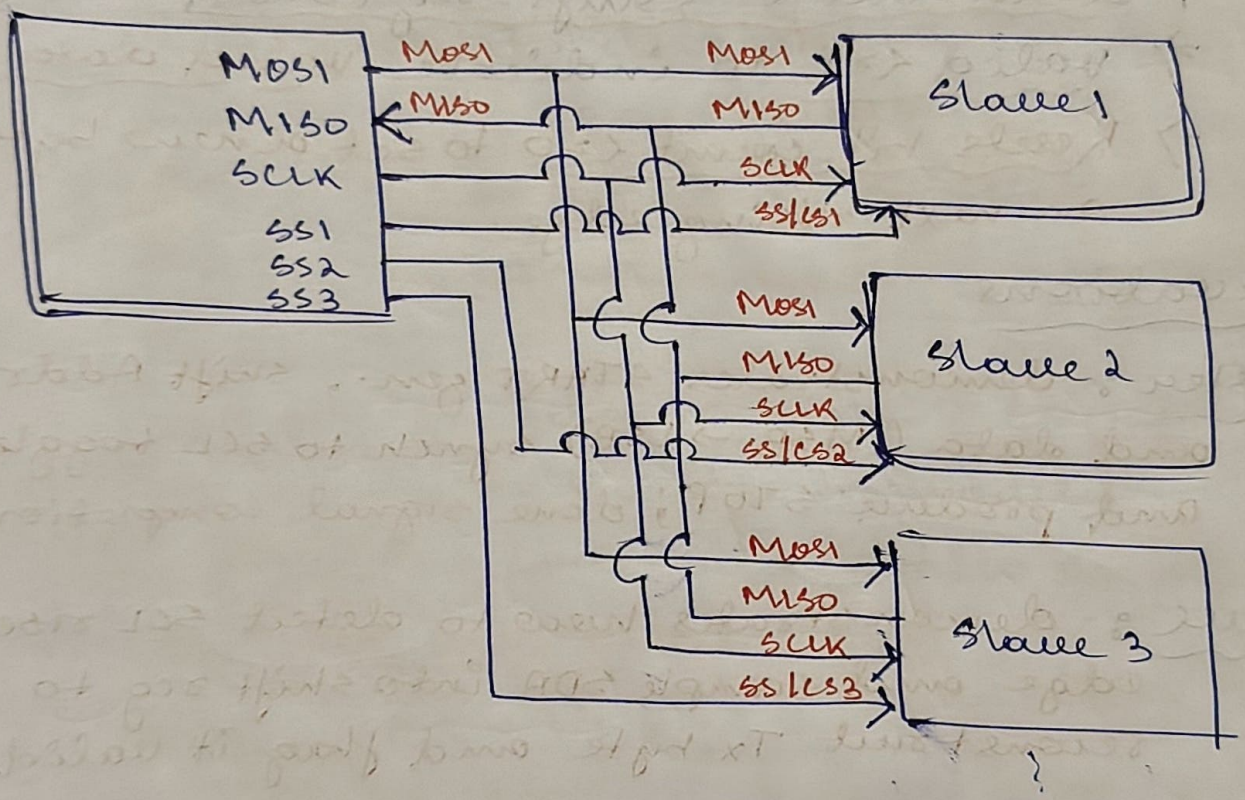
# SPI Protocol :- (Serial Peripheral interface)

- It is a synchronous, full duplex serial comms protocol used for short-distance comms bet devices
- It also uses Master-slave Architecture and 4 lines
  1. Master in - slave out (MISO)
  2. Master out - slave in (MOSI)
  3. serial clock - (SCLK)
  4. slave select (SS) / chip select (CS)




- High level purpose
  1. SPI master sends one 8-bit byte to slave
  2. Simultaneously Master receives one byte back (full-duplex)
  3. SS (slave select) gates the transfer

(4) SCLK ⇒ clks the serial exchange
(5) Mosi ⇒ Master → slave
(6) Miso ⇒ slave → Master
(7) Done flag ⇒ indicates when each side has completed its byte transfer.

- SPI Master (controller + serializer / receiver)

⇒ working (1) control ss (active low)
(2) Generate SCLK
(3) Serialises data_in MSB → LSB to MOSI
(4) samples MISO MSB → LSB into an int. shift reg to form data_out
(5) Asserts Done when 8-bit sent and received

⇒ Ports (for code) (1) clk, rst
(2) start - Begin new transfer.
(3) data_in [7:0] - byte to transmit MOSI
(4) data_out [7:0] - byte received from MISO
(5) done - asserted when transfer complete
(6) sclk - driven by Master
(7) mosi - master drives bit
(8) miso - master samples bit
(9) ss - slave select (active low)

⇒ Internal regs (1) bit_cnt (3bit) - counts transmitted bits (0 — 7)
(2) shift_reg - captures incoming bits from MISO until finalization.

