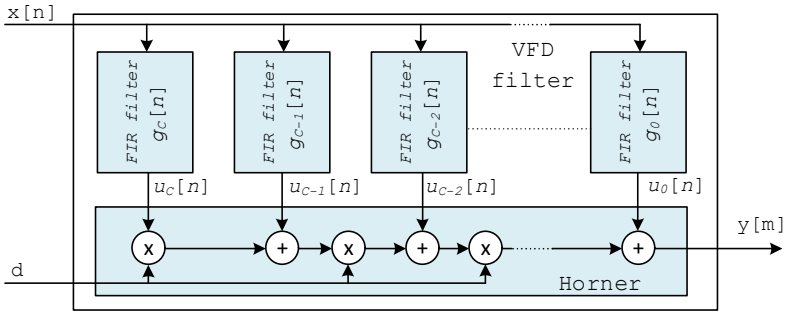


Fig. 4.13. VFD architecture based on the Farrow architecture and the Horner rule.

4.4. Implementation of the SRC architecture

The high-level architecture for implementation of the proposed resampler is presented in Fig. 4.14. From a logic point of view, only six signals constitute the interface of the architecture. These signals are the hardware equivalent to the functional ones presented in Fig. 4.1. At the input interface, the data-path input port feeds the available data samples; it contains the `data_i` bus signal and the `valid_i` qualification signal. The `data_i` bus signal corresponds to the $x[n]$ and the `valid_i` qualification signal to *new_spl* in Fig. 4.1. A second input port receives the control signal `T_out_n`, that lets the resampler know the relation between output and input rates. This signal corresponds to the R signal in Fig. 4.1 and feeds the value $(1/R)$ as presented in Eq.(4.8). The third input port is a clock port, `clk`, driving the hardware. This clock is the processing clock in the convention followed in this document. At the output, only the data-path port is present, it contains the `data_os` bus signal and the `valid_os` qualification signal. The first corresponds to $y[m]$, while the second to *valid_out* in Fig. 4.1. Looking at the internals of the resampler, it is composed of three functional blocks. The first entity, in red in Fig. 4.14, hosts the *DIANA* engine, which implements the algorithm presented in Fig. 4.5, for the computation of the time shift (`dly` signal). It also controls when an output data sample can be computed (`op` signal). The second block implements the VFD filter, in blue in Fig. 4.14. This hosts the bank of FIR filters to process the available samples, and the Horner chain of adders and multipliers combining these filter outputs as outlined in Fig. 4.13. Finally, the third block, in green in Fig. 4.14, contains the "Control Logic and Synchronization Memories". This block handles the communication and synchronization between the entities of the resampler and the interface ports.

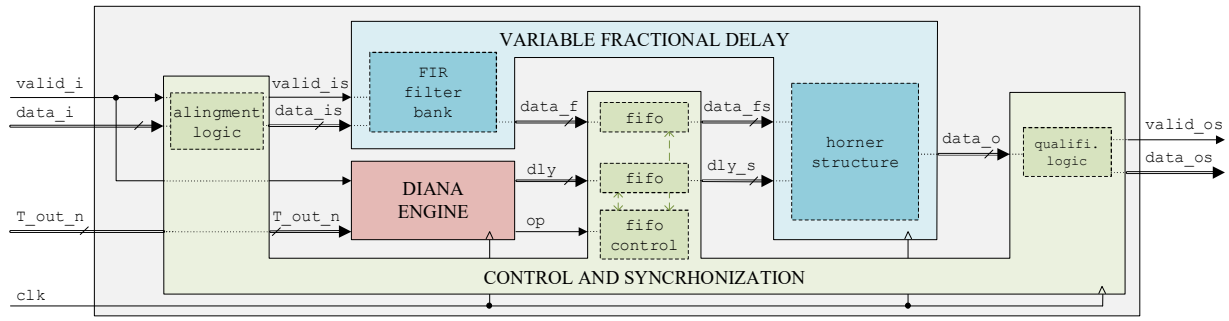


Fig. 4.14. High-level architecture of the implementation of the resampler.

4.4.1. Decoupled data-path SSRC architecture with arbitrary variable ratio

Sampling rate conversion architectures based on the use of a VFD and timing engine employ normally at least two clock domains, marked `clk_x` and `clk_y` in Fig. 4.15. They are better suited for arbitrary ratio ASRC implementations. The input clock domain is used to feed the data to the VFD filter bank. The output clock domain activates the Horner combiner when an output data sample is required. The timing unit computes the delay based on the difference of sampling instants between the clock domains. As presented in Chapter 2 and section 4.2.2 such an approach requires the output clock domain to sweep in frequency if the resampling ratio changes.

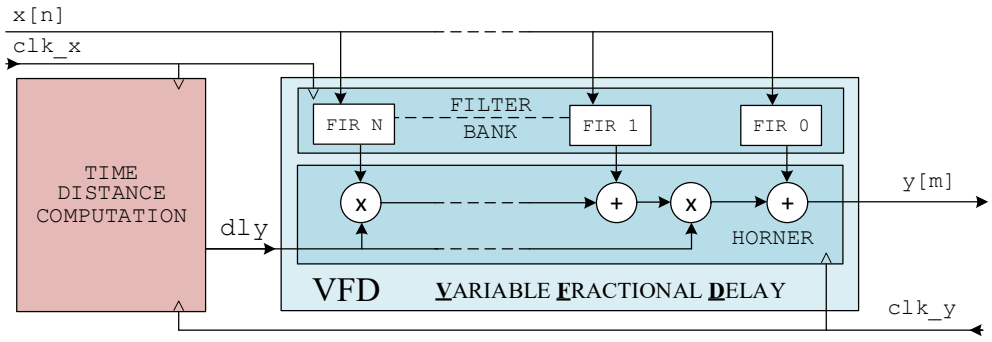


Fig. 4.15. Arbitrary ratio SRC architecture based on a Farrow VFD with different input and output clock domains.

In our case we focus on a SSRC architecture that accepts a variable resampling ratio. Our goal is to use a single processing clock, as depicted in Fig. 2.5. We thus use a decoupled data-path in the FPGA implemented on the *FRANCISCO* fabric. This combination makes it possible to have different sampling rates in the data-path using a single processing clock, and to vary sampling rates in real-time. From a functional point of view the VFD is the same element. It requires no architectural modifications for implementation within the *FRANCISCO* fabric. At implementation level, the registers in the taps of the filters use an enable signal as presented in Chapter 3 to propagate valid samples only. The common clock is used in both the filter bank and the Horner, however, the insertion of data in the filter is done based on the input data-path valid signal, and the validation after the Horner based on the output data-path valid qualification signal, `valid_is` and `op` (or its synchronized version `valid_os`) respectively in Fig. 4.14.

The same applies to the timing unit; from a functional point of view the use of the *FRANCISCO* fabric is transparent. At the implementation level it infers the resampling ratio from an external signal as

introduced in section 4.2, `T_out_n` in Fig. 4.14. This entity is the responsible for handling the different sampling rates in the data-path. It controls the incoming valid samples and modifies the valid signal of the decoupled data-path to reflect the change in ratio at the output of the resampler. The up-sampling ratio is limited by the relation between processing and sampling clock as presented in Chapter 3, section 3.3.3.

The main problem to be addressed in this implementation is thus to synchronize the data-path signals among the different elements of the VFD, and with the delay in the time-distance engine. The following sections address these implementation issues.

4.4.2. VFD implementation

Introduced in Fig. 4.13, the VFD unit contains two functional elements: The bank of FIR filters $g_c[n]$ and the Horner combiner. The architecture of the VFD and the interfacing signals are as depicted in Fig. 4.16. In the filter-bank, the `valid_is` signal enables the tapped delay line registers of the bank when a valid data is present in the data-path. The data signal `data_is` feeds the filters of the bank with the new samples in parallel. In the Horner structure, the `data_fs` signals feed C+1 buses containing the outputs of the filter bank, $\bar{u}_c[n]$ vector. The `dly_s` signal contains the delay value used for the current output sample. These signals need synchronization among themselves, hence the postfix "s". FIFO memories are used for this (in green in Fig. 4.16). The memories are part of the "Control Logic and Synchronization Memories" entity.

The filters in the bank are FIR ones. They are implemented with enabled registers in the taps, to cope with the decoupled data-path architecture. Recall that the resampler is implemented in a *FRANCISCO* fabric, and not all the clock cycles contain valid data in the data-path, hence only the data of the valid processing slots needs to be inserted in the filter. The even filters ($g_c[n]$ with c even) have coefficients which are even symmetric while the odd filters ($g_c[n]$ with c odd) have also odd symmetric coefficients. The number of coefficients (B in Eq.(4.29)) is dependent of the desired precision in the interpolation process of the VFD. The filters are implemented benefiting from the coefficient symmetry, they are folded around the central tap. The coefficients of the filter are scaled to avoid overflow in the internals of the filter. This scaling factor is compensated by re-scaling the data-path at the output of the filter.

A chain of adders and multipliers following the Horner rule combine the outputs of the different filters of the bank with the delay parameter, Eq.(4.33), signal `dly_s` in Fig. 4.16. There is one multiplier per filter bank branch, except for the last one. There is also one adder per branch except for the first one. The arithmetic elements, multipliers and adders, are pipelined with registers to segment the combinatorial path to ease the timing in the place and route process. The registers used in the pipeline and in the arithmetic operators do not have enable signal. This makes the propagation time between any of the data-path inputs and the output known. The inserted registers in the input data-path branches account for the pipelined arithmetic operators, and compensate the propagation of the signals in the arithmetic operators.

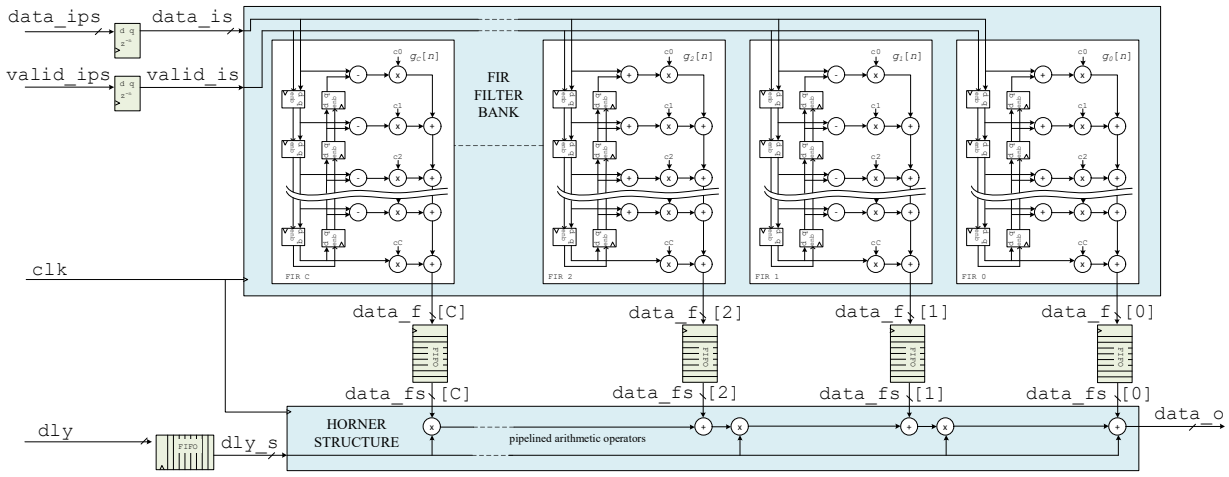


Fig. 4.16. Architectural view of the VFD filter. In blue, the FIR bank of filters and the Horner architecture. In green, the synchronization memories, not part of the entity.

4.4.2.1. The coefficients

The coefficients of the Farrow-based VFD of Fig. 4.16 have been obtained using the Weighted-Least-Square (WLS) method [104]. The delay signal `dly` takes values from -0.5 to 0.5 sample, `dly` $\in$ [-0.5, 0.5]. For each value we want the response from the input to the output to approximate an exact delay. Let $H(e^{j\omega},d)$ be this transfer function, with ω the normalized angular frequency in radian/sample and d the delay, the WLS computes the filter coefficients that minimize the cost function

$$J = \int_0^{\alpha\pi}\int_{-0.5}^{0.5}\left|H\left(e^{j\omega},d\right)-e^{-j\omega d}\right|^2 \mathrm{d}d\ \mathrm{d}\omega \qquad \text{Eq.(4.34)}$$

The parameter α is a fixed number, smaller than one, that specifies the pass-band (between 0 and π radian/sample) over which the cost is evaluated. The number of filters (C+1) and the filter length (B) must be chosen to achieve the required precision in the pass-band. In our design, we set $\alpha = 0.6$, and used six filters containing fifteen taps each. The resulting maximum square error for all values of d, in the pass-band is less than $5\cdot10^{-9}$. This was found to be good enough for our application. The number of taps has a big effect. When reducing it to seven, without changing any other parameter, the maximum square error increases to $3\cdot10^{-5}$. Further details are provided in Chapter 5.

4.4.3. *DIANA* algorithmic engine

The following section presents implementation details for the *DIANA* algorithmic engine. We first introduce the resulting features when implementing the *DIANA* engine with a decoupled data-path in the *FRANCISCO* fabric. Then we show the proposed implementation for the engine.

4.4.3.1. Ratio limits

We saw in section 4.2.3 that the use of a VFD as interpolating unit for resampling of a discrete sequence, results in a generic unit in the sense that it can be used for either up-sampling or down-sampling without modifications. The *DIANA* algorithm presented in section 4.3.1.2 is the solution in our architecture for the computation of the delay fed to such a VFD filter. This algorithm is also generic, it supports both

up-sampling and down-sampling, and solves our key concern; the capability to compute the delay with an arbitrary ratio value that changes in real-time. We focus in this section on the implementation of the algorithm; our main architectural decision is the use of the fixed processing clock to avoid clocking and synchronization problems related to the variable ratio. For this, we use the *FRANCISCO* fabric synthetizing a decoupled data-path, that copes with the variable sampling rate at the output port of the resampler, resulting from the variable ratio. This decoupled data-path brings, however, some implications in the performance that the algorithm can achieve once implemented.

Another functional requirement in our solution is the feasibility to be used in a feedback loop, where processing in real-time is required. We insert the incoming samples directly in the VFD to be able to perform online resampling; we do not want to store this data in a buffer where to perform subsequent offline resampling reading that data on demand. We need hence a unit that receives, processes the data as it arrives, and outputs the new sequence. The objective of the decoupled data-path is to make the use of the fixed frequency processing clock feasible in the system responsible for that. When this clock has a higher frequency than the data sampling rate, we ensure more processing slots than available samples in the decoupled data-path. This avoids its overflow within some boundaries and partially solves our swept clock concern; we have now void processing slots to populate with new data and we can use a fixed frequency processing clock.

The distribution of incoming samples in the data-path is also relevant. Note that all the input samples are inserted in the FIR filter bank as all of them are used in the interpolation process, however, not all of them are used as reference for an output when they reach the central tap. Take the case of down-sampling, as presented in Fig. 4.2 or Fig. 4.17; the resulting time spacing in the output sequence between sampling instants of the subjacent analog signal is always larger than the input sampling period, and larger than the VFD input range, *dly* ϵ [-0.5, 0.5] sample. We will hence never use an input sample as delay

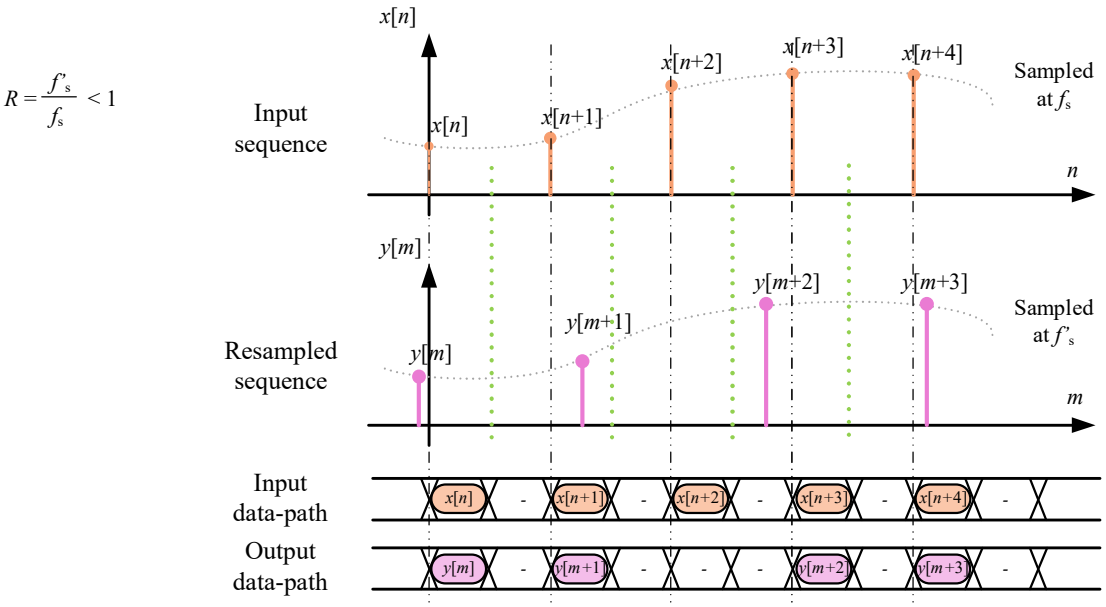


Fig. 4.17. Decrease in the number of populated slots in the data-path in the case of down-sampling.

reference for more than one output, and in some cases nor for one (the case when no output sample is calculated between two correlative input samples that result from a delay value larger than 0.5 sample). More void processing slots are thus found in the output sequence. This is depicted in Fig. 4.17 where we down-sample a sequence and we show the resulting distribution of samples populating the processing slots in the data-path (we neglect the processing time). The 0.5 sample delay window that the VFD accepts is depicted in the figure by the green vertical lines around each input sample. In this case, the sample $x[n+2]$ is not used as reference to compute any output sample; $y[m+1]$ is computed with $x[n+1]$ and as the delay value is above 0.5 sample for $x[n+2]$, the output $y[m+2]$ is computed using as reference $x[n+3]$.

Any down-sampling ratio, $R < 1$, is hence suitable for the decoupled data-path as there is no way to overflow it, no matter what the distribution of input samples we receive is. The figure depicts also for this case the output data-path that contains less populated processing slots at the output than at the input.

In the case of up-sampling, the decoupled-data (and the input distribution of valid samples) introduces some further considerations; we want to insert new valid samples in the output that alter the input distribution. The output spacing between sampling instants, that emulates the acquisition of the analog signal, is always smaller than the input sampling period, and hence smaller than the input delay range of the VFD (one input sampling period, from -0.5 to +0.5 sample). We will hence use each input sample, as delay reference, to compute at least one output and in some cases more than one. We need hence a data-path with enough void processing slots where to fit this new data. Depending on the population of the input data-path we will be able to do it so or not. The up-sampling rate is hence linked to the relation between processing clock and sampling rate at the output of the *MERCEDES Decouple* interface, $M = f_p / f_s$, and to the input sample pattern (*valids*) dependent on the physical position of the resampler in the data-path.

When the resampler is placed just after the *MERCEDES Decouple* interface, the activation rate of the data-path is $ar_{in} = (1 / M)$ and the input pattern is a uniform and periodic distribution of valid samples

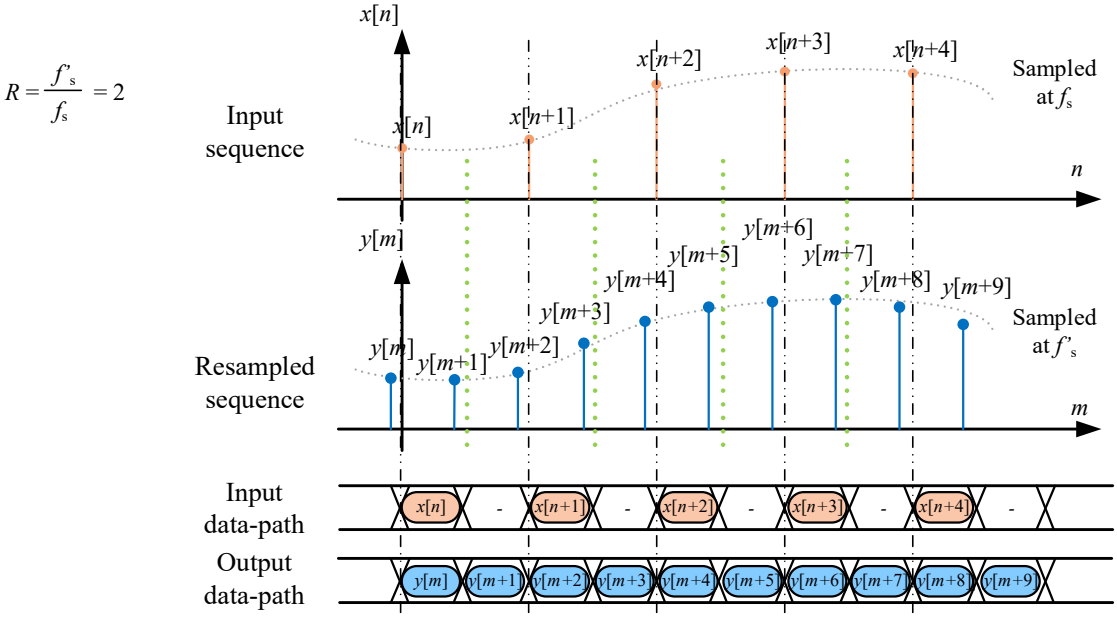


Fig. 4.18. Increase in the number of populated slots in the data-path in the case of up-sampling.

without bursts. One out of M processing slots contains valid data and the others are void. We will represent and refer such a pattern as UP_M : {1 0 0 … 0 1 0 0 … 0 1 0 0…}, with M - 1 zeros in between the ones. In this case, we have always free space, free processing slots, between input samples where to insert new outputs, and the *DIANA* engine has enough clock cycles to compute all the possible outputs for a given reference. The maximum possible up-sampling ratio for this configuration is hence $R = M$, as we can populate these M - 1 available processing slots in between. This is depicted in Fig. 4.18; the data-path has an $M = 2$ and we up-sample a sequence with a ratio $R = 2$, we show the input periodic pattern UP_2 : {1 0 1 0 … 1 0 1 0 … 1 0 …} and the resulting distribution in the data-path {1 1 1 1 … 1 1 1 1 … 1 1 …}.

Another situation that we can find happens when we do not satisfy this periodic and uniform distribution of samples in the input data-path port of the resampler. Note that this does not mean that the samples have not been periodically sampled, it means that the sequence of samples arrives to the resampler with a non-periodic and/or uniform pattern (for instance, some further processing between the *MERCEDES* and the resampler alters the distribution). In that case we can have two consecutive samples arriving in a burst, but keeping still an average sampling rate $ar_{in} = (1 / M)$, {1 0 … 1 0 1 1 0 0 1 0… }. In the middle of this burst, there are no free processing slot available. In this case, we cannot ensure that we will not overflow the output data-path of the resampler (and underflow the delay value in the *DIANA* engine). It might be the case that a new output sample needs to be inserted in a inexistent slot in the middle of the burst, or that the engine has no enough clock cycles to compute all the possible outputs for a given reference, resulting both in blocking of the architecture. When the input distribution is hence not known or not uniform and periodic (UP_M), we cannot operate the resampler in up-sampling mode safely.

An example is depicted in Fig. 4.19 where we up-sample a sequence with a ratio $R = 2$. The clock relation in the data-path is also $M = 2$, but now there is an input burst. With this ratio we compute two output samples per input reference, the maximum up-sampling value. We hence need an extra void

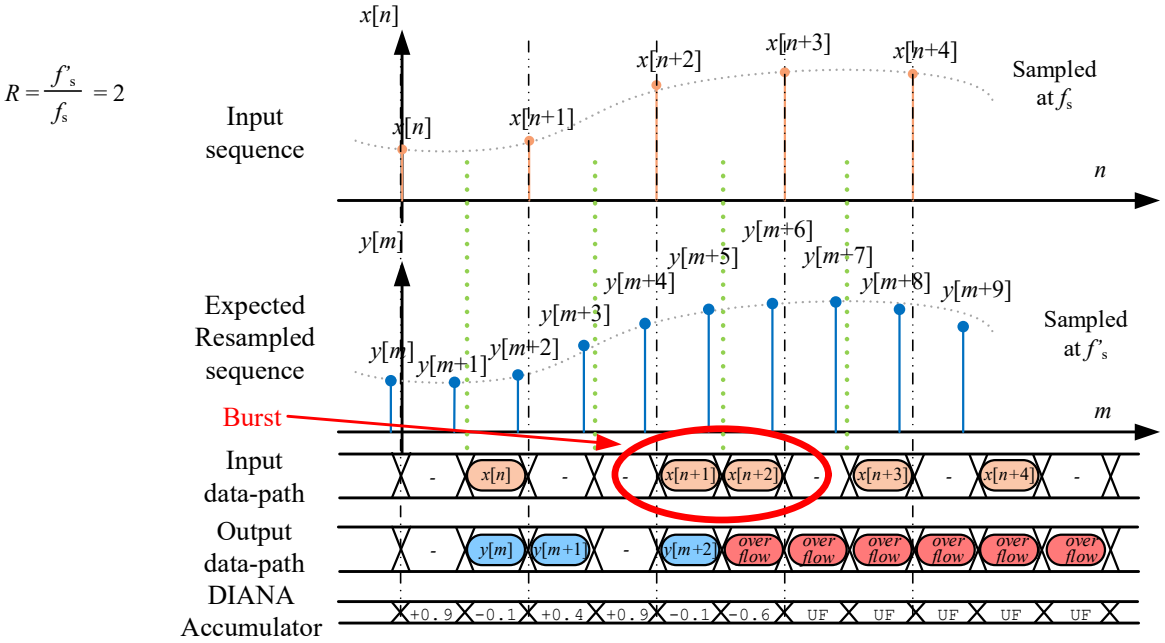


Fig. 4.19. Data-path overflow and accumulator underflow; the distribution of populated slots is altered and contains bursts.

processing slot per input sample. In the figure, we see the analog signal and the resulting periodically sampled sequence $x[n]$. We can observe as well the distribution of these samples in the processing slots of the input data-path, altered with respect to Fig. 4.18. The overflow event is depicted with a red ellipse. Take a look at the burst in the $x[n+1]$ and $x[n+2]$ samples; these arrive in consecutive processing clock cycles, with a delay value of 0.9 sample in the *DIANA* accumulator prior to the $x[n+1]$ arrival. The algorithm updates the value resulting in $0.9 - 1 = -0.1$ sample (arrival of a new sample but no valid output in the previous iteration). This value lets us compute $y[m+2]$. In the subsequent processing clock cycle, the algorithm updates again the value. If no new sample would arrive the result would be $-0.1 + 0.5 = 0.4$ sample and we could hence compute $y[m+3]$ as expected. However, $x[n+2]$ arrives within the burst, this results in a final computed delay value of $-0.1 + 0.5 - 1 = -0.6$ sample. As a result of the burst, the accumulator underflows the delay value ($-0.6 < -0.5$ sample, minimum VFD range value), and overflows the data-path as there is no available slot where to insert $y[m+3]$. The *DIANA* engine can detect this underflow condition when the delay adopts a value below -0.5 sample. Unfortunately, it cannot recover automatically as the only way to increment the accumulator delay is by computing new output samples; this is not any longer possible as the current value is under the valid range of the VFD, remaining thus blocked.

The maximum up-sampling ratio is hence a function of the input sample pattern and the relation between processing and sampling clocks. When the relation between frequencies in the sampling and processing clocks is M, being the processing clock higher, and the input samples are equispaced with (M-1) samples between valid inputs, $ar = 1/M$ satisfying UP_M : {1 0 0 … 0 1 0 0 … 0 1 0 0…}, the maximum ratio value is $R = M$. When this is not the case, the activation rate is not uniform or it contains bursts, we cannot ensure available processing slots and thus the resampler should not be used for up-sampling.

There are specific cases with this still being feasible; take for instance two up-samplers in series and a data-path with $M = 4$ at the input of the segment resulting from a *MERCEDES Decouple* interface as depicted in Fig. 4.20. In the input of the first resampler we have the uniform distribution UP_4 : {1 0 0 0 1 0 0 0 1 0 …} ensured by the *MERCEDES* interface. We first up-sample by $R_1 = 2$ in the first stage; with

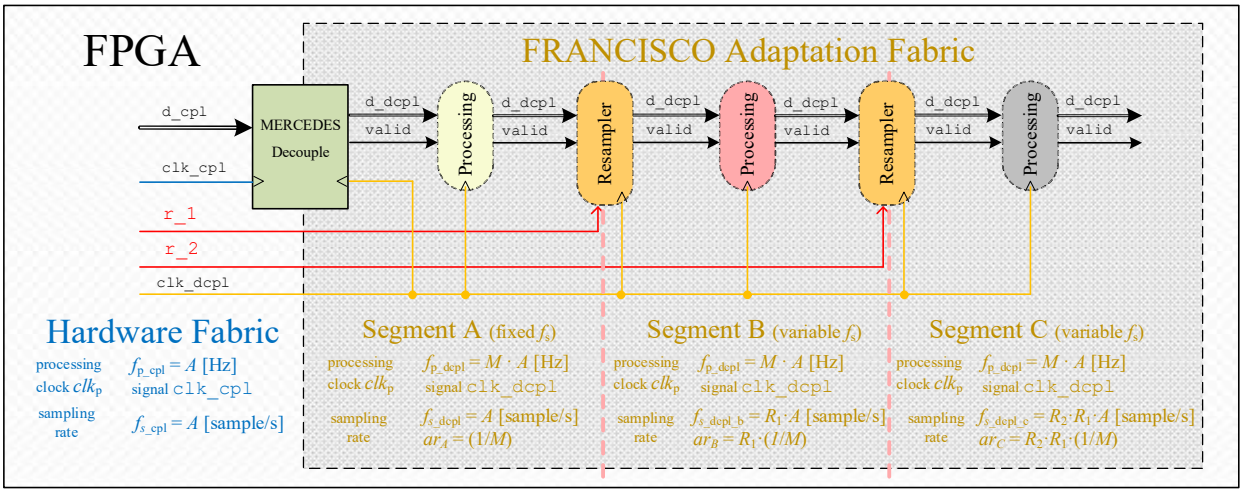


Fig. 4.20. Data-path architecture with multiple resamplers.

$f_{p\_dcpl} = 4 \cdot A$ Hz and $f_{s\_dcpl} = A$ sample/s this results in $f_{s\_dcpl\_b} = 2 \cdot A$ sample/s. The output distribution of this first resampler will be {1 1 0 0 1 1 0 0 1 1 …}. Some processing after the first resampler rearranges the data-path pattern without altering the sampling rate resulting in {1 0 1 0 1 0 1 0 1 0 …}. In this situation we could still place a second up-sampler with again a feasible maximum up-sampling ratio of $R_2 = 2$. Its input distribution is equispaced and has a pattern that makes it possible to insert the new second output sample per input sample without overflowing the maximum sampling rate $f_{s\_dcpl\_c} = 4 \cdot A$ sample/s of the data-path. The resulting pattern would be {1 1 1 1 1 1 1 1 1 1 …}. We can generalize the result; for a given up-sampling ratio $R = U$ with $U \leq M$ being M the relation between processing and sampling clocks in the data-path, we need at least $v = \text{ceil}(U) - 1$ void processing slots in between input samples to ensure the correct operation of the resampler. The ceil(X) operator results the least integer greater than or equal to X.

Table 4.1 lists the possible modes of operation for the resampler based on the pattern of valid samples at the input and for a relation between clocks M.

Table. 4.1 Modes of operation for the SSRC architecture

Mode	Pattern of valid samples at the input	Is the configuration accepted?	Accepted ratio R
Down-sampling	Any	*Accepted*	$R \in (0, 1)$
Up-sampling	Periodic, valid sample following by (M-1) void slots	*Accepted*	$R \in (1, M]$
Up-sampling	Others	*Not accepted* *	-
Transparent	Any	*Accepted*	$R = 1$

* As introduced in the section in some situations this configuration is also valid, however, it is not desirable as it requires further logic and/or constraints in the valid distribution or resampling ratio.

To cope with burst resulting in a non-uniform input pattern a feasible solution is to use a FIFO memory in the input of the resampler. We can ensure the required valid pattern by first writing the input in the FIFO and then reading the samples inserting void processing slots in the middle. However, for our BSP Architecture this is not needed as we use two resamplers in a sandwich configuration where the input one is placed after a *MERCEDES* interface, thus ensuring the necessary input pattern, and the output resampler operates in down-sampling configuration, hence with no constraints on the received valid pattern.

4.4.3.2. Engine implementation

The proposed implementation for the *DIANA* functional unit, depicted in Fig. 4.21, has three input ports. The first one is the valid signal, `valid_i`, indicating the validity of the data present in the engine input data-path. This indicates a new input sample arriving at the resampler and corresponds to the variable *new_spl* in the algorithm of Fig. 4.5. The second input port is a bus signal, `T_out_n` which provides the information about the resampling ratio, equal to $1/R$ as presented in 4.3.1.2. It maps the variable *dly_incr* in the algorithm. The width of this bus is dependent on the performance (desired precision) of the resampler, this is elaborated in Chapter 5. Finally, a clock input port receives the processing clock.

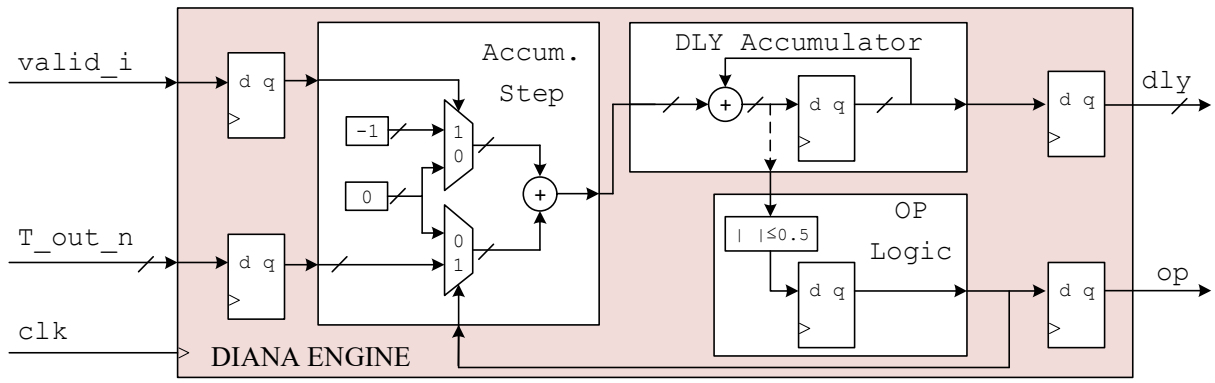


Fig. 4.21. Implementation of the *DIANA* engine.

At the output, the entity has two ports. The first is the computed delay, `dly`, implemented by means of a bus signal whose width is also dependent on the desired precision of the resampler. It maps the variable with same name, *dly* in the algorithm. The second port contains the `op` signal, which indicates that a new output sample can be computed using the delay value present in the delay bus. It corresponds to the variable *valid_out* in the algorithm. The *DIANA* algorithm of Fig. 4.5 is implemented by means of an accumulator (Fig. 4.21). Two multiplexers evaluate the accumulator step at each iteration. One is controlled by the `valid_i` signal, while the other is driven by feedback logic evaluating the `op` signal from the last iteration. The output of the multiplexor is accumulated by a dedicated DSP block.

The `op` signal is activated when a comparator evaluating the magnitude of the accumulator indicates a value less than or equal to 0.5 (half an input sampling period). This corresponds to the hard-coded variable *vfd_range* = 0.5. The `T_out_n` signal is evaluated at each iteration, for real-time variable and arbitrary ratio operation of the resampler. In case of a fixed ratio resampler, this signal could be hard-wired for resource optimization. The registers do not incorporate enable signal, because the algorithm runs an iteration with each processing clock cycle.

This engine implementing the algorithm exploits the decoupled data-path paradigm. It populates, in the case of an up-sampler, the void processing slots with new computed samples. For that to happen, the engine needs to run and evaluate the data-path at the processing clock rate, and not at the sampling rate. The algorithm can thus handle the delay computation as long as the processing clock frequency f_p is higher than both input and output sampling clocks (f_s and f'_s), and as we have seen in the previous point, the input distribution in the data-path contains available processing slots between incoming samples where to insert new output data.

As an example, take a case where the *MERCEDES Couple* interface is configure with $M = 2$, the processing clock operates at the double of the frequency of the sampling clock. A resampler is placed just after the interface. In that case the upper boundary for the resampling ratios is set by the data-path and the clock relation M resulting from the interface, being $R = M = 2$. We saw that there is no lower boundary for the resampling ratio. We nevertheless constrain it to $R = 0.5$ in our BSP Architecture. The resampling ratio can hence adopt values within the range $R \in [0.5, 2]$.

4.4.4. Synchronization

This block contains glue logic and synchronization memories connecting the main entities of the architecture. Synchronization is understood as alignment, at processing-clock cycle level, between different propagating signals. This entity is therefore responsible for the management of the signals between the different functional blocks. It comprises registers, memories, and logic elements used for synchronization purposes.

4.4.4.1. The signal paths

All these signals propagate through three different paths, depicted schematically in Fig. 4.22. These paths cross the different functional units of the architecture. The first path, in blue, hosts the data-path from the input to the output interface. The second, depicted in red, contains the signals related to the resampling ratio. This ratio is fed at the input as `T_out_n`. That signal is internally translated to delay to be used in the VFD. Finally, the third path, in green, groups the internal control signals. The alignment between these different signal types is required at certain interfaces. These interfaces lie between the main entities of the resampler. The interfaces are depicted as vertical lines in the same figure.

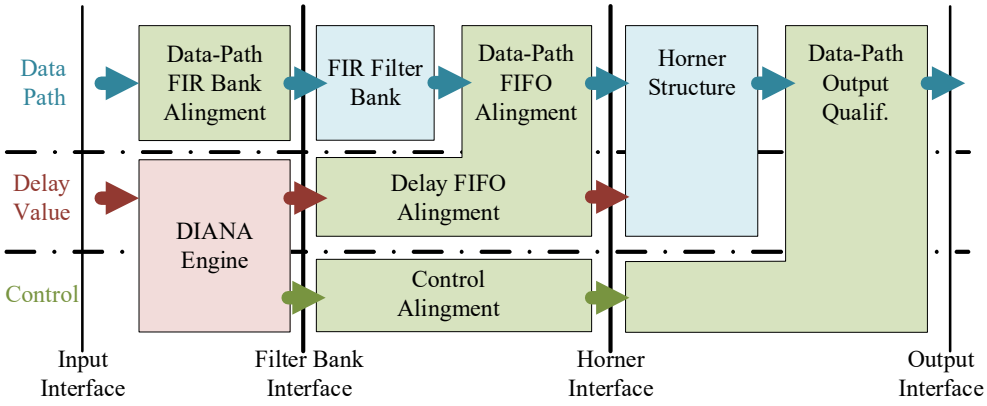


Fig. 4.22. Synchronization interfaces of the resampler (vertical lines), and signal paths (horizontal arrows).

4.4.4.2. The interfaces

The interfaces are hardware locations where the signals crossing the interface need to be aligned at clock cycle level. These are the results of the concurrent operation on data and control signals in the architecture. The signals follow therefore the different paths with different latencies. The first interface is placed at the input of the resampler. There the data-path signals `data_i` and `valid_i`, and the resampling ratio `T_out_n` must arrive aligned. The resampler operates at sample level, and the ratio signal specifies this quantity for each specific sample in the input data-path.

The second interface lies at the input of the bank of filters. At that point alignment is required between the data-path (signals `data_ips` and `valid_ips`), the computed delay signal `dly`, and the `op` control signal.

The third interface lies at the input of the Horner structure. This interface requires alignment between the data-path buses `data_fs`, the delay value `dly_s`, and the internal control signals. The buses `data_fs` are the filtered output of each of the bank of filters branches.

Finally, the fourth interface, is placed at the output of the resampler and it requires alignment between the data-path `data_os`, and the `valid_os` signal qualifying the output stream.

4.4.4.3. Input interface alignment

The input interface poses a constraint to the system hosting the resampler rather than to the resampler itself. It requires that the input signals arriving to the resampler and the clock are all aligned in phase and frequency.

4.4.4.4. Filter Bank interface alignment

The filter bank interface is used to align the computed delay to be applied at each sample, and the sample itself in the data-path. The synchronization problem is a trivial problem in this case, as the logic used for the computation of the delay is fully deterministic. This unit has been designed with a latency of three cycles, thus three registers are inserted in the data-path between `data_i` and `data_ips`, and between `valid_i` and `valid_ips`. Therefore, only glue logic (registers) is used to align the data-path to the output signals of the *DIANA* engine. The different data-path, delay and control path signals arriving to the Filter Bank Interface are depicted in Fig. 4.23.

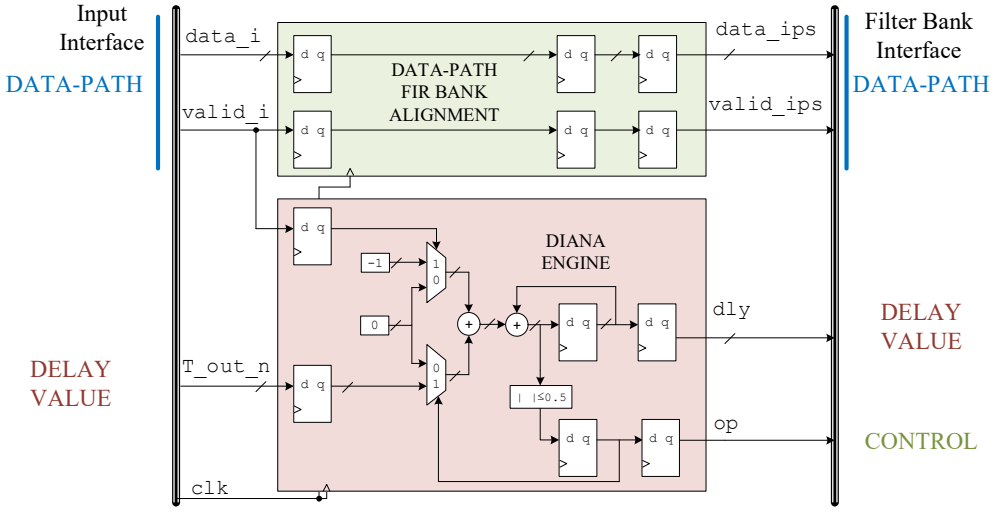


Fig. 4.23. Hardware and signal propagation arriving to the Filter Bank Interface.

4.4.4.5. Horner interface alignment

The synchronization at the Horner interface, lying at the input of the Horner combiner, is the most complex element of the architecture. It ensures the synchronized arrival to the combiner of the signal pair composed of the filtered reference sample, `data_fs` signal buses, and the delay value, `dly_s` signal in Fig. 4.16. This pair of signals is used by the Horner when computing a new output sample. The synchronization mechanism also aligns the signal `op_s` for the qualification logic at the output.

The interface and the preceding hardware elements are depicted in Fig. 4.24. The synchronization mechanism needs hence to manage signals propagating in parallel within the three paths; data-path, delay and control. The segment of the data-path preceding the interface contains the VFD filter bank. The filtering that it performs has a variable latency when implemented within the *FRANCISCO* fabric. This makes the synchronized propagation of reference samples and delay a non-trivial problem.

In the case of delay and control paths, the filter bank interface at the output of the previous stage provides the delay value `dly`, and the control signal `op` aligned at processing-clock cycle level. These signals have been generated by the *DIANA* engine. They contain the decision about the computation of a new output sample (algorithm in Fig. 4.5); if possible, the `op` signal is active, otherwise it remains low.

In the same preceding interface, the incoming data-path (`data_ips` and `valid_ips` signals) is also aligned at clock cycle level with these delay and control paths. The latter signal, `valid_ips`, flags when active that the data-path reference sample at the filter bank interface needs to be processed (inserted in the bank and filtered), otherwise the data-path remains halted.

