Estructura de Computadores 2 [08/09]

Buses de interconexión

1. Introducción. Estructura del bus y uso básico. Clases de buses y jerarquías
2. Elementos de diseño de un bus:
   - Tipos de buses, anchura del bus, temporización, método de arbitraje y tipos de transferencia de datos
3. Arbitraje del bus
4. Ejemplos de buses estándar
   - PCI, SCSI, USB y FireWire, AGP y PCI-Express

10 de mayo de 2009

Bibliografía

- *Computer Architecture: A Quantitative Approach (3rd or 4th ed.)*, John L. Hennessy y David A. Patterson. Morgan Kaufmann Publishers, Inc.
- *Organización y arquitectura de computadores (7th ed.)*, William Stallings. Prentice Hall.
- *Organización de Computadores*, C. Hamacher, Z. Vranesic y S. Zaky. Mc Graw Hill, 2003.
- *Computer Organization and Design: The hardware/software interface (3rd ed.)*, David A. Patterson and John L. Hennessy. Morgan Kaufmann Publishers, Inc.

GAC: Grupo de Arquitectura de Computadores. Dpt. Electrónica e Sistemas. Universidade da Coruña.

Introducción

- Las distintas unidades funcionales de un computador necesitan comunicarse. Deben existir, por lo tanto, líneas para interconectar estos módulos.

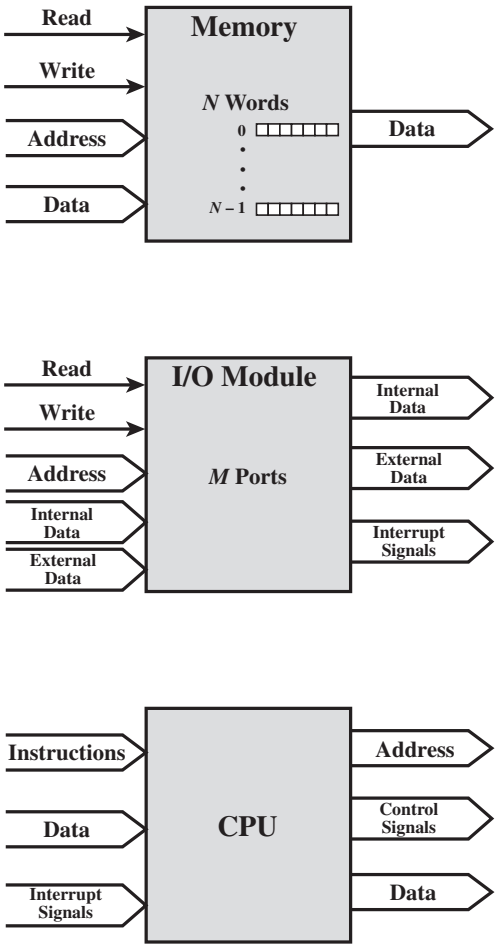


Figure 3.15 Computer Modules

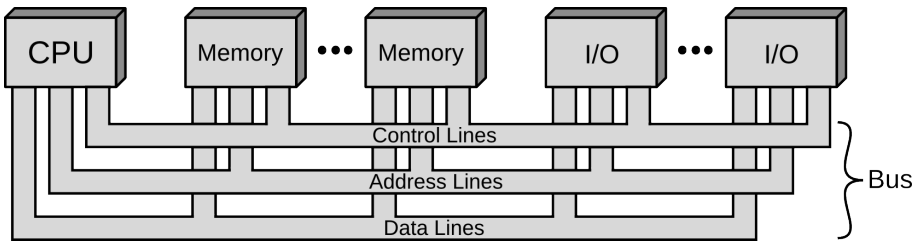
- Tipos de transferencias a las que debe dar cobertura la estructura de interconexión:
  - Procesador ⇔ Memoria
  - Procesador ⇔ E/S
  - Memoria ⇔ E/S (DMA)

Introducción

- Un bus es un canal de comunicación compartido que utiliza un conjunto de líneas para conectar múltiples subsistemas. Cada cable o línea transmite un único bit de información en un determinado momento.
- Ventajas:
  - versatilidad
  - bajo coste
- Desventaja: supone un cuello de botella, limitando la productividad máxima del sistema.
- A la hora de diseñar un sistema de buses, el reto consiste en cubrir la demanda de comunicación del sistema y permitir la conexión de un gran número de dispositivos de E/S.
- Características deseables:
  - heterogeneidad
  - escalabilidad
  - baja latencia
  - alto ancho de banda
- Principal problema en el diseño de un BUS: la velocidad máxima (y por tanto el rendimiento) está fuertemente limitada por cuestiones físicas:
  - longitud del bus
  - número de dispositivos conectados
- Y también por la necesidad de dar soporte a una gran variedad de dispositivos con muy distintas latencias y anchos de banda
- Los crecientes problemas que presentan las cada vez más **altas velocidades** de transmisión en múltiples hilos paralelos hacen que la industria se encuentre en plena transición: de buses paralelos compartidos a **interconexiones punto-a-punto de alta velocidad con switches**.

Estructura del bus

- Las líneas que componen un bus se pueden clasificar en tres grupos funcionales:



  - Las **líneas de datos** del bus proporcionan el camino para transmitir datos entre los módulos del sistema. El número de líneas del bus de datos determina el número máximo de bits que es posible transmitir al mismo tiempo.
  - Las **líneas de dirección** se utilizan para designar (direccionar) la fuente o el destino de los datos situados en el bus de datos. La anchura del bus de direcciones determina la cantidad máxima de memoria (y de dispositivos de E/S) direccionable en el sistema.
  - Las **líneas de control** se emplean para gestionar el acceso y el uso de las líneas de datos y dirección, señalizando peticiones y reconocimientos e indicando qué tipo de información pasa por las líneas de datos.

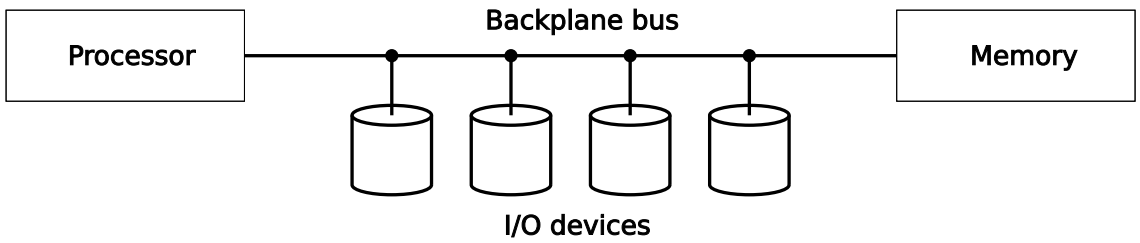
- Clasificación según las características eléctricas:
  - Unidireccionales con un transmisor y múltiples receptores
  - Unidireccionales con múltiples transmisores y un único receptor
  - Bidireccionales
- Algunas líneas de control típicas son:
  - Escritura en memoria (*Memory Write*)
  - Lectura de memoria (*Memory Read*)
  - Escritura de E/S (*I/O Write*)
  - Lectura de E/S (*I/O Read*)
  - Transferencia reconocida (*Transfer ACK*)
  - Petición de bus (*Bus Request*)
  - Cesión de bus (*Bus Grant*)
  - Petición de interrupción (*Interrupt Request*)
  - Interrupción reconocida(*Interrupt ACK*)
  - Reloj (*Clock*)
  - Inicio (*Reset*)

