Aulas 11 e 12

- Noções gerais sobre organização de barramentos de dados
- Protocolos síncrono, semi-síncrono e "*handshaken*"
- Ligação entre endereçamento e transferência de dados: Microciclo e "*Merged*"
- Multiplexagem entre endereços e dados
- Ciclos com transferências múltiplas: "*Read-Modify-Write*", "*Read-After-Write*", "*Block transfer*"
- Barramentos "*Multi-Master*"
- Políticas e tipos de arbitragem

José Luís Azevedo, Arnaldo Oliveira, Tomás Silva, Bernardo Cunha

Introdução (1)

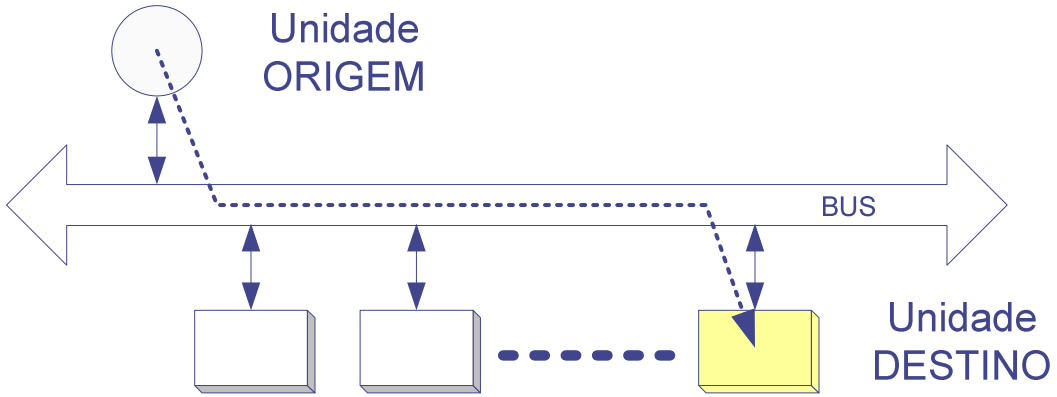
- Barramentos: interligação dos blocos de um sistema de computação
  - CPU, memória, unidades de I/O
- **Barramento de dados**:
  - Assegura a transferência de informação entre os blocos
  - O número de linhas (largura do barramento) determina quantos bits podem ser transferidos simultaneamente; a largura do barramento é um fator determinante no desempenho do sistema
- **Barramento de endereços**:
  - Especifica a origem/destino da informação
  - O número de linhas determina a capacidade máxima de memória que o sistema pode ter (2^N palavras, sendo N o número de bits do barramento de endereços)
- **Barramento de controlo**:
  - Composto por sinais que especificam operações e sinalizam eventos ou condições de múltiplas naturezas

Introdução (2)

- Tipos de dispositivos ligados a um barramento:
  - **Master** – Dispositivo que pode iniciar e controlar uma transferência de dados (exemplos: Processador, Módulo de I/O com DMA)
  - **Slave** – Dispositivo que só responde a pedidos de transferências de dados, i.e., não tem capacidade para iniciar uma transferência (exemplos: Memória, Módulo de I/O sem DMA)
- **Barramento de um só Master**: só há um dispositivo no barramento com capacidade iniciar e controlar transferências de informação
- **Barramento Multi-Master**: mais que um dispositivo capaz de iniciar e controlar transferências de informação (exemplos: vários CPUs, 1 ou mais controladores de DMA, um ou mais módulos de I/O com DMA)
- **Barramento paralelo**: os dados são transmitidos em paralelo (através de N Linhas)
- **Barramento série**: os dados são transmitidos em série (sequencialmente no tempo) através de 1 única linha de comunicação

Introdução (3)

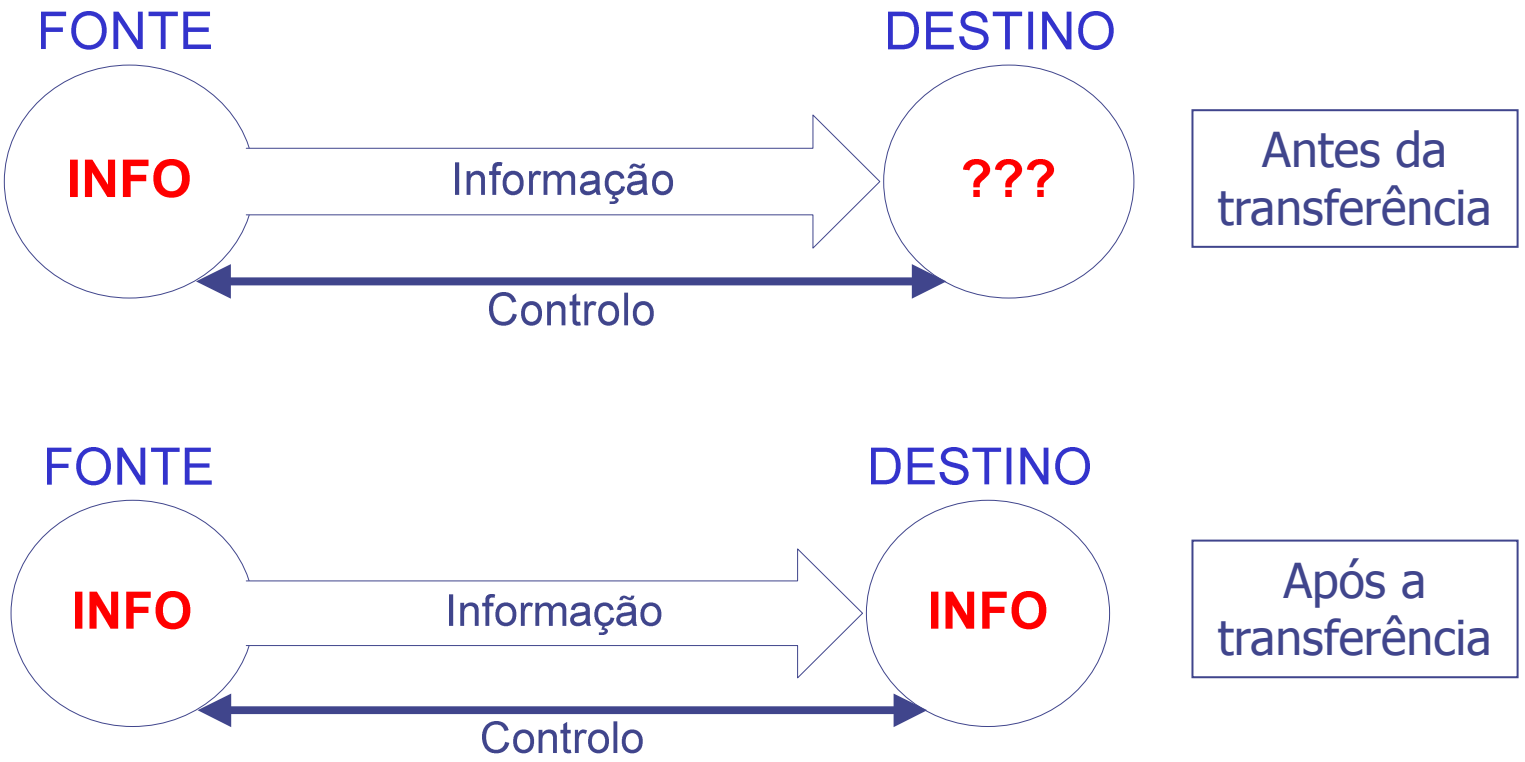
- Em geral os sistemas computacionais são compostos por diversas unidades que necessitam de trocar informação entre si
- A topologia em barramento permite que diversos módulos compartilhem a mesma ligação física (desde que os requisitos elétricos e físicos sejam respeitados)



- A partilha de um recurso (barramento) requer ainda um protocolo para seleção dos interlocutores
- A seleção do interlocutor é tipicamente efetuada através de um mecanismo de **endereçamento**

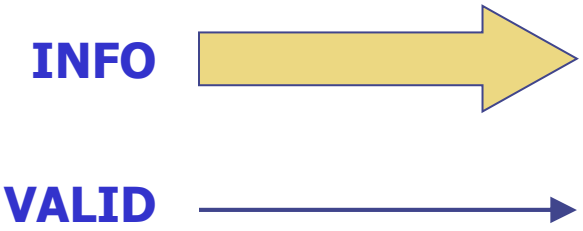
Transferência ponto a ponto

- O processo de transferência de informação pode ser descrito como uma sequência de **ações elementares**.



Transferência ponto a ponto

- Protocolo da ligação:
  - Especifica a **sequência de ações elementares** que coordenam a informação e o respetivo controlo
  - A ativação de cada ação elementar do protocolo está associada com o estado ou mudança de estado de uma variável booleana (sinal)
  - Essa variável pode estar associada a uma linha física ou estar codificada: N linhas podem servir para codificar 2^N ações.
- Ciclo:
  - **Sequência completa de ações elementares** que permitem transferir uma "unidade" de informação da fonte para o destino.
- Exemplo:



  - **VALID**: Sinal ativado pela fonte e usado pelo destino para armazenar INFO. É ativado quando as linhas de INFO já estão estáveis.

Transferência ponto a ponto

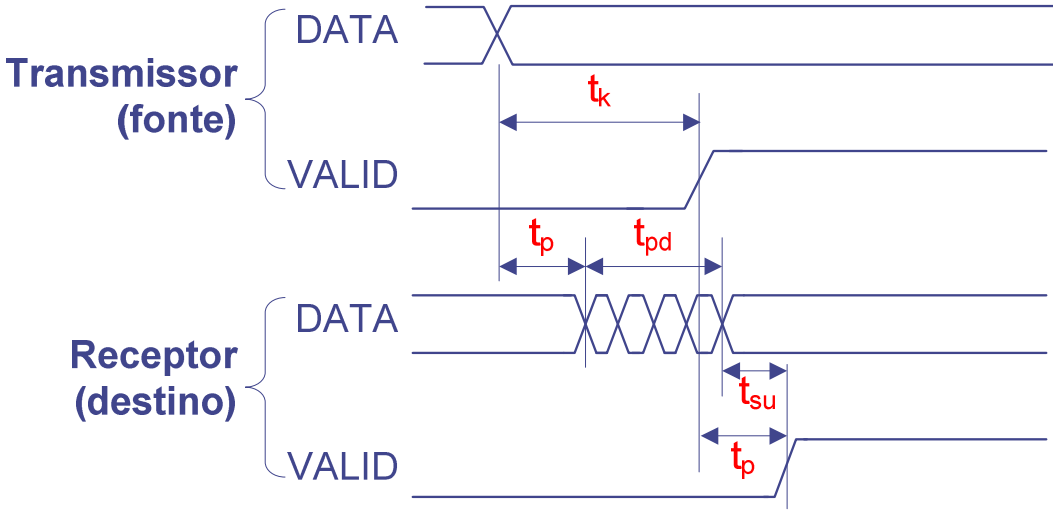
- Fluxo de ações elementares numa transferência:

FONTE coloca INFO no barramento

FONTE ativa sinal VALID

DESTINO armazena INFO em registo

- Diagrama temporal:



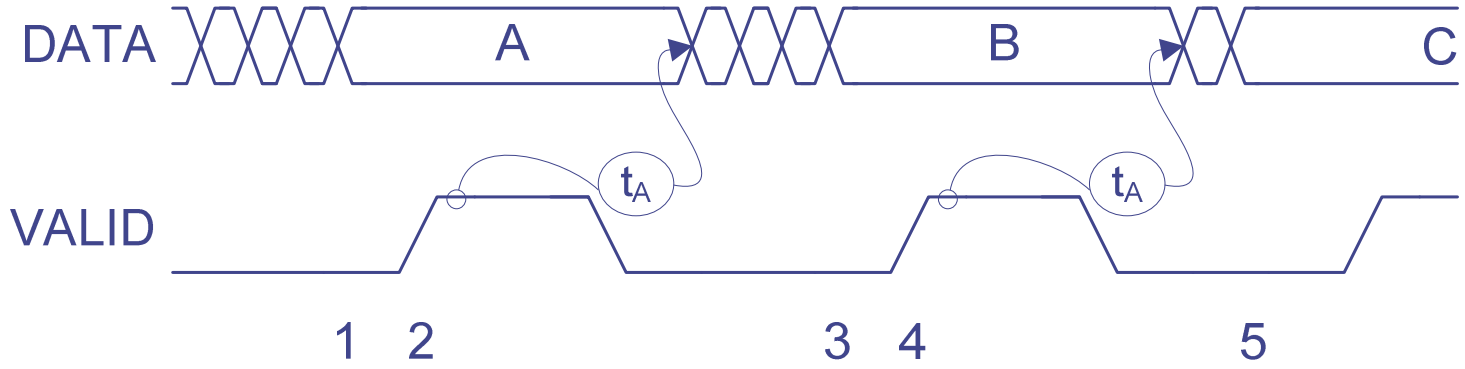
t_K – tempo de *setup* no tx *(source deskew)*

t_p, t_{pd} – tempos de propagação e *skew*

t_{su} – tempo de *setup* no rx

Transferência múltipla SÍNCRONA

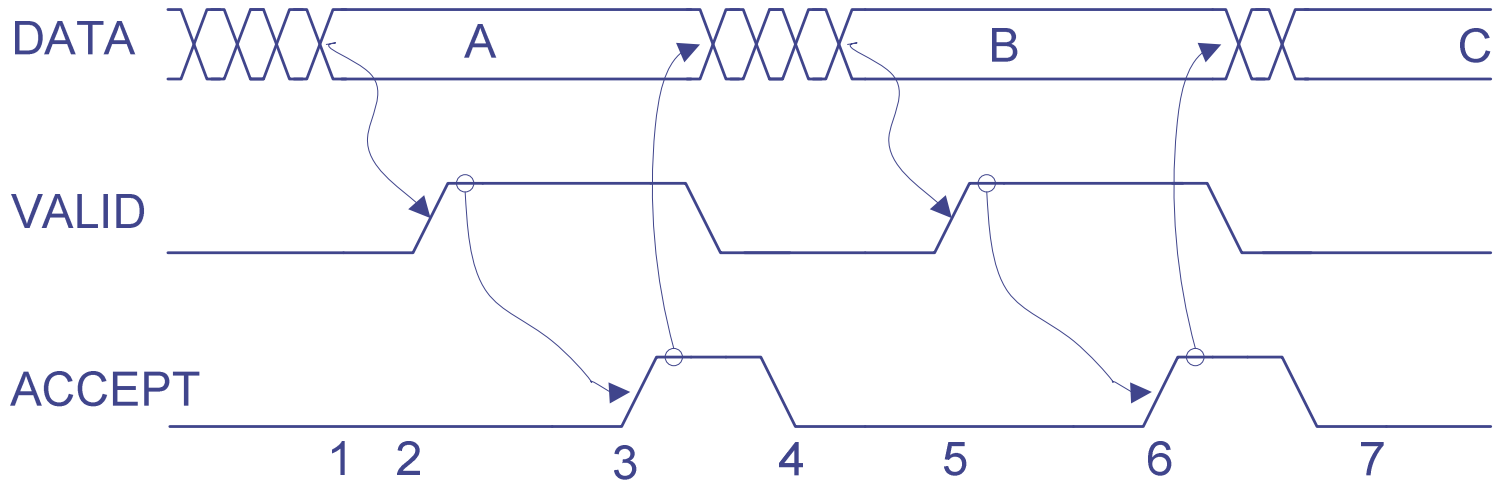
- Não utiliza sinais de reconhecimento (i.e. não há qualquer sinal que indique que a informação tenha sido consumida pelo destino)



