Tema 10: Buses de comunicación

Objetivos:

- Comprender la estructura y el funcionamiento de un bus como elemento de comunicación entre diferentes unidades de un computador.
- Analizar la forma de sincronización que utilizan los buses para que el intercambio de información sea correcto.
- Describir las diferentes alternativas que se utilizan en los buses para arbitrar su utilización cuando más de una unidad pretende acceder a ellos.
- Introducir la estructura jerárquica que adoptan los buses en un computador a fin de acomodar las diferentes velocidades de las unidades a ellos conectadas.
- Entender la importancia de normalizar las especificaciones de los buses y revisar el funcionamiento de algunos buses estándar.

Contenido:

1. Estructura de un bus

Un bus es un medio compartido de comunicación constituido por un conjunto de líneas (conductores) que conecta las diferentes unidades de un computador. La principal función de un bus será, pues, servir de soporte para la realización de *transferencias* de información entre dichas unidades. La unidad que inicia y controla la transferencia se conoce como *master* del bus para dicha transferencia, y la unidad sobre la que se realiza la transferencia se conoce como *slave*. Los papeles de *master* y *slave* son dinámicos, de manera que una misma unidad puede realizar ambas funciones en transferencias diferentes. Por ejemplo, una unidad de *DMA* hace de *slave* en la inicialización que realiza el *master*, la CPU, para una operación de E/S. Sin embargo, cuando comienza la operación, la unidad de *DMA* juega el papel de *master* frente a la memoria, que en esta ocasión hace de *slave*.

Para garantizar el acceso ordenado al bus, existe un sistema de *arbitraje*, centralizado o distribuido, que establece las prioridades cuando dos o más unidades pretenden acceder al mismo tiempo al bus, es decir, garantiza que en cada momento sólo exista un *master*.

Para establecer el tiempo de duración de las transferencias y que sea conocido tanto por el *master* como por el *slave*, un bus debe disponer de los medios necesarios para la *sincronización master-slave*.

1.1. Líneas

Las líneas de un bus podemos clasificarlas en grupos, atendiendo al papel que cumplen en las transferencias.

1.1.1 Líneas de información básica.

Las utiliza el *master* para definir los dos elementos principales de una transferencia, el *slave* y los datos. Se dividen, pues, en dos grupos: direcciones y datos.

1.1.1.1. Líneas de Direcciones

Determinan la unidad que hace de *slave* en la transferencia

1.1.1.2. Líneas de Datos

Transportan los datos de la transferencia.

Existen buses con líneas independientes para cada uno de los anteriores tipos de información. En cambio en otros se utilizan las mismas líneas multiplexadas en el tiempo.

1.1.2. Líneas de control

Transmiten las órdenes que determinan la operación de transferencia a realizar por las líneas de datos y direcciones, y marcan el ordenamiento temporal de las señales que circulan por el bus. Las primeras son las líneas de control propiamente dichas, de las que las más importantes son:

- Escritura en memoria
- Lectura de memoria
- Operación de salida
- Operación de entrada

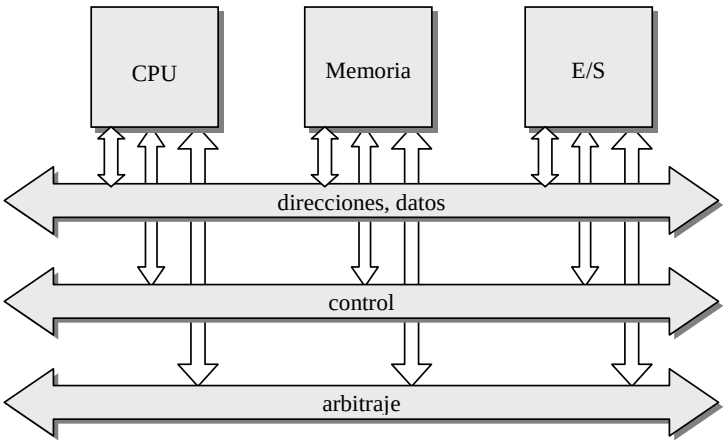
Las segundas son las líneas de sincronismo, entre las que cabe citar las siguientes:

- Reconocimiento de transferencia
- Reloj
- Reset

1.1.3. Líneas de arbitraje

Establecen la prioridad entre diferentes peticiones de acceso al bus. Por ejemplo:

- Petición del bus
- Cesión del bus
- Ocupación del bus

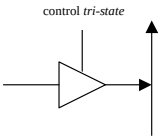


1.2. Direccionalidad de buses

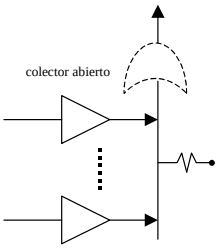
Las líneas de un bus podemos también clasificarlas en función de su direccionalidad:

1.2.1. Líneas unidireccionales

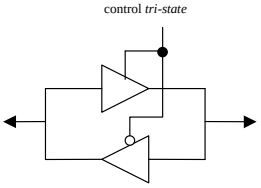
- Emisor simple



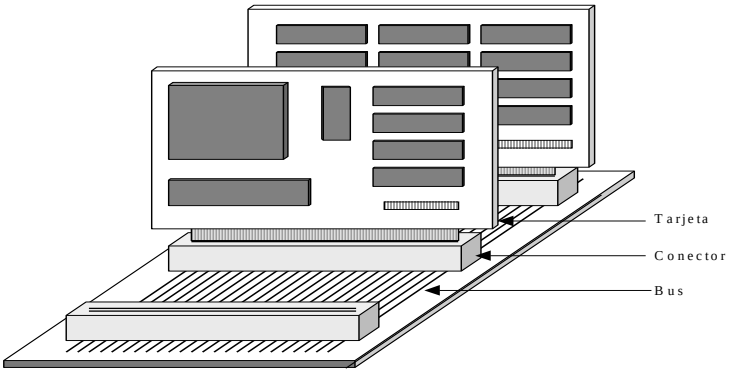
- Emisor múltiple



1.2.2. Líneas bidireccionales



Desde el punto de vista físico un bus es un conjunto de conductores eléctricos paralelos dispuestos sobre una tarjeta de circuito impreso. Los dispositivos del sistema se conectan a través de conectores (***slots***) dispuestas a intervalos regulares a lo largo del bus.



La disposición anterior corresponde generalmente a los buses del sistema. Los denominados buses de entrada/salida conectan las unidades a través de cables.

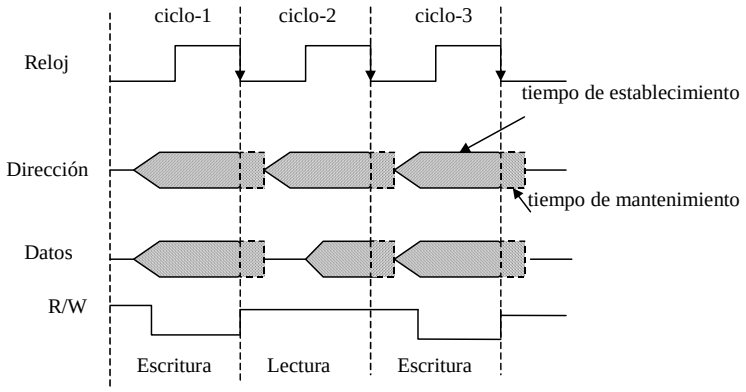
Los principales parámetros que caracterizan un bus son:

- Ancho de banda: velocidad de transferencia medida en Mb/s
- Anchura del bus: número de líneas que lo componen
- Ancho de datos: número de líneas de datos
- Capacidad de conexión: número de unidades conectadas al bus

2. Protocolos de transferencia

2.1. Síncronos

En los buses síncronos existe un reloj que gobierna todas las actividades del bus, las cuales tienen lugar en un número entero de ciclos de reloj. La transferencia propiamente dicha coincide con uno de los flancos del reloj (el de bajada en el ejemplo de la figura).