Uso básico del bus

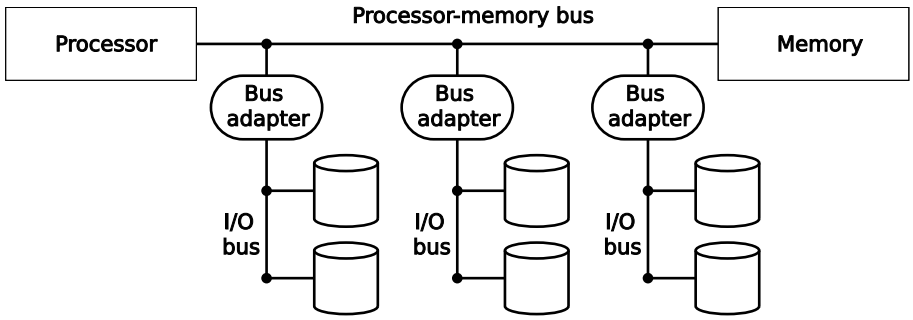
- Cuando un módulo necesita enviar un dato a otro módulo:
  1. Obtener el uso del bus
  2. Transferir el dato a través del bus
- Cuando un módulo necesita pedir un dato a otro módulo:
  1. Obtener el uso del bus
  2. Transferir la petición al otro módulo mediante las líneas de control y dirección apropiadas
  3. Esperar a que el segundo módulo envíe el dato

Clases de buses y jerarquías

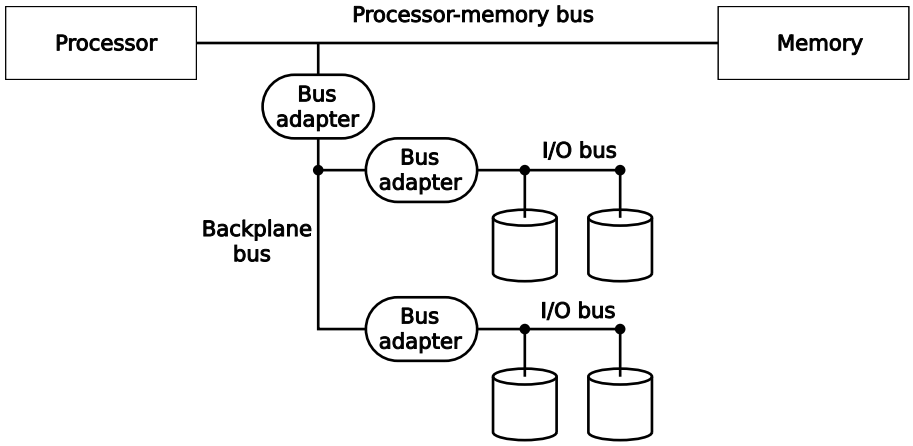
- Bus **Procesador-memoria**
- Bus **E/S**
- Bus ***Backplane*** o ***Mezzanine*** (Literalmente: «entresuelo». Ejemplo, PCI)
- Las prestaciones del bus disminuyen con la conexión de un gran número de dispositivos (cuello de botella). Solución: configuraciones con múltiples buses, normalmente organizadas jerárquicamente. Ejemplos:
  - IBM PC



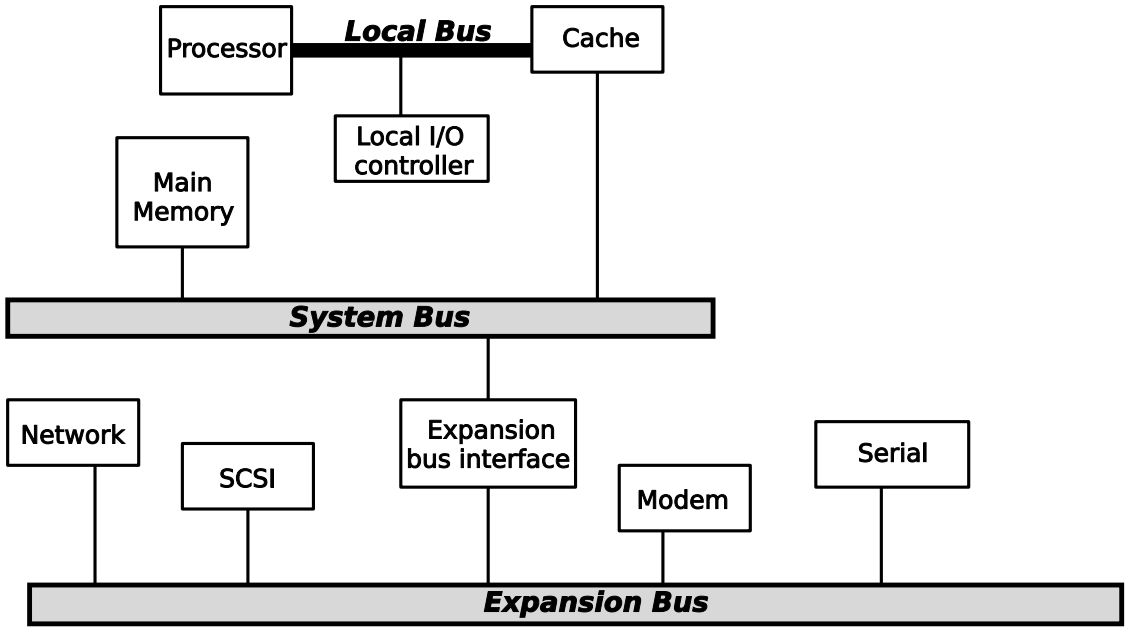
  - Apple Macintosh II



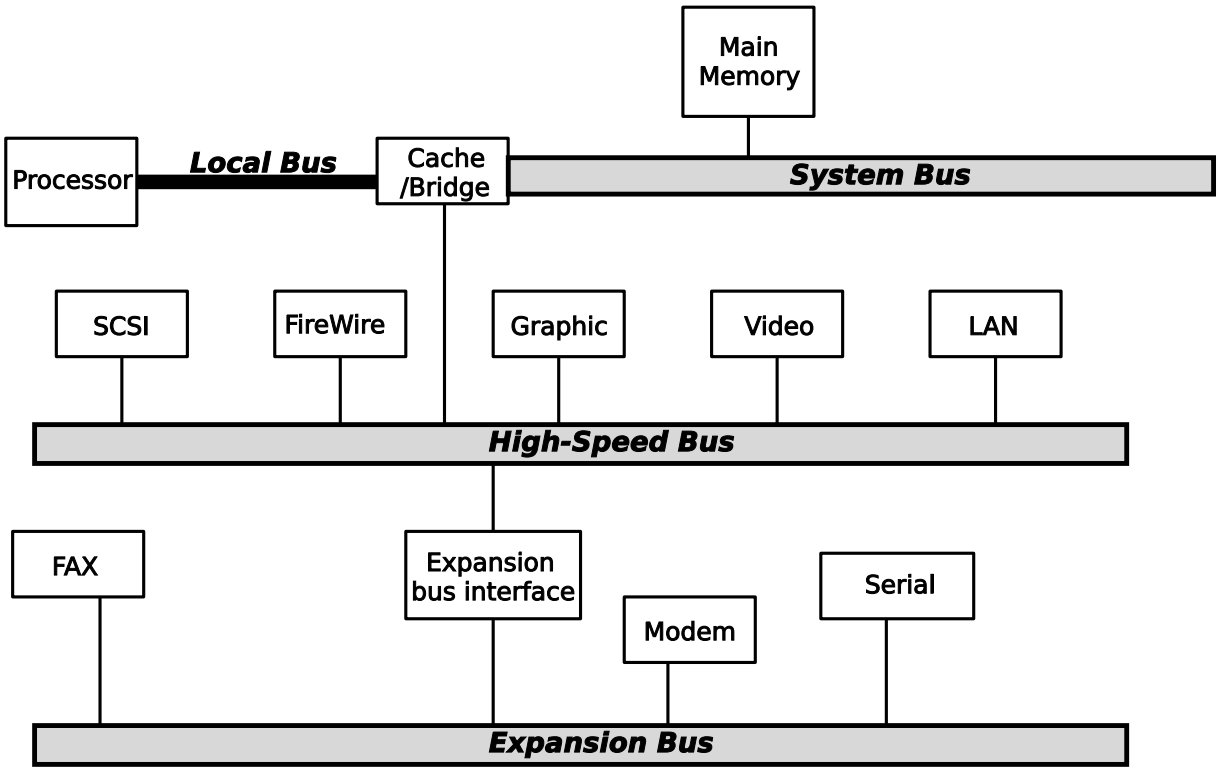
  - IBM RS-6000



– Más configuraciones



(a) Traditional Bus Architecture



(b) High-Performance Architecture

Diseño de un bus

- Tipo de bus
  - Dedicado
  - Multiplexado
- Anchura del bus
  - Dirección
  - Datos
- Temporización
  - Síncrono
  - Asíncrono
- Número de maestros de bus
  - Uno
  - Varios (opción: orientación a paquete)
- Método de arbitraje
  - Centralizado
  - Distribuido
- Tipo de transferencia de datos
  - Lectura
  - Escritura
  - Lectura-modificación-escritura
  - Lectura-después de-escritura
  - Bloque
  - Transacción partida (o *packet-switched bus*): Se dividen eventos del bus en *peticiones* y *respuestas*. Otro maestro puede usar el bus.

