transmisión y del tiempo de carga en los latches.

En la figura 2.6 mostramos la utilización de los buses, durante 2 ciclos (no hacemos distinción entre accesos para lectura o escritura), para 8 procesadores con peticiones activas a 8 módulos de memoria diferentes y 2 buses. Vemos como durante dos ciclos, los 8 procesadores han podido acceder a los 8 módulos de memoria, utilizando los 2 buses en forma multiplexada, mientras que sin multiplexar, sólo se hubièran podido realizar 4 accesos.

2.3 Sincronización de los procesadores

La realización de las funciones del sistema de arbitraje, debe tener en cuenta las diferentes formas o modos de sincronización entre los procesadores y las diferentes implicaciones para compartir el bus.

Describimos los diferentes modos de sincronización que hemos considerado. Estos modos estan relacionados como se muestra en la figura 2.7.

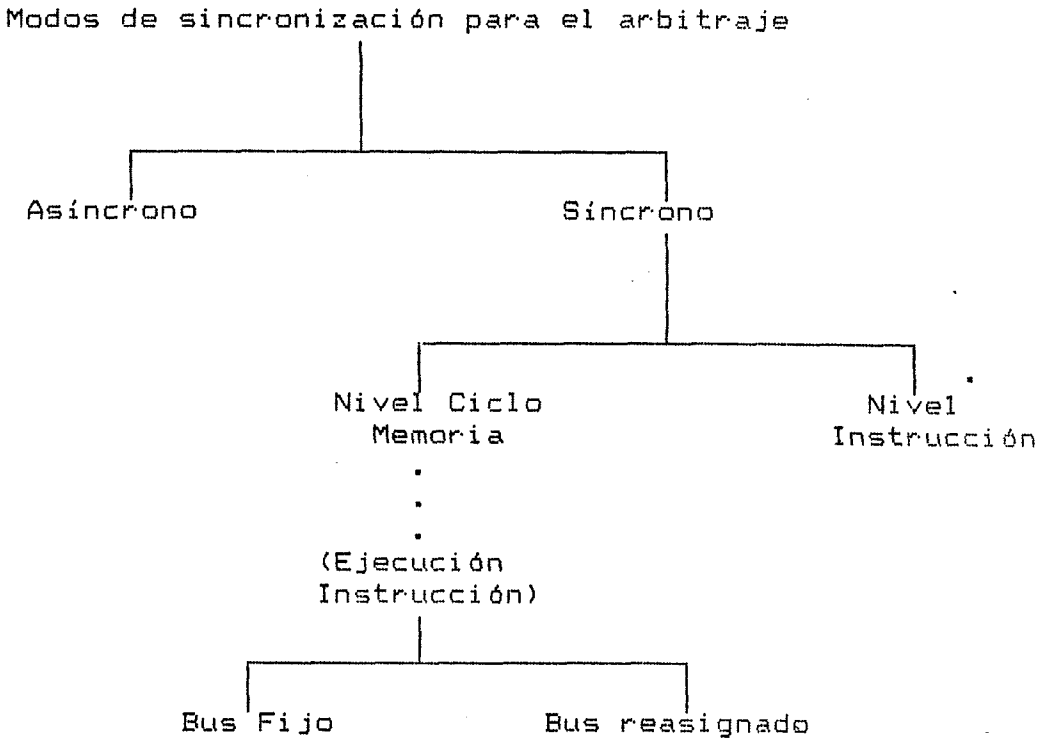


Figura 2.7 Modos de sincronización.

Asíncrono : Cada procesdor tiene su propio reloj, por lo tanto las peticiones de los procesadores pueden activarse en cualquier instante de tiempo y tener diferentes duraciones. En el momento de activarse una petición, pueden existir buses ocupados, asignados a otras peticiones activas.

Para los sistemas asíncronos nosotros proponemos el sistema de arbitraje centralizado, utilizando una red de interconexión completa. Para utilizar los esquemas reducidos, en un sistema asíncrono, se tendría que esperar a que todos los buses estuvieran libres para realizar una nueva asignación, no atendiendo a nuevas peticiones hasta que esto ocurra. Funcionando, por lo tanto el sistema en forma degradada.

Es muy complejo utilizar con un funcionamiento asíncrono los buses en forma multiplexada, puesto que el árbitro no tiene información sobre los momentos específicos de utilización del bus, ya que dependeran de cada procesador.

Síncrono : Existe un reloj que utilizan todos los procesadores del sistema. Este reloj nos indica los

ciclos de acceso al bus. En cada ciclo de reloj, se puede realizar un acceso desde cualquier procesador a un módulo de memoria común, a través de los buses múltiples. Normalmente este ciclo es igual al ciclo del procesador y de los módulos de memoria.

En un sistema síncrono todas las peticiones se activan o desactivan, en momentos determinados respecto al reloj del sistema, permaneciendo activas, como mínimo, durante un ciclo. Quedando por lo tanto definidos, dentro de un ciclos los momentos de arbitraje.

Los sistemas sícronos, pueden utilizar los sistemas de arbitraje centralizados o distribuídos que proponemos, utilizando tanto las redes de interconexión completas como las reducidas. Tambien pueden utilizar los buses en forma multiplexada.

Este modo general puede ser dividido en dos tipos, dependiendo de la forma de activación y desactivación de las peticiones:

a)**Síncronos a nivel de instrucción:** En esta forma de sincronización, se activan las peticones al comenzar un ciclo de búsqueda de la instrucción. La petición permanece activa, durante todos los ciclos de

la instrucción.

Este tipo de sincronización es necesario utilizarlo, cuando la ejecución de la instrucción es indivisible, por lo tanto, estas instrucciones no pueden ser ejecutadas ciclo a ciclo.

Puesto que normalmente, no todas las instrucciones duran el mismo número de ciclos, todas las peticiones de los diferentes procesadores, no se activaran y desactivarán simultáneamente.

En el instante de activar una petición pueden existir buses ocupados, asignados a peticiones y existentes en ciclos anteriores. Esto impide usar directamente los esquemas reducidos, ya que estos esquemas necesitan que todos los buses estén libres al realizar la asignación.

Para poder utilizar los esquemas reducidos, todas las instrucciones deben comenzar en el mismo ciclo, para que estén todos los buses libres.

Al no durar todas las instrucciones el mismo tiempo, para sincronizar en un mismo ciclo la activación de las peticiones, habrá que parar a los procesadores que han ejecutado las instrucciones más cortas, hasta que se termine de ejecutar la más larga. De este modo, podrán competir en el mismo ciclo, todas las peticiones, estando todos los buses libres.

Vemos ,que utizar los esquemas de

interconexión reducidos, con esta forma de sincronización tiene el inconveniente de que el sistema funciona a la velocidad de la instrucción más lenta.

b) **Síncronos a nivel de ciclo de memoria:** El sistema de arbitraje comienza una nueva operación de asignación de bus, cada vez que el reloj del sistema indica que comienza un ciclo.

En la ejecución de programas concurrentes en sistemas multiprocesadores, la estructura debe garantizar los accesos multiciclos a un recurso común, necesarios para la ejecución de operaciones indivisibles, como por ejemplo la ejecución de las instrucciones de "test and set". En estos accesos en los que la petición es mantenida durante más de un ciclo, consideramos dos solucciones diferntes en este nivel:

-Bus fijo : El bus inicialmente asignado, es mantenido a lo largo de todos los ciclos de ejecución de la instrucción. El procesador mantiene la petición fija durante más de 1 ciclo. En general, cuando un nuevo ciclo de memoria comienza, no todos los buses estan libres.

-Bus reasignado : En cada ciclo de la instrucción el sistema de arbitraje debe asignar un bus. Cuando un nuevo ciclo de memoria comienza, todos

los buses quedan libres, pero las peticiones para asignación de bus, correspondientes a instrucciones y en ejecución, son prioritarias. De este modo se garantiza, que todos los ciclos de una instrucción, tendran bus asignado en ciclos consecutivos, pudiendo estar asignados buses diferentes en los diferentes ciclos.

2.4 Gestion de buses

En los sistemas multiprocesadores en los que la conexión entre procesadores y memoria está realizada por un conjunto de buses (buses múltiples), existen unos recursos comunes compartidos por todos los procesadores (módulos de memoria y buses). La mayoría de estos recursos no permiten un acceso simultáneo. Debido a esta característica es necesario tener un sistema de arbitraje que decida, mediante un algoritmo, la asignación de los recursos compartidos.

El esquema de conexión más característico para el esquema de buses multiples, es en el que cada procesador está conectado a todos los buses y cada bus a todos los módulos de memoria, de modo que un procesador puede acceder a cualquier mòdulo de memoria

a través de cualquier bus compartido, si el módulo direccionado está libre y existe algún bus disponible.

Existen dos formas de obtener la asignación de un bus, para la comunicación entre un procesador y un módulo de memoria compartida :

a.- Solicitando la asignación de un bus determinado.

b.- Solicitando el acceso al módulo de memoria, siendo automática la asignación del bus que va a utilizar.

Nosotros hemos elegido la segunda opción, siendo por lo tanto, la asignación del bus transparente al usuario.

Proponemos un sistema de arbitraje realizado por hardware, para no añadir tiempo extra a un ciclo del sistema, de modo que en un ciclo se realice el arbitraje y el acceso a memoria sin degradar la velocidad del sistema. Esta unidad proporciona las funciones de control en las redes de interconexión, es decir, controla todos los puntos de conexión. Suponemos que todos los módulos de memoria son uni-entrada, de modo que sólo un procesador puede acceder a un módulo de memoria en un momento dado.

La figura 2.8 presenta un diagrama de bloques del sistema de arbitraje. cada procesador Pi (i=1,2,...,p), tiene asociadas las siguientes líneas:

una "Línea de petición Ri" que indica cuando el procesador i quiere realizar un acceso a un módulo de memoria común.

Un "código binario MPi" que indica el módulo de memoria al que desea acceder. Por lo tanto un procesador, no puede hacer una nueva petición a otro módulo de memoria, sin quitar la petición actual.

Una "línea de concesión Ai" que le indica cuando puede acceder al módulo de memoria pedido.

El sistema de arbitaje está encargado de realizar las siguientes funciones que son ejecutadas en paralelo:

a) Selección del procesador cuando existen varias peticiones simultáneas a un mismo módulo de memoria común.

