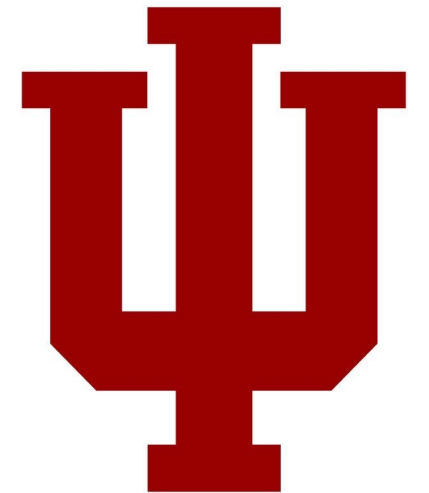


05: Buses II

Engr 315: Hardware / Software Codesign
Andrew Lukefahr
Indiana University



Announcements

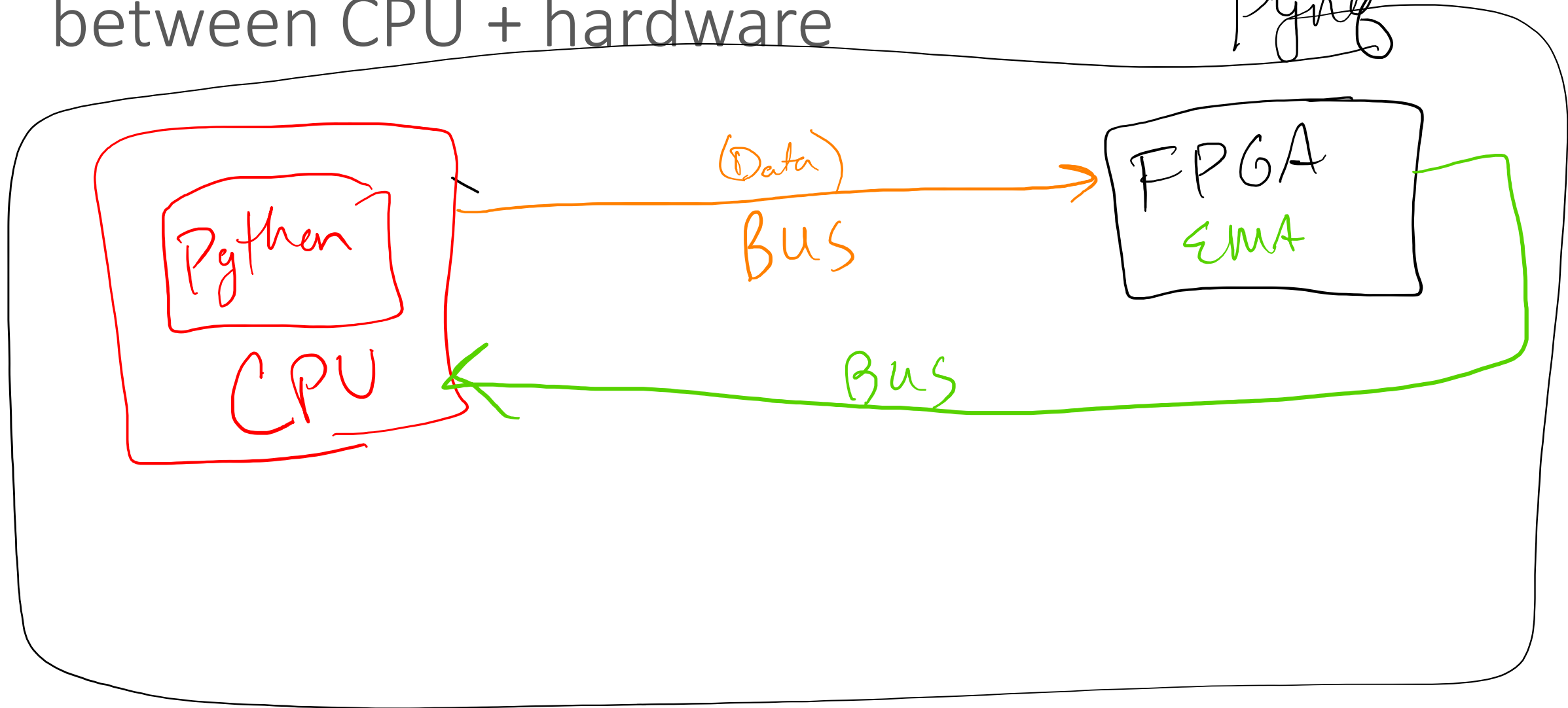
- Slack – See Website
- Office Hours – See Website / Syllabus
 - *Maybe change Thursday due to conflict
- P2: Due Wednesday
 - (New Project, could be some bumps)
 - Need a Pynq
 - Groups of 2 allowed
- P3: Out now!

