RISC-V

System-on-Chip Design

Harris, Stine, Thompson, & Harris

Chapter 9:

Bus Interface

Chapter 9 :: Topics

Privileged Operations

9.1 Principles

9.2 AMBA Buses: APB, AHB, AXI

9.3 Other SoC Buses (not covered in slides)

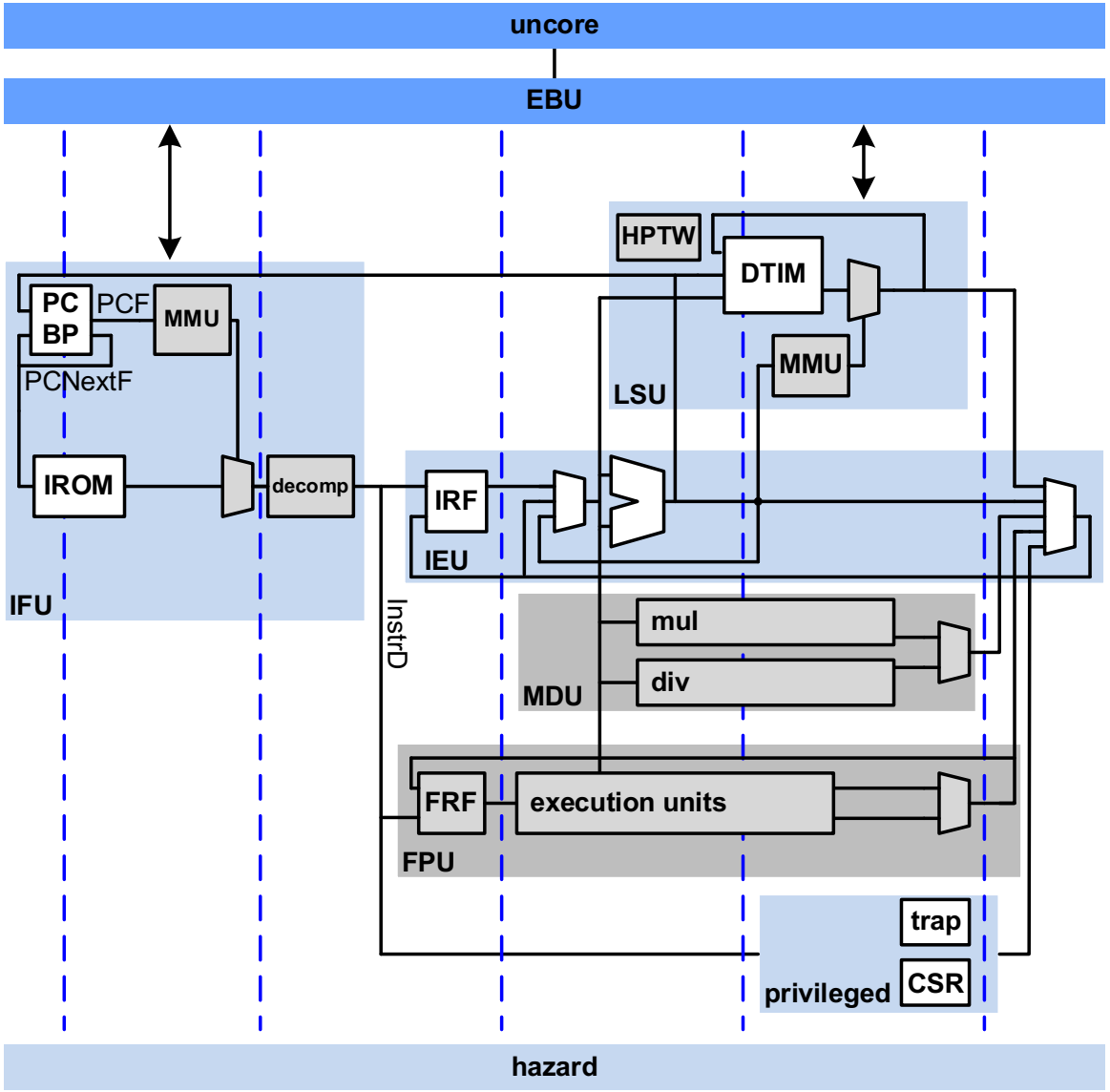
9.4 Test Plan (not covered in slides)

9.5 Wally Implementation

Principles

- **Bus interfaces:**
  - Connect components: e.g., cores, memories, peripherals
  - Standard protocol
    - Well-defined interaction
    - Makes it easy to attach off-the-shelf peripherals

Wally SoC with EBU & Uncore



EBU: External bus unit

Uncore:

- Everything except the processor core
- Includes: bus interfaces, memory, peripherals

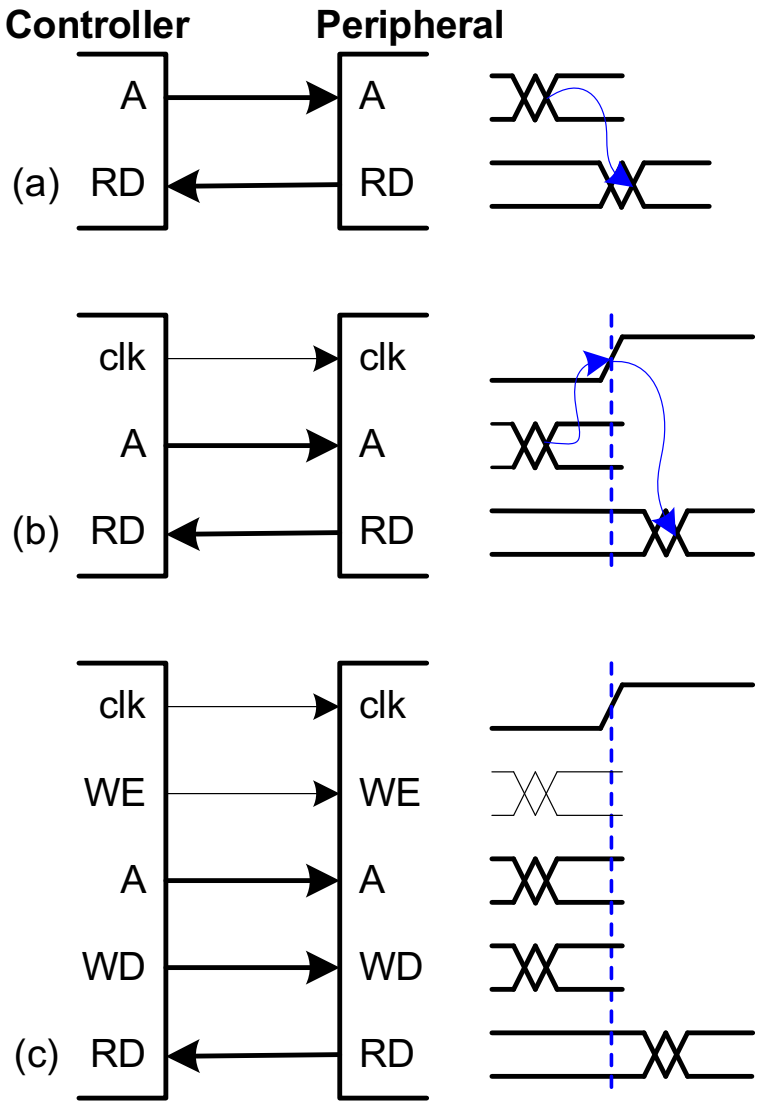
Bus Principles

- **Bus connects:**
  - **Controllers:** e.g., a processor
  - **Peripherals:**
    - Memory
    - I/O device: timer, interrupt controller, general-purpose I/O (GPIO), serial port, etc.

Peripherals are also called **responders** or **devices**.

- **Interactions:**
  - **Controller sends:**
    - Address: indicates which device to access (memory, peripheral device, etc.)
    - Data: read from/written to peripheral
  - **Peripheral sends:**
    - Data: in response to controller request

Simple Buses



Asynchronous Read-Only

A: Address

RD: Read data

Synchronous Read-Only

A: Address (sets up before clock edge)

RD: Read data (returned after clock edge)

Synchronous Read/Write

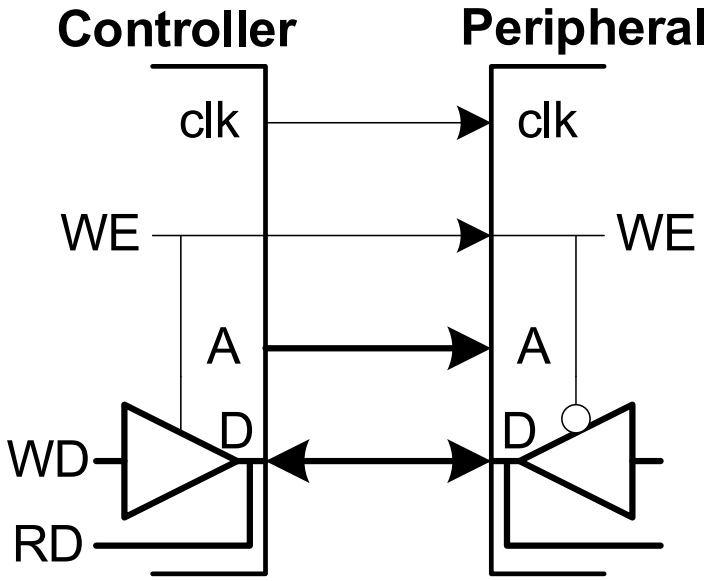
A: Address (sets up before clock edge)

RD: Read data (returned after clock edge)

WE: Write enable

WD: Write data (sets up before clock edge)

Bidirectional Data Port

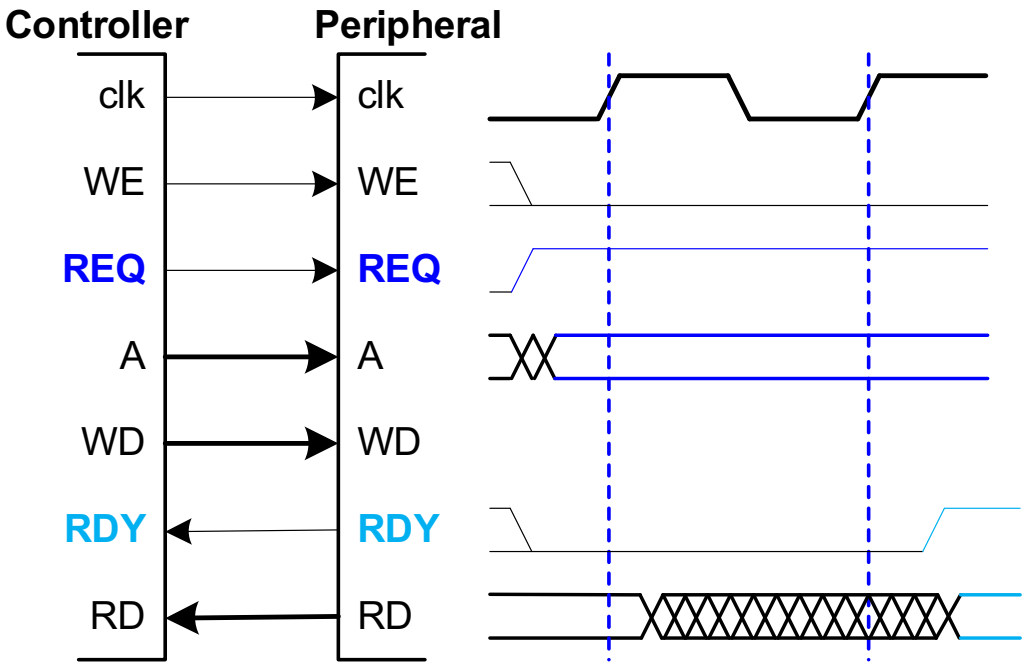


Synchronous Read/Write with Bidirectional Data Port

A: Address (sets up before clock edge)
D: Bidirectional data
WE: Write enable

Requires **tristate** so that only the controller or peripheral drives D at once.

Handshaking



Handshaking:

- Request/Acknowledge
- Saves power because controller only needs bus sometimes

REQ:

Controller asserts Request (REQ) when it has put a valid address on bus

ACK/RDY:

Bus asserts Acknowledge/Ready (ACK/RDY) when transfer is complete

Multiple Peripherals

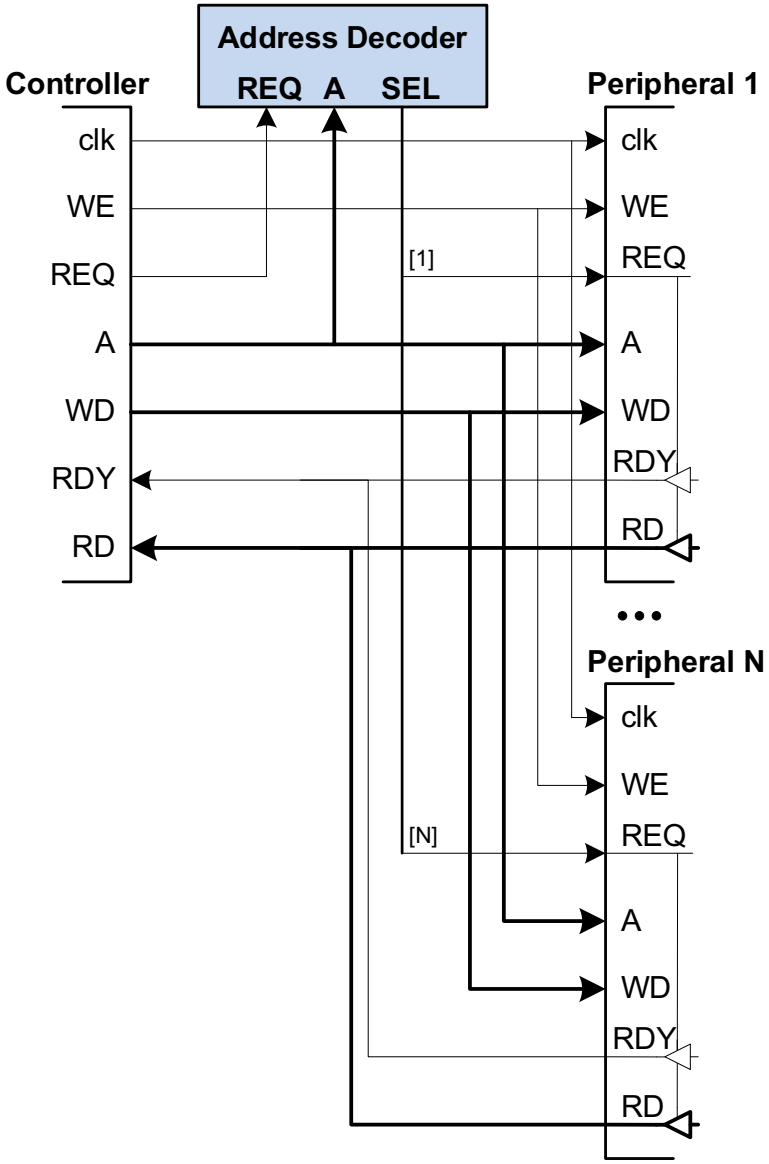
Address Decoder:

- Distinguishes which peripheral is being accessed
- Generates write enable signals

Multiplexer:

