Módulo 2. Entrada/salida
Tema 2.2. Sistema de interconexión y entrada/salida mediante DMA

Indice

- Sistema de interconexión
  - Introducción
  - Taxonomía de las líneas de comunicación
  - Buses. Organización. Transferencia de datos. Ejemplo: PCI.
  - Interconexiones punto a punto. Ejemplos: PCI express. QPI.
- Entrada/salida por Acceso Directo a Memoria
  - Necesidad del DMA
  - Controlador de DMA
  - Transferencia DMA. Tipos de transferencias.

BIBLIOGRAFÍA BÁSICA:

• W. Stallings; Computer Organization and Architecture, 9ed. Prentice Hall 2013.

•D. A. Patterson & J. L. Hennessy; Estructura y diseño de computadores. La interfaz hardware/software, 4ed. Reverté 2011.

•B. Parhami; Computer Architecture: From Microprocessors to Supercomputers, Oxford University Press, 2005.

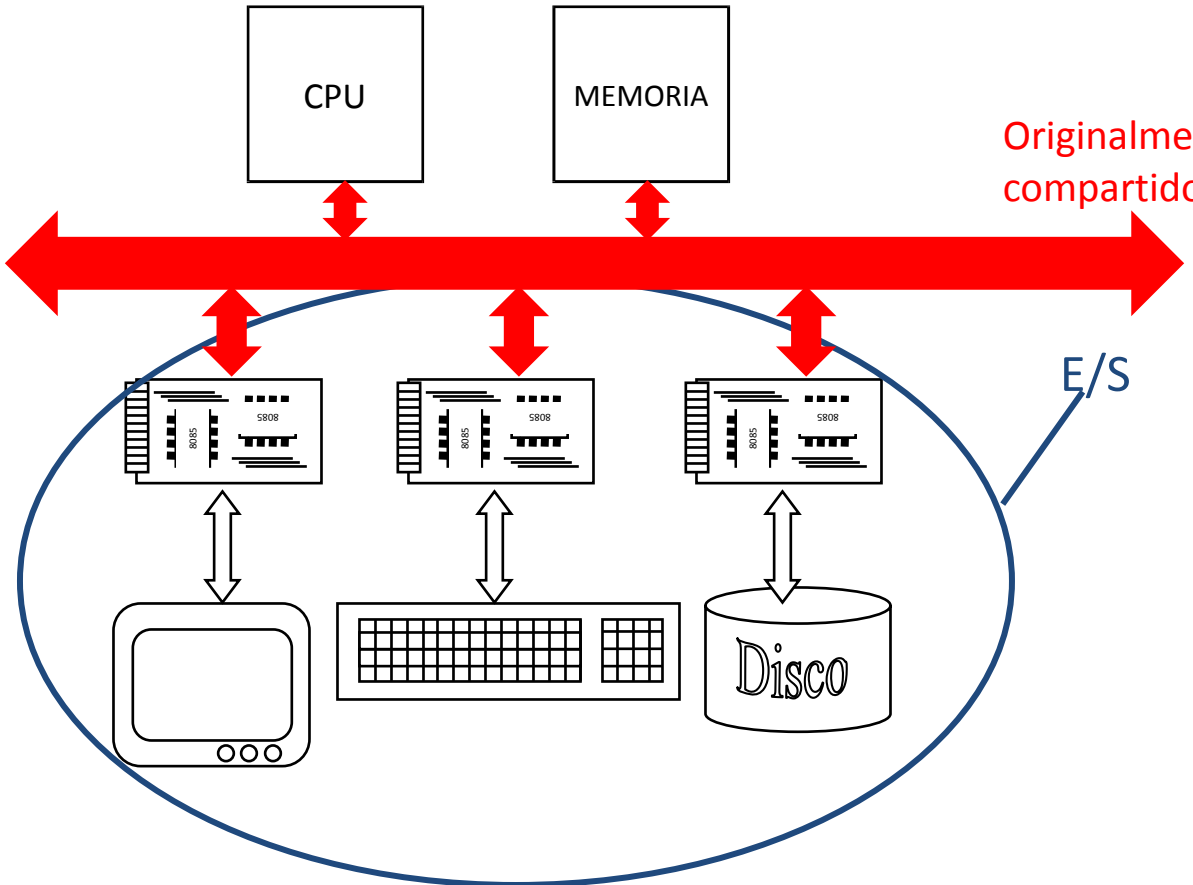
•S. Furber; ARM System-on-Chip architecture, 2ed. Addison-Wesley 2000.

Tema 2.2. Sistema de interconexión y entrada/salida mediante DMA

Sistema de interconexión

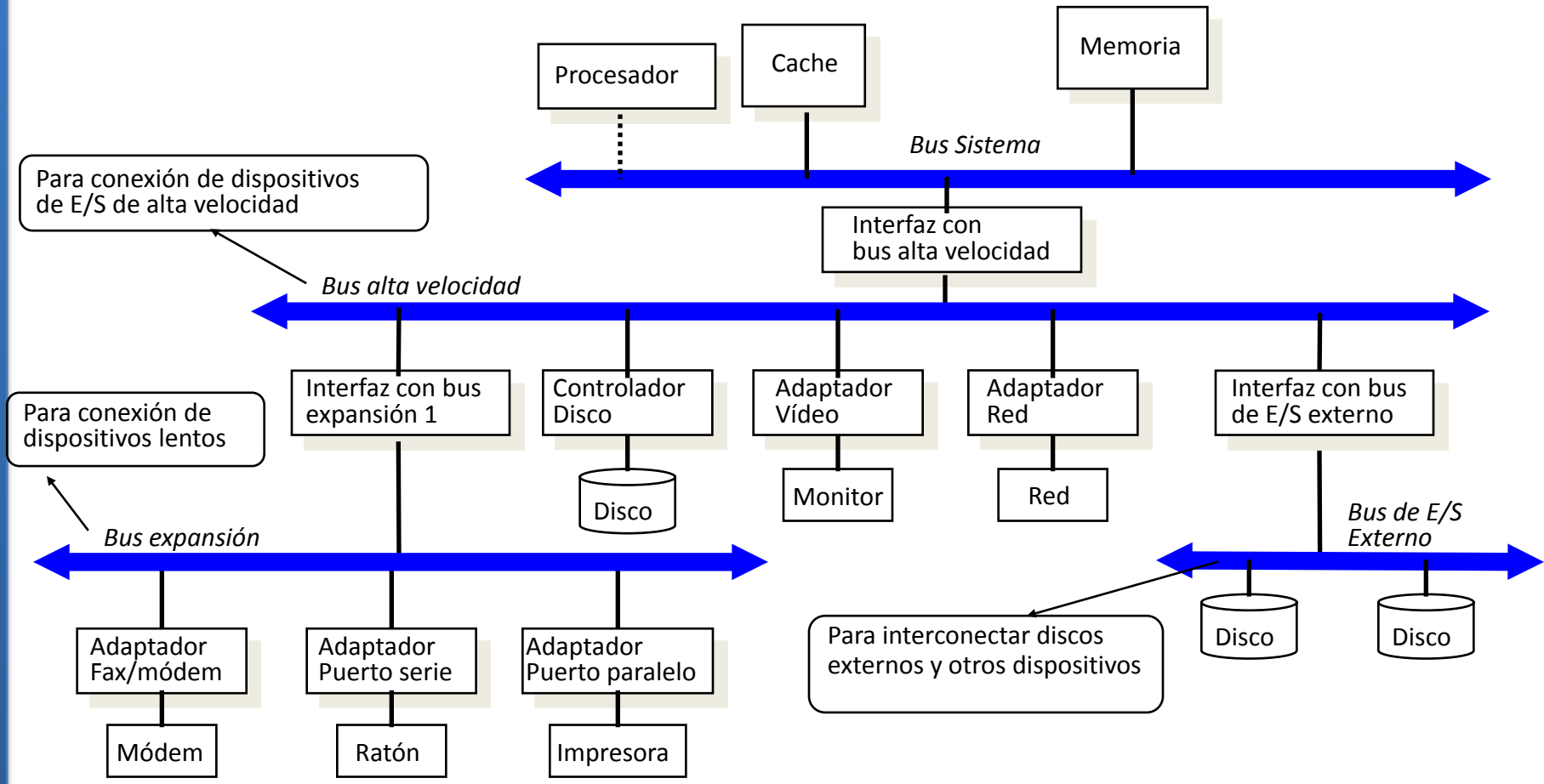
Sistema de interconexión

- Es necesario comunicar CPU con el sistema de memoria y los dispositivos de E/S



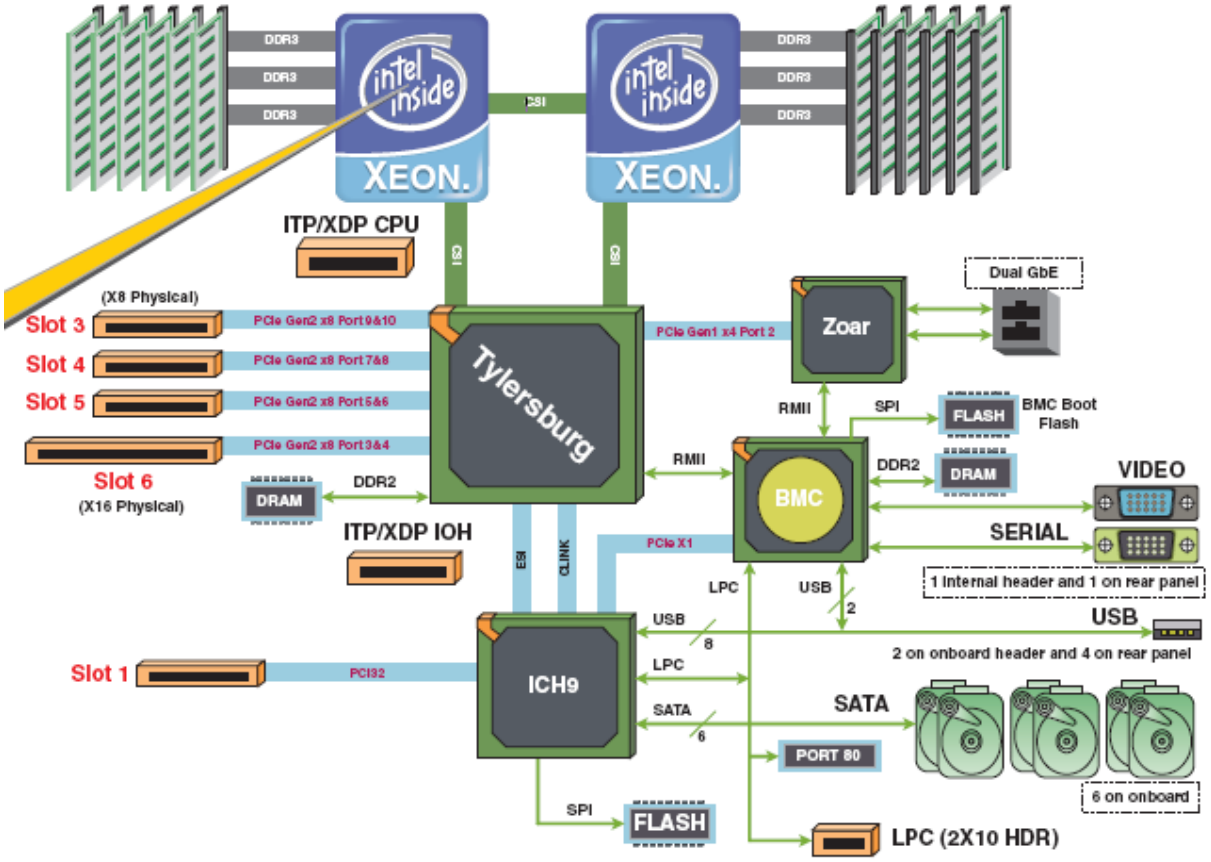
Sistema de interconexión (2)

- El bus no era capaz de conectar tantos dispositivos de velocidades tan distintas. Se usa una Jerarquía de buses.

