3. MULTIPROCESADORES CON MEMORIA COMPARTIDA

3.1. ASPECTOS FUNDAMENTALES EN EL DISEÑO DE UN MMC

El parámetro que fundamentalmente determina el rendimiento de un sistema MMC es el coste de los accesos a la memoria compartida. Este parámetro viene determinado por las características de la red de interconexión utilizada y por el tráfico que debe soportar dicha red.

En este apartado se describen los aspectos más importantes relacionados con el diseño de un sistema MMC. Concretamente, en el apartado 3.1.1 se describen las características principales de los diferentes tipos de redes de interconexión propuestos y cómo afectan estas características en el rendimiento del sistema.

3.1.1. Redes de interconexión para MMC

La red de interconexión de un sistema MMC debe permitir que cualquier procesador acceda a cualquier módulo de memoria. En esta sección se describen diferentes tipos de redes de interconexión y sus características más importantes. Para ello, realizaremos en primer lugar, algunas definiciones importantes.

A lo largo de la sección supondremos que cada módulo de memoria sólo puede servir una petición de acceso en cada instante. En el caso de que dos procesadores intenten acceder simultáneamente al mismo módulo de memoria se produce un conflicto y una de las peticiones debe ignorarse. La frecuencia con que se producen los conflictos es independiente del tipo de red de interconexión. Depende fundamentalmente de cómo se han distribuido los datos en los módulos de memoria.

En el caso de que dos procesadores intenten acceder simultáneamente a dos módulos diferentes de memoria y estas peticiones no puedan ser atendidas en

paralelo diremos que se ha producido contención. La frecuencia con que se producen contenciones depende del tipo de red de interconexión.

Por último, diremos que un sistema es tolerante a fallos si el sistema puede seguir funcionando, probablemante de forma degradada, aunque falle algunos de sus componentes.

A continuación, se describen diferentes tipos de redes de interconexión y se comparan entre sí. Se hace especial hincapié en el coste de la red, el grado de contención que produce y la tolerencia a fallos.

3.1.1.1. Bus común

Un sistema multiprocesador con bus común tiene el aspecto mostrado en la figura 5. Existe un único bus que atiende todas las peticiones de acceso a memoria. En este caso, basta con que dos procesadores deseen acceder a memoria en el mismo ciclo para que se produzca contención. La circuitería de arbitraje, no mostrada en la figura, es responsable de determinar qué procesador puede ocupar el bús en cada ciclo. En [HwBr84] pueden encontrarse algunos esquemas de arbitraje.

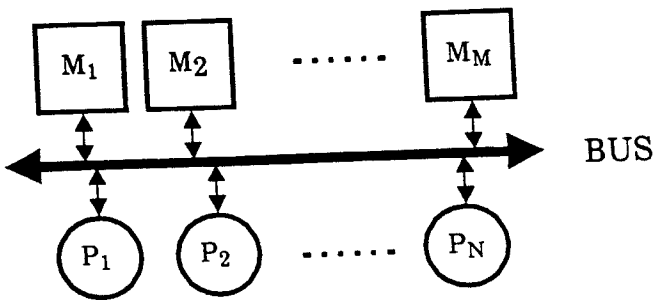

Figura 5: Sistema multiprocesador con bus común

Las características fundamentales de este tipo de red de interconexión son:

- Es barata (en relación a las redes de interconexión que se describirán a continuación).

- La contención es alta y, por tanto, el rendimiento es bajo, especialmente cuando el número de procesadores es elevado.
- No es tolerante a fallos. Si falla el bus el sistema no puede seguir funcionando.

3.1.1.2. Crossbar

La estructura de un crossbar es la mostrada en la figura 6. En este caso, existe un bus independiente para acceder a cada uno de los módulos de memoria. Cada procesador puede utilizar cualquiera de los buses. Por tanto, en este caso no existe contención. Dos peticiones a módulos diferentes siempre podrán ser atendidas simultáneamente.