- A lógica de controlo da FONTE assume que a informação foi armazenada pelo DESTINO ao fim de um tempo t_A
- A FONTE tem controlo total das temporizações, sincronizando a transferência
- **O que acontece se o DESTINO se atrasar ou não existir?**
- Sistemas heterogéneos têm que ser dimensionados de acordo com a velocidade da unidade mais lenta

Transferência múltipla ASSÍNCRONA ("handshaken")

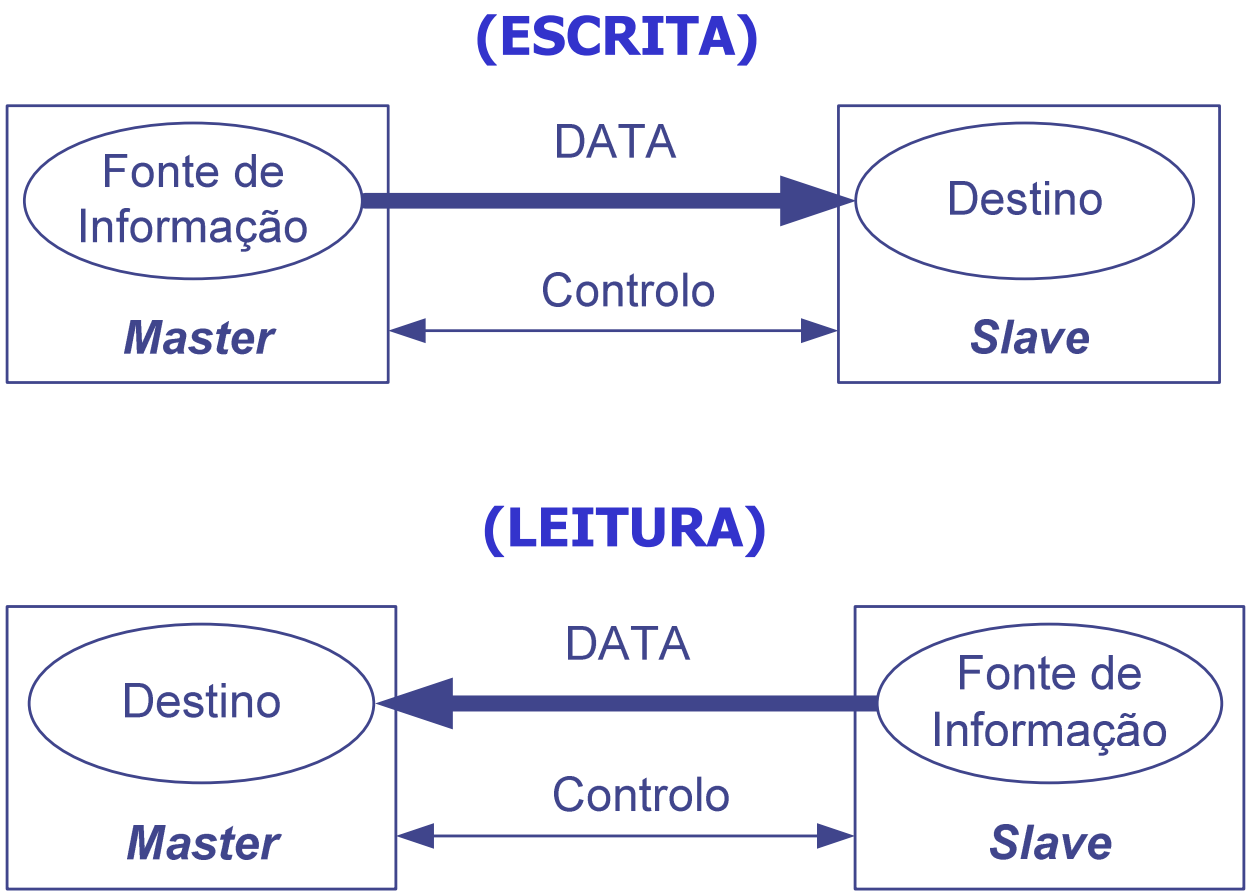
- Utiliza sinais de protocolo para assegurar a correta troca de informação entre a origem e o destino
- Pode ser utilizada na comunicação com dispositivos muito lentos (quando o tempo que decorre desde o pedido da operação até à sua conclusão é longo e variável)



- Sinal "ACCEPT", ativado pelo DESTINO (**Significado**: informação já foi usada / armazenada – pode ser retirada pela FONTE)
- Cooperação entre a FONTE e o DESTINO ("**handshaking**")

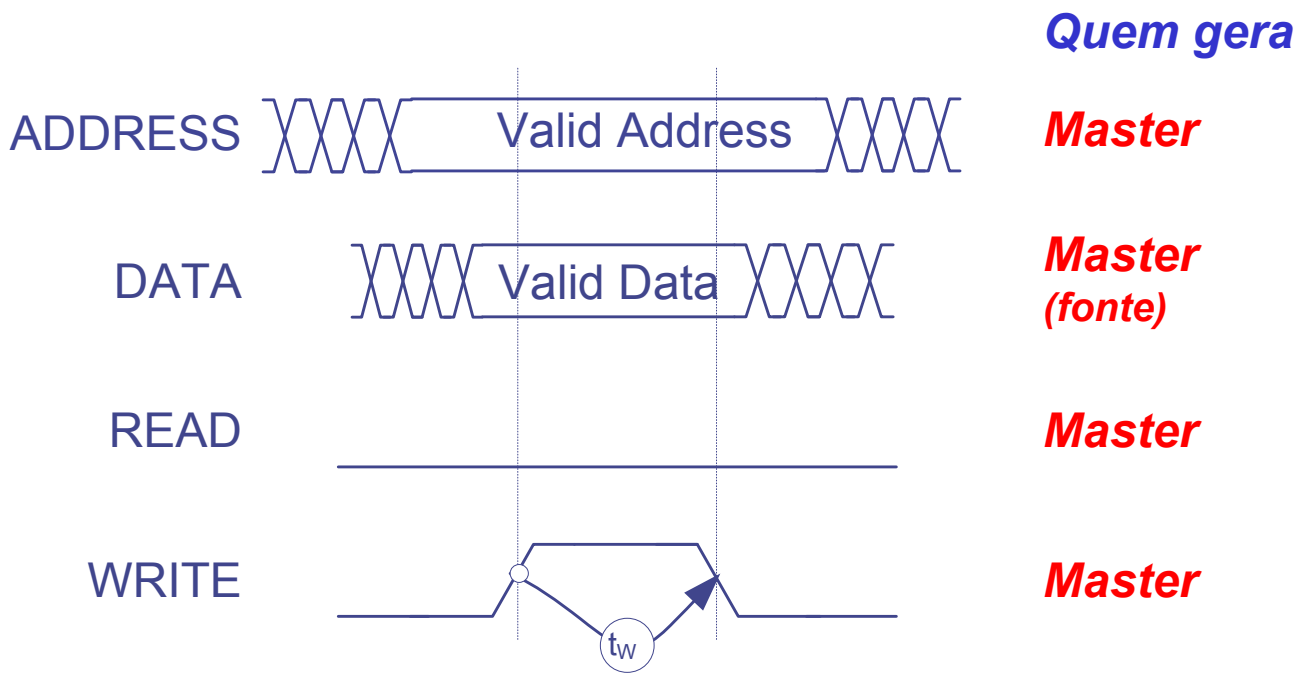
Controlo da transferência de informação

- Protocolos de escrita (Write) e de leitura (Read)



Controlo da transferência de informação

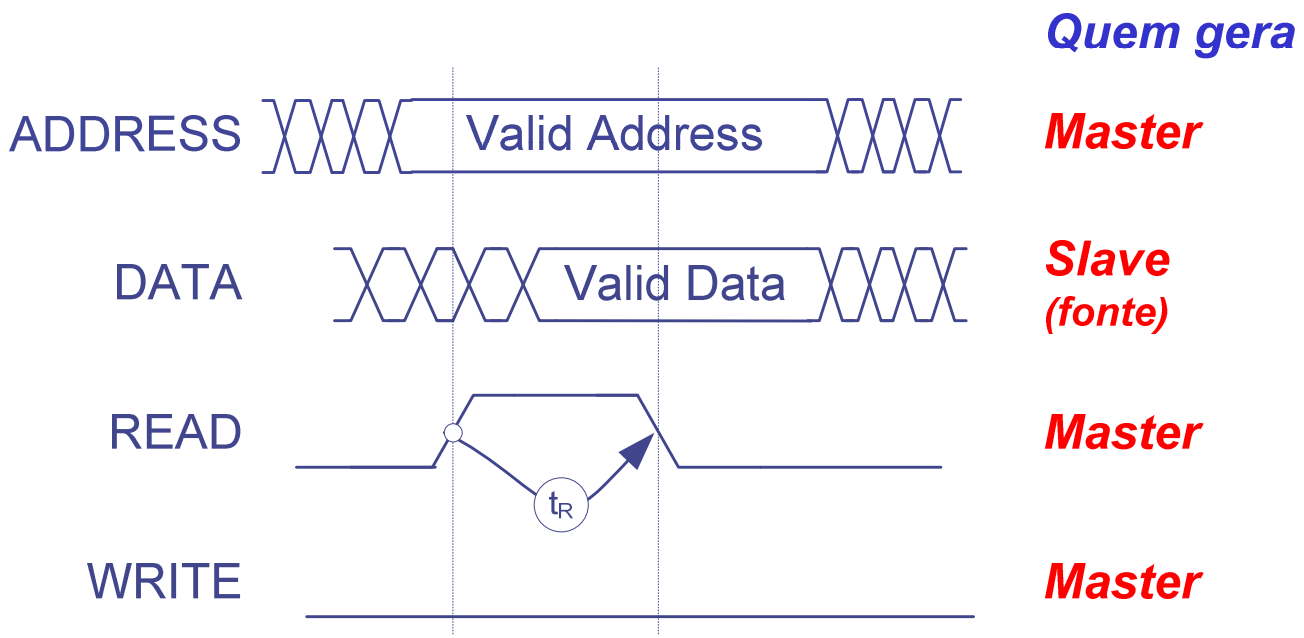
- Exemplo de **protocolo síncrono** com **sinais independentes** de Read e Write (operação de **escrita**)