- Distinguishes which peripheral is being read
- Multiplexes read data signals

Multiple Peripherals: Shared Bus

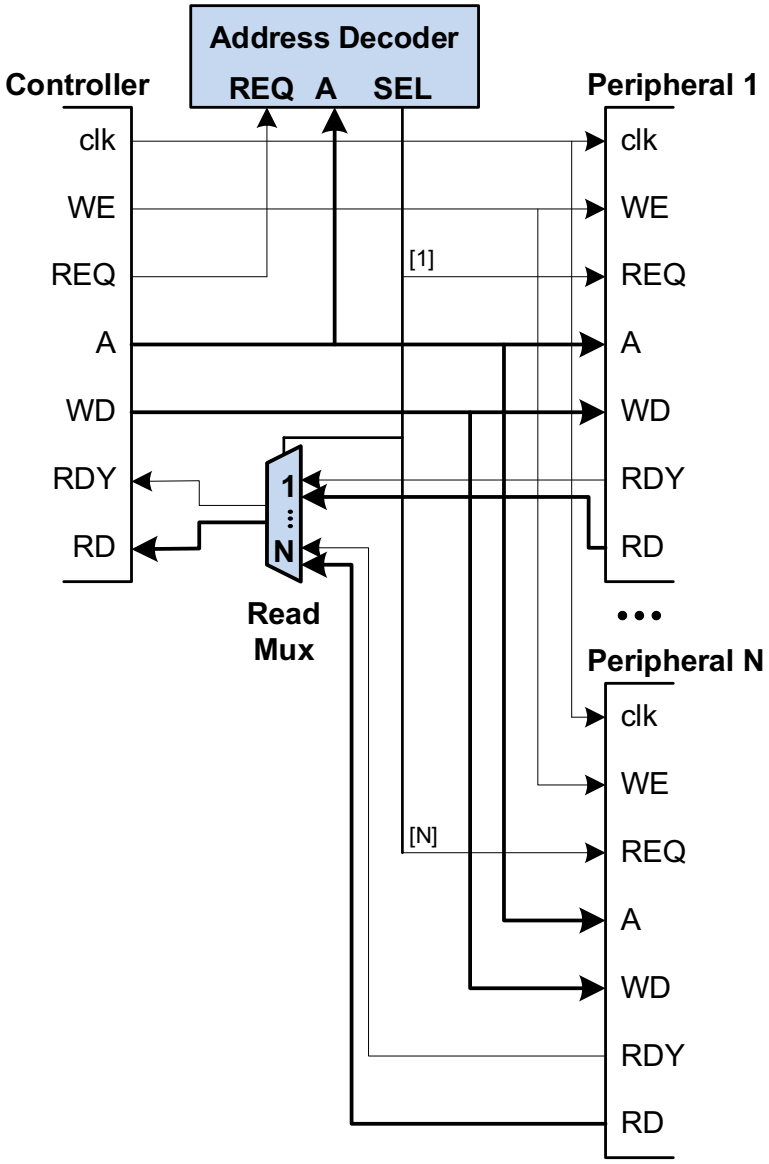


Address Decoder:

- Generates request (**REQ**) signals to peripherals depending on address (A)
- Only one peripheral can drive ready (**RDY**) and read data (**RD**) signals at once

Shared buses are typically **slower** than point-to-point buses, so they are typically not used in SoCs.

Multiple Peripherals: Point-to-Point



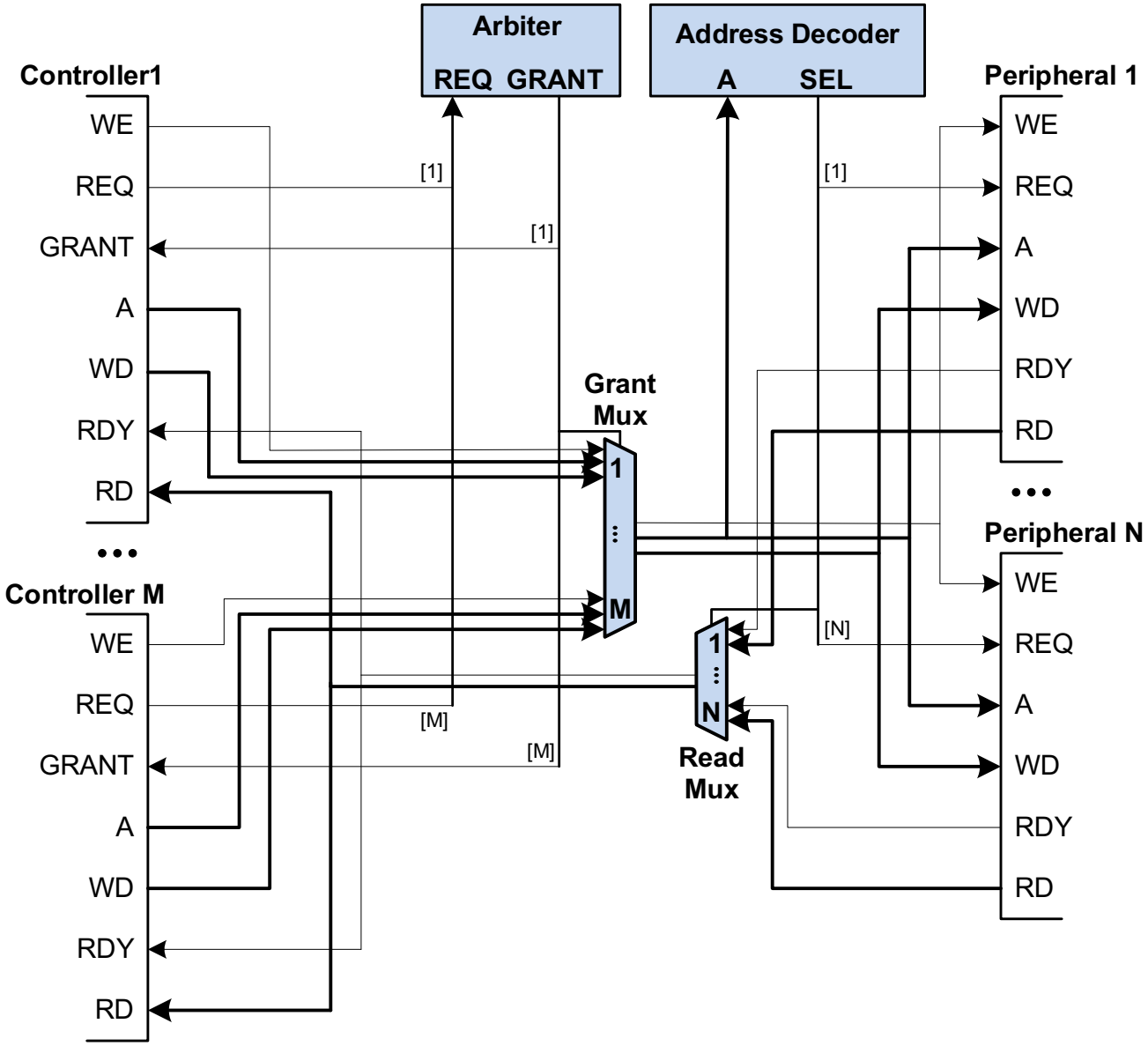
Address Decoder:

- Generates request (**REQ**) signals to peripherals depending on address (A)
- Generates **select signal** for read multiplexer (Read Mux)

Read Multiplexer (Read Mux):

- Selects one peripheral to read from based on the address being read
- Multiplexes ready (RDY) and read data (RD) signals from multiple peripherals

Multiple Controllers & Peripherals



Arbiter:

- Arbitrates between controllers and grants bus access to controllers

Grant Mux:

- Grants request access to one controller at a time

Chapter 9: Bus Interface

AMBA Buses

AMBA Buses

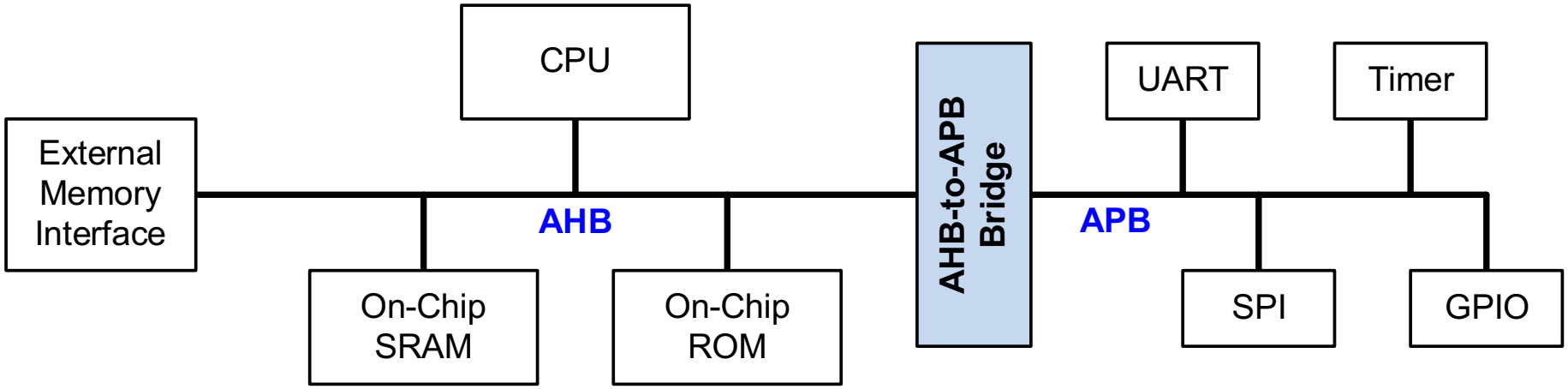
AMBA: ARM's Advanced Microcontroller Bus Architecture

- Open-standard on-chip buses
- Used in ASICs and FPGAs since 1996

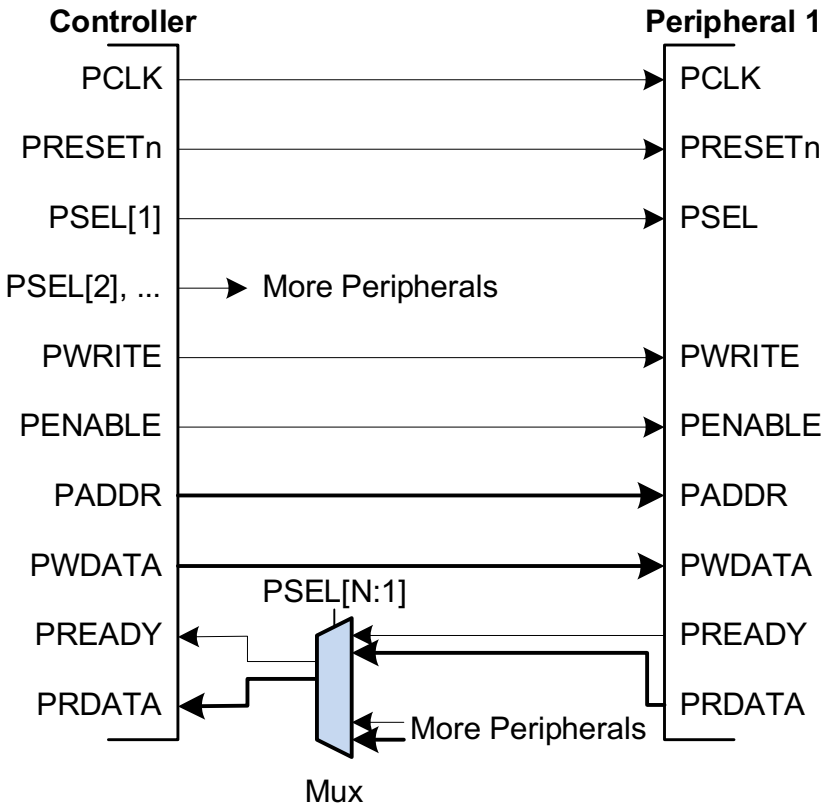
AMBA Flavors

- APB (Advanced Peripheral Bus): Simple unpipelined
- AHB (Advanced High-performance Bus): Pipelined
- AXI (Advanced eXtensible Interface): Multichannel, high-speed

Example System: AHB to APB Bridge



APB



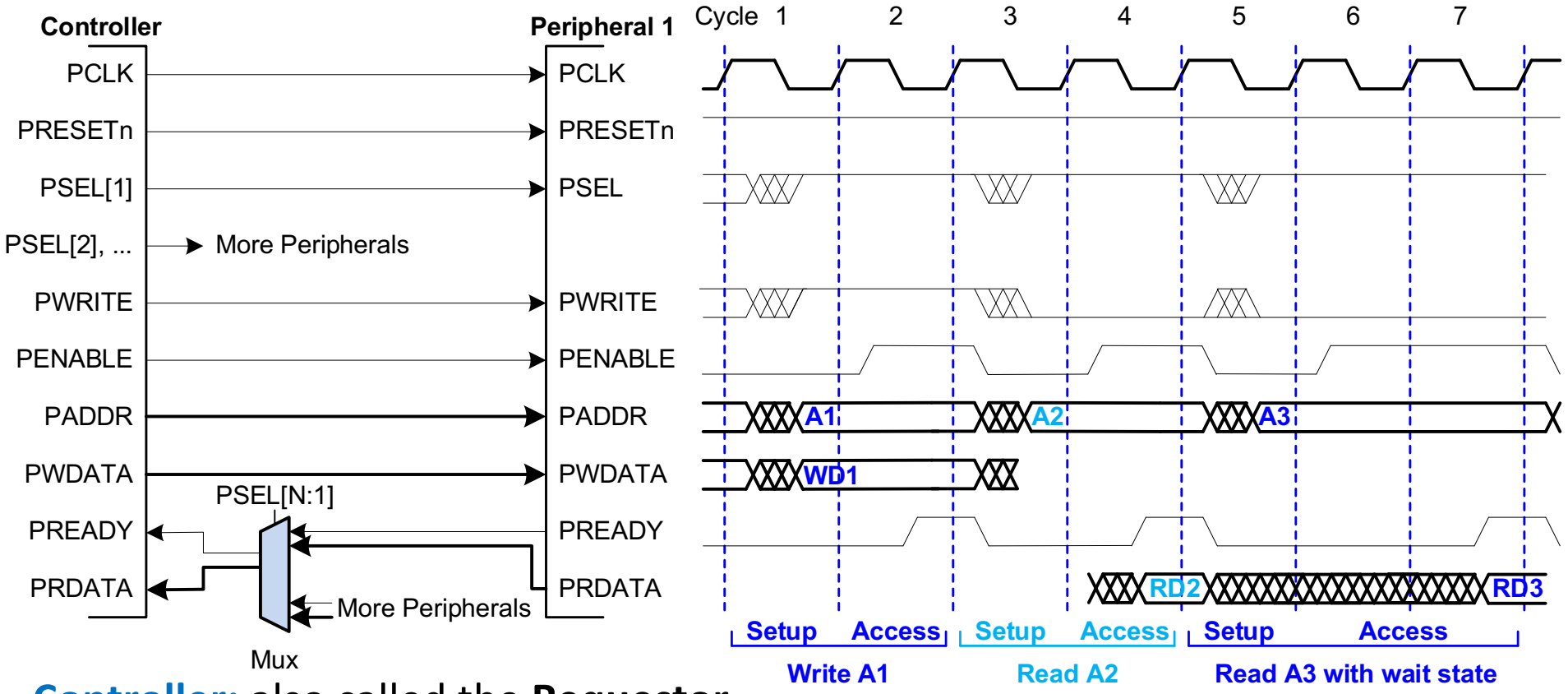
Controller: also called the **Requestor**

