Lecture 10 Introduction to AMBA AHB

Multimedia Architecture and Processing Laboratory
多媒體架構與處理實驗室
Prof. Wen-Hsiao Peng (彭文孝)
pawn@mail.si2lab.org
2007 Spring Term

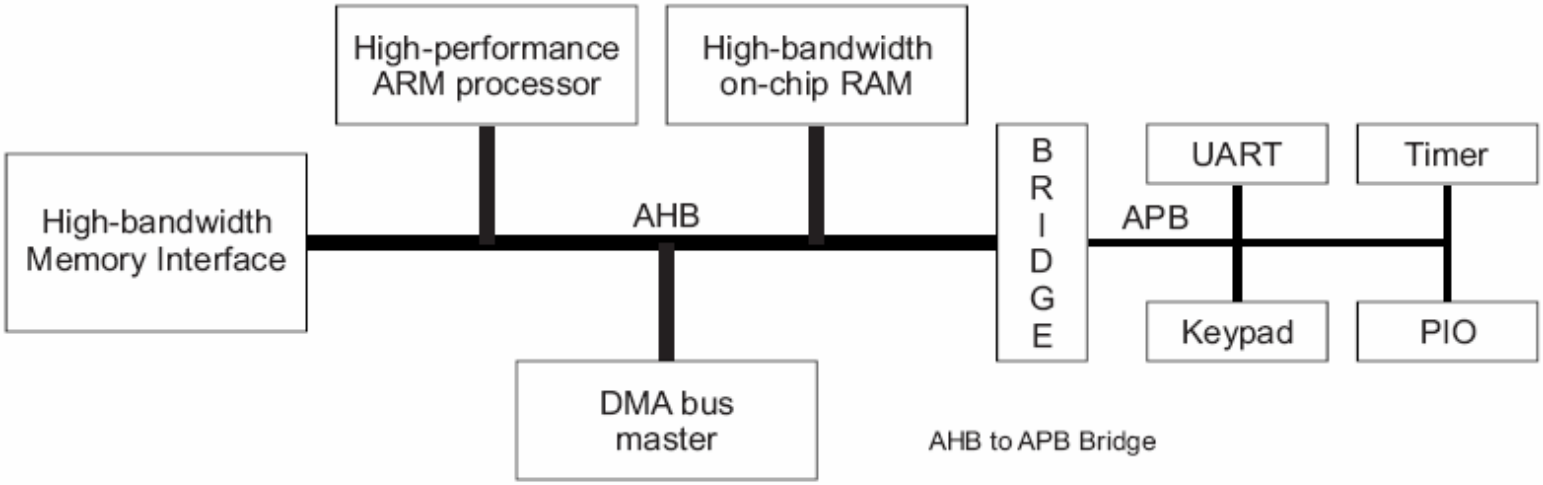
Reference

- AMBA Specification 2.0

AMBA AHB Bus 2.0

- Intended to address the requirement of high performance
- Sit about APB and implement high-performance features
  - Byte-addressable
  - Burst transfers
  - Split transactions
  - Single cycle bus master handover
  - Single clock edge operation
  - Non-tristate implementation
  - Wider bus configurations (64/128 bits)

Typical AMBA AHB-based System



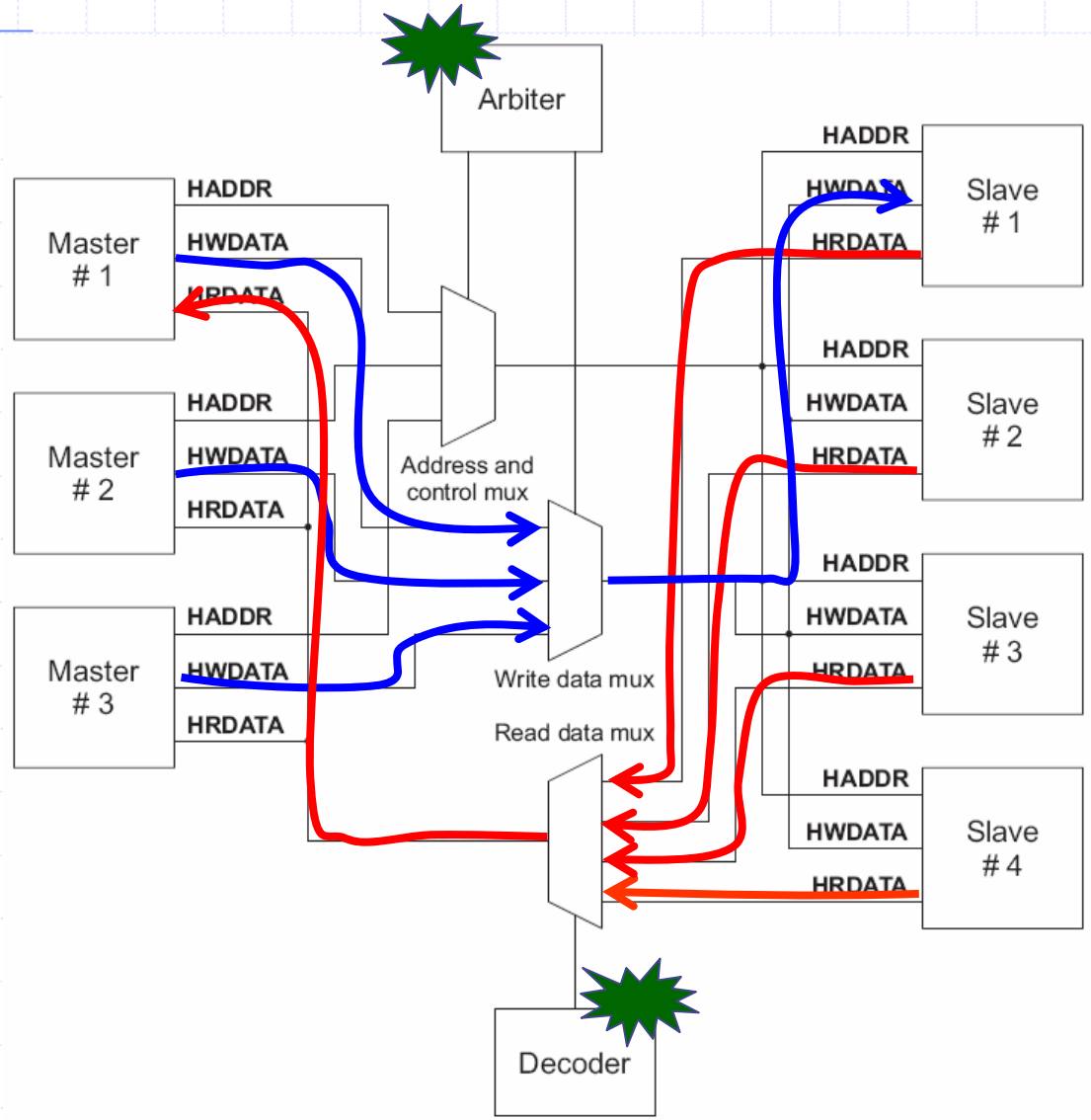
AMBA Advanced High-performance Bus (AHB)
* High performance
* Pipelined operation
* Burst transfers
* Multiple bus masters
* Split transactions

AMBA Advanced Peripheral Bus (APB)
* Low power
* Latched address and control
* Simple interface
* Suitable for many peripherals

