08: AXI4 Lite

Engr 315: Hardware / Software Codesign

Andrew Lukefahr
Indiana University

Announcements

- P2 is due tonight
- P3 is out
- P4 coming soon.

Register your Hardware

- Go to Canvas
- Select “P2 Box Number”

maybe another name

- Enter your box number

What's Your Board #?

Project ~~3~~: ~~Hardware Box Number~~

This is a preview of the published version of the quiz

Started: Feb 23 at 2:10pm

Quiz Instructions

Question 1 1 pts

Box # you picked up from the lab is [Select]

Not saved Submit Quiz

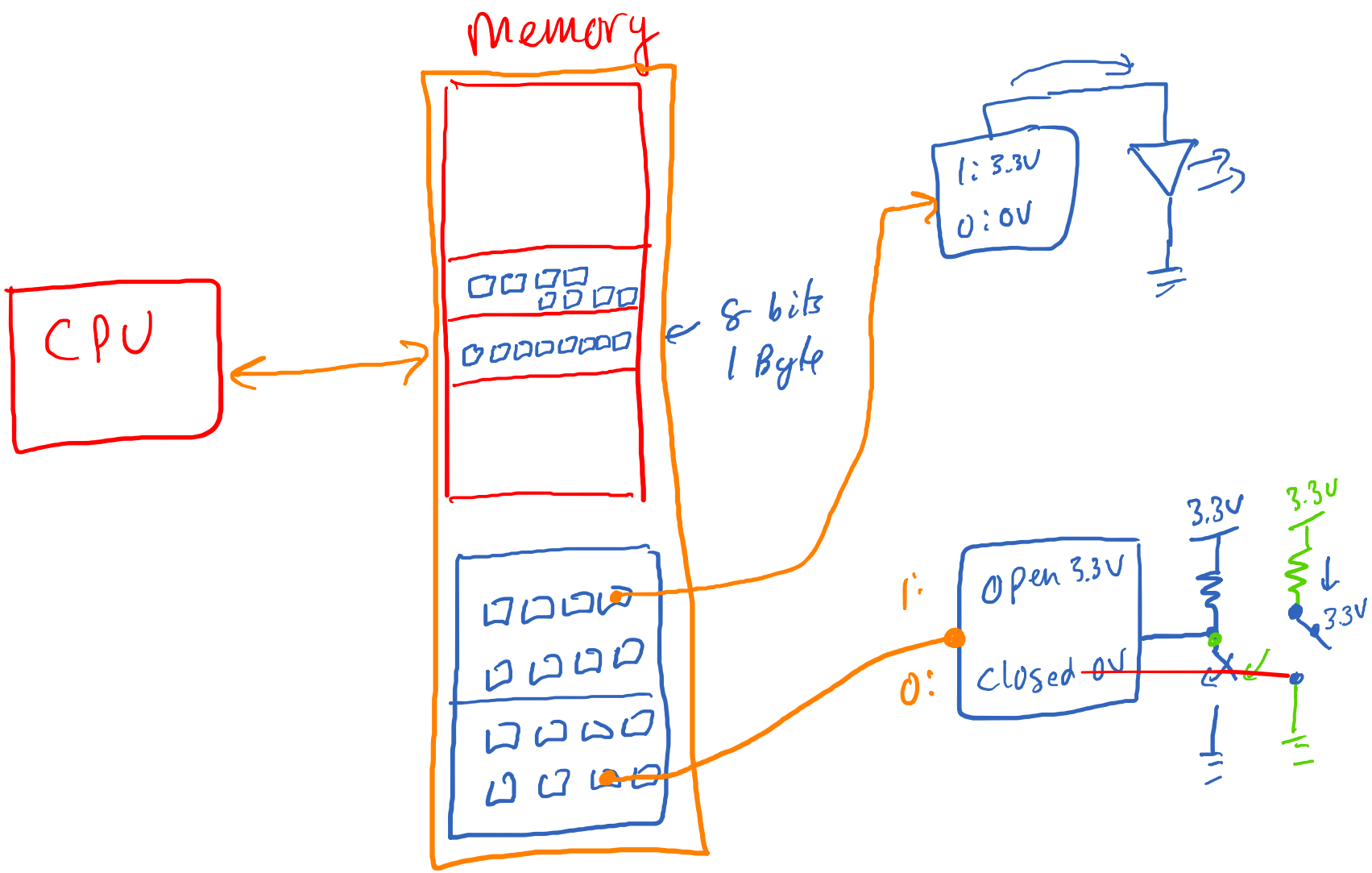
Optimizations thus far

- Algorithmic complexity
- Removing redundant computation
- ~~Multithreading~~
- ~~Multiprocessing*~~
- Python/C/Asm Interfacing
- Map to Hardware

↳ Bus

↳ mmIO

Review: Memory-Mapped I/O



Use volatile for MMIO addresses!

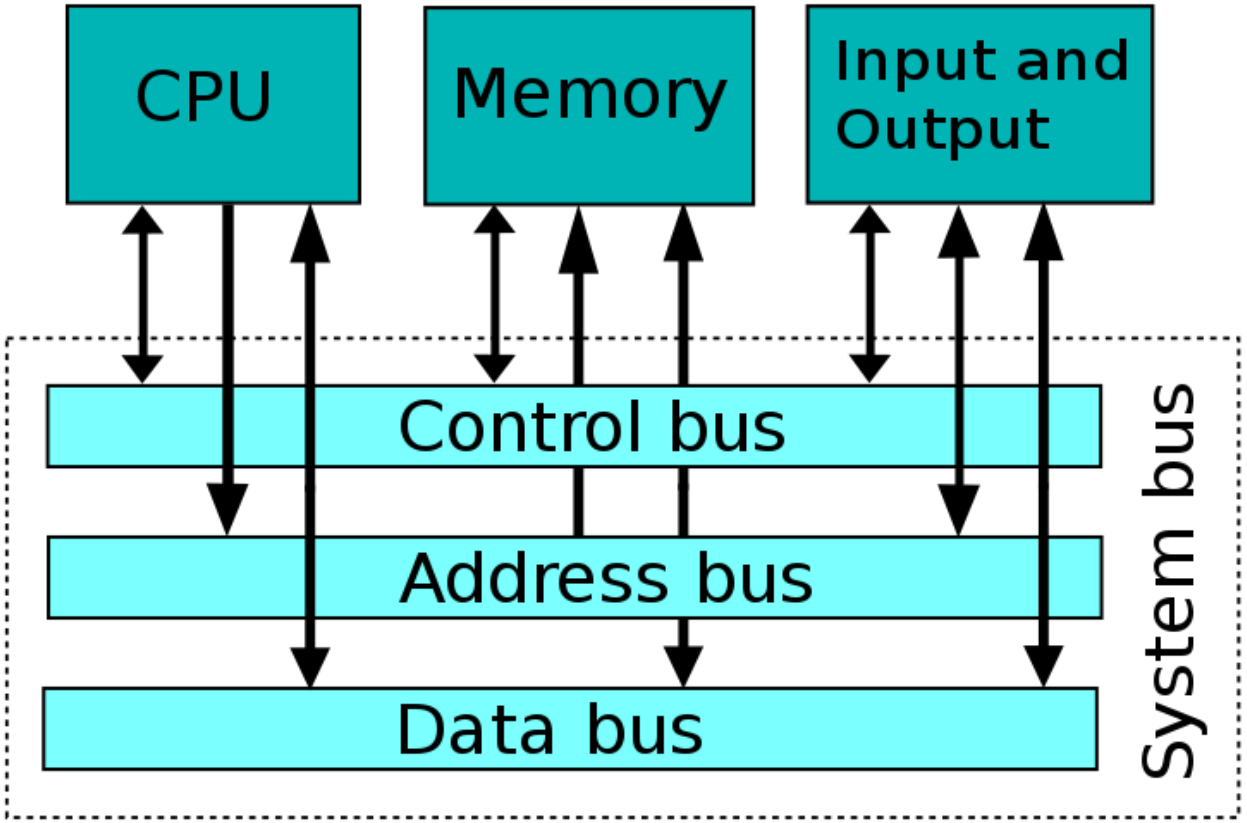
```
#define SW_ADDR 0xfffe
volatile uint32_t * SW_REG = (uint32_t * SW_ADDR);

int quit = (*SW_REG);
while(!quit)
{
    //more code
    quit = (*SW_REG);
}
```

Use **`volatile`** for all MMIO memory.

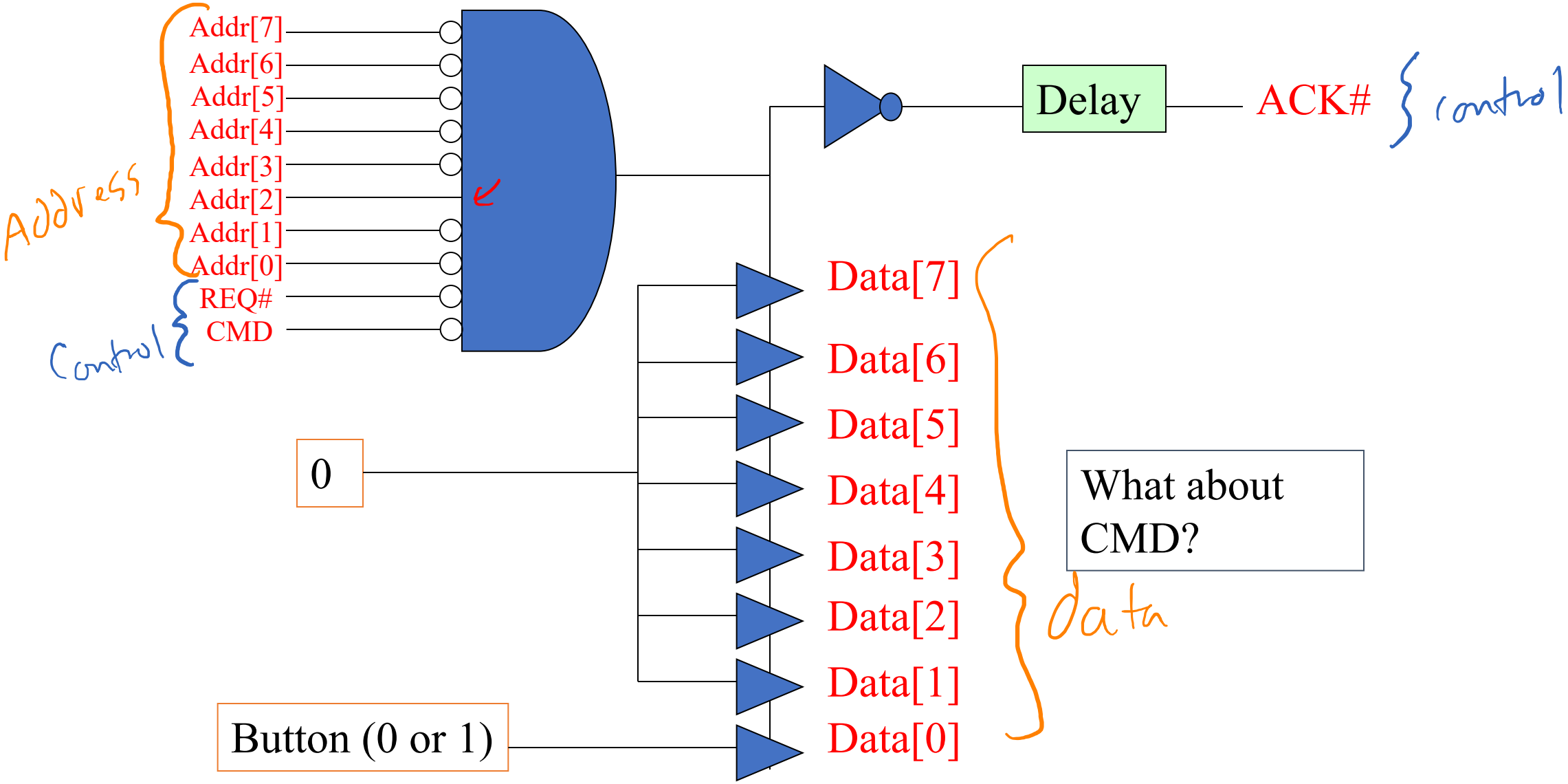
(hint: P4)