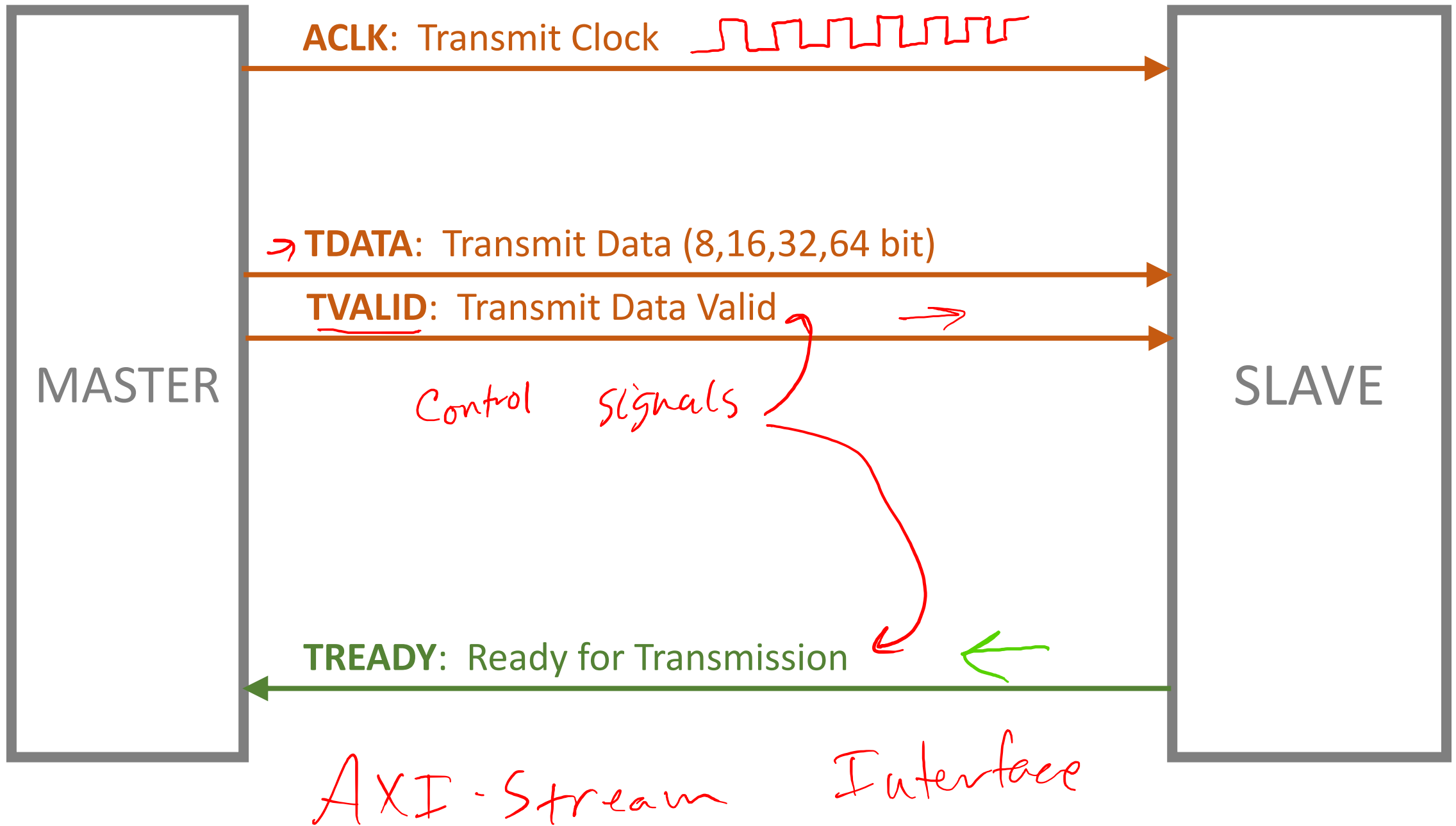
Bus terminology

- A “**transaction**” occurs between an “**initiator**” and “**target**”
- Any device capable of being an initiator is said to be a “**bus master**”
 - In many cases there is only one bus master (single master vs. multi-master).
- A device that can only be a target is said to be a “**slave device**”.

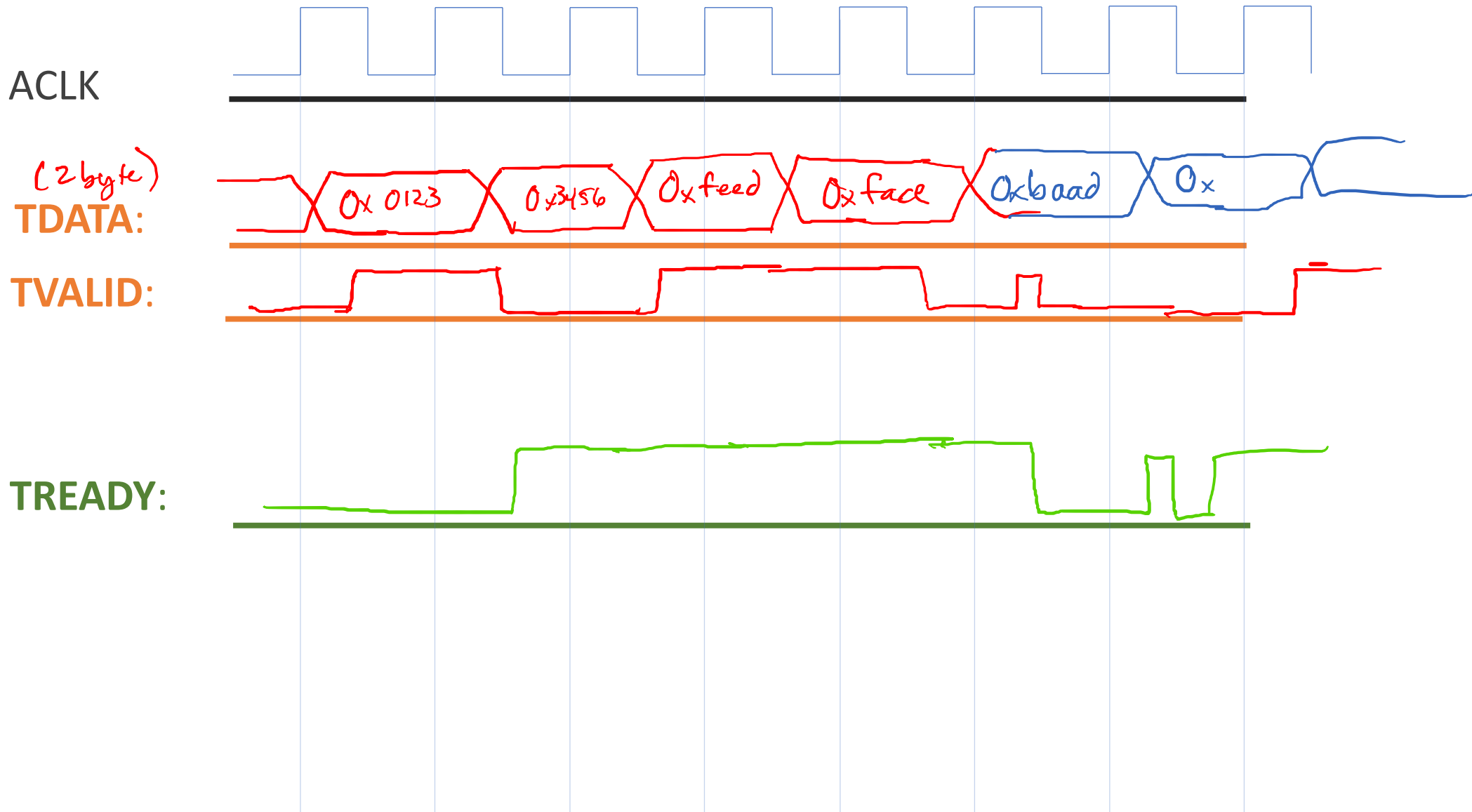
P3 “EMA” uses two buses to move data between CPU + hardware

