AMBA 2.0 AHB/APB Bus Protocol and Bus Components

2010-07-05

Kim, Si-Hyun

Design Technology (DT) Part
DIS Group
System IC Business Team
LG Electronics

Please send comments via email to: *sihyun1.kim@lge.com*

This documentation contain confidential and proprietary information that is the property of LG.

LGE CONFIDENTIAL

SYSTEM IC BUSINESS TEAM

Agenda

- Introduction to the AMBA Buses
- AMBA Signals
  - AHB Master, Slave
  - Arbiter
  - Decoder
  - APB Bridge, Slave
- AMBA H/W Components
- AMBA Bus Protocol Level Timing Diagram
- Multi-layer AHB, AHB-Lite
- Quiz

AMBA Spec., Design and Integration Guide: ARM Information Center

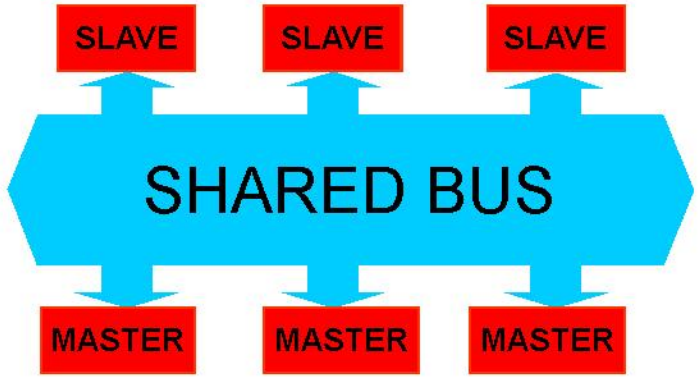
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.amba/index.html

Bus: Definition

- **Set of *wires* connected to many *bus masters and slaves***

- **Bus master and slave wires are grouped in functional groups:**
  - *Address ..... 32bit width*
  - *Data ..... 32/64... bit width*
  - *Control ..... burst type, burst size, data size, handshaking*

- **Wires are shared (Poor scalability but Low area)**
  - Control logic + control wires embody the protocol
    - ✓ Bus Components
      - – Bus arbiter
      - – Address decoder
      - – M2S/S2M MUX
      - – Bridge

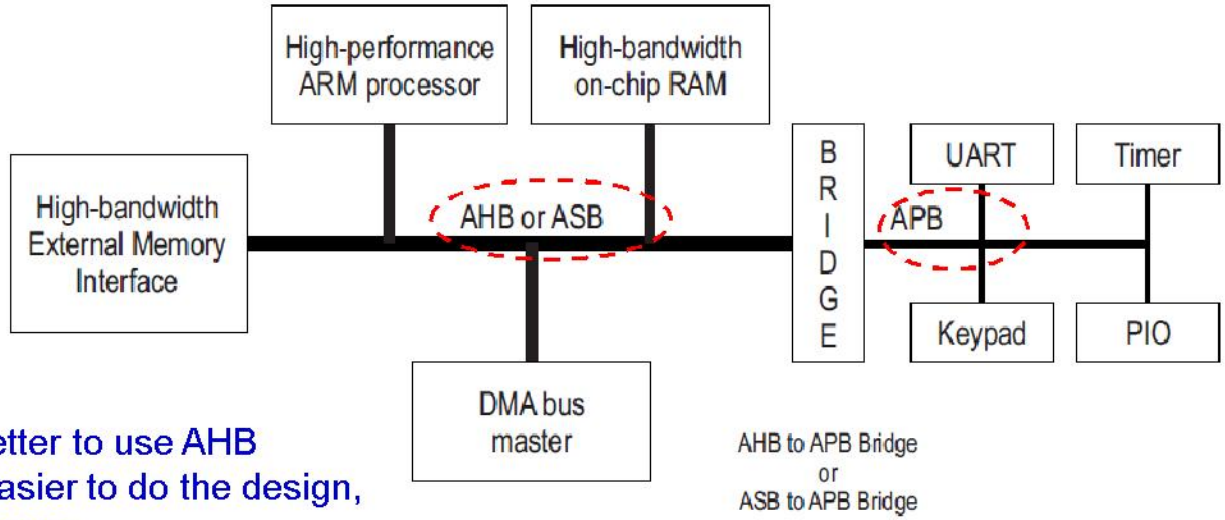


Bus: Sync vs. Async

- **In a synchronous bus**
  - operations are synchronized to a global bus clock
    - ✓ AMBA global bus clock
      - – AHB: HCLK
      - – APB: PCLK
- **In an asynchronous bus**
  - control signal edges signal bus events
- **On-chip buses are generally synchronous**
  - But asynchronous busses are emerging

A typical AMBA AHB-based System

- **Advanced Microcontroller Bus Architecture(AMBA) is a bridged bus**
  - Advanced High-performance Bus (AHB)
  - Advanced Peripheral Bus (APB)



In general it is much better to use AHB rather than ASB. It is easier to do the design, provide better performance.

AMBA AHB

* High performance
* Pipelined operation
* Multiple bus masters
* Burst transfers
* Split transactions

AMBA ASB

* High performance
* Pipelined operation
* Multiple bus masters

AMBA APB

* Low power
* Latched address and control
* Simple interface
* Suitable for many peripherals

AMBA Signal Names

- **All AMBA signals are named such that the first letter of the name indicates which bus the signal is associated with,**
  - AHB signal prefixes
    - ✓ **H** indicates an AHB signal
      - Ex. ***H***CLK, ***H***ADDR, …
  - APB signal prefixes
    - ✓ **P** indicates an APB signal
      - Ex. ***P***CLK, ***P***ADDR, …
- **A lower case n in the signal name indicates that the signal is active LOW, otherwise signal names are always all upper case (active HIGH).**
  - HRESET***n***

AHB Signal List(1)

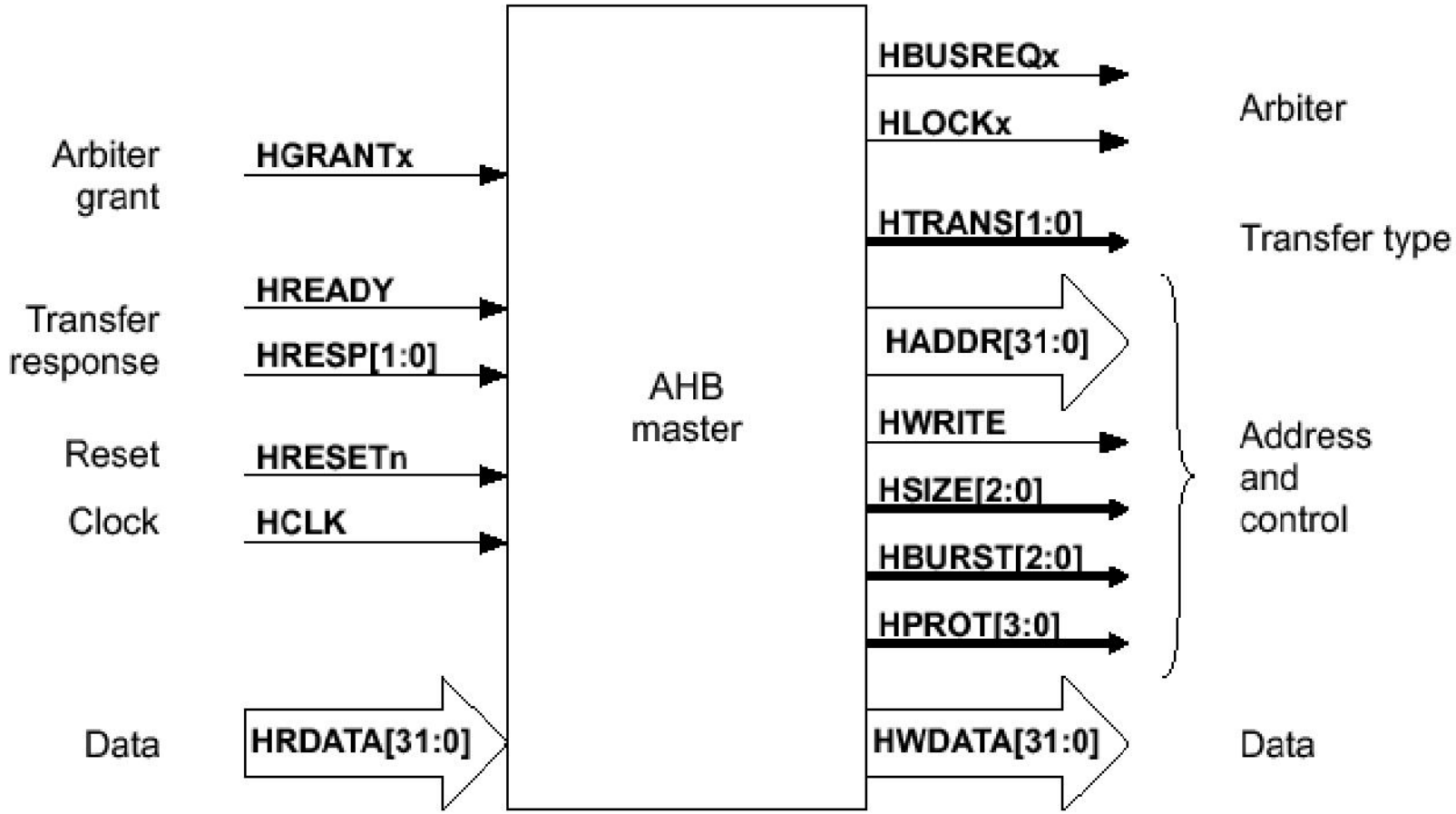
Name	Source	Description
HCLK	Clock source	Bus clock (rising edge)
HRESETn	Reset controller	Reset(Active low) connected directly to the system bus reset signal
HADDR[31:0]	Master	Address bus 32-bit system address bus
HTRANS[1:0]	Master	Transfer type NONSEQUENTIAL, SEQUENTIAL, IDLE or BUSY.
HWRITE Control Signals	Master	Transfer direction WR:HIGH, RD:LOW
HSIZE[2:0]	Master	Transfer size byte (8-bit), halfword (16-bit) or word (32-bit).
HBURST[2:0]	Master	Burst type 1,4,8,16,X (increment, wrap)
HPROT[3:0]	Master	Protection control
HWDATA[31:0]	Master	Write databus 32, 64, ...
HSELx	Decoder	Slave select There is a HSELx signal for each bus slave.
HRDATA[31:0]	Slave	Read data bus
HREADY	Slave	Transfer done

AHB Signal List(2)

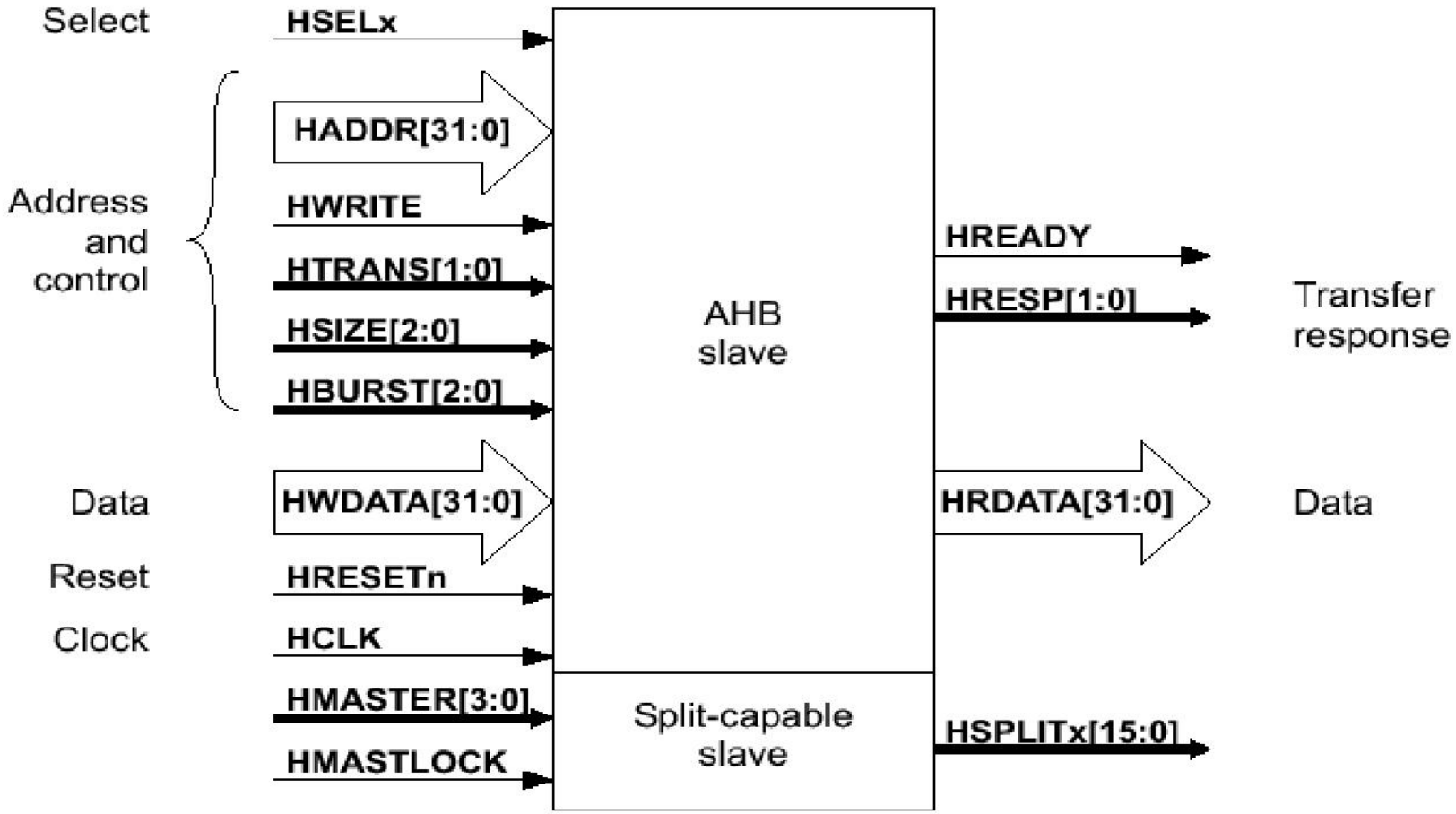
Name	Source	Description	
HRESP[1:0]	Slave	Transfer response	OKAY,ERROR, RETRY and SPLIT
HBUSREQx	Master	Bus request	maximum of 16 bus masters
HLOCKx	Master	Locked transfers	no other master should be granted the bus
HGRANTx	Arbiter	Bus grant	bus master x is currently the highest priority master.
HMASTER[3:0]	Arbiter	Master number	MUX(M2S, S2M) select signal
HMASTLOCK	Arbiter	Locked sequence	current M is performing a locked transfers.
HSPLITx[15:0]	Slave	Split completion request	

Arbitration signals

AHB Master


Arbiter grant
HGRANTx
Transfer response
HREADY
HRESP[1:0]
Reset
HRESETn
Clock
HCLK
Data
HRDATA[31:0]
AHB master
HBUSREQx
HLOCKx
Arbiter
HTRANS[1:0]
Transfer type
HADDR[31:0]
HWRITE
HSIZE[2:0]
HBURST[2:0]
HPROT[3:0]
Address and control
HWDATA[31:0]
Data

