

Entrada/salida y buses

- Funciones del sistema de E/S. Interfaces de E/S
- E/S programada
- Interrupciones
- DMA (Acceso directo a memoria)
- **Estructuras de bus básicas**
- Especificación de un bus. Transferencias. Temporización. Arbitraje
- Ejemplos y estándares

Estructuras de bus básicas

- **Un bus se utiliza para conectar dos o más elementos de un sistema digital.**
 - Es un **conjunto de líneas compartidas** por esos elementos del sistema y usadas para la **comunicación** entre ellos.
- **Un sistema de buses es un conjunto de buses usado para conectar los distintos elementos de un ordenador.**
- **El término “bus” suele implicar comunicaciones paralelas**
 - Varias señales viajan al mismo tiempo.
 - Sin embargo, esto puede simularse por una secuencia de señales en una única conexión: bus **serie**.
- **Los buses pueden adoptar diferentes formas y tamaños:**
 - interconexión de componentes en una tarjeta.
 - interconexión de tarjetas
 - interconexión de periféricos, etc.

Bus único

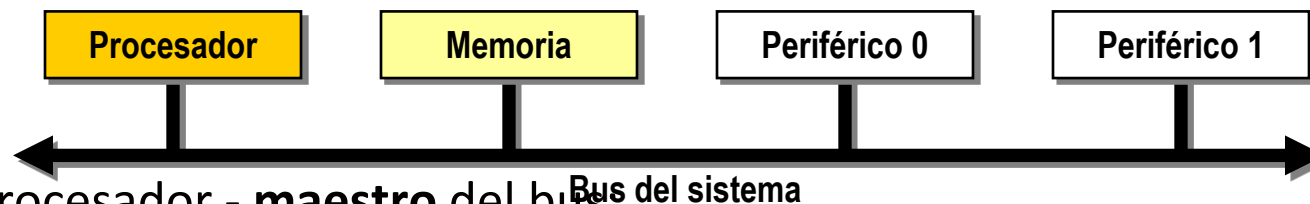
■ Necesidad:

El procesador necesita conectarse a otros elementos para funcionar:

- Memoria para programas / datos.
- E/S para almacenamiento / comunicación usuario.

■ Solución:

- Usar **las mismas líneas** para conectar todo.



- Procesador - **maestro** del bus.
 - Inicia todas las transferencias.
 - Controla si un dispositivo...
 - lee del bus.
 - escribe en el bus.
 - triestado.

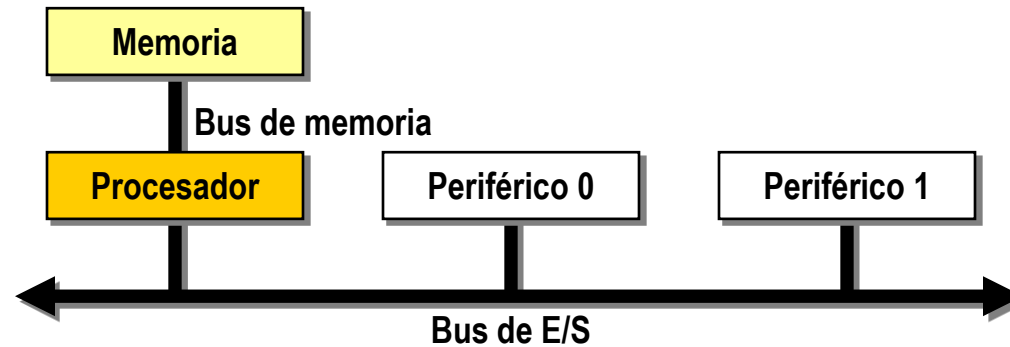
Bus único

■ Problemas:

- Cortocircuito:
 - Sólo un dispositivo puede escribir en un instante dado.
- Velocidad transferencia:
 - Procesador > Memoria >> Periféricos
 - Si el bus es síncrono:
 - Todas las transferencias duran el mismo tiempo.
 - ✗ O el bus (y procesador y memoria) va a la velocidad del más lento,
 - ✗ ...o no se pueden conectar periféricos lentos.

Buses separados

- Uno rápido (memoria) y otro lento (E/S)

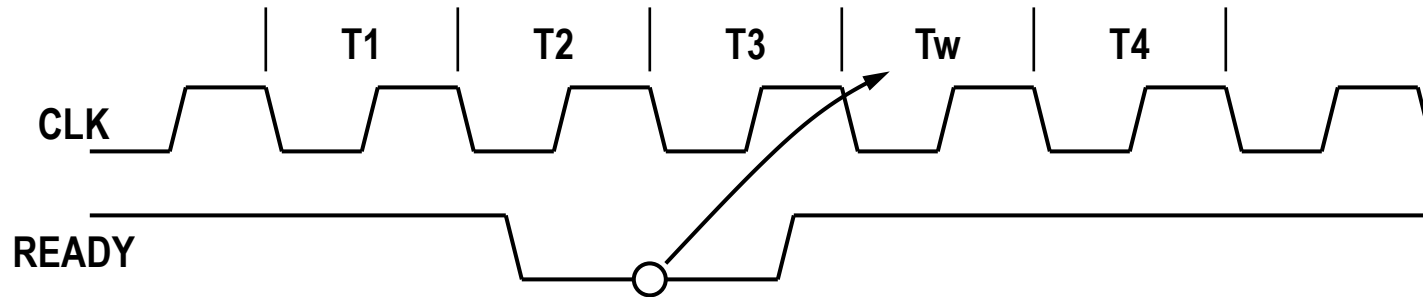


- Bus memoria va a la velocidad de la memoria.
- ✗ Bus E/S va a la velocidad del más lento.

Buses separados

■ Mejora: asincronismo bus E/S.

- Requiere una línea de control adicional READY.

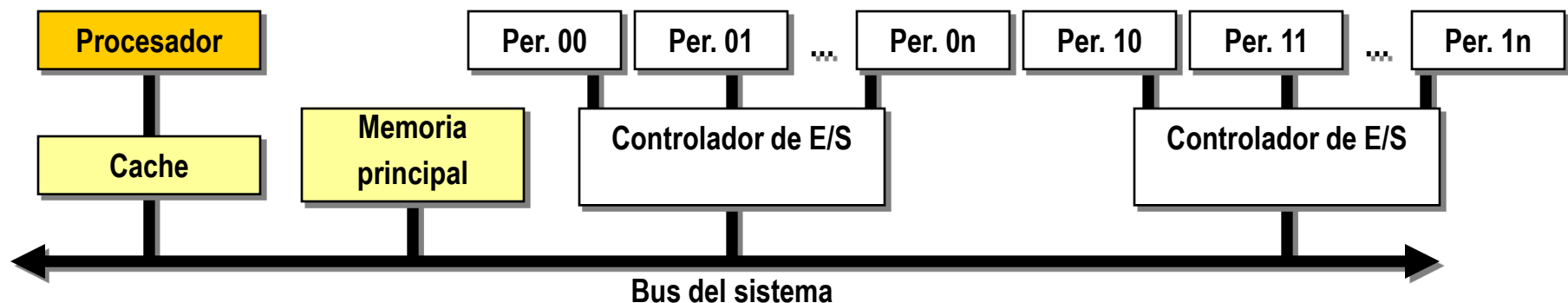


- Procesador intenta transferir en T3.
- Periférico no está listo aún, READY↓.
 - Procesador inserta estados Tw (wait).
- Periférico preparado ya, READY↑.
 - Procesador no inserta más Tw, pasa a T4.
 - Se transfiere el dato.

Bus único avanzado

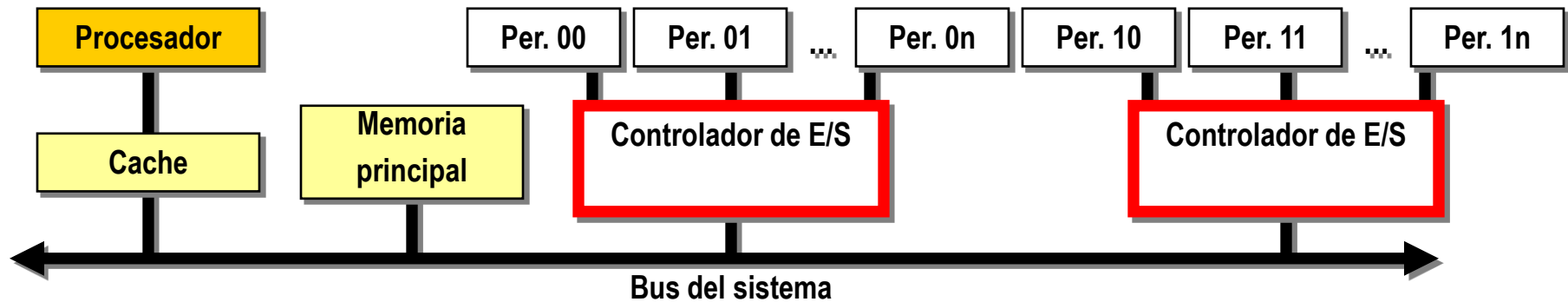
■ Objetivo:

- Que el procesador pueda seguir trabajando con la memoria mientras que los periféricos terminan su operación.



Bus único avanzado

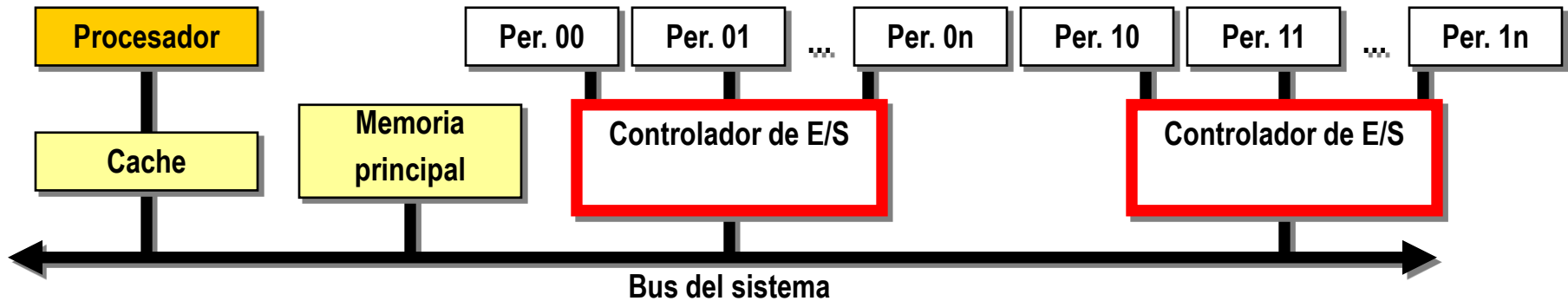
■ Controlador de E/S:



- Procesador programa controlador indicando:
 - Tipo de transferencia (R/W)
 - Periférico (0...n)
 - Tamaño del bloque de datos
- Procesador sigue trabajando con la memoria.

Bus único avanzado

■ Controlador de E/S:



- Controlador interrumpe a procesador cuando periférico preparado
 - Pide (W) el primer dato, o
 - proporciona (R) el primer dato.
- Procesador escribe / lee dato E/S.
- Procesador sigue trabajando hasta próxima interrupción.

Bus único avanzado

■ Buffers:

- Controlador puede disponer de un pequeño bloque de memoria propia (por ej. 1 KB).
- Procesador puede escribir / leer datos E/S de 1 KB en 1 KB.
 - Se interrumpe menos frecuentemente (1024 veces menos).
 - Es más rápido transferir una vez 1 KB que 1024 veces 1 B.
- Periféricos también pueden tener buffer propio.
 - Controlador queda libre para manejar otro periférico mientras tanto.

Bus único avanzado

■ Interrupciones:

- Mecanismo (hardware, línea INTR) por el cual el procesador
 - memoriza (normalmente en la pila) contexto actual,
 - contador de programa
 - indicadores de estado
 - otros registros
 - ejecuta Rutina de Servicio de Interrupción (ISR) muy breve,
 - retorna a contexto anterior.

Bus único avanzado

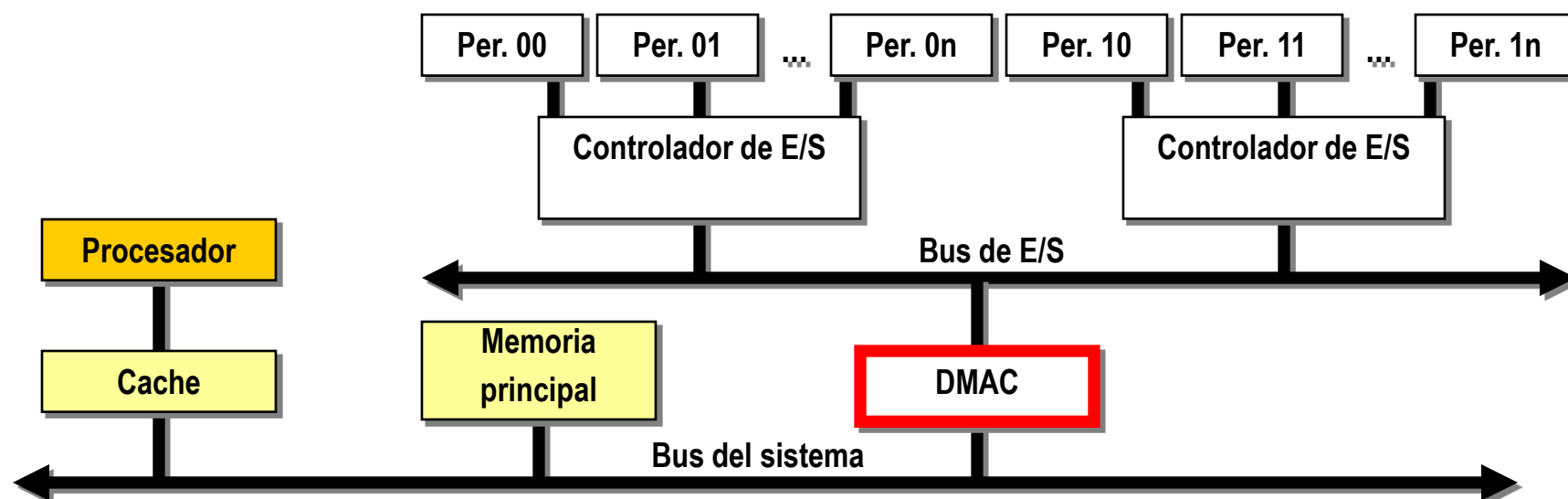
■ Interrupciones:

- ISR debe
 - identificar dispositivo,
 - interrupciones vectorizadas
 - » controlador pone vector en bus,
 - » vector identifica periférico e ISR.
 - consulta (polling) y tabla de saltos
 - » ISR va comprobando estado periféricos
 - » hasta localizar el causante de la interrupción.
 - atenderlo (R/W),
 - atender controlador (reprogramarlo).

Buses separados avanzados

■ Objetivo:

- Liberar procesador de tráfico E/S ↔ memoria.



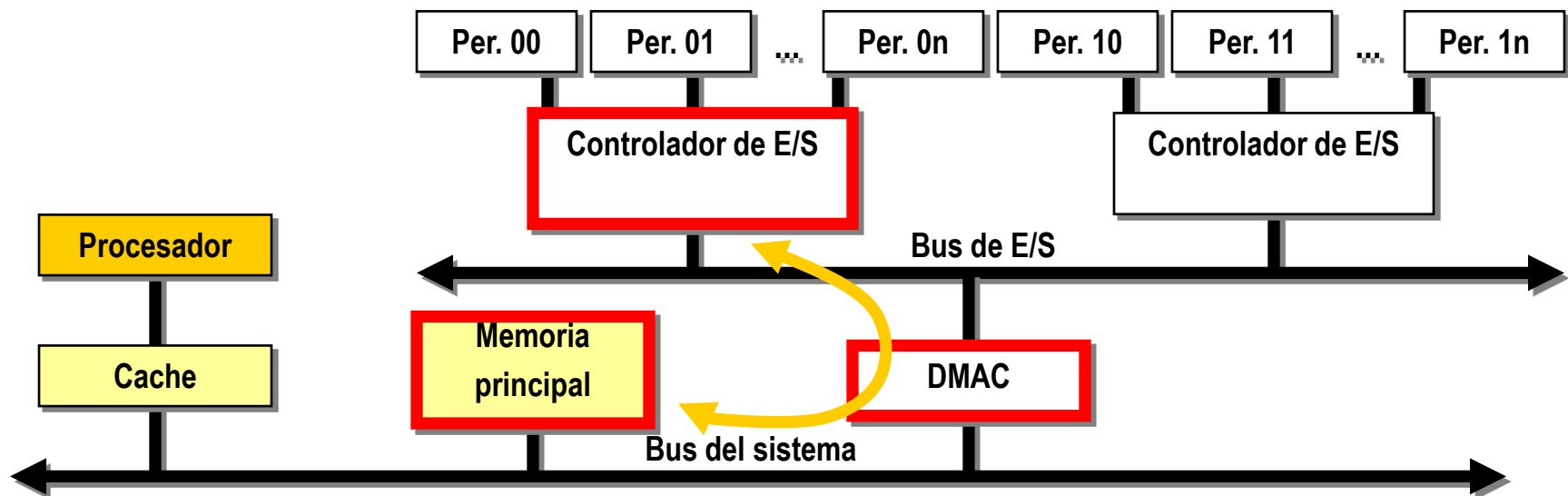
■ Controlador de DMA (DMAC):

- Procesador programa DMAC indicando
 - tipo de operación (R / W),
 - tamaño del bloque de memoria,
 - dirección inicial de memoria.

Buses separados avanzados

■ Controlador de DMA (DMAC):

- Controlador de E/S debe programarse para actuar conjuntamente.
- DMAC aprovecha cuando bus del sistema está libre...
 - Lee de memoria y escribe dato en bus de E/S, o escribe en memoria dato leído de bus de E/S.
 - Incrementa dirección y decrementa contador.
 - Interrumpe al procesador al acabar.



Buses separados avanzados

■ Controlador cache:

- Cache es una memoria más **rápida, cara y pequeña** que la MP.
- **Principio de localidad** de las referencias a memoria:
 - *Temporal*:
 - Tendencia procesador a referenciar dentro de poco datos referenciados hace poco.
 - *Espacial*:
 - Tendencia a referenciar datos cercanos al último referenciado.
- El controlador...
 - intenta que procesador encuentre en cache todo lo que necesita.
 - transfiere memoria \leftrightarrow cache en **pequeños bloques**.

Otras estructuras de bus

■ Debido a:

- Deseo de reducir el **tiempo** de ejecución.
- **Coste** cada vez menor de los μ procesadores.

■ Surgen sistemas con:

- Varios procesadores iguales
 - Repartir tiempo CPU.
 - Incrementar tolerancia a fallos.
- Coprocesadores específicos (gráficos, matemáticos).
- Todo trabajando simultáneamente en paralelo.

■ Viable sólo si:

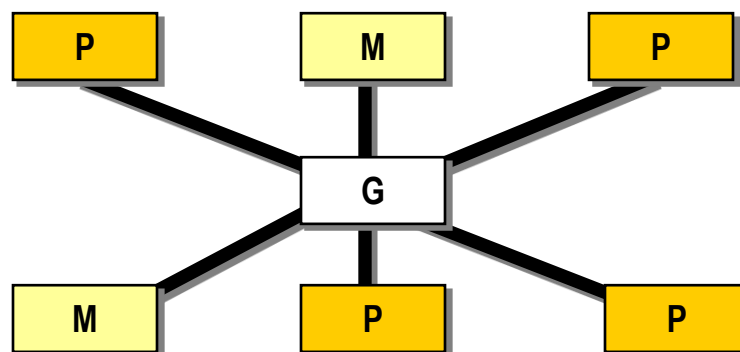
- Cada uno puede trabajar bastante tiempo aisladamente.
- Acceden ocasionalmente al bus para transferir ráfagas de datos.

Otras estructuras de bus

■ Tres estructuras típicas:

■ Estrella:

- Interconexión procesadores ↔ memoria compartida.