Si en un ciclo varios procesadores generan direcciones para acceder al mismo módulo de memoria,sólo uno de ellos puede ser atendido. Para seleccionar uno de los procesadores se utilizan árbitros "p a 1", puesto que existen "p" entradas, cada una asociada a un procesador y sólo una puede obtener

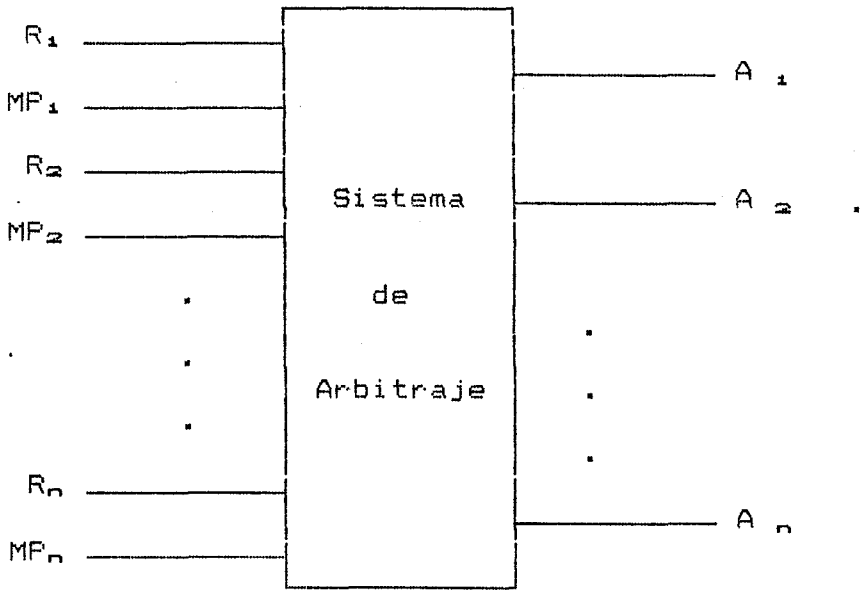


Fig. 2.8 Sistema de Arbitraje

concesión.

b) Asignación de un bus para una petición a un módulo de memoria común, elegiendo uno de los buses libres.

Cada petición corresponde a una petición hecha por uno o más procesadores a un módulo de memoria común M_k (k= 1,2,...,m).

Es necesario asignar un pool de buses B_j (j=1,2,...,b) a "i" peticionarios (i<=k).

Para realizar la asignación se utiliza un árbitro "m usuarios b servidores". Las salidas de este árbitro indican el bus que se le ha concedido a cada módulo de memoria con petición.

El sistema de arbitraje propuesto (Luq-84a) es regular, modular (la estructura está basada en bloques básicos), rápido y ofrece capacidad de expansión (es facilmente reconfigurable, permitiendo incrementar el número de buses, módulos de memoria y procesadores). Este sistema de arbitraje puede ser usado en sistemas síncronos y asíncronos y puede manejar redes de interconexión multibus completas ,reducidas y multiplexadas.

2.5 Estructura del sistema de arbitraje

La figura 2.9 presenta una primera descomposición

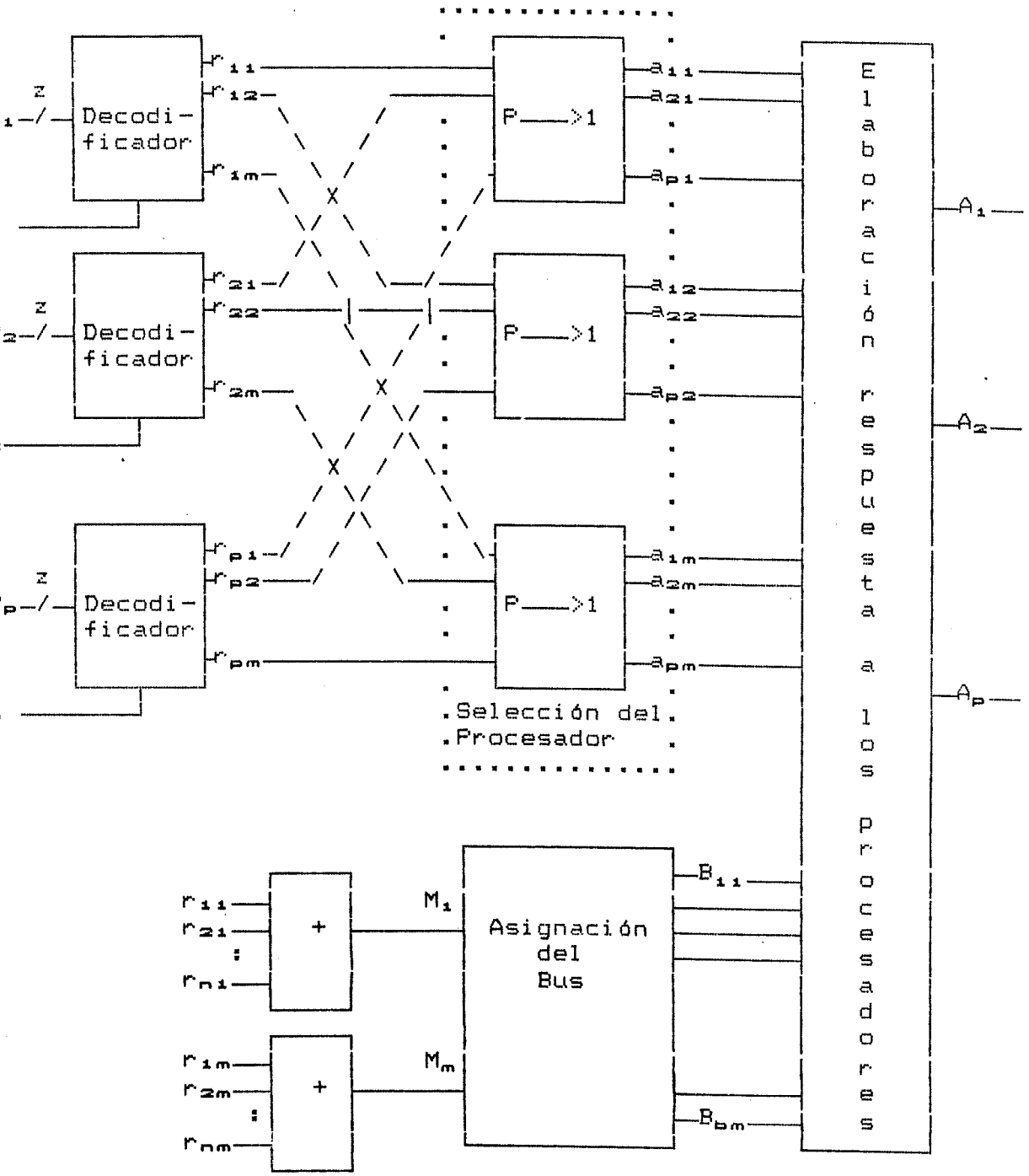


Fig. 2.9 Estructura del sistema de arbitraje

en bloques del árbitro, donde se reflejan las dos funciones que realiza en paralelo. Además de las entradas y salidas ya descritas, existen las siguientes variables internas:

Las salidas de los decodificadores r_{ik}, indican que el procesador i quiere acceder al módulo de memoria k. Estas señales son las entradas a los "m" árbitros que realizan la selección del procesador, que nos proporcionan las señales de concesión a_{ik} , indicando que el procesador i ha sido el seleccionado para acceder al módulo de memoria k.

Las señales "M_k" (k=1,2,...,m) indican que exite al menos una petición para acceder al módulo de memoria k. Estas señales son las entradas al árbitro "m usuarios b servidores" que realiza la aignación del bus. Las salidas de este dispositivo son las señales "B_{jk}" indicando el bus j que ha sido asignado para acceder al módulo de memoria k.

Las salidas de ambos módulos son las entradas al circuito que elabora la respuesta a los procesadores.

2.5.1 Selección del procesador

Existiran funcionando en paralelo, tantos árbitros "p a 1" como módulos de memoria existan en el sistema multiprocesador. Las entradas a estos árbiros son las señales de petición r_{ik} y las salidas a_{ik} son las señales que especifican el procesador seleccionado para acceder a cada módulo de memoria.

El algoritmo de arbitraje dependerá de los árbitros "p a 1" elegidos. En este trabajo proponemos, dos circuitos de arbitraje para la selección del procesador, uno centralizado y otro distribuído.

2.5.2 Elaboración de la respuesta a los procesadores.

Un procesador "i" pidiendo acceso al módulo de memoria k ($r_{ik}=1$), obtiene contestación afirmativa "$A_i=1$", si el procesador i es el seleccionado para acceder al módulo de memoria k, ($a_{ik}=1$) y en el circuito de asignación del bus , el módulo de memoria k obtiene algun bus j ($B_{jk}=1$).