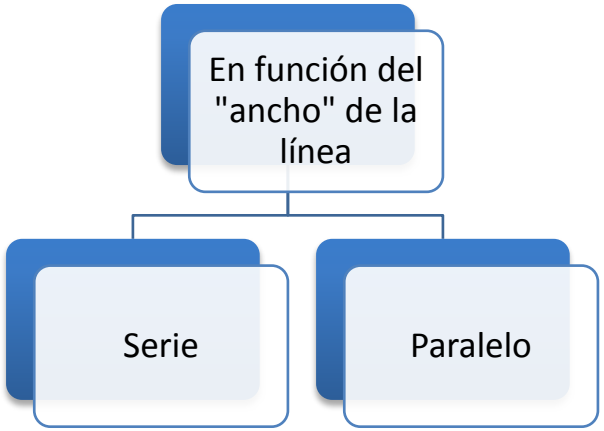


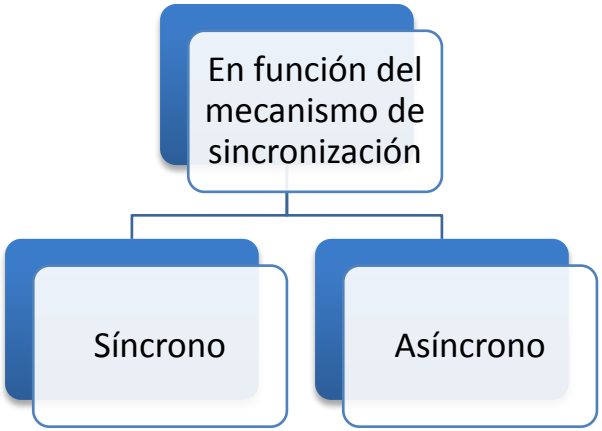
Sistema de interconexión (3)

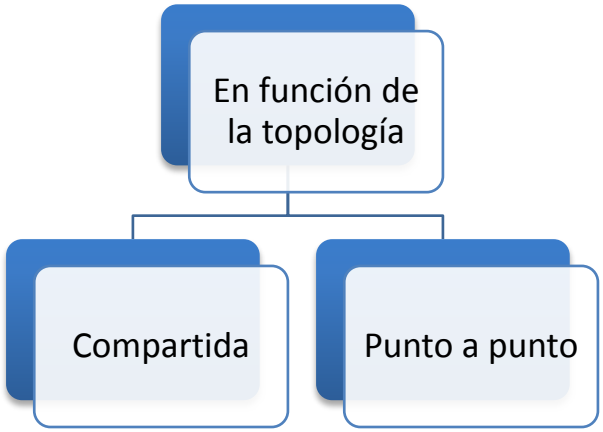
- Actualmente es un sistema complejo, con conexiones punto a punto (QPI, PCIe) y otras conexiones tipo bus.



Taxonomía de líneas de comunicación

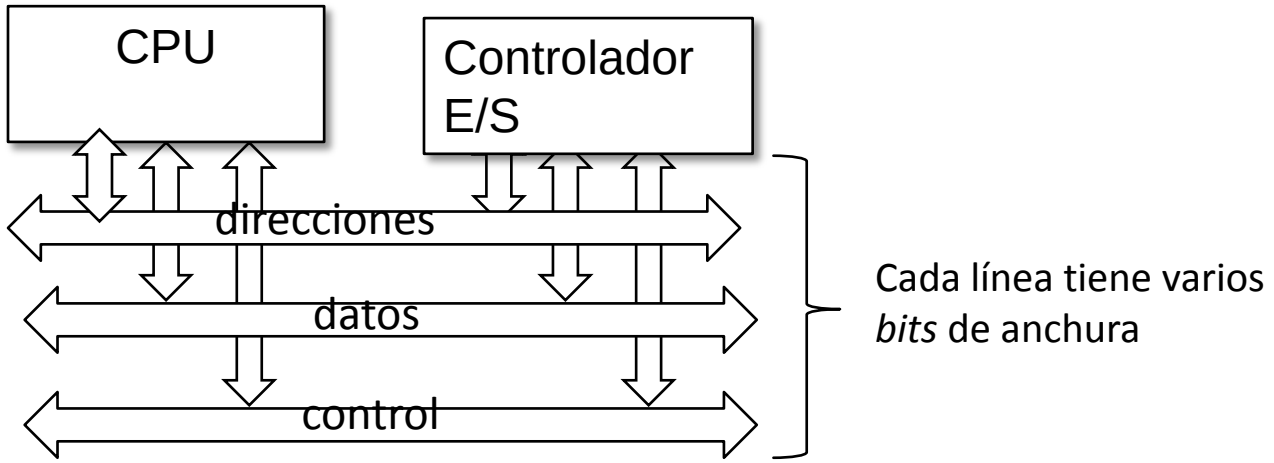






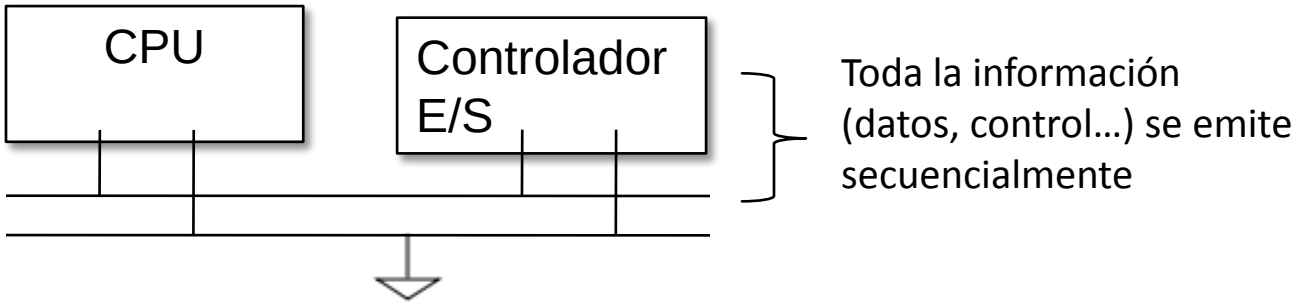
Transmisión paralela

- Utiliza varias líneas de comunicación (cables) a través de las cuales se envían varios **bits de información de forma simultánea**
  - ✓ Ancho de banda elevado (en teoría…)
  - ✗ Para conectar dispositivos a distancias medias o largas resulta muy costosa
  - ✗ Frecuencia de funcionamiento limitada por factores físicos
  - ✗ Los dispositivos de baja velocidad no aprovechan el potencial de la transmisión paralela (ratón, módem…)



Transmisión serie

- Utiliza una única línea de comunicación (2 cables: dato y tierra) a través de las cual se se envían los bits de información de **forma secuencial**
  - ✓ Es menos costosa que la E/S paralela
  - ✓ Permite frecuencias de funcionamiento mayores
  - ✗ A igualdad de frecuencia que el caso paralelo, el ancho de banda es menor



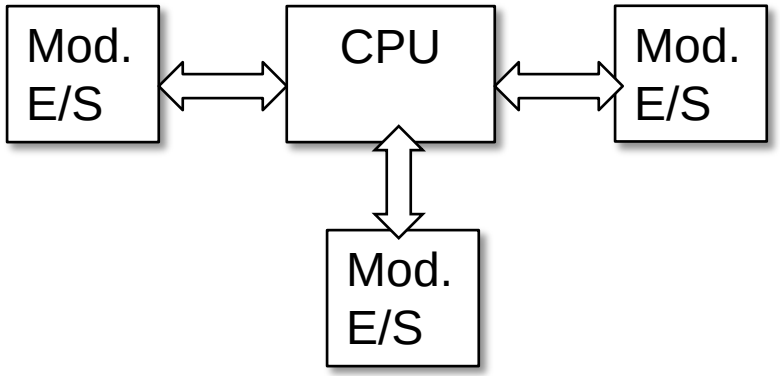
Es posible aumentar el ancho de banda entre dos dispositivos usando varias conexiones serie

Comunicación síncrona vs asíncrona

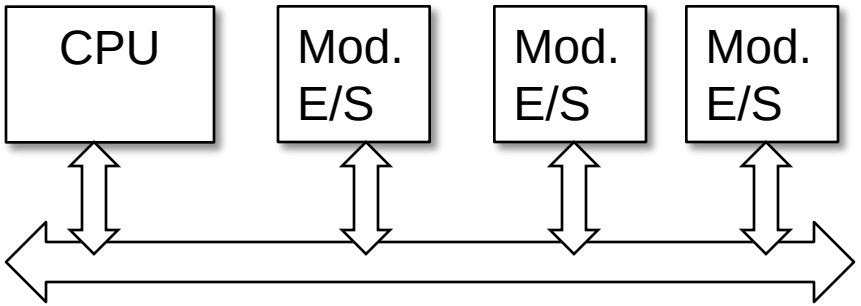
- ¿Cuándo empieza/acaba el envío de un dato?
- Asíncrona
  - La señal de reloj NO se envía por la línea de comunicación
    - Emisor y receptor utilizan sus propias señales de reloj.
    - Es necesario establecer un **protocolo** para la sincronización entre ambos
  - ✓ No es imprescindible que emisor y receptor funcionen a la misma frecuencia
- Síncrona
  - La señal de reloj viaja con el resto de señales
  - ✓ Más sencillo y, en general, más eficiente
  - ✗ Exige que emisor y receptor funcionen a la misma frecuencia
  - ✗ Debe ser corto si queremos que sea rápido (para que no le afecte el clock skew)

Punto a punto vs. Compartida

- Comunicación Punto a punto
  - ✓ Gran rendimiento
  - ✗ Exige un gran número de interfaces

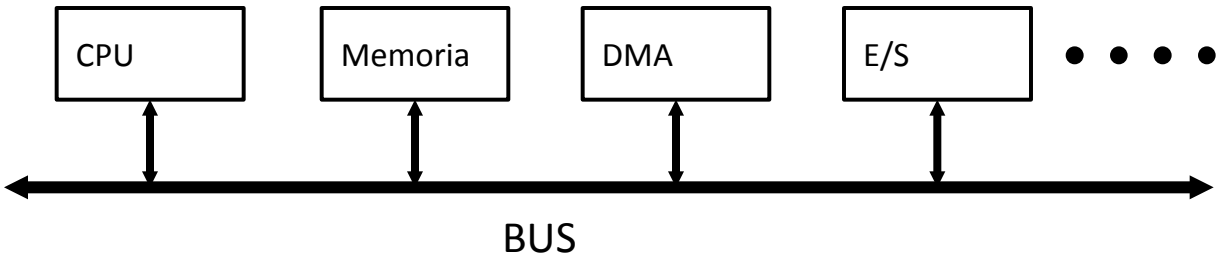


- Compartida
  - ✓ Más versátil y barato
  - ✗ Cuello de botella en la comunicación
  - ✗ Exige sincronización entre dispositivos para evitar escrituras simultáneas



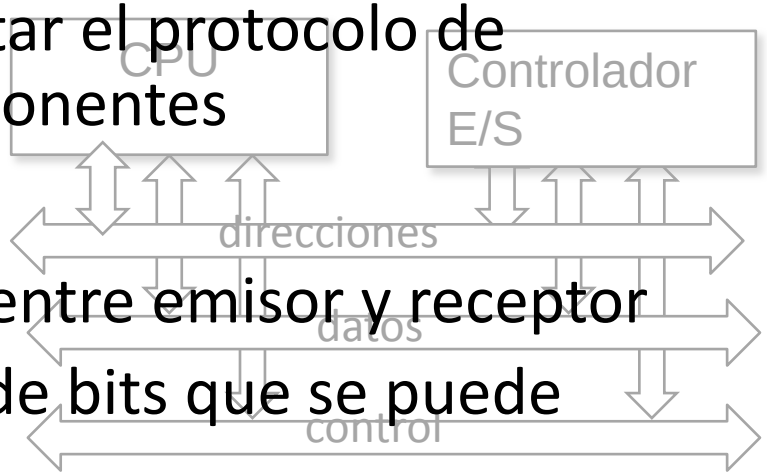
Comunicación tradicional en el computador: buses

- Conjunto de *cables* que conectan múltiples componentes:
  - Paralelo
  - Compartido
    - Es necesario **arbitrar** el uso del bus
  - Hay versiones síncronas y asíncronas



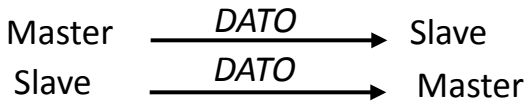
Organización general de un bus

- Líneas de control
  - Utilizadas para implementar el protocolo de comunicación entre componentes
- Líneas de datos
  - Transmite la información entre emisor y receptor
  - Anchura de bus: número de bits que se puede transmitir
- Líneas de dirección
  - Contienen la información de direccionamiento
  - Habitualmente, coinciden físicamente con las de datos



Transferencia de datos en un bus

- Tipos básicos de transferencia:
  - Escritura:
  - Lectura:

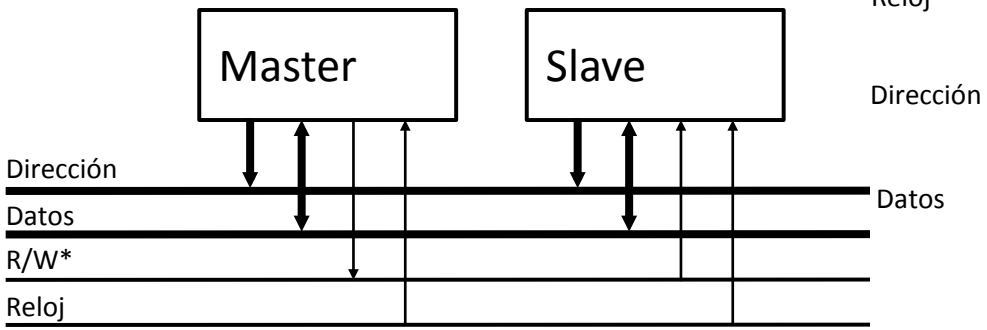


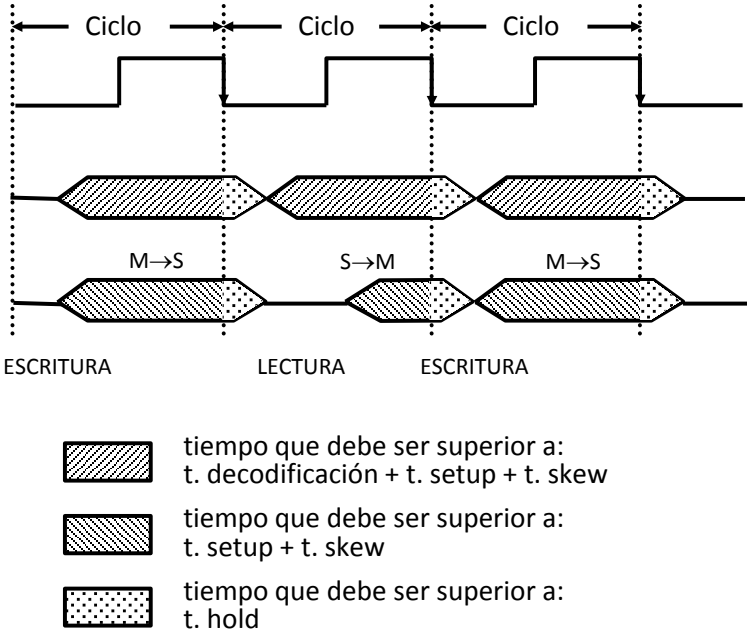
- Fases de una transferencia
  - 1. Direccionamiento (identificación) del *slave*
  - 2. Especificación del tipo de operación (lectura o escritura)
  - 3. Transferencia del dato
  - 4. Finalización del ciclo de bus

