Exam Review

Engr 315: Hardware / Software Codesign

Andrew Lukefahr

Indiana University

Some material taken from EECS370 at U. of Michigan

Announcements

- P7: Due Friday.
  - New SD Card / Linux Image

- Exam: on Monday

- P8: Coming Soon.

Exam Details

- Main 5 sections
  - Multiple questions / section

- Some short answer
- Some fill-in-the blank/code/table

A “Cheat” Sheet is Allowed

- 2-sided
- 8.5”x11” paper
- Handwritten (not photocopied)

Major Topics

- Performance Profiling
- Data Structures

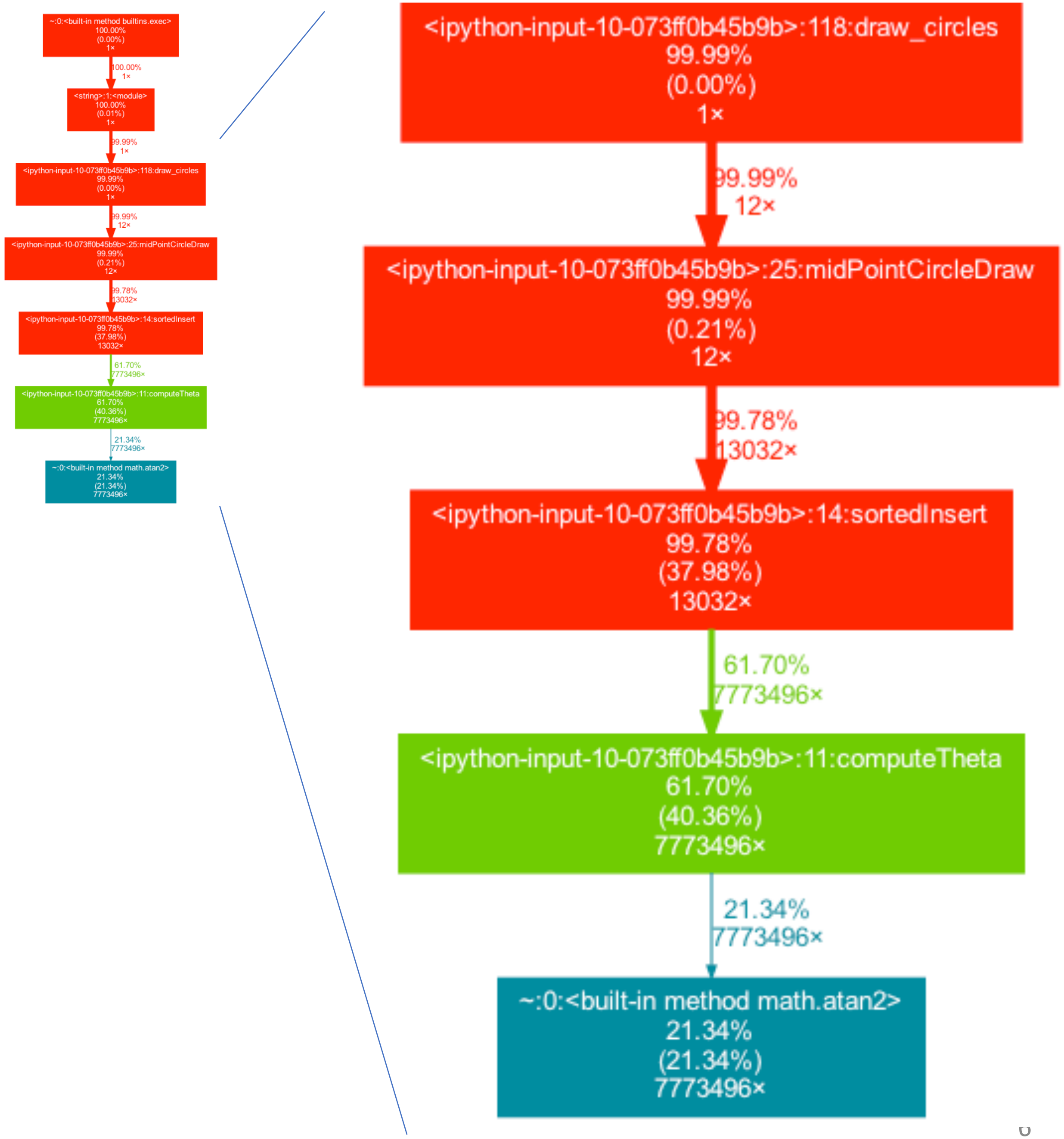
- Bus Interfaces
- MMIO
- DMA

Performance Profiling

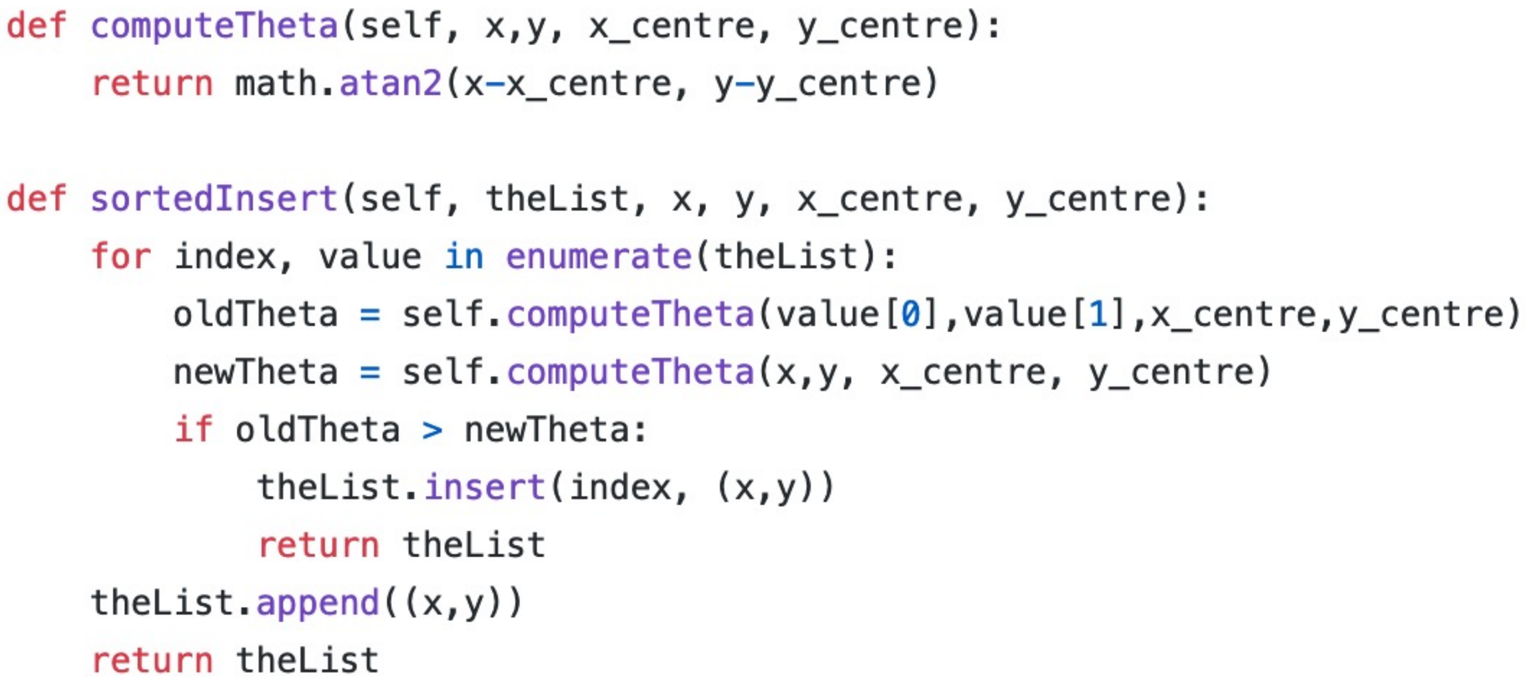
- What is profiling?

- What function should we be optimizing here?

- How do you know?


~:0:<built-in method builtins.exec>
100.00%
(0.00%)
1×
100.00%
1×
<string>:1:<module>
100.00%
(0.01%)
1×
99.99%
1×
<ipython-input-10-073ff0b45b9b>:118:draw_circles
99.99%
(0.00%)
1×
99.99%
12×
<ipython-input-10-073ff0b45b9b>:25:midPointCircleDraw
99.99%
(0.21%)
12×
99.78%
13032×
<ipython-input-10-073ff0b45b9b>:14:sortedInsert
99.78%
(37.98%)
13032×
61.70%
7773496×
<ipython-input-10-073ff0b45b9b>:11:computeTheta
61.70%
(40.36%)
7773496×
21.34%
7773496×
~:0:<built-in method math.atan2>
21.34%
(21.34%)
7773496×

Algorithm Tuning



```
def computeTheta(self, x,y, x_centre, y_centre):
    return math.atan2(x-x_centre, y-y_centre)

def sortedInsert(self, theList, x, y, x_centre, y_centre):
    for index, value in enumerate(theList):
        oldTheta = self.computeTheta(value[0],value[1],x_centre,y_centre)
        newTheta = self.computeTheta(x,y, x_centre, y_centre)
        if oldTheta > newTheta:
            theList.insert(index, (x,y))
            return theList
    theList.append((x,y))
    return theList
```