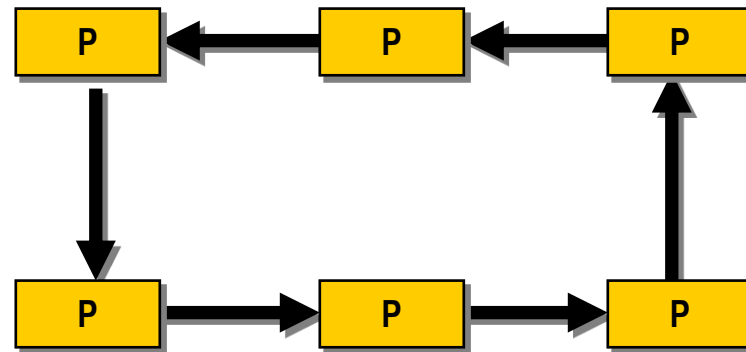


- Módulo central G (Gestor de comunicaciones)
 - Sistema de conmutadores (barras cruzadas, barril, etc.)
- ✓ Alta velocidad.
- ✗ Número de procesadores limitado por capacidad de G.
- ✗ Avería en G inutiliza el sistema.

Otras estructuras de bus

▪ Anillo:

- Interconexión procesadores mediante paso de testigo.



- Cada procesador...
 - Espera testigo,
 - lee mensajes de otros procesadores,
 - retira los que le corresponden,
 - añade los que desea transmitir,
 - escribe mensajes a siguiente procesador.
- ✓ Barato (2 hilos, coaxial, altas velocidades).
- ✓ Sencillo, ampliable (insertar P en cadena).
- ✗ Avería de un procesador inutiliza el sistema.

Otras estructuras de bus

- **Bus único multiplexado** en el tiempo:
 - Opción escogida por la mayoría de fabricantes y la mayoría de estándares.
 - ✓ Barato.
 - ✗ Relativamente lento.

Entrada/salida y buses

- Funciones del sistema de E/S. Interfaces de E/S
- E/S programada
- Interrupciones
- DMA (Acceso directo a memoria)
- Estructuras de bus básicas
- **Especificación de un bus. Transferencias. Temporización. Arbitraje**
- Ejemplos y estándares

Especificación de un bus

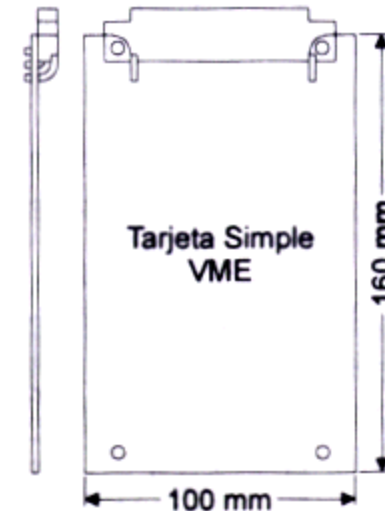
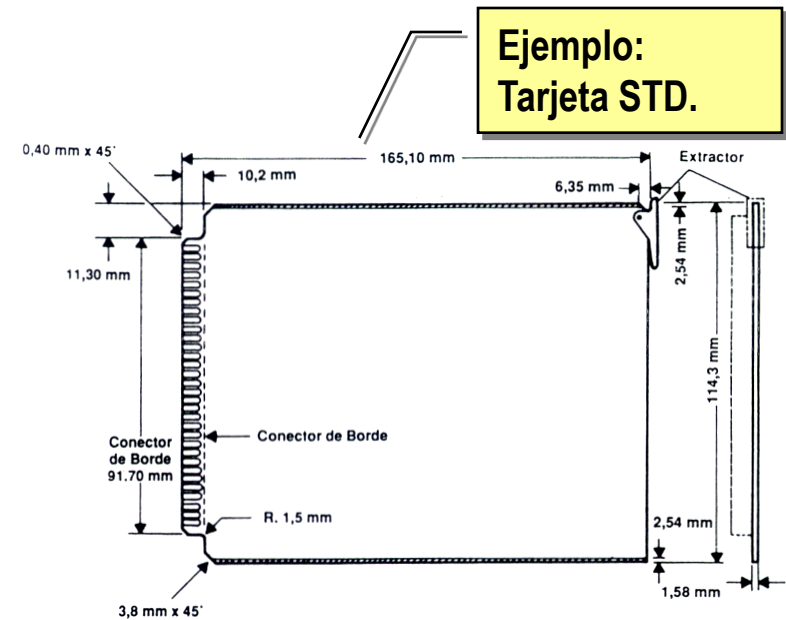
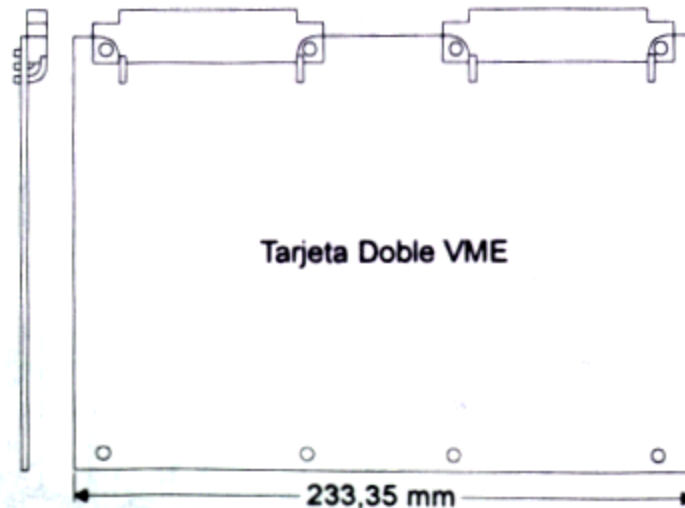
- **Debe incluir toda la información necesaria para saber conectar un dispositivo al bus, a varios niveles:**
 - Mecánico.
 - Eléctrico.
 - Lógico.
 - Temporización.
 - Transferencia simple.
 - Transferencia de un bloque.

Nivel mecánico

- Soporte (rack, PCB)
- Nº de líneas
- Tipo de conector
- Dimensiones de las tarjetas

...

Ejemplo:
Tarjeta bus VME.

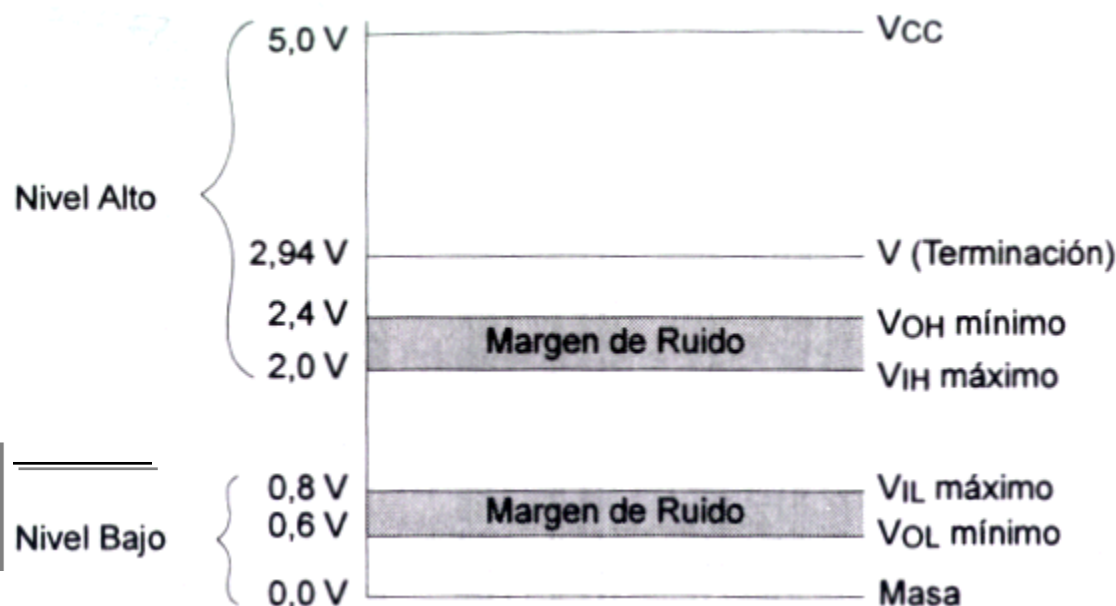


Nivel eléctrico

- Alimentación
- Impedancias
- Nivel de señal

...

Ejemplo:
Niveles bus VME.

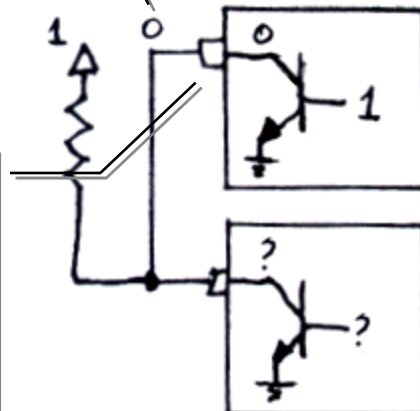


- Muy relacionado con el nivel mecánico, ambos influyen en:
 - Distancia máxima conexión / nº de tarjetas conectables
 - Inmunidad al ruido
 - Velocidad máxima de transferencia
 - Paralelismo / multiplexación
 - Coste (cables, conectores, racks, ...)

Nivel lógico

- Nº de señales
- Función de cada señal.
- Activas en alta / baja.
- Señales triestado.
- Señales en colector abierto ("open collector")

... $0 \text{ AND } ? = 0$



Las señales en colector abierto permiten conexión directa formando una AND cableada.

Ejemplo:
Señales bus STD.

| Contacto | Nombre | Descripción |
|----------|-----------------------|---|
| 31 | \overline{WR} | Escritura en memoria o E/S [†] |
| 32 | \overline{RD} | Lectura en memoria o E/S [†] |
| 33 | \overline{IORQ} | Selección de dirección de E/S [†] |
| 34 | \overline{MEMRQ} | Selección de dirección de memoria [†] |
| 35 | \overline{IOEXP} | Expansión de dirección de E/S |
| 36 | \overline{MEMEX} | Expansión de dirección de memoria |
| 37 | $\overline{REFRESH}$ | Temporización de refresco [†] |
| 38 | \overline{MCSYNC} | Ciclo de sincronización de la CPU [†] |
| 39 | $\overline{STATUS1}$ | Estado de la UCP [†] |
| 40 | $\overline{STATUS0}$ | Estado de la UCP [†] |
| 41 | \overline{BUSAK} | Cesión del bus por parte de la UCP |
| 42 | \overline{BUSRQ} | Petición del bus [†] |
| 43 | \overline{INTAK} | Aceptación de la interrupción |
| 44 | \overline{INTRQ} | Petición de interrupción [†] |
| 45 | \overline{WAITRQ} | Petición de tiempo de espera [†] |
| 46 | \overline{NMIRO} | Petición de interrupción no enmascarable [†] |
| 47 | $\overline{SYSRESET}$ | Reset del sistema [†] |
| 48 | $\overline{PBRESET}$ | Reset de interruptor [†] |
| 49 | \overline{CLOCK} | Reloj de la UCP |
| 50 | \overline{CNTRL} | Temporización auxiliar |
| 51 | PCO | Salida de la cadena de prioridad <i>daisy-chain</i> |
| 52 | PCI | Entrada de la cadena de prioridad <i>daisy-chain</i> |
| 53 | AUX GND | Masa auxiliar |
| 54 | AUX GND | Masa auxiliar |
| 55 | AUX +V | +12V |
| 56 | AUX -V | -12V |

[†] Triestado

[‡] Colector abierto

Nivel de temporización

Tipos de ciclo

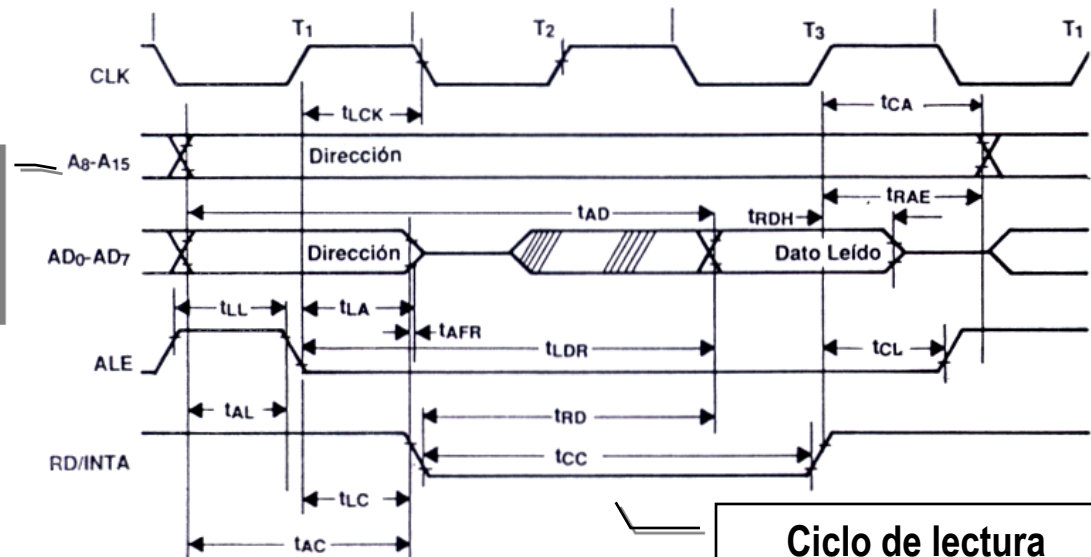
- R/W
- IO/M
- TW

Ejemplo:
Cronogramas
8085.

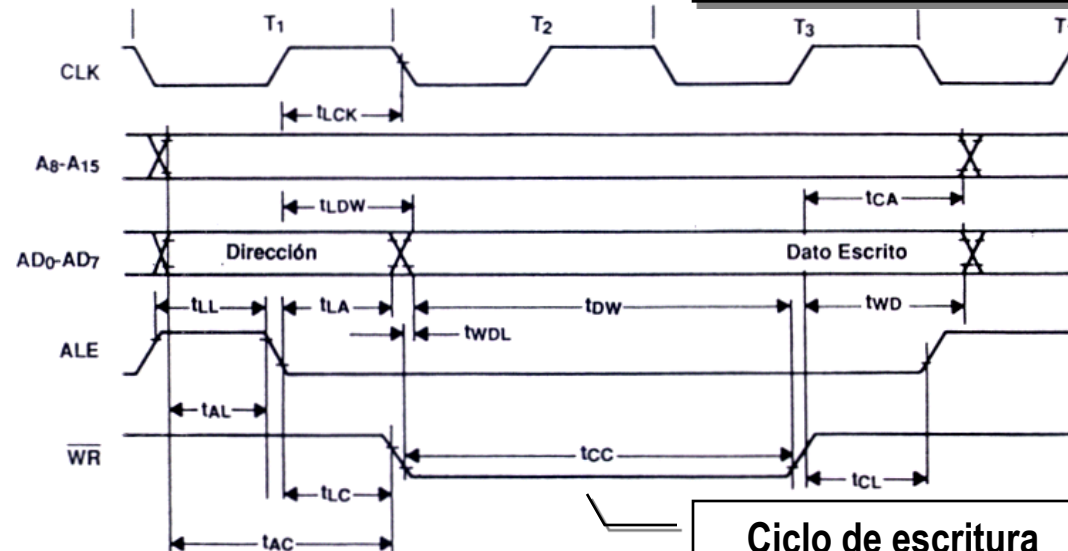
Cronogramas.

| | | | |
|-----|------------------------|---------------------------|-----|
| ... | $t_{LA} \rightarrow$ | $(1/2) \cdot T - 60$ | MIN |
| | $t_{LCK} \rightarrow$ | $(1/2) \cdot T - 60$ | MIN |
| | $t_{AD} \rightarrow$ | $(5/2 + N) \cdot T - 225$ | MAX |
| | $t_{RAE} \rightarrow$ | $(1/2) \cdot T - 10$ | MIN |
| | $t_{DW} \rightarrow$ | $(3/2) \cdot T - 60$ | MIN |
| | $t_{CC} \rightarrow$ | $(3/2 + N) \cdot T - 80$ | MIN |
| | $t_{ARY} \rightarrow$ | $(3/2) \cdot T - 260$ | MAX |
| | $t_{HABF} \rightarrow$ | $(1/2) \cdot T + 50$ | MAX |
| | $t_{AC} \rightarrow$ | $T - 50$ | MIN |
| | $t_2 \rightarrow$ | $(1/2) \cdot T - 40$ | MIN |
| | $t_{LDR} \rightarrow$ | $2 \cdot T - 180$ | MAX |
| | $t_{AL} \rightarrow$ | $(1/2) \cdot T - 45$ | MIN |
| | $t_{LL} \rightarrow$ | $(1/2) \cdot T - 20$ | MIN |
| | $t_{LC} \rightarrow$ | $(1/2) \cdot T - 30$ | MIN |
| | $t_{RD} \rightarrow$ | $(3/2 + N) \cdot T - 180$ | MAX |
| | $t_{CA} \rightarrow$ | $(1/2) \cdot T - 40$ | MIN |
| | $t_{WD} \rightarrow$ | $(1/2) \cdot T - 60$ | MIN |
| | $t_{CL} \rightarrow$ | $(1/2) \cdot T - 110$ | MIN |
| | $t_{HACK} \rightarrow$ | $(1/2) \cdot T - 50$ | MIN |
| | $t_{HABE} \rightarrow$ | $(1/2) \cdot T + 50$ | MAX |
| | $t_1 \rightarrow$ | $(1/2) \cdot T - 80$ | MIN |
| | $t_{RV} \rightarrow$ | $(3/2) \cdot T - 80$ | MIN |

T periodo de
reloj
 N nº de
periodos de
espera



Ciclo de lectura



Ciclo de escritura

Niveles de transferencia

■ Nivel de transferencia simple

- Protocolos de
 - arbitraje
 - transmisión
 - ciclo partido
 - detección de errores
- ...

■ Nivel de transferencia en bloque

- Protocolos de transferencia en bloque.
- Mecanismo de reintento / recuperación de bloque.
- ...

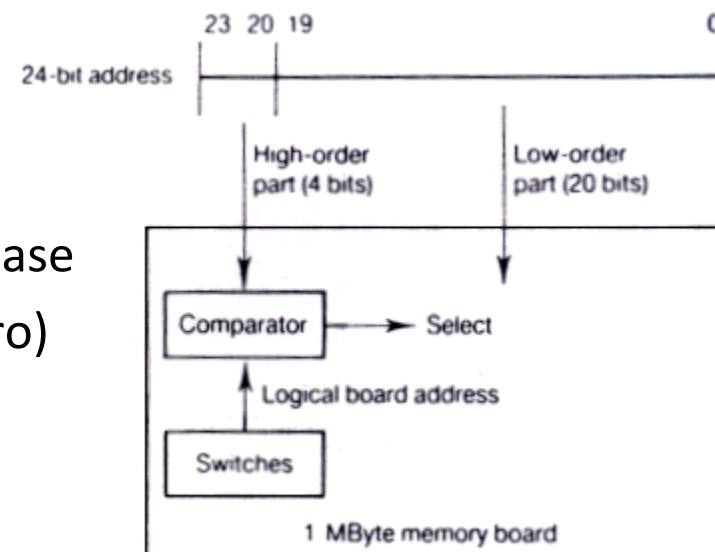
Direccionamiento

- **Una vez en posesión del bus, el maestro debe establecer comunicación con el esclavo**
 - seleccionarlo, ponerlo en funcionamiento (CS)
- **Cada tarjeta tiene...**
 - una **dirección base** (única, que la distingue del resto de tarjetas)
 - un **rango de direcciones** por el que se reparten sus elementos direccionables (memoria, registros de control/estado, registros de E/S...)
- **Según método de selección:**
 - Direccionamiento lógico
 - Direccionamiento geográfico
- **Según nº de esclavos:**
 - Direccionamiento normal
 - Direccionamiento extendido

Direccionamiento lógico / geográfico

■ Direccionamiento lógico

- Las tarjetas tienen
 - **conmutadores manuales (*switches*) o registro programable**
 - **comparadores**
- El operador (o el S.O.)
 - diseña el mapa de direcciones
 - qué tarjeta va en qué dirección base
 - configura los conmutadores (o registro)
- Ejemplo:
 - Bus de direcciones de 24 bits.
 - Todas las tarjetas tienen 4 conmutadores.
 - Hasta 16 tarjetas. $2^{20} = 1 \text{ MB} / \text{tarjeta}$.
 - Direcciones base: 000000, 100000, 200000, ..., E00000, F00000.
- No todas las tarjetas tienen que tener el mismo nº de conmutadores



Direccionamiento lógico / geográfico

■ Direccionamiento geográfico (Fastbus, Multibus II, NuBus)

- Las **ranuras** donde se insertan las tarjetas **tienen cableado un número**, en orden secuencial (*slot number*).
- El nº de ranura (posición de la tarjeta en el *rack*) determina la base y el rango de direcciones.
- Normalmente los buses con direccionamiento geográfico **pueden conmutar a direccionamiento lógico**:
 - Direccionamiento geográfico se usa durante inicializ. sistema
 - S.O. chequea todas las tarjetas / excluye las que fallan
 - Las tarjetas tienen un **registro base programable**
 - equivalente a *switches*.
 - **S.O. reparte mapa de direcciones** entre tarjetas
 - S.O. conmuta a direccionamiento lógico
 - S.O. puede volver a direccionamiento geográfico y reconfigurar sistema de nuevo

Direccionamiento normal / extendido

■ Direccionamiento normal

- Sólo un esclavo responde al maestro.

