

# ARM System on Chip

---

A processor based System-on-Chip will typically include:

- Hardware

- A processor core (e.g. ARM M0 DesignStart)
- Interconnect (e.g. AHB-Lite bus)
- Fixed program memory (ROM)
- Data memory (RAM)
- Application specific interfaces (for input and output)

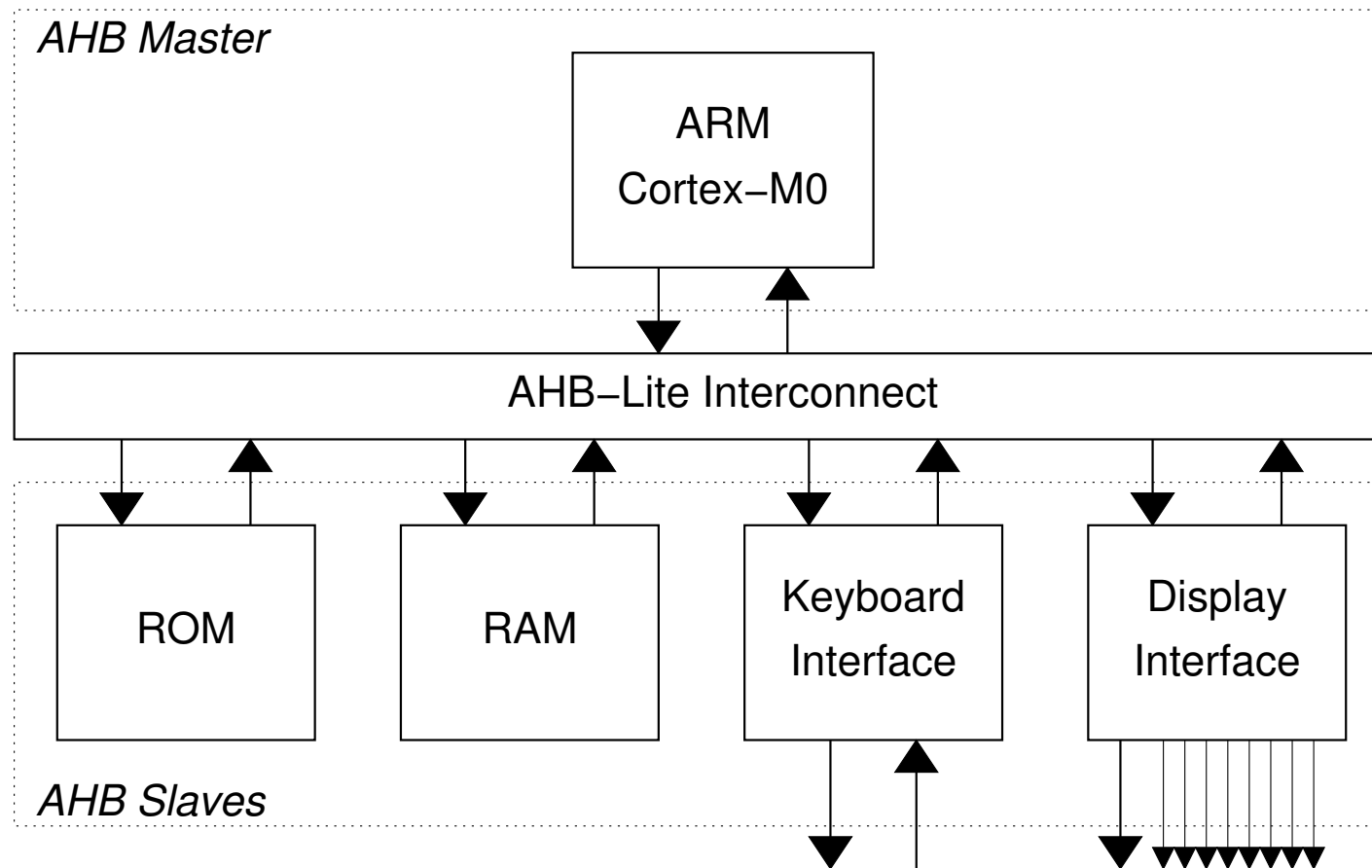
- Software

- Application specific software (typically written in C)

# ARM System on Chip

---

## Hardware



# ARM System on Chip

---

## Use of Third-Party Intellectual Property

The use of hardware and software modules created by others allows designers to build larger and more complex systems.

You may make use of third-party intellectual property in your design provided that:

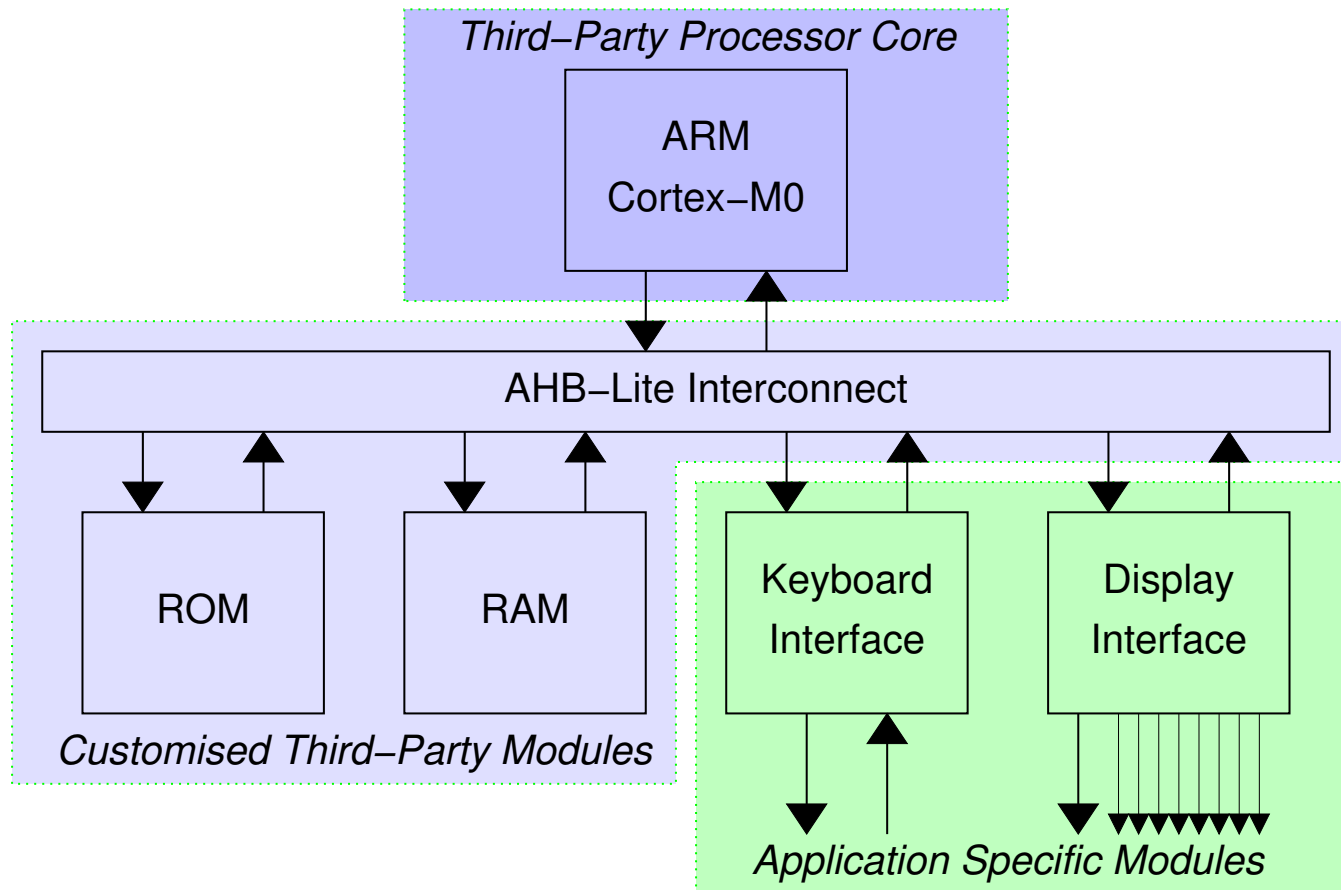
1. You have the permission of the designer
2. You acknowledge its use in your report
3. You do not remove copyright/licensing notices

*In a simple design you might expect to use a third-party processor core together with customised versions of third-party interconnect, ROM and RAM modules*

# ARM System on Chip

---

## Use of Third-Party Intellectual Property



# ARM System on Chip

---

## ARM Cortex-M0 DesignStart

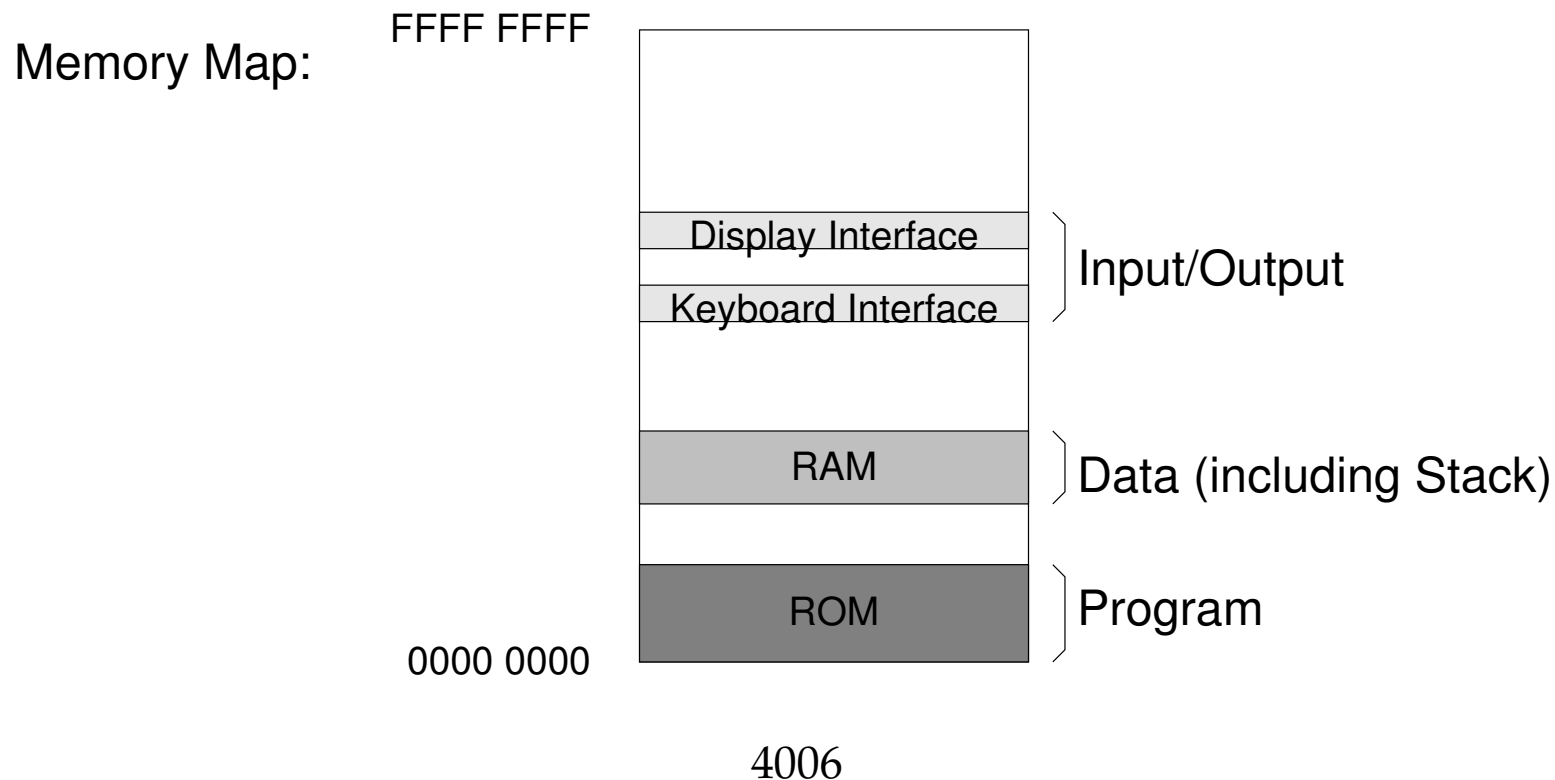
- Simple 32-bit ARM processor core  
Designed for embedded applications.
- Executes 16-bit instructions from the ARM Thumb2 instruction set  
Supported by standard GNU compilers.
- Obfuscated Verilog Source Code  
Designed to be usable but not to be reverse engineered or customised.
- Available free for academic and other non-commercial use  
You may use the provided core for this project. If you wish the Cortex-M0 DesignStart for another purpose, you should apply to ARM for permission.

# ARM System on Chip

---

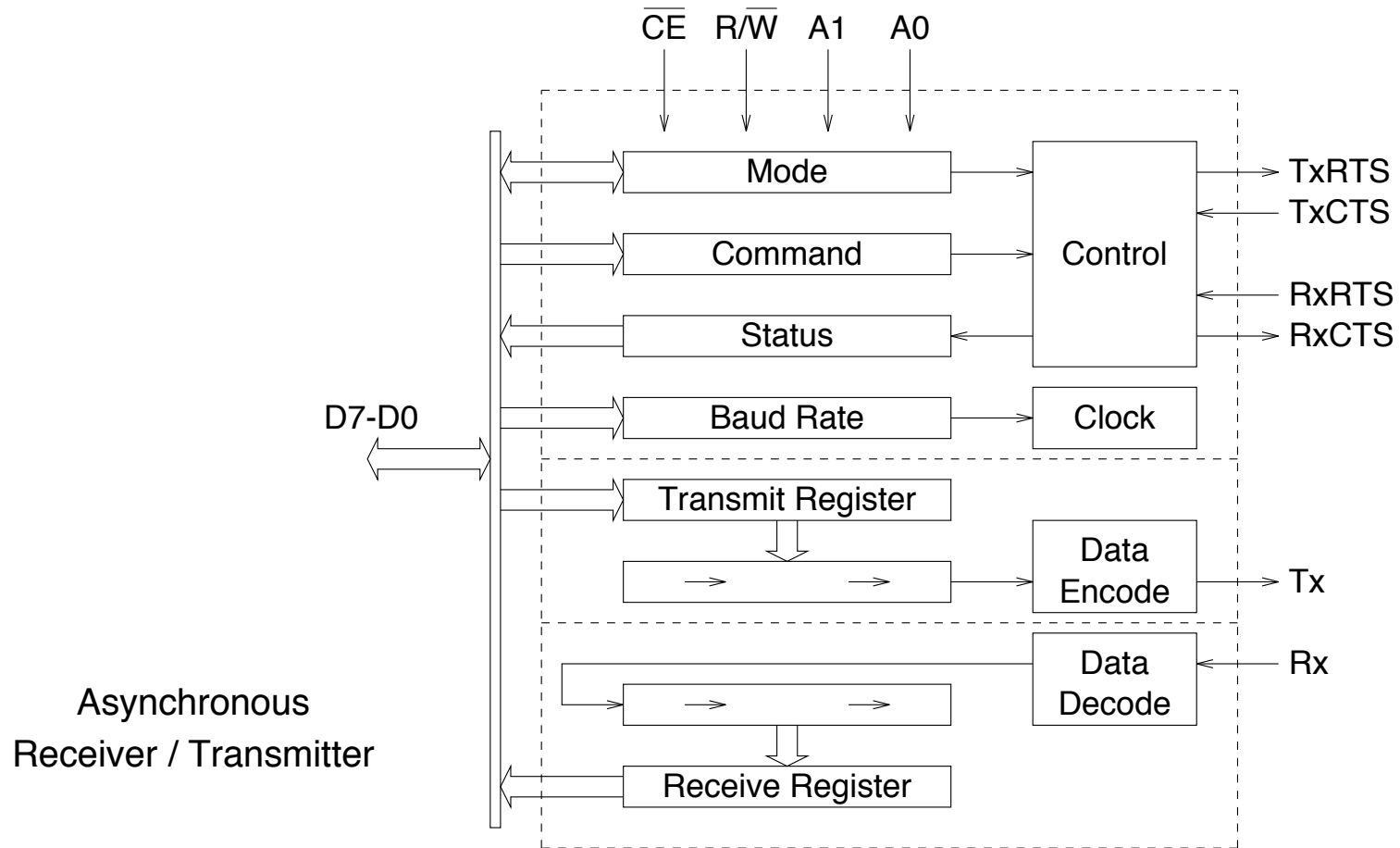
## Programmers View - Memory Mapped I/O

The programmer sees each input/output device as occupying one or more memory locations.



# I/O Devices

---



# I/O Devices

---

The registers of the asynchronous receiver/transmitter chip<sup>1</sup> are located at the following addresses.

A1,A0	Read	Write
00	Mode	Mode Select
01	Status	Baud Rate Select
10	NOT USED	Command
11	Receive Register	Transmit Register

## Register access modes

- read/write
- read only
- write only
- no direct access at all

---

<sup>1</sup>This chip is loosely based on the SCN2681 DUART as used on ECS SPARCboards.



# I/O Devices

---

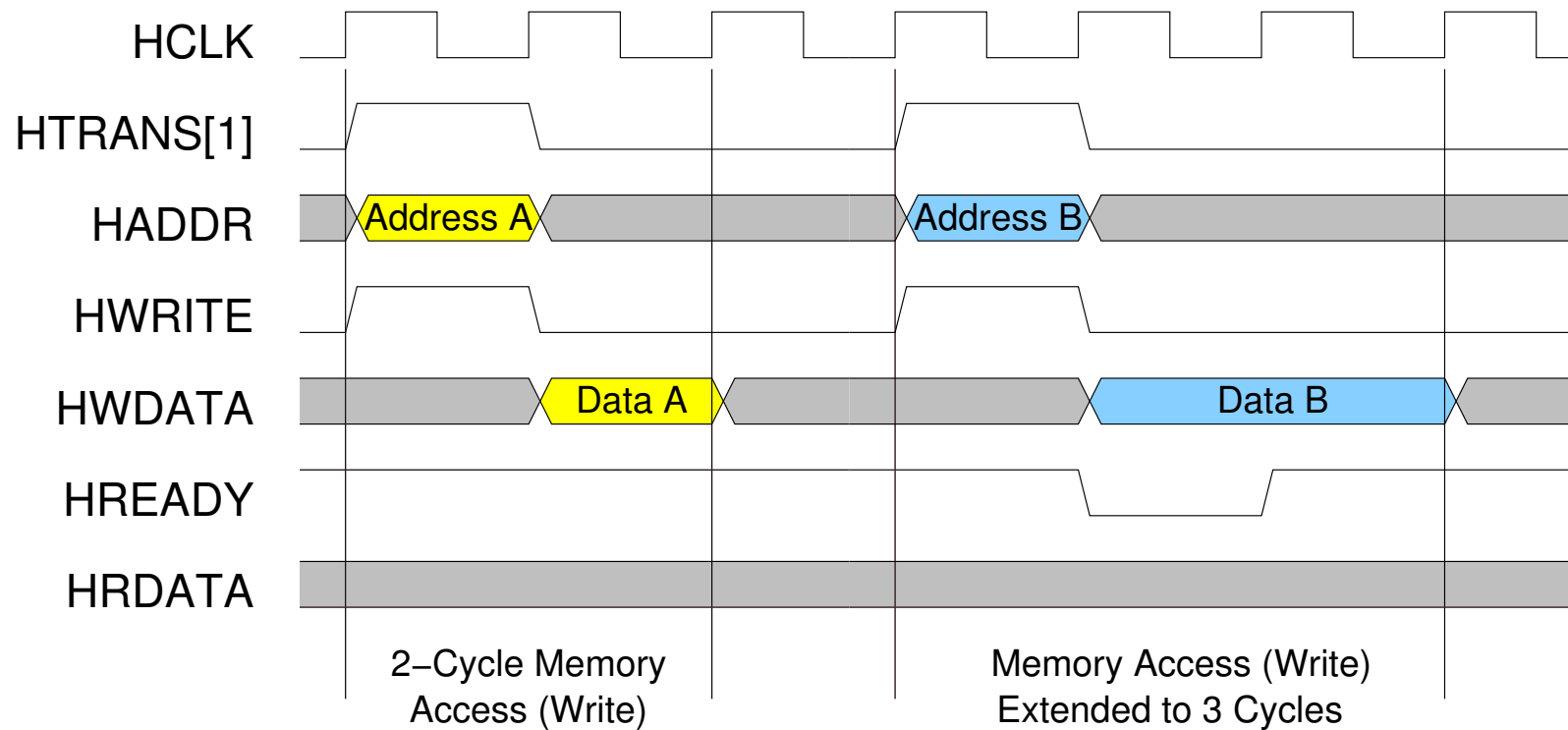
Although memory mapped, these registers do not act merely as memory locations used to transfer data. The action of accessing a register provides information to the device.