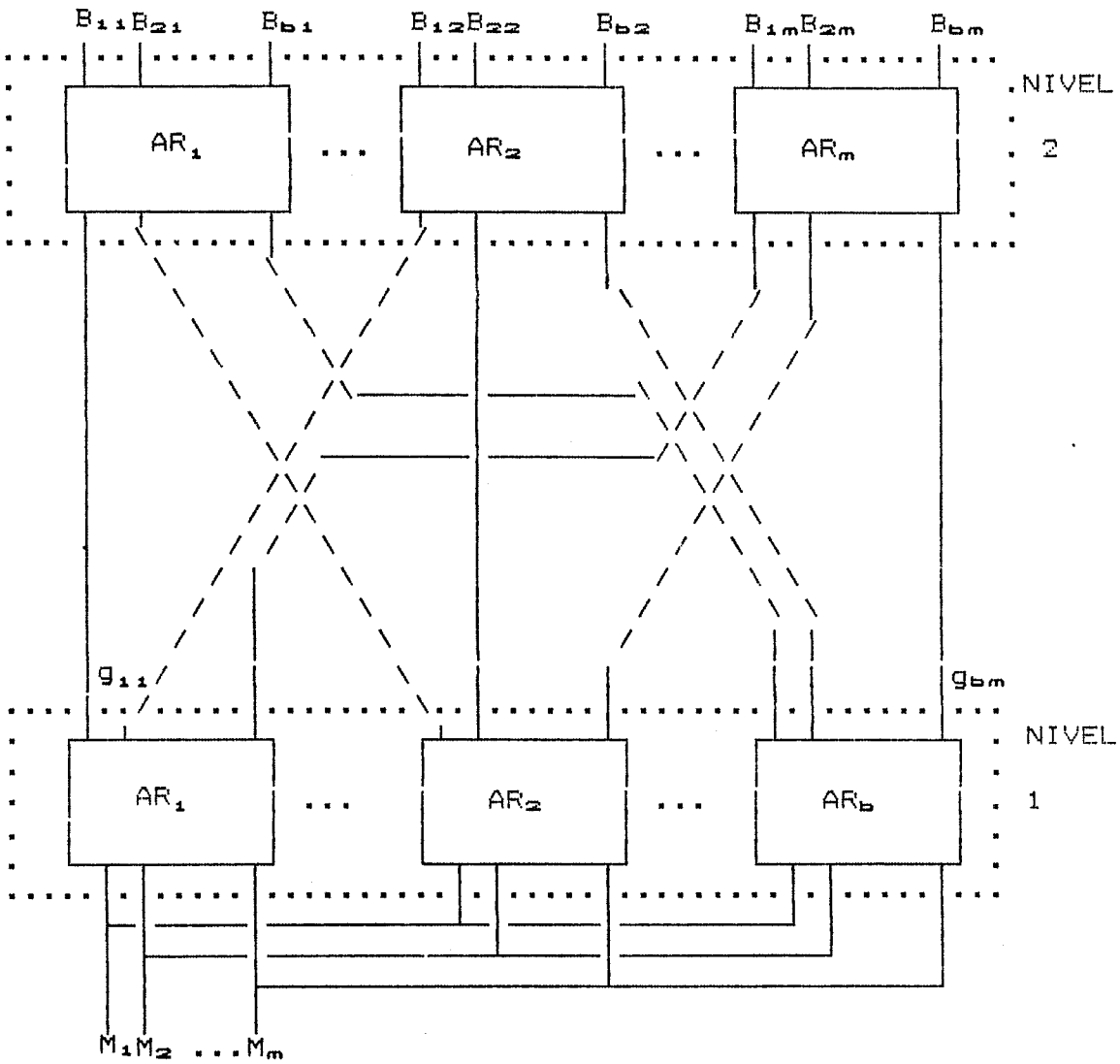
2.5.3 Asignación de buses

La estructura básica del dispositivo de asignación de bus está representada en la figura 2.10. Es una estructura simple, modular y regular, basicamente sólo usa un tipo de módulos (árbitros "n a 1")

El dispositivo de asignación de bus es un circuito asíncrono, que asigna bus a las peticiones que llegan en función de los buses libres. Una vez ha asignado el bus, éste no se libera hasta que no cese la petición.

En el caso de que existan más peticones que buses en el sistema, las peticiones que han quedado sin bus serán las más prioritarias en los buses que se vayan liberando.

En este circuito existe una realimentación desde las salidas B_{jk} a las entrada M_k. Esta realimentación garantiza que en el momento en el que a un módulo de memoria se le haya concedido un bus, éste quede asignado para todo el ciclo de acceso, y que el módulo de memoria no deje inactivo a ningún otro bus, es decir que no deje otros buses marcados en estado de ocupado sin utilizarlo.



NIVEL 1: Selección de 1 bus para cada módulo.

NIVEL 2: Peticiones a los módulos compitiendo por buses.

Figura 2.10 Estructura del dispositivo de Asignación de Bus.

El algoritmo ejecutado por este circuito puede ser descrito, considerando la función individual de cada uno de los dos niveles:

NIVEL 1: En el nivel 1 existen tantos árbitros "n a 1" como buses existan en el sistema. Cada señal M_k de petición a un módulo de memoria, rivaliza en todos los árbitros del nivel 1, para obtener tantos buses como sea posible entre los buses libres, ya que pueden haber buses asignados.

Las señales intermedias "g_{jk}", salidas de los árbitos del nivel 1, indican los buses que se le han concedido a cada peticionario, que pueden ser más de 1 y que han quedado marcados en estado de ocupado.

NIVEL 2: En este nivel existen tantos árbitros "n a 1" como módulos de memoria existan en el sistema. En este nivel se realiza para cada peticionario, la selección de un bus, entre todos los que haya capturado en el nivel 1.

Las salidas activas de los árbitros del nivel 2 (B_{jk}), indican para cada petiión a un módulo de memoria (k) cual es el bus (j) asignado para su acceso.

Los buses obtenidos en el nivel 1, pero no asignados en el nivel 2, si existen, pasan a ser buses libres en el nivel 1. Para realizar esto es necesaria

la realimentación antes mencionada.

Por lo tanto un bus queda finalmente ocupado, sólo si es el único que ha quedado asignado a un módulo de memoria con una petición activa, en caso contrario quedará libre.

Todas estas operaciones realizadas en los niveles 1 y 2, se realizan en lo que llamamos una "vuelta" en el circuito. Si alguna petición fué parada en el nivel 1, porque no obtuvo ningún bus, cuando los buses sean liberados a través de la realimentación del nivel 2 al nivel 1, comenzará una nueva vuelta para estos peticionarios. El número de vueltas antes de que la solucción final sea obtenida, dependerá del modelo de interconexión utilizado en las entradas al primer nivel y entre ambos niveles.

Es muy interesante notar, que a pesar de su apariencia serie, este circuito correctamente cableado, es capaz de realizar un funcionamiento paralelo.

Para realizar el estudio de este circuito, hemos utilizado árbitros con prioridad en ambos niveles.

Para simplificar este estudio, el dispositivo de

asignación de buses lo hemos descrito por dos matrices (Luq-85), mostradas en la figura 2.11:

-Una matriz del primer nivel (FLM), describiendo las interconexiones entre las señales de petición a los módulos de memoria (M_k) y las entradas a los árbitros del nivel 1, esta formulación es mostrada en la figura 2.11a.

La matriz tendrá tantas filas como buses existan en el sistema y tantas columnas como módulos de memoria.

Cada fila repesenta un árbitro del nivel 1, con un número de entradas igual al número de columnas de la matriz.

Cada elemento de la matriz indica la prioridad de entrada de la petición al módulo (indicado en la columna), con la que compite en el correspondiente bus (indicado por la fila). Por lo tanto, todos los elementos de una fila deben tener diferente prioridad, lo contrario sisgnificaría que existen peticiones conectadas a la misma entrada en un árbitro.

-Una matriz del segundo nivel (SLM), que describe las interconexiones entre las salidas de los árbitros del nivel 1 (g_{jk}) y las entradas de los

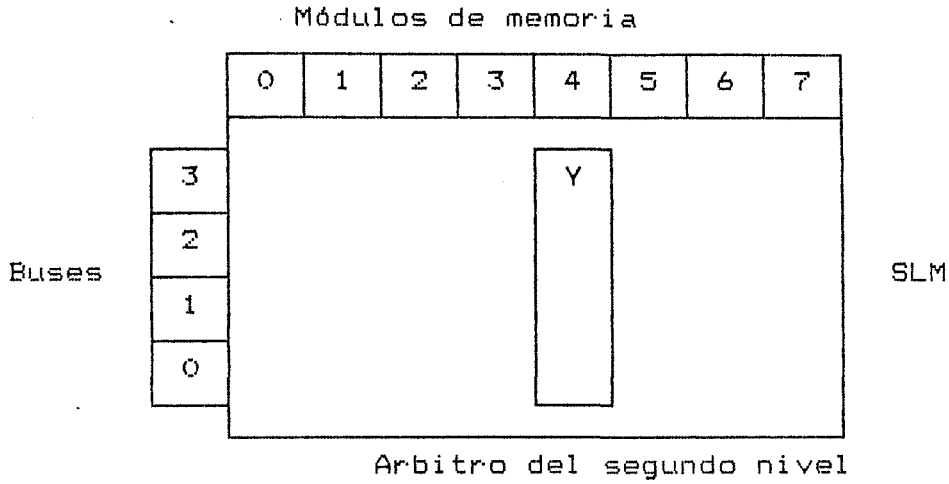


Figura 2.11b Matriz del segundo nivel (SLM)

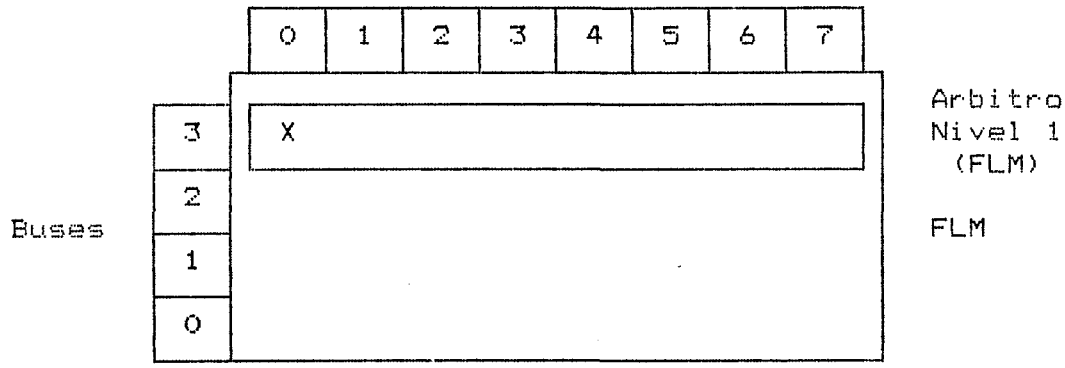


Figura 2.11a Matriz del primer nivel (FLM)

Y = Prioridad en el árbitro del módulo 4 de la concesión del bus 3 al módulo 4

X = Prioridad en el árbitro del bus 3 de la petición al módulo de memoria 0 R_o.

Figura 2.11 Representación del dispositivo de asignación de buses

árbitros del nivel 2, esta formulación se muestra en la figura 2.11b.

Al igual que en las matrices del nivel 1, existen tantas filas como buses y tantas columnas como módulos de memoria existen en el sistema.

Cada columna representa un árbitro del nivel 2, con un número de entradas igual al número de buses (filas en la matriz).

Cada elemento de una columna indica la prioridad de entrada de los diferentes buses asignados a una petición al mismo módulo de memoria. Por lo tanto en esta matriz todos los elementos de una columna deben tener diferente prioridad.

Para ilustrar esta formulación se muestra un ejemplo en la figura 2.12, donde se representa un dispositivo de asignación del bus para 4 módulos de memoria y 2 buses.

En el ejemplo de la figura 2.12b, es posible verificar, siguiendo los modelos de interconexión representados por las matrices, que para cualquier par de peticiones a módulos de memoria, los buses son asignados en una sola vuelta. El circuito con este modelo de interconexión realiza la asignación de buses en paralelo.

En la figura 2.12c, la asignación de bus se realiza en dos vueltas. Con este modelo de interconexión el circuito está asignando en forma serie.

