ENGR 210 / CSCI B441
“Digital Design”

SPI III

Andrew Lukefahr

Announcements

- P9 – SPI
  - This one is new. Might be some changes.
  - Last one

- Final: **Thursday 5/6 @ 12.40-2:40pm**

P9 SPI QuickStart

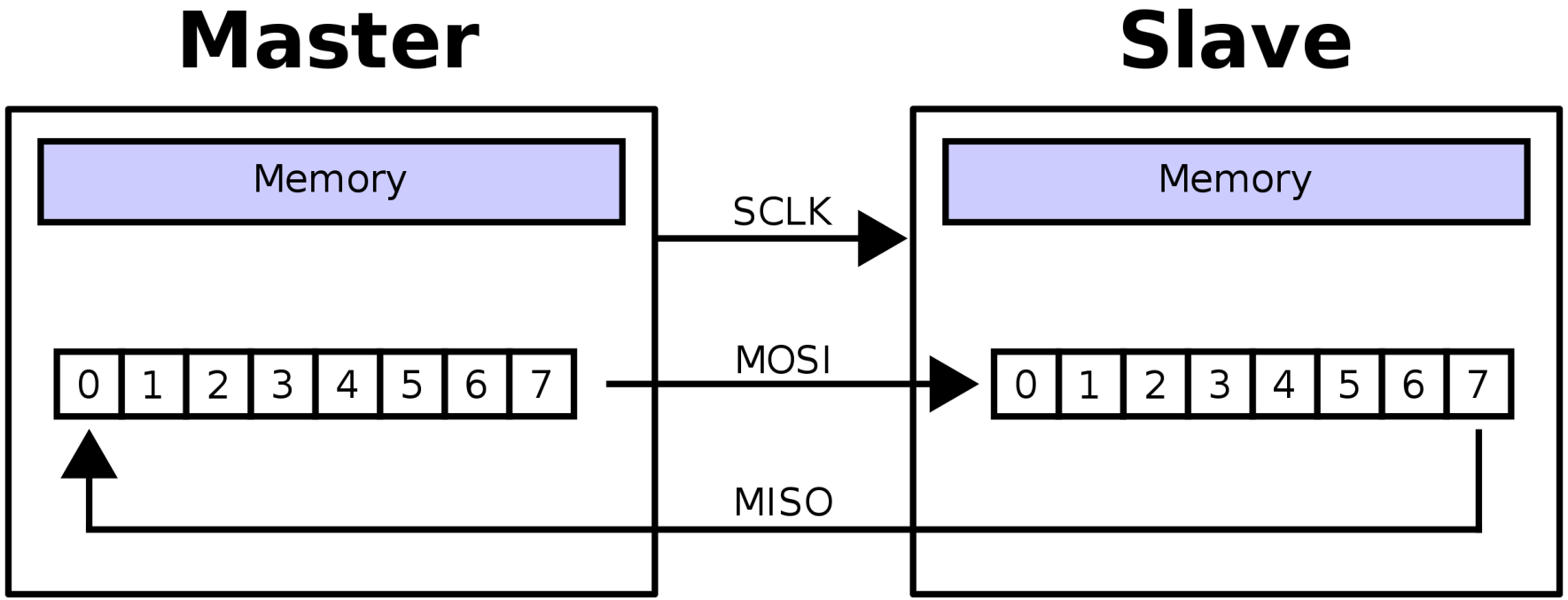
- We build the Vivado project for you:

```
git clone https://github.com/ENGR210/P9_SPI.git
cd P9_SPI
make setup
vivado vivado/vivado.xpr
```

- We provide you with Testbenches

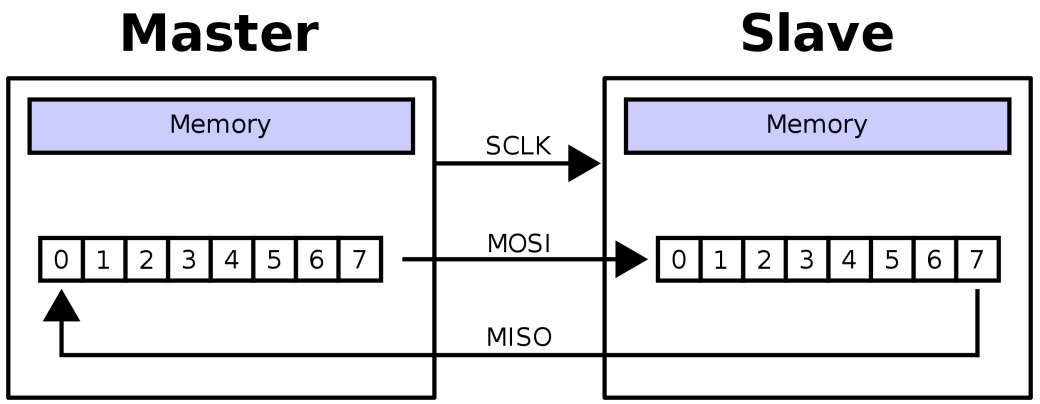
- Same ones as the Autograder!