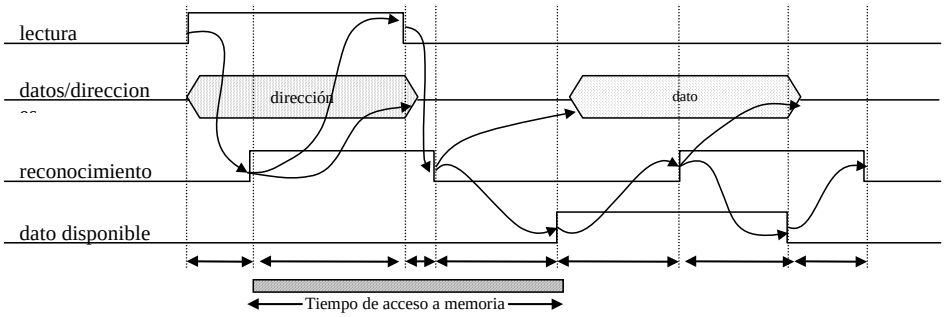
En este caso hemos supuesto que el tiempo de acceso al *slave* es menor de un ciclo, es decir, en cada ciclo tiene lugar una operación con memoria. Los buses síncronos son rápidos pero no tienen capacidad para conectar unidades con velocidad de transferencia baja o no conocida a priori.

2.2. Asíncronos

Los buses asíncronos utilizan un protocolo tipo *handshaking* para comunicarse el *master* con el *slave*. En el siguiente diagrama se presenta el diálogo de señales que tiene lugar durante una transacción de lectura de memoria por parte de la CPU utilizando un protocolo asíncrono (*handshaking*) sobre un bus que multiplexa las direcciones y los datos sobre las mismas líneas (*datos/dirección*).

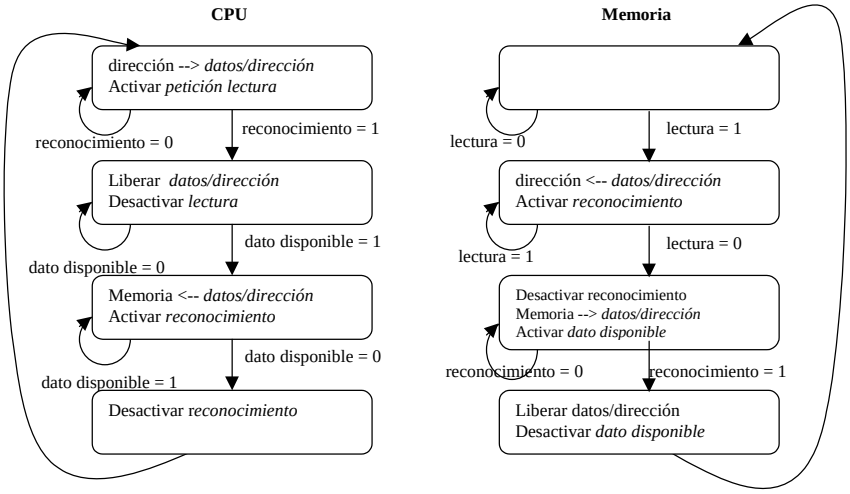
1) La CPU activa la señal de *lectura* al tiempo que coloca la dirección de la posición a leer en las líneas datos/*dirección*.
2) La Memoria detecta la activación de *lectura*, lee la dirección que hay en *datos/dirección* y activa la señal de *reconocimiento* para indicar que ha detectado la orden de *lectura*.
3) la CPU detecta la activación de *reconocimiento* y en respuesta desactiva la señal de *lectura* y libera las líneas de *datos/dirección*.
4) La memoria detecta que se ha desactivado la señal de *lectura* y desactiva la señal de *reconocimiento* para dar por terminada la orden de lectura.
5) Cuando la memoria ha accedido al dato y lo tiene preparado lo pone en *datos/dirección* y activa la señal de *dato disponible*.
6) La CPU detecta que *dato disponible* está activa y procede a leer los datos del bus y activar seguidamente la línea de *reconocimiento* para indicar que ya dispone del dato.
7) La memoria al detectar la señal de *reconocimiento* desactiva *dato disponible* y libera las líneas de *datos/dirección*.
8) Finalmente, la CPU al detectar que se desactiva *dato disponible*, desactiva, a su vez, la señal de *reconocimiento*, indicando que la transmisión ha finalizado.

A partir de este momento se puede iniciar una nueva transacción.



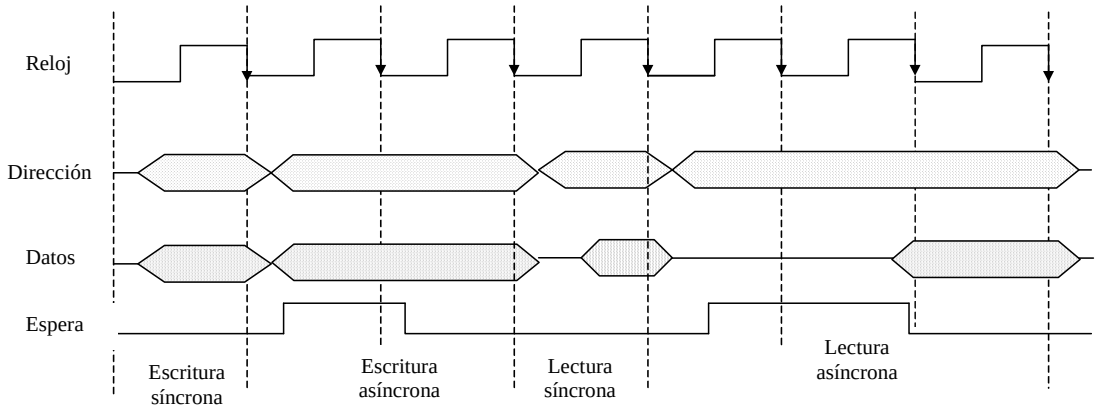
Un bus asíncrono trabaja igual que un par de máquinas de estados finitos que se comunican de tal forma que uno de los autómatas no avanza hasta que sabe que el otro autómata ha alcanzado un determinado estado, es decir, los dos autómatas están coordinados.

Los buses asíncronos se escalan mejor con los cambios de tecnología y pueden admitir una mayor variedad de velocidades de respuesta en los dispositivos.



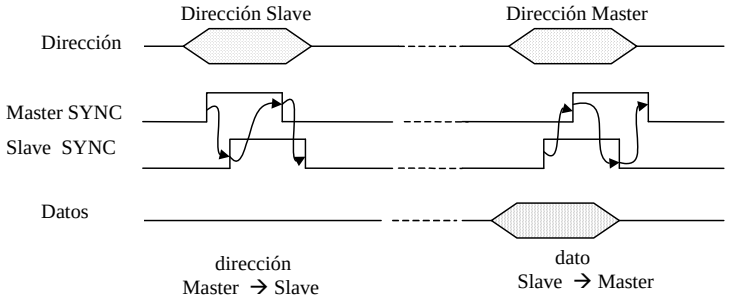
2.3. Semisíncronos

En los protocolos semisíncronos existe, como en los síncronos, un reloj que gobierna las transferencias en el bus. Sin embargo, en este caso existe, además, una señal de *espera* (*wait*) que es activada por el *slave* cuando la transferencia va a durar más de un ciclo de reloj. De esta forma, los dispositivos rápidos operarán como en bus síncrono, mientras que los lentos alargarán la operación el número de ciclos que les sea necesario.



2.4. Ciclo partido

En los buses de ciclo partido la operación de lectura se divide en dos transacciones no continuas de acceso al bus. La primera transacción es la de petición de lectura que realiza el *master* sobre el *slave*. Una vez realizada la petición el *master* abandona el bus. Cuando el *slave* dispone del dato leído, inicia un ciclo de bus actuando como *master* para enviar el dato al antiguo *master*, que ahora actúa como *slave*.



