Tema 12: Características y protocolos de los buses

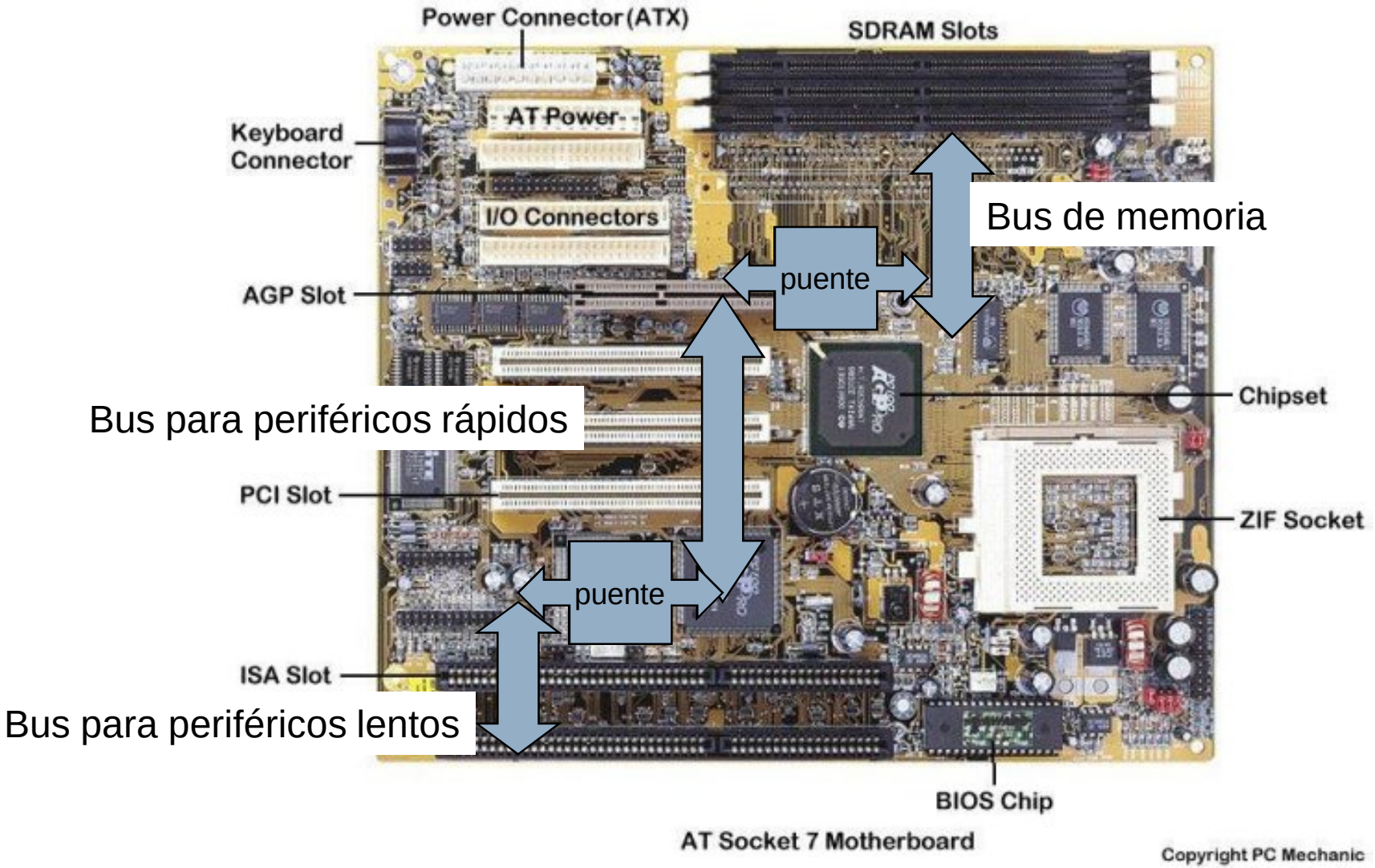
Departamento de
Informática e Ingeniería
de Sistemas

Universidad Zaragoza

Características y protocolos de los buses

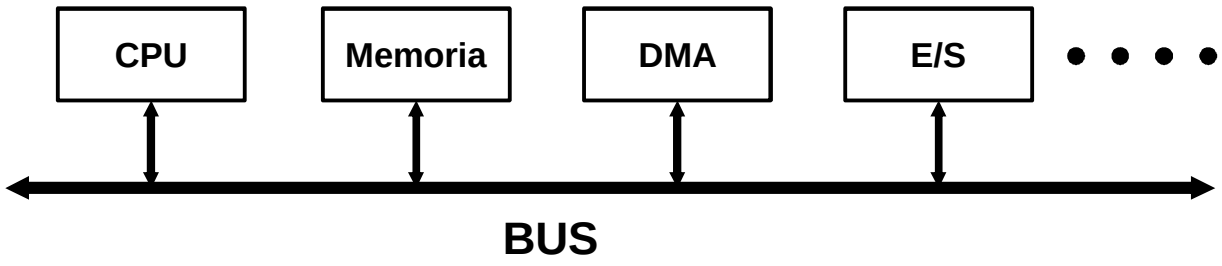
1. **Importancia de la comunicación entre dispositivos**
2. **Características de un bus**
3. **Protocolos de transferencia**
4. **Protocolos de arbitraje**

Motivación:¿cómo se comunican todos estos elementos entre si?



Motivación

- Un computador está formado por múltiples elementos que colaboran entre sí
- Estos elementos deben intercambiar datos y sincronizarse para funcionar correctamente
- Los buses de una computadora proporcionan el soporte necesario para estas comunicaciones
- Los buses son elementos caros así que se comparten entre varios dispositivos



Definiciones

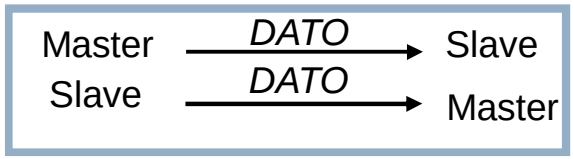
Un bus es un medio de transmisión compartido que interconecta dos o más dispositivos de un sistema digital

- **Función:** permitir la correcta comunicación entre los dispositivos interconectados

- **Estructura**:
  - Conjunto de líneas (conductores eléctricos) compartidas por todos los dispositivos
    - Cualquier señal transmitida por un dispositivo está disponible para los demás
    - Si dos dispositivos transmiten simultáneamente utilizando las mismas líneas las señales se solapan y la información resultante es inválida
  - Conexiones en paralelo: varias señales viajan juntas sobre distintas líneas

Definiciones

- **Elementos** implicados en una transferencia:
  - **Maestro (master)**: Inicia y dirige las transferencias (CPU, DMA, ...)
  - **Esclavo (slave)**: Obedece y accede a las peticiones del master (memoria, interfaz E/S, ...)
- **Tipos básicos** de transferencia:
  - **Escritura**: el maestro envía datos al esclavo
  - **Lectura**: el maestro pide datos al esclavo



- **Ciclo de bus**: pasos a seguir para realizar una transferencia (puede realizarse en uno o varios ciclos de reloj)
  - Operaciones básicas:
    1. **Direccionamiento** del slave
    2. Especificación del **tipo de operación** (lectura o escritura)
    3. **Transferencia** del dato
    4. **Finalización** del ciclo de bus