Signals: similar to generic names, with **P prefix**

PRESETn: asserts to **reset all peripherals**

Address Decoder: generates **PSEL** signals, is **part of the Controller**

APB



Controller: also called the **Requestor**

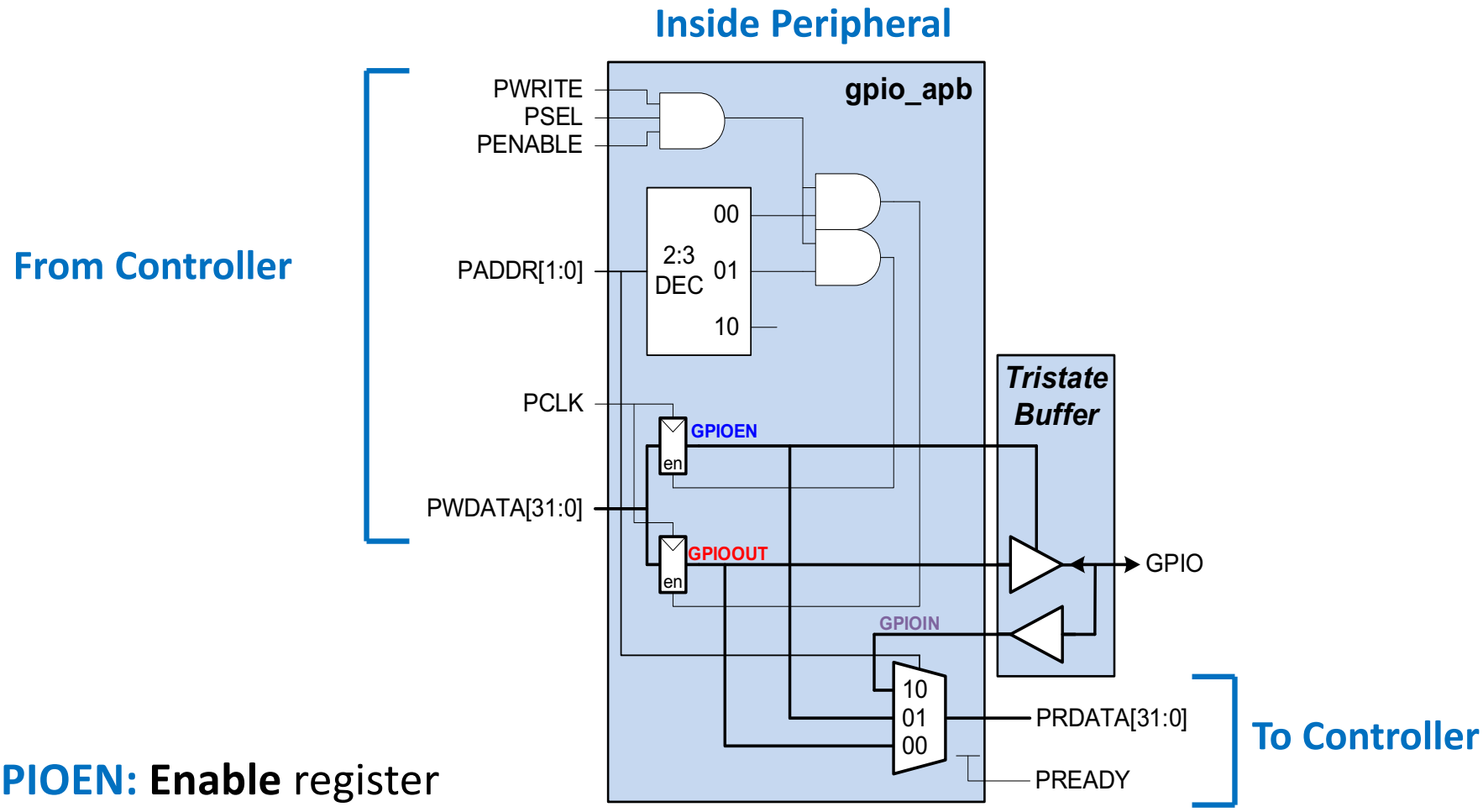
Signals: similar to generic names, with **P prefix**

PRESETn: asserts to **reset all peripherals**

Address Decoder: generates **PSEL** signals, is **part of the Controller**

Two Phases: **Setup** (Address) and **Access** (Data Read/Write)

APB: General-Purpose I/O (GPIO)



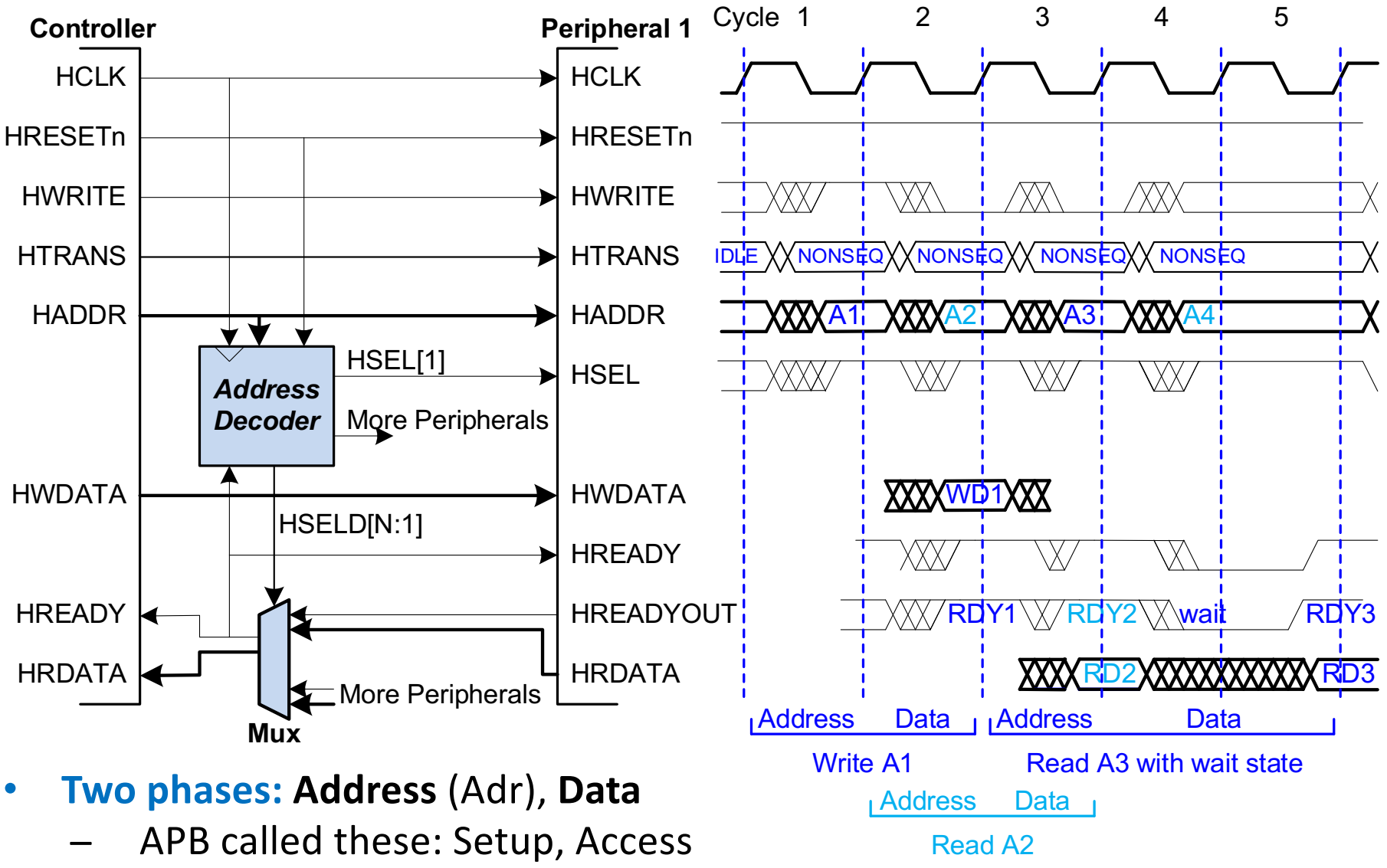
GPIOEN: Enable register

GPIOOUT: Output register

GPIOIN: Input register

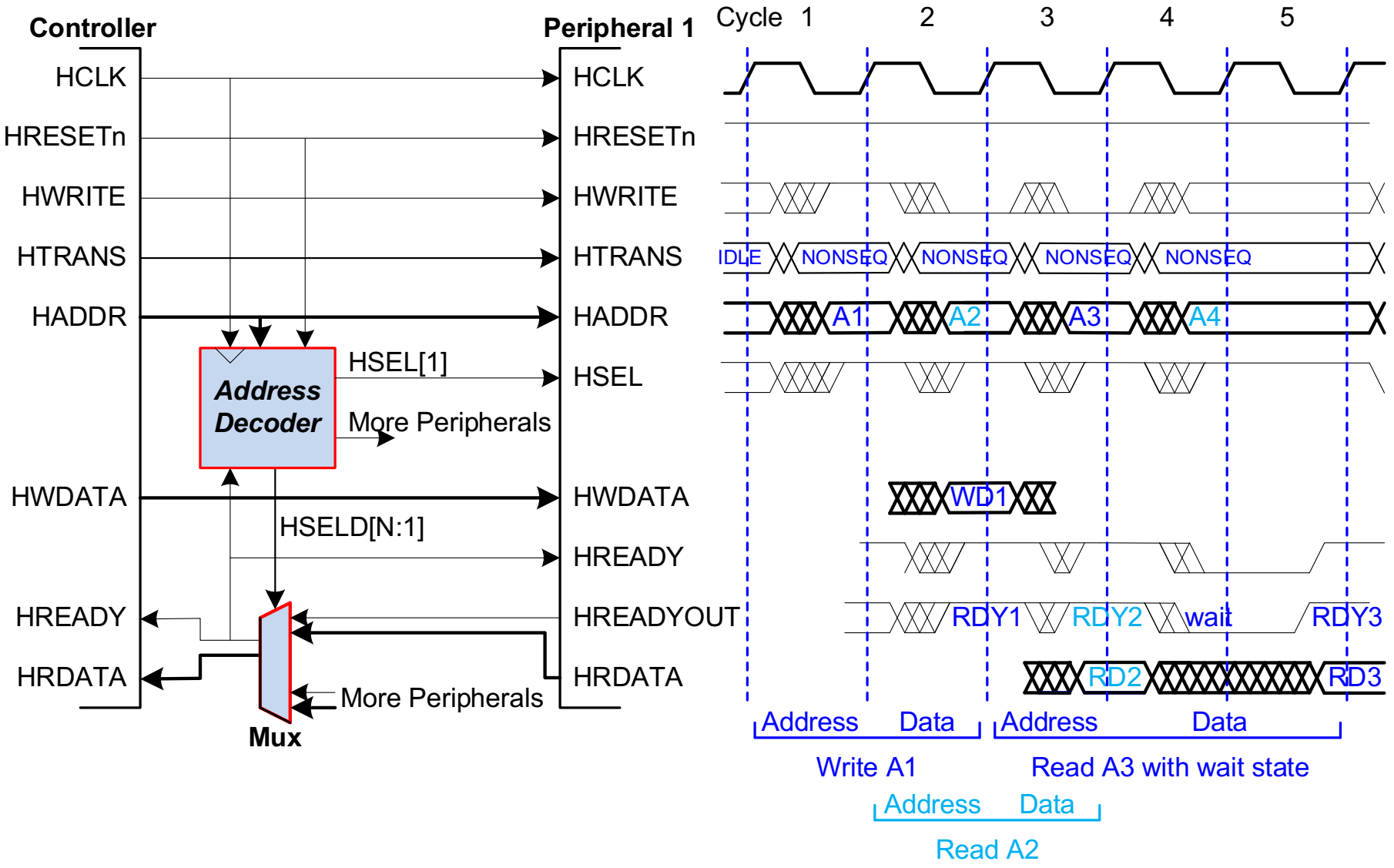
Mux: Sends input, output, or enable data to Controller

AHB

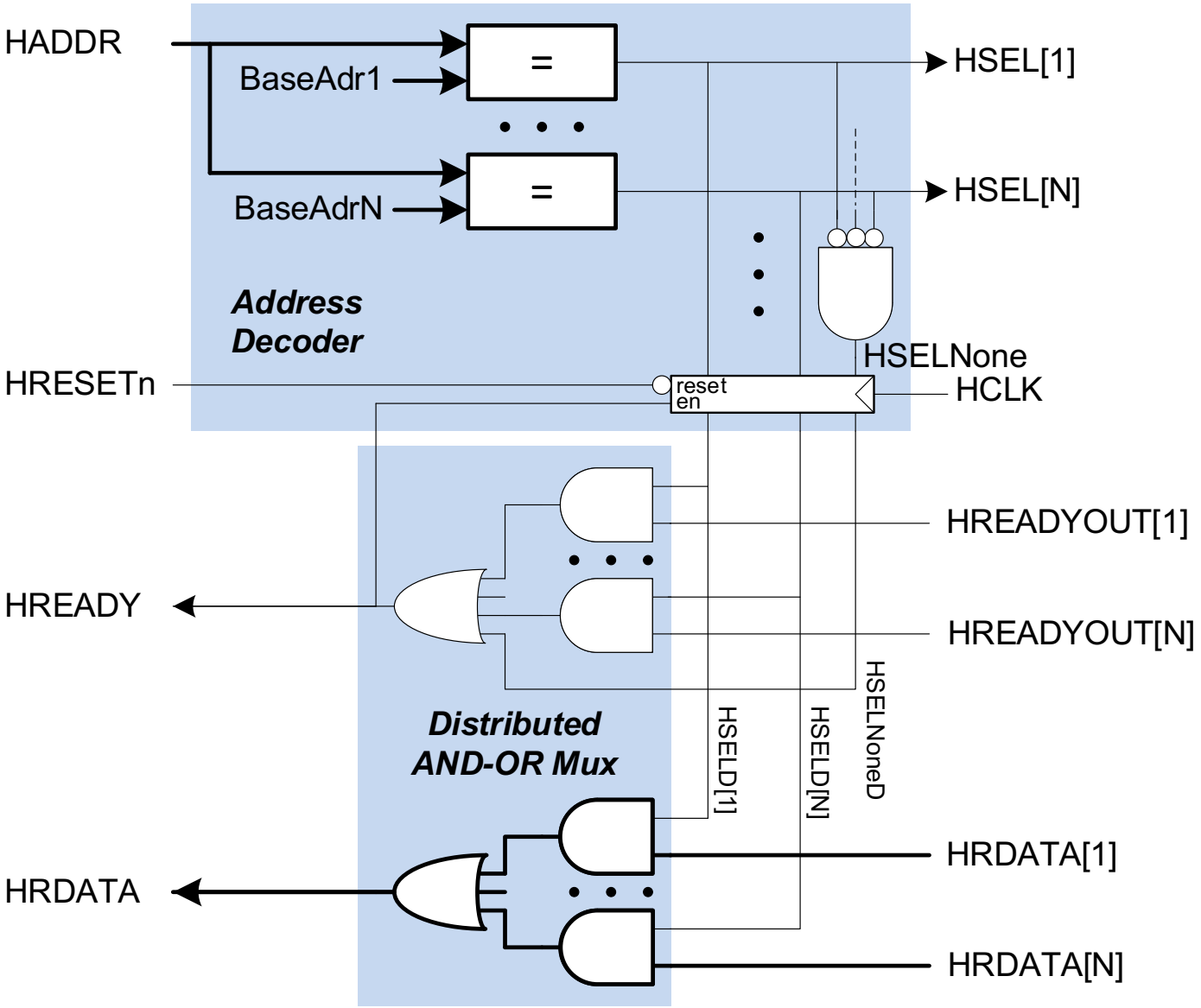


- **Two phases: Address** (Adr), **Data**
  - APB called these: Setup, Access
- **Note: HWRITE** occurs in Address phase