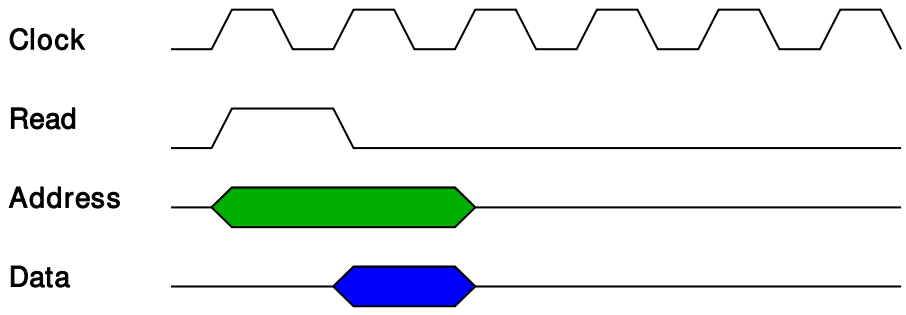
Temporización

- **Bus síncrono**. Entre sus líneas de control se incluye una señal de reloj. El protocolo para la comunicación es fijo y está gobernado por la señal de reloj ⇒ se asume que los envíos llegan correctamente.

  Ventajas: Puede funcionar a gran velocidad y se puede implementar con un sistema secuencial sencillo.

  Inconvenientes: No es adecuado para mezclar dispositivos con grandes diferencias de velocidad. Su diseño tiene que ser muy cuidadoso. Problema del sesgo del reloj (*clock skew*).

  Los buses de memoria suelen ser síncronos.



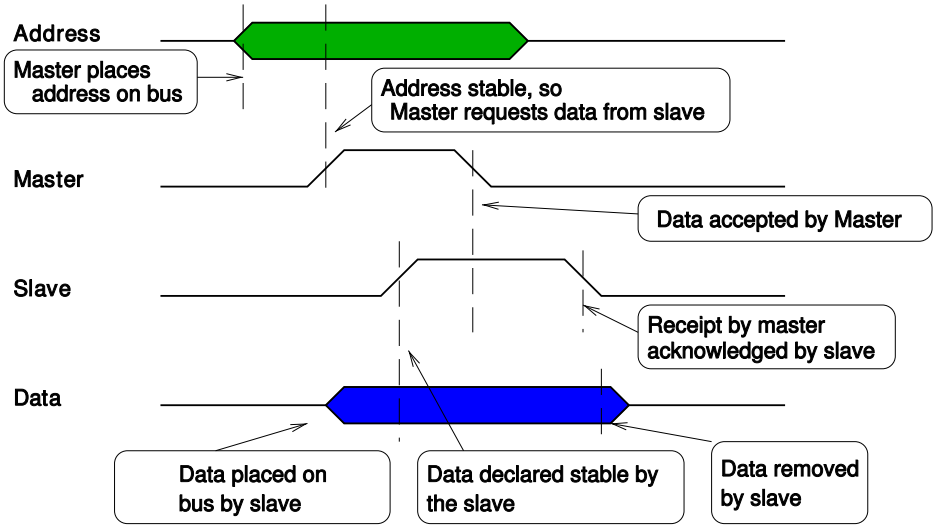
Ejemplo de lectura síncrona.

- **Bus asíncrono**. Sin reloj. Las transmisiones de datos se coordinan con un protocolo de *handshaking* entre emisor y receptor: un evento origina el siguiente, y así sucesivamente.

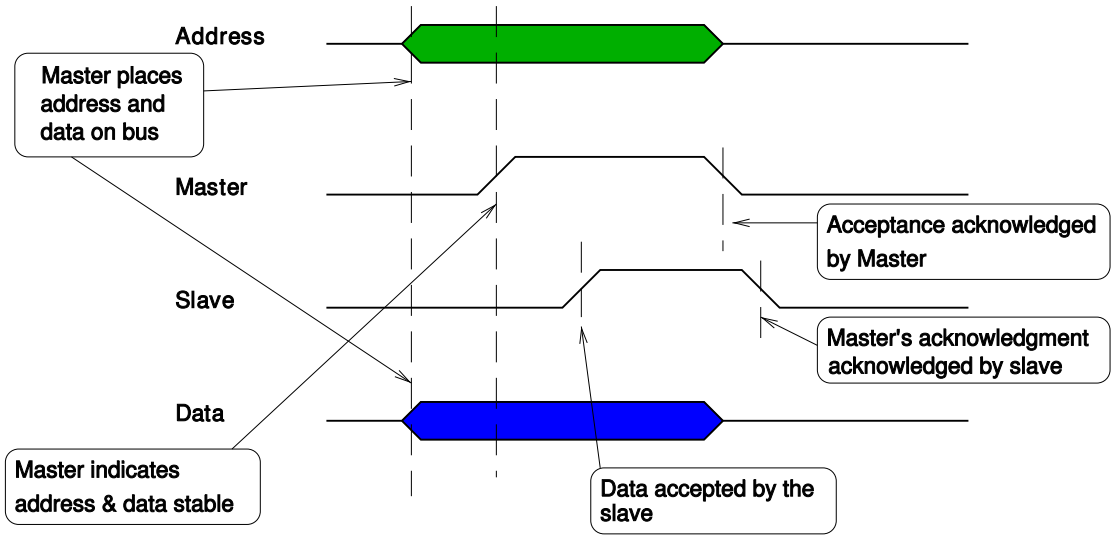
  Ventajas: Permite la conexión de dispositivos de un amplio rango de velocidades diferentes. Escalan mejor tanto con el número de dispositivos como con los cambios tecnológicos en los mismos. No hay problemas de sesgo de reloj, por lo que permite distancias más largas.

  Inconvenientes: Es más lento, debido a la sobrecarga introducida para sincronizar a emisor y receptor. Puede necesitar un cierto número de líneas de control adicionales para implementar el protocolo. Es más difícil predecir el tiempo que va a llevar una determinada transacción.

  Los buses de E/S son habitualmente asíncronos.



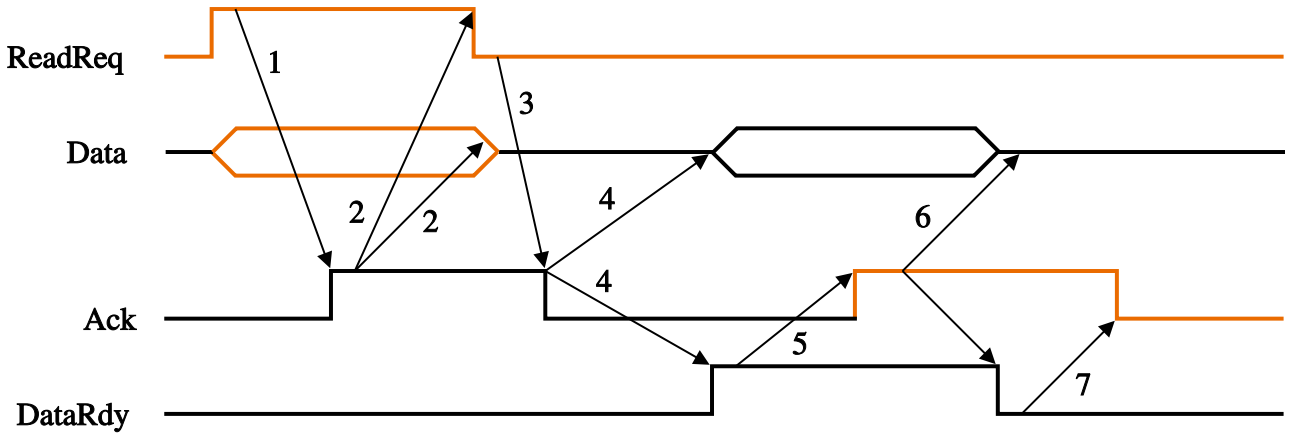
Ejemplo de lectura asíncrona.