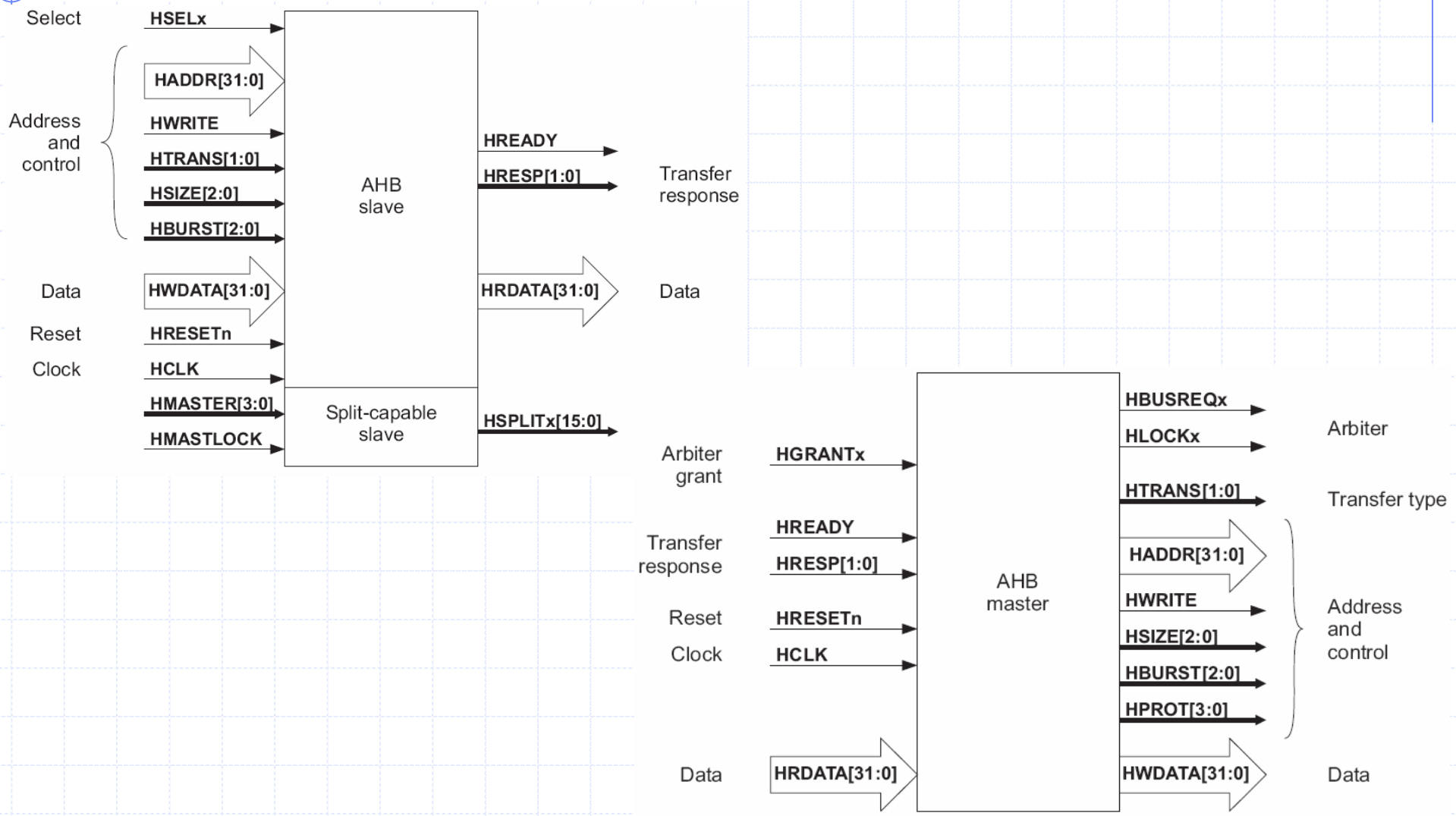
Bus Interconnection

- Two MUXs controlled by arbiter for Write Data and Address
  - All bus masters drive out the address, write data, and control signals
  - Arbiter determines which master has its signals roll to slaves
- One MUX controlled by central decoder for Read Data
  - All bus slaves drive out the read data and responses signals
  - Central decoder determines which slave has its signal roll to masters

Bus Interconnection (c. 1)



Master/Slave Interface

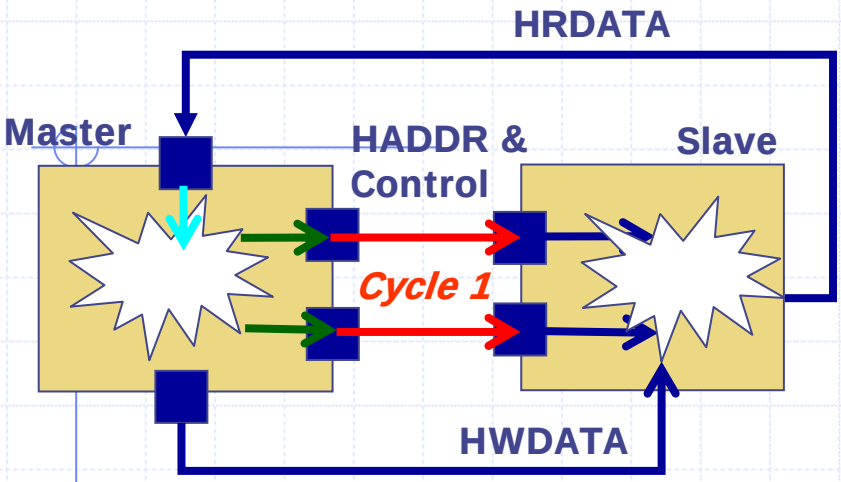


AHB Operations

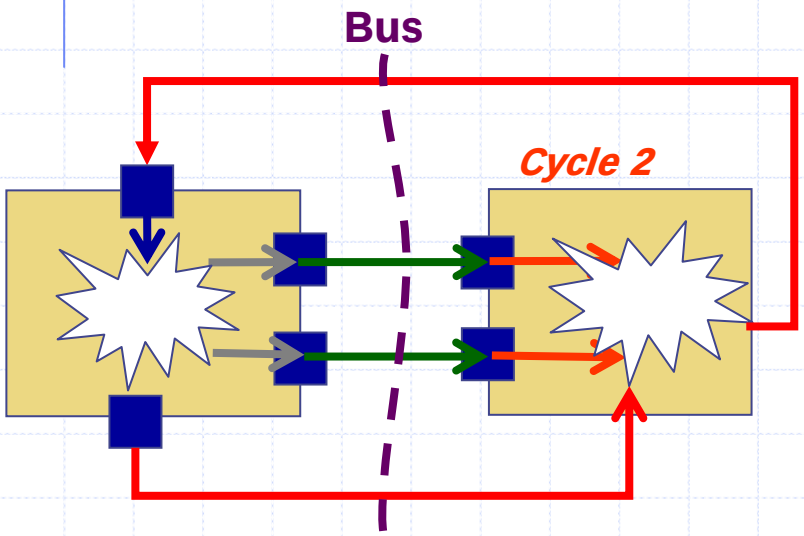
Overview of AHB Operations

- Masters must be granted bus for access before transfer
  - Masters assert request signals
  - Arbiter indicates when the master will be granted use of the bus
- An AHB-bus transfer
  - Address phase
    - A single cycle in which the master drives the address and control signals
    - Indicate direction, width of transfer, and if the transfer forms part of a burst
    - Each address correspond to 1-byte data (byte-addressable)
  - Data phase
    - One or more cycles controlled by HREADY from slave
    - The slaves sample and process the data
  - Address phase and data phase of different transactions are overlapped
- Response HRESP [1:0]
  - OKAY
  - ERROR
  - RETRY or SPLIT

Simple Transfer

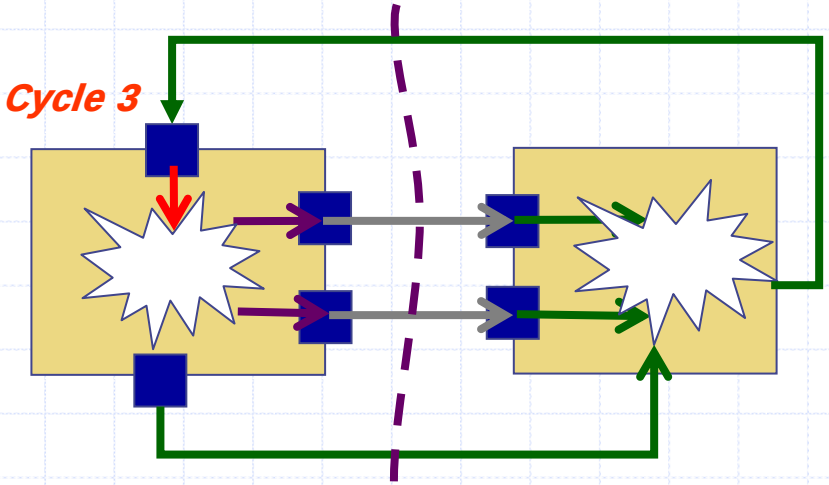


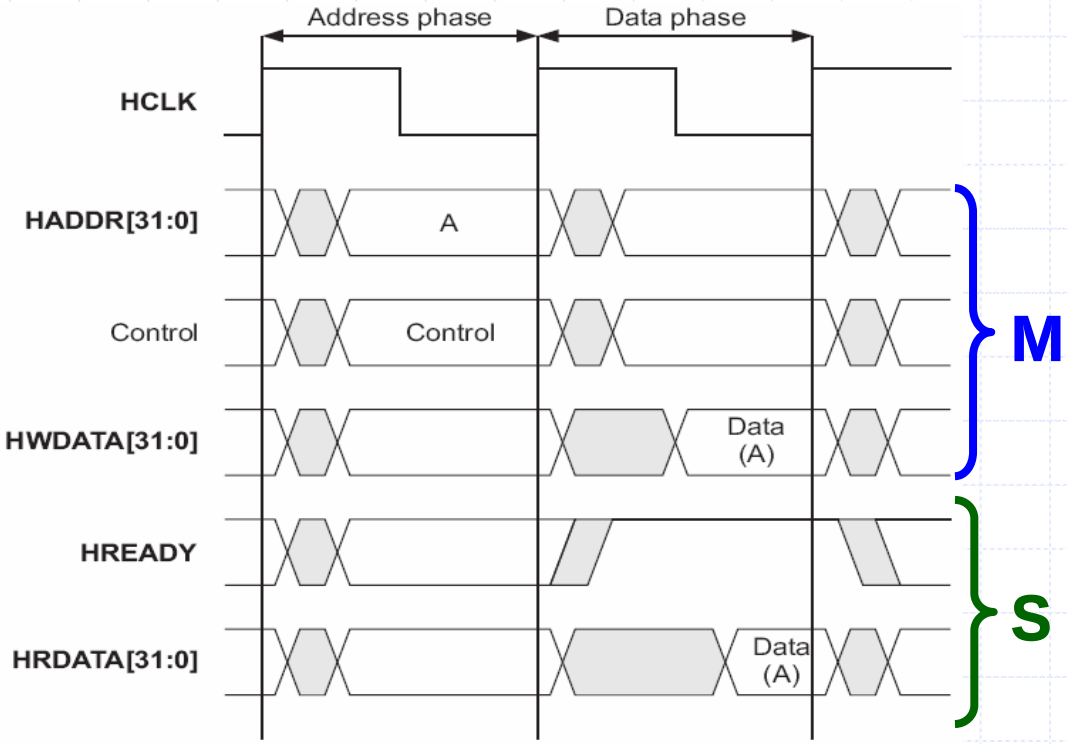
1. Master drives the address and control signals