AHB Address Decoder & Mux

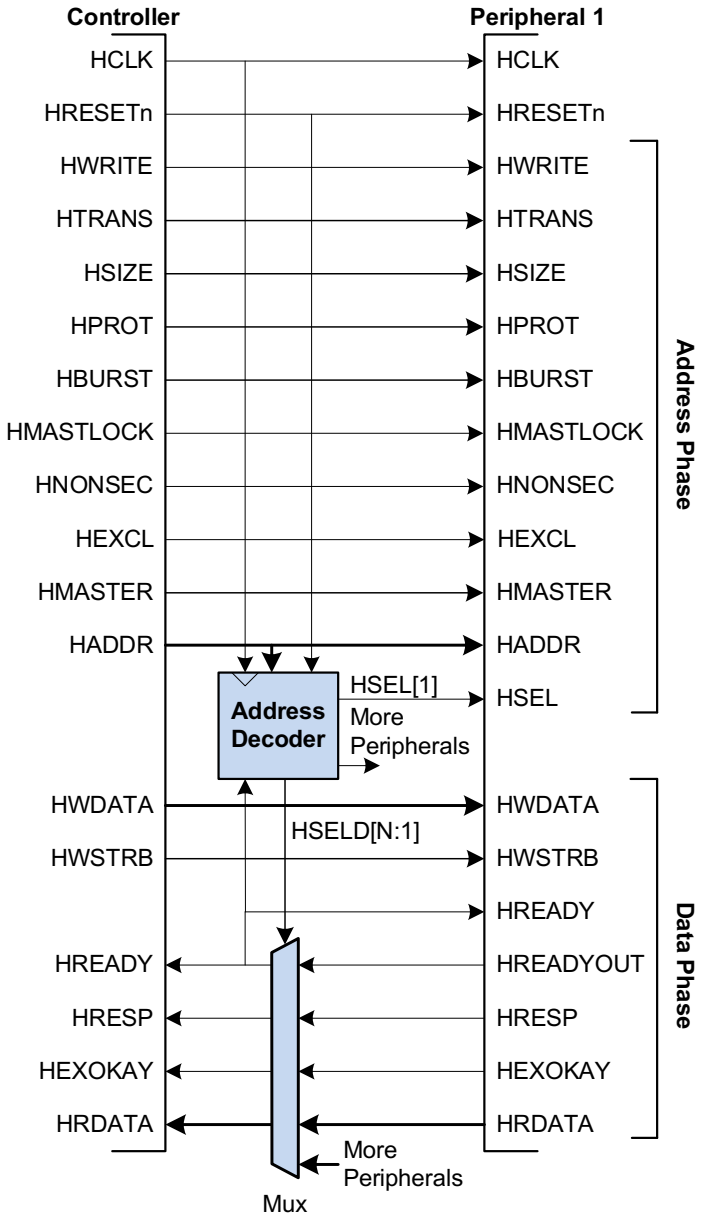


AHB Address Decoder & Mux



Full AHB Interface

We discuss some of these added signals next.



Transfer Size

HSIZE[2:0]	# Bits	Size
0	8	Byte
1	16	Half word
2	32	Word
3	64	Double word
4	128	
5	256	
6	512	
7	1024	

Examples: Write 1 Byte

Signal	Option 1	Option 2
HADDR	0x1000	0x1002
HSIZE	2 (4 bytes)	0 (1 byte)
HWSTRB	0b0100	0b0001
HWDATA	0x00420000	0x00000042

Write 0x42 to address 0x1002

HWSTRB (Write Strobe):

Indicates which byte(s) to write

Option 1: Sends whole word, enables 1 byte within word to be written

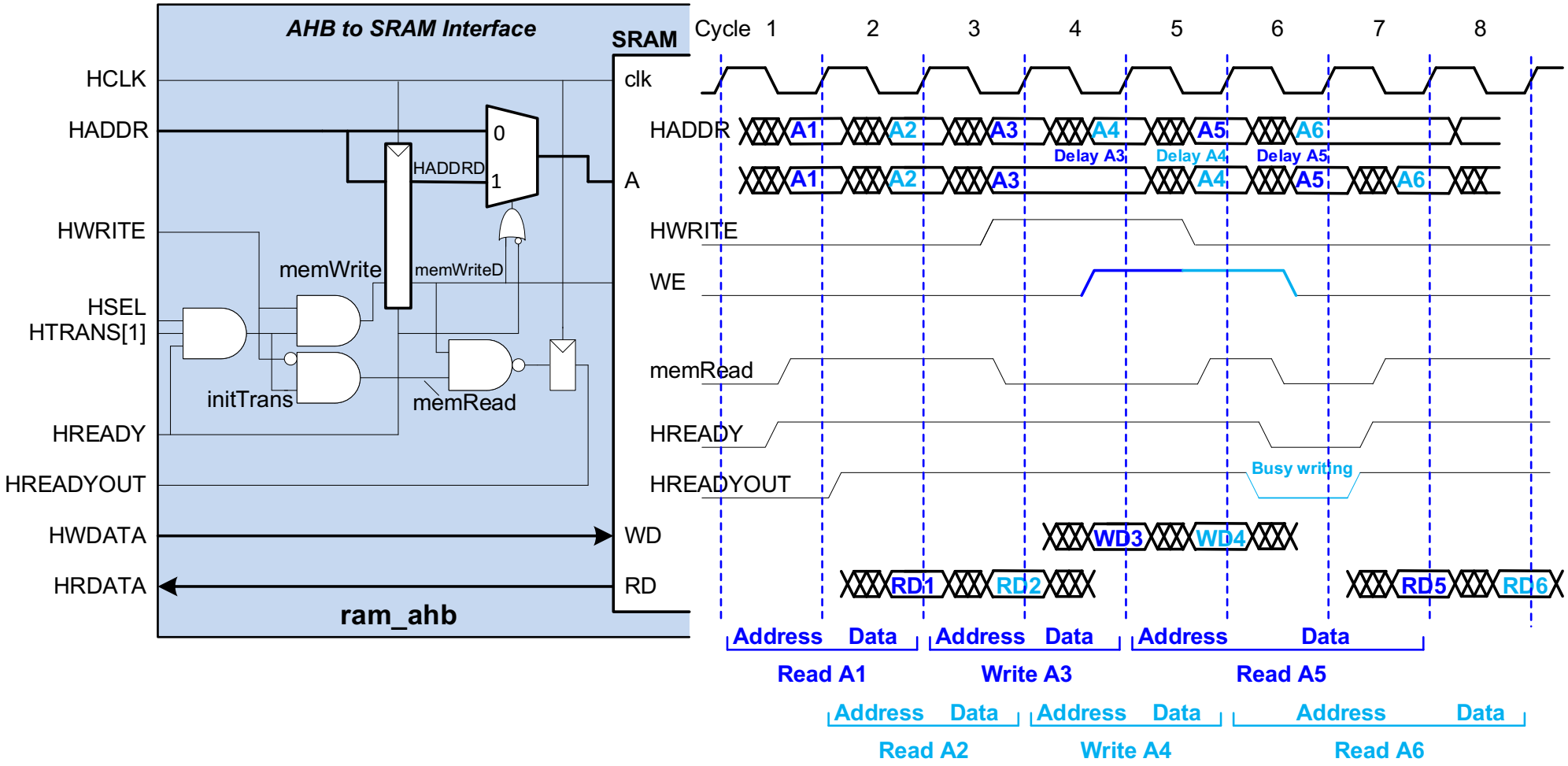
Option 2: Single-byte transfer

AHB Peripherals

Must:

- **Detect** new transactions
- Accept write data (**HWDATA**) in Data phase, 1 cycle after address
- **Read/write registers** in the peripheral
- Produce **HRDATA** upon read request
- Generate **HREADYOUT**

