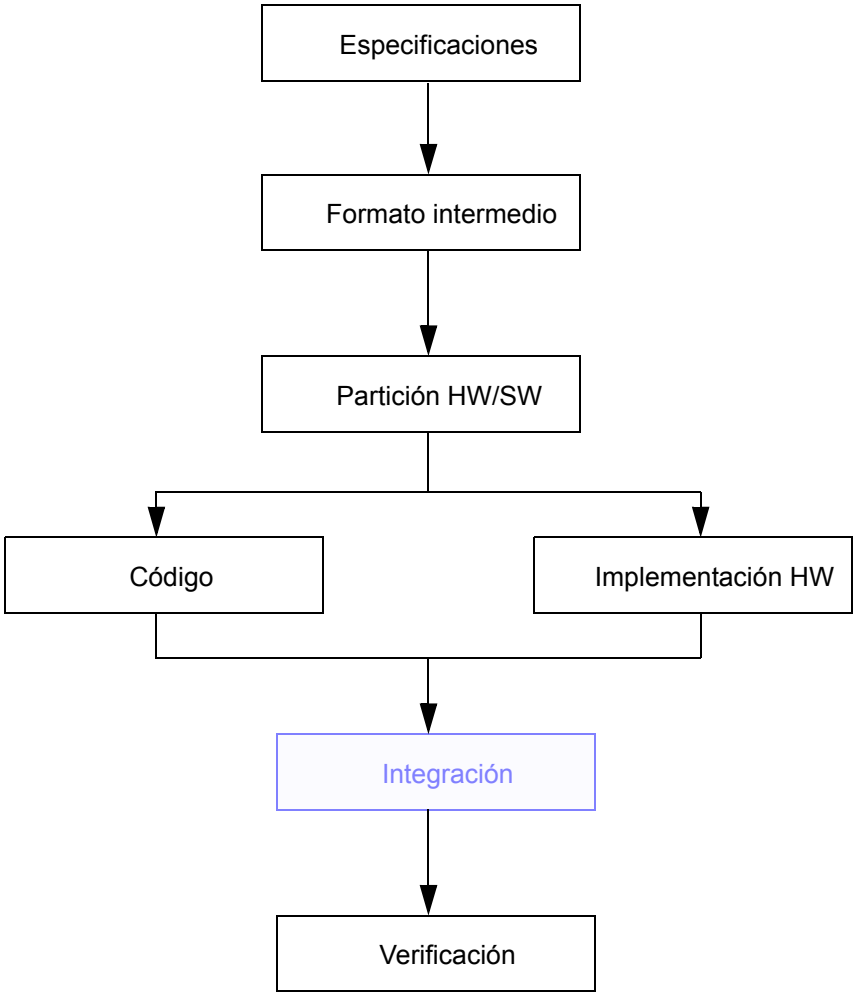
Tema VI

Comunicaciones

Una vez que hemos implementado todas las operaciones, ya sean vía hardware o vía software, debemos integrar todo el sistema. La tarea que queda previa al sistema final es dotar de mecanismos de comunicación a las diferentes operaciones, para que cada una de ellas trabaje con los datos necesarios. Esta última parte es la que se tratará en este tema.



6.1. Introducción

Hasta ahora hemos realizado un modelo del sistema que se desea implementar, se han dividido las operaciones en función del tipo de componentes que las van a implementar, y después se ha llevado a cabo dicha implementación. Por lo tanto, el siguiente paso que nos queda es la integración de todos ellos en un mismo sistema, a través de los mecanismos de comunicación entre los diferentes componentes.

Los procesos en el nivel de sistema se comunican a través de canales abstractos que modelan mecanismos de comunicación. Entre estos mecanismos podemos encontrar sistemas de mensajería síncrona o asíncrona, llamadas a procedimiento remotos y monitores. Para implementar dichos mecanismos, se debe generar el hardware y software que interconecta los componentes del sistema y permite que los procesos se comuniquen entre sí, y con periféricos y otras interfaces externas. Esta generación presenta una serie de dificultades que vienen motivadas entre otras razones por:

- una elevada diversidad de dispositivos y procesadores que necesitan comunicarse
- unas prestaciones complejas
- una falta de apoyo apropiado para el desarrollo y debugueo de los sistemas software

Por lo tanto, la existencia de herramientas de diseño es esencial para mejorar la productividad del diseño.

La síntesis del sistema de comunicación se puede ver como una tarea de diseño top-down consistente en tres procesos bien diferenciados:

- Unión de canales
- Refinamiento de la comunicación
- Generación de interfaces

De tal forma que el resultado final, es decir, el sistema en el que están integrados todos los elementos (ya sean hardware, software y de comunicaciones), debe garantizar el comportamiento inicialmente especificado y satisfaces las prestaciones y restricciones impuestas por las especificaciones de diseño.