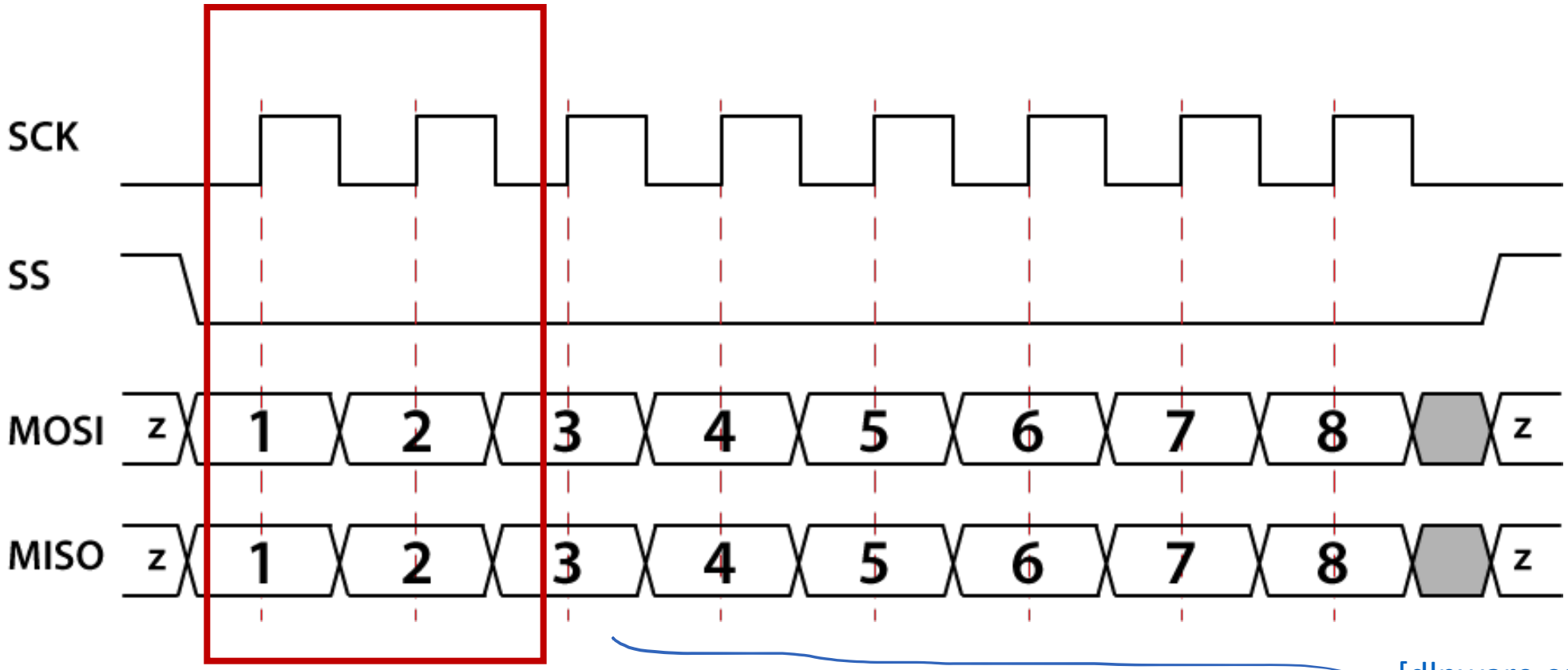
SPI


Master
Slave
Memory
Memory
SCLK
MOSI
MISO
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7

[wiki]