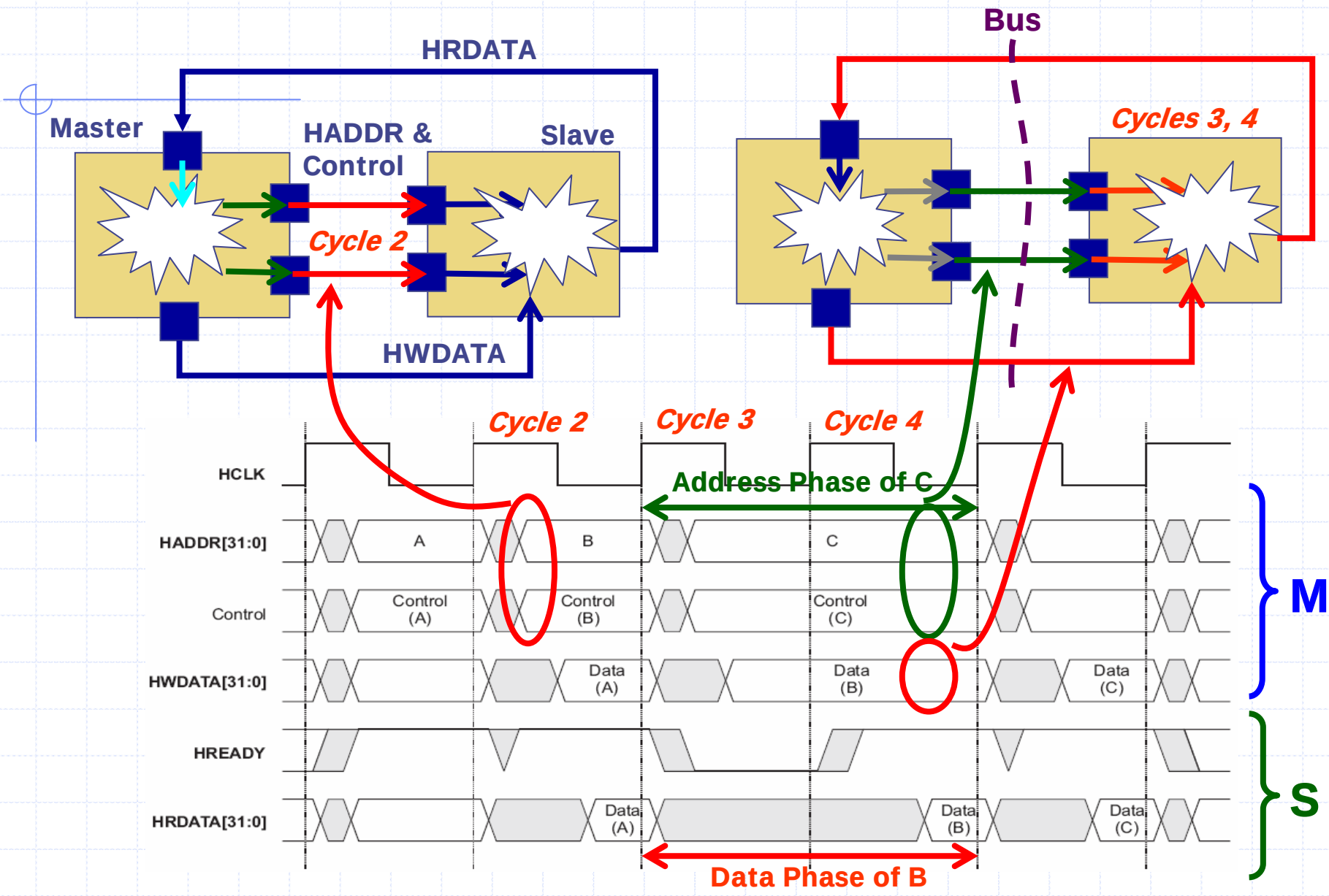
1. Slave samples the address and control signals
2. Slave starts to drive response

1. Master samples the read data and response





Transfer With Wait States



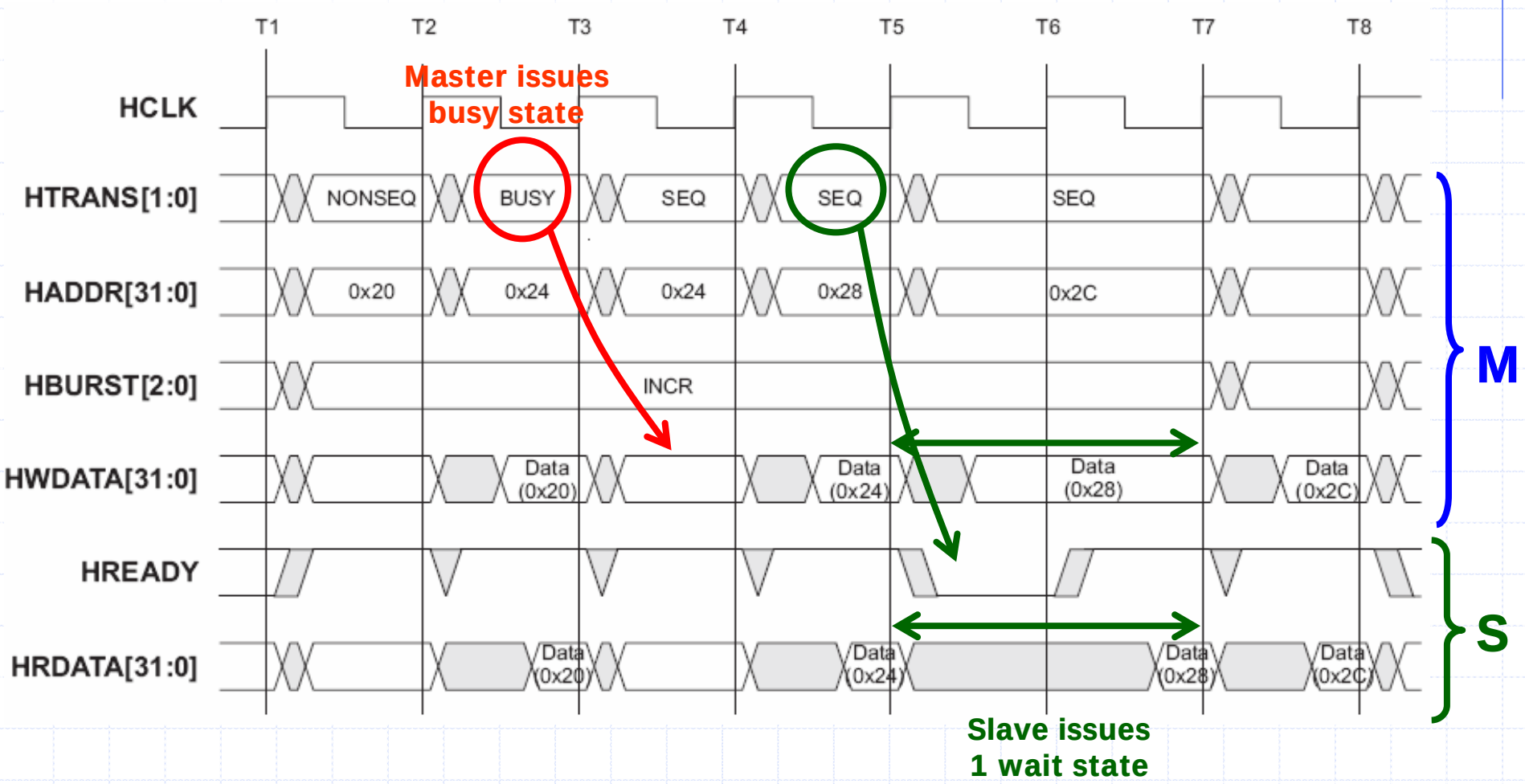
Transfer With Wait States (c. 1)

- The transfers to address A and C are both zero wait state
- The transfer to address B is 1 wait state
- Extending the data phase of the transfer to address B has the effect of extending the address phase of the transfer to address C

Transfer Type (HTRANS[1:0])

- IDLE (00) – Indicate a new transfer
  - No data transfer is required
  - Used when the master is granted the bus without performing transfer
  - The slave must provide a zero wait state OKAY response
- BUSY (01)
  - Allow master to insert IDLE cycle in the middle of bursts of transfers
  - Address and control signals must reflect the next transfer in the burst
  - The slave must provide a zero wait state OKAY response
- NONSEQ (10) – Indicate a new transfer
  - Indicate the first transfer of a burst or a single transfer
  - Address and control signals are unrelated to previous transfer
- SEQ (11)
  - Indicate a remaining transfer in a burst
  - Address = Address of previous transfer + size in bytes