Definiciones

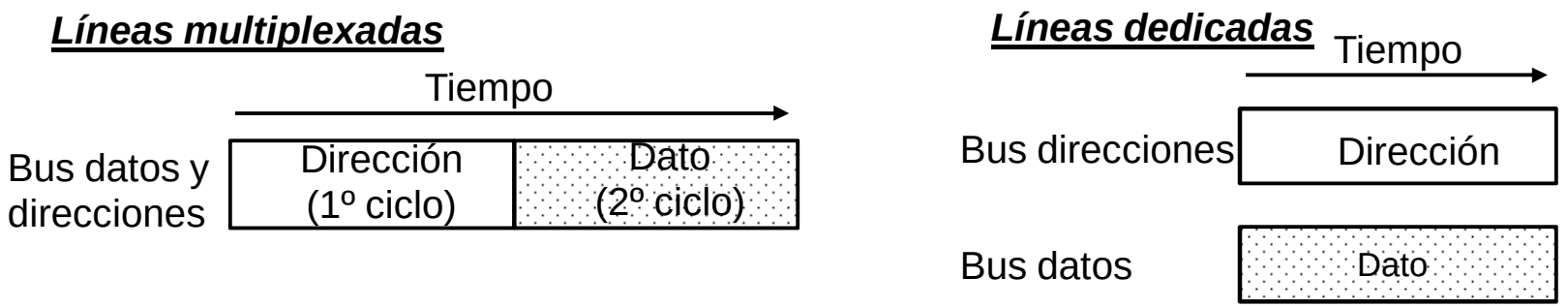
- **Sincronización**: método para determinar el inicio y el final de cada transferencia
- **Arbitraje**: método para controlar el acceso al bus en caso de varios masters
- **Capacidad de conexión**: número máximo de dispositivos conectables
- **Longitud de bus**: máxima distancia que puede separar a dos dispositivos conectados
- **Ancho de bus**: número total de líneas
- **Ancho de datos**: número total de líneas para datos
- **Ancho de banda**: caudal máximo de información que puede transmitirse

Clasificación de las líneas del bus según su función

- **Líneas de datos (*bus de datos*)**: transmiten datos
- **Líneas de direcciones (*bus de direcciones*)**: designan la fuente o el destino de un dato
- **Líneas de control**: gobiernan el acceso y el uso de las líneas de datos y direcciones
  - *Líneas de operación*: determinan el tipo de operación que debe realizar el slave
  - *Líneas de sincronización*: determinan el comienzo y final de cada transferencia
  - *Líneas de arbitraje*: determinan cuál de los elementos conectados puede usar el bus

En algunos buses una misma línea puede realizar más de una función

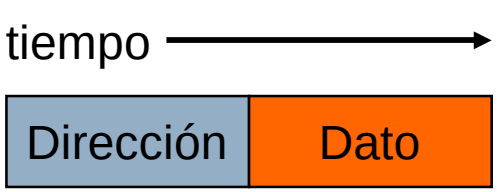
- **Líneas dedicadas**: tienen asignada una única función
- **Líneas multiplexadas**: realizan distintas funciones a lo largo del tiempo
  - *Ventajas*: menor número de líneas ⇒ ahorra espacio ⇒ menor coste
  - *Desventajas*: Circuitería más compleja y menor rendimiento (las funciones que realizan las líneas multiplexadas no pueden realizarse en paralelo)

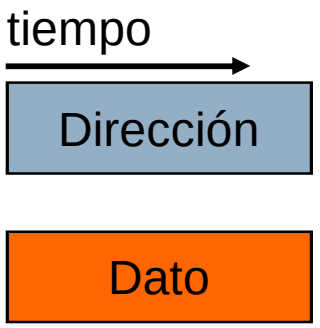


protocolos de transferencia

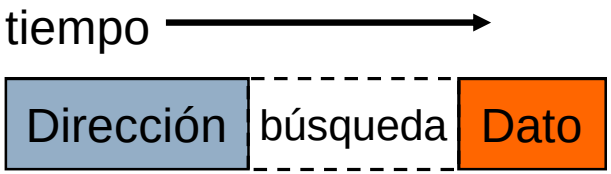
- **Función**:
  - Sincronizar los elementos implicados en una transferencia (master y slave)
  - Determinar el comienzo y el final de cada transferencia
- **Tipos de transferencia**:
  - Lectura
  - Escritura
  - Lectura de bloque
  - Escritura de bloque
  - Lectura-modificación-escritura
  - Lectura después de escritura
- **Tipos de protocolos de transferencia**:
  - Síncrono
  - Asíncrono
  - Semisíncrono
  - Ciclo partido

