Project

- Project team
  - — Teams of three

- Final Project Report
  - — Deadline: TBD in the lecture
  - — Submit the following
    - – PDF file with snippets
    - – Design files
    - – One Testbench file
    - – Do file to run the testbench
  - — Allow 2 days for unforeseen issues

Project Suggestions

- Split the work in parallel between design and verification
- Tasks will take longer than what you think
- Have a 2-3-day task list

SPI Interface

- One of the most popular Interfaces nowadays

- Stands for Serial-Peripheral Interface

- Four Wires
  — MOSI: Master-Out-Slave-In
  — MISO: Master-In-Slave-Out
  — SCK: Clock
  — SS_n: Slave Select

- High Data Rates

Project: 1- SPI Slave Interface

- One of the most popular Interfaces nowadays

- Stands for Serial-Peripheral Interface

- Four Wires
  — MOSI: Master-Out-Slave-In
  — MISO: Master-In-Slave-Out
  — SCK: Clock
  — SS_n: Slave Select

- High Data Rates
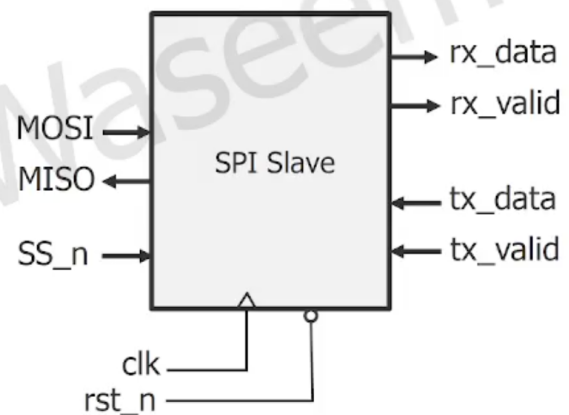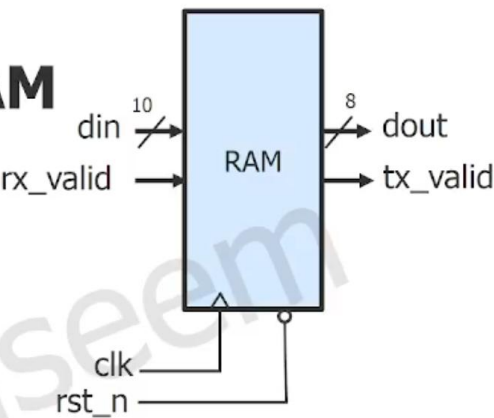
Project: 2- Single-port Sync RAM

- ## Parameters
  - — MEM_DEPTH, Default: 256
 - — ADDR_SIZE, Default: 8
- ## Ports

Name	Type	Size	Description
din	Input	10 bits	Data Input
clk		1 bit	Clock
rst_n		1 bit	Active low asynchronous reset
rx_valid		1 bit	If HIGH: accept din[7:0] to save the write/read address internally or write a memory word depending on the most significant 2 bits din[9:8]
dout	Output	8 bits	Data Output
tx_valid		1 bit	Whenever the command is memory read the tx_valid should be HIGH