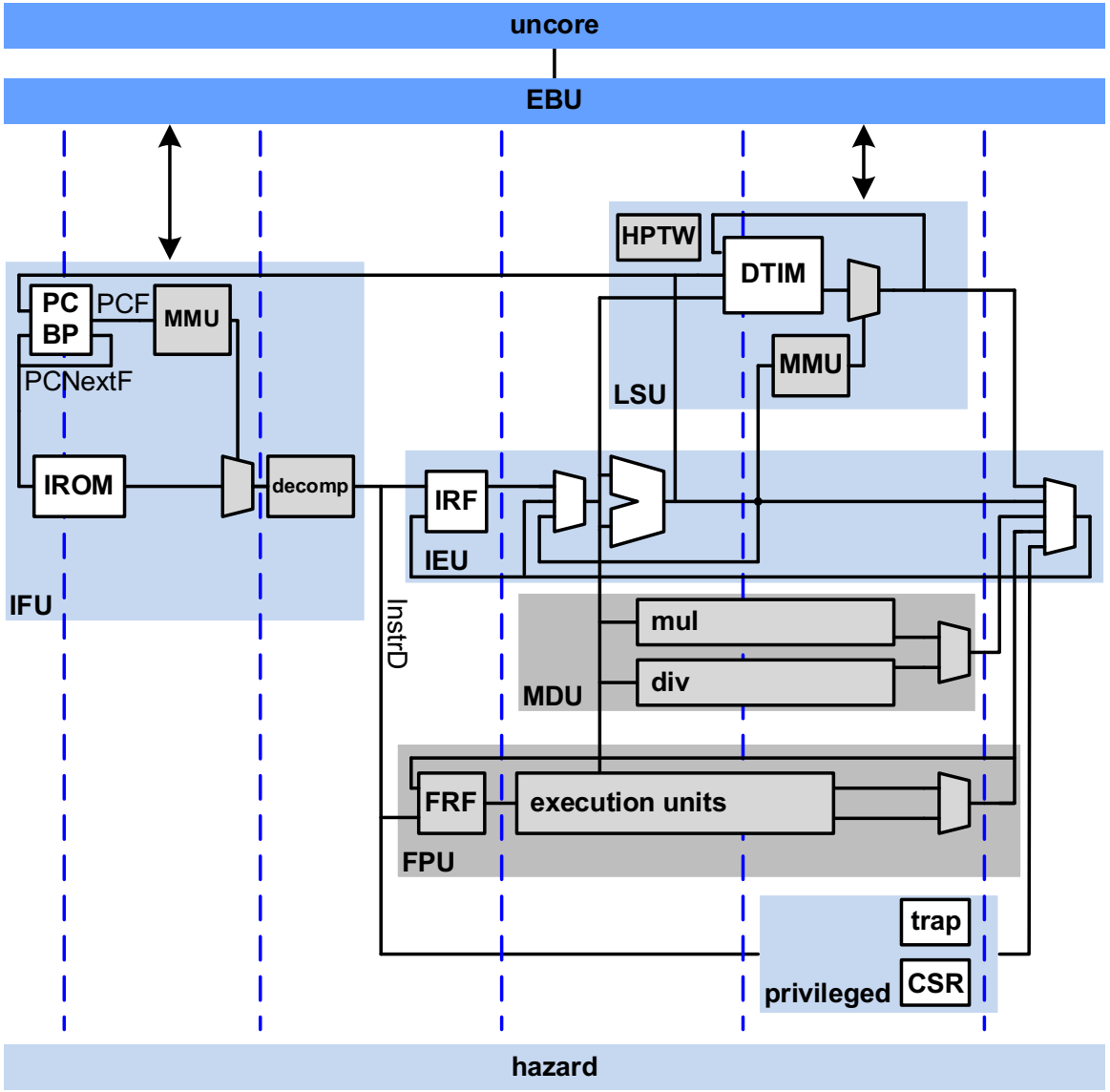
Example: AHB/SRAM Interface



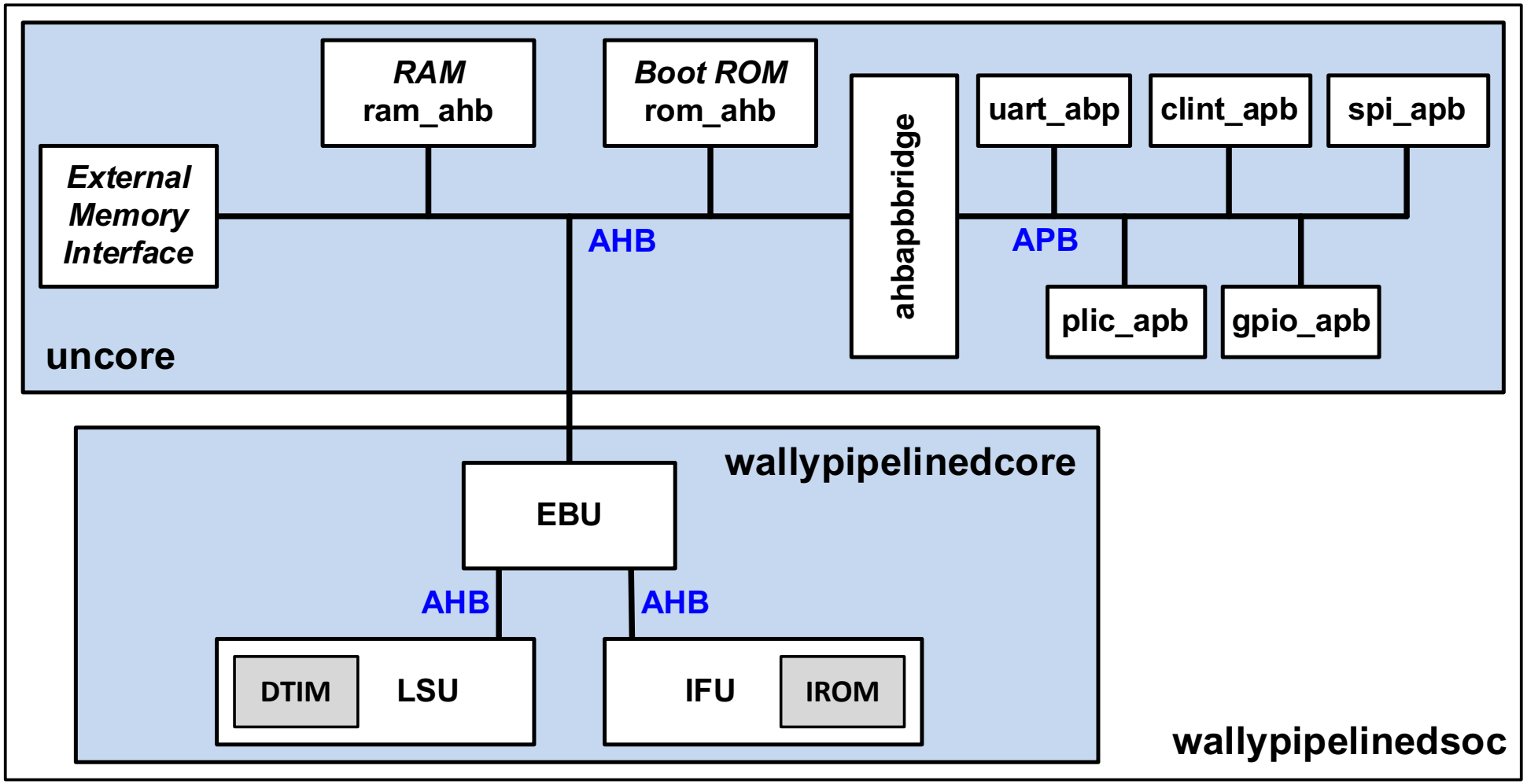
Chapter 9: Bus Interface

Wally Bus Interface

Wally SoC with EBU & Uncore



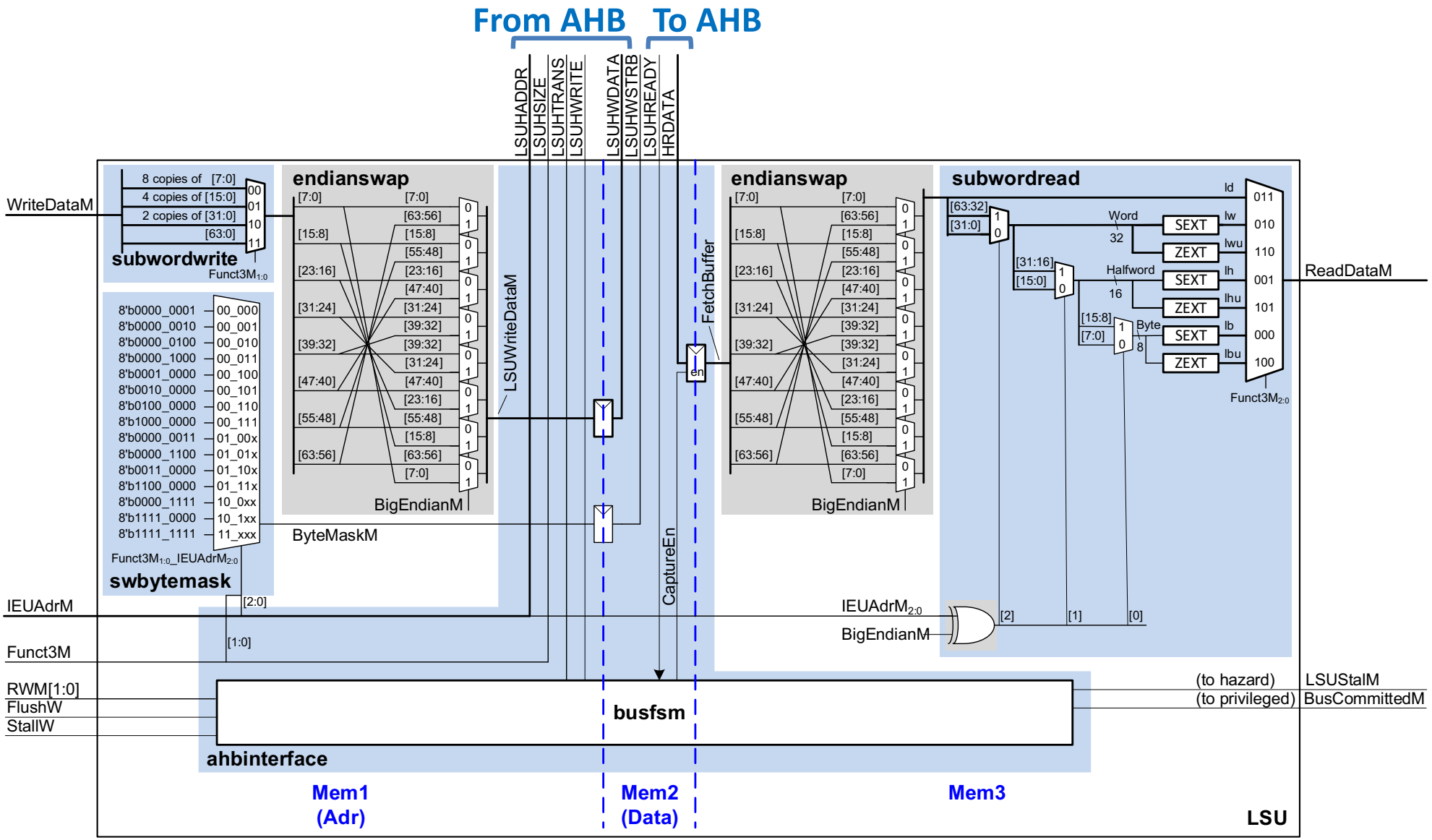
Details: Wally SoC w/ EBU & Uncore



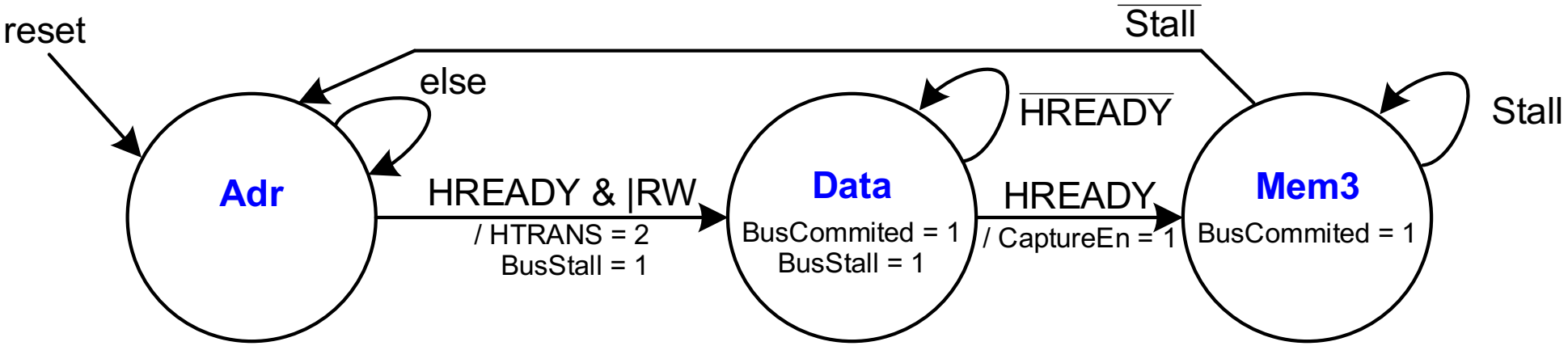
Wally Bus/Memory Configurations

Configuration Parameter	Meaning	Typical Address
UNCORE_RAM_SUPPORTED	RAM in the Uncore on AHB	80000000
BOOTROM_SUPPORTED	ROM in the Uncore on AHB for bootloader	1000
EXT_MEM_SUPPORTED	AHB interface to RAM outside the Uncore (e.g. via a DDR memory controller on an FPGA)	80000000

LSU with AHB Interface

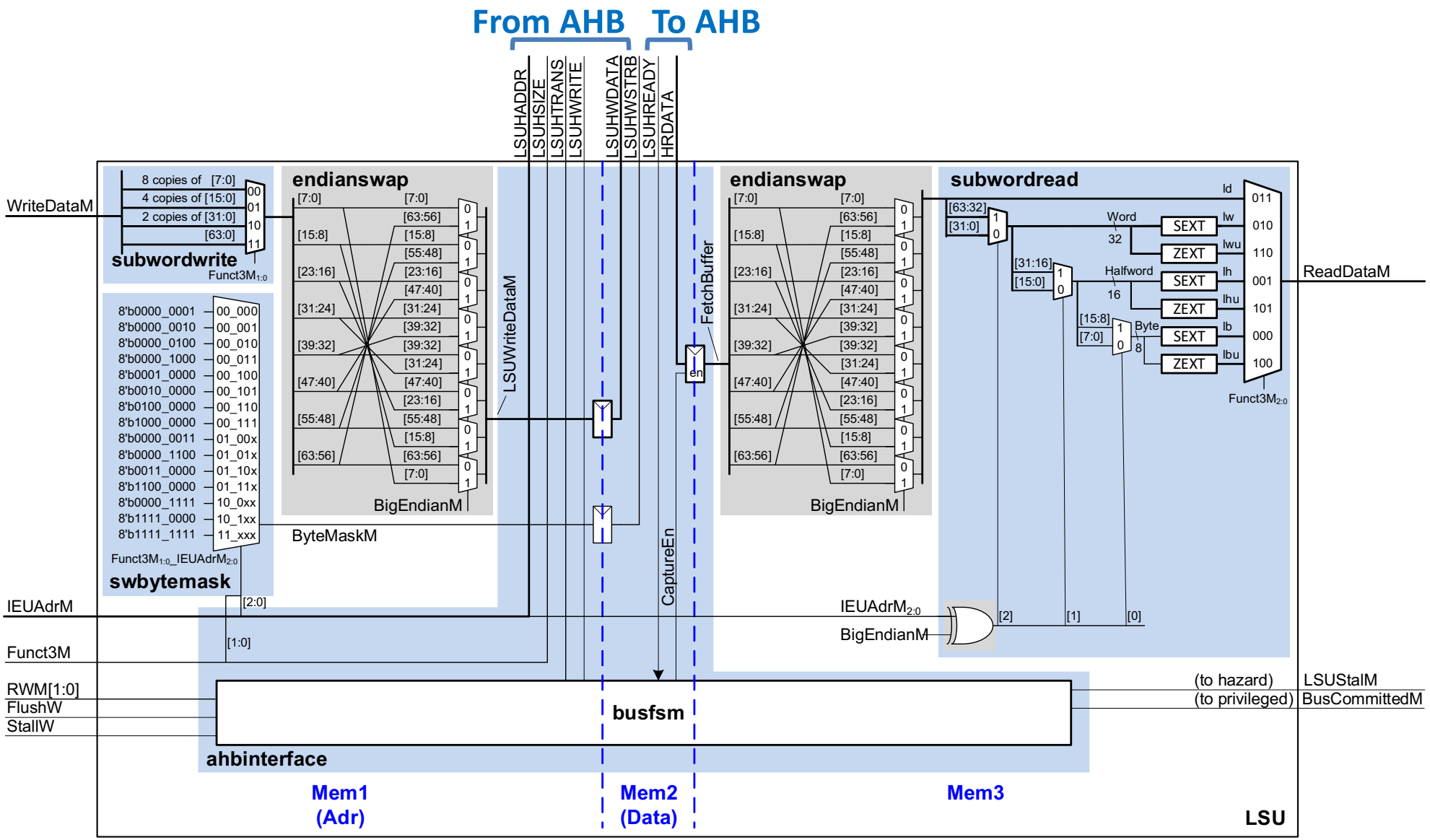


LSU's Bus FSM (busfsm)