Operação de ESCRITA

Controlo da transferência de informação

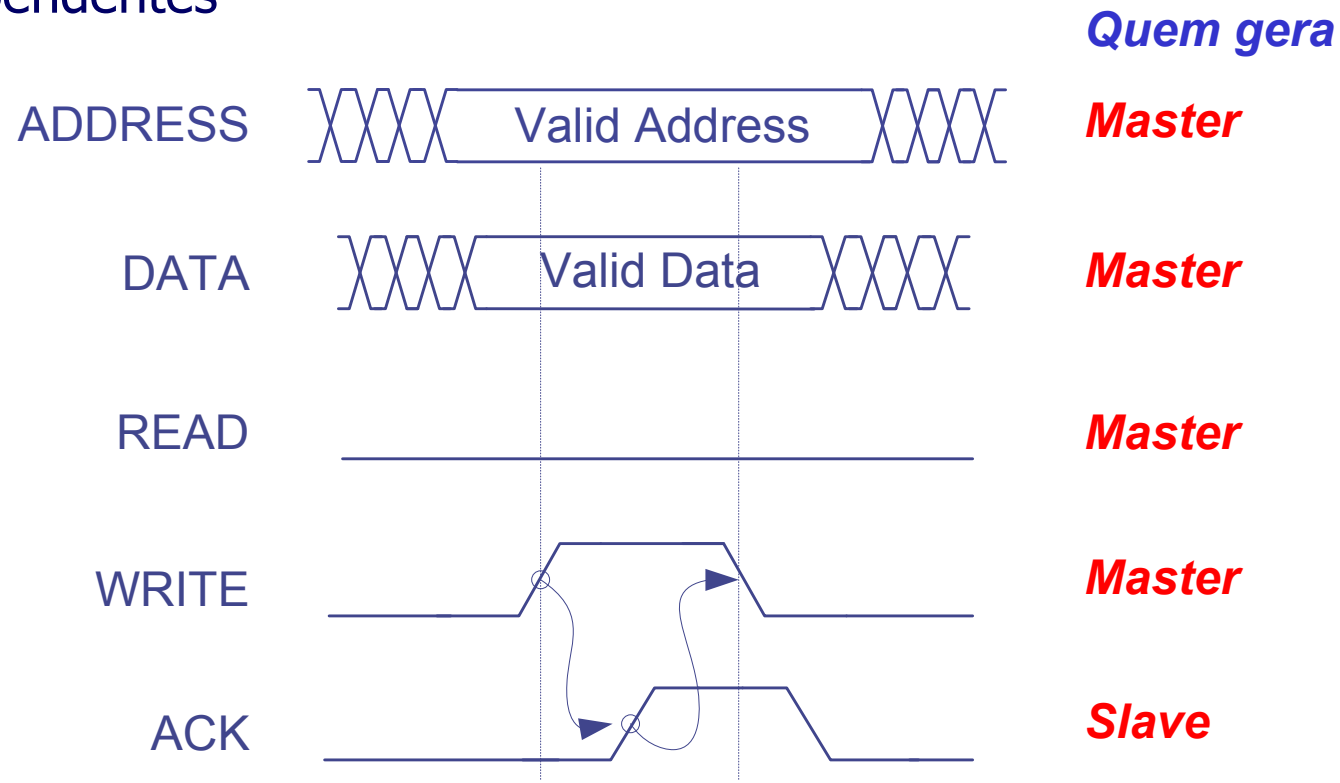
- Exemplo de **protocolo síncrono** com **sinais independentes** de Read e Write (operação de **leitura**)



Operação de LEITURA

Controlo da transferência de informação

- Exemplo de **protocolo "handshaken"**, com sinais de Read e Write independentes

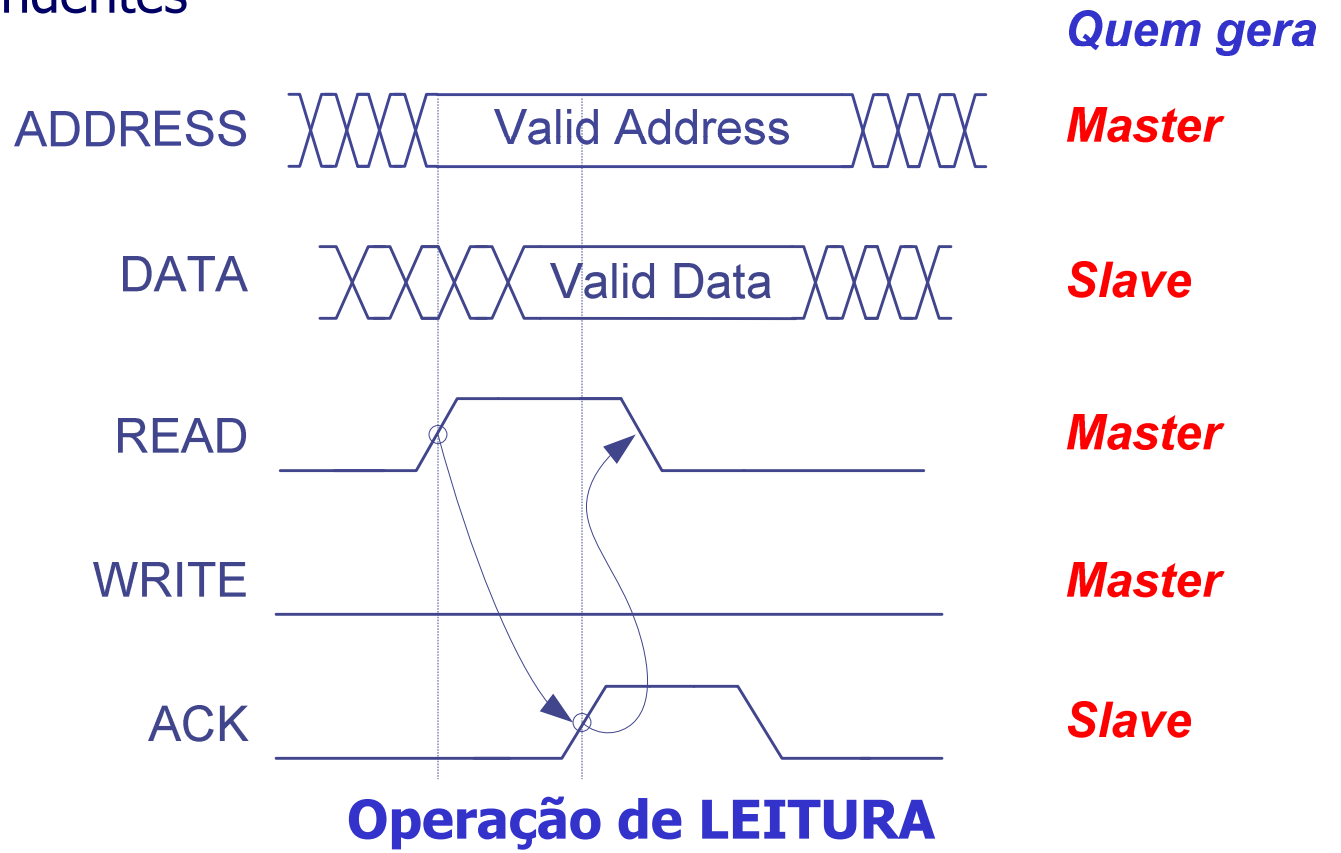


Operação de ESCRITA

- ACK = acknowledge (o mesmo que "accept")

Controlo da transferência de informação

- Exemplo de **protocolo "handshaken"**, com sinais de Read e Write independentes

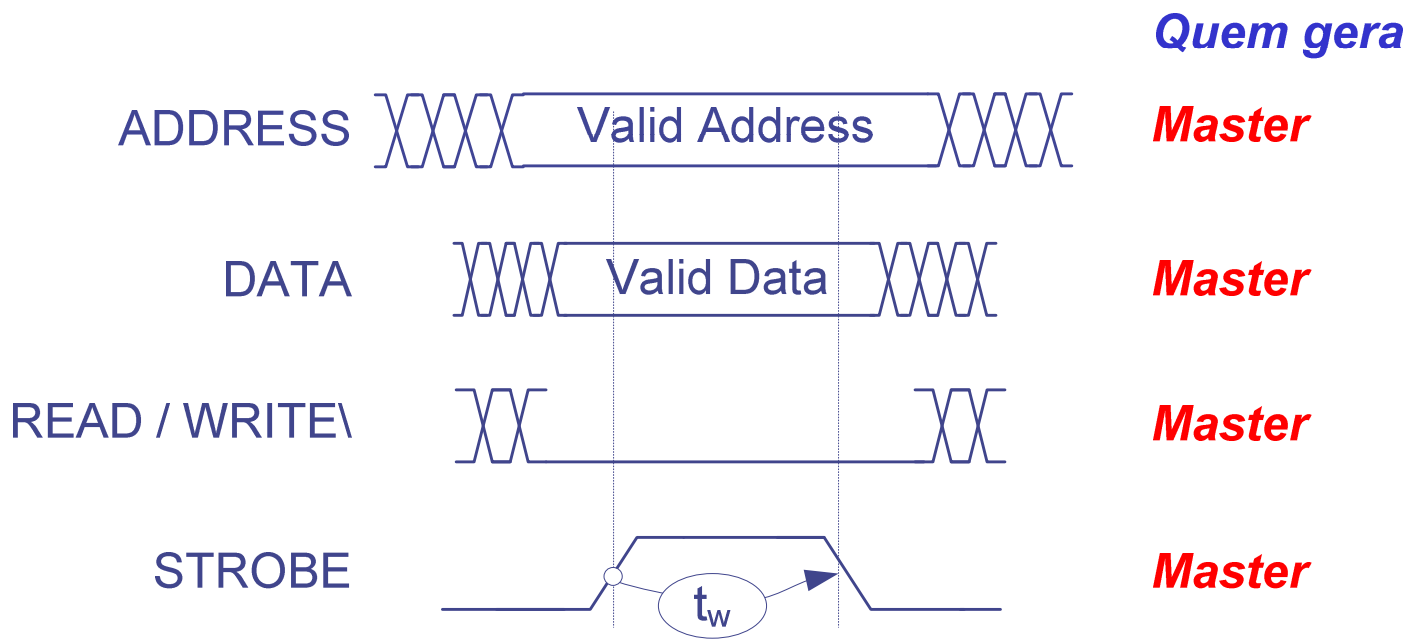


Operação de LEITURA

- **ACK** = *acknowledge* (significa "informação válida disponível no barramento")

Controlo da transferência de informação

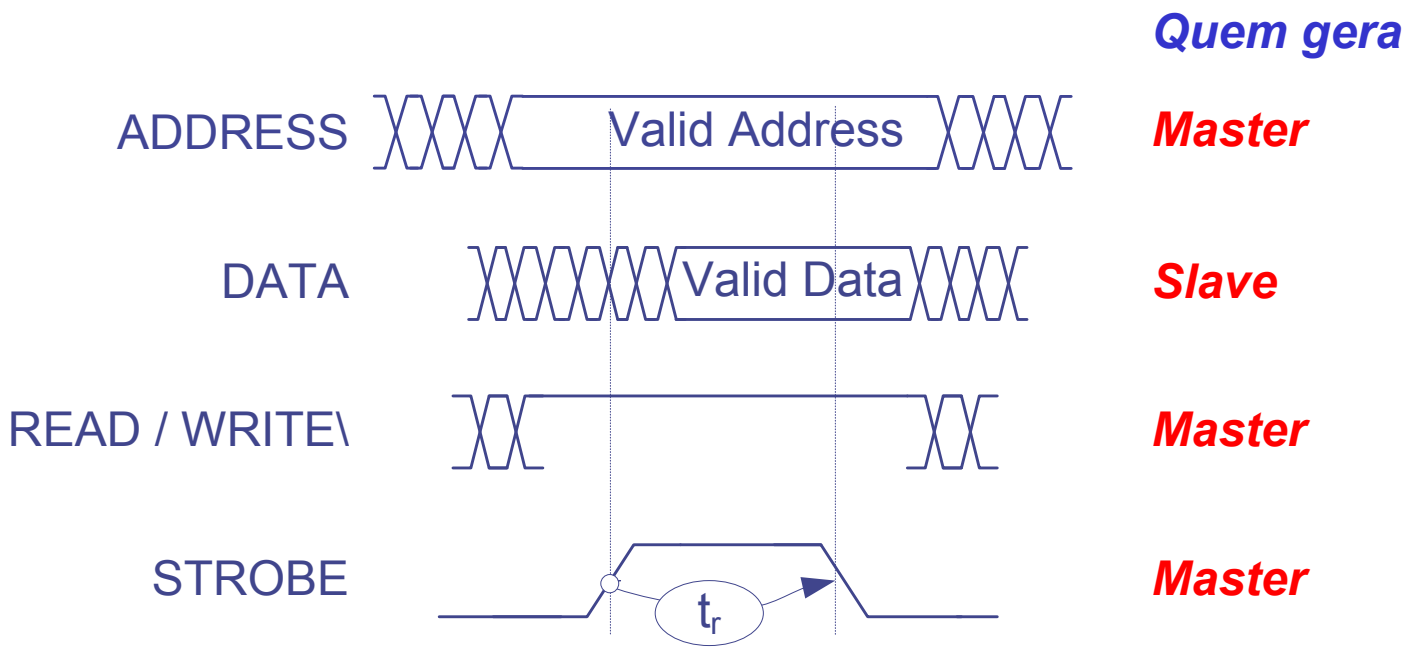
- Protocolo síncrono com sinalização por **sinal único** (escrita):



Operação de ESCRITA

Controlo da transferência de informação

- Protocolo síncrono com sinalização por **sinal único** (leitura):