Ejemplo de escritura.

Temporización

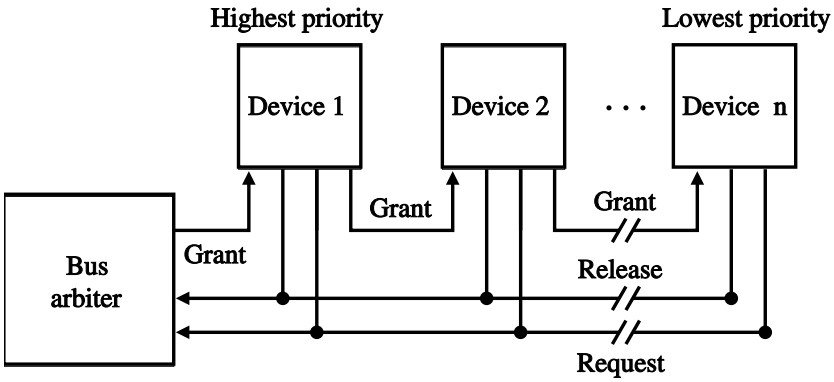
- Otro ejemplo diferente de protocolo *handshaking* para realizar una lectura asíncrona:



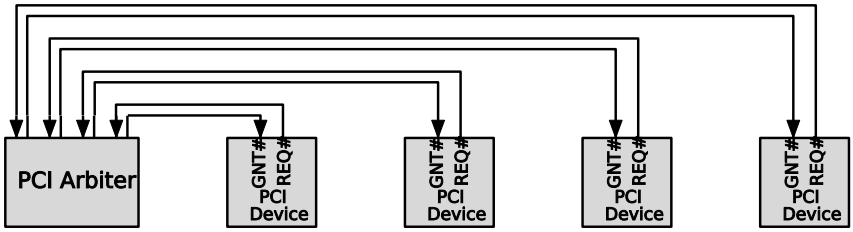
0. El protocolo comienza con el maestro (el procesador, por ejemplo) poniendo una dirección de memoria en el bus de datos y activando la señal de petición de lectura (*ReadReq*).
1. El esclavo (la memoria, por ejemplo) ve la petición y coge la dirección de memoria del bus de datos, activando la señal de aceptación (*ACK*).
2. El maestro ve activa la señal de *ACK* y libera el bus de datos y desactiva la señal de petición de lectura, que sirve de aviso al esclavo.
3. El esclavo ve la desactivación de la señal de petición y contesta así mismo con la desactivación de la señal de *ACK*.
4. Cuando el esclavo tiene listo el dato a leer lo pone en el bus y activa la señal que indica que el dato está listo (*DataReady*).
5. El maestro ve la señal *DataReady*, con lo que sabe que el dato está listo en el bus y realiza su lectura. Cuando completa la operación activa la señal de *ACK*.
6. El esclavo ve la señal de *ACK*, lo que le dice que el maestro ya tiene el dato leído, y desactiva la señal *DataReady* y libera el bus de datos.
7. Por último, el maestro responde a la desactivación de *DataReady* con la bajada de su señal de *ACK*, lo que da por concluida la transmisión.

Esquemas de arbitraje

- Gestión del uso del bus por parte de múltiples maestros.
- Dos factores a considerar:
  - prioridad
  - imparcialidad (*fairness*)
- Clasificación genérica de técnicas de arbitraje:
  1. **Arbitraje en serie** (*daisy chain arbitration*). La línea de concesión de bus (*grant*) recorre los dispositivos desde el más hasta el menos prioritario. Las prioridades se determinan así en función de la posición del dispositivo en el bus.

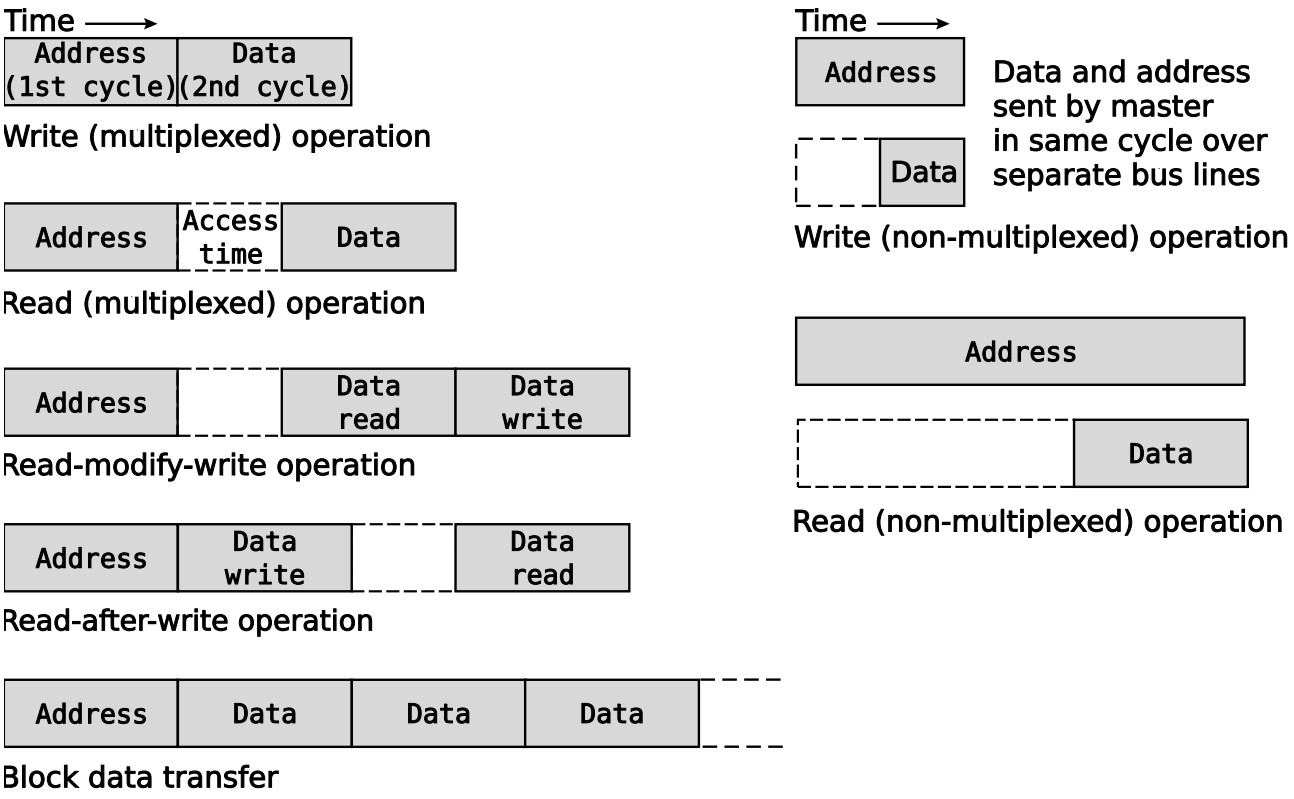


  2. **Arbitraje paralelo centralizado**. Se utilizan múltiples líneas de petición, por las que los diferentes dispositivos piden acceso al bus de forma independiente. Un árbitro centralizado selecciona uno de entre los dispositivos que han solicitado el bus y le notifica que ahora es el maestro del bus.



  3. **Arbitraje distribuido por autoselección**. Se emplean también múltiples líneas de petición de bus, pero ahora cada dispositivo determina de forma independiente si él es el solicitante de mayor prioridad sin necesidad de un árbitro.
  4. **Arbitraje distribuido por detección de colisión**. Cada dispositivo solicita de forma independiente el bus. En caso de múltiples peticiones simultáneas de bus se produce una colisión. Una vez detectada la colisión se aplica un esquema que determine el dispositivo que será maestro de bus entre las partes en colisión.