- Write to *Transmit register* triggers the output of data even if the data in the register is not changed.
- Read from *Receive register* indicates that the data is no longer required so that it may be overwritten.
- Data in the *Command register* is only consulted during a write, its effect is immediate – e.g. reset transmit machine.

# ARM System on Chip

## AHB-Lite Bus

Each memory access takes two or more clock cycles

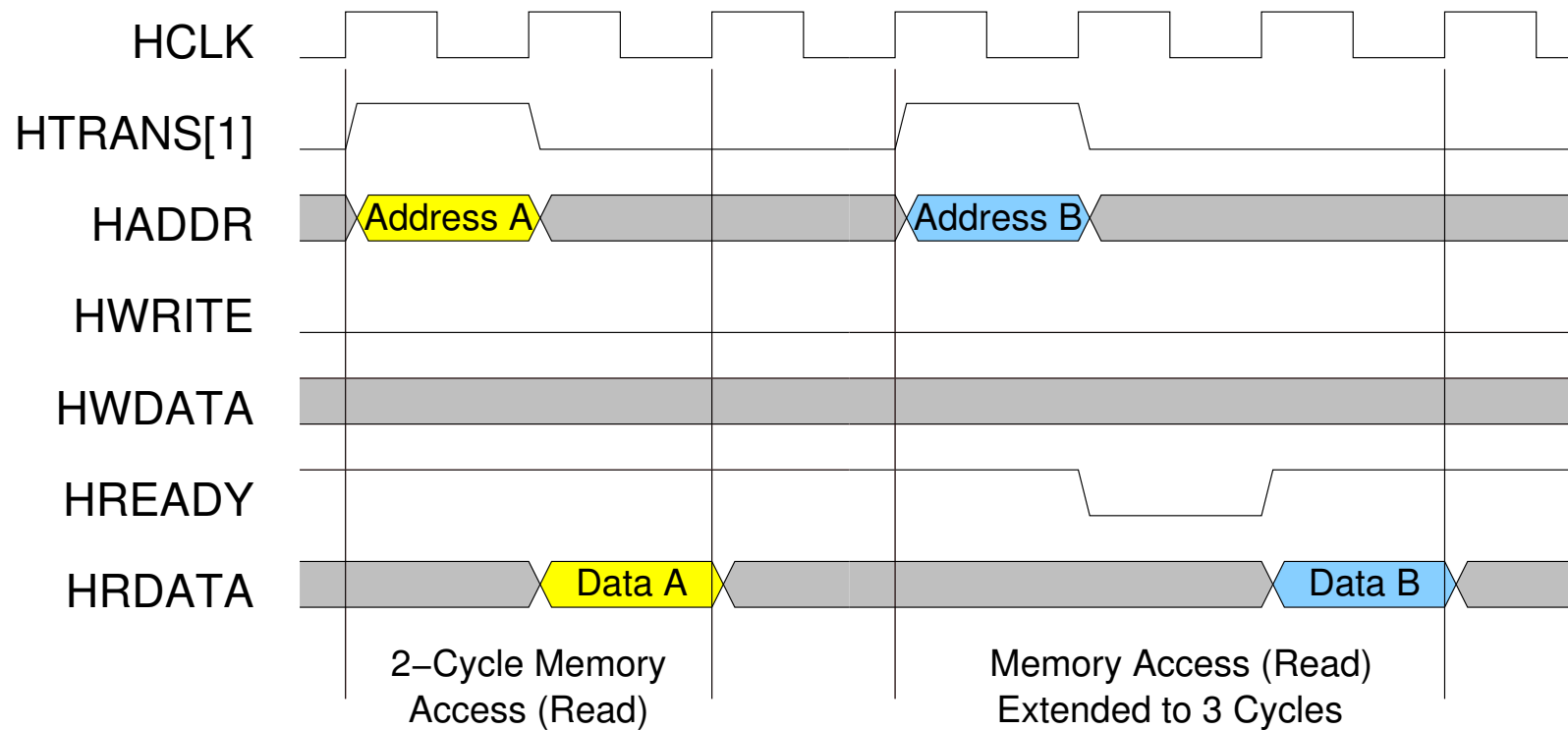


# ARM System on Chip

---

## AHB-Lite Bus

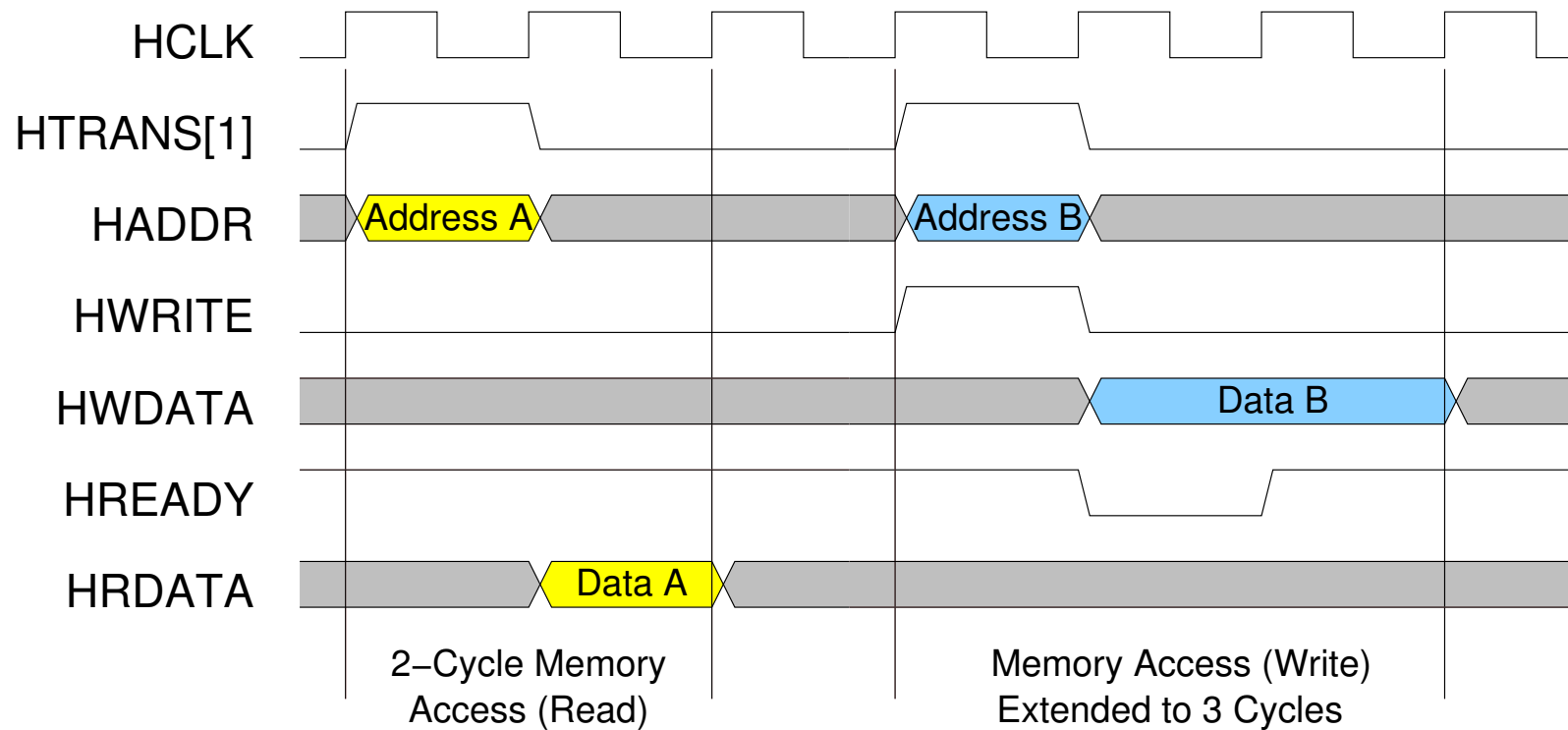
Each memory access takes two or more clock cycles



# ARM System on Chip

## AHB-Lite Bus

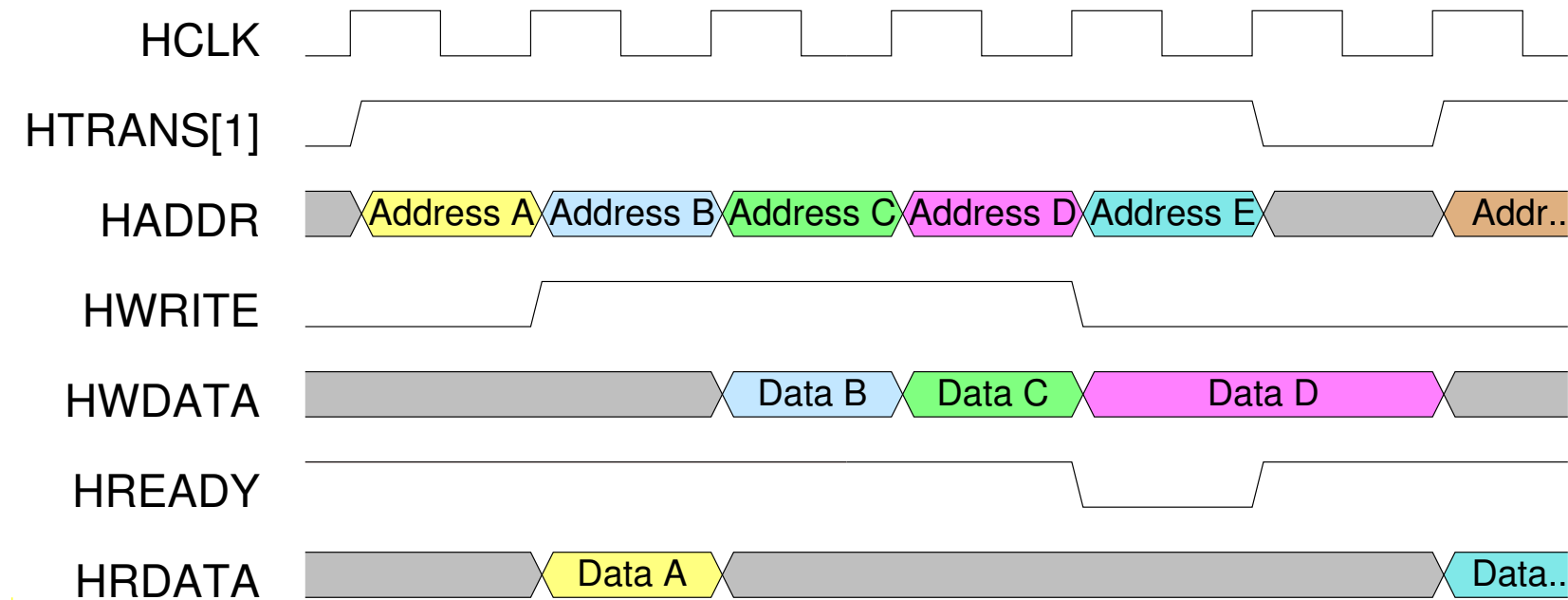
Each memory access takes two or more clock cycles



# ARM System on Chip

## AHB-Lite Bus

Memory accesses can be pipelined<sup>1</sup>

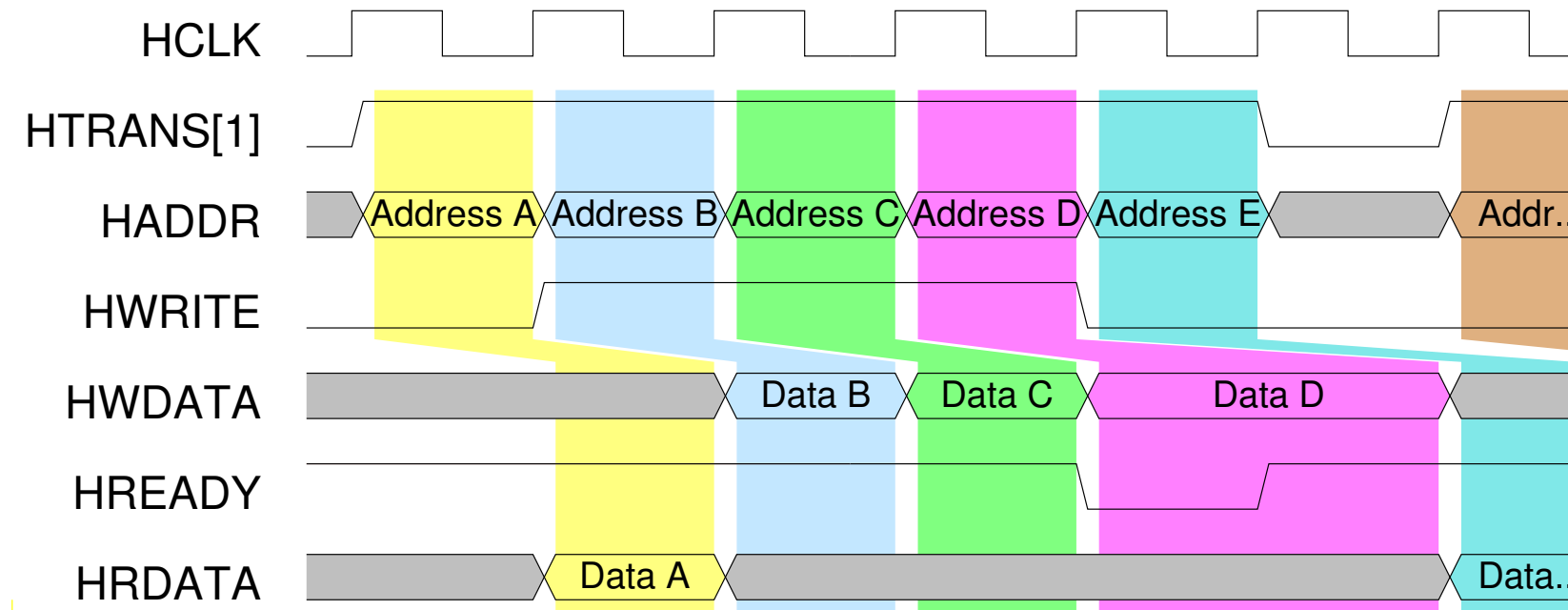


<sup>1</sup>the address phase of one memory access may start before the data phase of the previous access is complete

# ARM System on Chip

## AHB-Lite Bus

Memory accesses can be pipelined<sup>1</sup>

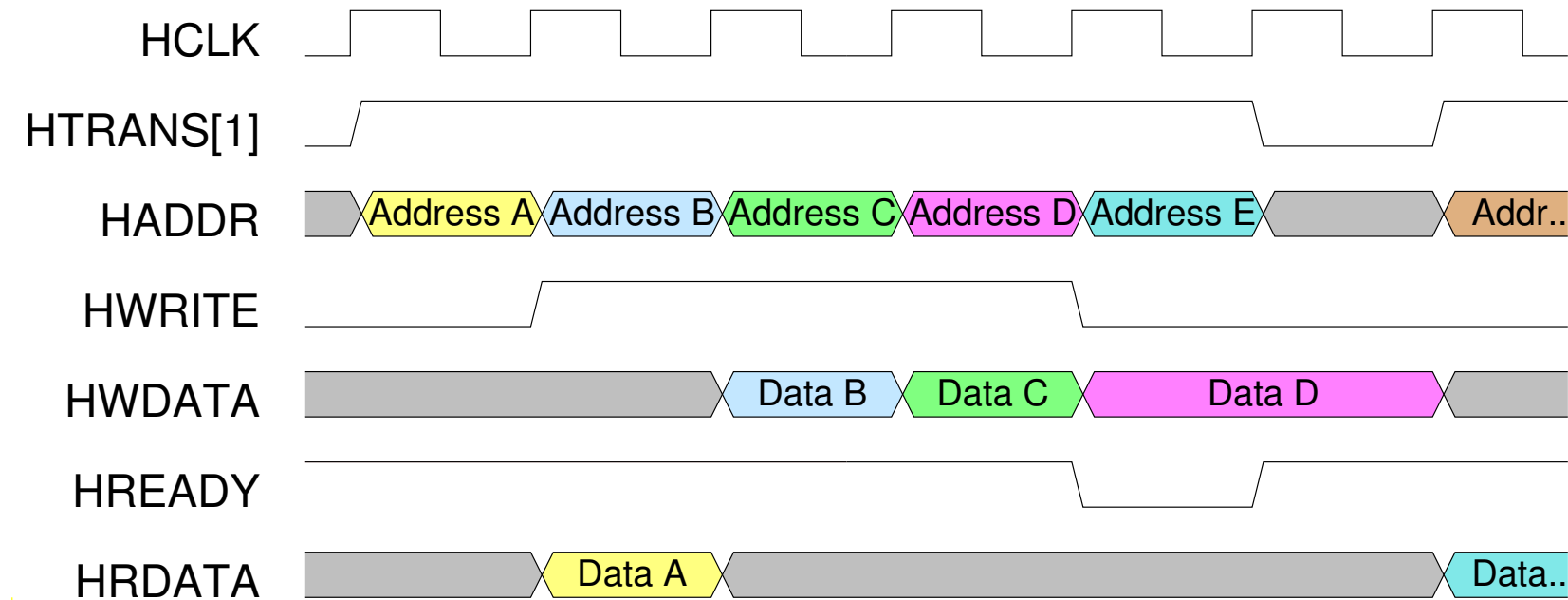


<sup>1</sup>the address phase of one memory access may start before the data phase of the previous access is complete

# ARM System on Chip

## AHB-Lite Bus

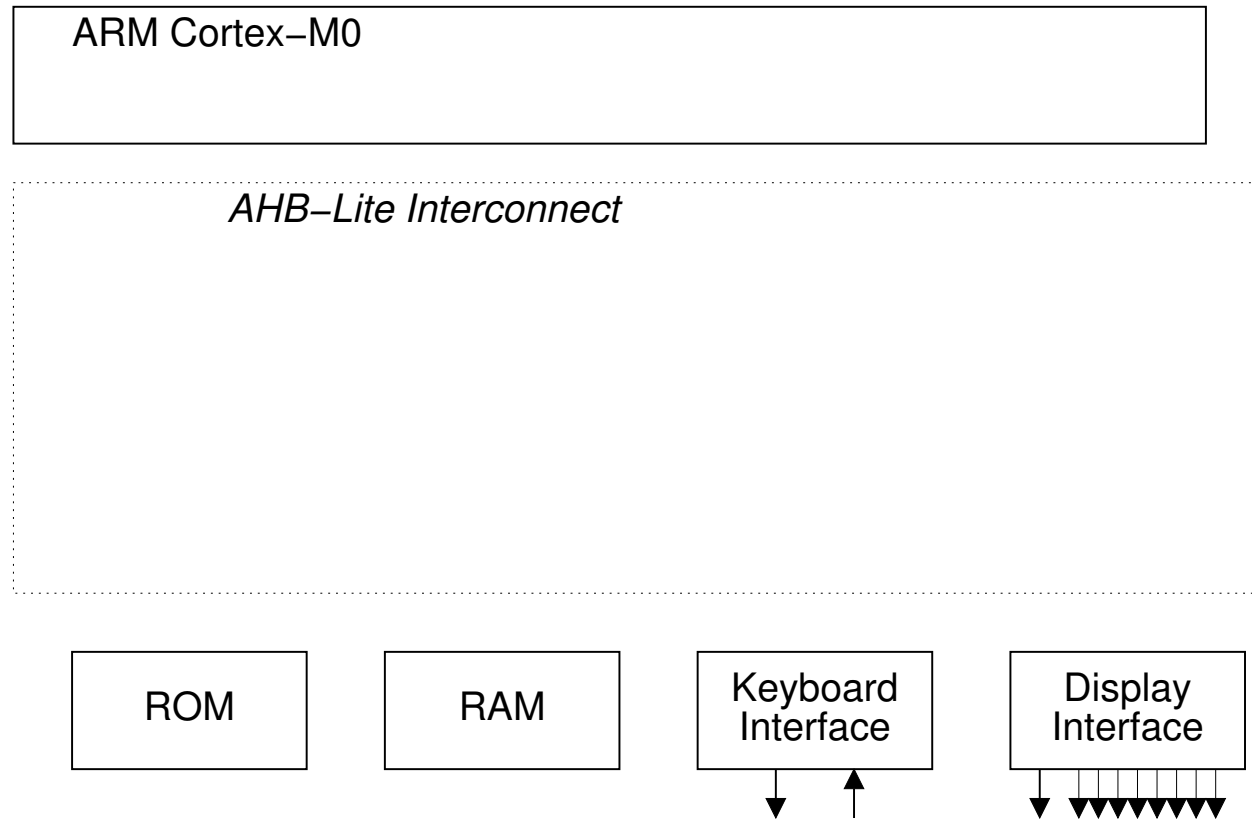
Memory accesses can be pipelined<sup>1</sup>