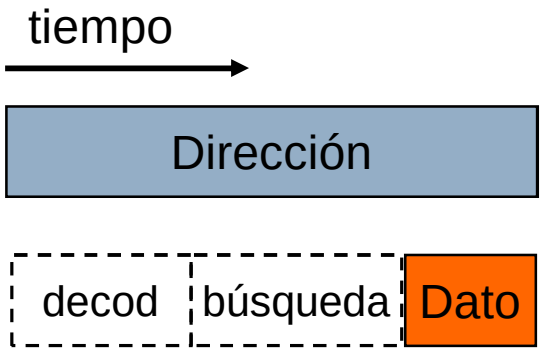
Tipos de transferencia





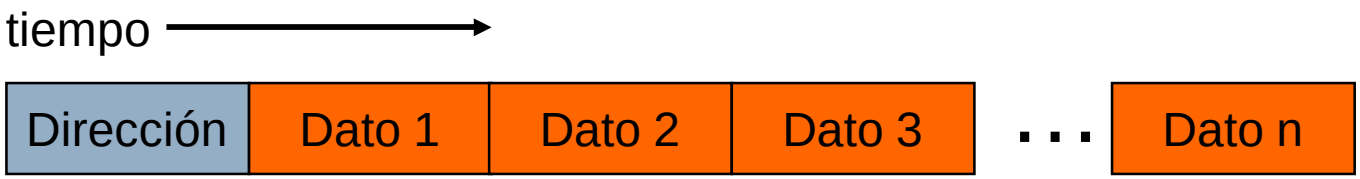
Operación de escritura



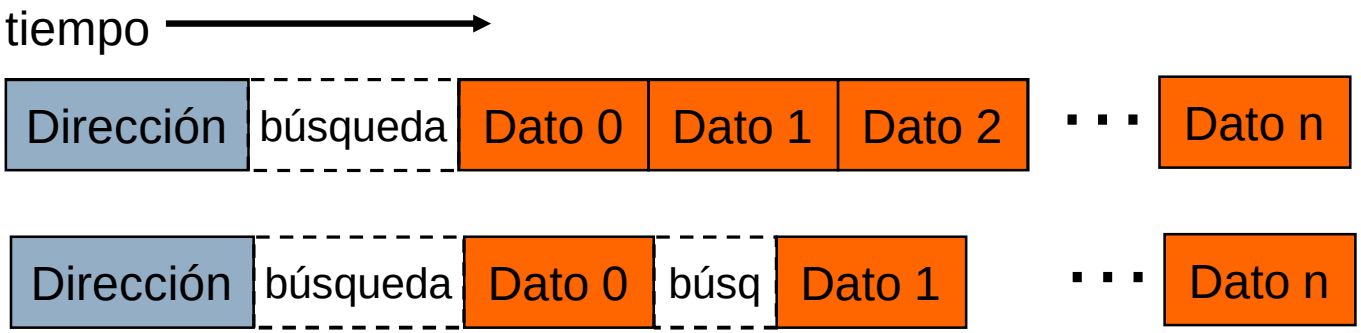


Operación de lectura

Tipos de transferencia

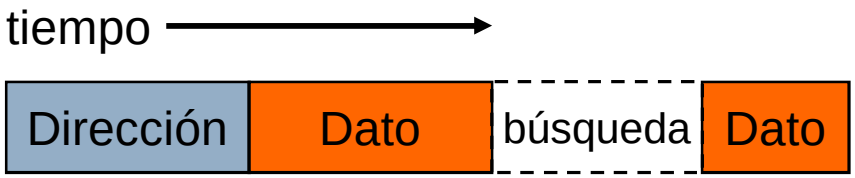


Operación de escritura de bloque



Operación de lectura de bloque

Tipos de transferencia



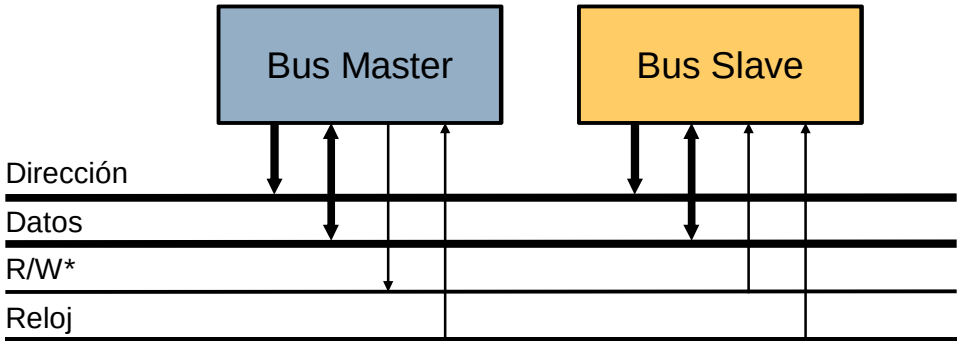
Operación de lectura tras escritura

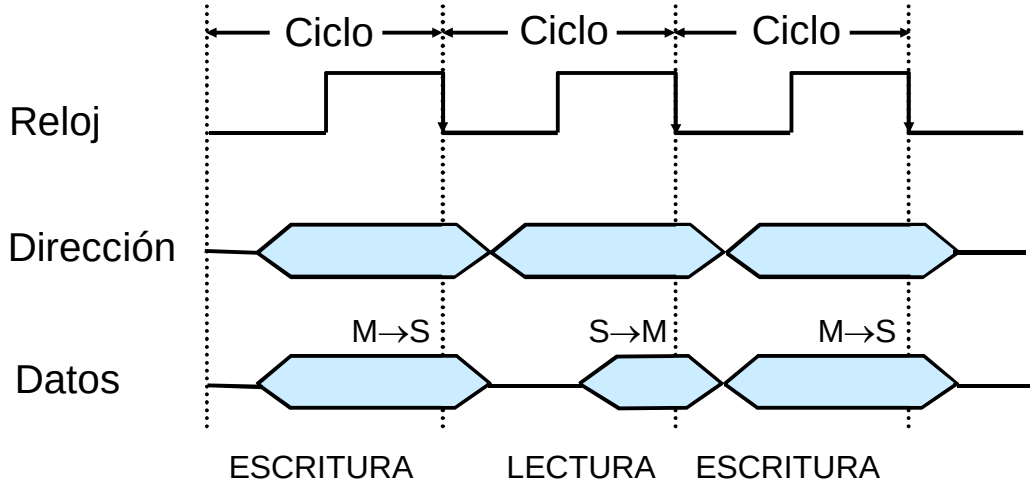
Operación de lectura, modificación y escritura

Protocolo de transferencia síncrono

- Las transferencias están gobernadas por una **única señal de reloj** compartida por todos los dispositivos
- 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 de transferencia síncrono

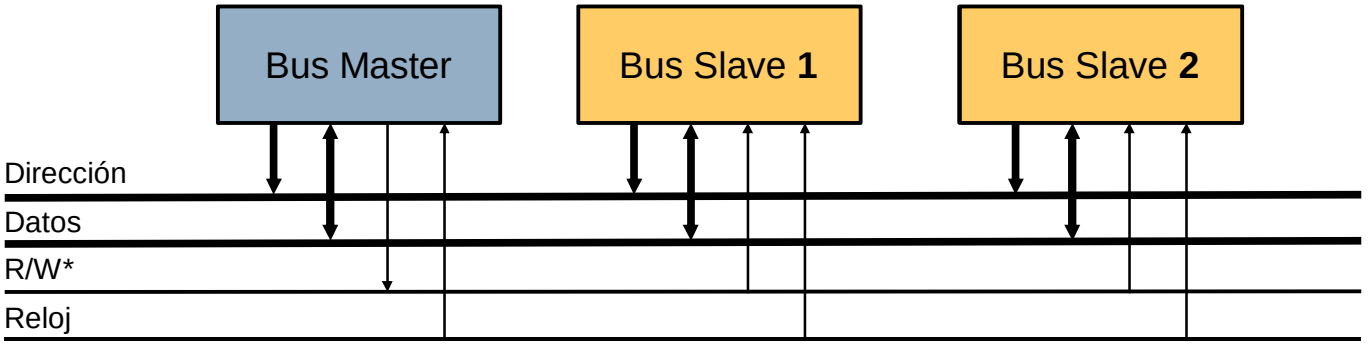




Protocolo de transferencia síncrono

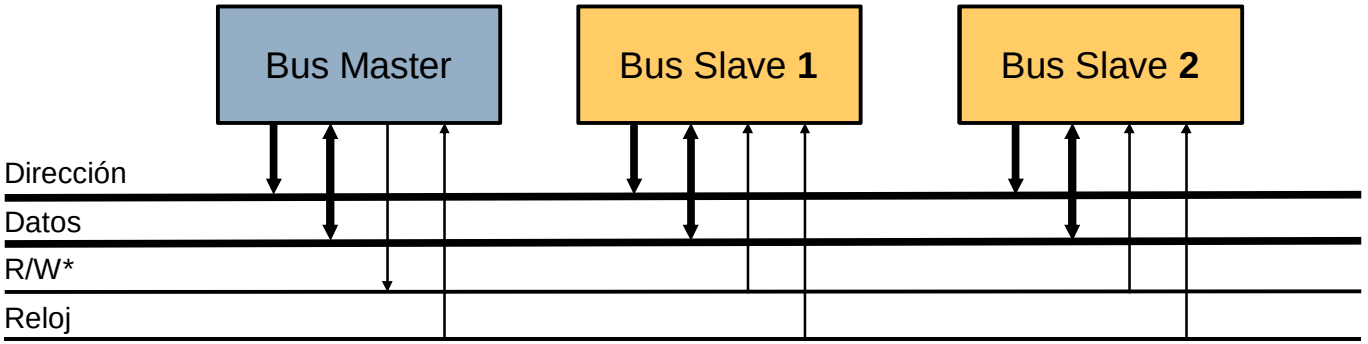
- *Ventajas*:
  - Simplicidad (de diseño y de uso)
  - Sólo se necesita una señal (reloj) para llevar a cabo la sincronización
  - Mayor velocidad (en relación a protocolo asíncrono)
- *Desventajas*:
  - El periodo de reloj **se tiene que adaptar a la velocidad del dispositivo más lento** (por lo que 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:
    - **Tiempo de desplazamiento relativo (*skew time*):** diferencia de tiempo entre la llegadas al receptor de dos señales que partieron del emisor simultáneamente, o la llegada de una señal a dos receptores distintos
    - **Cuanto mayor es el bus, mayor es el skew**

Protocolo de transferencia síncrono



Tiempo de decodificación: 30 ns
Tiempo de setup: 2 ns
Skew: 2 ns
Tiempo de búsqueda 1: 100 ns
Tiempo de búsqueda 2: 1000 ns
Si queremos que se haga una transferencia por ciclo:

Protocolo de transferencia síncrono



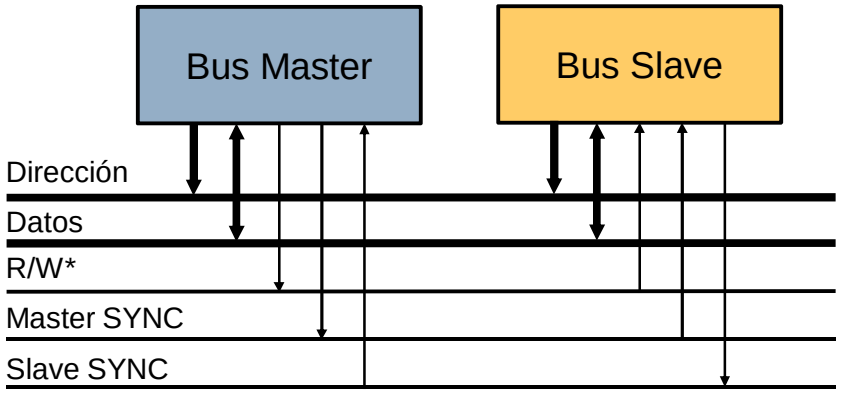
Tiempo de decodificación: 30 ns
Tiempo de setup: 2 ns
Skew: 2 ns
Tiempo de búsqueda 1: 100 ns
Tiempo de búsqueda 2: 1000 ns
Si queremos que se haga una transferencia por ciclo:

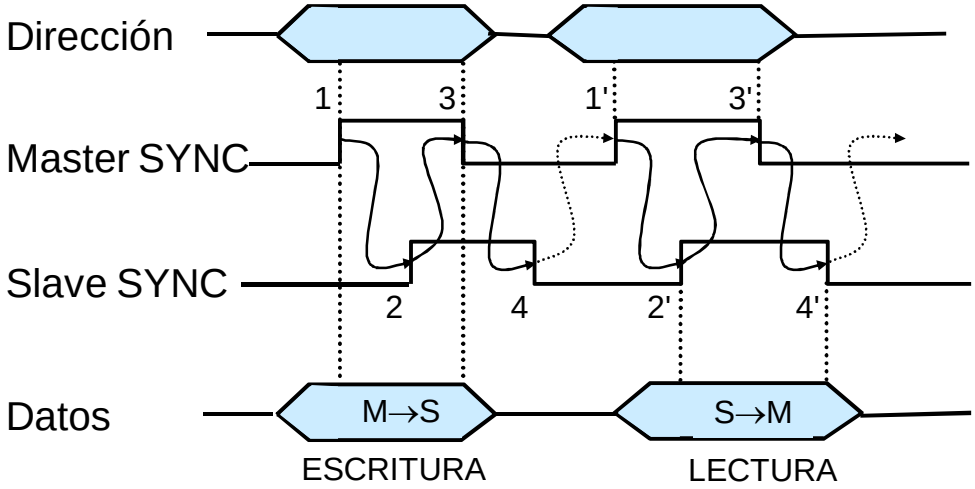
Tiempo de ciclo: 30+2+2+max(100,1000)= **1034ns**
Óptimo para el slave 2 pero muy ineficiente para el 1

Protocolo de transferencia asíncrono

- **La señal de reloj se sustituye por dos señales de sincronización:**
  - Master SYNC (procedente del master)
  - Slave SYNC (procedente del slave)
  - A cada flanco del 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)

Protocolo de transferencia asíncrono

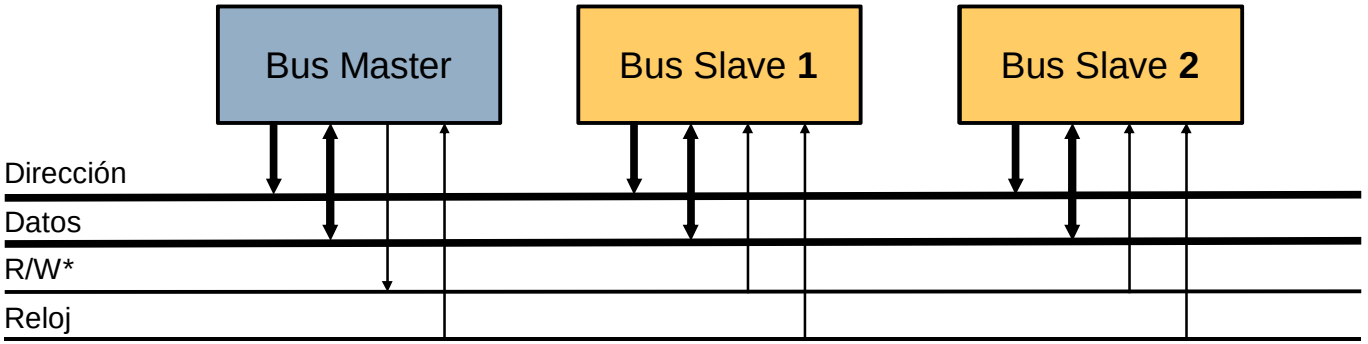




Protocolo de transferencia asíncrono

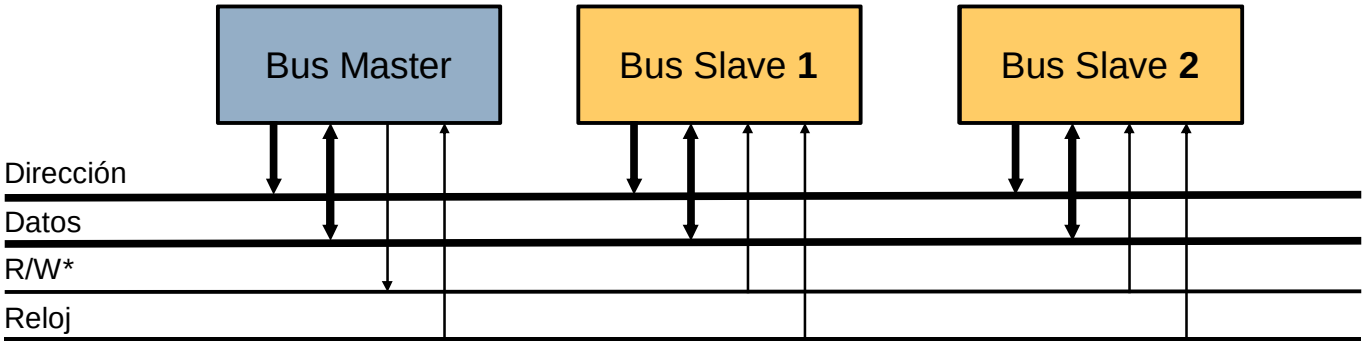
- *Ventajas:*
  - Facilidad para conectar elementos de diferentes velocidades
  - Fiabilidad: la recepción siempre se confirma
- *Desventajas:*
  - El intercambio de señales de control introduce retardos adicionales
  - A igualdad de velocidades de los dispositivos, menos eficiente que el síncrono
- *Ejemplos*: Unibus (PDP-11), MC680XX(10,20,30), Bus VME, FutureBus+

Protocolo de transferencia asíncrono



Tiempo de decodificación: 30 ns
Tiempo de intercambio de señales de sincronización: 10ns
Tiempo de búsqueda 1: 100 ns
Tiempo de búsqueda 2: 1000 ns
Si queremos que se haga una transferencia por ciclo:

Protocolo de transferencia asíncrono



Tiempo de decodificación: 30 ns
Tiempo de intercambio de señales de sincronización: 10ns
Tiempo de búsqueda 1: 100 ns
Tiempo de búsqueda 2: 1000 ns
Si queremos que se haga una transferencia por ciclo:

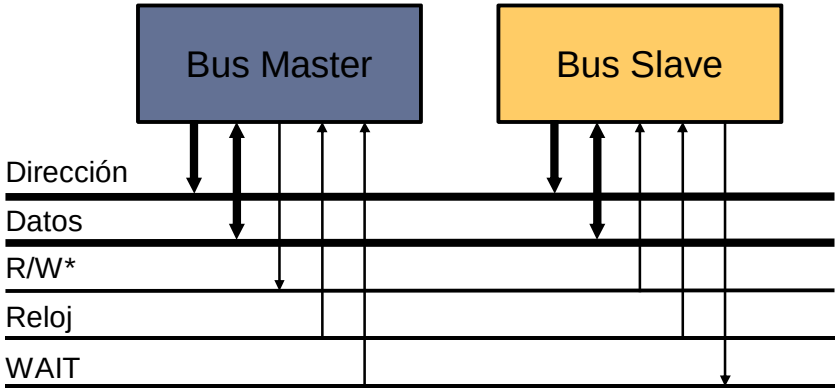
Tiempo de transferencia de lectura:

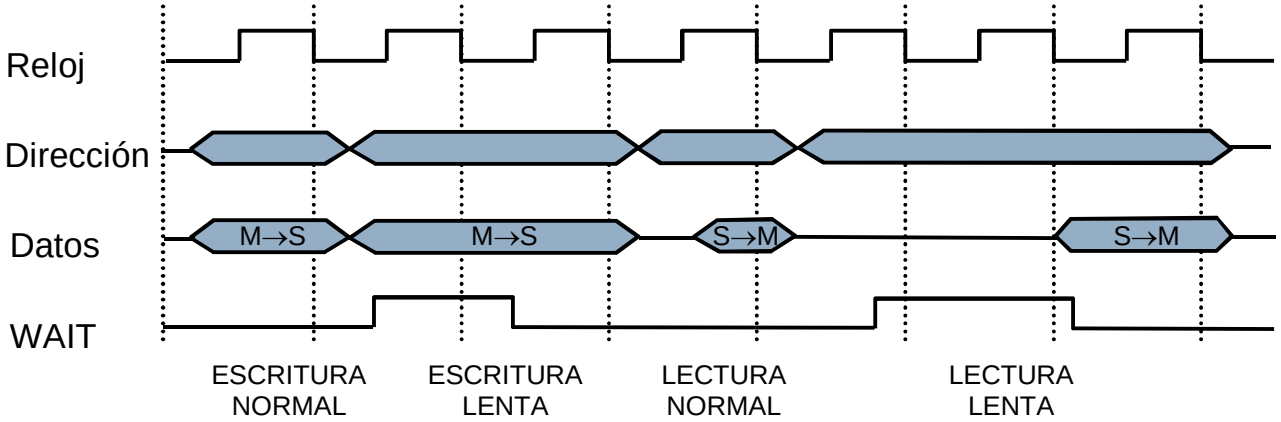
- slave1: 30+10+10+10+10+100 = 170ns
- slave2: 30+10+10+10+10+1000 = 1070ns
- tiempo medio: **620ns**

Protocolo de transferencia semi-síncrono

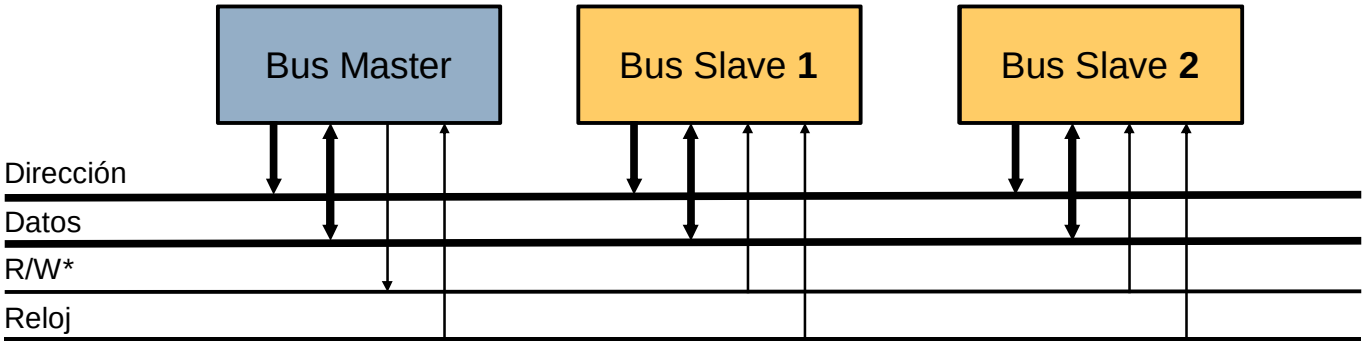
- Las transferencias se rigen por una **única señal de reloj**
- Cada transferencia puede ocupar uno o varios periodos de reloj
- Señal de WAIT: la activa el Slave si no es capaz de realizar la transferencia en el tiempo estipulado
  - **Dispositivos rápidos**: operan como en un bus síncrono
  - **Dispositivos lentos**: activan la señal de WAIT y congelan la actuación del Master
- *Ejemplos*: i80x86, MC68040, Bus PCI

Protocolo de transferencia semi-síncrono



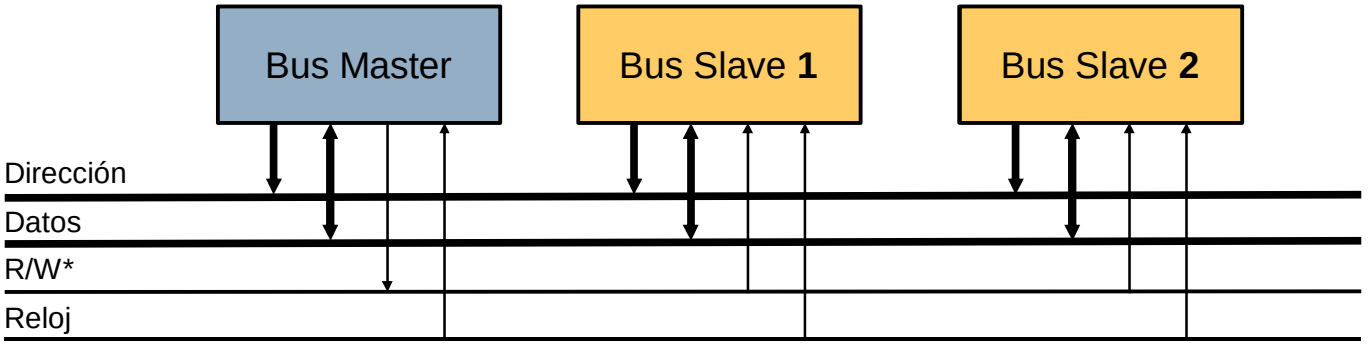


Protocolo de transferencia semi-síncrono



Tiempo de decodificación: 30 ns
Tiempo de setup: 2 ns ; Skew: 2 ns
Tiempo de búsqueda 1: 100 ns
Tiempo de búsqueda 2: 1000 ns
Queremos que los dispositivos rápidos hagan una transferencia por ciclo:

Protocolo de transferencia semi-síncrono



Tiempo de decodificación: 30 ns
Tiempo de setup: 2 ns ; Skew: 2 ns
Tiempo de búsqueda 1: 100 ns
Tiempo de búsqueda 2: 1000 ns
Queremos que los dispositivos rápidos hagan una transferencia por ciclo:

Tiempo de ciclo: 30+2+2+100= 134 ns

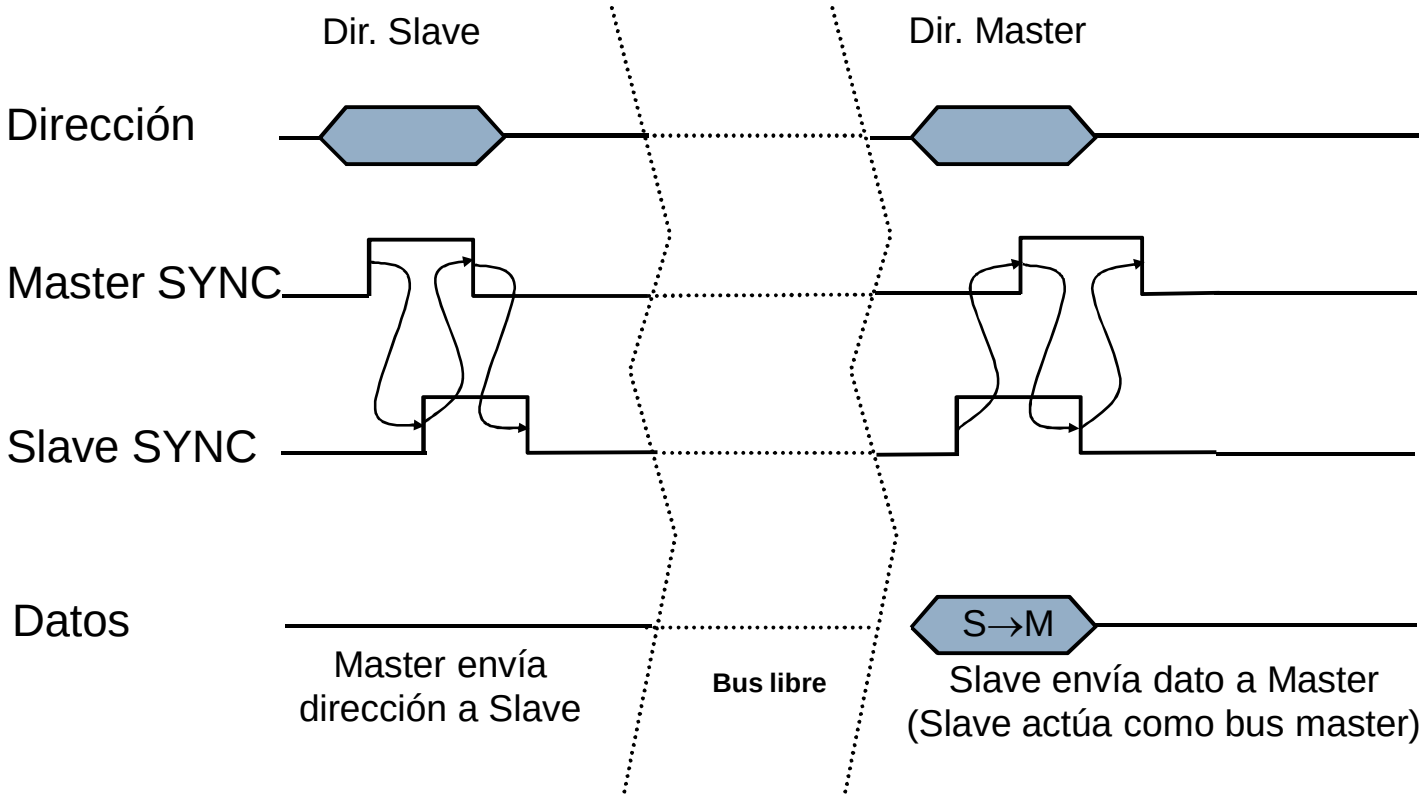
Tiempo de transferencia:

- slave1: 134 ns (1 ciclo)
- slave2: 134*8 ciclos= 1072 ns
- tiempo medio: **603 ns**

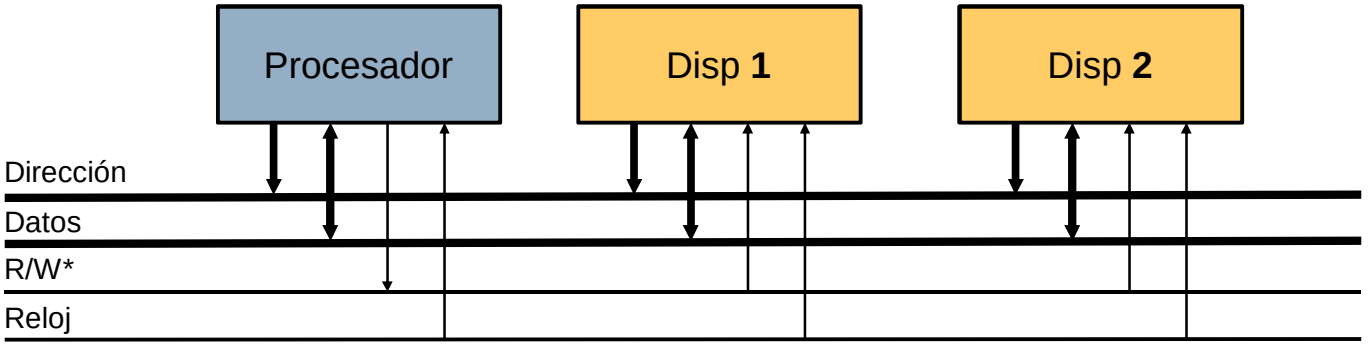
Transferencias de ciclo partido

- Mejora el rendimiento del bus en las operaciones de lectura
- El ciclo de lectura se divide en dos transacciones separadas
  - El Master envía al Slave la petición de lectura y deja el bus libre
  - Cuando el Slave dispone del dato solicitado, inicia un ciclo de bus y envía el dato al Master
    (Slave actúa como master del bus)
- *Desventajas*:
  - Lógica más compleja: ambos dispositivos deben ser capaces de actuar como Master y como Slave
  - Necesidad de incluir un protocolo de arbitraje
- *Ejemplos*: VAX-11/780, iAPX-432

Transferencias de ciclo partido

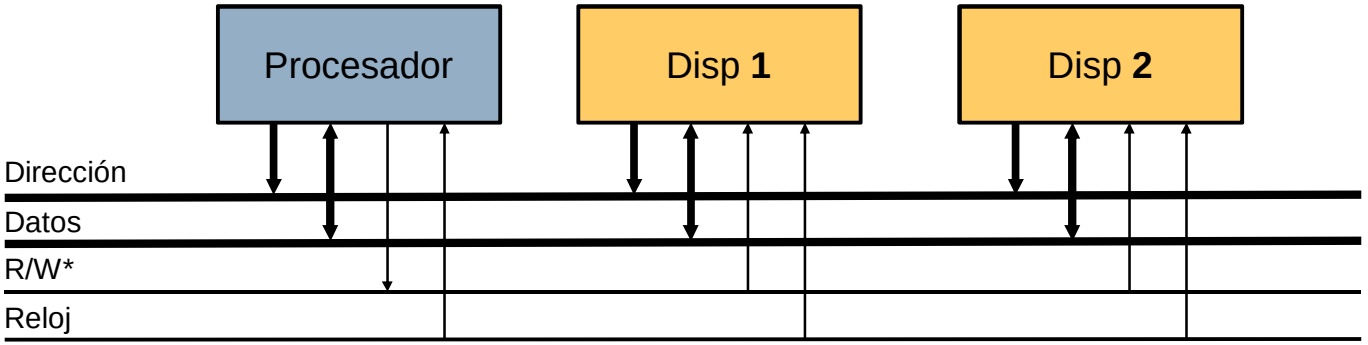


Transferencia ciclo partido asíncrona



Tiempo de decodificación: 30 ns
Tiempo de intercambio de señales de sincronización: 10ns
Tiempo de búsqueda 1: 100 ns
Tiempo de búsqueda 2: 1000 ns
Queremos que un dispositivo rápido pueda hacer una transferencia por ciclo:

Transferencia ciclo partido asíncrona



Tiempo de decodificación: 30 ns
Tiempo de intercambio de señales de sincronización: 10ns
Tiempo de búsqueda 1: 100 ns
Tiempo de búsqueda 2: 1000 ns
Queremos que un dispositivo rápido pueda hacer una transferencia por ciclo:

Transferencias de lectura:

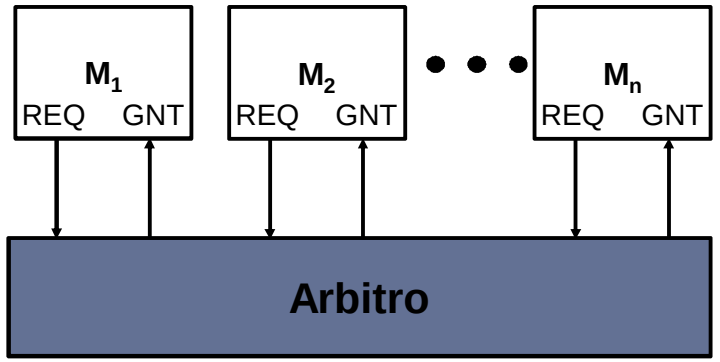
- envío dirección:
  30+10+10+10+10 = 70ns
- envío dato:
  30+10+10+10+10 = 70ns
- ocupación del bus: **140ns**
- tiempo mínimo para la lectura:
  - disp1: 240 ns
  - disp2: 1140 ns
  - tiempo medio: **690 ns**

Protocolos de arbitraje

- **Función**:
  - Garantizar el acceso al bus libre de conflictos cuando existen varios masters alternativos
    - Procesador + Controladores DMA
    - Procesador + Procesador de E/S + Coprocesador matemático + ...
    - Sistema multiprocesador
    - Buses con protocolo de ciclo partido
- **Tipos** de protocolos de arbitraje
  - *Centralizados*: existe un **árbitro** del bus o **master principal** que controla el acceso al bus
    - Protocolo en estrella
    - Protocolo daisy-chain
  - *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

Protocolo de arbitraje en estrella

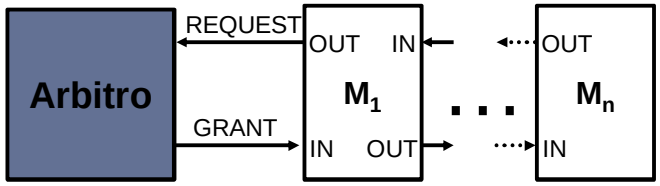
- Cada master se conecta al árbitro mediante dos líneas individuales:
  - BUS REQUEST (REQ): línea de petición del bus
  - BUS GRANT (GNT): línea de concesión del bus



- **Varias peticiones** de bus pendientes: el árbitro puede aplicar distintos algoritmos de decisión
  - FIFO
  - Prioridad fija o variable
- *Ventajas:*
  - Algoritmos de arbitraje simples
  - Pocos retardos de propagación de las señales (comparado con 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

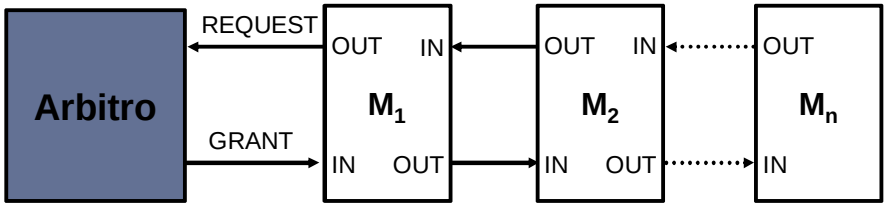
Protocolo de arbitraje daisy-chain

- Dos líneas de arbitraje comunes
  - BUS REQUEST: Línea de petición del bus
  - BUS GRANT: Línea de concesión del bus



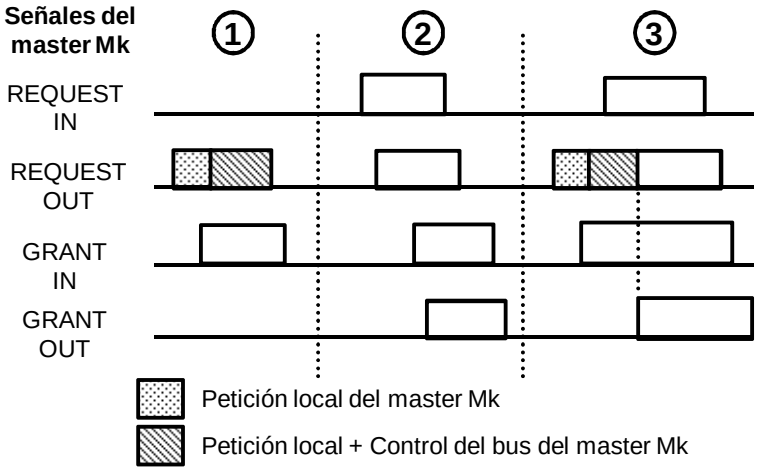
- *Funcionamiento:*
  - El master que quiere tomar el control del bus activa REQUEST
  - Los restantes masters propagan REQUEST hasta el árbitro
  - El árbitro genera un pulso en la línea GRANT
  - Si un master detecta un flanco de subida GRANT y no pidió el bus, lo propaga al siguiente
  - Si un master detecta un **flanco de subida** en GRANT y tiene una petición pendiente, toma el control del bus
- *Prioridades:*
  - El orden en que se conectan los Masters a las líneas de arbitraje determina la prioridad de los mismos
- *Ejemplo*: bus IDE

Protocolo de arbitraje daisy-chain



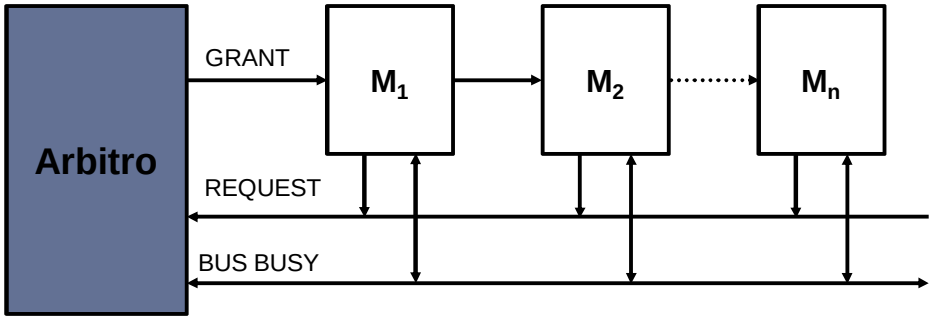
Situaciones típicas:

1) M_k pide el Bus
   Cuando Mk recibe GRANT toma el control del bus
2) M_k recibe una petición de M_{k+1}
   Cuando Mk recibe GRANT la propaga a M_{k+1}
3) M_k pide el bus
   Cuando M_k recibe GRANT toma el control del bus
   Mientras M_k controla el bus, recibe una petición de M_{k+1}
   Cuando M_k termina de usar el bus, propaga GRANT a M_{k+1}



Protocolo daisy-chain con 3 hilos

- *Líneas de arbitraje*:
  - **BUS REQUEST**: línea de petición de bus
  - **BUS GRANT**: línea de concesión del bus
  - **BUS BUSY**: línea de bus ocupado