3. Protocolos de arbitraje

La demanda para utilizar el bus en un computador puede provenir de diferentes unidades, no sólo de la CPU. Por ejemplo, si el computador tiene E/S por DMA, éste demandará el uso del bus cuando tenga el control de la operación con la memoria. Los procesadores de E/S necesitan acceder al bus no sólo para realizar las transferencias de datos por DMA, sino también para leer su programa de canal. Por supuesto, si el sistema es multiprocesador (más de una CPU con memoria compartida) los candidatos al uso del bus aumentan.

Para garantizar que en todo momento sólo una unidad acceda al bus, se utilizan los protocolos de arbitraje. Los protocolos de arbitraje organizan el uso compartido del bus, estableciendo prioridades cuando más de una unidad solicita su utilización y garantizando, sobretodo, que el acceso al bus es realizado por un solo *master*.

Existen dos grupos de protocolos de arbitraje, los *centralizados* y los *distribuidos*. En los primeros existe una unidad de arbitraje, el *árbitro del bus*, encargado de gestionar de forma centralizada el uso del bus. El árbitro puede ser una unidad físicamente independiente o estar integrado en otra unidad, por ejemplo, la CPU. Por el contrario, en los protocolos distribuidos no existe ninguna unidad especial para la gestión del bus. Esta se realiza de forma distribuida entre las unidades de acceso.

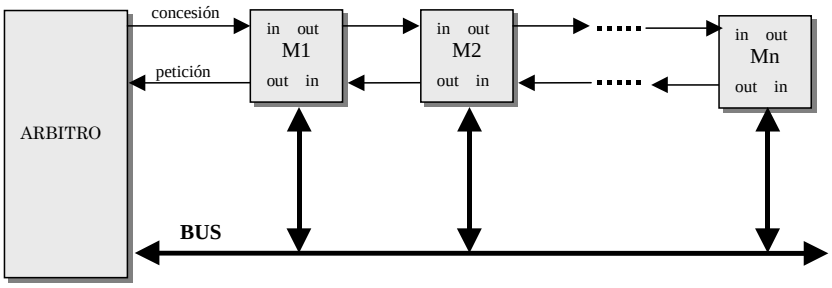
Examinaremos en los siguientes apartados diferentes protocolos de arbitraje, tanto centralizados como distribuidos.

3.1. Protocolo de encadenamiento (*daisy chaining*) de dos señales

Es el protocolo centralizado más sencillo ya que utiliza sólo dos señales encadenadas, una de *petición* del bus y otra de *concesión*.

El *master* que quiere acceder al bus activa la señal de petición (*out*) y los demás *masters* la propagan hasta el árbitro. El árbitro activa la señal de concesión que es propagada por los *masters* que no solicitaron el acceso al bus. El *master* que recibe la señal de concesión y tiene una petición pendiente toma el control del bus. Si un *master* recibe una señal de petición mientras está accediendo al bus, bloquea su propagación al árbitro hasta que finalice la utilización del bus.

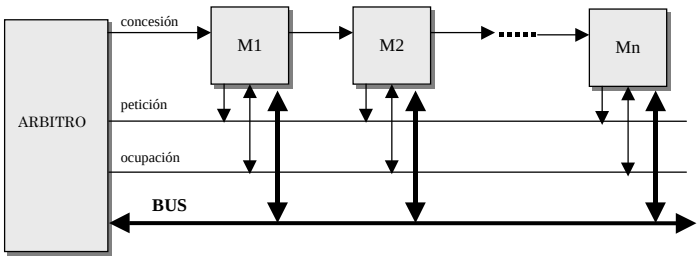
La prioridad viene determinada por la proximidad al árbitro.



3.2. Protocolo de encadenamiento (*daisy chaining*) de tres señales

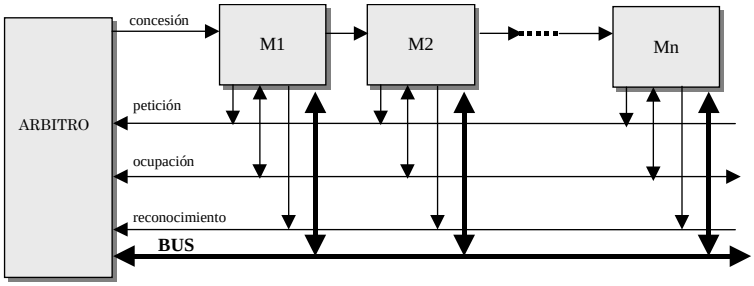
Utiliza una línea más que el protocolo anterior, la línea de *ocupación*. Además, la línea de *petición* no es encadenada sino compartida por todos los *masters* a través de una entrada al árbitro con capacidad de O-cableada.

Cuando un master toma el control del bus activa la línea de ocupación. El árbitro sólo activa la línea de concesión cuando recibe una petición y la línea de ocupación está desactivada. Como en el caso anterior, si un master recibe la concesión y no ha solicitado el bus, transmite la señal al siguiente master. Un master toma el control del bus si tiene una petición local pendiente, la línea de ocupación está desactivada y recibe el flanco de subida de la señal de concesión.



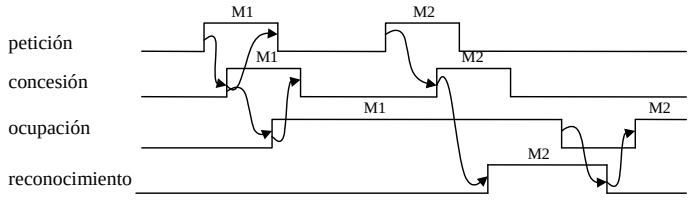
3.3. Protocolo de encadenamiento (daisy chaining) de cuatro señales

Este protocolo permite simultanear el uso del bus por un *master* con el proceso de arbitraje para la selección del *master* siguiente. De esta forma, cuando el primer *master* abandona el bus, no se pierde tiempo en el arbitraje para el siguiente porque ya se ha hecho, pasando directamente el *master* seleccionado a realizar su transacción, al tiempo que se realiza la selección del siguiente *master*. Para ello se añade una cuarta línea al esquema anterior, la línea de *reconocimiento*.



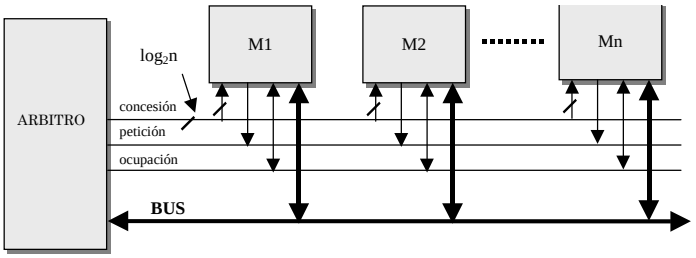
La línea de *reconocimiento* la activa un *master* que solicitó el bus (activó *petición*) y recibió la *concesión* pero la línea de *ocupación* estaba activa (bus ocupado). Cuando el árbitro recibe la activación de *reconocimiento* inhibe su actuación, es decir, deja de atender la señal de petición y generar la de concesión. El *master* queda en espera para ocupar el bus tan pronto lo abandone su actual usuario, que lo hará desactivando la señal de *ocupación*. Cuando esto ocurre, el *master* ocupa el bus y desactiva la señal de *reconocimiento*, con lo que el árbitro recupera su actuación, procediendo a un nuevo arbitraje entre los *master* solicitantes, simultáneamente con la operación de transacción en el bus. En la siguiente figura hemos representado el diálogo de señales

correspondiente a una ocupación del bus por el *master* M1, seguido por el arbitraje a favor de M2 mientras M1 realiza su transacción, y terminando con la ocupación del bus por M2 cuando M1 finaliza:



3.4. Protocolo con concesión por encuesta (polling)

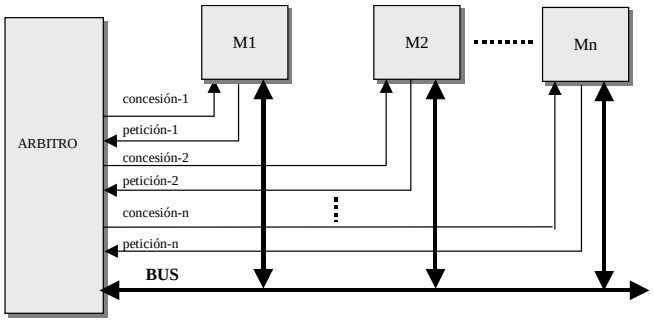
Este protocolo sustituye la línea encadenada de concesión del bus por un conjunto de líneas que permiten acceder de forma selectiva a la dirección asignada cada master sobre estas líneas.