La unión de los canales es similar a un proceso de allocation (como se muestra en el tema VI) con los registros donde amacenar la información. De tal forma que si dos datos no necesitan ser transmitidos en un mismo instante temporal, podemos compartir líneas del bus para dichos datos. Por lo tanto, las líneas que necesitará un bus serán las siguientes:

- Líneas de dato, serán las líneas a través de la cual irá la información que se desea transmitir.
- Líneas de identificación, serán las líneas que identifiquen el grupo de líneas de dato de cada uno de los canales de comunicación.
- Líneas de control, serán aquellas que gobiernen el protocolo de comunicaciones para garantizar una correcta transmisión de los datos.

6.2. Protocolos de comunicaciones

Una vez que tengamos identificados los canales de comunicaciones que necesitamos, el siguiente paso será seleccionar y generar el protocolo de comunicaciones para garantizar una comunicación correcta entre los diferentes componentes del sistema. En función del protocolo elegido, cambiará el número de señales de control, y por lo tanto, el tamaño del canal de comunicaciones.

Entre los protocolos más utilizados podemos encontrar el protocolo basado en strobe y el protocolo handshake. El protocolo strobe únicamente dispone de una sola señal de control cuyo tiempo a nivel alto indica la duración del intercambio de datos, es decir, el tiempo que permanece estable el dato de entrada, como se muestra en la figura 6.1. En este caso no existe una comunicación real entre el proceso productor de información y el receptor; únicamente el receptor mira la línea de control para ver si el dato de entrada existente en la línea de dato es válido. En cambio, el productor no tiene ninguna información del momento en el cual el receptor ha cogido la información, y así ha acabado el intercambio real de datos; por lo tanto, el tiempo que se dejan activos los datos suele estar fijado para todos los intercambios.

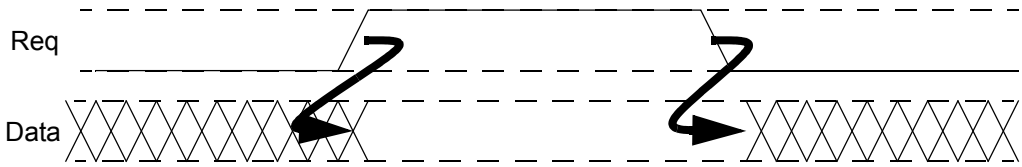


Figura 6.1.- Esquema temporal de protocolo strobe.

Por contra, el protocolo handshake dispone de dos señales de control utilizadas por el productor y receptor de información, produciéndose una comunicación real entre ambos procesos. La secuencia de eventos se muestra en la figura 6.2(a). El productor de información envia una petición (una transición de subida en la señal req) indicando que la información colocada en las líneas de datos es válida. Esta información es capturada por el receptor, indicándolo a través de una subida en la señal de reconocimiento (ack). En este momento ya se ha completado la comunicación, no obstante hay que devolver el medio de comunicación a su estado inicial; por lo tanto, la señal de petición debe bajar para que se puedan cambiar la información de las líneas de datos; y depués bajar la señal de reconocimiento indicando que ya no se está capturando ninguna información. Este protocolo es denominado protocolo handshake de cuatro fases, es decir, hay que completar cuatro eventos para poder realizar una nueva comunicación. En este caso, el significado de cada evento depende del nivel de cada señal. No obstante, existe otra versión de este protocolo denominada de dos fases en el cual el significado de cada evento depende de la transición de cada señal. Esta versión se muestra en la figura 6.2(b), y la secuencia de eventos es como sigue. El productor envia una petición (una transición en la señal req) indicando la validez de los datos, y se espera a que el receptor recoja la información (una transición en la señal ack). Una vez reconocida el traspaso de información, los datos pueden volver a cambiar, e iniciar una nueva comunicación. Este último protocolo tiene la ventaja de una mayor velocidad que el de cuatro fases (típicamente el doble, ya que en un ciclo completo se realizan dos comunicaciones); por contra tiene la desventaja de una mayor complejidad en la implementación de la interfaz ya que se deben utilizar componentes sensibles a la transición y no al nivel. Por lo tanto, vamos a centrarnos en el protocolo de cuatro fases.