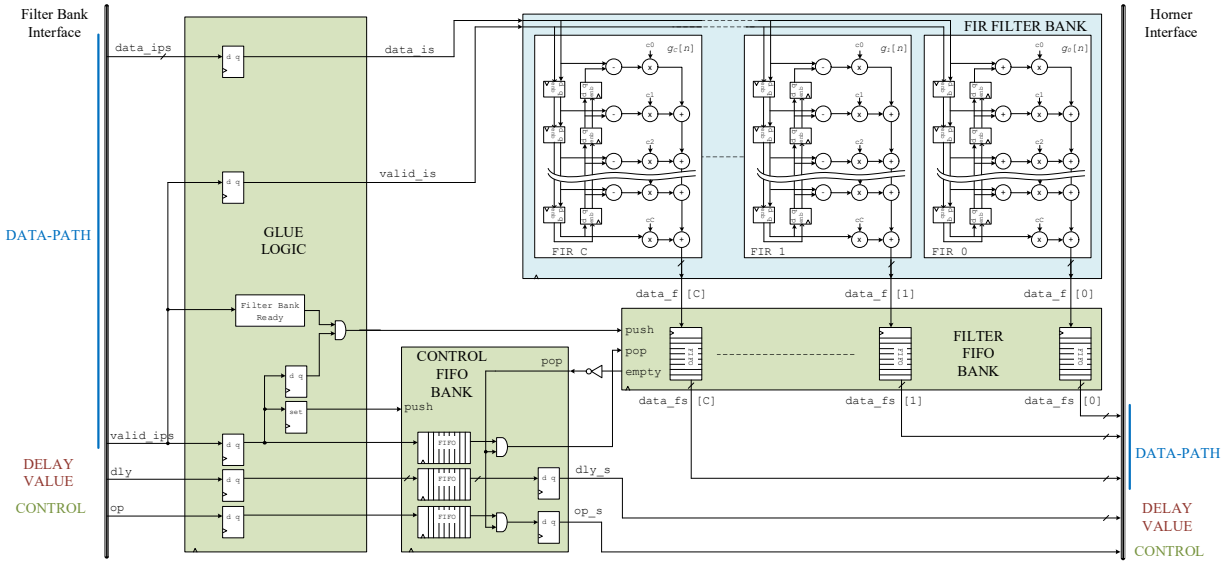


Fig. 4.24. Hardware and signal propagation arriving to the Horner Interface.

As we have just introduced, when a new output needs to be computed, and a new sample needs also to be filtered, the synchronization mechanism needs to propagate and recover the alignment of the pair `data_ips` and `dly` at the input of the Horner combiner (`data_fs` and `dly_s`). It can also be the case that a new output needs to be computed but no new reference sample is available in the data-path ($R > 1$ *with* `valid_ips` *low in the filter bank interface*). In these circumstances the last filtered input sample is used as reference. This filtered sample is already processed or being processed in the filter pipeline. In any case, the delay value `dly` that will be used in the new output needs to have synchronized its arrival at the input of the Horner structure together with its associated reference sample being filtered. The synchronization mechanism hence handles the filtering of the data-path and the arrival of the sample and the delay value to the Horner based on the control signals. The computation of a new sample requires therefore different mechanisms depending on the case. The possible scenarios are listed in Table 4.2.

Table. 4.2 Scenarios, control signals and actions to be done within the synchronization logic

Scenario	valid_ips	op	Actions of the VFD
1	0	0	Do nothing
2	0	1	Process new output
3	1	0	Propagate data-path to filter bank
4	1	1	Propagate data-path & process new output

In the first scenario nothing has to be done. The delay value to be applied to a sample for the computation of a new output cannot be handled by the VFD, therefore the `op` signal is not active. Concurrently there is no new sample arriving to the filter bank interface as the `valid_ips` signal is low. No new sample needs to be filtered.

Scenario two happens when a new output needs to be computed but there is no new sample arriving to the filter bank. In this case the delay value is within the limits of the VFD, and the last sample inserted in the filter bank pipeline needs to be reused as reference. The delay propagation needs hence to be synchronized with the filter bank output; the reference sample in the bank needs to be used twice.

Scenario three is present when a new sample arrives at the filter bank and the delay required for a new output cannot be handled by the VFD. In this case the sample is inserted in the filter pipeline but no new output is computed.

Finally, scenario four combines scenarios two and three. In that case a new sample has arrived at the filter bank. This sample needs to be filtered and propagated through the pipeline. Concurrently, a new output needs also to be computed with that sample; the *DIANA* engine has computed a delay value using that sample as reference that results within the VFD range. The alignment mechanism needs therefore to mimic the latency of the reference sample through the filter bank pipeline, and synchronize the propagation of the delay value with the same latency, to make it both to cross the Horner interface together.

The estimation of this latency through a filtering architecture implemented with a coupled data-path is a trivial problem. The signal advances one register in each clock cycle, as depicted in Fig. 4.25(a). In this case, for instance implementing the filter in the FPGA hardware fabric hosting BAP, the latency is a function of the architectural implementation of the filter, Eq.(4.22). This makes the solution for the synchronization of the delay signal easy; the filtering latency can be mimicked by adding the same fixed number of pipeline registers to the delay path.

This is not feasible for our filtering architecture, as we use a *FRANCISCO* decoupled data-path. In such a case, the filtering latency varies with the sampling rate and input pattern. This variation is a direct consequence of the decoupling and the valid signal: The data advances one register further in the filter's tapped delay line only when the enables are active (`valid` signal in Fig. 4.25(b)). The data-path propagation through the taps is now driven by the `valid` signal, and the latency (in processing clock cycles) is dependent on its activation rate and distribution. In this case, the filtering latency for a given processing clock, is hence a function of the sampling rate, that dictates the activation ratio *ar* of the `valid`

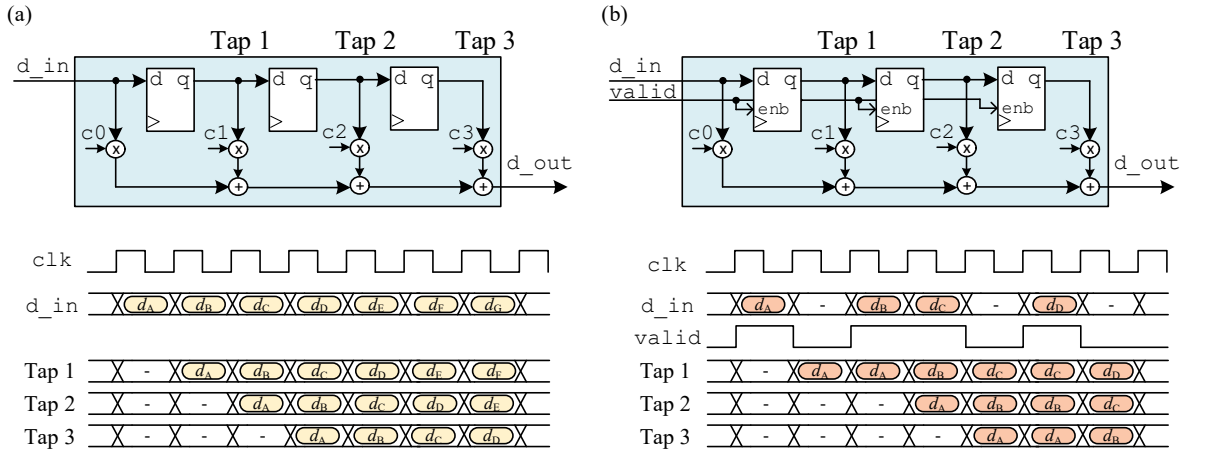


Fig. 4.25. Propagation through the tapped delay line of a filter with (a) a non-decoupled and (b) a decoupled data-path.

signal as stated in Eq.(3.2). If the sampling rate in the decoupled data-path is uniform, the propagation delay can still be mimicked with the same approach as before, adding registers to the delay path. As our architecture needs to handle different scenarios (a resampler accepting different input rates that result in different filtering latencies) and must be generic, the delay synchronization problem cannot be solved anymore by adding a fixed number of pipeline registers to the delay path.

In the filter bank, the data advance through the taps is hence based on a "`valid` driven propagation". Each delay value is computed and associated to its reference sample in the *DIANA* engine, arriving aligned to filter bank interface. Note now that the filtering of one sample requires several subsequent samples marked valid, the number depends on the tap length of the filter. With each valid sample, the data samples in the data-path advance one tap register in the filter, as depicted in Fig. 4.25(b). The process starts for a given reference sample with the active `valid` signal accompanying it; that `valid` signal triggers the filtering by inserting the sample in the first tap of the filter data-path. Then, the upcoming samples with the `valid` signal active, drive the sample through the filter. The new arriving samples are inserted in the pipeline and at the same time, the insertion propagates the precedent samples through the successive taps. A filtered sample is hence popped out of the filter only when a sufficient number of valid samples propagates it from the input of the filter to the central tap (we neglect here the latency and/or pipelining in the arithmetic hardware elements from that middle tap to the output of the filter).

To cope with the propagation based on upcoming samples, and the variable filtering latency, the solution adopted is to use FIFO memories for synchronization of the different paths. Two banks are used: CONTROL FIFO BANK and FILTER FIFO BANK, depicted in Fig. 4.24 in green. The first stores the delay value `dly`, the `op` and the `valid_ips` control signals in the filter bank interface, the second stores the outputs of the filter bank `data_f` at the output of the filters. By managing the reading of these banks, the synchronization mechanism can handle the aligned arrival of the filtered sample-delay pair at the Horner interface.

The first bank, CONTROL FIFO BANK (control signals), compensates (mimics) the propagation latency of the data-path through the filter bank pipeline. It acts handling the propagation latency of the

delay signal and control signals (current scenario) between the output of the *DIANA* engine (filter bank interface), and the input of the Horner structure (Horner interface). The scenario and delay are pushed in this bank, once the first valid data sample arrives at the filter bank interface. This associated reference sample is filtered in the parallel data-path and stored in the second FILTER FIFO BANK once ready. The filling level of this second memory bank is monitored by the synchronization mechanism. With this information this mechanism pops and validates that data out of the CONTROL FIFO BANK as long as the second bank is not empty.

The second memory bank, FILTER FIFO BANK in the figure, stores the valid processed outputs of the filter bank. This makes the controlled extraction of filtered samples from the memory possible, reusing them when needed. The `Filter Bank Ready` block latches an active signal after initialization of the data-path pipeline in the filters; the taps of the filter bank are populated with valid data after the start of the system. After the latching of that signal, data is pushed into the FIFO bank when new valid samples arrive (signal `valid_ips` active). The bank pops out filtered data, signals `data_fs`, when the `pop` control signal becomes active. This signal, that is a synchronized version of the valid flag in the input data-path, is extracted from the CONTROL FIFO BANK memory. It reproduces together with the delay value and the `op` signal the input scenario in the filter bank interface.

Although in our sandwich configuration we will not have bursts arriving to a resampler configured in up-sampling mode, our synchronization mechanism can handle also such a case. As presented in 4.3.1, a same sample will be used as reference for several output values in up-sampling configurations. When a burst arrives (successive input samples marked valid arriving at the data-path input in the filter bank), the filter produces consecutive outputs in consecutive processing slots; this pops several filtered samples out of the filter based on the consecutive `valid` of the burst. These bursts at the output need to be handled properly by the synchronization mechanism, otherwise corruption may occur. This is the case if the first popped sampled of the burst was used in the *DIANA* engine as reference in two delay values; it needs to be re-used twice in the Horner structure, to produce two output samples, with the two delay values (scenario 4 followed by scenario 2 at the filter bank interface). Imagine that a burst arrives at the input of the filter in such a circumstance, the precedent same sample needs to remain two consecutive cycles at the output of the filter. The filter will pop a second sample, making the needed one to remain only for one single slot at the output port. Without our memory-based mechanism this would violate the delay-sample combination, since the sample would only be available for the first of the two processing slots required. However, with the controlled extraction of the memories present in our mechanism, this would be safely handled.

Combining hence the operation of the two memory banks, the scenarios can be safely passed from the filter bank interface to Horner interface. The delay and filtered sample signals are thus fed aligned to the Horner structure, and a new output can be produced.

4.4.4.6. *Output interface alignment*

The propagation between Horner interface and output interface, through the Horner structure, is deterministic. The Horner entity is composed of adders and multipliers which combine and propagate the filtered data with the delay value. Only arithmetic operators implemented in DSP macros are present, and no enable register is used in this data-path pipeline. The latency through them can thus be estimated and mimicked with a chain of registers in the `op_s` line: This signal is propagated in parallel to the data-path with pipeline registers reproducing the delay. At the output of the Horner structure, the synchronized `op_s` signal qualifies the result of the resampler (signals `data_os` and `valid_os`). These elements are depicted in Fig. 4.26.

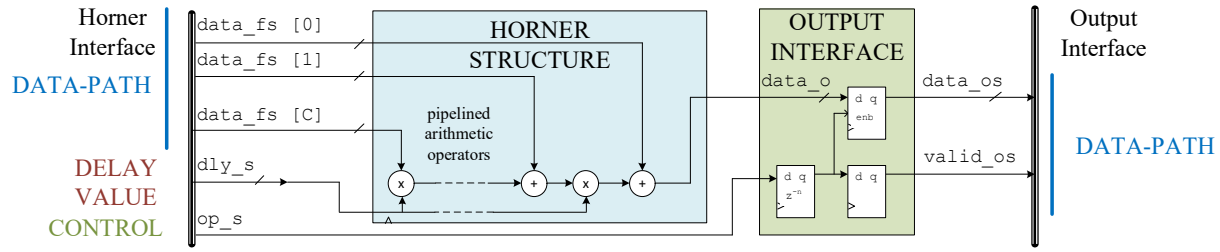


Fig. 4.26. Hardware and signal propagation arriving to the Output interface.

4.5. Conclusions

The chapter has presented a new all-digital sampling rate conversion architecture with a fixed frequency system/processing clock. The output data port is synchronous to the input that also serves as reference. The resampling ratio is externally provided. The architecture supports both up-sampling and down-sampling. The resampler contains an interpolating unit based on a Variable Fractional Delay filter and timing unit based on an algorithmic engine implementing the *DIANA* algorithm. The range of values for this resampling ratio is defined by the relation between sampling clock and processing clock in the decoupled data-path in which it is included, and the position of the resampler in the data-path. It uses the *FRANCISCO* fabric presented in Chapter 3. The ratio can adopt arbitrary values within the range and accepts its modification in real-time.

Chapter 5

Verification and Validation of the Resampler and the BSP Architecture

Abstract: *This Chapter presents the results of the functional verification and validation of the main units of the proposed solution. The entities of the resampler, DIANA engine, VFD filter and Synchronization logic are first analysed. They are later studied combined within a resampler. Finally, the BSP Architecture containing two resamplers and application specific processing is tested. The results of Hardware tests in a laboratory setup are also presented.*

5.1. Introduction

This chapter presents the functional verification of the resampler, its three functional units and the combination of two resamplers in a sandwich configuration originating the BSP Architecture. The chapter presents also the validation results of the tests that we have performed in the units affecting the final performance of the BSP Architecture.

In the verification process we have checked the operation and behaviour of the resampler, its units, and the BSP Architecture against the functional specification that we have introduced in Chapters 3 and 4. The validation procedures assessed the performance of these entities; we look at the key elements and the implications they have, that affect the quality of the results of the BSP Architecture when in operation. This analysis is closely related to the implementation details of the Architecture; it is defined by parameters such as the number of bits used in the data-path, the coefficients of the filter bank, or the delay/ratio data-path, the number of taps in the filters, the architecture of the VFD, etc. They all influence the final performance of the BSP Architecture. The optimization of all these parameters is out of the scope of the Thesis; we however present some guidelines that should be taken into account by the reader as an illustration on how all these trade-offs affect the performance when tailoring the Architecture to other implementations.

The methodology that we have followed in the development, implementation, verification and validation procedures is as follows. We have first studied the resampling algorithm with MATLAB simulations, regardless of any implementation details. Then a functional Simulink model was used for architectural exploration and its verification. The algorithmic operations and control structures were grouped by functionalities, and translated into system level blocks. These were integrated into a functional system level model of the resampler. Then two resamplers in "*sandwich configuration*" were connected to study the BSP Architecture. These simulations included the latency of the hardware blocks mapping the data-path and the control structures. No signal quantization constraints were incorporated at this stage, the only error source came from the coefficients of the VFD that approximates the response of a pure delay. This functional model was migrated to Xilinx System Generator primitives within our hardware implementation for verification and validation with simulations and also in real hardware in the laboratory. These hardware primitives and the System Generator suite make possible the simulation of the Architecture and its internal signals with an accuracy of a processing clock cycle. We run several simulations with test benches mimicking real scenarios and applications of the resampler and the BSP Architecture. These verified the feasibility of the solution. Finally, the resampler and the BSP Architecture have been migrated to a uTCA crate for test in the laboratory and real hardware.

5.2. Verification

The section presents the verification of the BSP Architecture against the functional specification. We first briefly present the procedure that we have followed to verify the resampler entities and the resampler as a whole. Then we concentrate on the BSP Architecture verification; its results implicitly double-confirm that the verification was also successful for the internal entities and single resampler. We use in all the verification and validation processes the same clocking configuration for the *FRANCISCO* fabric and the *MERCEDES* interfaces; the data stream at the input of the decouple interface arrives at sampling rate $f_s = 62.5 \cdot 10^6$ sample/s, we configure the interfaces with $M = 2$ (section 3.3.7), this results in a processing clock in the decoupled data-path at a frequency of 125 MHz. These values are also the ones that will be used in the OTFB application tests of Chapter 6 (the recovered WR clock is also a signal at 125 MHz).

5.2.1. Entities and resampler verification

We present in first place the verification procedure for the *DIANA* engine, the synchronization logic and the VFD. We have used System Generator instead of the Very high-speed integrated circuit Hardware Description Language (VHDL) for the hardware implementation of the resampler and BSP Architecture, this eases the design and also the verification process of a DSP hardware architecture. This graphical language instantiates hardware primitives present in Xilinx FPGAs and accepts easily parametric configuration of the hardware elements; we can play with different architectural configurations and, at the same time, System Generator makes possible computer simulation of the synthetized final hardware in the time domain. We prepared different test benches for the verified entities where we instantiated the System

Generator models of the hardware entities and excited them with signals generated in a computer workspace.

In the case of the *DIANA* engine verification, we have simulated the System Generator implementation of the entity depicted in Fig. 4.21 against the algorithmic specification (section 4.3.1.2). We studied different input stimulus; the different sampling rates in the input data-path of the resampler are defined by the valid signal. We have simulated different activation ratios and different distributions in the tests. The resampling ratio R (mapped as $1/R$ in the signal `T_out_n` as presented in Eq.(4.8)) also reproduces different scenarios; we performed tests with the ratio as a static signal, and dynamic scenarios where the ratio changed with time. The functional behaviour of the resampler is also controlled with this signal; we tested scenarios where it was configured in up-sampling mode and others were the resampler was configured in down-sampling mode. We analysed the correct generation and synchronization of the output signals `op` and `dly` according to the algorithm diagram of Fig. 4.5.

The VFD interpolating unit of the resampler was the next entity verified. It estimates the value of an analog subjacent signal at the requested instants by shifting (filtering) the available samples. These new sampling instants are defined by an amount of time (the delay) with respect to the reference handled by the *DIANA*. We have verified that our VFD satisfies the delay ranges (section 4.4.2.1) and the dynamic modification of its value. We performed this preliminary verification in the time domain, and we let the frequency-domain analysis for the validation study. The data-path of the VFD is implemented in the *FRANCISCO* fabric and uses hence a decoupled data-path. In the verification we do not focus on the data-path implementation issues that are tackled when verifying the synchronization logic between different elements of the resampler, among them the filter bank and the Horner. Moreover, from a functional point of view the use of a coupled or decoupled data-path is transparent.

We have performed simple tests on the VFD that had as objective the verification of the correct signal shift or "*estimation*", both for "*past*" and "*future*" sampling instants with respect to the reference, as presented in 4.3.1.1; the "delay by τ s" operation was modelled in the continuous-time in Eq.(4.10) as the impulse response $h_{id}(t)$. This impulse response is a shift operation of magnitude $|\tau|$ s, being τ negative valued. The discrete counterpart $h[n]$ was presented in Eq.(4.18), where the total delay D measured in samples (Eq.(4.20)) contains an integer part D_i that accounts for the latency of the filter and the fractional one d. We were interested in this fractional contribution; it defines towards what time direction and how much we shift our reference to obtain the estimated output sample. Note that as the contribution of the integer part (the hardware latency) is always larger than the fractional contribution, the total magnitude of the delay will always be $D < 0$. When d is also negative, $d < 0$, the VFD estimates the value of the signal in past sampling instants (looking *"backwards"*) with respect to the reference $x[n]$, that lies in the central tap of the filter. When the fractional delay value is positive, the VFD estimates the value of the signal in future sampling instants (looking *"forward"*) with respect to the reference $x[n]$. The test to verify this correct behaviour for positive and negative values fed fixed negative ($-d$) and positive (d) fractional delay values

to the VFD. Then the resulting delayed and advanced signals were post-processed; the latency D_i of the VFD was removed and the resulting signal aligned with the reference one. The VFD successfully passed all the tests. We performed more complex tests with the delay value modified in real-time. Note that the filter does not need any reconfiguration time, nor flushing of the internal pipeline and/or taps when changing the delay value. The Farrow-based VFD does not require re-computation of the filter coefficients as a function of the delay as presented in 4.3.2.5, instead it prefilters the input samples $x[n]$ in the parallel static branches that contain the sub-filters $g_c[n]$ of the filter bank. This results in a parallel set of pre-filtered data based on the single input of the VFD. These pre-filtered data are the coefficients $\bar{u}_c[n]$ (Eq.(4.30)) of the Taylor polynomial (Eq.(4.32)), with the delay value d as the variable in this case. We were able to change the delay value in real-time and obtain the estimate $y[n]$ of the signal at the sampling instant pointed by this delay value.

The glue logic and synchronization elements connecting the main entities of the architecture have also been verified. The process checked that the signals crossing the interfaces presented in section 4.4.4 arrive aligned at processing-clock cycle level to the following interface. The entities are the *DIANA* engine, and the VFD filter that contains the FIR filter bank and the Horner combiner. The *DIANA* engine requires alignment between data-path and resampling ratio. In the VFD, the filter bank interface requires alignment between the data-path and the delay value resulting from the engine. The Horner, that combines the processed data-path and the delay, requires also alignment among them, and with the control signal op of the engine, that flags when these signals will produce a valid output sample. The synchronization mechanism performed correctly.

After verification of the entities as isolated elements, we studied the entire resampler. We resampled different input signals and verified the output to properly represent the underlaying analog signal of the input. The ratio was configured in up-sampling mode, in down-sampling mode and to perform no operation, being transparent, i.e., $R = 1$. We also checked that the resampler operates as expected when modifying its input resampling ratio signal in real-time. We depict in Fig. 5.1 a summary plot of one of these tests with varying ratio in real-time, that implicitly also verifies the simpler cases. We used as input a sinusoidal signal at a frequency of 1.5 kHz. The ratio starts adopting a down-sampling value $R = 0.5$, then is modified towards up-sampling values according to $R = 0.7$, $R = 0.9$, $R = 1.1$, $R = 1.3$ and $R = 1.5$. We then decrease the resampling ratio towards down-sampling ratios; $R = 1.3$, $R = 1.1$, $R = 0.9$, $R = 0.7$ and $R = 0.5$. After each modification of the resampling ratio, the resampler operates in steady state for 0.1 ms. We have decimated the stored reference signal, the resampled signal and the ratio signal to better depict the results; we plot one sample in the figure per one-thousand-five-hundred samples in the simulation. In Fig. 5.1(a) we depict the input reference samples fed to the resampler and its subjacent analog signal. In Fig. 5.1(b) we depict the resulting signal that follows the resampling ratio, that it is also depicted in the third trace, Fig. 5.1(c). We observe at the beginning of the simulation, in the first steady period, that we have more samples in the reference trace than in the output trace. The reference trace contains six depicted samples while the resampled trace only three. This is coherent for $R = 0.5$. Then the ratio starts to grow and

around the simulation time 0.3 ms, it changes from down-sampling to up-sampling values. We see how in each segment, the number of samples in the resampled trace is larger than in the reference. Then, between 0.5 ms and 0.7 ms, the resampler operates at $R = 1.5$, we can observe how in the refence trace we have nine depicted samples while in the resampled one this value is increased up to thirteen depicted samples, this is again coherent for $R = 1.5$. We finally start to decrease the ratio, and at the end of the simulation it reaches the starting value $R = 0.5$, with again six depicted samples in the reference and three in the resampled trace. We can hence conclude that according to this functional test, the resampler is verified, and all the units and entities that integrate it and provide support for the implementation, operate as expected with static and dynamic ratios.

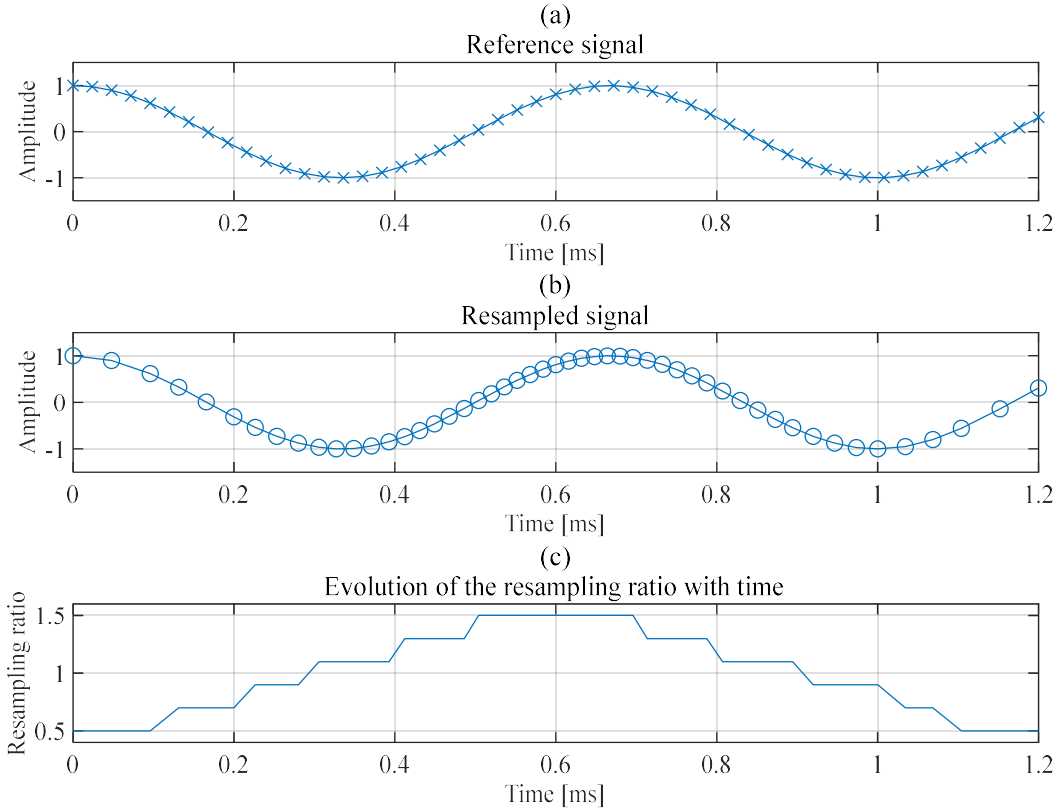


Fig. 5.1. Functional verification of the sweeping dynamic resampling ratio. (a) Reference signal. (b) Resampled signal. (c) Resampling ratio.

5.2.2. BSP Architecture verification

The verification of the BSP Architecture that we present in this point checked that the tuning between signal and BSP algorithm was achieved for spectral contents that vary with time. This requires dynamic reconfiguration of the resampling ratios in the sandwich. We assessed that our BSP Architecture effectively maps the signal spectrum to the BSP processing (filtering), and at the output recovers the original sampling rate, in a test bench emulating different scenarios found in LLRF applications.

The test bench that we used instantiates the segment of the data-path that contains the BSP Architecture implementation presented in Chapter 3. It spans between the input and output *MERCEDES* interfaces of Fig. 3.9. The data-path is decoupled before the first resampler and coupled back after the second. We used the presented clocking configuration in the introduction with $M = 2$. The valid signal

present after the *MERCEDES Decouple* interface has a uniform activation ratio *ar* = 0.5. The BSP region between the two resamplers in sandwich configuration is implemented in the *FRANCISCO* fabric. It contains the application-specific processing. The *MERCEDES Decouple* interface implements also in System Generator the proposed architecture of Fig. 3.19, and the *MERCEDES Couple* interface does the same with the architecture of Fig. 3.24. As the test bench aims to mimic LLRF applications, the BAP processing block (Fig. 3.9) in our application would be the responsible for the down-conversion of the digitized RF signal.

In the BSP region, we used for the verification a static processing filter inserted between the two resamplers. The first resampler increases the sampling rate of the discrete signal. The resampling ratio is modified in real-time proportionally to the instantaneous frequency of the signal. This operation tunes the discrete representation of the sweeping spectral components of the processed signal to fixed normalized frequencies according to Eq.(3.6). Finally, the filtered signal is brought back to the original sampling rate by the second resampler. The ratio is modified in real-time in this resampler with the inverse value of the input one. The BSP filter was also implemented in System Generator primitives. The stimulus and other auxiliary system blocks were implemented using Simulink blocks for simplicity. The tests were conducted without and with quantization of the signals to fixed-point arithmetic. We present here the quantized case; the data-paths at the input and output of the *MERCEDES* interfaces, BAP and BSP regions are sixteen bits wide, with fifteen fractional bits. The coefficients of the VFD filter within the resamplers and the BSP filter are twenty-seven bits wide, with twenty-six fractional part bits. The resampling ratio signals are thirty-two bits wide words, with twenty-nine fractional part bits, two integer part bits and one sign bit. The computed delay value is an eighteen bits wide word, with seventeen fractional bits and one sign bit.

5.2.2.1. Notching filter scenario

The use case scenario that we present in this section emulated an RF signal with harmonic and variable spectral content that needs to be processed. The stimulus signal at the input of the data-path test bench represents this harmonic signal after down-conversion to base-band: It contains a DC level and a tone at a frequency that changes with time. The DC level remains at that fixed position in the normalized spectrum during the whole simulation. The real frequency of the tone changes with time but we use the BSP Architecture to tune the sweeping tone to the processing, both at a fixed position in the normalized discrete spectrum. The signal processing is hence static, it is not reconfigured, and has a fixed frequency response addressing the harmonic content. We use for that a static periodic filter whose notches cancel the DC level and the variable frequency tone. In the test bench an extra line at a fixed static frequency is added as witness of the proper BSP behaviour. When the system performs as expected, the output of the simulation contains the witness line only.

In the simulation, at the beginning, the sweeping tone has a frequency F_{tone} = 5.375 MHz (ω_{tone} = $2\pi \cdot 0.086$ radian/sample). At the end of the simulation the tone has increased by 18.6%, its frequency reaching F_{tone} = 6.375 MHz (ω_{tone} = $2\pi \cdot 0.1020$ radian/sample). The witness tone remains at 3.375 MHz

during the whole simulation time, 500 ms. Digital frequencies are normalized to the $62.5 \cdot 10^6$ sample/s input sampling rate.

The Z Transform of the implemented notch filter within the BSP has been normalized to have unit gain outside the notches and is presented in Eq.(5.1).

$$H(z) = \frac{1+a}{2} \cdot \frac{1 - z^{-N}}{1 - a\, z^{-N}} \qquad \text{Eq.(5.1)}$$

The parameter values are a = 31/32 and N = 12. The normalized frequency response notches are located at $\omega_{\text{notch\_k}} = k \cdot (2\pi/12) = k \cdot 2\pi \cdot 0.0833$ radian/sample. There are 12 notches ($k \in [0, 11]$) from DC to the resampling frequency. Fig. 5.2 depicts the magnitude response of the filter with respect to the normalized BSP sampling frequency f'_s. The filter (encapsulated within the resampler sandwich) operates only on valid samples in the data-path. These valid samples are the data samples with the valid flag asserted. The filter thus processes samples in the BSP region of the sandwich at a rate f'_s and operates with a system clock at f_p. After tuning the spectral content of the signal to the filter frequency response (up-sampling with the first resampler), the DC notch ($k = 0$) cancels the DC component, and the first notch ($k = 1$) cancels the sweeping tone. The witness line should remain unaltered as it does not match any of the filtered frequencies.

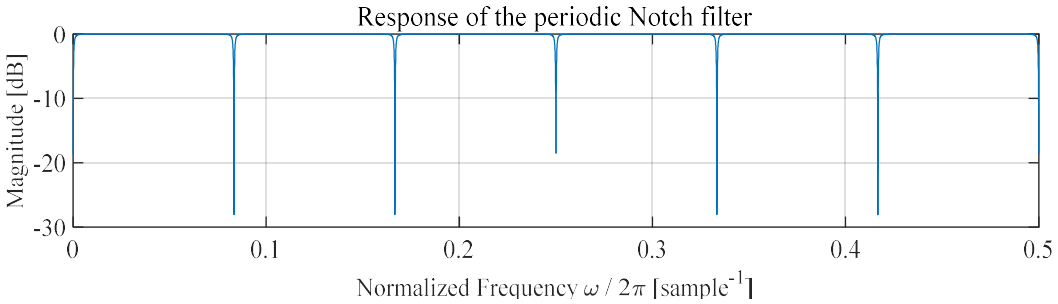


Fig. 5.2. Magnitude of the frequency response of the periodic notch filter normalized to the BSP sampling frequency f'_s.

The ratio of the input resampler tracks in real-time the instantaneous frequency F_{tone} of the sweeping tone. This resampling ratio is computed based on this known instantaneous sweeping frequency of the signal, according to the presented procedure in section 3.3.9. As a computation example, take the start of the simulation; the normalized frequency of the first peak in the notch filter above DC is

$$\omega_{\text{notch\_1}} = 1 \cdot (2\pi / 12) \text{ [radian/sample]} \qquad \text{Eq.(5.2)}$$

It corresponds to the processing frequency to which we want to tune the signal. The resampling ratio according to Eq.(3.8) is therefore

$$R_{\text{init}} = F_{\text{tone\_init}} / (f_{\text{notch\_1}} \cdot f_s) = 5.375 \cdot 10^6 / \left((1/12) \cdot 62.5 \cdot 10^6\right) = 1.032 \qquad \text{Eq.(5.3)}$$

At the end of the simulation, the resampling ratio is now

$$R_{\text{end}} = F_{\text{tone\_end}} / (f_{\text{notch\_1}} \cdot f_s) = 6.375 \cdot 10^6 / \left((1/12) \cdot 62.5 \cdot 10^6\right) = 1.224 \qquad \text{Eq.(5.4)}$$

The ratio of the output resampler is just the inverse ratio of the input.

The test was successfully completed; Fig. 5.3 shows the spectrogram of the input stimulus signal (a) and the output processed signal (b). The x axis shows the normalized frequency contents of the signal spectrum, the y axis shows the simulation time. In Fig. 5.3(a) one clearly identifies the DC component, the witness line at 0.054 sample^{-1} and the sweeping tone changing with time during the simulation. All lines have similar magnitudes in the order of 70 decibel (dB) above the noise floor. Then in Fig. 5.3(b) after filtering the input signal within the BSP region, the DC component of the resulting output signal is removed. The sweeping tone has been also tracked and cancelled by the notch filter, with its magnitude reduced down to the noise floor value. The witness line is not affected by the filtering as it is not tuned to any notch of the filter. The test hence verifies that the BSP Architecture operates as expected, from a functional point of view. It tunes the sweeping spectrum of the signal to the BSP processing, by modifying in real-time the sampling rate of the signal.

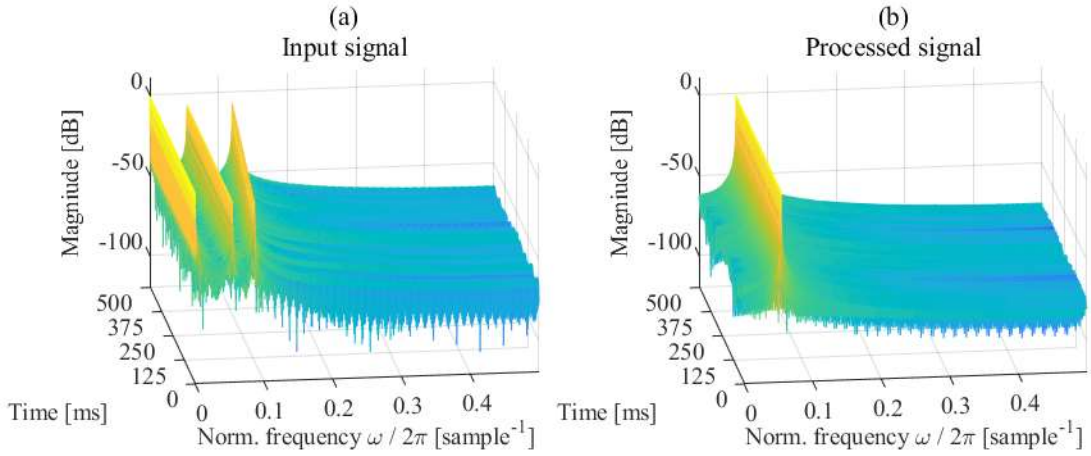


Fig. 5.3. Spectrograms at the input (a) and output (b) of the BSP Architecture.

5.3. Validation

This section presents the validation of the BSP Architecture, the resampler and the internal entities. We study how the implementation affects the resampler and BSP Architecture performance. We analyse how the finite precision of the fixed-point arithmetic signals degrades the performance.

5.3.1. Entities and resampler validation

5.3.1.1. DIANA engine validation

We start with the *DIANA* engine analysis, as it lies just after the Input interface of the resampler (section 4.4.4). Besides the monitoring of new samples arriving to the architecture, the *DIANA* does not perform arithmetic operations directly on the data-path signals (composed by `data_i` and the `valid_i` in that segment, Fig. 4.23); it only tests the state of the `valid_i` signal. The engine hence does not use or operate the `data_i` signal that contains the fixed-point sampled values; it does not directly act in the data-path processing (arithmetic operations on the `data_i` signal and the associated sampled values). Nonetheless, the engine computes the delay value (`dly` signal) based on the resampling ratio signal

`T_out_n` that it receives. Both signals, `dly` and `T_out_n`, deal with real-valued numbers and will therefore present some quantization error. The delay results from the time distance between the input reference and desired output, and it is later multiplied with the filter bank output samples in the Horner. The delay error, difference between the quantized value and its real counterpart, will hence influence the resampler performance. This error can be modelled, when looking at resampler level, as time jitter in the output sampling instant. This jitter shifts the time at which the output is estimated with respect to the real instant at which it should be. However, the delay value changes for each output sample, it does not therefore accumulate. Note that we will later present in section 5.3.1.2.1 that this jitter can also be modelled as an error in the frequency response of the VFD.

We can estimate the error and this time jitter for our fixed-point implementation. The inaccuracy, or shift in time, for a given data sample, is governed by the number of bits used in the fixed-point word representing the delay. We use rounding to deal with the truncation error; the maximum delay error will hence be in this case one half of an LSB. The objective is to dimension this fixed-point word to trade-off between hardware resources and error. The delay signal is used in the Farrow architecture to evaluate the polynomial that is composed of the different outputs of the VFD filter bank, as presented in Eq.(4.33), and depicted in Fig. 4.13. This evaluation is implemented within the Horner combiner by products between the delay and the outputs of the filter bank. The limiting factor will hence be the input width of the multipliers that deal with the delay. We use DSP blocks with hardware multipliers to increase the frequency performance of the resampler. In FPGAs, these DSP blocks usually offer a multiplicand input of eighteen bits and a second multiplicand input with a larger number of bits. For instance, the DSP48E2 block in the Xilinx Ultrascale devices contains a 18 x 27 bit hardware multiplier [105]. Our concern is now which of the two inputs of the multiplier to use to achieve better global performance within the architecture. We have two signals, the data-path and the delay, but only the first contains information directly related to the samples that represent the analog signal. It makes hence more sense to use the larger input of the multiplier with these discrete values that directly handle the samples of the analog signal. The delay will therefore be quantized to a maximum of eighteen bits in our implementation that makes use of a Xilinx FPGA.

Looking now at the time error, this delay value is normalized to the input sampling period T_s (Eq.(4.6)); a delay value `dly` = 1 corresponds to one input period. The maximum delay value accepted by the VFD is plus or minus half of a sampling period; the maximum delay magnitude that a valid sample will use is therefore |`dly`| = 0.5. We can use all the bits in the discrete word, besides the sign bit, to represent this fractional valued delay. If we use Xilinx FPGAs with DSP48E2 blocks, the word width will result in a fixed-point signal with eighteen bits, one sign bit plus seventeen fractional bits. The magnitude of the maximum quantization error (half of an LSB) results hence in this case in $3.8147 \cdot 10^{-6}$ sampling periods.

It is possible to translate this value in sampling periods to time jitter for any clock frequency; if we use a sampling clock at the input of the resampler with a sampling rate of $62.5 \cdot 10^6$ sample/s (16 ns sampling period), the error in sampling periods results in that case a time jitter of $6.1035 \cdot 10^{-14}$ seconds. The effect

that this instantaneous time jitter has in the computed sampling instant is hence negligible when compared to the delay error that results from the frequency response of the VFD filter (section 5.3.1.2.2). The latter approximates an ideal delay with a certain precision and the magnitude of the error in this approximation is larger than the error that the time jitter will induce. In any case, in our final application (Chapter 6) we limit the regulation bandwidth to 5 MHz per sideband around the RF. If we take for instance a tone at a frequency of 3 MHz, this time jitter will create a very small and negligible phase error of $6.5918 \cdot 10^{-5}$ degrees for such a signal.

There is a second error source in the *DIANA* engine related also to the quantization process. We have already presented in section 3.3.6 the problem that discrete systems present when dealing with fixed-point implementations of the resampling ratio signals in our BSP Architecture; the quantization errors in the ratio (signal *T_out_n* in our resampler implementation as presented in section 4.3.1.2 and Eq.(4.8)) result in a slightly different output sampling frequency in the resampler.

Take the case of an ideal resampler whose ratio signal `T_out_n` is not quantized; in this case the output sampling frequency matches the requested value $f'_s = R \cdot f_s = (1/T\_out\_n) \cdot f_s$. We now quantize the ratio into the signal `T_out_n`$_q$ (we define $T\_out\_n_q$ = Quant[(T_out_n)], where Quant[x] is the quantized value of x). In this second case the resulting sampling frequency $f'_{s\_q} = R_q \cdot f_s = (1/T\_out\_n_q) \cdot f_s$ incorporates a frequency deviation coming from the quantization error in the value of the ratio R_q. This error can be computed as $e_q = T\_out\_n_q - T\_out\_n$. These `T_out_n` and `T_out_n`$_q$ signals are processed in the *DIANA* engine. The engine settles hence the deviation of the output sampling frequency in the resampler. We define $\Delta f'_s = (f'_{s\_q} - f'_s)$ as the absolute frequency deviation in sample/s. We are interested in an estimate of the deviation based on the word width used in the fixed-point implementation of this ratio signal. We use again rounding for the implementation of these signals; the magnitude of the error will be at most half an LSB. For instance, if the fixed-point width of the ratio signal is thirty-two bits with twenty-nine fractional bits, the magnitude of the largest possible error e_{MAX} results

$$\left|e_{MAX}\right| \le 1/2^{29+1} = 9.3132 \cdot 10^{-10} \qquad \text{Eq.(5.5)}$$

We can compute the resulting quantized ratio value and the output sampling frequency if we know the error in the ratio signal. For this we incorporate the quantization error term into Eq.(4.8) as

$$R_q = \frac{f'_{s\_q}}{f_s} = \frac{f'_s + \Delta f'_s}{f_s} = \frac{1}{T\_out\_n_q} = \frac{1}{T\_out\_n + e_q} \qquad \text{Eq.(5.6)}$$

We define also the deviation with respect to the requested output sampling frequency as

$$\frac{\Delta f'_s}{f'_s} = \frac{f'_{s\_q}}{f'_s} - 1 = \frac{R_q \cdot f_s}{R \cdot f_s} - 1 = \frac{R_q}{R} - 1 = \\ = \frac{T\_out\_n}{T\_out\_n_q} - 1 = \frac{T\_out\_n}{T\_out\_n + e_q} - 1 = \frac{(1/R)}{(1/R) + e_q} - 1 = \frac{1}{1 + R \cdot e_q} - 1 \qquad \text{Eq.(5.7)}$$

Recall that we use $M = 2$ for the *MERCEDES* interfaces, this results in ratios $R \in [0.5 , 2]$; the frequency deviation that these ratios would induce if its quantization error adopts that maximum value, e_{MAX}, would result in $|\Delta f'_s / f'_s| = 4.6566 \cdot 10^{-10}$ and $|\Delta f'_s / f'_s| = 1.8626 \cdot 10^{-9}$, respectively. Note that the *T_out_n* values resulting for $R = 2$ and $R = 0.5$ can be represented without error in fixed-point arithmetic, the example intends however just to settle some boundaries for the deviation based on the ratios of the resampler.