Example of Transfer Type



Burst Transfer

Burst Operation

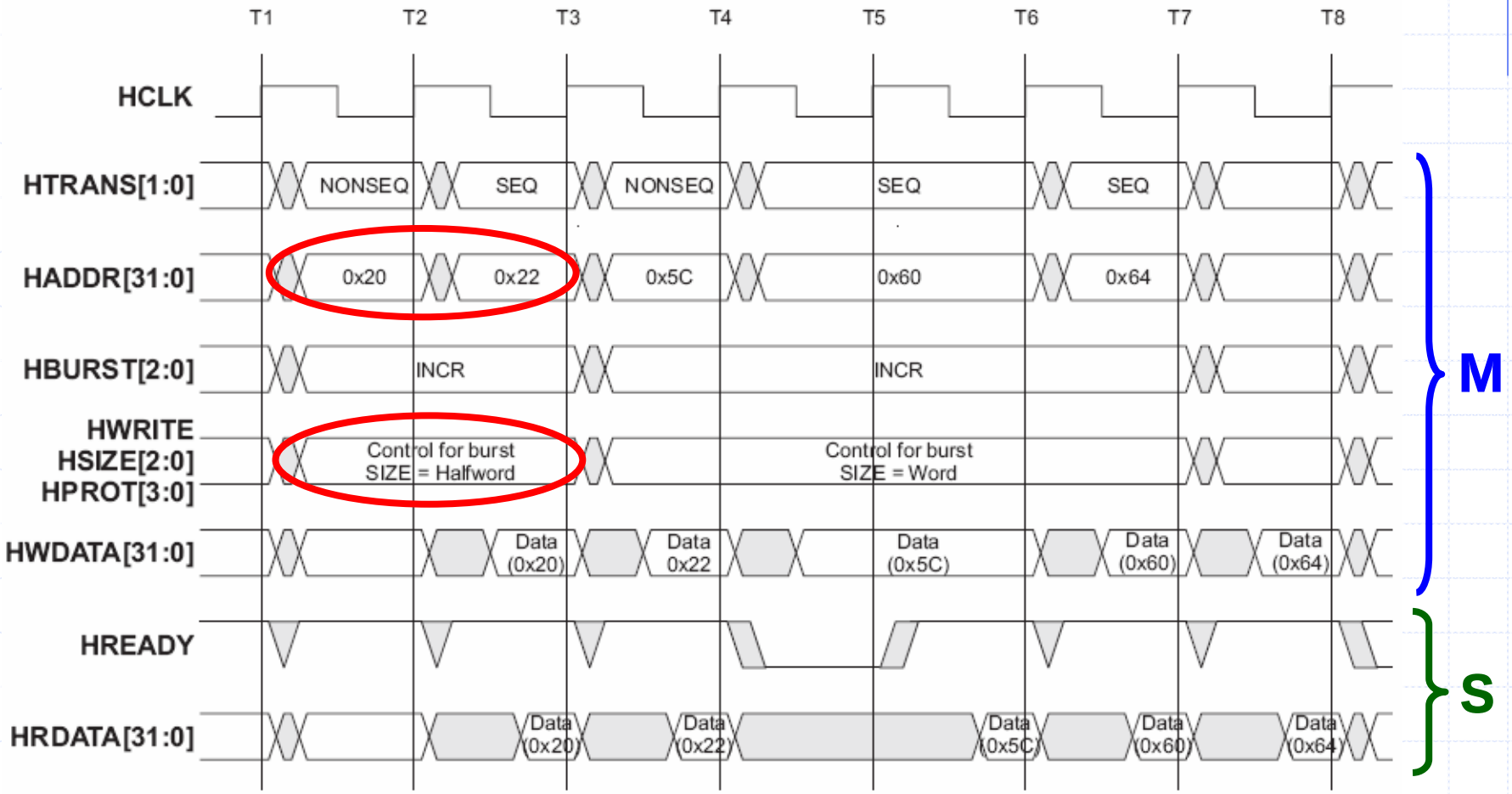
- 4/8/16-beat bursts, as well as undefined-length bursts
- Burst size indicates the number of beats in the burst
- Valid data transferred = (# of beats) x (data size in each beat)
  - Data size in each beat is indicated in HSIZE[2:0]
- Incrementing bursts
  - Access sequential locations with the address of each transfer in the burst being an increment of the previous address
  - An incremental burst can be of any length, but the upper limit is set by the fact that address must not cross a 1KB boundary
- Wrapping bursts
  - If the start address of the transfer is not aligned to the total number of bytes in a burst (size x beats) then the address of the transfers in the burst will wrap when the boundary is reached
  - A 4-beat wrapping burst of word (4-byte) access
  - If start address is 0x38 -> 0x38, 0x3C, 0x30, 0x34

Burst Operation (c. 1)

- Transfers within a burst must be aligned to the address boundary equal to the size of the transfer
  - Word transfers must have A[1:0] = 00
  - Halfword transfers must have A[0] = 0
- Early burst termination
  - Slave can monitor transfer type to detect early burst termination
    - Burst is started with a NONSEQ transfer followed by SEQ/BUSY transfers
    - A NONSEQ/IDLE transfer in a burst indicates the start of a new transfer
  - Master shall complete the burst transfer using undefined-length bursts after it next gains access to the bus

Example of Undefined Length Burst Operation

Valid data transferred = 1 word even though the data bus is of 32-bit width



HWDATA and HRDATA

- Separate read/write data buses in order not to use tristate drivers
- Minimum width of read/write data buses is 32 bits
- HWDATA - driven by the bus master
  - Hold the data valid until the transfer completes
  - Drive the appropriate byte lanes
    - Transfers that are narrower than the width of the bus
  - Burst transfers will have different active byte lanes for each beat
    - A transfer size less than the width of the data bus
- HRDATA - driven by the bus slave
  - Provide valid data only at the end of the final cycle
  - Provide valid data only on the active byte lanes
- All masters and slaves on the bus shall have the same endianness

Endianness

Transfer size	Address offset	DATA [31:24]	DATA [23:16]	DATA [15:8]	DATA [7:0]
Word	0	✓	✓	✓	✓
Halfword	0	-	-	✓	✓
Halfword	2	✓	✓	-	-
Byte	0	-	-	-	✓
Byte	1	-	-	✓	-
Byte	2	-	✓	-	-
Byte	3	✓	-	-	-

Little-Endian

Big-Endian

Transfer size	Address offset	DATA [31:24]	DATA [23:16]	DATA [15:8]	DATA [7:0]
Word	0	✓	✓	✓	✓
Halfword	0	✓	✓	-	-
Halfword	2	-	-	✓	✓
Byte	0	✓	-	-	-
Byte	1	-	✓	-	-
Byte	2	-	-	✓	-
Byte	3	-	-	-	✓

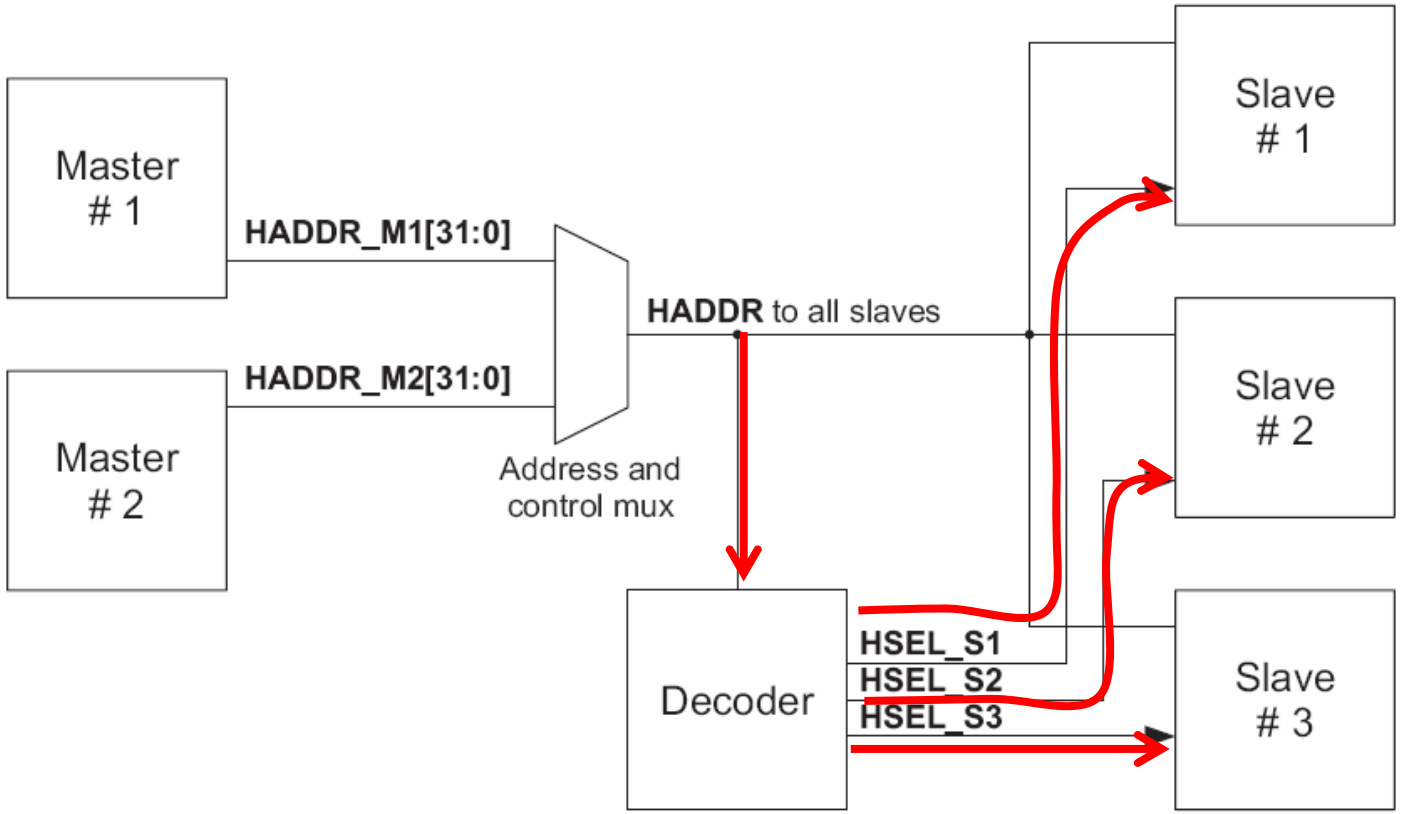
Slave Operation

Slave Address Decoding

- The select signal HSELx for each slave on the bus
  - A combinational decode of the high-order address signals
  - Slaves must only sample the address and control signals and HSELx when HREADY is HIGH
- The minimum address space for a single slave is 1kB
  - The incremental transfers shall not cross a 1kB boundary
  - Ensure a burst never crosses an address decode boundary
- Default slave provide a response when any of the nonexistent address locations are accessed
  - An ERROR response if a NONSEQ or SEQ transfer is attempted to a nonexistent address location
  - An OKAY response if an IDLE or BUSY transfer is attempted to a nonexistent address location