<sup>1</sup>the address phase of one memory access may start before the data phase of the previous access is complete

# ARM System on Chip

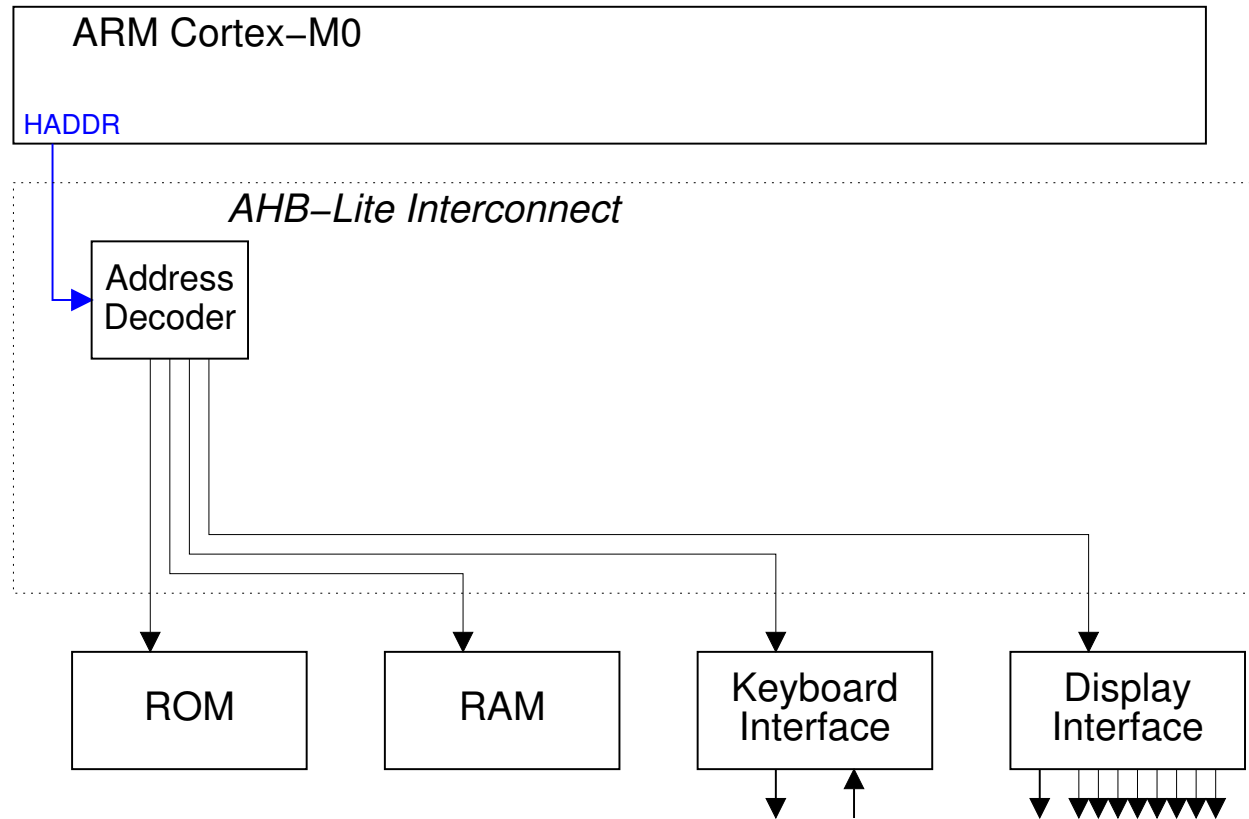
---





# ARM System on Chip

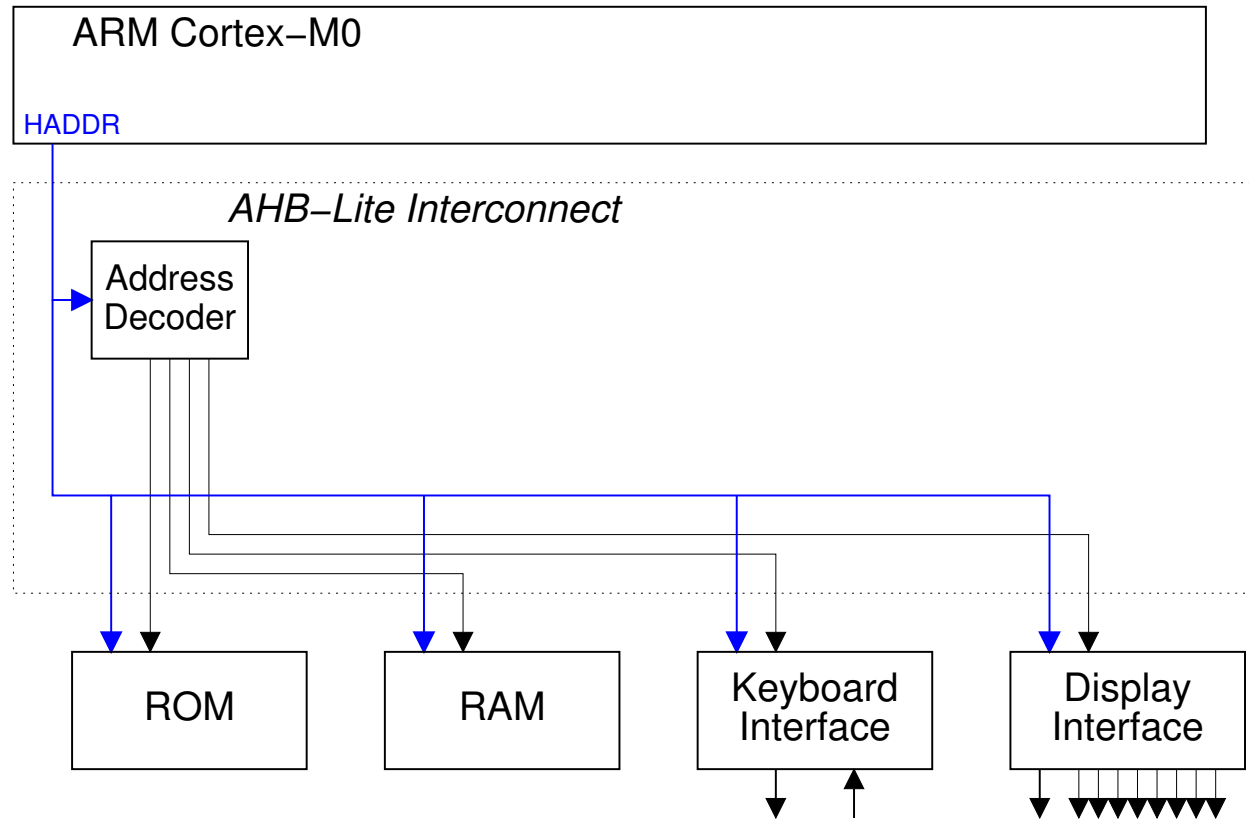
---



- Individual slave select signals are generated from the high order bits of HADDR

# ARM System on Chip

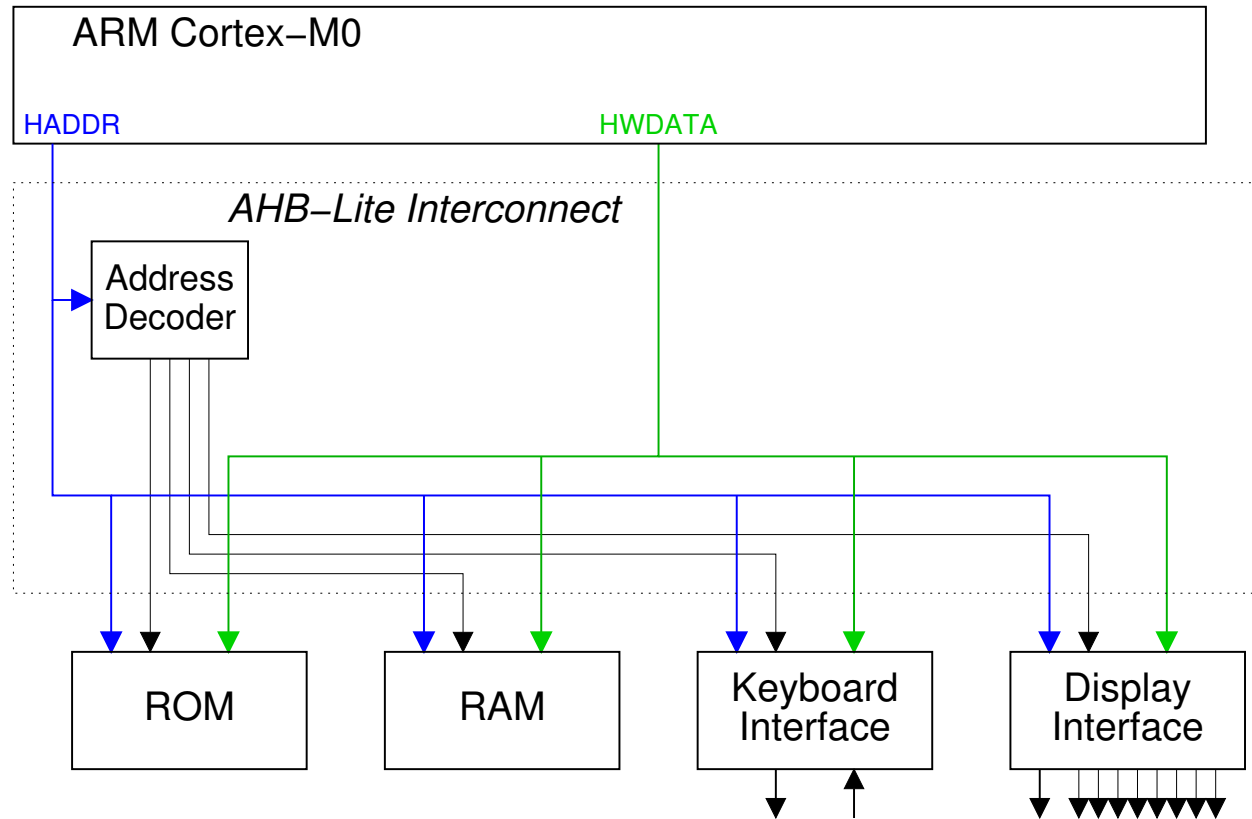
---



- The low order bits of HADDR are used to select registers within the slaves

# ARM System on Chip

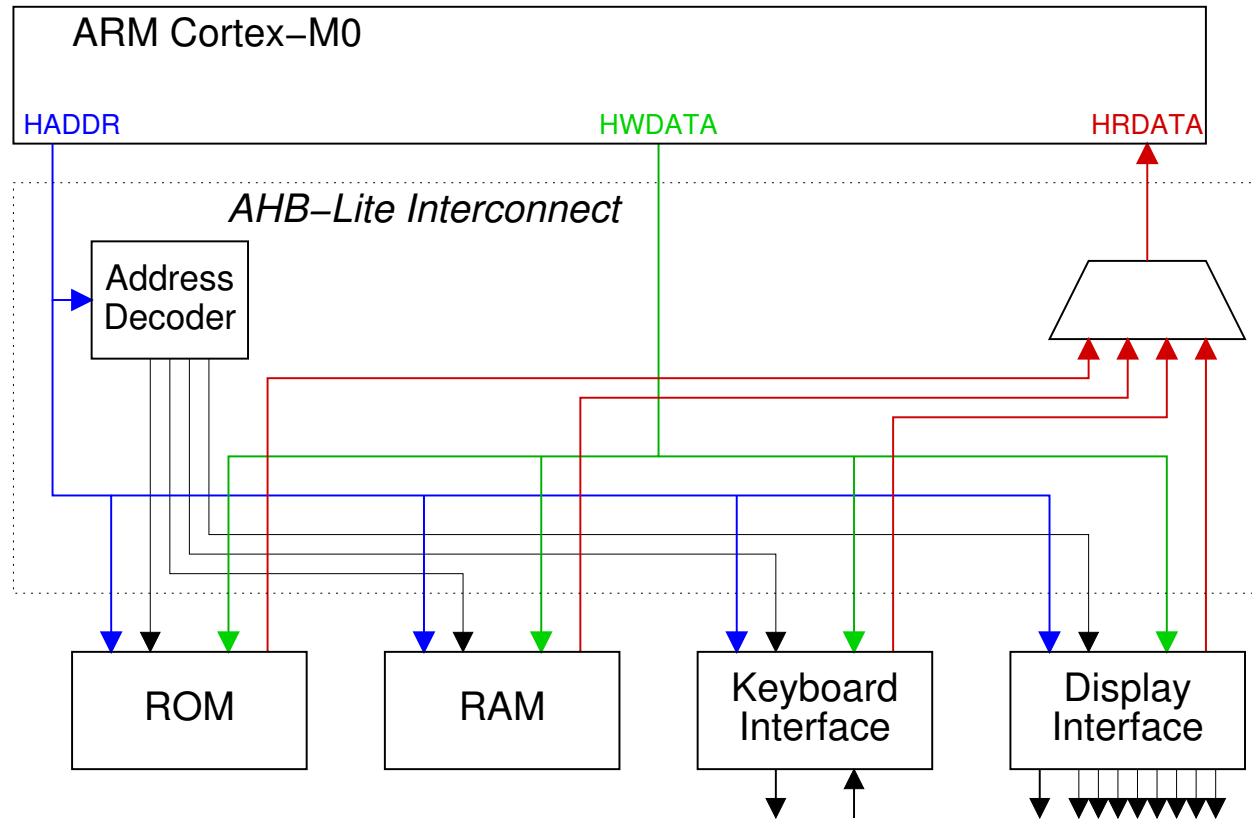
---



- The write data (HWDATA) is broadcast to all of the slaves

# ARM System on Chip

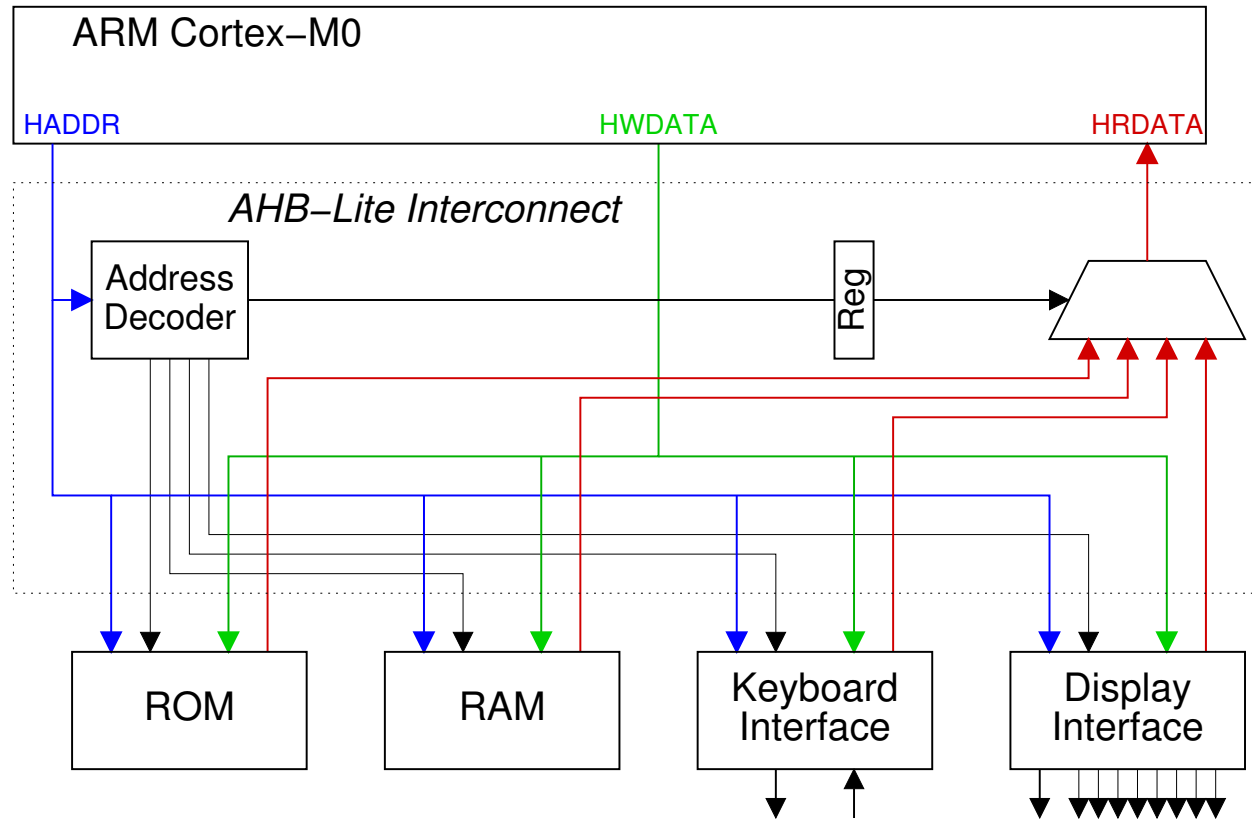
---



- A multiplexer selects the read data (HRDATA) from one of the slaves

# ARM System on Chip

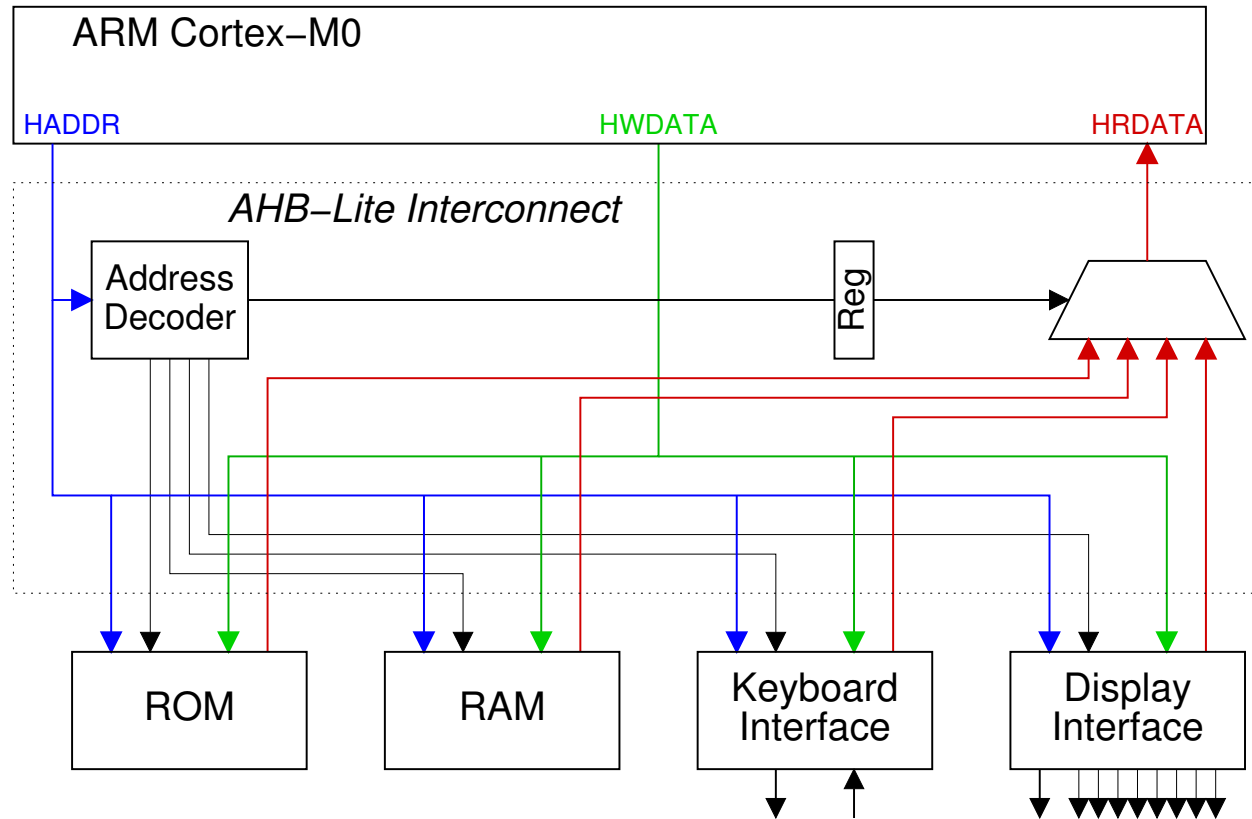
---



- The Address Decoder controls the multiplexer via a register which adds a single cycle delay