■ Direccionamiento extendido

- Varios esclavos (incluso todas las tarjetas).
- BROADCAST:
 - Escritura simultánea en varias tarjetas.
- BROADCAST:
 - Lectura simultánea de varias tarjetas.
 - OR cableado, o
 - multiplexado en el tiempo: las tarjetas responden una tras otra (Ej.: Ethernet).
- *Broadcast/call* **universal**:
 - Se dirige a todas las tarjetas.

Direccionamiento normal / extendido

- Dos formas de implementación del direccionamiento extendido:
 - **Dirección reservada:** 000...0h ó FFF...Fh
 - Leer de la dirección provoca *broadcast*.
 - Escribir en dirección provoca *broadcast*.
 - **Línea de control *broad*:**
 - Cuando se activa, indica operación a/de varios esclavos.
 - El bus de direcciones se utiliza para especificar subconjunto de esclavos.

Direccionamiento normal / extendido

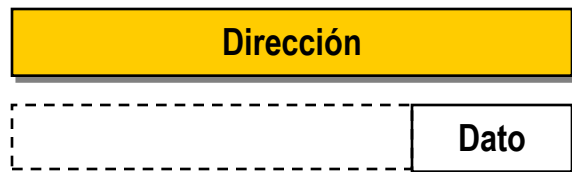
- Ejemplos de uso del direccionamiento extendido:
 - **Coherencia de cache** (*broadcast*).
 - Sistema multiprocesador. Cada procesador con cache propia. Datos compartidos.
 - Procesador modifica su cache. Debe avisar a otros procesadores y actualizar memoria compartida.
 - » Escribe simultáneamente en la cache propia, en memoria compartida, y en cache de los otros.
 - **Arbitraje distribuido** (*broadcast/call* simultáneos).
 - *cast*: cada árbitro escribe AP# para que el resto lo lea.
 - *call*: cada árbitro lee OR-lógico de varios AP# simultáneos.
 - **Lectura del estado de las IRQ** (*broadcastcall*).
 - A cada interrupción puede asignarse una posición (bit).
 - *Broadcastcall* 32 bits implicando a 32 tarjetas.

Tipos de transferencias

■ Según contenido:

■ Lectura

- Maestro requiere dato de esclavo.
- Esclavo escribe dato cuando se active (tiempo de acceso).



- No es grave multiplexar el bus:



■ Escritura

- Maestro proporciona datos a esclavo.
- Esclavo lee datos cuando se active (tiempo de selección).



Tipos de transferencias

- Modificación (*read-modify-write*)
 - Maestro lee e inmediatamente escribe. Indivisible.
 - Ej.: operación TEST&SET



- Comprobación (*read-after-write*)
 - Maestro escribe e inmediatamente lee.
 - Útil para chequeo.



- DMA (*three-party*)
 - Transferencia a tres partes.
 - El maestro (DMAC) no es fuente ni destino.
 - Hay dos esclavos.

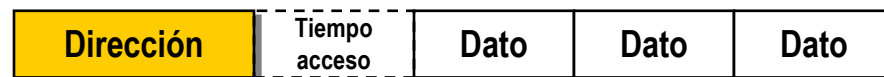
Tipos de transferencias

■ Según tamaño:

- Vacía (*dummy*).
 - No se transfieren datos, sólo se direcciona el esclavo.
 - Ej.: refresco de memorias dinámicas.
- Palabra (*single*).
- Bloque o ráfaga (múltiple, *burst*).
 - Suponer bus multiplexado direcciones / datos, y lectura de un bloque 1KB desde posiciones consecutivas.
 - *Single*: repetir 1024 transferencias completas.



- *Burst*: transferencia normal seguida de 1023 datos contiguos.

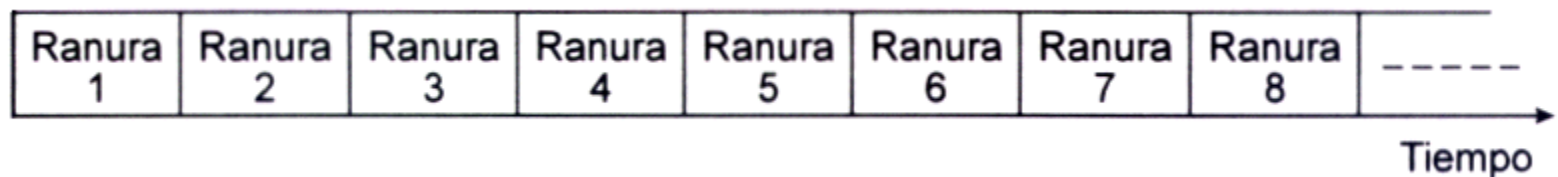


- ✓ Ventajoso sobre todo en buses multiplexados (ahorro de 1023 direcciones).
- ✗ Bloque de tamaño grande impide usar el bus a otros

Tipos de transferencias

■ Según procedimiento:

- Conectada (ciclo completo).
- Partida (ciclo partido).
 - Se divide el tiempo en slots (ranuras):



- La secuencia “petición-transferencia realizada” está partida.
- El maestro envía la petición “orden+dirección” en una ranura y libera el bus.
- Cuando el esclavo está listo, termina la transferencia en otra ranura.
- Durante el tiempo intermedio el bus está libre para ser utilizado por otro dispositivo.

Tipos de transferencias

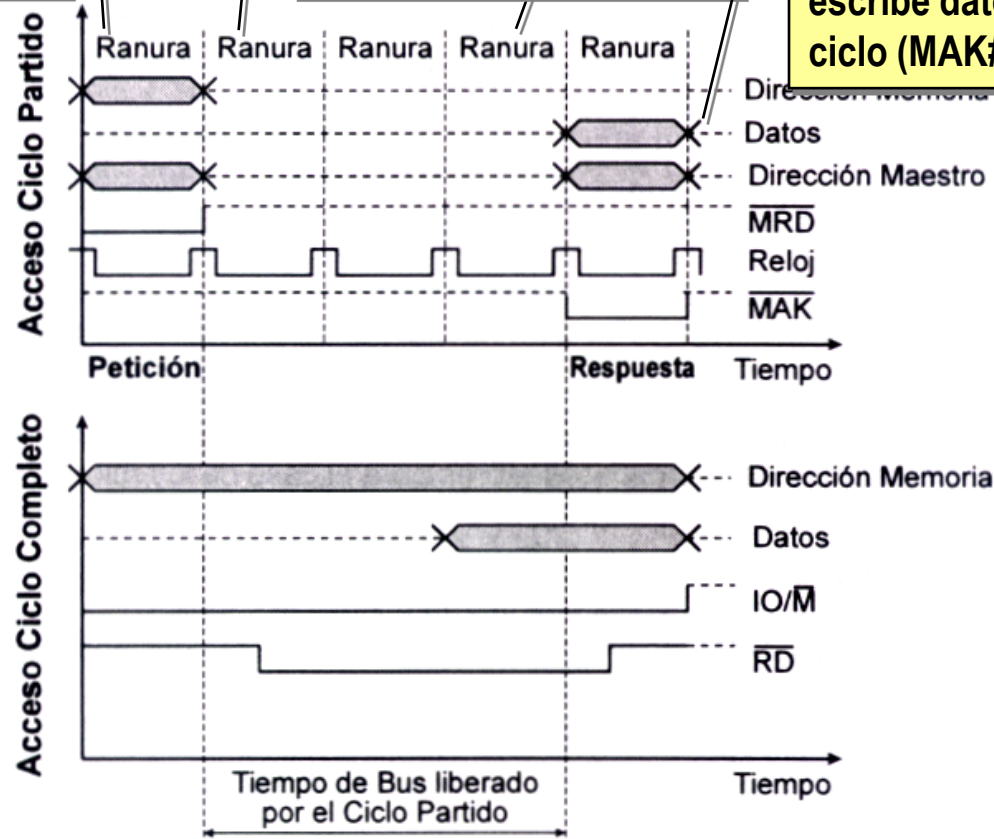
- Ejemplo: Lectura de memoria:

Maestro direcciona a esclavo y libera bus.

Otros maestros pueden usar el bus.

Esclavo es lento. Transcurre tiempo de acceso.

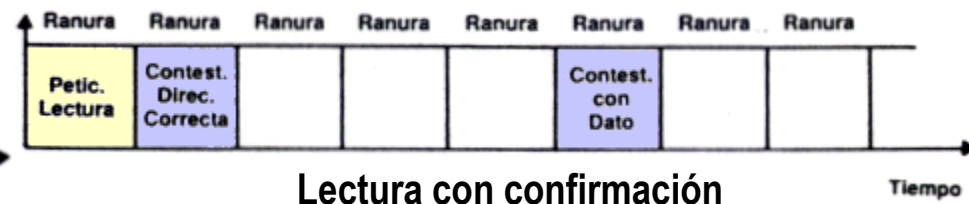
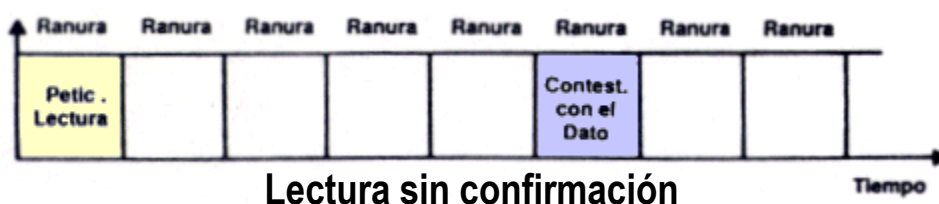
¡Esclavo direcciona maestro!, escribe datos, y reconoce fin de ciclo (MAK#)



Tipos de transferencias

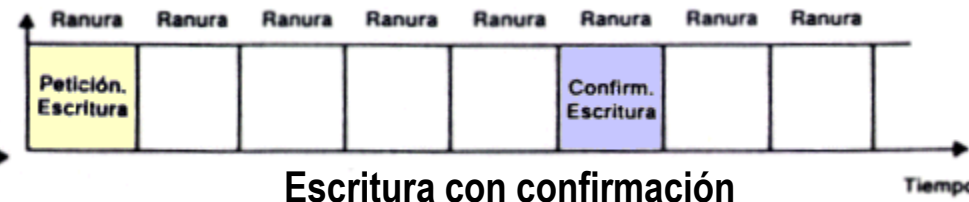
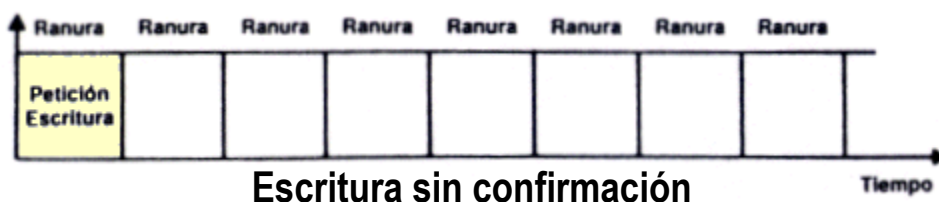
▪ Lectura:

- El dispositivo (lento) hace “*latch*” de la dirección.
- Responde una vez transcurrido el tiempo de acceso.
- Debe direccionar al maestro (maestro ha de leer dato).



▪ Escritura:

- El dispositivo hace “*latch*” de dirección/dato.
- No tiene mucho sentido hablar de escritura sin confirmación partida.



Temporización

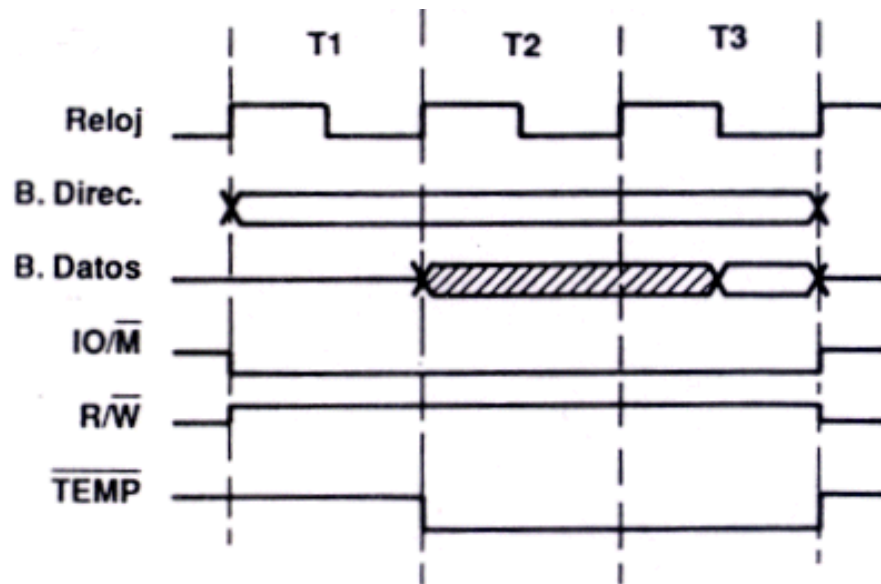
■ Síncrona sin confirmación

- Línea de reloj CLK común a todos los dispositivos.
- Protocolo de comunicación fijo relativo al reloj:
 - Transferencias en ciclo/flanco predeterminado de reloj.
- No hay mecanismo por el cual el maestro pueda recibir reconocimiento.
 - Maestro asume que todos los esclavos funcionan correctamente y a velocidad adecuada.
- ✓ Circuitería de control mínima, permite alta velocidad.
- ✓ Inmunidad al ruido (el bus se muestrea en flancos predeterminados de reloj) en comparación con asíncrona.
- ✗ Todos los dispositivos han de funcionar a la misma frecuencia de reloj: bus debe ir a la velocidad del más lento (o el dispositivo no reaccionará a tiempo).
- Típica en buses procesador-memoria.

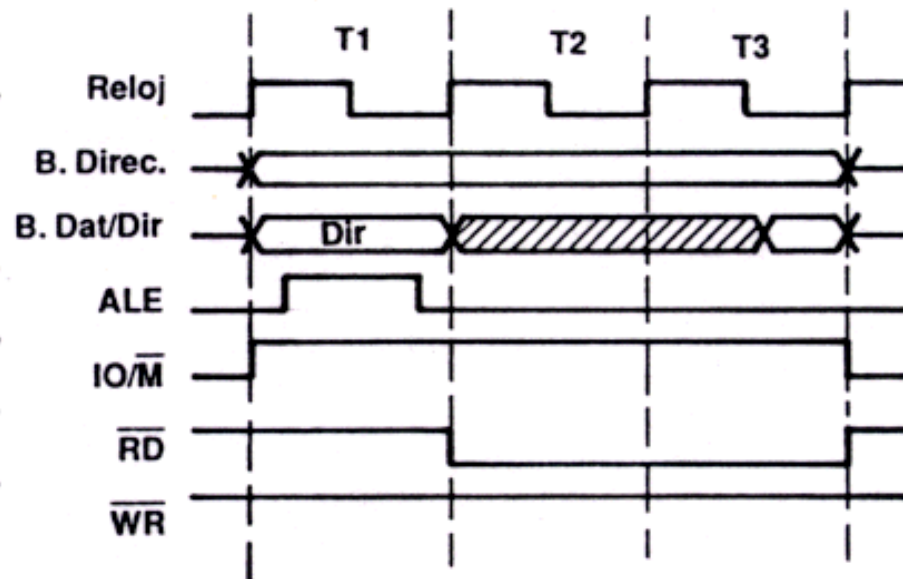
Temporización

■ Lectura síncrona:

- Maestro proporciona dirección en T1.
- Maestro lee datos en T3.
- ✗ Esclavo debe proporcionar datos en T3. Si no, dato = basura.



Ejemplo de lectura síncrona de memoria

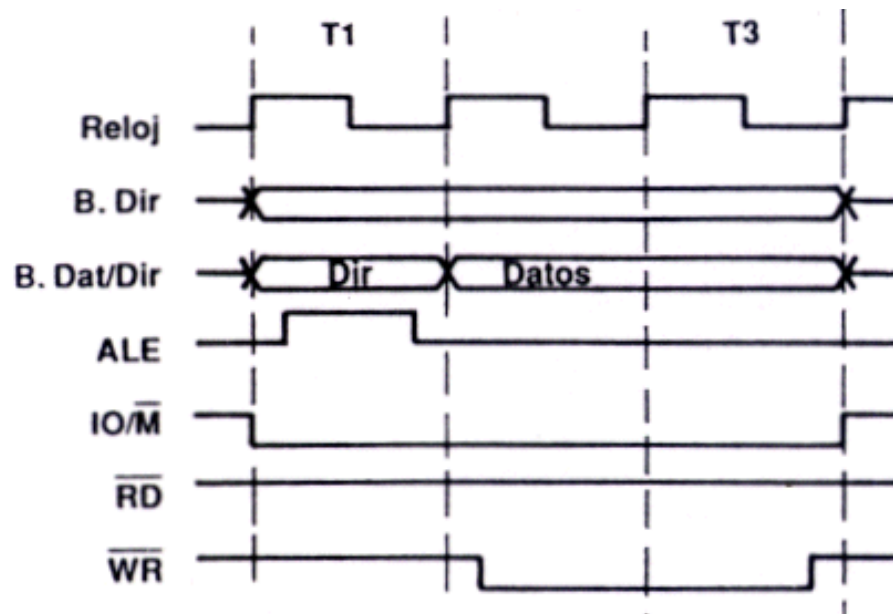


Ejemplo de lectura síncrona de E/S

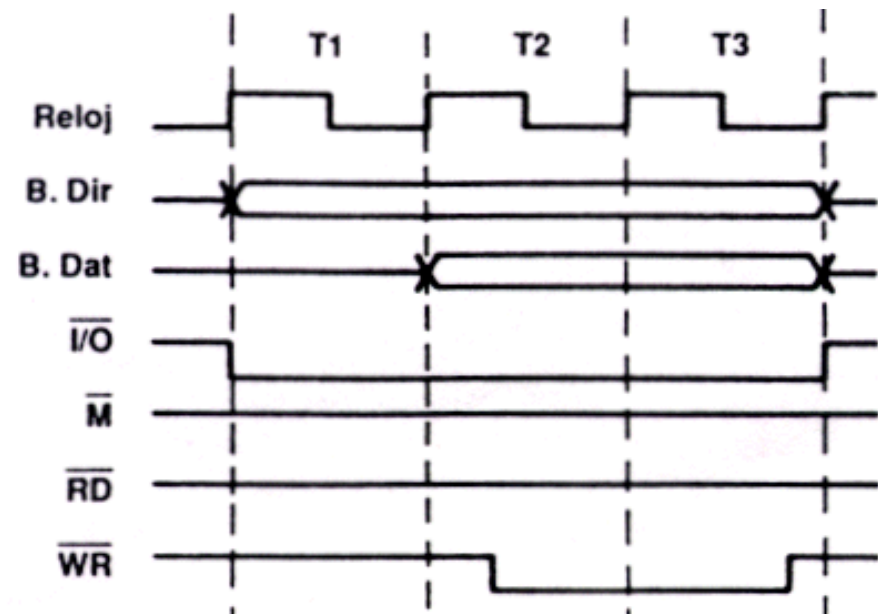
Temporización

■ Escritura síncrona:

- Maestro proporciona dirección en T1.
- Maestro escribe datos en T2 y los mantiene hasta fin de T3.
- ✗ Esclavo debe leer datos después de estabilizados en bus jantes de que acabe T3!. Si no, dato perdido.
- ✗ Maestro no puede garantizar que transferencia correcta.