3.5. Protocolo con señales independientes

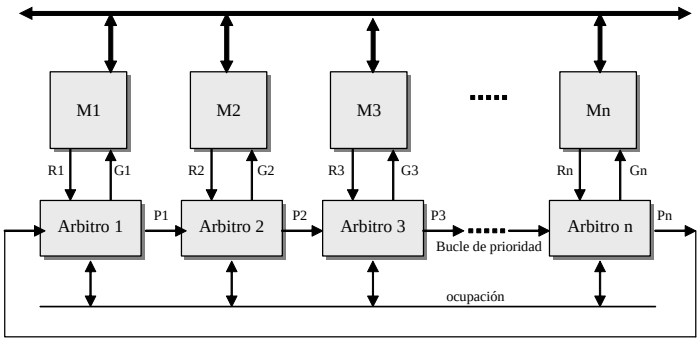
También denominado en estrella, utiliza una línea de concesión específica para cada línea de petición independiente. Esta alternativa tienen la ventaja que el árbitro puede aplicar distintos algoritmos de decisión en caso de peticiones simultaneas (FIFO, prioridad fija, prioridad variable). Además, los retardos de propagación de las señales son pequeños en comparación con las anteriores alternativas.

Tiene la desventaja del número elevado de líneas de arbitraje (una por posible master). El número de master queda limitado al número de líneas existentes.



3.6. Protocolo distribuido

En estos protocolos la responsabilidad del arbitraje no reside en una unidad independiente sino que se distribuye por los diferentes *masters* conectados al bus.

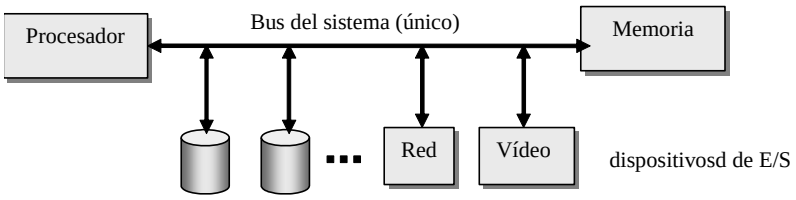


- Arbitro-i concede el bus al *master* Mi activando Gi si:
  - Mi ha activado su línea de petición de bus Ri,
  - La línea de ocupación está desactivada.
  - La línea de entrada de prioridad Pi-1 está activada
- Si el master Mi no ha activado su línea de petición de bus Ri, el Arbitro-i activa la línea de salida de prioridad Pi.

4. Jerarquía de buses

4.1. Bus del sistema (*backplane*)

Los ordenadores antiguos utilizaban una topología de bus único, denominado bus del sistema o *backplane,* para conectar procesador, memoria y los módulos de E/S, tal como la que se muestra en la siguiente figura:

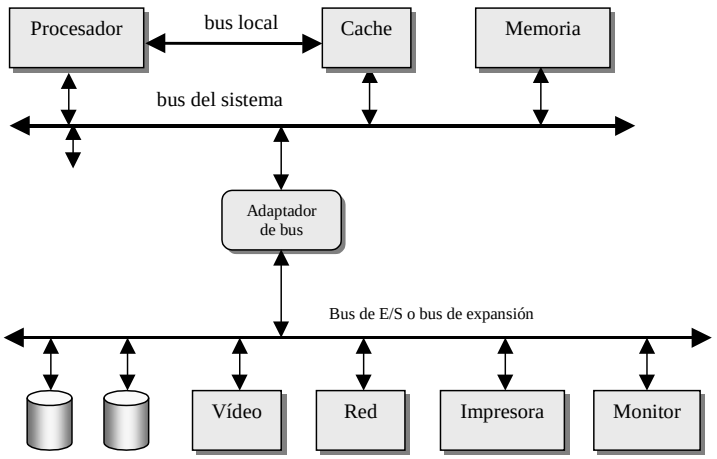


Sin embargo, cuando el número de dispositivos conectados a un bus aumenta disminuye su rendimiento. Esto es debido a dos motivos fundamentales. El primero el aumento del retardo de propagación de las señales debido al aumento de longitud de los conductores que dan soporte al bus. Esto afecta especialmente a la propagación encadenada de la señal de *concesión del bus*. El segundo el incremento de demanda de acceso que se produce al aumentar el número de dispositivos conectados. Este exceso de dispositivos puede crear un *cuello de botella* que haga que el rendimiento del sistema se degrade por la espera inútil que se origina cuando tienen que realizar transferencias.

Por otra parte, las diferencias en la velocidad de operación de los dispositivos conectados al bus, también repercuten negativamente en su rendimiento. En efecto, los dispositivos lentos pueden ocasionar retrasos importantes a los rápidos. Por ejemplo, supongamos que un procesador que opera a *200 MHz* (*Tc = 5 ns, 100 MIPS*) y con un *CPI = 2* se conecta a un único bus compartido por la cache, la memoria y los dispositivos de E/S, entre ellos un disco con *10 ms* de tiempo de acceso y *15 MB/seg* de velocidad de transferencia. Durante la transferencia de disco a memoria de un archivo de *512 KB* se emplearán *10 ms + 512 KB/ 15.000 KB/s = 44,1 ms*. En este tiempo la *CPU* podría haber ejecutado 0,0441 * 100 * 10^6 = 4,41 millones de instrucciones.

Para evitar la caída de rendimiento, el sistema de buses se jerarquiza, apareciendo dos buses más: el *bus local,* y el *bus de E/S*.

4.2. Buses locales



El *bus local* es de longitud pequeña, de alta velocidad, y adaptado a la arquitectura particular del sistema para maximizar el ancho de banda entre el procesador y la caché, por eso suele ser un bus propietario. Este bus aísla el tráfico procesador-caché del resto de transferencias del sistema.

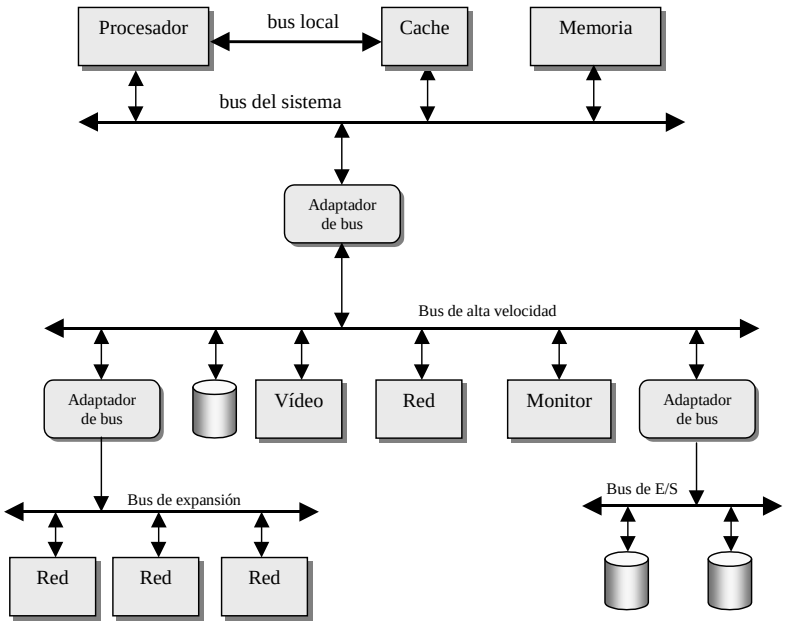
4.3. Buses de E/S o de expansión

El bus de E/S o de expansión reduce el tráfico en el bus del sistema, de manera que el procesador puede acceder a memoria en un fallo de caché mientras realiza una operación de entrada/salida.