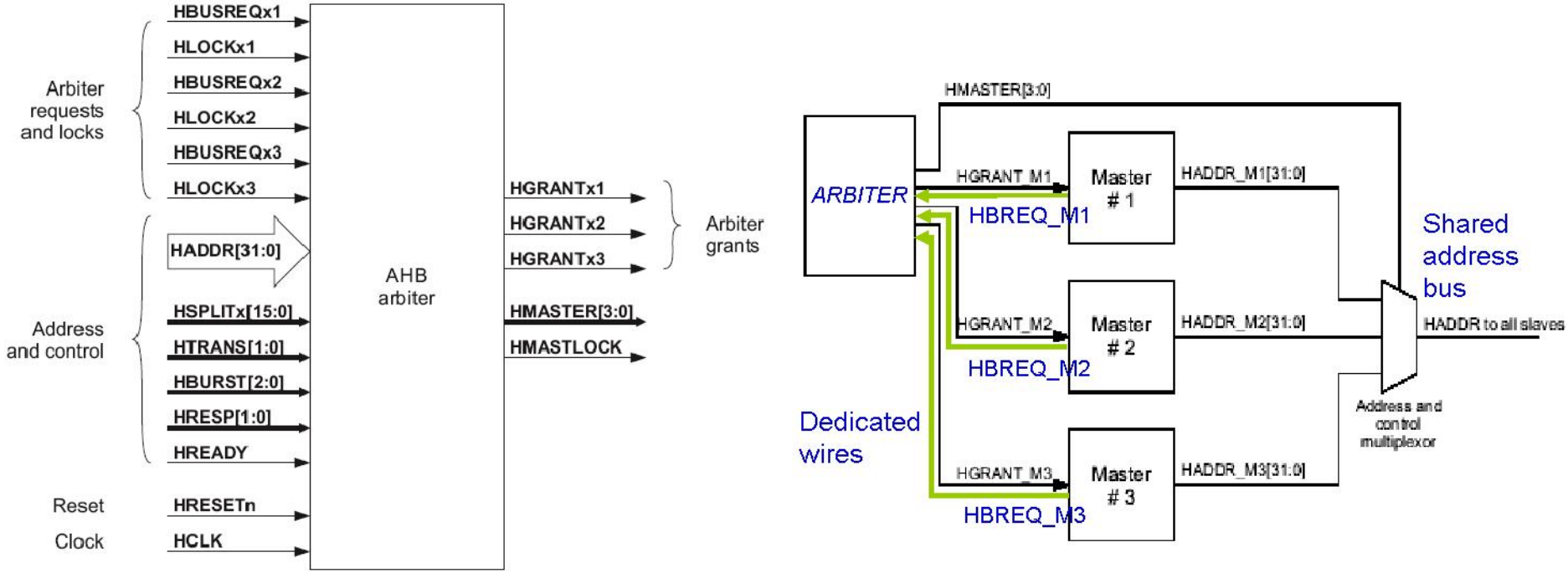
AHB Slave


Select
HSELx
HADDR[31:0]
Address and control
HWRITE
HTRANS[1:0]
HSIZE[2:0]
HBURST[2:0]
Data
HWDATA[31:0]
Reset
HRESETn
Clock
HCLK
HMASTER[3:0]
HMASTLOCK
AHB slave
Split-capable slave
HREADY
HRESP[1:0]
Transfer response
HRDATA[31:0]
Data
HSPLITx[15:0]

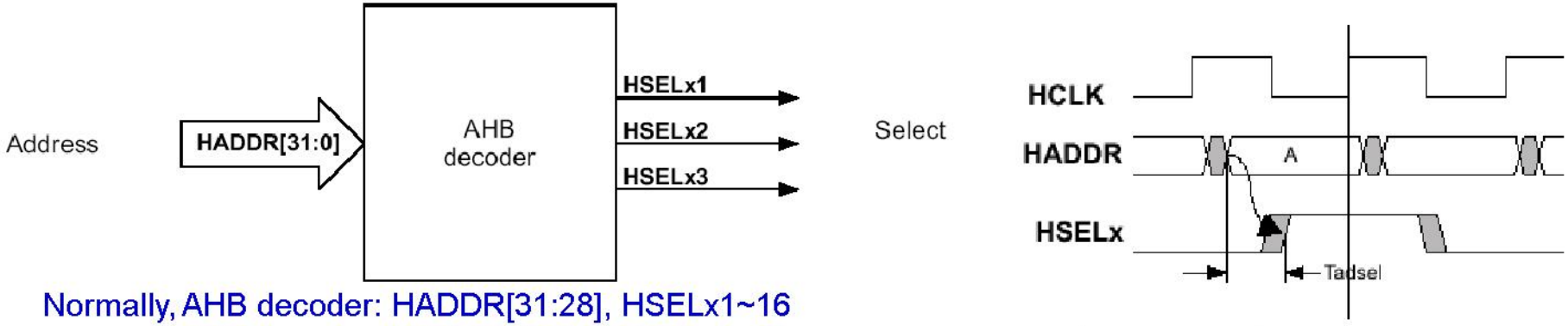
AHB Arbiter

❑ Every bus master has a REQUEST/GRANT interface to the arbiter

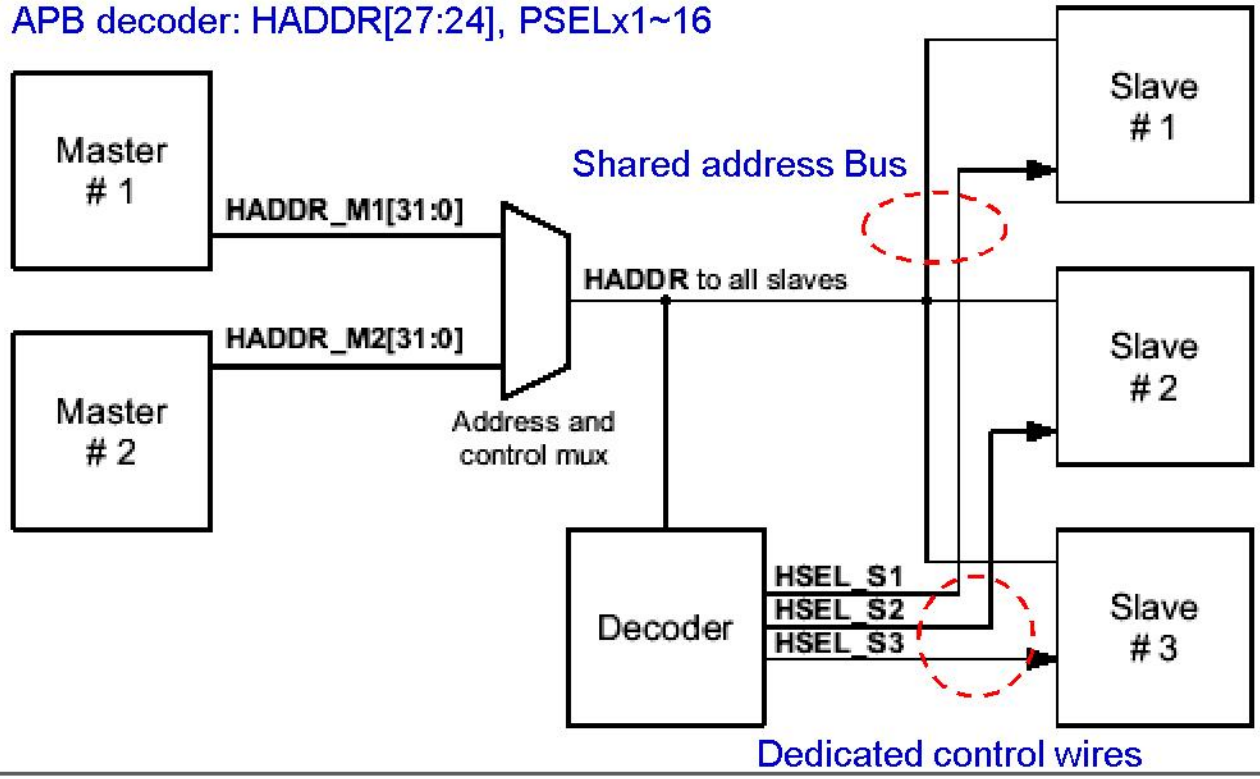
- Arbiter uses a prioritization scheme to decide which bus master is currently the highest priority master requesting the bus.
  - ✓ Arbitration policy is NOT specified by AMBA
    - – Fixed, round-robin, weight group, daisy-chain, …



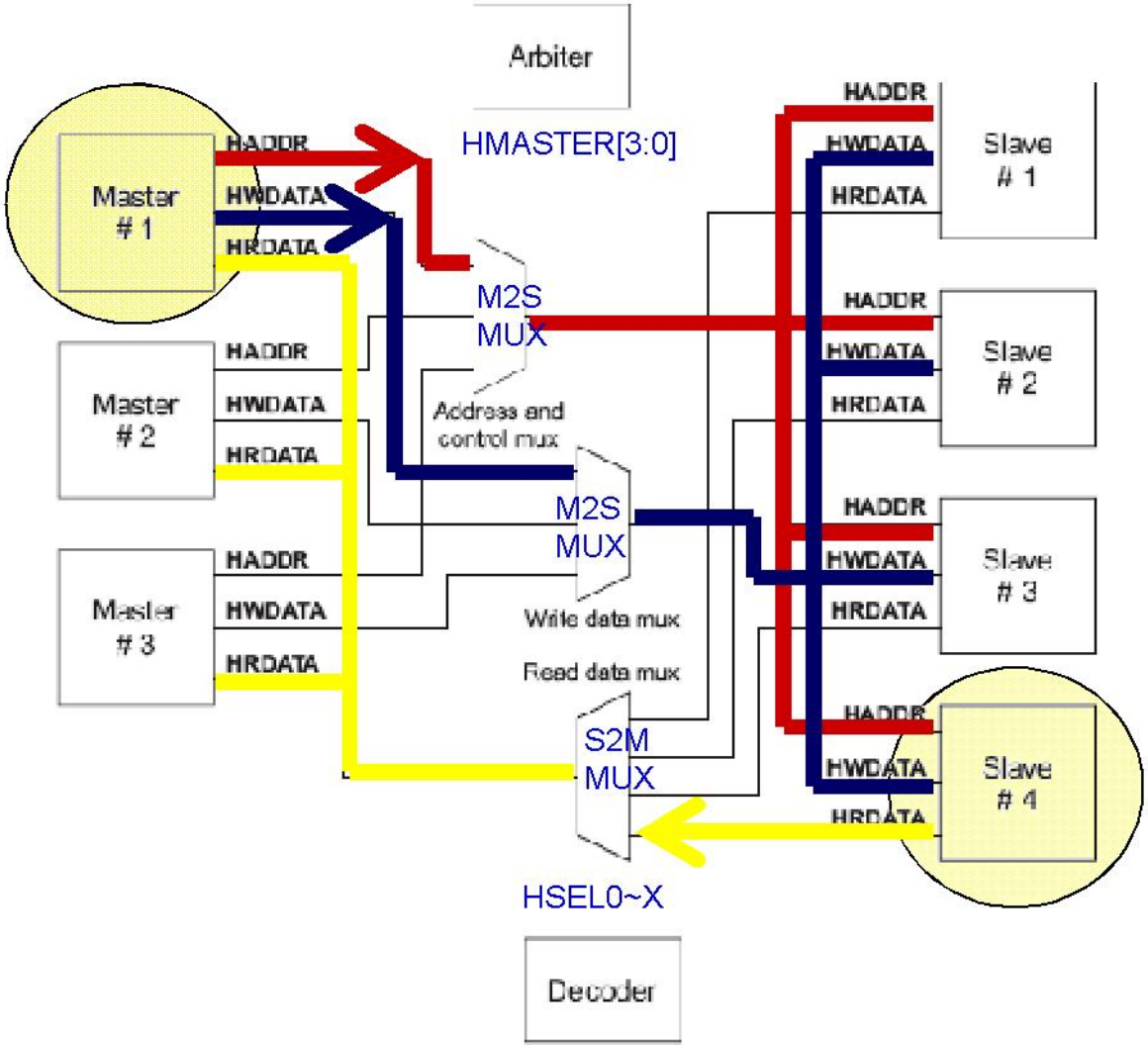
AHB Address Decoder



Normally, AHB decoder: HADDR[31:28], HSELx1~16
APB decoder: HADDR[27:24], PSELx1~16