Pyne





Transferring data on a AXI4-Stream Bus.



Data (**TDATA**) is only transferred when

TVALID is 1.

This indicates the **MASTER** is trying to transmit new data.

TREADY is 1.

This indicates the **SLAVE** is ready to receive the data.

If either **TVALID** or **TREADY** are 0, no data is transmitted.

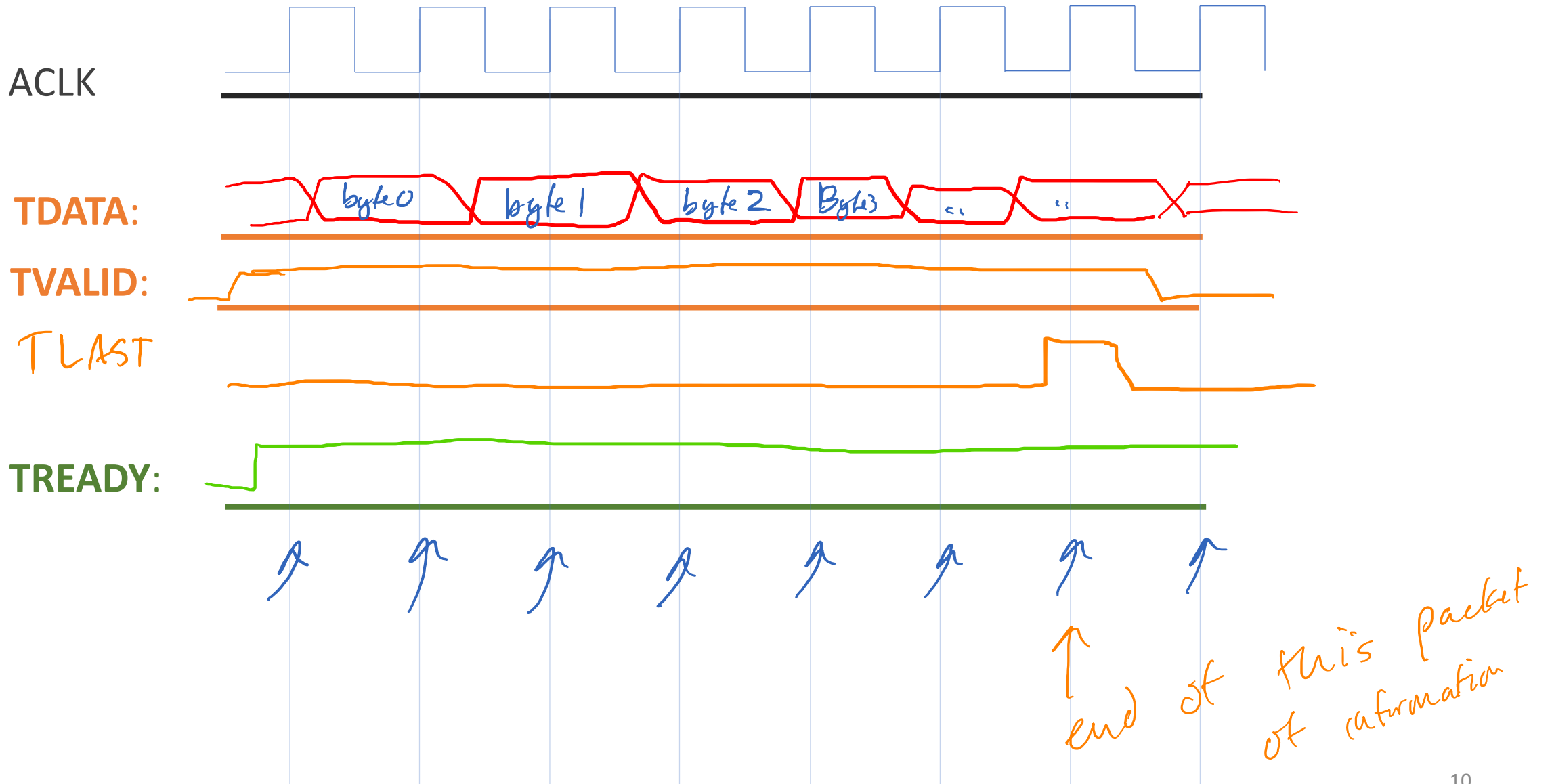
If **TVALID** and **TREADY** are 1, **TDATA** is transmitted