Es importante resaltar, que todos los modelos de interconexión, respetando las reglas de conexión antes descritas, proporcionan una correcta asignación de bus, la diferencia entre ellos, como se muestra claramente en las figuras 2.12b y 2.12c, es el número de vueltas necesarias para obtener la contestación y por tanto el tiempo de respuesta hasta evaluar la solucción correcta. El número de vueltas (V), será 1 como mínimo y como máximo igual al mínimo de número de buses o peticiones activas:

$$1 <= V <= \min(\text{buses}, \text{peticiones activas})$$

es decir, en cada vuelta se asignará como mínimo una asignación.

El estudio de este circuito, puede ser reducido, al estudio de los modelos de interconexión. Desgraciadamente se hace imposible un análisis exhaustivo de todos los modelos posibles, debido a su elevado número, como mostraremos a continuación.

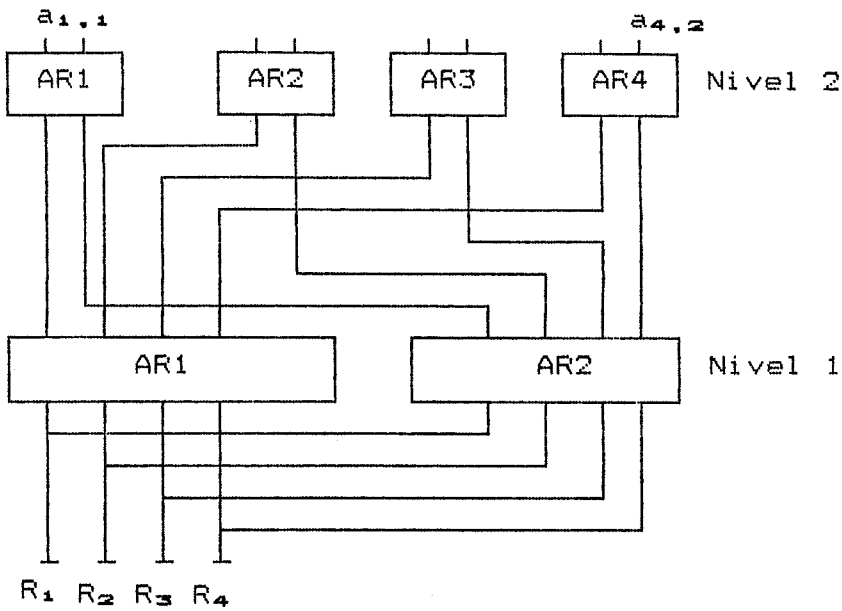


Fig. 2.12a Modelo de Interconexión para una asignacion paralelo

SLM

	0	1	2	3
0	0	0	0	0
1	1	1	1	1

FLM

	0	1	2	3
0	0	1	2	3
1	3	2	1	0

2.12b Asignación Paralelo

SLM

	0	1	2	3
0	0	0	0	0
1	1	1	1	1

FLM

	0	1	2	3
0	0	1	2	3
1	0	1	2	3

2.12c Asignación Serie

Figura 2.12

Si consideramos un sistema multiprocesador com "M" módulos de memoria y "B" buses, el número de posibles modelos de interconexión para el circuito (Luq-85a) es:

$$\text{Interconexiones } (M,B) = (M!)^B . (B!)^M$$

$(M!)^B$: Conexiones en las entradas al nivel 1.

$(B!)^M$: Conexiones en las entradas al nivel 2.

Podemos evaluar esta expresión para algunos valores de M y B.

$$\text{Interconexiones } (4,2) = 9216$$

$$\text{Interconexiones } (8,2) = 10^{11}$$

$$\text{Interconexiones } (8,4) = 10^{29}$$

El gran número de posibilidades nos ha obligado a reducir nuestro estudio a un subconjunto de modelos de interconexxión.

En el nivel 1 buscamos los modelos de interconexión, en los que una petición a un módulo gane el mínimo número de buses posible, ya que si en el primer nivel, a cada petición sólo se le asigna un bus,

el circuito realizará la asignación de los buses en una vuelta.

En matrices cuadradas, es decir aquellas en las que el número de buses es igual al número de módulos (M=B), es muy fácil lograr este objetivo, pero a medida que las matrices se van haciendo más rectangulares (M>B) es más dificil lograrlo.

Por lo tanto, en el segundo nivel buscamos que queden libres aquellos buses en los cuales, las peticiones sin bus asignado, tengan la máxima prioridad en buses diferentes en el nivel 1.

Hemos estudiado el tiempo de respuesta (retardo desde la petición al módulo de memoria hasta la contestación asignándole un bus) del circuito, variando el número de buses y/o el número de módulos de memoria.

Para obtener el tiempo de respuesta (RT), frente a cualquier posible combinación de las peticiones a la entrada, hemos realizado un programa que simula el comportamiento del circuito de asignación de bus . Las especificaciones de este programa se encuentran en el apéndice A.

Hemos obtenido una expresión que permite conocer, o al menos limitar, el tiempo de respuesta (RT) del dispositivo de asignación de bus. Este tiempo (RT) puede lograrse utilizando la matriz de interconexión apropiada. Esta expresión ha sido deducida considerando el proceso de asignación del bus como un proceso binario:

$$RT = m * \lceil \log_2 B \rceil$$

Siendo:

B : el número de buses.

$\lceil \log_2 B \rceil$: el número máximo de vueltas.

m : el tiempo tiempo de retardo producido en los árbitros del nivel 1 y el nivel 2. Este tiempo dependerá de los árbitros utilizados en ambos niveles.

Vemos que el tiempo de respuesta del circuito de asignación del bus, depende basicamente del número de buses, y es independiente del número de procesadores del sistema.

Debemos notar que el valor límite del tiempo de respuesta obtenido en esta fórmula, está aplicado a sistemas muy rectangulares (8*4), (12*4),... sin embargo, en sistemas cuadrados, o aún muy poco rectangulares (4*4), (5*4), el tiempo de respuesta es

sólo "m".

En la tabla de la figura 2.13 se puede observar el número máximo de vueltas que realiza el circuito, para obtener la asignación de los buses, variando el número de módulos y de buses. Estos resultados han sido obtenidos utilizando el programa de simulación antes mencionado.

Vamos a estudiar sobre un ejemplo el comportamiento del circuito, para un sistema con 16 módulos de memoria y 8 buses. Para ello utilizaremos las matrices de interconexión mostradas en la figura 2.14. Iremos viendo paso a paso como se va realizando la asignación de los buses a los diferentes módulos de memoria con petición activa.

Supongamos que existen peticiones activas para los módulos M_0, M_1, M_2, M_3, M_4, M_5, M_6, y M_7.

En el árbitro del nivel 1 para el Bus 0 (B_0), el módulo más prioritario es M_0, por lo tanto tendremos:

M_0----B_0

Para el Bus B_1 el módulo más prioritario es M_2 y tiene petición activa, por lo tanto:

M_2----B_1

En el B_2 el módulo más prioritario con petición activa es M_4:

M = 8	B	1	2	3	4	7	8
	V	1	1	2	2	1	1

M = 12	B	1	2	4	6	8
	V	1	1	2	3	3

M = 12	B	1	2	4	8	16
	V	1	1	2	3	1

M = 32	B	1	2	4	8	32
	V	1	1	2	3	1

M = nº de módulos

B = nº de buses

V = nº de vueltas

Figura 2.13

M_6----B_2

Analizando la matriz, para el resto de los módulos con petición activa, vemos que:

M_5----B_3

M_2----B_4

M_1----B_5

M_4----B_6

M_7----B_7

. Cada petición a un módulo sólo gana un bus en el nivel 1, por lo tanto los árbitros del nivel 2 sólo tendrán 1 entrada activa, siendo ésta la asignación definitiva, obtenida en 1 vuelta.

Supongamos ahora que existen peticiones activas a los módulos M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8. Vamos a ver como resuelven los árbitros del nivel 1:

M_1----B_0

M_3----B_1

M_6----B_2

M_5----B_3

M_8----B_4

M_8----B_5

M_8----B_6

M_8----B_7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	3	2	1	0	7	6	5	4	11	10	9	8	15	14	13	12
2	6	7	4	5	2	3	0	1	14	15	12	13	10	11	8	9
3	5	4	7	6	1	0	3	2	13	12	15	14	9	8	11	10
4	10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5
5	9	8	11	10	13	12	15	14	1	0	3	2	5	4	7	6
6	12	13	14	15	8	9	10	11	4	5	6	7	0	1	2	3
7	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Fig 2.14a Matriz Nivel 1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7

Fig. 2.14b Matriz Nivel 2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
1	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
5	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 2.14c Matriz Nivel 2

Vemos que la petición al módulo 8 obtiene 4 buses en el nivel 1, por lo tanto, en el nivel 2 el árbitro correspondiente al módulo 8, tendrá 4 entradas activas, asignará un bus para el módulo 8, liberando los otros 3 para que se asignen en otra vuelta a los módulos con petición activa y sin bus asignado.

Si utilizamos la matriz del nivel 2 de la figura 2.14b, vemos que para el árbitro del módulo 8 el bus más prioritario, entre los que ha obtenido, es el B_4, quedando libres los buses B_5, B_6, B_7.

Ahora comenzará uns segunda vuelta, en la que competiran las peticiones a los módulos sin bus asignado M_2, M_4, M_7, por los buses libres, obteniendo:

M_2----B_5

M_4----B_6

M_7----B_7

Obteniendose en esta segunda vuelta una asignación definitiva.

Vemos que si utilizamos la matriz del segundo nivel de la figura 2.14c, harían falta 3 vueltas. En la primera vuelta, al módulo 8 se le habría asignado el

bus B_7, quedando libres los buses B_4, B_5, B_6. En la segunda vuelta tendríamos:

M_2----B_4

M_2----B_5

M_4----B_6

En el segundo nivel el M_2 obtendría el bus B_5, liberando el B_4, que quedaría asignado en la tercera vuelta al M_7.

Las matrices de interconexión para estructuras (4*4), (8*4) y (16*4), están mostradas en la figura 2.15.

2.6 Sistema de arbitraje

Para realizar el sistema de gestión de buses descrito, proponemos dos tipos de sistemas de arbitraje,un sistema de arbitraje basado en un árbitro centralizado y otro basado en un árbitro distribuído. Ambos ejecutan la política de arbitraje descrita.