The System Bus



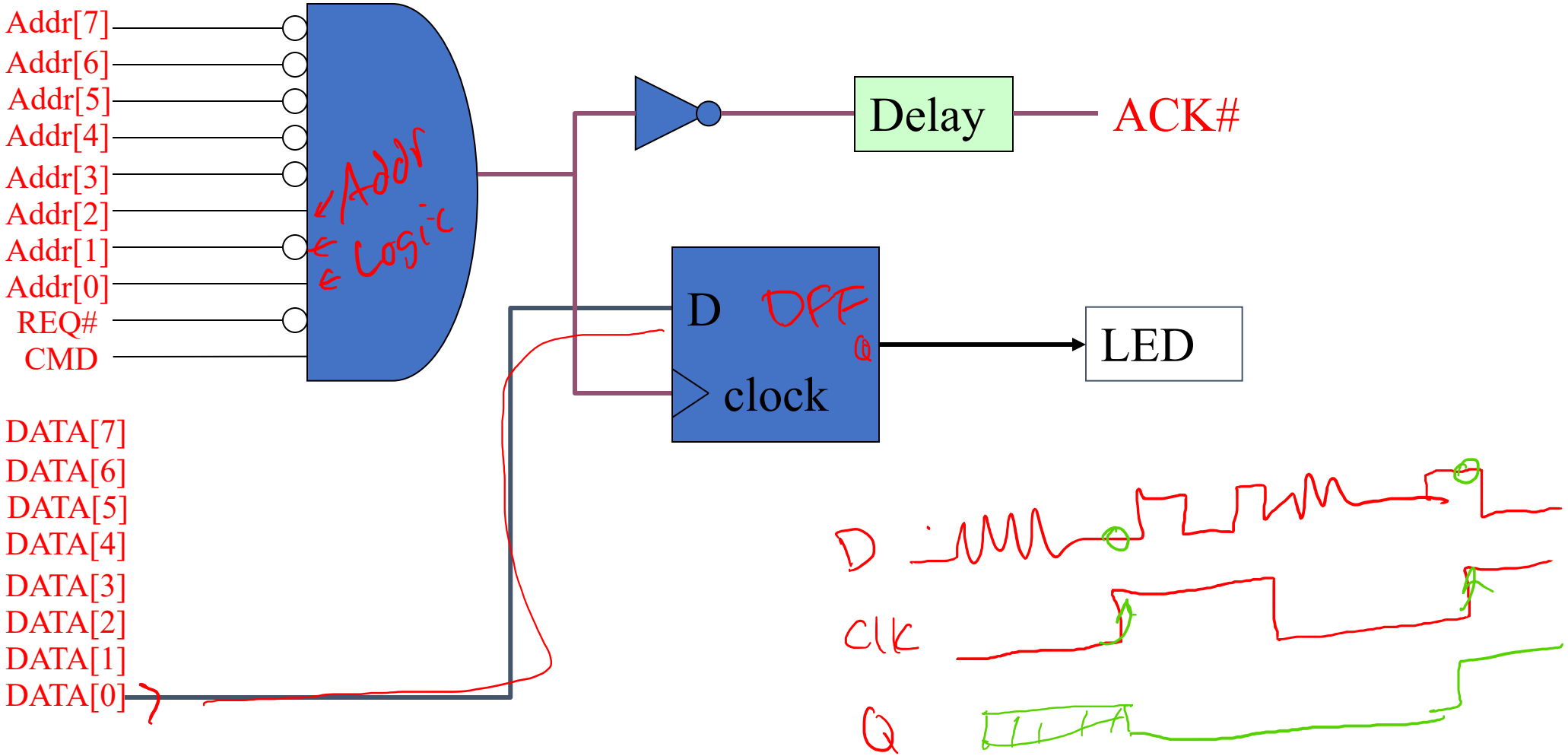
The push-button

(if Addr=0x04 ~~write~~ read 0 or 1 depending on button)



The LED

(1 bit reg written by LSB of address 0x05)



Let's write a simple C program to turn the LED on if button is pressed.

Peripheral Details

- 0x04: Push Button - Read-Only
  - Pushed -> 1
  - Not Pushed -> 0
- 0x05: LED Driver - Write-Only
  - On -> 1
  - Off -> 0

```
#define PB 0x04
#define LED 0x05

int main() {
register int val;
for (;;) {
val = *(volatile uint32_t *)(PB);
*(volatile uint32_t *)(LED) = val;
}
}
```

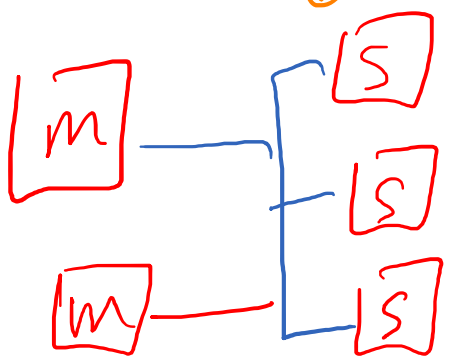
ARM AXI Bus

- “Advanced eXtensible Interface” Bus Version 4, “AXI4”

ARM AXI Bus

- "Advanced eXtensible Interface" Bus Version 4, "AXI4"

Mem-Mapped

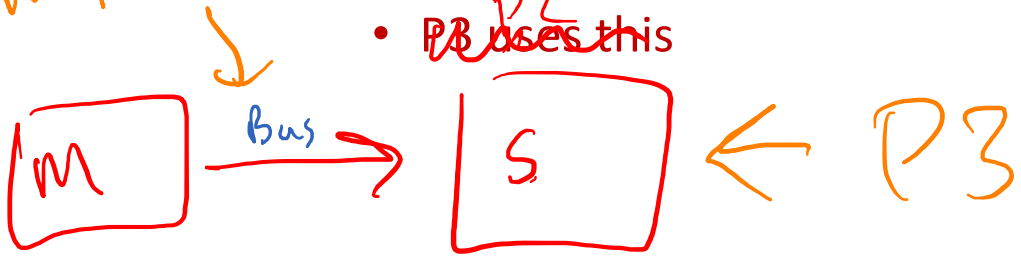


P4

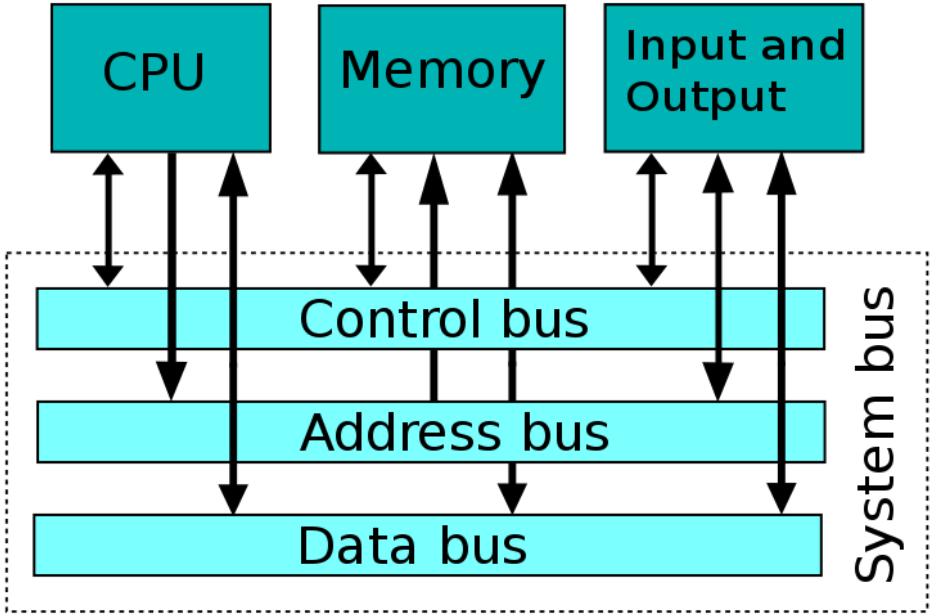
- Three Variants
  - AXI4: Fast but complicated; Memory-mapped
  - AXI4 Lite: Slow but simple; Memory-mapped

    P3
  - AXI4 Stream: Fast and simple; Not memory-mapped
    - ~~P3 uses this~~ P2

Not Mem Mapped



Why AXI4 Lite?