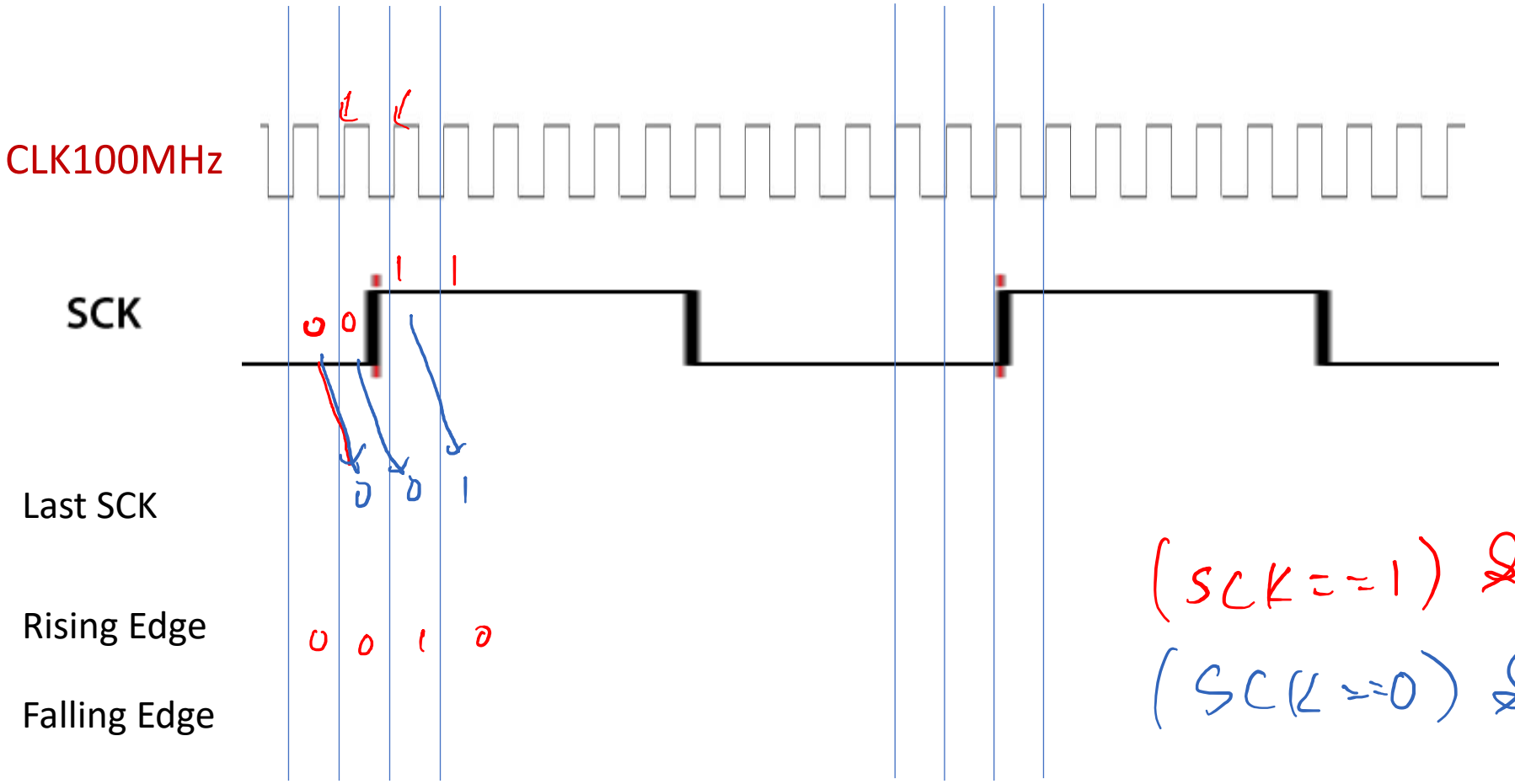
Clocks on Slave SPI





[dlnware.com]

Finding SCLK Edges



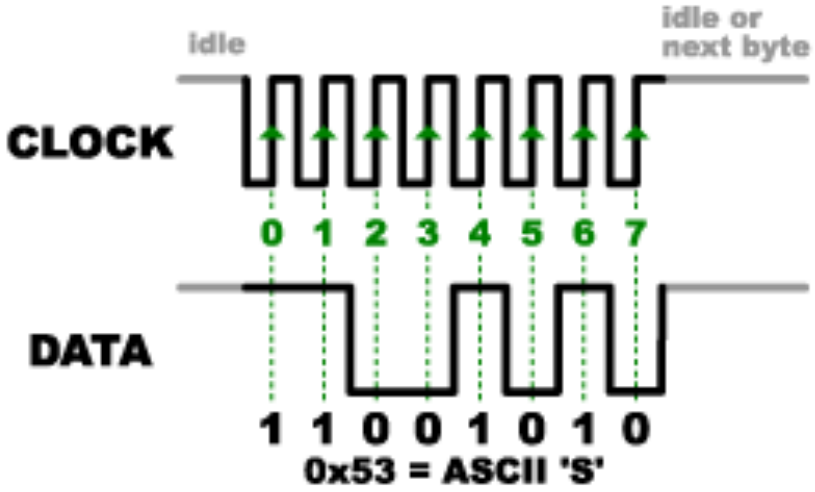
(SCK == 1) && (lastSCK == 0)

(SCK == 0) && (lastSCK == 1)

How to capture MOSI?

Not 100% right yet!

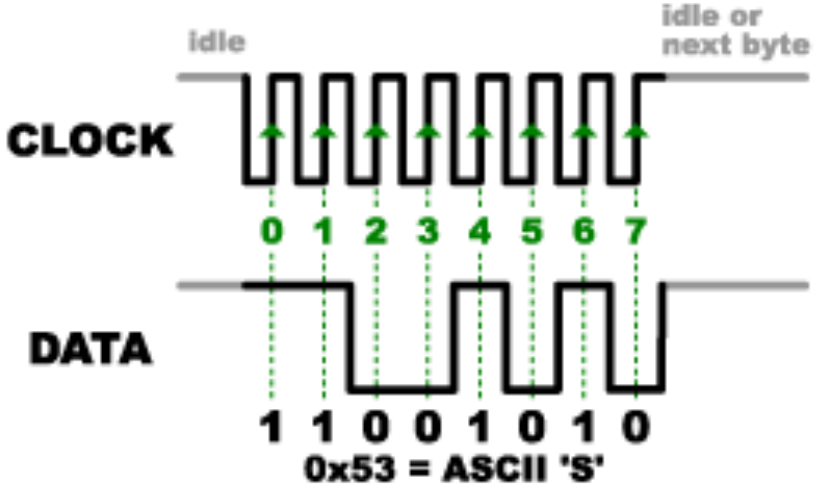
```
always_ff @(posedge clk) begin
        if (rst) q <= ‘h0;
        else q <= next_q;
end

always_comb begin
        next_q = q; //default
        if (rising_edge)
                next_q = MOSI;
end
```