Data Structures

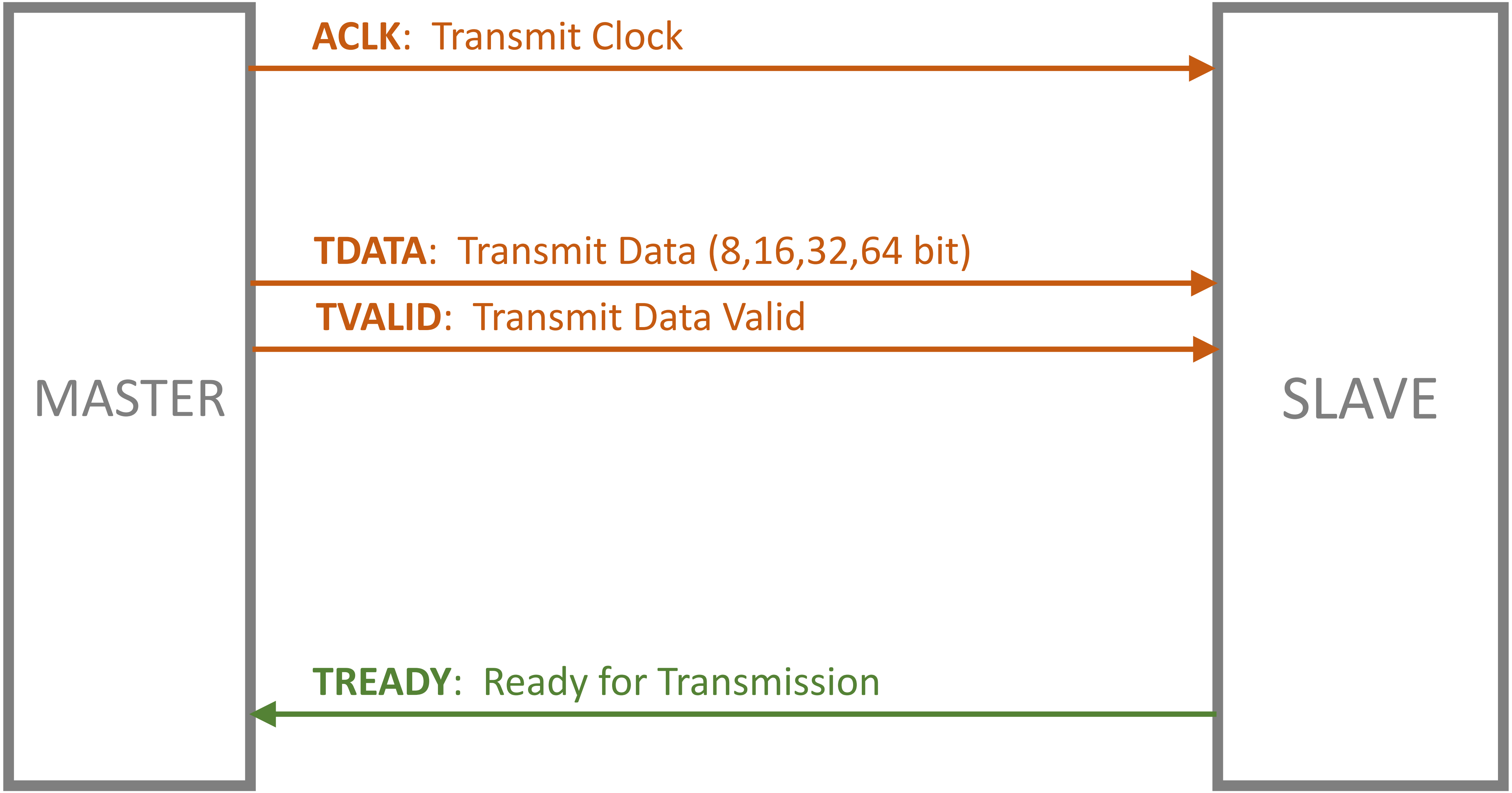
- When is better here? list or array?
  - Inserting at the beginning?
  - Accessing the element at position N (i.e. values[n])
  - Accessing elements sequentially?

- What’s funny about Python’s lists?