Tipo de transferencia de datos



Decisiones de diseño

Rendimiento vs. Coste

Opción	Alto rendimiento	Bajo coste
Ancho de bus	líneas de datos y direcciones indep.	líneas de datos y direcciones multiplexadas
Ancho de bus	Más ancho es más rápido	Menos ancho es más barato
Tamaño transfers	transfer bloq de múltiples palabras	transfer bloq. de una palabra
Maestros de bus	múltiples maestros (req. arbitraje)	un único maestro
Transac. partida	Sí. Aumenta ancho de banda (necesita varios maestros)	No. Conexión continua más barata (y con menor latencia)
Temporización	síncrono	asíncrono

Ejemplos de buses estándar

BUS ISA (*Industrial Standard Architecture*) (hoy obsoleto)

- Desarrollado por IBM primero para su *PC XT* (Intel 8088 y 8086), **bus XT**, y luego para el *PC AT* (80286), **bus AT**.
- Su éxito lo convirtió en un estándar *de facto*.
- Especificaciones: 8 *bits* - 4,77 Mhz Bus XT / 16 *bits* - 8 Mhz Bus AT ($5 - 8$ MB/s)
- Posteriormente IBM desarrollaría el **bus MCA** (*Micro Channel Architecture*) para sustituirlo (32 *bits* - 10 Mhz para 40 MB/s). La falta de compatibilidad hacia atrás con ISA lo hizo fracasar.
- Una alternativa algo más exitosa (relativamente) fue el **bus EISA** (*Extended Industrial Standard Architecture*), bus rápido de 32 bits (33 MB/s) con compatibilidad con ISA desarrollado por la competencia para competir con el MCA de IBM.
- El cuello de botella que supone el bus del sistema se hace cada vez más acusado en aplicaciones con cierta exigencia gráfica. La solución pasa por incorporar un nuevo bus a la arquitectura (*local I/O bus*) que permita una interfaz más directa con el procesador:
  1. **VESA** local bus (VL-bus), por la *Video Electronics Standards Association*. Ofrece acceso directo a memoria a la velocidad del procesador (132 MB/s). Fuerte dependencia del procesador a nivel físico (80486).
  2. **PCI**, desarrollado por Intel (Pentium en adelante). Aislado de la CPU aunque manteniendo el acceso a memoria.

BUS PCI (*Peripheral Component Interconnect*)

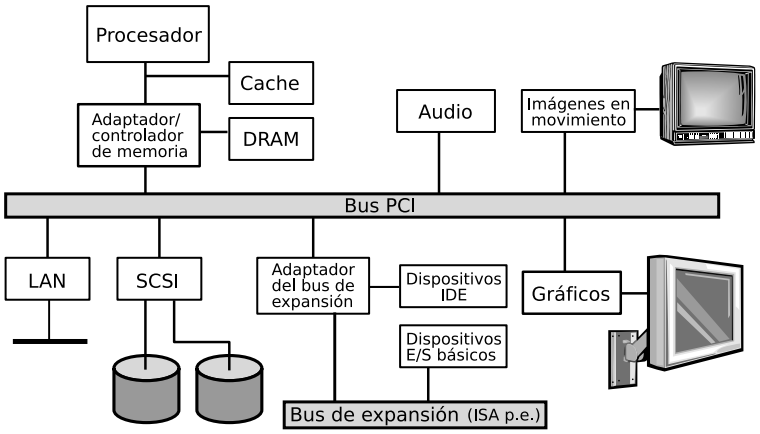
- Bus de 64 *bits* (64 líneas de datos), frecuentemente implementado como bus de 32 *bits*.
- Reloj: 33 o 66 Mhz. Ejemplos de configuración: 32 *bits* - 33 Mhz: 133 MB/s, 64 *bits* - 66 Mhz: 528 MBs/s
- Multiplataforma e independiente de la arquitectura del procesador.
- De diseño sencillo (pocos circuitos integrados) y económico, permite la interconexión de otros buses (como ISA) y es fácilmente ampliable.
- Frente al bus ISA tradicional incorpora mejoras como:
  - Permite la compartición de líneas de petición de interrupción (*IRQS*).
  - Importantes mejoras en cuanto a *bus mastering*, que permite la introducción de DMAs en los propios módulos de E/S.

Después de PCI:

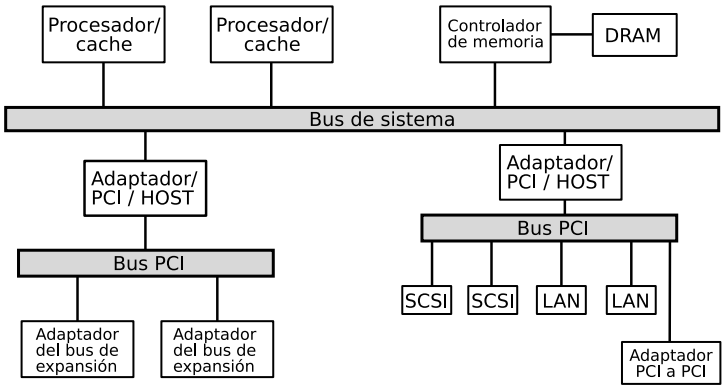
- **PCI-X** (*PCI eXtended*): diseñado para aumentar el rendimiento (hasta 1 GB/s) en la conexión de dispositivos con grandes necesidades de ancho de banda: Gigabit Ethernet, fibra óptica, procesadores en un cluster... Es compatible con PCI.
- **AGP** (*Accelerated Graphics Port*): interfaz diseñada por Intel, basándose en PCI, para conexión de tarjetas gráficas. Introduce canal punto-a-punto dedicado de 32 $bits$ - 66 Mhz para acceso de procesador gráfico a memoria principal (1X 266 MBs/s).
- **PCI Express**: diseñado para sustituir tanto a PCI como a AGP. Basado en canales serie punto-a-punto (200 MBs/s por canal).

Bus PCI

- Variedad de configuraciones diferentes, con uno o más procesadores.
- Conjunto de funciones de uso general.
- Temporización síncrona y esquema de arbitraje centralizado.



Sistema de sobremesa típico



Servidor típico

Bus PCI: estructura

- Variedad de configuraciones diferentes, con uno o más procesadores.
- 49 líneas de señal **obligatorias**: (# indica señal activa a nivel bajo)
  - Sistema: reloj y reinicio (*CLK* y *RST#*)
  - Direcciones y datos
    - 32 líneas multiplexadas para datos y direcciones (*AD*[31:0]).
      - ◇ PCI es *little endian*
    - 4 líneas de control multiplexadas (*C/BE*[3:0]#):
      - ◇ órdenes durante la fase de direccionamiento
      - ◇ indicador de validez de los datos (*byte enabled*) durante la fase de datos
    - línea de control de paridad (*PAR*)
  - Control de interfaz: temporización y coordinación de las transferencias (6 líneas)
    - Indicación de comienzo y duración de transferencia por maestro (*FRAME#*)
    - Maestro preparado (*IRDY#*, *Initiator Ready*)
    - Esclavo preparado (*TRDY#*, *Target Ready*)
    - Esclavo desea que maestro pare transacción actual (*STOP#*)
    - Respuesta del dispositivo esclavo seleccionado cuando reconoce su dirección (*DEVSEL#*)
    - Selección de un dispositivo durante la fase de inicialización del bus (*IDSEL*)
  - Arbitraje (2 líneas punto-a-punto)
    - Solicitud de bus (*REQ#*)
    - Concesión de bus (*GNT#*)
  - Señales de error (2 líneas)
    - Error de paridad (*PARR#*)
    - Error del sistema (*SERR#*)
- 51 señales **opcionales**
  - Interrupción
  - Soporte caché
  - Ampliación del bus a 64 bits
  - Test