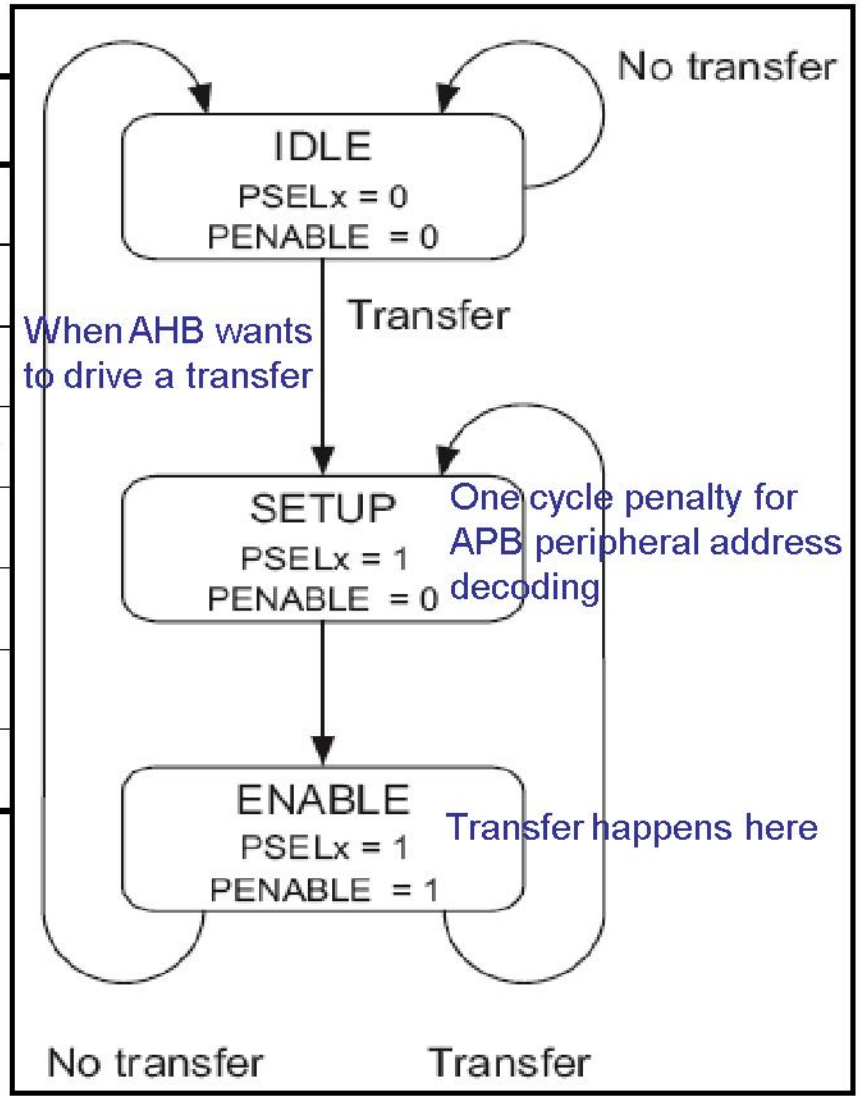


Multiplexor Interconnection



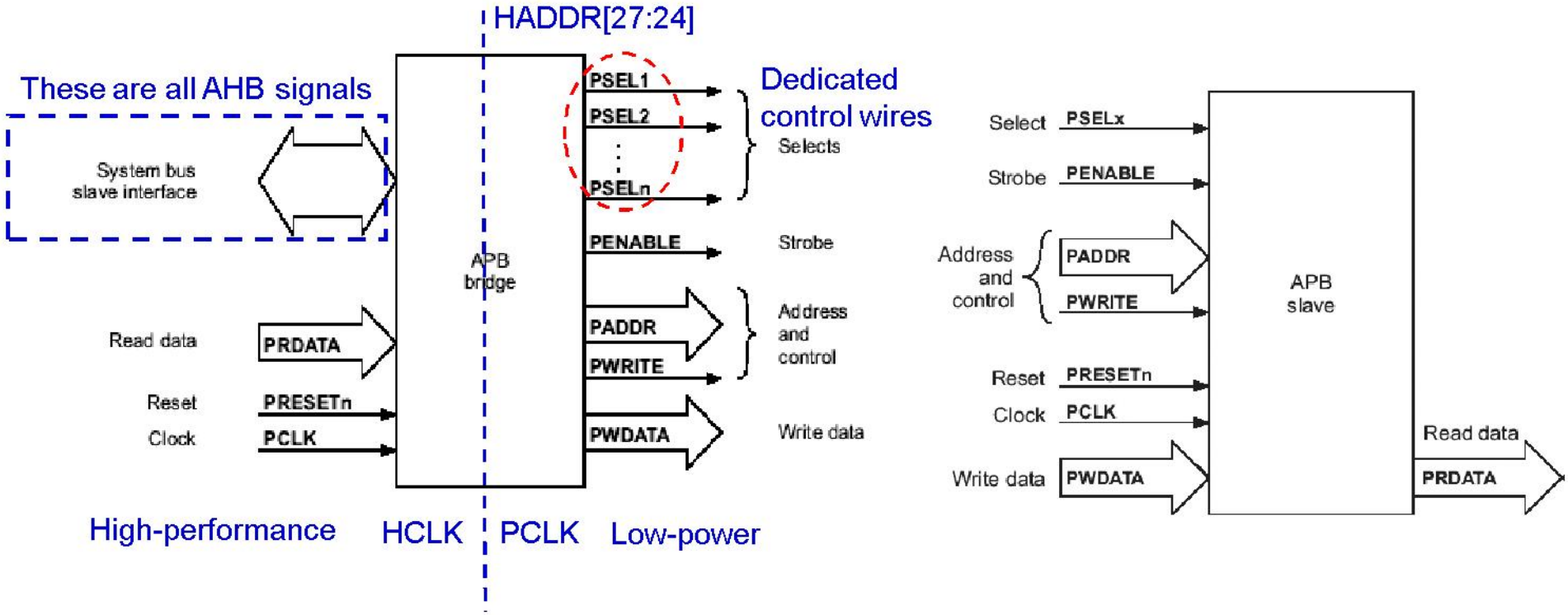
APB Signal List with State Diagram

Name	Description
PCLK	Bus clock (Rising edge)
PRESETn	APB reset (Active low)
PADDR[31:0]	APB address bus
PSELx	APB select Secondary decoder
PENABLE	APB strobe
PWRITE	APB transfer direction
PRDATA	APB read data bus
PWDATA	APB write data bus

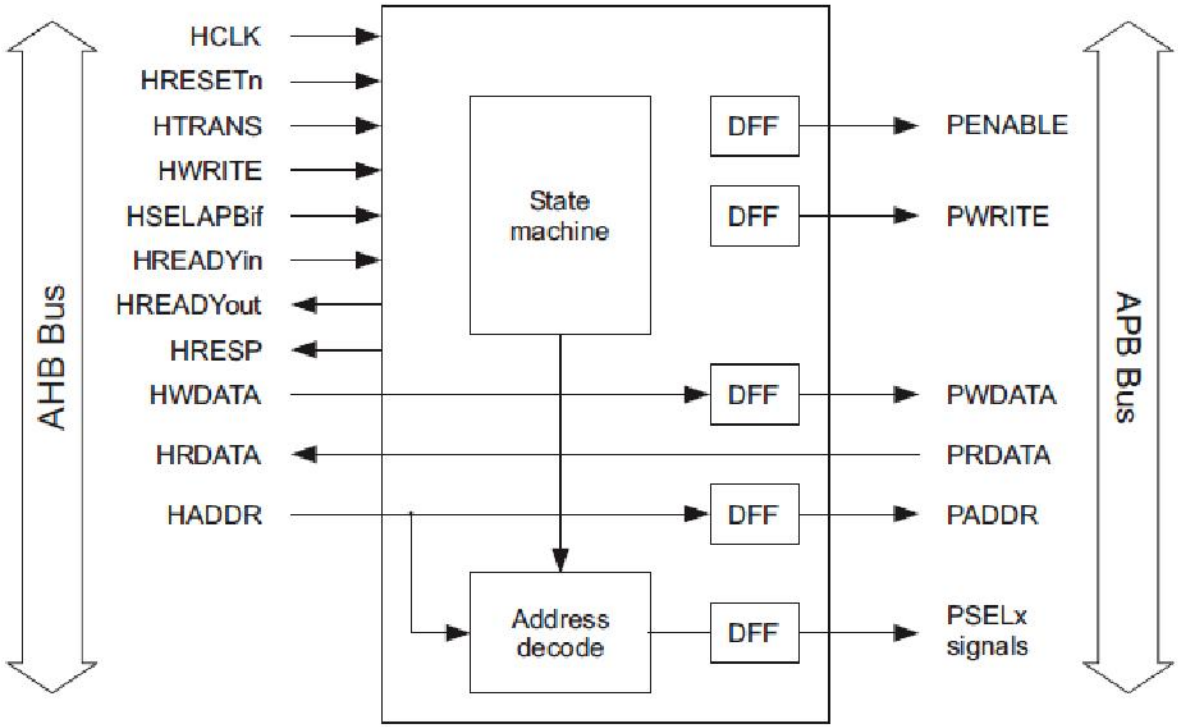


NOTE: no multi-cycle (burst) transfers, no pipelining

AHB-APB Bridge and APB Slave

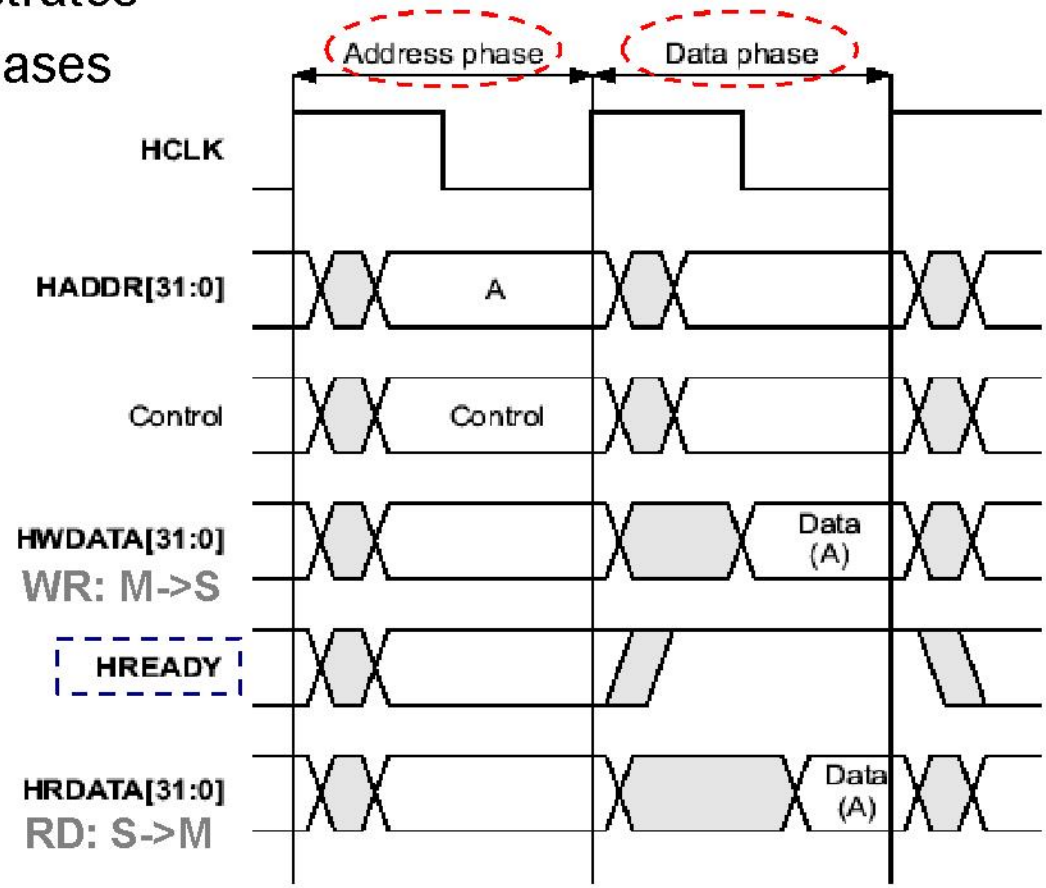


Block Diagram of Bridge Module



AHB Simple Transfer with *no wait states*

- This simple example demonstrates how the address and data phases of the transfer occur during different clock periods.



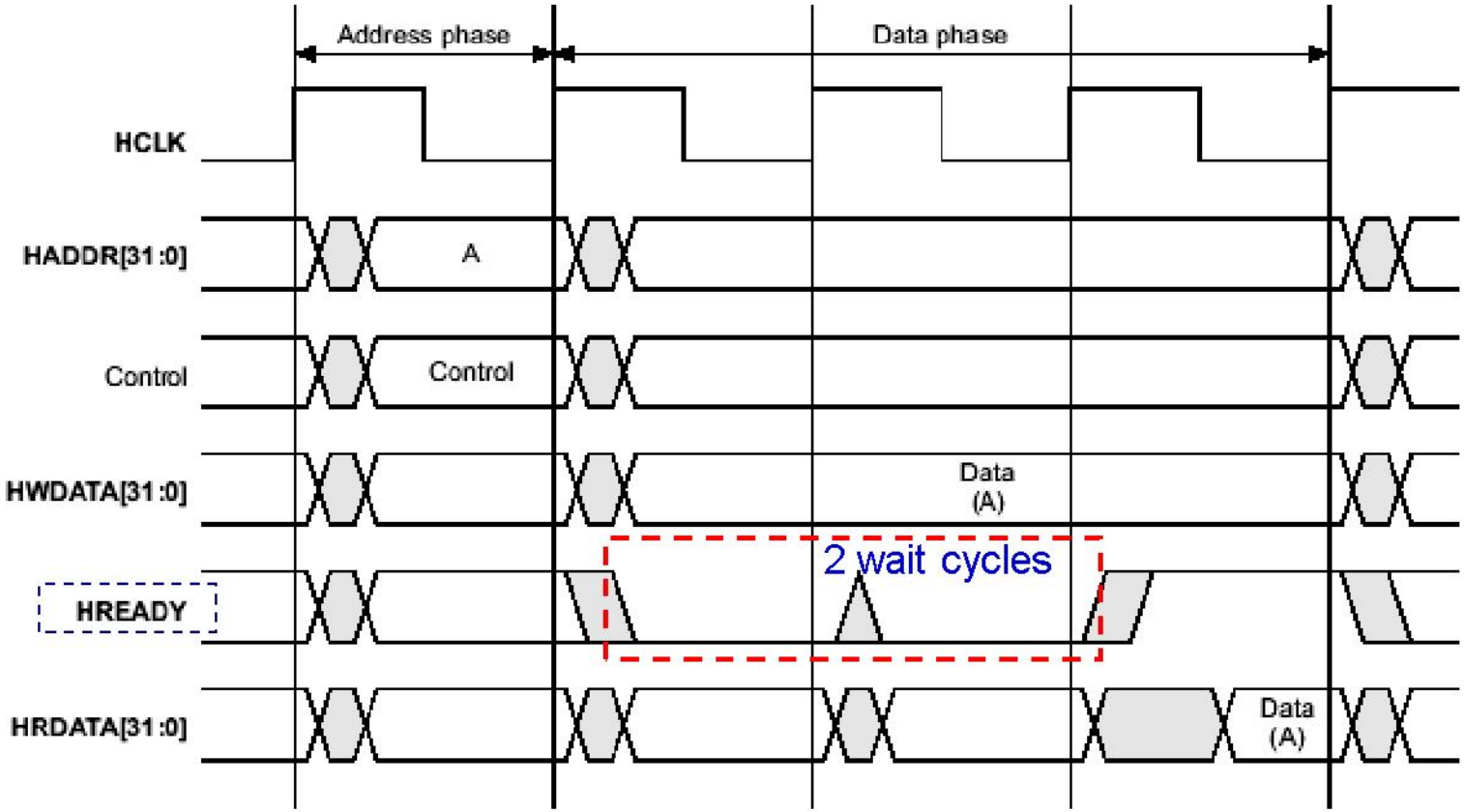
- In fact, the address phase of any transfer occurs during the data phase of the previous transfer.

- This overlapping of address and data is fundamental to the pipelined nature of the bus, and allows for high performance operation.

AHB Simple Transfer with *wait states*

❑ A slave may insert wait states into any transfer

- WR: master will hold the data stable throughout the extended cycles.
- RD: slave does not have to provide valid data until the transfer is about to complete.



Endianness of AMBA System

- AHB/APB does not specify the required endianness.
  - In order for the system to function correctly it is essential that all modules are of the **same endianness** and also that any data routing or bridges are of the same endianness.
  - **Dynamic endianness is not supported**, because in the majority of embedded systems, this would lead to a significant silicon overhead that is redundant.
  - 8-bit(byte) address increment.

Table 3-6 Active byte lanes for a 32-bit little-endian data bus

HSIZE[2:0] HADDR[1:0]

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	✓	-	-	-

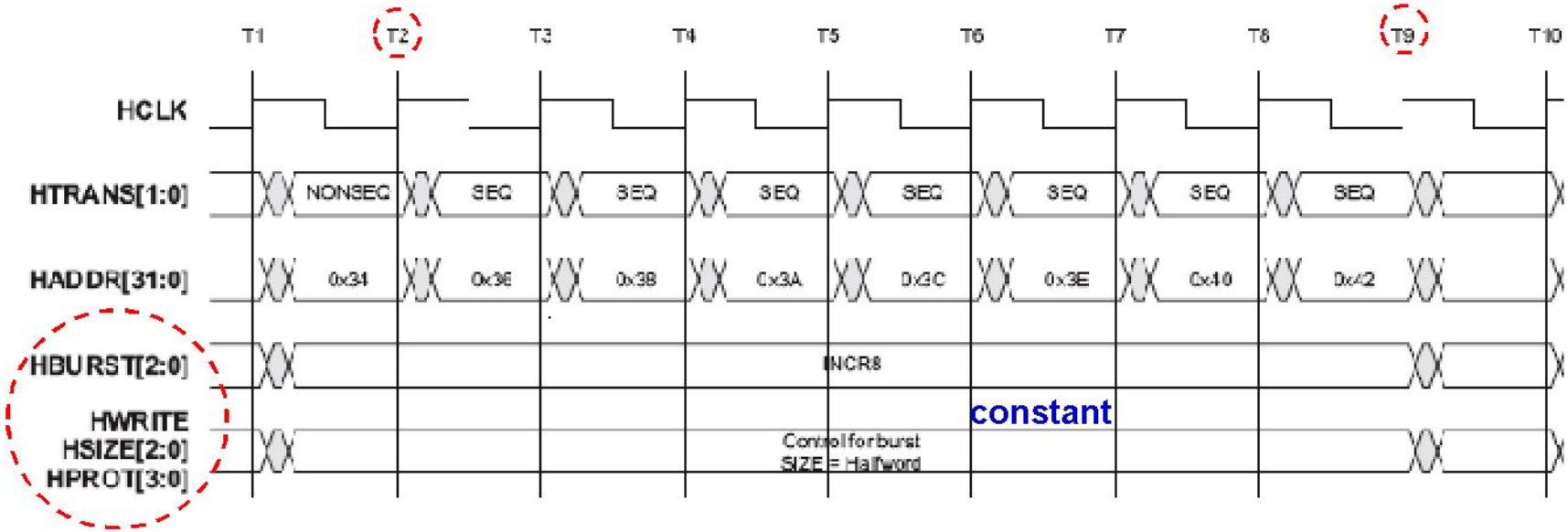
Table 3-7 Active byte lanes for a 32-bit big-endian data bus

HSIZE[2:0] HADDR[1:0]

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	-	-	-	✓

Control Signal Constant

- Master control signals have exactly the same timing as the address bus.
  - They must remain constant throughout a burst of transfers.
    - ✓ Constant:
      - HBURST, HWRITE, HSIZE, HPROT
    - ✓ Dynamic:
      - HTRNAS



Control Signal: Transfer Size

- **HSIZE[2:0]** indicates the size of the transfer.
- The size is used in conjunction with the HBURST[2:0] signals to determine the address boundary for wrapping bursts (slide #23).