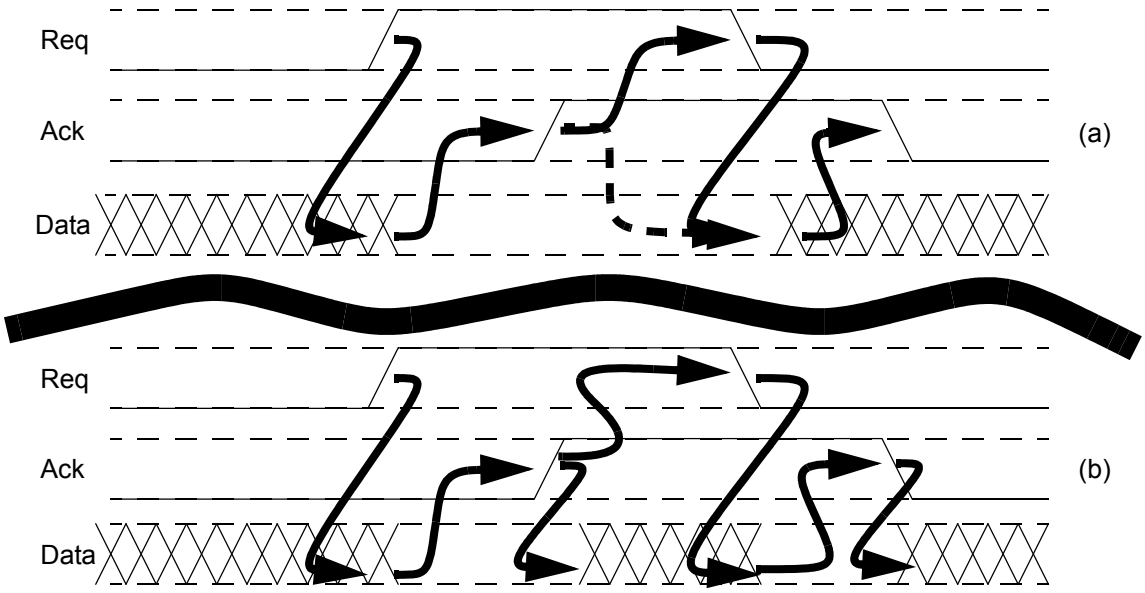


Figura 6.2.- Esquema temporal de protocolo handshake (a) de cuatro fases y (b) de dos fases.

Una forma de representar los ciclos de operación handshake es utilizando los denominados STG (Signal Transition Graph), que no es más que una red de petri modificada, en la que se representan las transiciones de señal. El STG correspondiente al protocolo de cuatro fases se muestra en la figura 6.3a, en el que podemos observar la secuencia de transiciones que se debe garantizar para que exista una correcta comunicación entre los dos procesos. La unión de dos ciclos de operación handshake de dos procesos comunicantes (un proceso y un recurso) generará la composición del interfaz de comunicaciones. Tenemos dos variantes principales mostradas en la figura 6.3b: protocolo fullhandshake, en el que con la secuencia Ack_1^+ -> Req_2^+ se está desvinculando las operaciones de los dos procesos y por lo tanto ambos pueden ejecutar operaciones simultáneamente; y protocolo halfhandshake, en el que con la secuencia Req_1^+ -> Req_2^+ se vincula ambos procesos y hasta que no acabe el proceso destino no puede comenzar un nuevo proceso fuente.

Además de los protocolos anteriores, podemos encontrar protocolos más comerciales y dirigidos hacia sistemas típicamente informáticos, tales como los sistemas ISA, PCI o USB.

Los protocolos utilizados deben realizar una segunda tarea, muy relacionada con la comunicación correcta, que se trata de gestionar la petición de varios procesos a un mismo recurso, es decir, lo que se suele conocer como conflictos de accesos. Este problema se puede definir de la siguiente forma:

> Un conflicto de accesos sucede cuando dos o más procesos intentan acceder al mismo recurso simultáneamente.

Las situaciones en las que se suelen producir los conflictos de accesos son las siguientes:

- Debido al agrupamiento de canales y variables durante la partición del sistema, y que dichos agrupamientos puedan ser accedidos simultáneamente por diferentes procesos.