- Control de la transferencia:
  - Sincronización: determinar el inicio y el final de cada transferencia
  - Arbitraje: controlar del acceso al bus en caso de varios masters

Sincronización: protocolo síncrono

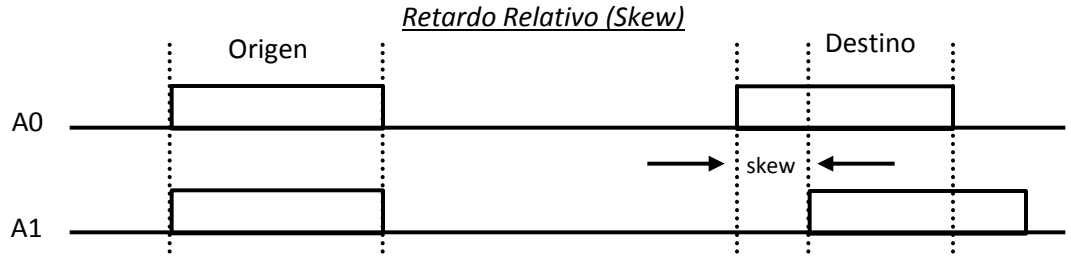
- Cada transferencia se realiza en un número fijo de periodos de reloj (1 en el ejemplo)
- Los **flancos del reloj** (de bajada en el ejemplo) determinan el comienzo de un nuevo ciclo de bus y el final del ciclo anterior





Protocolo síncrono

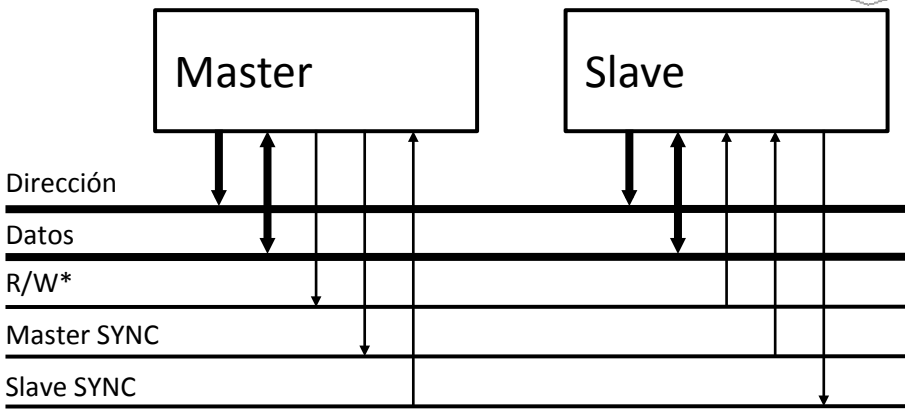
- *Temporización*:
  - **Tiempo de decodificación (*decode time*):** tiempo necesario para que slave decodifique la dirección
  - **Tiempo de estabilización (*setup time*):** tiempo antes del flanco de reloj que deben permanecer estables las señales para asegurar su correcto almacenamiento
  - **Tiempo de permanencia (*hold time*):** tiempo después del flanco de reloj que deben permanecer estables las señales para asegurar su correcto almacenamiento
  - **Tiempo de desplazamiento relativo de las señales (*skew time*):** diferencia de tiempo entre la llegadas al receptor de dos señales que partieron del emisor simultáneamente

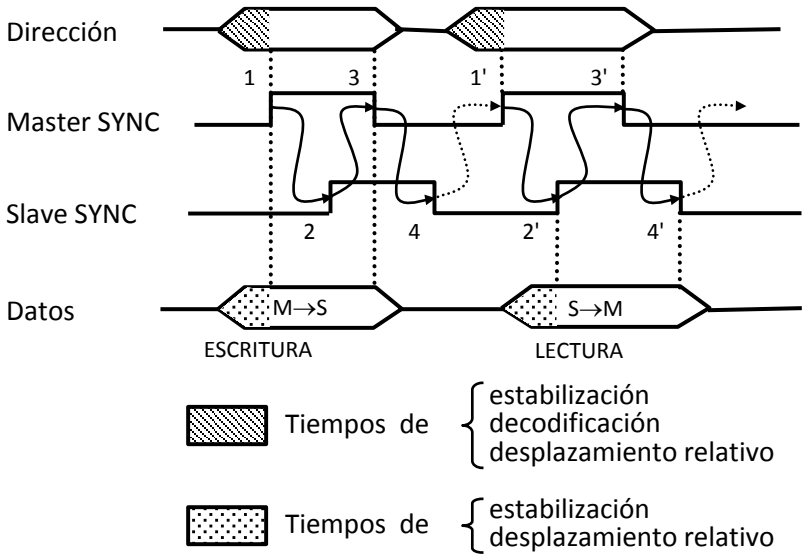


- *Ventajas*:
  - Simplicidad (de diseño y de uso)
  - Sólo se necesita una señal (reloj) para llevar a cabo la sincronización
  - Mayor velocidad (en relación a protocolo asíncrono)
- *Desventajas*:
  - El periodo de reloj se tiene que adaptar a la velocidad del dispositivo más lento (por lo que suele usarse para conectar dispositivos homogéneos)
  - No existe confirmación de la recepción de los datos
  - La necesidad de distribuir la señal de reloj limita la longitud del bus

Sincronización: protocolo asíncrono

- No existe señal de reloj
- Los dispositivos implicados en la transmisión fijan el comienzo y el final de la misma mediante el intercambio de señales de control (**handshake**)
- Se utilizan 2 señales de sincronización:
  - Master SYNC (procedente del master)
  - Slave SYNC (procedente del slave)
- Protocolo completamente interbloqueado:
  - A cada flanco de master le sigue uno del slave
- **Ciclo de escritura**:
  1: (M a S) Hay un dato en el bus
  2: (S a M) He tomado el dato
  3: (M a S) Veo que lo has tomado
  4: (S a M) Veo que lo has visto (Bus libre)
- ***Ciclo de lectura***:
  1': (M a S) Quiero un dato
  2': (S a M) El dato está en el bus
  3': (M a S) He tomado el dato
  4': (S a M) Veo que lo has tomado (Bus libre)





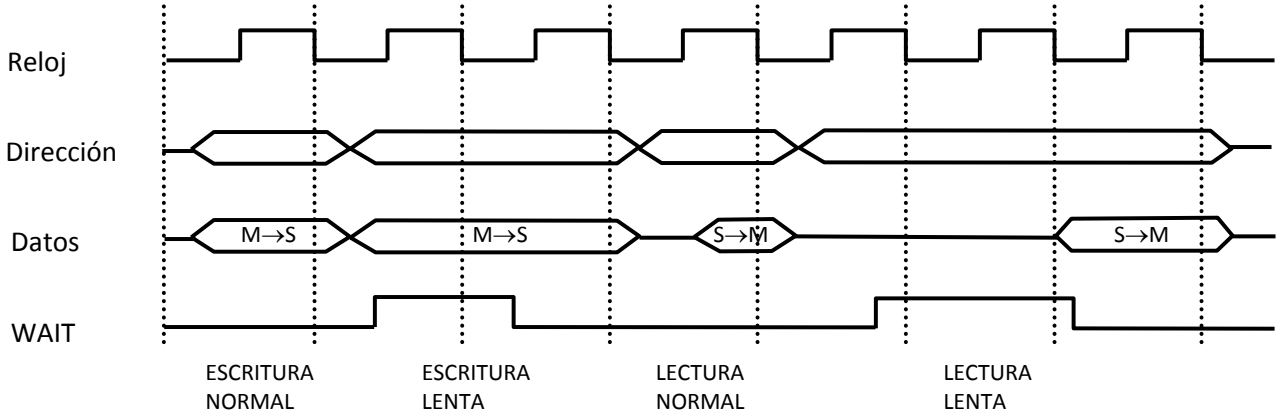
Sincronización: protocolo asíncrono

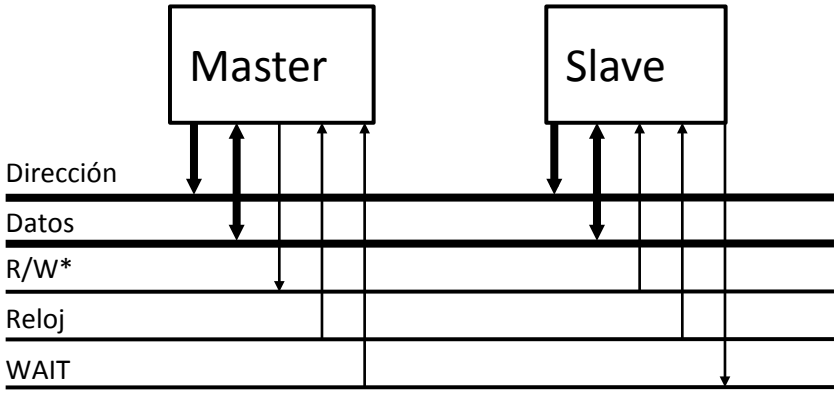
- *Ventajas:*
  - Facilidad para conectar elementos de diferentes velocidades
  - Fiabilidad: la recepción siempre se confirma.
- *Desventajas:*
  - El intercambio de señales de control introduce retardos adicionales
  - A igualdad de velocidades de los dispositivos, menos eficiente que el síncrono

Sincronización: protocolo semisíncrono

- Las transferencias se rigen por una **única señal de reloj**
- Cada transferencia puede ocupar uno o varios periodos de reloj
- Señal de WAIT (puede activarla cualquier Slave si no es capaz de realizar la transf. en 1 solo ciclo)
  - **Dispositivos rápidos**: operan como en un bus síncrono (una transferencia por ciclo)
  - **Dispositivos lentos**: activan la señal de WAIT y congelan actuación del Master (una transferencia puede ocupar varios ciclos)

Sincronización: protocolo semisíncrono



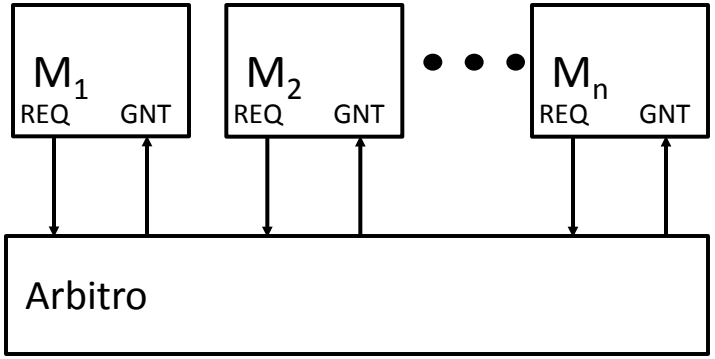


Conflictos en el acceso: arbitraje

- Si puede haber varios emisores (Masters), es necesario que el acceso al bus sea *libre de conflictos*
  - Se debe evitar que dos módulos escriban simultáneamente en el bus
  - Cada dispositivo solicita permiso para poder tomar el control del bus
- Protocolos centralizados: existe un **árbitro** del bus o **master principal** que controla el acceso al bus
  - En estrella
  - Daisy-chain de 2,3 y 4 hilos
- Protocolos distribuidos: el control de acceso al bus se lleva a cabo entre todos los posibles masters de una forma cooperante
  - Protocolo de líneas de identificación
  - Protocolo de códigos de identificación