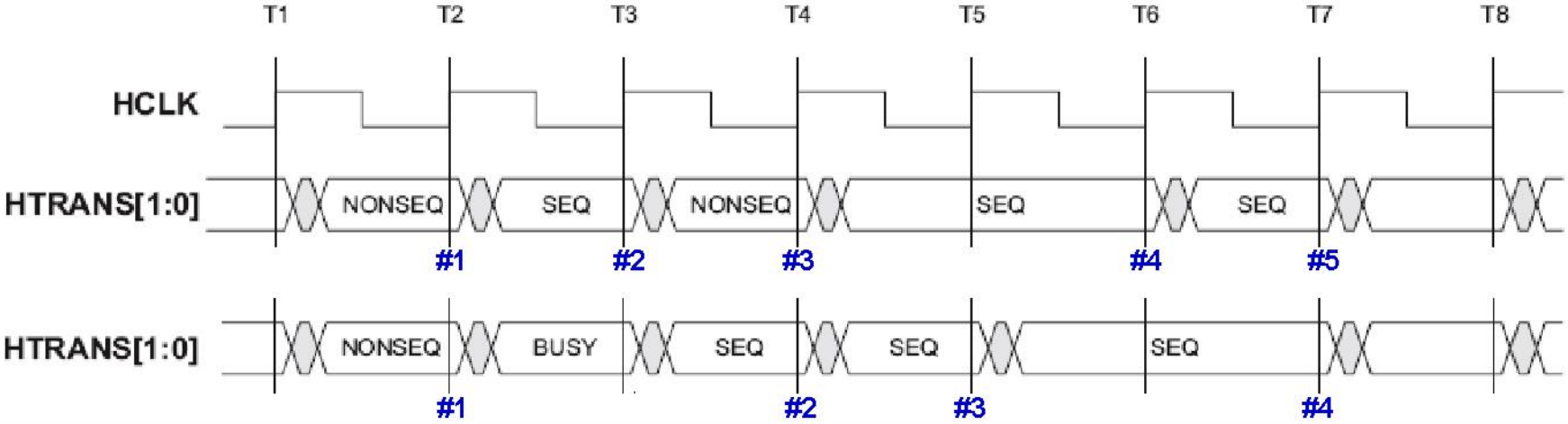
32-bit data bus

HSIZE[2]	HSIZE[1]	HSIZE[0]	Size	Description
0	0	0	8 bits	Byte
0	0	1	16 bits	Halfword
0	1	0	32 bits	Word
0	1	1	64 bits	-
1	0	0	128 bits	4-word line
1	0	1	256 bits	8-word line
1	1	0	512 bits	-
1	1	1	1024 bits	-

Control Signal: Transfer Type

- Every transfer can be classified into one of four different types, as indicated by the **HTRANS[1:0]** signals.

HTRNAS[1:0]	Type	Description
2'b00	IDLE	IDLE transfer is used when a master is granted but does not wish to perform a data transfer, or master idle(done).
2'b01	BUSY	Is used to insert NOP cycles in the middle of a burst transfer
2'b10	NONSEQ	Indicates the first transfer of a burst or a single transfer(start)
2'b11	SEQ	Indicates the remainng transfer of a burst



Control Signal: Burst Operation

❑ Burst Length

- Fixed-length
  - ✓ single
  - ✓ 4, 8, and 16 beat bursts
- undefined-length

❑ Incrementing(4, 8, 6, X) and wrapping(4, 8, 6) bursts are supported

- **Incrementing**
  - ✓ access sequential locations and increment of the previous address.
  - ✓ Ex. four-beat incrementing burst of word (4-byte) accesses
    - – start address 0x34 -> 0x38 -> 0x3C -> 0x40

0x3C('b0011_1100) + 0x4
= 0x40 ('b0100_0000) @INC
= 0x30 ('b0011_0000) @WRAP

- **Wrapping**
  - ✓ address wrap when the boundary is reached.
  - ✓ Ex. four-beat wrapping burst of word (4-byte) accesses
    - – four-beat X word = 16-byte boundaries. (5'b1_0000)
    - – start address 0x34 -> 0x38 -> 0x3C('b11_1100) -> 0x30 ('b11_0000).

Burst Signal Encoding

- **Burst must be aligned to the address boundary equal to the size (HSIZE[2:0]) of the transfer.**
  - Ex. HSIZE[2:0] = word -> HADDR[1:0] = 2'b00, (0, 4, 8, …)
    HSIZE[2:0] = half-word -> HADDR[0] = 1'b0, (0, 2, 4, …)

- **Bursts must not cross a 1kB address boundary.**
  - masters do not attempt to start a fixed-length incrementing burst which would cause this boundary

    (11'b1xx_xxxx_xxxx, x=don't' care)

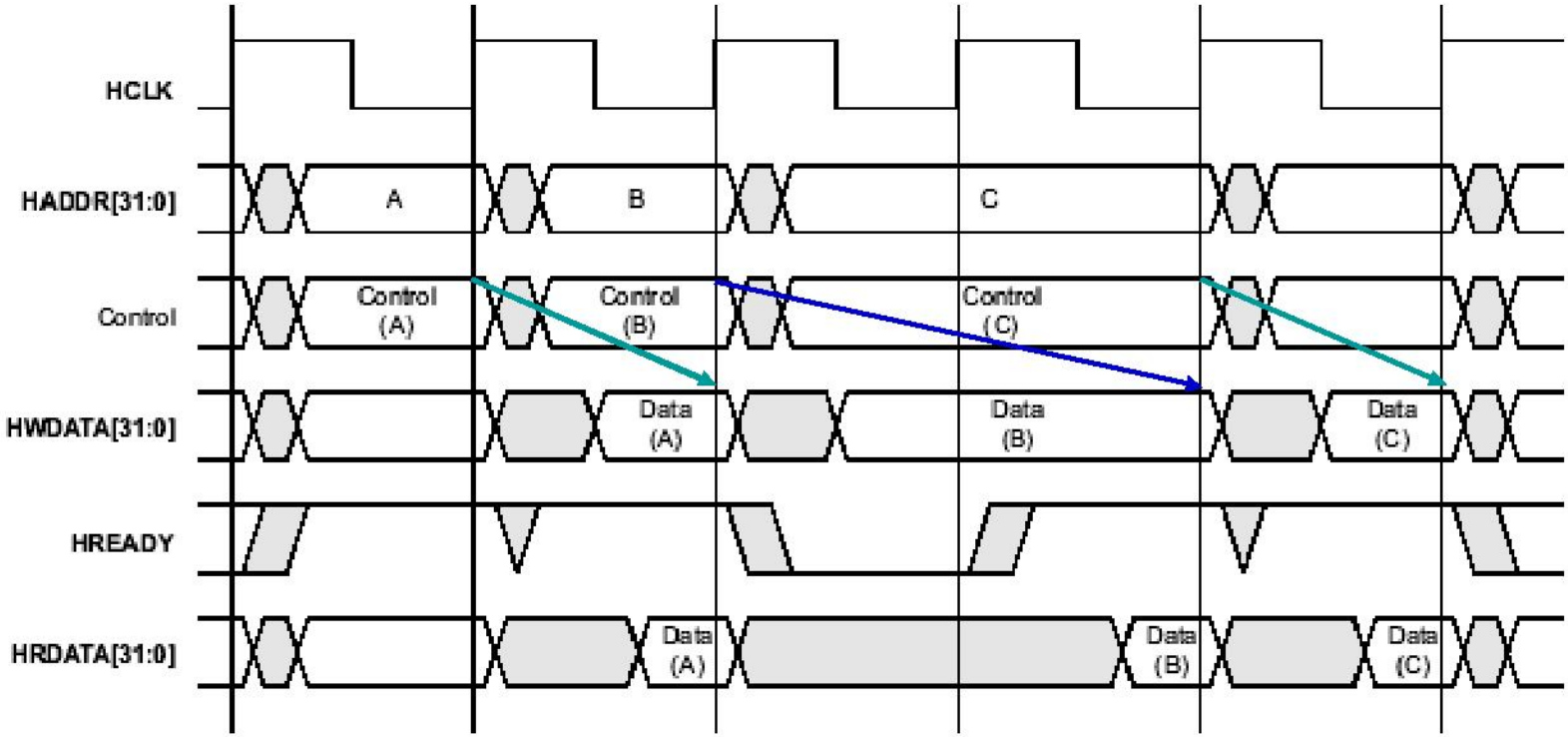
HBURST[2:0]	Type	Description
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing burst

Early Burst Termination

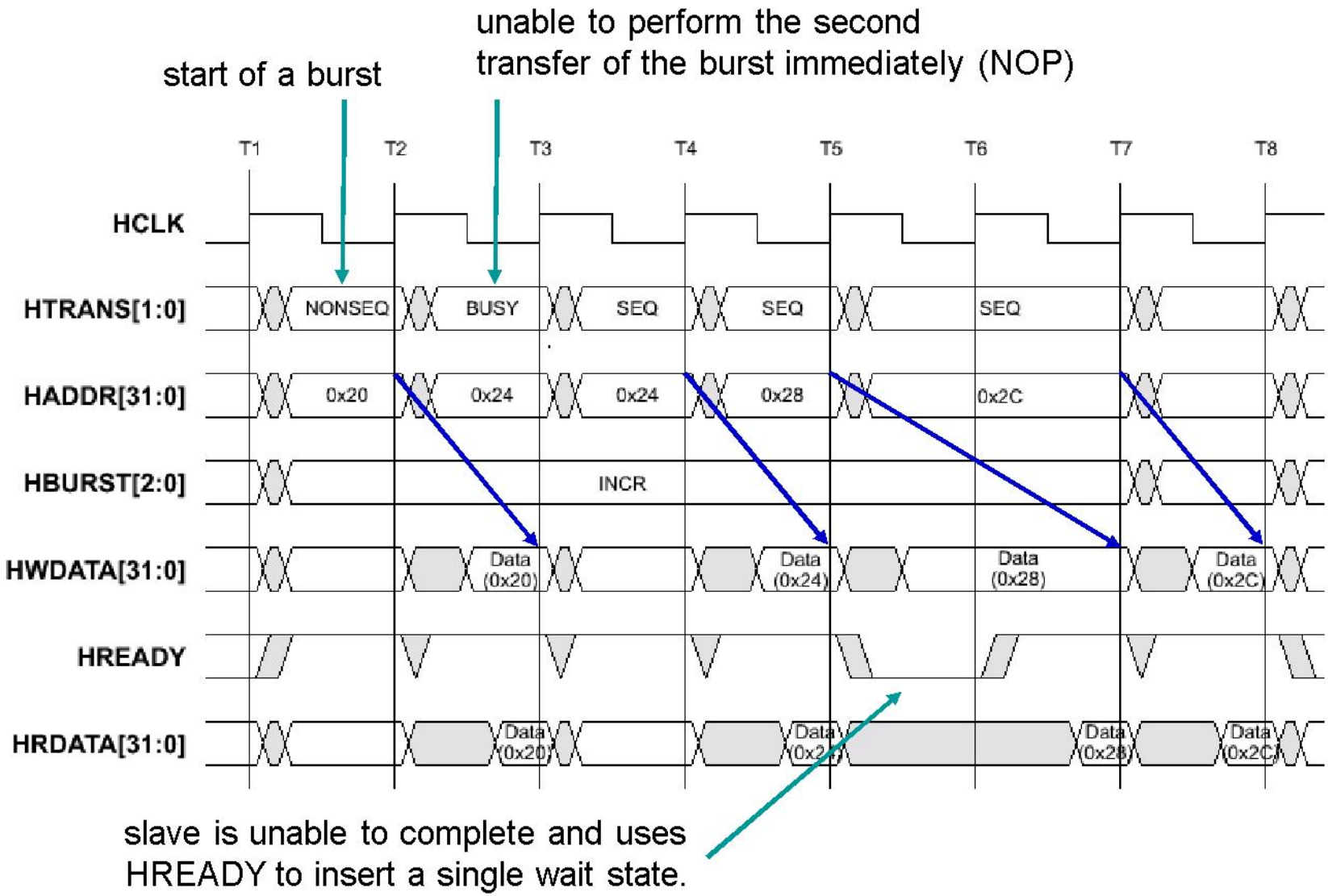
- **When a burst will not be allowed to complete (HGRANTx to low),**
  - **The slave**
    - ✓ can determine when a burst has terminated early by monitoring the HTRANS signals and ensuring that after the start of the burst every transfer is labelled as SEQUENTIAL or BUSY.
    - ✓ If a NONSEQUENTIAL or IDLE transfer occurs then this indicates that a new burst has started and therefore the previous one must have been terminated.
  - **The master**
    - ✓ it must rebuild the burst appropriately when it next gains access to the bus.
    - ✓ For example, if a master has only completed one beat of a four-beat burst then it must use an undefined-length burst to perform the remaining three transfers.