2.6.1 Sistema de arbitraje centralizado

El arbitraje centralizado necesita líneas privadas de conexión en el backplane, entre cada tarjeta del

Módulos de memoria

Buses	0	1	2	3
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3

SLM

Módulos de memoria

Buses	0	1	2	3
0	0	1	2	3
1	3	2	1	0
2	1	0	3	2
3	2	3	0	1

FLM

RT = 1 m

Fig. 2.15a sistema 4*4

Módulos de memoria

Buses	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3

SLM

Módulos de memoria

Buses	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	3	2	1	0	7	6	5	4
2	5	4	7	6	1	0	3	2
3	6	7	4	5	2	3	0	1

FLM

RT = 2 m

Figura 2.15b Sistema 8*4.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SLM

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	5	4	7	6	1	0	3	2	13	12	15	14	9	8	11	10
2	10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5
3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

FLM

RT = 2 m

Fig 2.15c Sistema 16*4

procesador o de memoria y la tarjeta del árbitro. Cada tarjeta del procesador necesita enviarle al árbitro,lassiguientes señales propias: la señal de petición "R_i", el código del módulo de memoria al que desea acceder "MP_i", y el árbitro le contestará con la señal de concesión "A_i".

Tambien circulan por el bus, todas las señales de apertura y cierre de los circuitos que controlan el flujo de información. Estas señales son líneas especiales del backplane, diferentes para cada tarjeta.

En el capítulo 2 se describe un prototipo realizado para 4 procesadores, 4 módulos de memoria y 4 buses, utilizando un árbitro centralizado.

2.6.2 Sistema de arbitraje distribuído

El arbitraje distribuído, utiliza sólo líneas tipo bus comunes a todas las placas, por lo tanto no requiere líneas especiales de backplane. Estas líneas son señales "O cableadas", salidas de puerta de colector abierto.

La idea básica fue desarrollada por Taub (Tau-84).

El método de Taub sólo usa líneas de buses, de forma que toda la información del arbitraje está presente en todo el bus; no tiene dependencia de la posición y no requiere líneas especiales de backplane. Utiliza señales "O cableadas" para señales de tiempo y control, y para realizar el arbitraje actual del bus. La idea básica es que cada dispositivo que quiere el bus, intenta poner su propia prioridad en el bus de arbitraje, quitando sus bits menos significativos si ve un número mayor presente; después de un retardo, sólo queda fijo en las líneas de arbitraje,el número de mayor prioridad entre los competidores. El dispositivo que ve su propia prioridad controla el bus. Cuando el dispositivo quita su número, gana el dispositivo de siguiente prioridad y así hasta el último.

Cada procesador debe tener un número de arbitraje diferente,que indique su prioridad . Las señales "O cableadas" de control, mantiene a los diferentes dispositivos sincronzados, durante todo el tiempo que dura el procedimiento de adquisición del bus

Todas las líneas son mantenidas por puertas de colector abierto, el 1 lógico está representado por el nivel menos positivo. Por lo tanto, la señal en una línea representa la función "O" de las señales

aplicadas a ella, por los diferentes dispositivos.

Para nuestro sistema de gestión de buses múltiples,el número de líneas necesarias para realizar el arbitraje distribuído, son las siguientes:

-M líneas de petición de módulo, una por cada módulo de memoria que exista en el sistema.

-M líneas para realizar el arbitraje.

-Señales de control que sincronizan los momentos de arbitraje.

Cuando un procesador desea acceder a un módulo de memoria, enviará por la línea correspondiente una señal de petición de módulo, y analizará las señales de control que indicarán cuando comenzar el arbitraje, en este momento irá colocando su número de arbitaje, en la línea de arbitraje correspondiente a dicho módulo. Primero colocará el bit más significativo de su número de arbitraje, si en la línea ve un valor diferente al suyo, cesa de sacar bits a la línea de arbitraje, ya que esto significa que existe un procesador más prioritario que él pidiendo acceso al mismo módulo de memoria.

Cada procesador después de analizar el bit

correspondiente en la línea de arbitraje, conmuta sus señales de control, sólo cuando el último ha conmutado, conmutará la línea "O cableada" de la señal de control. En este momento los procesadores a los que les va coincidiendo el número de arbitraje sacan el siguiente bit de su número de arbitraje. Esta operación se irá repitiendo para cada bit del número de arbitraje, sólo el procesador de mayor prioridad pidiendo al módulo de memoria, habrá reconocido todos los bits, el dispositivo que ha visto concordar todos los bits de su prioridad es el procesador seleccionado.

Para garantizar la imparcialidad entre los procesadores y prevenir que el dispositivo de mayor prioridad esté ganando todo el tiempo el acceso a un módulo, una vez el dispositivo ha terminado el acceso a un módulo de memoria, inicializa las señales de control y su línea de petición quedará aislada del bus de peticiones, hasta que todos los procesadores que hayan pedido simultaneamente con él y que tenían menor prioridad, hayan sido atendidos. La señal de petición de módulo m_i, tiene controlado su paso al bus por un árbitro binario, siendo la otra entrada la petición de módulo M_i, por tanto es necesario que esta señal se desactive, indicando que todos los procesadores que habían realizado petición a dicho módulo han sido

atendidos, para que puedan salir a la línea de petición nuevas peticiones al módulo.

El circuito de asignación del bus estará replicado en todas las tarjetas del sistema, las entradas son las señales M_i que indican que existe alguna petición al módulo i, este circuito irá funcionando en paralelo con el ciruito antes visto de selección del procesador.

2.6.3. Arbitraje de los esquemas reducidos

Vamos a ver como aplicamos el circuito de arbitraje antes descrito a los esquemas reducidos.

El circuito de gestión de buses, en nuestro caso no se ve simplificado por el hecho de reducir el número de conexiones, ya que es el mismo que el que controla el esquema completo de buses múltiples, variando unicamente las interconexiones, como ya veremos.

Para que todos los procesadores tengan la misma probabilidad de acceso a memoria, dividimos el proceso de arbitraje en dos partes:

a) Un proceso de selección imparcial, para

seleccionar, de los R módulos de memoria que tienen al menos una petición pendiente , min(B,R) peticiones a módulos de memoria. Este proceso de selección no depende del esquema de conexión.

Para realizar esta función utilizamos un árbitro "M usuarios B servidores", que controla el paso como máximo de tantas peticiones como buses existan en el sistema. Este árbitro ejecuta un algoritmo de asignación imparcial.

b) Un proceso de asignación, que asigne los buses a los módulos de memoria seleccionados. El proceso de asignación debe comenzar una asignación, disponiendo de todos los buses libres, ya que en caso contrario puede ocurrir que no pueda asignar bus a módulos de memoria seleccionados por estar ocupados los buses a los que estan conectados existiendo otros buses libres.

El dispositivo de asignación de bus propuesto, puede controlar el proceso de asignación de bus en esquemas reducidos, para ello basta con buscar para cada uno de los esquemas de conexión, las correspondientes interconexiones o sea, las prioridades de las conexiones en las entradas a ambos niveles.

Si la reducción de las conexiones se realiza minimizando las conexiones memoria bus, las peticiones a los módulos de memoria M_k, correspondientes a los buses de los que estan desconectados, no deben estar conectadas a las entradas del los árbitros del nivel 1.

Si la reducción en el número de conecciones se realiza entre buses y procesadores, existirán unas nuevas señales de petición de módulo M_{kj},diferentes para cada árbitro del nivel 1, puesto que dependerán de los buses a los que estén conectados los procesadores que pidan a un módulo .

La señal M_{kj}, entrada al árbitro del bus j en el nivel 1, será la "O lógica" de las peticiones r_{ik} de los procesadores conectados al bus j

Hemos buscado las matrices de interconexión para diferentes formas reducidas. La reducción se ha realizado en el crossbar memoria-bus. Hemos utilizado el programa que simula el comportamiento del circuito de asignación de buses, para comprobar el funcionamiento del árbitro frente a cualquier posible combinación de entrada de peticiones, comprobando que si existen peticiones a B módulos de memoria, se realizan B asignaciones de buses a los B módulos de

memoria y que no suceda que un bus quede libre existiendo una petición pendiente. Utilizando el programa, tambien hemos obtenido el número máximo de vueltas necesario para realizar las asignaciones, estando el tiempo de respuesta (RT), en los casos estudiados, acotado por la misma expresión que la obtenida en los sistemas completos.

Las matrices de interconexión obtenidas para cuatro formas reducidas son mostradas en la figura 2.16.

Vemos que los módulos de memoria con menor número de conexiones, tienen las máximas prioridades en algún bus en el nivel 1, para que los módulos con menos conexiones, obtengan al menos un bus en la primera vuelta.

2.6.4 Arbitraje de los esquemas multiplexados

Como hemos visto, el esquema multiplexado que estudiamos está basado en dividir un ciclo de acceso a la memoria en cuatro partes, tiempo de arbitraje, enviar la dirección (y el dato en un ciclo de escritura), esperar el tiempo de acceso de la memoria y

Módulos de memoria

Buses	0	1	2	3	4	5	6	7
3	0	0	0	0	3			
2		1	1	1	2	2		
1			2	2	1	1	1	
0				3	0	0	0	0

SLM

Módulos de memoria

Buses	0	1	2	3	4	5	6	7
3	0	4	3	2	1			
2		0	2	3	4	1		
1			1	4	3	2	0	
0				1	2	3	4	0

FLM

Figura 2.16a Rombico

Módulos de memoria

Buses	0	1	2	3	4	5	6	7
3				0	0	0	0	0
2			0		1	1	1	1
1		0			2	2	2	2
0	0				3	3	3	3

SLM

Módulos de memoria

Buses	0	1	2	3	4	5	6	7
3				0	1	2	3	4
2			0		4	3	2	1
1		0			2	1	4	3
0	0				3	4	1	2

FLM

Figura 2.16b Escalera

Módulos de memoria

Buses	0	1	2	3	4	5	6	7
3	2		2	2	0		2	0
2		0	1	1		0	1	1
1	1	1	0		1	1	0	
0	0	2		0	2	2		2

SLM

Módulos de memoria

Buses	0	1	2	3	4	5	6	7
3	5		3	4	1		0	2
2		5	4	3		1	2	0
1	0	2	1		3	4	5	
0	2	0		1	4	3		5

FLM

Figura 2.16c Ciclico

Módulos de memoria

Buses	0	1	2	3	4	5	6	7
3			2	2	0		1	0
2			1	1		0	0	1
1	1	0	0		1	1		
0	0	1		0	2	2		

SLM

Módulos de memoria

Buses	0	1	2	3	4	5	6	7
3			3	4	1		0	2
2			4	3		1	2	0
1	0	2	1		3	4		
0	2	0		1	4	3		

FLM

Figura 2.16d Balanceado

Figura 2.16 Matrices de interconexión para sistemas reducidos 8x4

devolver el dato. Si la memoria dispone de unos latches para almacenar la dirección (y los datos en los ciclos de escritura), nos permite utilizar el bus durante el tiempo de acceso de la memoria, para transferir información de otros procesadores.