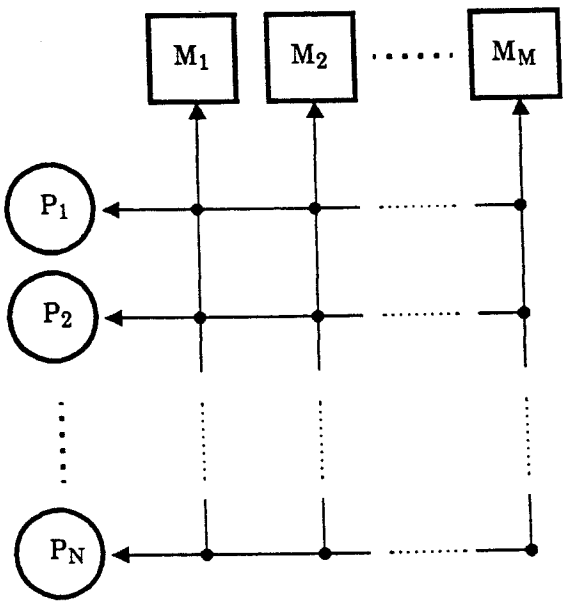

Figura 6: Estructura general de un crossbar

Las características fundamentales de este tipo de red son:

- Es cara. El coste de la red es proporcional al número de conexiones entre los procesadores y los buses (es decir, proporcional a NxM).

- El rendimiento es grande (no hay contención).
- Es tolerante a fallos. Si falla un bus uno de los módulos queda inutilizado pero el resto del sistema puede seguir funcionando.

3.1.1.3. Multiples buses

Un sistema multiprocesador con multiples buses tiene el aspecto mostrado en la figura 7. En este caso, existen B buses. Cualquiera de ellos puede ser utilizado para comunicar cualquier procesador con cualquier módulo de memoria.

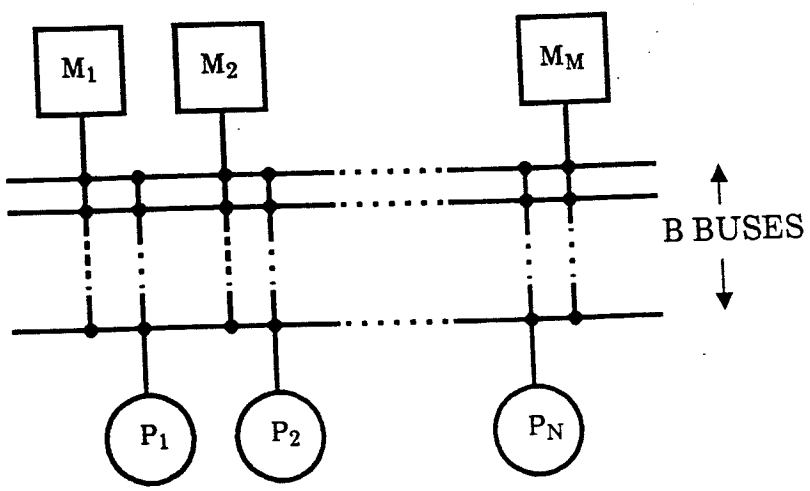

Figura 7: Red de interconexión basada en múltiples buses

Este tipo de red representa un compromiso entre las características de las dos redes descritas en los apartados anteriores. Concretamente, si $B=1$ tenemos un sistema con bus común y si $B=M$ entonces el sistema se comporta como un crossbar.

En cualquier caso, las redes de interconexión basadas en buses no son adecuadas para sistemas con elevado número de procesadores, ya sea por el bajo rendimiento del sistema (si existen pocos buses) o por el coste de la red (si el número de buses es elevado).

3.2. PROGRAMACION DE UN MMC

Para programar un sistema MMC es necesario descomponer los cálculos a realizar en un conjunto de tareas que puedan ejecutarse en paralelo. Cada uno de los procesadores se encargará de ejecutar una (o varias) de estas tareas. Para ello, tendrá acceso a los datos que necesite, y que se encuentran en la memoria común accesibles por todos los procesadores (figura 17).