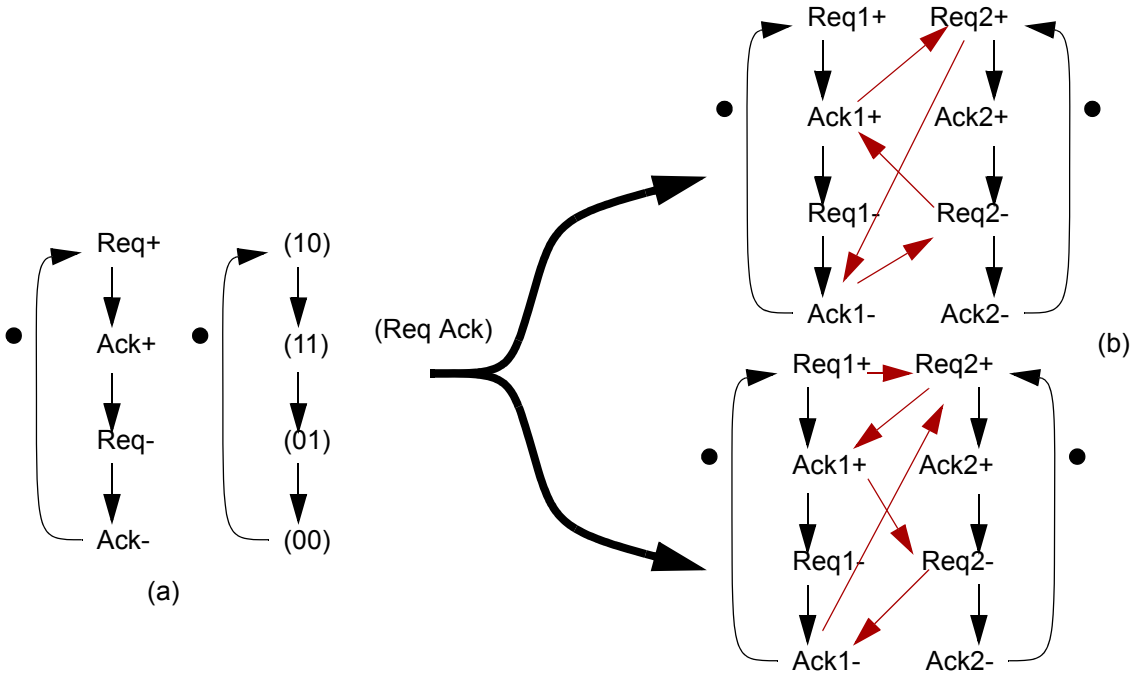


Figura 6.3.- STG correspondiente a un protocolo handshake de cuatro fases.

- Todos los canales de comunicacióon sean agrupados juntos para ser implementados en un único bus.
- El recurso dispone de menos canales de comunicación que el número de procesos que puede comunicarse con el simultáneamente.

La forma de resolver estos conflictos es a través de mecanismos de arbritación, que son implementados en un arbritador. La generación de un arbritador es una tarea de refinamiento que inserta un mecanismo de arbritación siempre y cuando haya un conflicto de acceso. Los modelos de arbritación más utilizados son la arbritación estática y dinámica. En el caso de la arbitración estática, la conexión a los canales de comunicación del recurso es estática, es decir, un mismo proceso siempre se conectará por el mismo canal. En cambio, en la arbitración dinámica, la conexión a los canales de comunicación es dinámica, y por lo tanto pueden cambiar de un acceso a otro. Esta situación se muestra en la figura 6.4, en el que en el caso estático, el proceso 1 siempre estará conectado al puerto 1; mientras que en el caso dinámico, el puerto al que estará conectado dependerá del arbritador.

La implementación del arbritador irá incluido en el circuito de interfaz de control del bus. La arbritación consiste en implementar un esquema de prioridades. Estos esquemas se pueden dividir en dos grandes grupos: esquemas de prioridad fija y esquemas de prioridad dinámica.

En el caso de la prioridad fija, cada canal tiene una prioridad predeterminada que no puede cambiar según el estado del sistema. La asignación de las prioridades no es una tarea simple ni sencilla, ya que puede darse el caso de que un proceso acapare toda la comunicación en detrimento del resto de procesos. Por lo tanto, el asignamiento de prioridades debe optimizar una determinada métrica, que suele ser el tiempo medio de espera de un proceso para acceder al recurso. En el caso que se minimice el tiempo medio de espera, se estará acelerando la