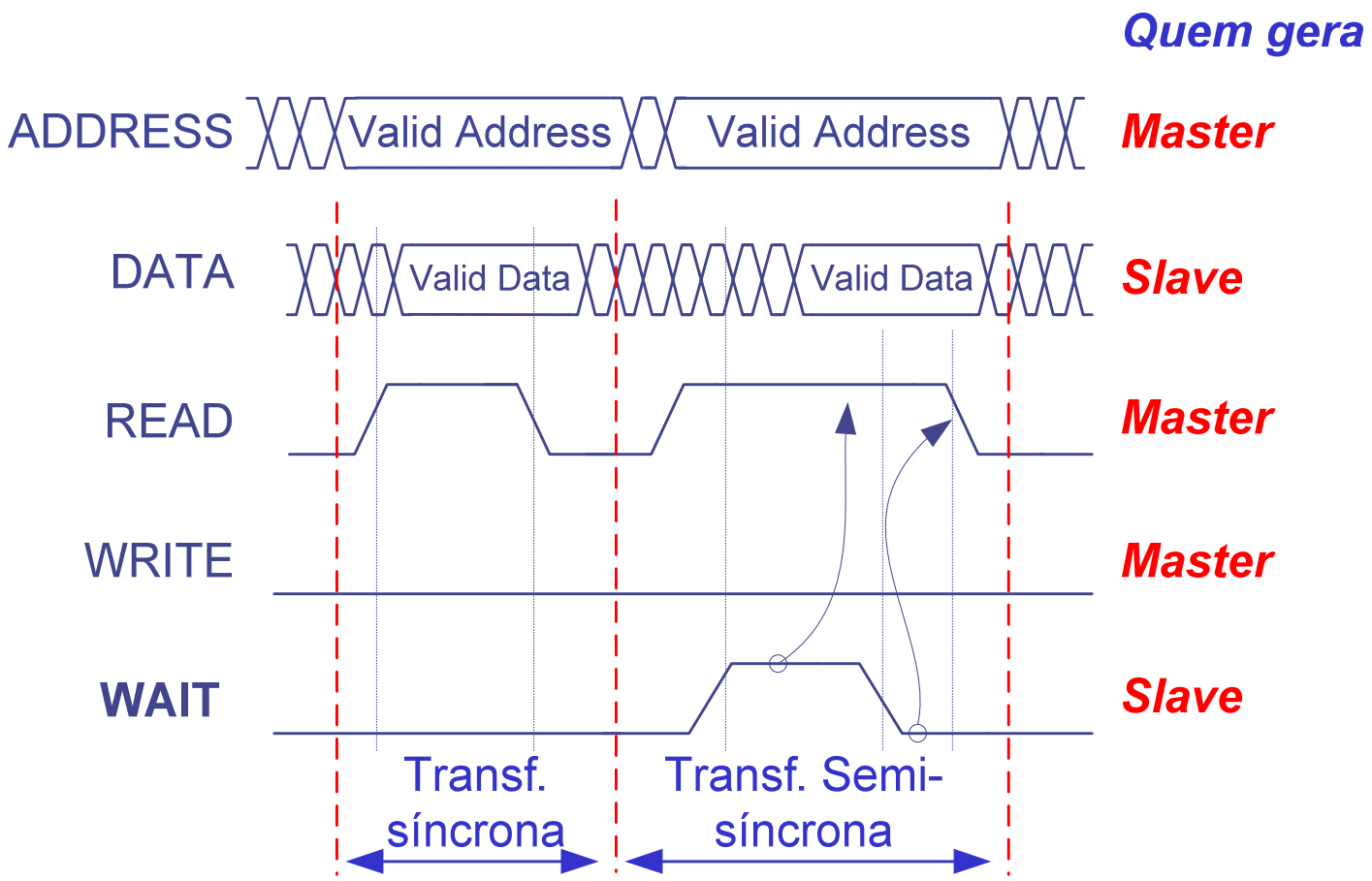


Operação de LEITURA

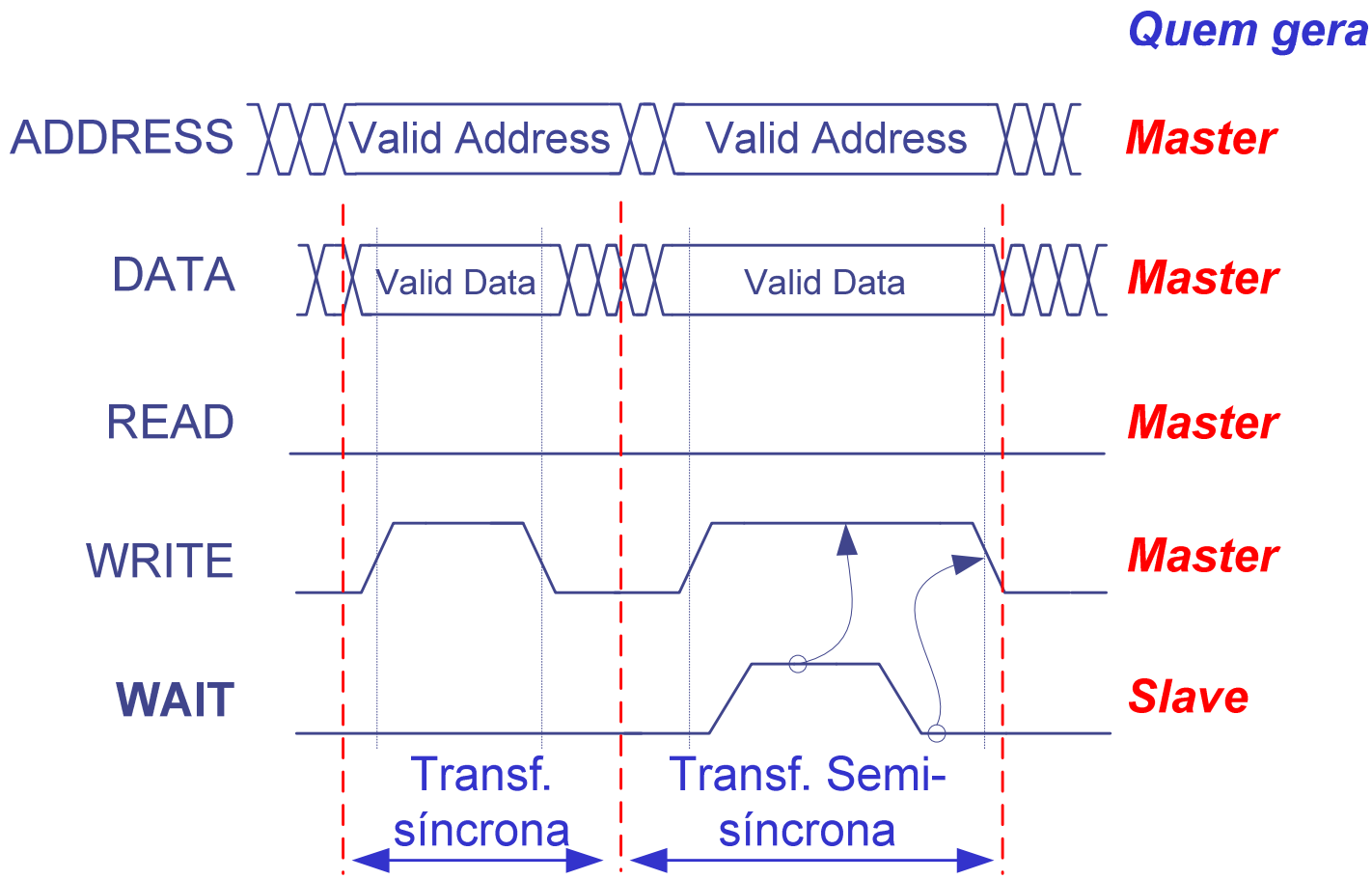
Protocolo semi-síncrono

- Protocolo **síncrono** é simples mas demasiado rígido para certas aplicações (a unidade mais lenta determina a máxima frequência a que o sistema pode funcionar)
- Protocolo "**handshaking**" é versátil em termos de requisitos temporais mas implica:
  - uma interface mais complexa no *slave*
  - um número elevado de linhas de controlo
- Solução intermédia – protocolo **semi-síncrono**:
  - Opera, por defeito, de forma síncrona mas torna-se "handshaken" quando houver um pedido explícito, por parte do *slave*, para alteração da temporização das ações
  - Um dispositivo lento pode, através da ativação de um sinal adequado, atrasar a conclusão da operação; no caso em que o sinal não é ativado a transferência é síncrona (só os dispositivos lentos necessitam de ativar esse sinal)
- O protocolo semi-síncrono permite a coexistência de dispositivos rápidos e de dispositivos lentos no mesmo sistema

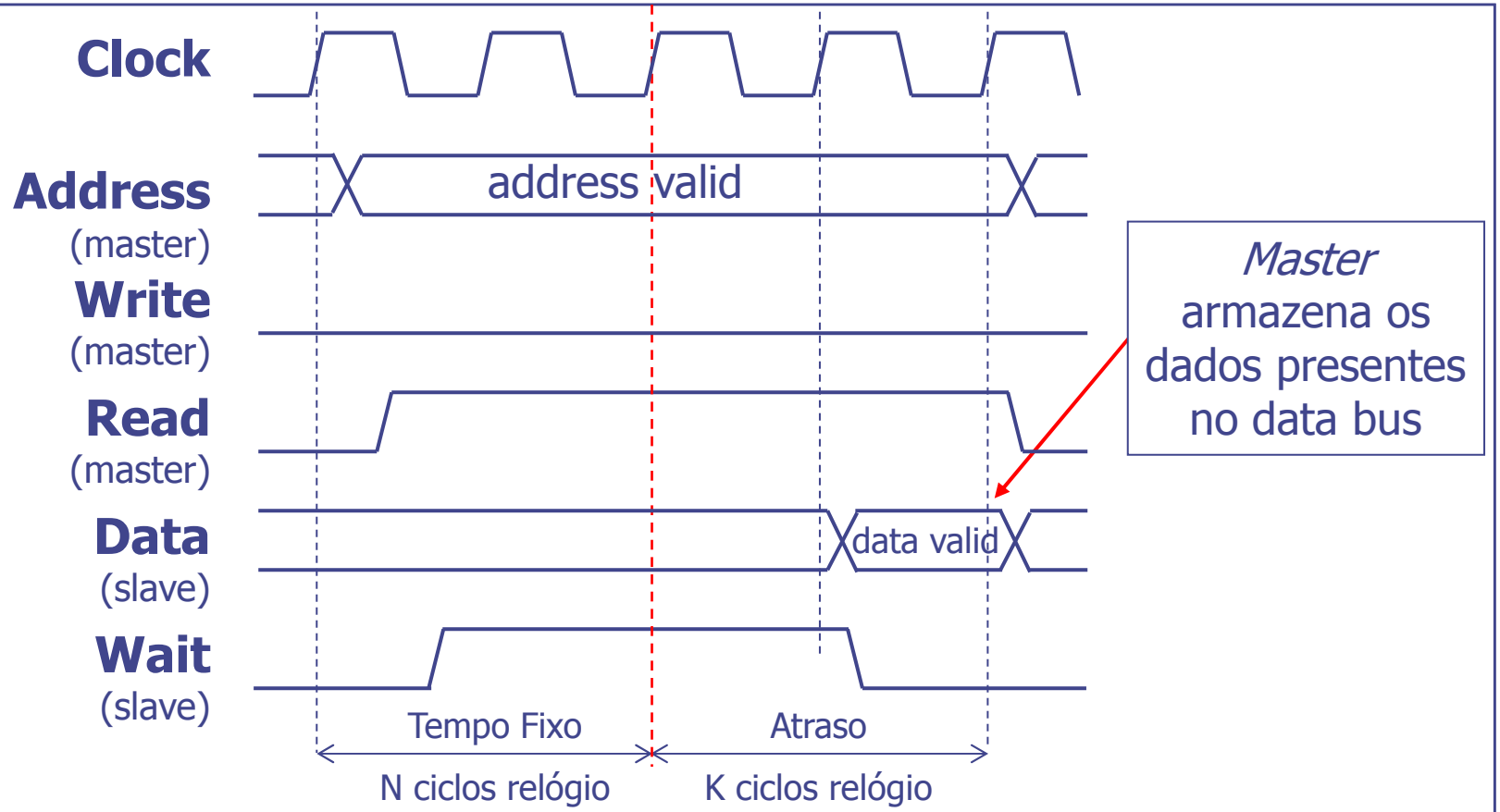
Protocolo semi-síncrono – Leitura



Protocolo semi-síncrono – Escrita



Transferência semi-síncrona (exemplo)



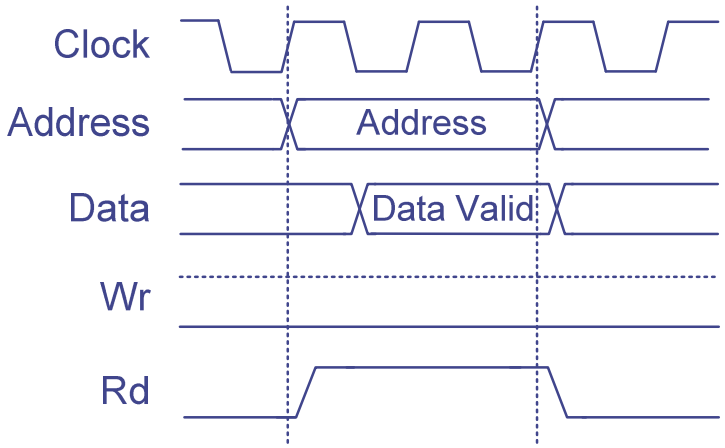
- O *master* atrasa a escrita dos dados; o atraso é um múltiplo do período de relógio: os ciclos de relógio correspondentes ao "atraso" introduzido designam-se por **wait-states** (2 *wait-states*, no exemplo da figura, i.e. K=2)

Comparação sumária

- **Complexidade**:
  - Protocolos síncronos são mais simples que os *handshaken*
- **Velocidade**:
  - Protocolos **síncronos**:
    - Potencialmente os mais rápidos desde que todas as unidades operem à mesma velocidade
    - Sistemas heterogéneos têm que ser desenhados de acordo com a velocidade da unidade mais lenta
  - Protocolos ***handshaken*** permitem que cada unidade seja operada à máxima velocidade (à custa de uma maior complexidade das unidades)
  - Protocolos **semi-síncronos** exibem características semelhantes aos protocolos síncronos

Exercício

- Considere-se um CPU que suporta transferências de tipo síncrono e de tipo semi-síncrono. O CPU funciona a uma frequência de 500 MHz e o ciclo de leitura pode ser descrito no diagrama temporal



- Pretende-se ligar a este CPU uma memória RAM com um tempo de acesso de 7 ns (tempo que decorre desde que a memória é selecionada até que a informação fica disponível no data bus)

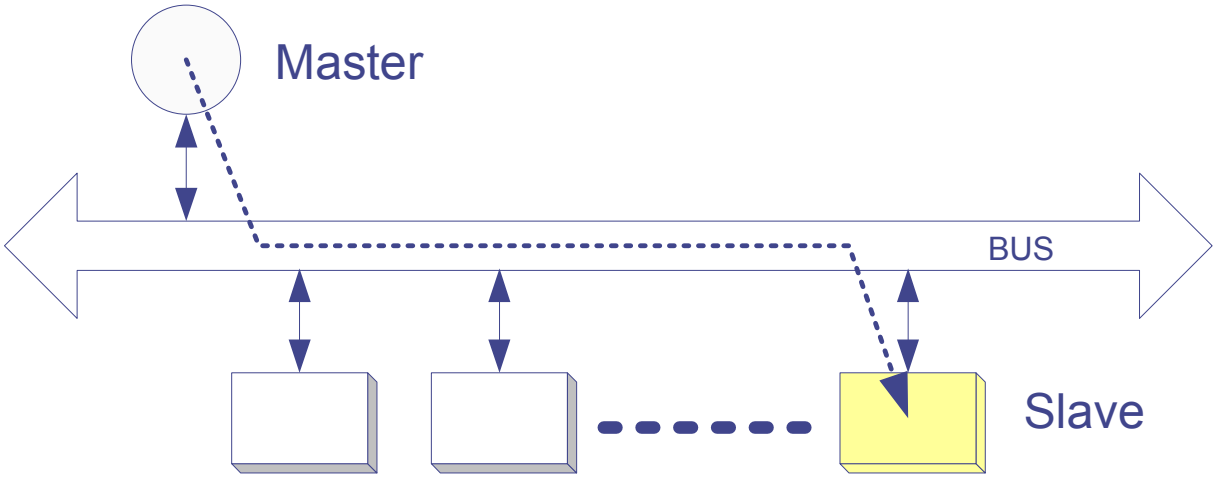
  Q1: É possível a utilização de uma transferência de tipo síncrono no acesso a esta memória?