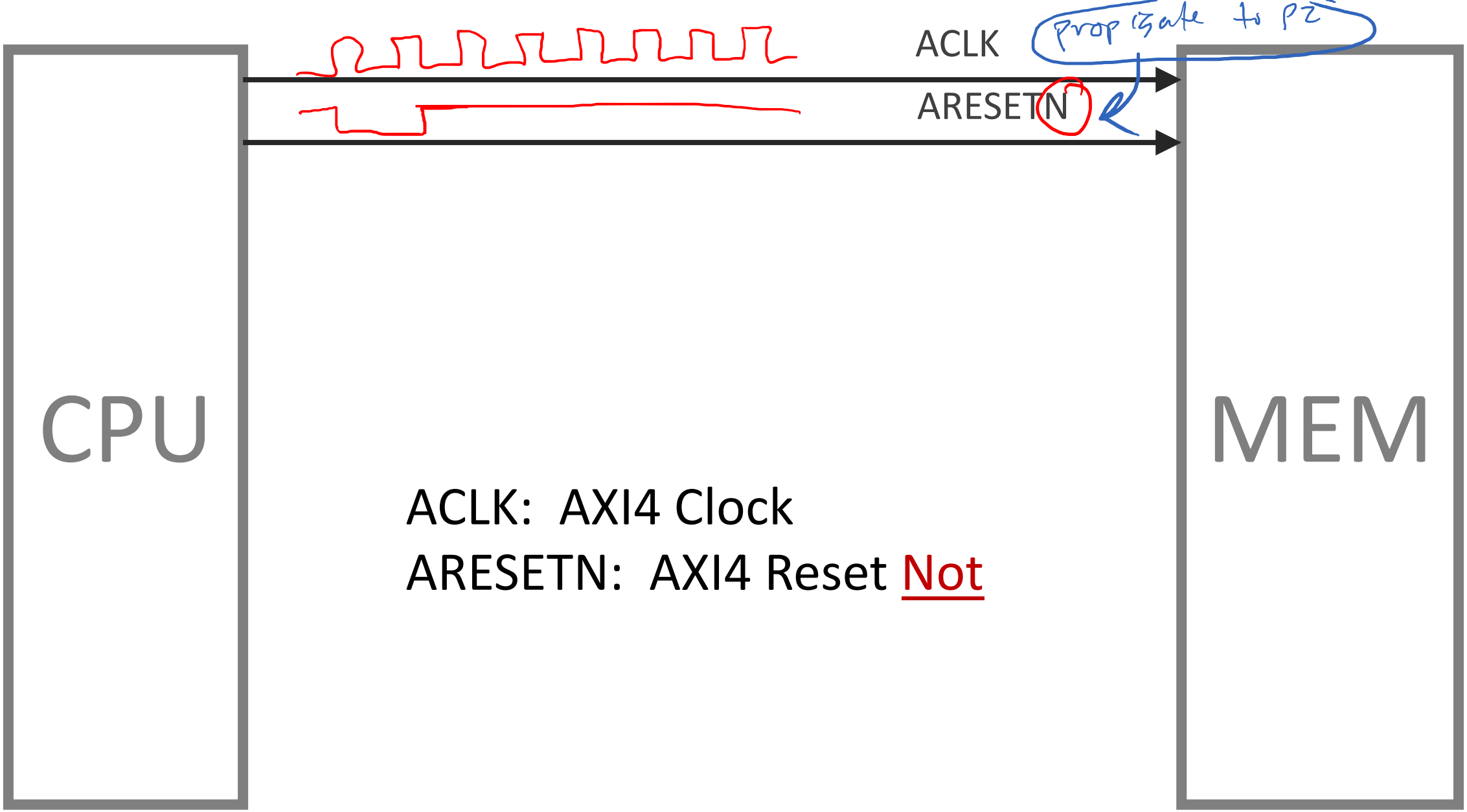


Xilinx AXI Reference Guide:

"AXI4-Lite is a light-weight, single transaction memory mapped interface. It has a small logic footprint and is a simple interface to work with both in design and usage. "

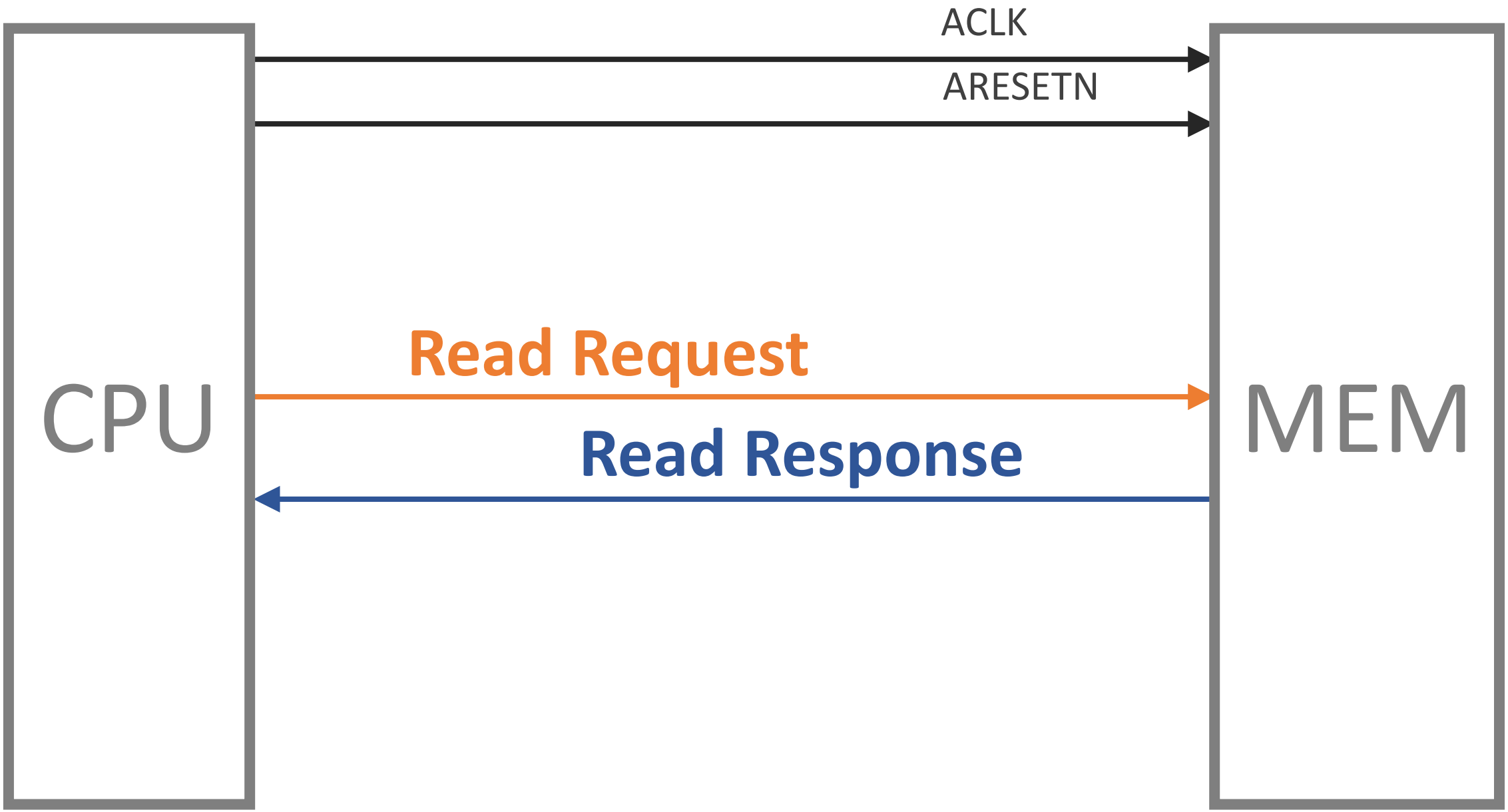
CPU

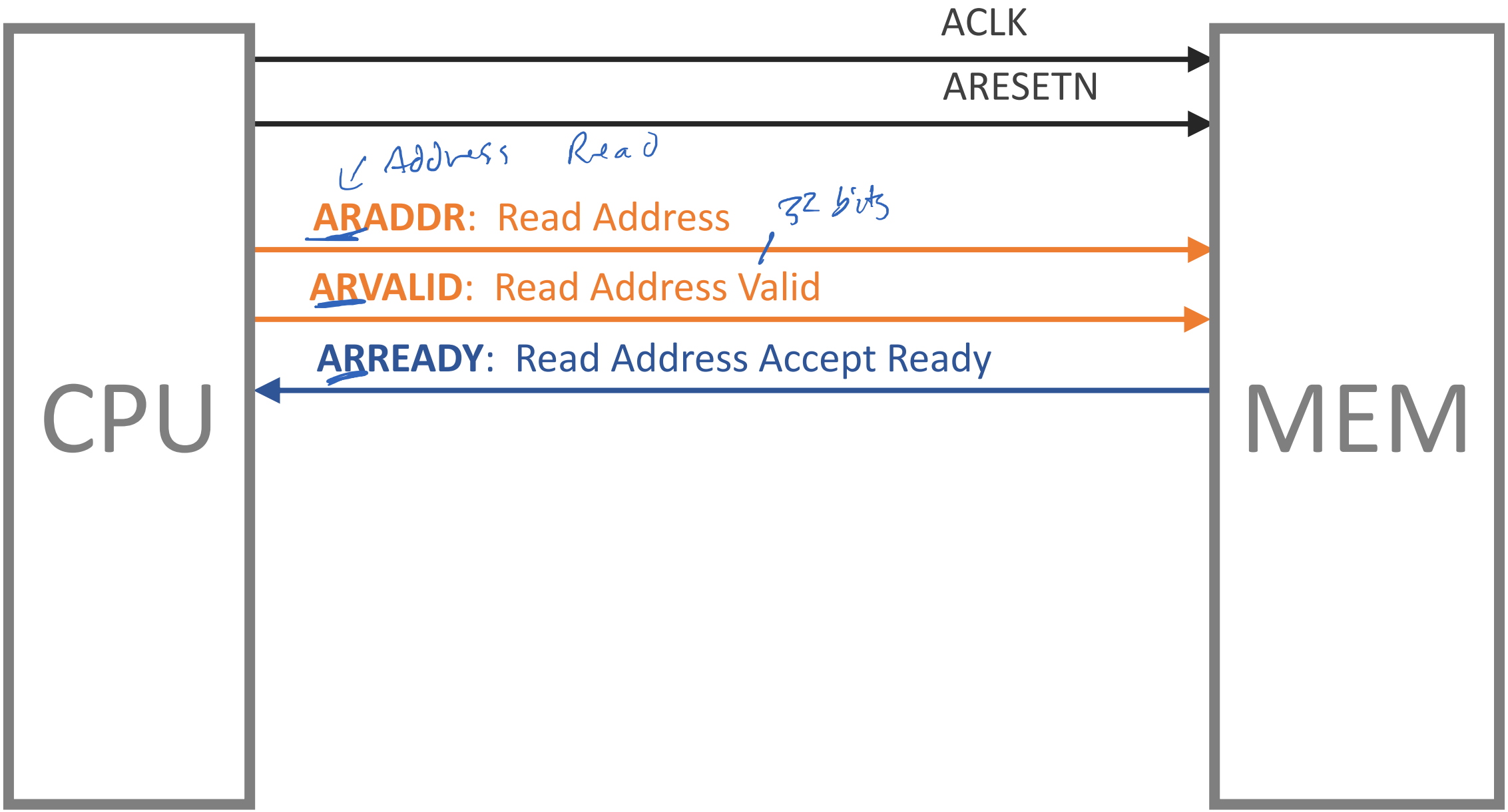
MEM



ACLK: AXI4 Clock

ARESETN: AXI4 Reset Not


ACLK
ARESETN
Read Request
Read Response
CPU
MEM


ACLK
ARESETN
Address Read
ARADDR: Read Address
32 bits
ARVALID: Read Address Valid
ARREADY: Read Address Accept Ready
CPU
MEM