Multiple Transfers (Pipelined)

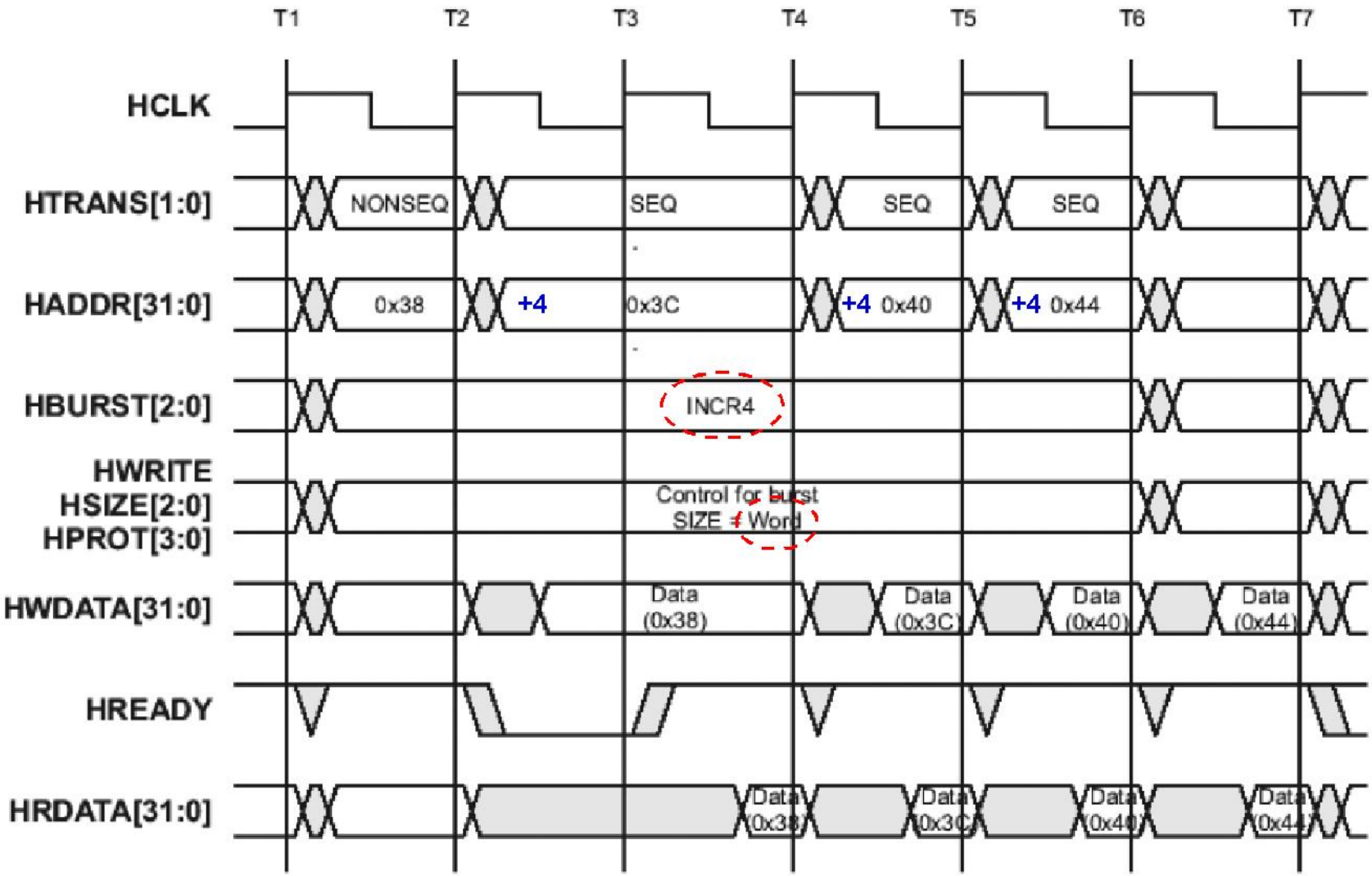
- **zero wait state**
  - transfers to addresses A and C
- **one wait state**
  - transfer to address B



Burst Transfer (BUSY, HREADY)

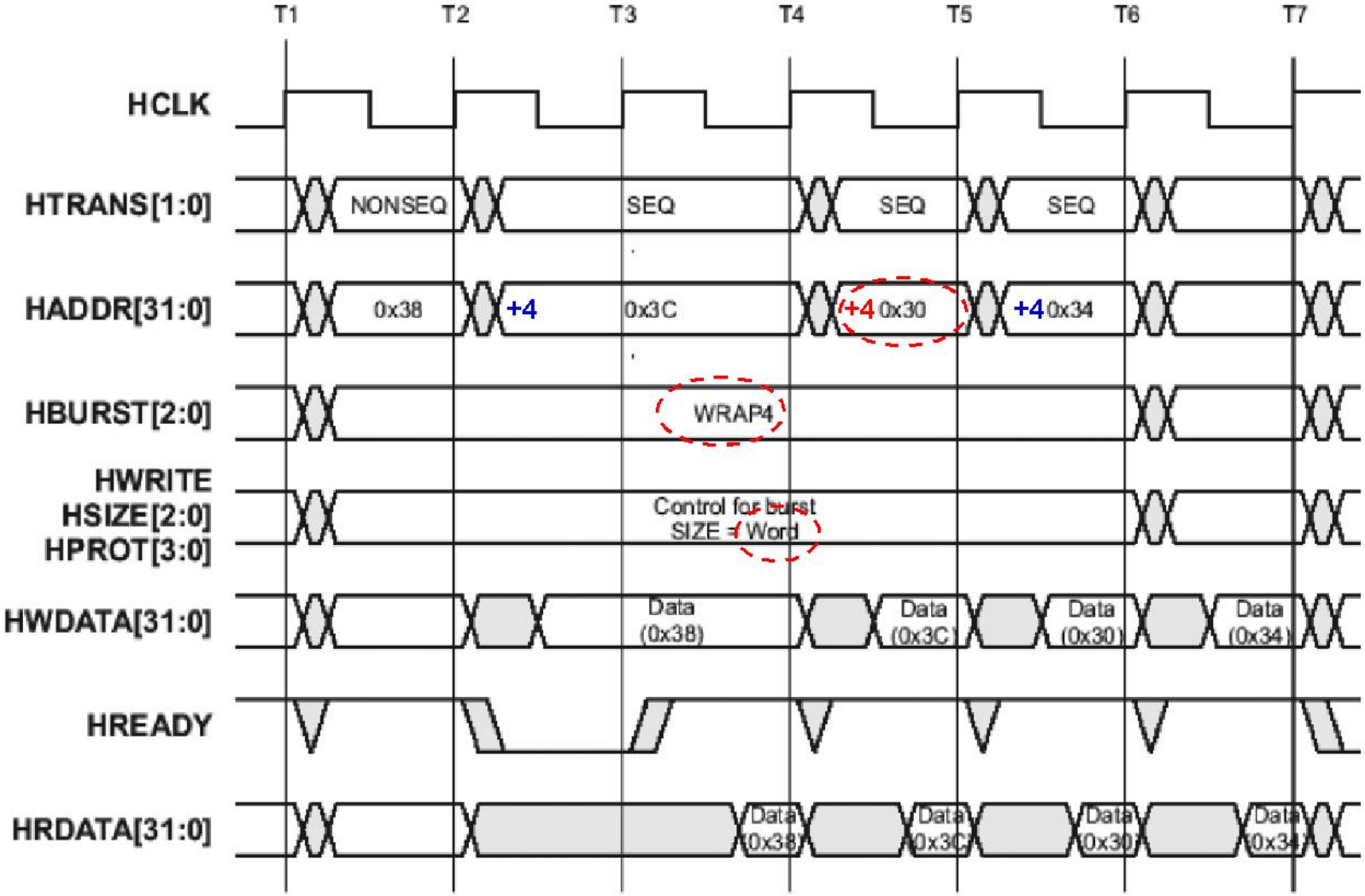


Burst Transfer (4-beat INCR)



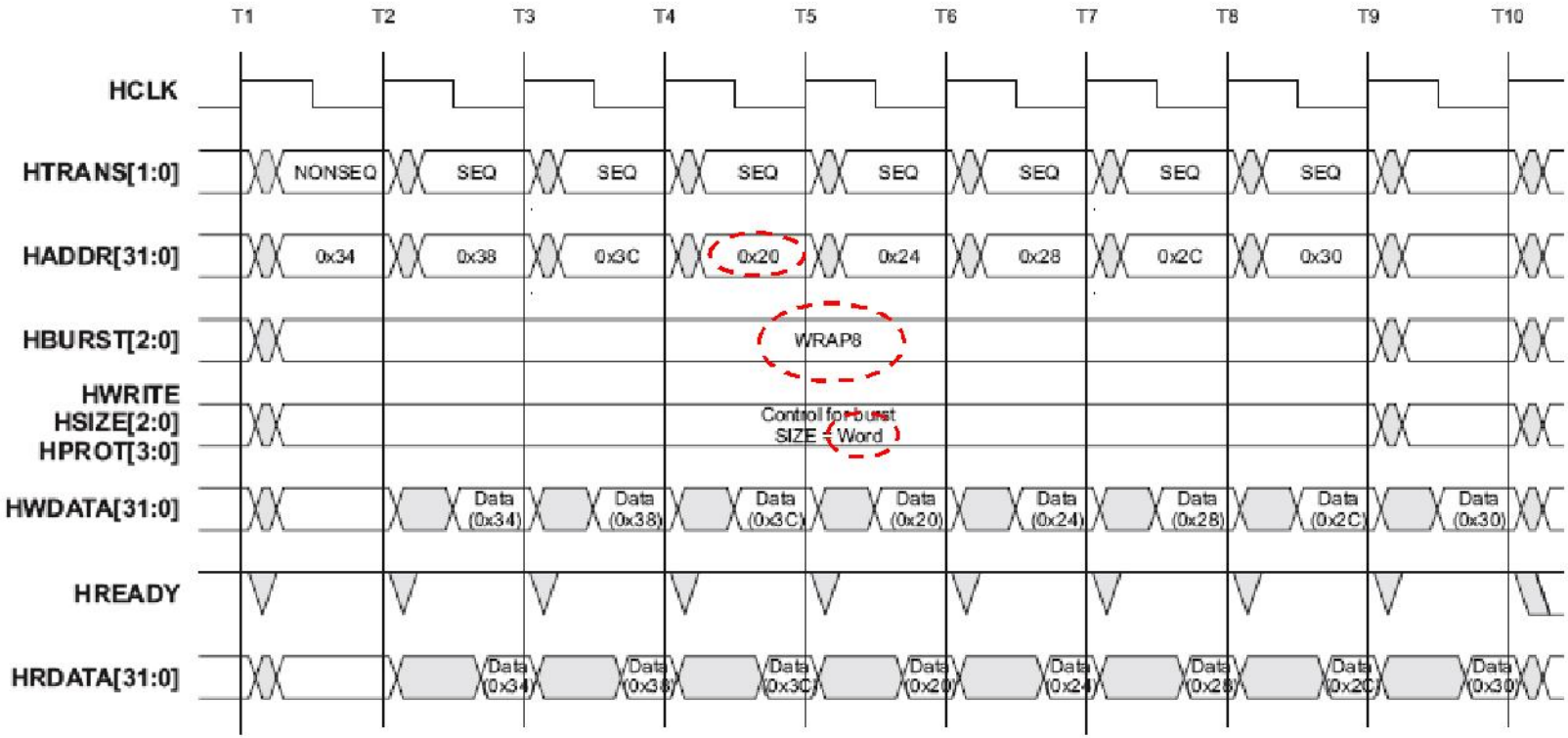
Burst Transfer (4-beat WRAP)

Address will wrap at 16-byte (4-beat X 4 byte) boundaries: **5'b1_0000**



Burst Transfer (8-beat WRAP)

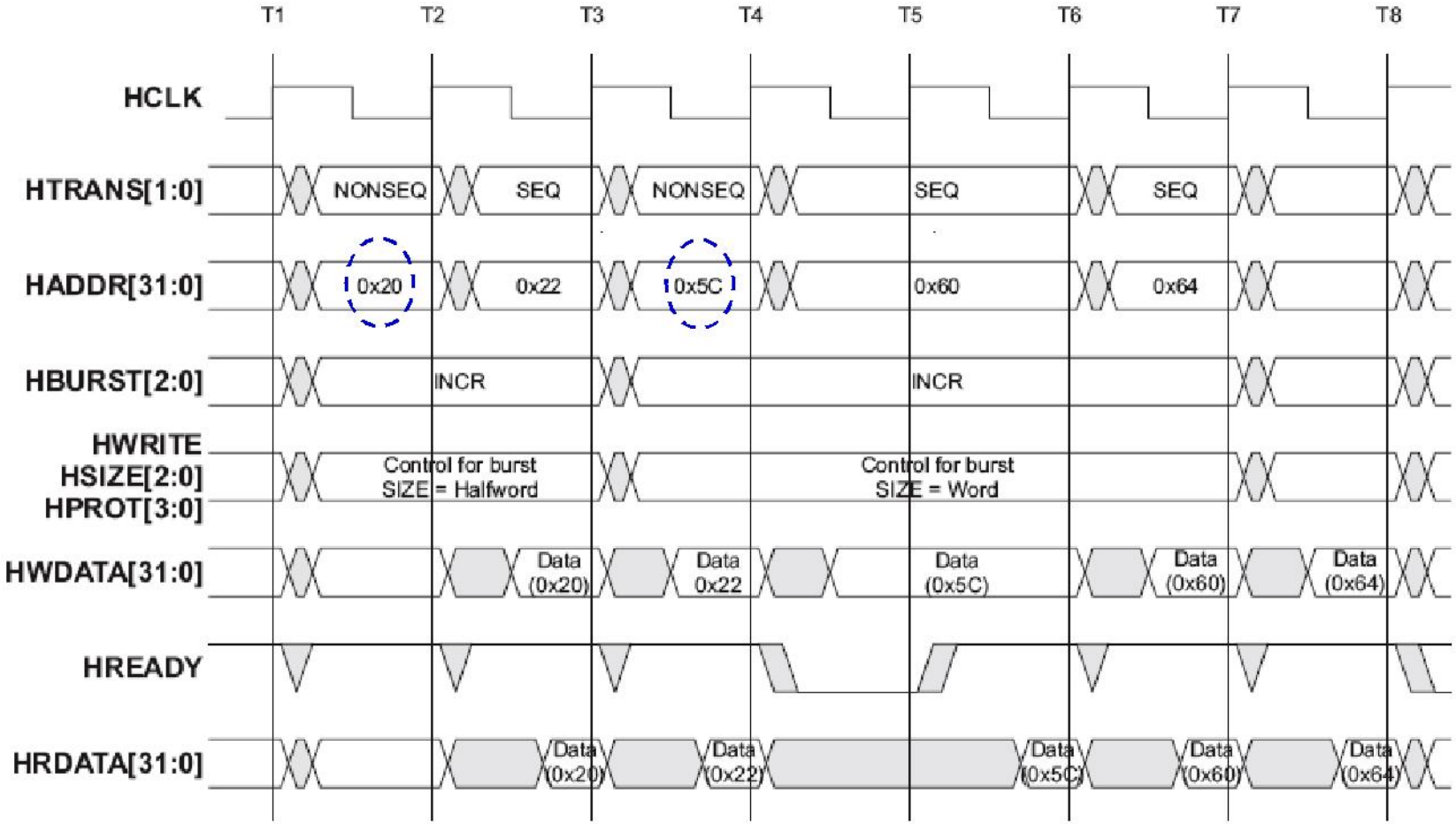
Address will wrap at 32-byte (8-beat X 4 byte) boundaries: **6'b10_0000**



0x3C('b0011_1100) + 0x4 = 0x40 ('b0100_0000) -> 0x20('b0010_0000)

Burst Transfer (Undefined-length bursts)

- Two halfword transfers starting at address 0x20.
- Three word transfers starting at address 0x5C.