Cuando comienza un ciclo, los procesadores que lo deseen activan las peticiones a los módulos de memoria, el sistema de arbitraje selecciona los procesadores que pueden realizar el acceso y les asigna bus. Durante el tiempo de acceso a la memoria se les asigna los buses a otros módulos de memoria con petición activa. El árbitro m a n selecciona de las "m" peticiones a los módulos de memoria con bus asignado cuales van a utilizar los "n" buses físicos que existen en el sistema durante el primer intervalo de tiempo. La información circulando por los buses se almacenará en unos latches en los módulos de memoria correspondientes y los buses serán nuevamente asignados a otros módulos de memoria con petición activa.

El circuito de gestión de buses activará la señal de concesión a tantos procesadores con peticiones activas como buses físicos existan en el sistema. Antes de que termine el ciclo, los buses serán nuevamente asignados a los módulos de memoria que tienen datos

válidos para enviar a los procesadores.

Para no añadir un elevado tiempo extra de arbitraje para asignar el bus a otros procesadores, el sistema de gestión de buses para los esquemas multiplexados, seleccionará todas las peticiones a módulos de memoria que puedan iniciar una transferencia durante un ciclo, dependiendo del factor de multiplexación, en lugar de como máximo tantas peticiones como buses físicos que existan en el sistema.

Una vez realizada la selección (en el circuito de asignación de buses) de todas las peticiones que van a utilizar buses, necesitamos un árbitro "m a b" que seleccione, durante cuantos de tiempo, tantas asignaciones como buses (b) existan fisicamente en el sistema. Hay que tener en cuenta que antes de acabar el ciclo,el bus debe ser de nuevo asignado al primer módulo de memoria accedido, para devolver los datos al procesador con la señal de concesión activa

El sistema de arbitraje, dispondrá de un reloj que le permitirá cuantificar los tiempos de concesión de los buses. Este reloj estará sincronizado con el reloj del sistema.

Respecto al sistema de gestión de buses mostrado en la figura 2.9, se ha añadido un nuevo módulo (árbitro "m a b"), conectado a las salidas $B_{i,j}$ del sistema de asignación de buses y un circuito para controlar las entradas al árbitro "m a b" durante los cuantos de tiempo establecidos.

CAPITULO 3

REALIZACION DE UN SISTEMA MULTIPROCESDOR

En este capítulo se describe el sistema multiprocesador con buses múltiples desarrollado, su arquitectúra, características, organización y las diferentes posibilidades que ofrece para la ejecución de aplicaciones. Se detallan tambien los árbitros utilizados y las características especificas del prototipo realizado.

3.1 Arquitectura del sistema

Vamos a dar una descripción de la arquitectura general del sistema multiprocesador. Tres módulos son los componentes fundamentales del multiprocesador:

1- Procesadores (P). Cada procesador incluye su propia memoria local (Lm), capacidades de Entrada/Salida e interconexión de buses (BI).

2- Módulos de Memoria Común (M). Que incluyen además de la memoria global, un espacio de direciones dedicados a periféricos comunes para entrada/salida y la interconexión a los buses.

3- Buses y la lógica de arbitraje que los gestiona.

Un diagrama de bloques mostrando la estructura de este sistema multimicroprocesador, con memoria compartida y buses multiples, está en la figura 3.1, donde los "p" procesadores y "m" módulos de memoria, están conectados por "b" buses, tal que b<= min (p,m)

Los parámetros que hemos considerado para elegir como topología de interconexión el esquema de buses multiples, son : Fiabilidad, tolerancia a fallos, modularidad, anchura de banda, número de procesadores, expandabilidad y coste moderado.

Una característica de nuestro sistema de buses, es que el software no necesita conocer como estan configurados y cuantos estan trabajando,siendo por lo tanto el número de buses transparente al usuario.

Todos los módulos de memoria común cubren las mismas direcciones del espacio de direcciones de los procesadores. El mecanismo de mapping (selección del módulo o banco de memoria común deseado por parte de los procesadores), definido e implementado en cada procesador (microprocesdor en el prototipo realizado),

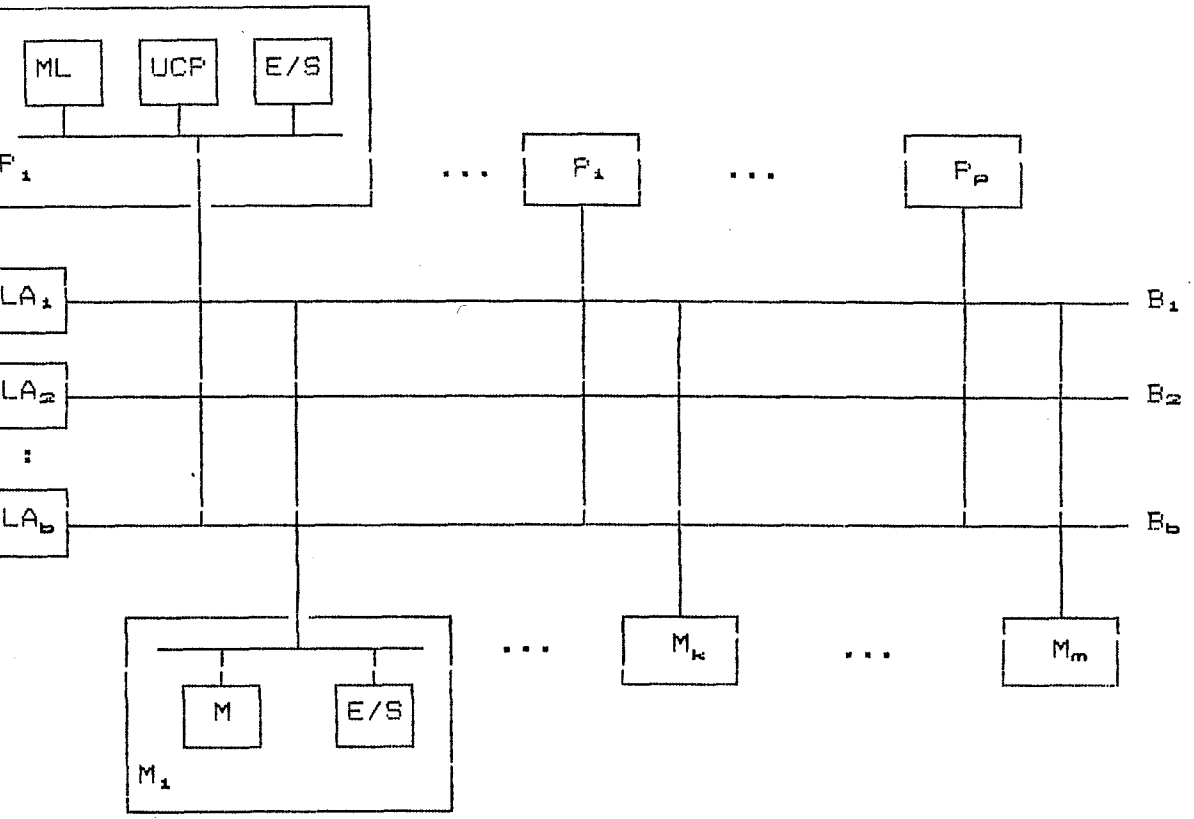


Figura 3.1 Estructura multiprocesador.

tiene dos funciones:

-Seleccionar uno de entre los diferentes módulos de la memoria global, para establecer la correspondencia entre la zona del espacio de direcciones reservada en el microprocesador (ventana de direcciones) y las direcciones correspondientes a un módulo de memoria común.

-Expandir el limitado espacio de direcciones de memoria que proporcionan los microprocesadores de 8 bits. Esta función se realiza a través de un registro (el .registro de memory-mapping), donde las direcciones lógicas, suministradas por el microprocesador, actúan como un desplazamiento dentro del módulo de memoria global capacitado por el hardware encargado de seleccionar un módulo.

La característica de memory-mapped de entrada/salida que poseen los microprocesadores del sistema permite usar ciertas direcciones de la memoria común para soportar los periféricos comunes.

El subsistema de entrada/salida del sistema multiprocesador, está organizado en dos niveles:

1- Entrada/salida local:Los dispositivos están directamente asociados a los procesadores, con objeto de proporcionar el tiempo mínimo de respuesta en

la atención a las tareas dirigidas por interrupción.

2- Entrada/Salida global: Las direcciones físicas de estos periféricos estan en la región de direcciones globales. Los periféricos globales permiten, al estar compartidos, optimizar la utilización de estos recursos. Dos grupos de periféricos pueden ser considerados:

Periféricos pasivos: que necesitan un procesador del sistema para dirigir su comunicación.

Periféricos inteligentes: que incluyen su propio procesador en la comunicación. Estos periféricos son mas recomendados para limitar el uso de buses comunes en funciones de entrada/salida

3.2 Caracteristicas del sistema

El sistema multiprocesador con buses múltiples que proponemos tiene las siguientes características.

-Memorias locales para cada procesador.

-Memorias compartidas, donde se almacenarán los monitores y los procesos menos prioritarios que aquellos que se ubiquen en las memorias locales. Un árbitro que garantiza la exclusión mútua está asociado a cada móodulo de memoria compartida.

-Software básico almacenado en cada uno de

los procesadores, encargado de gestionar las colas de los diferentes estados por los que pasan los procesos y los monitores.

-La gestión de los periféricos podrá ser de dos tipos: Los periféricos comunes, que serán gestionados a través de monitores del sistema y los periféricos locales, asociados a cada procesador como si se tratara de memorias locales.

-Otra característica del sistema es la transparencia, en el sentido de que sin modificar el software de la aplicación, se puede realizar una reconfiguración del sistema para aumentar o disminuir el número de procesadores, módulos de memoria y buses, para mejorar la adaptación del mismo al problema concreto que se trata de resolver en funciones de parámetros tales como las prestaciones y el costo del sistema.

-La asignación de un proceso al procesador puede ser fija en aplicaciones donde la carga del sistema es conocida a priori, o dinamicamente cambiada cuando la carga del sistema no es conocida, la asignación de procesos a los procesadores se hará en función del estado del sistema y del algoritmo a

ejecutar.

3.3 Organización de la memoria

En un sistema multiprocesador la memoria principal es un recurso básico que es normalmente compartido por todos los procesadores. Es importante la organización de la memoria del sistema, para evitar una degradación en el rendimiento, causada por interferencias en la memoria cuando dos o más procesadores intentan acceder simultaneamente al mismo módulo de la memoria. Por ejemplo no sería conveniente tener un único módulo de memoria compartido entre varios procesadores, ya que habría serios problemas de interferencia en la memoria.