Bus PCI: Espacio de direcciones y operaciones

- **Espacio de direcciones**: La especificación PCI establece tres espacios de direcciones diferentes:
  - Configuración
    - 256 *bytes*:
      - 64 *bytes* ($00h - 3Fh$) estándar predefinido PCI
      - 192 *bytes* ($40h - FF$) configurables
    - Información básica del dispositivo
    - Información *Plug´n Play*
  - E/S
    - Especificación PCI permite 4 *bytes* - 4 *Gbytes*
    - La compatibilidad ISA restringe a 256 *bytes*
    - Se suelen mapear periféricos como teclado, puerto serie...
  - Memoria (recomendado)
    - Hasta 4 *Gbytes* de propósito general
- Tipos de transferencia PCI: **operaciones**

C/BE	COMANDO
0000	Reconocimiento de interrupción
0001	Ciclo especial
0010	Lectura de E/S
0011	Escritura en E/S
0100	Reservado
0101	Reservado
0110	Lectura de memoria
0111	Escritura en memoria
1000	Reservado
1001	Reservado
1010	Lectura de configuración
1011	Escritura de configuración
1100	Lectura múltiple de memoria
1101	Ciclo de dirección dual
1110	Lectura de línea de memoria
1111	Escritura e invalidación de memoria

Transferencias de datos

- Cada transferencia de datos en el bus PCI es una transacción única.
- Transacción PCI: 1 fase de direccionamiento + 1 o más fases de datos.
- Todos los eventos se sincronizan en transiciones de bajada del reloj (mitad de ciclo de reloj).
- Los dispositivos del bus interpretan las líneas del bus en los flancos de subida (comienzo del ciclo del bus).
- **Fase de direccionamiento**
    1. Maestro de bus identifica dispositivo destino (esclavo) y tipo de transacción
    2. Maestro de bus activa la seña $FRAME\#$
    3. Cada dispositivo conectado al bus PCI decodifica la dirección para determinar si pertenece a su espacio de direcciones. El dispositivo al que pertenece la dirección activa la señal $DEVSEL\#$
- **Fase de datos**
    1. Las señales $C/BE\#$ determinan el número de bytes habilitados para la transmisión en el bus de datos
    2. Las señales $IRDY\#$ y $TRDY\#$ controlan la transferencia. Los datos son transmitidos solo cuando ambas señales están activada.
- **Duración y finalización de la transacción**
    1. El maestro de bus mantiene la señal $FRAME\#$ activada desde el comienzo de la transacción hasta que está preparado para finalizar la fase de datos
    2. La finalización de la transacción se indica mediante la desactivación de $FRAME\#$ junto con la activación de $IRDY\#$
    3. Cuando el último dato ha sido transferido se libera el bus mediante la desactivación de $IRDY\#$

Transacción de lectura

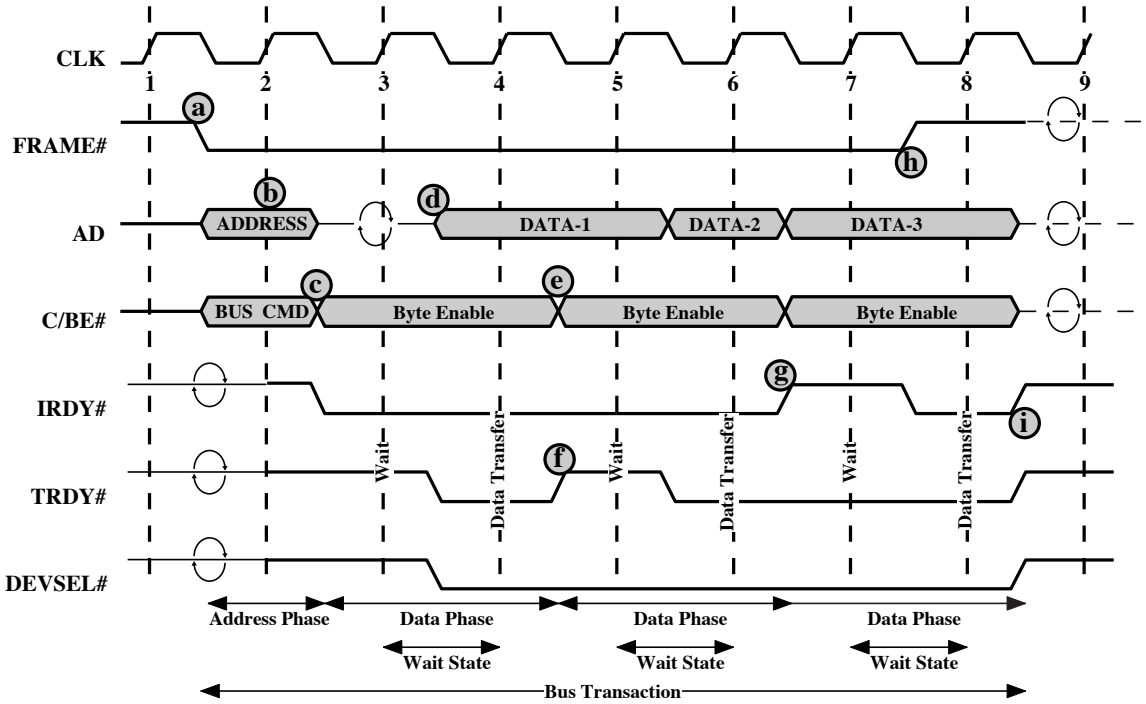


Figure 3.23 PCI Read Operation

a) El maestro del bus ha obtenido control del bus e inicia transacción activando $FRAME\#$. Esta línea permanecerá activa hasta que el maestro se disponga a finalizar la última fase de datos. El maestro sitúa la dirección de comienzo de lectura en el bus de direcciones y la orden de lectura en las líneas $C/BE\#$

b) A partir del comienzo del segundo ciclo de reloj el dispositivo del que se lee reconocerá su dirección en las líneas AD.

c) El maestro libera líneas AD. Ciclo de cambio para evitar contienda. Maestro cambia información en $C/BE\#$ para habilitar las líneas de AD a utilizar. También activa $IRDY\#$ para indicar que está preparado para recibir primer dato.

d) El dispositivo de lectura seleccionado activa $DEVSEL\#$ para indicar que ha reconocido la dirección y va a responder. Sitúa dato solicitado en AD y activa $TRDY\#$ para indicar que hay un dato válido en el bus.

e) El maestro lee el dato al comienzo del cuarto ciclo.