This frequency deviation in the resampler is only relevant if we use it within a sandwich configuration, or if the real frequency or phase of the resampled signal is of importance; the different output sampling frequency will result in a phase slippage in the resampled sequence with respect to the ideal one. This phase slippage is proportional to the signal frequency; take for instance the up-sampling by a ratio $R = 1.4$ of the same tone at 3 MHz sampled at $62.5 \cdot 10^6$ sample/s with our fixed-point implementation as before, thirty-two bits with twenty-nine fractional bits. The quantization error in this case results in $e_q = T\_out\_n_q - T\_out\_n = 2.6609 \cdot 10^{-10}$. The frequency deviation would result in this case $\Delta f'_s = -0.0326$ sample/s and $|\Delta f'_s / f'_s| = 3.7253 \cdot 10^{-10}$. The achieved output sampling frequency would be 87,499,999.9674 sample/s instead of $87.5 \cdot 10^6$ sample/s. The resampler would take 30.67 s to slip by one sample in the output (to produce one less sample) due to this quantization error in the ratio. If we reconstruct the analog signal using the resampled sequence with a clock at the exact output sampling frequency of $87.5 \cdot 10^6$ sample/s the resulting tone will be at 3,000,000.0011 MHz. For the resampler case, this frequency deviation of 0.0011 Hz will be the error in the mapping of the signal discrete representation to the processing. The slippage in terms of the reconstructed signal would result in an error of 90º in 223.6 s when compared against an exact reference. Note that this effect on the phase is hardly noticeable in a computer simulation; with relatively large sampling rates the simulation times used cannot be very long, otherwise the outputs of the simulation require large volume of memory making the treatment complex. If the simulation lasts for 1 ms, the phase slippage in the reconstructed signal would be $4.023 \cdot 10^{-4}$ degree that is a value very difficult to notice with a Fast Fourier Transform (FFT) as it would require a large number of samples.

The good news is that we can cope with this problem; in section 3.3.7 we presented a solution that keeps the resampling ratio in a "resampling sandwich" stable at a desired given value, the *JOAQUINA* Frequency Locked Loop. That solution, that is valid for our final resampler sandwich, can be generalized with a similar approach for a single resampler; the resampled data can be stored in a memory, and then read with a clock whose frequency is the ideal output sampling rate to which the resampler transforms the input sequence. The fluctuation in the filling level of this memory can be used to correct on average the quantized resampling ratio of the resampler. Fig. 5.4 depicts a sketch of the concept; we want to resample a sequence `x[n]` to a new sequence `y[m]` according to the ratio R. For this we connect in series a resampler and a memory. The sequence `x[n]` is fed into the resampler that outputs `y[m]` according the ratio R_{q_corr} that it receives. This ratio contains corrections from a feedback loop, depicted in red around the resampler. The loop monitors the filling level of the memory against a reference value and generates a difference signal *corr_R* based on it. The resulting sampling rate f'_{s_q} at the output of the resampler includes the frequency deviation caused by the quantization of R in R_q, and the corrections of the feedback loop that on average

cancel this deviation. This sampling rate f'_{s_q} of the signal `y[m]` oscillates around f'_s and on average it matches its value $<f'_{s\_q}> = f'_s$. The input sampling rate region is depicted in yellow in the left of the figure, while the output region is depicted in blue in the right of the figure.

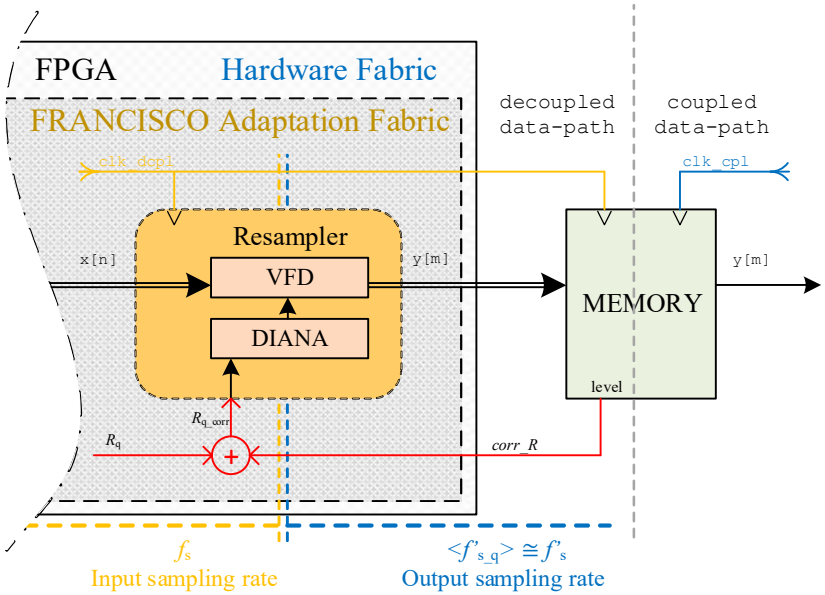


Fig. 5.4. *JOAQUINA* inspired feedback loop around the resampling ratio for a single resampler.

The input and resampled data streams use a decoupled data-path that has a processing clock `clk_dcpl`. This clock is also used to write the resampled sequence `y[m]` to the memory at the biased sampling rate f'_{s_q}. In the output port of the memory, we use a coupled data-path. This data-path uses a clock `clk_cpl` at a frequency F_{clk} to read the memory. This frequency exactly matches the desired output rate $F_{clk} = f'_s$ Hz. The resulting signal `y[m]` in the output port of the memory contains hence samples at an average sampling rate $<f'_{s\_q}> = f'_s$ sample/s and is clocked with at an exact frequency f'_s Hz.

We can hence conclude that the errors arising from the implementation in fixed-point of the *DIANA* engine are not relevant to affect the operation of the engine. The jitter in the sampling instant is masked by the VFD frequency response error and the error in output sampling frequency can be corrected with a feedback loop if needed.

5.3.1.2. VFD Filter Validation

This section presents the validation of the Variable Fractional Delay filter unit, the interpolator, of the resampler. This study analyses how well the filtered signal matches an ideal shifted signal (quality of the interpolation process), and how the different parameters of the VFD architecture affect the performance of the filter. The optimization of the data-path and the VFD filter-bank is out of the scope of the Thesis, however we present in the study some guiding results providing information to tailor the hardware resources of the VFD according to the required performance of the resampler. The study is based on a frequency-domain analysis, with the VFD optimized with $\alpha = 0.6$ in Eq.(4.34). This entity is the interpolating core of the resampler, whose performance will be deeply influenced by the VFD performance.

5.3.1.2.1. The VFD model

The discrete-time model $H_{VFD}(e^{j\omega},d)$ used in the study of the fixed-point VFD is depicted in Fig. 5.5. It includes the errors that make the resulting frequency response of the FIR filter bank (section 4.3.2) deviate from the ideal delay $H_{id}(e^{j\omega},d)$. The model has a data-path input port $X(e^{j\omega})$, a data-path output port $Y(e^{j\omega},d)$ and a delay input d. The VFD receives reference samples $X(e^{j\omega})$ with a certain spectral content ω and filters them according to the delay d with which it is configured.

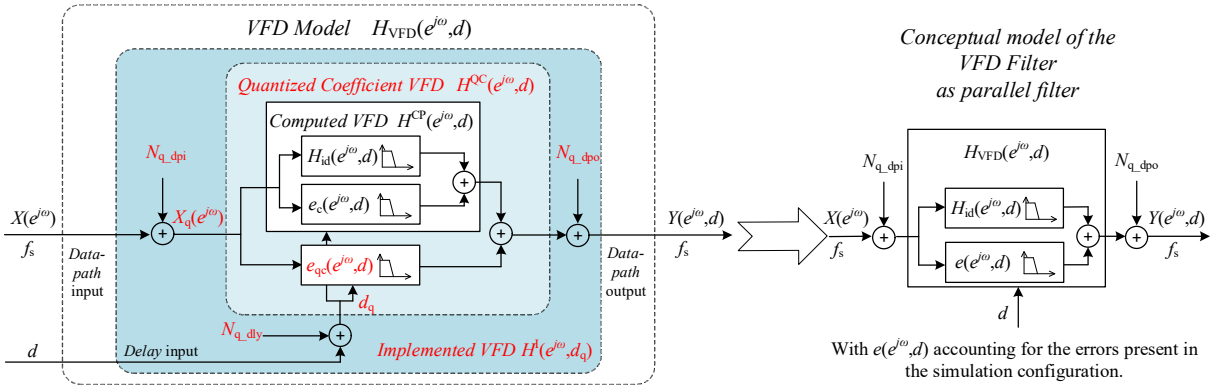


Fig. 5.5. Model of the VFD in the study.

Our analysis is based on the modelling of the deviation as noise added to the filtered output signal. The core of the model is composed by the discrete-time filter $H_{id}(e^{j\omega},d)$ (Eq.(5.8)) that is the ideal delay response (whose continuous-time frequency response introduced in Chapter 4 was presented in Eq.(4.11)), and the internal filter $e_c(e^{j\omega},d)$ that models the deviation error induced by the coefficients resulting from the optimization (section 4.4.2.1). The discrete-time frequency response of these time-dependent filters depends on ω, the normalized angular frequency in radian/sample, and d, the fractional delay in sample (that varies with time).

$$H_{id}(e^{j\omega},d) = 1 \cdot e^{-j\omega d} \qquad \text{Eq.(5.8)}$$

We call the addition of the two parallel filters $H_{id}(e^{j\omega},d)$ and $e_c(e^{j\omega},d)$ the Computed VFD filter $H_{VFD}(e^{j\omega},d) = H^{CP}(e^{j\omega},d) = H_{id}(e^{j\omega},d) + e_c(e^{j\omega},d)$. The parallel error filter $e_c(e^{j\omega},d)$ is not the only noise contribution present in the output $Y(e^{j\omega},d)$; the coefficients of the filter bank within the VFD are also quantized and implemented in fixed-point arithmetic. The quantization is modelled as an error term added to the coefficient value that also distorts the response of the VFD. The quantization is also present in the arithmetic signals of the architecture (input, output, delay and all the filter internal signals) implemented in fixed-point arithmetic, being also modelled as additive noise components. We rearrange the contribution of these quantization errors according to their effect in the frequency response. The quantization of the coefficients is modelled by the error filter/term $e_{qc}(e^{j\omega},d)$ in the model in parallel to $H^{CP}(e^{j\omega},d)$, and results in a deviation in the filter frequency response that is different for each delay and frequency combination. The quantization in the delay signal, modelled in the *DIANA* engine (section 5.3.1.1) as time jitter in the sampling instant, is here analysed looking at the frequency response. We can think of this delay error as another deviation in the frequency response of the filter; it results from the difference between the real delay

value d and the quantized delay value d_q received by the internals in $H_{VFD}(e^{j\omega},d)$. We model the delay error as the additive noise source N_{q_dly} that acts biasing the delay value that the filters receive, $d_q = d + N_{q\_dly}$, and hence the filter response. Recall that the magnitude of the maximum quantization error in the delay signal for our reference architecture results in $3.8147 \cdot 10^{-6}$ sampling periods (section 5.3.1.1); we can anticipate that such small value will not influence enough the filtered signal to notice this error. The last quantization effect comes from the truncation and/or rounding in the data-path. When this error is present in the input port, resulting from the sampling and processing in the precedent stages, we model it as the additive noise N_{q_dpi} that originates $X_q(e^{j\omega}) = X(e^{j\omega}) + N_{q\_dpi}$. The filter $H_{VFD}(e^{j\omega},d)$ processes hence a signal that contains a noise term that is also filtered. In the output this contribution is again an additive component $Y(e^{j\omega},d) = (X_q(e^{j\omega}) + N_{q\_dpi}) \cdot H_{VFD}(e^{j\omega},d) = (X_q(e^{j\omega}) + N_{q\_dpi}) \cdot (H_{id}(e^{j\omega},d) + e_{VFD}(e^{j\omega},d))$. To cope with the increment and propagation of this error, we have extended the word width in the internals of the filter after each arithmetic operation in the FIR bank and the Horner. This resulting extended word is only truncated in the output stages to the width of the input port. Our data-path also normalizes a full-scale input to a maximum magnitude of one; this lets us handle efficiently the quantization error by removing the excess LSBs in these output stages of the processing blocks. We include the quantization effects in the internals of the data-path (if any) as a second additive noise N_{q_dpo} at the output of the filter. This second component also accounts for the effect when the output data-path width is different from the input one. We call this model that incorporates all the quantization effects in signals and coefficients the Implemented VFD filter $H_{VFD}(e^{j\omega},d) = H^{I}(e^{j\omega},d_q)$ whose output signal results $Y(e^{j\omega},d) = (X_q(e^{j\omega}) + N_{q\_dpi}) \cdot H^{I}(e^{j\omega},d_q) + N_{q\_dpo} = (X_q(e^{j\omega}) + N_{q\_dpi}) \cdot (H_{id}(e^{j\omega},d_q) + e_I(e^{j\omega},d_q)) + N_{q\_dpo}$, where the term $e_I(e^{j\omega},d_q)$ models globally the errors in the implemented filter besides the data-path quantization error.

We performed analysis of the different error terms iteratively and we found that the data-path quantization is the most relevant contribution. We present here only the results for the Computed Filter $H^{CP}(e^{j\omega},d)$, the optimal frequency response approximating the ideal $H_{id}(e^{j\omega},d)$, and for the Implemented Filter $H^{I}(e^{j\omega},d_q)$, that incorporates all the errors present in the final implementation.

5.3.1.2.2. The computed filter

The Computed Filter (FIR bank architecture with six filters ($C = 5$) and fifteen taps each ($B = 15$) presented in section 4.4.2) whose pass-band has been optimized up to $\alpha = 0.6$ is referred as

$$H^{CP}(e^{j\omega},d)\big|_{\alpha=0.6} = H_{06}\left(e^{j\omega},d\right) = \left|\mathrm{mag}_{06}(e^{j\omega},d)\right| \cdot e^{-j\omega\cdot\tau_{06}(e^{j\omega},d)} \qquad \text{Eq.(5.9)}$$

This optimization corresponds to a cut-off frequency of $\omega_c = \alpha \cdot \pi = 1.885$ radian/sample (equivalent to $f_c = 0.3$ sample^{-1}), normalized with respect to the input sampling frequency. The error $e_c(e^{j\omega},d)$ resulting from the optimization when no other quantization effect is present can be obtained as

$$e_c\left(e^{j\omega},d\right)\Big|_{\alpha=0.6} = e_{06}\left(e^{j\omega},d\right) = H_{06}\left(e^{j\omega},d\right) - H_{id}\left(e^{j\omega},d\right) \qquad \text{Eq.(5.10)}$$

The magnitude of the error is a clear indicator of the degradation in the filter. The analysis of the phase is more complex; we instead study the group delay defining the group delay error function as the difference between the group delay of the computed filter $\tau_{06}(e^{j\omega},d)$ and the group delay d of the reference

$$e_{\tau 06}\left(e^{j\omega},d\right) = \tau_{06}(e^{j\omega},d) - d \qquad \text{Eq.(5.11)}$$

In Fig. 5.6 we depict the resulting magnitude (a) of the error function in Eq.(5.10) and the group delay error (b) of Eq.(5.11) in the optimized pass-band region for our VFD. The coordinates in the horizontal plane are the delay d in sample and the angular frequency ω normalized by 2π radians (resulting into a discrete frequency f in sample^{-1}). The resulting errors in magnitude and group delay are the surface whose magnitude is mapped to the z coordinate. The magnitude of the error function presents dependency with the frequency axis and is below $1.25\cdot10^{-5}$ in the most part of the optimized pass-band, and only for frequencies close to the cut-off, and for delay values close to $d = \pm0.5$ sample does it reach $7\cdot10^{-5}$. The surface presents soft wavy pattern/oscillations along the frequency axis, increasing at higher frequencies. The error dependency in the delay axis is symmetric around the origin, $d = 0$ sample, and presents the maxima for delay values around $d = \pm0.35$ sample, with exception in the frequencies very close to the cut-off where the error explodes for $d = \pm0.5$ sample. The group delay error presents the same dependencies in the frequency and delay axes but is larger in the lower frequency range instead reaching a value of $1\cdot10^{-4}$ sample.

We can also observe that in the lower frequencies the error magnitude decreases to zero for delay values close to the accepted VFD range, +/- 0.5 sample. For frequencies larger than 0.1 sample^{-1} the error grows also for these corner delays, and the contribution of the frequency axis becomes more relevant than the delay parameter in the response. It is also clearly noticeable that the error becomes zero for all frequencies, when the delay value is $d = 0$. That is sure expected; the Farrow architecture in which the VFD is implemented is based on a Taylor series. It evaluates the series with the delay value d as variable, as presented in Eq.(4.33). When the delay is $d = 0$, only the first coefficient (filtered data resulting in the coefficient $u_0[n]$ in Fig. 4.13) is present in the output. All the other filter branches are multiplied by powers of zero. Furthermore, the sub-filter $g_0[n]$ in the $u_0[n]$ branch has as impulse response a Dirac delta; the

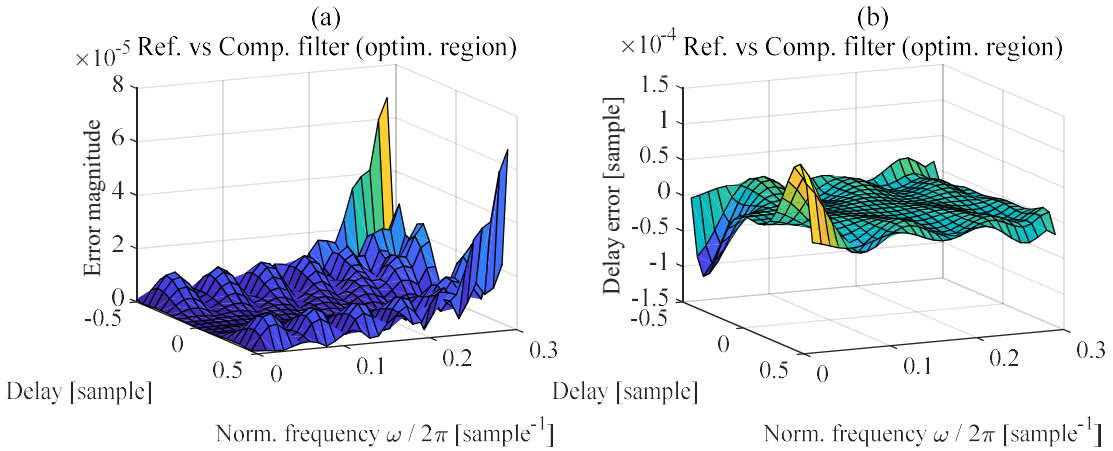


Fig. 5.6. Error function and group delay error function for the computed filter with $\alpha = 0.6$. Zoom in the pass-band region. (a) Magnitude of the error function. (b) Fractional group delay error.

output is a replica of the input. The error becomes zero in this case as this is the only sub-filter contributing to the bank output.

We have also computed the achievable Signal to Noise Ratio (SNR) as figure of merit when we scan the filter with a pure tone that has a magnitude of 1 V_{peak}. We use a 1 V_{peak} signal as this value is the full-scale input magnitude of our quantized data-path, and hence the test signal that in the VFD will achieve the maximum SNR. The noise term $\sigma^2_{N\_RMS}(e^{j\omega},d)$ contains only the coefficient optimization error for $H^{CP}(e^{j\omega},d)$, with SNR defined as

$$SNR\left(e^{j\omega},d\right) = \frac{\sigma^2{}_{X\_RMS}(e^{j\omega})}{\sigma^2_{N\_RMS}\left(e^{j\omega},d\right)} \qquad \text{Eq.(5.12)}$$

The resulting surface is presented in Fig. 5.7. The lowest SNR value results in 95 dB close to the cut-off frequency for a delay $d = 0.5$. In the rest of the pass-band it reaches peak values of $SNR_{MAX} = 130$ dB, with an average value $SNR_{AVG} = 110$ dB. The surface keeps the wavy pattern along the frequency axis. These large SNR values make our Computed Filter acceptable for our DSP application of Chapter 6.

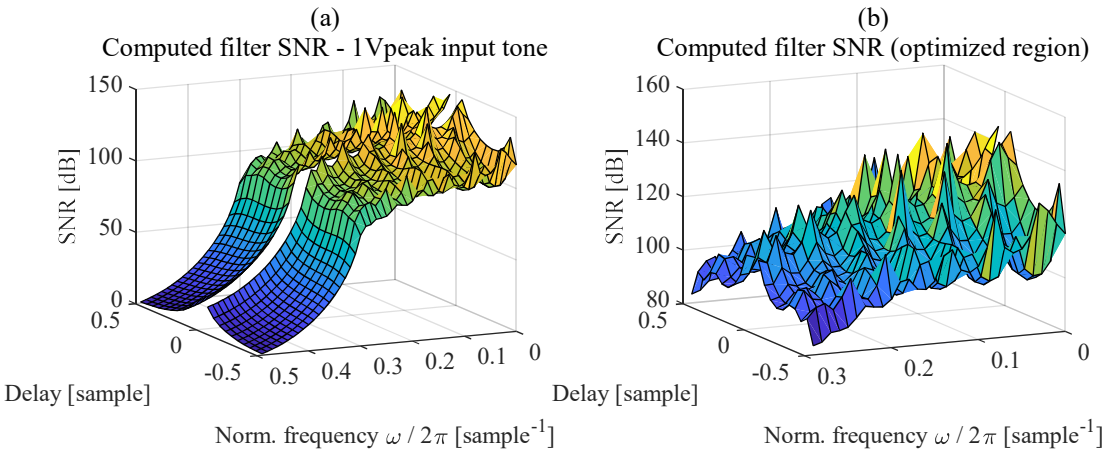


Fig. 5.7. SNR at the output of the computed filter when excited with a 1 V_{peak} tone with $\alpha = 0.6$. (a) First Nyquist zone. (b) Zoom in the pass-band region.

5.3.1.2.3. The implemented filter

We have conducted a similar study with the coefficients and the delay quantized to fixed-point arithmetic; no significant degradation was found in the filter response. We hence present here the Implemented Filter VFD filter, $H^I(e^{j\omega},d_q)$, that uses fixed-point arithmetic with rounding in all the signals. The width of the data-path is sixteen bits wide at both the input and output ports; one sign bit and fifteen fractional bits. This word is a commonly used value inherited from the many analog to digital converters in the market that use this width. The delay value has been quantized to eighteen bits as presented in 5.3.1.1, the resulting error will induce (as presented when describing our VFD model) a shift in the achieved frequency response with respect to the ideal filter. The coefficients have been quantized to a fixed-point word width of twenty-seven bits; one sign bit and twenty-six fractional bits.

We present the resulting error magnitude of our filter in Fig. 5.8 when we scan it with a full-scale input, a 1V_{peak} (we iterate different frequency-delay pairs $(e^{j\omega},d)$). The first significative difference that we

notice with $H^{CP}(e^{j\omega},d)$ is the larger error magnitude that now reaches in the optimized region a value that fluctuates around $2\cdot10^{-5}$. The largest contribution to this error is coming from the quantization of the data-path. Note that the largest quantization error, one half of an LSB, results in $MAX(e_q[n]) = \pm\ 1.52\cdot10^{-5}$ V; our error is larger as the input quantization noise is also being filtered by the VFD. Our fixed-point data-path masks hence the error of the $H^{CP}(e^{j\omega},d)$ within almost all the optimized pass-band. Only in the high frequency region, very close to the cut-off frequency, and in the corner delays |d| = 0.5 sample, is the error in the frequency response of the filter larger than the quantization error. In this case it is the computed filter response and not the fixed-point implementation of the data-path that degrades the VFD. These corner cases are however very irrelevant compared to all the rest of flat optimized region. We can hence assume as error a flat surface visible in the enlargement of Fig. 5.8(b); the quantization noise contribution, uncorrelated with the input signal, is at least twice larger in magnitude than the worst peaks and in general several orders of magnitude larger than the average value that we had in $H^{CP}(e^{j\omega},d)$. The error seems different when the input signal has frequency of 0 $sample^{-1}$ (a DC signal). In this case the error increases from $2\cdot10^{-5}$ to $3\cdot10^{-5}$ and seems unrelated to the rest of the error surface.

This result is expected; first recall that we use in the test a full scale 1 V_{peak} input that is quantized to a signed fixed-point arithmetic word. The number of quantizer levels in such a signed word might not be even; the positive range might have one less level. An input value with 0 V, that is neither positive nor negative, already consumes a discrete word that is formed with all the bits set to 0. Normally the missing level lies in the upper limit of the representable values. The error for a positive full-scale value (V_{peak} = 1V in our data-path) is hence 1LSB instead of ½ LSB. This is the case for the System Generator implementation of the quantizer, it maps the 0 V input to the middle word composed of all the bits set to 0. The error for a full-scale input is hence one LSB instead of just one half of an LSB. The DC error depicted in Fig. 5.8(b) verifies this adopting a constant value of $3.052\cdot10^{-5}$ consistent with our sixteen bits data-path word.

The last significant difference with $H^{CP}(e^{j\omega},d)$ is that now when using a delay value $d = 0$, the error is no longer 0 but the quantization error of the input signal. Recall that the frequency response error of the computed filter for this delay value is 0. However, if the input signal is quantized, this error will unavoidably be present at the output of the VFD.

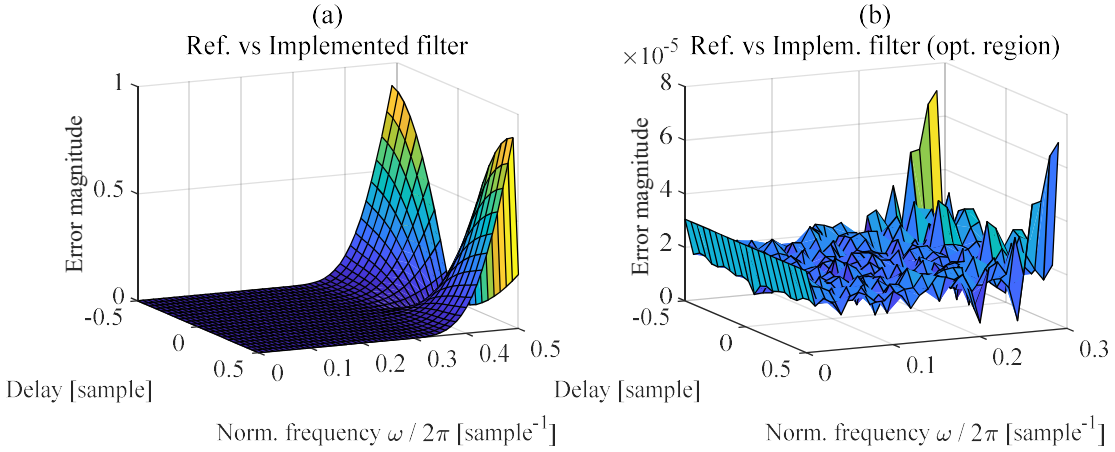


Fig. 5.8. Error function for the implemented filter with $\alpha = 0.6$ and sixteen-bit data-path. (a) Magnitude of the error function. (b) Zoom in the pass-band region.

The resulting peak SNR for the implemented (quantized) filter can be seen in Fig. 5.9; it decreases to an average value close to 94 dB in the same flat surface independent of the frequency. The implemented filter therefore degrades the SNR_{AVG} by 16 dB when compared to the computed filter in 5.3.1.2.2. The contribution of the VFD optimization error in the resampler is negligible in the optimized bandwidth compared to the data-path quantization error for the presented fixed-point implementation. However, the maximum achievable SNR is still very close to the limit Signal-to-Quantization-Noise-Ratio (SQNR) of 98.09 dB (sixteen bits). Note that the design data-path SQNR at the input of the VFD contains only noise coming from the quantization noise, while at the output the SNR includes the contribution of the Implemented VFD, 4 dB, that mainly result from the filtering of the input quantization noise.

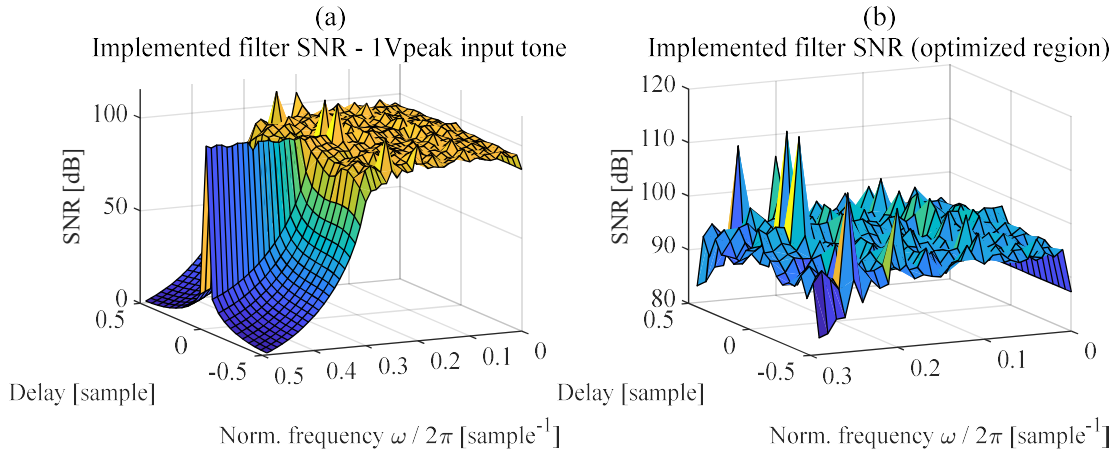


Fig. 5.9. SNR at the output of the implemented filter with $\alpha = 0.6$ and sixteen-bit data-path when excited with a 1 V_{peak} tone with $\alpha = 0.6$. (a) First Nyquist zone. (b) Zoom in the pass-band region.

5.3.1.3. Resampler Validation

We present in this section the validation of the resampler as a whole entity, that includes the *DIANA* engine, the VFD filter and the Synchronization elements implemented in the *FRANCISCO* fabric. We study how well these units operate together, and how well the resampler estimates the resulting values of the subjacent analog signal sampled at the output rate.

5.3.1.3.1. The resampler model

Fig. 5.10 depicts our reference model for the resampler, that makes use of the hardware architecture and functional units presented so far, and is also filter-inspired. We will study how the quantization error of the arithmetic operations within the data-path and the delay computation degrades the performance. It is based on the VFD model of Fig. 5.5, but we now incorporate also the *DIANA* engine, and the

characterization uses the frequency ω of the signal and the resampling ratio R, instead of the delay d, as input parameters. Our results are hence now based on the ratio-signal frequency pair ($e^{j\omega}$,R).

The VFD response presented in 5.3.1.2 is always reproducible for a given frequency-delay pair ($e^{j\omega}$,d). This is not any longer the case for the resampler; it uses a delay value (and a reference) that changes from one sample to the next (except for very specific cases as $R = 1$ or $R = 0.5$). The frequency response of the resampler might even be different for two identic ratio-signal frequency pairs; the input sample history and the ratio history determine the reference sample used in the computation of the delay value. We have hence a response that depends on the past. To cope with this memory effect in the resampler, we characterize statistically the ratio-signal frequency pairs; we average different iterations with the same configuration. We anticipate that the delay/reference variations have not much influence when averaging the performance of the implemented resampler. This assumption is based on the fact that the frequency response of the implemented VFD in the optimized region is uniform. The oscillations in the frequency and delay axis are only noticeable when we do not quantize the data-path. We hence expect results in the quantized implementation of the resampler very similar to the implemented VFD performance.

Looking at the internals of the resampler model we see that the VFD entity is the same as discussed in section 5.3.1.2. The only difference is now that the delay d_q is computed in the *DIANA* engine using the resampling ratio R. In the resampler model the data-path port receives reference samples $X(e^{j\omega})$ with a sampling rate f_s, the ratio input receives the ratio signal R with a rate f_p, the *DIANA* computes the delay based on the ratio and the output of the resampler provides the resampled signal $Y(e^{j\omega})$ at a sampling rate f'_s after filtering in the VFD. The quantization noises are depicted again as additive components N_q in red. The only new noise component is N_{q_r}, that is incorporated to the ratio input when we use a quantized implementation (section 5.3.1.1). In this filter-based model, the reader has to bear in mind that the input and output rates of the resampler are different (except for the case $R = 1$). This brings some mathematical inconsistencies to the figure. We however do not intend to obtain a precise mathematical model, but just a qualitative simple model that let us identify conceptually the contribution of the different errors present in the resampler.

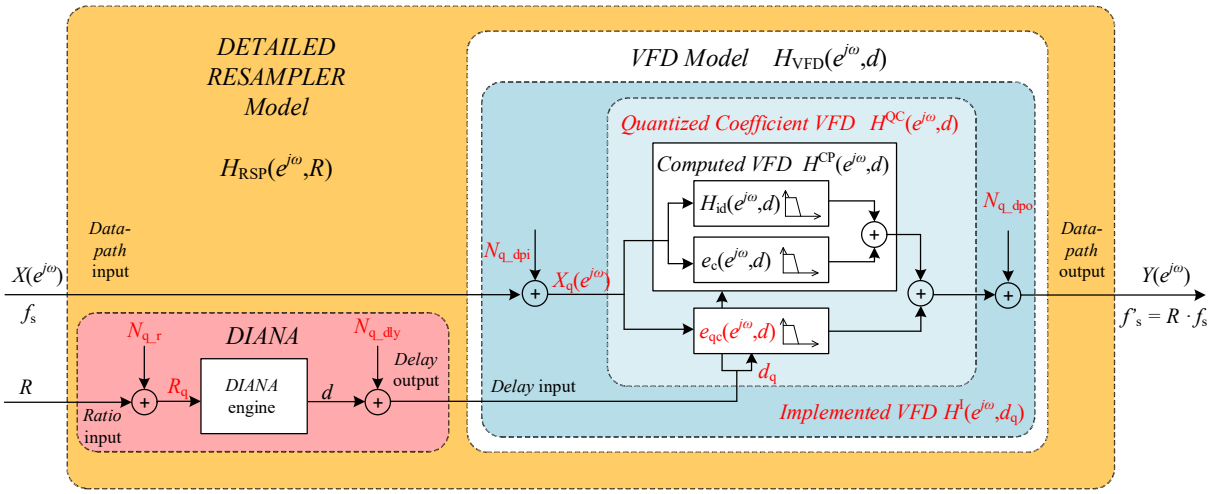


Fig. 5.10. Detailed model of the resampler in the study.

We will now analyse the differences that the *DIANA* errors introduce in the resampler model. We recall the results presented in 5.3.1.1; the quantization effects on the engine will influence the resampler twofold. The ones in the delay value d create a time jitter in the output sampling instant. This jitter results in a misconfiguration of the delay value used by the VFD, and deviation on the frequency response of the VFD. On the other hand, the quantization error in the ratio signal R will create a deviation in the output sampling frequency. The achieved resampling ratio might differ slightly from the desired one; this will result in a phase slippage when the time-domain signal is compared to an ideal reference at the ideal output sampling frequency. We also saw in Chapter 3 that when using two resamplers in sandwich configuration these errors in the ratio can lead to a misconfiguration between its inverse ratios. We offered solutions to both problems based on the *JOAQUINA* feedback loop; in the sandwich it keeps on average inverse ratios, and in the resampler it keeps on average the exact sampling frequency at the output. However, we analyse and simulate in this section the resampler in open-loop; we do not correct the errors resulting from the quantization of the ratio. As we anticipated already in the *DIANA* study, this error has a very small influence in the quantitative results (as opposed to the functional problems that it supposes for the sandwich). It depends on the value of the ratios and the resulting largest sampling frequency deviation for our implemented architecture remains below $|\Delta f'_s / f'_s| = 2 \cdot 10^{-9}$. Note also that the phase slippage results also in very small values for short simulations times, as this is the case. We can hence run the simulations in open-loop and consider the results as acceptable, as long as the simulation time last for a few ms, disregarding the influence of this sampling frequency offset.

As in the VFD case, we have studied the influence of these error sources by gradually incorporating them to our model. We present here however only the most relevant results. We call Computed Resampler $H^{CPR}(e^{j\omega},R)$ the resampler model that incorporates the $H^{CP}(e^{j\omega},d)$ VFD filter and does not quantize ratio or delay signals; its only error source comes from the computed coefficients of the VFD. This model lets us understand the effect (if any) of the different delays used by the VFD for a single ratio-frequency pair. When we include quantization to the model (additive noise sources in the data-path, delay and ratio signals, and also the frequency deviation due to the quantized coefficients) we call this model the Implemented Resampler $H^{IR}(e^{j\omega},R)$. To make the results more readable we simplify the detailed model of Fig. 5.10 to the simplified filter-based version of Fig. 5.11, that has a frequency response $H_{RSP}(e^{j\omega},R)$ and models as an average the delay time-dependent response of the resampler for a given frequency ω and ratio R based on several iterations of the simulation. Again, this is just a conceptual representation of the resampling process,

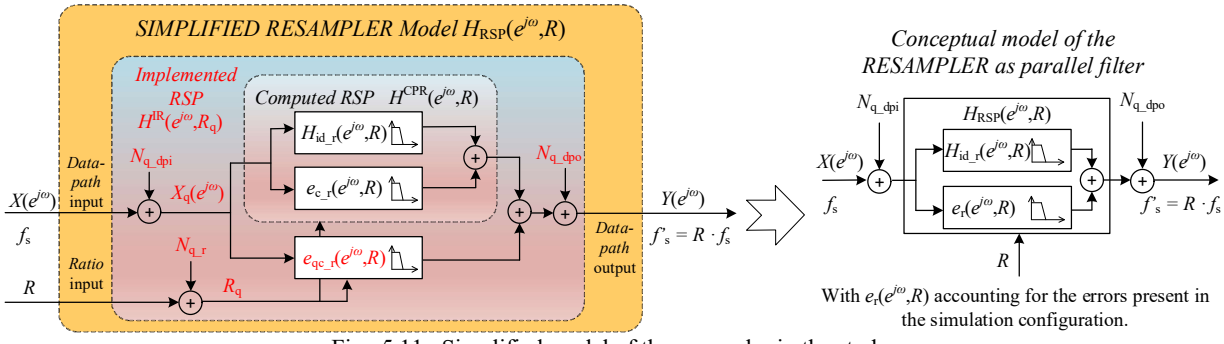


Fig. 5.11. Simplified model of the resampler in the study.

that does not intend to be mathematically rigorous. The core of the model, includes two parallel filters. One models the ideal response of a resampler $H_{id\_r}(e^{j\omega},R)$, as a noiseless resampling process. The second $e_{c\_r}(e^{j\omega},R)$ models the error in the output of the resampler due to the computed coefficients of the VFD (it averages several simulations for a given ratio). Then, when quantization is added to the model, we include the additive noise sources N_{q_dpi} and N_{q_dpo} in the data-path, N_{q_r} in the ratio and the parallel filter $e_{qc\_r}(e^{j\omega},R)$ that accounts for quantization in the VFD coefficients. This $e_{qc\_r}(e^{j\omega},R)$ filter also processes the input when present. All these resampler filters have as input parameter the ratio R instead of the delay d. In the time-domain simulations of the resampler, we use the SSRC hardware architecture presented in Chapter 4, that has been elaborated for each unit in the previous sections. The accepted ratios are hence for down-sampling, transparent and up-sampling modes $R \in [0.5 , 1)$, $R = 1$ and $R \in (1, 2]$ (*MERCEDES* configuration, $M = 2$).

5.3.1.3.2. The computed resampler

We first present the results for the Computed Resampler $H^{CPR}(e^{j\omega},R)$, that uses the VFD of 5.3.1.2.2. The error in the output of the resampler in this situation comes only from the frequency response deviation of the VFD as the delay and ratio signals are exact. This lets us identify the effect that the changing different delay values have in the resampler for each output sample. For sake of clarity, we present results and error surfaces differentiating simulations for down-sampling and up-sampling ratios using static values. These different configurations let us better relate the results with the limit bandwidth for the input signal presented in section 3.3.10. In any case we saw that both the VFD and the *DIANA* engine support ratio variations in real-time, that will not degrade the response of the resampler. Further simulations are presented in Chapter 6 where the entire BSP Architecture is operated with variable rates.

We first depict in Fig. 5.12 the error surface $|e_r(e^{j\omega},R)|$ with a down-sampling configuration (note that in this case $e_r(e^{j\omega},R) = e_{c\_r}(e^{j\omega},R)$). One of the axes of the response presents the frequency ω at which we evaluate the resampler while the other depicts the ratio R. We observe in Fig. 5.12(a) that the resampler response is very flat in the optimized pass-band region of the VFD. When analysing in detail this optimized region in Fig. 5.12(b), we observe that for each frequency the error response is almost independent of the resampling ratio. However, when looking at the response for a given ratio, we can still identify the periodic

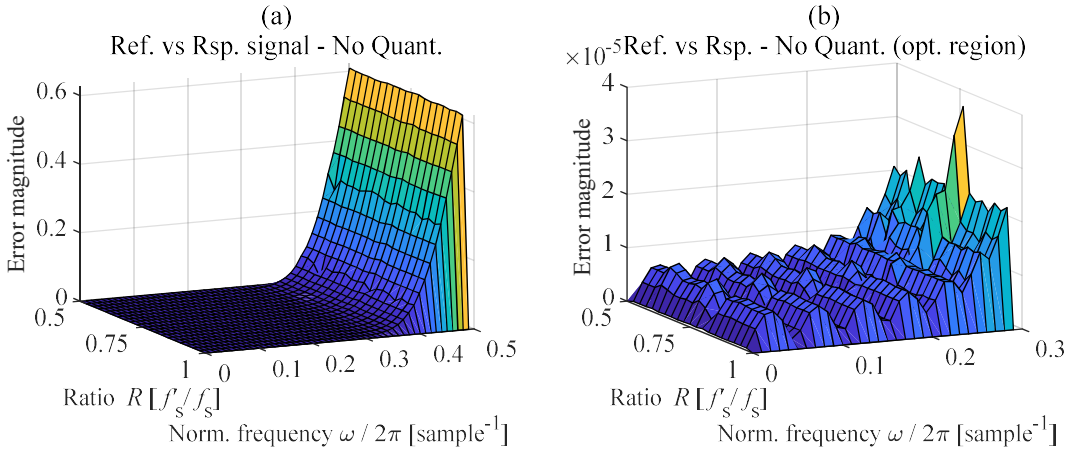


Fig. 5.12. Error function for the computed resampler, down-sampling ratios. (a) Magnitude of the error function. (b) Zoom in the pass-band region.

oscillations dependent with the frequency that we introduced in the ideal VFD. The magnitude of the error in the surface is of the same order as the one of its VFD (Fig. 5.6) with an average value below $1 \cdot 10^{-5}$ for frequencies below $f < 0.2$ sample^{-1}. In the upper frequencies of the pass-band the error increases to values around $3 \cdot 10^{-5}$, with a spike for a ratio value of $R = 0.85$. In this simulation, the error with $R = 1$ goes to 0; there is no resampling and the delay value $d = 0$ is predominant in the iterations run with this configuration. It is also the case for $R = 0.5$ that again presents a predominance of $d = 0$. This value results when the input and output sequences have the sampling instants aligned; the first output is sampled at the same time instant as the input, resulting in a delay $d = 0$, with the successive delay values repeating periodically the pattern $d = \{2\ 1\ 0\ 2\ ...\}$. We have intentionally configured the resampler and test bench for this to happen as a sanity check in the simulations. With further characterization of the resampler (giving more randomness to the reference history to achieve a delay $d \neq 0$ for $R = 1$) the resulting error would be an averaged value in the order of the rest of the surface; it would use other constant delay values averaged.