  Q2: No caso de não ser possível, pode-se recorrer a uma transferência de tipo semi-síncrono. Qual o número de wait-states que é necessário introduzir para que a operação decorra com sucesso, considerando um atraso de propagação no descodificador de endereços de 0 ns

  Q3: A questão anterior, considerando um atraso de propagação no descodificador de endereços de 1.1 ns

Endereçamento

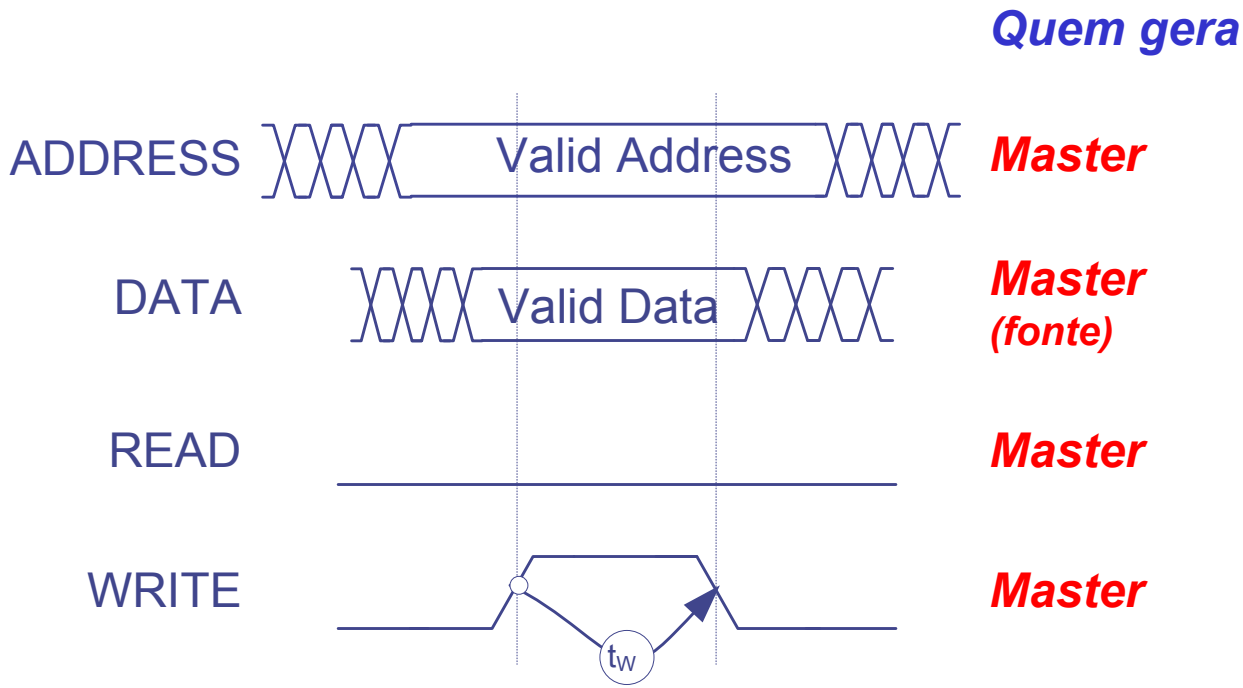
- Durante uma operação de transferência de dados, a seleção do interlocutor é feita através de **endereçamento**



- Ligação entre endereçamento e transferência de dados:
  - O endereçamento tem de ser efetuado sempre antes da transferência de dados (pode ou não ser mantido durante a mesma)
  - Podem existir sinais separados para ações de endereçamento e transferência de dados ou pode haver partilha de sinais

Endereçamento e transferência de dados – "MERGED"

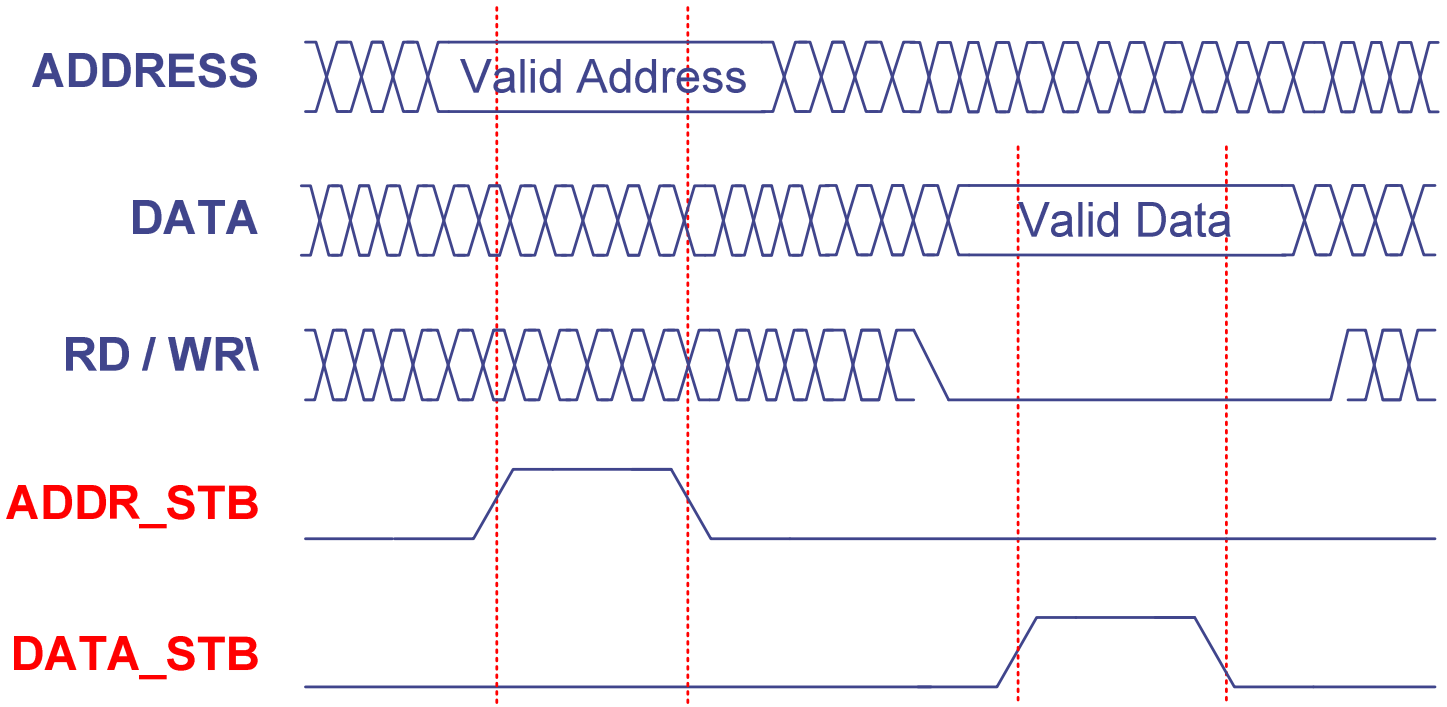
- Fase de endereçamento engloba a transferência de dados



Exemplo de um ciclo de escrita com protocolo síncrono

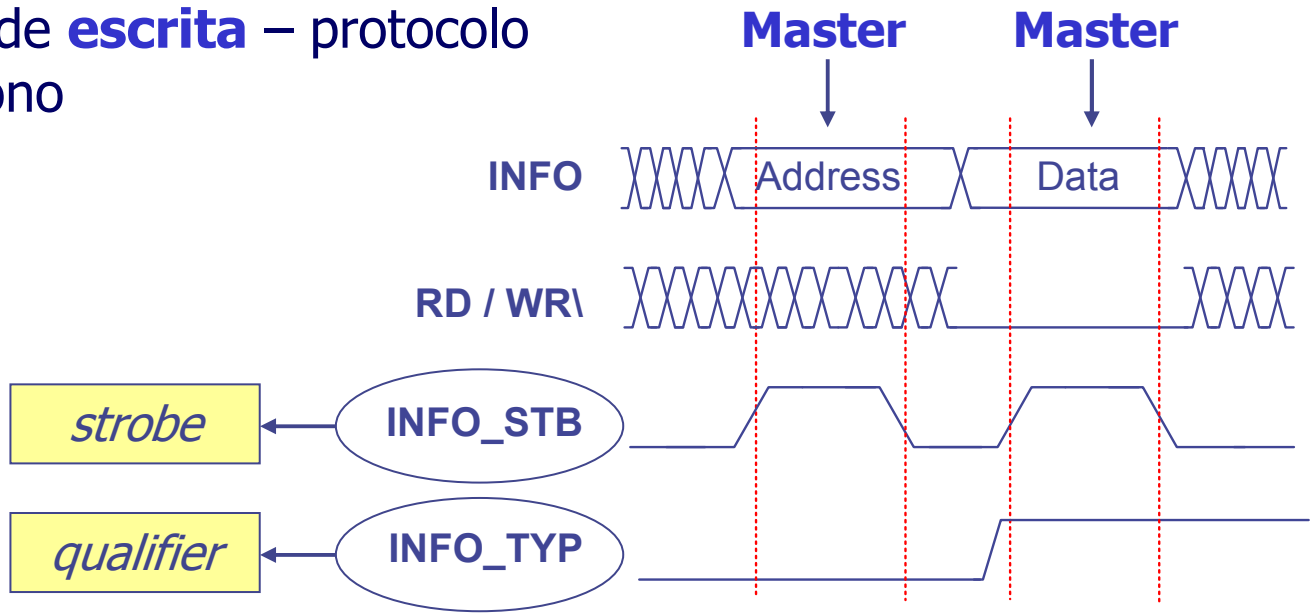
Endereçamento e transferência de dados – "MICROCICLO"

- O endereçamento e a transferência de dados são tratadas como operações autónomas com sinais de controlo e informação separados
- Exemplo de um ciclo de escrita com protocolo síncrono



Multiplexagem de endereços e dados

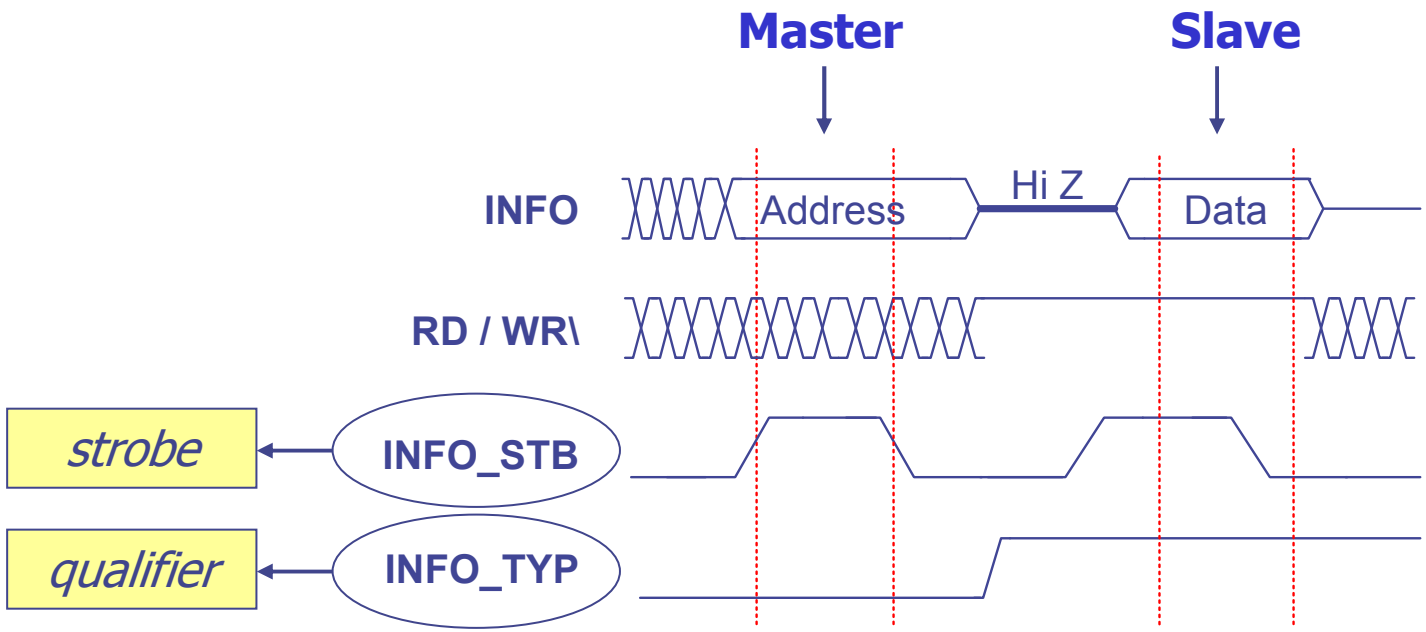
- Utilização das mesmas linhas físicas para envio de endereços e dados, em intervalos de tempo temporalmente disjuntos
- Adicionam-se sinais para indicação do tipo de informação que circula nas linhas (*qualifiers*)
- Ciclo de **escrita** – protocolo síncrono