Ejemplo de escritura síncrona en memoria

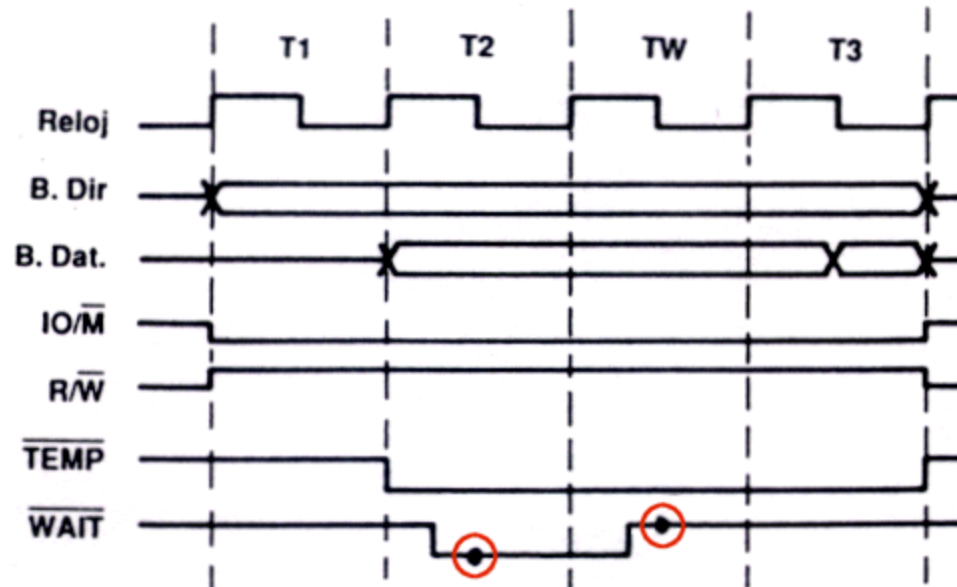


Ejemplo de escritura síncrona en E/S

Temporización

■ Síncrona con confirmación

- Síncrona, con una **línea adicional RDY** (o WAIT).
 - Maestro inserta **estados de espera** hasta que el esclavo reacciona.
 - ✓ Circuitería sencilla.
 - ✓ Inmunidad al ruido.
 - ✗ Tiempo uso del bus incrementado en múltiplos de ciclo.
- } en comparación con asíncrona.



Temporización

■ Asíncrona

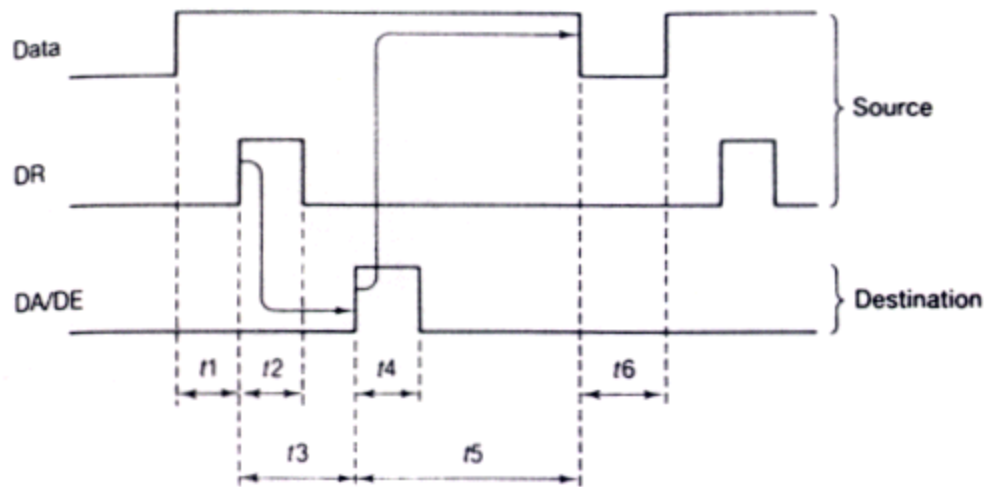
- No existe reloj común CLK.
- Conversación maestro/esclavo mediante “*handshake*”.

Handshake \equiv estrechar la mano (ofrece 1º, ofrece 2º, agitar, retira 1º, retira 2º)

- ✓ Permite conexión de periféricos a cualquier velocidad.
- ✗ Circuitería más compleja \Rightarrow coste \uparrow .

Temporización

- Asíncrona parcialmente interbloqueada:
 - Para simplificar circuitería $\rightarrow t_2, t_4 \equiv$ intervalos fijos de tiempo

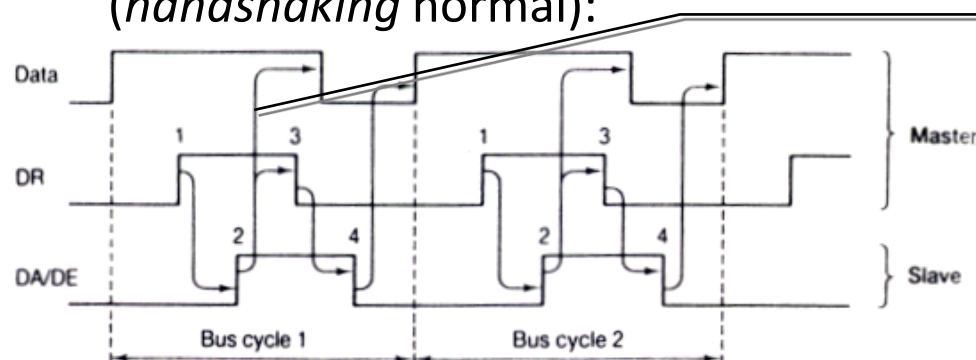


1. Fuente pone dato, estabiliza (t_1).
2. **Fuente señala dato listo ($DR \uparrow$) mediante pulso de ancho fijo t_2 .**
3. Destino tarda t_3 en ser seleccionado y almacenar el dato.
4. **Destino señala dato leído ($DA \uparrow$) o error ($DE \uparrow$) mediante pulso de ancho fijo t_4 .**
5. Fuente tarda t_5 en advertir señal de dato leído y quitar datos.
6. t_6 : bus sin uso, arbitraje, etc.

- ✓ t_3 bajo control de destino: podría ser arbitrariamente lento.
- ✓ t_5 bajo control de fuente: podría ser arbitrariamente lento.
- ✗ t_4 fijo: si fuente muy rápido, puede iniciar siguiente ciclo antes de que acabe destino ($t_4 > t_5 + t_6$).
- ✗ t_2 fijo: si destino muy rápido, inicia siguiente ciclo antes de que acabe fuente (si $t_2 > t_3 + t_4$, destino vuelve a ver DR a 1).

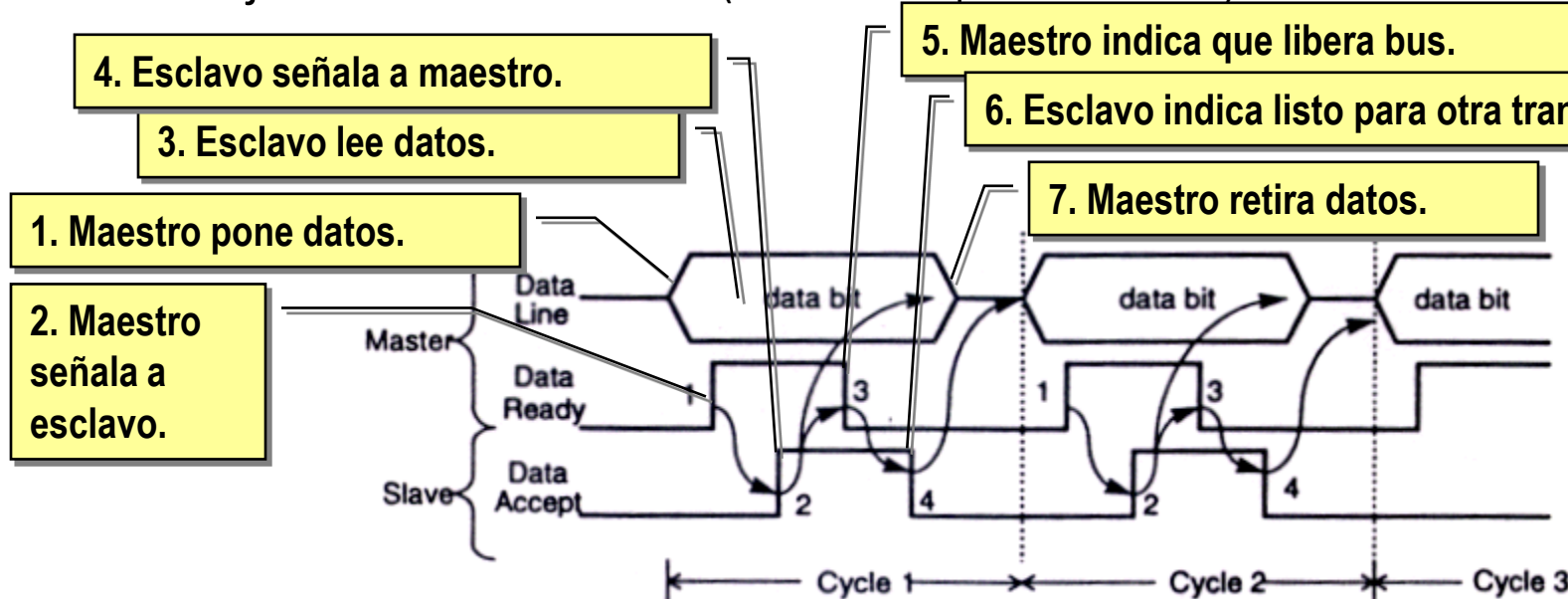
Temporización

- Asíncrona completamente interbloqueada, a 4 flancos (*handshaking normal*):



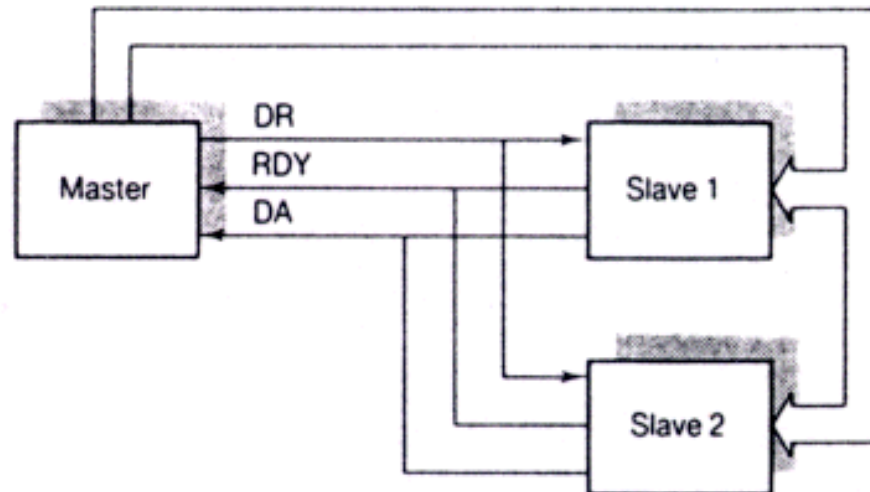
Cada paso no puede comenzar hasta que se haya completado el paso anterior (señalado mediante flancos 1 a 4).

- Ej.: Escritura asíncrona (con interbloqueo a 4 flancos):



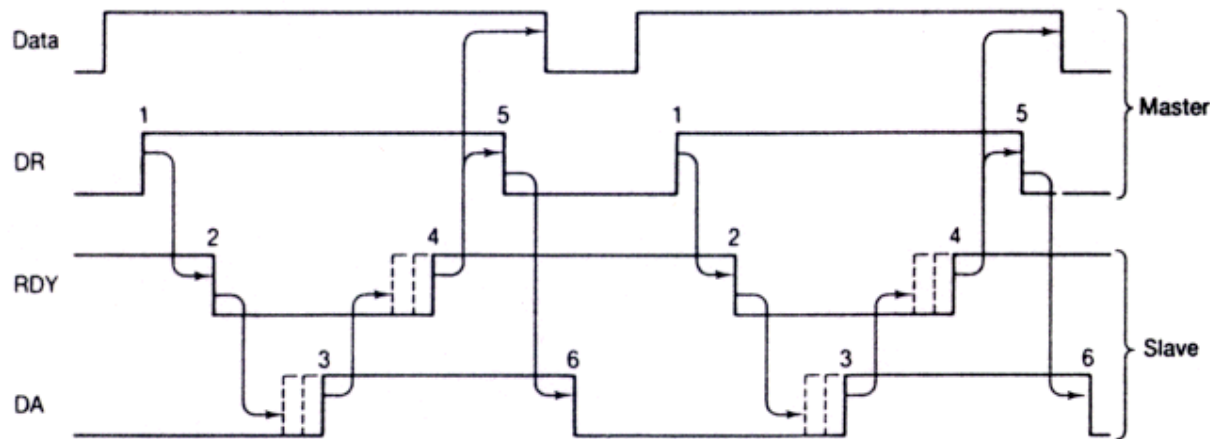
Temporización

- Asíncrona completamente interbloqueada, a 6 flancos:
 - Ej.: Escritura en Bus HP-IB o GPIB (IEEE-488):
 - Posibilidad de escritura en varios esclavos.
 - » $DAV \equiv Data\ Available$ ($DR \equiv Data\ Ready$)
 - Cada esclavo genera dos señales de *handshake*:
 - » $DAC\ (DA) \equiv Data\ Accepted$: se almacenó el dato.
 - » $RFD \equiv Ready\ For\ Data$ ($RDY \equiv Ready$): listo para nueva transferencia.



Temporización

- Debido a que hay varios esclavos.
 - hay que distinguir inicio DA (DA) y fin DA (RDY).
 - obliga a usar colector abierto.
 - permite DA↑ y RDY↑ sólo cuando último esclavo.



1. Maestro pone dato: DR↑.
2. En cuanto 1er. esclavo reacciona, RDY↓.
3. 1er. esclavo termina: activa DA pero colector abierto.
4. Último esclavo termina: DA↑.
5. 1er. esclavo libera bus: activa RDY pero colector abierto.
6. Último esclavo libera: RDY↑.
7. Maestro libera bus: DR↓.
8. Esclavos esperan datos: DA↓.

Concepto de arbitraje

■ Tipos de dispositivos (o tarjetas)

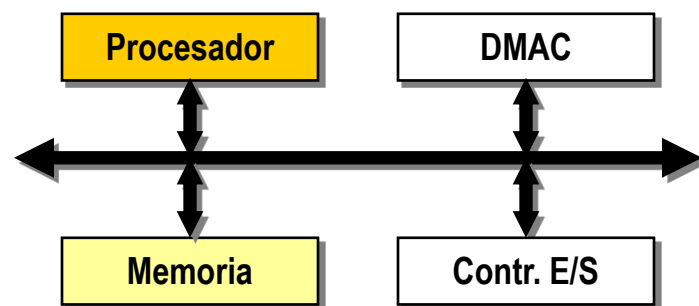
- Activos:
 - Eventualmente pueden requerir el uso del bus para iniciar una transferencia.
- Pasivos:
 - Sólo pueden responder a una transferencia, nunca iniciarla.
- Maestro (*Master, Initiator*):
 - En cada transferencia el bus es gobernado por un maestro que envía direcciones e información de control.
- Esclavo (*Slave, Target*):
 - Dispositivo(s) que responde(n) al maestro.

Pasivo → esclavo siempre

Activo → maestro ocasionalmente

Concepto de arbitraje

■ Ejemplo: Bus único, DMA (robo de ciclo).



Activos:

Procesador, DMAC

Pasivos:

Memoria, E/S

(E/S sólo puede interrumpir)

Maestro:

normalm. procesador

Esclavo:

normalm. memoria

■ Controlador de E/S

- dispone de un buffer de datos.
- interrumpe a procesador cuando buffer lleno.

■ DMAC

- roba bus (HOLD ↑).
- transfiere E/S → memoria.

Maestro: DMAC.

Esclavos: Mem, E/S.

Fuente: E/S.

Destino: Mem.

■ Procesador

- recupera bus (HOLD ↓).
- lee de memoria.

Maestro: Proces.

Esclavo: Mem.

Fuente: Mem.

Destino: Proces.

Concepto de arbitraje

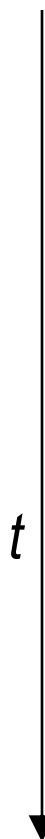
■ Con bus único:

- Situación de “lucha” por el bus (*bus contention*):
 - Activos deben acordar quién usa el bus. Si no \Rightarrow **cortocircuito**.
- Tipos de bus único:
 - Maestro único (procesador):
 - No hay lucha por el bus.
 - DMA (robo de ciclo):
 - Hay un elemento activo que suele ser el maestro (proces.).
 - Hay otro activo (DMAC) que ocasionalmente pide el bus por muy poco tiempo (HOLD-HLDA).
 - No hay arbitraje: proc. cede el bus, y lo recupera pronto.
 - **Arbitraje** (p. ej. bus PCI):
 - **Varios elementos activos.**
 - **Requiere protocolo justo** (no dejar esperando eternam.).

Concepto de arbitraje

■ Secuencia de operaciones de una transacción en el caso más general (arbitraje):

- **Petición bus**: Activos envían petición para controlar el bus.
- **Arbitraje**: Se decide el ganador (maestro). Varias técnicas.
- Direccionamiento.
 - Maestro envía direcciones en el bus.
 - Esclavo es seleccionado tras estabilización bus direcciones.
- Transferencia de datos.
 - Maestro activa señales de temporización y sentido de la transferencia. Ej. R/W#
 - Fuente pone datos en el bus de datos.
 - Destino toma los datos cuando están estabilizados.
- Liberación.
 - Destino indica que ha tomado los datos.
 - Maestro libera el bus. Varias técnicas.
- Detección de errores.



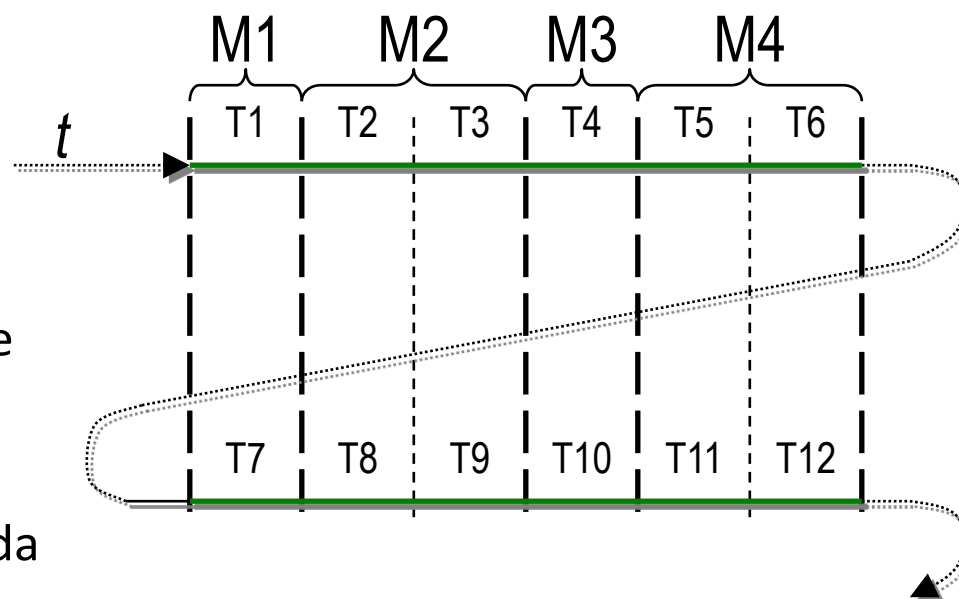
Políticas de arbitraje

- **Estática**
- **Dinámicas:**
 - de petición:
 - FIFO
 - Prioridad fija
 - Equidad
 - Combinada
 - de liberación:
 - ROR
 - RWD
 - *Pre-emption*

Políticas de arbitraje

■ Estática:

- Repartir turnos (*transaction slots*) entre maestros, de manera prefijada.
- Ejemplo: 4 tarjetas activas, potencialmente maestros.
 - A M1 y M3 se les asigna 1 slot o ranura temporal.
 - A M2 y M4 se les asignan 2 slots.
- Temporización síncrona (duración T fija).
- ✓ Esquema sencillo de implementar.
- Requiere calcular velocidad de transferencia pico para cada maestro, para asignar *slots*.
- ✗ Desperdicia *slots* si no hay nada que transmitir
⇒ ancho de banda del bus ↓.



Políticas de arbitraje

■ Dinámicas:

- Permiten cambiar el maestro según la situación en cada instante (requieren hardware apropiado).
- De petición:
 - Cómo decidir, de entre las tarjetas activas que solicitan el bus, el siguiente maestro.
 - **FIFO** (*First-in, First-out*):
 - Se concede el bus a quien lleva más tiempo pidiéndolo.
 - **Prioridad:**
 - Cada tarjeta tiene una prioridad. La petición de mayor prioridad gana.