Note also that according to section 3.3.10, the bandwidth of the input signal cannot exceed $\omega = 2\pi \cdot f = 2\pi \cdot 0.25$ when the ratio is $R = 0.5$ (with a clocking relation $M = 2$), increasing linearly up to $\omega = 2\pi \cdot f = 2\pi \cdot 0.3$ in $R = 0.6$. However, the simulation has been run with frequencies spanning up to $\omega = 2\pi \cdot f = 2\pi \cdot 0.5$. The results seem to be valid as the SNR value is in the same order of magnitude than for the valid frequencies. However, this is an artifact resulting from a folded image of the input tone in the second Nyquist zone. This image lies back folded to the first Nyquist zone in the sampling (or resampling) process. This artefact is present in both the resampler and our resampled reference signal. The error signal for these frequencies results then false with apparently no error (or a very low value) as the reference and resampled signal contains the identic artifacts. We will better depict this in the entire BSP Architecture verification. In any case, the resulting surface has to be considered only in the input frequency ranges of section 3.3.10.

We made the assumption that the resampler error response for a given ratio is an average of the responses of its different delays. Comparing Fig. 5.12 with Fig. 5.6(a), we verify that the magnitude of the resampler error (i.e., below $1 \cdot 10^{-5}$ for frequencies below $f < 0.2$ sample^{-1}) for each frequency is indeed in this same order of magnitude as the average of the VFD, and very flat for a given frequency. The oscillations of the error along the frequency axis adopt also the same previous VFD pattern.

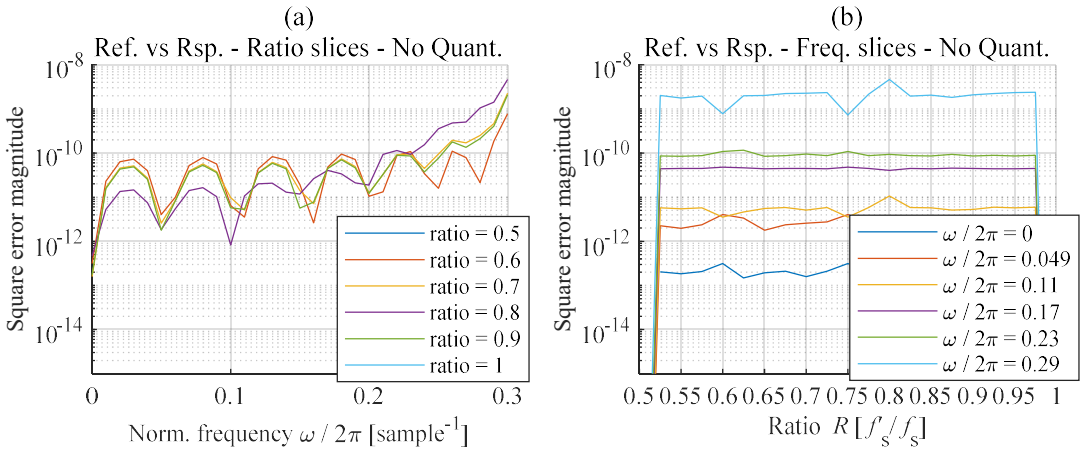


Fig. 5.13. Magnitude of the square error function for the computed resampler in down-sampling configuration. Zoom in the pass-band region. (a) Slices along the ratio axis. (b) Slices along the frequency axis.

In Fig. 5.13 we slice the square error in ratio and frequency traces. We clearly identify the periodic oscillations along the frequency axis in the ratio slices and the flatness of the response for each ratio in the frequency slices. We can therefore conclude that our resampler response assumption, as an average of the error response of the VFD, was right.

Fig. 5.14 presents the results for up-sampling ratios with identical conclusions.

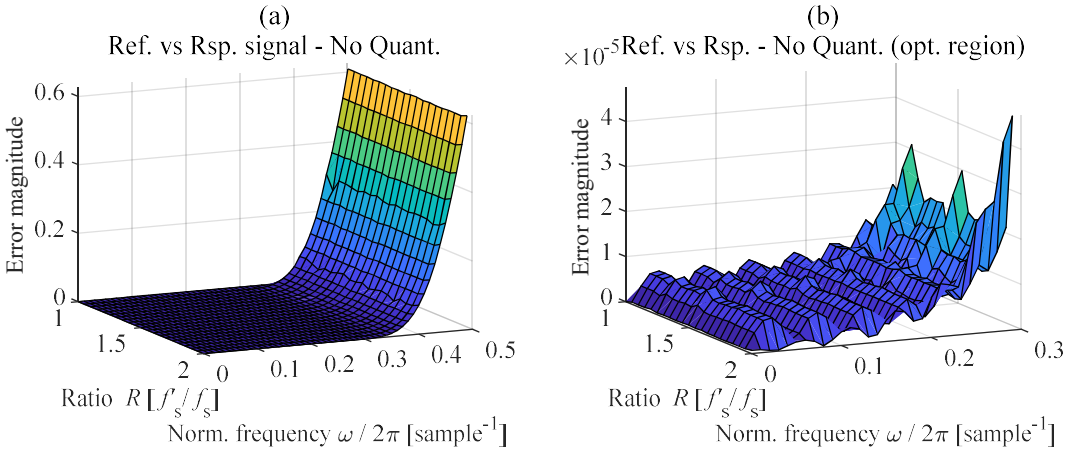


Fig. 5.14. Error function for the computed resampler, up-sampling ratios. (a) Magnitude of the error function. (b) Zoom in the pass-band region.

We finally depict in Fig. 5.15 the achieved SNR surface (note that the surface depicts results with the same folded artifacts presented before). We achieve SNR ratios above 115 dB for frequencies below $f < 0.2$ sample^{-1} with an $SNR_{AVG} = 110$ dB, and in any case for the cut-off frequencies our SNR decreases to 95 dB only. The SNR surfaces for ratios $R = 1$ and $R = 0.5$ tend to infinite as the error tends to 0, and hence are omitted in the plot. Note that the SNR presents again the same periodic oscillations along the frequency axis and remains flat for a given frequency along the ratio axis averaging the VFD response of the different delays as expected. The degradation of the resampler with respect to the VFD is hence negligible, and the varying delays can be neglected.

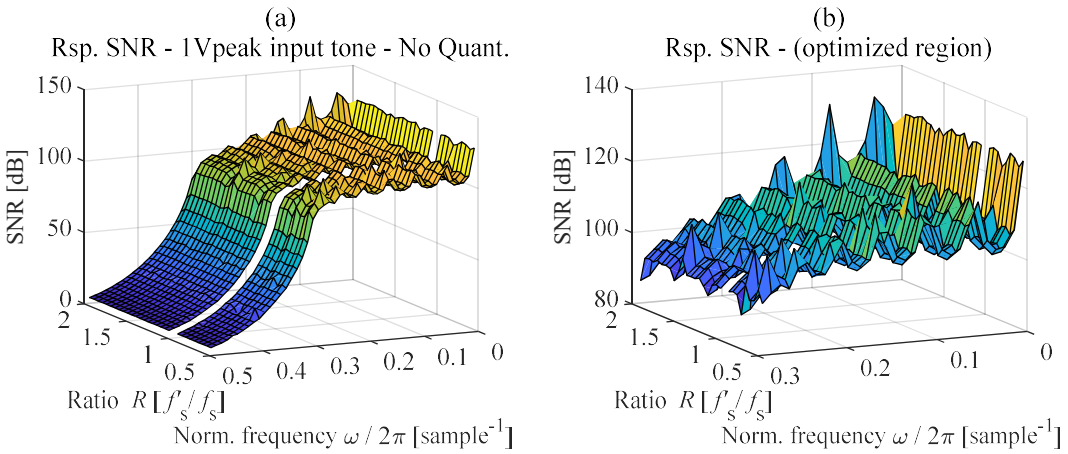


Fig. 5.15. SNR at the output of the computed resampler, both down-sampling and up-sampling ratios, when excited with a 1 V_{peak} tone. (a) First Nyquist zone. (b) Zoom in the pass-band region.

5.3.1.3.3. The implemented resampler

We now present the results of the Implemented Resampler $H_{RSP}(e^{j\omega},R) = H^{IR}(e^{j\omega},R_q)$ in which the signals are quantized to fixed-point arithmetic. The study is analogous to the Computed Resampler and we present only the more remarkable results. We use a width of thirty-two bits with twenty-nine fractional bits for the fixed-point signal representing the ratio; the resampling ratio and the computed delay are hence not exact. Recall that in the *DIANA* engine validation in 5.3.1.1 we computed the sampling frequency deviation error for such a ratio signal implementation using a resampling ratio R = 1.4 with a 3 MHz input signal. The error in the quantized ratio was $e_q = T\_out\_n_q - T\_out\_n = 2.6609 \cdot 10^{-10}$ and the frequency deviation resulted in this case $\Delta f'_s$ = -0.0326 sample/s and $|\Delta f'_s / f'_s| = 3.7253 \cdot 10^{-10}$. By examining the frequency response of the computed resampler in Fig. 5.14, the deviation resulting from this error seems negligible; the response does not vary very abruptly around that resampling ratio. We can hence run safely the simulation in open-loop as we did in the previous section even with the fixed-point implementation.

We depict slices of the magnitude of the square error function in down-sampling and up-sampling configurations respectively in Fig. 5.16 and Fig. 5.17. These slices are traces along the ratio axis in the left plots and slices along the frequency axis in the right plots of these two figures. We can observe that the implemented resampler response remains flat in the optimized pass-band region of the VFD; independent

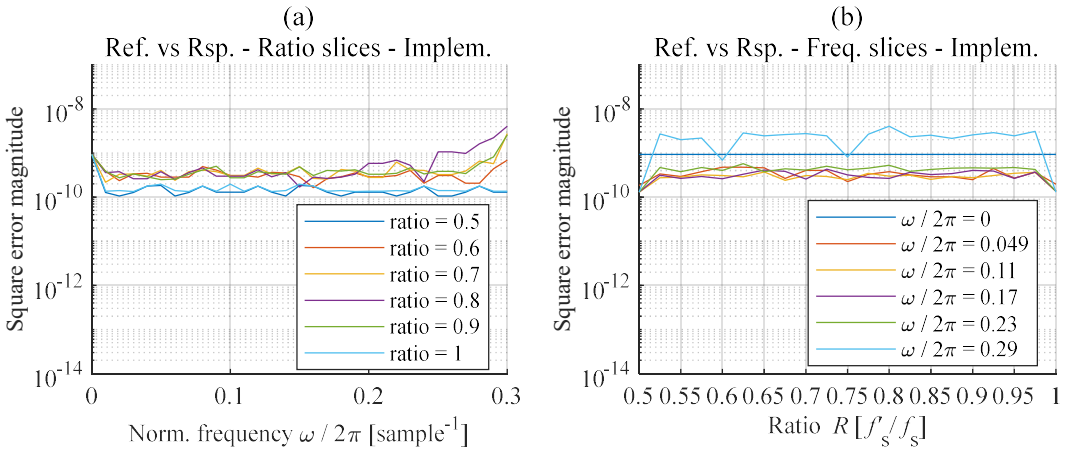


Fig. 5.16. Magnitude of the square error function for the implemented resampler in down-sampling configuration. Zoom in the pass-band region. (a) Slices along the ratio axis. (b) Slices along the frequency axis.

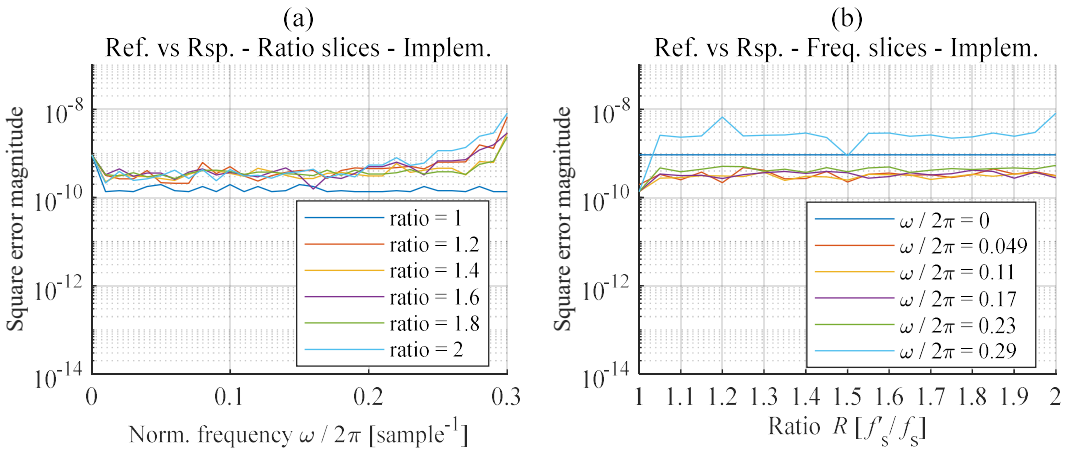


Fig. 5.17. Magnitude of the square error function for the implemented resampler in up-sampling configuration. Zoom in the pass-band region. (a) Slices along the ratio axis. (b) Slices along the frequency axis.

of the resampling ratio. However, now the data-path quantization noise effect dominates the ripples of the frequency response error; we do not see any more the periodic oscillations fluctuating in the frequency axis that were induced by the VFD. The magnitude of this square error, that we modelled as the error filter $e_r(e^{j\omega},R)$ in Fig. 5.11 adopts a value around $3\cdot10^{-10}$. This value is as expected larger than the one in the Computed Resampler, which was below $1\cdot10^{-10}$. Note also that the error with $R = 1$ and for $R = 0.5$ is now no longer zero, it is close to the quantization noise of the data-path (with these ratios the delay value $d = 0$ is still predominant in the different iterations performed).

We finally present the maximum achieved SNR surface (full scale input, $1V_{peak}$ tone) in Fig. 5.18. The achieved value in the final implemented resampler remains similar to the one of the VFD. We obtain a flat surface with SNR in the order of 95 dB that is very close to the limit SQNR of 98.09 dB of the data-path. Particularly, the SNR for ratios $R = 1$ and $R = 0.5$ adopts a value that matches the SQNR of the data-path. We can hence generalize our conclusions presented so far, but now for the entire optimized pass-band region; the contribution of the VFD frequency response error, and other quantization errors, besides the data-path, have a negligible effect. These effects are masked by the data-path quantization error when using a sixteen-bit implementation, but the result still reaches a value only 3 dB below the limit SQNR. In any case, the obtained SNR is an accurate average of the SNR of the different delay values of the VFD being used with a given resampling ratio R. Note that the surfaces are plotted for all the ratio values along the entire first Nyquist region, however the valid frequency ranges of the surface must be interpreted according to section 3.3.10.

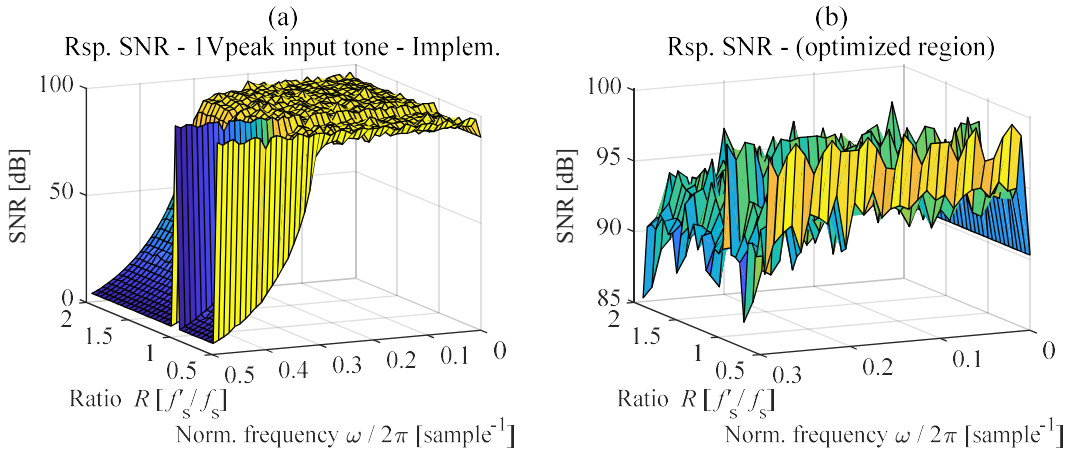


Fig. 5.18. SNR at the output of the implemented resampler, both down-sampling and up-sampling ratios, when excited with a 1 V_{peak} tone. (a) First Nyquist zone. (b) Zoom in the pass-band region.

5.3.2. BSP Architecture validation

This section presents the validation of the Beam Synchronous Processing Architecture, what we call the resampler sandwich. We study how well the BSP performs the double sampling rate conversion processes by studying the SNR at the output of the sandwich, when no application specific processing is performed within the BSP sandwich. In this situation the input signal crosses the Architecture being only up-sampled in the input resampler and down-sampled in the output resampler without any intermediate

degradation. We hence analyse the decrease in the spectral purity for the processed signal crossing the BSP Architecture that is implemented using two instances of the resampler presented in 5.3.1.3.

5.3.2.1. The BSP Architecture model

In Fig. 5.19 we present our sandwich model $H_{SND}(e^{j\omega},R_{in})$ based upon the one for the resampler in Fig. 5.11. Now it incorporates an output stage with a second resampler. Our results are averages of time simulations computed for different signal frequency-input resampling ratio pairs ($e^{j\omega}$, R_{in}) (note that the output ratio must be the configured with the inverse $R_{out} = 1 / R_{in}$). We perform statistical averaging to cope with the dependence of the sandwich on the past history of the input signal and ratio. Recall that for the single resampler case, the quantization noise in the input data-path is filtered once, while now in the BSP Architecture case, this noise is filtered twice (double resampling process that includes a second output resampler). The second resampler filters also all the noise contributions present at the output of the input resampler, plus the noise added in the BSP processing region (if any). We depict this in the model; the first resampling stage modifies the sampling rate of $X(e^{j\omega})$ resulting in the intermediate signal $Y(e^{j\omega})$ at a sampling rate f'_s that is propagated to the BSP region. The output resampler recovers, using the inverse ratio R_{out}, an output signal $Z(e^{j\omega})$ that (ideally) has the original sampling rate $f''_s = f_s$ of the signal $X(e^{j\omega})$. If no errors are present in the double resampling process, the input and output signals would hence be identical. This is not the case for ratios different than $R = 1$, as for instance, the approximation error of the computed VFD will always be present.

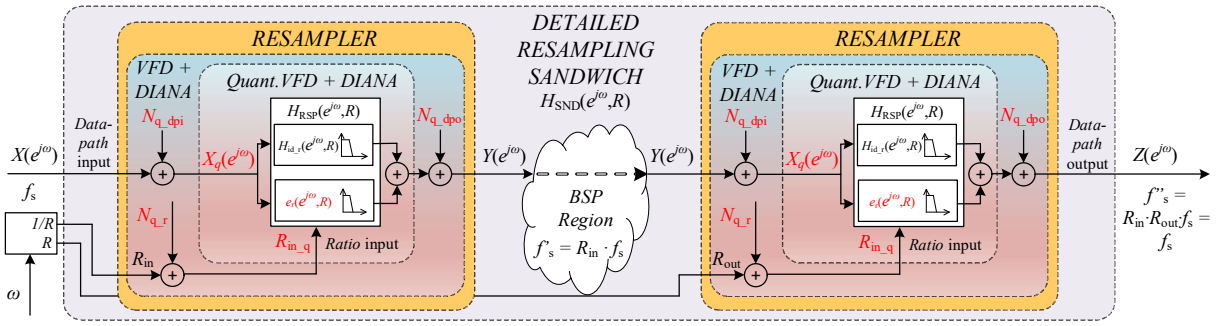


Fig. 5.19. Detailed model of the BSP sandwich Architecture in the study.

We will run the tests without the feedback loop around the ratio signals (in open-loop); the recovered output sampling frequency will incorporate an offset when quantizing the input and output resampling ratios. The magnitude of these errors is very small, and they impact the functional behaviour rather than the performance as we have already presented in the *DIANA* and the resampler verification (sections 5.3.1.1 and 5.3.1.3). Note also that our simulation iterations do not last for very long, the simulation time is less than a ms, and hence the phase slippage is almost negligible in this case. However, by including these effects, the resulting characterization of the final performance of the BSP Architecture is complete and sets a lower boundary for the performance when using a feedback loop. For this reason, we do not include the *MERCEDES* interfaces in this model, but just the decoupled data-path segment of the BSP region. These *MERCEDES* interfaces have no influence on the performance of the resampling process, they are just needed from a logic and physical point of view, hence not biasing our results.

In the *DIANA* engine validation, section 5.3.1.1, we presented an example with figures for these ratio quantization errors in the case of a single resampler when no regulation loop was used (we introduced these results as the *DIANA* engine performance). Consider that resampler as the input stage of the BSP test bench in Fig. 5.19. We can compute the resulting sampling frequency offset in the BSP sandwich output incorporating the output resampler to the analysis when there is no regulation around the ratios.

In the ideal case (when no quantization is performed), the inverse ratio for the output resampler would be $R_{out} = 1 / R_{in} = 1 / 1.4 = 0.7143$, with $R_{in} = 1.4$ the ideal input ratio. However, we are interested in the quantization effects on the ratio (signal *T_out_n* in our resampler implementation as presented in section 4.4.3.2 and Eq.(4.8)). We will call $T\_out\_n_{in} = 1 / R_{in}$ and $T\_out\_n_{out} = 1 / R_{out}$. After quantization, the first stage receives hence $T\_out\_n_{in\_q}$ = Quant[$T\_out\_n_{in}$] while the output stage receives $T\_out\_n_{out\_q}$ = Quant[$T\_out\_n_{out}$]. We will call the sampling frequency at the output of the input resampler resulting from $T\_out\_n_{in\_q}$ as f'_{s_q}, in contrast with f'_s that results from $T\_out\_n_{in}$ when there is no quantization error in the ideal case. Note also that the $T\_out\_n_{out\_q}$ value does not include the error in $T\_out\_n_{in\_q}$.

What the regulation loop does is to act on average on the output resampler ratio; it bears in mind the error in $T\_out\_n_{in\_q}$ to properly recover the original sampling rate f_s at the output of the sandwich, $f''_s = f_s$, when the input sampling frequency in the output resampler is f'_{s_q} instead of f'_s. In closed-loop the output resampler hence uses $T\_out\_n_{closedloop}$ that results on average in the inverse of the quantized value fed to the input resampler $T\_out\_n_{closedloop} = 1/ T\_out\_n_{in\_q}$ instead of $T\_out\_n_{out\_q}$.

We can hence compute the error $e_{openloop}$ in our quantized open-loop simulation; $e_{openloop} = T\_out\_n_{out\_q} - T\_out\_n_{closedloop} = 8.9407 \cdot 10^{-10}$. Using this error with Eq.(5.7) we can compute the output sampling frequency $f''_s = 62{,}499{,}999.9601$ sample/s, instead of the supposed $f_s = 62.5 \cdot 10^6$ sample/s, that results in $|\Delta f''_s / f''_s| = 6.3862 \cdot 10^{-10}$.

Continuing the example, if we reconstruct the input 3 MHz tone after processing, using a clock at the exact $f_s = 62.5 \cdot 10^6$ sample/s behind the output of the sandwich, we will get a signal whose frequency is 3,000,000.0019 MHz due to the deviation error in the output sampling frequency. This frequency deviation induces a phase slippage of 90º in 130.5 s and for a simulation lasting 1 ms, the slippage would be $6.8971 \cdot 10^{-4}$ degree, again hardly noticeable in a computer simulation.

As these numbers indicate, we expect a performance at the output of the sandwich similar to the performance with the single resampler. The BSP Architecture will not induce notable further degradation coming from this second stage, the output resampler. This is due to the fact that the main noise contribution, the data-path quantization, is an additive source already considered in the input signal of the input resampler. This means that this noise is not doubled with the second resampler, that uses the same quantized data-path, but filtered a second time with no significative increase due to the already present error with the same order of magnitude. We now present the results that validate our statements.

As we did for the resampler, we have gradually incorporated the errors to our sandwich model; we started studying the unquantized sandwich, that incorporates the $H^{CP}(e^{j\omega},d)$ VFD filters in the $H^{CPR}(e^{j\omega},R)$ resamplers. This BSP sandwich is what we call the Computed Sandwich $H^{CPS}(e^{j\omega},R_{in})$ and it averages different time-domain simulations for a given input ratio-frequency pair $(e^{j\omega},R_{in})$. The input resampler contains only as error the frequency response deviation caused by the computed coefficients of the VFD. Recall that now the output resampler receives that error within its input signal, and it also filters it. This model lets us understand the effect (if any) of the double resampling process, this is, the propagation of the input resampler error and subsequent filtering by the output resampler. We later incorporated the quantization to the model; additive noise sources in the data-path, delay and ratio signals, and also the frequency deviation due to the quantized coefficients. We call this model the Implemented Sandwich $H^{IS}(e^{j\omega},R_{in\_q})$.

We evolve our sandwich model without intermediate processing as a filter that performs the double resampling process. It is depicted in Fig. 5.20. This is again just a conceptual representation and it does not intend to be mathematically rigorous. We define the ideal response of the sandwich $H_{id\_s}(e^{j\omega},R)$ as the one that lets us reconstruct the input signal at the output without error and with the exact same sampling rate, when no processing is performed between resamplers. The added noise in the sandwich is represented by the parallel error filter $e_s(e^{j\omega},R)$, that encompasses the frequency deviation of the VFDs, the quantization of the coefficients (filters $e_{c\_s}(e^{j\omega},R)$ and $e_{qc\_s}(e^{j\omega},R)$ in Fig. 5.20), the quantization of the ratio signals and the data-path, and the filtering by the output resampler of the noises in the output of the first resampler. The resulting implemented filter that models the sandwich, $H_{SND}(e^{j\omega},R_{in})$, results from the combination of the contributions of these two filters, $H_{id\_s}(e^{j\omega},R)$ and $e_s(e^{j\omega},R)$. The response includes averaging of the simulations to include the different variable delays computed based on the ratios.

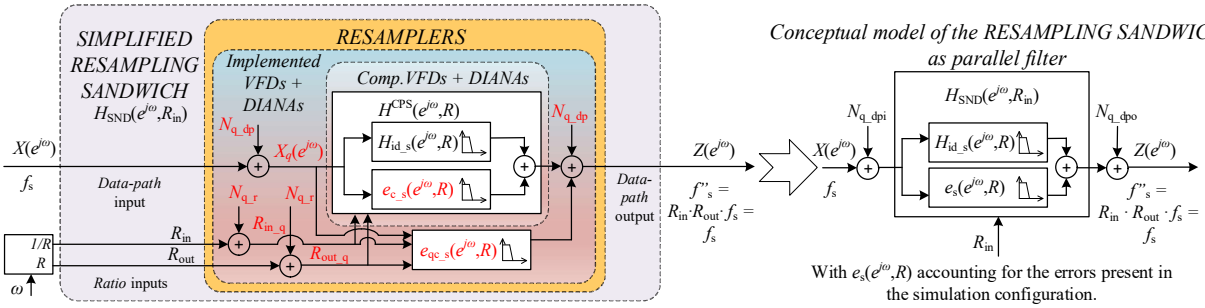


Fig. 5.20. Simplified model of the BSP sandwich Architecture in the study.

This new model let us hence compute the degradation in the signal that is caused by the sandwich; the added noise that comes from the error $e_s(e^{j\omega},R_{in}) = H_{SND}(e^{j\omega},R_{in}) - H_{id\_s}(e^{j\omega},R_{in})$ in the double resampling process. Note that the simulations span the entire first Nyquist zone for the frequency of the input signal $x_\omega[n]$. However, the reader must understand that the accepted input frequencies must span the ranges presented in section 3.3.10 only. In any case, the artefacts that we saw in the resampler are now not present. The double resampling process implicit in the reference signal of the sandwich avoids these folding artifacts. As for the VFD and resampler, we included the noise and error sources in different steps; we first analysed what we called the Computed Sandwich $H_{SND}(e^{j\omega},R_{in}) = H^{CPS}(e^{j\omega},R_{in})$. This sandwich results when

no quantization is present in the data-path nor in the ratio/delay or the VFD coefficients. Then we added quantization and called this the Implemented Sandwich $H_{SND}(e^{j\omega},R_{in}) = H^{IS}(e^{j\omega},R_{in\_q})$

5.3.2.2. *The computed BSP Architecture*

We start presenting the results for the Computed Sandwich $H_{SND}(e^{j\omega},R_{in}) = H^{CPS}(e^{j\omega},R_{in})$. We first depict in Fig. 5.21 the error surface $|e_s(e^{j\omega},R)|$ at the output of this unquantized sandwich configured with down-sampling ratios $R_{in} \in [0.5, 1]$. One of the axes of the response presents the frequency ω at which we evaluate the sandwich while the other depicts the ratio R_{in}. We observe in Fig. 5.21(a) that the sandwich response is now very flat in the optimized pass-band region in contrast to the resampler and VFD. When analysing in detail this optimized region in Fig. 5.21(b), we observe that for each frequency, the error response is independent of the resampling ratio. There is only a slight decrease in the error magnitude when the ratio adopts values close to $R_{in} = 1$. Note that the error for that ratio tends to 0; both resamplers are in transparent mode and there is no resampling in either of the two. Again, in these simulations, the delay value $d = 0$ is predominant in the iterations run with this configuration. It is also the case for $R_{in} = 0.5$ that again presents a predominance of $d = 0$.

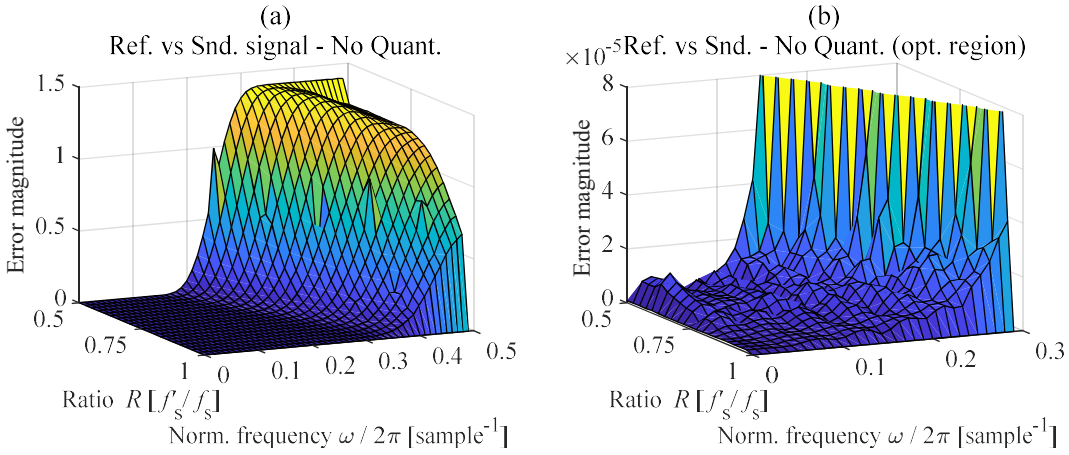


Fig. 5.21. Error function for the computed sandwich, down-sampling ratios in the input resampler. (a) Magnitude of the error function. (b) Zoom in the pass-band region.

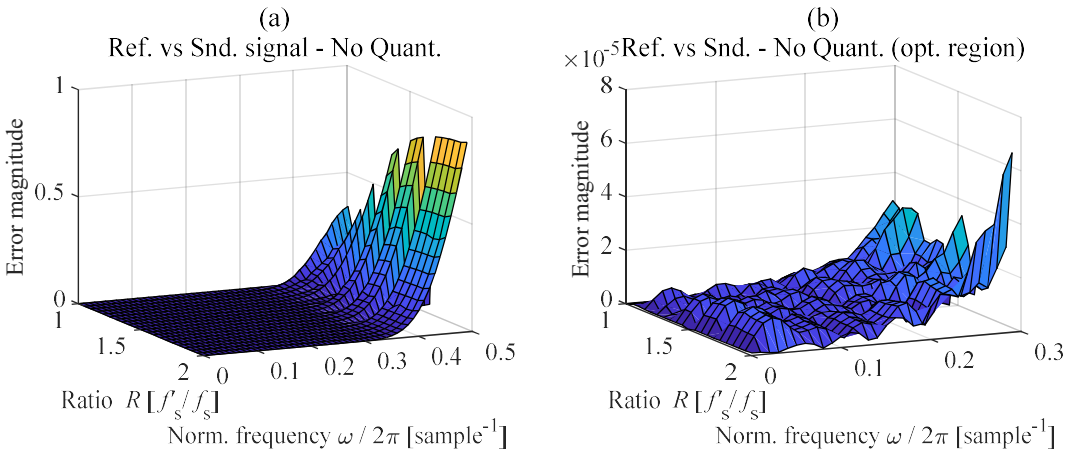


Fig. 5.22. Error function for the computed sandwich, up-sampling ratios in the input resampler. (a) Magnitude of the error function. (b) Zoom in the pass-band region.

The magnitude of the error in the surface is in the same order of magnitude as the one of its VFD, Fig. 5.6, and the resampler, Fig. 5.12, with an average value clearly below $1 \cdot 10^{-5}$ in the optimization, only increasing for frequencies close to the upper input bandwidth of the sandwich. We saw how the resampler error response averages the VFD responses combining different delays for a given ratio. In the present case we combine delays in two resamplers with inverse ratios, this, as the reader will notice, completely smooths the combined response for a given input ratio R_{in} within the sandwich. The oscillations of the error along the frequency axis are now modified. These oscillations are still present in the response of an isolated resampler. However now the normalized frequency of the signal at the input of each of the resamplers is different. This results in a combined response that does not present the previous characteristic oscillatory pattern of the VFD. In any case we can still see some reduced oscillations resulting from the mixed response. Fig. 5.22 depicts the results for up-sampling ratios with identical conclusions.

Fig. 5.23 depicts the achieved SNR surface. The first thing that we clearly observe in the figure is that the SNR is now also flat without any pattern in ratio or frequency axis (the spikes present in the simulation result from the numerical simulation). In the optimized region we achieve a SNR above 90 dB for any ratio, reaching for frequencies below $f < 0.2$ sample^{-1} in up-sampling configurations 110 dB with peaks up to 120 dB. The SNR surfaces for ratios $R = 1$ and $R = 0.5$ tend to infinite as the error tends to 0, and hence are omitted from the plot. The performance of the BSP Architecture is therefore well balanced and with enough spectral purity at the output when we use the $H_{06}(e^{j\omega},d)$ VFD filter in the resamplers. The double processing and resampling do not degrade the performance with respect to a single resampler.

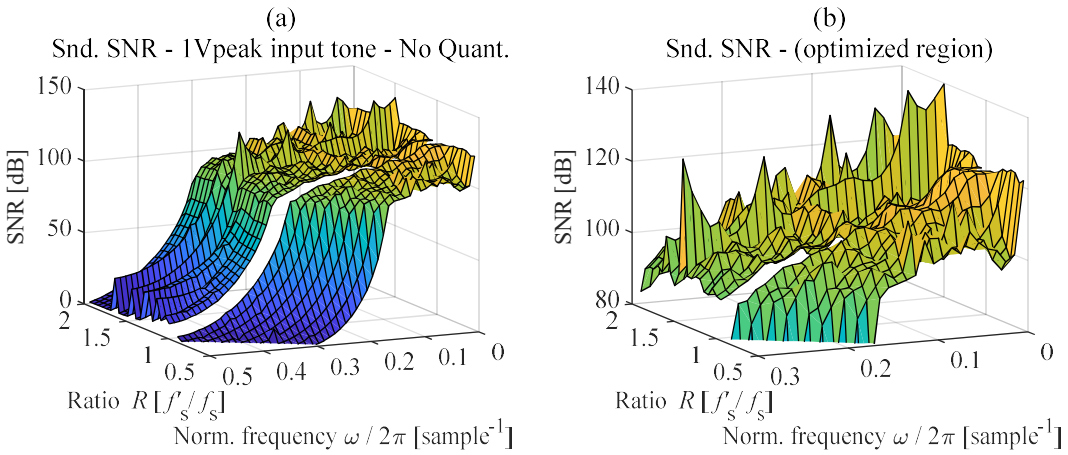


Fig. 5.23. SNR at the output of the computed sandwich, both down-sampling and up-sampling ratios in the input resampler, when excited with a 1 V_{peak} tone. (a) First Nyquist zone. (b) Zoom in the pass-band region.

5.3.2.3. The implemented BSP Architecture

We now present the results of the Implemented BSP Sandwich $H_{SND}(e^{j\omega},R_{in}) = H^{IS}(e^{j\omega},R_{in\_q})$ including quantization noise. The used resamplers have the same fixed-point configuration as the one presented in 5.3.1.3.3. Recall that in the BSP model presented in 5.3.2.1 we computed the sampling frequency deviation error for such a ratio signal implementation using an input value $R_{in} = 1.4$ with a 3 MHz input signal. The sampling frequency deviation at the output of the BSP sandwich resulted in this case $\Delta f''_s = -0.0399$ sample/s and $|\Delta f''_s / f''_s| = 6.3862 \cdot 10^{-10}$. By examining the error response of the computed

sandwich in Fig. 5.22, the deviation resulting from this error seems negligible; the response does not vary very abruptly and the response of the sandwich is qualitatively equivalent. We can hence run the simulations in open-loop and consider the results as acceptable, as long as the simulation time lasts for a few ms, disregarding the influence of this sampling frequency offset and the phase slippage in the processed signal.

We present on Fig. 5.24 and Fig. 5.25 plots with slices of the square of the error magnitude, in the first figure for down-sampling ratios and in the second for up-sampling ratios. These traces are slices along the ratio axis in the left plots and slices along the frequency axis in the right plots. The square error magnitude, that we modelled as the error filter $e_s(e^{j\omega},R)$, adopts a value around $4\cdot10^{-10}$ in the pass-band region; the quantization error is larger than the VFD frequency response error and masks all the other errors and deviations in the implemented sandwich. We do not see the fluctuations in the frequency axis either. Note also that the error with $R = 1$ is now no longer zero, it is instead a value close to the quantization noise of the data-path (the delay value $d = 0$ is still predominant in the different iterations performed). We can clearly observe the accepted input bandwidth of the sandwich (Fig. 3.30 in section 3.3.10); in Fig. 5.24, where we have a down-sampling configuration, we observe how the bandwidth increases as the value of the resampling ratio does. For $R_{in} = 0.5$ the bandwidth reaches $\omega = 2\pi\cdot0.15$, the design limit, and when the ratio reaches $R_{in} = 1$ the bandwidth has increased up to $\omega = 2\pi\cdot0.3$, the design bandwidth for this ratio.

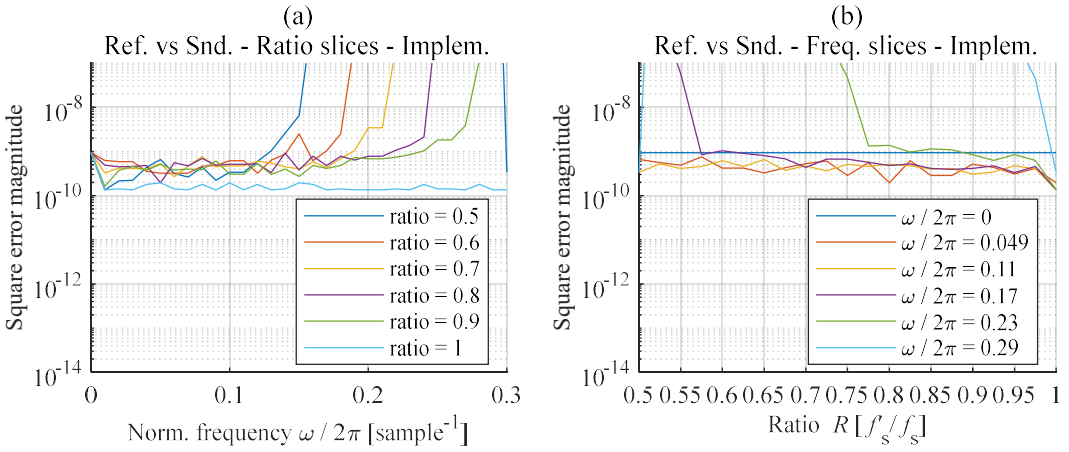


Fig. 5.24. Magnitude of the square error function for the implemented sandwich with down-sampling ratios in the input resampler. Zoom in the pass-band region. (a) Slices along the ratio axis. (b) Slices along the frequency axis.

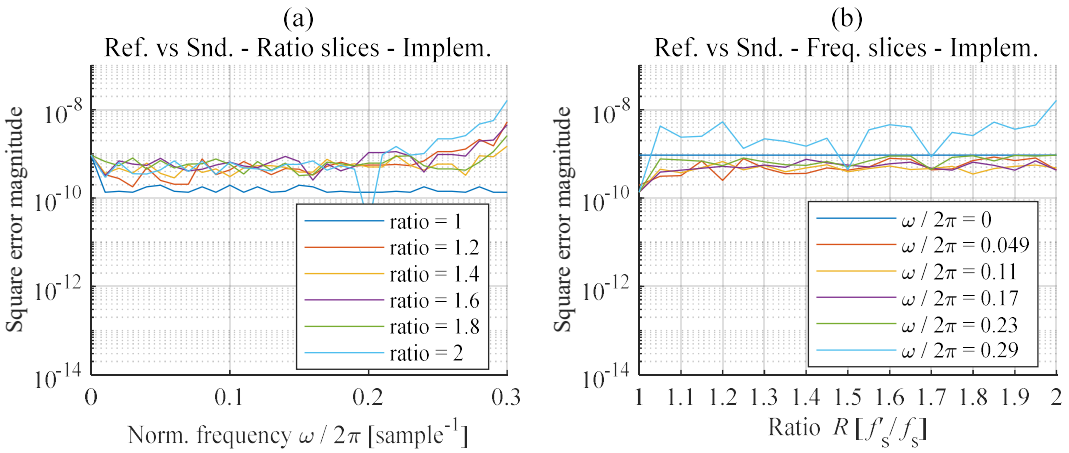


Fig. 5.25. Magnitude of the square error function for the implemented sandwich with up-sampling ratios in the input resampler. Zoom in the pass-band region. (a) Slices along the ratio axis. (b) Slices along the frequency axis.

We finally present the achieved SNR surfaces (full scale input with a $1V_{peak}$ tone) in Fig. 5.26 and Fig. 5.27, the first as a 3D surface and the second as a 2D colour coded surface. In this SNR plot, we can observe that the Implemented Sandwich response remains flat in the bandwidth of the sandwich. The achieved value in the final implemented sandwich remains similar to the one of the VFD and resampler. We observe a flat surface on the order of 93 dB that is very close to the limit SQNR of 98.09 dB of the input data-path. Particularly, the SNR for $R = 1$ adopts a value that matches the SQNR of the data-path. In Fig. 5.27 we can clearly observe that the SNR degrades drastically for frequencies larger than the upper limit of the input bandwidth of the sandwich (section 3.3.10). With this configuration this upper limit is a straight line between $\omega = 2\pi \cdot f = 2\pi \cdot 0.15$ radian/sample for $R_{in} = 0.5$ and $\omega = 2\pi \cdot 0.3$ radian/sample for $R_{in} = 1$. The colour coded spaces denote the regions where the SNR is poor, in dark bluish colours, and where the SNR is large in bright colours.