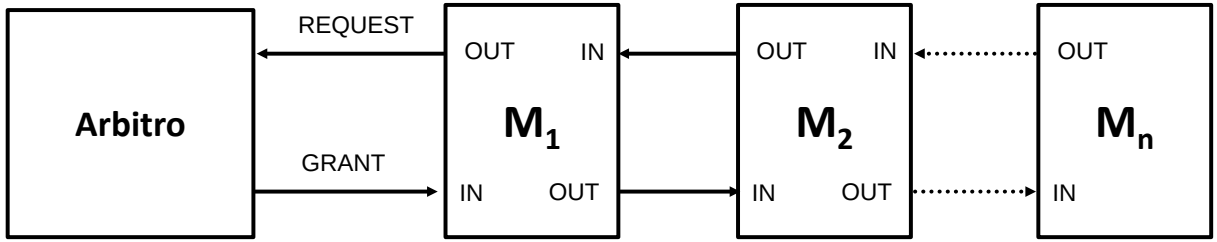
Protocolos centralizados: en estrella

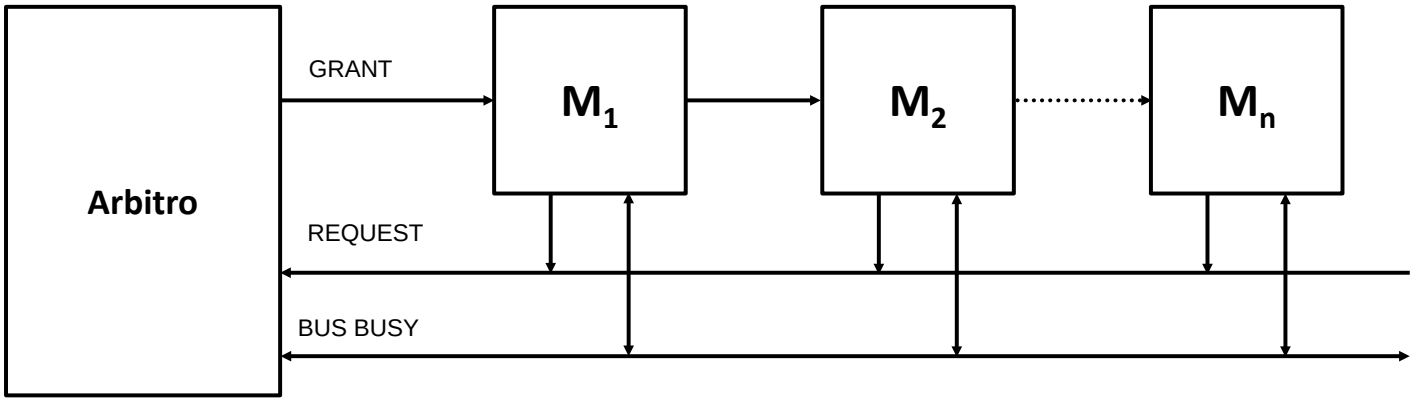
- Cada master se conecta al árbitro mediante dos líneas individuales:
  - BUS REQUEST (REQ): línea de petición del bus
  - BUS GRANT (GNT): línea de concesión del bus
- Varias peticiones de bus pendientes: el árbitro puede aplicar distintos algoritmos de decisión
  - FIFO
  - Prioridad fija
  - Prioridad variable



- Ventajas:
  - Algoritmos de arbitraje simples
  - Pocos retardos de propagación de las señales (en comparación con protocolos daisy-chain)
- Desventajas:
  - Número elevado de líneas de arbitraje en el bus (dos por cada posible master)
  - Número de masters alternativos limitado por el número de líneas de arbitraje
- Ejemplo: PCI

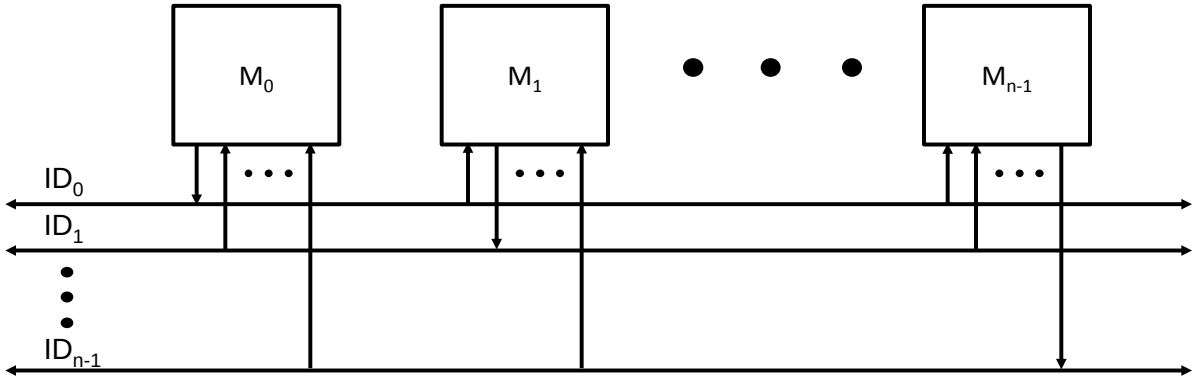
Protocolos centralizados: Daisy-chain (2-3 hilos)



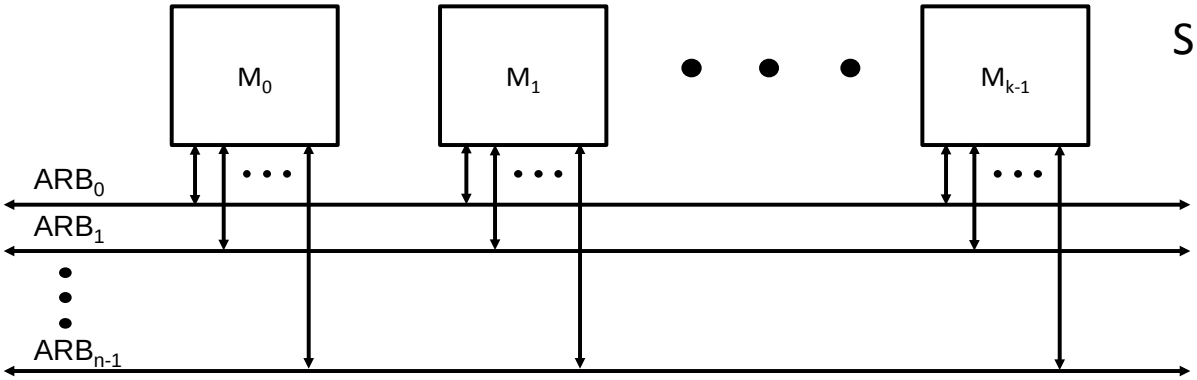


Protocolos distribuidos

Líneas de identificación



Códigos de identificación



Siendo $k \leq 2^n$

Ejemplo de un bus: PCI

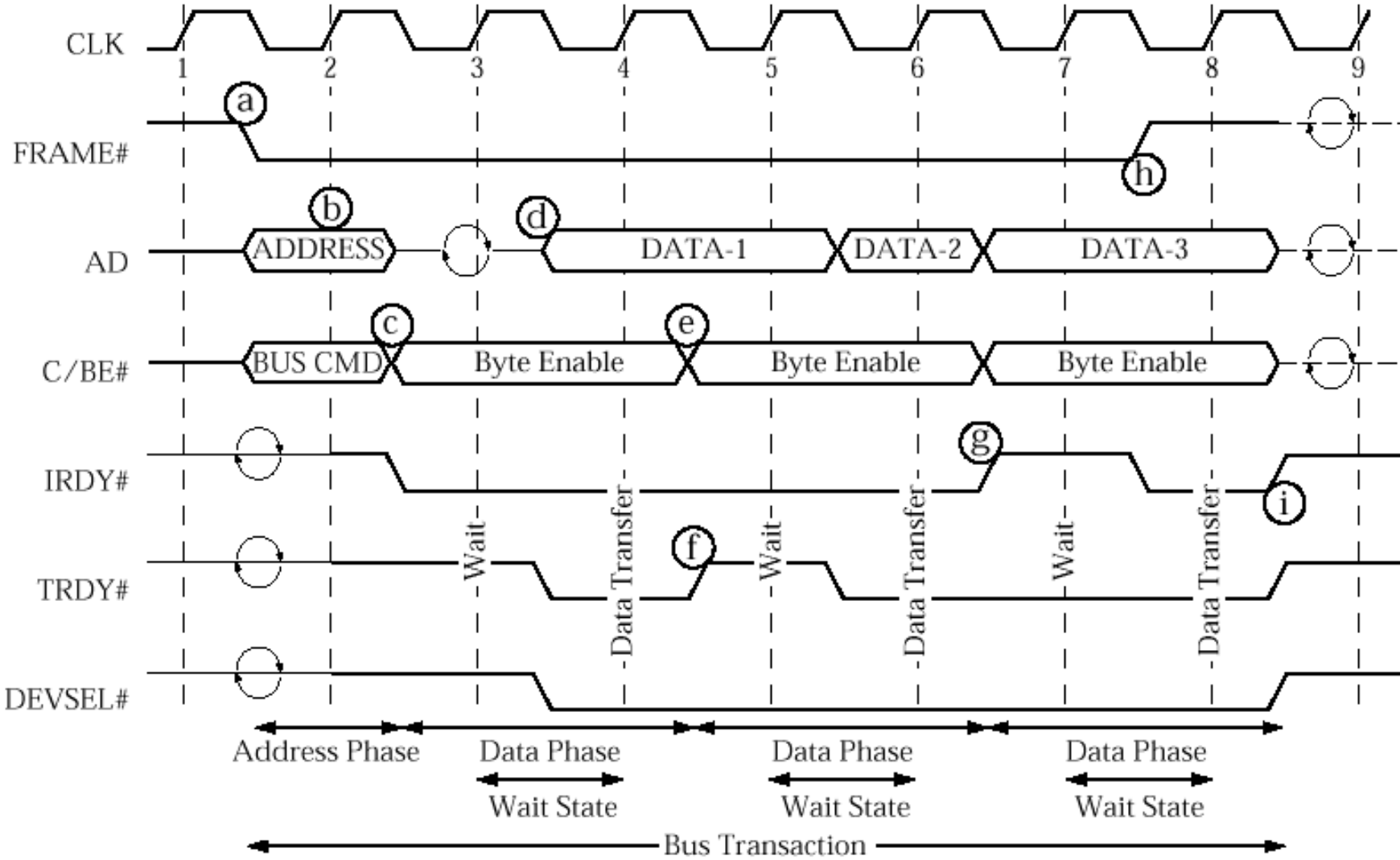
- Bus de expansión diseñado para los ix86 y Pentium (1993).
  - Actualmente se encuentra en todos los computadores personales aunque ya está en fase de extinción.
- Soporta hasta 10 periféricos de alta velocidad.
- Multiplexación de las líneas de datos y direcciones:
  - Bus de datos de 32 bits en la versión 2.0 y de 64 bits en la versión 2.1.
  - Bus de direcciones de 32 bits.
- Funciona a 33 MHz (versión 2.0) o a 66 MHz (versión 2.1).
  - La velocidad de transferencia máxima es de 132 MB/s o de 528 MB/s.
- Tipos de protocolos:
  - Protocolo de arbitraje: Centralizado en estrella.
  - Protocolo de sincronización: Semisíncrono, con dos modos de transferencia
    - Modo ráfaga: Se transfiere una única palabra (de 1, 2, 3 ó 4 bytes) a una dirección de memoria o de E/S.
    - Modo bloque: Se transfiere un bloque de datos desde/hacia posiciones de memoria consecutivas (especificando la posición inicial).

Ejemplo de un bus: PCI

Líneas del bus

- **CLK:** señal de reloj
- **AD0-AD31:** Líneas multiplexadas de datos y direcciones
- **C0*-C3*/BE0*-BE3*:** Líneas multiplexadas de *orden (command)* / *byte activo (byte enabled)*
  - **Orden (C0*-C3*):** la activa el master durante el primer ciclo de la transferencia para especificar el tipo de transferencia a realizar
    - ✓ Lectura de memoria, escritura de memoria, lectura de E/S, escritura de E/S, etc.
  - **Byte activo (BE0*-BE3*):** la activa el master durante la transferencia de datos para indicar qué líneas del bus transportan los datos
    - ✓ BE0* activada ⇒ AD0-AD7 transporta datos
    - ✓ BE1* activada ⇒ AD8-AD15 transporta datos
    - ✓ BE2* activada ⇒ AD16-AD23 transporta datos
    - ✓ BE3* activada ⇒ AD24-AD31 transporta datos
- **FRAME*:** Señal para indicar el comienzo y la duración de una transferencia
  - La activa el master al poner la dirección en el bus para indicar el comienzo de la transferencia
  - Si la transferencia es *modo bloque* la señal se mantiene activa durante toda la transferencia del bloque y se desactiva al transferir la última palabra
- **DEVSEL*:** Señal de *dispositivo seleccionado* (device selected)
  - La activa el slave para indicar que ha reconocido su dirección
- **TRDY*:** Señal de *slave preparado (target ready)*
  - La activa el slave al inicio de la transferencia junto con DEVSEL*
  - El slave desactiva esta señal en caso de que no pueda completar la transferencia en un solo ciclo de reloj
- **IRDY*:** Señal de *master preparado (initiator ready)*
  - La activa el master al inicio de la transferencia
  - El master desactiva esta señal en caso de que no pueda completar la transferencia en un solo ciclo de reloj
    - ✓ Por ejemplo, en caso de que el master se quede accidentalmente sin capacidad de almacenamiento

PCI: ejemplo de operación de lectura (3 bloques)



PCI: ejemplo de operación de lectura

a El master realiza las siguientes acciones
- Pone la dirección en el bus (AD0-AD31)
- Indica el tipo de operación a realizar (C0*-C3*)
- Activa FRAME* para indicar el inicio de la transferencia

b El slave descodifica y reconoce su dirección en el bus

c El master deja libre el bus de datos e indica en BE0*-BE3* qué líneas transportarán los datos y activa IRDY* para indicar que está preparado para recibir el 1er dato

d Cuando el slave tiene el 1er dato válido realiza las siguientes acciones
- Activa DEVSEL* para indicar ha reconocido su dirección
- Pone el dato en el bus y activa TRDY* para indicar que el dato está en el bus

e El master lee el dato
- A partir de aquí, mientras esté la señal FRAME* activada, se leerá un dato en cada ciclo de reloj (siempre que el slave no desactive TRDY*)

f El slave necesita más de 1 ciclo para poner el 2º dato en el bus
- Desactiva TRDY* hasta que tiene el nuevo dato preparado

g El master no está preparado para recibir el 3er dato
- Desactiva IRDY* hasta que está preparado para poder recibir correctamente el siguiente dato