...

Políticas de arbitraje

- **Equidad:**

- Garantizar que no se concede al mismo dos veces habiendo otra petición pendiente.
- Ej.: “*Round-robin*”: ir concediendo en ronda (entre los que lo soliciten).
- Evita “*starvation*” (privación o inanición) \equiv si máxima prioridad solicita mucho, restantes prioridades mueren de hambre.

- **Combinada:**

- Típico en sistemas multiprocesador | E/S.
- Usar política de prioridad para tareas ocasionales y de duración corta.
- Usar política de equidad para tareas normales.
- Ej.: E/S se atienden inmediatamente (si hay varias, en orden de prioridad). Procesadores se atienden cíclicamente mientras no haya E/S.

Políticas de arbitraje

- De liberación:

Cómo decidir cuándo deja libre el bus el maestro actual.

- **ROR**, *Release On Request* (Liberar cuando haya otra petición):

- Típico en sistemas monoprocesador.

- » Procesador posee el bus casi siempre

- » Sólo cede el bus si robo de ciclo DMA.

- ✓ Ahorra tiempo de arbitraje: procesador accede frecuentemente sin tener que competir.

...

Políticas de arbitraje

- **RWD, Release When Done** (Liberar al acabar):
 - Típico en sistemas multiprocesador o *bus master*.
 - » El maestro sólo usa el bus durante una transacción.
 - » Si quiere más, debe competir.
 - Cada transacción requiere ciclo previo de arbitraje.
 - » Si líneas de arbitraje en el bus de control → en paralelo con transacción anterior.
 - » Si multiplexación en el mismo bus de datos/dir. → arbitraje integrado.



El maestro de la próxima transferencia se decide:
tras el direccionamiento,
antes de los datos,
...de la transacción actual.

...

Políticas de arbitraje

- ***Pre-emption:***

- Una transferencia en curso puede ser interrumpida (maestro libera bus) por una petición de mayor prioridad.
- Típico en sistemas de paso de mensajes.
 - » Mensajes son transferencias de bloques de datos (relativamente largos).
 - » Una transferencia prioritaria no tiene por qué esperar → interrumpe.
 - » Requiere mecanismo de recuperación:
retransmitir después de prioritaria
sólo la parte restante del mensaje

Hardware de arbitraje

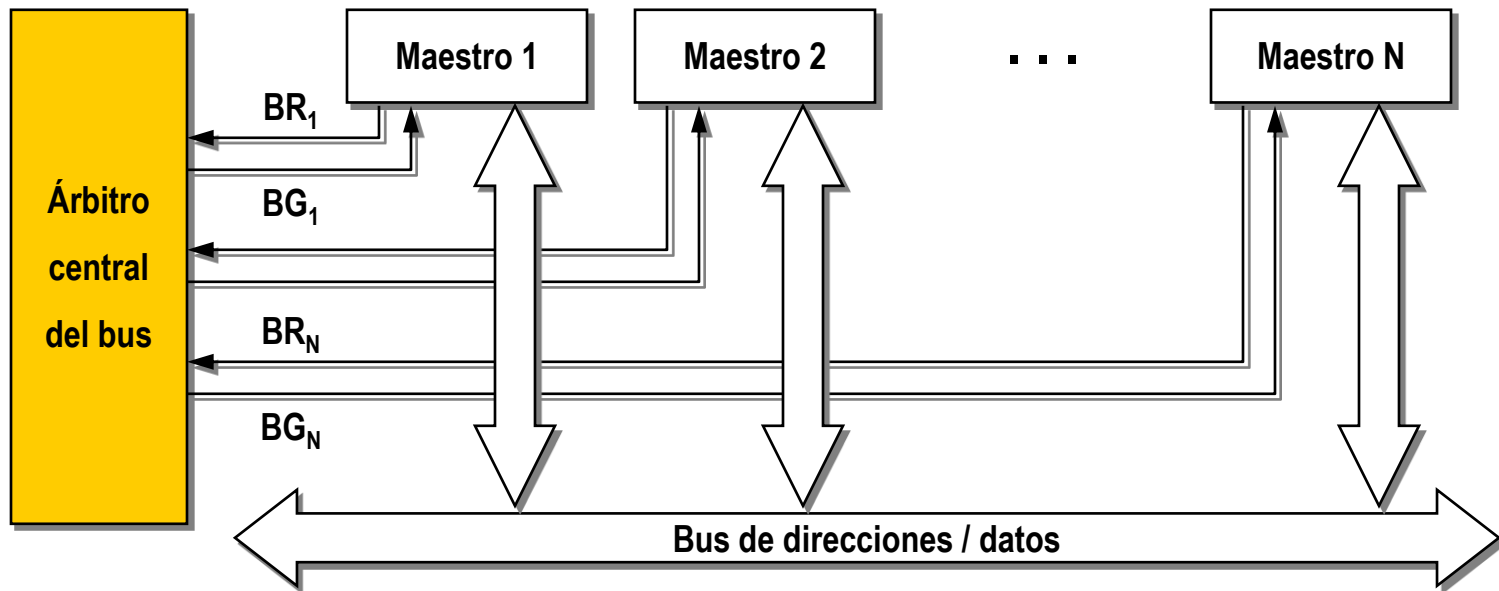
■ Hardware de arbitraje:

- Cada política de arbitraje puede realizarse (en hardware) de diversas maneras.
 - Extremos:
 - **Módulo árbitro**: programable para implementar diversas políticas.
 - **Lógica distribuida**: cada tarjeta activa contiene circuitería de arbitraje.
-
- Gestión centralizada
 - *Daisy-chain*
 - Híbrida (combinada)
 - Gestión distribuida

Hardware de arbitraje

■ Gestión centralizada:

- Cada tarjeta activa dispone de líneas de arbitraje:



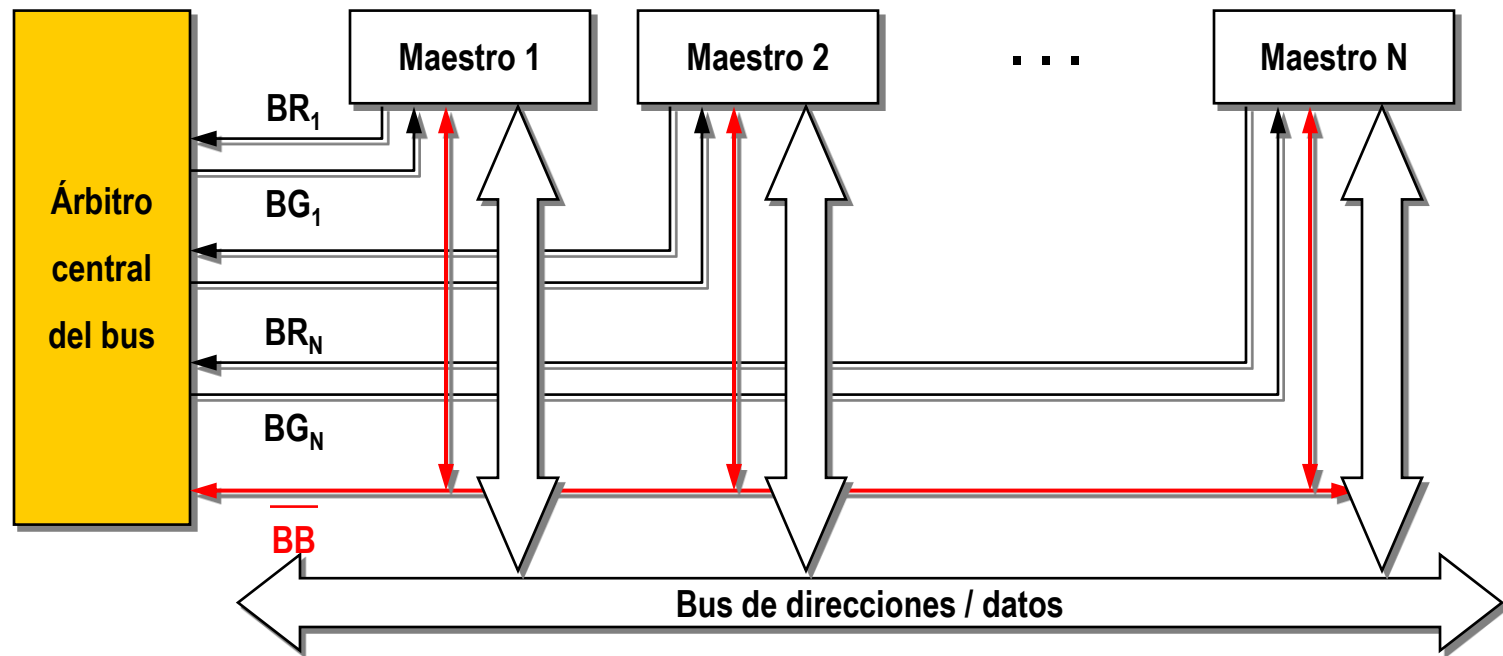
- BR_i (*Bus Request*): Petición del bus al árbitro.
- BG_i (*Bus Grant*): Árbitro concede el bus a tarjeta i .
- Maestro actual indica al árbitro fin de transferencia haciendo $BR\downarrow$.
- El árbitro escoge entonces un nuevo maestro.

Hardware de arbitraje

- Tarjeta árbitro programable para cualquier política:
 - Ej.: $N = 16$, 8 tarjetas E/S, 8 tarjetas procesador.
 - Tratar $N = 1...8$ por prioridad decreciente.
 - Tratar $N = 9...16$ prioridad mínima, por *Round-Robin*.
- ✓ Esquema flexible (política programable).
- ✓ Velocidad de arbitraje.
 - no *daisy-chain*.
 - líneas separadas.
 - durante transferencia en curso se arbitra siguiente maestro.
- ✗ Coste: árbitro, líneas BR/BG.
- ✗ Expandibilidad: máximo N tarjetas activas.

Hardware de arbitraje

- También es posible encontrar una 3ª línea (común) en colector abierto: BB# (**Bus Busy**, ocupado).

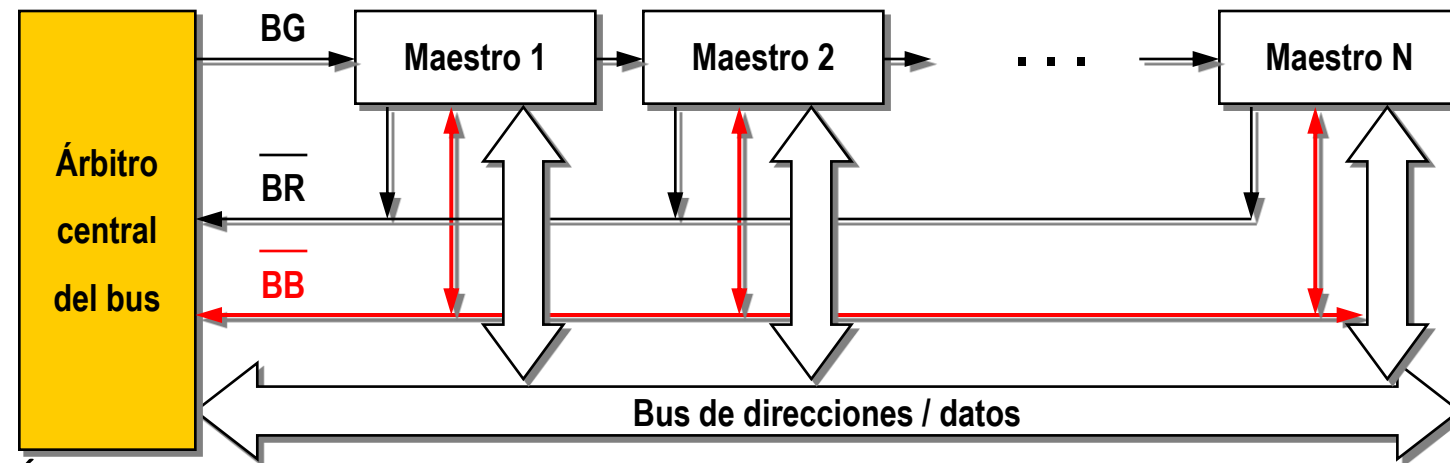


- Ej.: BR_1 , BR_3 2 peticiones. Supongamos árbitro decide BG_1 .
 - Maestro 1 lee BG_1 e indica $BB\# \downarrow$ a todos (árbitro y demás activos); puede desactivar BR_1 una vez ocupado BB.
 - Árbitro espera $BB\# \uparrow$ y decide entre BR_i pendientes; retira BG_1 .

Hardware de arbitraje

■ Daisy-chain o serie:

- La mayoría de autores considera *daisy-chain* una variante de centralizada (hay árbitro central):



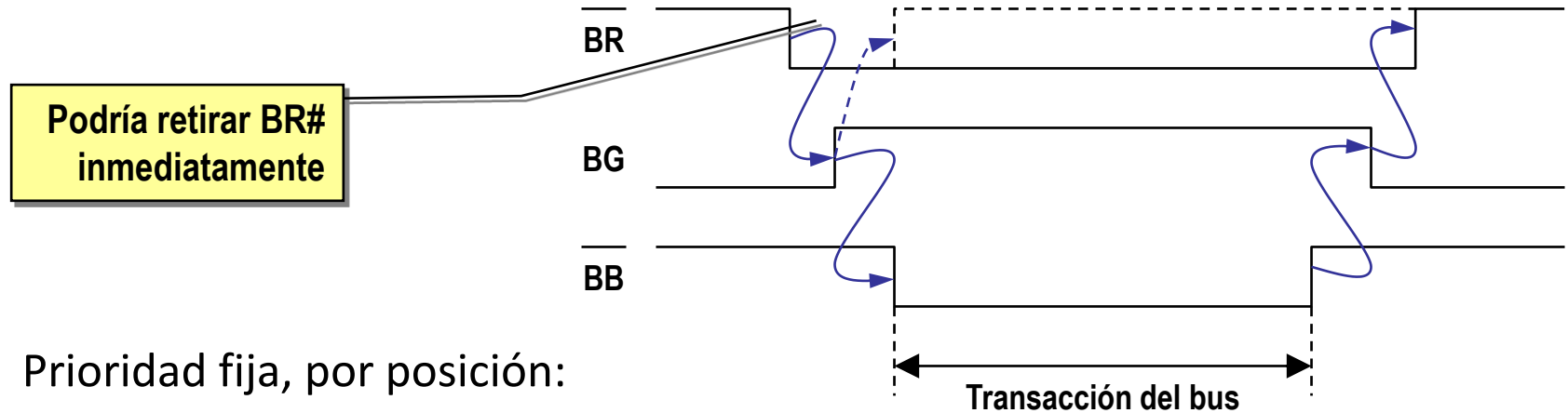
Árbitro sólo 3 líneas:

Cada tarjeta activa 4 líneas:

- Lee BR#: Compartida entre activas (colector abierto)
- Lee BB#: Compartida entre activas (colector abierto)
- Cede BG: Si hay alguna BR#, y no está BB#↓
- Escribe BR#: Si quiere pedir bus
- Lee BG IN: Cadena *Bus Grant*
- Escribe BG OUT: Puede quedarse BG (negar BG al siguiente)
- Lee y escribe BB#



Hardware de arbitraje



Prioridad fija, por posición:

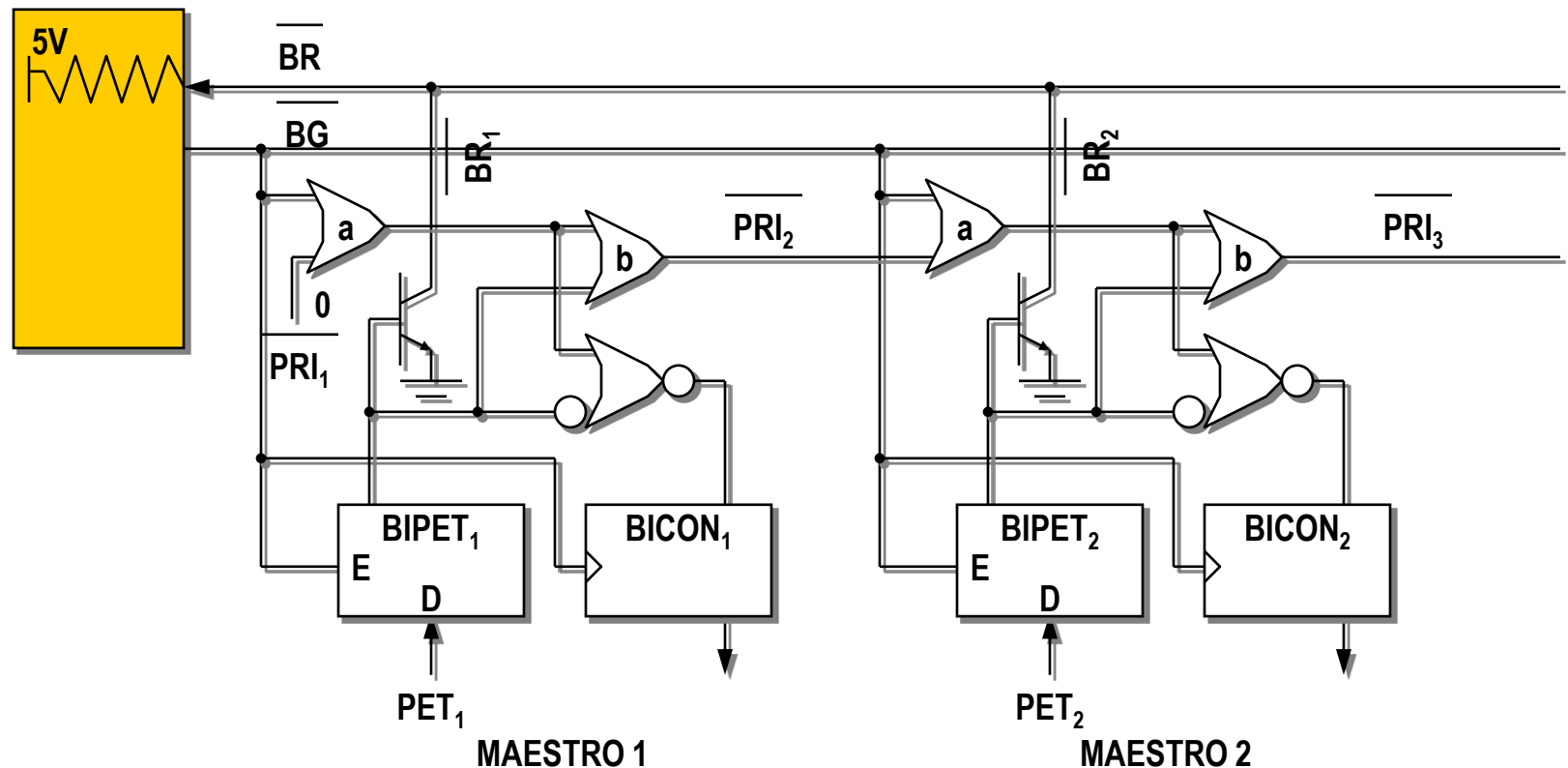
- Varias tarjetas (M2, M5) activan BR#.
- Como bus libre (BB#↑) ⇒ árbitro cede BG↑.
- M1 no pidió BR# ⇒ copia BG IN → BG OUT.
 - así sucesivamente hasta...
- M2 pidió BR# ⇒
 - lee BG IN concedido,
 - escribe BG OUT denegado (M5 no recibirá BG),
 - escribe BB#↓.
- M2 termina ⇒ libera bus BB#↑.
- Árbitro recupera bus BG↓, lee petición BR#, volver a empezar.

Hardware de arbitraje

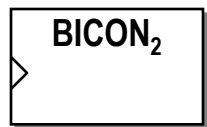
- Suponer que mientras M2 utiliza el bus (BB#↓), M1 quiere pedir BR#.
 - No se le puede dejar: M1 recibe BG antes que M2, ambos usarían el bus.
 - Solución 1:
 - » M1 lee BB#↓ y se da cuenta.
 - » Espera BB#↑ y BG↓ antes de pedir BR#.
 - Solución 2:
 - » *Pre-emption*: Abortar M2.
 - » Requiere *daisy-chain* adicional A IN, A OUT.