We can hence conclude that for our VFD coefficient optimization and sandwich implementation the data-path quantization is the major noise contribution in the Architecture. The contribution of the VFD frequency response error, and other quantization errors besides the data-path, have negligible effects. These effects are masked by the latter when using a sixteen-bit data-path. However, the result still reaches a value

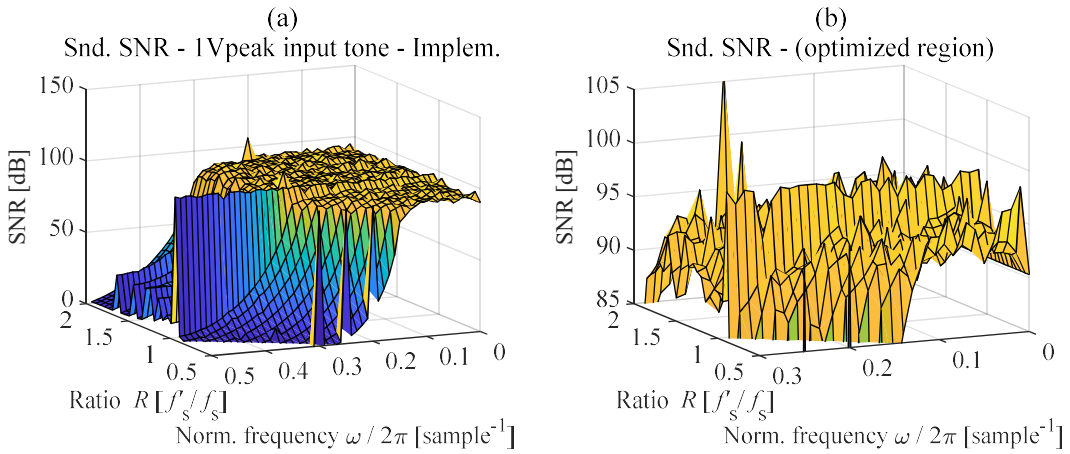


Fig. 5.26. SNR at the output of the implemented sandwich, for both down-sampling and up-sampling ratios in the input resampler, when excited with a 1 V_{peak} tone. (a) First Nyquist zone. (b) Zoom in the pass-band region.

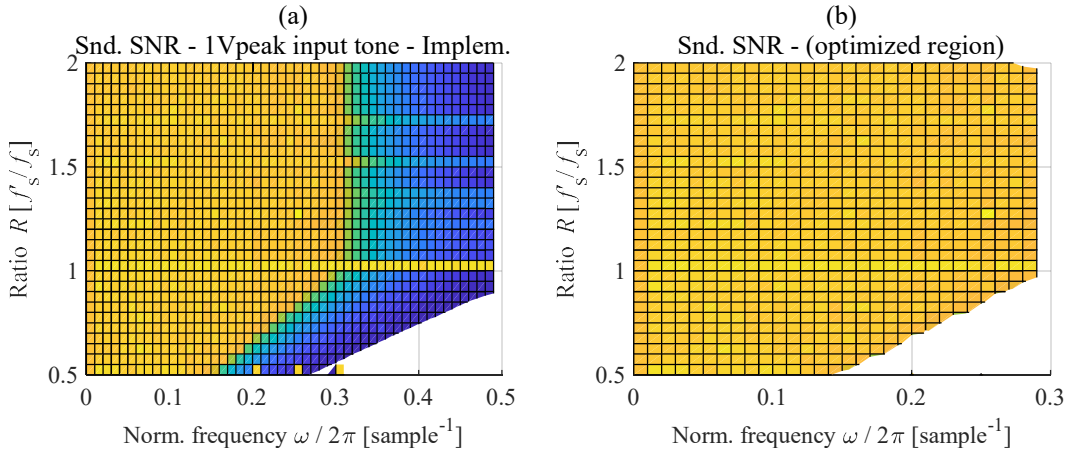


Fig. 5.27. 2D colour coded plot of the SNR surface at the output of the implemented sandwich, for both down-sampling and up-sampling ratios in the input resampler, when excited with a 1 V_{peak} tone. (a) First Nyquist zone. (b) Zoom in the pass-band region.

only 5 dB below the limit SQNR, SNR = 93 dB. This value is largely sufficient for our intended application presented in Chapter 6. The resulting SNR error surface is very flat in the optimized region making the response of the sandwich very uniform.

5.3.2.4. The BSP Architecture optimization

We finally analyse the influence that the FIR filter bank architecture, and the size of the data-path have on the performance of the BSP Architecture. The SNR at the output of the sandwich is the parameter that we use to measure the performance; we study first how it is degraded by different architectures using an unquantized data-path. We have performed time-domain simulations of the sandwich in which we change the VFD architecture; different number of FIR filters in the bank, C, and different number of taps, B, per FIR filter. We use as reference performance the one achieved by the architecture used in the VFD validation (5.3.1.2), six FIR filters in the bank and fifteen taps per FIR. For each one of the simulations in this section, we first modify the architecture, and then we recompute the coefficients of the filter with the WLS method [104]. Finally, we simulate that architecture and analyse how that affects the SNR. We emulate the non-quantized data-path by setting the fixed-point word widths to a large value of fifty-eight bits. The resampling ratio was set to R_{in} = 1.4. We depict traces that span the sandwich input frequency along the first Nyquist zone with the different architectural configurations.

In Fig. 5.28 we first depict the results for the Computed Sandwich. The reference architecture has performance baseline between 110 dB and 100 dB, with only the peaks being softened with respect to the resampler and VFD. The results show that when reducing the number of filters in the bank, C parameter, the cut-off frequency of the sandwich is degraded, and the SNR becomes hence worse for the high frequency components. This is especially relevant when we compare the results against the reference architecture that is configured with the largest number of taps, B = 15; if we reduce the number of filters from six to five, we decrease the cut-off frequency by 33%, and when reducing the number of filters down to four it decreases the frequency by 66%. When we instead reduce the number of taps B, the resulting effect is a degradation of the achieved SNR value, but without significant degradation of the cut-off

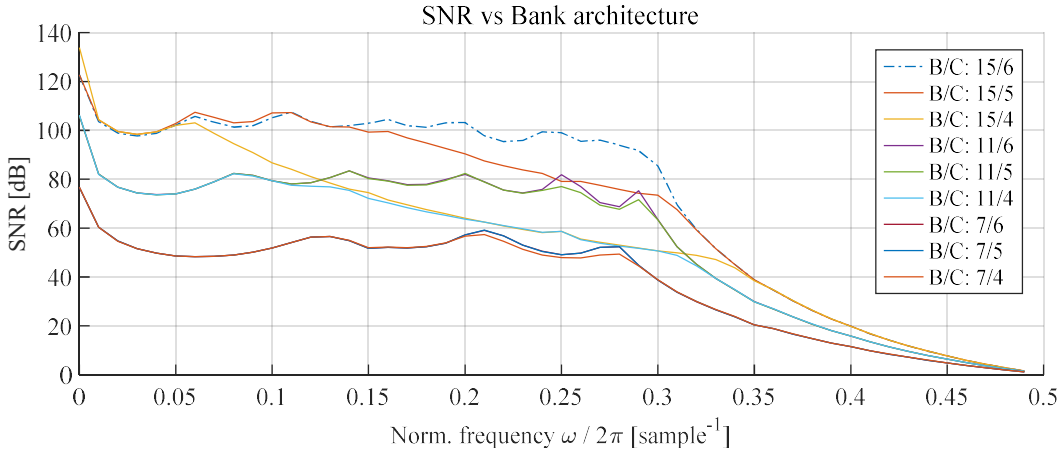


Fig. 5.28. SNR at the output of the BSP sandwich Architecture for α = 0.6 with resampling ratio R_{in} = 1.4. SNR vs filter bank architecture and data-path without quantization error.

frequency. In this case the entire trace lowers the SNR value. We lose around 20 dB when changing the configuration from $B = 15$ to $B = 11$, and additional 20 dB also when going from $B = 11$ to $B = 7$.

We now analyse how the data-path width affects the performance to be able to select a width that offers a trade-off between the resources used and the achieved SNR including the entire model of the sandwich. We have benchmarked the sandwich degradation in terms of output SNR for different data-path widths between twelve bits and twenty bits. The resampler and VFD architecture remains the reference one, a FIR bank composed of six filters with fifteen taps each.

Fig. 5.29 presents the results of the analysis; the maximum reference SNR at the output remains the same of Fig. 5.28 between 110 dB to 100 dB in the optimized pass-band region, when the data-path is not quantized. When the data-path is quantized, the reference SNR trace is achievable only using a fixed-point word of at least twenty bits width. The performance begins to degrade around the periodic maxima using eighteen bits, and for values below this width the quantization error limits the SNR in the bandwidth of the resampler, below $\omega = 2\pi \cdot 0.3$ radian/sample. The purple trace in Fig. 5.29 that implements a data-path with sixteen bits (the value used in all previous studies) is close to the maximum SNR value in the entire sandwich bandwidth without excessive degradation. We can hence confirm that the data-path width used in our simulations, sixteen-bit width, is a good choice that masks the error of the sandwich in its processing bandwidth and offers a SNR value very close to the ideal SQNR of 98.09 dB balancing the hardware resources and resampler SNR performance.

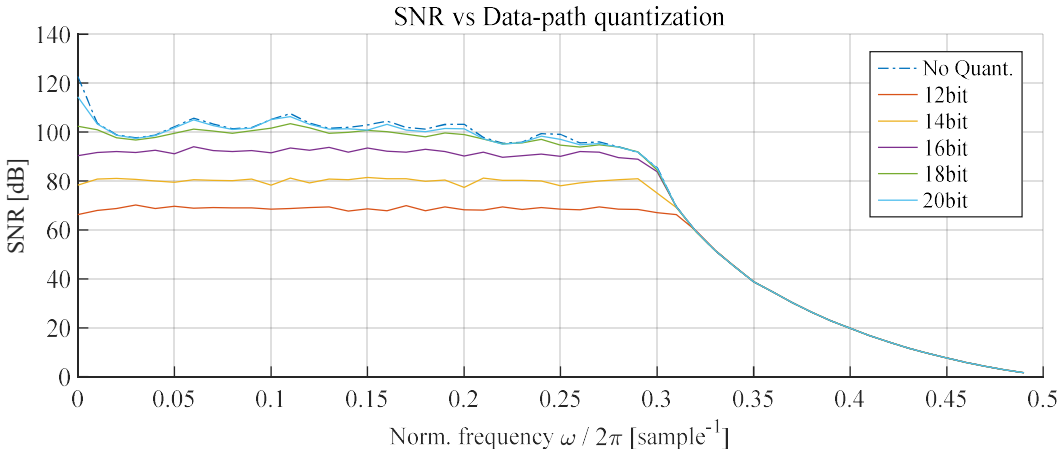


Fig. 5.29. SNR at the output of the BSP sandwich Architecture for $\alpha = 0.6$ with resampling ratio $R_{in} = 1.4$. SNR vs Data-path width and reference architecture, six filters with fifteen taps each.

5.4. Implementation results

In this section we present the hardware implementation results of the principal building blocks of the BSP Architecture. We target FPGA as technology, however they can also be easily implemented in ASICs. We first present the results for the case of a single resampler; the implementation and Place and Route (PAR) has been successfully performed using the Vivado software suite available from the FPGA manufacturer Xilinx. We have selected two devices, one representing the low-cost commercial segment, cheap and

accessible for the general market. The second one is a high-performance FPGA, not affordable for simple applications. The goal is to demonstrate how the resampler can be used as a feasible IP in both segments of the market. The high-grade FPGA is a Xilinx Kintex-7 XCKU040-1FFVA1156C and the small device is a Xilinx Artix-7 XC7A75T-1FGG676. The Kintex Ultrascale device is the FPGA in which the hardware tests of our Architecture will be performed. The resampler has been packed as single IP without any of the *MERCEDES* interfaces. We used a target 125 MHz processing clock for the two devices. The 125 MHz frequency of the clock results from the hardware configuration in which the design will be evaluated in the laboratory, the uTCA card SIS8300-KU [106] from Struck that contains the Kintex Ultrascale FPGA. We will use this card with a system clock recovered from a White Rabbit link.

The System Generator sources map the presented design stressing the readability of the Architecture, even when this might result in a penalty in performance. The VFD architecture uses the configuration presented in this chapter; it contains six FIR sub-filters with fifteen coefficients each. The FIR architecture is folded around the central tap resulting in eight coefficient multipliers per filter. The widths of fixed-point implementation of the signals are as presented above in this chapter; the data-paths at the input and output of the resampler are sixteen bits wide, with fifteen fractional part bits and one sign bit. The FIR filter bank coefficients are twenty-seven bits wide, with twenty-six fractional bits and one sign bit. The data-path is extended to forty-eight bits, and truncated back to twenty bits at the output in the internals of each of the sub-filters. In the internals of the Horner chain the data-path is constrained to twenty-seven bits, with twenty-three fractional bits and one sign bit. The resampling ratio signal of the resampler is a thirty-two bits wide word, with twenty-nine fractional part bits, two integer part bits and one sign bit. The computed delay value is an eighteen bits wide word, with seventeen fractional bits and one sign bit. The low-end device offers DSP48E1 macros instead of the DSP48E2 of the Kintex Ultrascale. The two inputs of the multiplier of the DSP48E1 are eighteen bits and twenty-five bits wide, respectively, while the DSP48E2 supports eighteen bits and twenty-seven bits. For this reason, the FIR filter bank coefficients and the Horner internal data-path have been reduced in the low-end device design to twenty-five bits wide signals, keeping the same integer width and reducing fractional part bits accordingly.

The timing constraints were met for the high-performance FPGA, and the timing report shows that the maximum achievable clock is 171.03 MHz. The achievable clock for the low-end device was 130.2 MHz. The PAR strategy for the synthesis and implementation was “Flow_AreaMultThresholdDSP” and “Area_ExploreWithRemap”, respectively. The first strategy enforces the use of DSP blocks increasing the performance of the results. The latter enforces the optimization of the logic to reduce the used resources. No optimization of the Architecture has been done to increase the achievable frequency. The objective is to proof the feasibility of the design and its implementation, not to optimize the operating frequency. Table 5.1 shows resource utilization after the PAR. The used glue logic and memory requirements are very limited and negligible for both devices. The DSP blocks are mainly consumed by the bank of filters of the VFD; six FIR filters with eight multipliers per filter. The other five DSP blocks are used in the Horner architecture

Table. 5.1 FPGA resource utilization after PAR for a single resampler

	XC7A75T			XCKU040		
Resource	Available	Used	%	Available	Used	%
Slice Registers	94400	4878	5.16	484800	4945	1.02
LUT as Logic	47200	3107	6.8	242400	3285	1.35
LUT as Memories	19000	105	0.55	112800	121	0.1
Block Ram Tiles	105	3.5	3.33	600	3.5	0.58
DSP blocks	180	53	29.4	1920	53	2.76

Table. 5.2 FPGA resource utilization after PAR for the BSP Architecture

	XC7A75T			XCKU040		
Resource	Available	Used	%	Available	Used	%
Slice Registers	94400	12346	13.07	484800	12446	2.56
LUT as Logic	47200	6668	14.12	242400	6745	2.78
LUT as Memories	19000	2499	13.15	112800	2531	2.24
Block Ram Tiles	105	8	7.61	600	8	1.33
DSP blocks	180	116	64.4	1920	112	5.83

multipliers. The DSP consumption reaches 30% in the low-grade FPGA (XC7A75T) that is a considerable value but remains around 3% in the high-performance FPGA (XCKU040).

We then performed the synthesis for the entire BSP Architecture. Now the sandwich contains a *MERCEDES Decouple* interface, an up-sampling resampler, a periodic IIR comb filter, a down-sampling resampler and the *MERCEDES Couple* interface. These blocks have been again packed as single IP. The PAR process has also been successfully completed for the same two FPGAs with the same constraints and IP configuration. The data-path in the *MERCEDES* interfaces has the same dimensions as in the input and output ports of the resampler. The data-path within the processing between resamplers is sixteen bits wide, with fifteen fractional part bits and one sign bit. The periodic IIR comb coefficients have a twenty-seven bits wide word, with twenty-six fractional part bits and one sign bit. The report shows that the maximum achievable clock is 169.5 MHz for the high-grade device and 128.2 MHz for the low-end one. Table 5.2 presents the resource utilization results after the PAR. The used glue logic and memory requirements are again very limited and negligible for the Kintex device. The Artix however reaches around 10% utilization for memory, 15% for logic and around 65% for the DSP blocks. These blocks are mainly consumed by the two resamplers, two times 53 DSP blocks. The other six DSP blocks in the Kintex device are used for data-path scaling in the IIR comb filter and for the ratio Frequency Locked Loop scaling in the *MERCEDES Couple* interface. The Artix implementation requires four additional DSP blocks resulting from the inappropriate dimensioning of the IIR coefficients that does not suit well the DSP48E1 multiplier architecture, twenty-seven bits vs twenty-five bits per multiplier.

5.5. Hardware tests

The Architecture has been validated in real hardware in the laboratory. We placed a filter within the resampler sandwich and observed its frequency response when modifying the resampling ratio; the response tracked the sampling rate in the sandwich. We leave for Chapter 6 a more detailed analysis of the functional behaviour of the BSP Architecture when used for TBLC as in the intended application. Then the performance of the implementation has been analysed looking at the spectral purity of the output when no processing is placed within the BSP sandwich; in this way we observe only the degradation due to the BSP Architecture. We present in this section the results and the measured performance.

5.5.1. The crate

The test setup has been implemented in a uTCA crate. The input and output analog RF front-ends use a Rear Transition Module (RTM) card, the DS8VM1 from Desy/Struck [107]. The sampling and processing are implemented in the SIS8300-KU card [106], also from Struck, that contains the Kintex Ultrascale FPGA presented in 5.4. The system is controlled from a desktop PC via Python scripts. Two channels, the In-phase and in-Quadrature, I/Q, perform the processing in parallel (complex or Cartesian processing). The fixed frequency clock, the analog LOs and the input RF stimulus signals are synthetized by signal generators feeding the crate. The RF is close to 200 MHz, as the one used in the SPS. We require a processing bandwidth of 5 MHz per sideband around the RF, similar to the SPS OTFB requirements. This processing bandwidth is the frequency region that needs to effectively tune the processing in the BSP and the spectral content of the signal. In the input, the front-end performs direct sampling at f_s = 125 MHz and exploits the aliasing of the signal for base-band down-conversion. The frequency of the input RF signal lies in the fourth Nyquist zone, between 187.5 MHz and 250 MHz, and appears folded down in the first Nyquist zone at around 50 MHz. This folded component is digitally down-converted to base-band in the FPGA with a digital mixer, whose LO is variable to keep the demodulated RF in base-band. In the output stage a vector modulator is used to synthetize the RF signal with the processed base-band I/Q pair. First a digital mixer brings the base-band processing to an IF frequency. This mixer uses a variable digital LO synthetized in the FPGA with the information about the RF frequency value, the frequency tuning word. The resulting digital LO adopts a frequency value computed as the difference between the instantaneous value of the RF and the analog LO of the vector modulator. This second analog modulator brings the IF back to the nominal RF value. It uses an analog LO with fixed frequency at 223.5 MHz. The complete implementation contains further processing, however from a functional point of view all that processing is transparent for the test purposes.

5.5.2. The processing

In the digital signal processing stages, after demodulation and conditioning, the *MERCEDES Decouple* interface at the input of the BSP Architecture receives a data stream at $f_s = 62.5 \cdot 10^6$ sample/s. At the output of the interface the data-path remains at the same sampling frequency but the processing clock

operates at f_p =125 MHz, that results from M = 2 in the *MERCEDES* interface. The `valid` line is now present in the data-path that is fed to the resampling sandwich. The sandwich implements the BSP Architecture presented in Chapter 3; it first up-samples the data, then comes the processing (if any) and it finally recovers the original sampling rate. The tuning between the processing and the fundamental frequency is managed by the resampling ratio of the sandwich, that sets a sampling rate within the sandwich at $f_s = R \cdot 62.5 \cdot 10^6$ sample/s. The value of R for both resamplers is controlled within the FPGA that receives the FTW from a White Rabbit link. The resampling sandwich and processing data-path is implemented with the same digital word widths presented in 5.4 for the Kintex Ultrascale device.

5.5.3. Performance tests

We now present the results of the measured performance in the LLRF setup. The parameters used as key performance indicators are the spectral purity and phase noise. Note that these are also dependent of the analog front-end of the LLRF system, but we just want to see if the BSP sandwich degrades the performance. The measurements are performed in the RF domain, including the whole uTCA crate, from RF input to RF output.

In the test setup, the results have been measured with a Spectrum Analyzer (SA). The RF input of the uTCA crate (the antenna input of the LLRF) is connected to a clean synthesizer. The output drive signal of the crate is connected to the SA input. The RF frequency value to which the uTCA crate is tuned is provided to the test setup via a discrete frequency tuning word coming from the White Rabbit link. The synthesiser output, the analyser centre frequency and the uTCA crate are configured and centred at an RF value of 200.2 MHz. We present in Fig. 5.30 the measured spectrum when there is no processing within the resampler sandwich. The span in the analyser is set to 2 MHz in Fig. 5.30(a), and to 100 kHz in Fig.

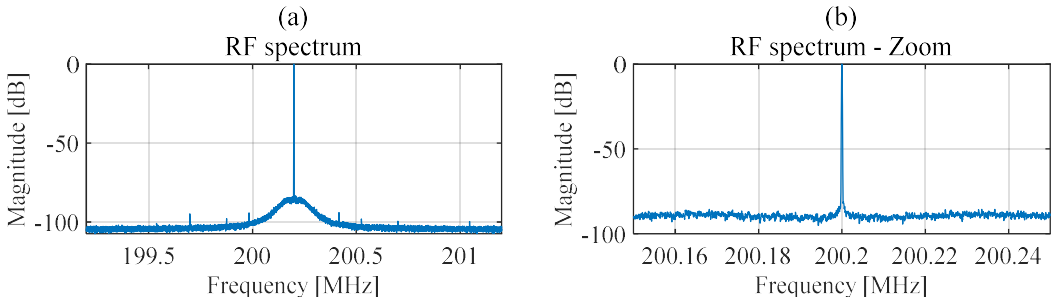


Fig. 5.30. Spectrum of the LLRF drive. BSP and synthesizer tuned to 200.2 MHz. Span of (a) 2 MHz and (b) 100 kHz.

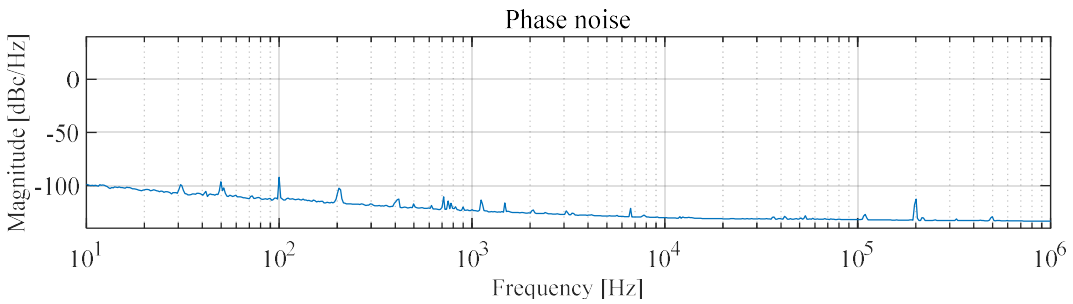


Fig. 5.31. Phase noise measurement. BSP tuned to 200.2 MHz, bandwidth of 1 MHz.

5.30(b). We observe that the noise floor adopts a value around -85 dBm. This measurement shows that there is no significant noise added by the two resamplers.

We now perform a phase noise measurement in the processing chain of this setup. The measurement includes the whole LLRF chain as above. The RF input of the uTCA crate remains connected to a clean synthesizer and the output of the system is measured by a phase noise analyser.

The synthesiser, analyser and the BSP in the uTCA crate remain tuned to a frequency of 200.2 MHz. The results are presented in Fig. 5.31. The covered bandwidth is 1 MHz and the phase noise is -98 dBc/Hz. Similar measurements have been performed exciting the system at different frequencies with the setup on tune with the input signal, and with signals whose frequency is not the one tunned in the BSP, but still within the processing bandwidth (5 MHz). The results were similar with the noise floor remaining at a value around -85 dBm, and with phase noise floor at -98 dBc/Hz, hence validating the RF front-end and system performances.

5.6. Conclusions

The chapter has presented the verification and validation results of the proposed Beam Synchronous Processing Architecture and the elements that compose it, with special attention to the VFD and resampler. In the verification process we have presented the results that show how the functional behaviour and capabilities of the Architecture satisfy the requirements. These needs were defined for the BSP Architecture in Chapter 3 and the resampler in Chapter 4. They match the requirements and give solutions to the problems introduced in Chapter 1 and Chapter 2. The validation process has studied if the performance of the Architecture and its units is sufficient to implement the new One Turn FeedBack control system for Transient Beam Loading Compensation that will be presented in Chapter 6. All the elements of the Architecture successfully passed all the qualitative tests. The VFD has been found as the bottleneck of the Architecture. The ultimate performance of the resampler and the BSP sandwich is related and limited by the performance of this interpolator and its implementation (namely the data-path width).

The *DIANA* engine has proven feasible both as functional solution for the timing unit of a resampler, and qualitatively achieving enough performance to be used in our final application. It computes the required delay for the VFD interpolator and it handles modifications of the ratio in real-time. The errors in the delay signal resulting from the *DIANA* will not induce performance problems to the resampler and the BSP Architecture. We use a fixed-point implementation for the resampling ratio signal with thirty-two bits wide word, twenty-nine fractional part bits, two integer part bits and one sign bit, and for the delay signal with eighteen bits, one sign bit plus seventeen fractional bits.

The functional verification of the VFD has assessed the operation of the filter for the specified input delay range, plus or minus half of an input sampling period. The verification also demonstrated that the VFD clearly distinguishes between positive and negative delay values. This delay value, that can be

modified in real-time, is tracked by the VFD according to the specification. The validation study has analysed the performance of the VFD filter when implemented in fixed-point arithmetic and for different hardware configurations related to architectural decisions. We conclude that the optimization of the filter $H_{06}(e^{j\omega},d)$, that is our target filter for the application presented in Chapter 6, is well balanced when using a data-path with sixteen bits word width, coefficients with twenty-seven bits word width, and a bank architecture composed of six filters and fifteen taps per filter. With these parameters, the SNR at the output of the VFD reaches, for a full-scale input signal, a value around 94 dB in the entire optimized region. The limiting factor is the data-path quantization noise, that is close but above the noise power added by the frequency response deviation of the filter.

The *MERCEDES* interfaces handle the transfer between the *FRANCISCO* fabric and the FPGA fabric. The Architecture is generic in the sense that it can host almost any processing algorithm within the resamplers, and adapt the clocking architecture to the user needs, limited only by technology constraints and not by functional requirements of the Architecture.

The verification of the resampler has assessed the feasibility and suitability of the proposed resampling architecture for BSP processing. The *DIANA* engine, together with the VFD filter and the synchronization elements fulfil the need for a resampling solution where the resampling ratio can be modified in real-time, while accepting different input and output sampling rates. It performs according to the specification. The performance of the resampler is limited by the performance of the interpolating unit, the VFD, that depends on the computed coefficients and the used data-path width. With the presented implementation we achieve a flat surface with SNR = 95 dB. This limit is achieved with the sixteen bits VFD. The achieved SNR is suitable for the intended application detailed in Chapter 6.

The BSP Architecture has demonstrated its feasibility as a functional solution for the processing of periodic signals with known but possibly varying fundamental frequency. The resampler sandwich can tune the representation of the processed signal (in the discrete domain) to the desired normalized frequency in real-time. For this, the input resampler modifies the sampling rate of the input sequence of samples. It does it by using a fixed frequency processing clock. The validation study has analysed the performance of the BSP sandwich Architecture for different hardware configurations related to architectural decisions in the VFD and the resampler. With the presented fixed-point implementation for the resampler sandwich we achieve a flat SNR surface in the order of 93 dB, very close to the SQNR. Again, the limiting factor is the data-path quantization noise. The frequency deviation resulting from the quantization in the ratio signal is negligible from a qualitative point of view for our BSP Architecture. However, if the exact sampling frequency at the output of the resampler is of importance, a *JOAQUINA* inspired feedback loop can solve the issue. From a functional point of view, this feedback loop is not necessary for a single resampler, but mandatory in the BSP Architecture that uses two resamplers in a sandwich configuration. It deals with the sampling frequency deviation that can misconfigure the resampling sandwich.

The proposed Architecture has also been tested in real hardware. A laboratory test setup has verified and validated the functionality and performance of the BSP Architecture implemented on a uTCA platform. The implementation successfully passed the Place and Rout process and was mapped to a high performance FPGA. No processing was placed within the sandwich; the objective was to validate the degradation in the resampling chain. The noise floor in the uTCA implementation went down to -85 dBm and the measured phase noise was -98 dBc/Hz.

Chapter 6

Beam Synchronous Processing Architecture for Transient Beam Loading Compensation in the CERN SPS Accelerator

Abstract: *This Chapter presents the results of the verification and validation of the BSP Architecture when used to implement a One Turn FeedBack for Transient Beam Loading Compensation. We first present the implemented OTFB, then the simulations verifying the proposed Architecture and finally the validation of the results in hardware tests in a real CERN SPS cavity.*

6.1. Introduction

The CERN LLRF system in the SPS synchrotron has historically used a master-slave architecture clocking the digital electronics with a reference synchronous with the RF [10]. This sweeping clock locked the processing to the beam by design. A classic example of this method is the TBLC by means of the One Turn FeedBack algorithm installed in the SPS in the mid-1980s [15]. This algorithm implements the feedback loop filtering the revolution frequency harmonics and an exact one turn delay in the processing path. The variable sampling period resulting from the swept clock automatically tunes the frequency response of the filter and the one turn delay to the beam velocity. With the new fixed frequency clock extracted from the WR, a novel solution is required for TBLC, and in general for BSP, which reconfigures the processing as a function of the beam energy. This chapter presents the application of the BSP Architecture presented in Chapter 3 to satisfy these needs; the chapter proposes a new implementation of the OTFB by means of the BSP Architecture using a fixed frequency clocking scheme. We present the verification results of the solution with simulations and the validation in real hardware. This architectural solution automatically tunes the processing response to the beam parameters (as the swept clock did in the original solution) and avoids

the real-time reconfiguration of the processing elements (solution employed in other engineering fields for variable frequency responses) [5], [37], [38]. The solution is based on the *resampling sandwich* of Chapter 3 in which the BSP is encapsulated. The Architecture is targeted at FPGA technologies and the solution has been implemented in a Xilinx Kintex-7 FPGA as proof of concept.

The chapter is structured as follows: We first review the characteristics of the SPS accelerator, and we depict the proposed LLRF system at high-level. We present the conceptual network architecture that is now based on deterministic protocols. Then we show the partitioning of the LLRF system; we present at logical level the different modules that compose it, and then at physical level we analyse the architecture of the nodes and stations. We continue with the description of the OTFB implementation making use of the BSP Architecture. Then we show the results of the functional verification with simulations of the LLRF system. Finally, we present the measured results in the test-stand mimicking the LLRF system with one SPS cavity.

6.2. The SPS

The Super Proton Synchrotron, switched on in 1976, is the second largest accelerator in the CERN complex injecting beams to the LHC. It has a circumference of nearly seven kilometres, with more than a thousand magnets and lies between France and Switzerland, Fig. 6.1. In his early years, the SPS beams probed the inner structure of protons and investigated about antimatter and exotic forms of matter. In 1983 running as a proton-antiproton collider it made possible the discovery of W and Z particles [108]. The SPS operates at up to 450 GeV, and it has handled many different kinds of particles: Sulphur and oxygen nuclei, electrons, positrons, protons and antiprotons [109].

6.2.1. The RF and LLRF systems

The protons arriving in the SPS have very relativistic energy levels (26 GeV). The SPS accelerates them further to 450 GeV; the increase in speed is small, but still relevant to the RF system. This system uses a swept RF frequency harmonic to the revolution period to cope with the change in speed. The revolution period T_{rev} of the SPS beams results in approximately 23 μs, and the revolution frequency F_{rev} is

Fig. 6.1. Aerial view of the CERN SPS layout (left), and BA3 location detail in the tunnel hosting the SPS RF cavities (right).

approximately of 43 kHz. The RF system is composed of two types of TWC cavities that operate at 200 MHz and 800 MHz. Each group of cavities has an associated sub-system [18]. These frequencies result from the use of harmonic number $h = 4620$ in the 200 MHz sub-system, that is the main RF sub-system. It is responsible for the acceleration and used with all types of beams. Before the LIU-SPS upgrade this sub-system was composed of four cavities, two of them with five-sections and two with four-sections. After the upgrade, it will contain four three-sections cavities and two four-sections cavities [12], [14]. The second 800 MHz sub-system is used for beam stabilization against longitudinal single- and multi-bunch instabilities. It is composed of two cavities with three-sections [12], [14]. Both configurations are depicted in Fig. 6.2, where we present on the left the old configuration, and on the right the new layout after the LIU upgrade. All cavities are located in the same machine straight section of the accelerator within the SPS tunnel, the BA3 location in Fig. 6.1. The transmitters and the LLRF system are also placed in that location but on the surface.

The LLRF system has been upgraded several times. The original LLRF system was commissioned in 1975 using NIM modules. The SPS operated with low beam intensities at that time and there was no schema or algorithm to compensate beam loading. This original system was composed by a cavity controller based on narrowband feedback amplitude/phase loops, that were implemented at a 10.7 MHz IF, a technology common at the time in the field of Radio-Transmission. In the 1980s the beam proton intensity was increased and the first problems with beam loading appeared. As a solution, the first OTFB [10] was proposed in a dedicated 6U Europa crate in 1984. In the early 1990s the SPS accelerated *Pb ions* and other species. The Fixed Frequency Acceleration (FFA) [110] system was then implemented in analog NIM modules to extend the accelerating frequency range of the SPS to lower frequencies. One significant machine development step came in the late 1990s with the preparation of the SPS as LHC injector. This upgrade comprised among others objectives the mitigation of longitudinal Coupled-Bunch instabilities. A new feedforward system and a longitudinal damper were added [50], and the OTFB was also upgraded reducing the cavity impedance at the fundamental. In the early 1980s an 800 MHz Landau system had been developed. It was also upgraded in the late 1990s. All these systems were developed in dedicated 6U Europa crates. In the early 2000s there was an upgrade of the SPS beam control for transfers to the LHC. Some of the electronic modules developed for the LHC LLRF were re-used and new ones were developed based on a VME platform. Lately, in 2015 the 800 MHz Cavity Controller was upgraded to the VME platform. The launch of the SPS LLRF upgrade as a part of the LIU project was started and the first functional specifications were released. The complete renovation of LLRF was confirmed in 2017; it will adopt an

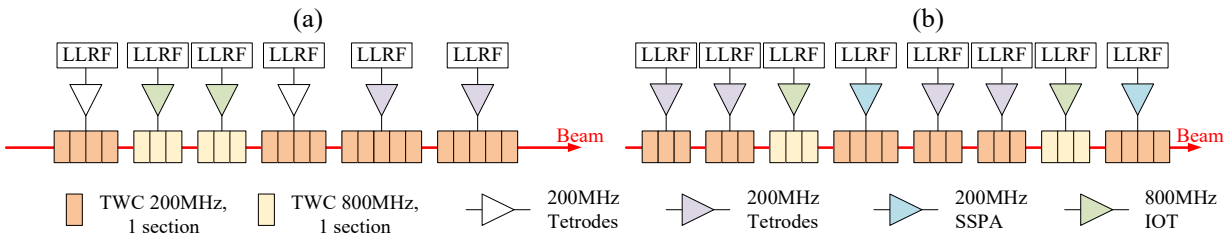


Fig. 6.2. Schematic representation of the SPS RF System; (a) prior to LIU SPS upgrade, (b) after the LIU SPS upgrade. Reproduced from [18].

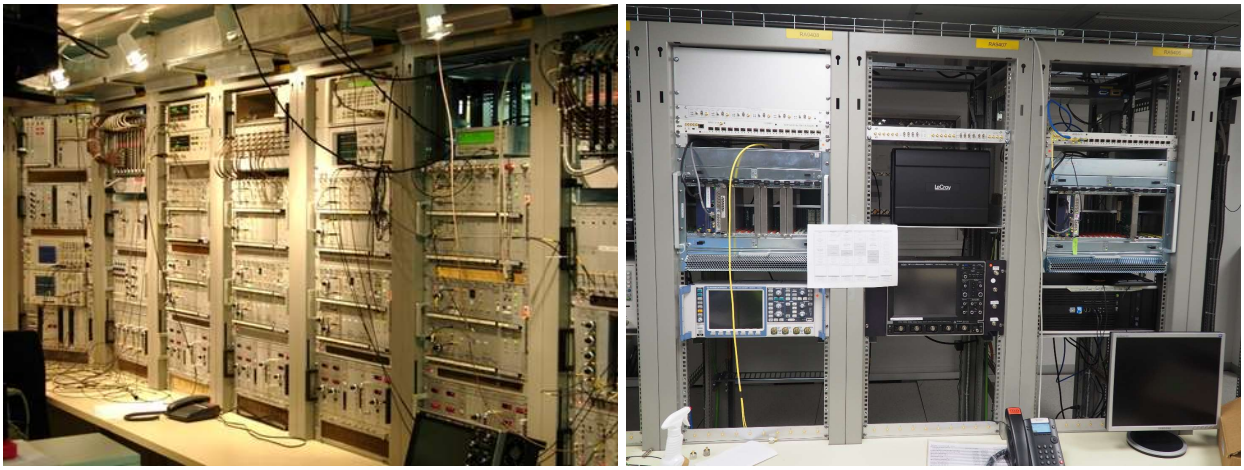


Fig. 6.3. Image of the SPS 200 MHz LLRF system in the Faraday cage: the pre-LIU SPS upgrade configuration (2018) (left), and the new configuration after the LIU SPS upgrade (right).

alternative synchronization method based on deterministic protocols using COTS as much as possible [111]. We present on the left-hand side of Fig. 6.3 some of the VME controllers of the LLRF prior to the LIU SPS upgrade; with that technology and the old NIM modules several crates were required for the regulation. The new uTCA crates, introduced in Chapter 5 in which we tested our Architecture are depicted on the right-hand side of Fig. 6.3; one of the crates handles now 6 Cavity Controllers and the second hosts the Beam Control module.

6.2.2. The SPS 200 MHz TWC cavity

The TWC cavities of the SPS, as the one of Fig. 6.4, are waveguides loaded with stems and drift tubes in order to reduce the phase velocity of the accelerating mode, and terminated into a matched load [13]. The beam crosses these TWC cavities along its longitudinal axis, and receives energy from a wave fed by the RF transmitter that propagates in the structure. The SPS cavity was designed to have a centre frequency of 200.222 MHz. Depending on the beam, the synchrotron ramps the RF from 199.950 MHz at injection up to 200.395 MHz at extraction to the LHC. The phase velocity of the RF wave and the speed of the particles crossing the cavity are only identical when the RF matches the centre frequency of the cavity. At other frequencies the different velocities result in some slippage, and hence less effective accelerating voltage [13], [50]. This effective accelerating voltage V seen by the beam upon one traversal of a n-cell TWC, is the sum of the contributions from all cells. Similarly, the beam-induced voltage must be summed on all cells. The result is not trivial as it strongly depends on the phase slip between the beam and the accelerating wave travelling in the cavity [13]. Let I_g and I_b be the generator current and beam current respectively, the effective voltage $V(F)$ for excitation at frequency F is given by

$$V(F) = Z_g(F) I_g(F) + Z_b(F) I_b(F) \qquad \text{Eq.(6.1)}$$

In a standing wave cavity, the two impedances, Z_g being the impedance seen by the generator, and Z_b being the impedance seen by the beam, are proportional [112]. This is not the case for these TWCs. For the generator-induced accelerating voltage we have the impedance

Fig. 6.4. CERN SPS travelling wave cavity in BAF3 test-stand during the Long Shutdown 2, before installation in tunnel.

$$Z_g(F) = R_1 \left\{ \mathrm{sinc}\left[\tau(F-F_0)\right] e^{-j\pi\tau(F-F_0)} + \mathrm{sinc}\left[\tau(F+F_0)\right] e^{-j\pi\tau(F+F_0)} \right\} \qquad \text{Eq.(6.2)}$$

For the beam-induced voltage the impedance becomes

$$\begin{aligned} Z_b(F) = &-R_2 \left\{ \mathrm{sinc}^2\left[\tau(F-F_0)\right] + \mathrm{sinc}^2\left[\tau(F+F_0)\right] \right\} \\ &+ jR_2 \left\{ \frac{1-\mathrm{sinc}\left[2\tau(F-F_0)\right]}{\pi\tau(F-F_0)} + \frac{1-\mathrm{sinc}\left[2\tau(F+F_0)\right]}{\pi\tau(F+F_0)} \right\} \end{aligned} \qquad \text{Eq.(6.3)}$$

In Eq.(6.2) and Eq.(6.3) F_0 is the cavity centre frequency, τ is the cavity filling time, and the sinc(x) function is the normalized sinc(x) defined as in Eq.(4.14). For additional details, refer to [112]. The two impedances are plotted on Fig. 6.5 around the central frequency of the cavity; in the left, Fig. 6.5(a), we find the generator impedance Z_g, and in the right, Fig. 6.5(b), the beam impedance Z_b. After compensation for the delay $\tau/2$, the impedance Z_g is purely real but it presents zeros and sign inversions. This is caused by the slippage between particle velocity and wave velocity in the structure. At the first zero the slippage results in a $2\pi/n$ phase shift of the voltage seen by the particle in two consecutive cells resulting in zero total voltage over the n-cell structure. The series impedance of the accelerating structure is 27.1 kOhm/m^2, the generator impedance of the cavity at its central frequency is 13.2 kOhm and the beam loading impedance of the cavity at its central frequency is 876 kOhm [13], [50]. On the SPS cavities we have one antenna per cell. A passive RF summing network (rectangular box visible on Fig. 6.4) adds the individual contributions with delays inserted (orange cables), equal to the particle transit time. The result is a measurement of the effective voltage $V(F)$, Eq.(6.1), for the LLRF [10].

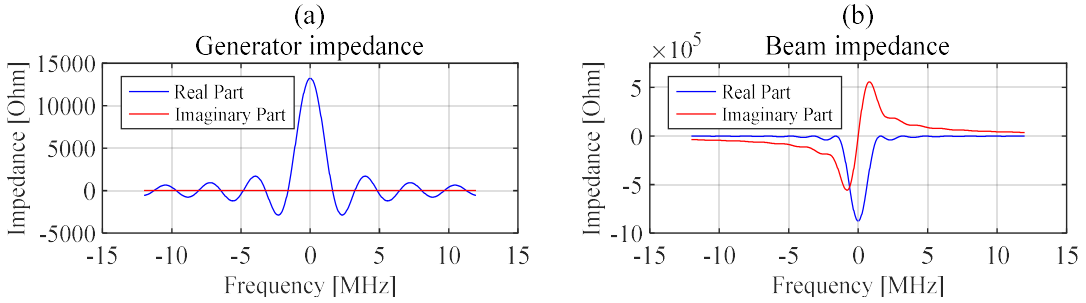


Fig. 6.5. (a) Impedance Z_g (real part in blue and imaginary part in red) after compensation of the $\tau/2$ delay, around the central frequency of the cavity. (b) Impedance Z_b (real part in blue and imaginary part in red). Four-section cavity.

6.3. Synchronization and fixed-frequency clocks

The LLRF system that implements the BSP Architecture of Chapter 3 results from the two paradigm changes that we incorporate: Distributed topology and fixed frequency system clock. In the presented system the deterministic protocol, White Rabbit in the SPS case, is the enabling element for the LLRF synchronization between machines, nodes and sub-systems at accelerator complex level. The BSP Architecture presented in Chapter 3 is in the other hand what enables and extends the life of the OTFB algorithm with a new implementation that now uses a fixed frequency processing clock, at node level. We now present the main concepts and elements (for which we have developed a simulation model) of this LLRF system with emphasis around the BSP Architecture for TBLC.

6.3.1. The synchronization, and the distribution of clocks and data

Traditional LLRF systems rely on dedicated distribution systems (coaxial cables or fibres) to transmit the RF frequency of an analog RF signal to the various stations, and for fine synchronization [22]. Our test system for BSP relies nevertheless on new distributed topologies that have been adopted at CERN for data-broadcasting and synchronization [24], [55], [113]. The use of White Rabbit [25] in our network architecture motivates the need for our BSP solution to treat beam related signals.