Slave Address Decoding (c. 1)

- A typical address decoding system and slave select signals



Slave Response

- The slave must determine how the transfer should progress after a master has started a transfer
  - Complete the transfer immediately
  - Insert one or more wait states to allow time to complete the transfer
  - Signal an error message indicating that the transfer has failed
  - Delay the completion of the transfer, but allow the master and slave back off the bus, leaving it available for other transfers
- Whenever a slave is accessed it must provide a response indicating the status of the transfer
  - HREADY
  - HRESP

Slave Response (c. 1)

- HREADY signal is used to extend the transfer
  - Low indicates that the data phase is to be extended
  - High indicates that the transfer can complete
- Every slave must have a predetermined maximum wait states
  - Allow the calculation of the latency of accessing the bus
  - It is recommended that slaves do not insert more than 16 wait states
- When it is necessary for a slave to insert a number of wait states prior to deciding what response will be given then it must drive OKAY response

Slave Response (c. 2)

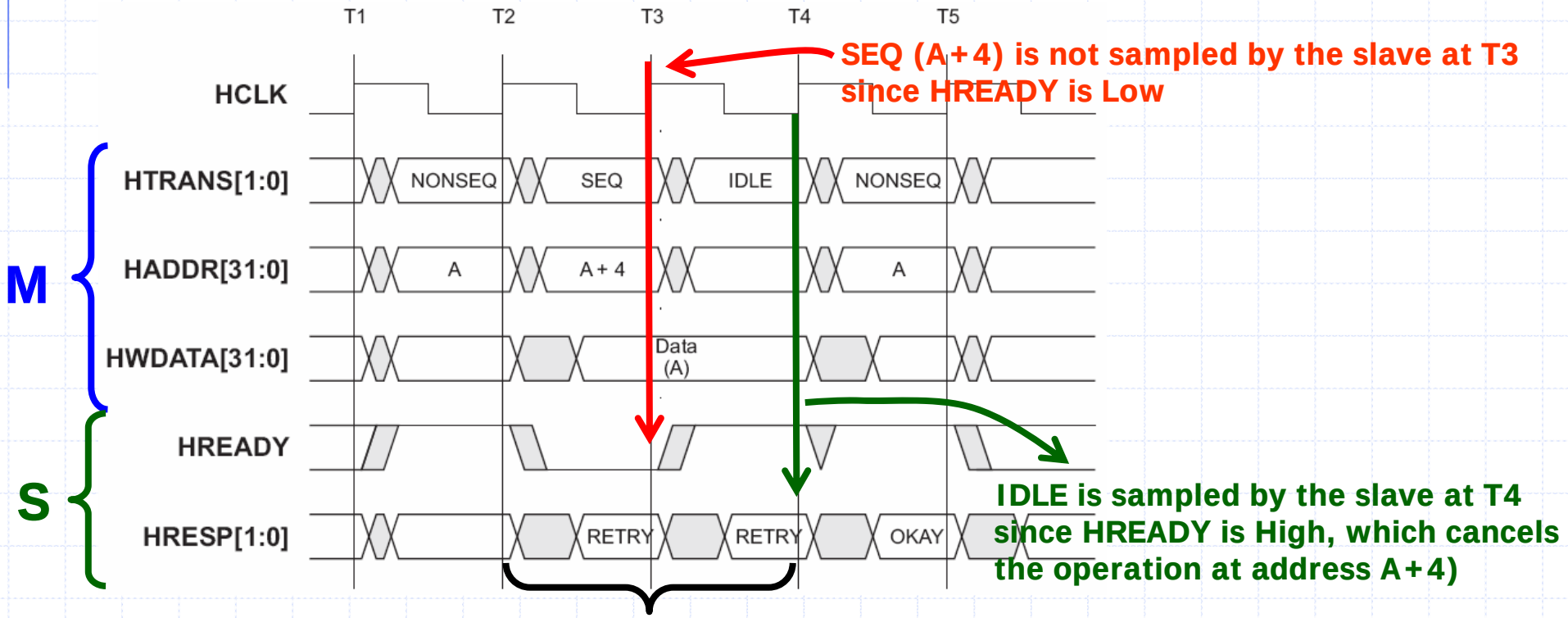
- OKAY
  - Work together with HREADY to indicate the success of the transfer
  - Also used for any additional cycles (wait states) with HREADY low
- ERROR
  - Show an error has occurred
  - Typically used for a protection error, such as an attempt to write to a read-only memory location
- RETRY
  - Show the transfer has not yet completed
  - The master should continue to retry the transfer until it completes
- SPLIT
  - Show the transfer has not yet completed
  - The master must retry the transfer when it is next granted access to the bus
  - The slave will request access to the bus on behalf of the master when the transfer can complete

Two-cycle Response

- OKAY response can be given in a single cycle
- ERROR, SPLIT and RETRY responses require at least two cycles
  - Additional cycles (wait states) at the start of the transfer
    - HREADY will be LOW and the response must be set to OKAY
  - In the penultimate cycle
    - The slave drives HRESP[1:0] to indicate these responses
    - The slave drives HREADY Low to extend the transfer
  - In the final cycle
    - HRESP[1:0] remains driven
    - HREADY is driven High to end the transfer
- SPLIT and RETRY
  - The following transfer must be canceled
- ERROR
  - Optional to cancel the following transfer

Two-cycle Response (c. 1)

- The master starts with a transfer to address A
- The master moves the address on to A+4
- The slave at address A is unable to complete the transfer immediately
- The RETRY response indicates to the master
- The transfer at address A+4 is concealed and replaced by an IDLE transfer



SPLIT and RETRY

- Allow slaves to delay the completion of a transfer
- Free up the bus for use by other masters
- Used by slaves with long latency
- Distinctions between SPLIT and RETRY
  - RETRY – allow masters with higher priority to access the bus
    - The arbiter uses normal priority scheme
    - Masters with higher priority will gain the bus
  - SPLIT – free the bus for the use by other masters (even with lower priority)
    - The arbiter masks the request from the master that has been SPLIT by a slave
    - Any other masters requesting the bus will get access
    - The slave indicates to the arbiter using HSPLIT when it is ready to complete
    - The arbiter unmasks the master and the master in due course will regain the bus
- A bus master should treat RETRY and SPLIT in the same manner
  - Continue to request the bus and attempt the transfer