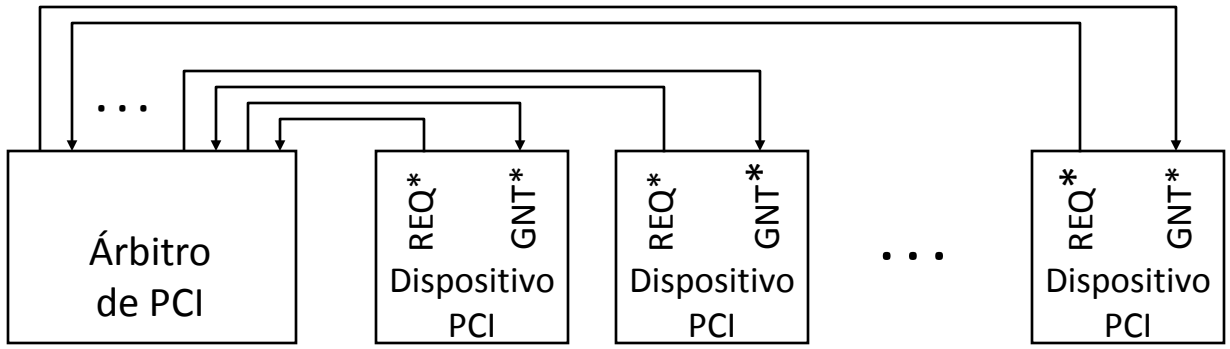
h Transferencia del último dato
- El master desactiva FRAME* para indicar el final de la transferencia del bloque

i El master desactiva IRDY* y el slave desactiva TRDY* y DEVSEL*
- El bus queda libre para la siguiente transferencia

PCI: arbitraje

- Conexión de cada dispositivo con el árbitro:
  - Línea de petición (REQ)
  - Línea de concesión (GRANT)
- PCI no especifica algoritmo de arbitraje. Se puede utilizar diferentes algoritmos: FIFO, round-robin, Prioridad fija,...

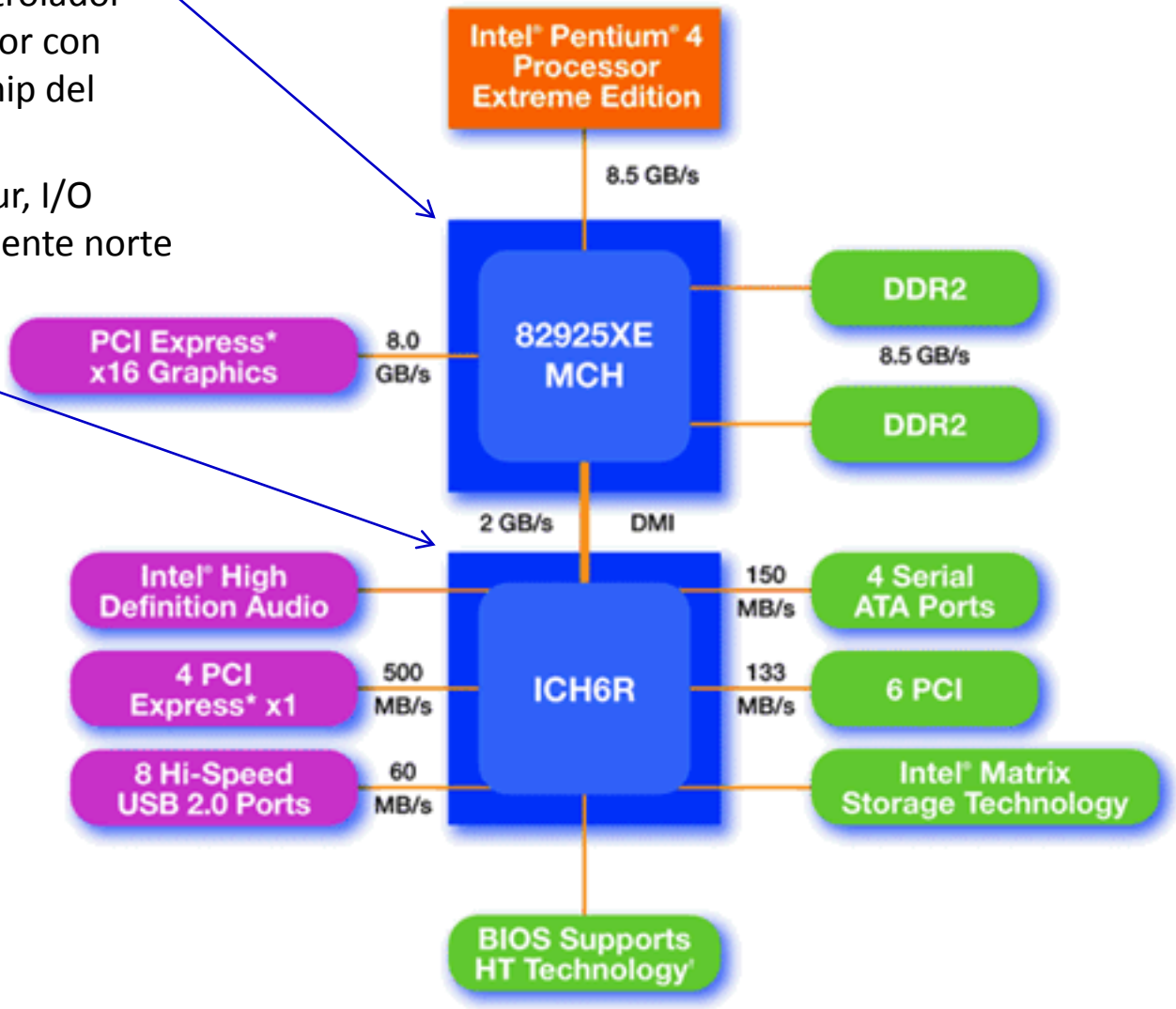
Líneas de arbitraje del bus PCI



Ejemplo: Chipsets de Intel. 2004

Controlador de memoria (puente norte, Memory Controller Hub): controlador DMA que conecta el procesador con memoria, el bus gráfico y el chip del puente sur

Controlador de E/S (puente sur, I/O Controller Hub): conecta el puente norte a los buses de E/S



Evolución de las interconexiones

- Los buses paralelos presentan limitaciones físicas:
  - ✘ A alta frecuencia, cada línea del bus tendrá diferencias de retardo significativas
  - ✘ La frecuencia de funcionamiento está muy limitada
  - ✘ El arbitraje introduce más latencia
  - ✘ Alto consumo energético
  - – Ejemplos: PCI, SCSI, IDE, VMEbus
- Resulta más ventajoso construir canales más estrechos pero de mayor velocidad
  - – **Comunicación serie, síncrona y punto a punto**
- El cambio de paradigma proporciona más flexibilidad:
  - ✓ Alto rendimiento en conexiones punto a punto
  - ✓ Permite múltiples transferencias en paralelo
  - ✓ Permite sintonizar el ancho de banda necesario para cada dispositivo con varios canales serie en paralelo
  - ✓ Bajo consumo energético
  - – PCI Express, Infiniband, Serial ATA (SATA), USB
- Actualmente, a medida que aumenta la velocidad necesaria se añaden más conexiones punto a punto.
  - – **Comunicación punto a punto con arquitectura de comunicaciones**

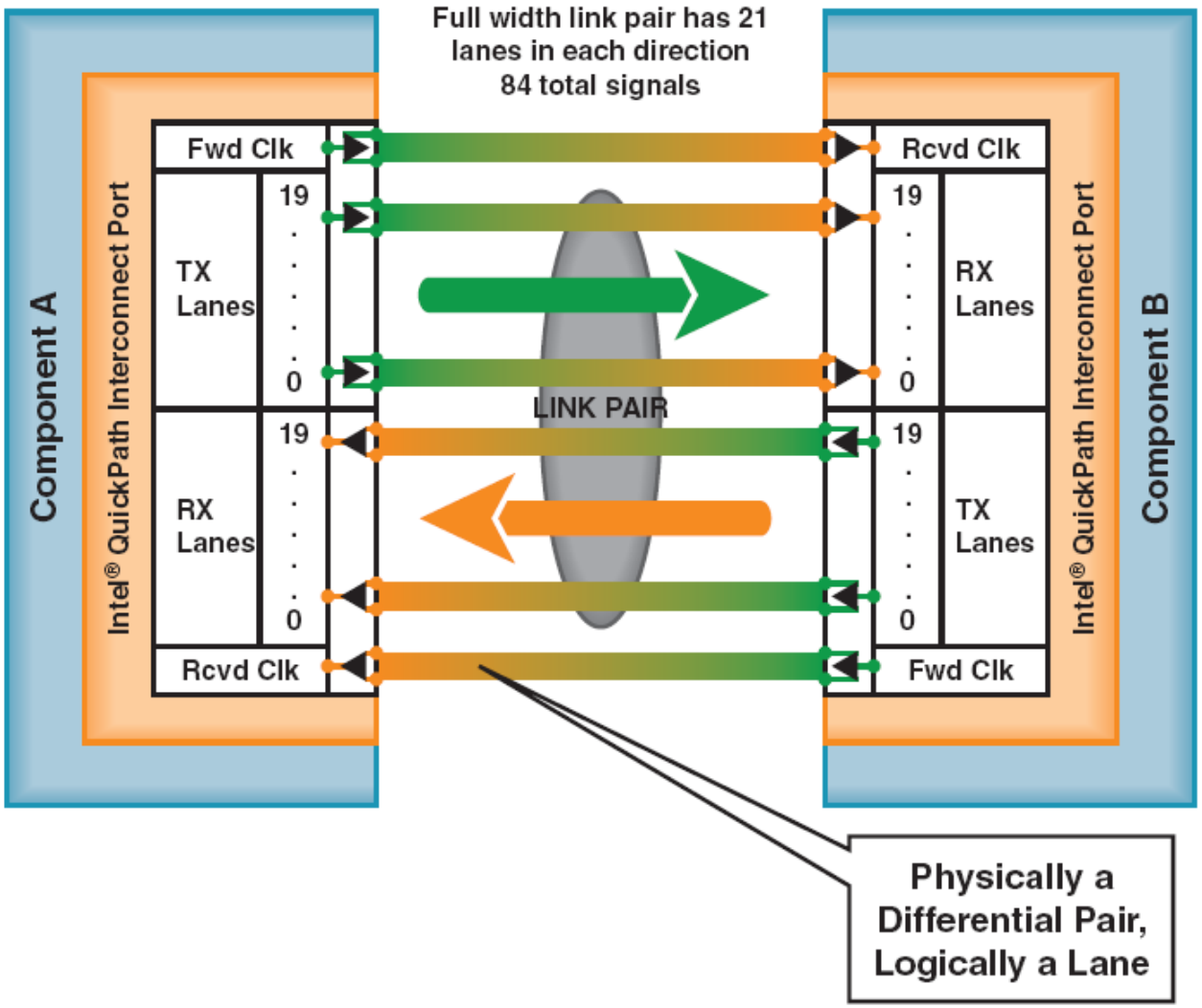
Ejemplo: PCI Express

- Tecnología de conexión punto a punto “serie”.
- Se conecta al procesador y a la memoria a través del chipset.
- El chipset o host bridge actúa como dispositivo de almacenamiento temporal (buffer) y para traducir entre los formatos de las transacciones PCIe y los del procesador y memoria.
- Se puede conectar más de un PCIe si se usa un switch.
  - El switch actúa como controlador de todos los dispositivos que utilizan PCI-Express.
- Ancho de bus (versión 3.0, 2010) de 1, 2, 4, 6, 16 ó 32 bits, con un canal para cada dirección y señalización diferencial. Al contrario que QPI no hay reloj y se añaden bits de sincronización (128b/130b)
- Compatible con bus PCI.
- Transferencia máxima de 16GB/seg Gen2, 32GB/seg Gen3

Ejemplo de conexión punto a punto: QPI (Intel)