Bus Interfaces

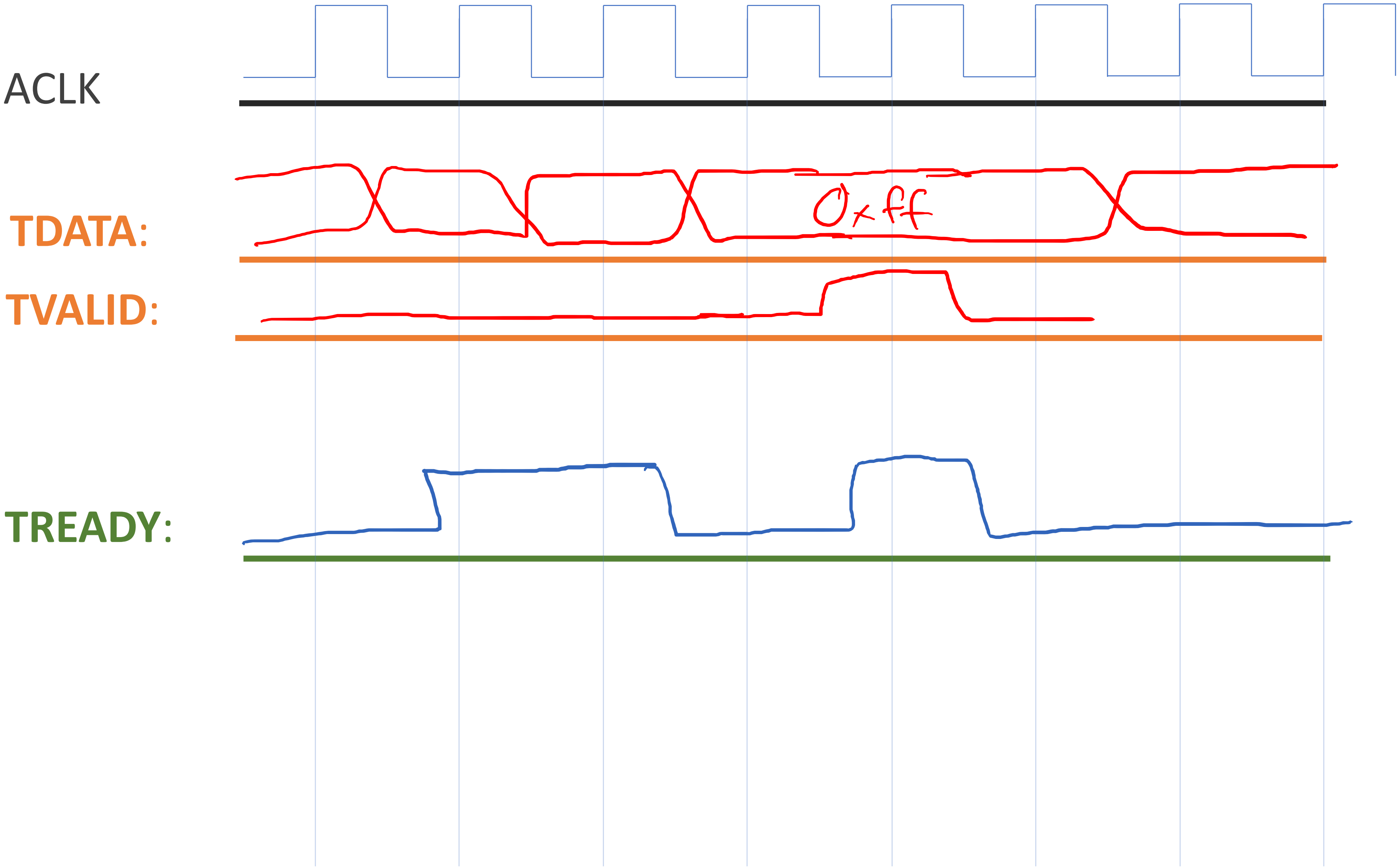
- AXI4 “Full” vs. AXI4 Lite vs. AXI4 Stream

- What are the benefits of each?
- Where do we use them?