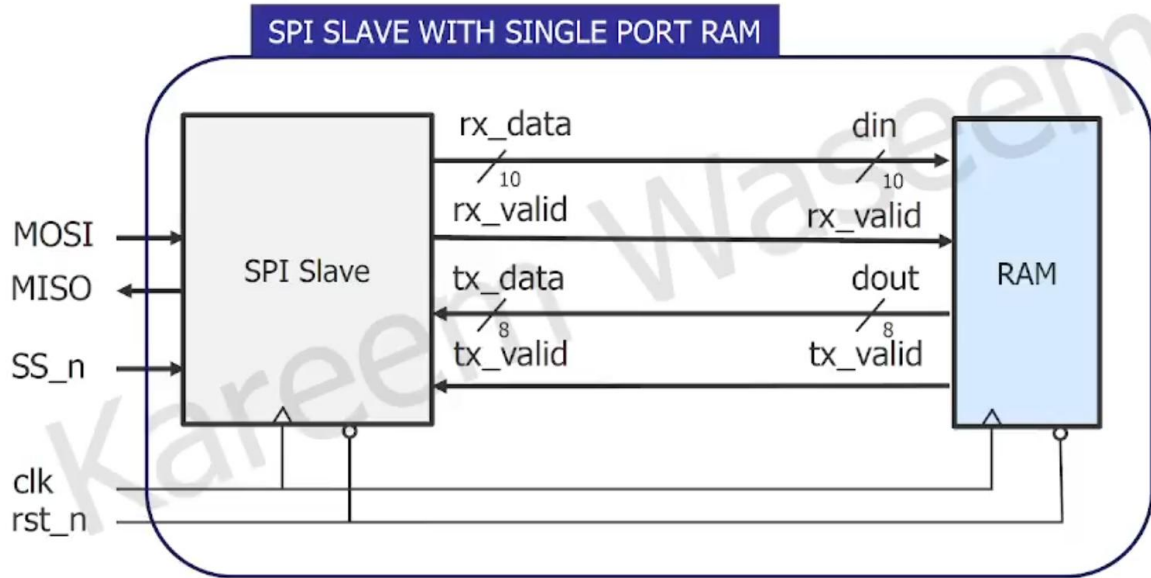# Project: 2- Single-port Sync RAM

- ## Parameters
  - — MEM_DEPTH, Default: 256
  - — ADDR_SIZE, Default: 8
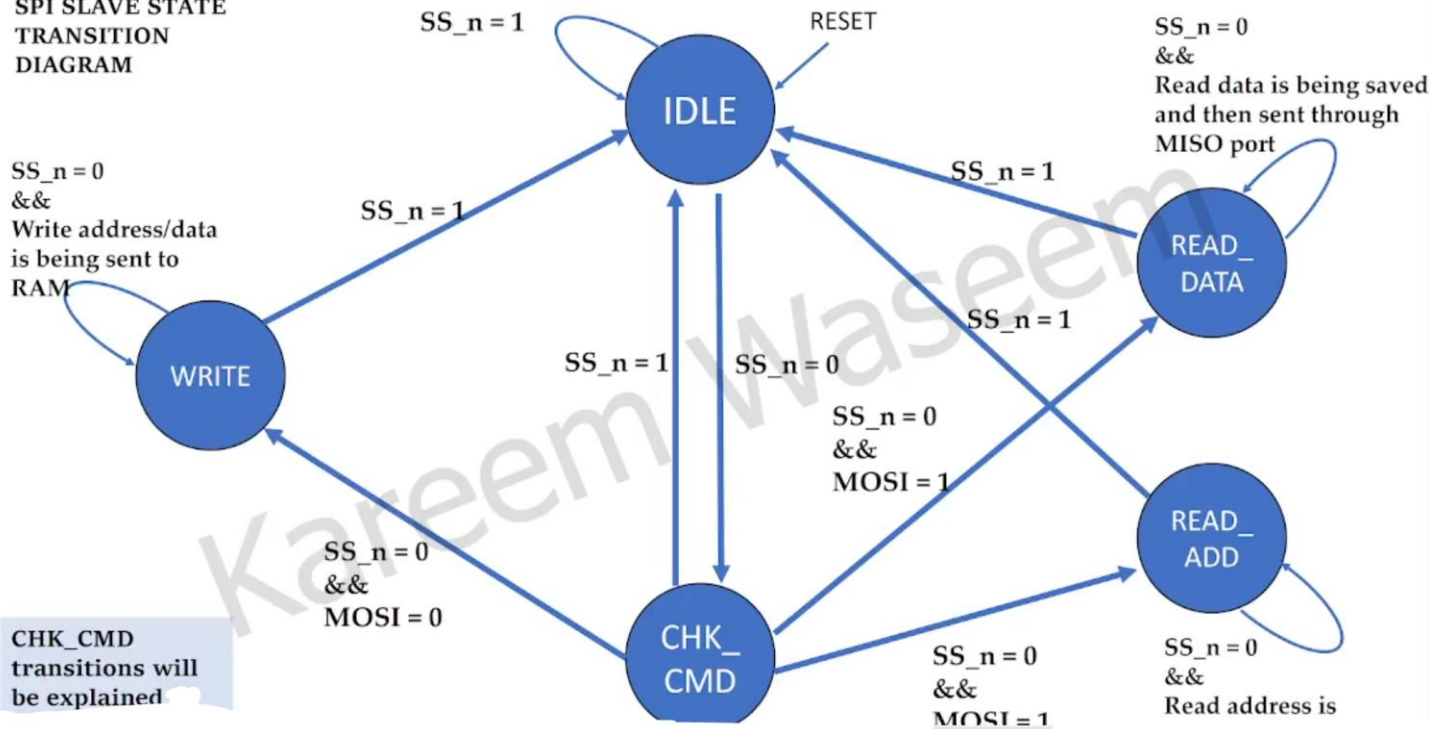- Most significant din bit "din[9]" determines if it is a write or read command
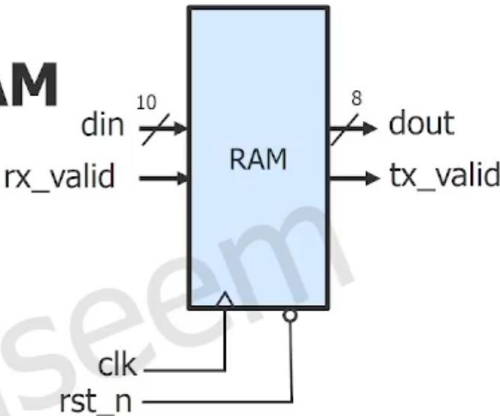
Port	Din[9:8]	Command	Description
din	00	Write	Hold din[7:0] internally as write address
	01		Write din[7:0] in the memory with write address held previously
	10	Read	Hold din[7:0] internally as read address
	11		Read the memory with read address held previously, tx_valid should be HIGH, dout holds the word read from the memory, ignore din[7:0]