These deterministic protocols ensure known propagation delays between nodes easing synchronization schemes; they calibrate the propagation time of the different paths at initialization of the

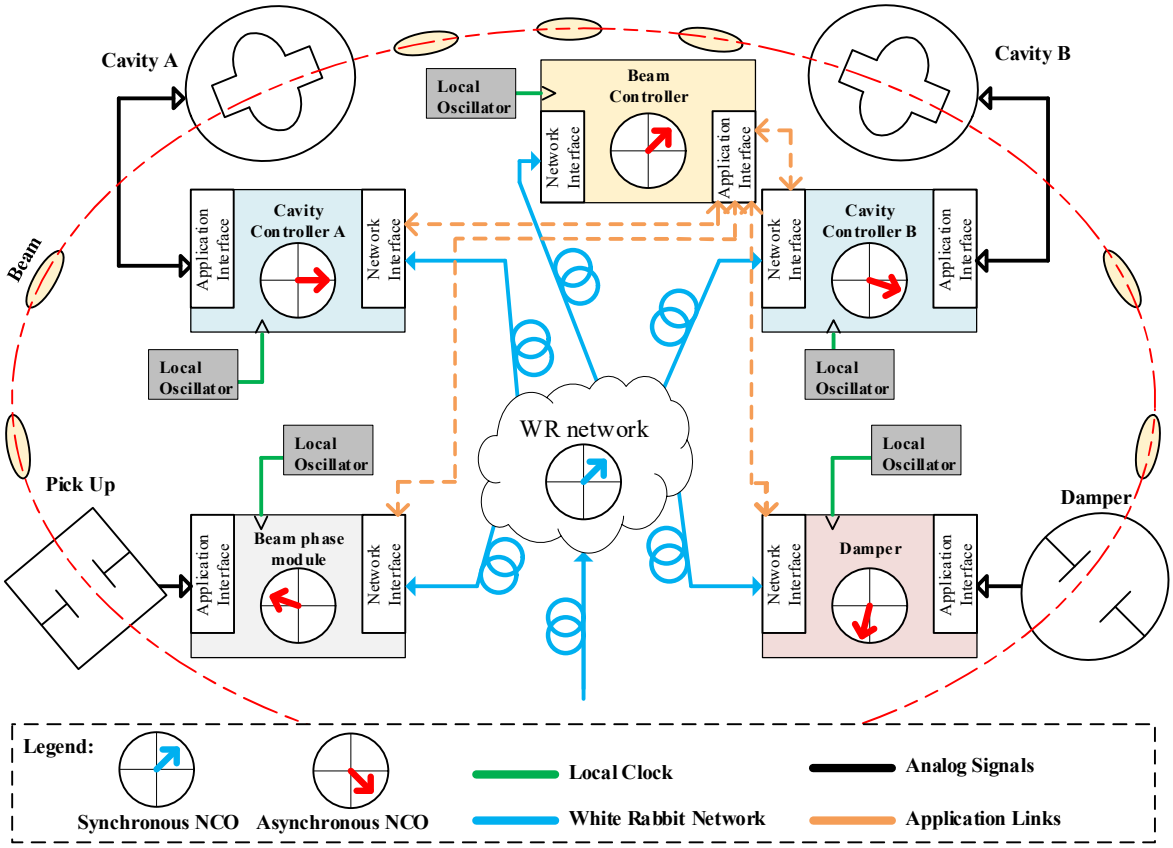


Fig. 6.6. Schematic representation of the network architecture in a synchrotron that uses White Rabbit for synchronization. The nodes use local free running oscillators for the clocking of the hardware.

system. Thanks to this, it is possible to ensure a certain time precision and accuracy among all the nodes of the network. This time certainty makes the information and reference signals to arrive at the same absolute time instant to all the nodes. Fig. 6.6 is a simplified schema of the different elements and systems in a synchrotron. These are implemented in the figure in dedicated nodes and sub-systems that are synchronized at accelerator level thanks to the deterministic network. Note however that the local computations, for instance, the phase of a signal between different nodes, are not yet synchronous. Each node runs with its own local oscillator and hence the initial conditions among them are random.

The problem is not new [24], [55]; take the case for instance of two Direct Digital Synthesis (DDS) synthesizers that regenerate the RF in two distant cavities. Its internals are schematically depicted in Fig. 6.7, and contain a phase accumulator, a look-up table converting phase to amplitude and a DAC. All these synthesizers use as control parameter a Frequency Tuning Word (FTW), and an optional instantaneous phase offset φ_i. They are digital systems and as such they need a system clock. The phase and frequency tuning word of the phase accumulator remain synchronous among all the nodes of the accelerator, this is ensured by WR. Unfortunately, the clocking schema in Fig. 6.6, which uses local versions of free running oscillators, is not locked among the different nodes. In this case, these different DDSs run hence asynchronously, making it impossible to have the phase of the RF signal aligned between nodes [24], [113].

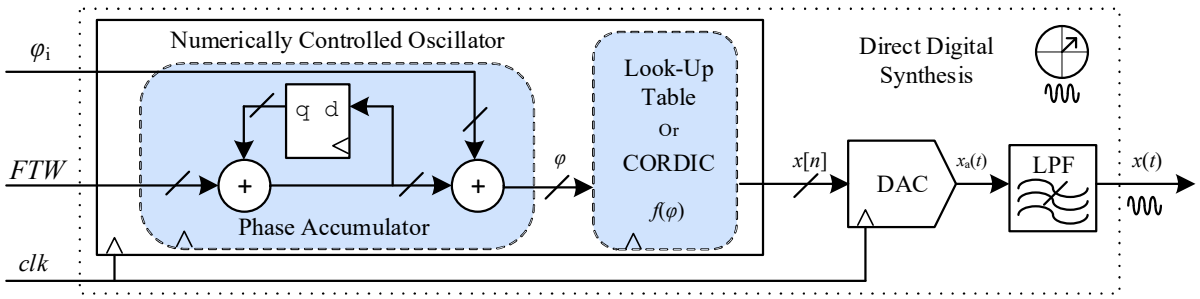


Fig. 6.7. Schematic representation of a signal synthesizer based on Direct Digital Synthesis.

To cope with this, WR provides a fixed frequency system clock extracted from the data stream. The concept was presented in the schematic representation of Fig. 1.1 where each node includes a unit responsible for the WR clock recovery. The clock is hence extracted from the same single fibre used for data distribution and synchronization. The absolute time reference and control signals synchronized among nodes, and the fixed frequency clock that is now also locked among all the nodes, make it now possible that different systems or accelerators can be played in phase; during filling, acceleration and transfer different sub-systems or machines are now in sync.

All these elements can be located in different places thanks to the WR determinism. This makes the acquisition of synchronous data at distant points of the accelerator possible. It also enables the distributed processing and/or control of the accelerator elements and parameters. We can now place cavities in distant places that will synchronously accelerate the beam. Although this is not relevant to the SPS (with all cavities in the same machine straight section), it can be very attractive for the Future CERN Collider (FCC) with a one-hundred kilometre circumference and cavities in two opposite locations [21]. This

distributed control is not new, similar approaches are implemented at GSI [30] and BNL [60], however there the fixed clock is transmitted on a separated dedicated link from the data and timings, whereas WR avoids that limitation at CERN. However, now the clocking architecture is asynchronous to the beam; we use a fixed-frequency system clock. This brings the new problem that the use of deterministic protocols introduces; how to perform the BSP.

This is solved by our BSP Architecture; we can now use the fixed frequency clock derived from the WR for the sampling and system clocks, while the resampler sandwich in the Architecture performs the tuning of the processing to the beam. We resample the data acquired with the fixed frequency periodic sampling to a variable rate that is proportional to the beam revolution period and hence beam synchronous. We also exploit the synchronized arrival of the FTW and references to the nodes to locally compute the required resampling ratio of the sandwich. The BSP Architecture of Chapter 3 suits seamless this distributed network topology with fixed frequency system clock.

6.3.2. Node and station hardware architecture

At physical level, the nodes of the network refer to the locations in the accelerator that contain one or more dedicated stations. Each station is a physical hardware element that hosts one or more functional modules. We use the uTCA standard for implementation of the hardware stations [19]. These stations are composed of a crate manager computer and one or multiple processing platforms. In the uTCA standard these processing platforms are the so-called Advanced Mezzanine Cards (AMC) that commonly implement as processing devices FPGAs. The station has also communication interfaces. These interfaces are, for instance, the Rear Transition Modules (RTM) in the uTCA standard, that are application specific front-ends dealing with the node antennas, pick-ups, dampers, cavities… The uTCA crate backplane serves also as interface; in this case, it links different AMC and RTM cards among them, and also with the network interfaces (White Rabbit and the system links).

We depict in Fig. 6.8 a schematic representation of the interrelations and partition of the elements presented so far in an uTCA station that could host our OTFB implementation. We focus on the AMC card, the FPGA, and the processing as it is the element that hosts our BSP Architecture. This FPGA contains the digital region of two RF front-ends, the application interfaces. The first front-end acquires signals from the accelerator plant, and the second front-end regenerates the output signals that are fed back to the accelerator after processing. The figure depicts also the analog part of these two application interfaces implemented in two different RTM cards of the uTCA crate. In the FPGA, we can see also the different units performing housekeeping tasks, and implementing network interfaces dealing with both the system links and the WR deterministic network. The core of the FPGA in the AMC card hosts the BSP processing enabled by the WR fixed clock and the FTW received via the deterministic protocol. The uTCA backplanes communicate the different elements of the station besides the direct connections they might have between them, as for instance between RTMs and AMC.

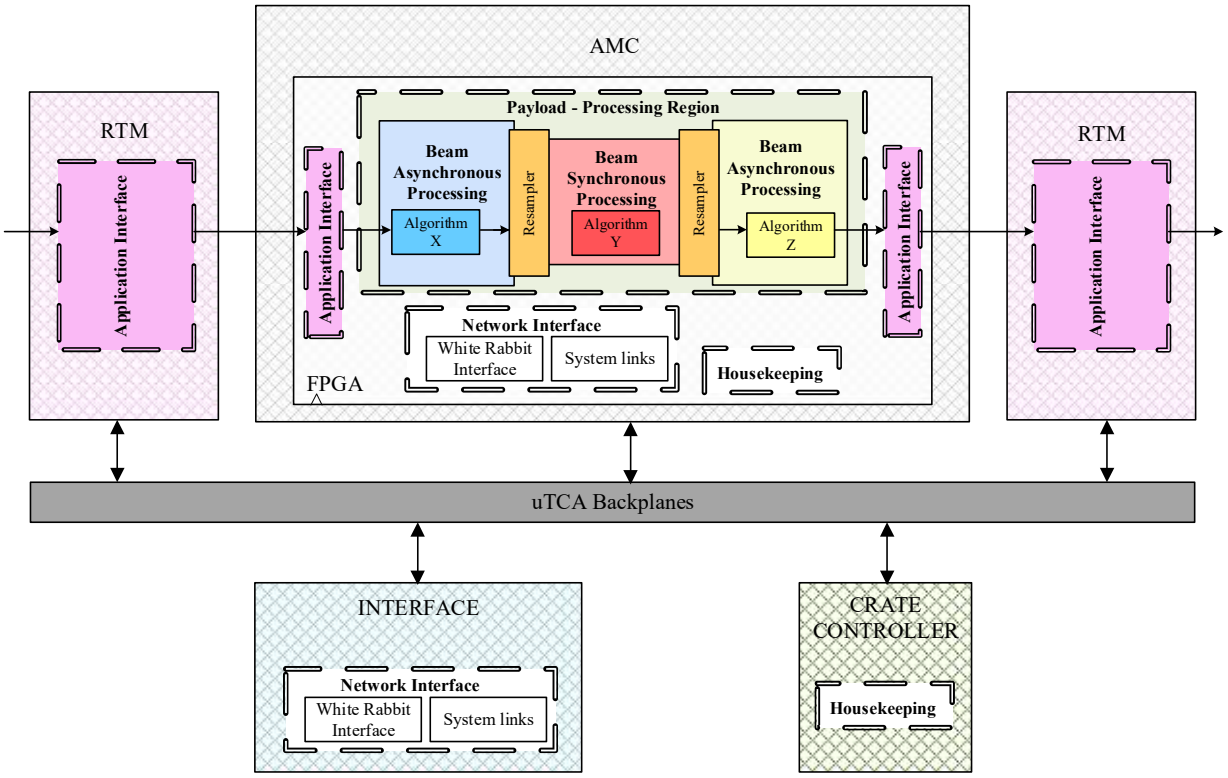


Fig. 6.8. Schematic representation of a uTCA station.

6.4. The BSP Architecture implementing the OTFB

The OTFB is the LLRF feedback loop selected to demonstrate the feasibility of the proposed BSP Architecture within the CERN SPS LLRF. This algorithm is implemented in the Cavity Controller of the LLRF system to deal with TBLC. In the SPS case it is present in the 200 MHz and 800 MHz sub-systems, but we focus only in the 200 MHz in this chapter. This section presents its implementation within the LLRF architecture [114].

6.4.1. The SPS 200 MHz LLRF System, Beam and Cavity Controllers

The LLRF logical schema of the SPS in the post-LIU project is presented in Fig. 6.9 [18], [115]. The system is composed of an RF-Synchro module, a Beam Phase module, a Radial Position module and six Cavity Controllers for the 200 MHz sub-system plus for Cavity Controllers for the 800 MHz sub-system. It contains also WR network infrastructure and receivers.

The module responsible for the acceleration is the Beam-Control. This module computes the RF frequency of the accelerating field based on the measurements of the magnetic field in the bending magnets of the accelerator. This magnetic field value is sent to the module over the WR network. The module is implemented in an AMC card within an uTCA station. This module also receives the cavity voltage measurements, the bunch-by-bunch phase, and the intensity and radial position measurements. Based on this information, the module also computes corrections to the RF that is widespread to the LLRF network

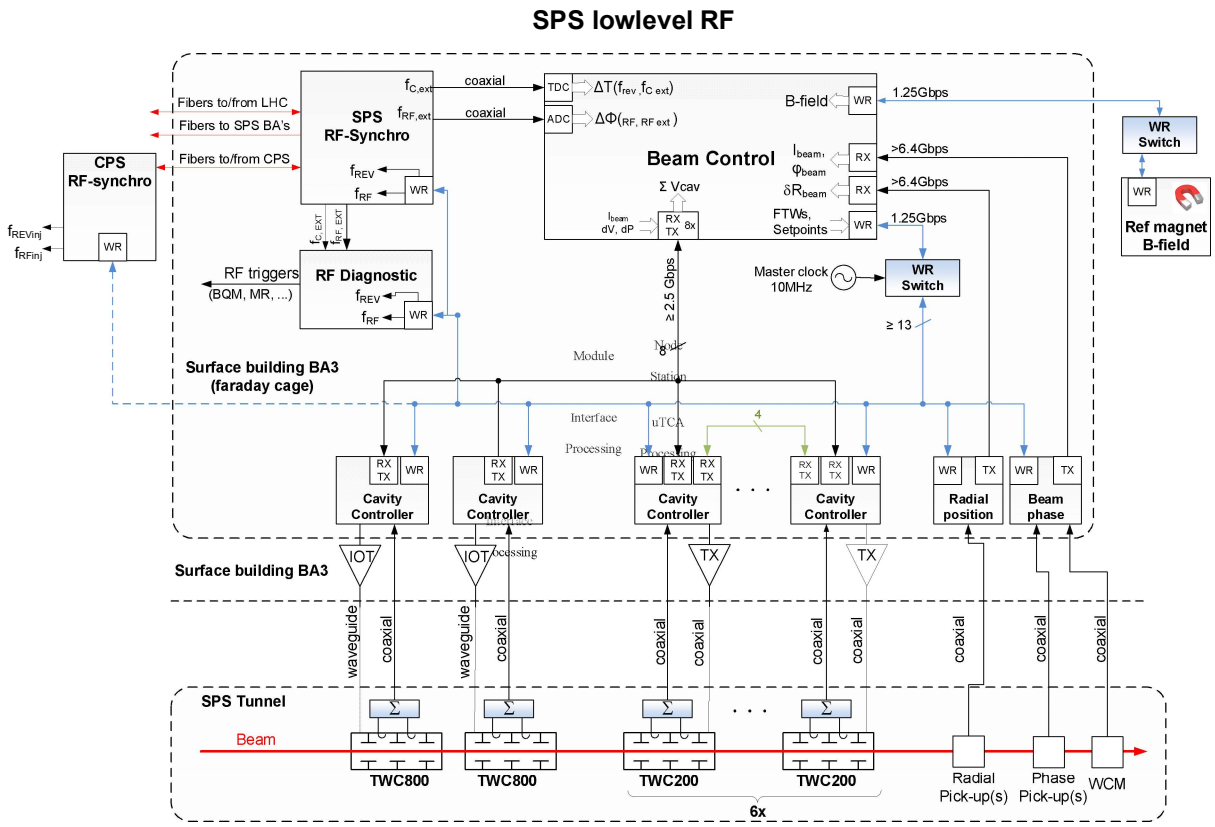


Fig. 6.9. SPS Low Level RF schematic architecture. Reproduced from [18].

in numerical format as the FTW. The RF-Synchro module is used for synchronization of the SPS with other machines at injection and extraction of the beam.

The Cavity Controllers are the modules responsible for the regulation of the accelerating RF field inside the cavity [50], and there is one per cavity. It probes the cavity voltage using one or several antennas(s) that couple(s) to the accelerating field, processes the signal and generates the drive sent to the amplifier. This node receives the FTW, the voltage set-point and other machine data from the beam controller via the WR link.

The core of the Cavity Controller module in the SPS is the OTFB algorithm, that settles the steady field in the cavity and also mitigates Transient Beam Loading. The regulation bandwidth extends 5 MHz on each sideband around the RF, covering around 116 revolution frequency lines per sideband [18]. The processing part of the module is implemented in the AMC card SIS8300-KU from Struck, while the RF front-ends span between this card and the RTM DS8VM1 from DESY/Struck. The hardware station is located in the SPS Faraday cage that is not very far from the cavities, in the underneath tunnel. This configuration results in a cabling delay between the cavity and the LLRF of 850 ns.

6.4.2. The OTFB algorithm

The OTFB is a feedback around the cavity-amplifier. First introduced in the early 1980s for the SPS [10] it has since been installed on many machines, sometimes as a complement to a Direct RF feedback [16], [112]. The algorithm performs two main actions; processing of the cavity field signal with a filter

tuned to the revolution frequency harmonics, and extension of the loop delay to properly match the corrective action to the RF field in the cavity with the next passage of the same beam portion. The extended loop delay matches one revolution period of the beam, one exact turn.

A schematic representation of the algorithm is presented in Fig. 6.10. The time τ plant is the time it takes the signals to travel from the output of the LLRF back to the input of the LLRF, after crossing the transmitter, the cavity, and the cabling between cavity and the LLRF. The time τ DSP is the processing time in the platform that comprises down and up-conversion, filtering… The addition of these two times, τ plant + τ DSP, is what we referrer as the loop delay; the measured time from the output of the OTFB algorithm to the input of the One Turn Matching block. This block is the responsible to increase (by adding the time τ match) and match the loop delay (τ plant + τ DSP + τ match) to exactly one revolution period of the beam, 1 TURN. The loop controller, block Revolution Frequency Filter in the figure, is the filter with its gain limited to narrow frequency bands around the revolution frequency harmonics [15]. Thanks to this combination, the OTFB algorithm reduces the beam loading, including the transients caused by the gaps in beam current, thereby equalizing the bunch parameters (length) and increasing the longitudinal coupled-bunch instability threshold [116].

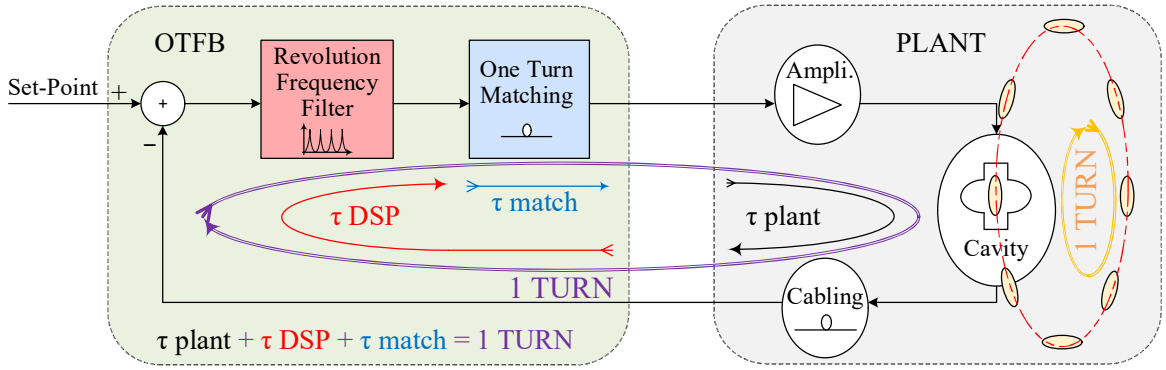


Fig. 6.10. Schematic representation of the One Turn FeedBack algorithm.

6.4.3. Partitioning of the OTFB between BSP and BAP

We have presented in the previous sections that the old SPS LLRF uses a swept RF resulting from the velocity increase of the beam. We have also introduced that the OFTB contains a filter matching the revolution frequency harmonics. We need hence to tune the filter to the varying harmonics resulting from the variable revolution frequency; the harmonics change in position and spacing in the signal of the cavity that contains the voltage perturbation caused by the beam. More complex processing schemes or algorithms can require also more operations that need to track the beam revolution frequency. There is hence a need in LLRF systems for real-time reconfiguration of parameters to tune varying signals as the revolution frequency.

With the new fixed frequency clock that it is fed to the uTCA stations after extraction from WR, we use our BSP solution to tune the processing elements (the filter in our case) to the beam revolution frequency (and to the spectrum of the sampled signal). The processing platform in the Cavity Controller

needs to deal with BAP also in the same single device, for instance in the RF front-ends. The BSP Architecture of Chapter 3 is an alternative offering flexibility to implement both types of processing inside the same FPGA, BSP and BAP. It avoids the burden of algorithm reconfiguration, or the need for differentiated hardware elements for BAP and BSP.

In the case of the OTFB, we need to partition its two elements between the BSP region, the sandwich between resamplers in the *FRANCISCO* fabric on top of the FPGA fabric, and the BAP region, the rest of the device. We implement the one-turn delay in the BAP region of the processing platform; any delay can be efficiently synthesized with a fixed clock using the FTW information. This region is depicted in blue in the schematic representation of Fig. 6.11. The filter is implemented in the BSP region of the processing Architecture, in red in the figure; the frequency response of the filter needs to track the beam energy ramping. The BSP Architecture, thanks to the resampler sandwich, performs automatically this tuning between the filter in the processing and the processed beam signal.

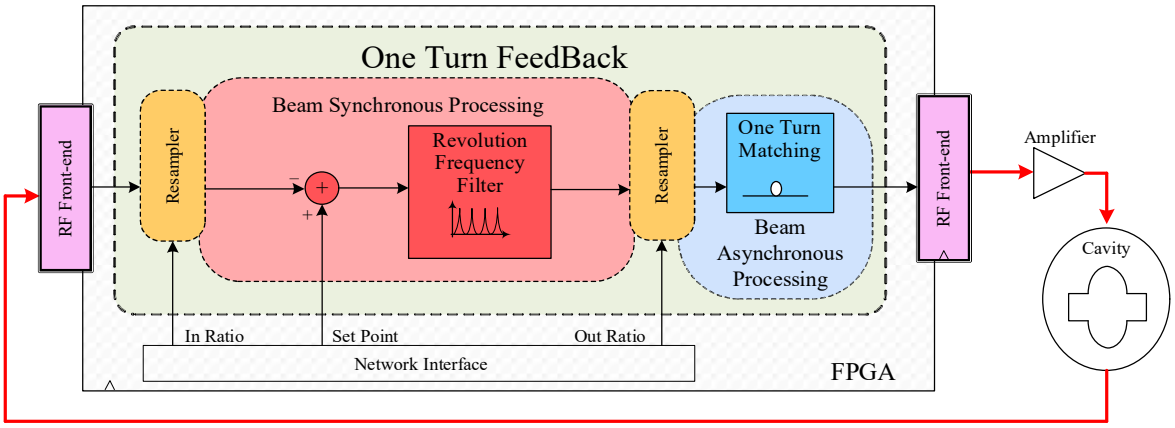


Fig. 6.11. Partitioning of the OTFB units between the BSP and BAP regions of the processing device.

6.4.4. The clocking architecture in the BAP and BSP

We depict in Fig. 6.12 a schematic representation of the OTFB implementation mapped to the processing regions within the FPGA. We present also the data-path related signals and clocks. The green entities in the figure, the *MERCEDES* interfaces presented in Chapter 3, are the boundaries of the data-path between the *FRANCISCO* fabric implementing the BSP, and the FPGA fabric implementing the BAP. They handle the data-path clock domain crossing. The clock relation M in the interfaces is defined by the frequencies of the clocking architecture in the BAP and BSP. In our case the system clock `clk` extracted from the WR network, is a f_p = 125 MHz signal. This clock is used in ADCs and DACs for acquisition and regeneration of the analog signals (cavity antenna and amplifier drive signal). The system clock of the FPGA is also this 125 MHz signal. Within the processing, the sampled data stream of the cavity field results, after conditioning in the RF front-end (blue BAP region in the figure), in an I/Q pair. This I/Q channels have a sampling rate of $f_{s\_cpl} = 62.5 \cdot 10^6$ sample/s each one.

They cross the hardware fabric within two parallel coupled data-paths that use a sub-multiple `clk_cpl` of the system clock at f_{p_cpl} = 62.5 MHz. The OTFB lies in the *FRANCISCO* fabric. The *MERCEDES* interfaces, configured with M = 2, decouple and couple back the data-path; in the

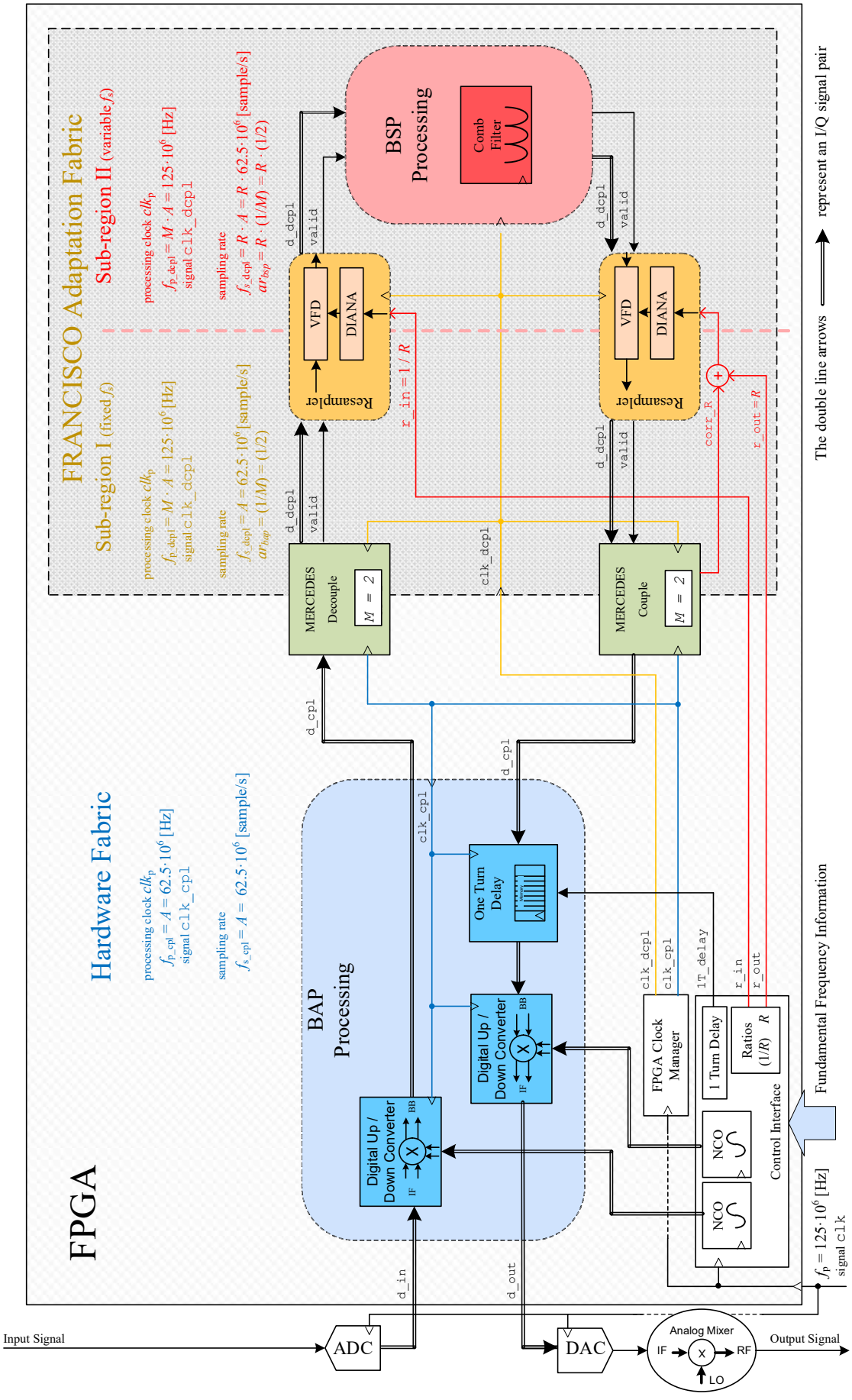


Fig. 6.12. Simplified representation of the OTFB implementation, and the clocking architecture in the processing device.

FRANCISCO fabric the clock `clk_dcpl` runs at a frequency f_{p_dcpl} = 125 MHz, with a sampling rate $f_{s\_dcpl} = 62.5 \cdot 10^6$ sample/s in the sub-region I (BAP), and $f_{s\_dcpl} = R \cdot 62.5 \cdot 10^6$ sample/s in the sub-region II (BSP) between resamplers.

6.4.5. The 1T Delay in the BAP

We have presented in section 6.4.2 the different delays in the LLRF-Amplifier-Cavity chain. The cabling, amplifiers and the LLRF DSP have a fixed latency. The One Turn Matching block synthesized by the LLRF is, on the other hand, a variable delay component; it is added to match the variable revolution frequency. In the original system this was easily done using a FIFO memory clocked with the swept clock of the system, whose frequency was a harmonic of the revolution frequency. The FIFO memory, with L memory positions, resulted in the variable delay proportional to the revolution frequency. And more important, a swept clock multiple of the revolution frequency makes the delay L a constant integer. The fixed-depth FIFO implemented hence the variable delay,

$$H(z) = z^{-L} \qquad \text{Eq.(6.4)}$$

The fixed delay was compensated in the RF front-end mixers by inserting a delay between the LO used in the demodulator and modulator mixers [10].

In our BSP fixed clock solution this is no longer applicable. We use instead a dual-port memory but now clocked with a fixed frequency signal. This memory is placed, as presented in the partitioning of the OTFB, in the BAP region. The one turn delay is achieved by updating the read and write pointer offset dynamically according to the revolution period (considering the known fixed delay).

To compute this offset, the FPGA implements an algorithm that divides the current variable delay needed by the BAP sampling clock period. The revolution frequency information is received in the node via the WR. The resulting value contains an integer part and a fractional part, the integer part is synthesizable with the memory achieving a time accuracy of a clock cycle (maximum error of half a clock cycle, i.e., 0.5 · 16 ns = 8 ns at 62.5 MHz). The fractional part is synthesized with a VFD filter placed behind the memory, with an architecture similar to the one used in the resampler. This VFD filters the corrective signal and modifies its value recreating the fractional delay contribution that adopts a value between -0.5 and 0.5 clock cycle.

6.4.6. The comb filter in the BSP and the regulation

The frequency response of the filter in the OTFB tracks the revolution frequency harmonics of the beam in the demodulated signal of the cavity. This filter is therefore a comb filter, with large gain on the revolution frequency harmonics; we have implemented the Z Transform shown in Eq.(6.5). The parameter a in the equation governs the bandwidth of the filter around each revolution frequency harmonic, and G is the gain of the filter on the resonances. The filter has zero phase shift on the revolution frequency harmonics, and the magnitude of the filter frequency response is depicted in Chapter 2, Fig. 2.8, for a unit

gain with $a = 31/32$ which gives around 36 dB peak to valley gain. The figure uses $N = 24$ which results in twelve harmonics per Nyquist zone.

$$H(z) = G \cdot (1-a) \cdot \frac{1}{1 - a\, z^{-N}} \qquad \text{Eq.(6.5)}$$

In the old system, the implementation of the tuning to the revolution frequency harmonics was very easy with the ratio of sampling clock to revolution frequency being the integer number N [15]. The swept system clock, which is a multiple of the revolution frequency, results in a delay N that is a constant integer.

That is not the case when the clock is fixed, our WR fixed frequency clock. In this case, the parameter N is not an integer anymore and it should be changed continuously during the acceleration ramp, so that N fixed clock cycles equal one varying revolution period. The solution that we propose using the BSP Architecture is different.

We implement the comb filter in the BSP region with a fixed frequency response, while we exploit the resampling of the incoming data to a sampling rate f'_s that is a multiple of the revolution frequency, and hence matches the response of the filter. The data stream to the filter has therefore a variable sampling rate, as in the old swept clock solution. The system therefore reproduces the old behaviour, but our implementation uses a fixed system clock, with a fixed filter (N does not need to be updated). We thereby avoid a clock tracking the beam revolution frequency and any reconfiguration in the processing system as in the original solution [15].

The frequency response of the BSP is normalized to the peak value to avoid saturation of the data-path. The recursive architecture of the filter is presented in Fig. 6.13. The filter is implemented using a decoupled data-path, with the valid flag signal controlling the enable input of the filter recursive register.

We need to choose the parameter N in conjunction with the resampling parameters in this fixed clock implementation as presented in section 3.3.9 or the example of Chapter 5; the resampler modifies the sampling period of the input according to the resampling ratio R. It generates an output signal with sampling rate f'_s (Eq.(2.2)) tuned to the filter response defined in the resampled domain. There, the filter response in Eq.(6.5) has peaks at the normalized angular frequencies

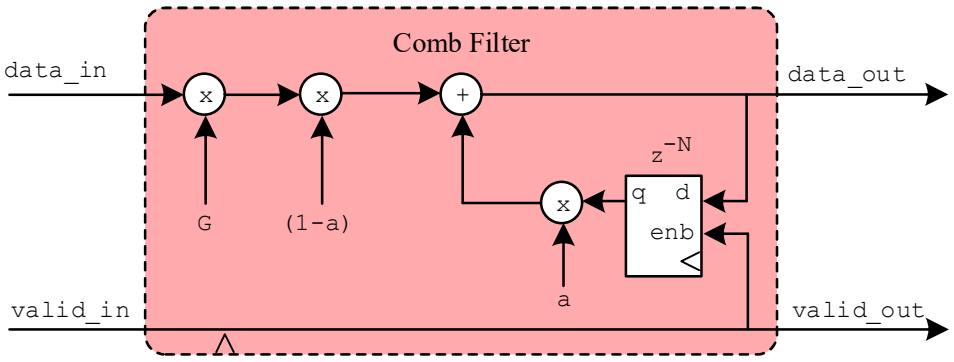


Fig. 6.13. IIR comb filter.

$$\omega_k = 2\pi \frac{k}{N} \quad \text{with } k = 0, 1, \ldots, N-1. \qquad \text{Eq.(6.6)}$$

As the filtering is done after resampling by frequency f'_s, the absolute positions of the peaks are

$$F_k = \frac{k}{N} f'_s \quad \text{with } k = 0, 1, \ldots, N-1. \qquad \text{Eq.(6.7)}$$

We want these peaks on the harmonics of the revolution frequency F_{rev}

$$F_k = k \cdot F_{rev} \quad \text{with } k = 0, 1, \ldots, N-1. \qquad \text{Eq.(6.8)}$$

Now merging Eq.(2.2), Eq.(6.7) and Eq.(6.8), we get

$$R = N \frac{F_{rev}}{f_s} \qquad \text{Eq.(6.9)}$$

In any resampler the error grows significantly when the input signal frequency approaches the Nyquist rate. N must be chosen depending on the filtering bandwidth, that is, it must be at least twice the number of revolution harmonics to be filtered ($N > 232$, two times 116 revolution frequency harmonics in the 5 MHz regulation bandwidth). We selected N to be 1442 sampling periods so that the frequency band of interest (5 MHz) extends to about one sixth of the Nyquist rate (31.25 MHz), and remains within the accepted input bandwidth for the resampling architecture (18.75 MHz section 3.3.10).

As the acceleration proceeds, the revolution frequency increases and so does the resampling ratio. At the output of the BSP region the second resampler, in the bottom of Fig. 6.12, recovers the original sampling rate by performing down-sampling. This down-sampling ratio is the inverse value of the up-sampling.

We count on the OTFB to regulate the voltage set-point. The filter and the one turn delay have unit gain in base-band; the static field in the cavity will hence reach, for $G = 10$ for instance, the value

$$V_{cav} = V_{set\_p} \frac{1}{1-G} = V_{set\_p} \frac{10}{1+10} = V_{set\_p} \frac{10}{11} \qquad \text{Eq.(6.10)}$$

6.5. Functional validation

The validation of the BSP solution implementing the OTFB for TBLC has been performed in first place by means of simulations of an accelerator plant. It represents part of the CERN SPS 200 MHz post-LIU accelerating system. It includes a model of the cavity, the amplifiers and the part of the LLRF responsible for the OTFB. Ancillary systems as the WR receiver and the associated blocks for local regeneration of the LO and other auxiliary signals used in the RF front-ends have also been included as functional models. The test bench reproduces also the RF modulation architecture with the input and output front-ends. This modulation architecture is staged in different IFs to match the frequency at which the

transfer function of the different elements of the model are described. This aims at being as close as possible to the real SPS. The system can reproduce the beam energy ramping with swept RF.

In the model, a pick-up measures the field in the cavity, an RF front-end down-converts the signal to base-band and feeds the LLRF. There the signal is compared against the reference value and the error signal is filtered and delayed (OTFB) to apply a correction to the RF field in the next bunch passage through the cavity. The objective is the reduction of the field perturbation induced by the beam passage. When the system behaves as expected, the field in the cavity in closed-loop shows rejection of the induced beam voltage, and the total voltage in the cavity approximates the set-point voltage. The OTFB uses a fixed frequency processing clock and the BSP Architecture for tuning.

6.5.1. The test bench model

The test bench used for validation is presented in Fig. 6.14. It emulates the LLRF regulation around a SPS 200 MHz cavity. The specification for the regulation bandwidth is that it must cover 5 MHz on each sideband around RF; this results in around 116 revolution frequency lines per sideband. The simulation can be programmed to reproduce a given accelerating cycle. The central frequency of the cavity is set to 200.242 MHz and the maximum and minimum values of the RF within a cycle are parameters of the simulation.

The blocks directly synthetizing the BSP region, and the OTFB, are implemented in Xilinx System Generator hardware primitives. The remaining elements of the model, used for validation of the solution have been modelled in Simulink. In the development process of the test bench, all the presented accelerator plant blocks have been first validated with MATLAB simulations. Complex elements as the cavity, amplifier and filters have been studied and modelled first as single blocks and later integrated into the system level model. The transfer function of these elements has been validated and the results have shown that the models behave as expected. The BSP region has also been modelled first in MATLAB, to assess the functional behaviour of the blocks.

Later, after deep understanding of the behaviour of all the elements, the blocks were integrated into a functional system level solution. These simulations included the processing latency of the blocks mapping the different elements, together with other latencies of the model, as cable delays. The functional model of the BSP elements was then migrated to Xilinx System Generator primitives for hardware verification. The hardware primitives make possible the simulation of the hardware architecture and its internal signals with a processing clock cycle accuracy.

Finally, the BSP Architecture has been integrated into the system level simulation test bench reproducing the TBLC with the new OTFB.

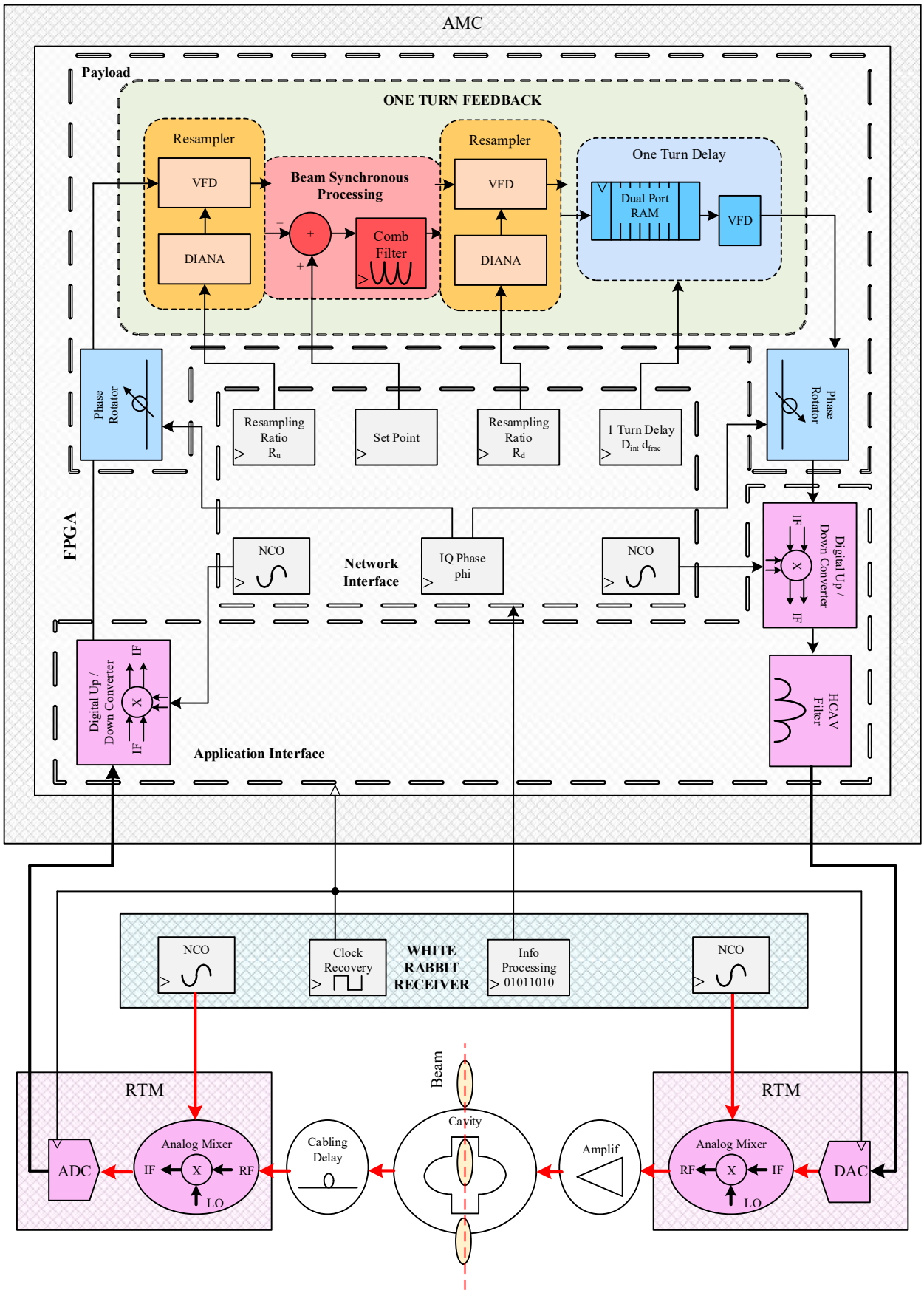


Fig. 6.14. Test bench architecture.

6.5.1.1. Modulation architecture

The LLRF control algorithms perform the processing in base-band. The OTFB, as one of such algorithms, performs the computation of the error signal in the cavity field after down-conversion of the RF signals. Machines operating in the lower radio-frequency range can sample the RF signals directly. Above few hundreds MHz, demodulation is used first, preferably in I/Q coordinates where beam loading is an issue. In these machines, one or multiple mixing stages down-convert a narrowband RF signal to base-band. The down-conversion process brings also to base-band some revolution frequency harmonics (depending on the regulation bandwidth) present in the sidebands around the RF. The resulting signal is sampled as an I/Q pair that after processing is up-converted back to the required RF.