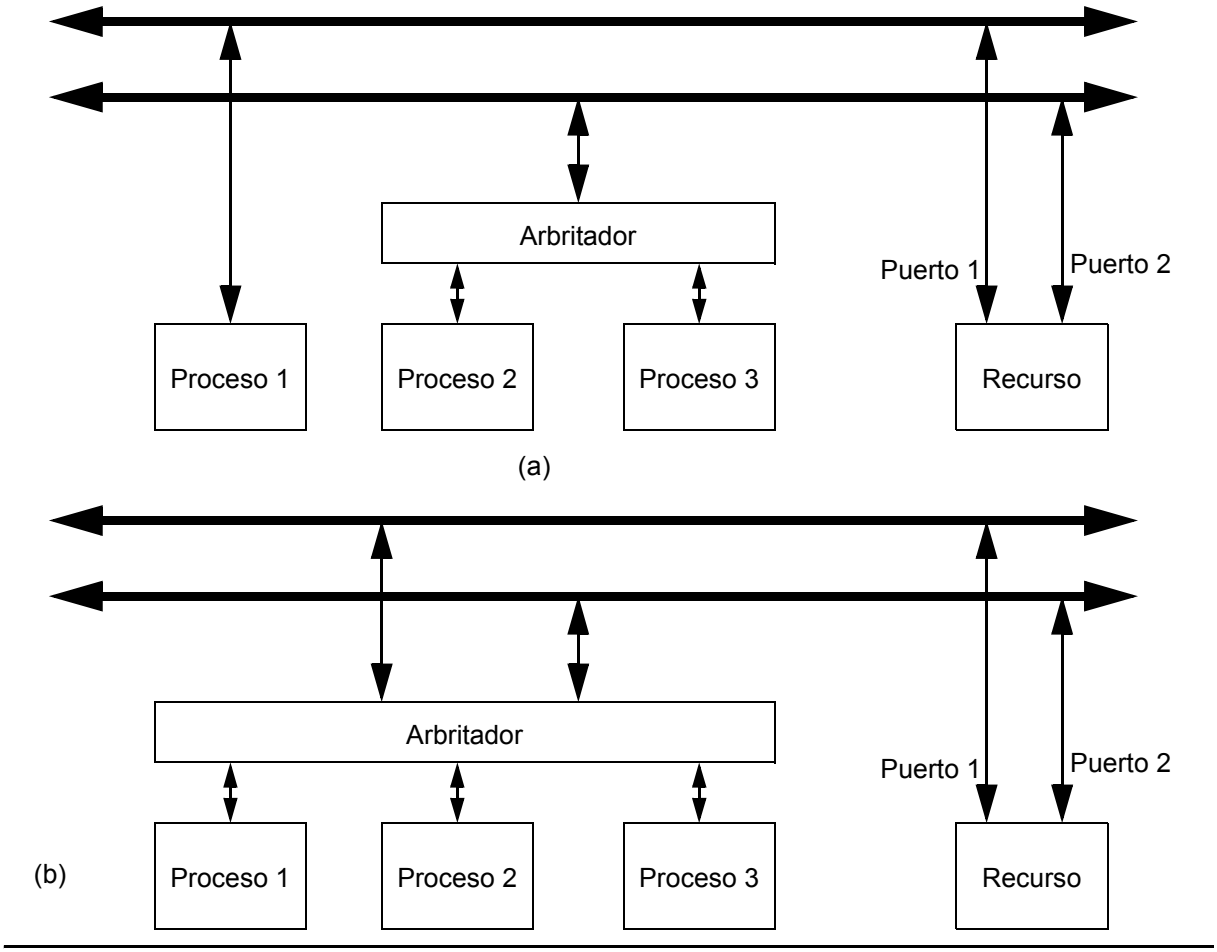


Figura 6.4.- Esquemas de arbritación estática (a) y dinámica (b).

velocidad de acceso al recurso por todos los procesos. Otra métrica utilizada es el tamaño de los datos asociados a un acceso; ya que si no se restringe este parámetro, un acceso podría utilizar un tiempo excesivamente grande por el elevado tamaño de datos que tienen que ser transferidos. Luego, para determinar la prioridad de cada proceso, el diseñador puede utilizar un solo criterio o una combinación ponderada de ellos.

En cambio, en el caso de la prioridad dinámica, las prioridades de cada canal cambian en función del estado del sistema. Así podemos encontrar un esquema rotativo en el cual, el proceso que acaba de acceder al recurso pasa a tener la prioridad más baja. De esta forma se garantiza que todos los procesos tienen garantizados el acceso al recurso común.

6.3. Diseño de un bus

Un bus es uno de los canales más utilizados en los sistemas empotrados, ya que el número de elementos a conectar es relativamente alto. En este caso, las conexiones punto a punto no serían apropiadas ya que ocuparían la mayor parte del área del sistema. Esta situación también implica que la adición de nuevos subsistemas sea relativamente simple, al no necesitar ningún cambio en los subsistemas ya existentes.

En primer lugar vamos a definir que es lo que entendemos por un bus:

> un bus se puede definir como un canal de comunicación distribuido para conectar múltiples subsistemas, es decir, una herramienta para componer sistemas.

En nuestro caso, los subsistemas que vamos a conectar serán los elementos software (como procesadores, microcontroladores o DSPs) y elementos hardware (como ASICs o FPGAs), como se puede ver en la figura 6.5. No obstante, un bus no se limita a un conjunto de cables por los que van los bits de información, sino que también debe garantizar unas especificaciones de protocolo y señales. Para esto último es necesario utilizar un controlador del bus (que generalemente no aparece en los esquemas).

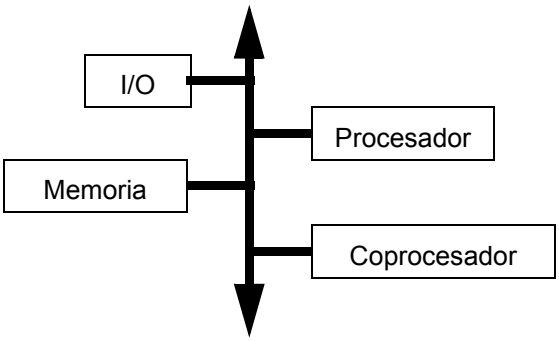


Figura 6.5.- Arquitectura clásica de un sistema empotrado.