Los buses de expansión son buses estándar o abiertos (ISA, EISA, PCI, etc.) es decir, independientes del computador y con unas características bien definidas en el correspondiente documento de normalización. La existencia de estos buses permite diseñar una amplia gama de controladores de periféricos compatibles.

Para conectar los buses del sistema y de expansión se requiere un Adaptador de Bus, dispositivo que permite adaptar las distintas propiedades de ambos buses: velocidad, carácter síncrono o asíncrono, multiplexación, etc.

El proceso de jerarquización se puede complicar más con otras topologías que den cabida a dispositivos de distinta velocidad. De esta forma se equilibra mejor el tráfico de información en sistemas que operan con muchos dispositivos conectados. En la figura siguiente se muestra una topología jerárquica en la que se contempla un bus de alta velocidad del que cuelgan dos buses, uno de expansión para dispositivos rápidos y otro de E/S para dispositivos lentos.



5. Buses normalizados

Las especificaciones de un bus estándar están perfectamente definidas en un documento de estandarización respaldado por alguna sociedad de prestigio en el área (IEEE, etc.). En las especificaciones se distinguen varios niveles:

- **Nivel mecánico**, en el que se recoge la forma y tamaño de los conectores, el número de contactos por conector y el número de dispositivos que soporta el bus.
- **Nivel eléctrico**, en el que se especifican los valores de tensión de las diferentes señales, polarización, etc.
- **Nivel lógico**, especifica la función de cada señal del bus: direcciones, datos, control, etc.
- **Nivel de temporización o sincronismo**, que especifica el protocolo de transferencia empleado
- **Nivel de arbitraje**, especifica el protocolo de arbitraje que utiliza el bus.

5.1. Bus PCI (Peripheral Component Interconnect)

El bus **PCI** es un bus de ancho de banda elevado e independiente del procesador. El estándar actual permite el uso de hasta 64 líneas de datos a 66 MHz, para una velocidad de transferencia de 528 MBytes/s, o 4,224 Gbps. El PCI está diseñado para permitir una cierta variedad de configuraciones basadas en microprocesadores, incluyendo sistemas con uno o varios procesadores. Utiliza temporización semisíncrona y un esquema de arbitraje centralizado.

5.1.1. Estructura

El bus PCI puede configurarse como un bus de 32 o 64 bits. La siguiente tabla define las líneas más importantes obligatorias en el PCI:

CLK (reloj) Señal de reloj que es muestreada en el flanco de subida.

RST# (reset) Hace que todos los registros y señales específicas del PCI pasen al estado inicial.

Señales de direcciones y datos

AD[31:0] Incluye 32 líneas para datos y direcciones multiplexadas en el tiempo.

C/BE[3:0]# Se utilizan para interpretar y validar las líneas de datos y direcciones.

Señales de control de interfaz

FRAME# Activada por el master para indicar el comienzo y la duración de una transferencia. Las activa al comienzo y la desactiva al final de la fase de datos.

IRDY# Señal de *master* preparado (*Initiator Ready*). La proporciona el *master* actual del bus (el iniciador de la transacción). Durante una lectura, indica que el master está preparado para aceptar datos; durante una escritura indica que el dato válido está en AD.

TRDY# Señal de *slave* preparado (*Target Ready*). La activa el *slave* al principio de la transferencia, y la desactiva cuando no puede completar la transferencia en un solo ciclo de reloj

DEVSEL# Señal de *slave* (dispositivo) seleccionado (*Device Select*). Activada por el *slave* cuando ha reconocido su dirección.

Señales de arbitraje

REO# Indica al árbitro que el dispositivo correspondiente solicita utilizar el bus. Es una línea punto-a-punto específica para cada dispositivo.

GNT# Indica al dispositivo que el árbitro le ha cedido el acceso al bus. Es una línea punto-a-punto específica para cada dispositivo.

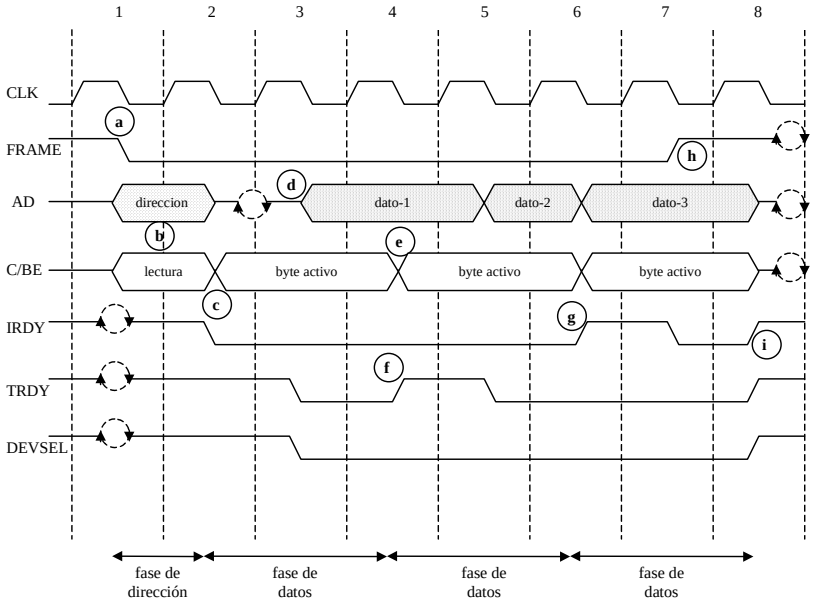
5.1.2. Ordenes

La actividad del bus consiste en transferencias entre dos elementos, denominándose *Initiator* (master) al que inicia la transacción. Cuando un master del bus adquiere el control del mismo, determina el tipo de transferencia que se producirá a continuación. Los tipos, entre otros son los siguientes:

- Reconocimiento de interrupción
- Lectura de E/S
- Escritura en E/S
- Lectura de memoria
- Escritura en memoria

5.1.3. Transferencia de datos

Toda transferencia de datos en el bus PCI es una transacción única, que consta de una fase de direccionamiento y una o más fases de datos. La siguiente figura muestra la temporización de una operación de lectura. Todos los eventos se sincronizan en las transiciones de bajada del reloj, cosa que sucede a la mitad de cada ciclo de reloj.



Los dispositivos interpretan las señales del bus en los flancos de subida, al comienzo del ciclo. A continuación, se describen los eventos significativos señalados en el diagrama:

a) Una vez que el *master* ha obtenido el control del bus, inicia la transacción:
- activando *FRAME*, que permanece activa hasta la última fase de datos
- situando la dirección de inicio en el bus de direcciones
- situando la orden de lectura en las líneas *C/BE*.

b) El *slave* reconoce su dirección en las líneas *AD* al comienzo del ciclo de reloj 2.

c) El *master* deja libre las líneas *AD* del bus y cambia la información de las líneas *C/BE* para indicar qué líneas *AD* se utilizan para transportar datos (de 1 a 4 bytes). También activa *IRDY* para indicar que está preparado para recibir el primer dato (*).

d) El *slave* activa *DEVSEL* para indicar que ha reconocido su dirección. Después sitúa el dato solicitado en las líneas *AD* y activa *TRDY* para indicar que hay un dato válido en el bus.

e) El *master* lee el dato al comienzo del ciclo de reloj 4 y cambia las líneas *C/BE* según se necesite para la próxima lectura.

f) En este ejemplo el *slave* necesita algún tiempo para preparar el segundo bloque de datos para la transmisión. Por tanto desactiva *TRDY* para señalar al *master* que no proporcionará un nuevo dato en el próximo ciclo. En consecuencia el *master* no lee las líneas de datos al comienzo del quinto ciclo de reloj y no cambia la señal *C/BE* durante ese ciclo. El bloque de datos es leído al comienzo del ciclo de reloj 6.

g) Durante el ciclo 6 el *slave* sitúa el tercer dato en el bus. No obstante, en este ejemplo, el *master* todavía no está preparado para leer el dato. Para indicarlo, desactiva *IRDY*. Esto hará que el *slave* mantenga el tercer dato en el bus durante un ciclo de reloj extra.

h) El master sabe que el tercer dato es el último, y por eso desactiva *FRAME* para indicárselo al *slave*. Además activa *IRDY* para indicar que está listo para completar esa transferencia.