In communication systems, where the RF is fixed, or accelerators, that use also fixed RF as LINACs, these modulation stages are often simple. This is not the case of the SPS where the RF is swept to accelerate the beam. In our case, a varying LO frequency is required to bring the sweeping RF to base-band. This LO is reconstructed locally in the processing nodes with the clock and information extracted from the deterministic link (FTW). Down and up-conversion can be implemented by means of an analog RF front-end (the acquired signal is mixed with a real analog swept LO), or digitally by means of a digital mixer driven from an NCO. We use both solutions in the SPS 200 MHz system. In the input RF front-end we perform direct-sampling of the RF signal followed by numerical down-conversion to base-band. The output RF front-end uses a vector modulator with an IF intermediate stage.

The simplified modulation architecture of the SPS, present also in our test bench model of Fig. 6.14, follows the schematic representation of Fig. 3.8 in Chapter 3. There the input direct-sampling stage has been replaced by a combination of two mixers, one analog and one digital, to ease the understanding of the different stages. In the schema, several IF frequency regions are defined to host different processing elements. In the LLRF region IF2, where the BSP hosts the OTFB, the goal is to process the RF in base-band. The region that performs BAP, IF3, contains the HCAV filter, that will be presented in 6.5.1.4, and the amplifier model. This region aims at demodulating the centre frequency of the cavity to base-band. The region containing the cavity model modulates the RF and cavity to its nominal values. Note that we use a base-band model of the amplifier in contrast with Fig. 6.14 that places the amplifier in the RF region as in the real SPS. The LO2 and LO3 are swept according to the RF ramping.

6.5.1.2. The cavity model

The accelerating structure can be configured to match any SPS TWC cavity configuration in number of sections and cells. The present simulation uses a single four-section cavity. Our mathematical model of the Travelling Wave Cavity is defined in 6.2.2. As presented in [112], we have incorporated two I/Q mixers excited by an LO at a frequency f_c, one at the input and the other at the output, so that the model can be placed in any IF region. This results in the I/Q frequency-domain model of Eq.(6.11).

$$\begin{bmatrix} V_{\mathrm{I}} \\ V_{\mathrm{Q}} \end{bmatrix} = \begin{bmatrix} Z_{\mathrm{g,s}} & -Z_{\mathrm{g,c}} \\ Z_{\mathrm{g,c}} & Z_{\mathrm{g,s}} \end{bmatrix} \begin{bmatrix} I_{\mathrm{g,I}} \\ I_{\mathrm{g,Q}} \end{bmatrix} + \begin{bmatrix} Z_{\mathrm{b,s}} & -Z_{\mathrm{b,c}} \\ Z_{\mathrm{b,c}} & Z_{\mathrm{b,s}} \end{bmatrix} \begin{bmatrix} I_{\mathrm{b,I}} \\ I_{\mathrm{b,Q}} \end{bmatrix} \quad \text{Eq.(6.11)}$$

The diagonal *C* terms in the impedance matrices denote the cross coupling between I and Q channels, while the *S* terms denote the direct contribution of the channel. Then the frequency responses are transformed into the impulse responses of Eq.(6.12) and implemented in the Matlab simulation as FIR filters. The generator impulse responses $h_{\mathrm{g,s}}(t)$ and $h_{\mathrm{g,c}}(t)$, and the beam impulse responses $h_{\mathrm{b,s}}(t)$ and $h_{\mathrm{b,c}}(t)$ of Eq.(6.12) are respectively presented through equation Eq.(6.13) to Eq.(6.16).

$$\begin{bmatrix} v_{\mathrm{g,I}} \\ v_{\mathrm{g,Q}} \end{bmatrix} = \begin{bmatrix} h_{\mathrm{g,s}}(t) & -h_{\mathrm{g,c}}(t) \\ h_{\mathrm{g,c}}(t) & h_{\mathrm{g,s}}(t) \end{bmatrix} * \begin{bmatrix} i_{\mathrm{g,I}} \\ i_{\mathrm{g,Q}} \end{bmatrix} + \begin{bmatrix} h_{\mathrm{b,s}}(t) & -h_{\mathrm{b,c}}(t) \\ h_{\mathrm{b,c}}(t) & h_{\mathrm{b,s}}(t) \end{bmatrix} * \begin{bmatrix} i_{\mathrm{b,I}} \\ i_{\mathrm{b,Q}} \end{bmatrix} \quad \text{Eq.(6.12)}$$

$$h_{\mathrm{g,s}}(t) = \frac{R_{\mathrm{I}}}{\tau} \Pi\left(\frac{t}{\tau} - \frac{1}{2}\right) \cos\left(2\pi \cdot \Delta f_0 \cdot t\right) \quad \text{Eq.(6.13)}$$

$$h_{\mathrm{g,c}}(t) = \frac{-R_{\mathrm{I}}}{\tau} \Pi\left(\frac{t}{\tau} - \frac{1}{2}\right) \sin\left(2\pi \cdot \Delta f_0 \cdot t\right) \quad \text{Eq.(6.14)}$$

$$h_{\mathrm{b,s}}(t) = -R_2 \left\{ \frac{1}{\tau} \Lambda\left(\frac{t}{\tau}\right) \left(\cos\left(2\pi \cdot \Delta f_0 \cdot t\right) + \mathrm{sgn}\left(\frac{t}{\tau}\right) \cos\left(2\pi \cdot \Delta f_0 \cdot t\right) \right) \right\} \quad \text{Eq.(6.15)}$$

$$h_{\mathrm{b,c}}(t) = R_2 \left\{ \frac{1}{\tau} \Lambda\left(\frac{t}{\tau}\right) \left(\sin\left(2\pi \cdot \Delta f_0 \cdot t\right) + \mathrm{sgn}\left(\frac{t}{\tau}\right) \sin\left(2\pi \cdot \Delta f_0 \cdot t\right) \right) \right\} \quad \text{Eq.(6.16)}$$

In the equations, the fixed difference Δf between demodulation frequency f_{c} and cavity centre frequency f_0 is defined as in Eq.(6.17). Refer to [50], [112] for more details on the cavity model.

$$\Delta f \triangleq f_{\mathrm{c}} - f_0 \quad \text{Eq.(6.17)}$$

6.5.1.3. The TX driver

The amplifier block models the base-band response of one of the tetrode amplifiers of the SPS around its centre frequency. It is implemented with a Butterworth filter having single-sided pass-band and stop-band edge frequencies at 1.5 MHz and 4 MHz respectively. The attenuation is 3 dB at the pass-band edge and 15 dB at the start of stop-band.

The loop delay including cables, amplifier, cavity and LLRF is modelled according to the real conditions; we use 850 ns as delay between LLRF and cavity that results in 1.6 µs round-trip delay.

6.5.1.4. The HCAV filter

The 5 MHz regulation bandwidth covers several zeros of the frequency response of the cavity [50], [112]. The first side lobes of this response have a phase shift of 180 degrees, and for stability, the feedback frequency response must also change sign at these frequencies. A filter HCAV is included in series with

the OTFB, which adds 180 degree extra phase shift to these lobes. HCAV is implemented in base-band, with the same response as Z_g to properly match the cavity zeroes.

6.5.1.5. The beam model

The beam is modelled as a DC current lasting for part of the turn (the populated buckets). This current is modulated at the RF frequency and injected into the cavity model (beam impedance). The simulation is performed for zero-degree stable phase (synchrotron convention). The cavity voltage and RF component of beam current are therefore in quadrature. As the feedback is implemented in I/Q coordinates, performances in term of beam loading compensation do not depend much on the stable phase. This beam model ignores longitudinal beam dynamics and is therefore valid in static conditions only (no oscillation of the beam in the RF potential). It is not valid during the injection transient caused by phase, momentum and bucket mismatch between the Proton Synchrotron (PS) injector and SPS buckets. The current intensity is adjusted in the simulations so that the beam-induced voltage equals the generator-driven voltage.

6.5.2. The simulations

We now present the simulation results of the implemented OTFB algorithm using the BSP Architecture. In the simulation the RF frequency ramps from 200.242 MHz to 200.342 MHz in four thousand turns lasting 92 ms. According to the RF values and given the use of a harmonic number equal to 4620, the revolution frequency of the simulated beam ramps from 43.342 kHz to 43.364 kHz. This is much faster than any SPS ramp but the limit is caused by the computation load. The ramping is linear with steps of 25 Hz per turn, which gives a dF/dt of 1.08 MHz/s, a value more than seven times faster than the maximum SPS ramping rate of 142 kHz/s. The simulation calls for 1 MV set-point in the cavity. The RF component of the beam current is set at 1.14 A so that the beam-induced voltage also equals 1 MV. The stable phase set to 0 degrees for simplicity (synchrotron convention) is a non-realistic value for acceleration (above transition) but eases the simulation analysis. The parameters are in accordance with the specifications after the LIU upgrade [12] (total voltage of 6 MV at injection, set-point of 1 MV per cavity), but the RF component of the beam current will peak at 2 A in the HL-LHC era.

6.5.2.1. Open-loop response of the regulation

The magnitude of the open-loop transfer function of the model is presented in Fig. 6.15, after phase alignment of the system. The model is excited with an impulse (delta) injected in the I channel only. The

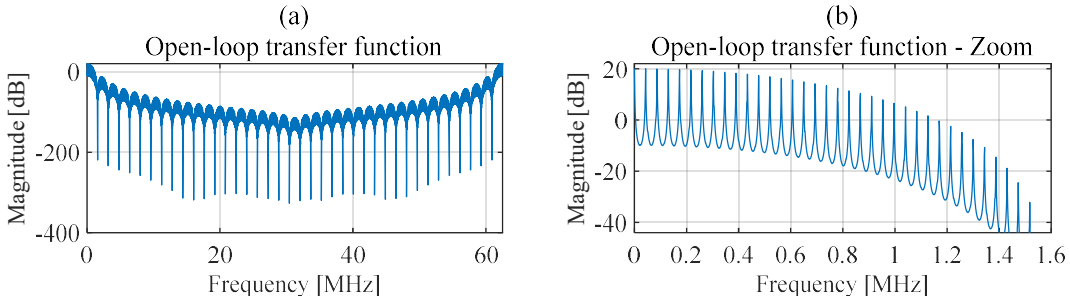


Fig. 6.15. (a) Open-loop transfer function of the feedback system, RF at base-band. (b) Zoom of the first 1.6 MHz.

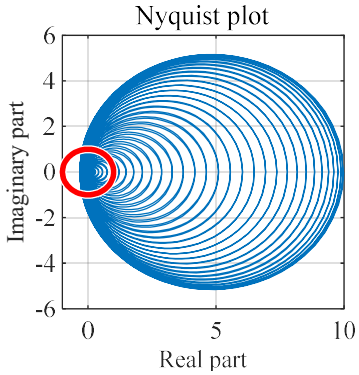


Fig. 6.16. Nyquist plot of the open-loop transfer function of the simulation.

open-loop response (I and Q channels) is measured. Then the transfer function of the model is computed with the Fourier Transform of the response. This response is obtained with an RF frequency at the cavity centre frequency (the plot depicts the I to I+jQ open-loop response with a sampling frequency of $62.5 \cdot 10^6$ sample/s). The zeroes of the cavity are located at 1.6 MHz and its multiples, as shown on Fig. 6.15(a). Fig. 6.15(b) enlarges the low frequency part of the response to show the comb filter peaks at the revolution frequency harmonics. The spacing is 43.342 kHz.

The Nyquist plot of the transfer function is presented in Fig. 6.16. The 30 dB gain of the comb filter can be clearly seen in the real axis. The red circle depicts the unit circle.

6.5.2.2. The cavity voltages and TBLC compensation

After the alignment and verification of the response of the system, we close the feedback loop around the cavity and study the Transient Beam Loading Compensation performance of the OTFB with the

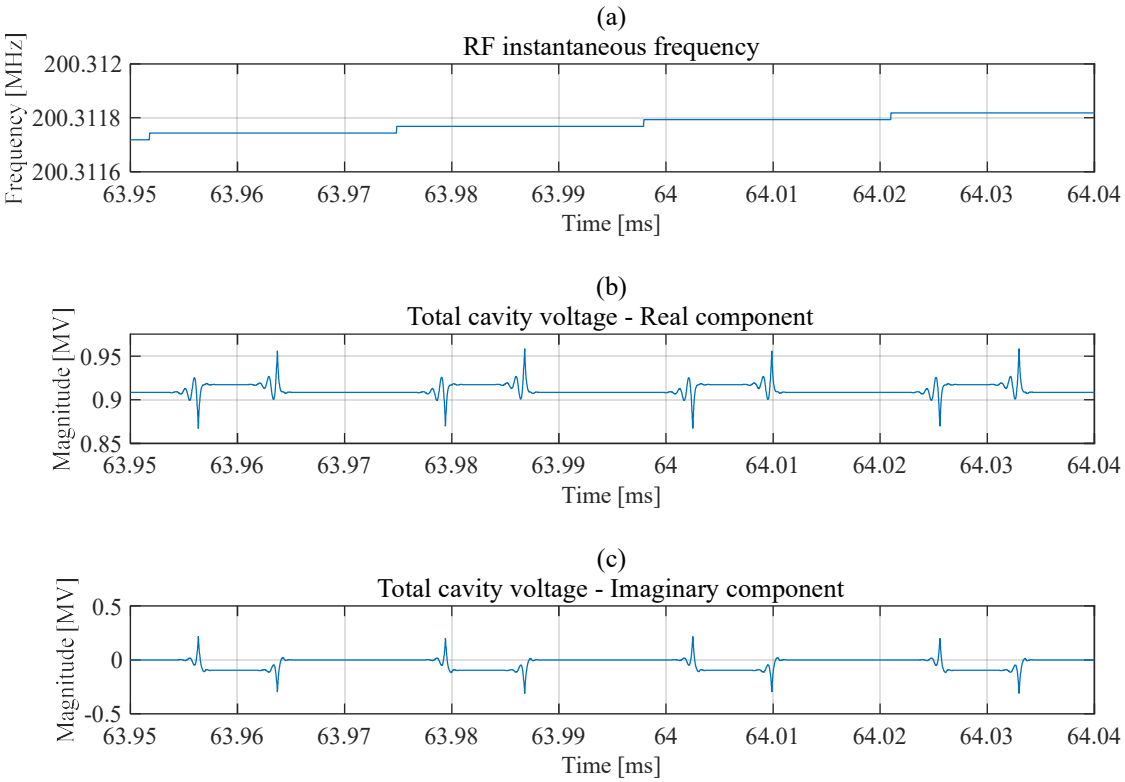


Fig. 6.17. Cavity voltage during the simulated ramp, zoom around simulation time 64 ms. (a) RF instantaneous frequency, (b) Cartesian I component and (c) Cartesian Q component.

BSP Architecture. We start in Fig. 6.17 depicting the measured voltage in the cavity and the update ratio of the RF frequency ramp, once per revolution period. The figure is zoomed around simulation time 64 ms, some turns after the start. The beam spans one third of the ring (6.6 μs). The gaps in the signal correspond to the empty buckets spanning the remaining two thirds of the ring. When the beam crosses the structure the voltage in the cavity is perturbed by the beam loading. The steps of the linear frequency ramp are shown in Fig. 6.17(a), the real component of the cavity voltage is presented in Fig. 6.17(b) and the imaginary component in Fig. 6.17(c). The cavity filling time is 620 ns, for a revolution period around 23 μs. Therefore, the full voltage is induced at each turn and does not accumulate over the turns.

The voltage in the cavity is depicted in Fig. 6.18 for the first 1.1 ms. The left column depicts the real component (I) and the right column the imaginary (Q). Fig. 6.18(a1) and Fig. 6.18(a2) show the beam-induced voltage. The beam is injected after 0.48 ms, inducing -1.061 MV in the reactive Q channel.

Fig. 6.18(b1) and Fig. 6.18(b2) show the voltage in the cavity driven by the generator. The set-point is set to 1 MV and the regulation is closed after 0.07 ms. The voltage magnitude reaches a stable value equal to 0.909 MV after 0.2 ms. This value is consistent with Eq.(6.10). The different steps correspond to the response of the OTFB turn after turn. After one turn, the voltage in the cavity reaches a value around 0.65 MV. The time constant of the regulation in this case is hence one turn, 0.023 ms. This validates the regulation as this reaction time is much smaller than the synchrotron period, in the order of 2.5 ms.

Fig. 6.18(c1) and Fig. 6.18(c2) depict the total voltage in the cavity, being the addition of both the generator and the beam-induced voltage. At the first passage, the beam induces -1.061 MV reactive (Q

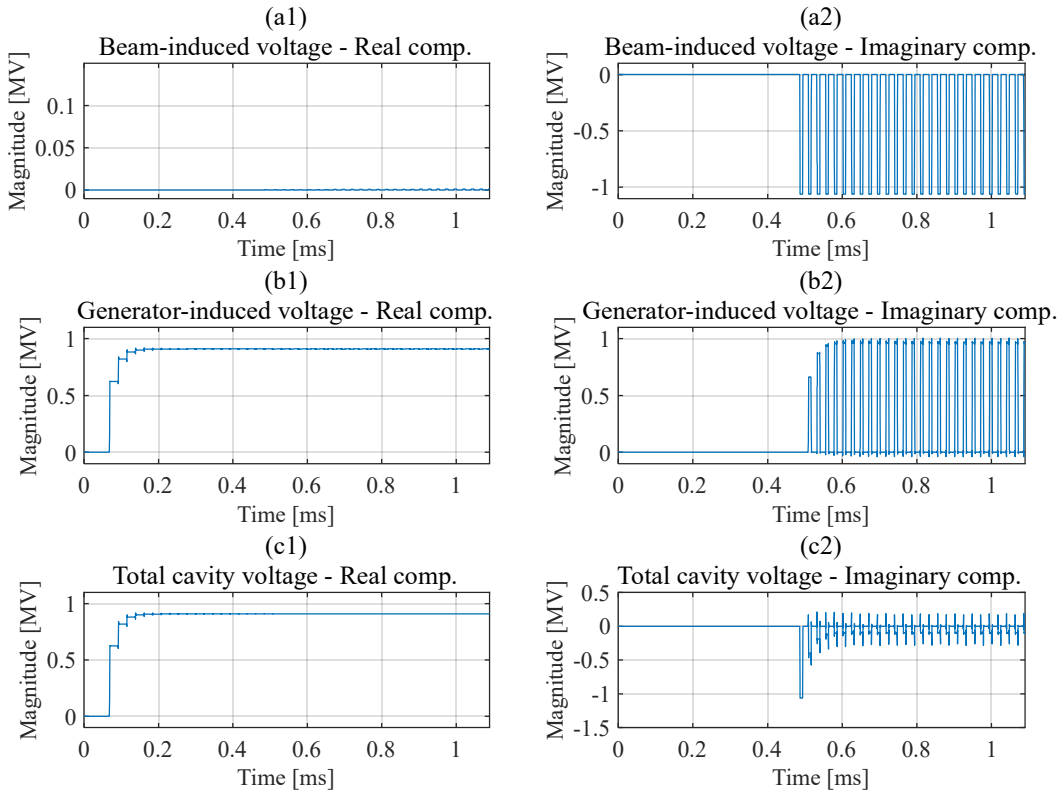


Fig. 6.18. Cavity voltage for the first 1.1 ms of the simulation in Cartesian I (left) and Q (right) components: (a) Beam-induced voltage, (b) Generator-induced voltage, (c) Total cavity voltage. RF in the beginning at 200.242 MHz.

channel). The beam current is in quadrature with the cavity voltage (zero-degree stable phase). At the beginning of the simulation the cavity is on tune (RF frequency equal to the cavity centre frequency) so there is no beam loading in the I channel. The first correction of the OTFB arrives on the second turn. After five turns the beam-induced voltage has been reduced to -0.1 MV in the middle of the beam batch. The factor ten reduction is consistent with the feedback gain at low offset frequencies ($G = 10$). Larger transients remain at the head and tail of the beam batch, caused by the reduced gain at large offset frequencies. As visible on Fig. 6.15(b), the open-loop gain has dropped to 0 dB at 1.2 MHz offset from cavity centre frequency.

The voltage in the cavity for the last 0.25 ms of simulation is depicted in Fig. 6.19. Fig. 6.19(a1) and Fig. 6.19(a2) show the demodulated beam-induced voltage, which now reaches 0.14 MV and -1.047 MV in the I and Q channels respectively, in the middle of the beam batch. The RF is now offset by 100 kHz with respect to the cavity centre frequency resulting in beam loading in both channels. Fig. 6.19(b1) and Fig. 6.19(b2) show the generator-induced voltage in the cavity set by the regulation. When the beam crosses the cavity, the regulation reduces the voltage in the I channel from the 0.9 MV to 0.78 MV to compensate the beam loading in the I channel. The same behaviour is observed in the Q channel, the regulation increases the voltage from 0 MV to 0.95 MV in the middle of the beam batch. The total voltage in the cavity, addition of the beam and generator-induced voltages is depicted in Fig. 6.19(c1) and Fig. 6.19(c2). When there is no beam crossing the cavity, the regulation sets 0.9 MV in the I channel. In the middle of the beam batch, the I channel reaches 0.92 MV while the Q channel has -0.1 MV. Again,

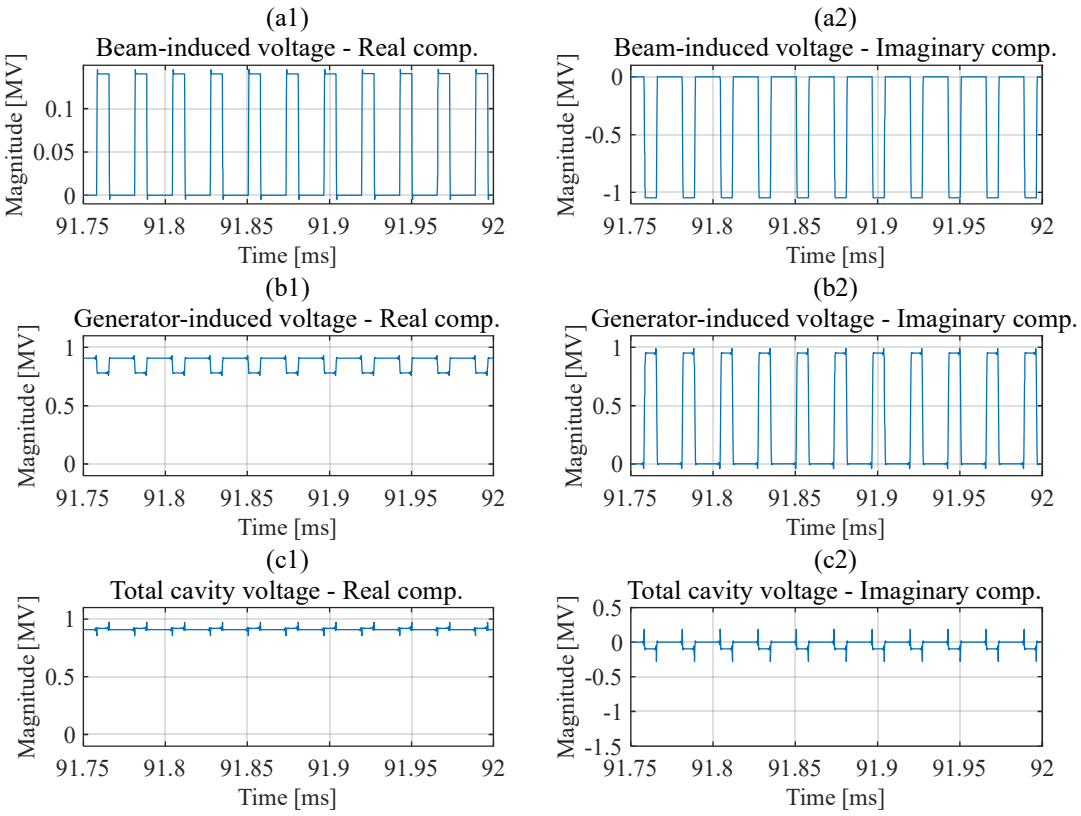


Fig. 6.19. Cavity voltage for the last 0.25 ms of the simulation in Cartesian I (left) and Q (right) components: (a) Beam-induced voltage, (b) Generator-induced voltage, (c) Total cavity voltage. RF in the end at 200.342 MHz.

there is a factor ten reduction of the beam-induced transients, consistent with the feedback gain at low offset frequencies ($G = 10$).

6.5.3. Hardware tests

We performed tests with the implementation of the OTFB in real hardware after the simulations. This section presents the results of these tests using a SPS cavity and a uTCA station hosting the OTFB; the algorithm has been successfully implemented in the AMC FPGA SIS8300-KU from Struck, targeting a 125 MHz processing clock for its high-grade Xilinx Kintex-7 XCKU040-1FFVA1156C. No hardware optimization has been done, resulting in a maximum achievable FPGA clock of 200.8 MHz. The data-path at the input and output of the system has the same width as the one presented in Chapter 5 for the resampler sandwich, which remains unmodified; sixteen bits wide, with fifteen fractional part bits and one sign bit.

6.5.3.1. The hardware setup

The setup is placed in the CERN BAF3 building where the SPS LLRF and power plant are placed. The test uses a real SPS cavity and the pre-driver of the power plant. The system responses have been measured with a Vector Network Analyzer (VNA). The stimulus output of the VNA is connected to the antenna input of the LLRF (uTCA crate). The output drive signal of the crate is connected to the VNA input. The analog LO signal of the crate is generated with an external precision RF synthesizer. For spectrum measurements we replace the antenna signal by a clean synthesizer, and we acquire the output of the system with a SA. The RF frequency value to which the uTCA crate is synchronized, is provided to the test setup via a discrete FTW coming from the White Rabbit link.

6.5.3.2. Open-loop response

We have first performed functional tests to verify the tuning capabilities and functionality of the BSP Architecture within the OTFB. We study the open-loop response of the LLRF with the comb filter in the processing chain. It needs to adapt the spacing between peaks to the revolution frequency, sub-multiple of the tuned RF. We have configured the setup with the SPS harmonic number h = 4620.

We measure the transfer function with different RF frequencies and spans in the RF domain, including the whole uTCA crate, from RF input to RF output. We present here the most relevant results of these measurements; in Fig. 6.20 the RF frequency is set to 200.2 MHz and the span covers 2 MHz, twenty-

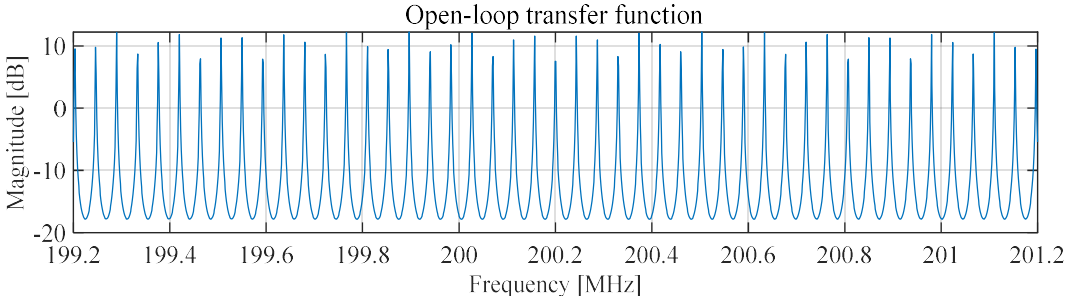


Fig. 6.20. Transfer function of the OTFB processing chain. The BSP, analyser and measurement are tuned to 200.2 MHz with 2 MHz span.

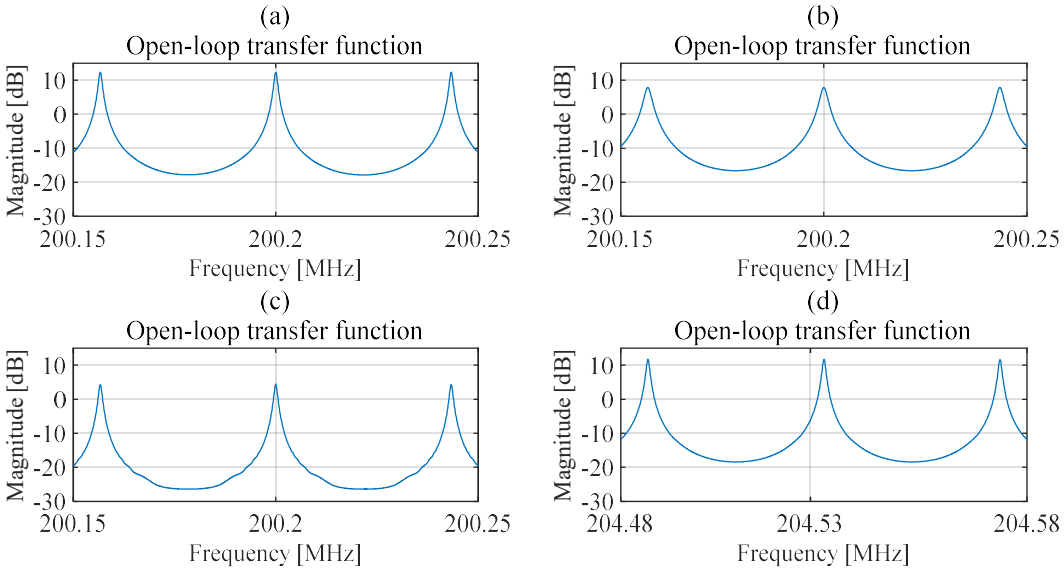


Fig. 6.21. Enlargements of the OTFB magnitude transfer function: (a) RF frequency at 200.2 MHz and span covering the first harmonics. (b) Bandwidth modified with $a = 7 / 8$. (c) Gain modified to $G = 1.5$. (d) RF frequency of 200.2 MHz and zoom around harmonic h_{100}.

three revolution frequency harmonics on each side of the RF, spaced by the corresponding 43.3 kHz. The BSP effectively tunes the comb response to the revolution frequency harmonics; the spacing between peaks matches the revolution frequency at which the harmonics repeat. The BSP filter is configured with 30 dB peak to valley gain in the resonances, $a = 15 / 16$. The global gain is set to a linear factor $G = 4$ that results in 12 dB gain in the peaks of the filter response. The measurement agrees with the simulations, Fig. 6.15, and theory Eq.(6.5).

We present in Fig. 6.21(a) an enlargement around the centre frequency with a span of 100 kHz, that covers the first harmonics around the RF spaced by the revolution frequency, 43.33 kHz. We modified the filter gain G and bandwidth parameter a without changing the RF; in Fig. 6.21(b) the coefficient $a = 7 / 8$ increases the bandwidth of the resonances without affecting its position. In Fig. 6.21(c) the parameter a is reconfigured back to $a = 15 / 16$ with the analyser covering a 100 kHz span, but we use a linear factor $G = 1.5$ instead of 4. This results in 4 dB global gain for the normalized filter response, as presented in the figure, with the spacing between peaks remaining unaltered at 43.33 kHz. Fig. 6.21(d) is centred on the harmonic h_{100} at 204.533 MHz with the harmonics h_{101} at 204.576 MHz and h_{99} at 204.49 MHz. The BSP effectively tunes the comb response to the beam revolution frequency in all the regulation bandwidth. The test has been repeated with different fixed and swept RF frequencies with satisfactory results.

We continue depicting in Fig. 6.22 the Nyquist plot of the system response. Note that in this hardware measurement we set the global gain $G = 4$ instead of $G = 10$ as in Fig. 6.16. The BSP remains tuned to an RF frequency of 200.2 MHz with a span of 500 kHz in the instrument. The comb filter (without the global gain G) and the LLRF input to output response have been designed to have unit gain at DC. In the figure however, $G = 4$ displaces the crossing with real axis in the right plane to Real$[H(F)] = 3.75$. The difference with respect to 4 is due to the fast sweep configured in the analyser. In the left side crossing with

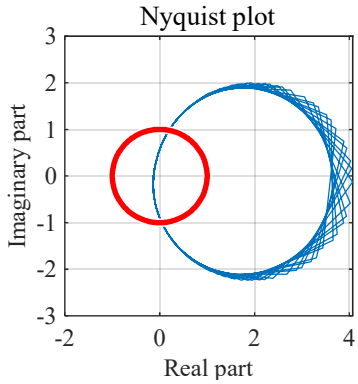


Fig. 6.22. Measured open-loop response, Nyquist plot.

the real axis, the one turn delay in series with the comb filter shifts the circle from the point Real[$H(F)$] = 0.129 to the depicted Real[$H(F)$] = -0.129. The value results from the bandwidth parameter that remains at a = 15 / 16. This configures a peak to valley linear factor of value 31 and hence the quotient between peak and valley results 4 / 31 = 0.129.

We now perform measurements of the spectral purity of the RF signal at the output of the OTFB. Fig. 6.23(a) shows the spectrum of the RF output (amplifier drive) when we excite the system with a clean signal and measure the output with a SA. The analyser is centred on the RF value of 200.2 MHz, with a span of 2 MHz. The noise floor adopts a value of -88 dBm for a 0 dBm input. This measurement is consistent with the simulations of Chapter 5 and shows that there is no significant noise added by the BSP. In Fig. 6.23(b) we present an enlargement around the RF with the span reduced to 100 kHz and with no spurious present around the RF.

We continue with a phase noise measurement on this RF signal. The synthesiser, analyser and the BSP in the uTCA crate are tuned to a frequency of 200.2 MHz. The results are presented in Fig. 6.24. The bandwidth covered is 1 MHz and the measured phase noise is -98 dBc/Hz at 10 Hz offset from the carrier.

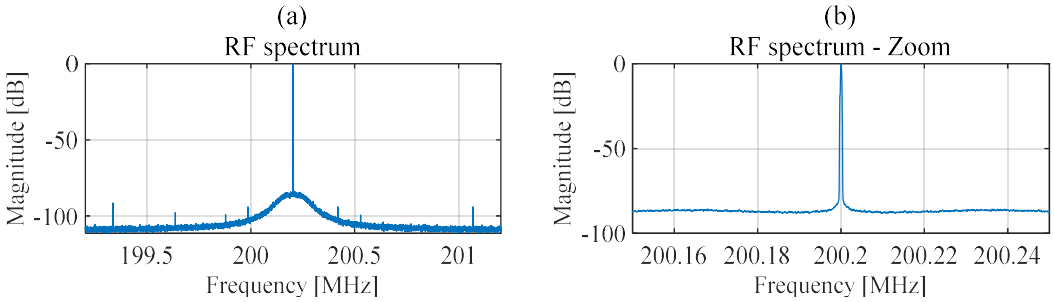


Fig. 6.23. Spectrum of the RF signal at the output of the OTFB, RF at 200.2 MHz. Span of (a) 2 MHz and (b) 100 kHz.

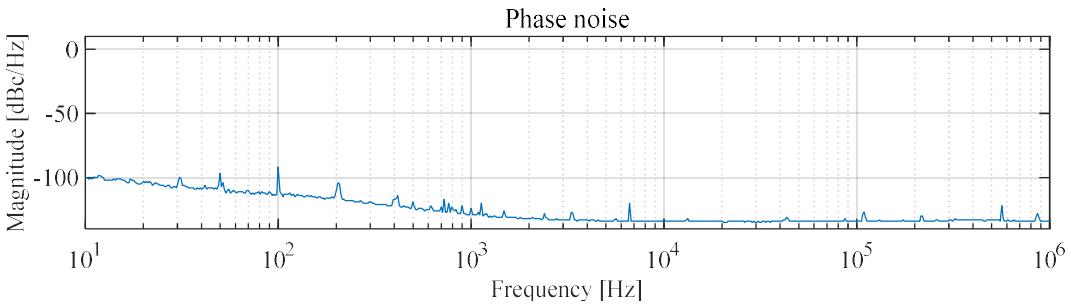


Fig. 6.24. Phase noise measurement. BSP tuned to 200.2 MHz, bandwidth of 1 MHz.

6.5.3.3. *Closed-loop response*

We now close the feedback around the cavity to perform measurements of the accelerating voltage inside the structure and to study the compensation of the Transient Beam Loading. The cavity is placed in a test-stand that uses as amplifier the 1 kW pre-driver of the SPS amplifier. We could not use the full 1 MW amplifier as the cavity was not conditioned to high field yet. The measurements are obtained with the sum of antennas from all cells (RF summing network) [50]. The first measurement is the spectral purity of the field when no beam is present. Fig. 6.25 shows the measured spectrum by the SA in the cavity when the regulation sets 0 dBm as defined by the voltage set-point (Fig. 6.14). The analyser and the RF are centred at a value of 200.347 MHz, with a span of 2 MHz in Fig. 6.25(a) and 100 kHz in Fig. 6.25(b). The noise floor remains similar to the open-loop measurement (Fig. 6.23) of the RF signal at a value of -85 dBm.

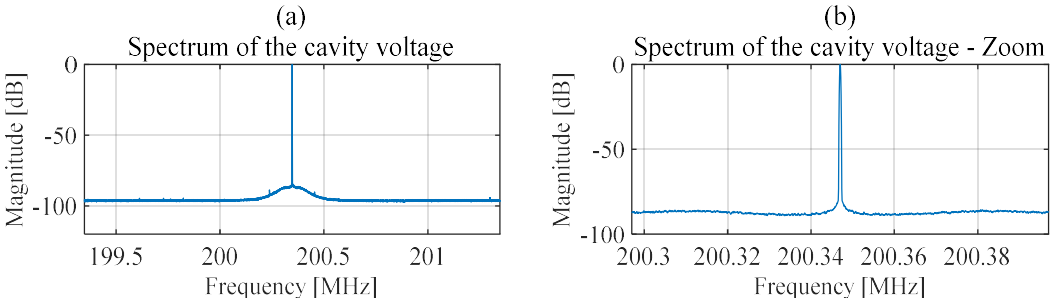


Fig. 6.25. Spectrum of the measured cavity field, RF frequency at 200.347 MHz. Span of (a) 2 MHz and (b) 100 kHz.

We present in Fig. 6.26 a spectrogram acquired when the RF is swept following a linear ramp between 199.8 MHz and 200.4 MHz in 250 ms and then back to 199.8 MHz in the same time (sawtooth at 2 Hz repetition rate). The regulation and the BSP follow in real-time the frequency ramp while tuning the processing to the RF signal as expected.

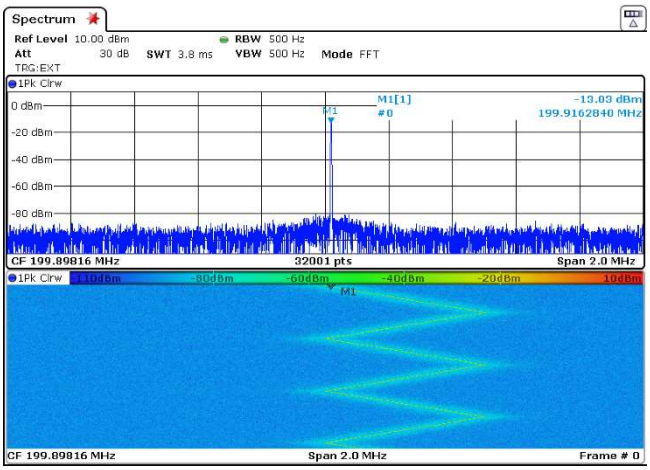


Fig. 6.26. Spectrogram of the measured cavity field; the RF is swept following a linear sawtooth pattern (600 kHz peak-peak).

The SPS accelerator was initially stopped till April 2021, and after the CoVid-19 pandemic its restart has been further delayed, so beam test was not possible. The beam was therefore emulated by injecting a perturbation, periodic at the revolution frequency, into the cavity drive signal. The OTFB is then expected to reduce the effect of this *beam loading* perturbation.

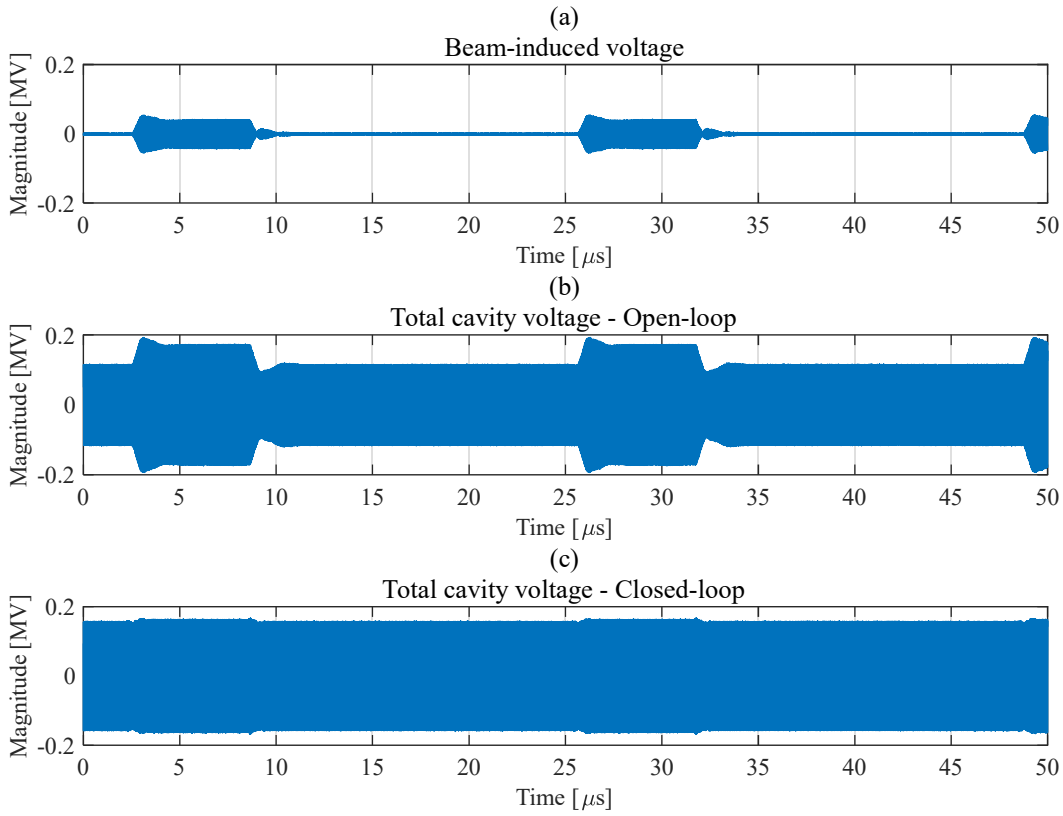


Fig. 6.27. Measured RF field in the cavity. (a) Beam-induced voltage, (b) total cavity voltage in open-loop, (c) total cavity voltage in closed-loop.

We start presenting in Fig. 6.27 time-domain measurements of the voltage in the cavity for a bit more than two turns, 50 µs. The top trace, Fig. 6.27(a), shows the beam-induced voltage when the regulation is open and the set-point requires 0 MV. The emulated beam passage induces a perturbation that reaches 0.05 MV peak. The beam segment lasts for 6.6 µs with gaps that correspond to the empty buckets, it repeats at a revolution frequency of 43.268 kHz for an accelerating RF of 199.89 MHz. The full voltage is induced at each turn and does not accumulate over the turns because the cavity filling time, 718 ns, is much shorter than the revolution period, 23 µs [13]. In Fig. 6.27(b) the regulation remains open but the set-point is now configured to 0.15 MV peak. The plot depicts the total voltage in the cavity that results from the addition of the beam-induced voltage and the generator-induced voltage when the beam is present. The perturbation after the initial transitory remains flat at 0.18 MV peak. In the last trace, Fig. 6.27(c), the regulation is closed. The field in the cavity approaches the required 0.15 MV and the beam perturbation is significatively reduced to 0.03 MV peak, that corresponds to a reduction factor $G = 20$.