Si nos centramos en los cables, en el interior de un bus podemos encontrar tres tipos diferentes de señales:

- Señales de dato, en las que va la información real que se va a tranferir.
- Señales de control, en las que va la información relativa al estado de la transferencia, y serán fundamentalmente señales de protocolo.
- Señales de identificación, en las que va la información relativa al subsistema que está tranferiendo la información.

Veamos un ejemplo de transferencia en el sistema de la figura 6.5, en el cual tanto el procesador como el coprocesador intentan comunicarse con el sistema de memoria y el último (el coprocesador) tiene una prioridad mayor, mostrado en la figura 6.6. En este caso, tanto el procesador como el coprocesador han realizado una petición de transferencia. En primer lugar ha sido atendida la petición del coprocesador debido a su mayor prioridad. Este elemento se identifica y coloca los datos en su línea de datos; seguidamente completa el protocolo de comunicaciones. Una vez que el coprocesador ha realizado su transferencia, el procesador puede empezar con un comportamiento similar. Obviamente en dicho esquema no aparecen todas las señales que están involucradas en la transferencia ya que no se ha considerado el controlador del bus, en el que estaría implementado el esquema de arbritación y el protocolo.

Si consideramos el controlador del bus y separamos las líneas de protocolo y de dato, como se muestra en la figura 6.7a, nos quedaría el cronograma representado en la figura 6.7b. En este cronograma sí podemos apreciar la operación full-handshake del protocolo elegido (Ack_1^+ -> Req_2^+), por lo que el coprocesador o el procesador pueden realizar una nueva operación mientras se está realizadno la transferencia, situación determinada porque cuando el sis-

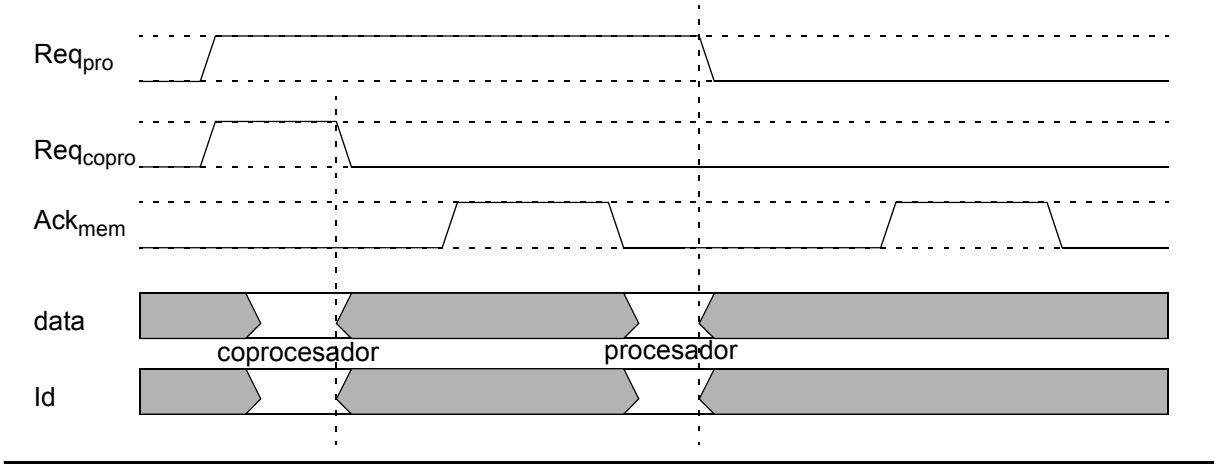


Figura 6.6.- Esquema de señales de un transferencia entre procesos.

tema de memoria está realizando la operación (tiempo durante el cual se encuentra la señal Req_{mem} activa), los elementos anteriores están llevando a cabo un nueva operación que aun no ha terminado (tiempo durante el cual las señales Req_{copro} o Req_{pro} no están activas). También podemos apreciar que los datos de cada elemento (procesador o coprocesador) no se colocan en el bus hasta que el controlador no ha reconocido la transferencia de dicho elemento; para lo cual es necesario que exista en el controlador un registro para almacenar la información mientras se realiza la transferencia y el elemento generador de información empieza el procesamiento de una nueva información.

Una vez que hemos visto las líneas que van agrupadas en el bus, pasaremos al controlador. El control del bus puede realizarse de tres formas diferentes:

- Control centralizado, en el que existe un único controlador que gobierna el bus y todos los elementos deben realizar las peticiones a él para que gestione las transferncias. También puede darse el caso de que el controlador sea un tarea implementada por uno de los elementos del sistema (como puede ser el procesador) en lugar de ser un elemento dedicado exclusivamente a esta tarea.
- Control descentralizado, en el que cada elemento conectado al bus dispone de su propio controlador que gobernará la transferencia de dicho elemento. La arbitración será realizada por todos ellos, en los cuales se implementará un sistema de arbitración local.
- Control semicentralizado, en el que cada elemento conectado al bus dispone de su propio controlador que gobernará la transferencia de información de dicho elemento. La diferencia con el esquema anterior es que existe un controlador central que se encargará básicamente de las tareas de arbitración, en lugar de llevar el peso de toda la transferencia.