La memoria común puede ser dividida en varios módulos independientes y el espacio de direcciones distribuído a través de estos módulos. Este esquema llamado "interleaving" resuelve algunas de las interferencias permitiendo accesos concurrentes a más de un módulo.

Existen dos métodos básicos de distribuir las direcciones ertre los módulos de memoria (Hwa-84). Asumimos que existe un total de $N=2^n$ palabras en la

memoria principal, entonces las direcciones físicas para una palabra en la memoria consta de "n" bits, $a_{n-1}a_{n-2}...a_1a_0$. Los dos métodos para distribuir las direcciones son:

- **Interleaving de orden alto**, distribuye las direcciones en $M=2^m$ módulos, tal que cada módulo "i", para $0<=i<=M-1$, contiene direcciones consecutivas $i*2^{n-m}$ a $(i+1)*2^{n-m}-1$, inclusive. Los "m" bits de orden alto son utilizados para seleccionar el módulo, mientras que los restantes n-m bits seleccionan la dirección dentro del módulo, como muestra la figura 3.2a

- **Interleaving de orden bajo**, distribuye las direcciones de modo que direcciones consecutivas estan situadas en módulos consecutivos. Los "m" bits de orden bajo de la dirección seleccionan el módulo, mientras que los restantes n-m bits seleccionan la dirección dentro del módulo, como muestra la figura 3.2b. Aquí. una dirección A esta situada en el módulo identificado por "A mod.M".

El interleaving de bajo orden, presenta ventajas en sistemas de multiprocesamiento cuando el espacio de direcciones común de los procesos activos esta compartido intensamente (Hoa-77). Sin embargo la colocación de direcciones contiguas de la memoria

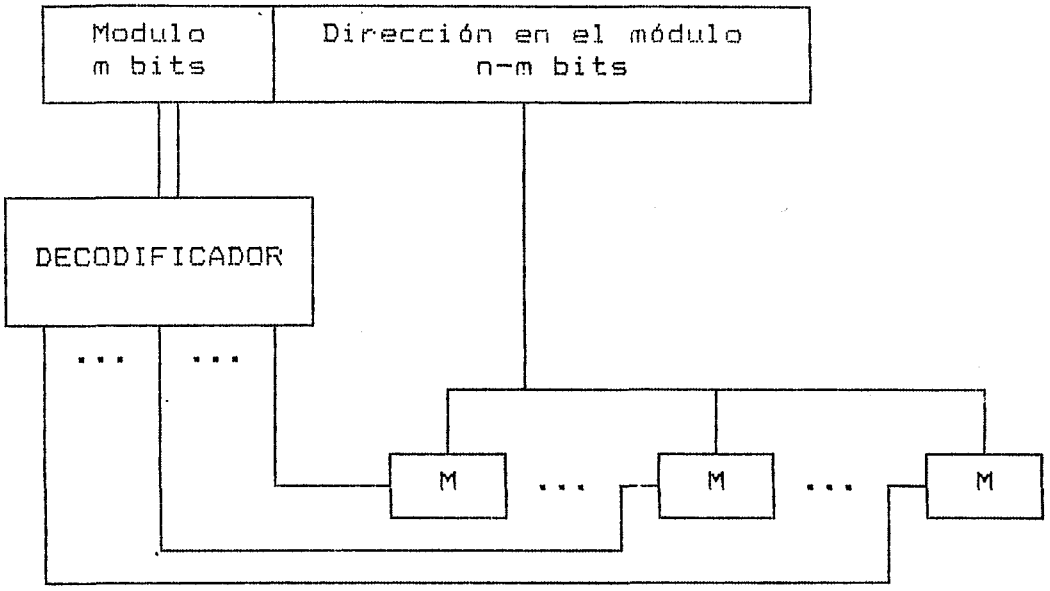


Fig 3.2a Interleaving de alto orden con palabras consecutivas e un módulo.

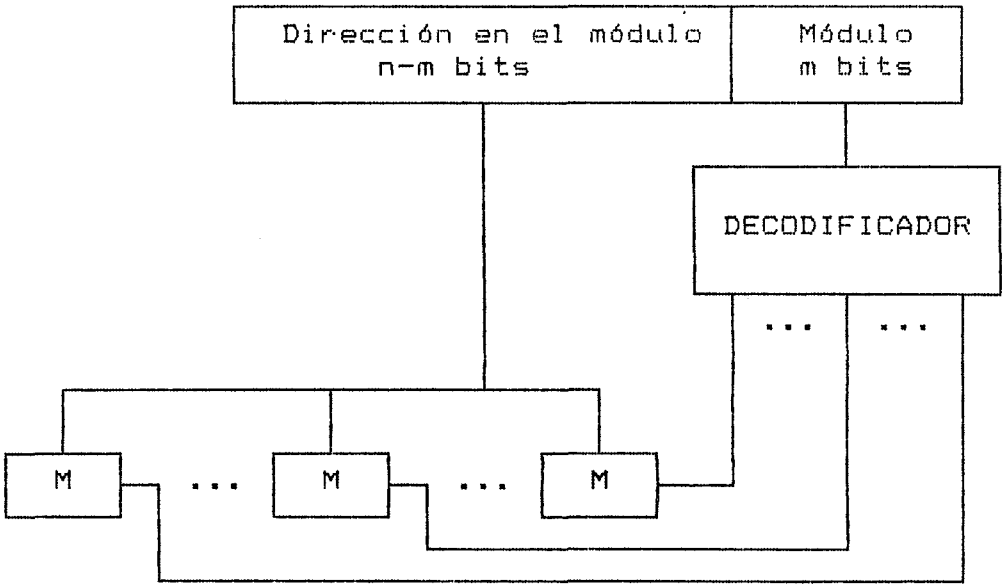


Fig. 3.2b Interleaving de bajo orden con palabras consecutivas e módulos consecutivos.

dentro de un módulo, puede causar conflictos en la memoria debido a la secuencialidad de instrucciones en programas y la secuencialidad de datos en vectores, esto hace que esten en el mismo módulo intrucciones y datos consecutivos. Este conflicto puede ser evitado si los módulos están separados para procesos disjuntos o no interactuando, concentrando páginas de un mismo proceso en un módulo de memoria, de este modo se logra que diferentes procesadores no accedan al mismo módulo de memoria.

El interleaving de orden alto permite una fácil expansión de memoria añadiendo uno o más módulos, tantos como se necesiten, hasta un máximo de M-1.

Otra ventaja del interleaving de orden alto es que proporciona mejor fiabilidad, puesto que cuando falla un módulo afecta sólo a un área localizada del espacio de direcciones y por lo tanto proporciona una fácil degradación en rendimiento. El módulo que ha fallado puede ser logicamente aislado del sistema y el manejador de memoria puede ser informado del espacio de direcciones del módulo que ha fallado. Un fallo de cualquier módulo en el esquema de interleaving de orden bajo podría ser un fallo catastrófico del sistema.

En el sistema multiprocesador que proponemos la memoria está dividida en:

- Memorias locales para cada procesador. En estas memorias estaran almacenados programas y datos locales. Estas memorias no son accesibles desde otros procesadores.

- Memoria común. Todos los procesadores pueden acceder a la memoria comun. La memoria compartida está dividida en módulos de memoria independientes utilizando un interleaving de alto orden.

Las peticiones de un procesador a su propia memoria local, son llamadas peticiones internas y son llevadas a cabo a través de un bus interno del procesador. Las peticiones a los módulos de memoria común son realizadas a través de la red de interconexión de buses multiples.

Según esto, podemos clasificar los estados del procesador, referidos a su acceso a memoria, como sigue:

-Activo: Un procesador está ocupado leyendo o escribiendo en su memoria local.

-Accediendo a la memoria común: Los procesadores pueden estar leyendo o escribiendo en la memoria común a través de la red de interconexión de

buses múltiples.

-Bloqueado: El procesador no ha podido establecer conexión con el módulo de memoria pedido y está esperando para acceder en el próximo ciclo procesador/memoria.

Para aumentar la capacidad de la memoria común, se utiliza un esquema simplificado de ampliación mediante un "mapping" en la memoria común. Este sistema implica la existencia de "$L=2^{l}$" bloques de memoria común de $N=2^{n}$ palabras, ocupando todos ellos las mismas posiciones del espacio de memoria. Estos bloques de memoria pueden ser utilizados de forma independiente desde cada procesador, almacenando en un registro propio del procesador (registro de mapping), el código de identificación correspondiente a cualquier bloque.

El interleaving puede realizarse a partir del registro de mapping,como se muestra en la figura 3.3a. El registro de mapping se utiliza para seleccionar el módulo de memoria al que se desea acceder y los "n" bits más bajos de las direcciones lógicas suministradas por el procesador actúan como un desplazamiento dentro del módulo de memoria seleccionado por los "1" bits del registro de mapping.

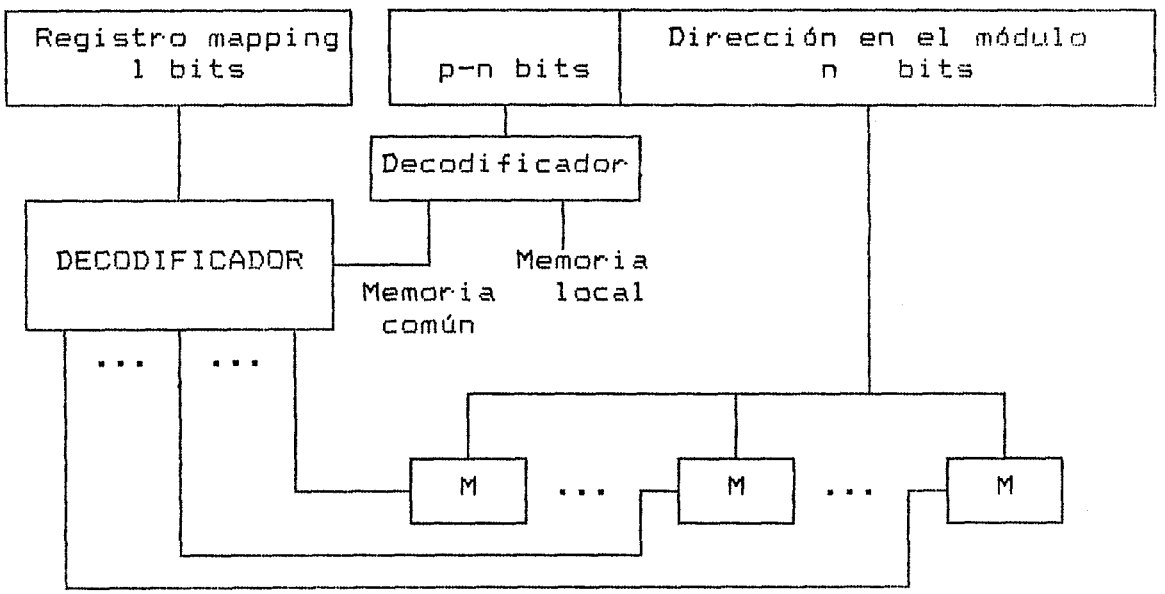


Fig 3.3a Interleaving utilizando un registro de mapping con palabras consecutivas en un módulo.