Hardware de arbitraje

- Otros autores consideran *daisy-chain* una variante de distribuida (cada tarjeta tiene circuitería propia; la tarea del árbitro es casi trivial, no programable):



Hardware de arbitraje

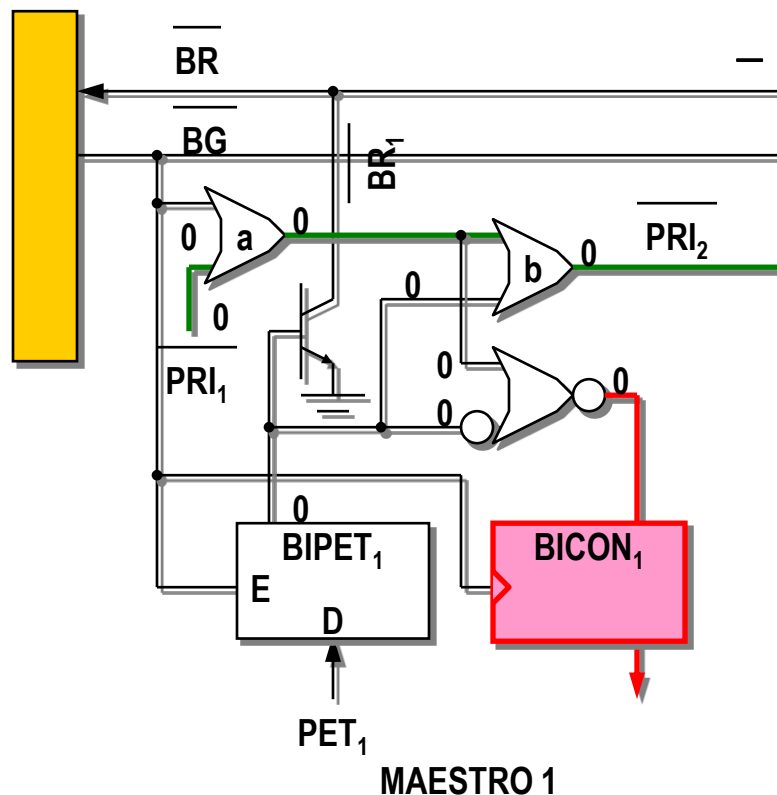


- BR# : Petición del bus. Colector abierto.
- BG# : Concesión del bus. *Daisy-chain*.
- BIPET \equiv Biestables para almacenar peticiones.
 - E \equiv *Enable*. Permite $\text{BIPET}_i \leftarrow \text{PET}_i$ 
- BICON \equiv Biestables para almacenar concesión.
 - Activados en flanco de subida.

Hardware de arbitraje

Ejemplo:

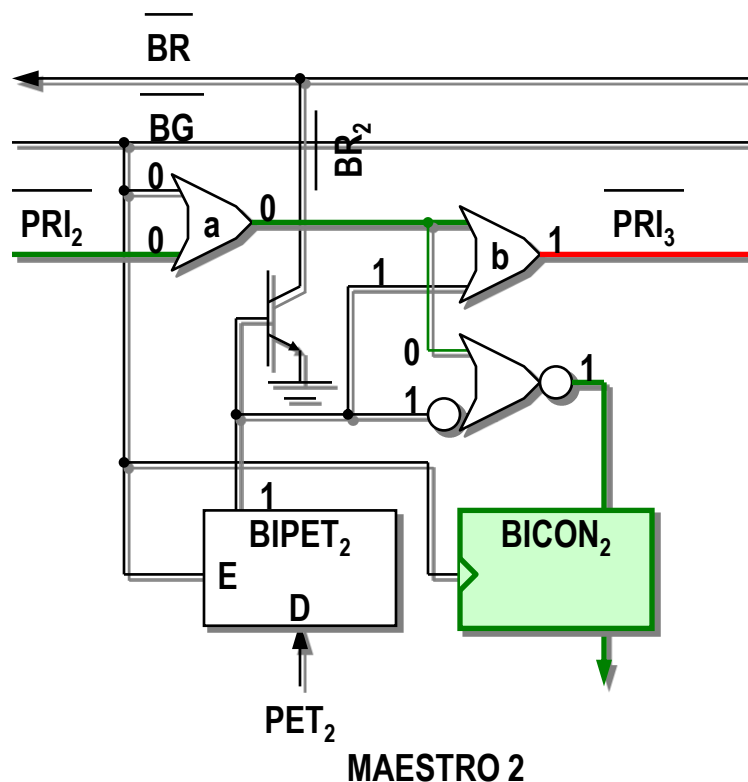
- BG# está a nivel alto mientras nadie pide el bus.
 $\Rightarrow E \uparrow$ permite $BIPET_i \leftarrow PET_i$
- Supongamos $PET_2 \uparrow \Rightarrow BIPET_2 \uparrow \Rightarrow BR\#_2 \downarrow \Rightarrow BR\# \downarrow$



– En este esquema, concesión por pulso

- $BG\# \downarrow$
 $\Rightarrow BIPET_i$ bloqueados momentáneamente.
- $PRI\#_1=0 \mid CONC\#=0 \Rightarrow OR_a=0$.
- $OR_a=0 \mid BIPET_1=0 \Rightarrow OR_b=PRI\#_2=0$.
- $OR_a=0 \mid BIPET_1\#=1 \Rightarrow NOR=BICON_1=0$.

Hardware de arbitraje

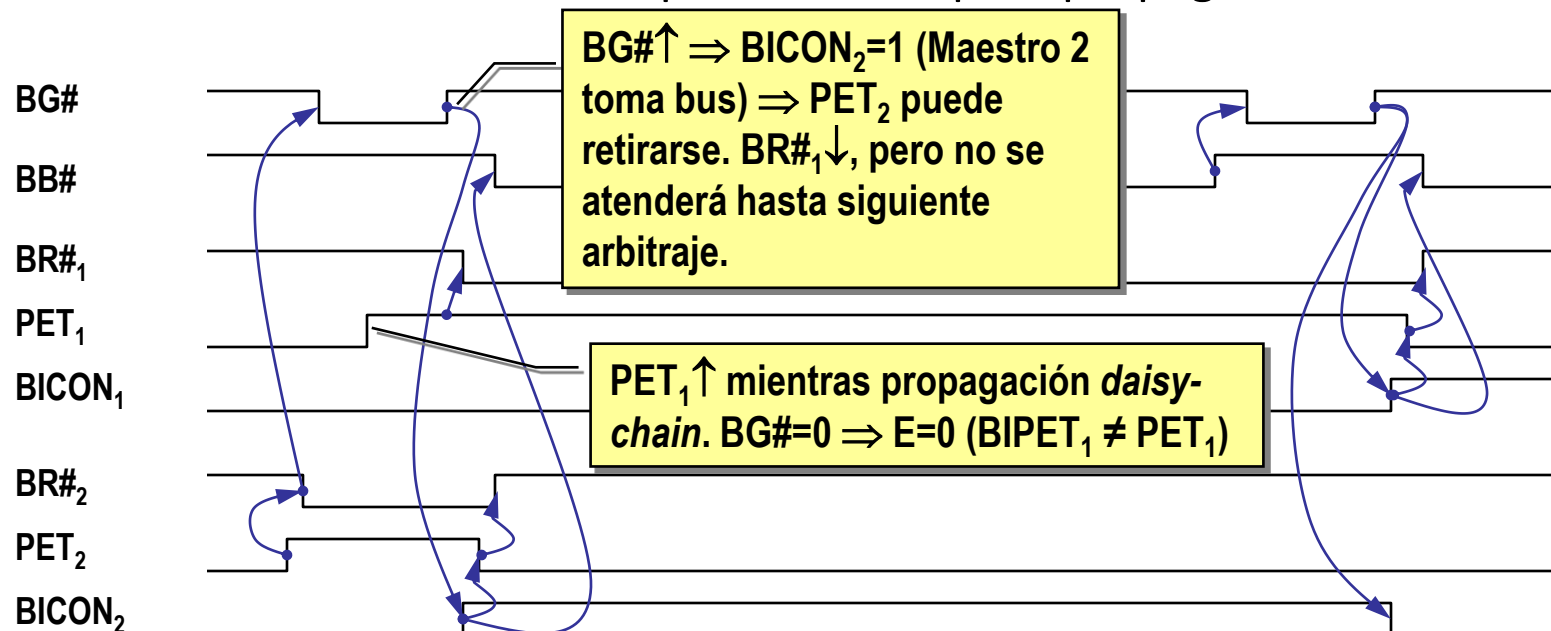


- $PRI\#_2=0 \mid BG\#=0 \Rightarrow OR_a=0$.
- $OR_a=0 \mid BIPET_2=1 \Rightarrow OR_b=PRI\#_3=1$.
- $OR_a=0 \mid BIPET_2\#=0 \Rightarrow NOR=BICON_1=1$.
- A partir de aquí ($i>2$):
 - » $PRI\#_i = 1 \Rightarrow BICON_i=0$
independientemente de PET_i
- $BICON_i = 1 \Rightarrow BIPET_i\#=0$ ($BIPET_i=1$) & $OR_{a,i}=0$ ($PRI\#_i=0$ & $BG\#=0$)
 - » (sólo toma el bus quien lo haya pedido y le llegue el *daisy-chain*)
- $PRI\#_i = 0 \Rightarrow OR_{a,i}=0$ & $BIPET_i=0$
 - » (sólo se propaga daisy-chain si llega hasta allí y no se pidió el bus)

- $BG\#\uparrow$
 \Rightarrow "Latch" de $BICON_i$
- Cadena $OR_a-OR_b = 1$
- $BIPET_i$ habilitados otra vez.

Hardware de arbitraje

- Cada OR tiene un tiempo de respuesta (orden ns).
- BG#=0 tiempo suficiente para propagar la señal.



- Este esquema requiere señalar fin de transferencia.
 - BB#: bus ocupado, colector abierto.
 - Maestro 2 señala que acabó BB#↑.
 - Árbitro lee BB#↑, pendiente BR#1 ⇒ concede BG#.

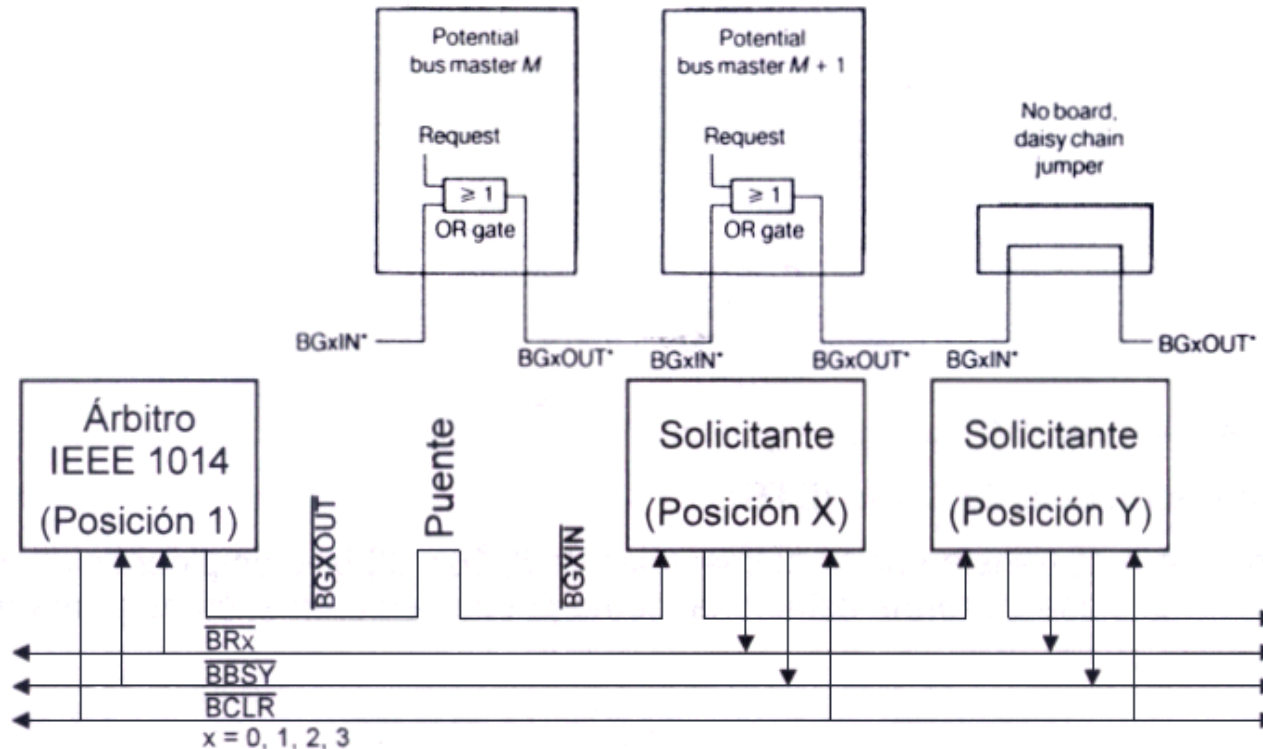
Hardware de arbitraje

■ Ventajas e inconvenientes de *Daisy-chain*:

- ✓ Simplicidad lógica de control: Poca circuitería, pocas líneas, coste↓
- ✓ Expandibilidad: Nº de maestros ilimitado (limitado sólo por tiempo de propagación)
- ✗ Lentitud: Ciclo de arbitraje suficientemente largo para garantizar propagación.
- ✗ Prioridad fija \Rightarrow no equidad:
 - Cambiar prioridades \Rightarrow cambiar orden físico tarjetas en *daisy-chain*.
 - Prioridad determinada por posición en la cadena (posible *starvation*).
- ✗ Poca tolerancia a fallos: Fallo en tarjeta activa \Rightarrow ruptura de la cadena.

Hardware de arbitraje

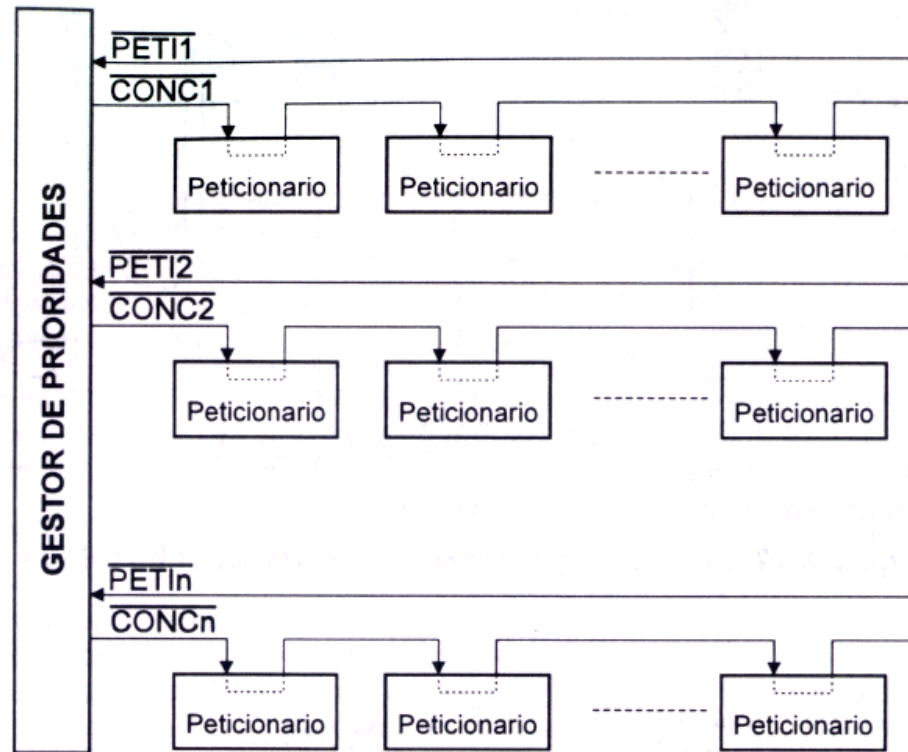
- Ej.: Bus VME permite “puentear”.
 - Tarjetas *daisy-chain* debieran estar contiguas (vecinas físicamente) en el *rack*.
 - Cambiar prioridad \Rightarrow cambiar orden.
 - Si no hay tarjeta o es tarjeta pasiva \Rightarrow puentear (*jumper*).



Hardware de arbitraje

■ Gestión híbrida (combinada):

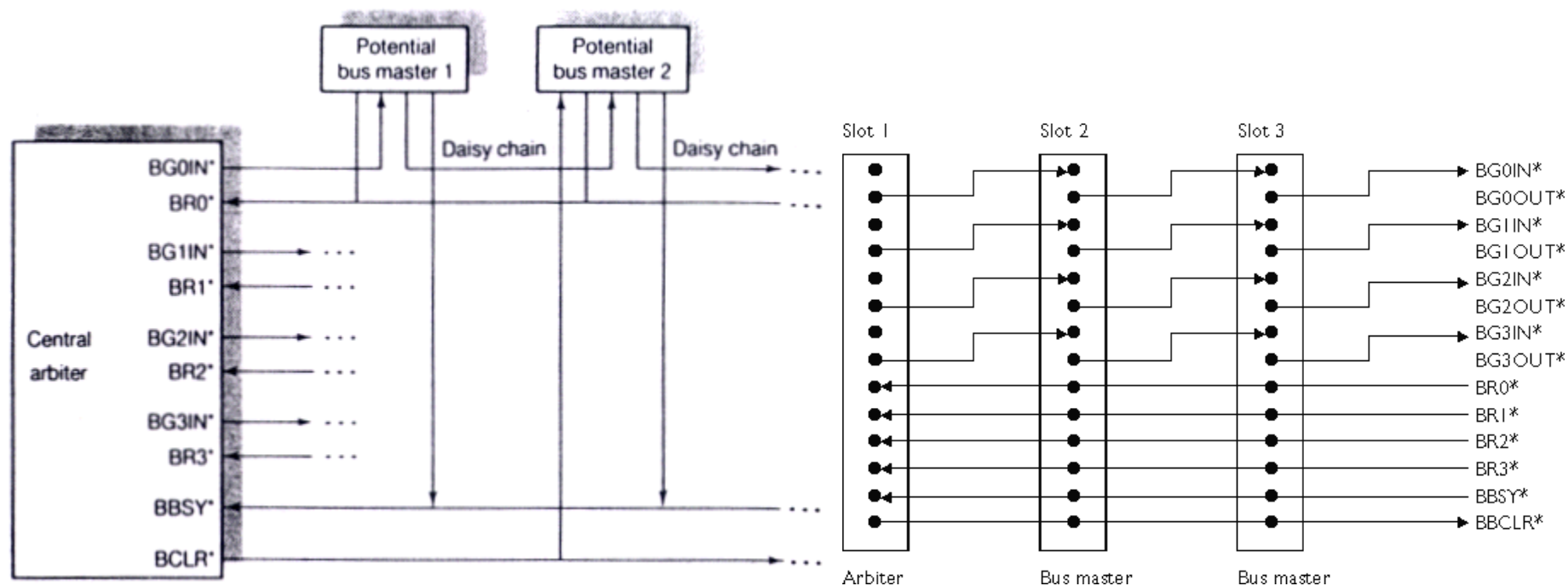
- Conectar un *daisy-chain* a cada línea del árbitro central:



- Política programable entre cadenas.
- Prioridad fija (*daisy-chain*) dentro de cada cadena.