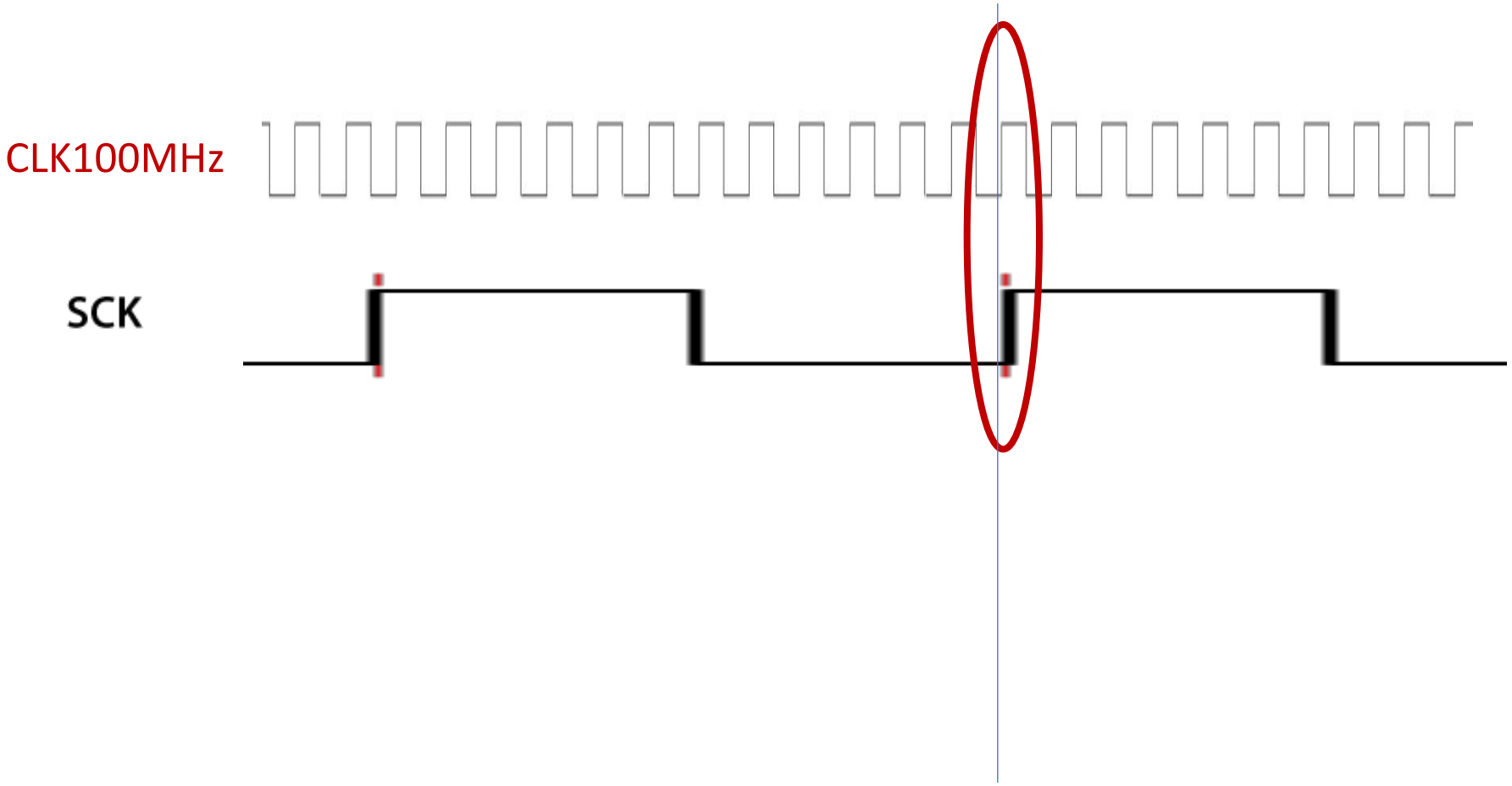
How to send MISO?

Not 100% right yet!

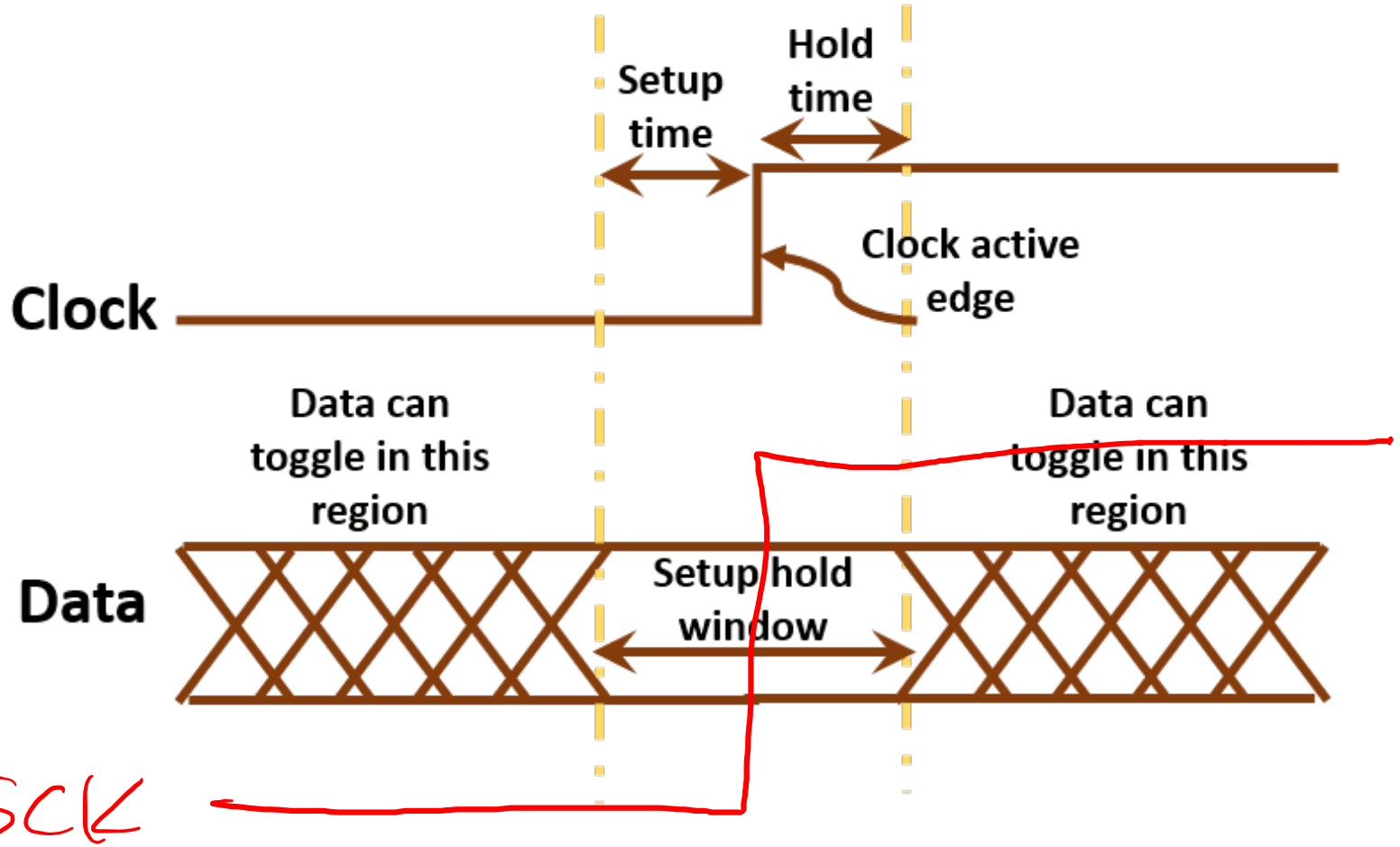
```
always_ff @(posedge clk) begin
        if (rst)  MISO <= ‘h0;
        else MISO <= next_MISO;
end

always_comb begin
        next_MISO = MISO; //default
        if (falling_edge)
                next_MISO = new_data_out;
end
```