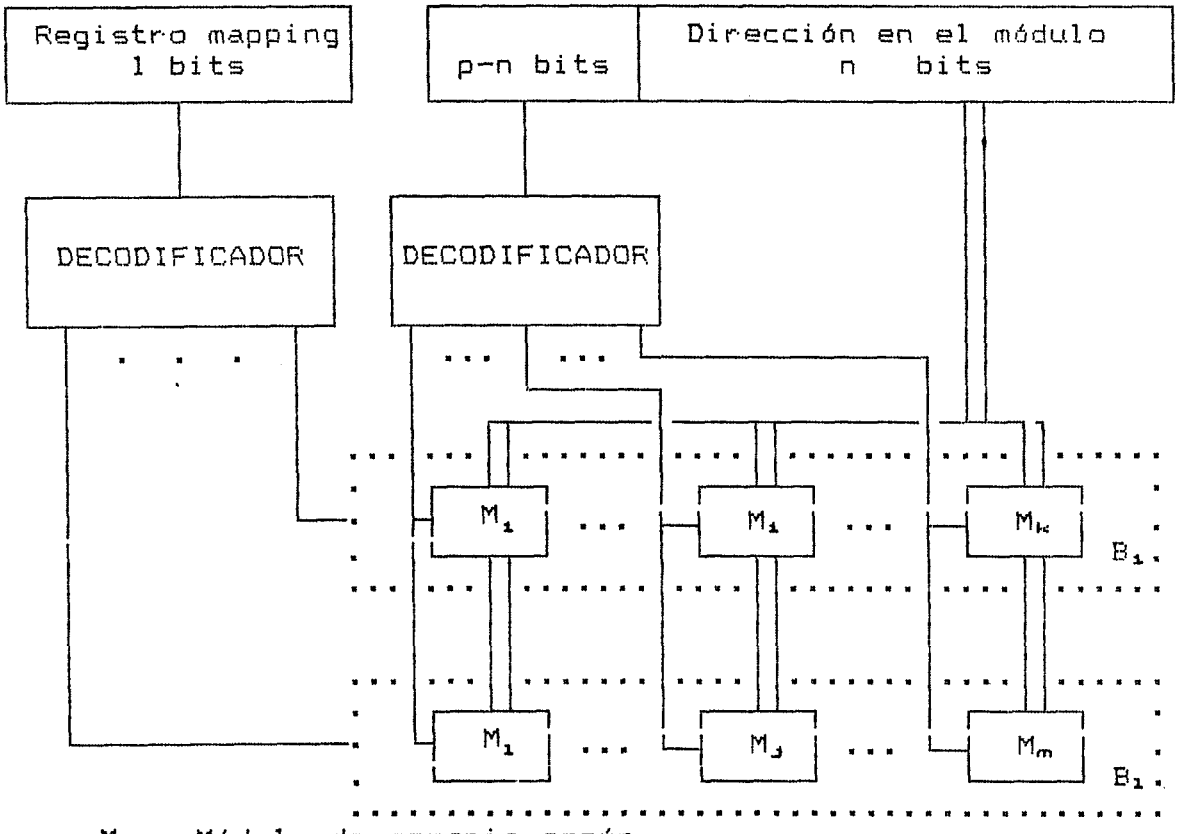
El interleaving tambien puede realizarse utilizando el registro de mapping y los "m" bits más altos de la dirección, de forma que en uno de los "L" bloques de la memoria común seleccionado por el registro de mapping, pueda existir más de un módulo de memoria independiente (figura 3.3b).

3.4 Acceso a recursos comunes

La concepción y realización de sistemas multiprocesadores utilizando procesadores no diseñados originalmente para arquitecturas multiprocesador, implica la creación, evaluación y realización de circuitos hardware específicos, que permiten la comunicación entre procesadores, buses globales y módulos de memoria, exigida por la propia filosofía de un sistema multiprocesador.

El hardware del sistema multipocesador propuesto ha sido concebido en parte para facilitar un esquema de programación concurrentre basado en la utilización de procesos y monitores.

Cuando un procesador desea acceder a un módulo de memoria común, envía una petición de uso de dicho



M_m : Módulo de memoria común.

B_1 : Bolque de memoria común. Todos los bloques ocupan las mismas posiciones del espacio de memoria.

Fig 3.3b Interleaving de alto orden utilizando un registro de mapping con palabras consecutivas en un módulo.

módulo. Realizada la petición el procesador entrará en un "estado de espera" hasta que se le concede el acceso al recurso. Este estado de espera puede ser de 2 tipos:

- El procesador queda detenido hasta que se puede realizar el acceso al recurso. Este modo de espera es imprescindible para las llamadas a monitores desde procesos en la memoria común o desde otros monitores, ya que en estas llamadas el procesador depende de la concesión de un recurso común para poder seguir ejecutando instrucciones.

- El procesador debe ir preguntando sobre la condición de acceso hasta que esta se cumpla y poder acceder al recurso. En este modo de espera el procesador no queda bloqueado. Este modo de espera sólo lo pueden utilizar procesos en memorias privadas, para que el procesador pueda seguir ejecutando un programa que no dependa de la concesión de un recurso común.

El uso de un módulo de memoria común, una vez concedido el acceso al mismo, puede realizarse de dos modos diferentes, seleccionables desde cada procesador:

-Bloqueando el recurso (módulo de memoria).

-Haciendo Interleaving.

Estos dos modos de acceso están soportados basicamente por hardware. Modificando el contenido de un campo dentro de un cierto registro de entrada/salida (campo

de modo), antes de acceder a un recurso común, se podrá seleccionar en cada procesador, el modo en el que dicho recurso va a ser utilizado.

-**Bloqueando el recurso:** La petición de acceso al módulo de memoria se realiza al almacenar el código correspondiente a este modo de operación en el campo de modo del mencionado registro de entrada/salida. Una vez concedido el recurso, éste quedará bloqueado, no permitiendose accesos al mismo desde otros procesadores mientras se mantenga activa la petición actualmente en servicio.

Acceder al recurso bloqueandolo, nos permite la sincronización entre procesos a través de los monitores, puesto que estos necesitan realizar la ejecución de las regiones críticas de forma indivisible.

-**Interleaving:** Con este segundo método ofrecemos la posibilidad de que un módulo de la memoria global pueda ser utilizado por más de un procesador para ejecutar concurrentemente los distintos procesos y/o monitores ubicados en dicho módulo.

Una vez indicado este método de operación en el

campo de modo del registro de entrada/salida, la petición al bloque de memoria seleccionado se realizará cada vez que la dirección que circule por el bus interno del procesador corresponda al espacio de direcciones de la memoria común.

El interleaving se puede realizar "ciclo a ciclo" o "instrucción a instrucción". En el prototipo realizado se hace instrucción a instrución y no ciclo a ciclo debido a características internas del microprocesador utilizado, que no permite que se le pare en un ciclo en el que va a ejecutar una operación de escritura.

Haciendo interleaving, podemos ejecutar en el mismo módulo programas independientes o monitores una vez se haya realizado el cierre correspondiente (regiones críticas). Sin embargo para muchos procesadores el interleaving puede degradar el rendimiento de la memoria incrementando el número de conflictos en accesos a memoria que ocurre cuando ciclos continuos de los procesadores siguen una secuencia de direcciones consecutivas de la memoria.

Bloqueando el recurso un procesador puede ejecutar los procesos y monitores correspondientes sin retardos

extras por conflictos de acceso a los recurso comunes, pero con la desventaja de quedar dicho recurso bloqueado para el acceso desde otros procesadores.

La elección de uno u otro modo en cada petición de acceso a memoria común será responsabilidad del programador, en función de las necesidades impuestas por el mecanismo de acceso. En cualquier caso, hay que señalar que el uso del bloqueo del recurso, debe limitarse tanto como sea posible, mediante la optimización de los códigos residentes. Además debe garantizarse la terminación del bloqueo en todas las condiciones, para evitar un colapso en el uso de un recurso común.

3.5 Comunicación entre recursos comunes.

Debido al "esquema de mapping" adoptado, las llamadas de un módulo de memoria común a otro en diferentes bloques (comunicación entre módulos de memoria de bloques diferentes), no se puede hacer directamente, ya que los procesadores ven los diferentes bloques de memoria común en las mismas posiciones del espacio de direcciones.

El establecimiento de una comunicación entre módulos en diferentes bloques, ya sea por la llamada de un proceso global a un monitor, o bien porque un monitor llame a otro monitor, se realizará mediante el uso de un programa almacenado en la memoria local de los procesadores. Dicho programa necesita los siguientes parámetros:

-Bloque de memoria al que se desea acceder.

-Modo de acceso

-Dirección de acceso.

-Bloque desde el que se hace la llamada.

-Modo en el que se debe hacer el retorno.

-Dirección de retorno.

3.6 Clasificación de los procesos.

En el sistema multiprocesador que proponemos existe la posibilidad de definir un mayor número de procesos que de procesadores. Las tareas (procesos), pueden ser clasificados en:

-Procesos locales, son los procesos que disponen de una asignación fija de procesador. Estos procesos estaran almacenados en las memorias locales de los procesadores.

-Procesos globales, son procesos que no tendran

procesador propio, pueden ser ejecutados por cualquiera de los procesadores del sistema y se les asigna procesador cuando existe alguno libre. Los procesos globales están almacenados en la memoria común.

3.7 Distribución de trabajo en los procesadores.

En un sistema multiprocesador existen basicamente dos decisiones de asignación de recursos, una es la "decisión de colocación", es decir, cómo distribuir el código y los datos en la memoria física y la otra es una "decisión de asignación", es decir en qué procesador se va a ejecutar cada proceso.

En nuestro sistema hemos considerado que los procesos estan ordenados en niveles de prioridad, dependiendo de su función y especificaciones en la aplicación tales como procesos dirigidos por interrupción, tiempo de respuesta limitado por la actuación del proceso,... Los procesos de mayor nivel de prioridad estaran almacenados en memorias locales (procesos locales), de esta forma cuando son activados tienen su propio procesador asignado y sólo pueden ser ejecutados por dicho procesador.