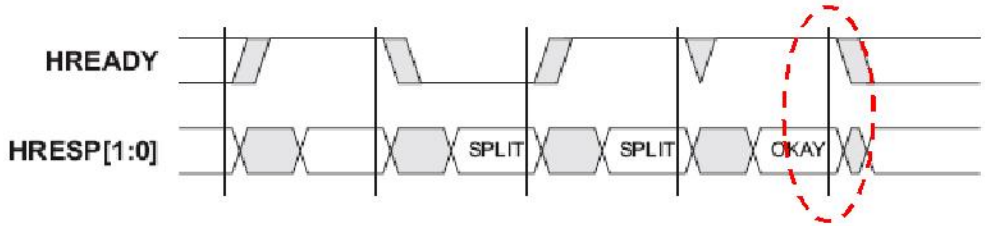
Slave Transfer Responses

- **After a master has started a transfer, the slave then determines how the transfer should progress.**

  - Transfer done, slave latency

HREADY	Description
1'b1	Transfer done
1'b0	Transfer extended

transfer completes



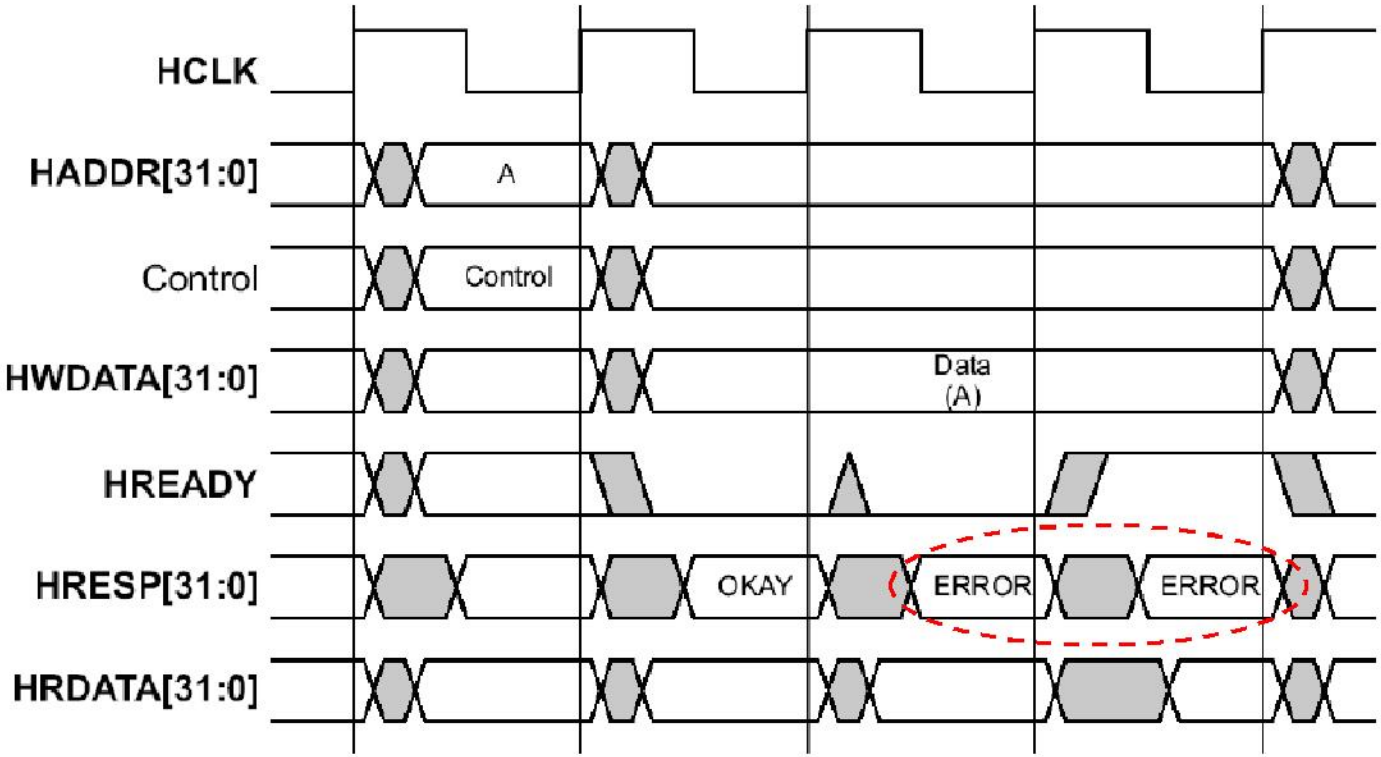
  - Transfer response

HRESP[1:0]	Response	Description
2'b00	OK	the transfer is progressing normally
2'b01	ERROR	a transfer error has occurred
2'b10	RETRY	The master should retry the transfer until it completes
2'b11	SPLIT	The master should retry the transfer when it is next granted

Two-cycle response

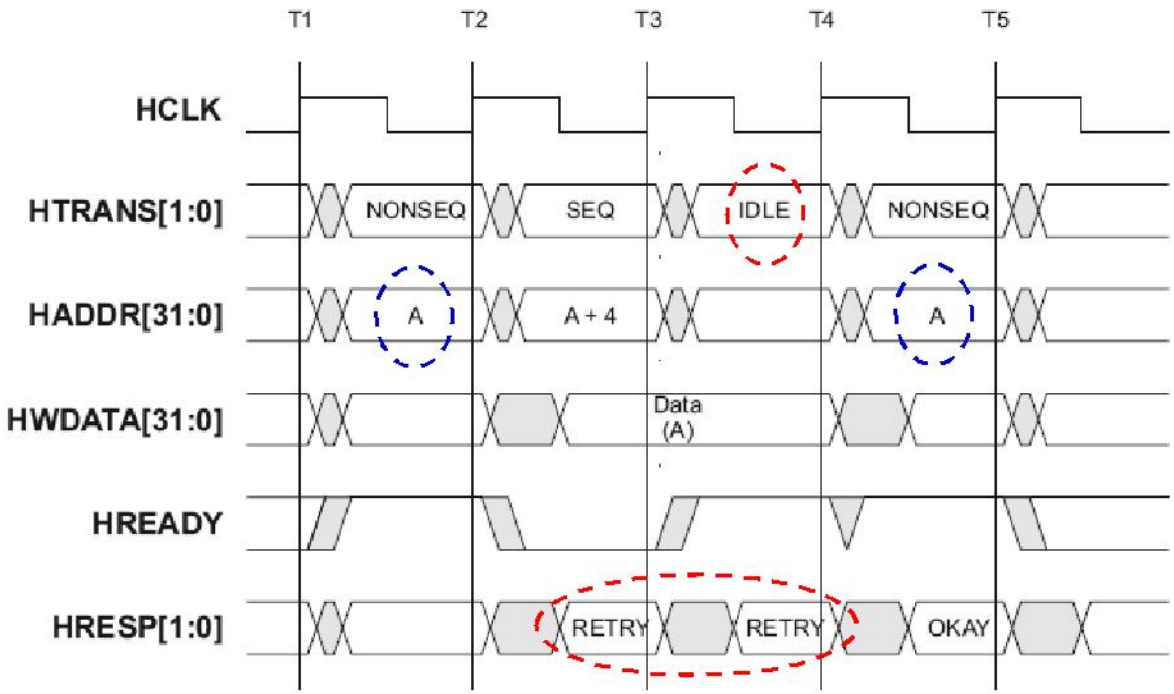
AHB Slave ERROR Response

- **The ERROR response is used by a slave to indicate some form of error condition with the associated transfer.**
  - Ex. protection error
    - ✓ Attempt to write to a read-only memory location.



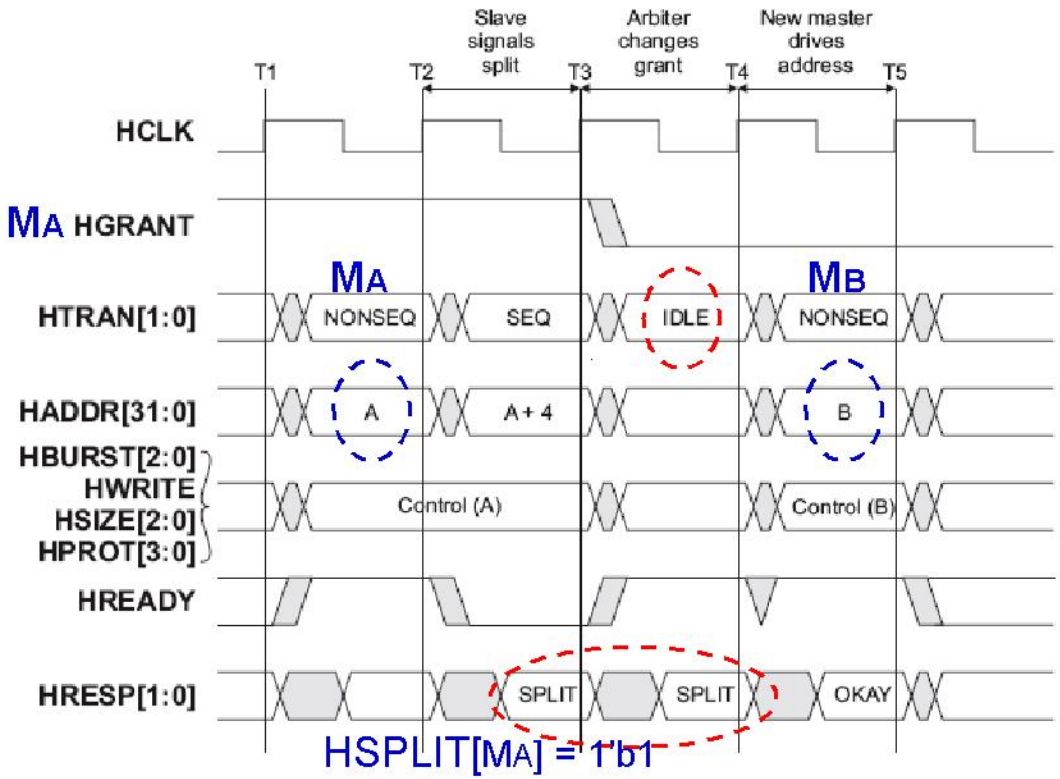
AHB Slave RETRY Response

- **The RETRY response shows the transfer has not yet completed.**
  - The bus master
    - ✓ performs an IDLE transfer immediately after receiving a RETRY or SPLIT.
    - ✓ continue to retry the transfer until it completes.



AHB Slave SPLIT Response

- **The transfer has not yet completed successfully.**
  - Bus handover after a SPLIT
  - The bus master
    - ✓ retry the transfer when it is next granted access to the bus.



T3: Slave stores master A ID

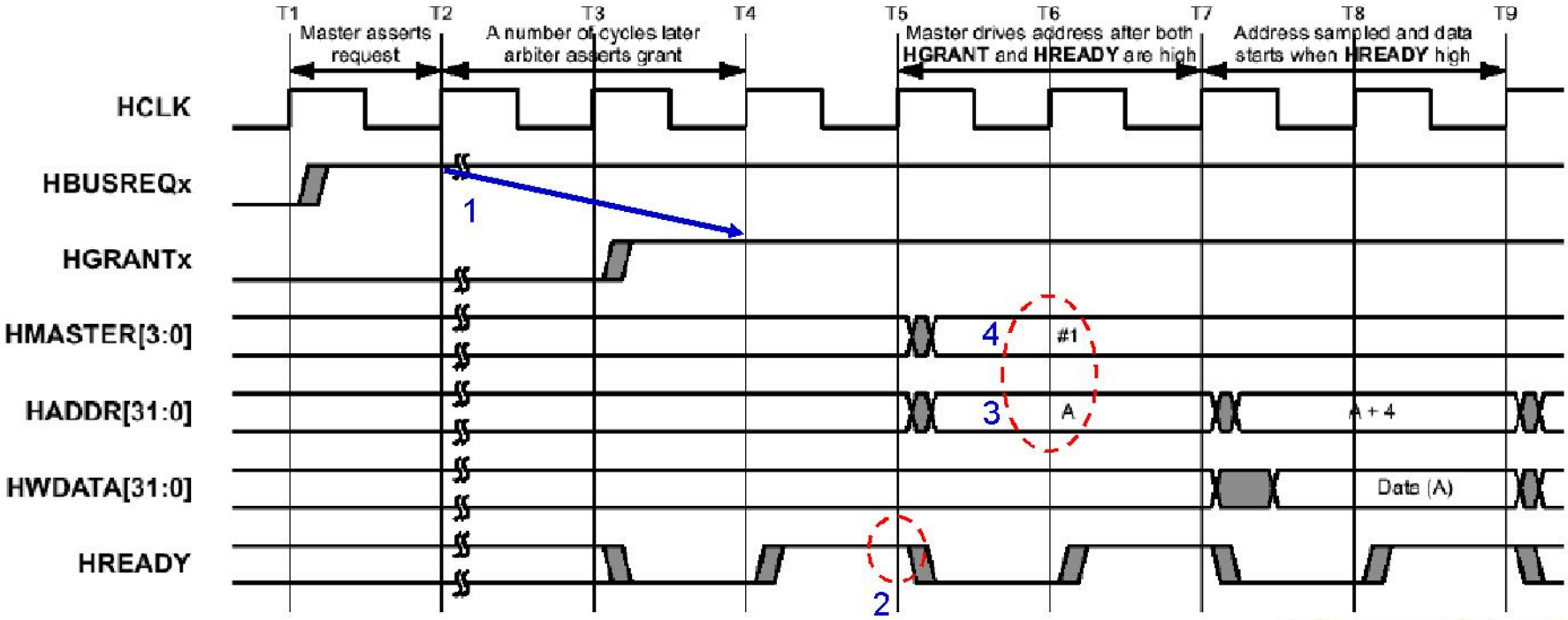
T4: Bus handover after a split

T5: Master A is masked

AHB Master Granting Access

- Arbiter : 1. Master (HBUSREQx) -> HGRANTx
- AHB Slave : 2. HREADY HIGH
- AHB Master : 3. Start Transfer
- Arbiter : 4. change the HMASTER[3:0]

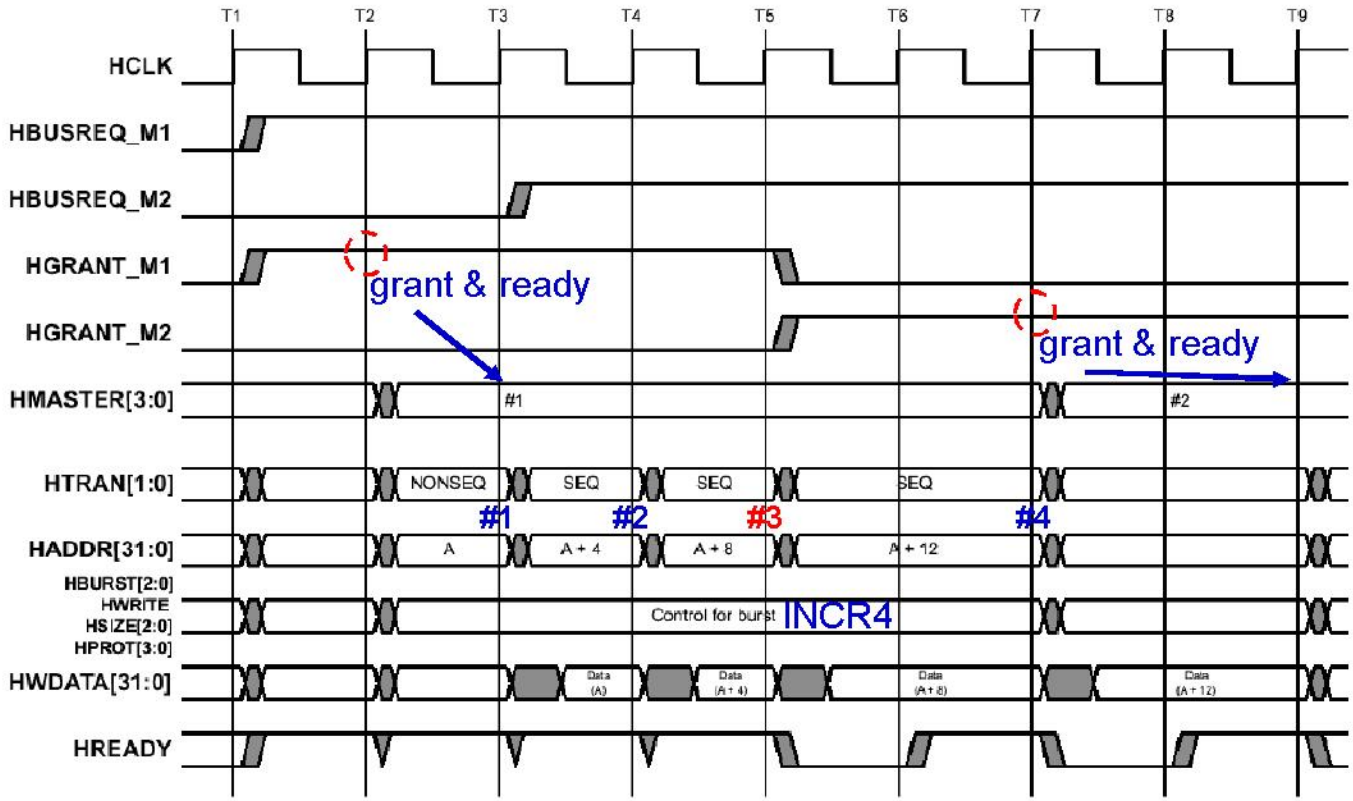
Cost of arbitration (bus access latency)