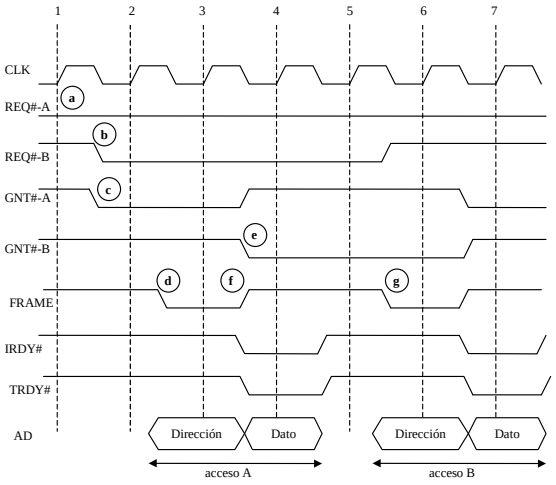
i) El *master* desactiva *IRDY*, haciendo que el bus vuelva a estar libre, y el *slave* desactiva *TRDY* y *DEVSEL*.

(*) **Nota**: En todas las líneas que pueden ser activadas por más de un dispositivo se necesita un ciclo de cambio (indicado por las dos flechas circulares) para que pueda ser utilizado por el dispositivo de lectura.

5.1.4. Arbitraje

El bus PCI utiliza un esquema de arbitraje centralizado síncrono, en el que cada maestro tiene una señal propia de petición (REQ) y cesión (GNT) del bus. Estas líneas se conectan a un árbitro central. La especificación PCI no indica un algoritmo particular de arbitraje. El árbitro puede utilizar un procedimiento de *primero en llegar primero en servirse*, un procedimiento de *cesión cíclica* (*round-robin*), o cualquier clase de esquema de prioridad. El maestro del PCI establece, para cada transferencia que desee hacer, si tras la fase de dirección sigue una o más fases de datos consecutivas.

La siguiente figura es un ejemplo en el que se arbitra a cuál de los dispositivos A y B se cede el bus. Se produce la siguiente secuencia:



a) En algún momento anterior al comienzo del ciclo de reloj 1, *A* ha activado su señal *REQ*. El árbitro muestrea esa señal al comienzo del ciclo de reloj 1.

b) Durante el ciclo de reloj 1, *B* solicita el uso del bus activando su señal *REQ*.

c) Al mismo tiempo, el árbitro activa *GNT-A* para ceder el acceso al bus a *A*.

d) El maestro del bus *A* muestrea *GNT-A* al comienzo del cielo de reloj 2 y conoce que se le ha cedido el acceso al bus. Además, encuentra *IRDY* y *TRDY* desactivados**,** indicando que el bus está libre. En consecuencia, activa *FRAME* y coloca la información de dirección en el bus de direcciones, y la orden correspondiente en las líneas *C/BE*. Además mantiene activa *REQ-A*, puesto que tiene que realizar otra transferencia después de la actual.

e) El árbitro del bus muestrea todas las líneas *GNT* al comienzo del ciclo 3, y toma la decisión de ceder el bus a *B* para la siguiente transacción. Entonces activa *GNT-B* y desactiva *GNT-A*. *B* no podrá utilizar el bus hasta que éste no vuelva a estar libre.

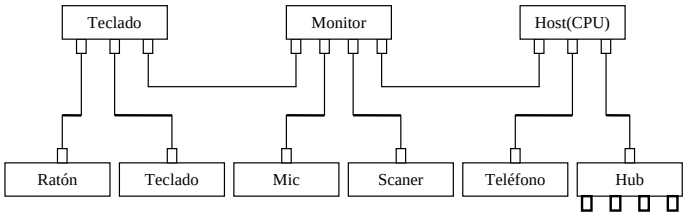
f) *A* desactiva *FRAME* para indicar que la última transferencia de datos está en marcha. Pone los datos en el bus de datos y se lo indica al dispositivo destino con *IRDY***.** El dispositivo lee el dato al comienzo del siguiente ciclo de reloj.

g) Al comienzo del ciclo 5, *B* encuentra *IRDY* y *FRAME* desactivados y, por consiguiente, puede tomar el control del bus activando *FRAME*. Además, desactiva su línea *REQ*, puesto que sólo deseaba realizar una transferencia.

Hay que resaltar que el arbitraje se produce al mismo tiempo que el maestro actual del bus está realizando su transferencia de datos. Por consiguiente, no se pierden ciclos de bus en realizar el arbitraje. Esto se conoce como *arbitraje oculto o solapado* (*hidden arbitration*).

5.2. Bus USB (*Universal Serial Bus*)

El bus *USB* (*Universal Serial Bus*) es un bus normalizado para la conexión de periféricos, desarrollado por un conjunto de empresas de informática y telecomunicaciones (7 compañías: Compaq, DEC, IBM, Intel, Microsoft, NEC y Northern Telecom). Permite conectar de forma sencilla dispositivos periféricos al computador, sin necesidad de reiniciarlo ni de configurar el sistema. Se pueden conectar hasta 127 dispositivos, con una longitud máxima de cable de 5 metros para cada uno, con lo que una conexión en cadena permitiría que el último dispositivo estuviese a 635 metros del ordenador. Trabaja en dos modos, a baja velocidad, 1,5 Mbps, para dispositivos lentos como teclados y ratones, y a alta velocidad, 12 Mbps, para dispositivos rápidos, como CD-ROM , módems, etc. Utiliza un cable de cuatro hilos, dos para datos y dos para alimentación. El bus USB está organizado en una estructura de árbol descendente, con unos elementos especiales, llamados *hubs* que encaminan las señales desde un dispositivo al *host* o viceversa. En la raíz está el host, que es el interfaz entre el bus USB y el bus del ordenador. De él cuelgan los dispositivos USB y los *hubs*, que también son dispositivos USB. A un *hub* se puede conectar uno o más dispositivos, que a su vez pueden ser otros *hubs*.



Cuando se conecta un dispositivo, se detecta la diferencia de tensión en la red *USB* y procede a determinar las características del dispositivo (vendedor, funcionalidad, ancho de banda requerido, etc.). El *host* le asigna una dirección única ID para diferenciarlo del resto de los dispositivos de la red *USB*. Después el *SO* carga los *drivers* del dispositivo, solicitándolos al usuario si es necesario. Cuando se desconecta el dispositivo, el *host* lo detecta y descarga los *drivers*. El *host* USB tiene, entre otras, las siguientes funciones:

- Detectar la conexión de nuevos dispositivos al sistema.
- Detectar la desconexión de dispositivos previamente conectados,
- Enumerar y configurar los dispositivos conectados al sistema.
- Administrar y controlar el flujo de datos entre el *host* y los dispositivos *USB*.
- Administrar y controlar las transferencias síncronas y asíncronas de información.
- Recoger y resumir estadísticas de actividad y estado de los elementos del sistema.
- Proporcionar energía eléctrica a algunos dispositivos (teclado, ratón, etc.).