Fig. 6.28 presents the demodulated cavity voltage when the set-point is configured to 0.4 MV with a phase of 45º, that corresponds to 0.28 MV in the real and imaginary components of the set-point. In Fig. 6.28(a1) and Fig. 6.28(a2) the regulation is off and the RF remains tuned to 199.89 MHz, the plots depict respectively the real and imaginary components of the measurement. The voltage induced by the generator in open-loop contains some error caused by improper gain in the feedforward, that prepares the field in cavity before closing the regulation. The measured real component reaches a value of 0.1 MV and the imaginary component 0.25 MV. When the beam crosses the cavity, the beam-induced voltage is added

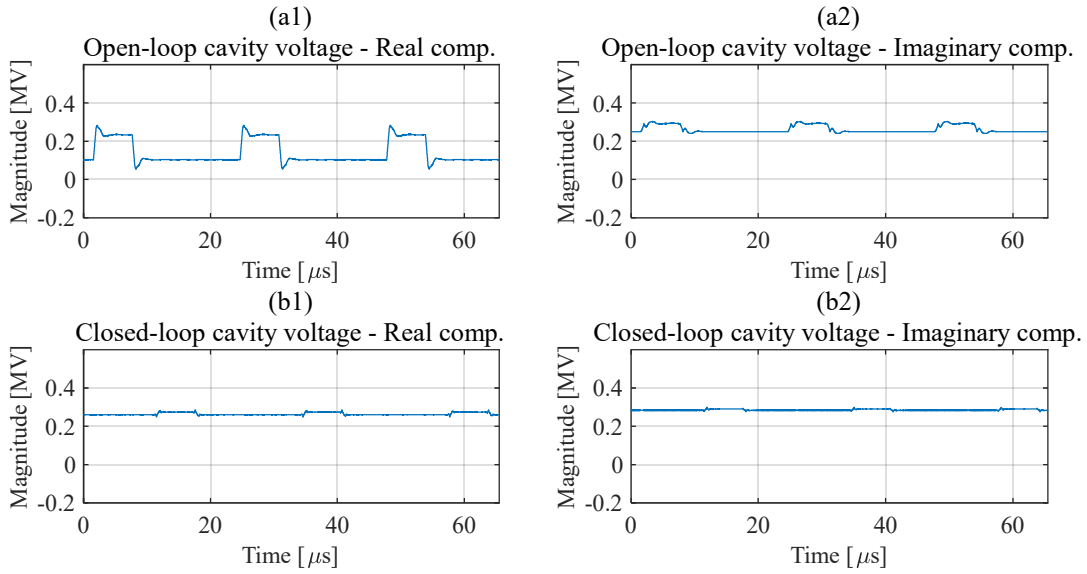


Fig. 6.28. Cavity voltage measured during 65 µs in Cartesian I (left) and Q (right) components: (a) Open-loop measurement, (b) closed-loop measurement. RF at 199.89 MHz.

to the generator one reaching 0.23 MV and 0.29 MV respectively for the real and imaginary components. When we close the regulation, the voltage in the cavity, when the beam does not cross it, reaches 0.26 MV and 0.28 MV respectively for the real and imaginary components (Fig. 6.28(b1) and Fig. 6.28(b2)). In this situation, with no beam, the regulation sets in the cavity a field close to the desired set-point. When the beam crosses the cavity, the measured voltage in the real component reaches 0.274 MV and the imaginary 0.29 MV. The reduction in beam loading is consistent with the gain $G = 4$ as the RF is not matched to the centre frequency of the cavity. These measurements agree with the simulations shown in Fig. 6.19.

We finally present in Fig. 6.29 a frequency-domain measurement of the Transient Beam Loading Compensation. The figure corresponds to an RF at 199.898 MHz (centre frequency of the Spectrum Analyzer span). It shows the main accelerating frequency (at the centre) and sidebands at the revolution harmonics induced by the *beam* signal (43.3 kHz spacing). The traces in red were captured when the compensation was not active (OTFB Off), while the blue traces have the OTFB active. The reduction in the revolution harmonics reaches 20 dB for the first few harmonics, a figure consistent with the gain set in the system (linear factor of 10).

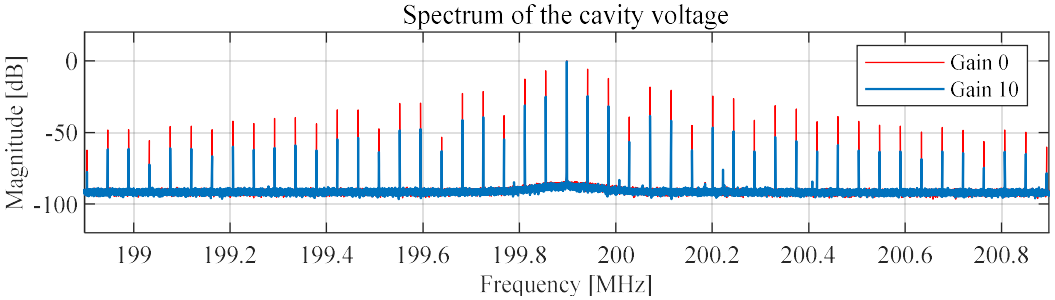


Fig. 6.29. Performance of the beam loading compensation: Spectrum of the cavity voltage with OTFB OFF (red trace) and OTFB ON (blue trace). The RF frequency is at 199.898 MHz so that the revolution harmonics induced by the beam are spaced by 43.3 kHz.

6.6. Conclusion

The chapter has presented the use of the Beam Synchronous Processing solution for Transient Beam Loading Compensation implementing the One Turn Feedback algorithm. The solution is well suited for new distributed network architectures, such as the one being commissioned at CERN for the SPS upgrade. The SPS architecture exploits deterministic protocols for data distribution and uses a network synchronous clock extracted from the deterministic link for network wide synchronization and synchronous processing.

This new OTFB implementation incorporates both BSP and Beam Asynchronous Processing (BAP) regions. The BAP contains filters whose functionalities are not related to the beam (compensation of amplifier frequency responses, calibrations of antenna, etc.), while the BSP tunes the algorithms to the beam revolution frequency automatically. The BSP and BAP regions support the porting of any new or existing algorithms, requiring no specific modifications of such algorithms. The BSP region adapts the data sampling rate to the beam revolution frequency. This avoids the reconfiguration of BSP algorithms in real-time. The data-path interfaces the BSP region by means of two resamplers. The input resampler performs the conversion of the fixed sampling rate, at which the data arrives into this BSP region, to a new rate proportional to the beam revolution frequency. At the output port, a second resampler brings the signal back to the original fixed rate. The resampling ratios of the resamplers are reciprocal (inverse) and vary dynamically during the acceleration ramp.

The new OTFB has been validated with system level simulations first. The filter is implemented in the BSP while the one-turn delay is in the BAP region. The performance is similar to the classic swept clock implementation. The resampler sandwich and the one-turn delay have also been validated in hardware. A new uTCA platform has been used, and laboratory tests have validated the correct tuning of the processing to the RF instantaneous value. No significant degradation of the signals spectral purity was observed as a consequence of the two resamplers. The CERN SPS accelerator is stopped till spring 2021 and no test with real beam will be possible before mid-2021. The beam was therefore emulated by injecting a perturbation, periodic at the revolution frequency, into the drive signal of a spare CERN SPS cavity, demonstrating the compensation of Transient Beam Loading with performances that agree with the simulations.

Chapter 7

Conclusions and Future Work

Abstract: *This chapter presents conclusions that the author has achieved after the study, design, implementation, validation and verification of the BSP Architecture. Suggestions for future work are also mentioned.*

7.1. Conclusions

A novel *Beam Synchronous Processing Architecture* that makes the treatment of periodic signals with known and varying fundamental frequency possible has been developed in this Thesis. The work responds to the BSP needs in the field of *Particle Accelerators*, especially at CERN, where the LLRF needs to accommodate the processing to the *slow* energy ramps found in hadron machines. The Architecture is a feasible alternative to the reconfiguration of the treatment algorithms; our technique performs *resampling* in the input data to tune the signal to the processing.

The Architecture suits seamless the two technological *paradigm changes* adopted for the new SPS LLRF system; the instantaneous value of the RF frequency is transmitted as a numerical word, the *Frequency Tuning Word*, using a deterministic network, *the White Rabbit*, and the reference signal is now a *fixed frequency clock* recovered from the WR. The Architecture uses the FTW to compute the ratio needed to *resample* the incoming signal. This operation results in a discrete signal spectrum matching the frequency at which the processing has been defined. The original sampling rate is recovered after processing. Thanks to the fixed frequency clock we can fully exploit State-Of-the-Art technologies such as the high-performance FPGAs in the new *uTCA* platform currently deployed in the new SPS LLRF systems.

We have demonstrated the new solution by implementing the *One Turn FeedBack Algorithm*; a simulation model of the SPS RF plant was used as a test bench for verification of the proper BSP operation. The simulations demonstrated the compensation of the *Transient Beam Loading* resulting from the beam-cavity interaction successfully. Then the uTCA implementation was validated with a *real SPS cavity* in a test stand. The system was successfully commissioned regulating the field in the cavity and compensating the perturbations of a simulated beam. The applications and results of the work are not limited to the One Turn FeedBack: The Architecture is also being commissioned in the Beam and Cavity controllers to interface and process data between beam synchronous and asynchronous domains.

7.1.1. Tangible contributions

We presented a few of the *many tangible contributions* and benefits that the new Architecture introduces for LLRF problems. Among these, RF gymnastics is now made simple; without the need for a swept clock, and with an absolute time reference at accelerator complex level, we can now compute the phase of the beam easily. This simplifies the synchronization and beam transfer and enables complex RF manipulations. The fixed frequency clock makes it possible to improve the spectral purity of the signals, as more precise PLL based architectures can be used to clean the references and clocks. This also improves the RF derived from these systems. In line with that, we can now exploit all the features of State-Of-the-Art FPGAs; we do not need the swept clock tuning the processing, that is performed by the BSP Architecture. We can also use any hardware macro or primitives, as for instance differential serial interfaces, that include or are based on PLLs that may unlock with swept clocks. This also facilitates the use of modern ADCs and DACs. We avoid the negative effects of phase jumps and interruptions in the LLRF. Many other use cases appear day after day enabled by the BSP Architecture and the paradigm changes that it makes possible to adopt.

7.1.2. Resampler

The Thesis has developed a new *All-Digital Synchronous Sampling Rate Conversion Architecture*. We have verified and validated its operation in both simulations and real hardware. Its resampling ratio accepts arbitrary values that can be modified in real-time. It can be configured for both up-sampling and down-sampling and to operate in transparent mode, being possible to change among any of these configurations dynamically. The hardware implementation uses a single fixed frequency system clock, making its implementation feasible for FPGA and ASICs. The accepted limits for the resampling ratio are dependent on the relation between the input sampling rate and the processing clock; if the frequency of the clock is the double of the input sampling rate, the ratio range is $R \in [0.5, 2]$. The input and output port signals are synchronous to this clock, and as the data-path is decoupled (it is implemented with the *FRANCISCO* fabric presented in Chapter 3), its sampling rate can be modified also in real-time, both at the input and output ports.

The resampler's multi-rate operation is based on the *DIANA Algorithm* implemented in the timing unit, that controls the interpolator unit, a *Variable Fractional Delay* filter. The *DIANA* computes the output sampling instant from the input reference and the resampling ratio. The VFD, implemented using a Farrow architecture, filters the input data with the required delay shift, in the plus or minus half a sampling period range. The input bandwidth of the VFD filter implementation has been optimized up to one third of the sampling rate because this was the required bandwidth for the targeted SPS OTFB application, but this can be customized. With the OTFB optimization, this same input bandwidth is accepted by the resampler for up-sampling and down-sampling ratios down to $R = 0.6$, decreasing monotonically to one fourth of the input sampling rate for $R = 0.5$.

The fixed-point implementation of the resampler using a sixteen bits data-path provides a SNR = 95 dB (very close to the theoretical SQNR = 98 dB) for a full-scale input. This performance is limited by the truncation error in the data-path. It could be upgraded to 110 dB by using eighteen bits without being limited by the current optimization of the VFD. The truncation in the ratio signal, implemented in thirty-two bits with twenty-nine fractional bits, results in a negligible maximum deviation in the output sampling frequency of $|\Delta f'_s / f'_s| = 3.7253 \cdot 10^{-10}$.

7.1.3. The BSP Architecture

Two resamplers have been combined in the so-called *resampling sandwich*. The application algorithm requiring synchronism with the input signal is placed in the middle defining the BSP Architecture. We have verified the sandwich functionality and validated its feasibility to tune the signal to the beam frequency and the static processing algorithm. The variable rate of the signal in the sandwich is controlled by the input resampler and recovers the original rate after the second resampler. It can easily be implemented in either FPGA or ASIC technology, as it only requires a fixed frequency system clock and the decoupled data-path, the *FRANCISCO* fabric. The *MERCEDES* interfaces handling the data-path clocking and the *JOAQUINA* loop controlling the inverse resampling ratios of the sandwich have also been validated. The combined operation of the sandwich and these entities performs the tuning of the processing in real-time. In the data-path region, between the *MERCEDES* interfaces and the resamplers, the Architecture can also host a BAP region to perform beam asynchronous processing. The combination of the BSP and BAP regions makes possible the implementation of any algorithms, including adaptive ones.

Very similar performance with a SNR = 93 dB is obtained at the output of the sandwich after the double resampling process when no algorithm is hosted in the BSP region. This fixed-point implementation uses the same quantization widths of the resampler and is still limited by the noise resulting from quantization of the data-path. The *JOAQUINA* loop around the resampling ratios corrects its value making them exactly inverse on average. The error in open-loop is very small and results in deviation in the recovered sampling frequency at the output of the sandwich of $|\Delta f''_s / f''_s| = 6.3862 \cdot 10^{-10}$. However, the loop is needed to prevent the de-synchronization in the *MERCEDES* interfaces arising from the truncation errors. As the correction magnitudes are in the order of one ratio LSB no modulation is observed in the output of

the sandwich. The implementation was tested also in a real system, implemented in a uTCA crate with a real SPS cavity. The operation and control of the Architecture were validated and it completed all the tests successfully.

7.1.4. The application of the BSP Architecture in the CERN SPS OTFB

The BSP Architecture has been benchmarked in *a real application in the CERN SPS LLRF system*. The OTFB algorithm was first simulated, then implemented in the new Architecture and finally commissioned in a uTCA crate controlling a real SPS cavity. The algorithm used both, the BSP and the BAP regions of the data-path, validating their combined operation. The controller operated as expected regulating the field in the cavity according to the configured regulation parameters. The regulation, and the BSP, were able to track in real-time an energy ramp with sweeping RF frequency. The system implemented the two architectural paradigms triggering the Thesis work, the distribution and synchronization of the instantaneous RF frequency of the LLRF with WR, and the use of a fixed frequency clock at 125 MHz regenerated from the WR. We simulated a beam by injecting a periodic perturbation at the revolution frequency in the drive signal of the cavity. The field in the cavity was measured with the regulation both in open and closed-loop; when the regulation was closed the beam loading was reduced as expected. The entire BSP Architecture and the complete uTCA implementation successfully passed all the functional validation and qualitative tests performed.

7.2. Future work

On the hardware side, future work needs to evaluate the performance obtained in the resampler when using other optimization methods for the VFD coefficients such as the Offset Window method [117], [118]. The implementation can be also optimized; the current VFD is implemented using the Direct-Form folded around the centre tap for the FIRs and the delay as the control parameter. Other architectures for the FIRs or in general for the entire VFD, such as the modified or transposed Farrow [89], could offer some optimization in the hardware implementation and computation.

On the application side, further algorithms can benefit from the BSP Architecture; longitudinal and transverse damper applications, for instance, as the solution facilitates the migration of any existing solution. In the case of the OTFB, we performed the test with the most basic comb filter at the revolution frequency. More complex filters such as triple comb of the SPS 800 MHz system dealing with the synchrotron sidebands to increase the beam stability could be tested.

The use of the BSP Architecture can also be extended to other fields of Science, where a frequency-sweeping signal needs to be processed such as in audio processing, mechanical noise removal, biomedical signal filtering, communications, etc. In these fields, the problem is normally addressed by reconfiguring the processing elements in real-time. The alternative solution presented here keeps all processing static avoiding the burden of real-time reconfiguration by using resampling to cope with the change in frequency.

References

[1] A. V. Oppenheim and R. W. Schafer, *Discrete-time signal processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.

[2] 'Algorithm definition', *Cambridge Dictionary*. https://dictionary.cambridge.org/dictionary/english/algorithm

[3] H. L. Krauss, C. W. Bostian, and F. H. Raab, *Solid state radio engineering*. New York, NY, USA: Wiley, 1980.

[4] B. Widrow and S. D. Stearns, *Adaptive signal processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1985.

[5] B. Widrow *et al.*, 'Adaptive noise cancelling: Principles and applications', *Proceedings of the IEEE*, vol. 63, no. 12, pp. 1692–1716, 1975, doi: 10.1109/PROC.1975.10036.

[6] B. Widrow and M. E. Hoff, 'Adaptive Switching Circuits', *IRE WESCON Convention Record*, pp. 96–104, 1960.

[7] H. Wiedemann, *Particle Accelerator Physics*. Cham, Germany: Springer International Publishing, 2015.

[8] S. Y. Lee, *Accelerator physics*. New Jersey, NJ, USA: World Scientific, 2019.

[9] P. Baudrenghien, 'Low level RF systems for synchrotrons: part I and II', presented at the CAS - CERN Accelerator School: Specialised Course on Radio Frequency Engineering, Seeheim, Germany, 2000, doi: 10.5170/CERN-2005-003.175.

[10] D. Boussard and G. Lambert, 'Reduction of the Apparent Impedance of Wide Band Accelerating Cavities by RF Feedback', *IEEE Transactions on Nuclear Science*, vol. 30, no. 4, pp. 2239–2241, Aug. 1983, doi: 10.1109/TNS.1983.4332774.

[11] S. Chatrchyan *et al.*, 'Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC', *Physics Letters B*, vol. 716, no. 1, pp. 30–61, Sep. 2012, doi: 10.1016/j.physletb.2012.08.021.

[12] J. Coupard *et al.*, 'LHC Injectors Upgrade. Technical Design Report Vol. I: Protons', CERN, Geneva, Switzerland, CERN-ACC-2014-0337, 2014. [Online]. Available: https://cds.cern.ch/record/1976692

[13] G. Dôme, 'The SPS acceleration system travelling wave drift-tube structure for the CERN SPS', CERN, Geneva, Switzerland, CERN-SPS-ARF-77-11, 1976. [Online]. Available: https://cds.cern.ch/record/319440

[14] J. Coupard *et al.*, 'LHC Injectors Upgrade. Technical Design Report Vol. II: Ions', CERN, Geneva, Switzerland, CERN-ACC-2016-0041, 2016. [Online]. Available: https://cds.cern.ch/record/2153863

[15] D. Boussard, 'Control of Cavities with High Beam Loading', *IEEE Transactions on Nuclear Science*, vol. 32, no. 5, pp. 1852–1856, Oct. 1985, doi: 10.1109/TNS.1985.4333745.

[16] P. Baudrenghien, G. Hagmann, and J. Molendijk, 'The LHC Low Level RF', CERN, Geneva, Switzerland, CERN-LHC-Project-Report-906, 2006. [Online]. Available: https://cds.cern.ch/record/971742

[17] P. Baudrenghien, J. Galindo, G. Hagmann, J. Noirjean, D. Stellfeld, and D. Valuch, 'Commissioning of the Linac4 Low Level RF and Future Plans', CERN, Geneva, Switzerland, CERN-ACC-2014-375, 2014. [Online]. Available: https://cds.cern.ch/record/2062609

[18] G. Hagmann *et al.*, 'The CERN SPS Low Level RF upgrade Project', presented at the Int. Particle Accelerator Conf. IPAC2019, Melbourne, Australia, 2019, doi: 10.18429/jacow-ipac2019-thprb082.

[19] PICMG, 'MicroTCA PICMG Specification MTCA.4.1 R1.0'. 2016.

References

[20] A. Abada *et al.*, 'FCC-hh: The Hadron Collider: Future Circular Collider Conceptual Design Report Volume 3', *The European Physical Journal Special Topics*, vol. 228, no. 4, pp. 755–1107, Jul. 2019, doi: 10.1140/epjst/e2019-900087-0.

[21] F. Zimmermann, 'Status and Challenges for FCC-ee', CERN, Geneva, Switzerland, CERN-ACC-2015-111, 2015.
[Online]. Available: https://cds.cern.ch/record/2057706

[22] B. G. Taylor, 'Timing Distribution at the LHC', presented at the 8th Workshop on Electronics for LHC Experiments, Colmar, France, 2002, doi: 10.5170/CERN-2002-003.63.

[23] J. M. Byrd, L. R. Doolittle, A. Ratti, J. W. Staples, and R. B. Wilcox, 'Timing Distribution in Accelerators via Stabilized Optical Fiber Links', presented at the LINAC 2006, Knoxville, Tennessee, USA, 2006.
[Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.463.2836

[24] J. Gill, 'RF distribution over White Rabbit. WR2RF - a replacement board for the BOBR.', presented at the High Precision Timing Distribution - Meeting #4, Geneva, Switzerland, 2018.
[Online]. Available:
https://indico.cern.ch/event/771447/contributions/3205240/attachments/1764135/2863327/wr2rf_hptd_03_12_2018.pdf

[25] J. Serrano *et al.*, 'The White Rabbit project', presented at the Int. Conference on Accelerator and Large Experimental Physics Control Systems ICALEPCS2009, Kobe, Japan, 2009.
[Online]. Available: https://accelconf.web.cern.ch/icalepcs2009/papers/tuc004.pdf

[26] M. Lipinski *et al.*, 'White Rabbit Applications and Enhancements', in *2018 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication ISPCS2018*, Geneva, Sep. 2018, pp. 1–7, doi: 10.1109/ISPCS.2018.8543072.

[27] 'Brookhaven National Laboratory'.
https://www.bnl.gov

[28] F. Severino, M. Harvey, T. Hayes, G. Narayan, and K. S. Smith, 'Distributed feedback loop implementation in the RHIC Low Level Platform', presented at the Int. Conference on Accelerator and Large Experimental Physics Control Systems ICALEPCS2013, San Francisco, CA, USA, 2014.
[Online]. Available: https://accelconf.web.cern.ch/icalepcs2013/papers/frcobab05.pdf

[29] 'Helmholtzzentrum für Schwerionenforschung'.
https://www.gsi.de

[30] H. Klingbeil, U. Laier, K.-P. Ningel, S. Schäfer, C. Thielmann, and B. Zipfel, 'New digital low-level rf system for heavy-ion synchrotrons', *Physical Review Special Topics - Accelerators and Beams*, vol. 14, no. 10, Oct. 2011, doi: 10.1103/PhysRevSTAB.14.102802.

[31] S. Baird, 'Accelerators for pedestrians', CERN, Geneva, Switzerland, AB-Note-2007-014, 2007.
[Online]. Available: https://cds.cern.ch/record/1017689

[32] H. Klingbeil, U. Laier, and D. Lens, *Theoretical Foundations of Synchrotron and Storage Ring RF Systems*. Cham, Germany: Springer International Publishing, 2015.

[33] J. W. Staples, 'Beam signal spectra, signal sampling, and noise', Univ. California, Santa Cruz, US Particle Accelerator School-Microwave Measurement and Beam Instrumentation Laboratory, 2008.
[Online]. Available: https://uspas.fnal.gov/materials/08UCSC/mml18_beam_signals.pdf

[34] D. Boussard, 'Beam Loading', presented at the CAS - CERN Accelerator School: Course on Advanced Accelerator Physics, Rhodes, Greece, 1993, doi: 10.5170/CERN-1995-006.

[35] R. Garoby, 'Beam loading in RF cavities', in *Frontiers of Particle Beams: Intensity Limitations*, vol. 400, M. Dienes, M. Month, and S. Turner, Eds. Berlin, Germany: Springer International Publishing, 1992, pp. 509–541.

[36] F. J. Galindo Guarch, J. M. Moreno Aróstegui, and P. Baudrenghien, 'An Architecture for Real-Time Arbitrary and Variable Sampling Rate Conversion with Application to Processing of Harmonic Signals', *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 5, pp. 1653–1666, May 2020, doi: 10.1109/TCSI.2019.2960686.

[37] A. Nehorai and B. Porat, 'Adaptive comb filtering for harmonic signal enhancement', *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 5, pp. 1124–1138, Oct. 1986, doi: 10.1109/TASSP.1986.1164952.

[38] M. Niedźwiecki and M. Meller, 'Generalized adaptive comb filters/smoothers and their application to the identification of quasi-periodically varying systems and signals', *Automatica*, vol. 49, no. 6, pp. 1601–1613, Jun. 2013, doi: 10.1016/j.automatica.2013.02.037.

[39] S.-C. Pei, Y.-D. Huang, S.-H. Lin, and J.-J. Shyu, 'Design of variable comb filter using FIR variable fractional delay element', *Signal Processing*, vol. 92, no. 10, pp. 2409–2421, Oct. 2012, doi: 10.1016/j.sigpro.2012.03.001.

[40] A. Gamp, W. Ebeling, W. Funk, J. R. Maidment, C. W. Planner, and G. H. Rees, 'The Radio Frequency System for Protons in HERA', presented at the 1st European Particle Accelerator Conference EPAC88, Rome, Italy, 1988.
[Online]. Available: https://accelconf.web.cern.ch/e88/PDF/EPAC1988_1105.pdf

[41] F. J. Galindo Guarch, J. M. Moreno Aróstegui, and P. Baudrenghien, 'Compensation of Transient Beam Loading in Ramping Synchrotrons using a Fixed Frequency Processing Clock', *Journal of Physics: Conference Series*, vol. 1067, p. 072033, Sep. 2018, doi: 10.1088/1742-6596/1067/7/072033.

[42] M. E. Angoletta *et al.*, 'CERN's LEIR Digital LLRF: System Overview and Operational Experience', presented at the 1st Int. Particle Accelerator Conf. IPAC2010, Kyoto, Japan, 2010.
[Online]. Available: https://accelconf.web.cern.ch/IPAC10/papers/TUPEA057.pdf

[43] J. Molendijk, 'Digital Receiver and Modulator Architecture for Multi-harmonic RF Finemet Operation', presented at the Low Level RF Workshop LLRF15, Shanghai, China, 2015.

[44] F. Bertin, Y. Brischetto, H. Damerau, A. Jibar, and D. Perrelet, 'Impedance reduction of the High-frequency Cavities in the CERN PS by Multi-harmonic Feedback', presented at the Low Level RF Workshop LLRF19, Chicago, IL, USA, 2019.
[Online]. Available: http://arxiv.org/abs/1910.06874

[45] J. Molendijk, 'Introducing Fixed Frequency Clock Operation on the CERN VXS LLRF Platform', presented at the Low Level RF Workshop LLRF17, Barcelona, Spain, 2017.
[Online]. Available: https://public.cells.es/workshops/www.llrf2017.org/pdf/Orales/O-22.pdf

[46] F. Tamura *et al.*, 'Multiharmonic vector rf voltage control for wideband cavities driven by vacuum tube amplifiers in a rapid cycling synchrotron', *Physical Review Accelerators and Beams*, vol. 22, no. 9, Sep. 2019, doi: 10.1103/PhysRevAccelBeams.22.092001.

[47] 'Japan Proton Accelerator Research Complex'.
https://j-parc.jp

[48] J. Dey, I. Kourbanis, J. Reid, and J. Steimel, '53 MHz Feedforward beam loading compensation in the Fermilab main injector', presented at the 2003 Particle Accelerator Conference, Portland, OR, USA, 2003, doi: 10.1109/PAC.2003.1289912.

[49] 'Fermi National Accelerator Laboratory'.
https://www.fnal.gov

[50] P. Baudrenghien and G. Lambert, 'Reducing the impedance of the Travelling Wave Cavities Feed-forward and one turn delay feed-back', presented at the 10th Workshop on LEP-SPS Performance, Chamonix, France, 2000.
[Online]. Available: https://cds.cern.ch/record/485863

[51] W. Hofle, 'Towards a transverse feedback system and damper for the SPS in the LHC era', presented at the Workshop on High Brightness Beams for Large Hadron Colliders, Montreux, Switzerland, 1996.
[Online]. Available: https://cds.cern.ch/record/326868

[52] W. Hofle *et al.*, 'Impact of a Wideband Feedback Prototype System on TMCI in the SPS', presented at the 9th Int. Particle Accelerator Conf., Vancouver, BC, Canada, 2018, doi: 10.18429/JACOW-IPAC2018-TUZGBD4.

[53] G. Apollinari, I. Béjar Alonso, O. Brüning, M. Lamont, and L. Rossi, 'High-Luminosity Large Hadron Collider (HL-LHC) Preliminary Design Report', CERN, Geneva, Switzerland, CERN-2015-005, 2015.
[Online]. Available: http://cds.cern.ch/record/2116337

[54] H. Abramowicz and R. Forty, 'Physics Briefing Book : Input for the European Strategy for Particle Physics Update 2020', CERN, Geneva, Switzerland, CERN-ESU-004, 2019.
[Online]. Available: https://cds.cern.ch/record/2691414

[55] T. Wlostowski, J. Serrano, G. Daniluk, M. M. Lipinski, and F. Vaga, 'Trigger and RF Distribution Using White Rabbit', presented at the 15th Int. Conf. on Accel. and Large Experimental Control Systems, ICALEPCS2015, Melbourne, Australia, 2015, doi: 10.18429/JACoW-ICALEPCS2015-WEC3O01.

[56] R. Garoby, 'RF gymnastics in synchrotrons', presented at the CAS - CERN Accelerator School: Specialised Course on RF for Accelerators, Ebeltoft, Denmark, 2010, doi: 10.5170/CERN-2011-007.431.

[57] A. W. Chao, Ed., *Handbook of accelerator physics and engineering*. Hackensack, New Jersey, USA: World Scientific, 2013.

[58] D. Quartullo, T. Argyropoulos, and A. Lasheen, 'Momentum Slip-Stacking Simulations for CERN SPS Ion Beams with Collective Effects', presented at the 61st ICFA ABDW on High-Intensity and High-Brightness Hadron Beams, HB2018, Daejeon, Korea, 2018, doi: 10.18429/jacow-hb2018-tup2wa02.

[59] T. Ferrand, H. Klingbeil, and H. Damerau, 'Synchronization of Synchrotrons for bunch-to-bucket Transfers', CERN, Geneva, Switzerland, CERN-ACC-NOTE-2015-0025, 2015.
[Online]. Available: https://cds.cern.ch/record/2053285

[60] T. Hayes, F. Severino, and K. S. Smith, 'A Deterministic, Gigabit Serial Timing, Synchronization and Data Link for the RHIC LLRF', presented at the Particle Accel. Conf PAC11, New York, NY, USA, 2011.
[Online]. Available: https://inspirehep.net/literature/1187389

[61] T. Hayes, M. Harvey, G. Narayan, F. Severino, K. S. Smith, and S. Yuan, 'A High Performance DAC/DDS Daugther Module for the RHIC LLRF Platform', presented at the Particle Accel. Conf PAC11, New York, NY, USA, 2011.
[Online]. Available: https://inspirehep.net/literature/1187387

[62] XILINX, 'UltraScale Architecture Clocking Resources User Guide', User Guide UG572, 2018.

[63] M. E. Angoletta *et al.*, 'CERN's PS Booster LLRF Renovation: Plans and Initial Beam Tests', presented at the 1st Int. Particle Accelerator Conf. IPAC2010, Kyoto, Japan, 2010.
[Online]. Available: https://cds.cern.ch/record/1287908

[64] T. Hentschel, *Sample rate conversion in software configurable radios*. Norwood, MA, USA: Artech House, 2002.

[65] F. Sheikh and S. Masud, 'Efficient sample rate conversion for multi-standard software defined radios', in *Proc IEEE International Conf. Acoustics, Speech and Signal Processing, ICASSP 2007*, Honolulu, HI, USA, 2007, vol. 2, p. II–329.
[Online]. Available: http://ieeexplore.ieee.org/abstract/document/4217412

[66] Long and Torres, 'High Throughput Farrow Re-samplers Utilizing Reduced Complexity FIR Filters', presented at the IEEE Military Comms Conf. MILCOM12, Orlando, FL, USA, 2012.
[Online]. Available: https://ieeexplore.ieee.org/document/6415616

[67] B. Markovic and J. Certic, 'FPGA realization of Farrow structure for sampling rate change', *Serbian Journal of Electrical Engineering*, vol. 13, no. 1, pp. 83–93, 2016, doi: 10.2298/SJEE1601083M.

[68] A. Tkacenko, 'Variable sample rate conversion techniques for the Advanced Receiver', *Interplanetary Network (IPN) Progress Report*, vol. 42, p. 168, 2007.

[69] M. A. Siddiqi, N. Samad, S. Masud, and F. Sheikh, 'FPGA-based Implementation of Efficient Sample Rate Conversion for Software Defined Radios', in *Proc. 10th IEEE Int. Conf. Comp. Info. Tech*, Bradford, UK, Jun. 2010, pp. 2387–2390, doi: 10.1109/CIT.2010.410.

[70] ALTERA, 'Using the DSP Builder AdvancedBlockset to Implement Resampling Filters', Altera Application Note AN 623, 2010.

[71] U. Meyer-Baese, *Digital signal processing with field programmable gate arrays*. Berlin, Germany: Springer, 2007.

[72] F. Harris, *Multirate signal processing for communication systems*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

[73] M. Blok and P. Drózda, 'Variable Ratio Sample Rate Conversion Based on Fractional Delay Filter', *Archives of Acoustics*, vol. 39, no. 2, Jan. 2015, doi: 10.2478/aoa-2014-0027.

[74] R. Adams and T. Kwan, 'Theory and VLSI Architectures for Asynchronous Sample-Rate Converters', *J. Audio Eng. Soc.*, vol. 41, no. 7/8, 1993.
[Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=6993

[75] G. Evangelista, 'Design of digital systems for arbitrary sampling rate conversion', *Signal Processing*, vol. 83, no. 2, pp. 377–387, Feb. 2003, doi: 10.1016/S0165-1684(02)00421-8.

[76] A. Chinaev, P. Thune, and G. Enzner, 'Low-Rate Farrow Structure with Discrete-Lowpass and Polynomial Support for Audio Resampling', presented at the 26th European Signal Processing Conference, EUSIPCO, Rome, Italy, 2018, doi: 10.23919/EUSIPCO.2018.8553469.

[77] C. W. Farrow, 'A continuously variable digital delay element', in *IEEE International Symposium on Circuits and Systems*, 1988, pp. 2641–2645.
[Online]. Available: http://ieeexplore.ieee.org/abstract/document/15483

[78] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine, 'Splitting the unit delay [FIR/all pass filters design]', *IEEE Signal Processing Magazine*, vol. 13, no. 1, pp. 30–60, Jan. 1996, doi: 10.1109/79.482137.

[79] D. E. Knuth, *Evaluation of polynomials, in The art of computer programming*. Boston, MA, USA: Addison Wesley, 1997.

[80] A. Gamp, 'Servo Control of Cavities under Beam Loading', presented at the CAS - CERN Accelerator School: Specialised Course on RF for Accelerators, Seeheim, Germany, 2000.
[Online]. Available: https://arxiv.org/ftp/arxiv/papers/1303/1303.1358.pdf

[81] F. Tamura *et al.*, 'Multiharmonic rf feedforward system for compensation of beam loading and periodic transient effects in magnetic-alloy cavities of a proton synchrotron', *Physical Review Special Topics - Accelerators and Beams*, vol. 16, no. 5, May 2013, doi: 10.1103/PhysRevSTAB.16.051002.

[82] F. Tamura *et al.*, 'Multiharmonic rf feedforward system for beam loading compensation in wide-band cavities of a rapid cycling synchrotron', *Physical Review Special Topics - Accelerators and Beams*, vol. 14, no. 5, May 2011, doi: 10.1103/PhysRevSTAB.14.051004.

[83] M. Lonza, 'Multi-bunch feedback systems', presented at the CAS - CERN Accelerator School: Course on Digital Signal Processing, Sigtuna, Sweden, 2007.
[Online]. Available: https://cds.cern.ch/record/1100539

[84] J. Fox and E. Kikutani, 'Bunch Feedback Systems and Signal Processing', in *Join US-CERN-Japan-Russia School on Particle Accelerators*, Montreux, Switzerland, 1998, pp. 579–620.

[85] T. Schilcher, 'RF applications in digital signal processing', presented at the CAS - CERN Accelerator School: Specialised Course on Digital Signal Processing, Sigtuna, Sweden, 2007.
[Online]. Available: https://cds.cern.ch/record/1100538

[86] R. Garoby, 'Low-Level RF and feedback', *Joint CERN-US-Japan Accelerator School: Course on Frontiers of Accelerator Technology: RF Engineering for Particle Accelerators*, 1997.

[87] F. Tamura *et al.*, 'Multiharmonic vector rf voltage control for wideband cavities driven by vacuum tube amplifiers in a rapid cycling synchrotron', *Physical Review Accelerators and Beams*, vol. 22, no. 9, Sep. 2019, doi: 10.1103/PhysRevAccelBeams.22.092001.

[88] D. G. Manolakis and V. K. Ingle, *Applied digital signal processing: theory and practice*. New York, NY, USA: Cambridge University Press, 2011.

[89] A. Franck, K. Brandenburg, J. O. Smith III, and V. Välimäki, 'Efficient algorithms for arbitrary sample rate conversion with application to wave field synthesis', Univ.-Verl. Ilmenau, Ilmenau, 2012.

[90] A. V. Oppenheim, A. S. Willsky, and S. Hamid Nawab, *Signals and systems*. Harlow, Essex, UK: Pearson Education, 2013.

[91] J. Belleman, 'From analog to digital', presented at the CAS - CERN Accelerator School: Specialised Course on Digital Signal Processing, Sigtuna, Sweden, 2007.
[Online]. Available: https://cds.cern.ch/record/1100535

[92] M. Bellanger, 'Multirate digital signal processing', *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 4, pp. 941–941, 1984.

[93] W. Kester, 'What the Nyquist Criterion Means to Your Sampled Data System Design', Analog Devices MT-002 Tutorial, 2008.

[94] J. Serrano, 'Introduction to FPGA design', presented at the CAS - CERN Accelerator School: Specialised Course on Digital Signal Processing, Sigtuna, Sweden, 2007.
[Online]. Available: https://cds.cern.ch/record/1100537

[95] A. DeHon and J. Adams, *Design Patterns for Reconfigurable Computing*, FCCM 2004: 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines. Los Alamitos, CA, USA, 2004.

[96] XILINX, '7 Series DSP48E1 Slice User Guide', User Guide UG479, 2018.

[97] K. Pearson, 'The Problem of the Random Walk', *Nature*, vol. 72, no. 1865, pp. 294–294, Jul. 1905, doi: 10.1038/072294b0.

[98] V. Valimaki, 'Discrete-Time Modeling of Acoustic Tubes Using Fractional Delay Filters', Helsinki University of Technology, Helsinki, Finland, 1995.

[99] G. M. Phillips, *Interpolation and approximation by polynomials*. New York, NY, USA: Springer, 2003.

[100] R. E. Crochiere and L. R. Rabiner, *Multirate digital signal processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1983.

[101] R. E. Crochiere and L. R. Rabiner, 'Interpolation and decimation of digital signals—A tutorial review', *Proceedings of the IEEE*, vol. 69, no. 3, pp. 300–331, 1981.

[102] P. P. Vaidyanathan, *Multirate systems and filter banks*. Englewood Cliffs, NJ, USA: Prentice Hall, 1993.

[103] C. E. Shannon, 'Communication in the Presence of Noise', *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, Jan. 1949, doi: 10.1109/JRPROC.1949.232969.

[104] T.-B. Deng and Y. Lian, 'Weighted-Least-Squares Design of Variable Fractional-Delay FIR Filters Using Coefficient Symmetry', *IEEE Transactions on Signal Processing*, vol. 54, no. 8, pp. 3023–3038, Aug. 2006, doi: 10.1109/TSP.2006.875385.

[105] XILINX, 'UltraScale Architecture DSP Slice', User Guide UG579, 2019.

[106] Struck, 'SIS8300-KU MTCA.4 Digitizer'.
[Online]. Available: https://www.struck.de/sis8300-ku.html

[107] Struck, 'DS8VM1 MTCA.4 Direct Sampling/Vector Modulator RTM'.
[Online]. Available: https://www.struck.de/ds8vm1.html

[108] 'The Nobel Prize in Physics 1984'.
https://www.nobelprize.org/prizes/physics/1984/press-release

[109] 'The CERN SPS'.
https://home.cern/science/accelerators/super-proton-synchrotron

[110] D. Boussard, J. M. Brennan, and T. Linnecar, 'Fixed Frequency Acceleration in the SPS', CERN, Geneva, Switzerland, CERN-SPS-89-49-ARF, 1989.
[Online]. Available: https://cds.cern.ch/record/204684

[111] P. Baudrenghien, 'The new SPS LLRF', Geneva, Switzerland, 2020.
[Online]. Available: https://indico.cern.ch/event/895500

[112] P. Baudrenghien and T. Mastoridis, 'I/Q Model of the SPS 200 MHz Travelling Wave Cavity and Feedforward Design', CERN, Geneva, Switzerland, CERN-ACC-NOTE-2020-0032, 2020.
[Online]. Available: https://cds.cern.ch/record/2719232

[113] J. Gill, 'RF signal distribution over White Rabbit.', Geneva, Switzerland, 2019.
[Online]. Available:
https://indico.cern.ch/event/865008/attachments/1949767/3236439/BE_seminar_WR_Applications-RFoWR.pdf

[114] F. J. Galindo Guarch, P. Baudrenghien, and J. M. Moreno Arostegui, 'A new beam synchronous processing architecture with a fixed frequency processing clock. Application to transient beam loading compensation in the CERN SPS machine', *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 988, p. 164894, Feb. 2021, doi: 10.1016/j.nima.2020.164894.

[115] P. Baudrenghien *et al.*, 'SPS Architecture', CERN, Geneva, Switzerland, BE-RF-FB Internal Report, 2020.

[116] P. Baudrenghien and T. Mastoridis, 'Fundamental cavity impedance and longitudinal coupled-bunch instabilities at the High Luminosity Large Hadron Collider', *Physical Review Accelerators and Beams*, vol. 20, no. 1, Jan. 2017, doi: 10.1103/PhysRevAccelBeams.20.011004.

[117] A. Yardim, G. D. Cain, and P. Henry, 'Optimal two-term offset windowing for fractional delay', *Electronics Letters*, vol. 32, no. 6, pp. 526–527, 1996.

[118] Blok, 'Fractional delay filter design with extracted window offsetting', presented at the 19th International Conference Mixed Design of Integrated Circuits and Systems - MIXDES 2012, Warsaw, Poland, May 2012.
[Online]. Available: https://ieeexplore.ieee.org/document/6226239

In Particle Accelerators, the Low-Level RF (LLRF) is the control system of the RF, and in the end, of the purpose of the machine, that is the energy transfer and acceleration of particles. It implements algorithms synchronizing the RF conveying the energy to the beam and tailoring its longitudinal parameters. For this, the LLRF uses beam-related signals whose spectral content changes during the acceleration. The increase in energy results in an increase of the beam velocity, and for circular accelerators (Synchrotrons) a decrease in revolution period. This is especially relevant for Hadron machines whose injection energy is low resulting in a significant increase of their velocity before reaching relativistic speeds. Hence, the LLRF needs to continuously tune its processing to the beam; we call this technique Beam Synchronous Processing.

The Thesis presents a novel Beam Synchronous Processing Architecture, using a fixed frequency clocking, and capable of treating periodic signals with known and varying fundamental frequency. The Architecture is an alternative to the burden of reconfiguration in processing algorithms; it tunes the spectrum to the processing by resampling the input data. Two Resamplers are combined in the so-called resampling sandwich. The application algorithm requiring synchronism with the input signal is placed in the middle. The key element is a novel All-Digital Farrow-based Resampler, that accepts arbitrary resampling ratios that can be modified in real-time. The hardware uses a single fixed frequency system clock, making its implementation feasible in State-Of-the-Art FPGAs, ASICs and systems such as the new uTCA platform currently being deployed in the CERN SPS LLRF system. The input and output ports of the Resampler, and all the processing within the Architecture, are synchronous to this fixed frequency clock and accept data streams whose sampling rate can be variable and modified in real time.

The Architecture suits seamless the two technological paradigm changes adopted for the new CERN SPS LLRF system; first, the instantaneous value of the RF frequency is transmitted as a numerical word (used to set the resampling ratio), via a deterministic network, the White Rabbit. And second, the reference signal is now the fixed frequency clock recovered from this network. Both paradigms benefit from the all-digital Resampler and the Beam Synchronous Architecture that fulfil the techniques and technological needs for its implementation enabling novel LLRF algorithms and solutions.