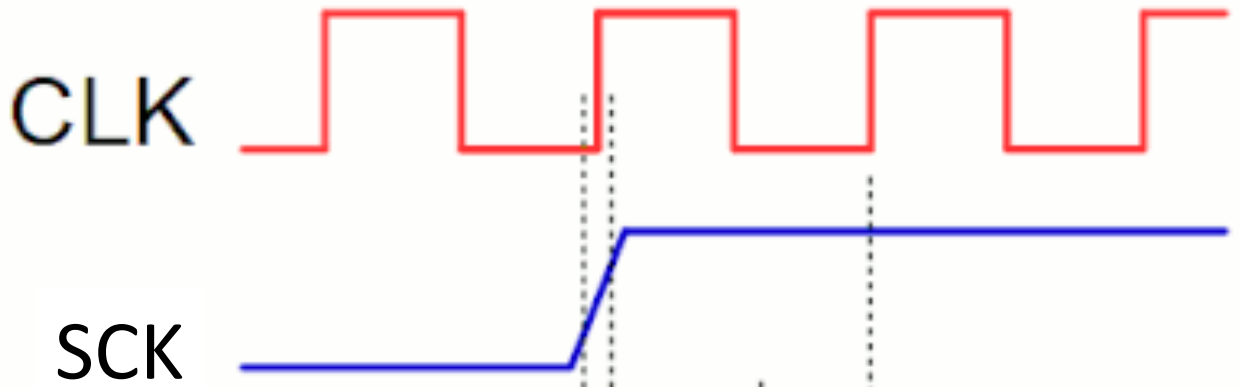
Timing Issues



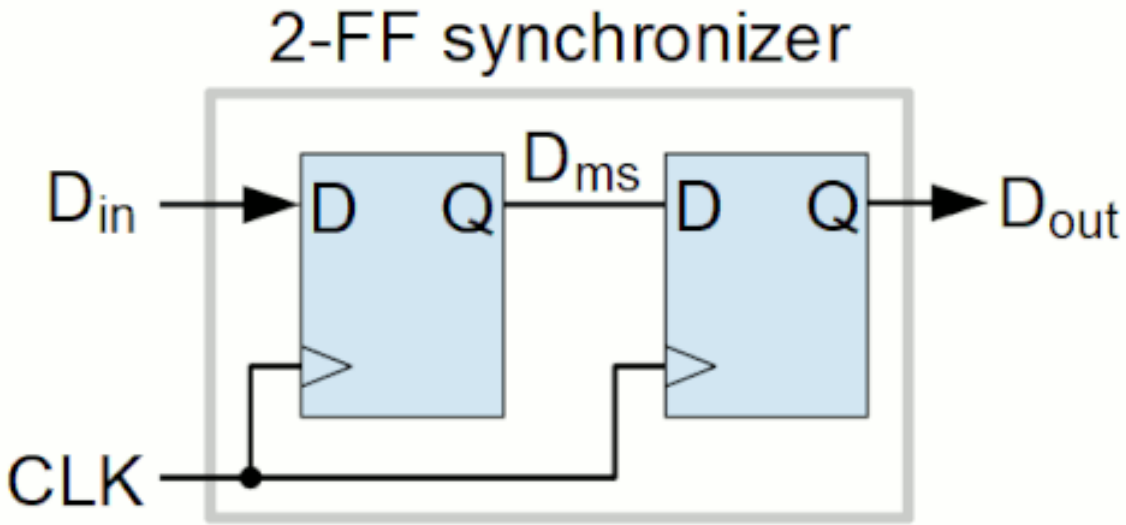
Setup/Hold Time

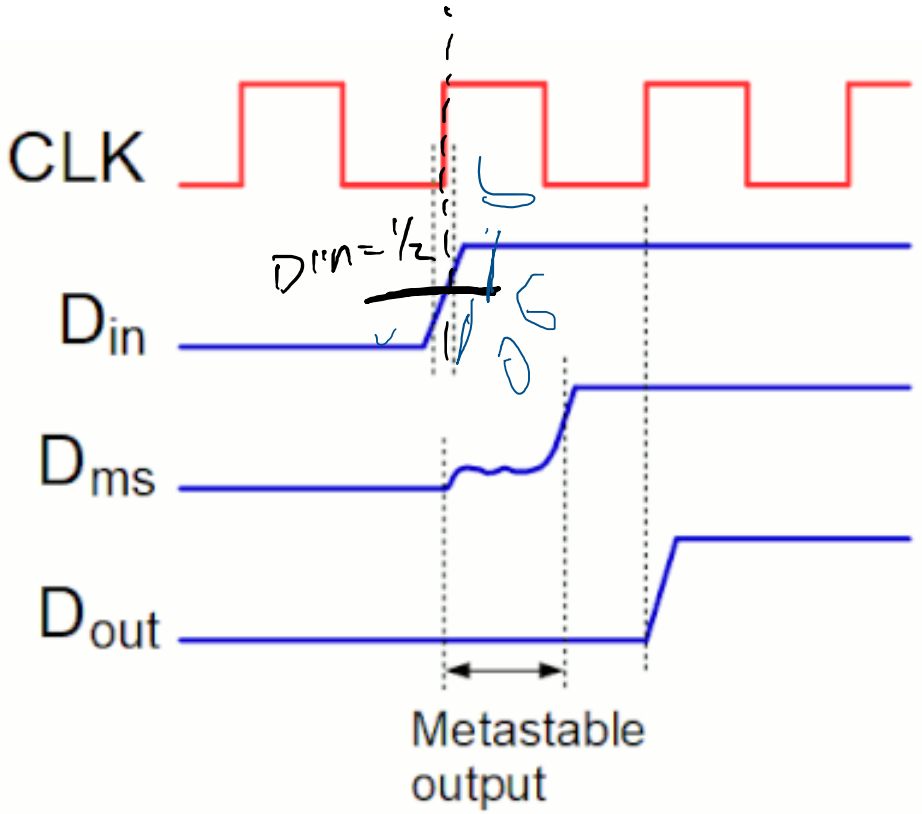


Why Synchronizers?



Synchronizers