AXI4 Handshaking

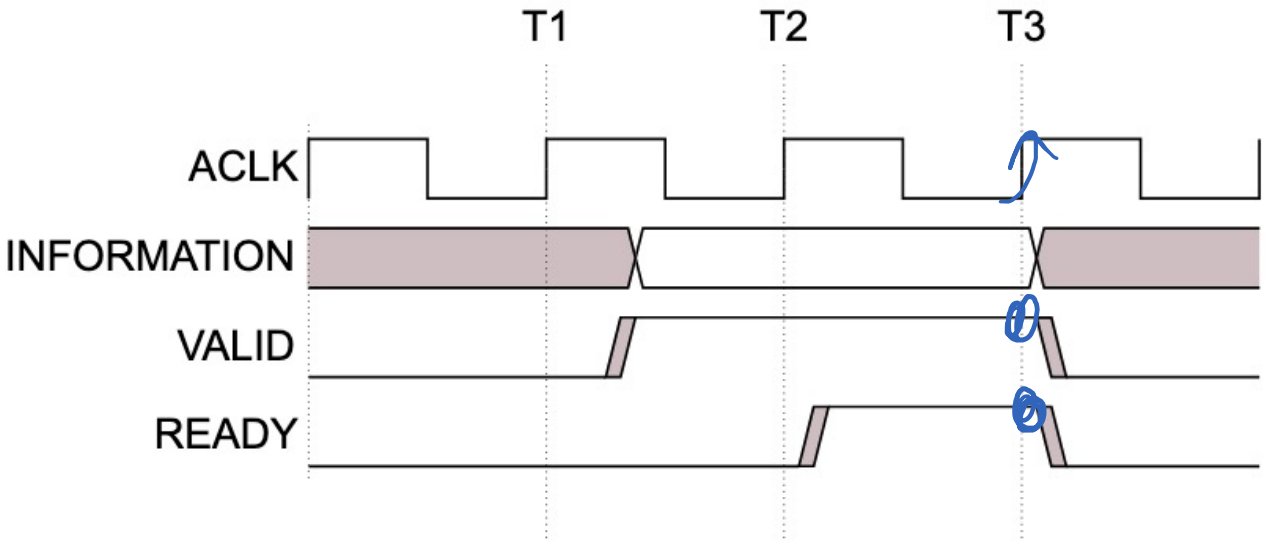
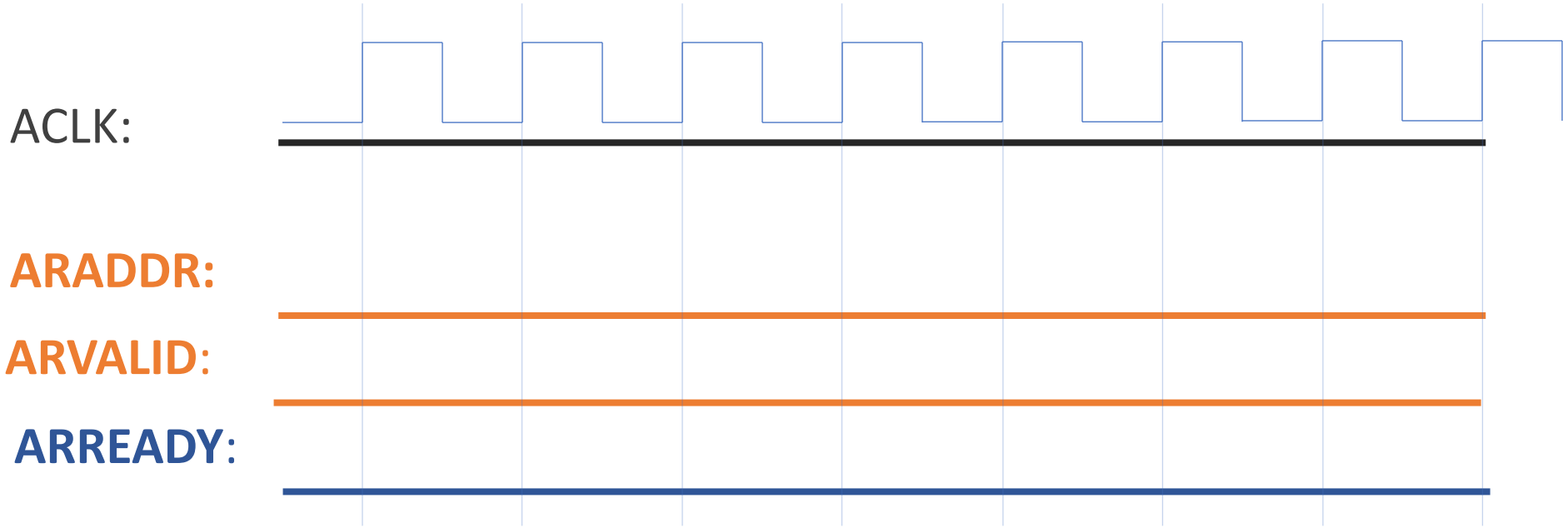
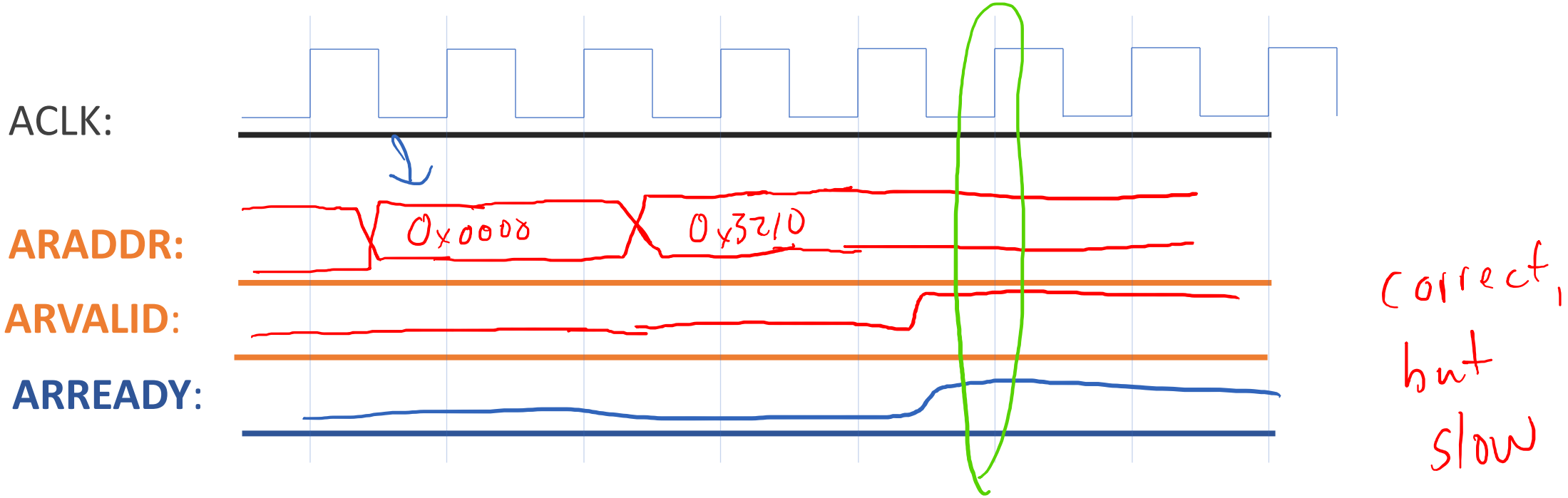