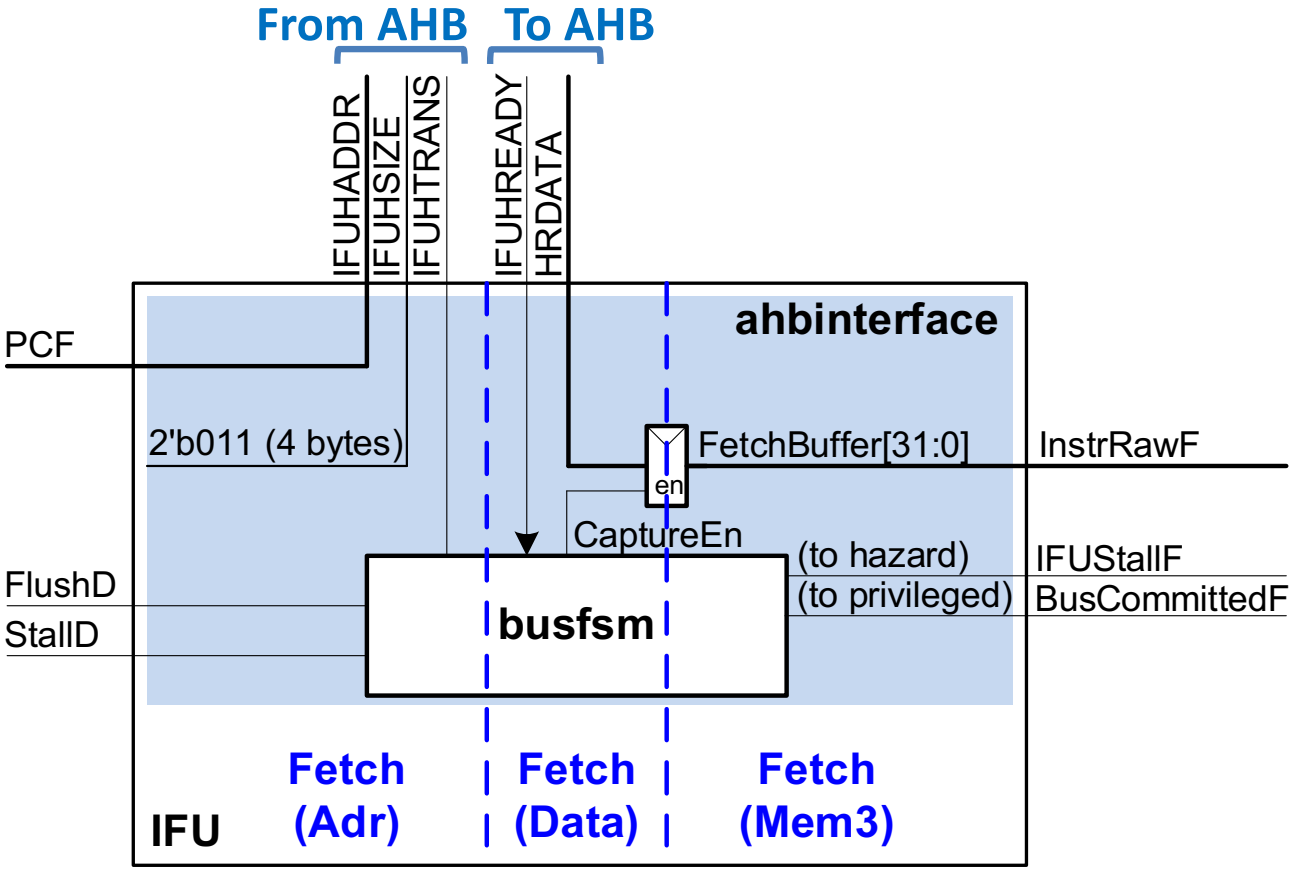


Stall = StallW

Review: LSU with AHB Interface

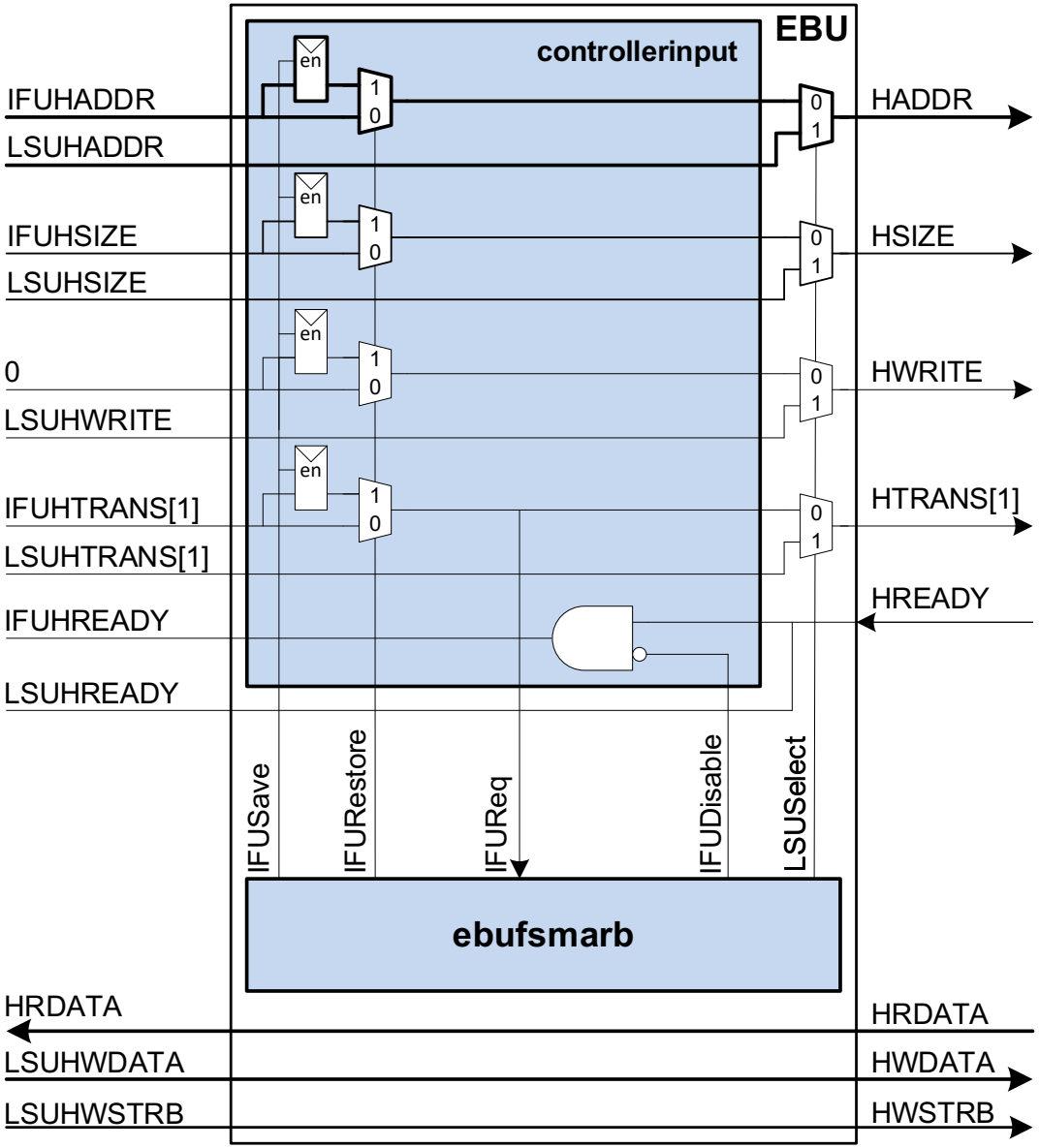


IFU with AHB Interface

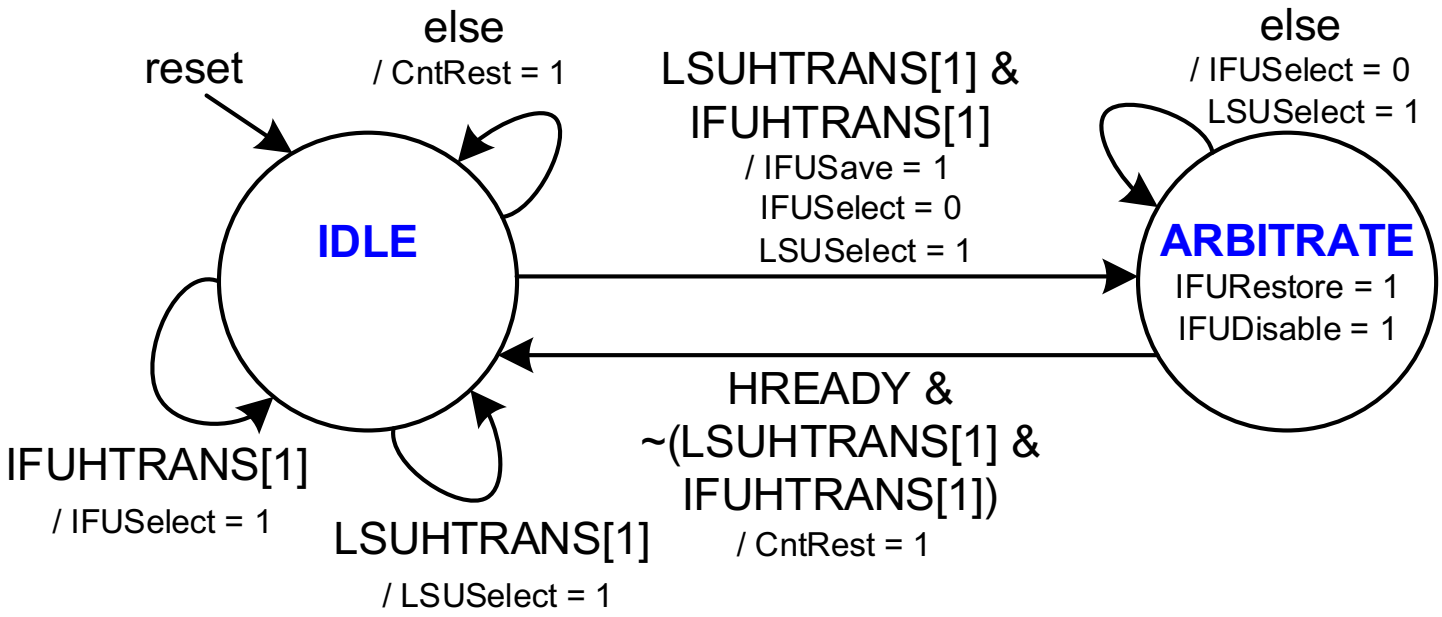


- **Read-only:** so no HWDATA, HWRITE, HWSTRB
- **Little-endian:** so no byteswap modules
- **32-bit reads only:** so no subwordread modules
- StallD rather than StallW