- **INFO_TYP** é um "**qualifier**": distingue dados de endereços (neste exemplo, INFO_TYP=0 → endereços, INFO_TYP=1 → dados)

Multiplexagem de endereços e dados

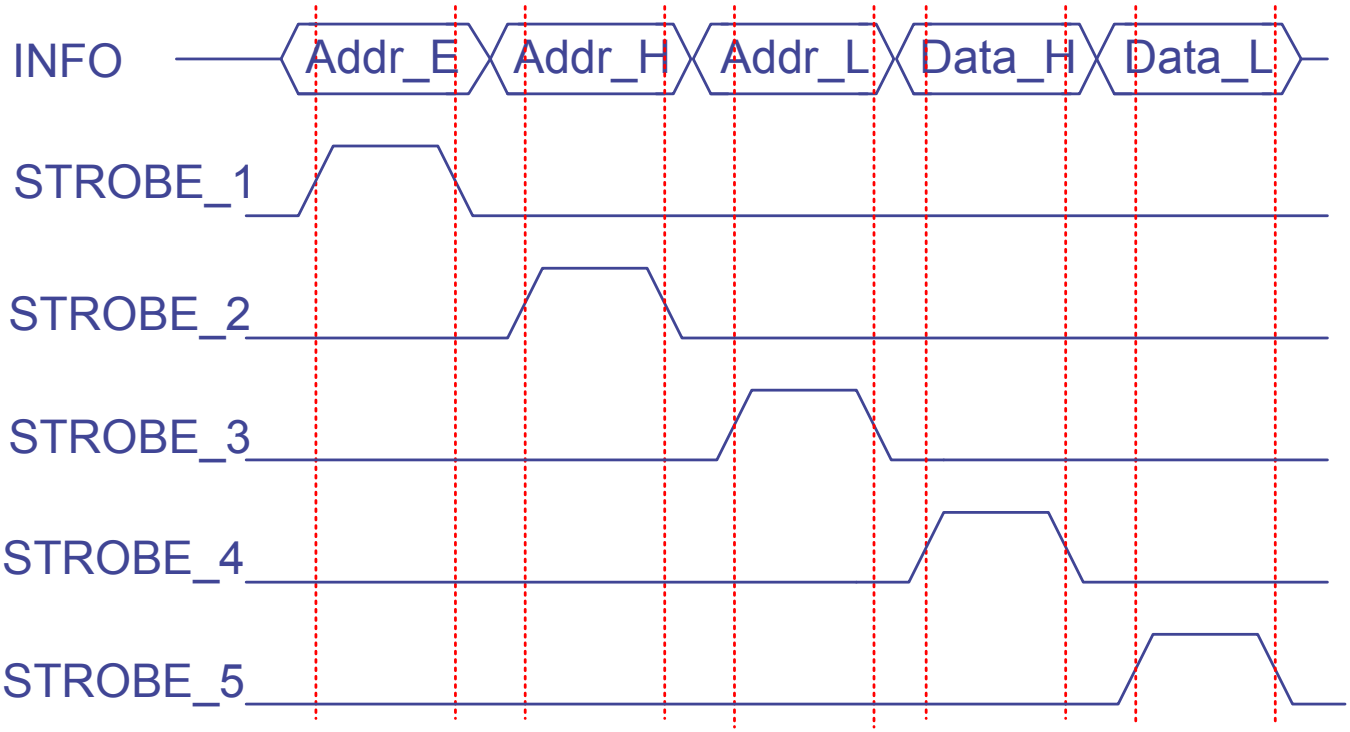
- Ciclo de **leitura** (protocolo síncrono)



- INFO_TYP é um *qualifier*: distingue dados de endereços

Multiplexagem de endereços e dados

- Multiplexagem com *strobes* independentes
- Um *strobe* para cada tipo de informação (validação e identificação simultâneas)

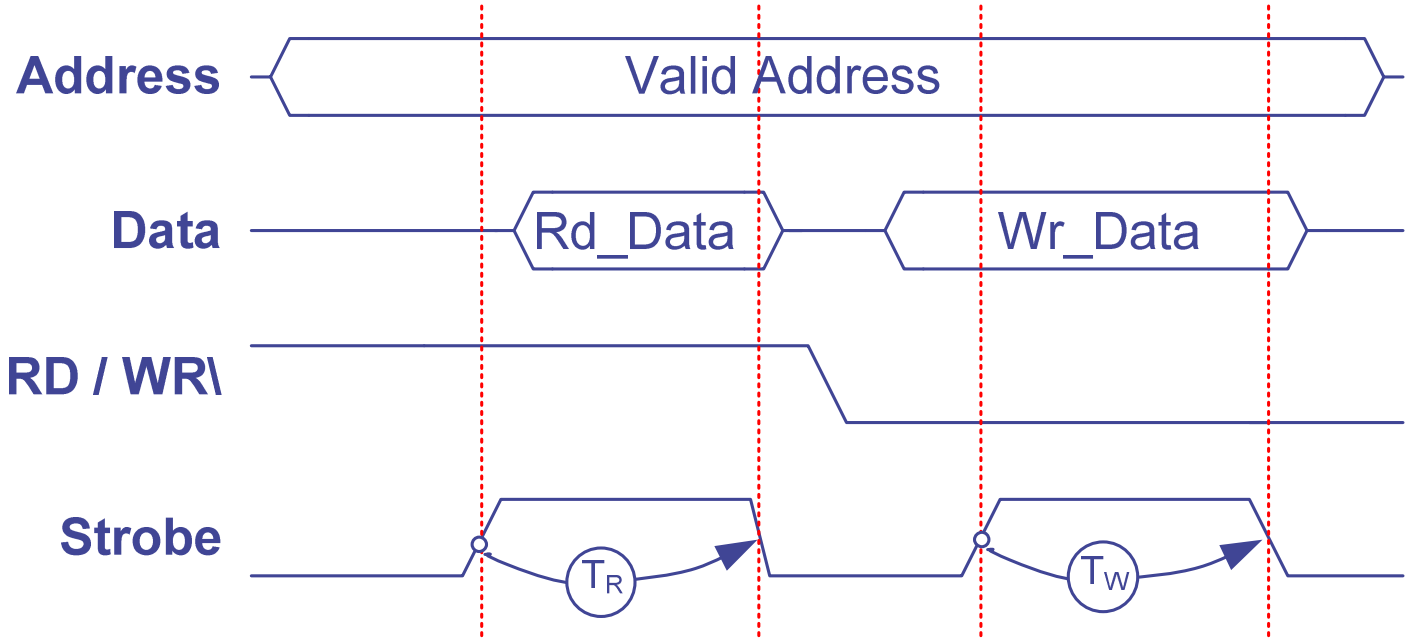


Ciclos com transferências múltiplas

- **Read-Modify-Write** (operação atómica, indivisível)
  - Permite, por exemplo, modificação parcial de bits
  - Acesso duplo com um só endereçamento
  - Primeiro leitura, segundo escrita
  - Os processadores modernos têm instruções especiais que usam este modo de transferência atómica de informação (por exemplo, para trocar o conteúdo de um registo com uma posição de memória)
  - Em sistemas com mais de um processador/core, este tipo de transferência é usado como primitiva de sincronização (por exemplo, semáforos).
- **Read-After-Write**
  - Acesso duplo com um só endereçamento
  - Primeiro escrita, segundo leitura (para verificação)
- **Block**
  - Acesso múltiplo incremental, iniciado por um endereçamento

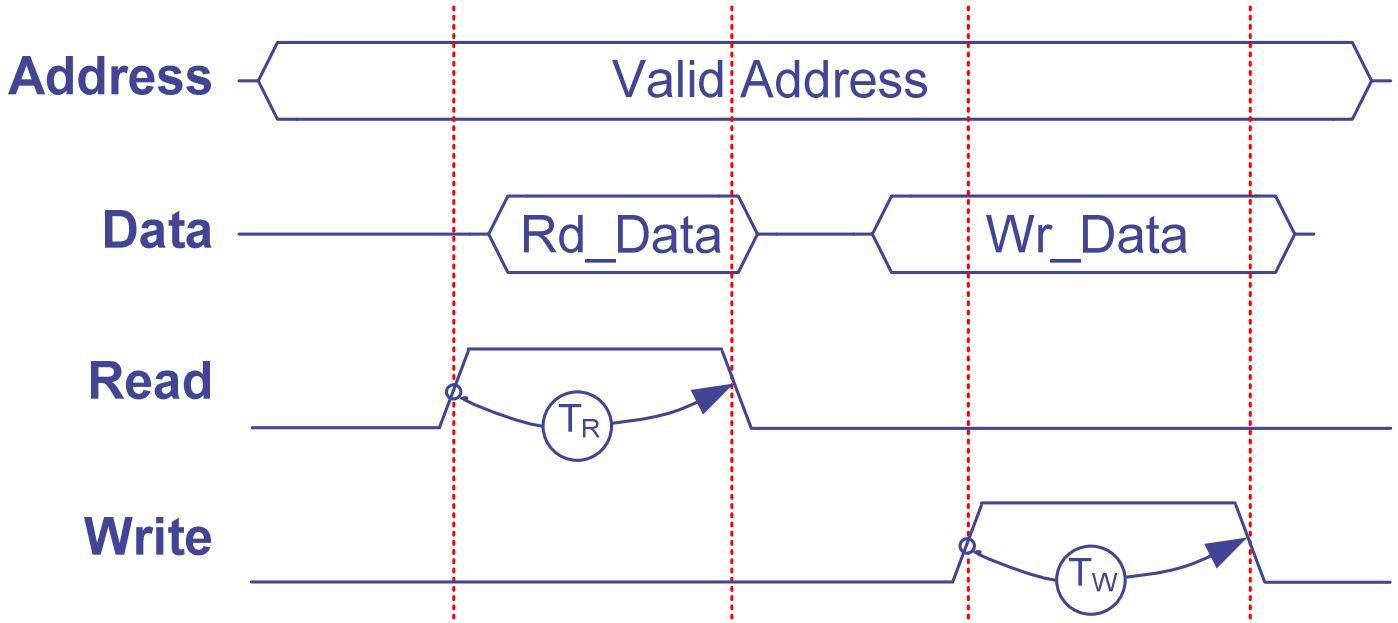
Ciclos com transferências múltiplas – "Read-Modify-Write"

- **Exemplo**: protocolo síncrono, sinal único para sinalização Read / Write



Ciclos com transferências múltiplas – "Read-Modify-Write"

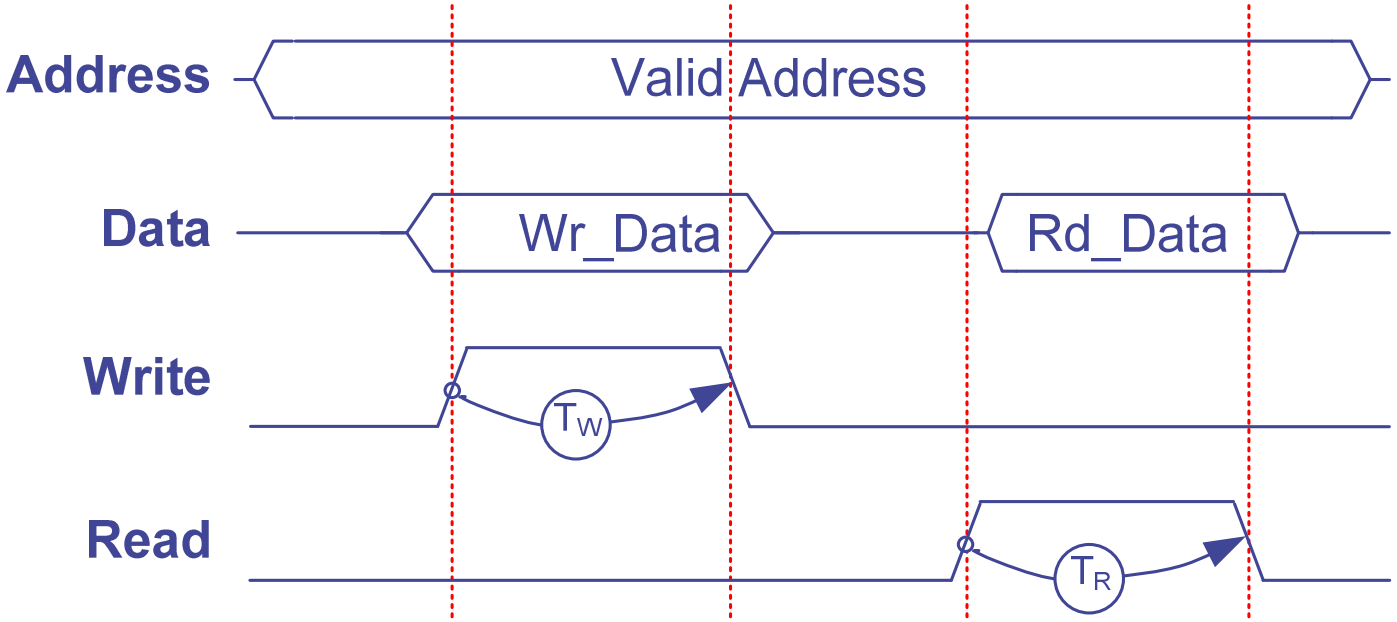
- **Exemplo**: protocolo síncrono, sinais de Read e Write independentes