Figure A3-2 VALID before READY handshake

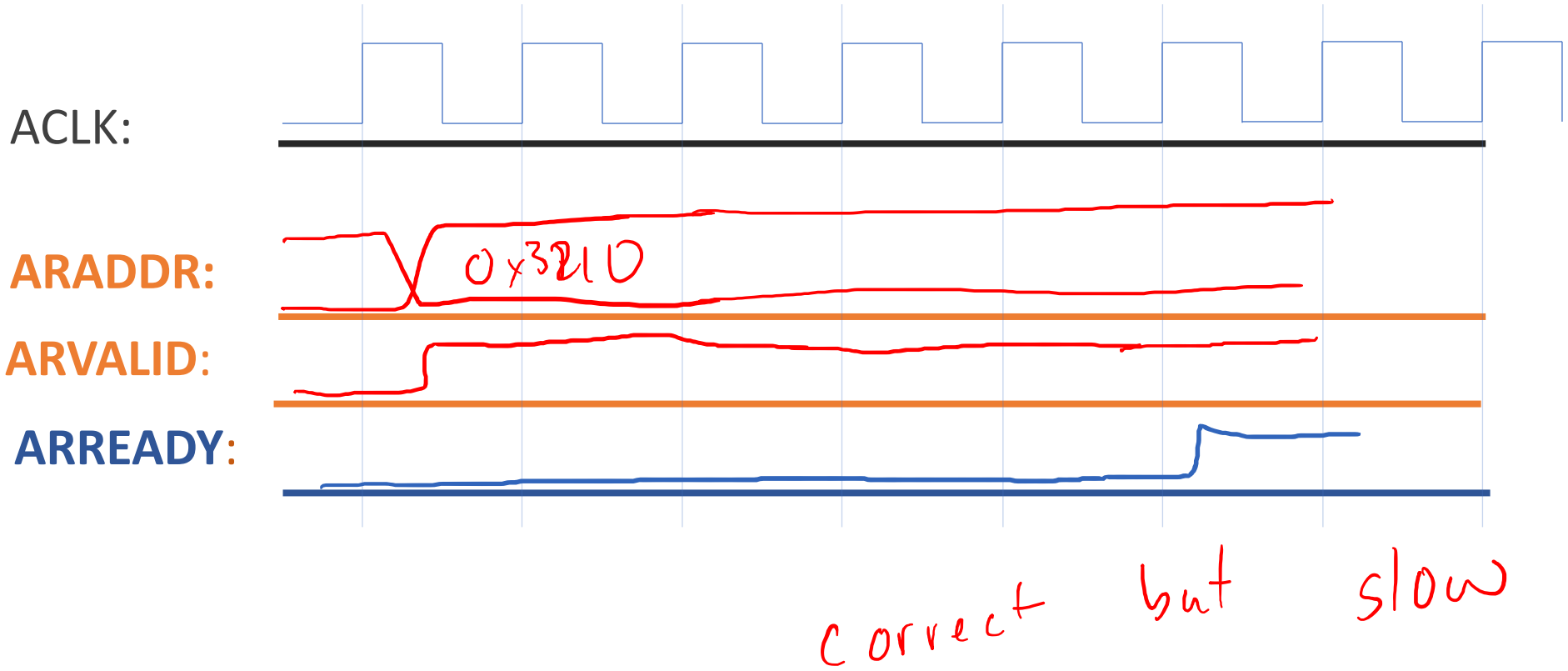
AXI4 Lite Read Transaction

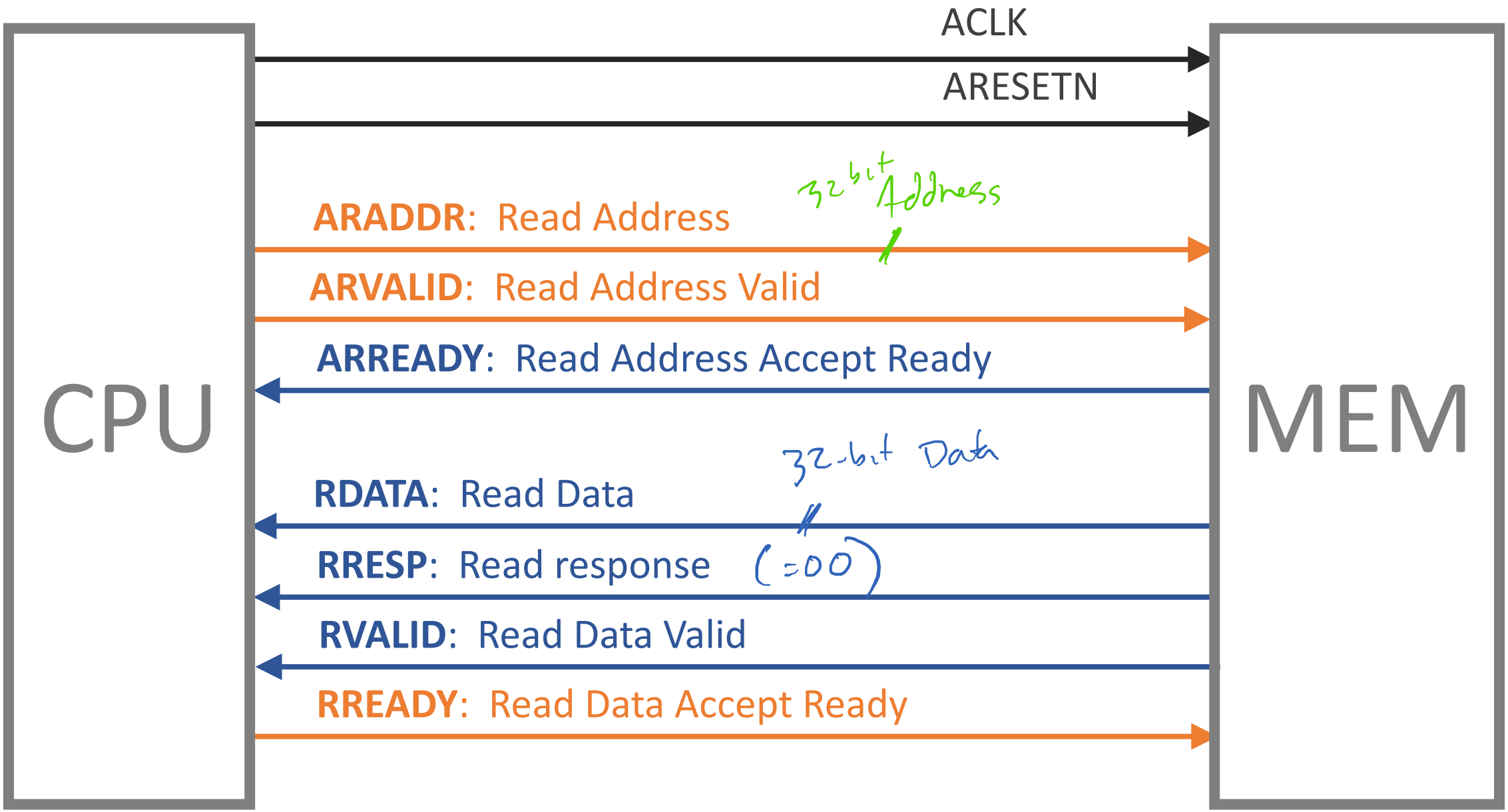


What if?


ACLK:
ARADDR:
0x0008
0x3210
ARVALID:
ARREADY:
correct, but slow

What if?




ACLK
ARESETN
CPU
MEM
ARADDR: Read Address
32 bit Address
ARVALID: Read Address Valid
ARREADY: Read Address Accept Ready
32-bit Data
RDATA: Read Data
RRESP: Read response
(=00)
RVALID: Read Data Valid
RREADY: Read Data Accept Ready

What is RRESP?

Table A3-4 RRESP and BRESP encoding

RRESP[1:0] BRESP[1:0]	Response
0b00	OKAY
0b01	EXOKAY
0b10	SLVERR
0b11	DECERR

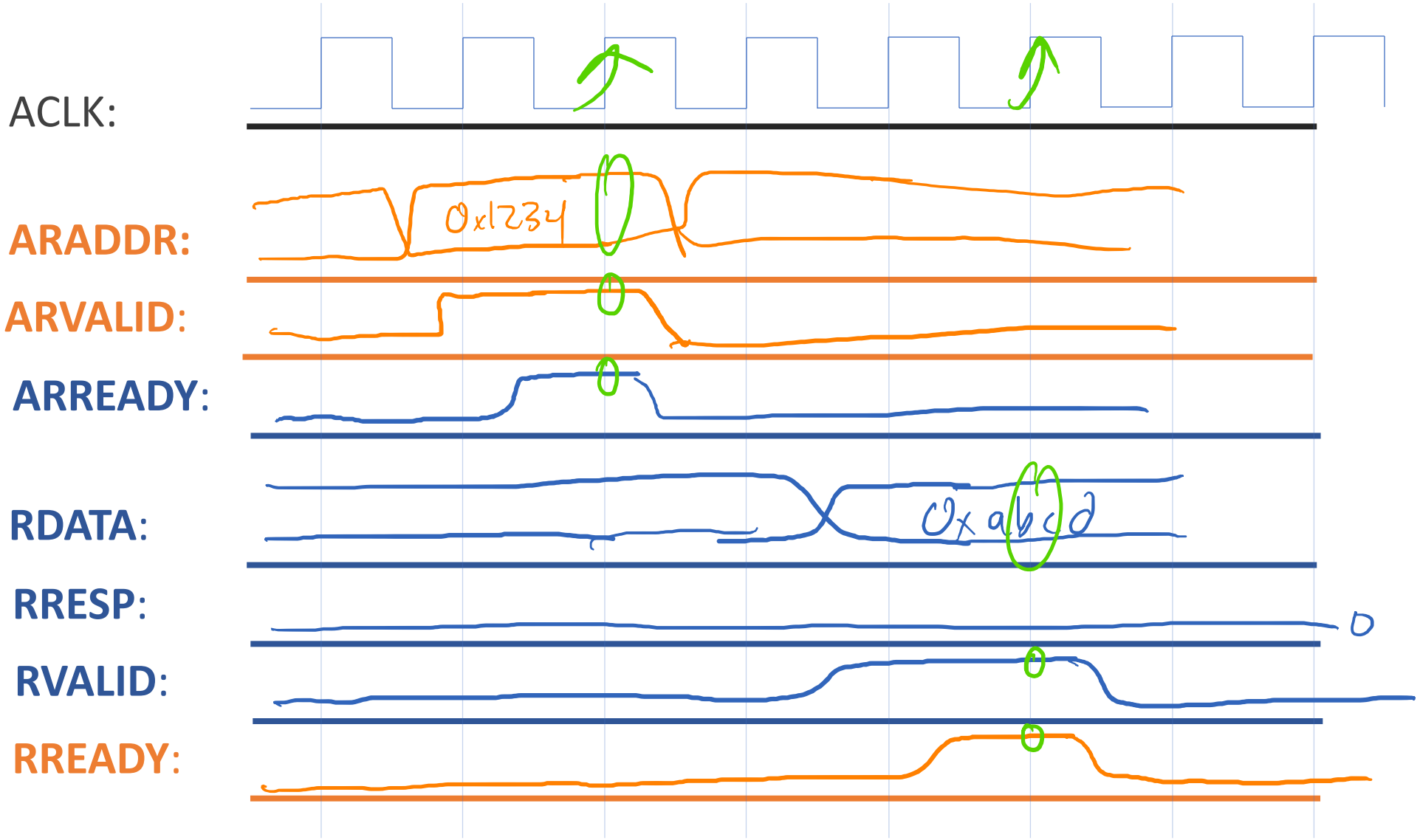
- Mostly used to send error codes back to CPU
- We'll always just use 0b00