⇒ Reset Behav:

On reset : ss=1 (deselected), done=0, sclk=0, mosi=0, data_out=0 bit_cnt=0, shift_reg=0

⇒ Transfer flow (main always block on posedge clk)

① ss=0 ⇒ slave selected

② sclk toggled each clk cycle (sclk <= ~sclk) for which Master generates sq. wave sclk

③ on falling edge of sclk (if (sclk == 1'b0))

* Master drives Mosi with current bit data_in[7-bit_cnt] - i.e MSB first

④ on Rising edge of sclk (else)

* Master samples Miso into shift_reg[7-bit_cnt] and increment bit_cnt

⑤ After 8th bit is sampled (bit_cnt == 7 && sclk == 1)

Master :- Moves shift_reg ⇒ data_out
- done = 1
- Releases ss = 1 (deselects slave)

⑥ when done = 1, Master holds sclk = low

• So the Master procedure goes like

Reset ⇒ ss=0 (slave select) ⇒ Master sclk toggles ⇓

Mosi ⇒ next bit (with falling edge) / Miso ⇒ sampled (with rising edge) ⇐ Received byte at data_out (after 8 cycles) ⇒ done flag = 1 ⇒ ss=1 (deselect) ⇒ sclk=0

# • SPI Slave (Serializer / Receiver)

⇒ working :- ① Monitor ss (active low) to enable or disable the transfer

② Samples incoming Mosi on sclk edges and shifts bits into data_out

③ Drives Miso with its data_in bits (MSB - LSB) so Master samples them simultenously

④ Asserts its own done when 8 bits have been captured

⇒ ports :- ① sclk - driven by Master

② rst ③ ss - slave select (active low)

④ mosi - i/p from master

⑤ miso - o/p to master

⑥ data_in [7:0] - byte that slave should shift out on Miso

⑦ data_out [7:0] - byte the slave received from Mosi

⑧ done - captured 8-bit

⇒ internal reg :- ① bit_cnt - counts 0...7 bit shift

② shift_reg - intermediate storage to data_out

⇒ clocking & sampling :-

at always @(posedge sclk or posedge rst)

slave samples on rising edge of sclk.

⇒ On rst :- clears, counters, data_out, done, shift_reg, miso

⇒ Active transfer (ss = low)

(1) on each posedge sclk when ss low

- slave samples Mosi and stores in data_out [7 - bit_cnt] (MSB first)
- slave places the nxt o/p bit on miso from data_in [7 - bit_cnt] so the master can sample it on its rising edge
- increment bit_cnt and when bit_cnt == 7 sets done = 1 (transfer complete for slave)

- so slave works like

synch to sclk ⇒ rising edge ⇒ next edge
read Mosi updates Miso

⇓

After 8 clk, done = 1

- Master - slave working (testbench)

(1) clk toggles every 5 units. Rst asserted-deasserted

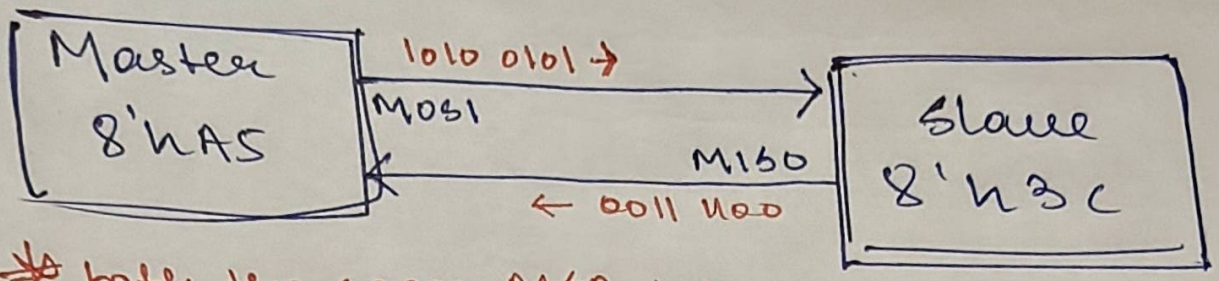
(2) master_data = 8'hA5 (master tx A5)
slave_data = 8'h3C (slave tx 3C)

(3) start = 1, Master toggle sclk, drive ss, o/p Mosi bit and sample Miso bit. Slave respond when ss = low and sclk toggle.

(4) o/p :- Master_out = 0x3C and slave_out = 0xA5

- So, both Master-slave transfers create a full-duplex communication protocol

→ Example we took




* both the cases MSB 1st

And after 8 clk cycle when 8 bit are transfered and sampled simultenously done = 1 along with Master_out = 0x3C
Slave_out = 0xA5