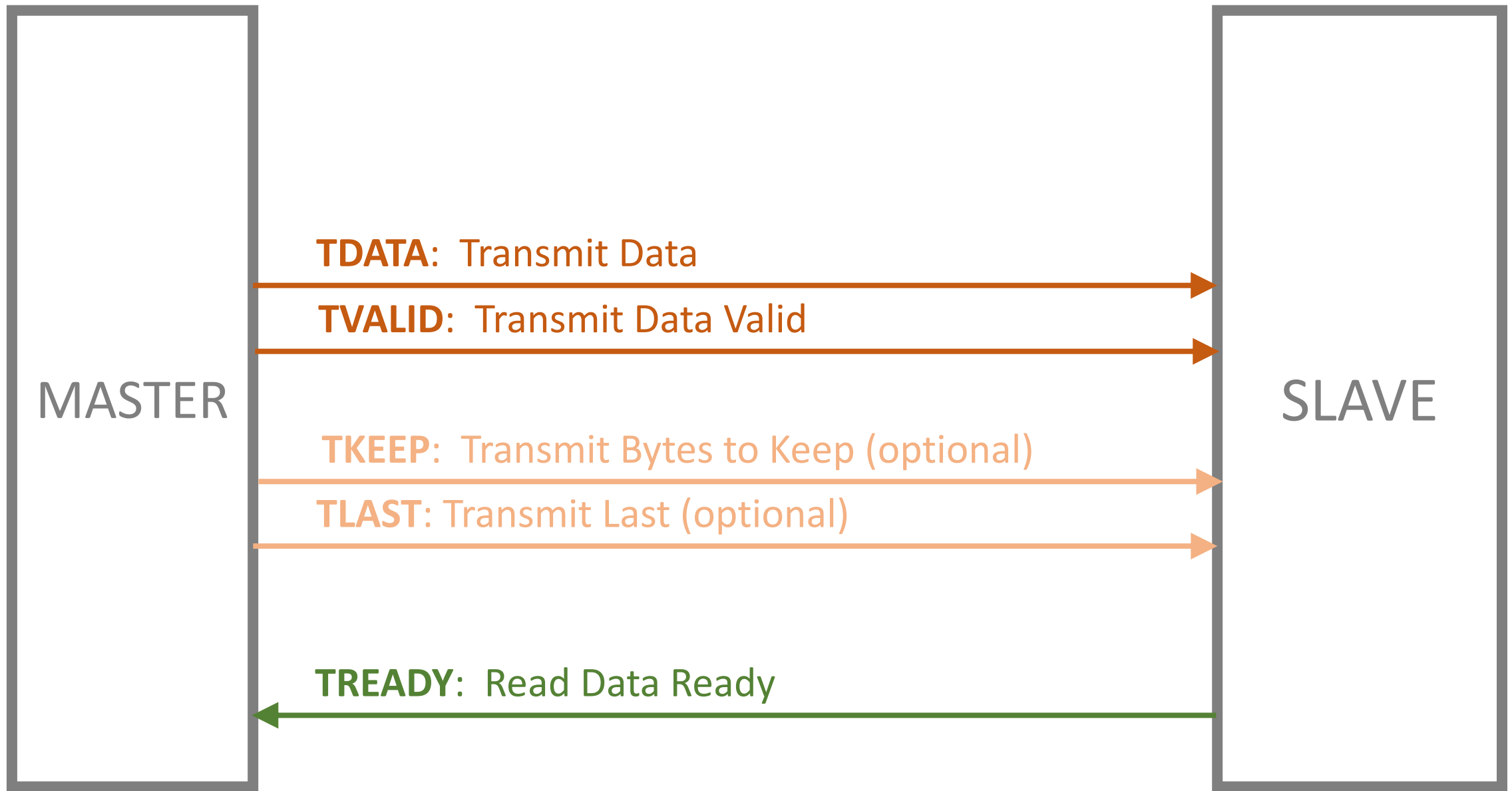
at the positive edge of **ACLK**

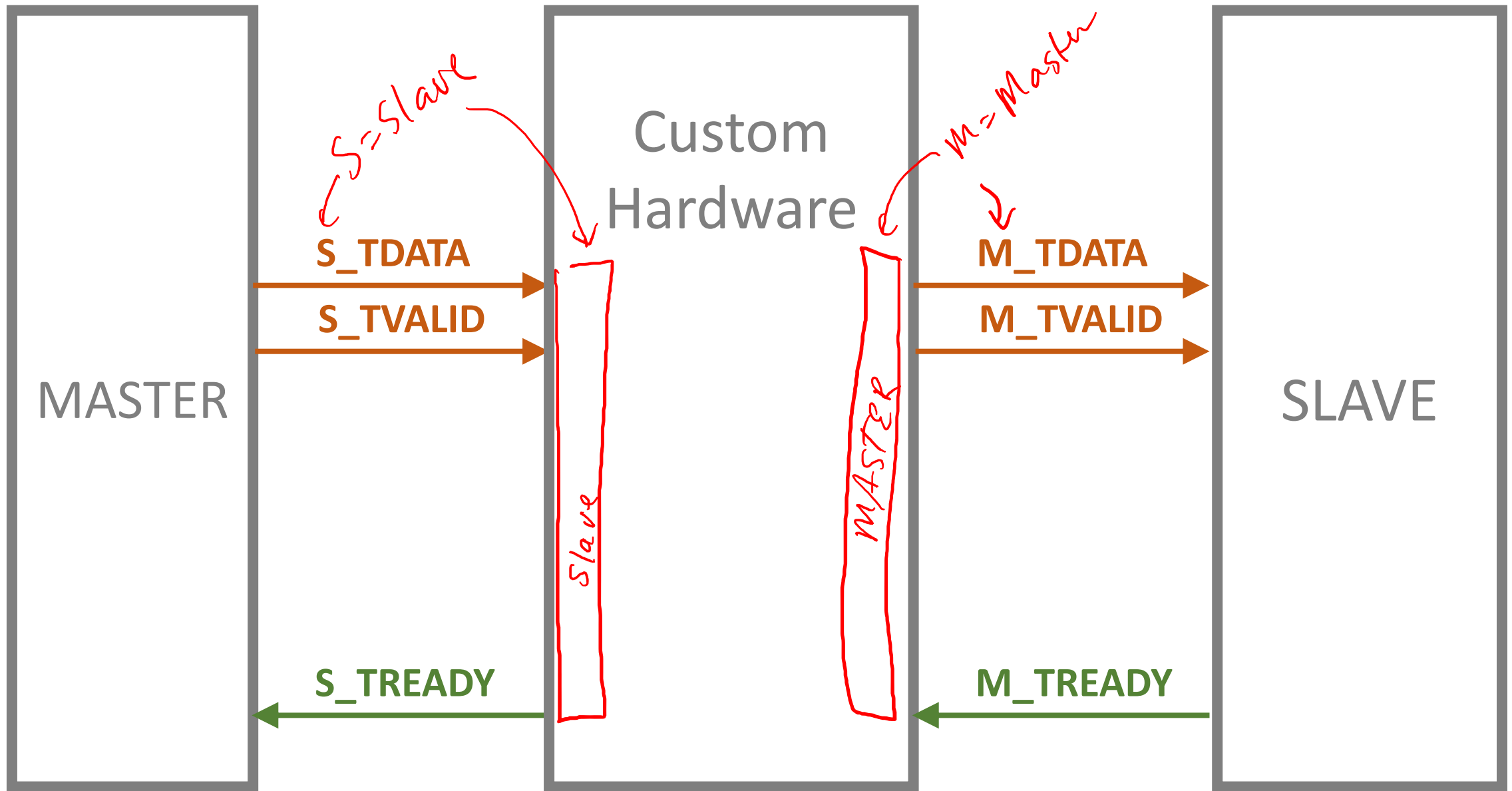
TLAST

- Special signal to indicate a group or “burst” of transmissions is complete.
- “Indicates the boundary of a packet”

Transferring data on a AXI4-Stream Bus.

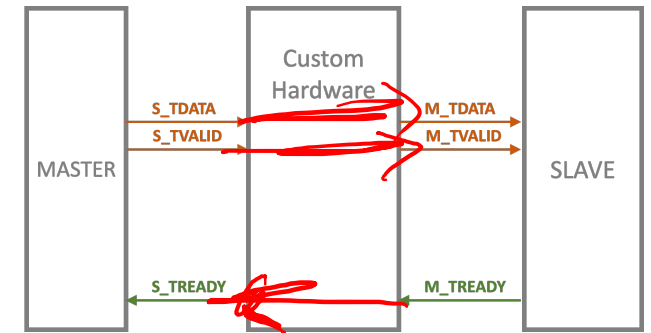






Let's build a custom block that does nothing!

```
module custom_hw (  
    input          ACLK,  
    input          ARESET,  
    input [31:0]   S_TDATA,  
    input          S_TVALID,  
    output         S_TREADY,  
    output [31:0] M_TDATA,  
    output         M_TVALID,  