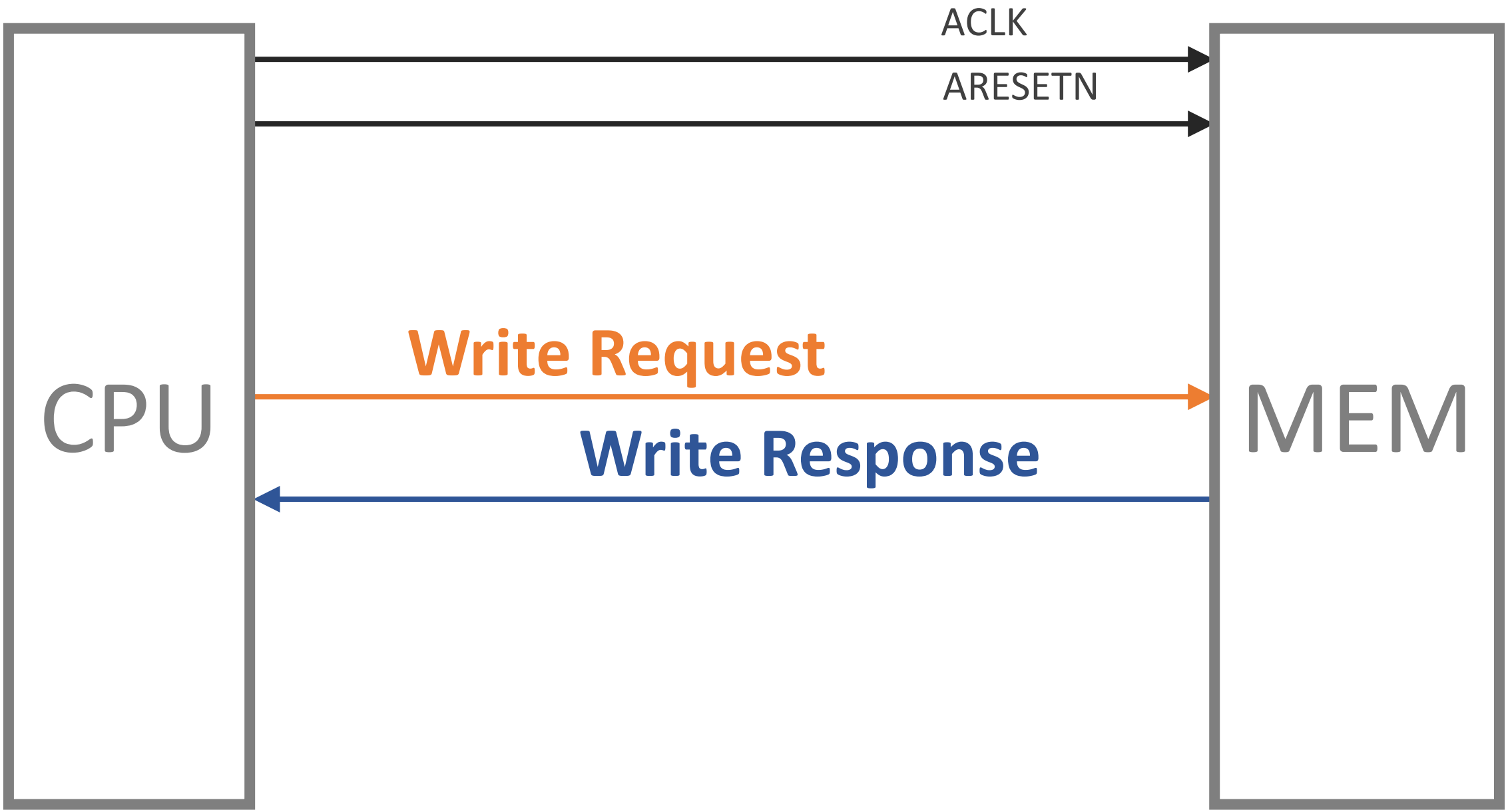
Load 0x1234, response: 0xabcd

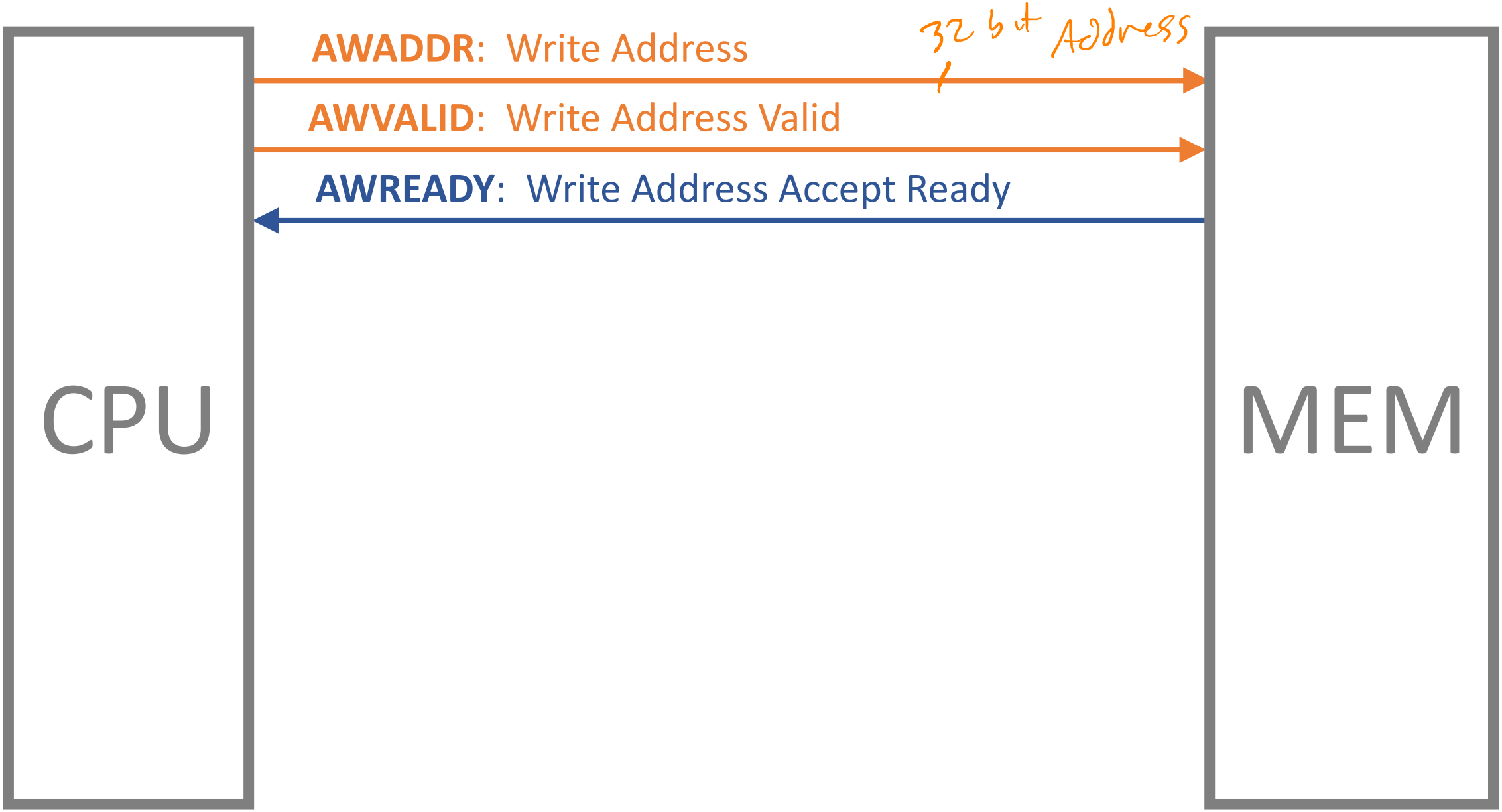
Address

Data

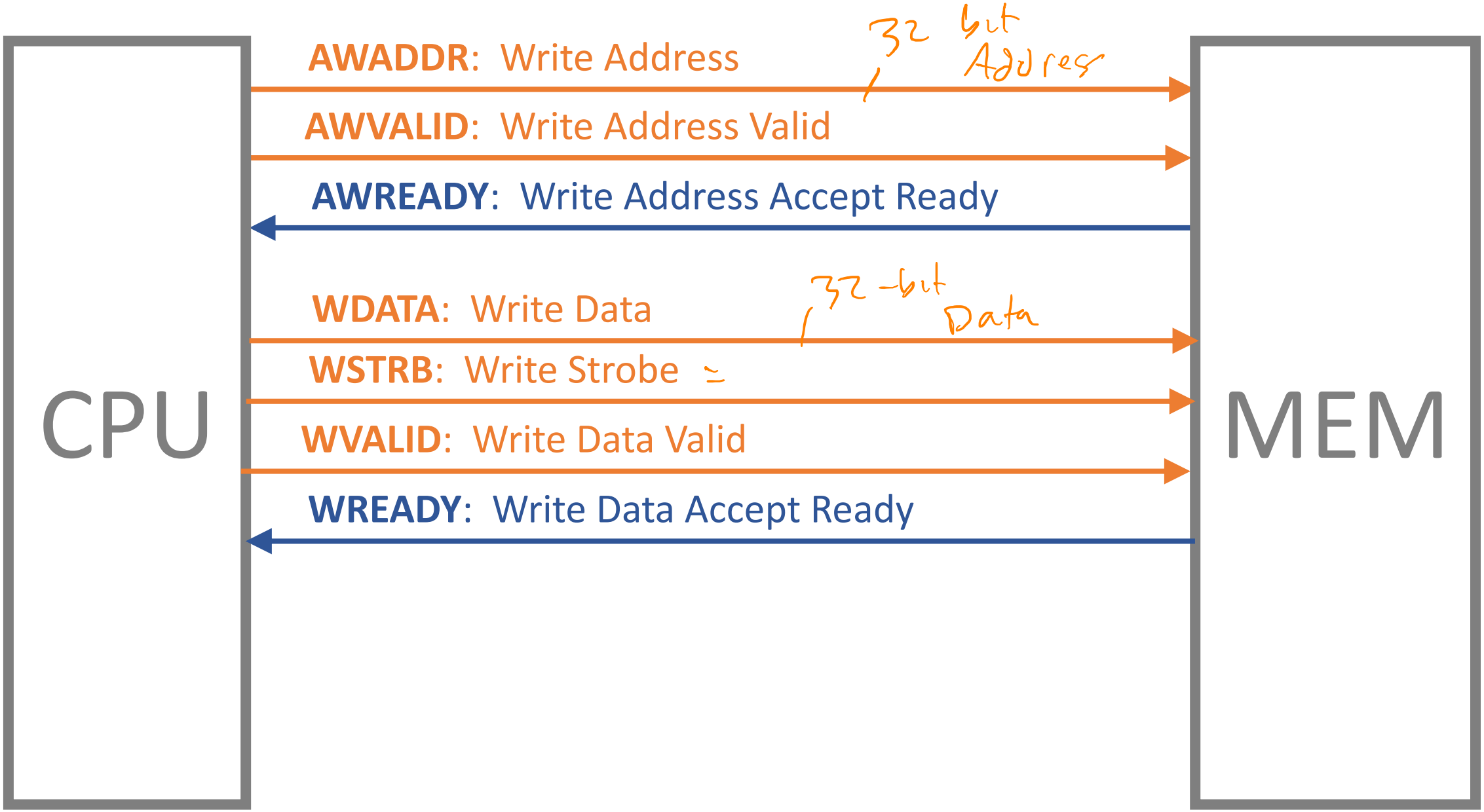
assume ARESETN = 1




ACLK
ARESETN
Write Request
Write Response
CPU
MEM



ACLK and ARESETN not shown



ACLK and ARESETN not shown

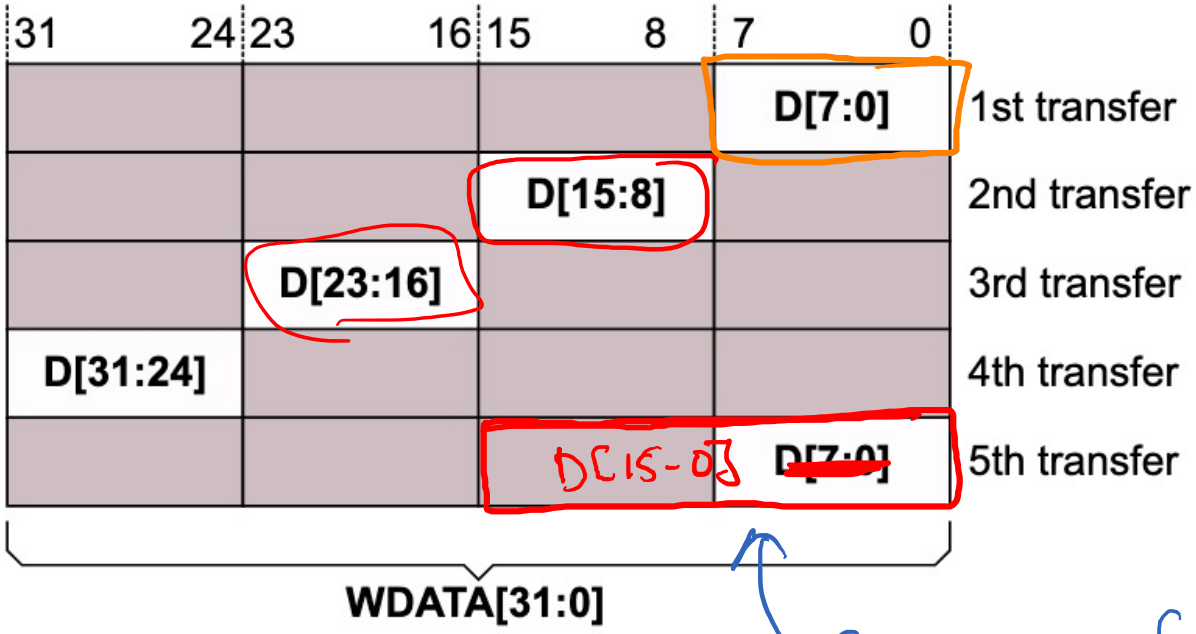
Q: How do you send a 1-byte (8-bit) value on a 32-bit bus?

- A: WSTB: Write Strobe

What is WSTRB?

The **WSTRB[n:0]** signals when HIGH, specify the byte lanes of the data bus that contain valid information. There is one write strobe for each eight bits of the write data bus, therefore **WSTRB[n]** corresponds to **WDATA[(8n)+7: (8n)]**

What is WSTRB here?