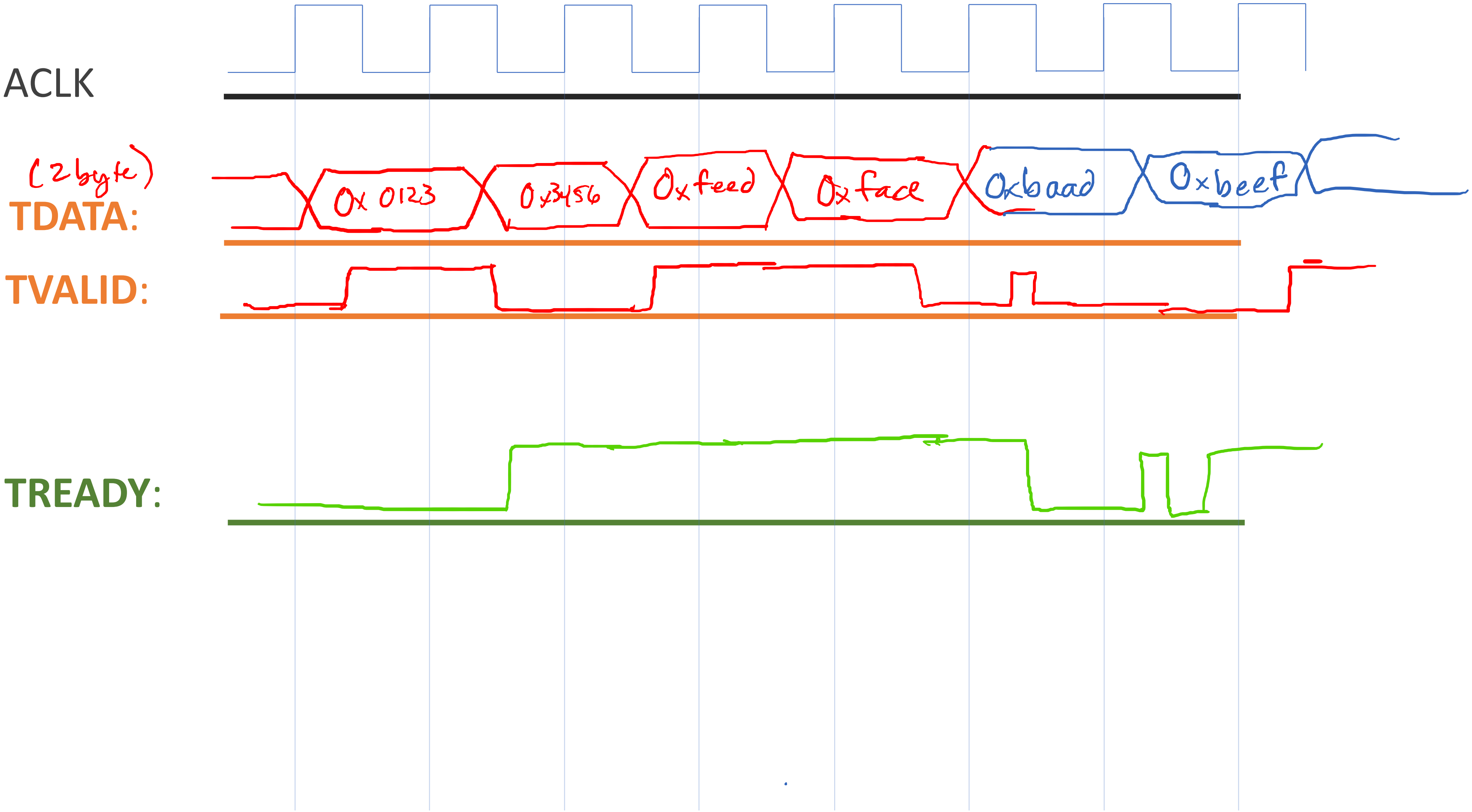

ACLK: Transmit Clock
TDATA: Transmit Data (8,16,32,64 bit)
TVALID: Transmit Data Valid
MASTER
SLAVE
TREADY: Ready for Transmission

When is a transmission valid?

Transferring data on a AXI4-Stream Bus.



Transferring data on a AXI4-Stream Bus.



MMIO

- Define MMIO?
- What is MMIO?
- Why do we use it?

MMIO Loads

- In ASM?

```
    mov r2, 0x40000000 ;
    ldr r3, [r2, 0x144];
```

- In C?

MMIO Loads and Stores

- In ASM?