    input          M_TREADY  
);
```

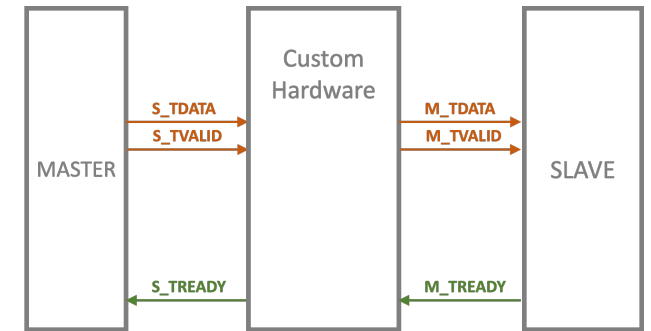


assign M_TDATA = S_TDATA;
assign M_TVALID = S_TVALID;
assign S_TREADY = M_TREADY;

```
endmodule
```

Let's build a custom block that does nothing!

```
module custom_hw (  
    input        ACLK,  
    input        ARESET,  
    input [31:0] S_TDATA,  
    input        S_TVALID,  
    output       S_TREADY,  
    output [31:0] M_TDATA,  
    output       M_TVALID,  
    input        M_TREADY  
);  
  
    assign M_TDATA = S_TDATA;  
    assign M_TVALID = S_TVALID;  
    assign S_TREADY = M_TREADY;  
  
endmodule
```