Hardware de arbitraje

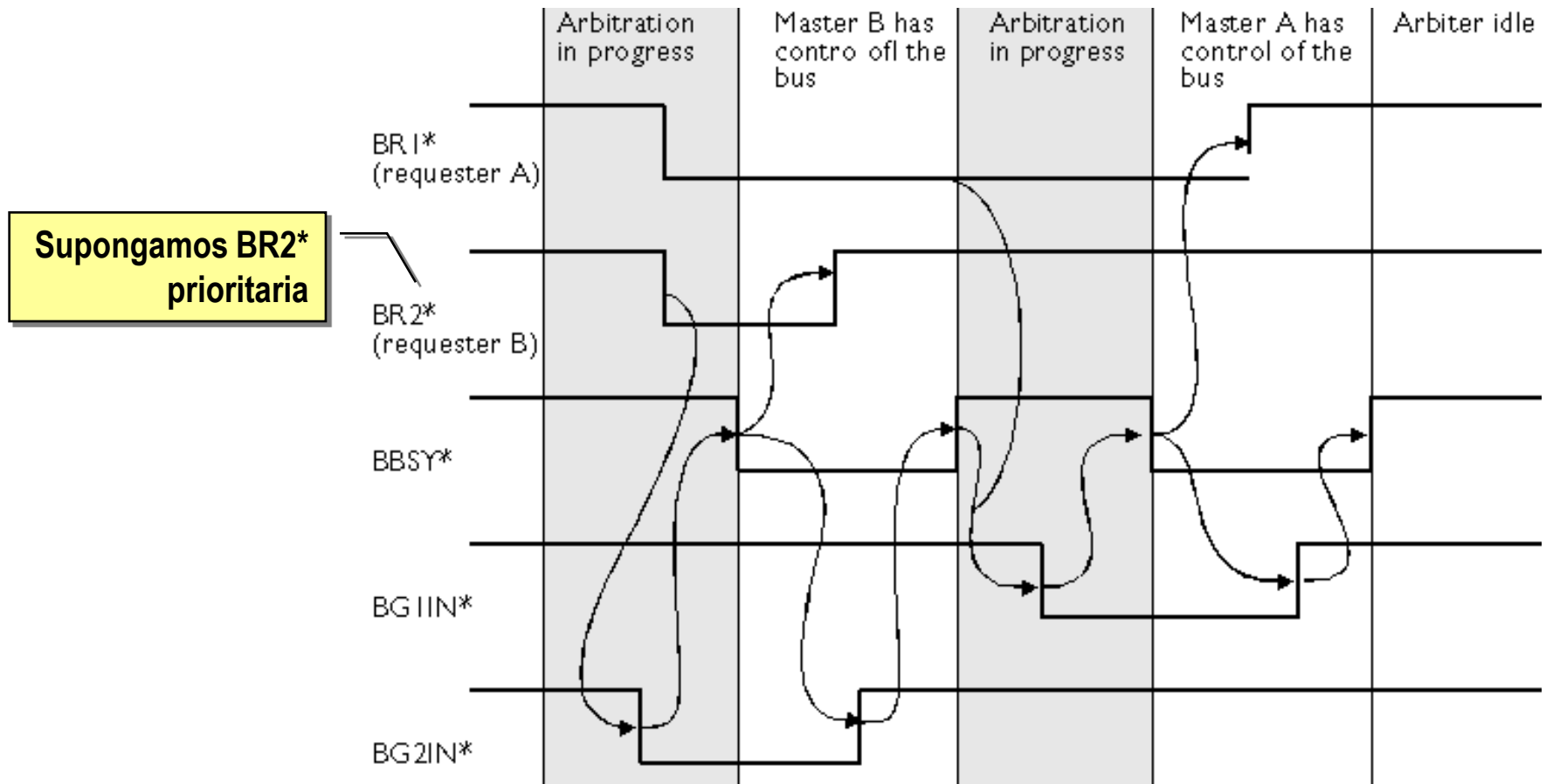
- Ejemplo: arbitraje del bus VME.



- BRi* compartida por todas las tarjetas.

Hardware de arbitraje

- BBSY* (*Bus busy*, ocupado). Árbítro lee, activos escriben (colector abierto).
 - Supongamos dos peticiones A y B por cadenas distintas:



Hardware de arbitraje

- BCLR* (*Bus clear*, liberar). Árbitro escribe, activas leen.
 - Maestro actual debe liberar el bus (*pre-emption*):

Árbitro en el slot 1

Tarjeta colocada en
el slot M con
prioridad < i

Maestro actual del
bus

Tarjeta colocada en
el slot N con
prioridad i

Requiere el bus

Activa BRi* □

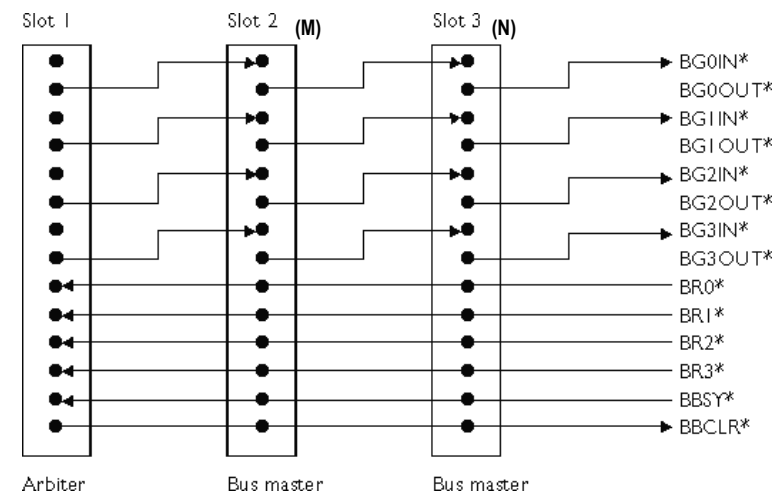
IF prioridad del maestro actual es menor
que i THEN BCLR* □

Libera el bus B
BBSY* □

Activa BGiOUT* □

Copia BGiIN* en
BGiOUT*

Espera BGiIN* □
Activa BBSY* □



Hardware de arbitraje

■ Gestión distribuida:

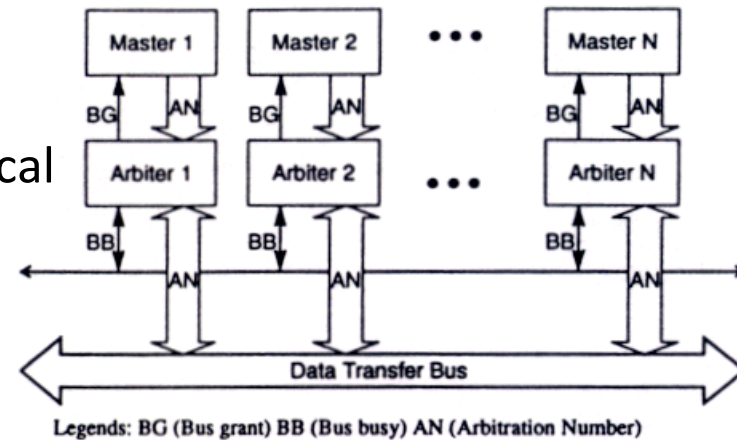
- No hay árbitro central, sino que la circuitería de arbitraje se reparte por las tarjetas activas.
- Típico en sistemas multiprocesador.
- Ejemplo 1: Multibus II
 - Arbitraje integrado
 - Direccionamiento.
 - Arbitraje para siguiente transferencia.
 - Datos transferencia actual.

| | | |
|-------------------|-----------|------|
| Orden / dirección | Arbitraje | Dato |
|-------------------|-----------|------|

- Política de prioridad fija.
- Cada tarjeta tiene un nº de arbitraje (*Arbitration Number, AN*).

Hardware de arbitraje

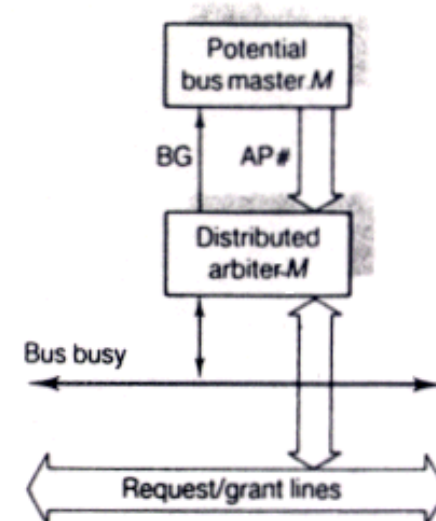
- Petición:
 - tarjetas escriben AN al árbitro local
 - árbitros
 - » escriben AN en bus de datos/direcciones (colector abierto)
 - » leen AN resultante (OR lógico)
 - » retiran AN si resultante > local
- Tras algunas oscilaciones, gana árbitro con mayor AN.
- Árbitro ganador
 - espera a que acabe la transferencia actual (BB↓).
 - se apodera del bus y notifica a tarjeta (BG↑, BB↑).



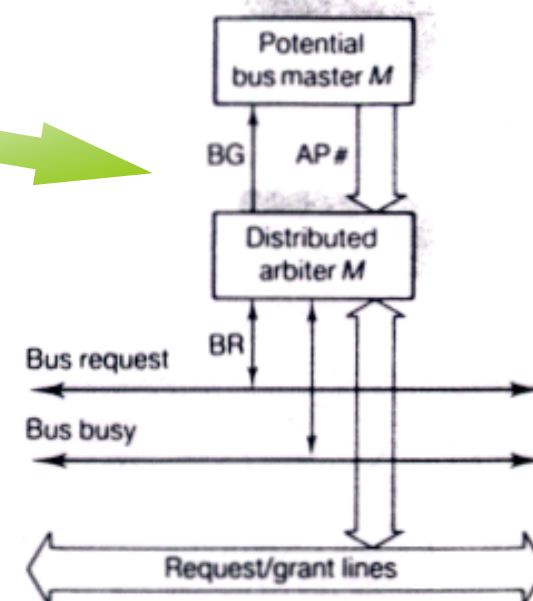
Hardware de arbitraje

■ Ejemplo 2: NuBus

- AP# \equiv *Arbitration Priority Number*
- SBRG \equiv *Shared Bus Request / Grant*
 - Bus separado de arbitraje (colector abierto).

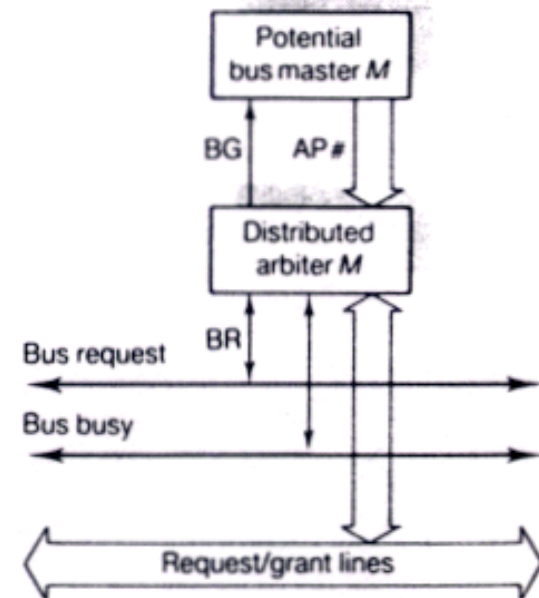


- **Arbitraje por prioridad:** idéntico a Multibus II.
- **Arbitraje por equidad-prioridad:**
 - Se añade BR, línea compartida.
 - Equidad: Una tarjeta que ha sido atendida no tiene derecho a pedir bus hasta que las demás hayan sido atendidas
 - » en ese momento BR↓ \Rightarrow empieza otra ronda.



Hardware de arbitraje

- Permite equidad-prioridad simultáneamente:
 - » Ej.: 8 tarjetas E/S priorizadas: configuradas con AP#16..AP#9 (prioridad decreciente) y para no atender línea BR.
 - » 8 tarjetas CPU *Round-Robin*: configurarlas con AP#8..AP#1 para atender a línea BR.
- Ej.: 1ª ronda:
 - Solicitan bus AP#1,2,3,4 \Rightarrow BR \uparrow
 - Arbitraje:
 - » AP#4 gana, BB \uparrow , acaba, BB \downarrow
 - » AP#3 gana, BB \uparrow , acaba, BB \downarrow
 - Vuelven a solicitar bus AP#3,4
 - » pero ya fueron atendidos
 - » aún hay BR pendientes, #1,2
 - » árbitros 3,4 no activan BR \uparrow

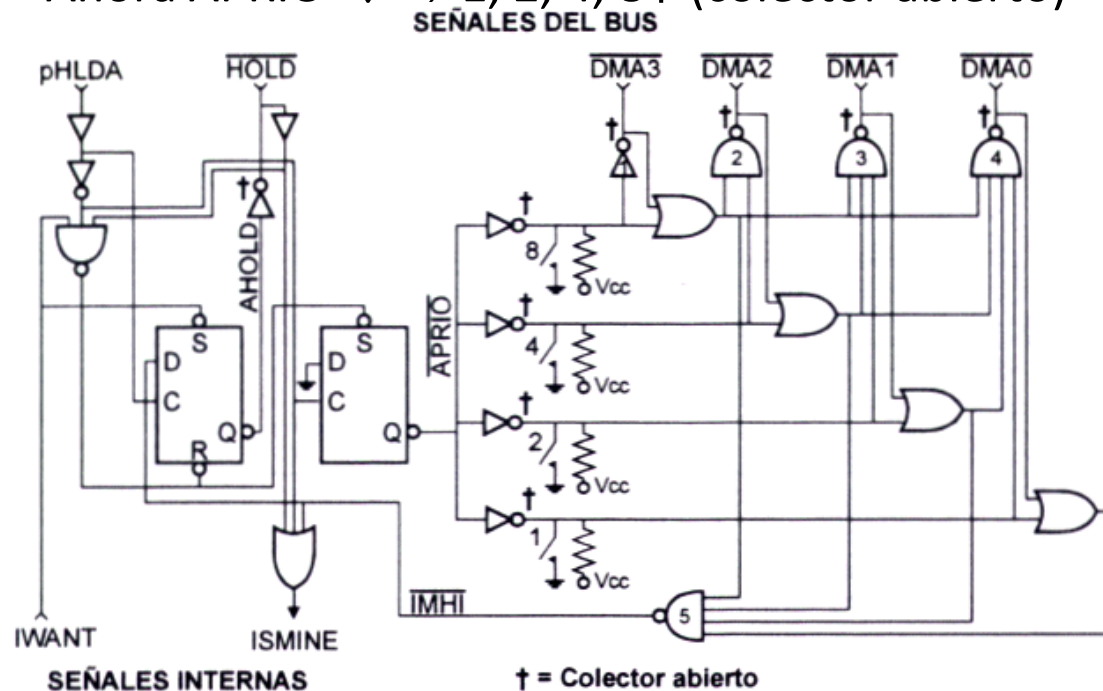


Hardware de arbitraje

- Arbitraje:
 - » AP#2 gana, BB $\uparrow\downarrow$
 - » AP#1 gana, BB $\uparrow\downarrow$
- Cuando AP#1 hizo BB \uparrow (ocupar bus), desactivo la última BR pendiente.
- BR $\downarrow \Rightarrow$ los árbitros locales con AP#3,4 inician 2ª ronda de arbitraje.
- Arbitraje para próxima transferencia
 - » en paralelo con transferencia actual
 - » aprovechando el bus separado de arbitraje
- 2ª ronda:
 - Dos árbitros pendientes han observado BR \downarrow
 - Arbitraje:
 - » AP#4 gana. Cuando AP#1 libere BB \downarrow , lo ocupará #4.
 - » AP#3 gana.

Hardware de arbitraje

- Ejemplo 3 (más detallado): bus S100
 - Señales internas: IWANT \equiv BR, ISMINE \equiv BG
 - Ignoremos parte izquierda del esquema.
 - Creer que $IWANT \uparrow \Rightarrow APRIO^* \downarrow$
 - Antes $APRIO^* \uparrow \Rightarrow 1, 2, 4, 8 \downarrow$
 - Ahora $APRIO^* \downarrow \Rightarrow 1, 2, 4, 8 \uparrow$ (colector abierto)



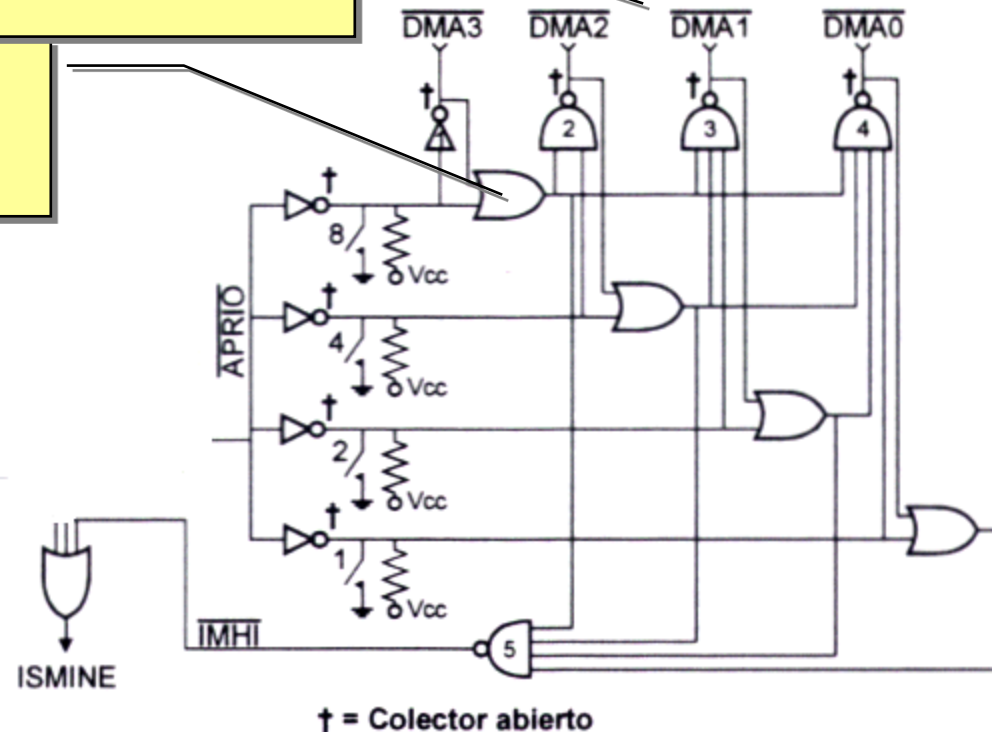
Hardware de arbitraje

- Las líneas horizontales son el AP# (15 en este caso)
 - Cerrando *jumper*s se puede cambiar (forzar línea↓)
- DMA*3-0 es el OR lógico de los AP# que compiten, pero invertido (NOR lógico).

2. Salida OR → al resto de las NAND. Salida 0 ⇒ NAND=1 (retirar AP# de competición). Se compite de msb a lsb.

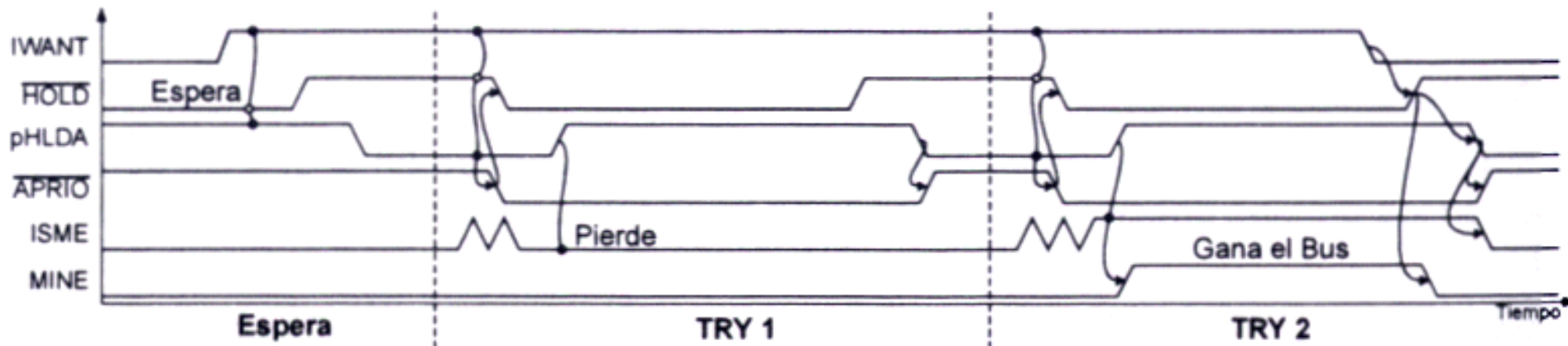
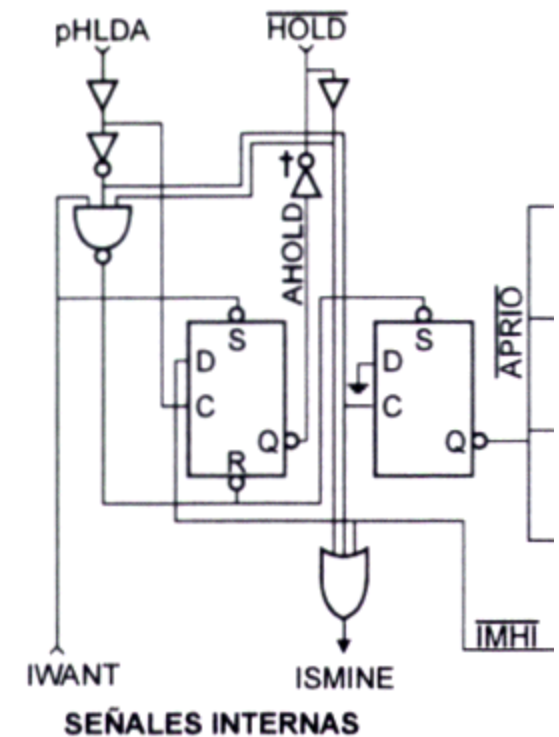
1. Puertas OR reciben bit AP# y DMA*. Salida 0 ⇒ bit DMA*=0 (número en bus es alto) & bit AP#=0 (AP# no es tan alto)

| AP#3 | DMA*3 | OR | AP# | DMA |
|------|-------|----|-----------|---------|
| 0 | 0 | 0 | 0XXX | < 1XXX |
| 0 | 1 | 1 | 0XXX | <= 0XXX |
| 1 | 0 | 1 | 1XXX | <= 1XXX |
| 1 | 1 | 1 | Imposible | |



Hardware de arbitraje

- En general, IMHI* (ISME en la figura) oscila hasta hallar ganador.
- pHLDA es un reloj que marca fases de arbitraje
 - 0 \Rightarrow se admiten peticiones. Competición.
 - 0 \rightarrow 1 \Rightarrow se decide ganador. Transferencia.
- Ganador pone HOLD* = 0 \Rightarrow bus ocupado, no se admiten peticiones.