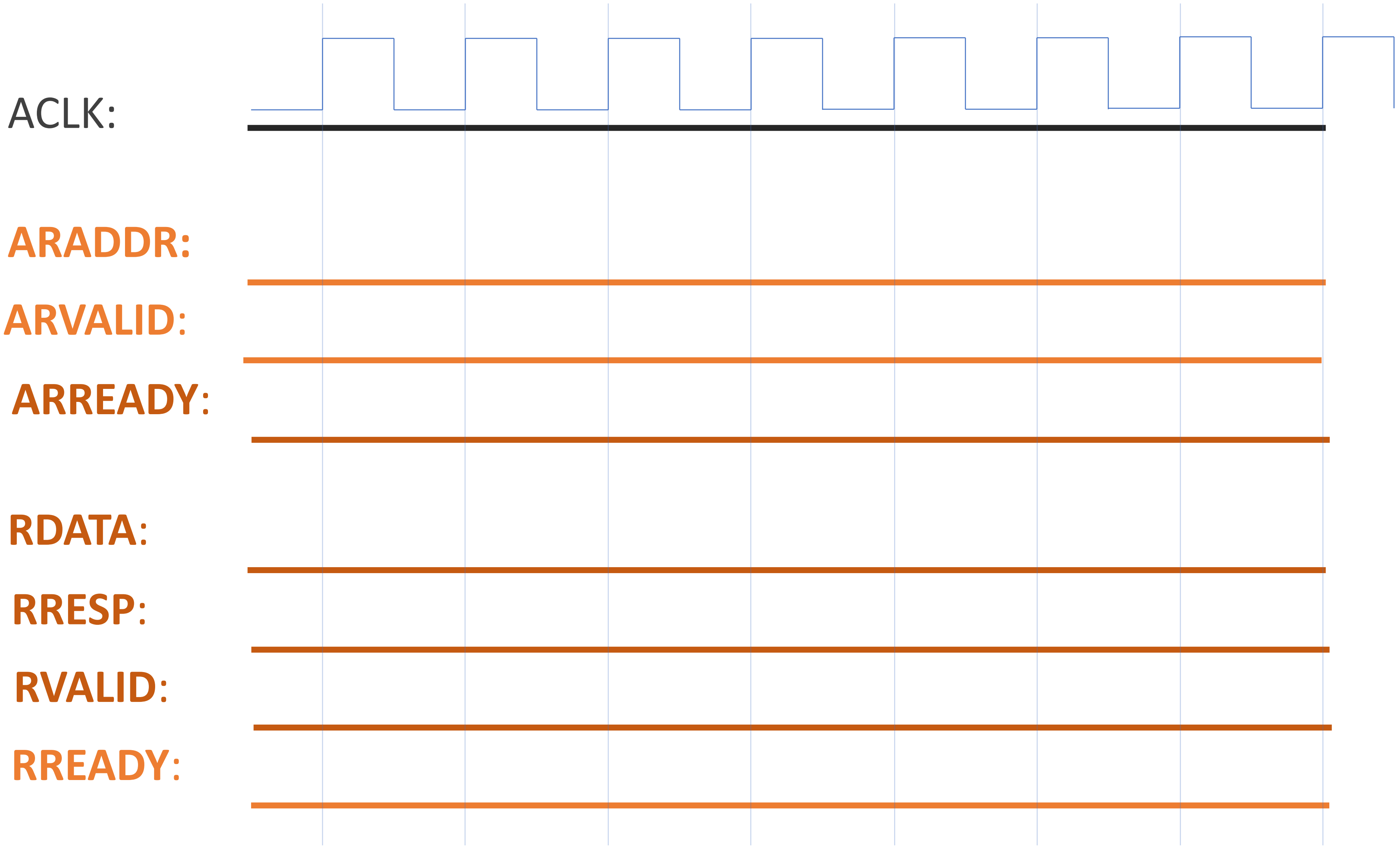
```
mov r2, 0x40000000 ;
ldr r3, [r2, 0x144];
```

- In C?

```
uint32_t  x = *(volatile uint32_t *)(0x40000144);
```

Store: *(volatile uint32_t *)(0x40000144) = 32;

AXI4Lite: Load 0x1234, response: 0xabcd



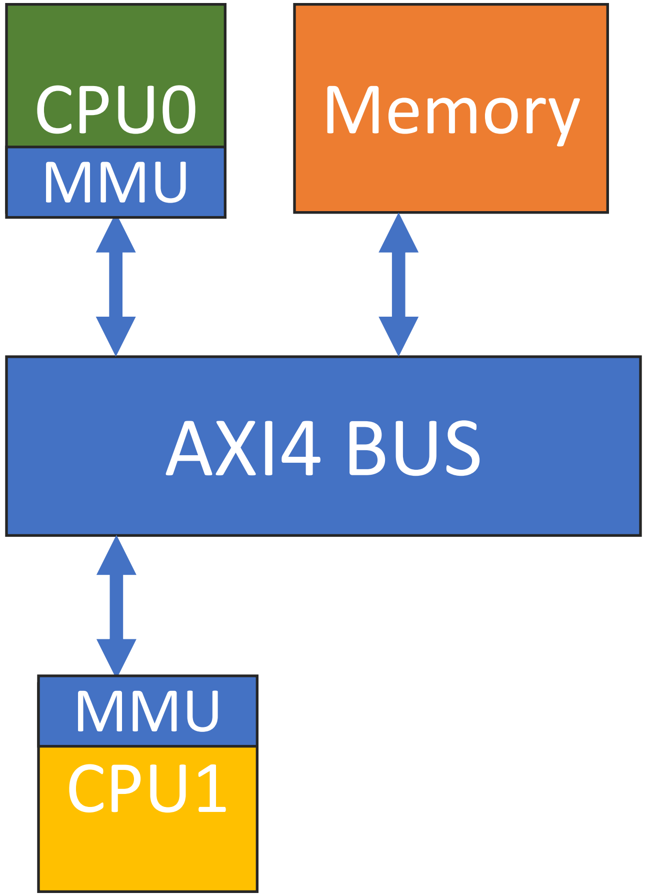
Linux MMIO?