fix in future

Figure A3-8 Narrow transfer example with 8-bit transfers

MSB LSB

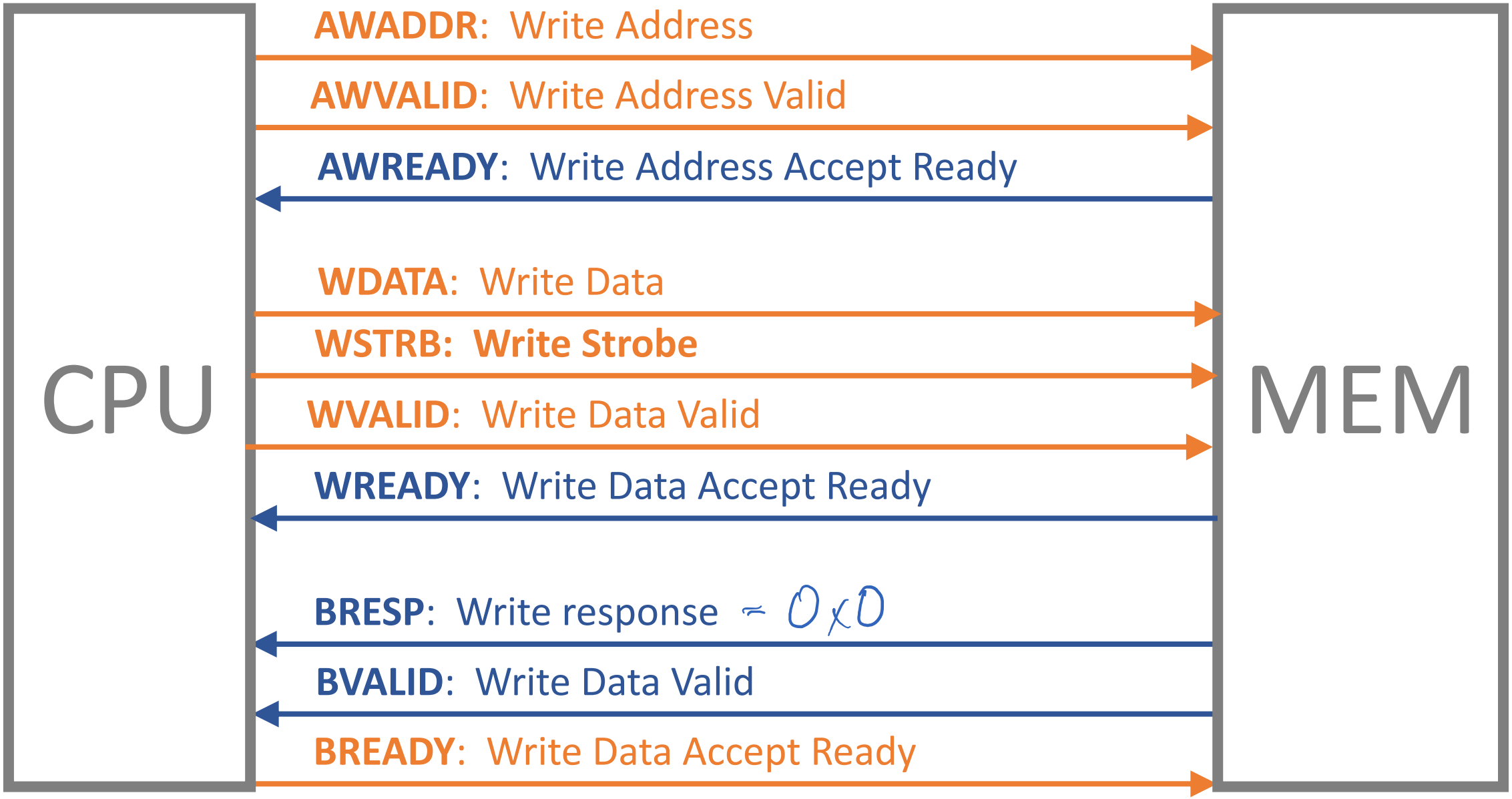
WSTRB = 0001 = 0x1

= 0010 = 0x2

= 0100 = 0x4

= 1000 = 0x8

= 0011 = 0x3



ACLK and ARESETN not shown

BRESP is just like RRESP

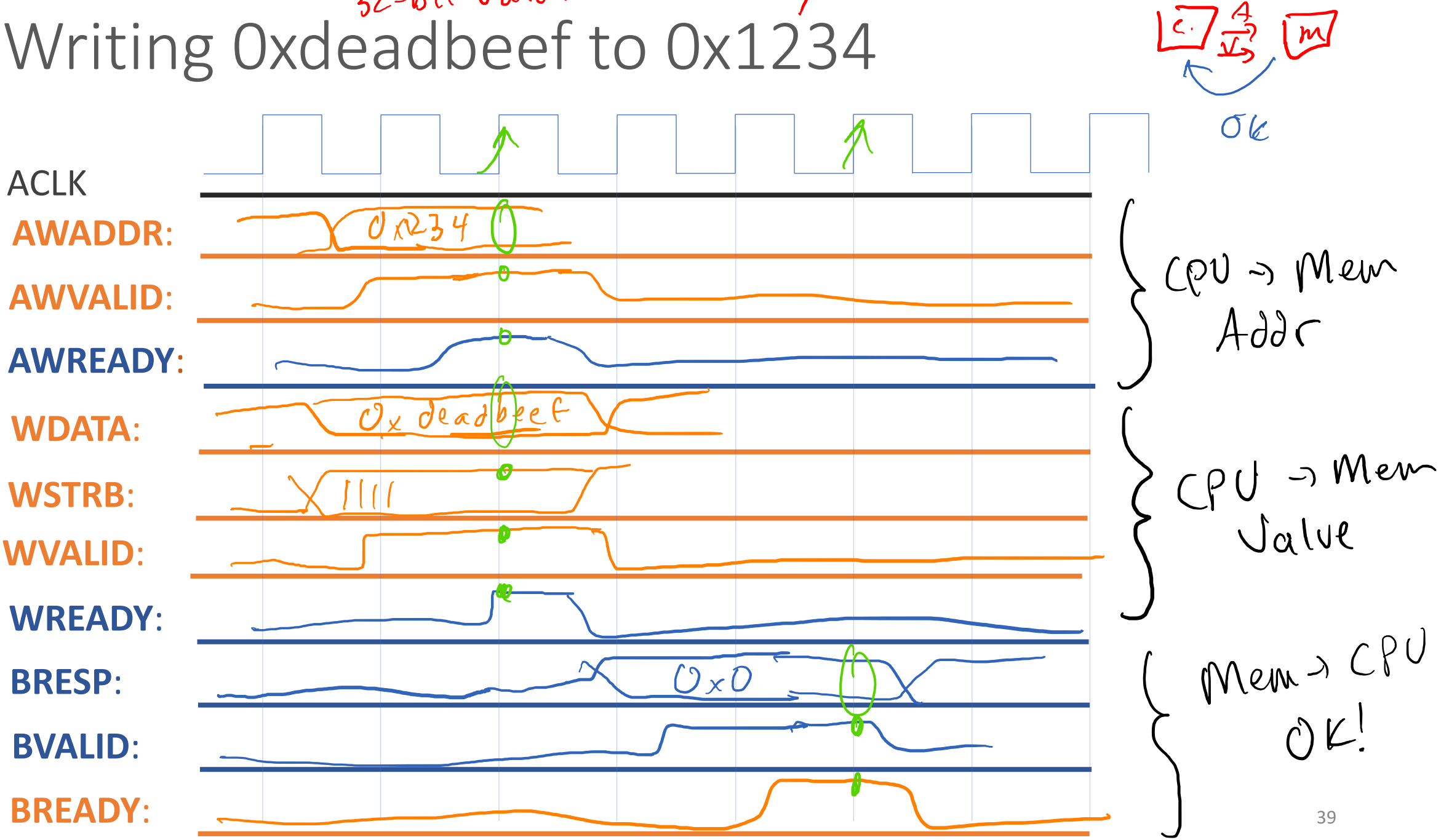
Table A3-4 RRESP and BRESP encoding

RRESP[1:0] BRESP[1:0]	Response
0b00	OKAY
0b01	EXOKAY
0b10	SLVERR
0b11	DECERR

- Mostly used to send error codes back to CPU
- We'll always just use 0b00

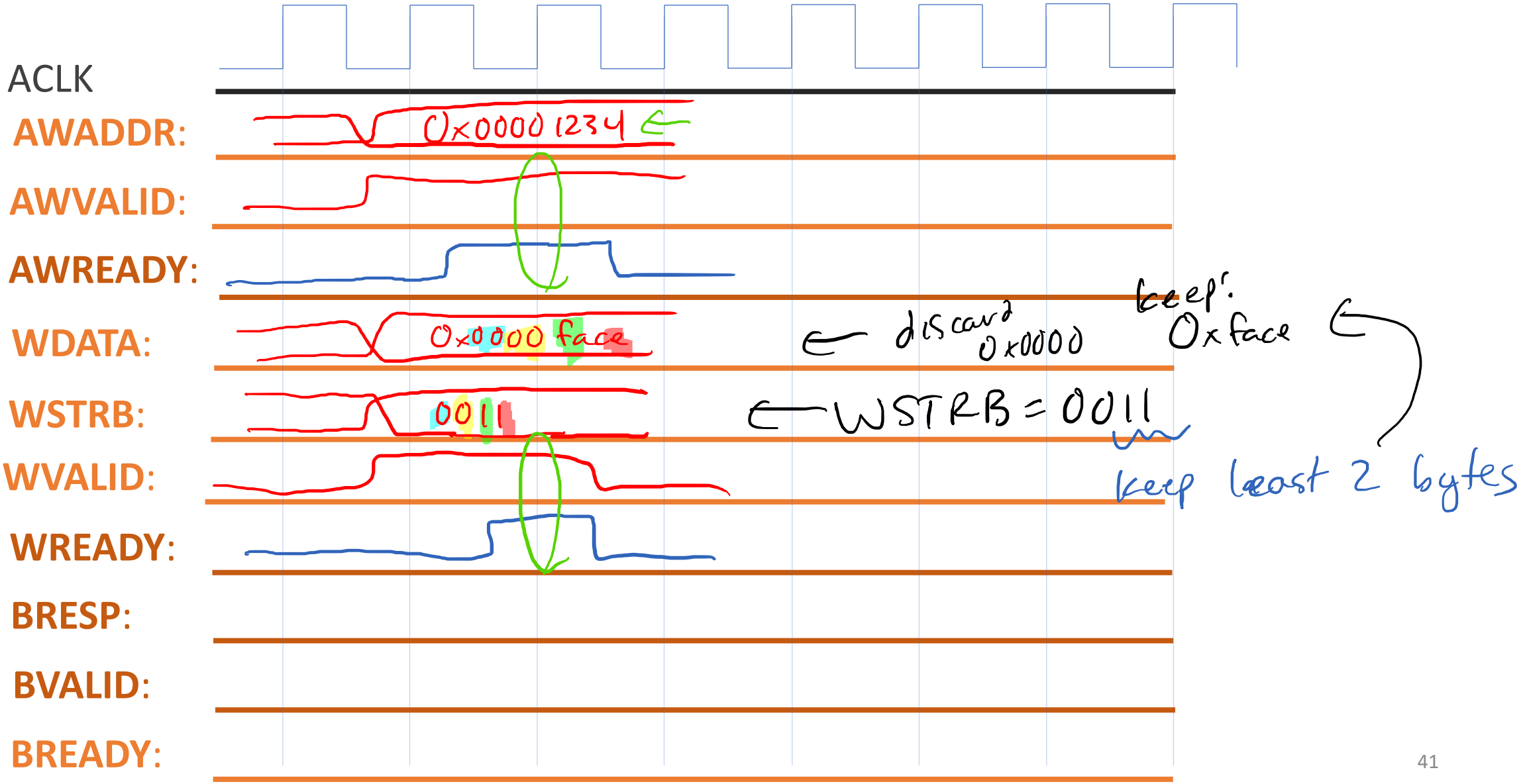
Writing 0xdeadbeef to 0x1234

32-bit value



Writing 0xface to 0x1234

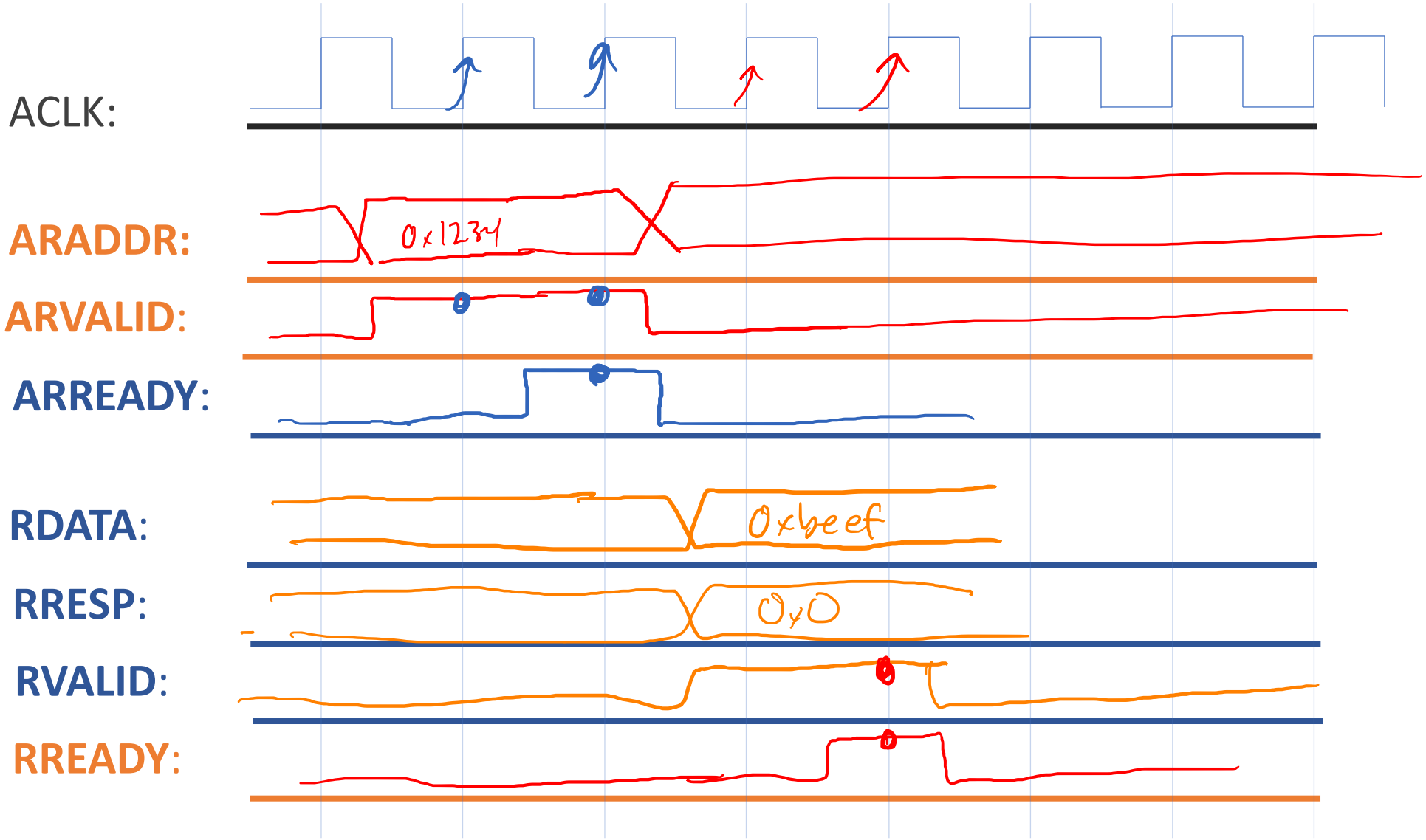
16-bit



ARM AXI Bus

- “Advanced eXtensible Interface” Bus Version 4, “AXI4”
- Three Variants
  - AXI4: Fast but complicated; Memory-mapped
  - AXI4 Lite: Slow but simple; Memory-mapped
  - AXI4 Stream: Fast and simple; Not memory-mapped

How long does a read(load) take?