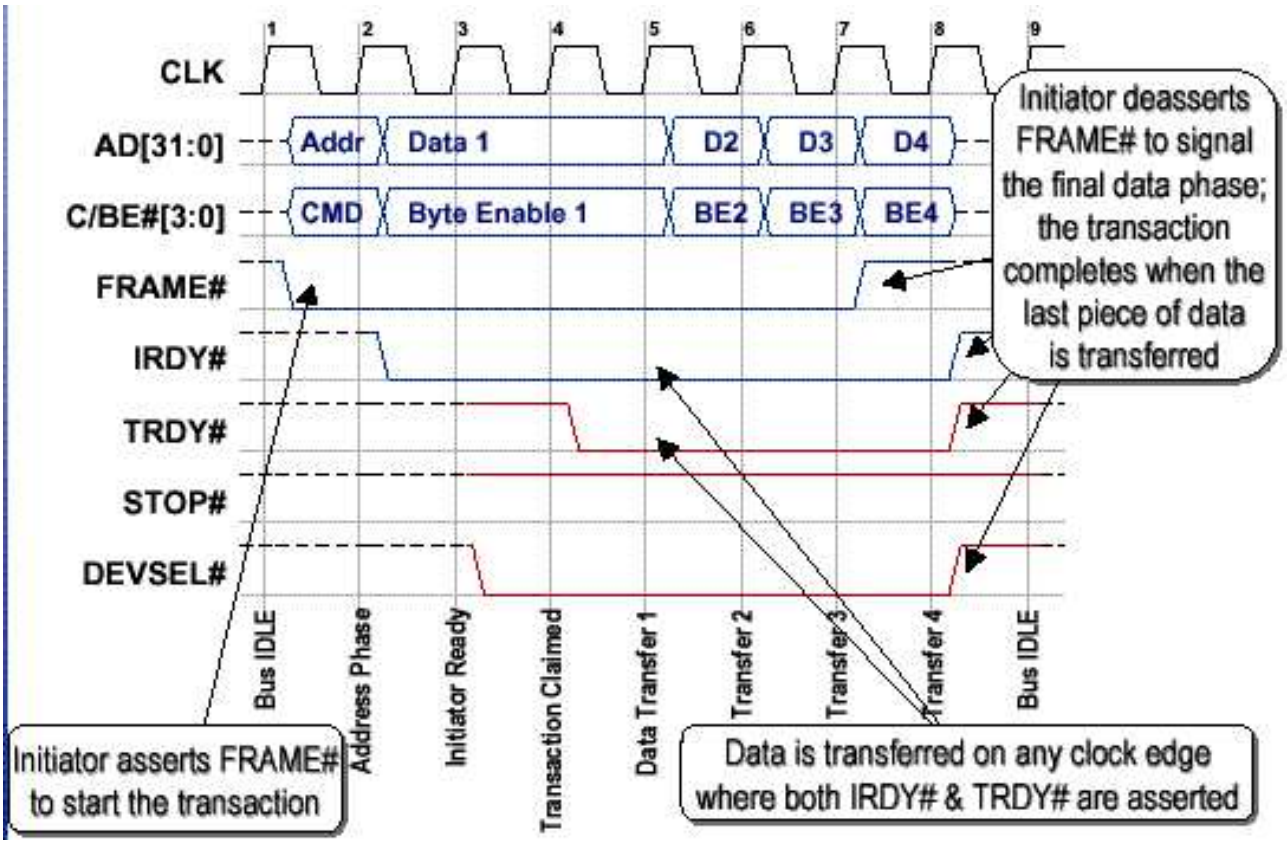
f) En este ejemplo dispositivo de lectura requiere tiempo para preparar segundo bloque de datos. Desactiva $TRDY\#$ para indicar a maestro que NO proporcionará nuevo dato en siguiente ciclo. Maestro no leerá bloque de datos hasta el sexto ciclo.

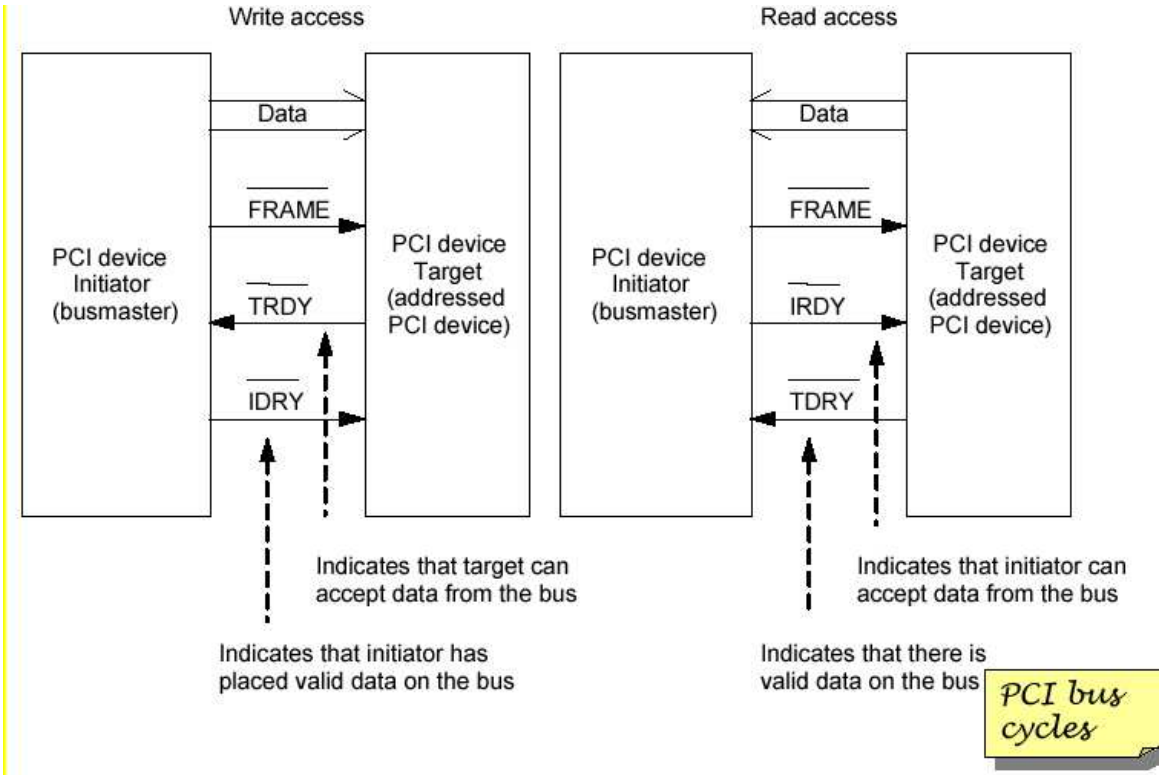
g) Dispositivo de lectura coloca tercer dato en el bus. Ahora es maestro el que no está preparado para leer el dato, desactivando $IRDY\#$.

h) El maestro desactiva $FRAME\#$, para indicar a dispositivo que es el último dato a transferir, y activa $IRDA\#$ para indicar que está listo para completar la transferencia.

i) El maestro desactiva $IRDY\#$, liberando el bus. El esclavo desactiva $TRDY\#$ y $DEVSEL\#$.

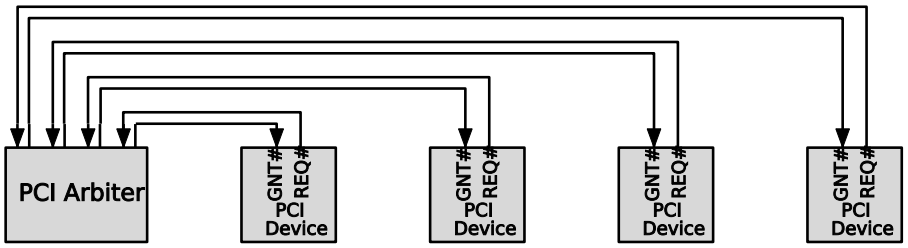
Transacción de escritura (4 palabras 32 bits)





Arbitraje del bus

Esquema de arbitraje **paralelo centralizado** síncrono:



El arbitraje se solapa con la comunicación del maestro de bus actual, de forma que no se pierden ciclos: **arbitraje de bus solapado u oculto**.
La especificación PCI no indica un algoritmo particular de arbitraje.

Ejemplo de procedimiento de arbitraje:

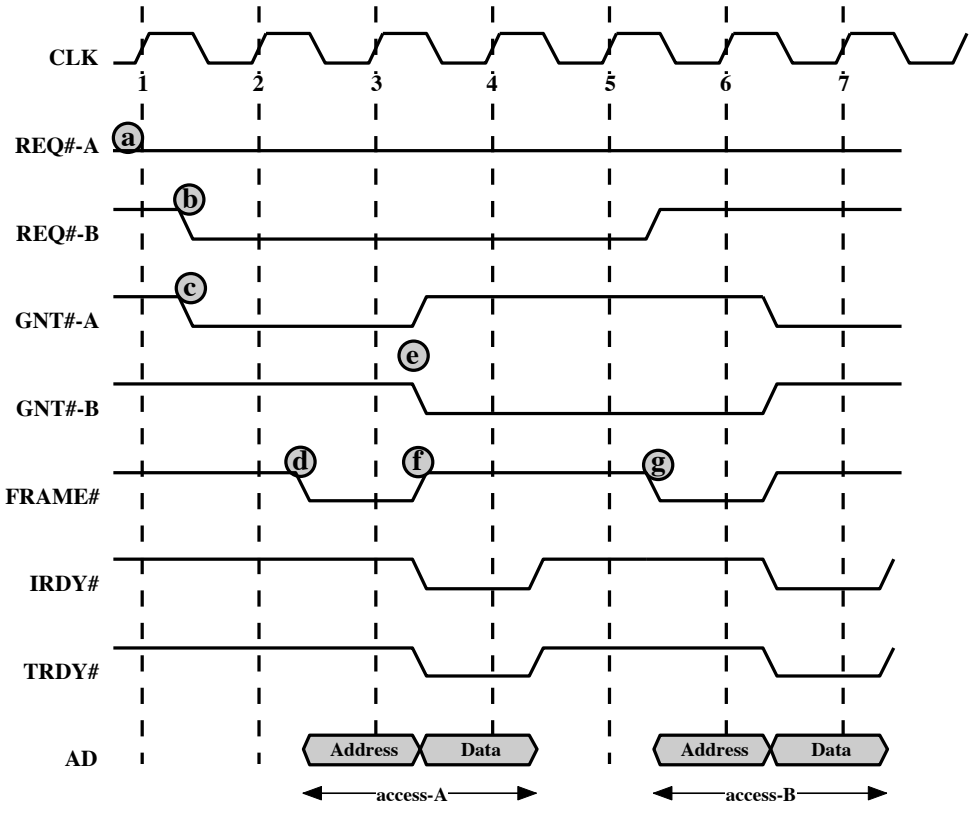
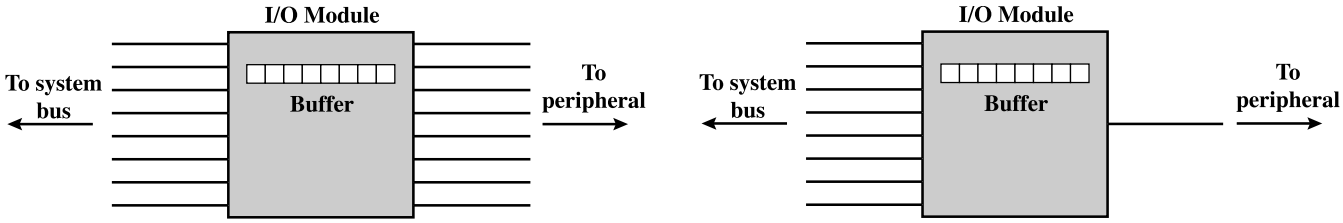


Figure 3.25 PCI Bus Arbitration Between Two Masters

La interfaz externa

Interfaz de comunicación **módulo E/S - periférico**

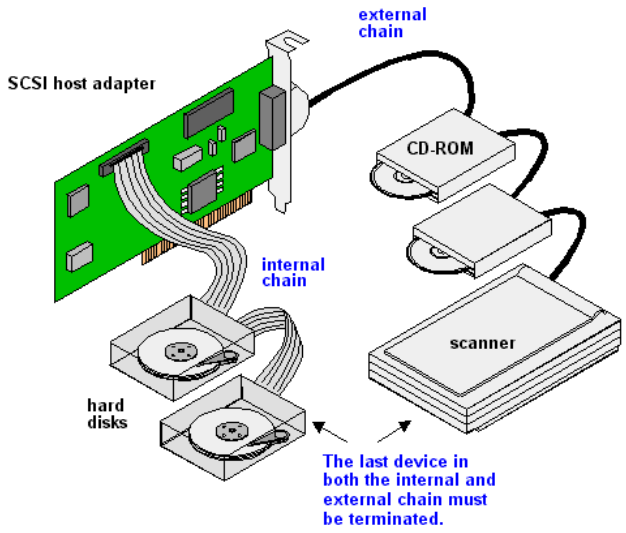
- Interfaz paralela vs. interfaz serie



- Conexión punto-a-punto vs. multipunto
  - Enlaces externos punto-a-punto típicos: teclado, puerto serie, puerto paralelo...
  - Enlaces externos multipunto: buses externos. Importancia creciente. Ejemplos: SCSI, USB y FireWire.

Interfaz SCSI (*Small Computer System Interface*)

- Diseñado para la conexión de múltiples periféricos a ordenadores personales, aunque también es ampliamente utilizado en servidores.
- Se trata de una interfaz paralela con, inicialmente, 8 líneas de datos y una velocidad de 5 MB/s (SCSI-1). Versiones posteriores aumentaron la velocidad, permitieron anchos de 8, 16 y hasta 32 líneas y aumentaron el número de dispositivos manejable (SCSI-2, Fast Wide SCSI, Ultra SCSI...).
- Modos de transferencia síncronos y asíncronos.
- Aunque utiliza estructura de bus y se suele considerar como tal, en el fondo se trata de una conexión encadenada de dispositivos (*daisy chain*).



Nuevos estándares en la interfaz externa

- El puerto serie tradicional, basado en el estándar $RS232$ y limitado a velocidades del orden de 115 Kb/s, se ha venido utilizando de siempre para la conexión de los periféricos más lentos.
- El puerto paralelo ha venido supliendo la necesidad de conexión de dispositivos algo más rápidos, aunque a costa de una mayor complejidad en el cableado.
- La industria ha acabado desarrollando nuevas interfaces serie más rápidas que presentan importantes ventajas en velocidad y abaratamiento de costes de frente a la mezcla existente de puertos serie, paralelo, de juegos, PS/2 y demás interfaces de E/S presentes en los equipos: *USB* y *FireWire*.
- Características:
  - Interfaz de conexión muy simple.
  - *Plug´n Play*: detección y configuración automática de los dispositivos conectados.
  - Conexión y desconexión *en caliente*.
  - Conector simple y robusto para todos los dispositivos.
  - Alimentación para los dispositivos incluida (solo dispositivos de bajo consumo).

USB (***Universal Serial Bus***)

- Se trata de un **eje raíz** que se conecta al bus principal.
- Se permiten ramificaciones del eje principal mediante *hubs*: total de 127 dispositivos manejables en una topología de estrella.
- Hasta 5 metros de longitud por segmento de cable.
- 1,5 Mb/s, 12 Mb/s, 480 Mb/s (USB 2.0).

FireWire ($IEEE$ 1394)

- Tasa de transferencia de hasta 400 Mb/s en su especificación inicial (IEEE 1394b hasta 800 Mb/s).
- Hasta 63 dispositivos en un único bus.
- Mayor rendimiento bruto que USB: se utiliza principalmente para dispositivos de almacenamiento y vídeo.
- Permite trabajar sin *host* y comunicaciones *peer-to-peer*.
- Hasta 100 metros en un único segmento.

AGP (*Accelerated Graphics Port*)

- Canal de comunicación punto-a-punto de 32 bits de alta velocidad para tarjetas gráficas (no es un medio compartido)
- Desarrollado por Intel en 1996 como solución al cuello de botella que el bus PCI suponía para las tarjetas gráficas, cada vez más demandantes de ancho de banda
- Su diseño parte de la especificación PCI 2.1, y se lleva a cabo teniendo en cuenta exclusivamente al hardware gráfico
- Su arquitectura inicial permitía una única ranura de conexión (AGP 1.0 y AGP 2.0). Posteriores revisiones eliminaron esta limitación (AGP 3.0)

PCI Express (PCIe)

- Pensado para reemplazar a PCI, PCI-X y AGP, al proporcionar una interfaz de comunicación para dispositivos de E/S más rápida
- Su diseño se basa en un sistema de comunicación serie con varios enlaces llamados canales (*lanes*). Con PCIe 1.1 cada canal tiene un ancho de banda de 250MB/s full duplex, con hasta 32 canales (hasta 8 GB/s totales)
  -Nomenclatura número de canales: x16 = 16 canales
- Mantiene compatibilidad software con PCI, cambiando solo la capa física
- Conexiones punto a punto (red de conmutación en lugar de medio compartido), permitiendo full duplex
- La red de canales serie, dinámicamente conmutados, permite varias comunicaciones al mismo tiempo, frente a la aproximación de medio compartido de un bus tradicional. Además, los canales se pueden agrupar para aumentar el ancho de banda
- Rendimiento aproximado: un canal da casi el doble de AdB que PCI, una configuración con 4 canales se equipara a PCI-X más rápido, y 8 canales al AGP más rápido
- De momento sus características y velocidad todavía no le permiten sustituir al bus de memoria o soportar comunicación entre procesadores