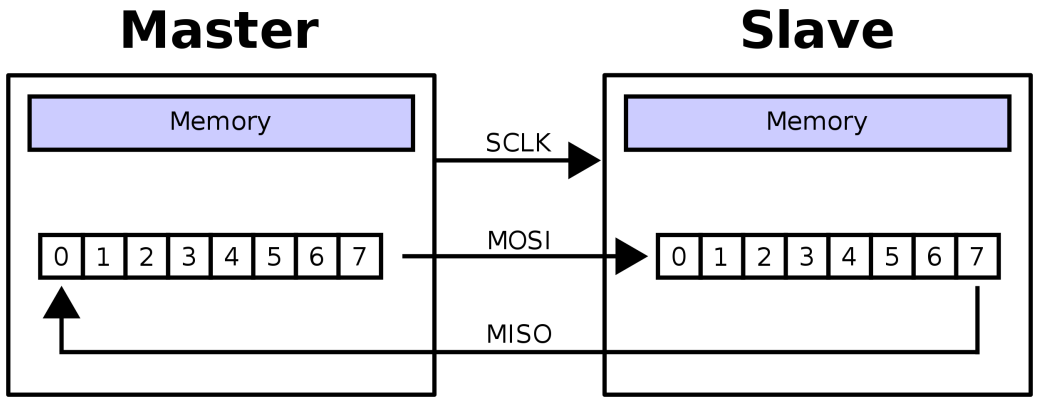
Synchronizers in P9

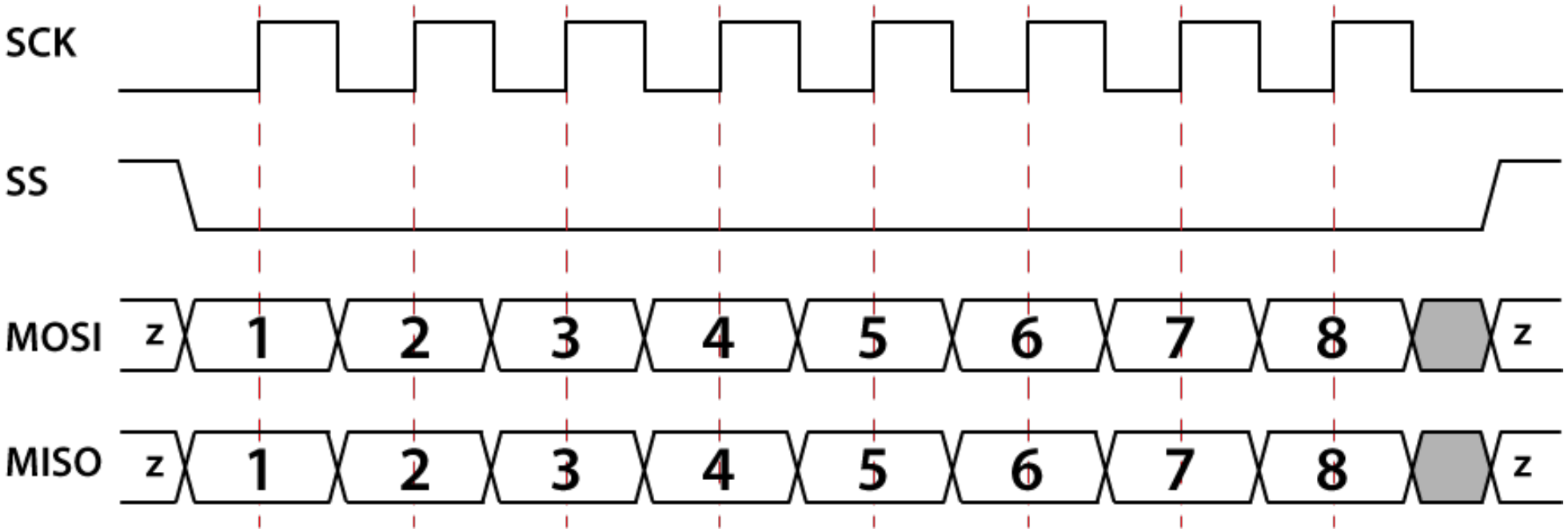
- This has a synchronizer inside it.

```
debounce db_sclk(
    .clk,
    .rst,
    .bouncy(sck),
    .stable(sck_)
);
```