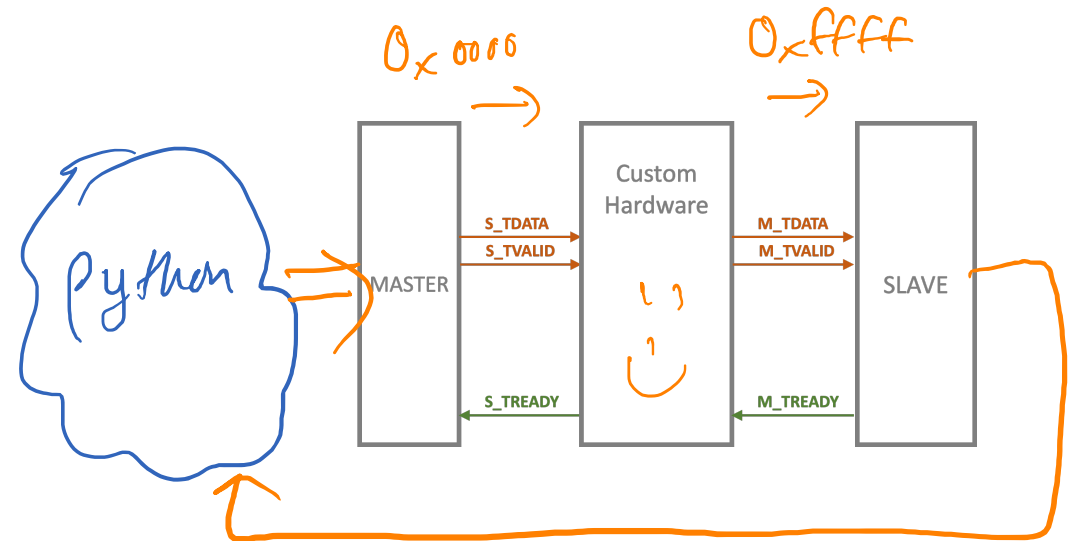


How would I flip all the bits of TDATA?

```
module custom_hw (  
    input          ACLK,  
    input          ARESET,  
    input [31:0]   S_TDATA,  
    input          S_TVALID,  
    output         S_TREADY,  
    output [31:0]  M_TDATA,  
    output         M_TVALID,  
    input          M_TREADY  
);
```

```
    assign M_TDATA = S_TDATA;  
    assign M_TVALID = S_TVALID;  
    assign S_TREADY = M_TREADY;
```

```
endmodule
```

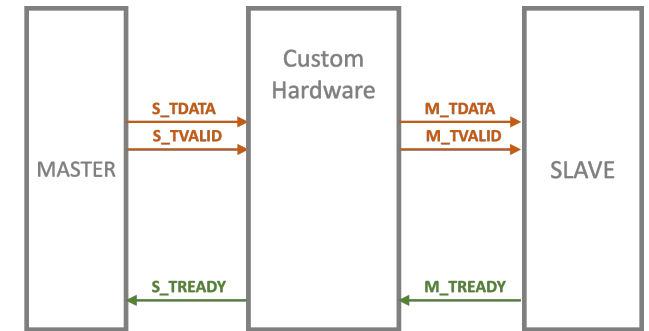


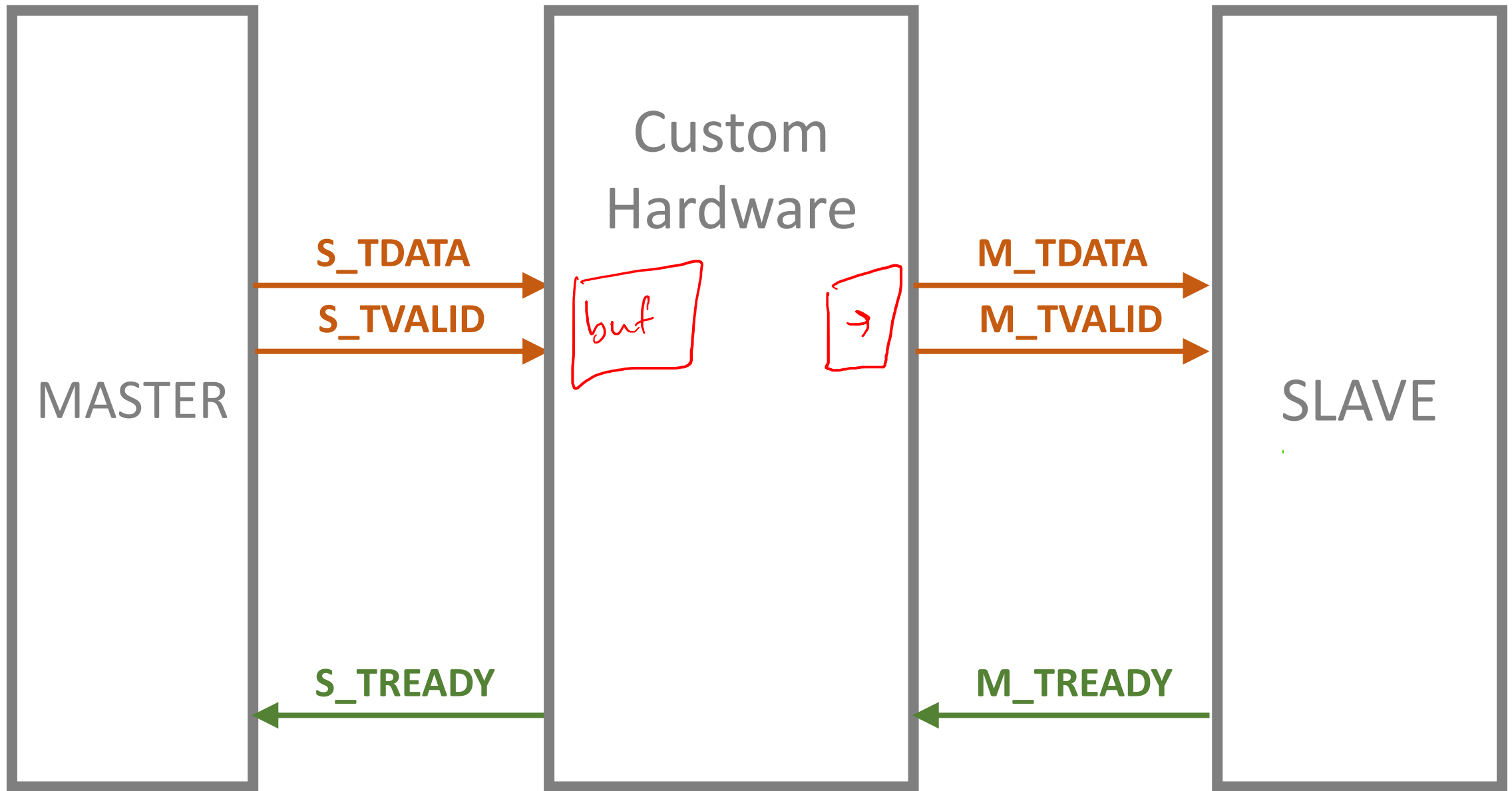
How would I flip all the bits of TDATA?

```
module custom_hw (  
    input        ACLK,  
    input        ARESET,  
    input [31:0] S_TDATA,  
    input        S_TVALID,  
    output       S_TREADY,  
    output [31:0] M_TDATA,  
    output       M_TVALID,  
    input        M_TREADY  
);
```