Bus Arbitration and SPLIT/RETRY Transfers

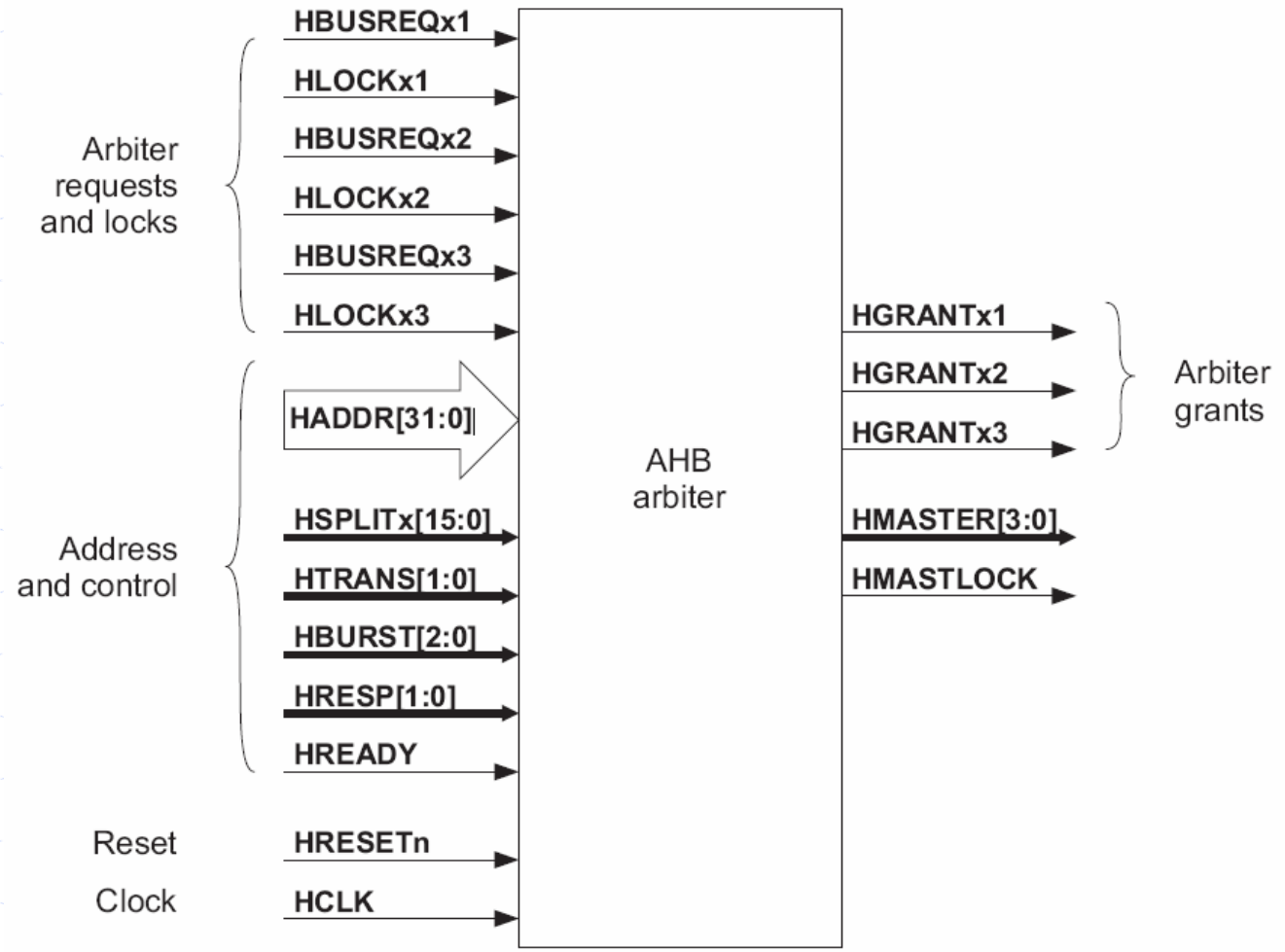
Arbitration

- Ensure that only one master has bus access at any one time
  - Decide which master is of the highest priority
  - Receive requests from slaves that wish to complete SPLIT transfers
- Arbitration signals
  - HBUSREQx
    - Used by a bus master to request access to the bus
    - There can be up to 16 separate bus masters
  - HLOCKx
    - Asserted by the master at the same time as HBUSREQx
    - Indicate that the master is performing a number of indivisible transfers
  - HGRANTx
    - Generated by the arbiter and indicate the master for bus access
    - A master gains ownership of the address bus when HREADY=HIGH, HGRANTx=HIGH at the rising edge of HCLK

Arbitration (c. 1)

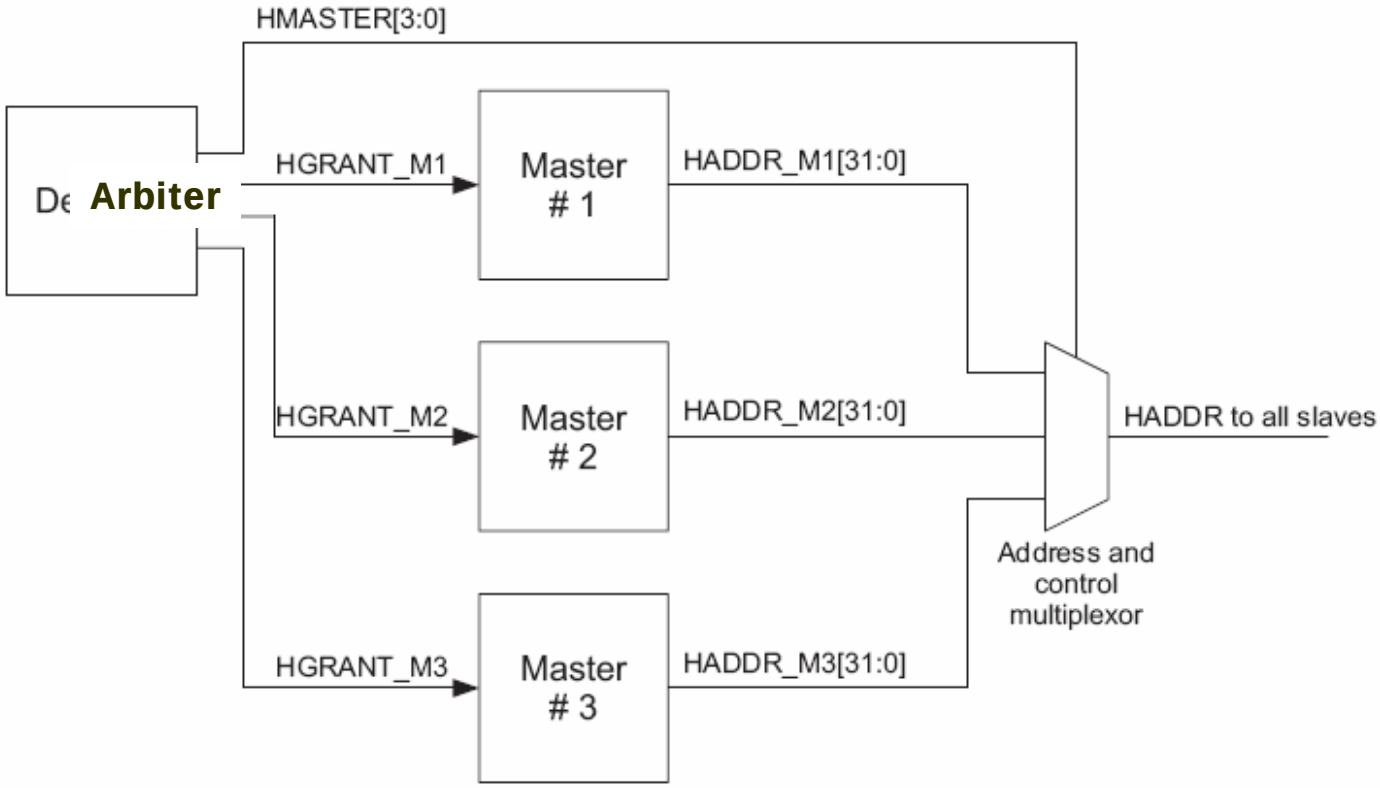
- HMASTER[3:0]
  - Arbiter indicates which master is currently granted the bus
  - Control the central address and control MUX
  - Required by SPLIT-capable slaves so that they can indicate to the arbiter which master is able to complete a SPLIT transaction
- HMASTERLOCK
  - The arbiter indicates that the current transfer is part of a locked sequence by asserting the **HMASTLOCK** signal
- HSPLIT[15:0]
  - The 16-bit bus is used by a SPLIT-capable slave to indicate which bus master can complete a SPLIT transaction
  - Required by the arbiter

Arbitration (c. 2)



Arbitration (c. 3)

- The **HGRANTx** signal is used by the master to determine when it owns the address bus
- Since a central multiplexor is used, each master can drive out the address of the transfer immediately and it does not need to wait until it is granted the bus
- A delayed version of the **HMASTER** bus is used to control the write data multiplexor

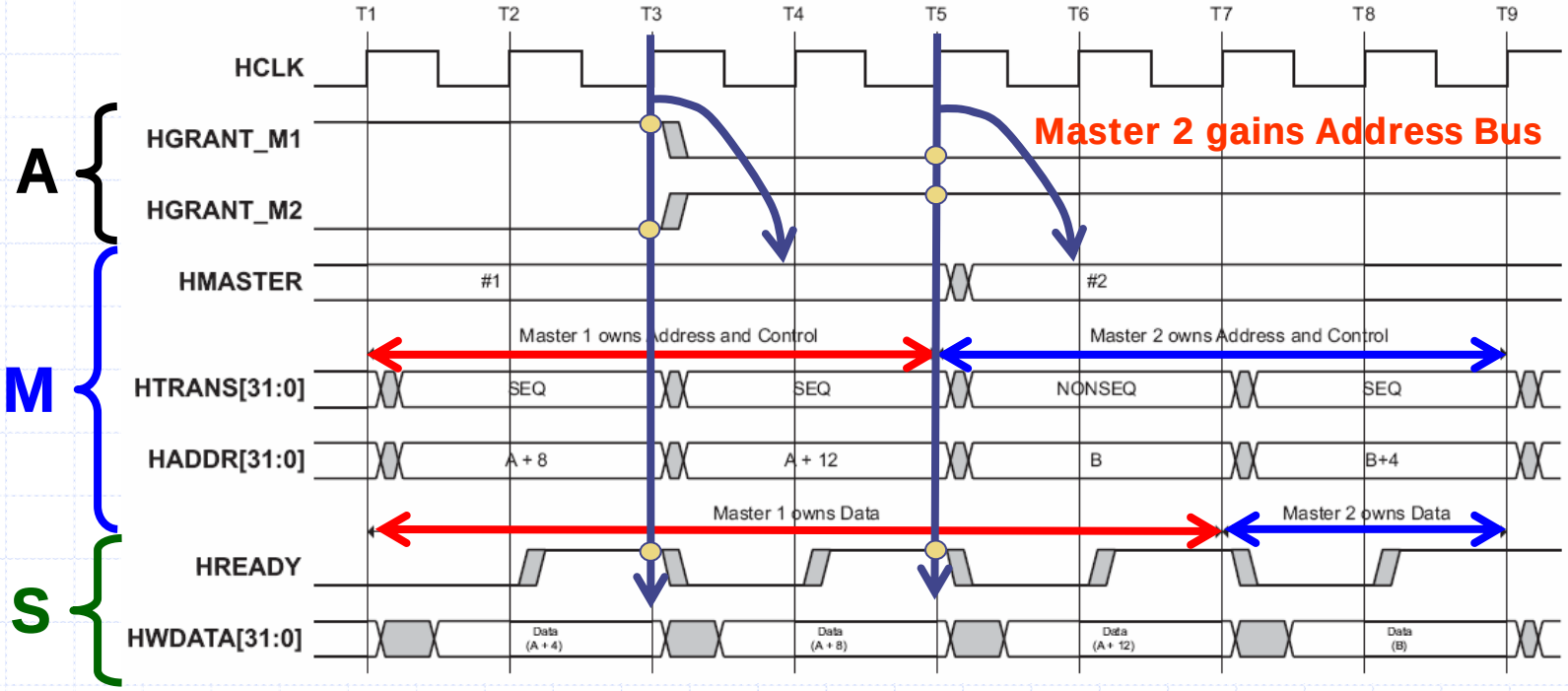


Request Bus Access

- A bus master uses the **HBUSREQx** signal to request bus at any time
  - Assert the **HLOCKx** signal for locked accesses
  - An undefined length burst
    - The master asserts the request until it has started the last transfer
  - A fixed length burst
    - Not necessary to keep requesting the bus during the burst transfer
- The arbiter will sample the request on the rising of the clock
  - Normally grant a different bus master when a burst is completing
  - If required, the arbiter can terminate a burst early
  - The master must re-assert **HBUSREQx** to regain access to the bus
- A master can be granted the bus when it is not requesting it
  - Occur when no masters are requesting the bus
  - The arbiter grants access to a default master
  - The master must drive the transfer type **HTRANS** to indicate an IDLE