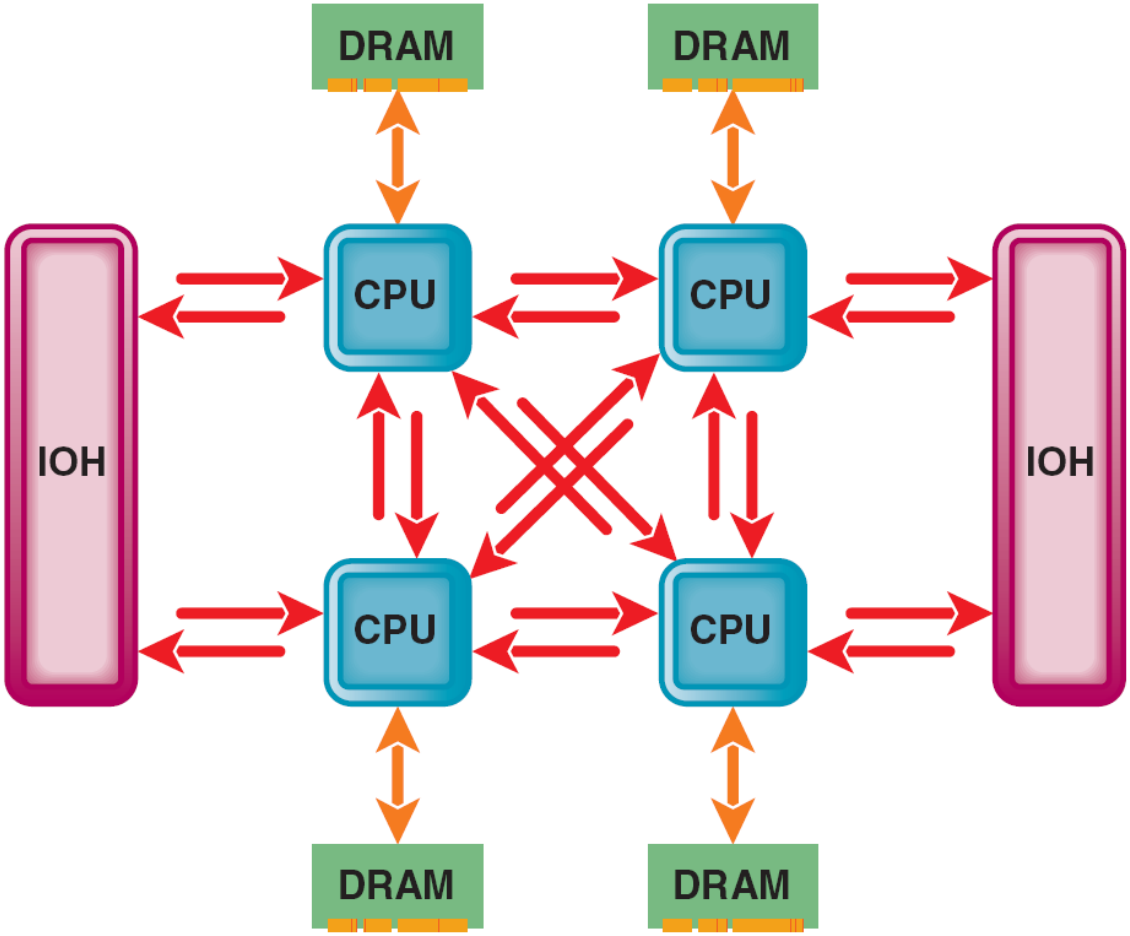
- Conexiones dedicadas.
  - Se elimina el arbitraje.
  - Mejor escalabilidad.
- QuickPath Interconnect (Intel 2008)
  - Conexión entre procesadores y entre un procesador y el hub de E/S.
  - Es unidireccíonal.
  - Cada puerto tiene 20 lanes de datos y uno de reloj, formados por 2 líneas cada uno, en cada dirección (84 líneas).
    - Usa señalización diferencial.
    - Hasta 6,4GT/S en cada dirección.

Conexión QPI



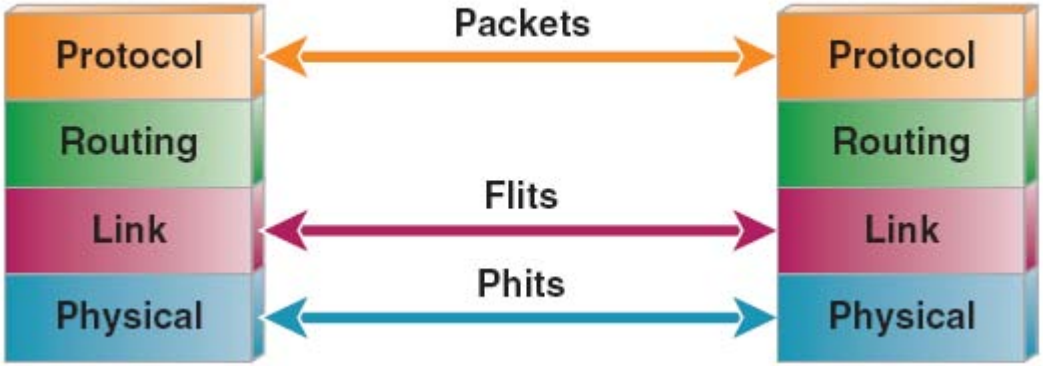
Ejemplo: QPI

- Plataforma con 4 procesadores usando tecnología QPI (líneas rojas)



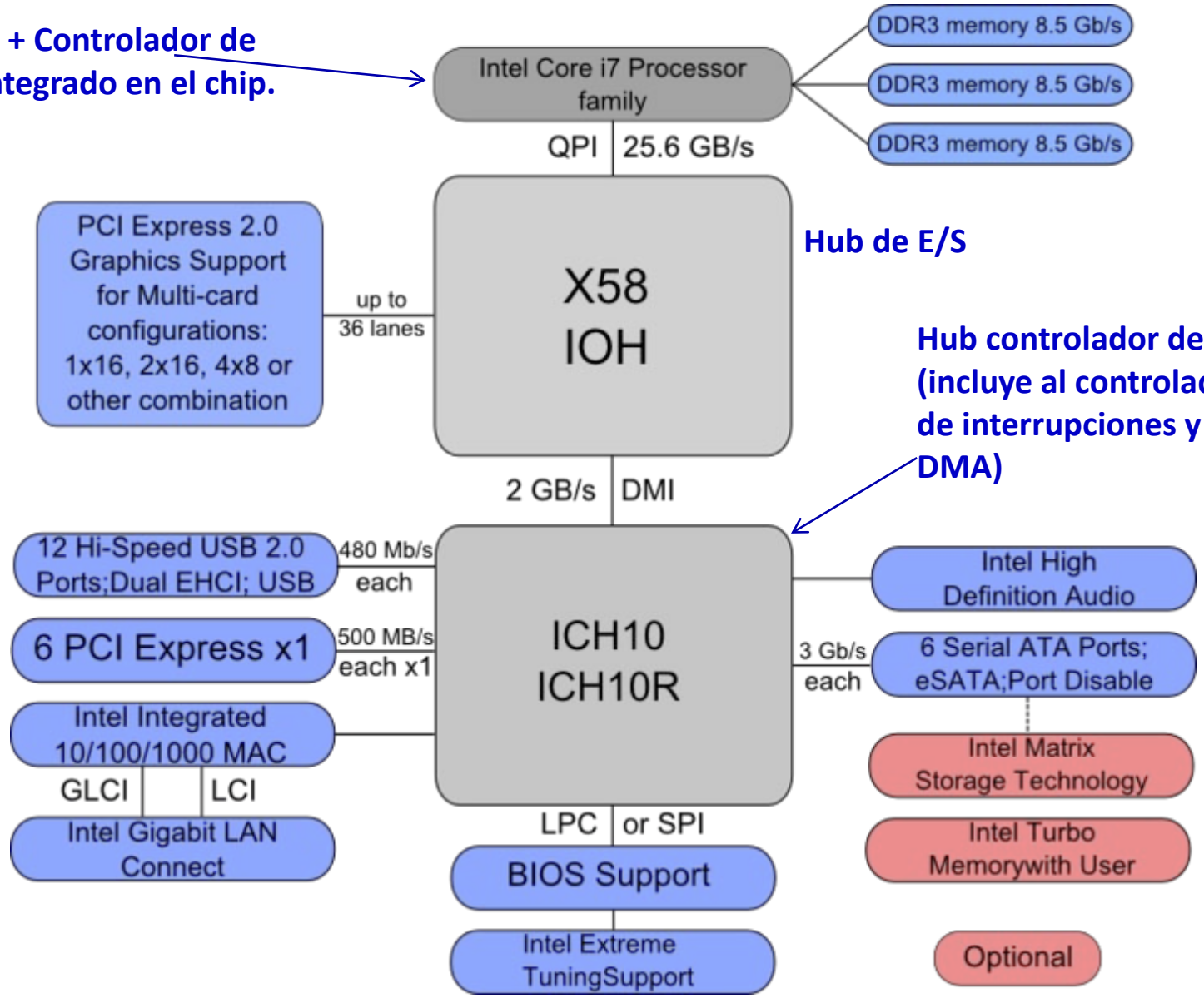
QPI: arquitectura de comunicaciones

- La arquitectura de comunicaciones tiene 4 niveles:
  - Físico: unidad de transferencia de 20 bits (uno por lane)
  - De enlace: unidad de transferencia de 80 bits
  - De encaminamiento
  - De protocolo



Ejemplo: Chipsets de Intel. 2008

Procesador + Controlador de memoria integrado en el chip.



Tema 2.2. Sistema de interconexión y entrada/salida mediante DMA

Entrada/salida mediante acceso directo a memoria

Necesidad del DMA: recordemos la E/S básica

2. Ejemplo de periférico rápido

- Procesador a 200 MHz (tiempo ciclo = 5 ns.); Ciclos medios por instrucción: CPI = 2 ciclos
  - Una instrucción tarda en promedio 2 x 5 ns = 10 ns ⇒ el computador puede ejecutar ~100 MIPS
- Disco con velocidad de transferencia de 10 Mbytes/s (1 byte cada 10^{-7} seg)
- Queremos transferir un fichero de memoria a disco de 10 Mbytes

a) E/S con espera de respuesta

- La CPU entra en un bucle y envía un nuevo byte cada vez que el disco está preparado para recibirlo
  - El disco tarda 1 seg en recibir un fichero de 10 Mbytes
  - **La CPU está ocupada con la operación de E/S durante 1 s** (en ese tiempo la CPU podría haber ejecutado 100 millones de instrucciones)

b) E/S por interrupciones

- El disco genera una interrupción cada vez que está preparado para recibir un nuevo byte
  - Suponemos que la RTI tiene 10 instrucciones (salvar contexto, comprobar estado, transferir byte, restaurar contexto, RTE)
  - Para transferir 10 Mbytes tenemos que ejecutar 10^7 veces la RTI

    ⇒ hay que ejecutar 100 millones de instrucciones para atender al periférico ⇒ la CPU tarda 1 s
  - **La CPU está ocupada con la operación de E/S durante 1 s**

CONCLUSIÓN

- La E/S por interrupciones no mejora el tiempo que la CPU está ocupada en atender al periférico

La E/S con espera de respuesta o por interrupciones resulta inadecuada para periféricos de alta velocidad, sobre todo si hay que transferir una gran cantidad de datos

Necesidad del DMA

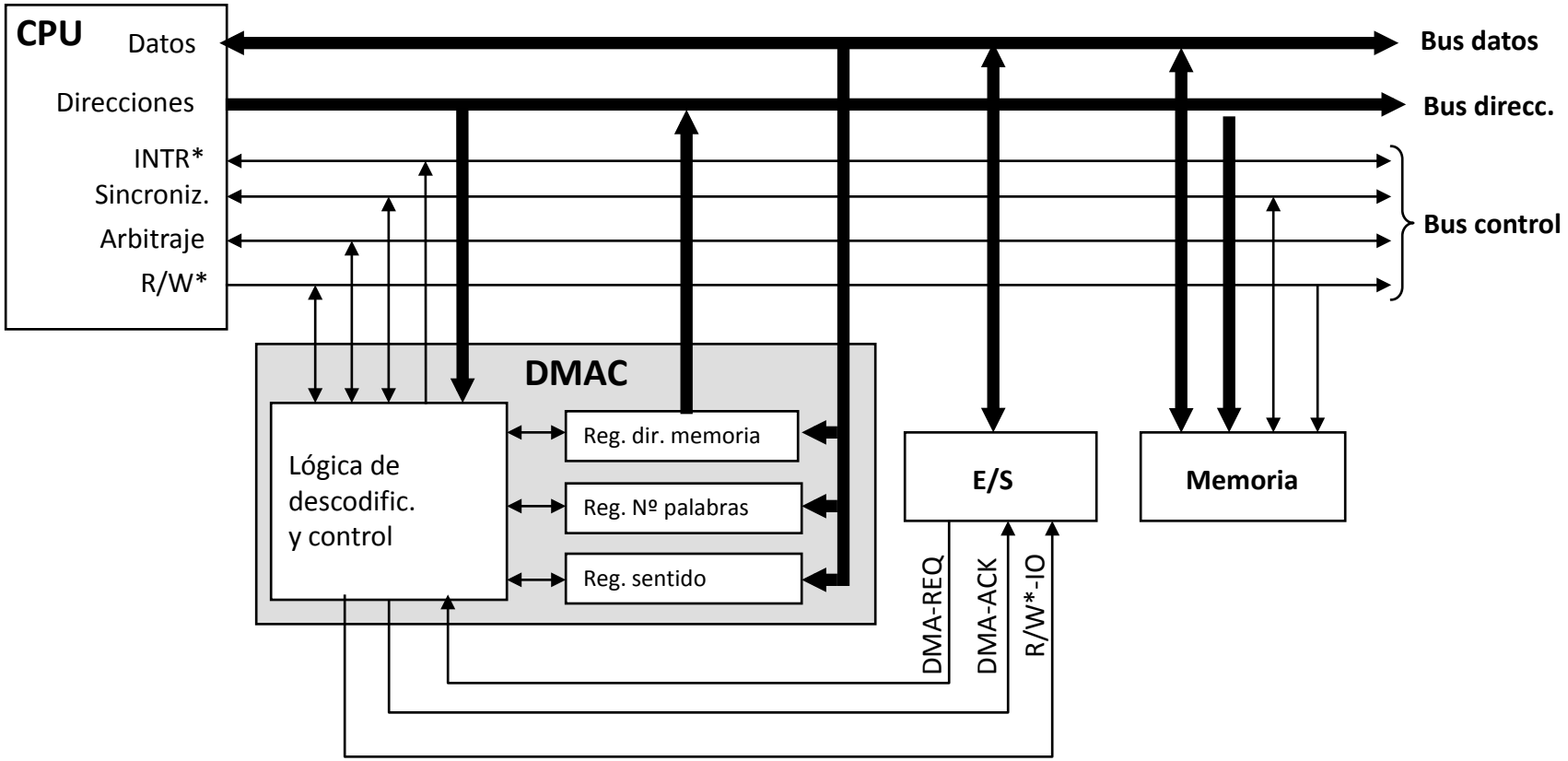
- ¿En qué consiste el Acceso Directo a Memoria (DMA)?
  - La técnica de DMA permite la transferencia de datos entre un periférico y la memoria sin intervención de la CPU (salvo en la fase de inicialización de los parámetros de la transferencia)
    - Con interrupciones se evita el bucle de espera pero la transferencia la lleva a cabo el procesador
      - Para un bloque de N bytes, se generan N interrupciones
    - Con DMA toda la transferencia la realiza el controlador de E/S
      - Solo una interrupción al final