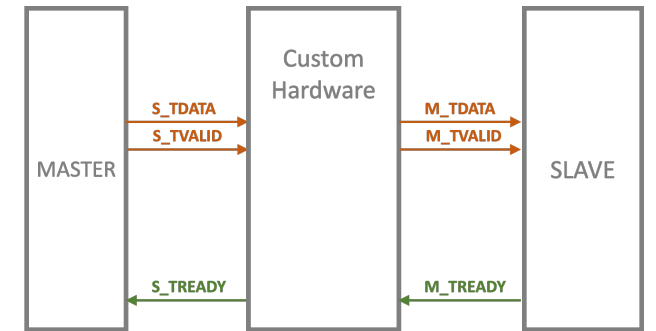
```
assign M_TDATA = ~S_TDATA;  
assign M_TVALID = S_TVALID;  
assign S_TREADY = M_TREADY;
```

```
endmodule
```

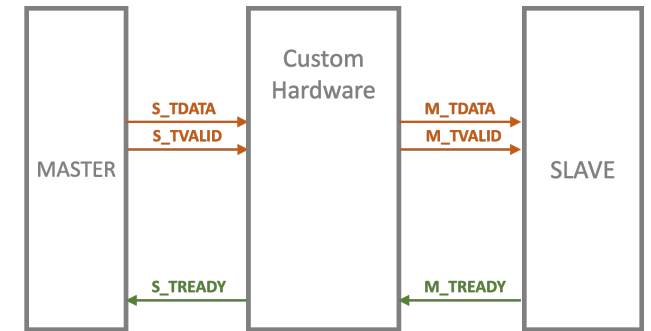
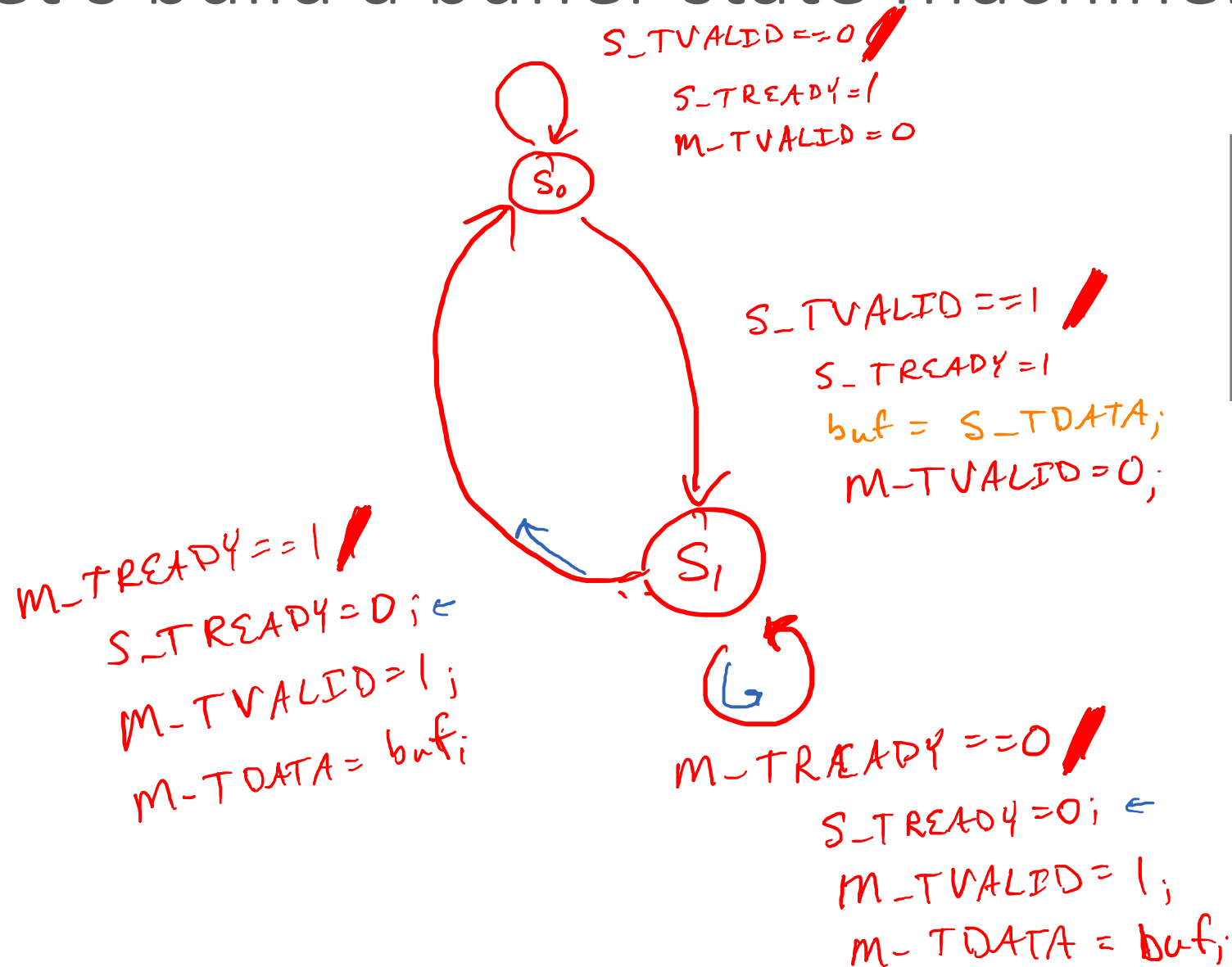