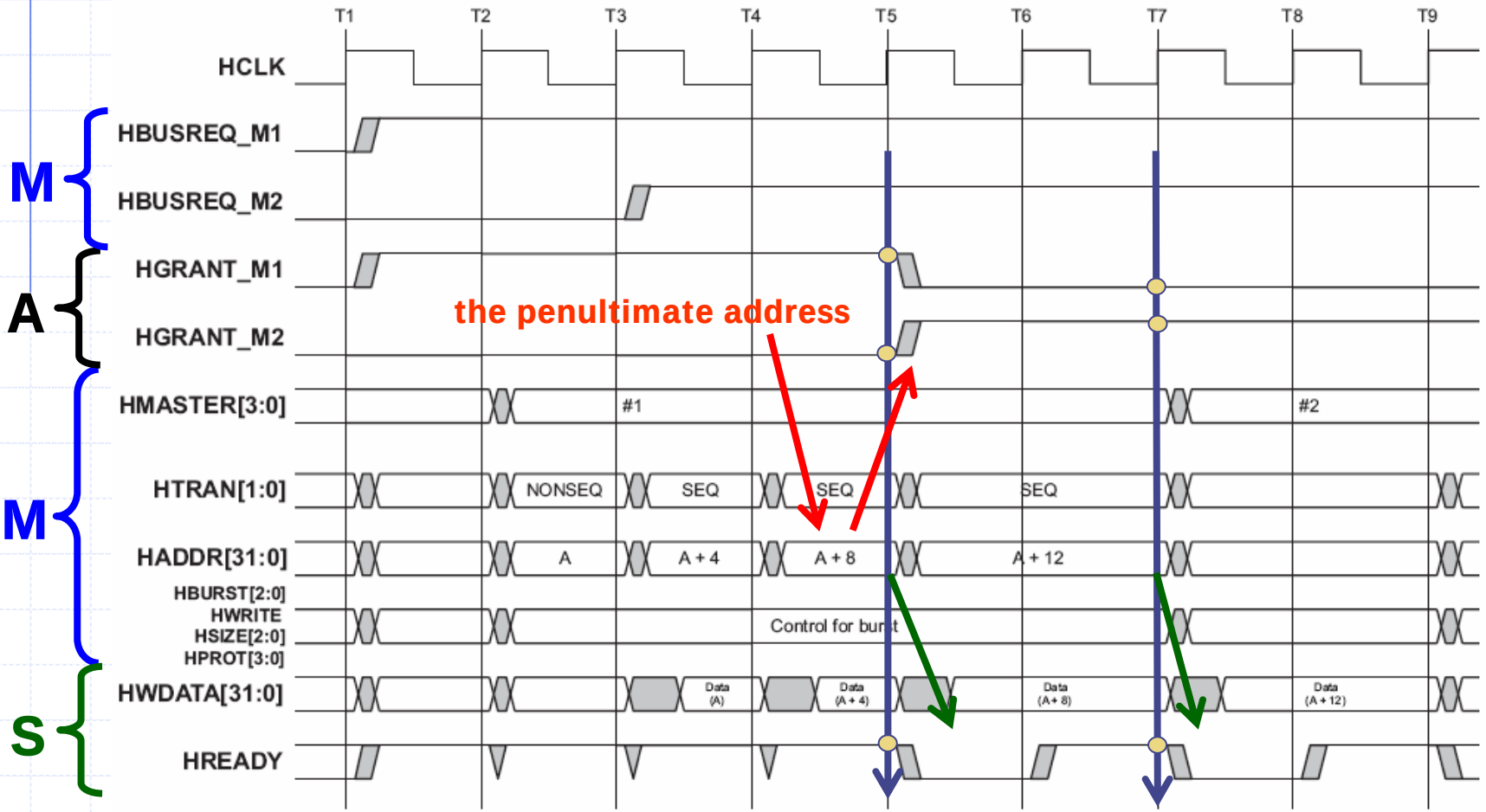
Data Bus Ownership

- Arbiter indicates the bus master for access by **HGRANTx**
- When the current transfer completes (HREADY=High)
  - The master will become granted
  - The arbiter will change the **HMASTER[3:0]** to indicate the master
  - The ownership of the data bus is delayed from that of the address bus



Bus Handover after a BURST Transfer

- Arbiter changes the **HGRANTx** signals when the penultimate address has been sampled
- The new **HGRANTx** will then be sampled as the last address of the burst is sampled

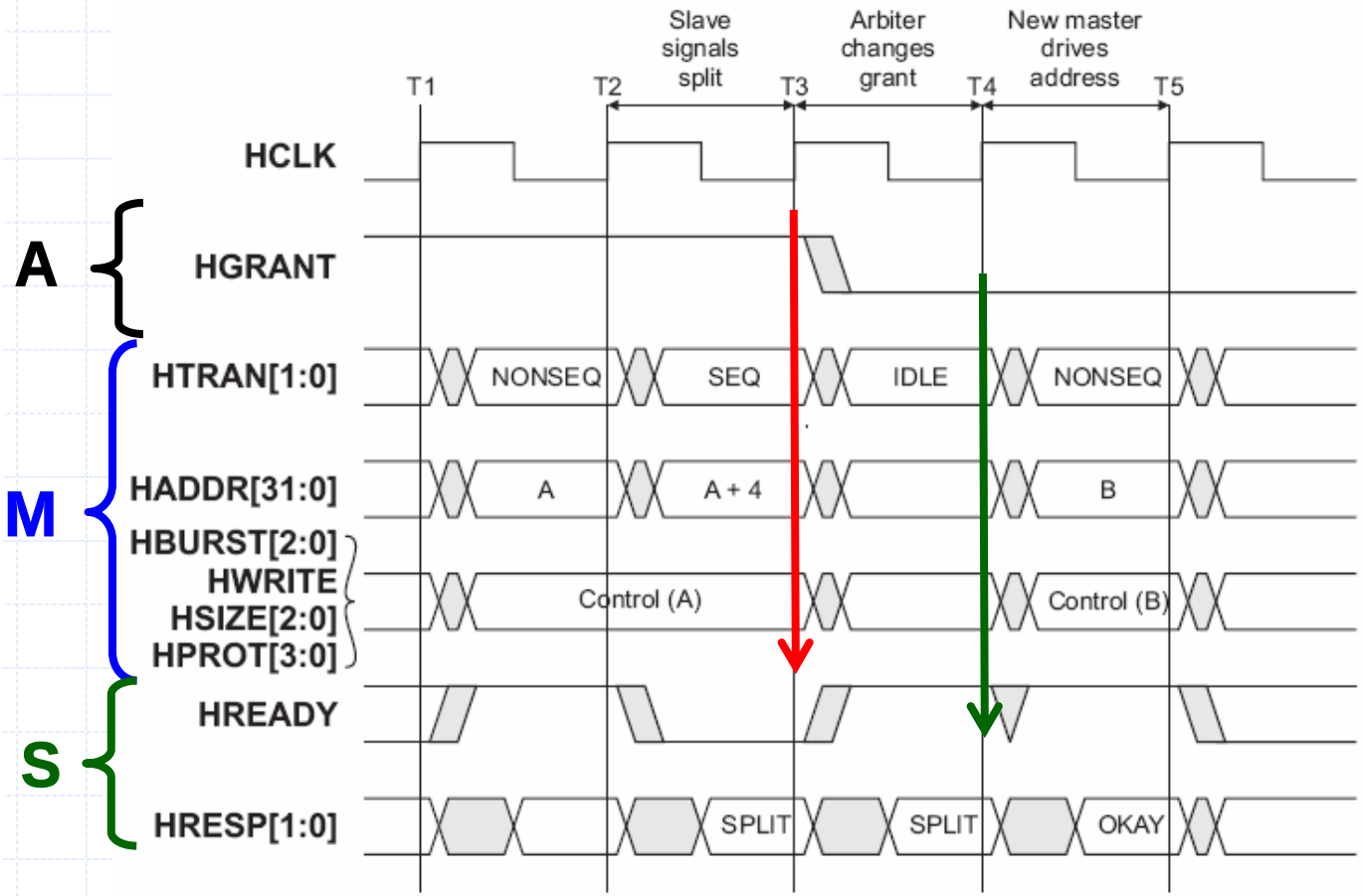


SPLIT Transfer

- Improve the overall utilization of the bus by separating
  - The master providing the address to a slave
  - The slave responding with the appropriate data
- Concepts of operations
  - The arbiter generates the master number on HMASTER[3:0]
  - The slave makes note of HMASTER[3:0] and signals to the arbiter
  - The arbiter masks any requests from masters which have been SPLIT
  - The arbiter grants other masters use of the bus
  - The slave is ready to complete the transfer
  - The slave asserts the appropriate bit on HSPLIT[15:0]
  - The arbiter restores the priority of the appropriate master
  - The master eventually will be granted the bus and re-attempt the transfer
  - When the transfer takes place, the slave finishes with an OKAY response