- What’s weird about C/MMIO with Linux?

Virtual Memory

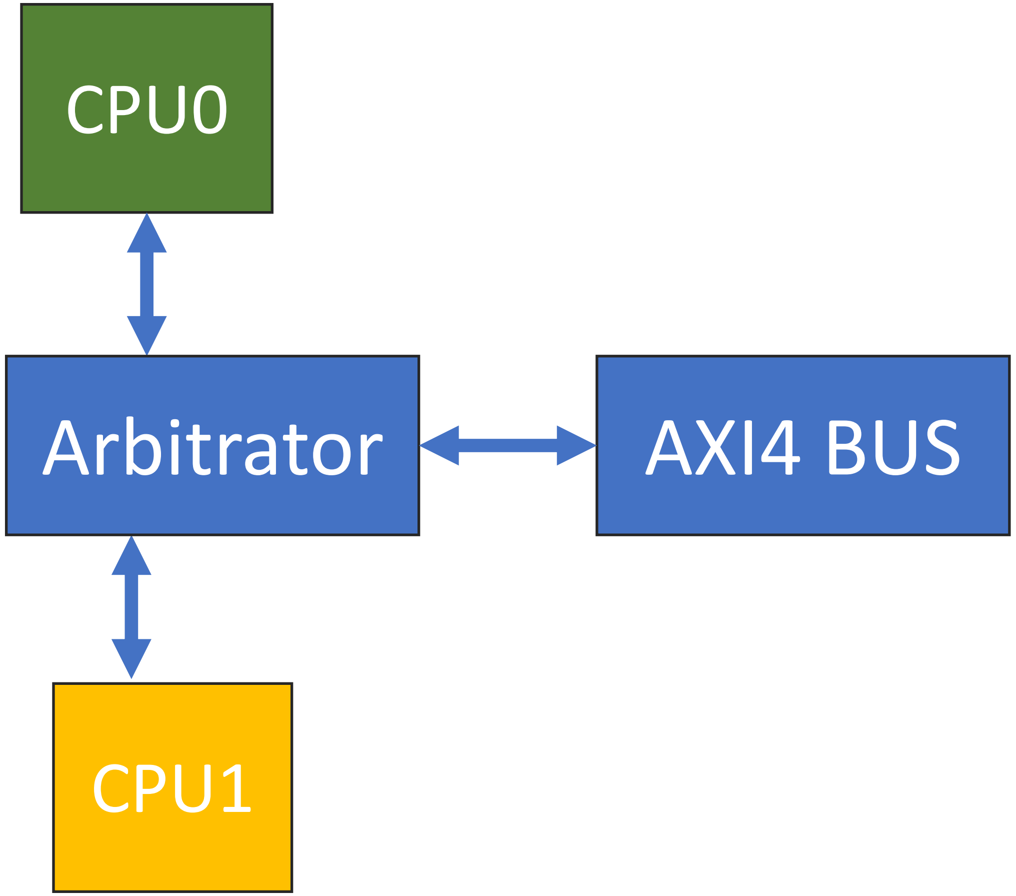
- Linux/Hardware “translates” CPU’s memory addresses into real memory addresses.
- CPU   physical address
- Memory   virtual address

Multiple Masters



- What happens if both CPUs request a transaction at the same time?