El controlador DMA (DMAC)

- El controlador de DMA (DMAC) es un dispositivo capaz de controlar una transferencia de datos entre un periférico y memoria sin intervención de la CPU
- El DMAC debe actuar como máster del bus durante la transferencia DMA y debe ser capaz de
  - Solicitar el uso del bus mediante las señales y la lógica de arbitraje necesarias
  - Especificar la dirección de memoria sobre la que se realiza la transferencia
  - Generar las señales de control del bus
    - Tipo de operación (lectura/escritura)
    - Señales de sincronización de la transferencia

El DMAC: registros

- **Reg. dir. memoria:** almacena la dir. inicial de memoria y se incrementa/decrementa después de transferir cada palabra
- **Reg. Nº palabras:** almacena el número de palabras a transferir y se decrementa después de transferir cada palabra
- **Reg. sentido:** almacena el sentido de la transferencia (lectura o escritura)

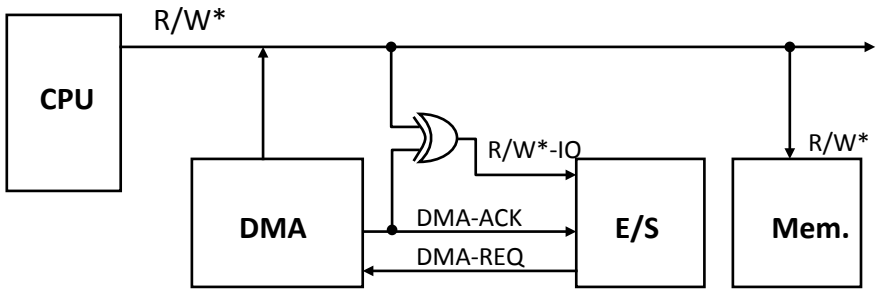


El DMAC: señales

- DMA-REQ: solicitud de servicio DMA
  - La activa el periférico para indicar al DMAC que está listo para transmitir/recibir
- DMA-ACK: Concesión del servicio DMA
  - La activa el DMAC para indicar al periférico que puede realizar la transferencia
  - Antes de activar esta señal el DMAC debe estar en posesión del bus
- R/W*-IO: Sentido de la transferencia para el periférico
  - El periférico no puede utilizar la misma señal de R/W* que la memoria, ya que la transferencia tiene sentidos opuestos desde el punto de vista de uno y otro dispositivo
  - Operación DMA de lectura (memoria $\rightarrow$ periférico)
    - Para la memoria es una lectura: R/W* = 1
    - Para el periférico es una escritura: R/W*-IO = 0
  - Operación DMA de escritura (periférico $\rightarrow$ memoria)
    - Para la memoria es una escritura: R/W* = 0
    - Para el periférico es una lectura: R/W*-IO = 1

Generación de las señales de control

Posible esquema de conexión de R/W* y R/W*-IO



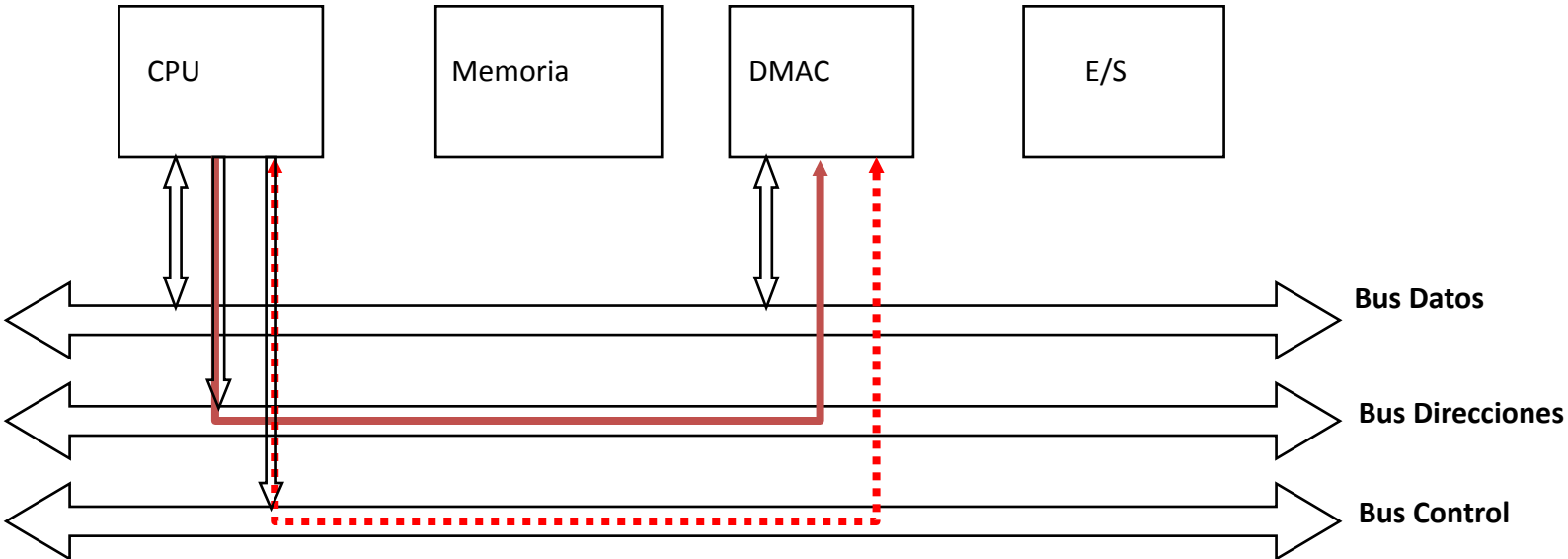
DMA-ACK	R/W*	R/W*-IO	
0	0	0	Operación de E/S normal (controlada por la CPU) R/W*-IO = R/W*
0	1	1	
1	0	1	Operación de E/S controlada por DMA R/W*-IO = NOT (R/W*)
1	1	0	

Etapas de una transferencia DMA

- Inicialización de la transferencia
  - La CPU debe enviar al DMAC los parámetros de la transferencia
  - Inicialización del controlador DMA (Bus master: CPU - Bus slave: DMAC)
    - Nº de bytes o palabras a transferir
    - Tipo de transferencia (lectura/escritura)
    - Dirección de memoria inicial para la transferencia
    - Nº de canal (para DMAs con varios canales)
  - Después de la inicialización la CPU retorna a sus tareas y ya no se preocupa más de la evolución de la transferencia
  - El DMAC envía al controlador del periférico los parámetros de la transferencia
  - Inicialización del controlador del periférico (Bus master: DMA - Bus slave: Controlador del periférico)
    - Nº de bytes a transferir
    - Tipo de transferencia (lectura/escritura)
    - Otra información de control (pista, sector, etc.)

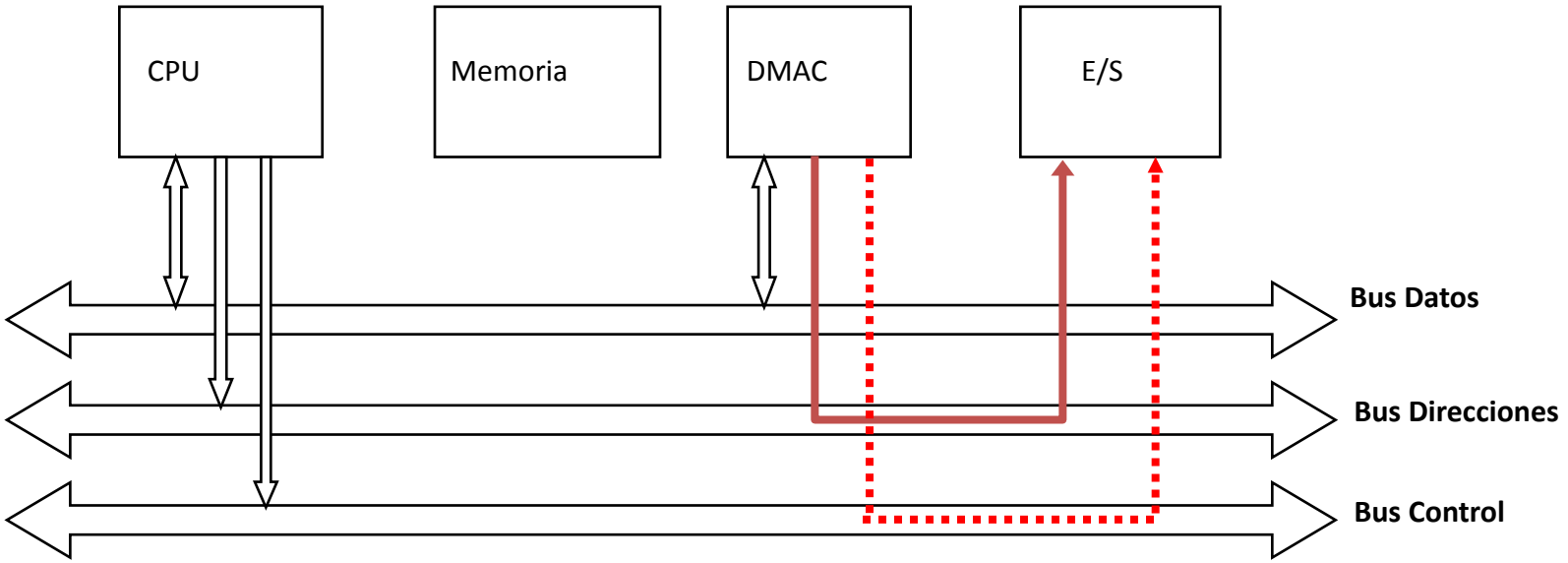
Inicialización de la transferencia (1)

1. La CPU programa al controlador de DMA para que ejecute la operación de E/S completa (todas las transferencias).
   - Dirección inicial de memoria → Reg. dir. memoria
   - Número de palabras a transferir → Reg. Nº palabras
   - Sentido de la transferencia → Reg. Sentido



Inicialización de la transferencia (2)

2. El controlador de DMA programa al controlador de dispositivo para realizar una transferencia.
   - Como en E/S programada.