Bus Handover after SPLIT Transfer

- T1
  - The master issues the address/control
- T2
  - The slave samples the address/control
  - The slave decides to give a SPLIT and set HREADY to Low
- T3
  - The arbiter samples the response and changes arbitration and grant signals
  - The master samples the response and cancel the following operation by putting an IDLE
- T4
  - The new master is granted



Multiple SPLIT Transfers

- AHB protocol only allows one outstanding transaction per master
  - However, a single module may appear as a number of masters
- A SPLIT-capable slave could receive multiple requests
  - A number of masters attempt to access a SPLIT-capable slave
  - The slave issues a SPLIT for each attempt and records the master number without latching the address/control
  - The slave then works through the outstanding requests in an orderly manner
  - The slave asserts the appropriate HSPLITx to indicate that it is ready to complete the transfer
  - The arbiter in due course grants the associated master
  - The master retries the transfer with the address and control
  - The slave latches the address/control and may issue another SPLIT
  - A master thus may be granted the bus a number of times before it is finally allowed to complete the transfer it requires

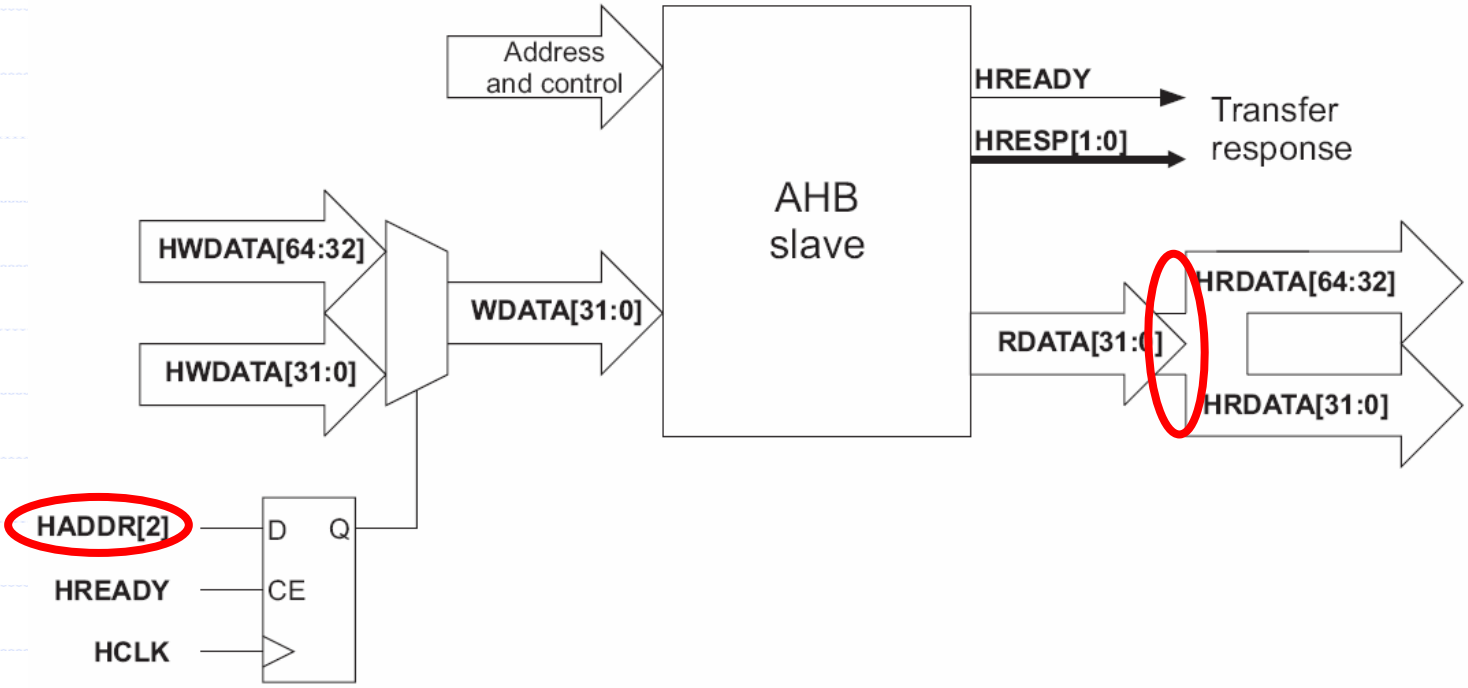
RETRY Transfer

- Slaves that issue RETRY are mostly peripherals and must be accessed by just one master at a time
  - Not enforced by the protocol of the bus
  - Should be ensured by the system
- The slave can check every transfer attempt that is made to ensure the master number is the same
- If the master number is different, then it can take an alternative course of action, such as:
  - An ERROR response
  - A signal to the arbiter
  - A system level interrupt
  - A complete system reset

Portability

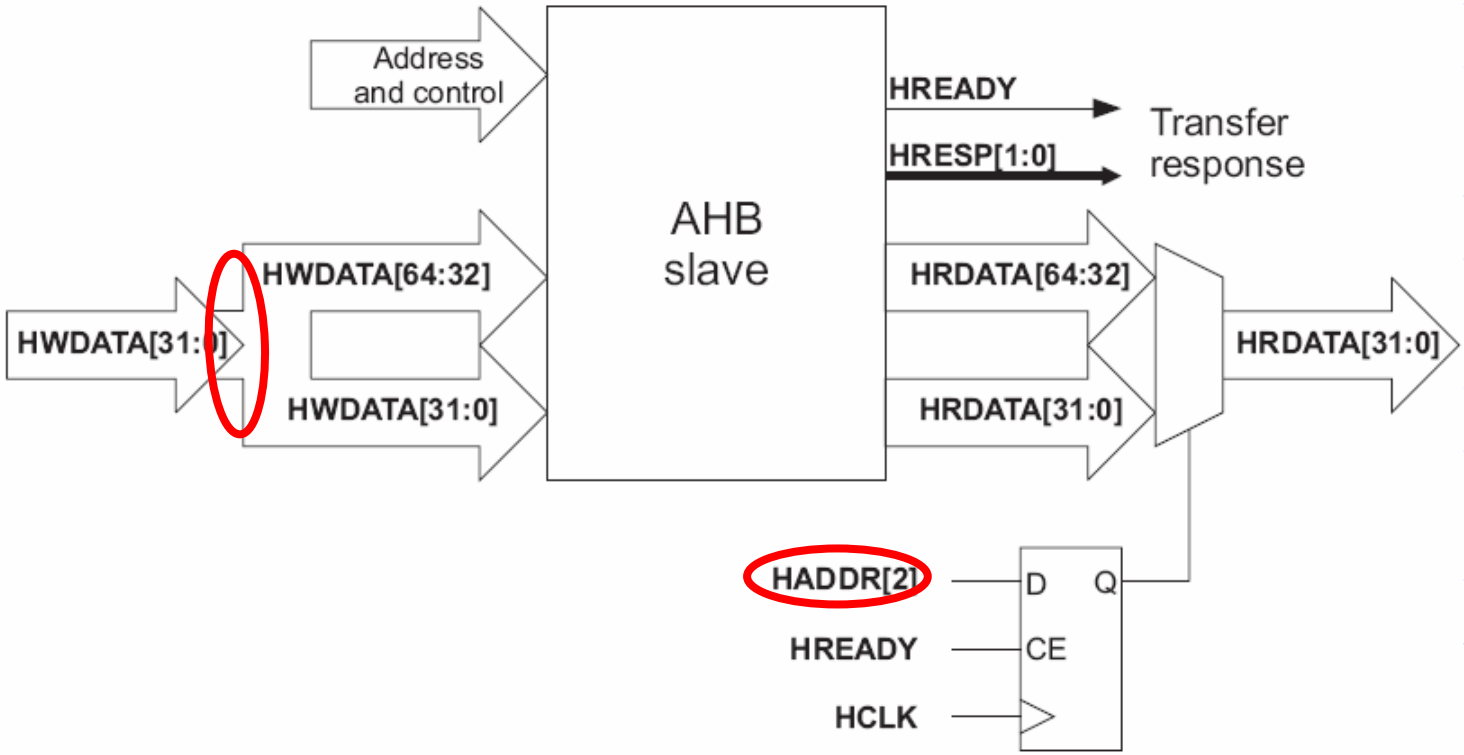
Implementing a Narrow Slave on a Wider Bus

- Only require the addition of external logic
- A slave can only accept transfers that are as wide as its natural interface
- If a master attempts a transfer that is wider than the slave can support then the slave can use the ERROR transfer response



Implementing a Wide Slave on a Narrow Bus

- Only require the addition of external logic



Master

- Masters can easily be modified to work on a wider bus similarly
  - Multiplexing the input bus
  - Replication of the output bus
- Masters cannot be made to work on a narrower bus
  - There must be some mechanisms to limit the width of transfers that the bus master attempts
- Masters must never attempt a transfer where the width (as indicate by **HSIZE**) is wider than the data bus