Un puerto serie es capaz de transmitir hasta 112,5 KB/s y un puerto paralelo entre 600KB/s y 15MB/s, sin embargo la velocidad de transferencia de un puerto USB está entre 1,5MB/s y 12MB/s. El FIREWIRE (IEEE 1394), maneja transferencias entre 100MB/s y 400MB/s, que permite conectar hasta 63 dispositivos y un cable de 4.5 metros por dispositivo, permitiendo al igual que el USB la conexión en operación. Uno de los problemas del puerto USB es que suministra solamente 500 miliamperios de corriente para los dispositivos conectados, que aunque es suficiente para la mayoría de los dispositivos que se conectan a este puerto, resulta pequeña cuando conectamos varios dispositivos sin fuente de alimentación propia.

Otra de las funciones importantes de los *hubs* es la de aislar a los puertos de baja velocidad de las transferencias de alta velocidad, proceso sin el cual todos los dispositivos de baja velocidad conectados al bus entrarían en colapso. La protección de los dispositivos lentos de los rápidos ha sido siempre un problema importante en el diseño redes mixtas, como es USB. El *hub* está compuesto por dos unidades principales: el *Controlador* y el *Repetidor*.

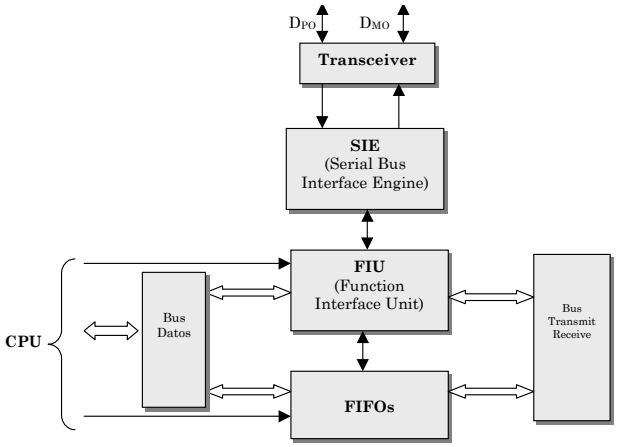
El *Repetidor* tiene la función de analizar, corregir y retransmitir la información que llega al *hub* hacia los puertos del mismo. Contiene una memoria de varios registros para sus funciones.

El *Controlador* es una pequeña CPU de supervisión de las múltiples funciones realiza un *hub*.

Todos los dispositivos conectados acceden al canal o medio para transmitir sus datos siguiendo un protocolo análogo al *token ring,* que consecutivamente va dando la posibilidad de transmisión a cada periférico.

5.2.1. Arquitectura

En la siguiente figura aparece un diagrama de bloques de un microcontrolador USB:



- **Transceiver** El cable USB dispone de solo cuatro hilos: Vbus, D+, D- y GND. La información y los datos van por los hilos D+ y D-, con dos velocidades: 12Mbps o 1.5Mbps. El *Transceiver*, incorporado dentro del chip controlador, constituye la interfaz de un dispositivo externo con el resto del sistema.
- **SIE (Serial Interface Engine)** Tiene la función de *serializar* y *paralelizar* las transmisiones, además maneja los protocolos de comunicación, las secuencias de paquetes, el control CRC y la codificación NRZI.
- **FIU (Function Interface Unit)** Administra los datos que son transmitidos y recibidos por el cable USB. Se basa y apoya en el contenido y estado de las memorias FIFOs. Controla los estados de las transacciones, los buffer FIFO, y solicita atención para diversas acciones a través de interrupciones.
- **FIFOs** Los controladores típicos disponen de un total de 8 buffer tipo FIFO, cuatro destinadas a la transmisión y cuatro a la recepción de datos. Tanto para la transmisión como para la recepción, los buffer soportan cuatro tareas o funciones, numeradas de 0 a 3.

5.2.2. Protocolo

Toda transferencia de datos o transacción que emplee el bus, involucra al menos tres paquetes de datos. Cada transacción se da cuando el *host* decide qué dispositivo hará uso del bus. Para ello envía un paquete al dispositivo específico. Cada dispositivo tiene un número de identificación, asignado por el *Controlador* de *host* cuando arranca el computador o cuando un se conecta un dispositivo nuevo al sistema. De esta forma, cada periférico puede determinar si es el receptor de un paquete de datos. Técnicamente un paquete de datos se denomina *Token Packet*.

Este protocolo tiene un sistema muy eficiente de recuperación de errores, empleando uno de los modelos más seguros como es el CRC (Código de Redundancia Cíclica). Y puede estar implementado al nivel de software y/o hardware de manera configurable.

5.2.3. Modos de transmisión

- **Asíncrona**
- **Síncrona**
- **Bulk**

La transmisión *Bulk*, es una comunicación no periódica, utilizada por transferencias que requieren todo el ancho de banda disponible o en su defecto son retrasadas hasta que el ancho de banda completo esté disponible. Aparecen en el movimiento de imágenes o vídeo, donde se requiere una gran capacidad de transferencia en poco tiempo.

USB permite dos tipos más de transferencias de datos:

- **Control**

Es un tipo de comunicación exclusivamente entre el *host* y el dispositivo que permite configurar este último. Sus paquetes de datos son de 8, 16, 32 o 64 bytes, dependiendo de la velocidad del dispositivo que se pretende controlar.

- **Interrupción**

Este tipo de comunicación está disponible para aquellos dispositivos que demandan mover muy poca información y con poca frecuencia. Su paquete de datos tiene las mismas dimensiones que el de las transmisiones de control.

5.3. Bus SCSI (Small Computer System Iterface)

El bus SCSI se utiliza en ordenadores personales y en muchas estaciones de trabajo. Se trata de una interfaz paralela, con 8, 16 o 32 líneas de datos. Cada dispositivo SCSI tiene dos conectores, uno de entrada y otro de salida, conectándose en cadena (*daisy chain*). Todos los dispositivos funcionan independientemente, y pueden intercambiar datos entre ellos, igual que con el computador. Por ejemplo, un disco duro puede guardar su contenido en una cinta sin que tenga que intervenir el procesador. Los datos se transfieren mediante paquetes, que componen un mensaje.

- Versiones

La especificación original SCSI, la SCSI-1, se desarrolló en los años 80. SCSI-1 utiliza 8 líneas de datos y opera a una frecuencia de reloj de 5 MHz, o a una velocidad de datos de 5 MBytes/s. SCSI-1 permite conectar al computador hasta siete dispositivos. En 1991 se introdujo SCSI-2. Los cambios más notables fueron la expansión opcional a 16 o 32 líneas de datos y el aumento de la frecuencia de reloj a 10 MHz. Como resultado se tiene una velocidad de datos máxima de 20 o 40 Mbytes/s.

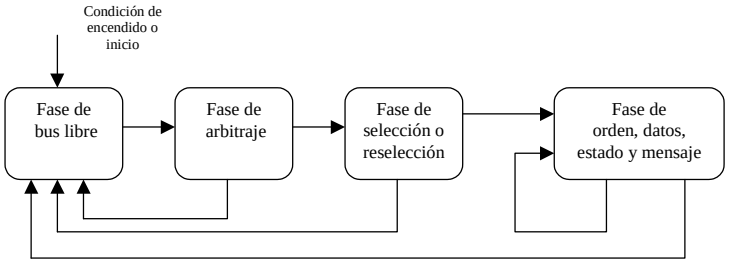
- Fases

Todos los intercambios en el bus SCSI se producen entre un dispositivo *iniciador* y un dispositivo *seleccionado*. Normalmente, el computador es el iniciador y un controlador periférico el dispositivo seleccionado, pero algunos dispositivos pueden asumir los dos papeles. En cualquier caso, toda la actividad del bus se produce en una secuencia de fases:

- **Bus Libre:** ningún dispositivo está utilizando el bus, está disponible.
- **Arbitraje:** determina el dispositivo que toma control del bus, de manera que pueda iniciar o reanudar una operación de E/S.
- **Selección:** de un dispositivo para realizar una operación, por ejemplo, una orden de lectura o escritura.
- **Reselección:** permite que el dispositivo seleccionado se vuelva a conectar al iniciador para reanudar una operación iniciada previamente, pero suspendida por el dispositivo.
- **Orden:** el dispositivo solicita una orden de información al iniciador.