# ARM System on Chip

---

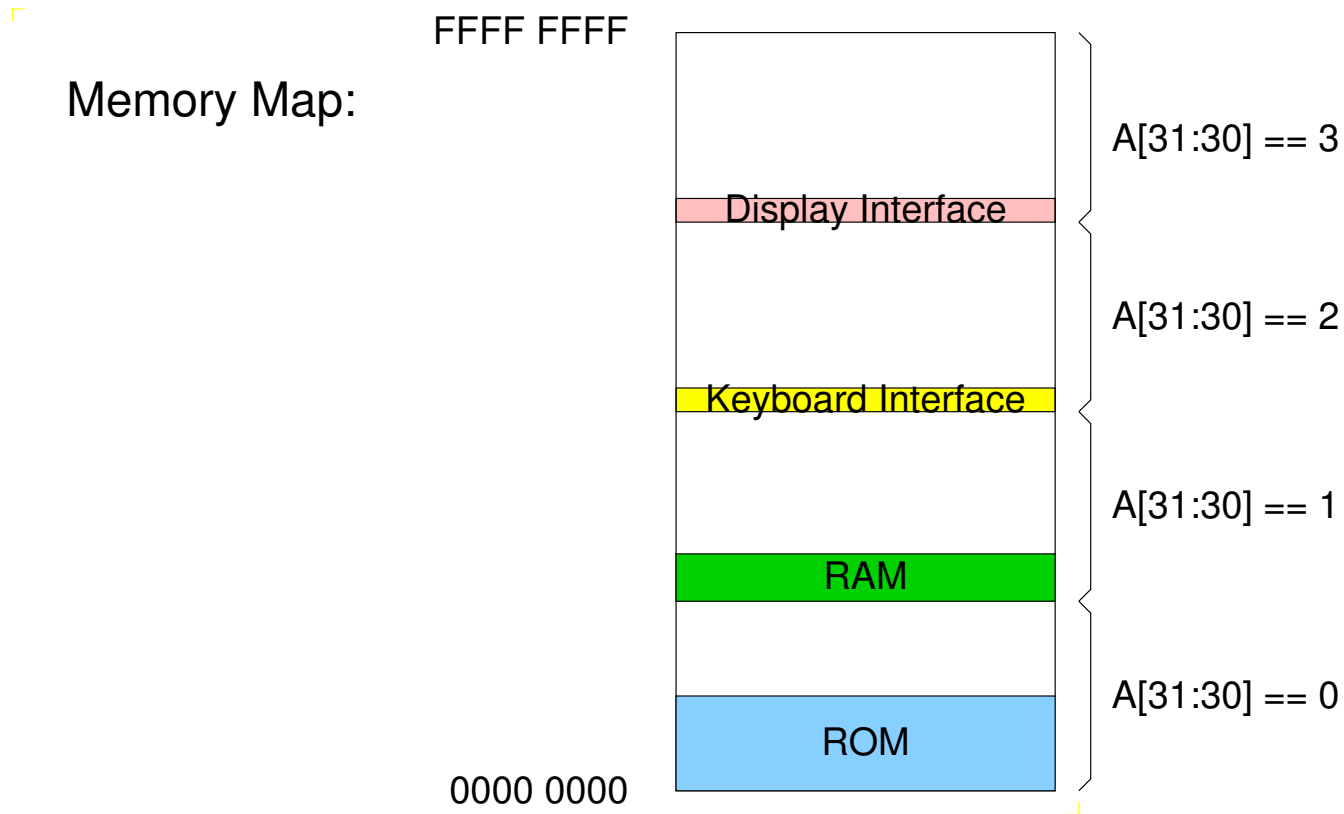


## AHB-Lite Interconnect

showing HADDR, HWDATA, HRDATA and individual slave HSEL select signals.

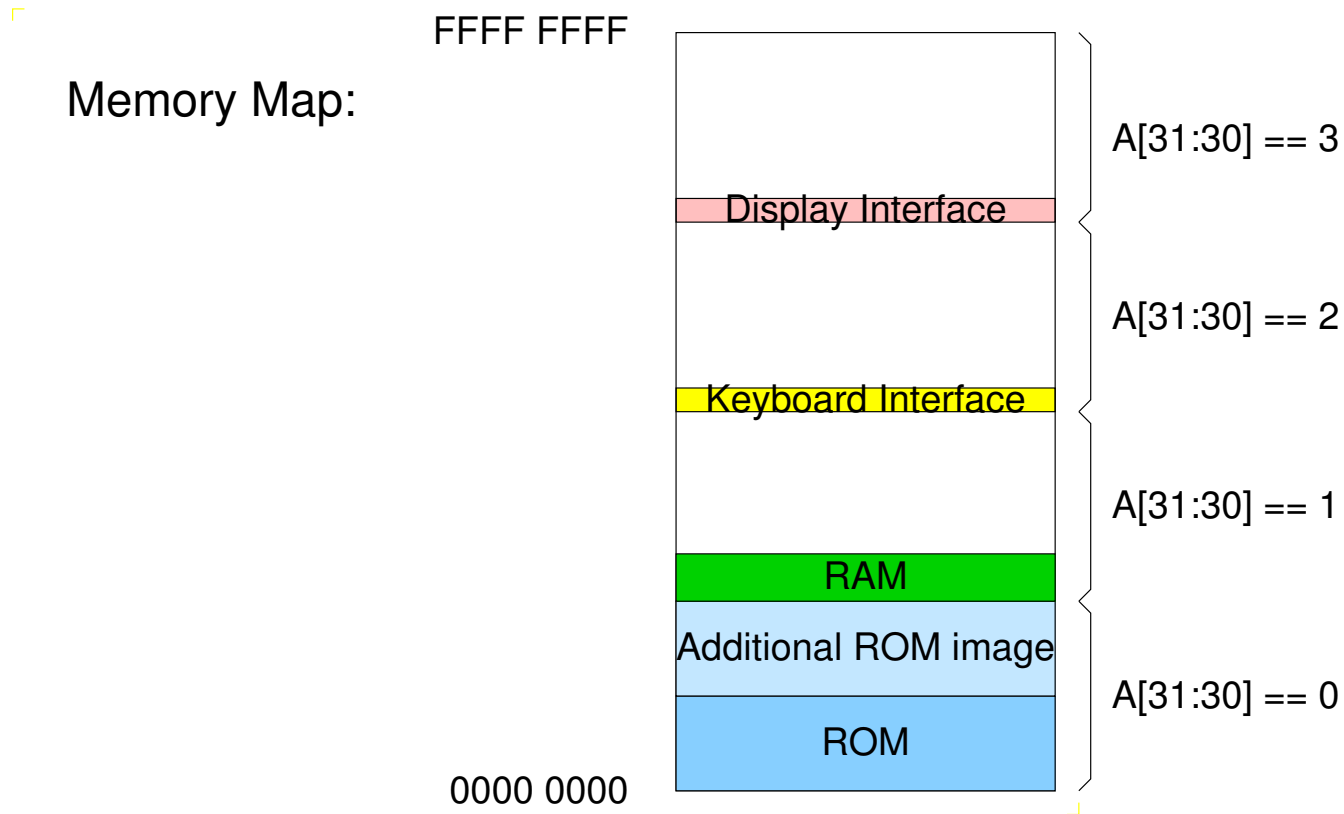
# ARM System on Chip

---



- With partial address decoding we use fewer address lines (A31 and A30 in the example above) and less logic for decoding.
- A side effect of this is that we get multiple images of some or all of the slave devices.

# ARM System on Chip



- With partial address decoding we use fewer address lines (A31 and A30 in the example above) and less logic for decoding.
- A side effect of this is that we get multiple images of some or all of the slave devices.

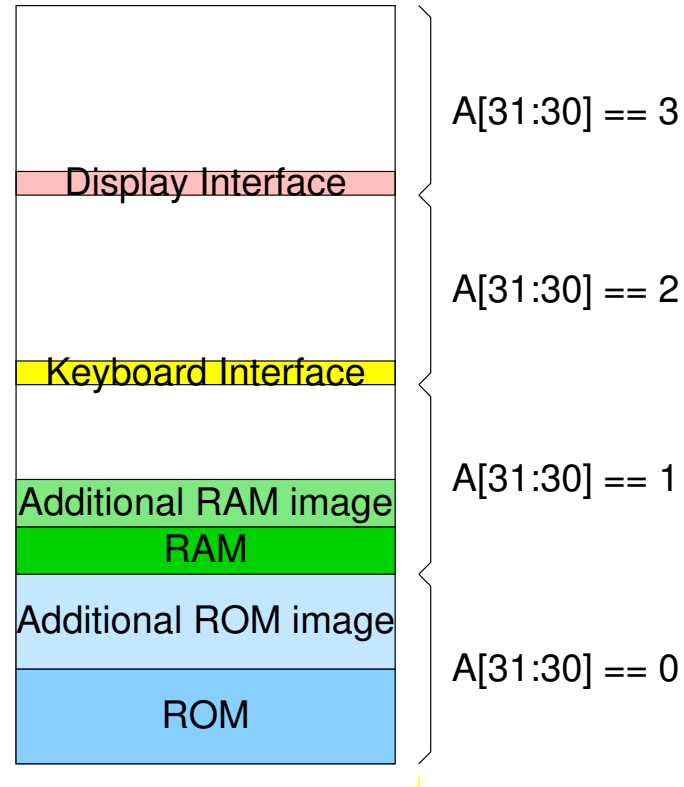


# ARM System on Chip

Memory Map:

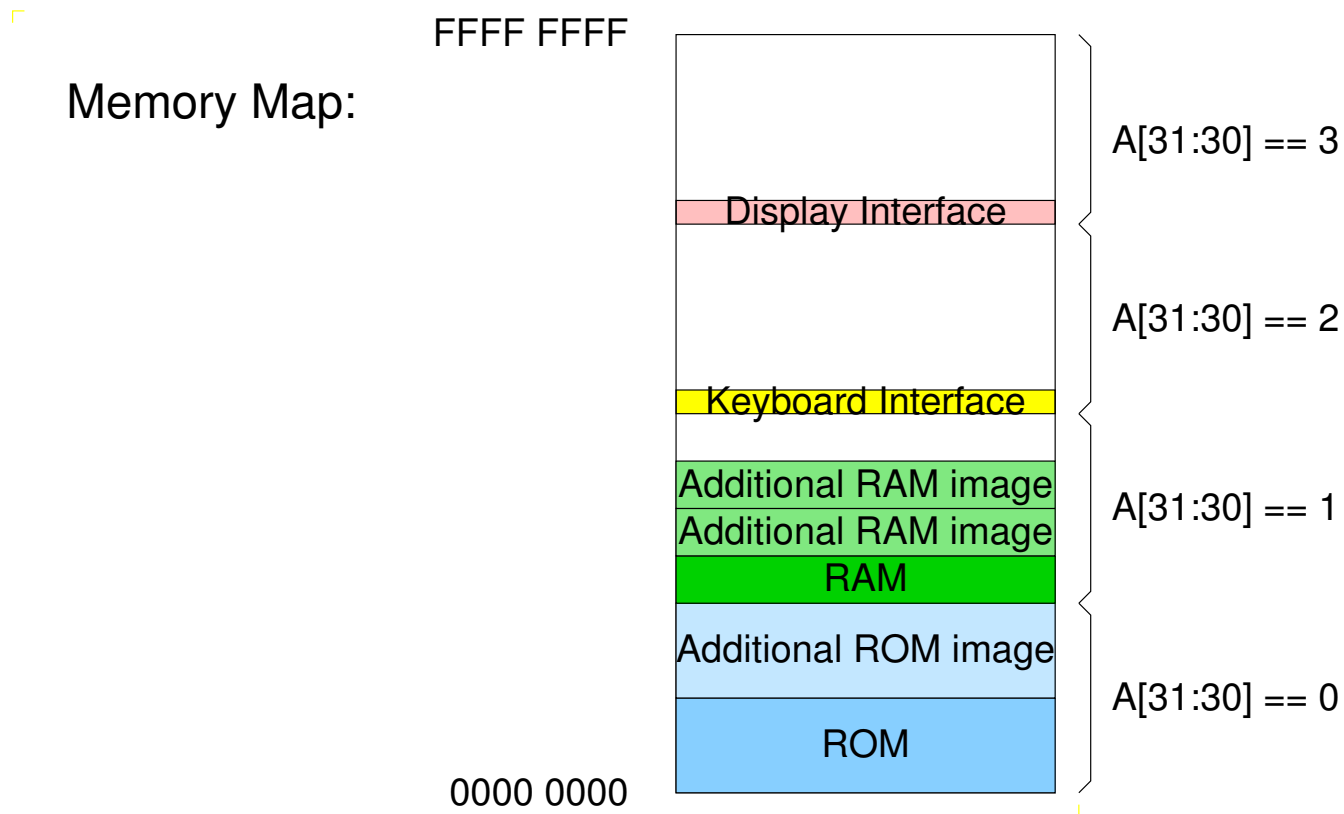
FFFF FFFF

0000 0000



- With partial address decoding we use fewer address lines (A31 and A30 in the example above) and less logic for decoding.
- A side effect of this is that we get multiple images of some or all of the slave devices.

# ARM System on Chip



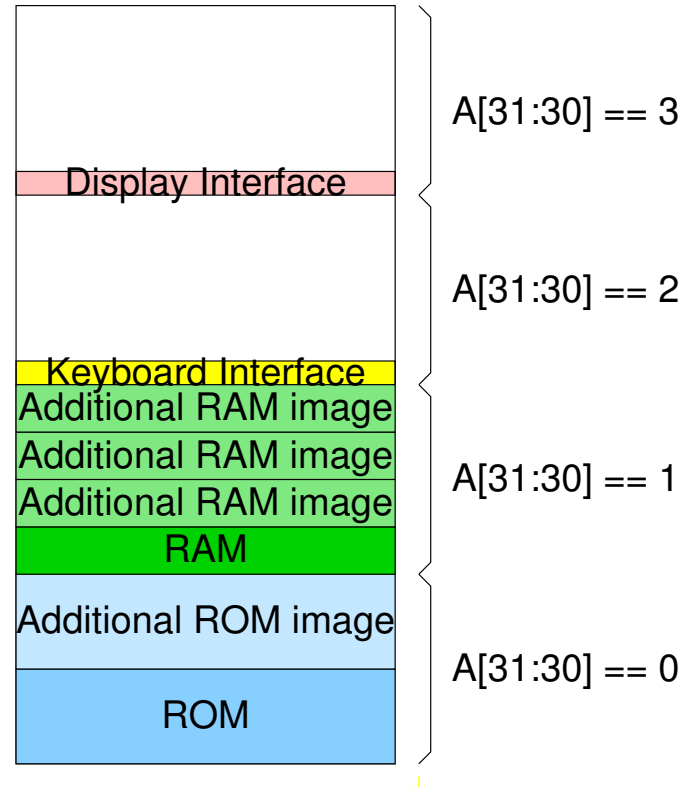
- With partial address decoding we use fewer address lines (A31 and A30 in the example above) and less logic for decoding.
- A side effect of this is that we get multiple images of some or all of the slave devices.

# ARM System on Chip

Memory Map:

FFFF FFFF

0000 0000

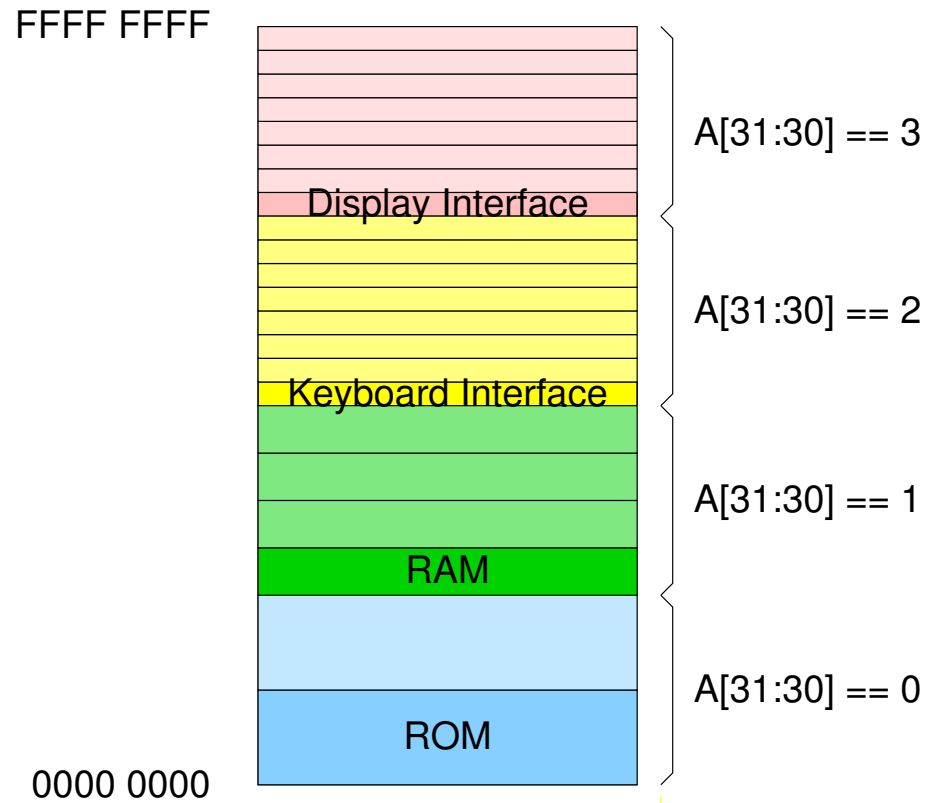


- With partial address decoding we use fewer address lines (A31 and A30 in the example above) and less logic for decoding.
- A side effect of this is that we get multiple images of some or all of the slave devices.

# ARM System on Chip

---

Memory Map:



- With partial address decoding we use fewer address lines (A31 and A30 in the example above) and less logic for decoding.
- A side effect of this is that we get multiple images of some or all of the slave devices.

ARM® AMBA®3 AHB-Lite

## Overview

# ARM® AMBA® Open Specification

- Open standard (No License required)
- The de facto standard for on-chip communication
- Used as on-chip interconnect for connecting and managing functional blocks in a System-on-Chip
- Promotes design re-use by defining common interface standards for SoC modules
- AMBA Family: AMBA 5, AMBA 4, AMBA 3 & AMBA 2
- AMBA 5 CHI (Coherent Hub Interface) specification is the latest addition to the AMBA (mainly used for server and networking SoCs)
- More info: <http://www.arm.com/products/system-ip/amba/amba-open-specifications.php>

# AMBA Acronyms

## ■ Acronyms

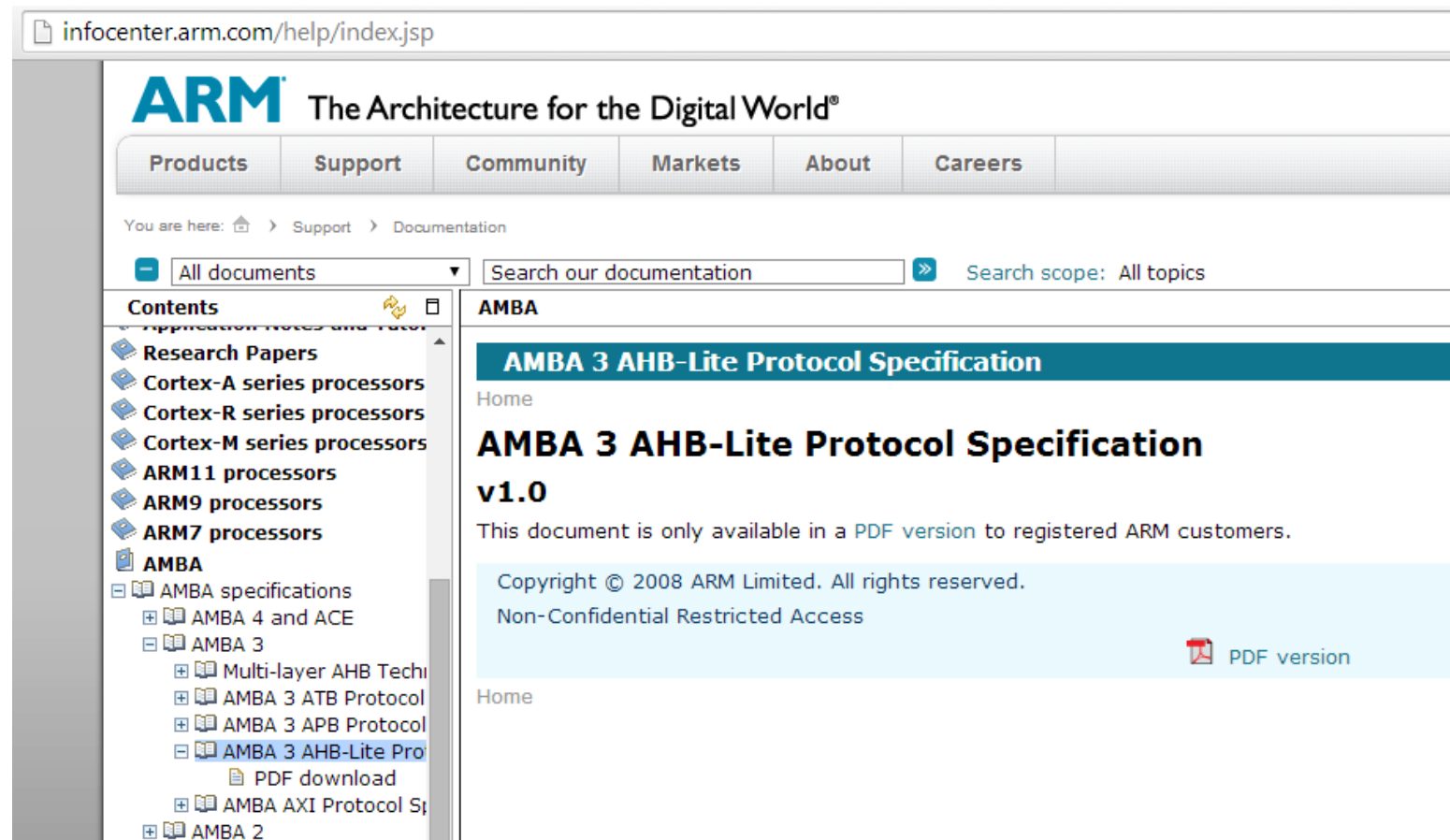
- AMBA<sup>®</sup> → Advanced Microcontroller Bus Architectures
- AXI → Advanced eXtensible Interface
- ACE → AXI Coherency Extensions
- AHB → Advanced High-Performance Bus
- APB → Advanced Peripheral Bus
- ATB → Advanced Trace Bus
- ASB → Advanced System Bus