Let's build a buffer state machine.



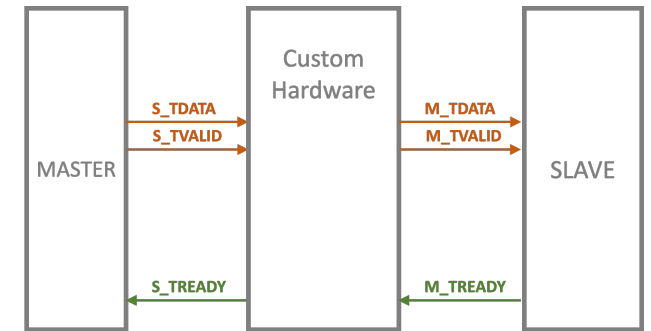
Let's build a buffer state machine.



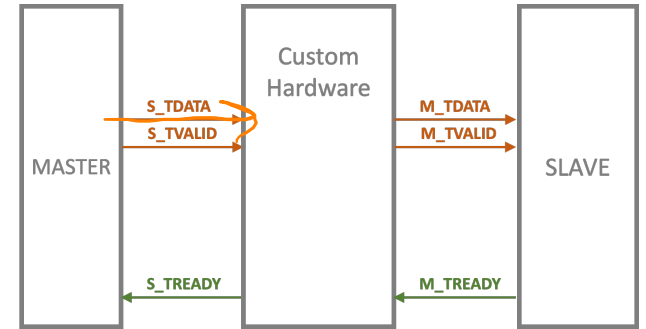
Let's build a buffer state machine.

```
module custom_hw_buf (  
    input          ACLK,  
    input          ARESET,  
    input [31:0]   S_TDATA,  
    input          S_TVALID,  
    output         S_TREADY,  
    output [31:0]  M_TDATA,  
    output         M_TVALID,  
    input          M_TREADY  
);
```

```
endmodule
```



Let's build a buffer state machine.



```

module custom_hw_buf (
    input          ACLK,
    input          ARESET,
    input [31:0]   S_TDATA,
    input          S_TVALID,
    output         S_TREADY,
    output [31:0]  M_TDATA,
    output         M_TVALID,
    input          M_TREADY
);

enum {S0, S1} state, nextState;
reg [31:0] nextVal;

always_ff @(posedge ACLK) begin
    if (ARESET)begin
        state <= S0;
        M_TDATA <= 32'h0
    end else begin
        state <= nextState;
        M_TDATA <= nextVal;
    end
end

end

```

```

always_comb begin
    S_TREADY = 'h1;
    M_TVALID = 'h0;
    nextState = state;
    nextVal = M_TDATA;
    case(state)
        S0: begin
            if (S_TVALID) begin
                nextState = S1;
                nextVal = S_TDATA;
            end
        end
        S1: begin
            S_TREADY = 'h0;
            M_TVALID = 'h1;
            if (M_TREADY) begin
                nextState = S0;
            end
        end
    endcase
end

endmodule

```

Vivado Demo

Next Time

- Memory-Mapped I/O
- Memory-Mapped Buses

References

- Zynq Book, Chapter 19 “AXI Interfacing”
- [Practical Introduction to Hardware/Software Codesign](#)
 - Chapter 10
- AMBA AXI Protocol v1.0
 - http://mazsola.iit.uni-miskolc.hu/~drdani/docs_arm/AMBAaxi.pdf
- <https://lauri.võsandi.com/hdl/zynq/axi-stream.html>

05: Buses II

Engr 315: Hardware / Software Codesign
Andrew Lukefahr
Indiana University