Ciclos com transferências múltiplas – "Read-After-Write"

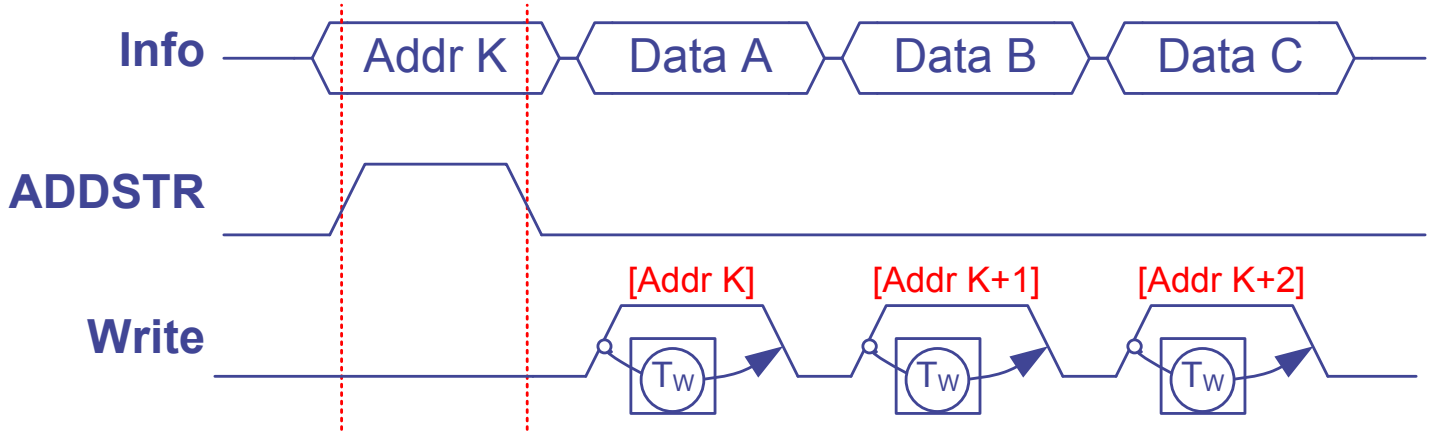
- Acesso duplo com um só endereçamento
- Primeiro escrita, depois leitura (para verificação)

- **Exemplo**: protocolo síncrono, sinais de Read e Write independentes



Ciclos com transferências múltiplas – "Block"

- Um ciclo com protocolo do tipo "block transfer" permite a transferência de um bloco de dados (n bytes/words) com apenas 1 ciclo de endereçamento
- Neste modo de transferência, a primeira transferência inclui um ciclo de endereçamento e as transferências subsequentes apenas incluem ciclos de dados.
- O *slave* é responsável pelo incremento do endereço após cada ciclo de dados (de modo a que a transferência se processe em endereços consecutivos)
- Exemplo: escrita com protocolo síncrono e **endereçamento microciclo** em **barramento multiplexado**

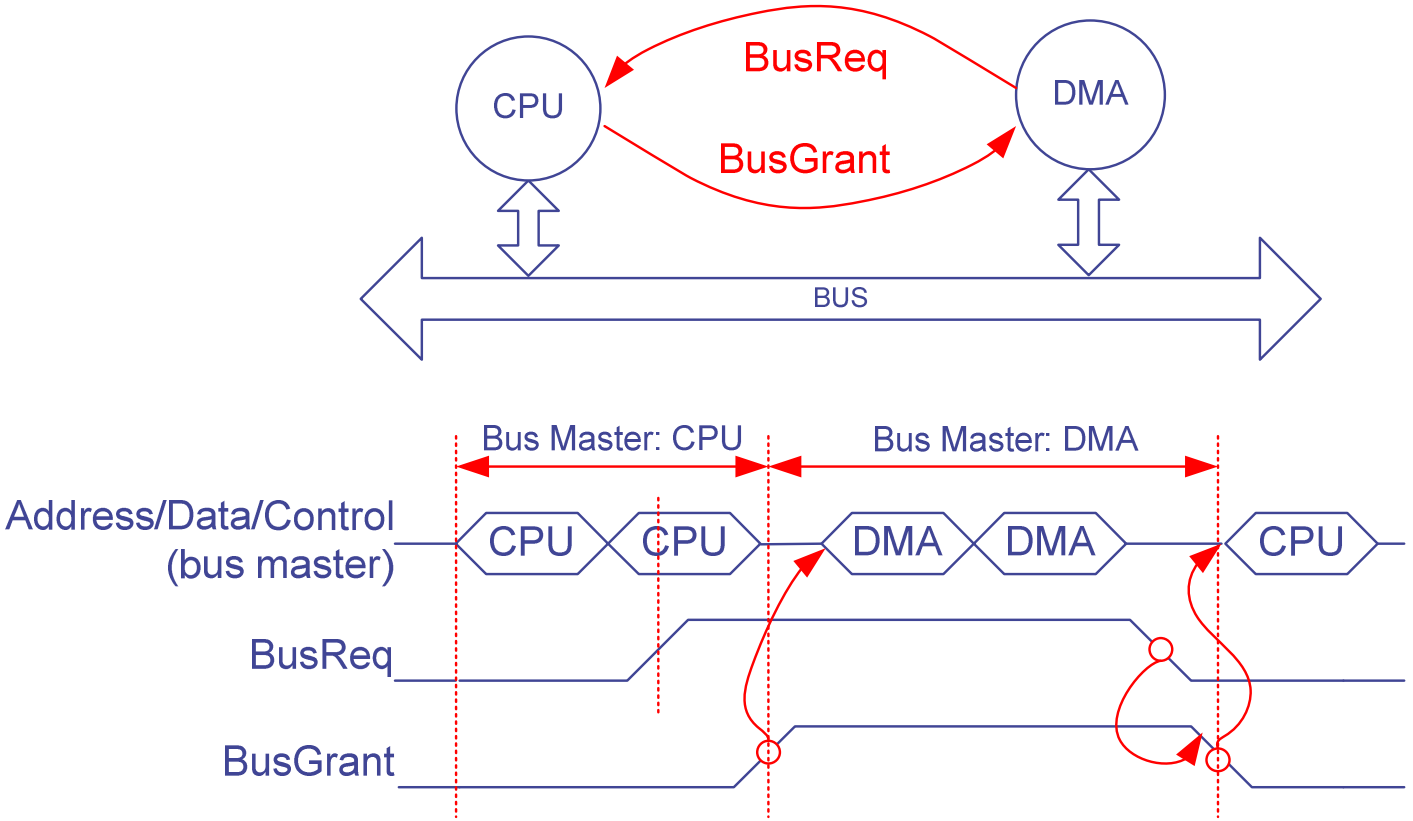


Tipos de Barramentos

- **Barramentos síncronos**
  - O barramento inclui um sinal de relógio
  - Protocolo de comunicação síncrono ou semi-síncrono
  - Todos os dispositivos têm de ser capazes de processar os sinais à frequência do relógio do barramento
  - Dispositivos mais lentos limitam a taxa de transferência no barramento
- **Barramentos assíncronos**
  - O barramento não inclui um sinal de relógio
  - Protocolo de comunicação "handshaken"
  - Permite a ligação de uma grande variedade de dispositivos operando com relógios de diferentes frequências
  - Módulos de interface podem responder ao seu próprio ritmo – mais fácil acomodar periféricos com diferentes tempos de resposta

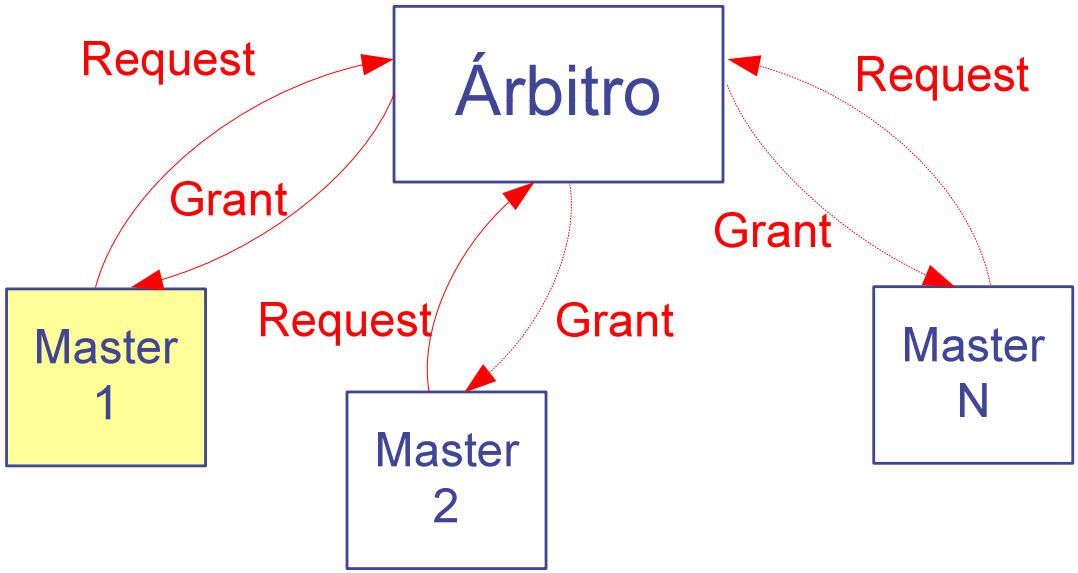
Barramentos "Multi-Master"

- Mais do que uma unidade capaz de iniciar e controlar transferências de dados (master)
- E.g.: múltiplos DMA, múltiplos processadores



Barramentos "Multi-Master" – Arbitragem

- Apenas um *master* pode estar ativo em cada instante - o barramento é um recurso partilhado
- É necessário um **mecanismo de arbitragem** para decidir qual o *master* que assume o controle do barramento



- **Árbitro**: entidade que faz a gestão do acesso ao barramento, garantindo a **exclusão mútua**

Tipos de arbitragem

- **Centralizada:**
  - A arbitragem é feita por um único dispositivo no barramento (pode ser o microprocessador ou outro dispositivo que apenas realiza essa tarefa)
  - Exemplo: *masters* organizados numa estrutura "daisy-chain": entrada do árbitro é uma linha de "Bus Request" única para todos os *masters*; árbitro responde com um sinal de "Bus Grant" que é propagado pelos *masters* até ao que fez o pedido

- **Distribuída:**
  - Os mecanismos de atribuição do barramento estão integrados nos potenciais utilizadores. Ou seja, todos os *masters* ligados ao barramento que pretendam ter acesso participam na seleção do próximo "bus master"
  - Exemplo: barramento CAN

Barramentos "Multi-Master" – Conceitos

- "**Starvation**":
  um elemento não tem acesso ao barramento devido a constantes pedidos provenientes de elementos de prioridade superior

- "**Fair**":
  um sistema de arbitragem que garante o acesso ao barramento por todos os elementos, evitando "starvation". Pode basear-se numa das seguintes técnicas:
  - Aumento da prioridade dos elementos que esperaram para além de um dado intervalo
  - Diminuição da prioridade dos elementos já servidos
  - Redistribuição das prioridades de forma regular ou aleatória
  - Não aceitação de novos pedidos até os pendentes serem satisfeitos

  ("**Fairness**" - até um dispositivo com a mais baixa prioridade não deverá ficar impedido de aceder ao barramento)

Políticas de arbitragem

- **Prioridades Fixas**: Quando existem pedidos "simultâneos" o barramento é atribuído ao elemento de maior prioridade. A prioridade (estática) é um parâmetro de configuração do sistema

- **Round-Robin**: acesso ao barramento é atribuído rotativamente – um *master* quando termina a transferência passa o controlo ao *master* seguinte

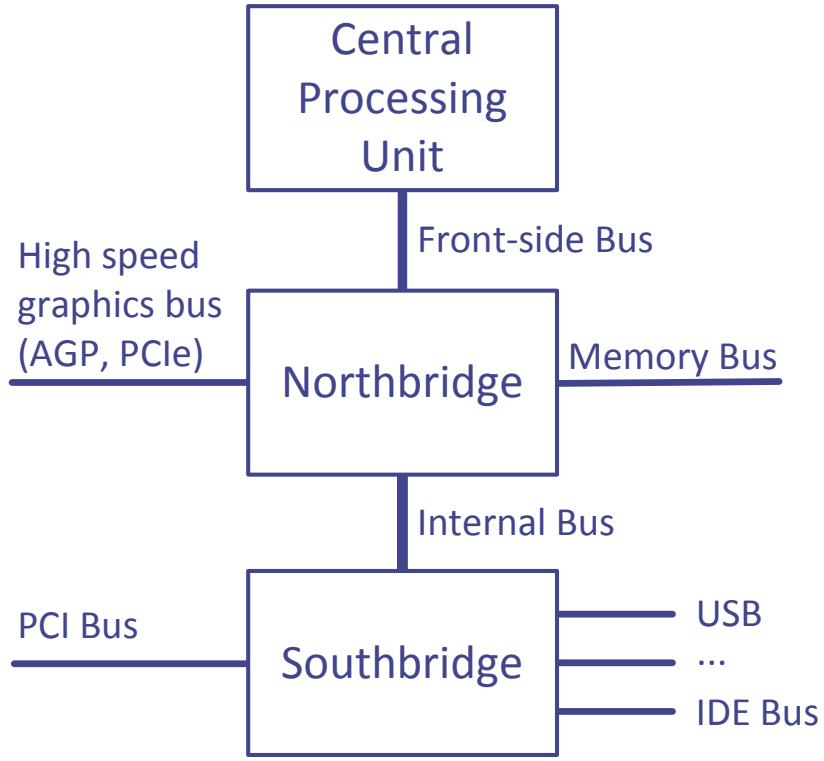
- **Prioridades dinâmicas**: As prioridades mudam ao longo do tempo, para evitar problemas tais como a impossibilidade de acesso por parte dos elementos menos prioritários (*starvation*).
  - **FIFO ou FCFS**: (***First-In-First-Out*** ou ***First-Come-First-Served***) os elementos são ordenados por ordem do pedido e o barramento é atribuído por ordem de chegada