Image Source: Google.com



# How to access the full specification?

- Go to <http://infocenter.arm.com/>



The screenshot shows the ARM infocenter website. The browser address bar displays `infocenter.arm.com/help/index.jsp`. The page header features the ARM logo and the tagline "The Architecture for the Digital World®". A navigation bar includes links for Products, Support, Community, Markets, About, and Careers. Below this, a breadcrumb trail indicates the current location: "You are here: > Support > Documentation". A search bar is present with a dropdown menu set to "All documents" and a search scope of "All topics". The left sidebar contains a "Contents" section with a tree view. Under the "AMBA" category, the "AMBA 3 AHB-Lite Protocol Specification" is highlighted. The main content area displays the title "AMBA 3 AHB-Lite Protocol Specification v1.0" and a notice that the document is only available in a PDF version to registered ARM customers. A copyright notice for 2008 ARM Limited is also visible, along with a "PDF version" link.

infocenter.arm.com/help/index.jsp

**ARM** The Architecture for the Digital World®

Products Support Community Markets About Careers

You are here: > Support > Documentation

All documents Search our documentation Search scope: All topics

**Contents**

- Research Papers
- Cortex-A series processors
- Cortex-R series processors
- Cortex-M series processors
- ARM11 processors
- ARM9 processors
- ARM7 processors
- AMBA**
  - AMBA specifications
    - AMBA 4 and ACE
    - AMBA 3
      - Multi-layer AHB Tech
      - AMBA 3 ATB Protocol
      - AMBA 3 APB Protocol
      - AMBA 3 AHB-Lite Pro**
        - PDF download
      - AMBA AXI Protocol S
    - AMBA 2

**AMBA**

**AMBA 3 AHB-Lite Protocol Specification**

Home

**AMBA 3 AHB-Lite Protocol Specification v1.0**

This document is only available in a [PDF version](#) to registered ARM customers.

Copyright © 2008 ARM Limited. All rights reserved.  
Non-Confidential Restricted Access

[PDF version](#)

Home



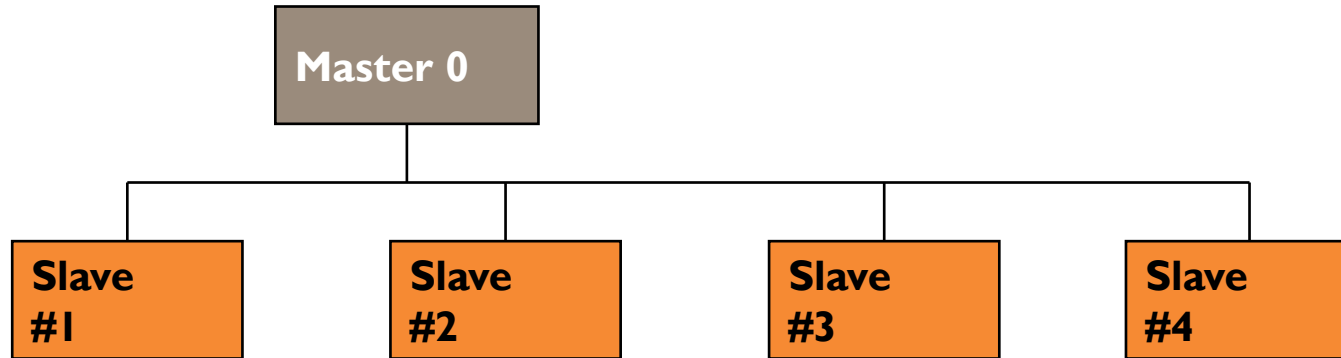
# AMBA 3 AHB-Lite

- Original AHB Specification was part of AMBA 2
- Subset of original AHB
- Reduced interconnect logic
- Simplifies slave design
- Master slave architecture
- Most of the designs have single master in the system
- Multiple masters still possible on multi-layer interconnect

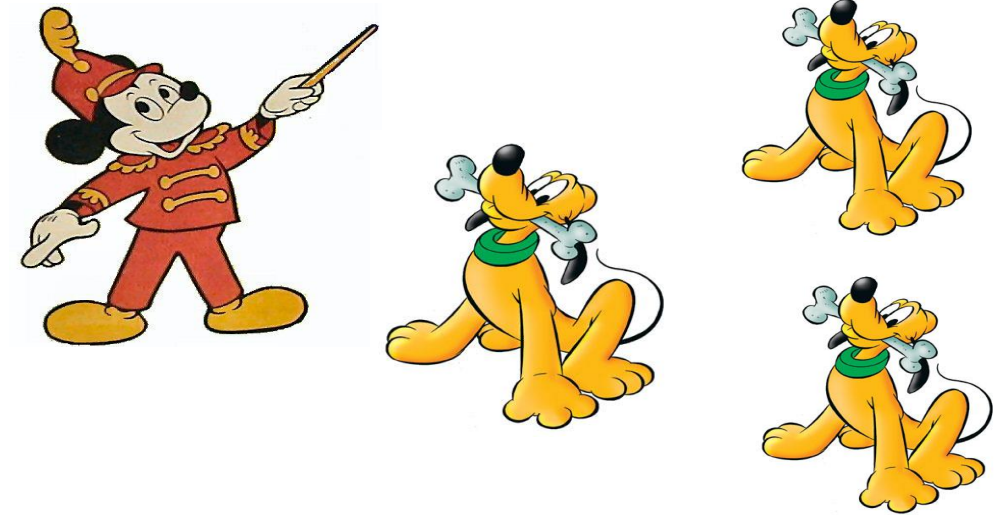
# AMBA 3 AHB-Lite



# AMBA 3 AHB-Lite



- Single Master
- Simple slaves
- Easier module design/debug
- No arbitration issues



# AHB-Lite transactions

- Master

- Register Read
- Register Write
- Burst Read
- Burst Write



- Slave/Peripheral

- Can make Master wait
- Can give error response



Image Source: Walt Disney


# AHB-Lite Features

- Single Clock Edge operation
- Uni-directional busses
  - No tri-state signals
  - Good for synthesis
- Pipelined Operation

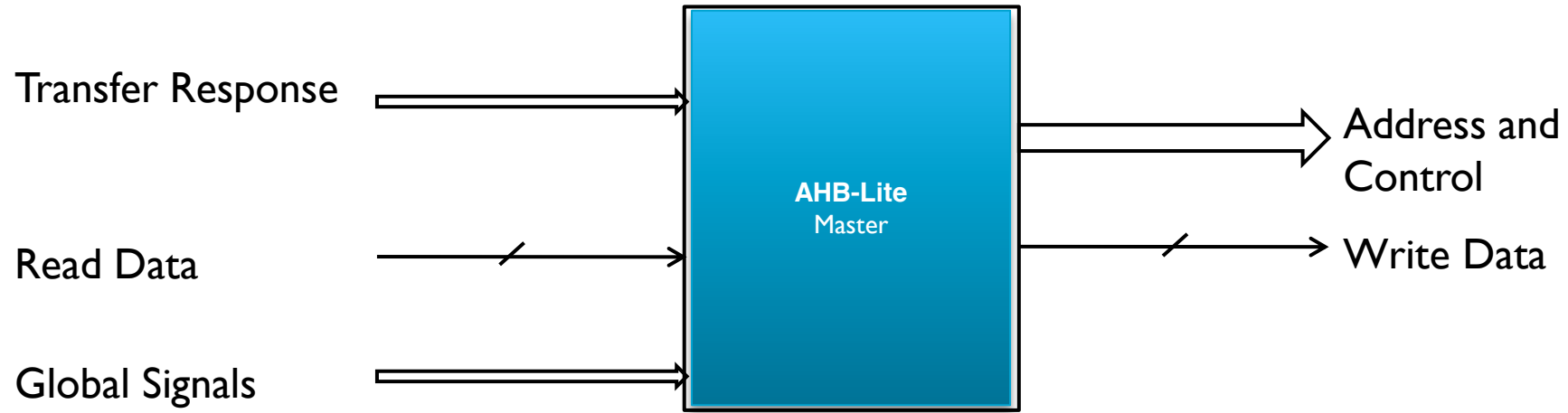


A system based on AHB-Lite

# Components of AHB-Lite System

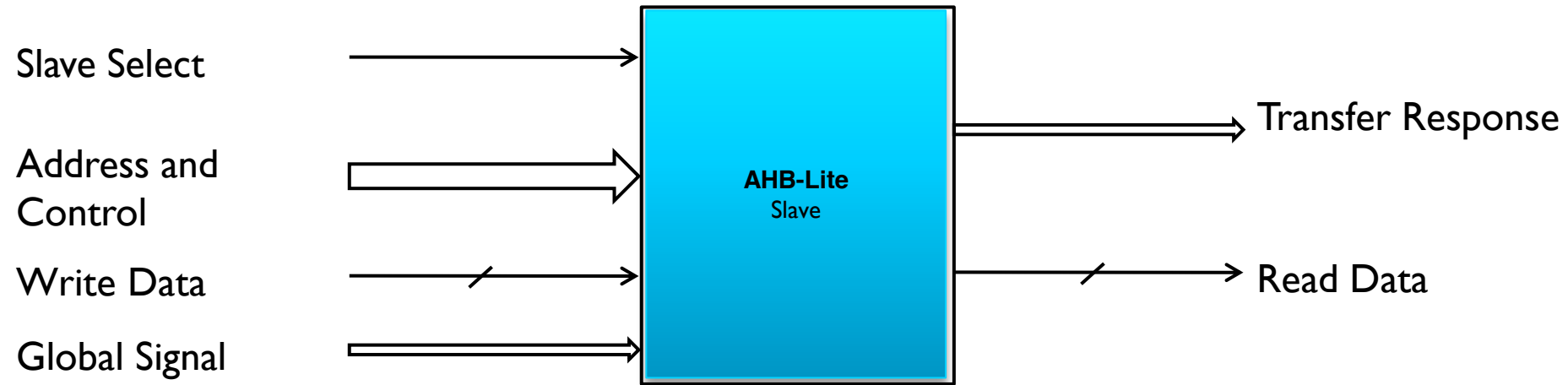
- Master
  - Slaves/Peripherals
  - Address Decoder
  - Multiplexor
- 
- AHB-Lite Interconnect**