- What other signals need synchronizers?

What about ’Z’?





[dlnware.com]

What is ‘Z’?

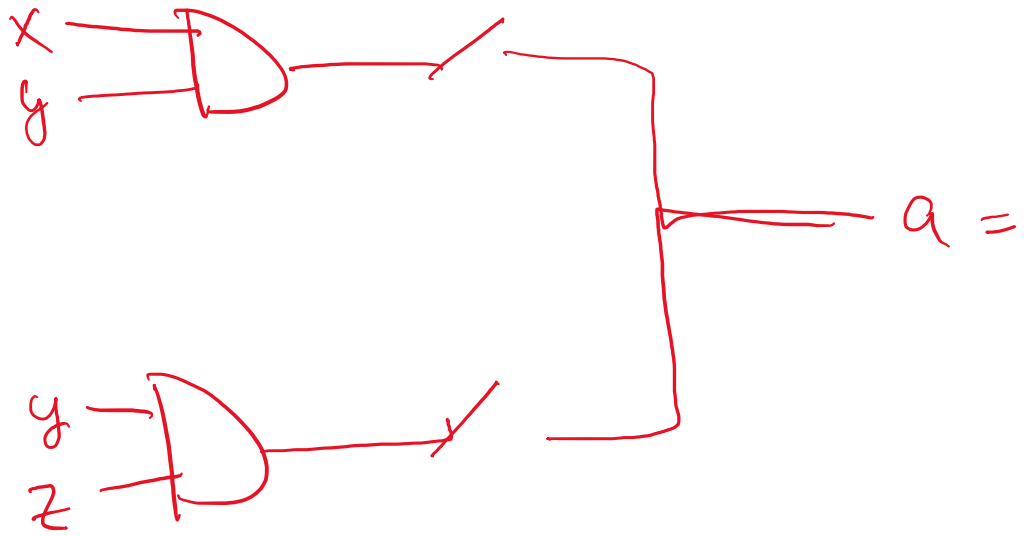
- Z: High Impedance
  - **DONT** drive a logical value
  - Pretend I’m disconnected

- “Tri-State” signals:
  - 1: this is logical true
  - 0: this is logical false
  - X: The simulation tools don’t know if it’s 1 or 0
  - Z: this is “high impedance”

Tri-State Logic

Problems with Tri-State Logic

- What if two signals “drive” at once?



Solution: Don’t Do That!

NEVER DO THAT!

SPI Memory Mapped Interface

P9 Goal

- Make all the Switches and LEDs accessible to the PI
- Use SPI

How do we read/set values?

- Switches: How to “read” the values?

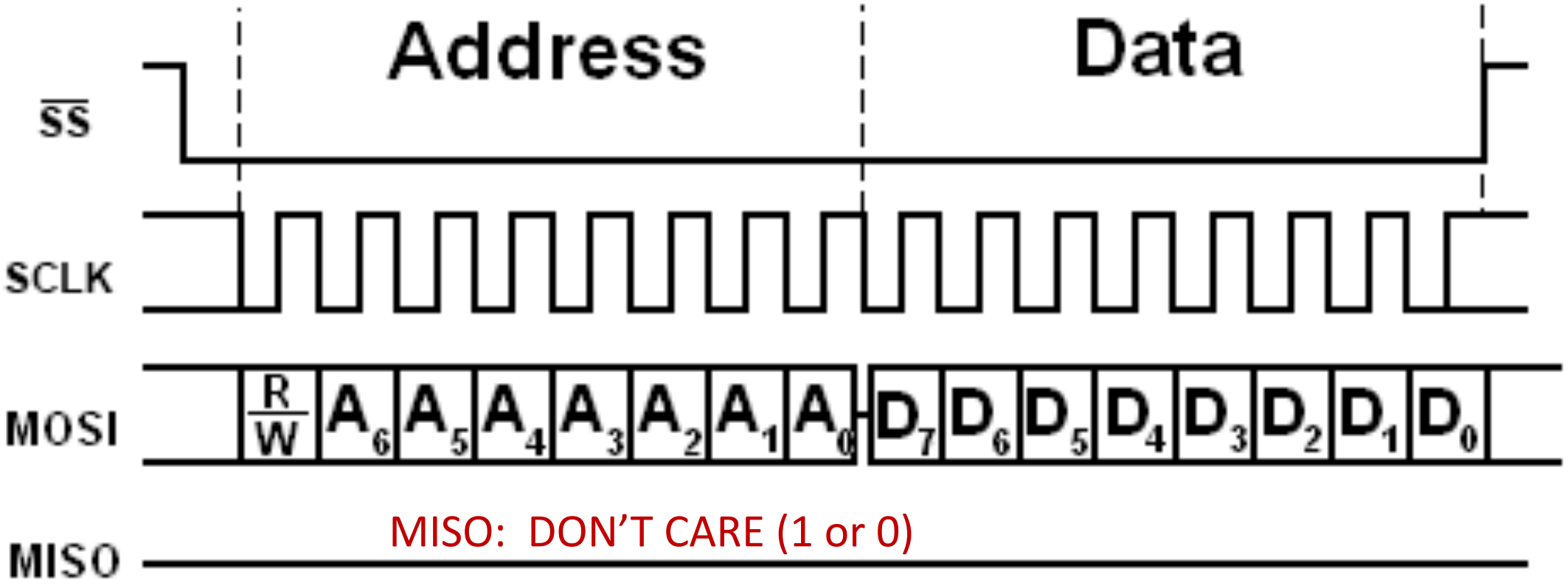
- LEDs: How to “write” the value?
  - Can we also “read” them?