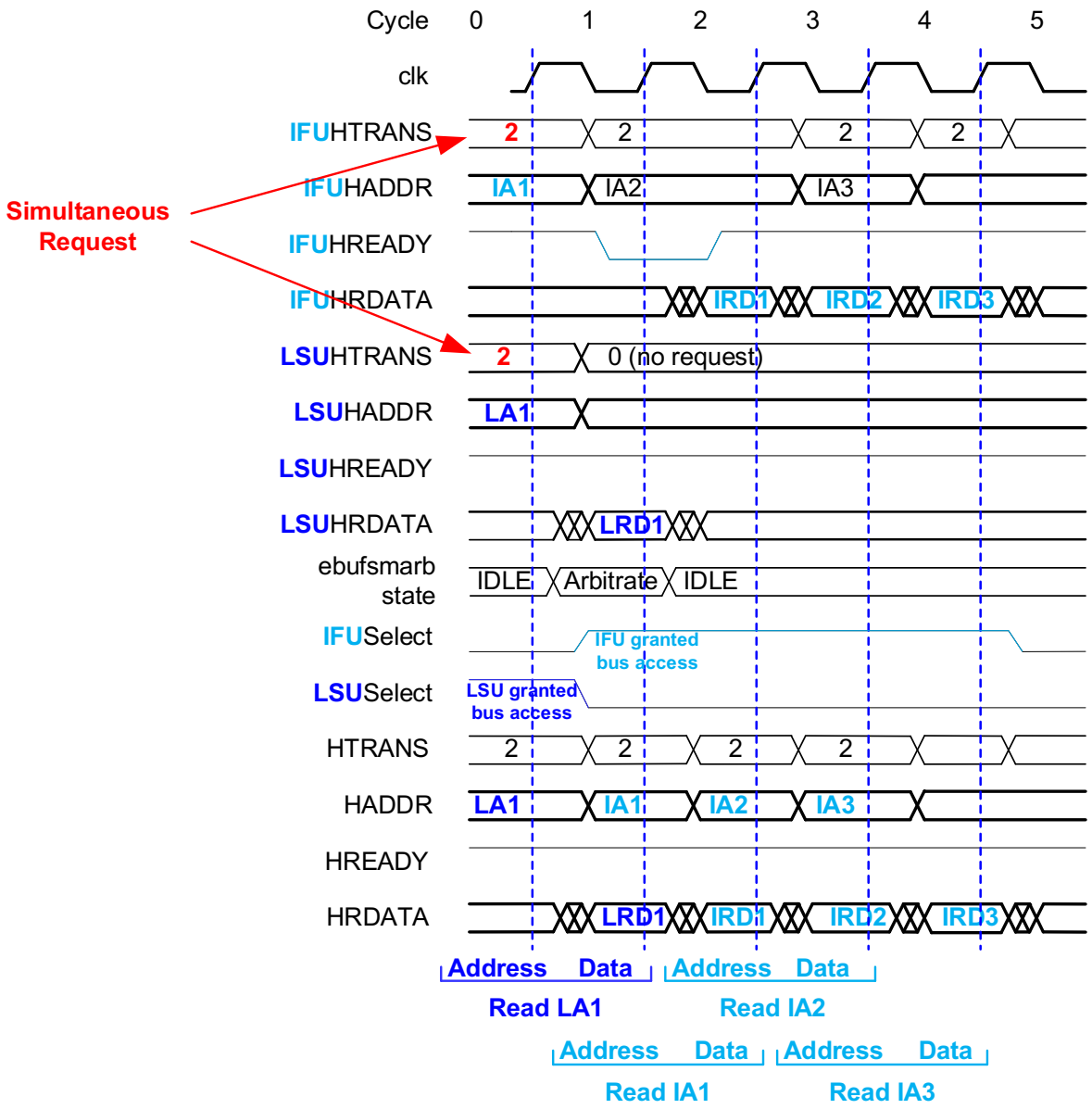
EBU: External Bus Unit



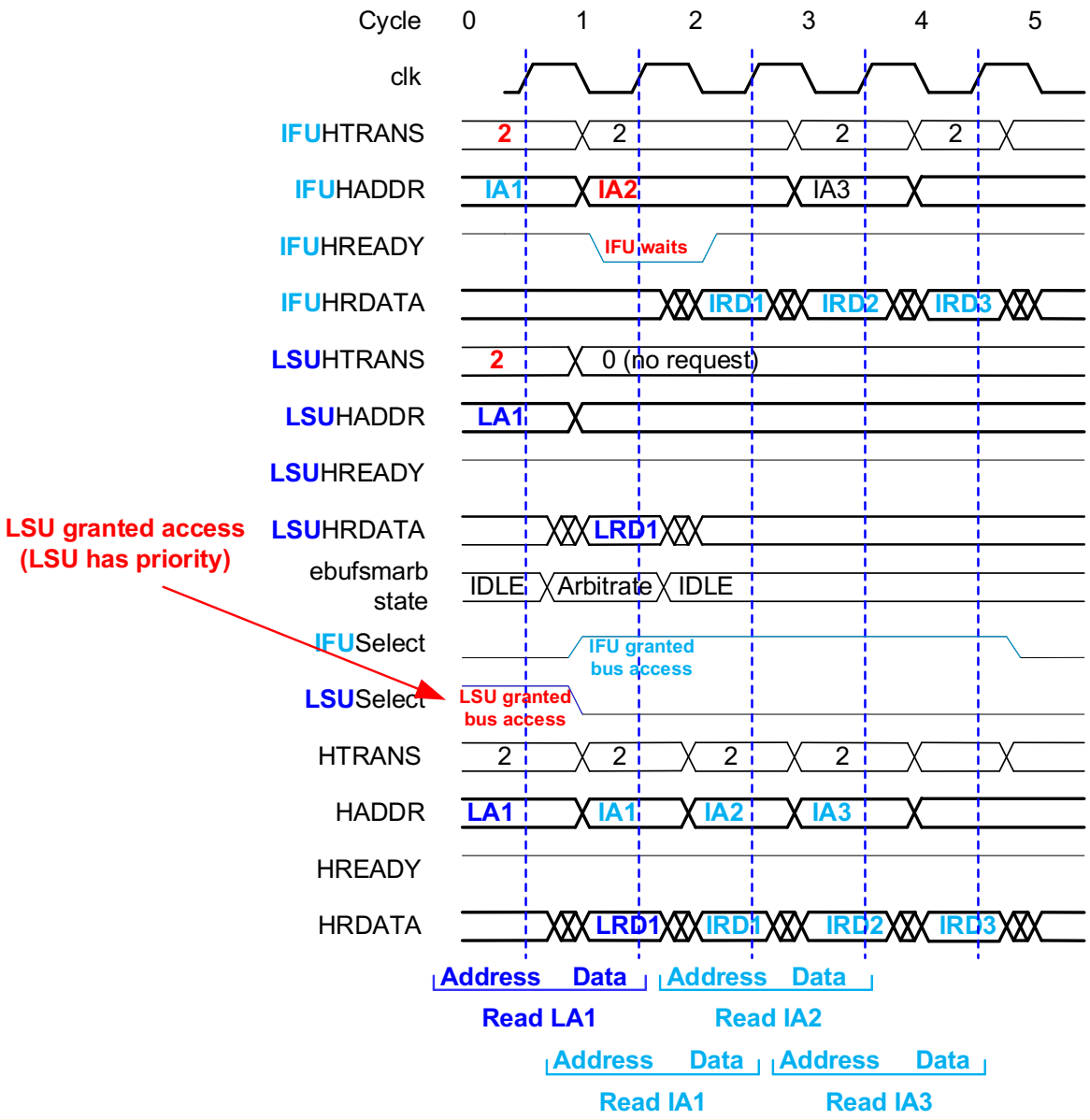
EBU FSM Arbiter: ebufsmarb



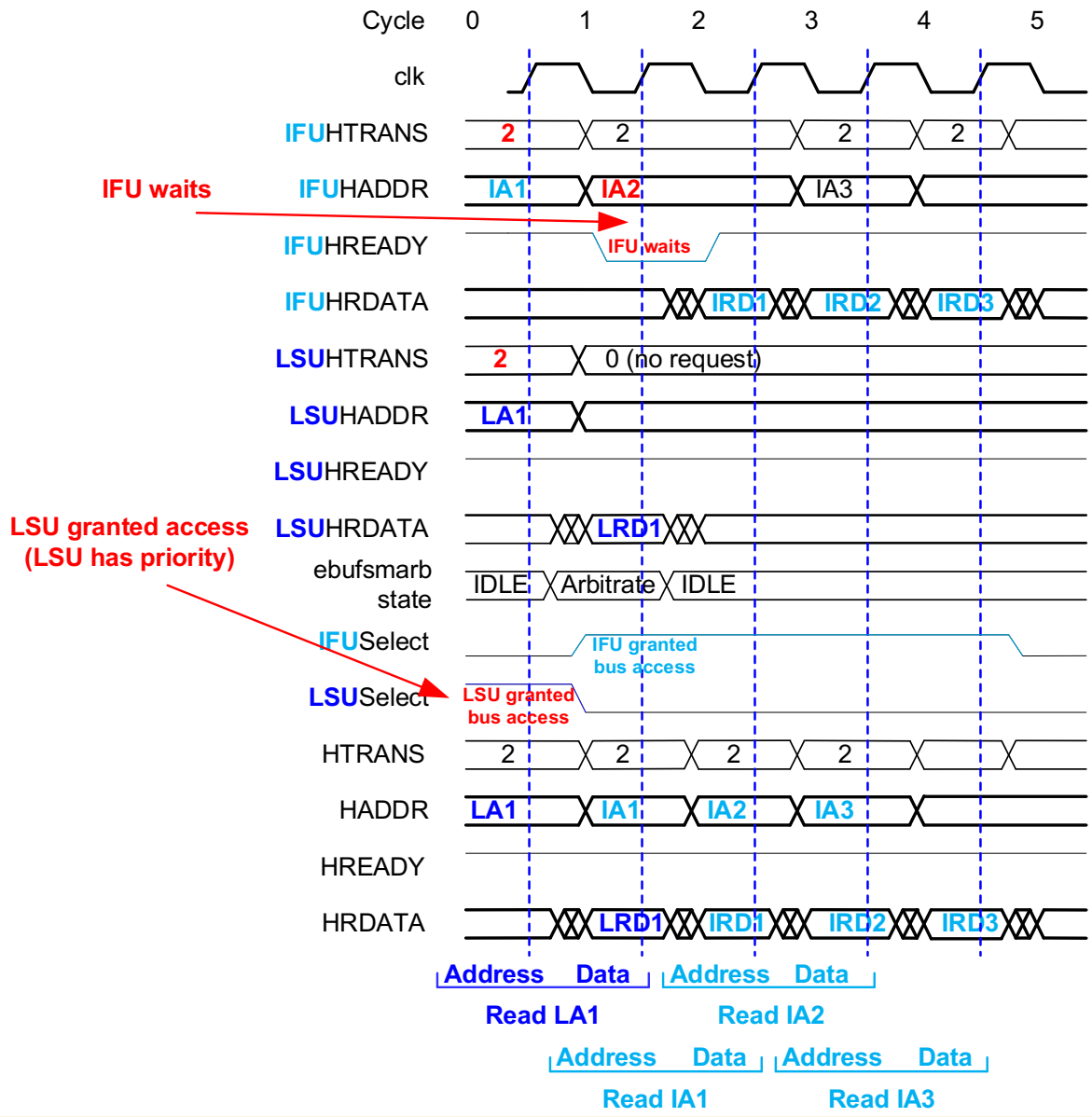
Simultaneous Requests



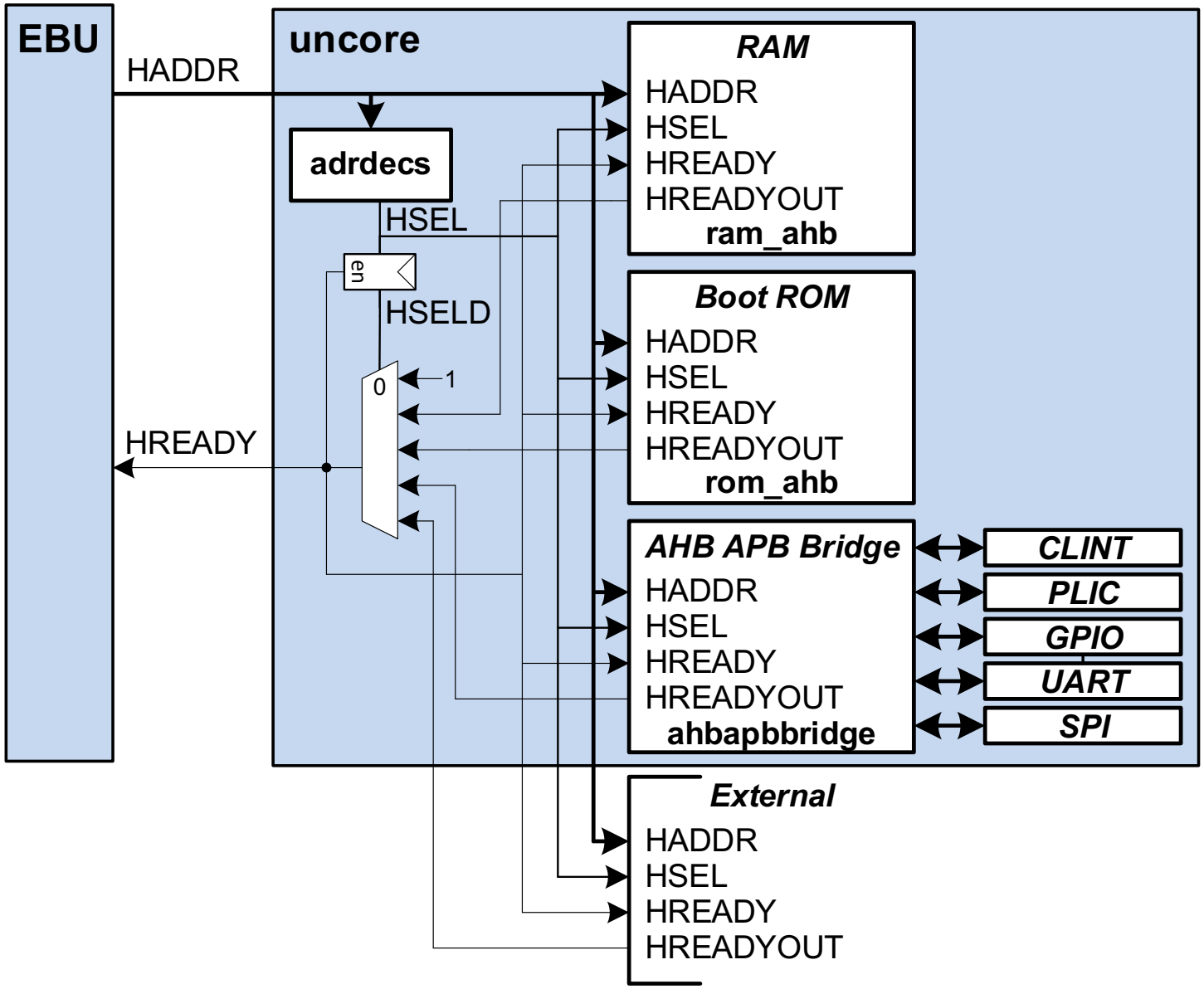
Simultaneous Requests



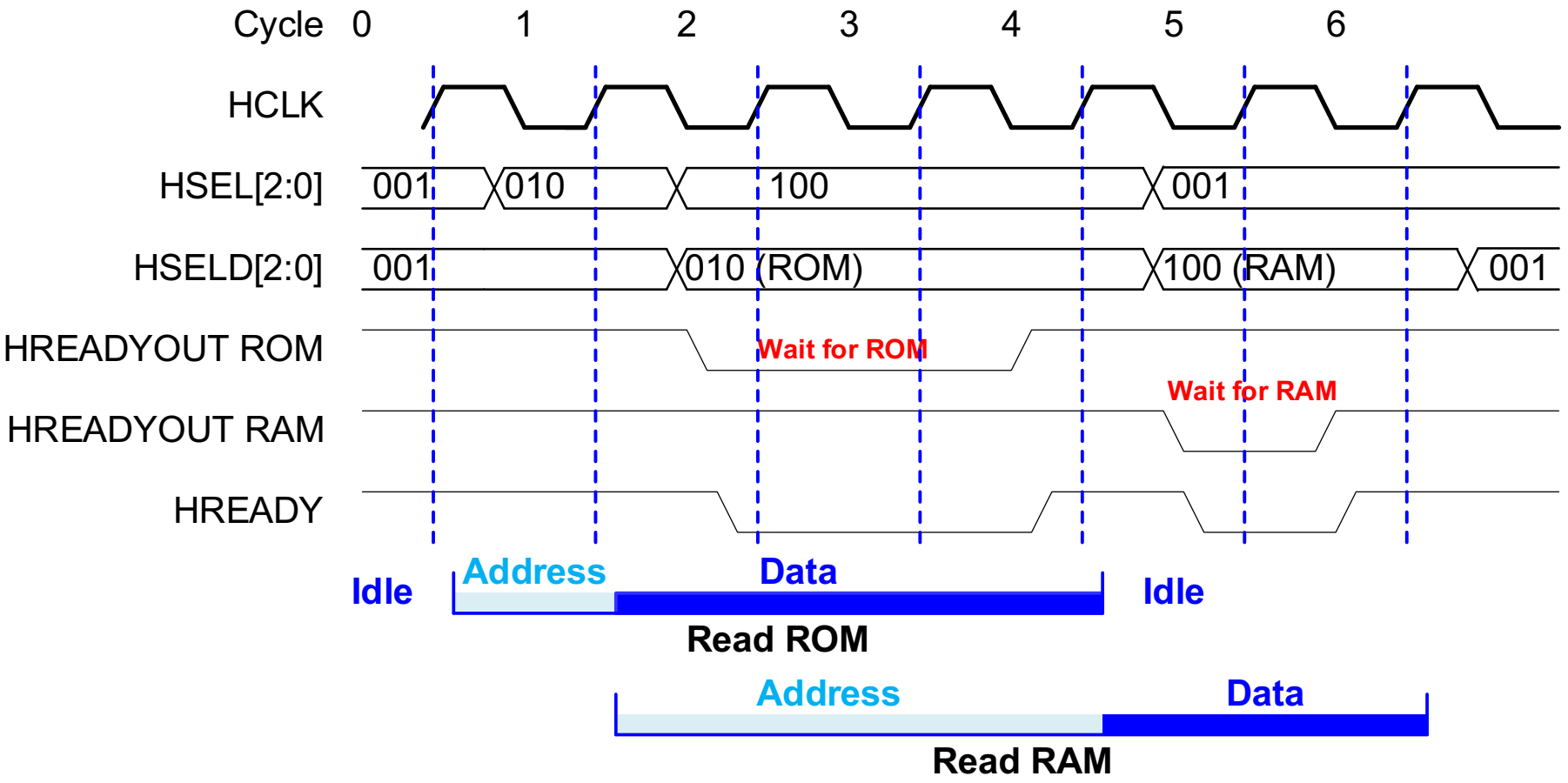
Simultaneous Requests



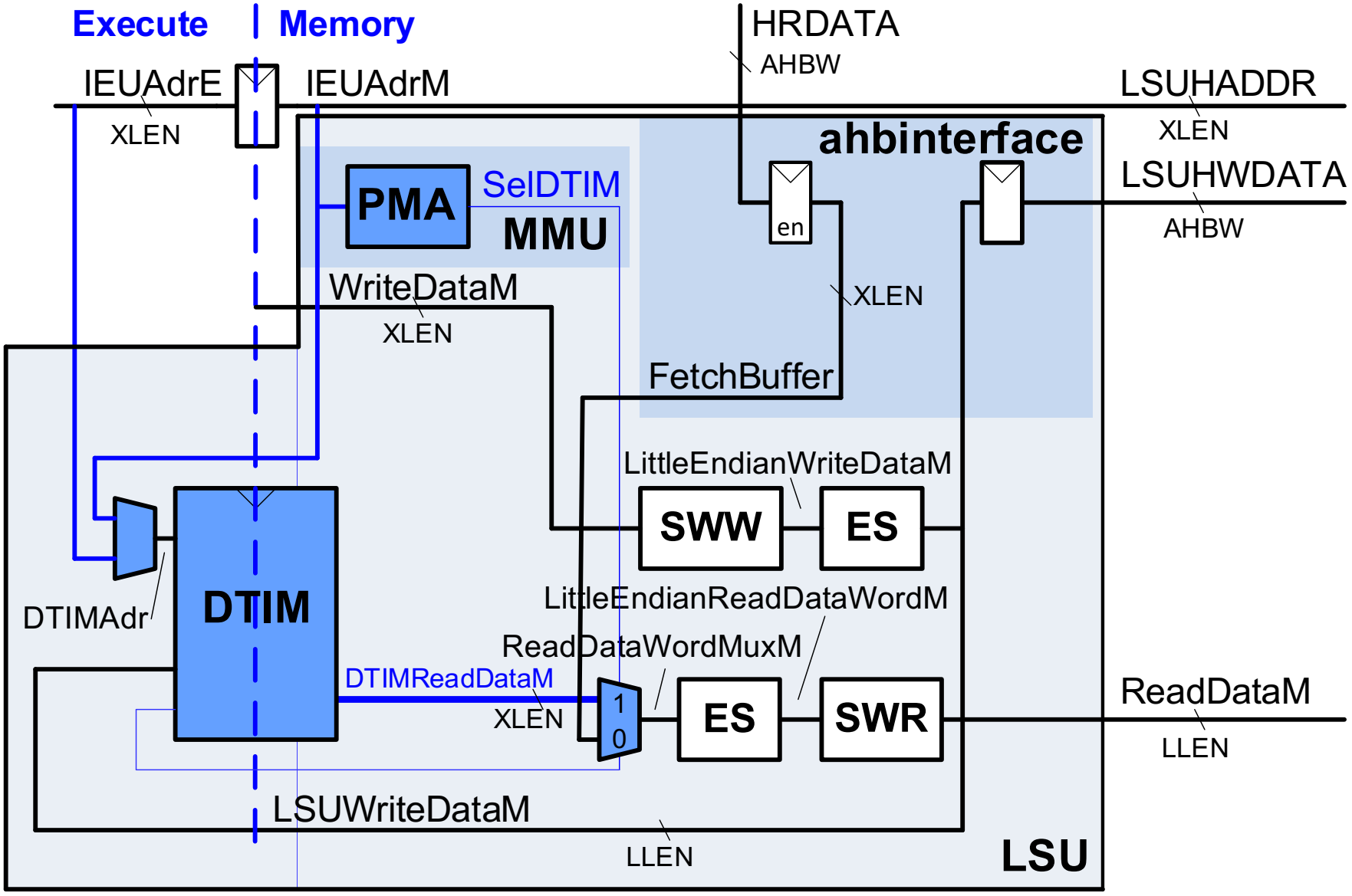
Interface: EBU & Devices



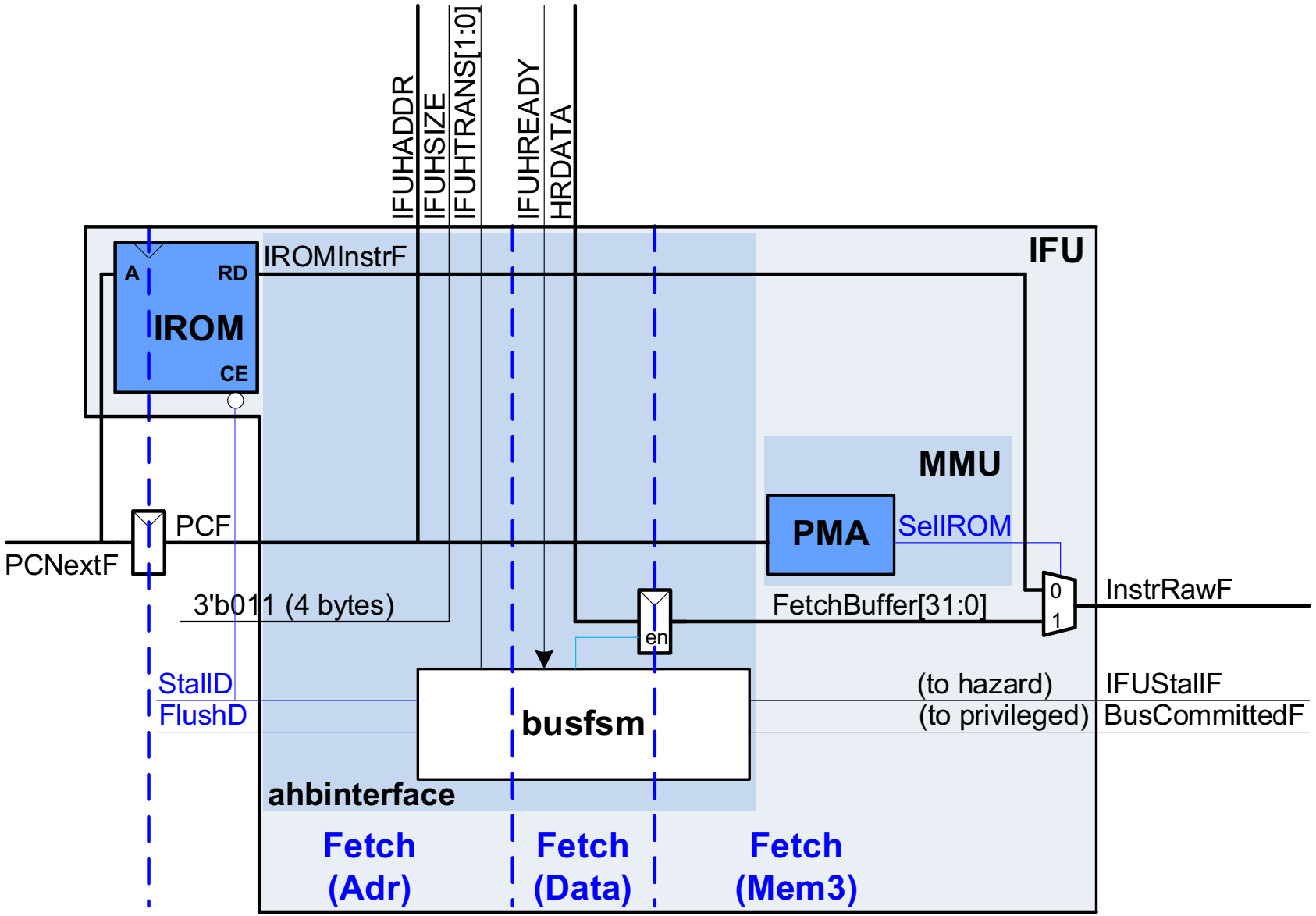
Consecutive Reads & Wait States



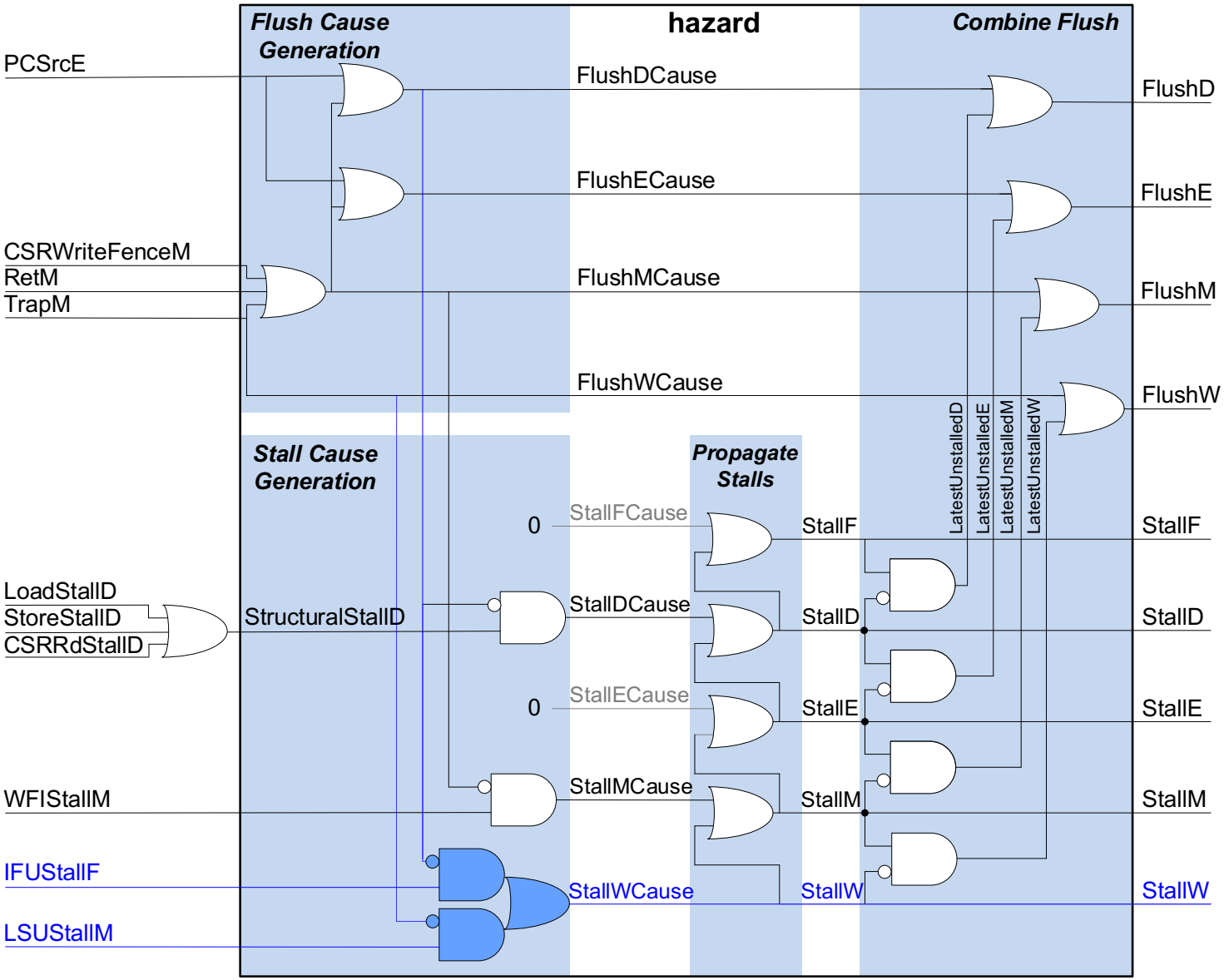
Abstracted LSU with DTIM & Bus



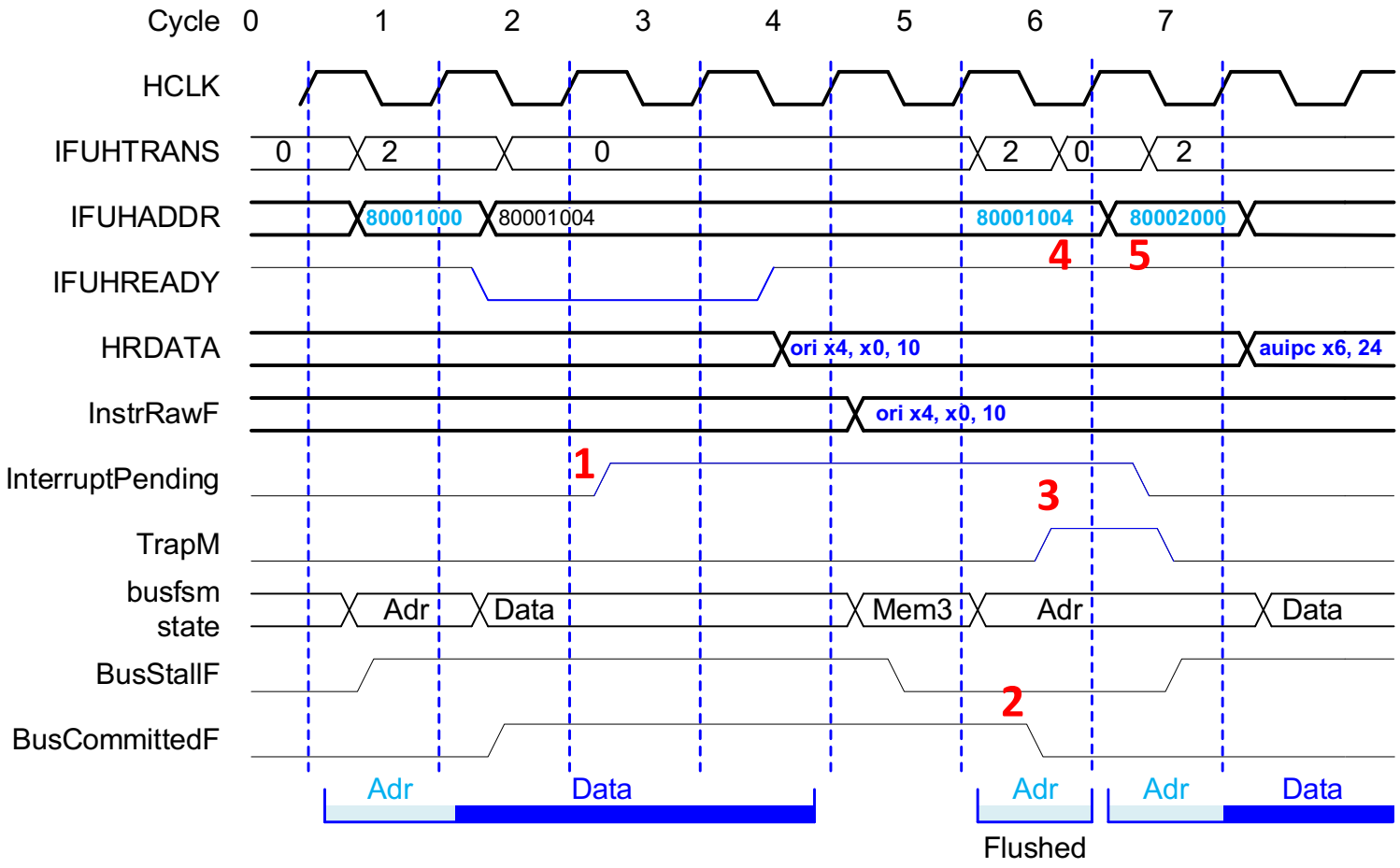
Abstracted IFU with IROM & Bus



Hazard Unit with Bus Support