# AHB-Lite Master

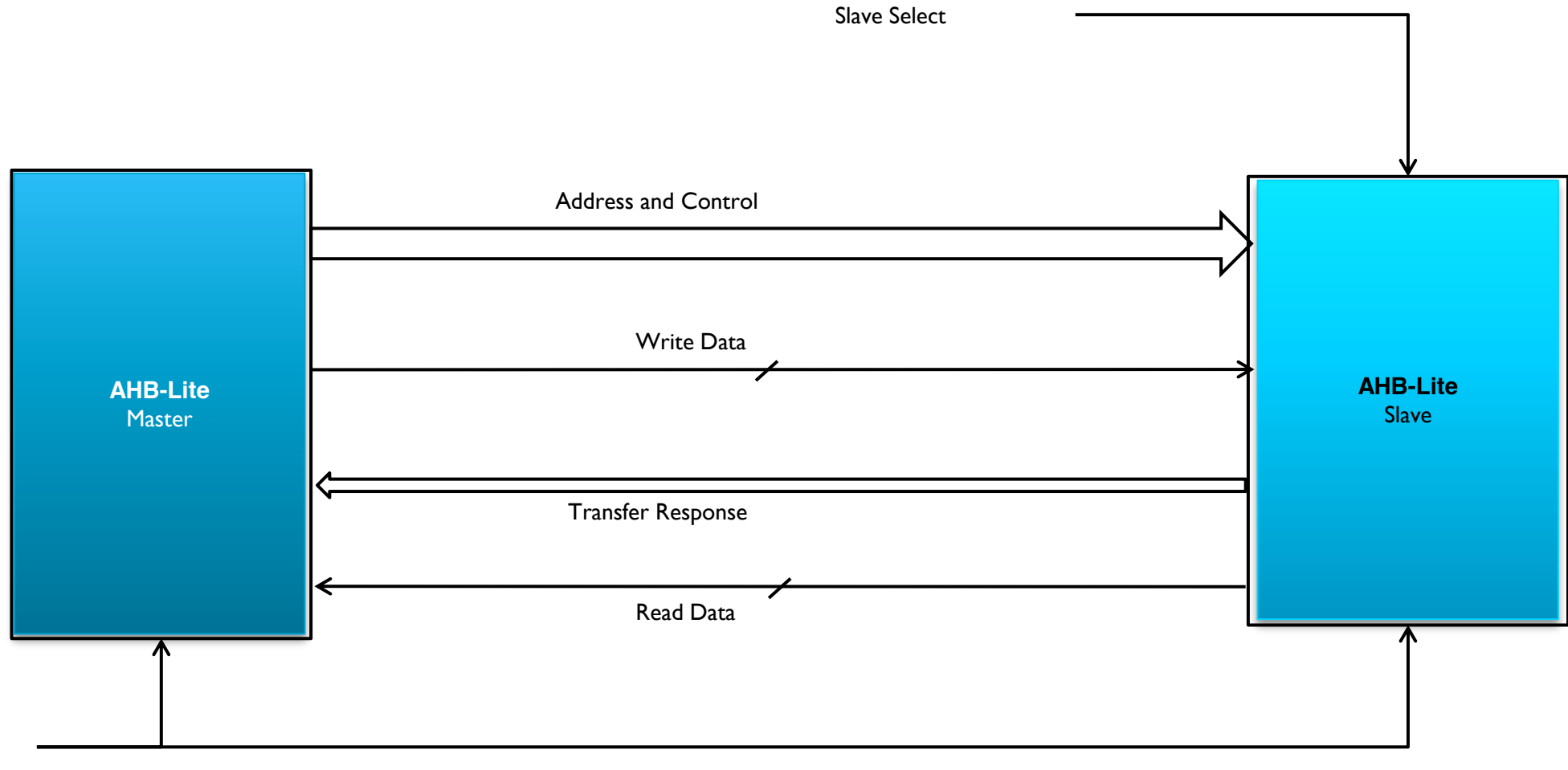




# AHB-Lite Slave

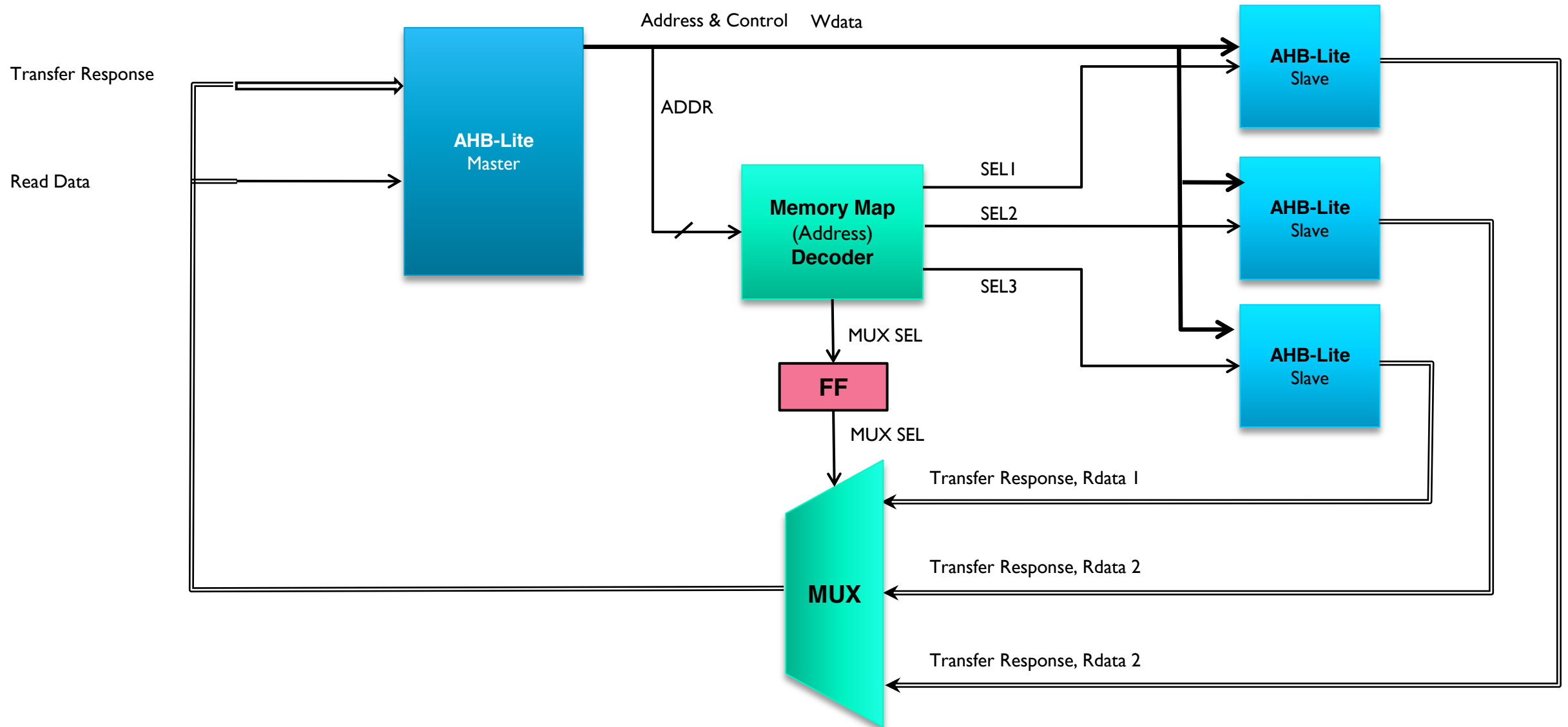


# AHB-Lite Master & Slave

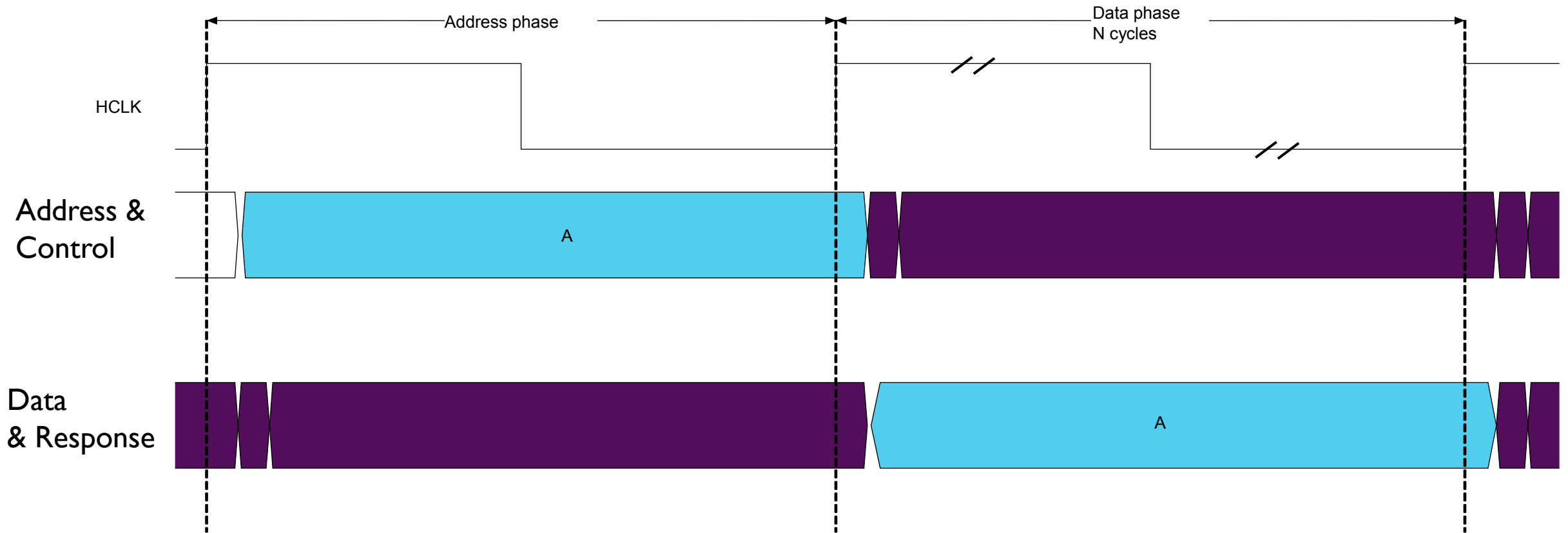


Global Signals

# Decoder & MUX

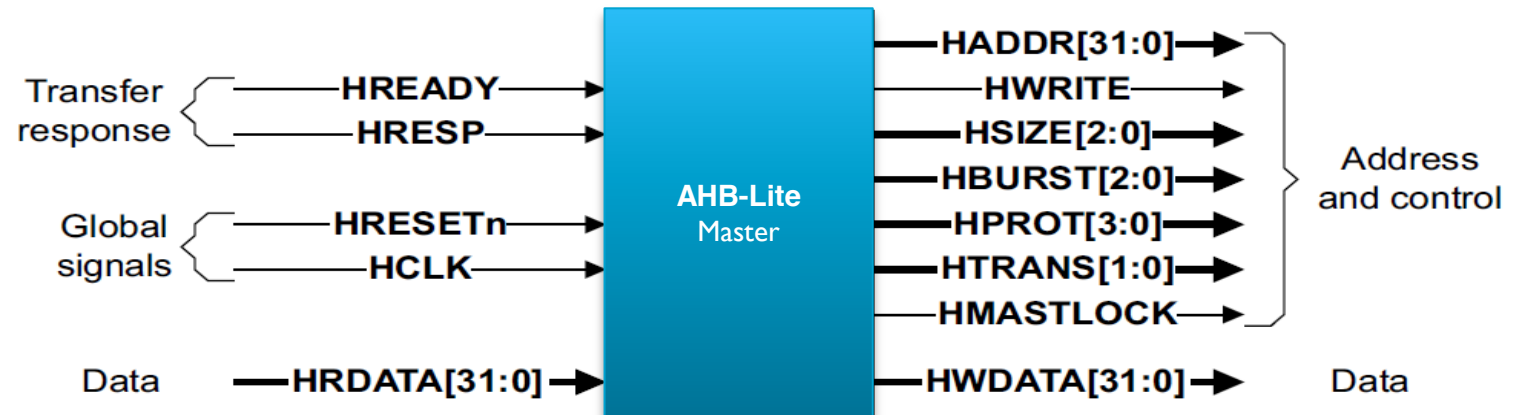
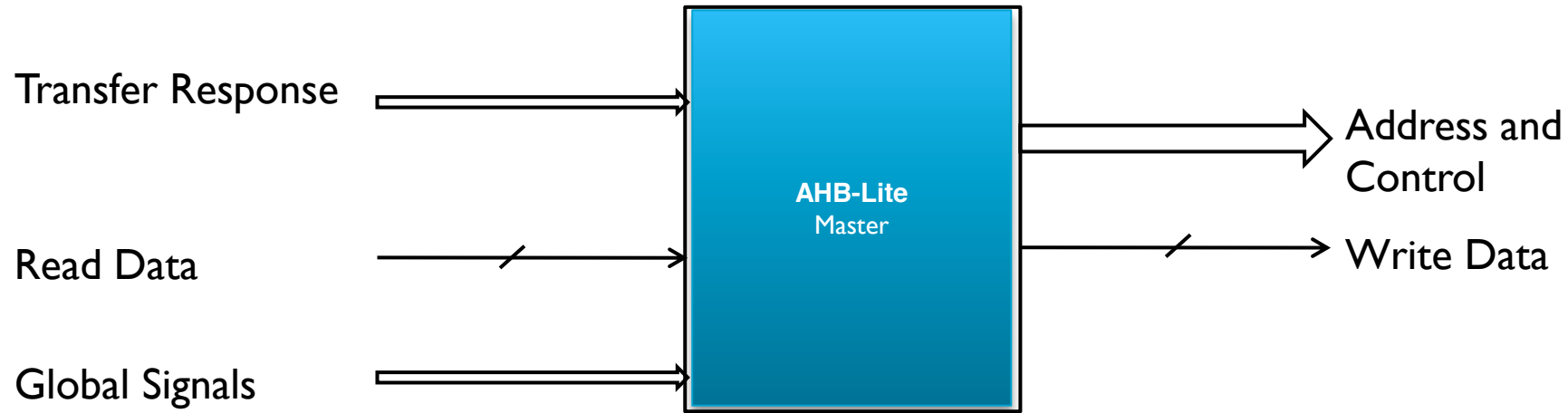


# Pipelined Transactions (Conceptual Level)

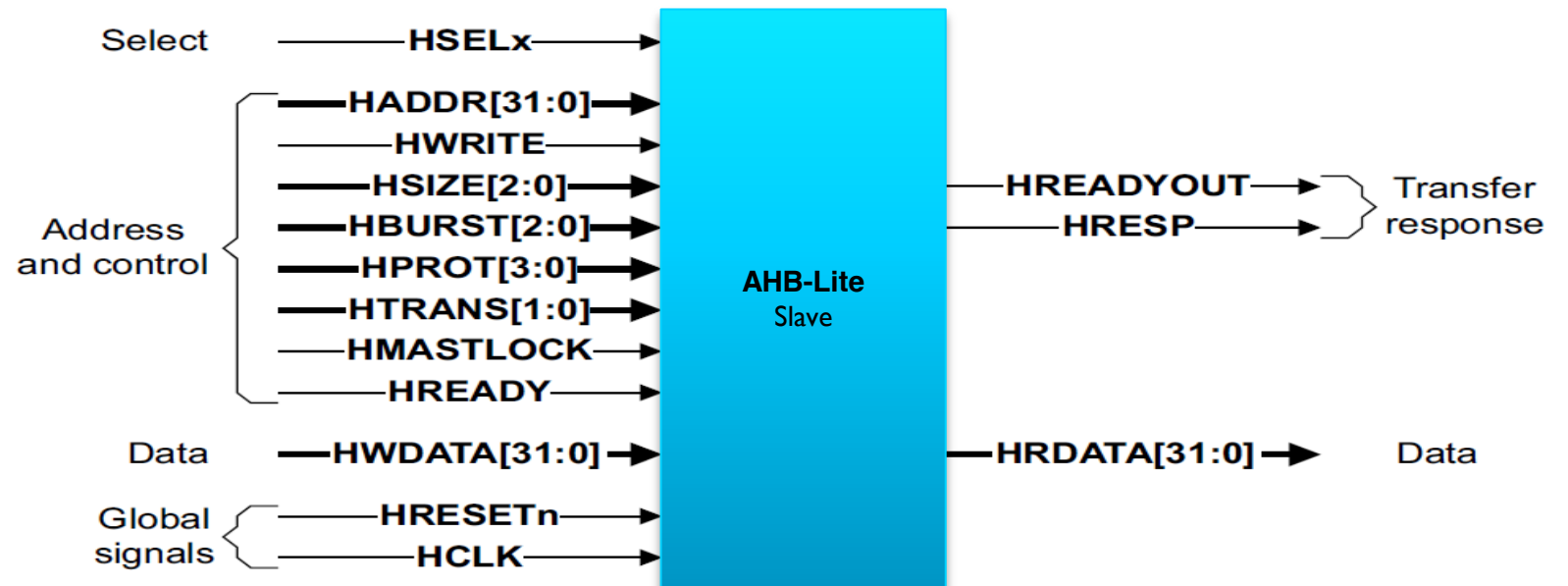
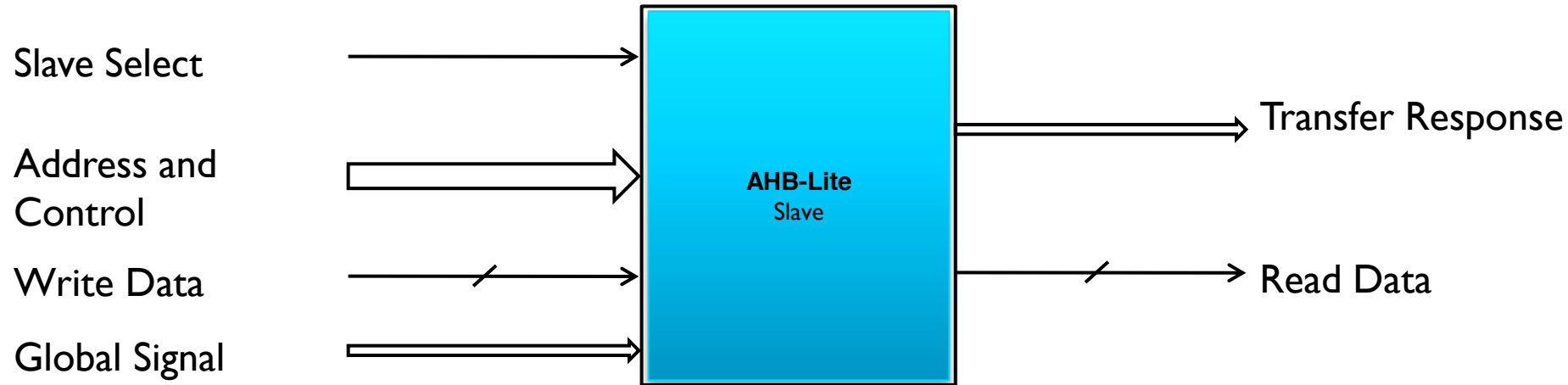


# AHB-Lite Signals

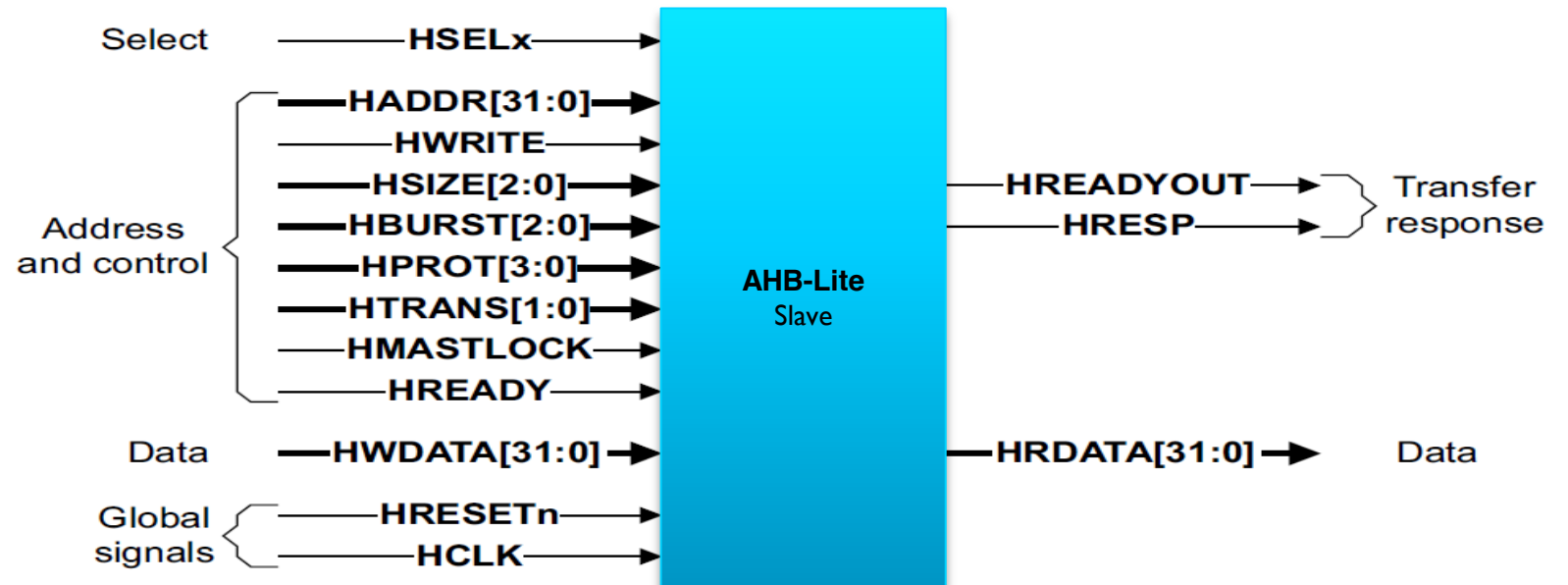
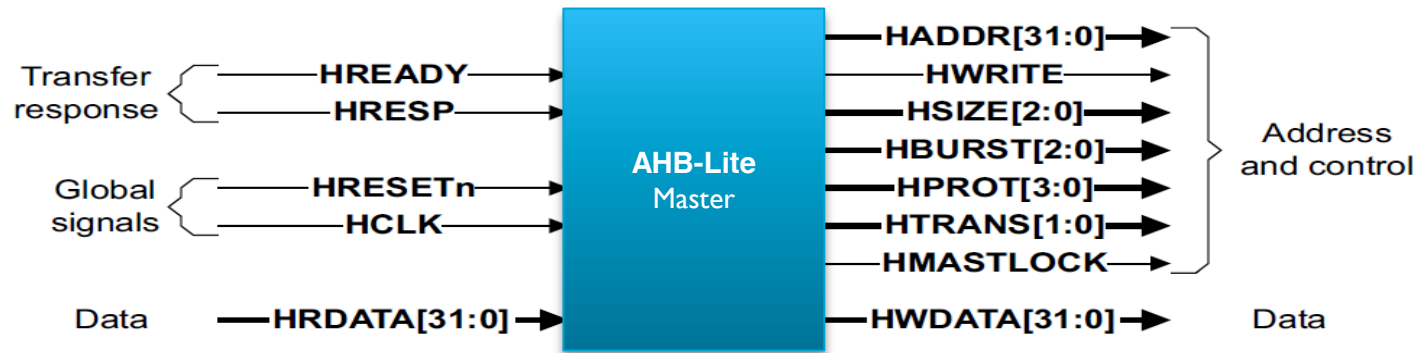
# AHB-Lite Master Signals



# AHB-Lite Slave Signals

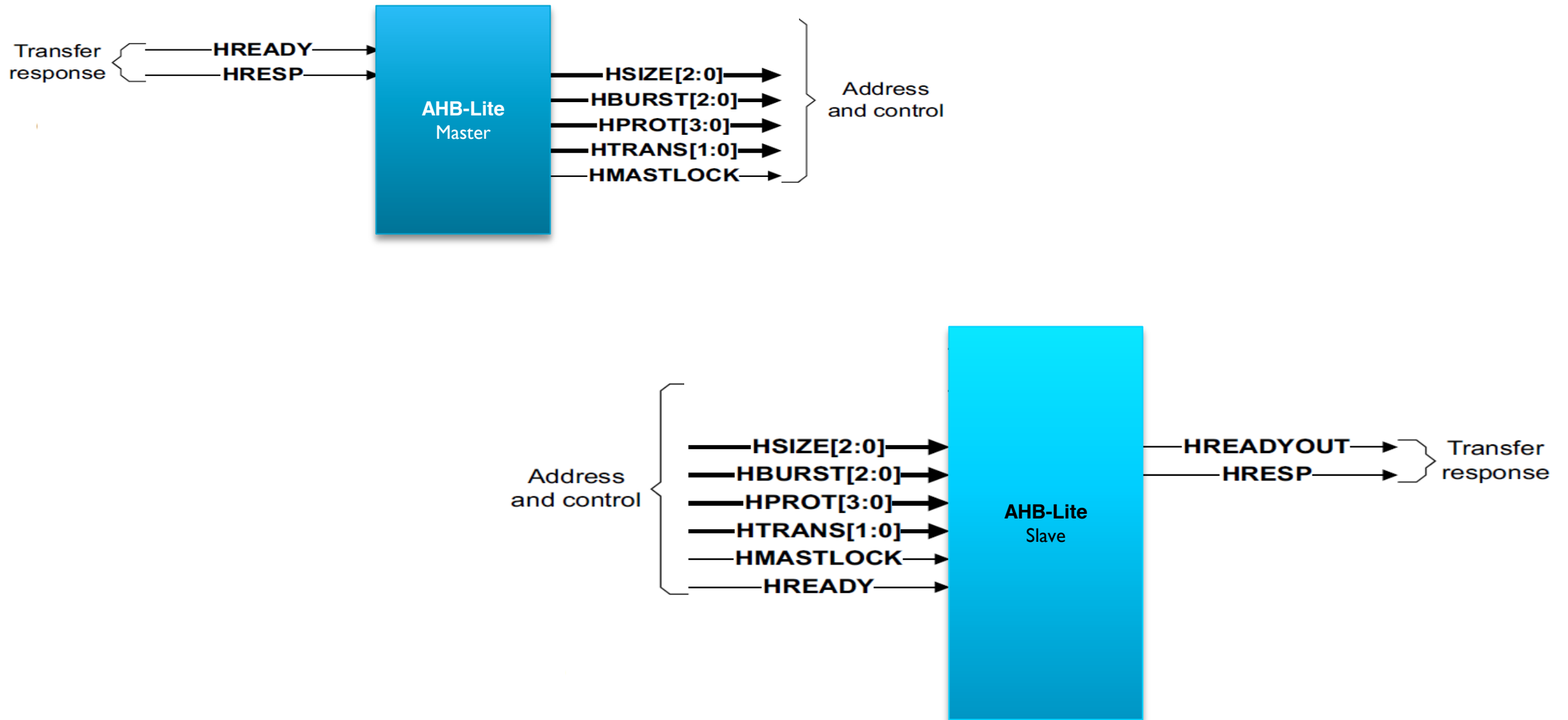


# AHB-Lite Master & Slave





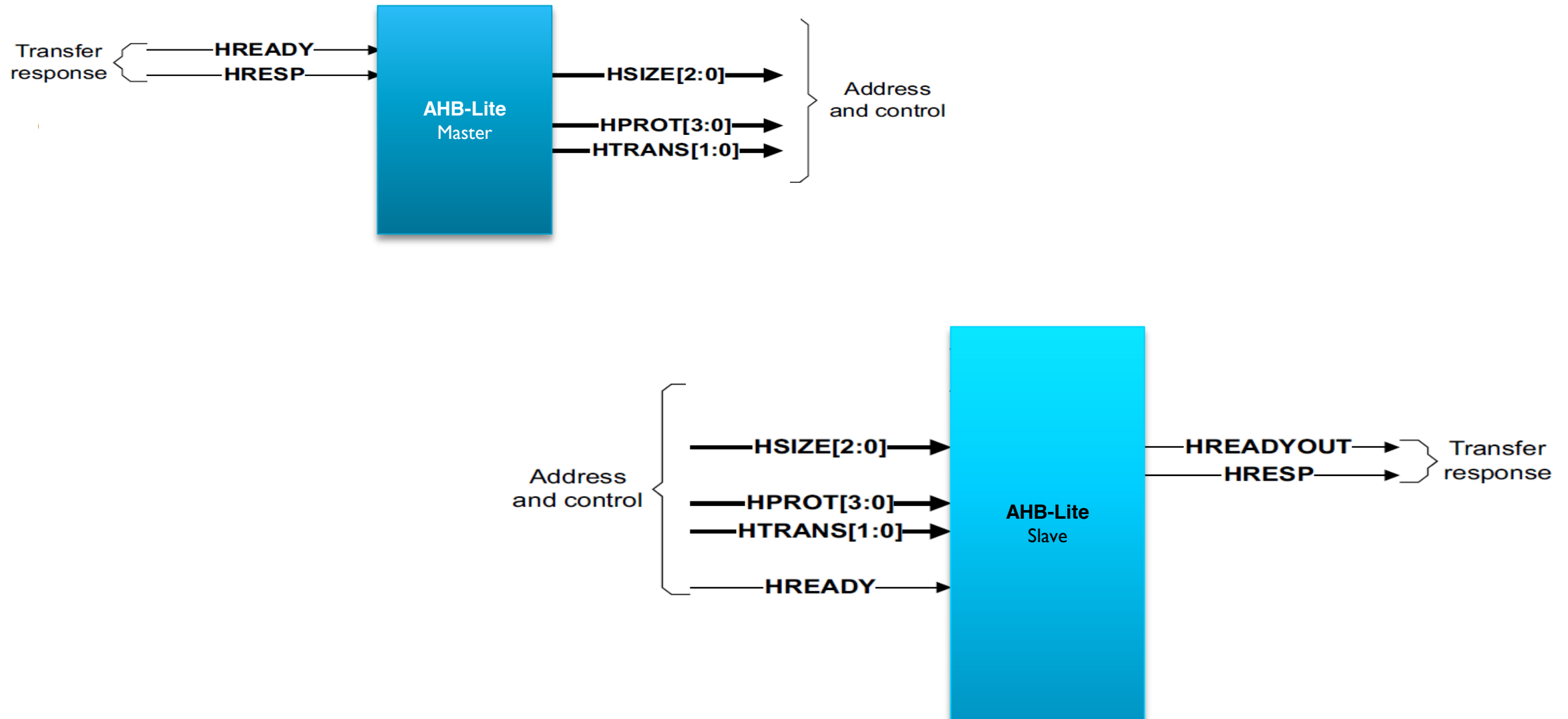
# AHB-Lite Master & Slave



# Cortex M0 doesn't speak the entire language !

- Cortex M0 does not support BURST transaction
  - HBURST[2:0] is always 3'b000
- Cortex M0 does not support locked transactions
  - HMASTLOCK is always 1'b0
- Cortex M0 issues only non-sequential transfers
  - HTRANS[1:0] is either 2'b00 (IDLE) or 2'b10 (Non Sequential)