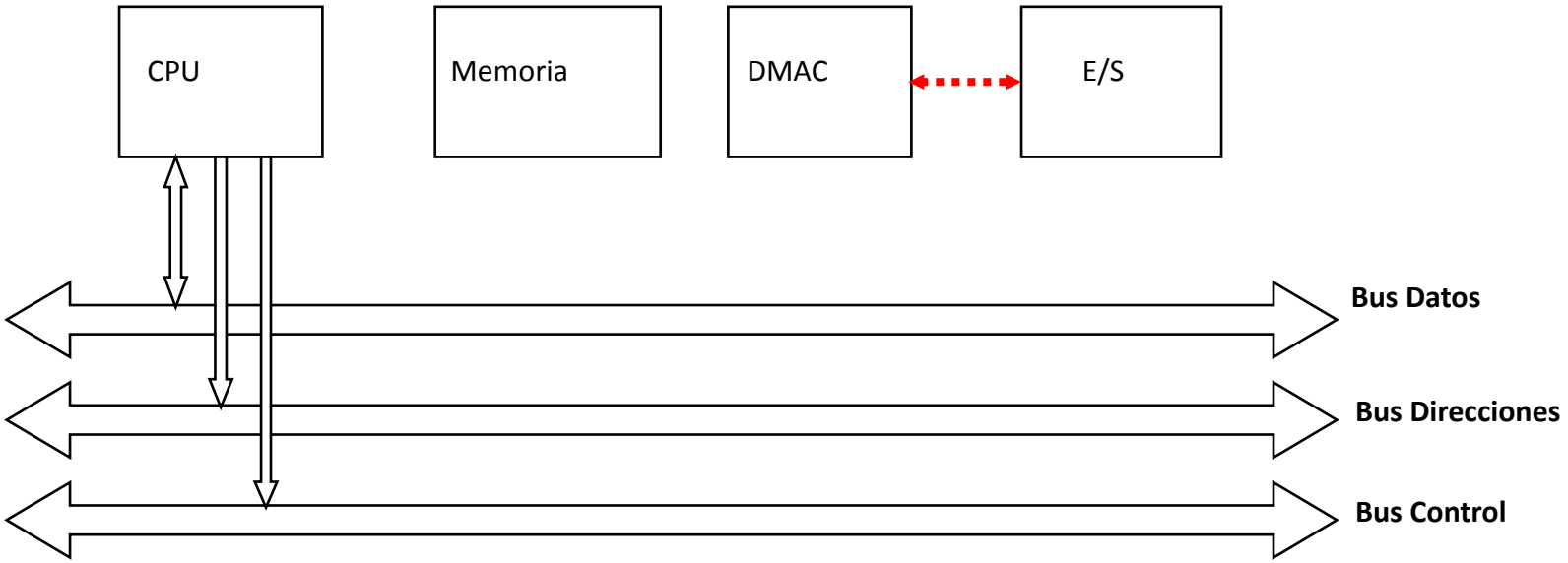
Etapas de una transferencia DMA (2)

- Realización de la transferencia
  - Cuando el periférico está listo para realizar la transferencia se lo indica al DMAC
  - El DMAC pide el control del bus y se realiza la transferencia entre el periférico y la memoria
    - Bus máster: DMAC + Periférico - Bus slave: Memoria
    - Después de la transferencia de cada palabra se actualizan los registros del DMA
      - Nº de bytes o palabras a transferir
      - Dirección de memoria
  - El DMAC devuelve el control del bus (esto dependerá del tipo de transferencia)

Realización de la transferencia

3. El controlador del dispositivo avisa al controlador de DMA de que está listo para realizar la transferencia.

   Activando la señal ***DMA-REQ***

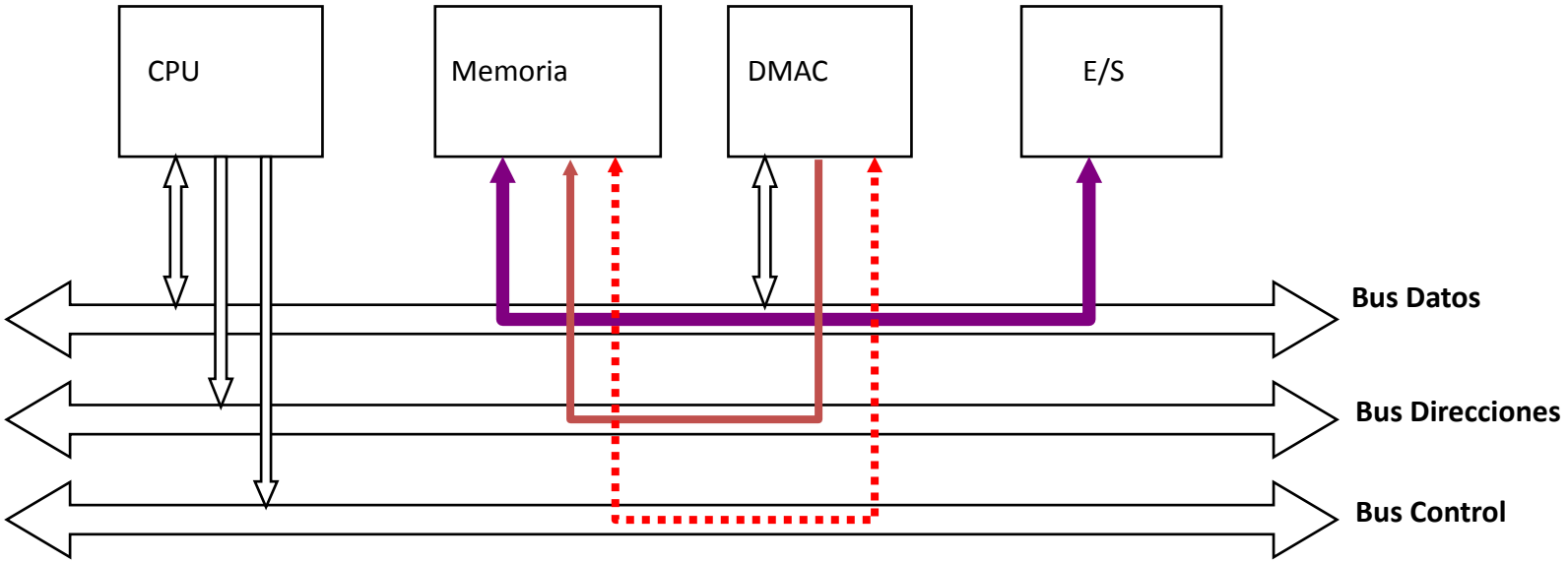


Realización de la transferencia (2)

4. Transferencia del dato:

El DMAC solicita el control del bus mediante las líneas de arbitraje
El DMAC recibe la concesión del bus y activa la señal **DMA-ACK** para indicar al periférico que puede iniciar la transferencia.
El DMAC debe generar y procesar las señales del bus adecuadas

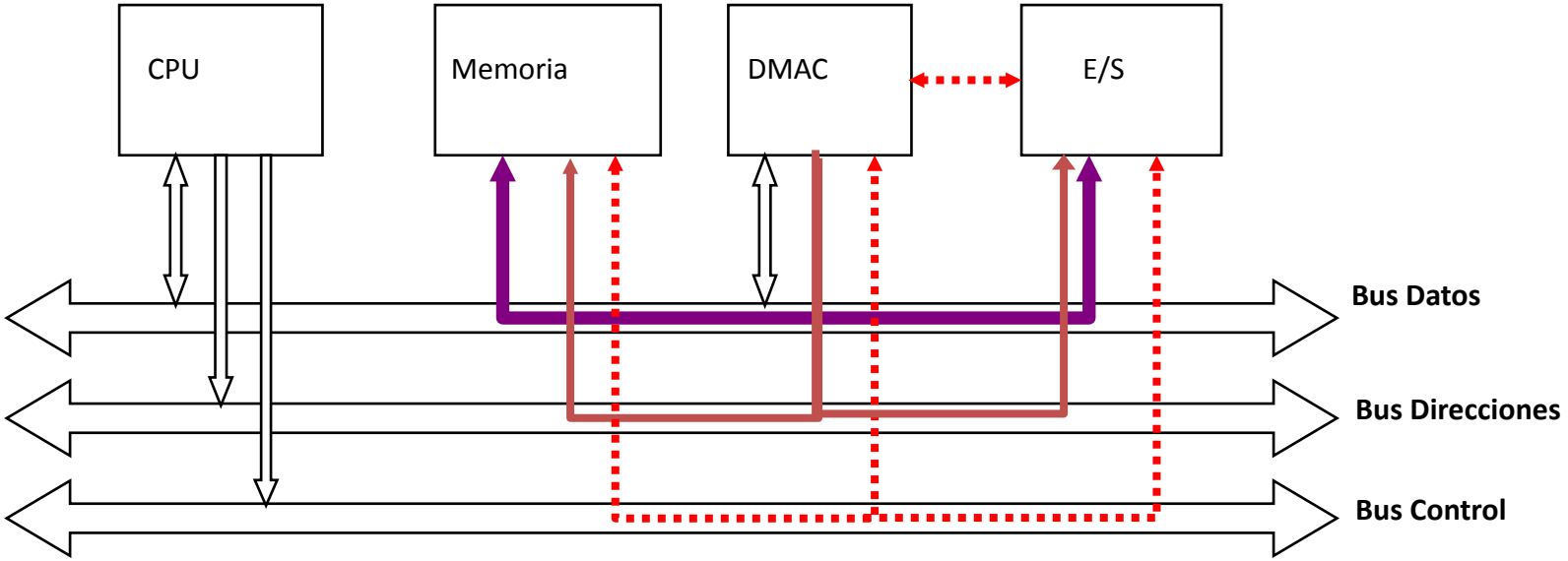
- ✓Dirección de memoria sobre la que se realiza la transferencia
- ✓Señales de sincronización de la transferencia (master-syncro, slave-syncro, etc.)
- ✓Señales de lectura/escritura (R/W* y R/W*-IO)



Intervienen: Controlador del dispositivo+ DMAC-Memoria

Realización de la transferencia (3)

- Después de transferir cada palabra el DMAC debe actualizar sus registros
  - Decrementar el registro de nº de palabras
  - Incrementar/decrementar el reg. de direcciones de mem. (según sean direcciones crecientes o decrecientes
- Se repiten los pasos 2, 3, 4 hasta que se hayan realizado todas las transferencias.



Etapas de una transferencia DMA (3)

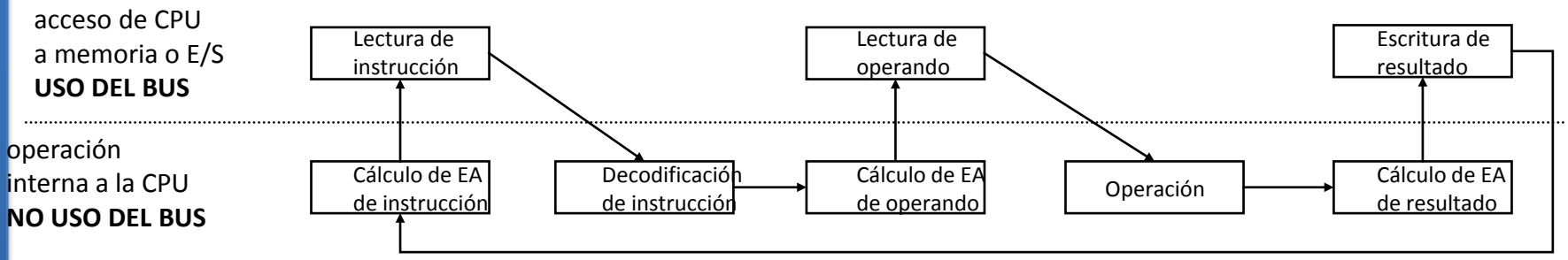
- Finalización de la transferencia: cuando el registro nº de palabras llega a cero.
  - El DMAC libera el bus y devuelve el control a la CPU
  - El DMAC suele activar una señal de interrupción para indicar a la CPU la finalización de la operación de E/S solicitada

Problema que puede presentar el DMA

- Puede degradar el rendimiento de la CPU si el DMAC hace uso intensivo del bus
  - Si el bus está ocupado en una transferencia DMA, la CPU no puede acceder a memoria para leer instruc./datos
- Este problema se reduce con el uso de memoria cache
  - La mayor parte del tiempo, la CPU lee instruc. de la cache, por lo que no necesita usar el bus de memoria
  - El DMAC puede aprovechar estos intervalos en los que la CPU está leyendo instrucciones de la cache (y por tanto no usa el bus de memoria) para realizar las transferencias
- En caso de computadores sin cache
  - El procesador no utiliza el bus en todas las fases de la ejecución de una instrucción
  - El DMAC puede aprovechar las fases de ejecución de una instrucción en las que la CPU no utiliza el bus para realizar sus transferencias

Problema que puede presentar el DMA

- **Conclusión**
  - Si el DMAC sólo toma el control del bus durante los intervalos de tiempo en los que la CPU no hace uso del mismo ⇒ *el rendimiento del sistema no sufrirá degradación alguna*



  - Se distinguen, por tanto, tres tipos de transferencias

    →**Por ráfagas (burst)**

    →**Por robo de ciclo (cycle-stealing)**

    →**Transparente**

Transferencia DMA en modo ráfaga

- El DMAC solicita el control del bus a la CPU
- Cuando la CPU concede el bus el DMAC no lo libera hasta haber finalizado la transferencia de todo el bloque de datos completo.

- VENTAJAS:
  - La transferencia se realiza de forma rápida
- DESVENTAJAS:
  - Durante el tiempo que dura la transferencia la CPU no puede utilizar el bus con memoria, lo que puede degradar el rendimiento del sistema

Transferencia DMA en modo robo de ciclo

- El DMAC solicita el control del bus a la CPU, que se lo cede al terminar de ejecutar la instrucción en curso.
- Cuando la CPU concede el bus al DMAC, se realiza la transferencia de una única palabra y después el DMAC libera el bus
- El DMAC vuelve a solicitar el control del bus tantas veces como sea necesario hasta haber finalizado la transferencia del bloque completo
- VENTAJAS:
  - Se degrada poco el rendimiento del sistema (1 ciclo por instrucción como máximo)
- DESVENTAJAS:
  - La transferencia tarda más tiempo en llevarse a cabo

Transferencia DMA en modo transparente

- El DMAC solicita el control del bus a la CPU, que se lo cede cuando no lo necesita.
- Cuando la CPU concede el bus al DMAC, se realiza la transferencia de una única palabra y después el DMAC libera el bus
- El DMAC vuelve a solicitar el control del bus tantas veces como sea necesario hasta haber finalizado la transferencia del bloque completo
- La CPU solo cede el control del bus durante los ciclos que no hace uso del mismo
- VENTAJAS:
  - No se degrada el rendimiento del sistema
- DESVENTAJAS:
  - La transferencia tarda más tiempo en llevarse a cabo

DMA y el sistema de memoria

- Sin el DMA el único acceso a la jerarquía de memoria es a través del procesador.
  - La traducción de direcciones virtuales se realiza en la MMU.
  - Hay datos que están en alguna de las memorias cache y que pueden no estar actualizados en memoria principal.
- El controlador de DMA accede directamente a memoria.
  - Problema con la traducción de direcciones y la coherencia de cache.