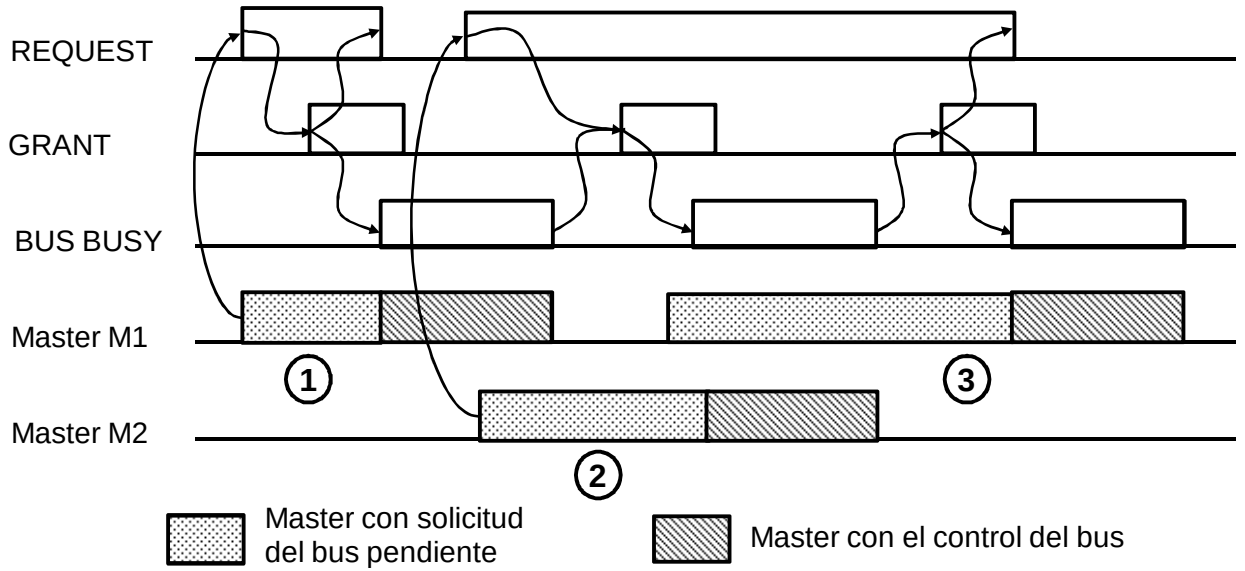


- ***Funcionamiento*:**
  - Cuando un master toma el control del bus activa BUS BUSY
  - Un master solicita el bus activando REQUEST
  - El árbitro activa GRANT cuando detecta REQUEST activado y BUS BUSY desactivado
  - Si un master recibe GRANT y no ha pedido el bus, entonces propaga GRANT al siguiente
  - Un master puede tomar el control del bus si se cumplen las tres condiciones siguientes:
    - a) El master tiene una petición local pendiente
    - b) La línea BUS BUSY está inactiva
    - c) El master recibe el **flanco de subida** de la señal GRANT
- *Ejemplo*: VME

protocolos de arbitraje

Situaciones típicas

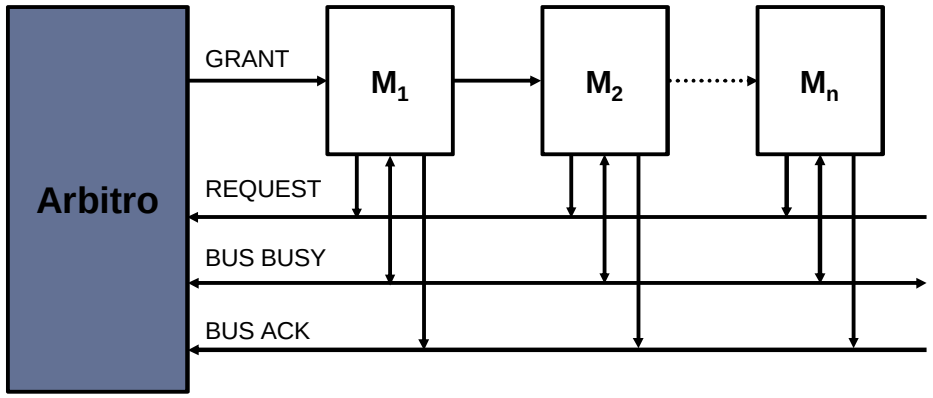
1) M_1 solicita el bus. Bus libre
   Cuando M_1 detecta el flanco de GRANT toma el control del bus
2) M_2 solicita el bus. Bus ocupado
   M_2 debe esperar bus libre y detectar el flanco de GRANT
3) M_1 solicita el bus. Bus libre. GRANT activado
   M_1 debe esperar que GRANT baje y vuelva a subir

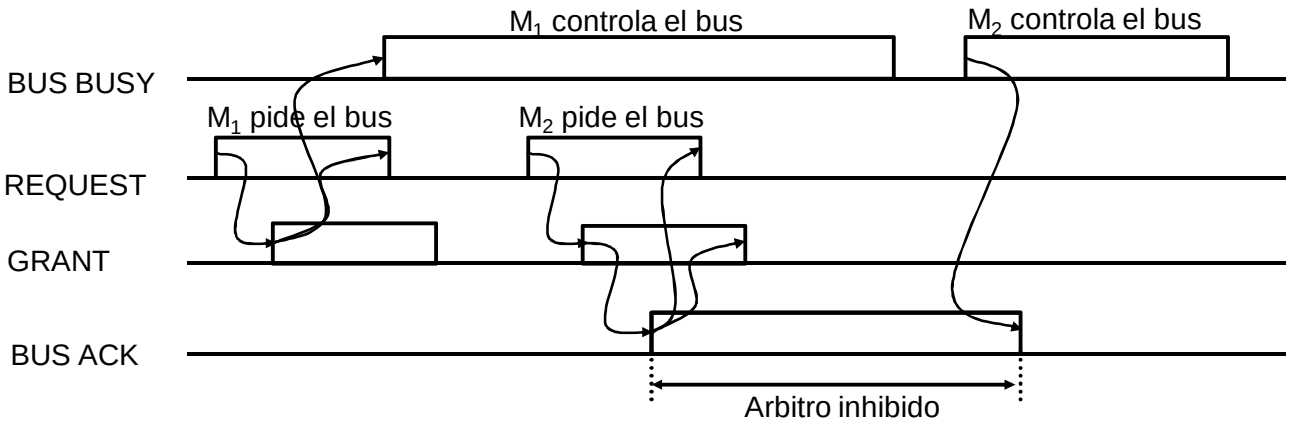


Protocolo daisy-chain con 4 hilos

- Permite solapar la transferencia del ciclo actual con el arbitraje del ciclo siguiente
- *Líneas de arbitraje*:
  - **BUS REQUEST**: línea de petición de bus
  - **BUS GRANT**: línea de concesión del bus
  - **BUS BUSY**: línea de bus ocupado
  - **BUS ACK**: línea de confirmación
    - La activa el master que solicitó el bus en respuesta BUS GRANT, cuando el bus está ocupado
    - Cuando está activada el árbitro queda inhibido
- *Ejemplo*: Unibus (PDP-11), MC68000

Protocolo daisy-chain con 4 hilos





Protocolo de arbitraje distribuido: líneas de identificación

- *Funcionamiento:*
  - Cuando un master quiere tomar el control del bus activa su línea de identificación
  - Cada línea de identificación tiene asignada una prioridad: Prioridad(ID_0)<Prioridad(ID_1)< ....< Prioridad(ID_{n-1})
  - Si varios masters activan simultáneamente sus líneas de identificación, gana el de mayor prioridad
  - Funcionamiento alternativo: las prioridades pueden ser variables
- *Desventajas:*
  - Número de dispositivos limitado por el número de líneas de arbitraje
- *Ejemplos:*
  - **Prioridad fija**: VAX SBI, SCSI
  - **Prioridad variable**: DEC 70000/10000 AXP, AlphaServer 8000

Protocolo de arbitraje distribuido: códigos de identificación

- *Funcionamiento*:
  - Cada master tiene un código de identificación de n bits (máximo 2^n masters)
  - Existen n líneas de arbitraje: ARB_0, ARB_1, ..., ARB_{n-1}
  - Cuando un master quiere tomar el control del bus pone su código en las n líneas de arbitraje
  - Si varios masters compiten por el bus, gana el de mayor identificador
- Requiere menos líneas de control que con líneas de identificación pero necesita lógica de decodificación mayor coste y retardo
- *Ejemplos*: Multibus II, Futurebus+

Conclusiones

- Los buses permiten la comunicación entre los dispositivos
- El rendimiento de un bus depende de el ancho de banda de datos, el protocolo de transferencia, la longitud y el número de dispositivos conectados
- Los protocolos síncronos (o semi-síncronos) son más adecuados para buses pequeños que deban proporcionar alto rendimiento
- Los asíncronos son más adecuados para buses grandes con elementos heterogéneos conectados
- Si hay varios “masters” en un mismo bus, se debe incluir un mecanismo de arbitraje que evite las colisiones