Hazard Unit with Bus Support



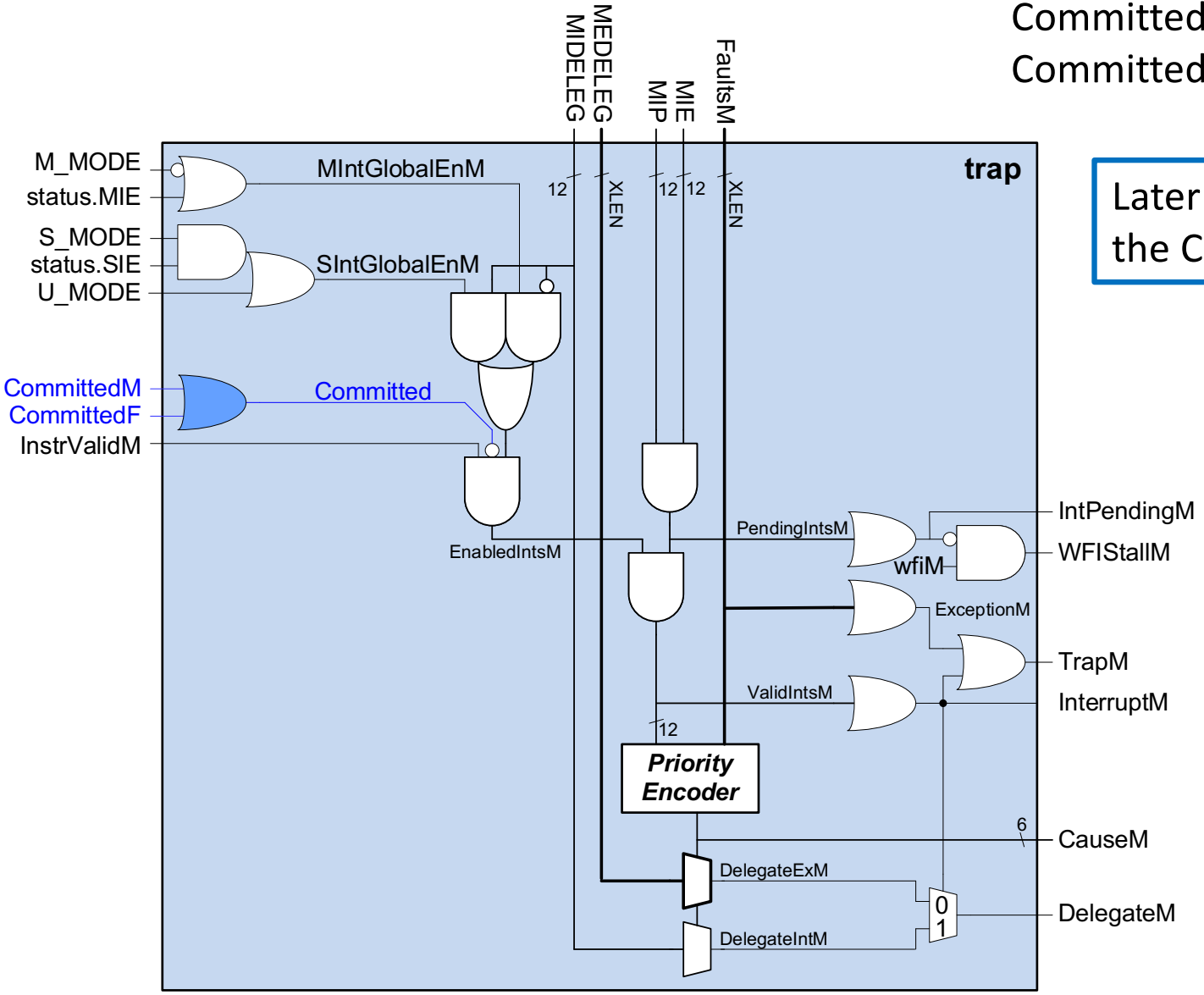
Trap occurs

```
80001000:       ori  x4, x0, 10
80001004:       addi x5, x4, 10
…
TrapHandler:
80002000:       auipc x6, 24
```

1. Trap occurs
2. Bus is committed (is in Data phase), so must wait for bus transaction to complete
3. Bus transaction completes, so trap can be taken
4. Instruction in Adr phase (addi) is flushed
5. Instruction at the trap handler (auipc) fetched

Trap Unit with Committed Signals

CommittedM = BusCommittedM
CommittedF = BusCommittedF



Later chapters will expand the Committed signals

About these Notes

RISC-V System-on-Chip Design Lecture Notes