An Arbitrator selects who gets to use the bus

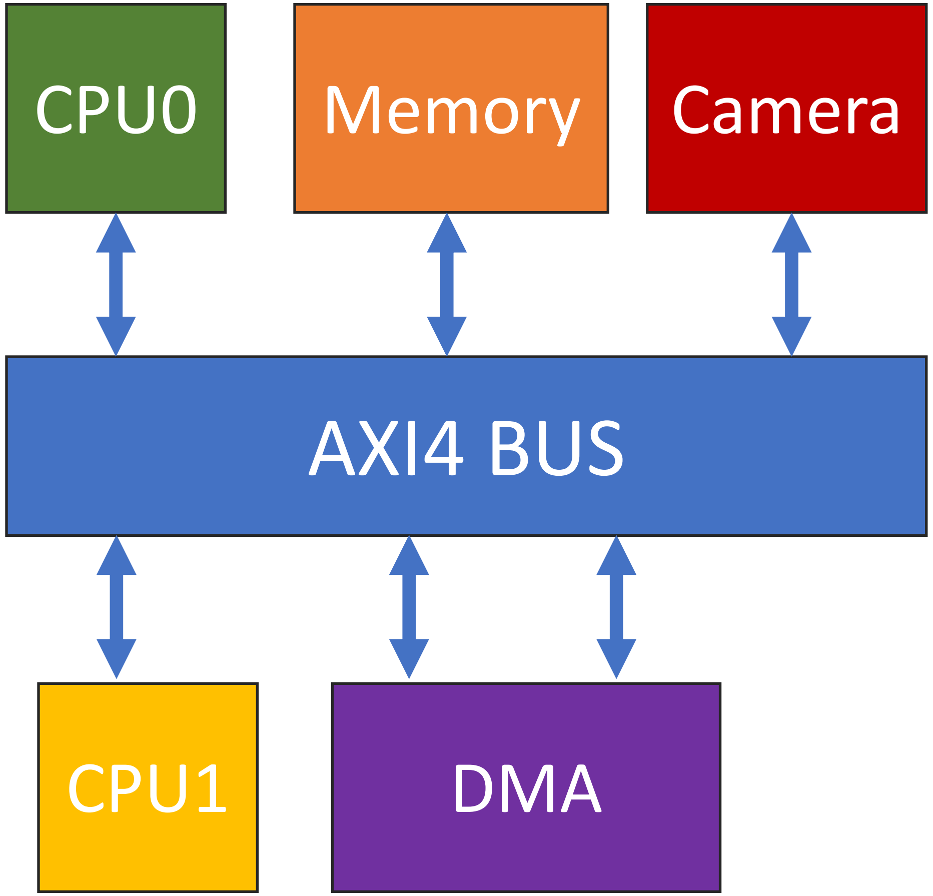


- What happens if both request a transaction at the same time?

- Arbitration: Pick a winner!

- What Arbitration scheme to use?

DMA



- Define DMA?
- What's the goal of DMA?
- What steps are involved?

DMA Control Design

```
void dma (uint32_t * from,
              uint32_t * to,
              uint32_t size)
{
   register uint32_t reg;

   for (int i = 0; i < size; ++i){

      reg = from[i]; //load

      to[i] = reg; //store
   }
}
```

DMA Control Design

```
void dma (uint32_t * from,
              uint32_t * to,
              uint32_t size)
{
   register uint32_t reg;

   for (int i = 0; i < size; ++i){

      reg = from[i]; //load

      to[i] = reg; //store
   }
}
```

- What interfaces do you need?
- How do you start/stop DMA?
- Design a DMA state machine?

All DMA Registers

Control - 0x0400	31-1 Reserved	0 Start
Status - 0x0404	31-1 Reserved	0 Done
Source - 0x0408	31-0 DMA Source Address	
Destination - 0x040C	31-1 DMA Destination Address	
Size - 0x0410	31-16 Reserved	15-0 DMA Transfer Size (in Bytes)

DMA Control

```
void dma (uint32_t * from,
            uint32_t * to,
            uint32_t size)
{
   register uint32_t reg;

   for (int i = 0; i < size; ++i){
      reg = from[i]; //load
      to[i] = reg; //store
   }
}
```

- AXI4 Master Interface
  - Loads + Stores

- 5 MMIO registers
  - Control (Start)
  - Status (Done)
  - Source (`from`)
  - Destination (`to`)
  - `size` (in Bytes)

0x0400: Control Register
0x0404: Status Register
0x0408: Source Address
0x040C: Destination Address
0x0410: Transfer Size in Bytes

Using DMA from CPU’s side:

```
void dma (uint32_t * from,
          uint32_t * to,
          uint32_t size)
{









}
```

0x0400: Control Register
0x0404: Status Register
0x0408: Source Address
0x040C: Destination Address
0x0410: Transfer Size in Bytes

Using DMA from CPU’s side:

```
void dma_copy ( uint32_t * src,
                          uint32_t * dest,
                          uint32_t size){

        *((volatile uint32_t *)(0x0408))=src;
        *((volatile uint32_t *)(0x040C))=dest;
        *((volatile uint32_t *)(0x0410))=size;
        *((volatile uint32_t *)(0x0400))= 0x1; //start

        //spin until copy done
        while( *((volatile uint32_t *)(0x0404)) != 0x1){;}
}
```

DMA System Interface

Exam Review

Engr 315: Hardware / Software Codesign

Andrew Lukefahr

Indiana University