- **Datos:** el dispositivo solicita la transferencia de un dato hacia el iniciador (entrada de datos, Data In) o viceversa (salida de datos, Data Out).
- **Estado:** el dispositivo solicita la información de estado desde al iniciador.
- **Mensaje:** el dispositivo solicita la transferencia de uno o más mensajes desde el iniciador (entrada de mensaje, Message In) o viceversa (salida de mensaje, Message Out).

La siguiente figura ilustra el orden en que se producen las fases del bus SCSI:



Después de conectarse el sistema, o después de un reinicio, el bus pasa a la fase de *bus libre*. A ésta le sigue la *fase de arbitraje*, que da lugar a que un dispositivo tome el control. Si el arbitraje falla, el bus vuelve a la fase de bus libre. Cuando el arbitraje termina con éxito, el bus pasa a una *fase de selección o reselección*, en la que se designa un iniciador y un dispositivo seleccionado para realizar la transferencia. Después de determinados los dos dispositivos, se producirá una o más *fases de transferencia de información* entre ambos (*fases de orden*, *datos*, *estado* y *mensaje*). La fase final de transferencia de información es normalmente la fase de entrada de mensaje, en la que un mensaje de desconexión o de orden completa se transfiere al iniciador, seguida de una fase de bus libre. Una característica importante del bus SCSI es la capacidad de reselección. Si una orden enviada necesita algún tiempo para completarse, el dispositivo seleccionado puede liberar el bus y volverse a conectar al iniciador más tarde. Por ejemplo, el computador puede enviar una orden a un dispositivo de disco para que dé formato al disco, y el dispositivo realiza la operación sin necesidad de acceder al bus.

5.3.1. Líneas de señal

BSY: Bus ocupado, activada por el iniciador o dispositivo seleccionado

SEL: Selección de un dispositivo por el iniciador o del iniciador por un dispositivo

C/D: Indicación por el dispositivo si el bus de datos lleva información de control (orden, estado o

mensaje) o datos

I/O: Control por el dispositivo de la dirección del movimiento de datos en el bus de datos.

REQ: Petición por el dispositivo de una transferencia de información o datos.

ACK: Reconocimiento por el iniciador de la señal de REQ del dispositivo.

DB(7-0): Bus de datos

5.3.2. Operación (temporización)

Para aclarar el papel de cada señal en las distintas fases (estados) del bus, analizaremos una operación de lectura que transfiere datos desde un dispositivo periférico al iniciador (computador). Esta sigue las siguientes fases:

Arbitraje	Selección	Orden	Transferencia de datos	Estado	Mensaje

El **arbitraje** se realiza con la señal BSY y las 8 señales de datos, la 7 asignada al computador, y

las otras 7 asignadas a otros tantos dispositivos (7 es el número máximo de dispositivos que se pueden conectar al SCSI-1). El dispositivo que quiere acceder al bus activa BSY y su señal de datos asociada. Entre las señales de datos existe una prioridad, siendo la 7 la más prioritaria y la 0 la menos prioritaria. Cada dispositivo conoce si puede acceder al bus leyendo las líneas de datos. Si no hay otra activa con mayor prioridad accederá al bus. En nuestro caso es la CPU quien accede (máxima prioridad) para realizar la lectura.

El dispositivo que ha ganado el arbitraje se convierte en el iniciador y comienza la **fase de selección** del dispositivo con el que quiere realizar la transacción. Para eso activa la señal SEL y las dos líneas de datos: la suya propia y la del dispositivo con el que quiere conectar. El dispositivo reconoce su identificación activando la señal BSY que ha desactivado previamente el iniciador después de su uso en el arbitraje.

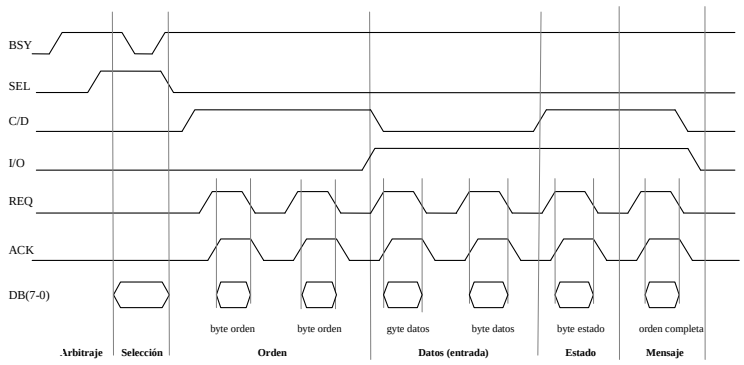
Después se pasa a **la fase de orden** en la que el iniciador transmite al dispositivo en forma asíncrona el conjunto de bytes que componen la orden a ejecutar (lectura en nuestro caso). El sincronismo se realiza con la ayuda de las señales conocidas REQ (controlada por el dispositivo y leída por el iniciador) y ACK (controlada por el iniciador y leída por el dispositivo).

Una vez que el dispositivo conoce que es una orden de lectura se pasa a la **fase de transmisión de datos** que de nuevo se realiza byte a byte de forma asíncrona con la ayuda de REQ y ACK..

Finalizada la transmisión el dispositivo pasa el bus a la **fase de estado** en la que transmite un byte en el que se codifica que la transmisión se ha realizado con éxito.

Finalmente el dispositivo hace pasar al bus a la **fase de mensaje** en la que se transmite un byte con el mensaje de orden completa.

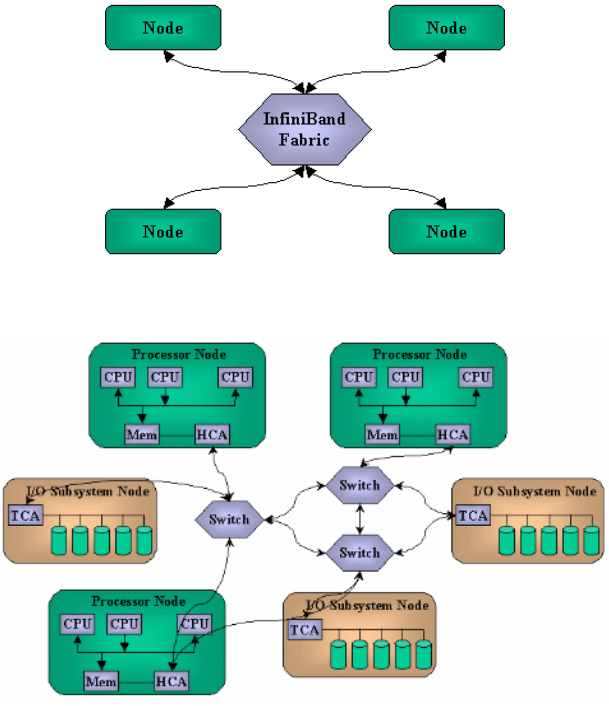
El diagrama de tiempos para la orden de lectura con las principales señales implicadas sería el siguiente:



5.3.3. Mensajes

- Orden completa
- Desconexión
- Error detectado en el iniciador
- Abortar
- Transferencia síncrona de datos

5.4. Arquitectura del standard InfiniBand


Node
Node
InfiniBand
Fabric
Node
Node
Processor Node
CPU
CPU
CPU
Mem
HCA
Processor Node
CPU
CPU
CPU
Mem
HCA
I/O Subsystem Node
TCA
Switch
Switch
Switch
I/O Subsystem Node
TCA
Processor Node
CPU
CPU
CPU
Mem
HCA
I/O Subsystem Node
TCA