High-Performance Bus Ideas

- Make single transaction faster

AXI Handshake Speedup

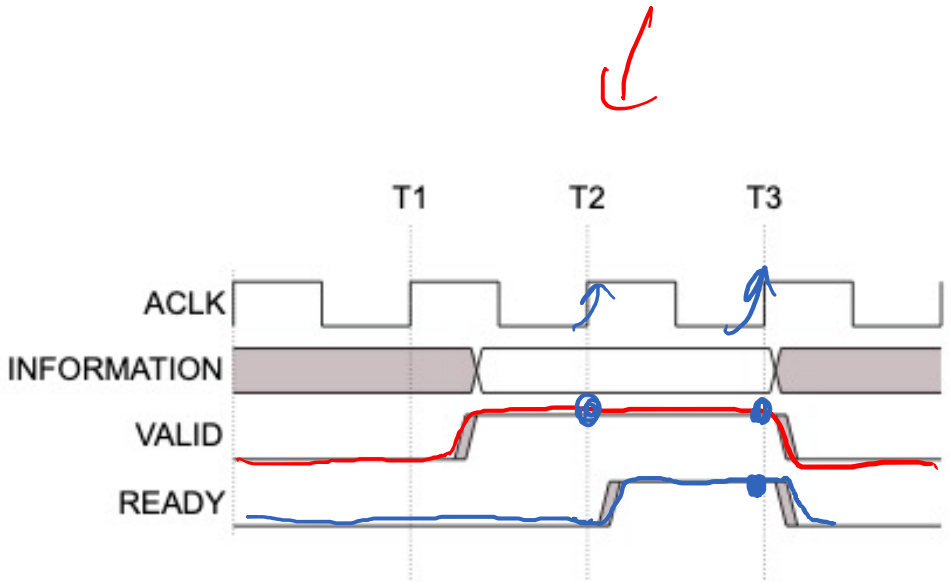


Figure A3-2 VALID before READY handshake

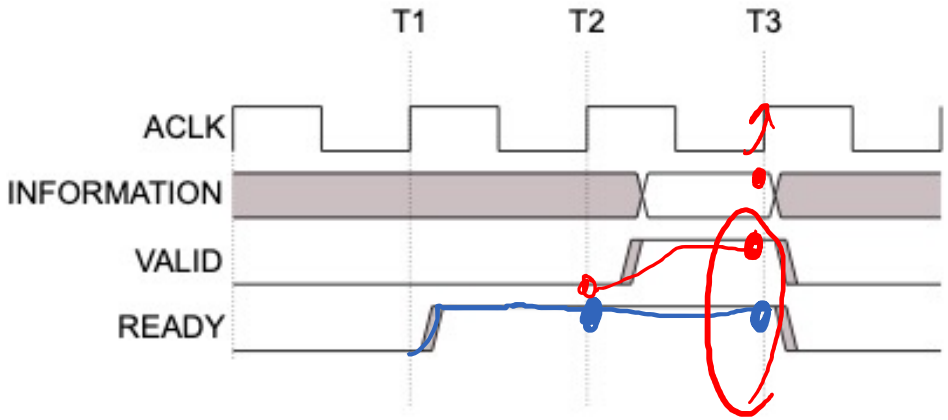
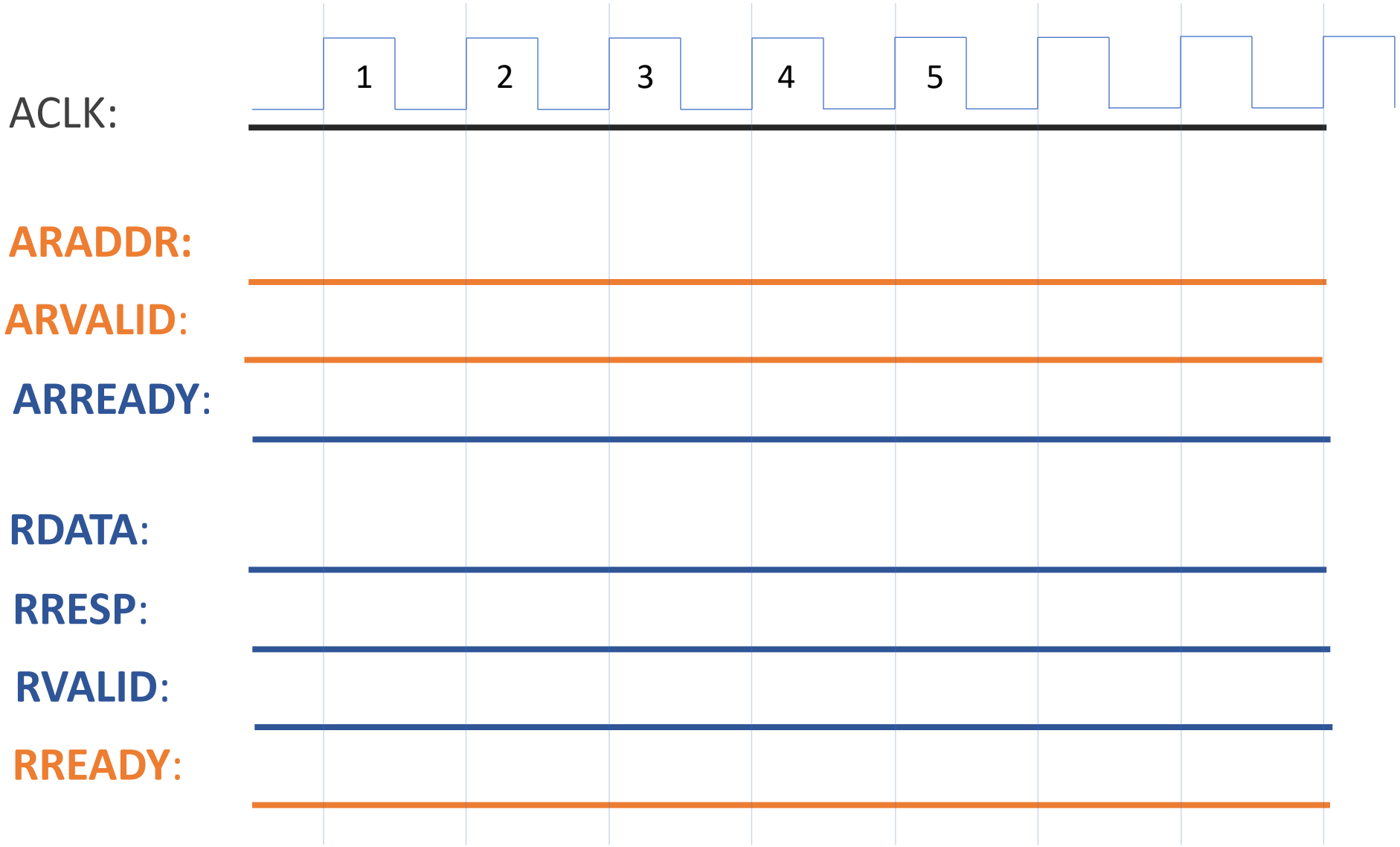


Figure A3-3 READY before VALID handshake

- Both are valid
- Right is faster

What can we do to make this faster?



High-Performance Bus Ideas

- Make single transaction faster
- Overlap multiple transactions

Next Time

- High-Performance Busses

~~Monday~~!

Tuesday

References

- https://www.youtube.com/watch?v=okiTzvihHRA
- https://web.eecs.umich.edu/~prabal/teaching/eecs373/
- https://en.wikipedia.org/wiki/File:Computer_system_bus.svg
- https://www.realdigital.org/doc/a9fee931f7a172423e1ba73f66ca4081

- AMBA® AXI™ and ACE™ ProtocolSpecification

08: AXI4 Lite

Engr 315: Hardware / Software Codesign

Andrew Lukefahr
Indiana University