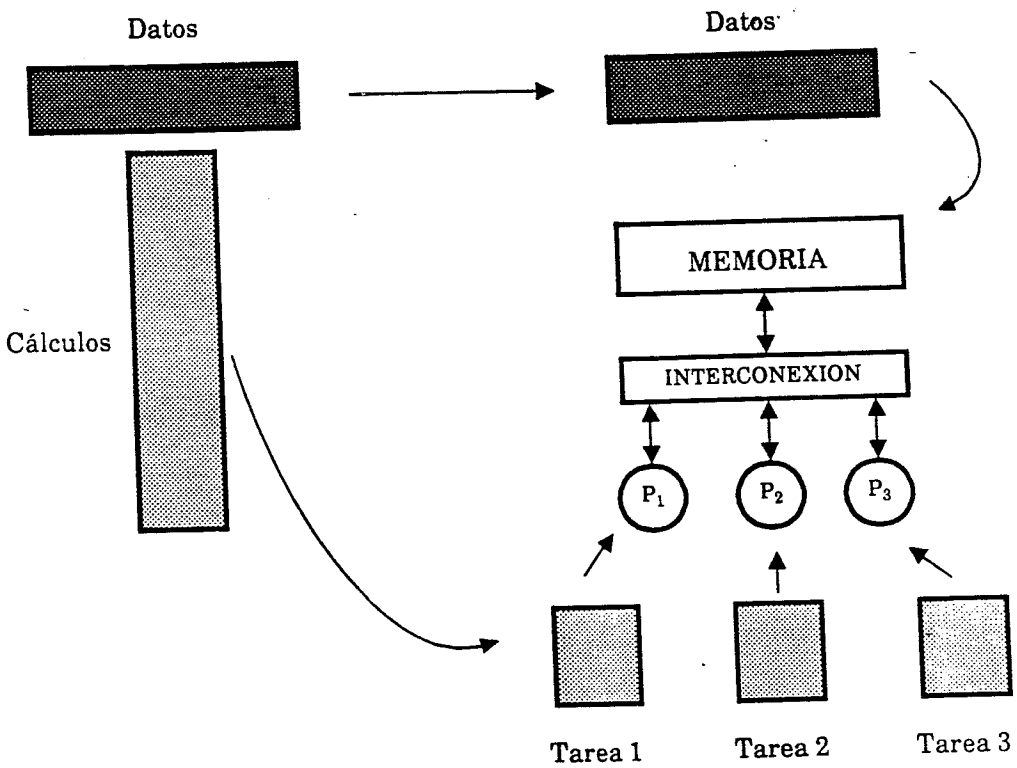

Figura 17: Descomposición de un problema en tareas para su ejecución en un sistema multiprocesador con memoria compartida

En general, no será posible descomponer los cálculos en tareas totalmente independientes entre sí. Es pues necesario dotar al sistema multiprocesador de algún mecanismo que permita a los procesadores sincronizar e intercambiar información.

La descomposición de los cálculos en tareas puede realizarse de forma estática (por parte del programador o por parte del compilador) o de forma dinámica, en tiempo de ejecución. Si la descomposición la realiza el programador, el lenguaje de programación utilizado debe incluir construcciones que permitan al programador especificar las tareas en las que ha descompuesto el problema. Si la descomposición se hace de forma automática, el compilador debe incorporar teécnicas de análisis de dependencias. Este análisis le permitirá determinar, a apartir de la descripción del algoritmo, qué grupos de sentencias pueden ejecutarse en paralelo y en que puntos de la ejecución es necesario incluir operaciones de sincronización. Finalmente, en el caso de que la detección del paralelismo sea dinámica es necesario incorporar al sistema algún mecanismo que permita determinar en cada instante, durante la ejecución, qué operaciones pueden ser realizadas en paralelo. La detección dinámica del paralelismo lleva consigo, en general, un alto overhead en tiempo de ejecución y sólo se ha utilizado en máquinas especializadas en ejecución de lenguajes de programación tales como LISP o Prolog. En este tipo de lenguajes existe un alto nivel de paralelismo implícito que no puede ser detectado totalmete de forma estáatica.

Una vez realizada la descomposición del problema en tareas es necesario realizar la planificación (scheduling) de las tareas. Esta operación consiste en determinar en qué procesador y en qué instante debe iniciarse la ejecución de cada una de las tareas. La estrategia de planificación afectará en gran medida en el rendimiento del sistema durante la ejecución paralela del programa.