# AHB-Lite Master & Slave



# HTRANS[1:0]

HTRANS	Type	Description
00	<b>IDLE</b>	Master does not wish to perform a transfer
01	<b>BUSY</b>	Bus Master is in the middle of a burst but cannot immediately continue with the next transfer
10	<b>NON-SEQ</b>	Indicates the first transfer of a burst or a single transfer
11	<b>SEQ</b>	The remaining transfers in the burst are sequential address steps from the previous transfer. Step size is that of data width of transfer (which is shown by HSIZE)

Cortex M0 Always generates NON-SEQ Transactions

# HSIZE[1:0]

Address-phase:		Data-phase:			
HSIZE [1:0]	HADDR [1:0]	HxDATA [31:24]	HxDATA [23:16]	HxDATA [15:8]	HxDATA [7:0]
00	00	-	-	-	Rd[7:0]
00	01	-	-	Rd[7:0]	-
00	10	-	Rd[7:0]	-	-
00	11	Rd[7:0]	-	-	-
01	00	-	-	Rd[15:8]	Rd[7:0]
01	10	Rd[15:8]	Rd[7:0]	-	-
10	00	Rd[31:24]	Rd[23:16]	Rd[15:8]	Rd[7:0]

# HPROT[3:0] Protection Signal Encoding

HPROT[3] Cacheable	HPROT[2] Bufferable	HPROT[1] Privileged	HPROT[0] Data/Opcode	Description
-	-	-	0	Opcode fetch
-	-	-	1	Data access
-	-	0	-	User access
-	-	1	-	Privileged access
-	0	-	-	Non-bufferable
-	1	-	-	Bufferable
0	-	-	-	Non-cacheable
1	-	-	-	Cacheable

# Transactions

Transaction	Access
HTRANS[1:0] = 2'b00	IDLE
HTRANS[1:0] = 2'b10	FETCH
HPROT[0] = 1'b0	
HSIZE[1:0] = 2'b10	
HWRITE = 1'b0	

## Instruction Fetch

## Data Access

Transaction	Access
HTRANS[1:0] = 2'b10	BYTE
HPROT[0] = 1'b1	
HSIZE[1:0] = 2'b00	
HTRANS[1:0] = 2'b10	HALF-WORD
HPROT[0] = 1'b1	
HSIZE[1:0] = 2'b01	
HTRANS[1:0] = 2'b10	WORD
HPROT[0] = 1'b1	
HSIZE[1:0] = 2'b10	

# Control Signals Recap

## HTRANS[1:0]

IDLE  
BUSY  
NONSEQ  
SEQ

## HSIZE[2:0]

Byte  
Halfword  
Word  
Doubleword  
...

## HBURST[2:0]

SINGLE  
INCR  
WRAP[4|8|16]  
INCR[4|8|16]

## HPROT[3:0]

Data/Opcode  
Privileged/user  
Bufferable  
Cacheable

## HMASTLOCK

UNLOCKED  
LOCKED

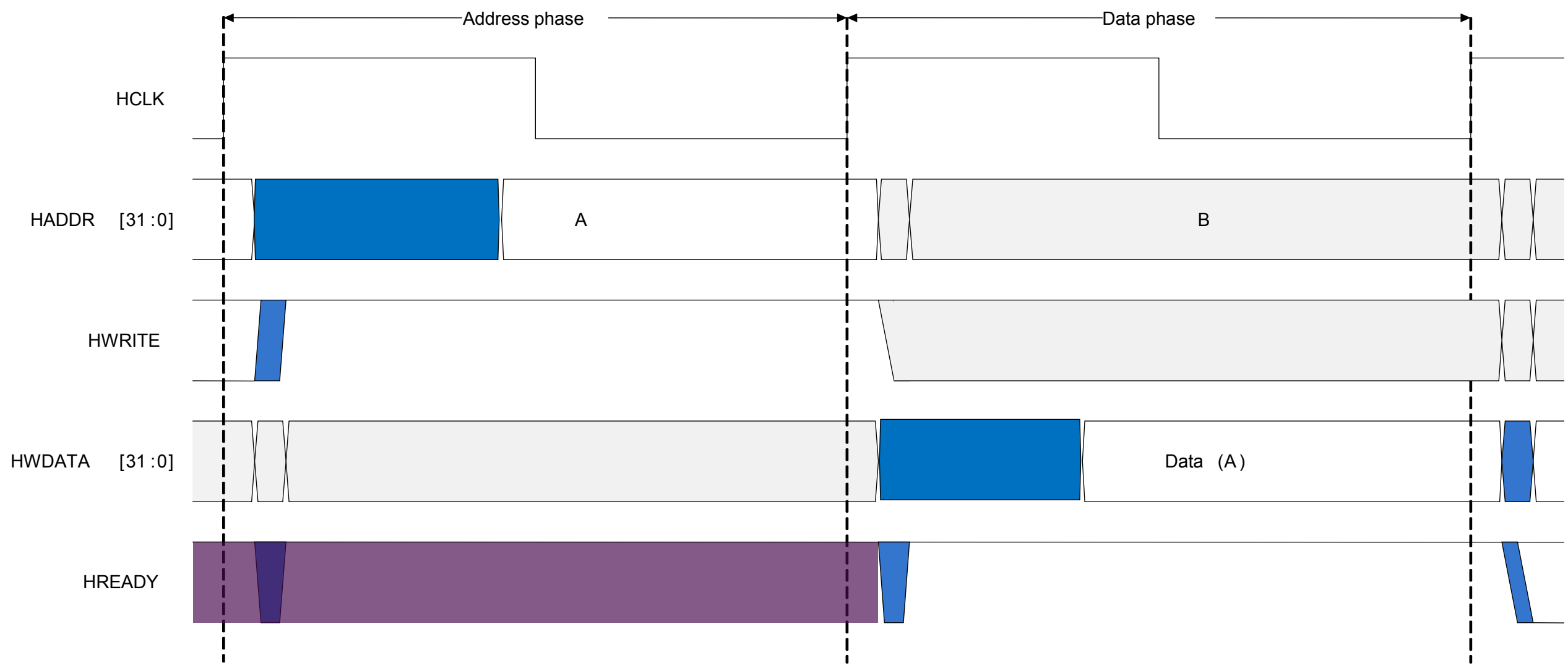


# Transfer Response Signals

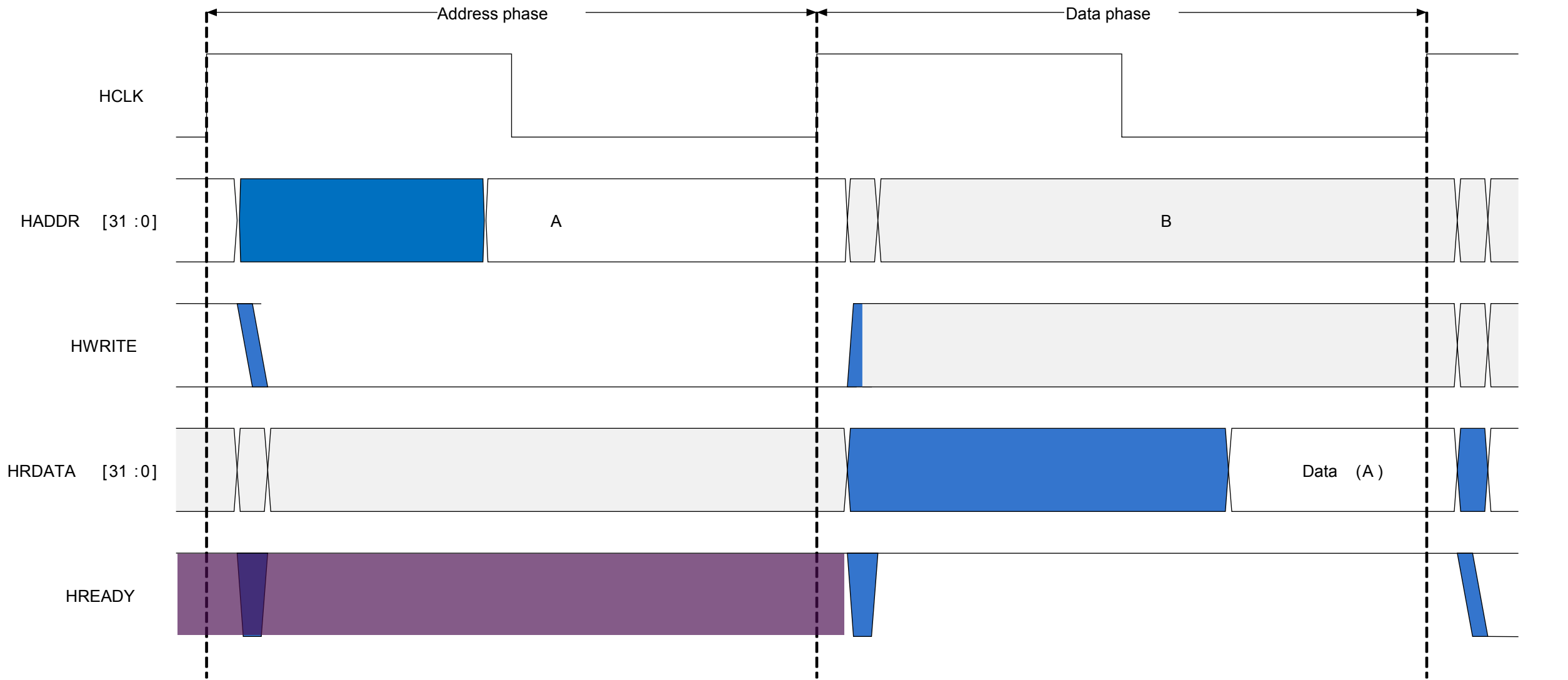
<b>HREADYOUT</b>	Multiplexor	When HIGH, the <b>HREADYOUT</b> signal indicates that a transfer has finished on the bus. This signal can be driven LOW to extend a transfer.
<b>HRESP</b>	Multiplexor	The transfer response, after passing through the multiplexor, provides the master with additional information on the status of a transfer. When LOW, the <b>HRESP</b> signal indicates that the transfer status is OKAY. When HIGH, the <b>HRESP</b> signal indicates that the transfer status is ERROR.