Hierarquia de barramentos

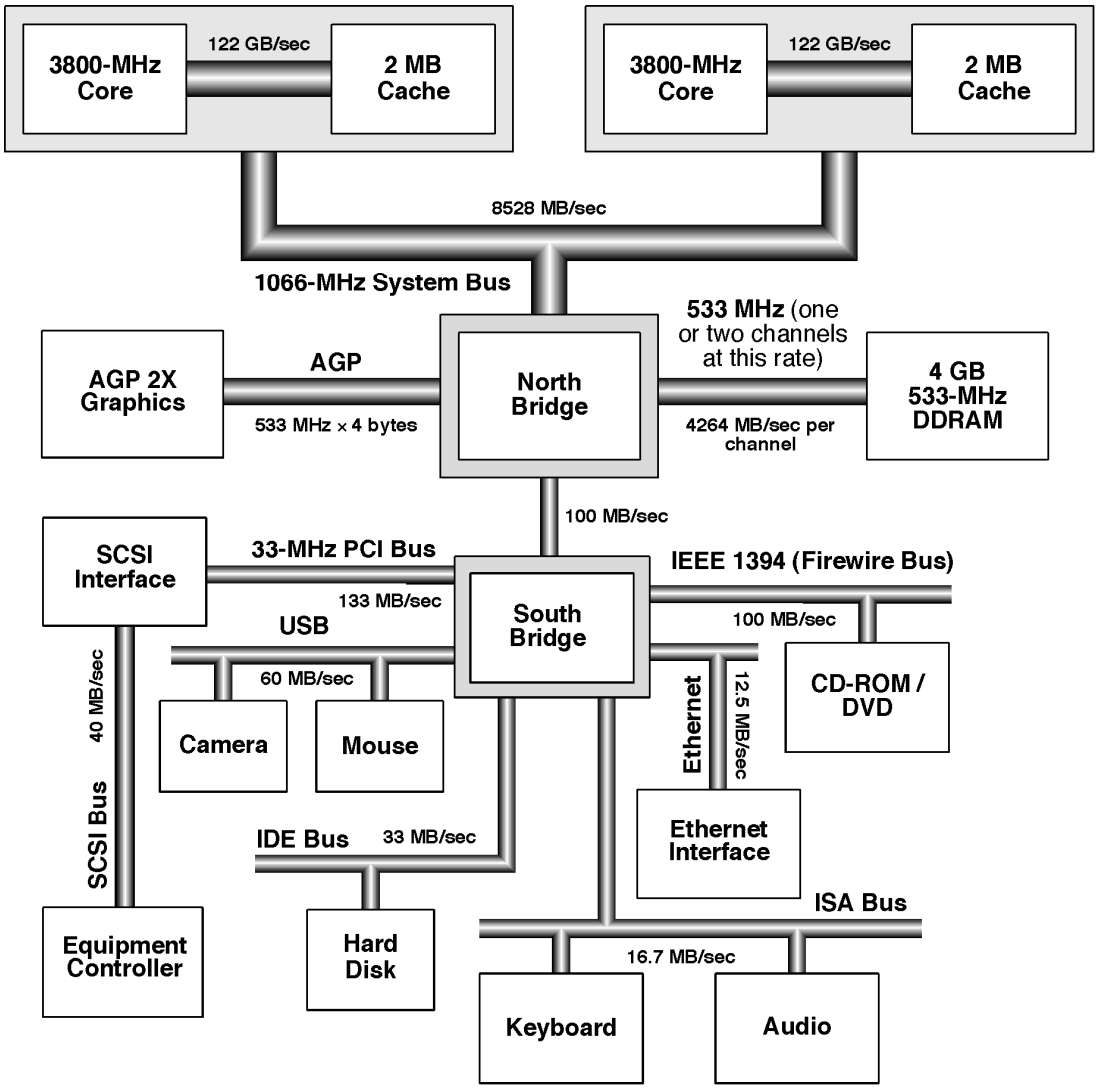
- Se um grande número de dispositivos estiver ligado ao mesmo barramento o desempenho global do sistema degrada-se:
  - Mais dispositivos implicam maior comprimento do barramento, ou seja, o tempo de propagação dos sinais aumenta
  - Num barramento síncrono, os dispositivos mais lentos limitam o desempenho dos mais rápidos (tipicamente CPU e Memória)
  - O barramento pode tornar-se um ponto de estrangulamento do sistema quando as taxas de transferência de informação entre os dispositivos do sistema se aproximam do limite do barramento
- Solução:
  - Usar diversos barramentos com diferentes níveis de desempenho, a funcionar de forma independente
  - Obriga à utilização de dispositivos de interligação dos barramentos, designados por *bridges*

Hierarquia de barramentos (arquitetura Intel até 2011)

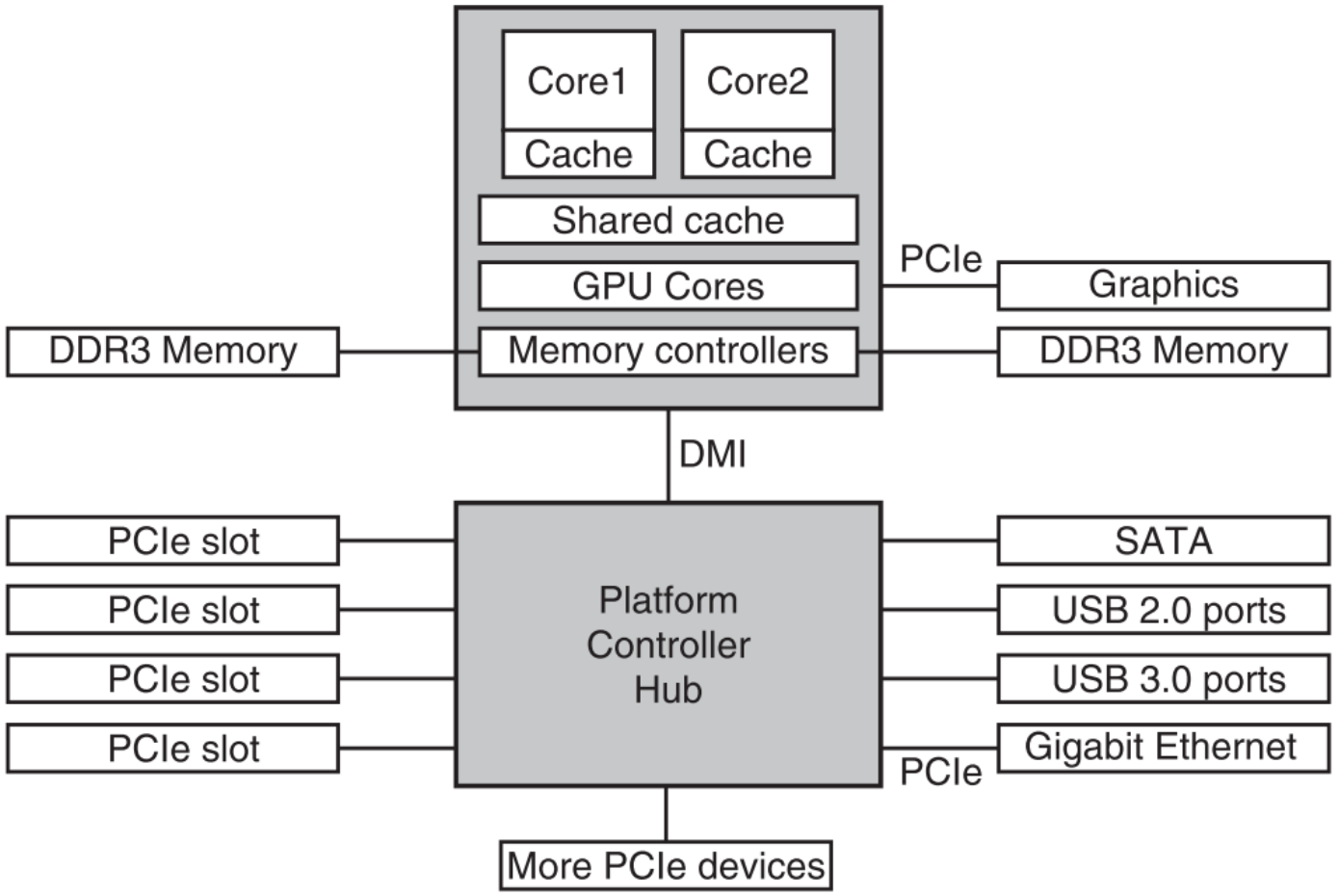
- Múltiplos barramentos interligados por *bridges*
- Várias transações podem estar simultaneamente em curso nos diferentes barramentos
- Os dispositivos mais lentos não condicionam a operação dos mais rápidos
- AGP – Accelerated Graphics Port
- PCI – Peripheral Component Interconnect
- IDE – Integrated Drive Electronics
- PCIe – PCI express



Hierarquia de barramentos



Hierarquia de barramentos – arquitetura Intel



Barramento PCI

- Barramento paralelo
- Protocolo síncrono
  - Versão de 32 bits: relógio de 33 MHz (132 MB/s)
  - Versão de 64 bits: relógio de 66 MHz (528 MB/s)
- Multiplexado, i.e. linhas comuns para endereços e dados
- Barramento "multi-master" com arbitragem centralizada
- "Plug and Play": permite que dispositivos ligados ao barramento sejam automaticamente detetados e configurados (por exemplo, vetor de interrupção atribuído, gama de endereços atribuída)
- Limitações:
  - Largura de banda
  - O aumento da frequência de relógio não é solução (*skew*, *crosstalk* entre linhas, efeito capacitivo)
  - O barramento tem muitos sinais (conetores de 124 / 188 pinos) o que se traduz em placas de dimensão significativa
  - Incapacidade de suportar transferências com largura de banda e latência garantidos (exigência de, por exemplo, *streaming* de vídeo)

Barramento PCI - exemplo

- Reprodução de um vídeo com resolução (720p) de 1280x720 pixels
  - Cada pixel é codificado com 3 bytes (R, G e B)
  - Cada imagem tem assim 1280x720x3 = 2.7 Mbytes (2.764.800 bytes)
  - Uma reprodução suave do vídeo obriga a um mínimo de 30 imagens/segundo o que significa um débito de 81.0 MB/s
- Se o ficheiro com o vídeo residir no disco externo, os dados têm que passar, através do barramento, para a memória do sistema e daí novamente através do barramento para a memória do controlador gráfico
- Isto significa que o barramento terá uma ocupação (só para a reprodução do vídeo) de aprox. 162 MB/s. A somar a esta largura de banda há ainda as necessidades do CPU e de outros dispositivos
- Necessidade de um novo barramento
- Questão: qual seria a taxa de transferência para FullHD (1080p)

Barramento PCI Express (PCIe)

- Arquitetura baseada num barramento série dedicado, em vez de um barramento paralelo partilhado
- Ligações ponto-a-ponto: ligações diretas entre dispositivos eliminam a necessidade de arbitragem
- A informação é enviada numa mensagem através de uma ligação dedicada, designada por "lane" (elimina o problema do "skew time" dos barramentos paralelo)
- O paralelismo também é assegurado porque a arquitetura do barramento permite a existência de múltiplas "lanes" o que possibilita o envio em paralelo de múltiplas mensagens
- Taxas de transferência nas diferentes configurações (e gerações):

		Link width				
	Ano	x1	x2	x4	x8	x16
Geração 1	2003	250 MB/s	500 MB/s	1 GB/s	2 GB/s	4 GB/s
Geração 2	2007	500 MB/s	1 GB/s	2 GB/s	4 GB/s	8 GB/s
Geração 3	2011	.985 GB/s	1.97 GB/s	3.94 GB/s	7.88 GB/s	15.8 GB/s
Geração 4	2017	1.97 GB/s	3.94 GB/s	7.88 GB/s	15.75 GB/s	31.51 GB/s
Geração 5	2019	3.94 GB/s	7.88 GB/s	15.75 GB/s	31.51 GB/s	63.0 GB/s

Alguns Acrónimos

- ATA – AT Attachment
- AGP – Accelerated Graphics Port
- ISA – Industry Standard Architecture
- USB – Universal Serial Bus
- IDE – Integrated Drive Electronics (variante do ATA)
- SATA – Serial ATA
- DMI – Direct Media Interface
- PCI – Peripheral Component Interconnect
- SCSI – Small Computer System Interface
- SPI – Serial Peripheral Interface bus
- CAN – Controller Area Network bus
- I2C – Inter-Integrated Circuit bus