data access latency

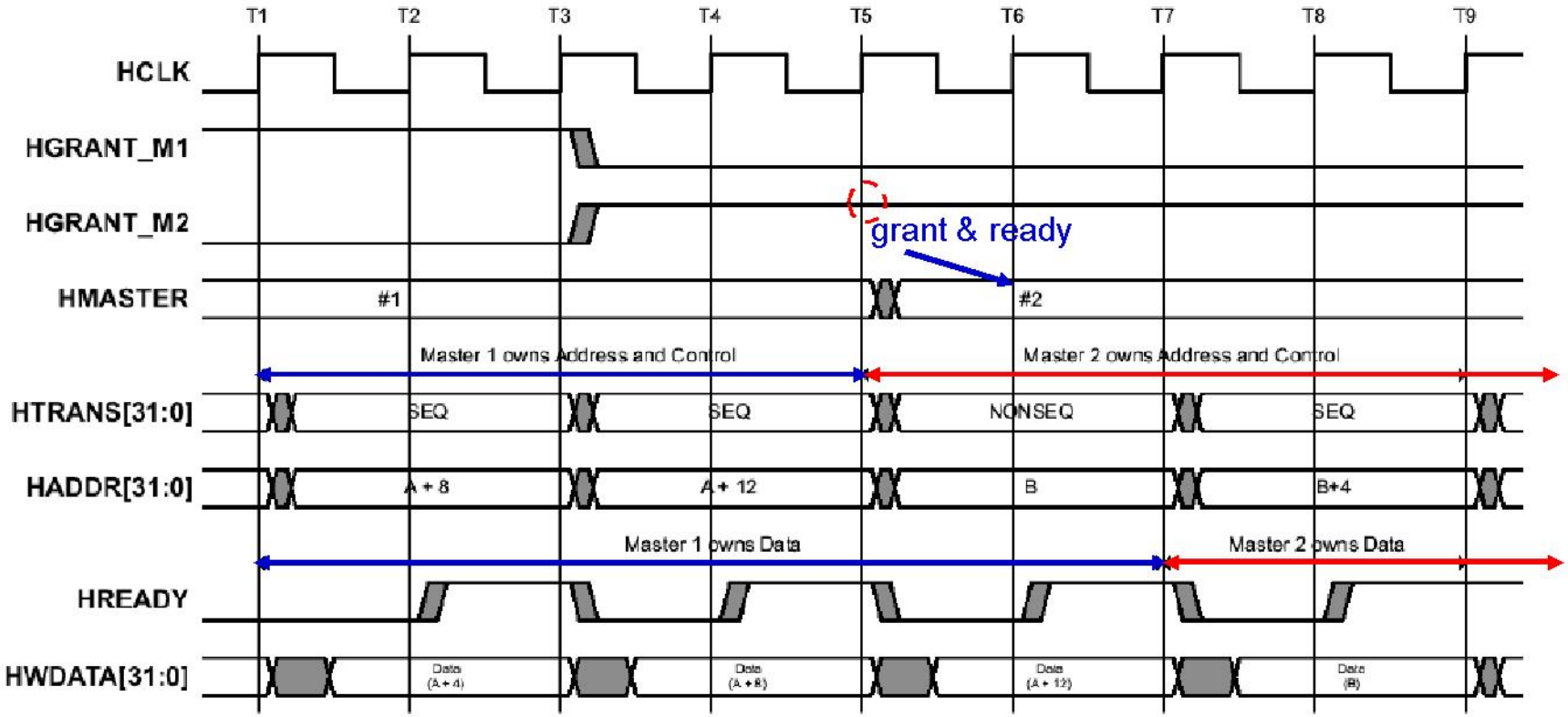
AHB Master Handover after Burst

- Figure shows an example of how the arbiter can hand over the bus at the end of a burst of transfers.
  - The arbiter changes the HGRANTx signals when the penultimate (one before last) address has been sampled.



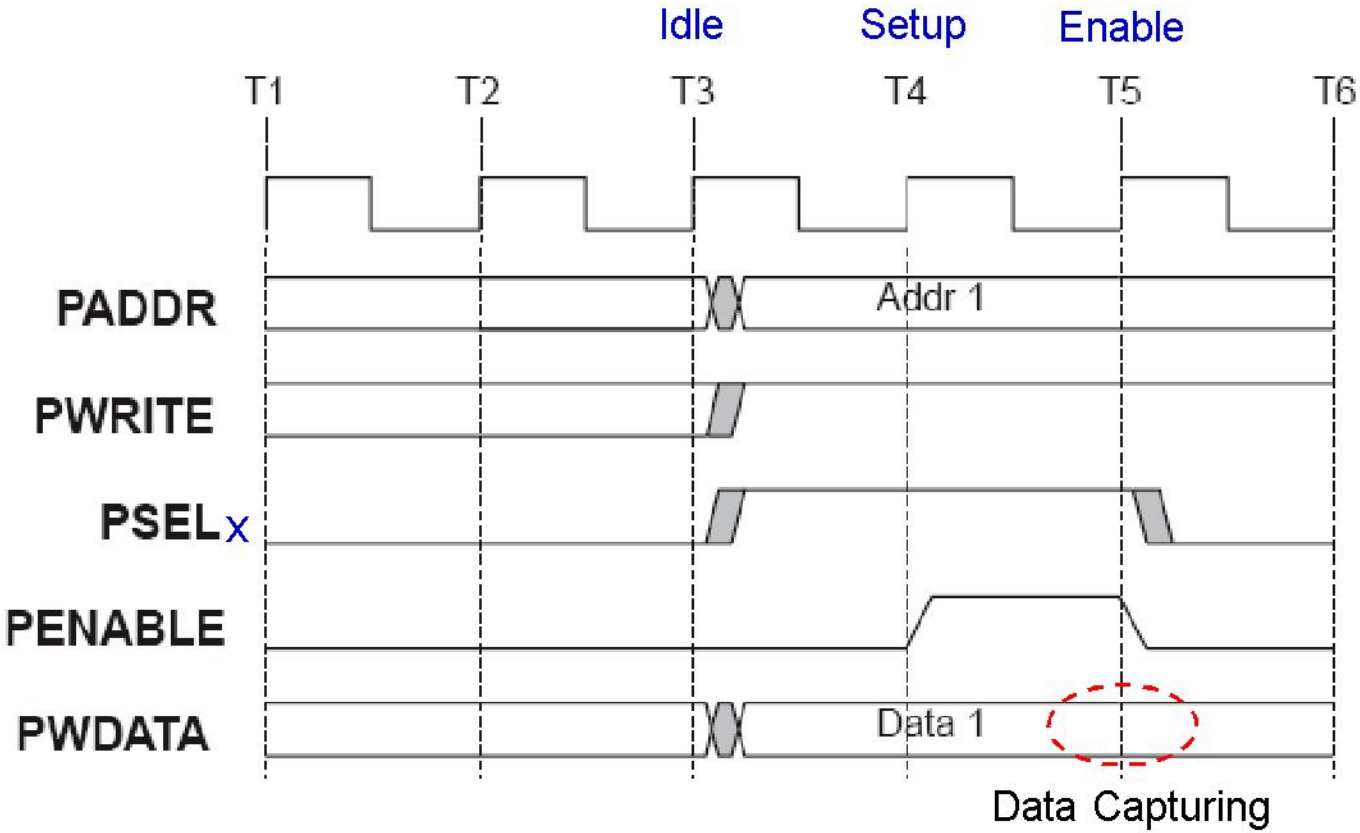
AHB Master Data Bus Ownership

- Figure shows how the ownership of the data bus is transferred when handover occurs between two bus masters.
  - The ownership of the data bus is delayed from the ownership of the address bus.

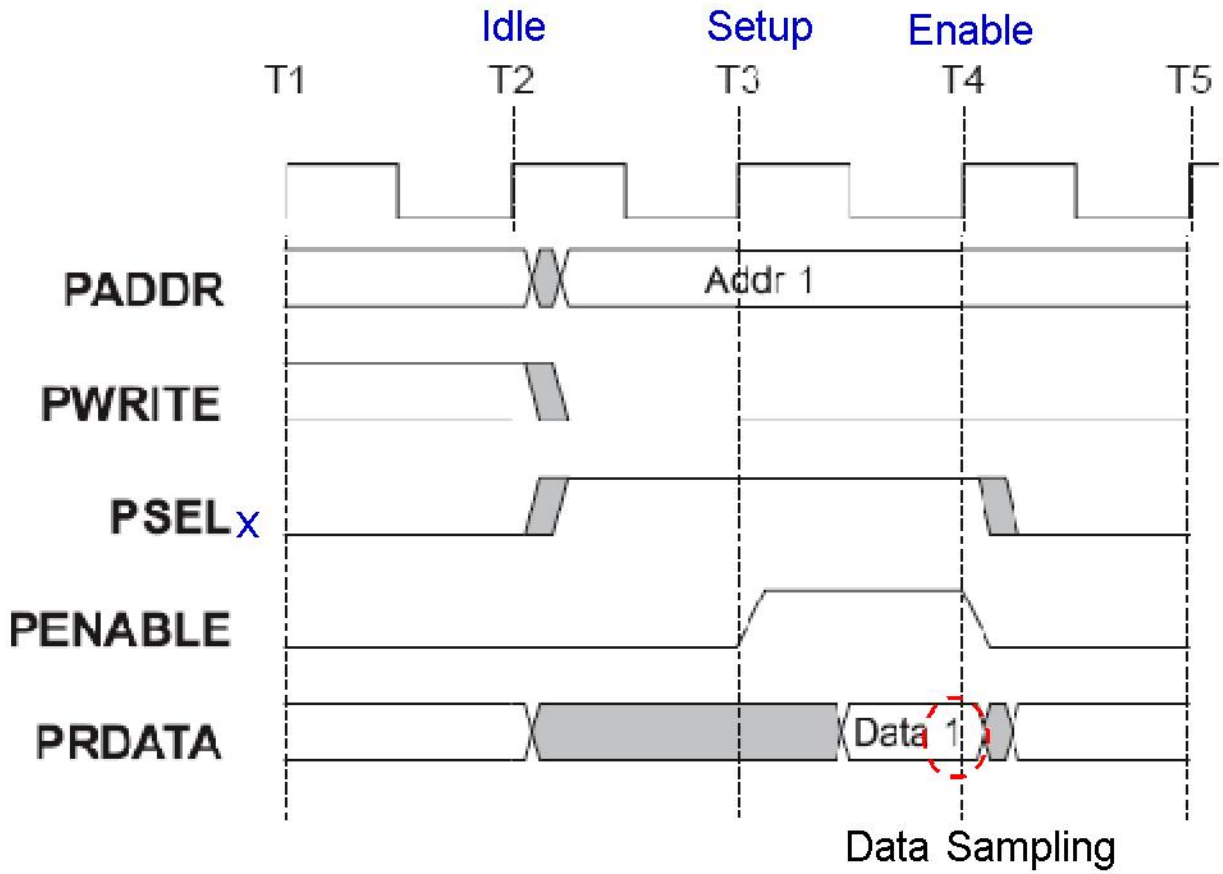


APB Write Transfer

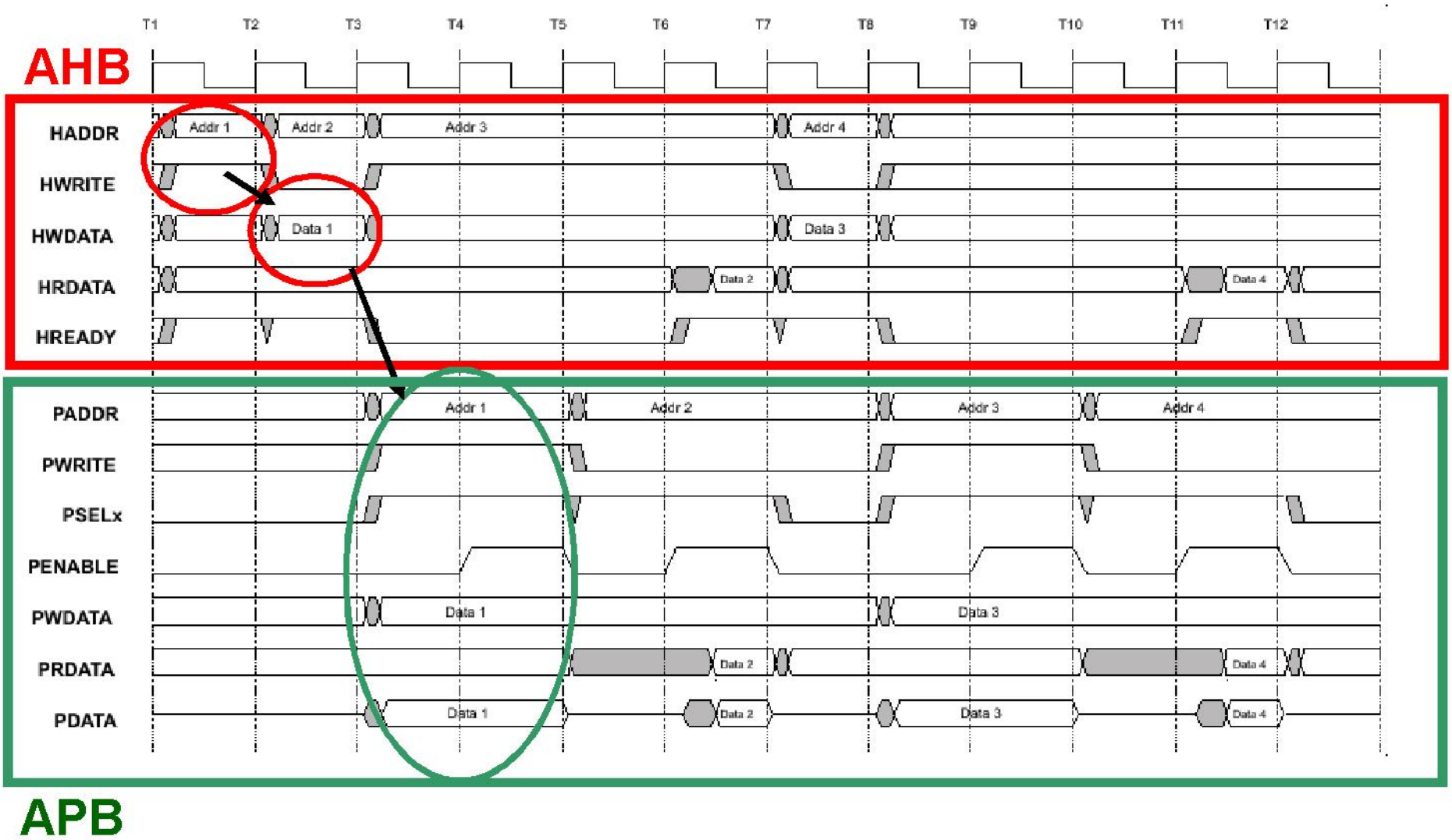
- For low power, after the end of the enable cycle addresses and control signals do not change.