Write Protocol

- “Please Set the LEDs to ON-OFF-ON-OFF-ON-OFF-ON-OFF”

Write Protocol

Operation: WRITE (**`h0**)
Address: LEDs (**`h3**)
Data: ON-OFF-ON-OFF (**`b10101010**)



[pyroelectro]

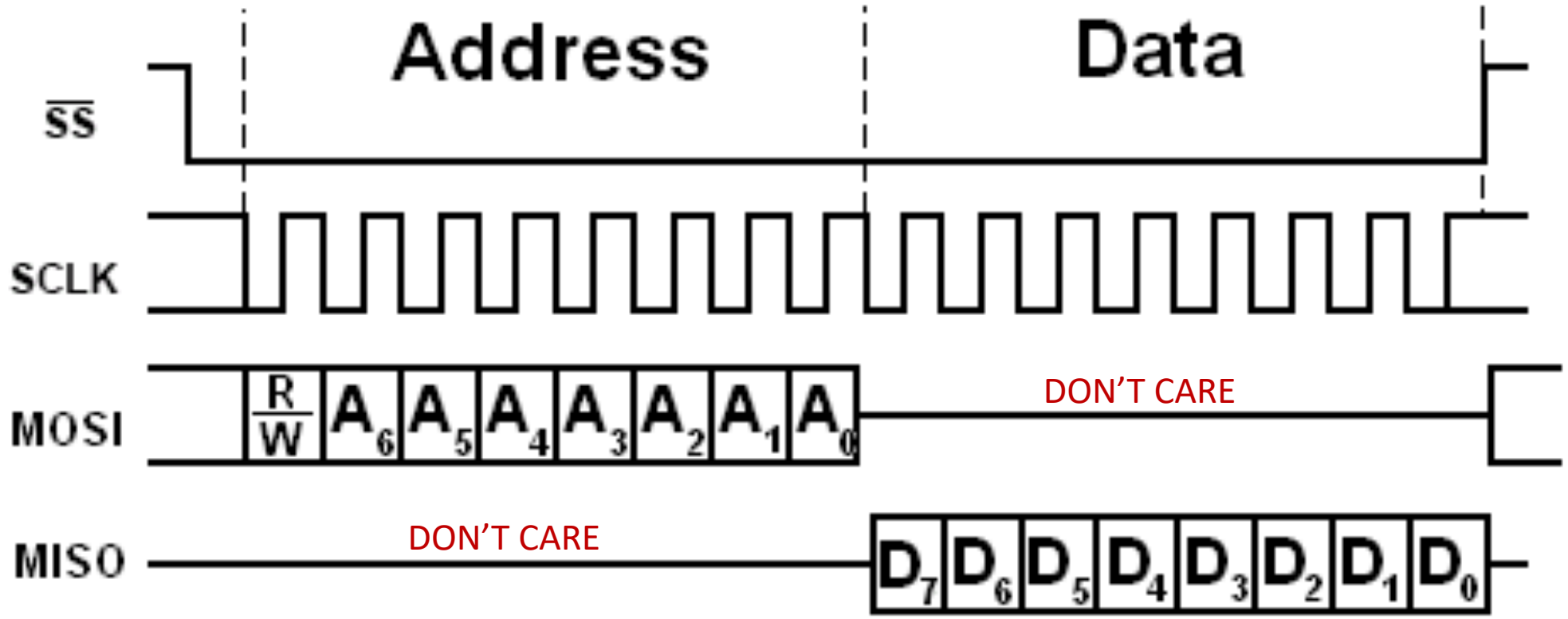
Read Protocol

- “Please TELL ME the value of the Switches”

- Operation:
- Address:
- Data:

Read Protocol

Operation: READ (`'h1`)
Address: SWITCHES (`'h1`)
Data: ??



[pyroelectro]

Protocol Address Map

Address	Mapping	Type
7'h00	chip_id (8'h7)	Read-Only
7'h01	switches[7:0]	Read-Only
7'h02	switches[15:8]	Read-Only
7'h03	leds[7:0]	Read/Write
7'h04	leds[15:8]	Read/Write

P9 Controller

- What implements the “Read/Write” Protocol.

- Inputs: Raw SPI, Switches
- Outputs: Raw SPI, LEDs

- Task: Change LEDs based on WRITEs
- Task: send Switch/LED values for READs

Next Time

- The “guts” of an FPGA