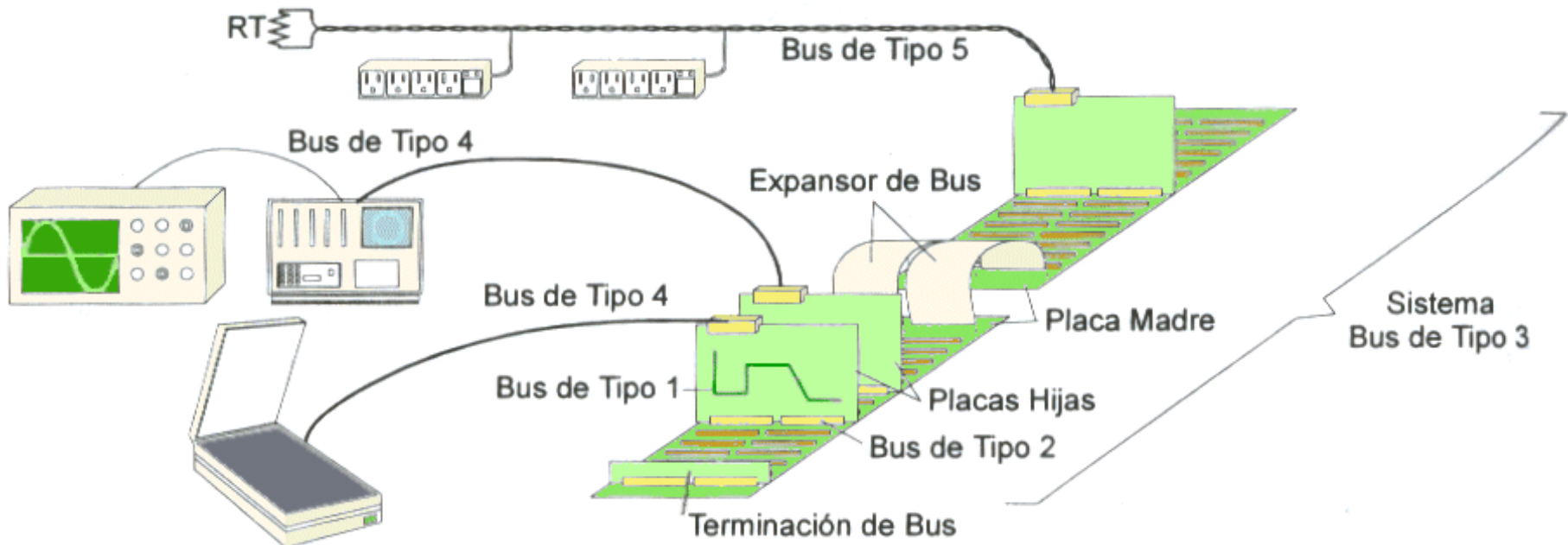


Entrada/salida y buses

- Funciones del sistema de E/S. Interfaces de E/S
- E/S programada
- Interrupciones
- DMA (Acceso directo a memoria)
- Estructuras de bus básicas
- Especificación de un bus. Transferencias. Temporización. Arbitraje
- **Ejemplos y estándares**

Clasificación de buses según jerarquía

- **Tipo 0**: Internos al chip. Inaccesibles. No documentados.
- **Tipo 1**: Interconexión componentes tarjeta PCB.
- **Tipo 2**: Interconexión tarjetas de panel posterior.
- **Tipo 3**: Expansión del bus (mayores distancias).
- **Tipo 4**: Conexión periféricos en paralelo.
- **Tipo 5**: Conexión periféricos en serie.

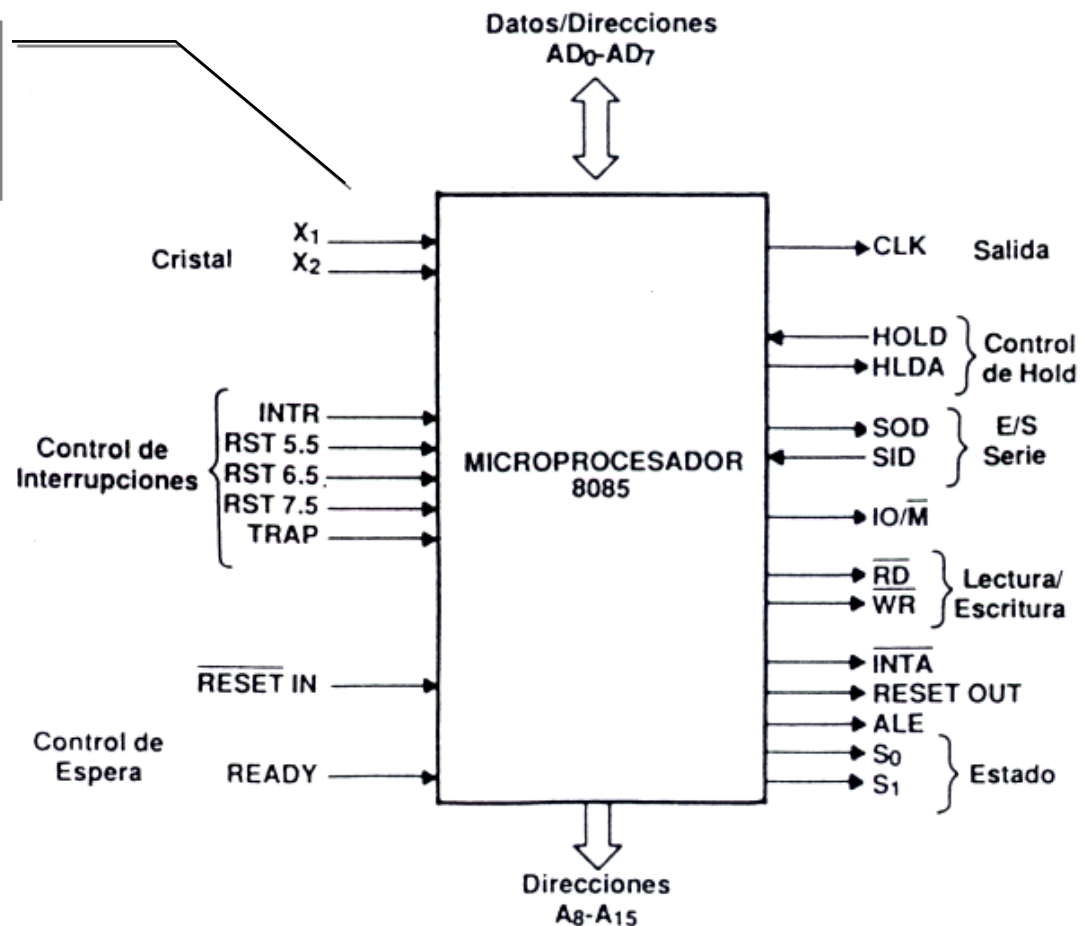


Clasificación de buses según jerarquía

■ Buses de PCB, locales o *board* (tipo 1)

- Documentados con procesador y chipset (circuitos de apoyo).

Ejemplo:
Señales de conexión
del 8085.



Clasificación de buses según jerarquía

...

- Distancias cortas $\approx 10 - 50$ cm (tamaño placa) y pequeña impedancia.
 - Altas velocidades (≈ 100 MB / s).
 - No requieren terminadores.
- Señales típicas de control:
 - Selección R/W IO/M
 - Espera WAIT RDY
 - DMA HOLD HLDA
 - IRQ INTR NMI
 - y otras señales (conocidas como “housekeeping”) dedicadas entre determinados componentes para controlar aspectos específicos del chipset o el procesador en concreto. Ej.:
 Refresco de memoria.
- En el apéndice: señales del 8086

Clasificación de buses según jerarquía

■ Buses de panel posterior o *backplane* (tipo 2)

- Conectan tarjetas procesadoras.
 - Cada tarjeta podría ser una CPU distinta.
- Velocidad 10 - 200 MB /s
- Multitud de estándares de bus único, independ. del procesador:
 - STD: una serie de normas estándares para los fabricantes de tarjetas, particularizadas para varias familias de microprocesadores de 8 bits (casi buses de tipo 1).
 - Multibus II (IEEE-1296) de Intel; Nubus (IEEE 1196) de Texas Instruments; Futurebus de IEEE.
 - VME (IEEE 1014): un estándar muy utilizado en el pasado reciente, hasta 24 MB/s.
 - PCI y otros buses del PC (en el apéndice).
- Muchos de ellos permiten fabricar sistemas hardware multiprocesador independientes del fabricante.

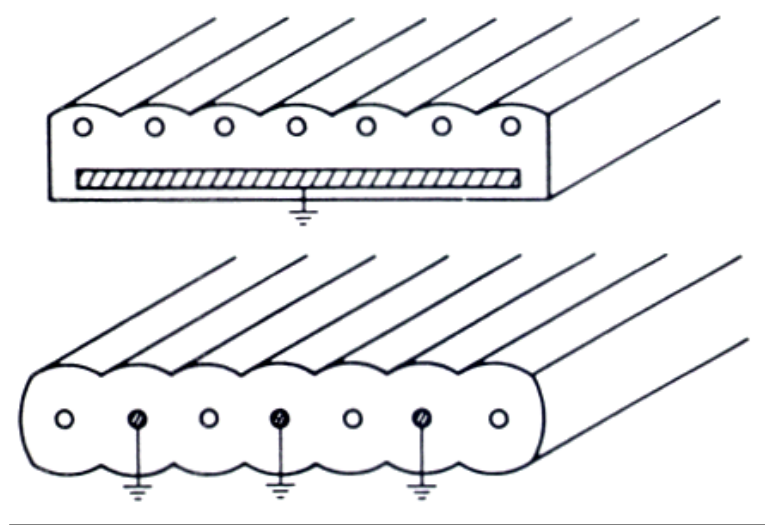
Clasificación de buses según jerarquía

■ Buses de expansión (tipo 3)

- Interconectan subsistemas a mayores distancias (≈ 10 m).
- Sufren el **efecto de línea de transmisión**:
 - A altas frecuencias dos hilos paralelos...
 - ...no son un circuito abierto, sino que
 - ...equivalen a una impedancia resistiva y capacitiva;
 - una señal...
 - ...tarda en llegar al extremo del cable,
 - ...rebota en el extremo (si no tiene terminador), y
 - ...retrocede por el cable interfiriendo con otras señales.
- Retardo bus mayor.
- Ancho de banda menor (≈ 1 MB/s).

Clasificación de buses según jerarquía

- Necesidad de apantallar cables.
 - Se rodean las líneas de un potencial fijo muy estable.



⇒ Se reducen las interferencias...

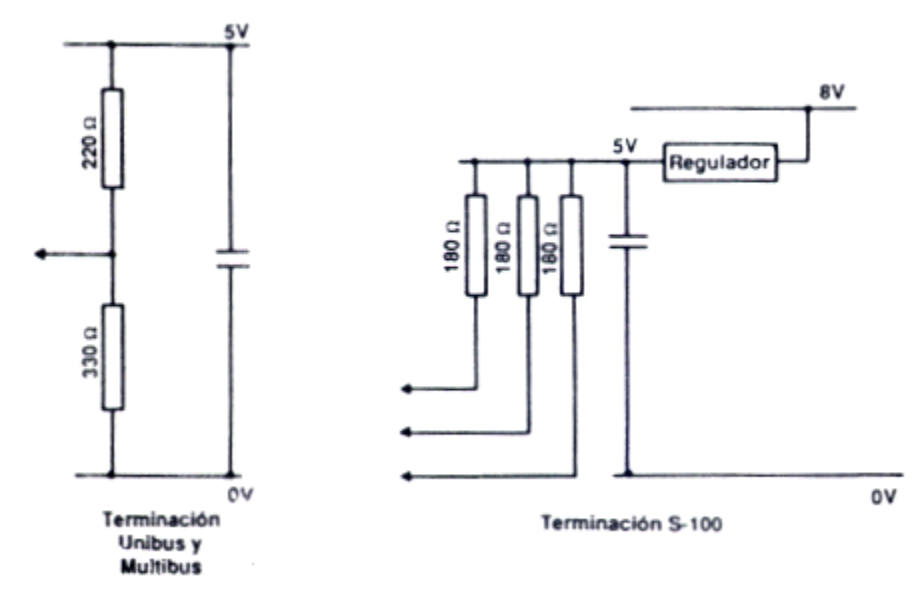
- externas, e
- internas (“crosstalk”)

una línea cambia de potencial (ej.: 0V → 5V)

⇒ en la línea vecina se induce un cambio menor y opuesto (ej. 5V → 3V); mayor según proximidad de las líneas, velocidad y amplitud del cambio.

Clasificación de buses según jerarquía

- Necesidad de incluir terminadores de baja impedancia.
 - Son circuitos electrónicos que combinan R y C para conseguir la misma impedancia característica del cable.



- Deben colocarse a ambos extremos del bus.
- Hacen que se extinga la señal.
- Simulan que el cable es de longitud infinita.

Clasificación de buses según jerarquía

■ Buses para periféricos o de interfaz

paralela (tipo 4)

serie (tipo 5)

■ Distintas soluciones según:

■ Tipo de dispositivo:

- Interno (Disco, cinta, CD-ROM).
- Externo (Impresora, escáner).

■ Distancia de conexión:

- Dispositivos locales.
- Terminales CRT, POS.
- Sensores, equipos de medida.

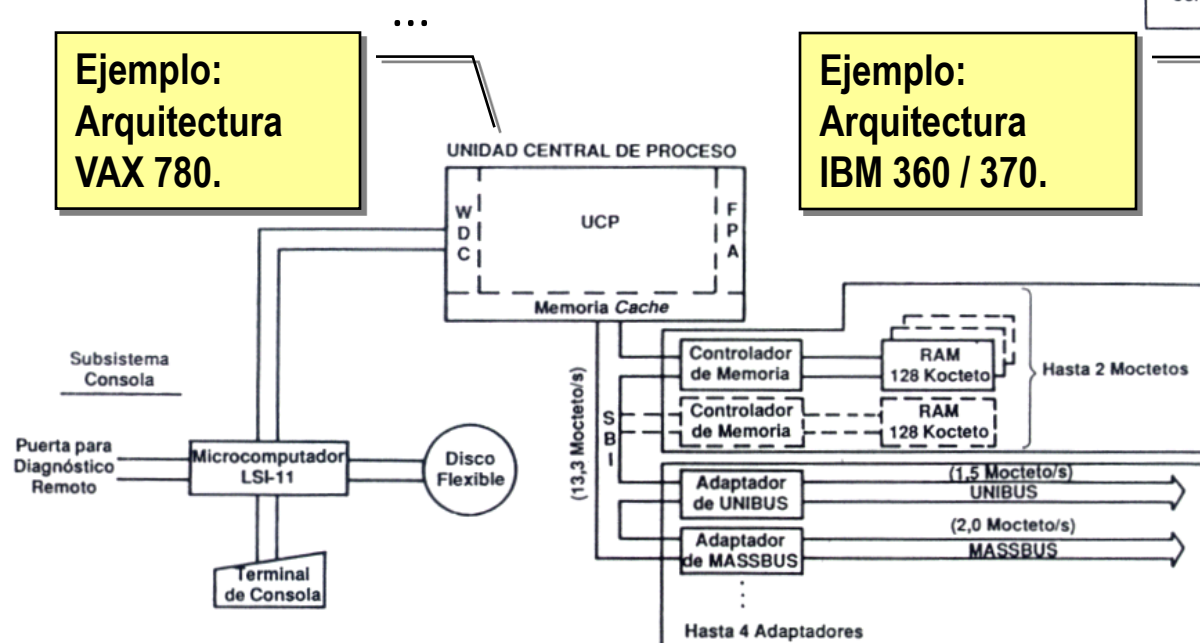
■ Coste deseado:

- Aumenta con el número de hilos (paralelismo), la distancia y la velocidad requeridas.

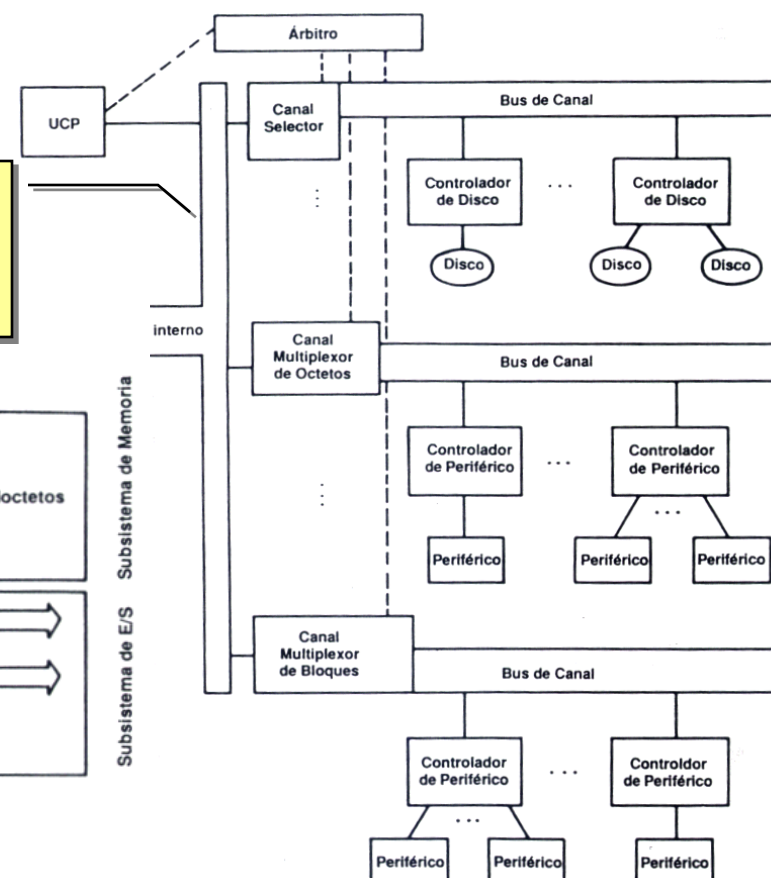
Clasificación de buses según jerarquía

- Buses de interfaz paralela (tipo 4):
 - Para dispositivos rápidos (<varios MB/s) no muy lejanos (<15m)
 - Distintas soluciones:
 - Bus doble

**Ejemplo:
Arquitectura
VAX 780.**



**Ejemplo:
Arquitectura
IBM 360 / 370.**



Clasificación de buses según jerarquía

- ...
- Controladora que sigue las especificaciones del bus del sistema (tipo 2)
 - » Ej.: Disquetera.
- Estándares de bus de interfaz en paralelo.
 - » Ejs.: ATA, SCSI, HP-IB → GPIB (IEEE-488).
 - » En el apéndice: ATA y SCSI
- Prototipos.
 - » Típico de ambientes académicos o de I+D.
 - » Toman del sistema las señales necesarias para la aplicación / experimento y las acondicionan según los requisitos del dispositivo.

Clasificación de buses según jerarquía

- Buses de interfaz serie (tipo 5):
 - Multiplexación extrema: toda la información por 1 bit (2 hilos).
 - Obligatorios en largas distancias (por coste, retardos, ruido).
 - Tradicionalmente han sido la solución para
 - Interconexión de periféricos lentos / lejanos.
 - Redes de ordenadores.
 - Sin embargo, debido al problema de *skew* a altas velocidades, existe una tendencia a pasar de paralelo a serie, utilizando técnicas para enviar la señal de reloj junto con los datos.
 - En el apéndice: USB e IEEE 1394 (FireWire).