# AHB-Lite Transactions

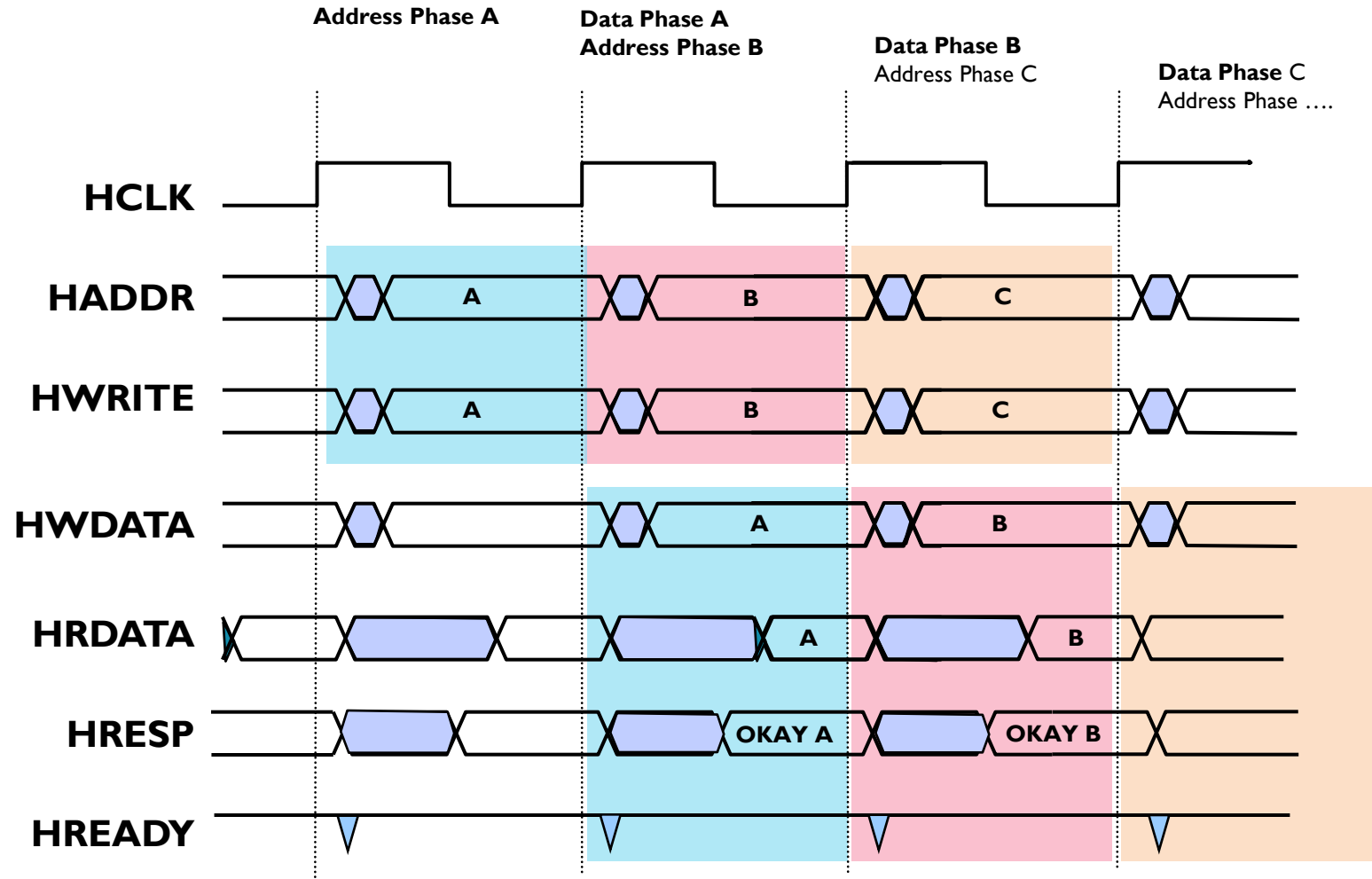
# Basic transfer - Write



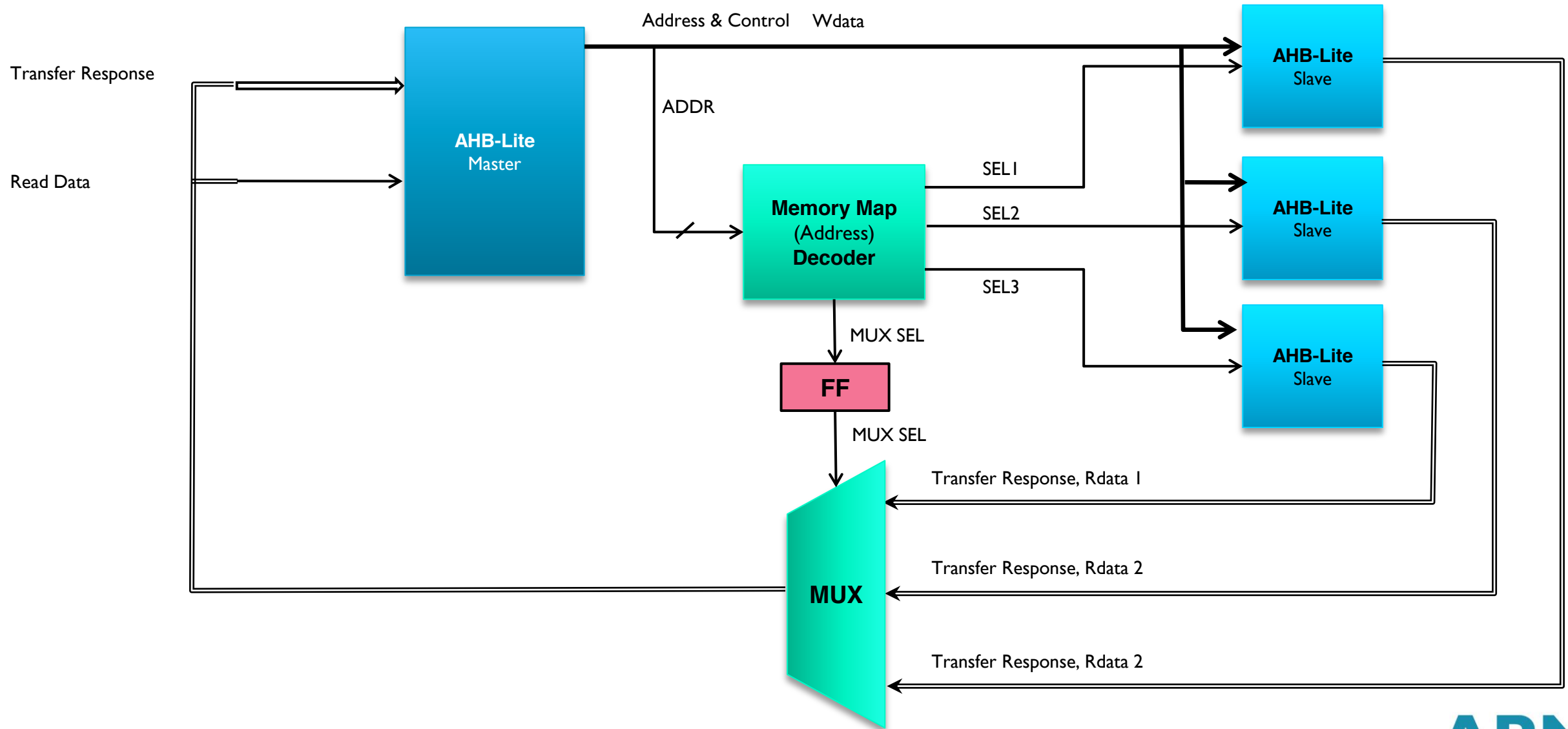
# Basic transfer - Read



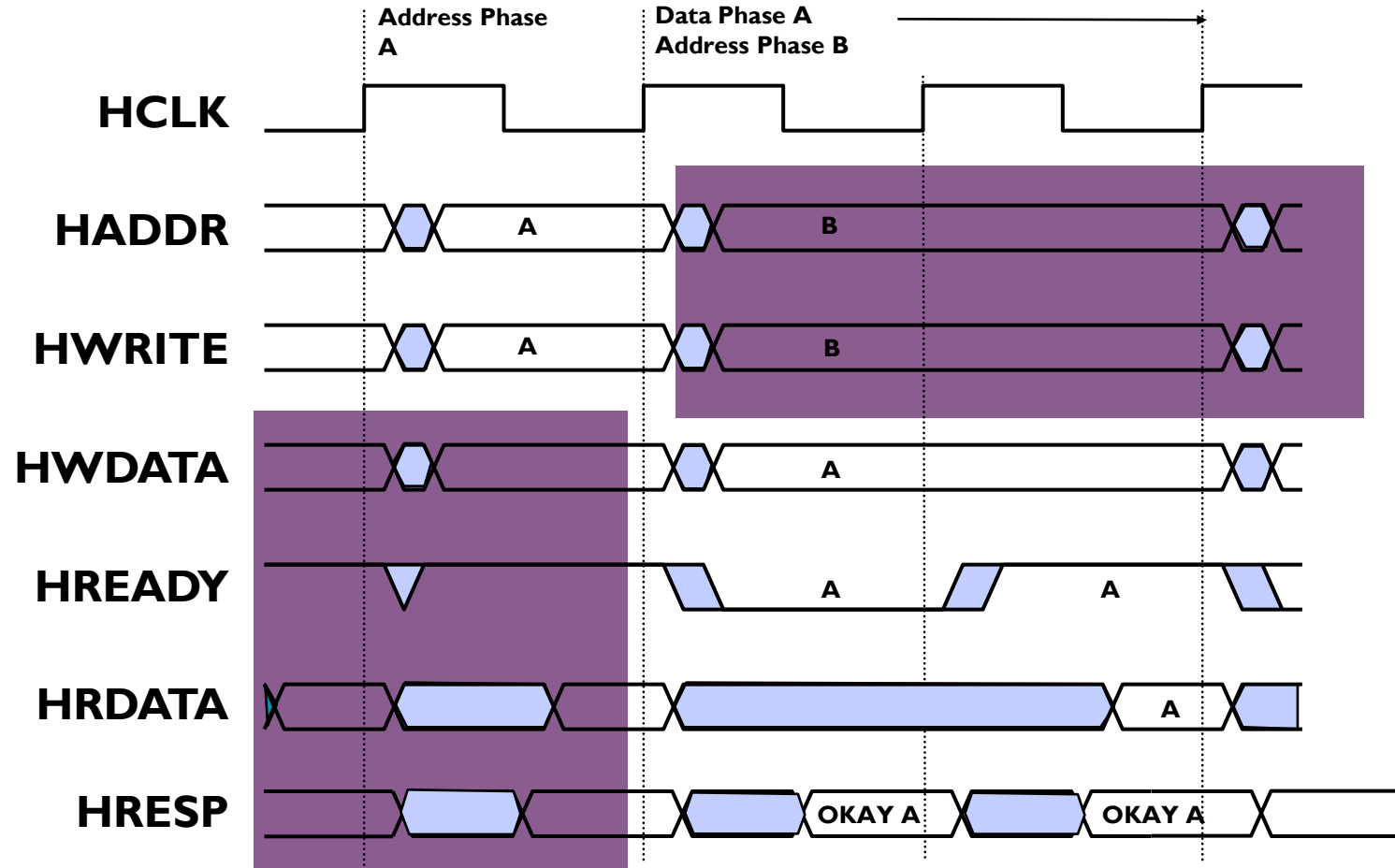
# AHB Pipelined Transaction



# Pipelined Operation

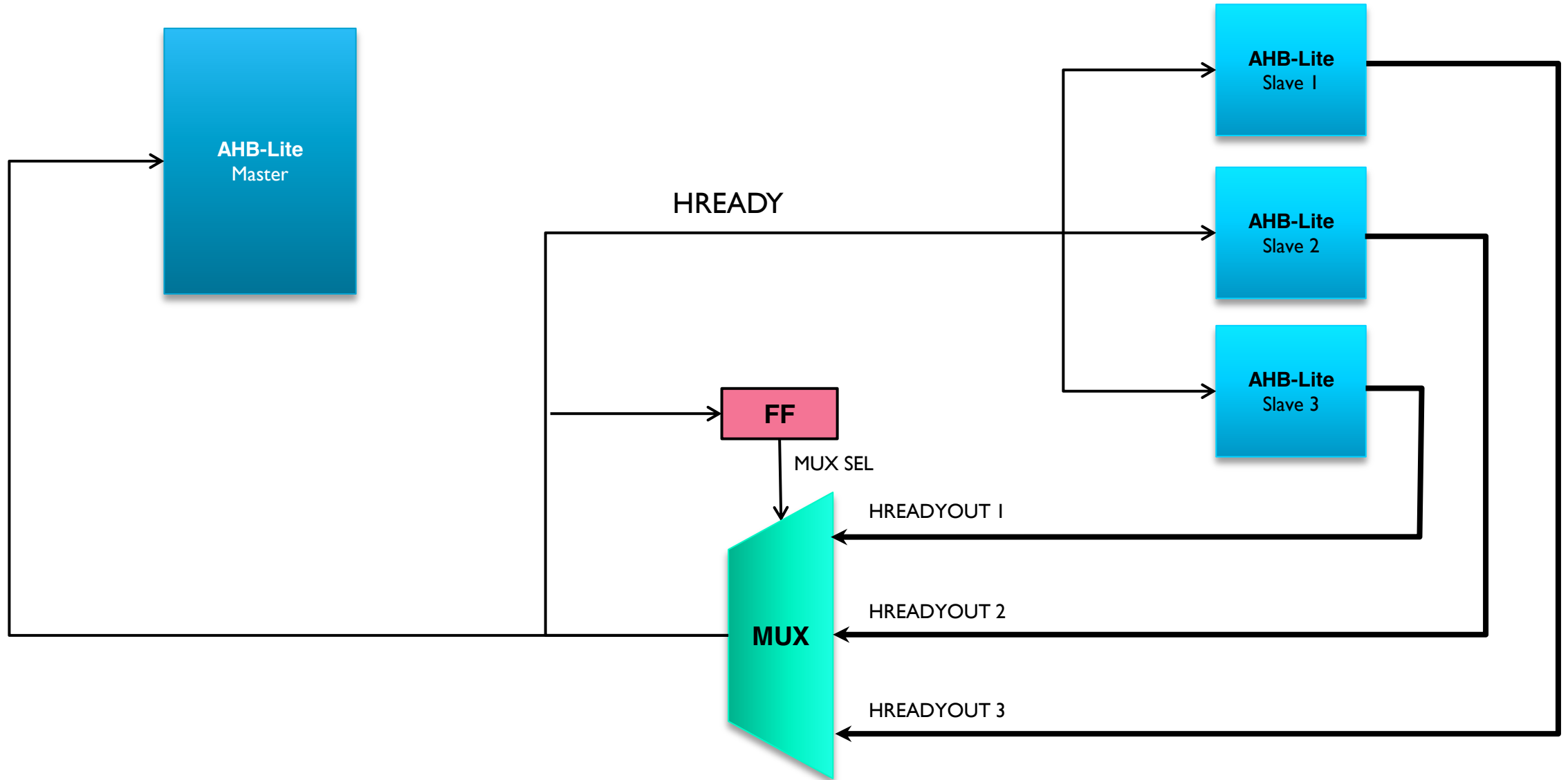


# AHB basic signal timing – Adding wait states



Master will extend Address Phase B

# HREADY (Inform all)



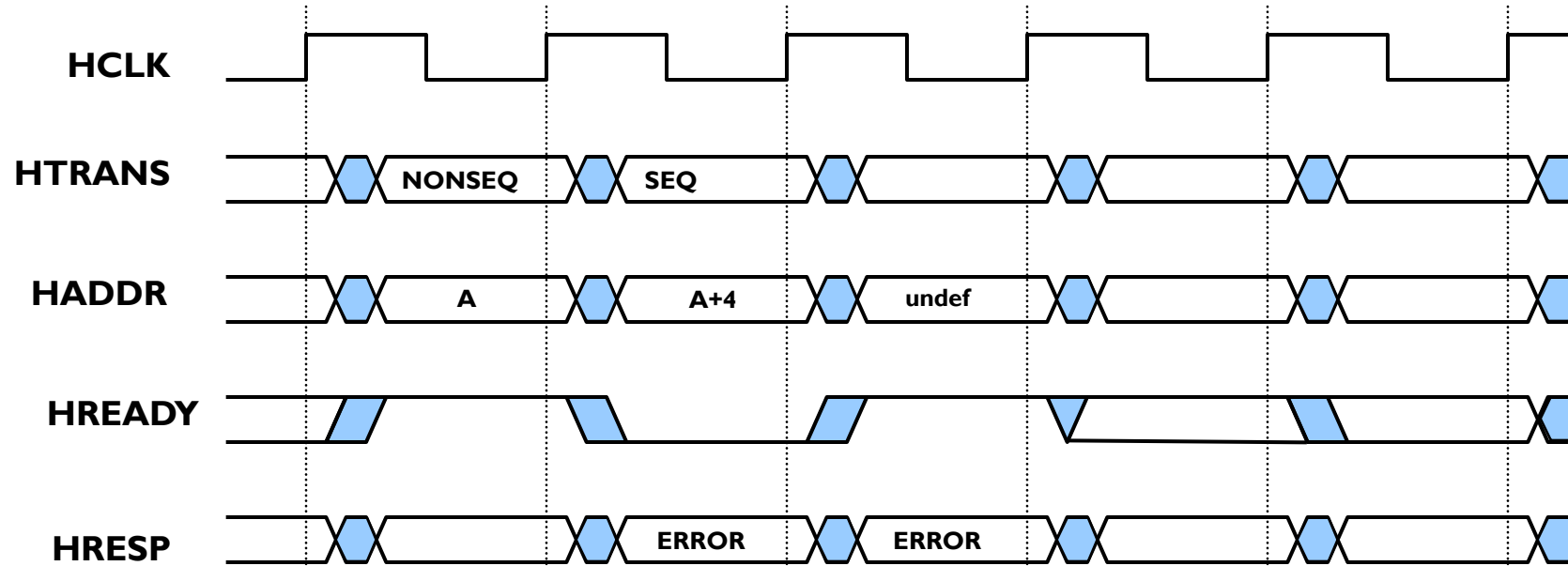


# HRESP – Slave Response

<u>HRESP</u>	<u>Event</u>	<u>Bus Master operation</u>
OKAY	Access completed normally	
ERROR	Slave aborts access, (2 cycle response)	Master has option of continuing or terminating a burst containing an ERROR

It is permissible to continuously drive HRESP Low in a system which does not wish to generate any errors.

# ERROR Response

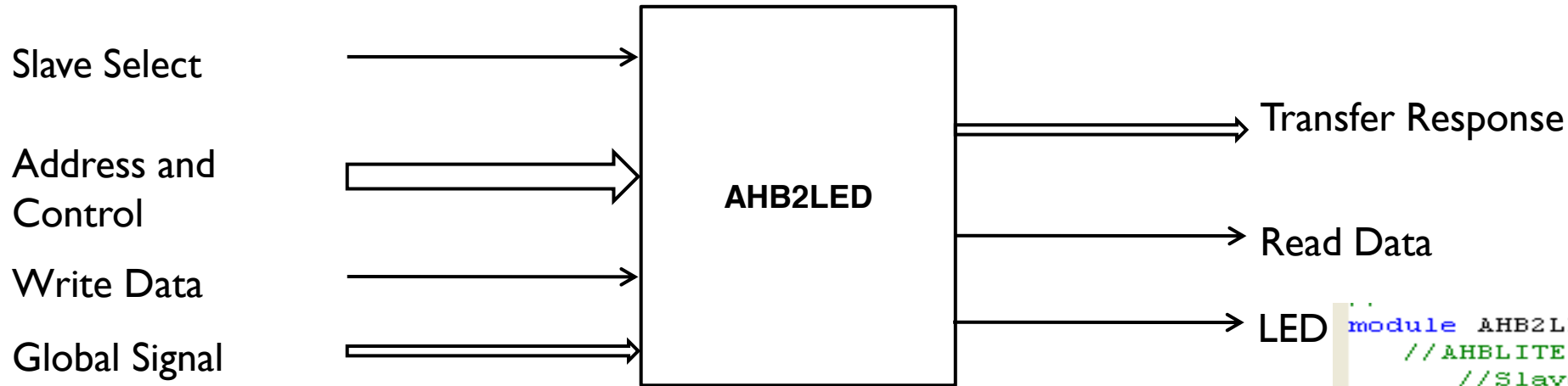


- If **HRESP = ERROR**, CM0-DS takes an exception and you should implement appropriate exception handler to catch the error

# A simple AHB-Lite Slave

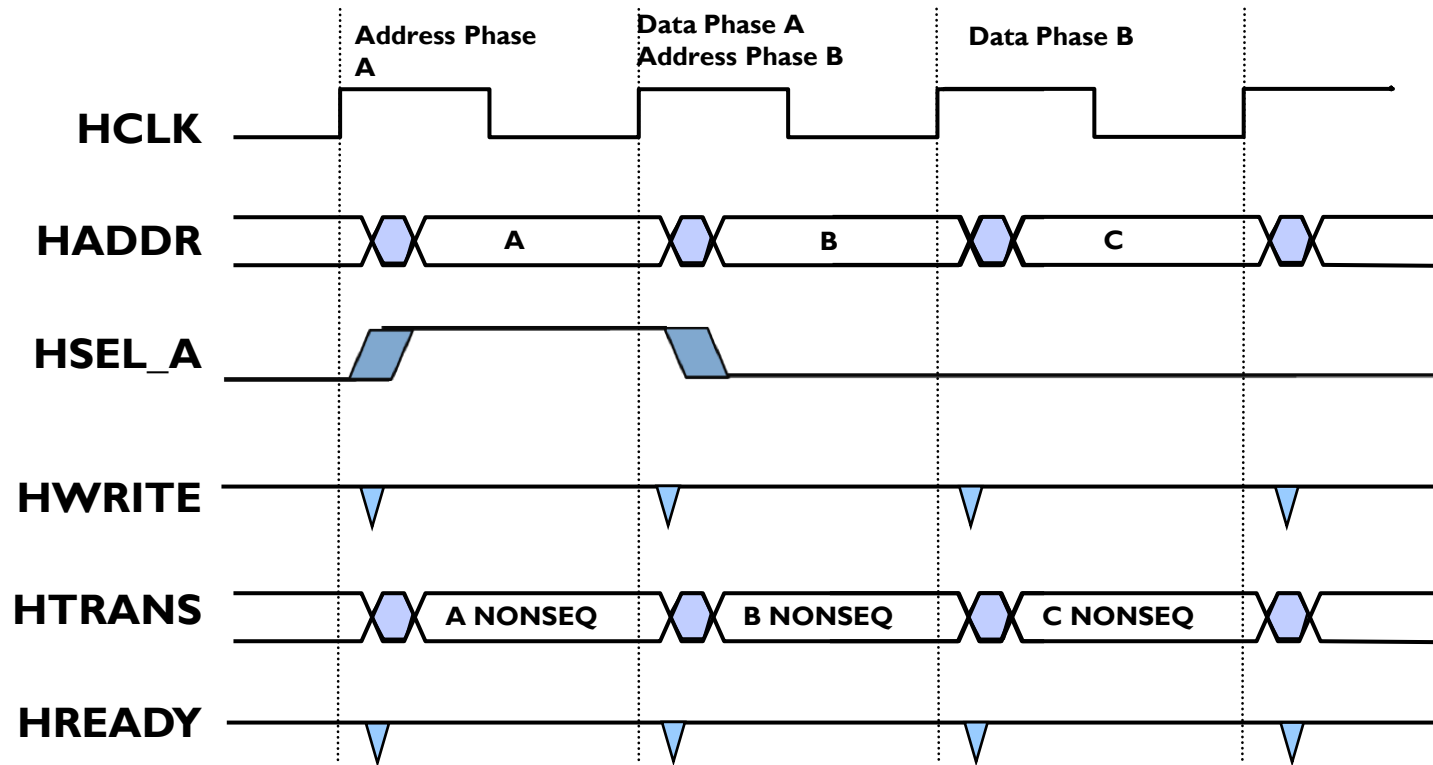
AHB2LED.v

# AHB2LED TOP LEVEL



```
module AHB2LED(  
    //AHBLITE INTERFACE  
    //Slave Select Signals  
    input wire HSEL,  
    //Global Signal  
    input wire HCLK,  
    input wire HRESETn,  
    //Address, Control & Write Data  
    input wire HREADY,  
    input wire [31:0] HADDR,  
    input wire [1:0] HTRANS,  
    input wire HWRITE,  
    input wire [2:0] HSIZE,  
  
    input wire [31:0] HWDATA,  
    // Transfer Response & Read Data  
    output wire HREADYOUT,  
    output wire [31:0] HRDATA,  
  
    //LED Output  
    output reg [7:0] LED  
);
```

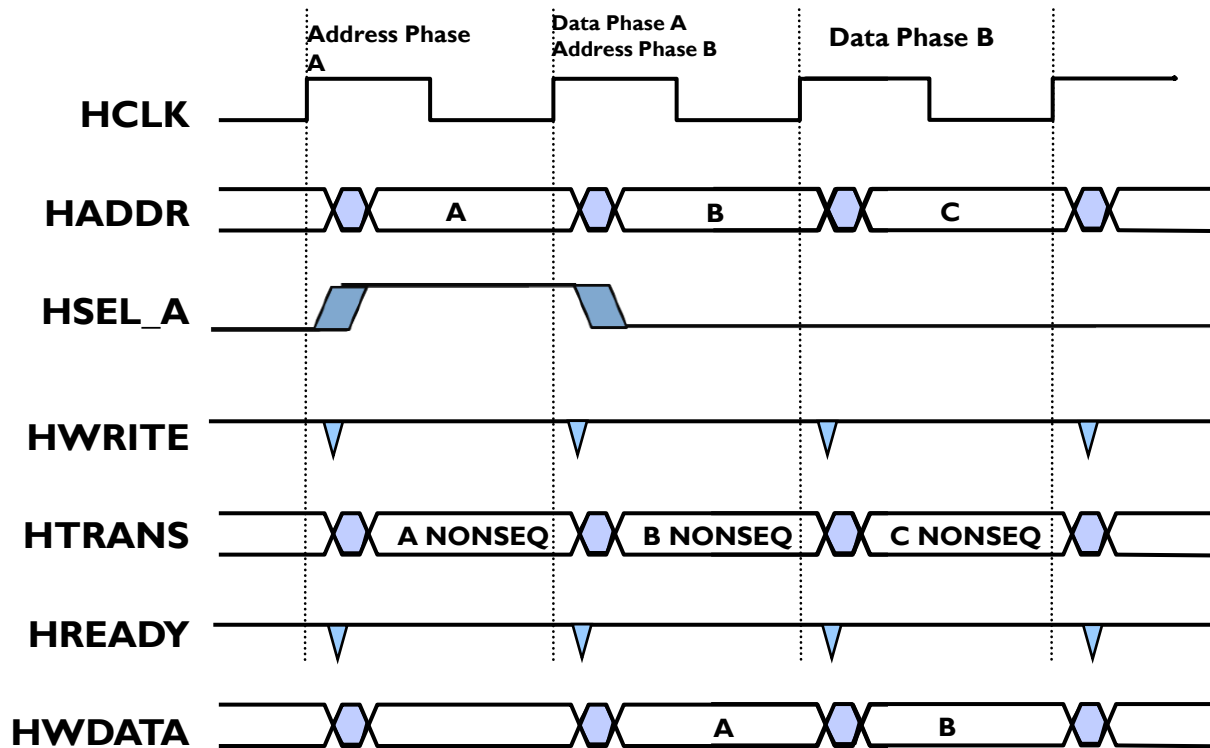
# Sampling Address & Control



```
//Address Phase Sampling Registers
reg rHSEL;
reg [31:0] rHADDR;
reg [1:0] rHTRANS;
reg rHWRITE;
reg [2:0] rHSIZE;

//Address Phase Sampling
always @(posedge HCLK or negedge HRESETn)
begin
    if(!HRESETn)
    begin
        rHSEL    <= 1'b0;
        rHADDR    <= 32'h0;
        rHTRANS    <= 2'b00;
        rHWRITE    <= 1'b0;
        rHSIZE    <= 3'b000;
    end
    else if(HREADY)
    begin
        rHSEL    <= HSEL;
        rHADDR    <= HADDR;
        rHTRANS    <= HTRANS;
        rHWRITE    <= HWRITE;
        rHSIZE    <= HSIZE;
    end
end
end
```

# Sampling Address & Control



```

67 //Data Phase data transfer
68 always @(posedge HCLK or negedge HRESETn)
69 begin
70     if(!HRESETn)
71         LED <= 8'b0000_0000;
72     else if(rHSEL & rHWRITE & rHTRANS[1])
73         LED <= HWDATA[7:0];
74 end
75
76 //Transfer Response
77 assign HREADYOUT = 1'b1; //Single cycle Write & Read. Zero Wait state operations
78
79 //Read Data
80 assign HRDATA = {24'h0000_00,LED};
81

```

```

//Address Phase Sampling Registers
reg rHSEL;
reg [31:0] rHADDR;
reg [1:0] rHTRANS;
reg rHWRITE;
reg [2:0] rHSIZE;

//Address Phase Sampling
always @(posedge HCLK or negedge HRESETn)
begin
    if(!HRESETn)
    begin
        rHSEL    <= 1'b0;
        rHADDR   <= 32'h0;
        rHTRANS  <= 2'b00;
        rHWRITE  <= 1'b0;
        rHSIZE   <= 3'b000;
    end
    else if(HREADY)
    begin
        rHSEL    <= HSEL;
        rHADDR   <= HADDR;
        rHTRANS  <= HTRANS;
        rHWRITE  <= HWRITE;
        rHSIZE   <= HSIZE;
    end
end

```

# Lab

- Analyse the AHB2LED.v file provided
- In the next lab, we will look into system integration, simulation and implementation of a complete AHB-Lite System using Cortex M0 Design Start core