Vamos a centrarnos en un controlador central ya que suele ser una buena opción por requerimientos de espacio y velocidad.

En este controlador se debe implementar el esquema de arbritación y de protocolo. En la figura 6.8 mostramos un controlador que implementa un protocolo de comunicación fullhandshake con un esquema de prioridad cíclica. El esquema fullhandsake está compuesto por dos elementos C (que son unos elementos básicos en la implementación de protocolos handshake).

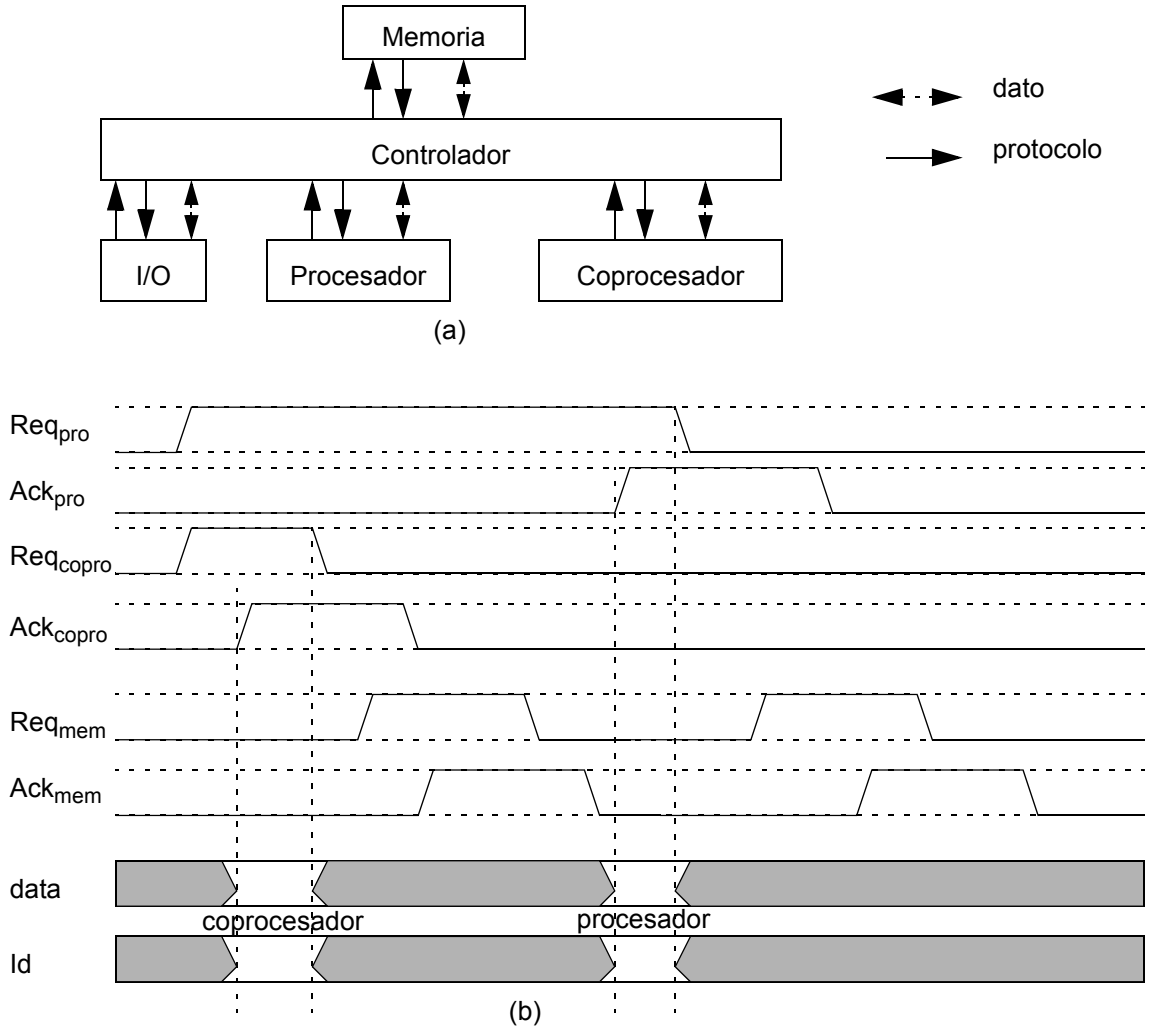


Figura 6.7.- Esquema de señales de un transferencia entre procesos.