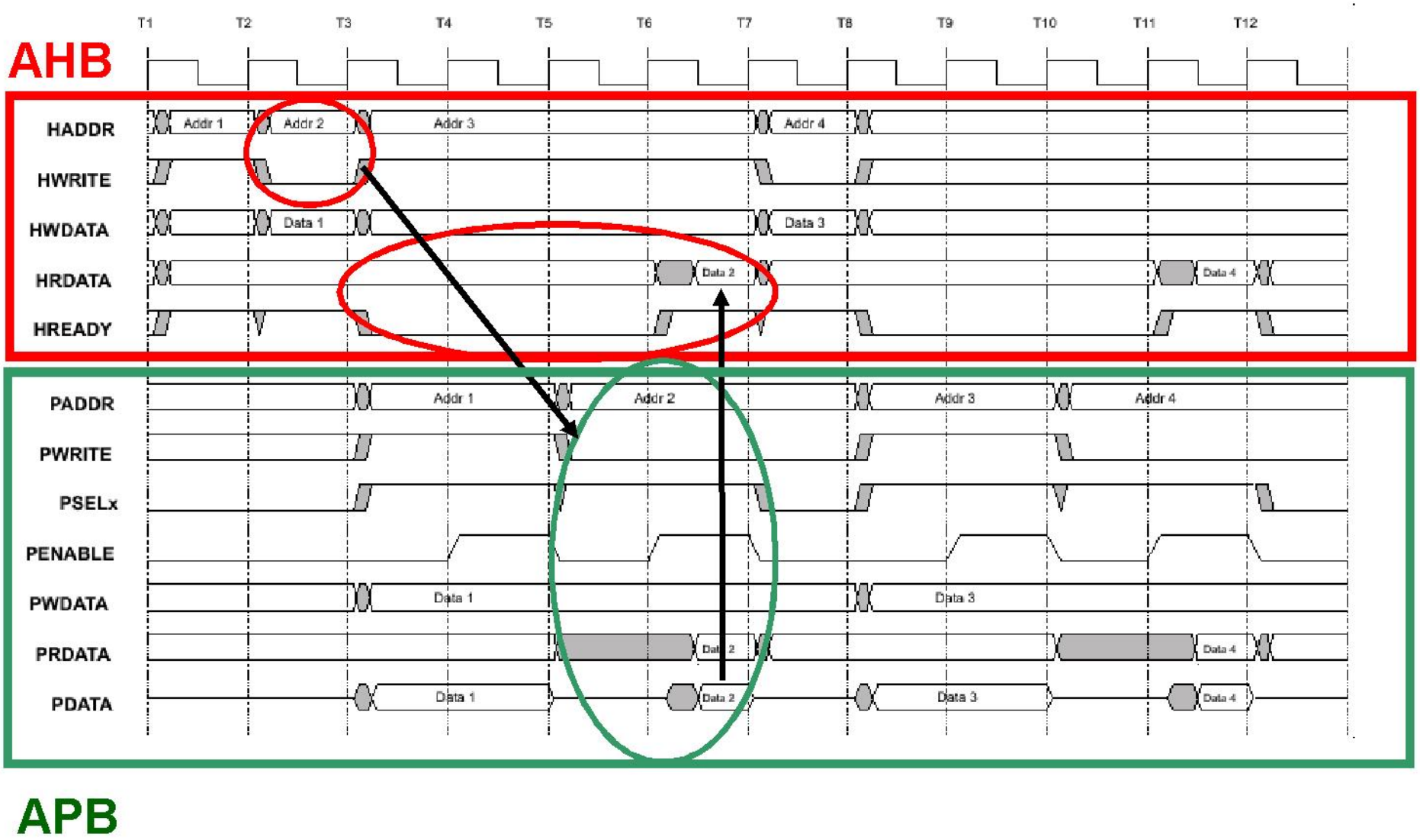
APB Read Transfer



Ex. AHB to APB Transfer (Write)

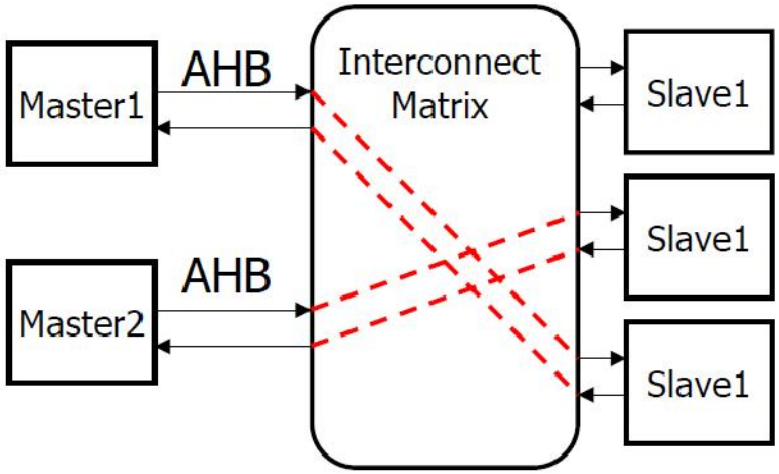


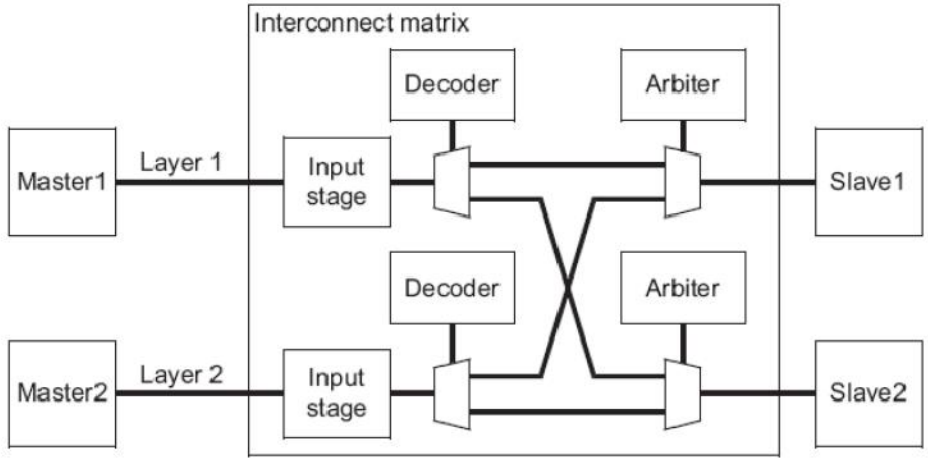
Ex. AHB to APB Transfer (Read)

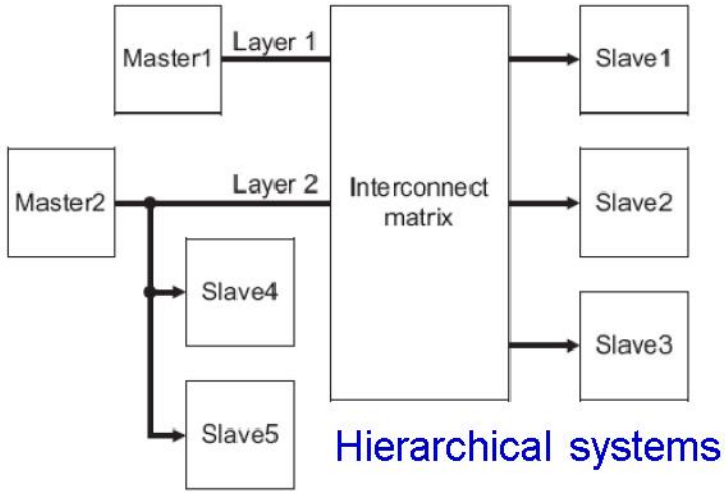


AMBA Multi-layer AHB

- Enables parallel access paths between multiple masters and slaves
- Fully compatible with AHB wrappers





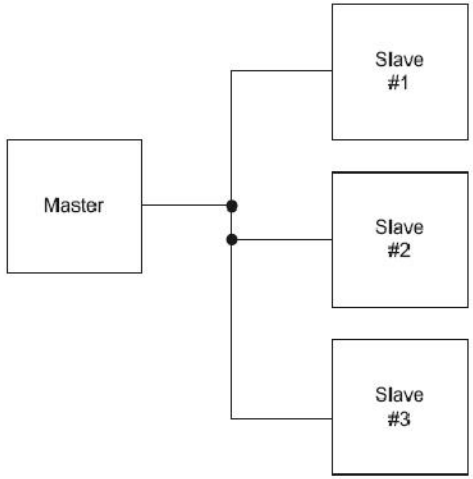


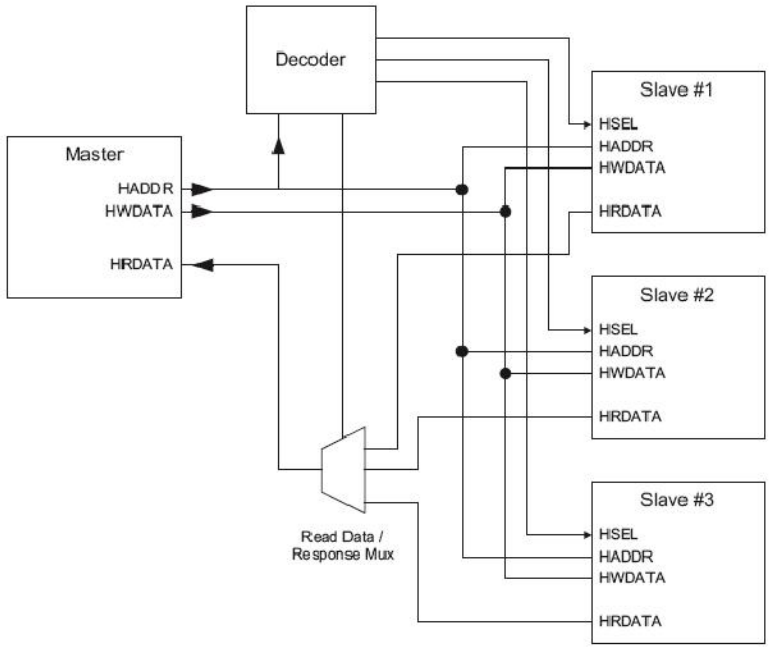
Hierarchical systems

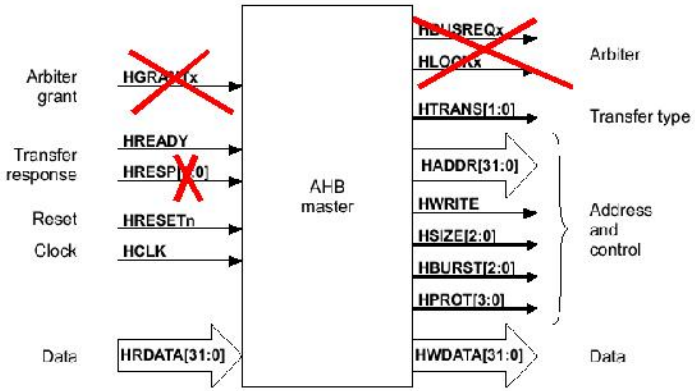
AHB-LITE

- Where no arbitration is required between multiple masters, significant simplifications to the AHB interface can be made, resulting in the AHB-Lite interface.

- AHB-Lite simplifies the AHB specification by removing the protocol
  - request/grant protocol to the arbiter and
  - split/retry responses from slaves.







What is the difference between a default and a dummy master?

❑ Default bus master

- Default master that is granted when none of the masters in the system are requesting access to the bus.
- 1-hot protocol

❑ Dummy bus master

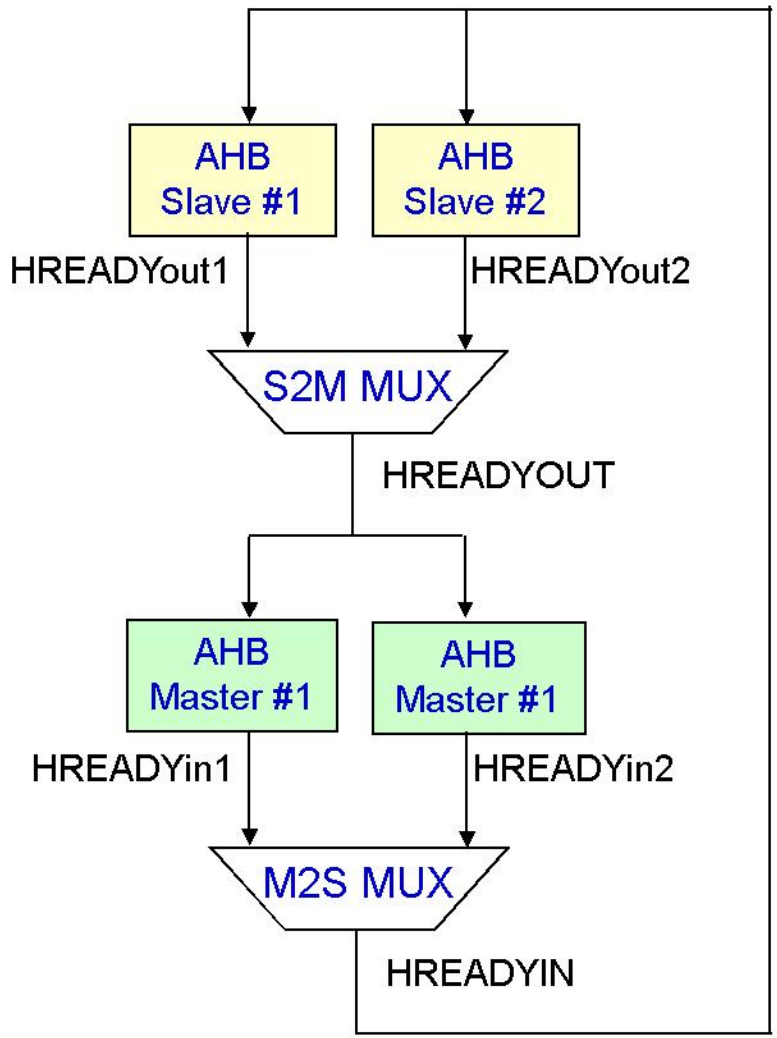
- The dummy bus master is a master which only performs IDLE transfers.
- It is required in a system so the arbiter can grant a master which is guaranteed not to perform any real transfers.
- The two cases,
  - ✓ Split response is given to a locked transfer and
  - ✓ Split response is given and all other masters have already been split.

Why is a burst not allowed to cross a 1 Kbyte boundary?

- HSELx = HIGH
  - AHB slave start of a burst transaction
- 1 kB is the smallest area an AHB slave may occupy in the memory map.
  - if a burst did cross a 1 kB boundary, the access could start accessing one slave at the beginning of the burst and then switch to another on the boundary, which must not happen for the above reason.
- The 1 kB boundary has been chosen as it is large enough to allow reasonable length bursts.
  - but small enough that peripherals can be aligned to the 1 kB boundary without using up too much of the available memory map.

Slaves require HREADY as both an input/output signal, why?

- The output of this S2M multiplexer is the global HREADY signal which is routed to all masters on the AHB and is also fed back to all slaves as the HREADY input.
- When the slave is responsible for driving HREADY (when it is being accessed) then this signal will be the same as the HREADY output from the slave.
- However, when another slave is being accessed, the HREADY signal will indicate if that slave is inserting wait states and this is important when a master changes from one slave to another.



What is the different between PSELx and PENABLE signals?

❑ PSELx (APB select)

- A signal from the secondary decoder, within the peripheral bus bridge unit, to each peripheral bus slave x.
- This signal indicates that the slave device is selected and a data transfer is required.
- There is a PSELx signal for each bus slave.

❑ PENABLE (APB strobe)

- This strobe signal is used to time all accesses on the peripheral bus.
- The enable signal is used to indicate the second cycle of an APB transfer.
- The rising edge of PENABLE occurs in the middle of the APB transfer.

When should a master de-assert its HBUSREQ signal?

- **For an undefined length burst (INCR) a master**
  - must keep its HBUSREQx signal asserted until it has started the address phase of the last transfer in the burst.
- **For a defined length burst the master**
  - can deassert the HBUSREQx signal once the master has been granted the bus for the first transfer.
  - This can be done because the arbiter is able to count the transfers in the burst and keep the master granted until the burst completes.
- **However it is not a mandatory requirement for an Arbiter to allow a burst to complete.**
  - the master will have to re-assert HBUSREQx if the Arbiter removes HGRANT before the burst has been completed (a so-called early burst termination, next slide).

When can Early Burst Termination occur?

- Bursts can be early terminated either as a result of the Arbiter removing the HGRANT to a master part way through a burst, or after a slave returns a non-OKAY response to any beat of a burst.

- Note however that a master cannot decide to terminate a defined length burst unless prompted to do so by the Arbiter or Slave responses.

- All AHB Masters, Slaves and Arbiters must be designed to support Early Burst Termination.