Un elemento C es una puerta que toma el valor de sus entradas cuando todas ellas tienen el mismo valor; mientras que si alguna de las entradas tiene un valor diferente a las otras, la salida toma el mismo valor que el instante anterior. Así, la señal Ack1 tomará el valor alto cuando la petición sube y la petición del sistema de memoria baja.

El caso del arbitrador está compuesto por un sistema basado en una FIFO cíclica en la que v pasando un ‘1’ a través de ella. Cada registro de la FIFO identifica a uno de los procesos que se quieren comunicar. Cuando uno de ellos hace una petición, espera a que el token (el ‘1’ que va circulando a través de la FIFO) llegue a su registro. Una vez que ha capturado el token, pasa un ‘0’ hasta que la comunicación ha concluido, momento en el cual vuelve a pasar un ‘1’.

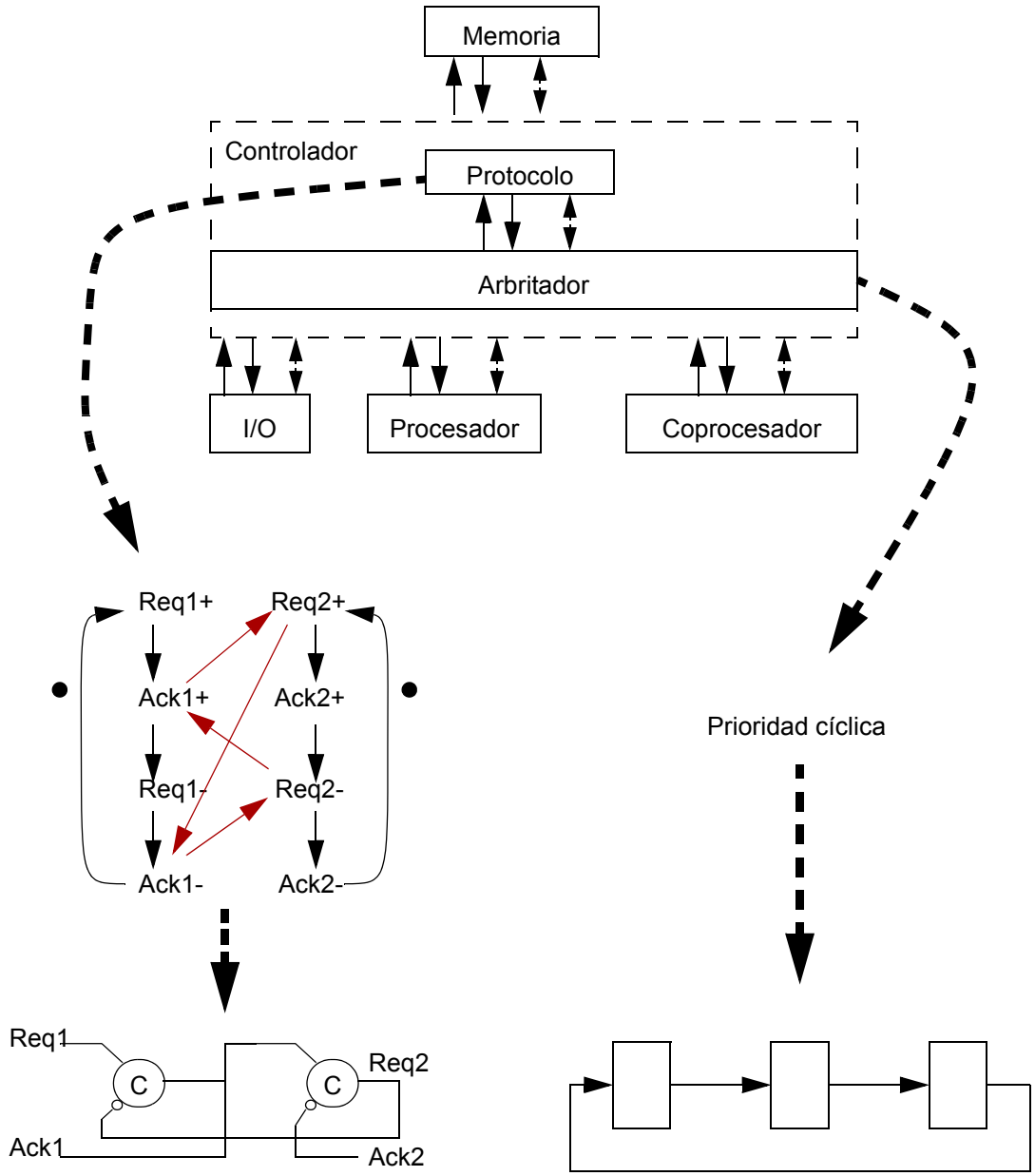


Figura 6.8.- Posible esquema de un controlador que implementa un protocolo fullhandshake con prioridad cíclica.