Project: 3- SPI Wrapper

SPI SLAVE WITH SINGLE PORT RAM

SPI SLAVE STATE TRANSITION DIAGRAM

SS_n = 1

RESET

SS_n = 0
&&
Read data is being saved
and then sent through
MISO port

SS_n = 0
&&
Write address/data
is being sent to
RAM

IDLE

SS_n = 1

SS_n = 1

READ_
DATA

WRITE

SS_n = 1   SS_n = 0

SS_n = 0
&&
MOSI = 1

SS_n = 0
&&
MOSI = 0

CHK_
CMD

READ_
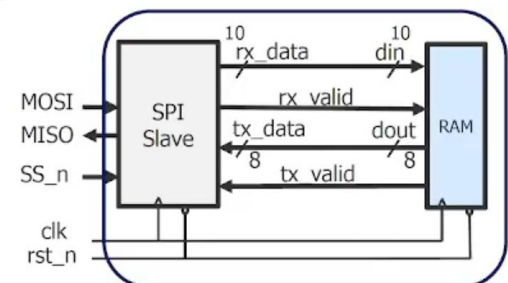ADD

**CHK_CMD
transitions will
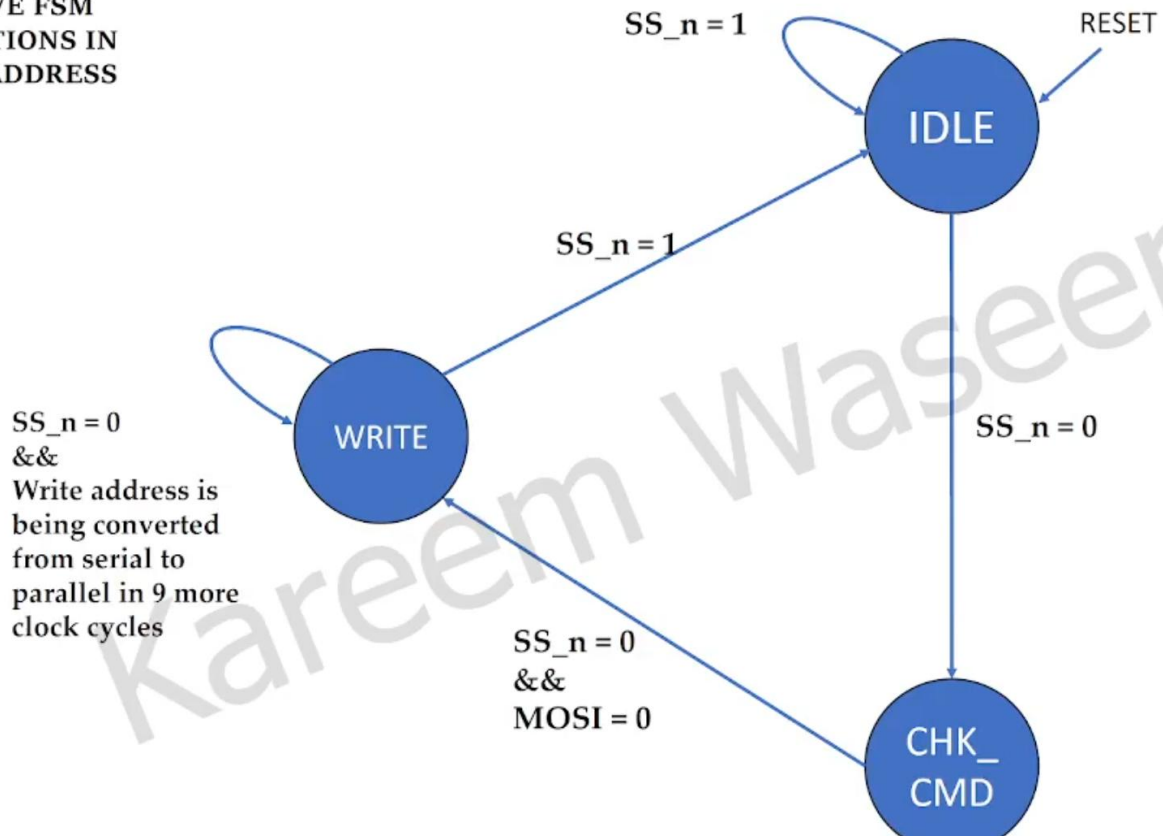be explained**

SS_n = 0
&&
MOSI = 1

SS_n = 0
&&
Read address is
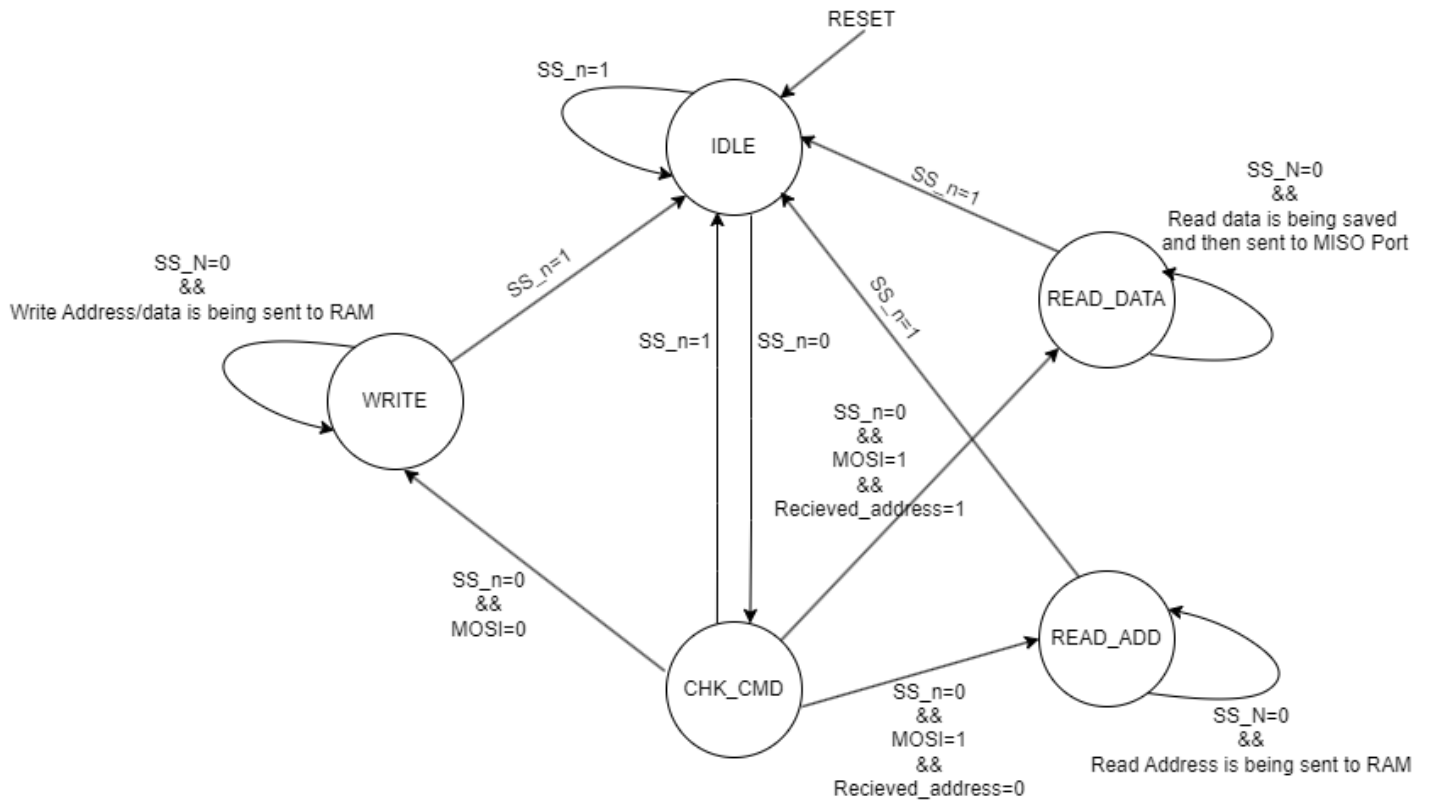
RAM Write Command – Write Address

1. Master will start the write command by sending the write address value, rx_data[9:8] = din[9:8] = 2'b00

2. SS_n = 0 to tell the SPI Slave that the master will begin communication

3. SPI Slave check the first received bit on MOSI port '0' which is a control bit to let the slave determine which operation will take place "write in this case". SPI Slave then expects to receive 10 more bits, the first 2 bits are "00" on two clock cycles and then the wr_address will be sent on 8 more clock cycles

4. Now the data is converted from serial "MOSI" to parallel after writing the rx_data[9:0] bus

5. rx_valid will be HIGH to inform the RAM that it should expect data on din bus

6. din takes the value of rx_data

7. RAM checks on din[9:8] and find that they hold "00"

8. RAM stores din[7:0] in the internal write address bus
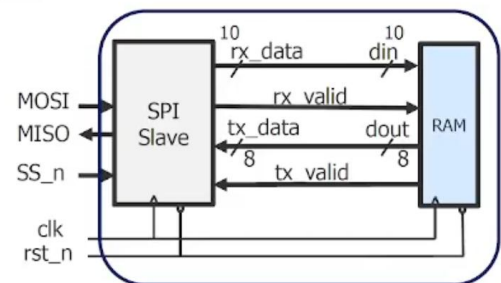
9. SS_n = 1 to end communication from Master side

SPI SLAVE FSM
TRANSITIONS IN
WRITE ADDRESS

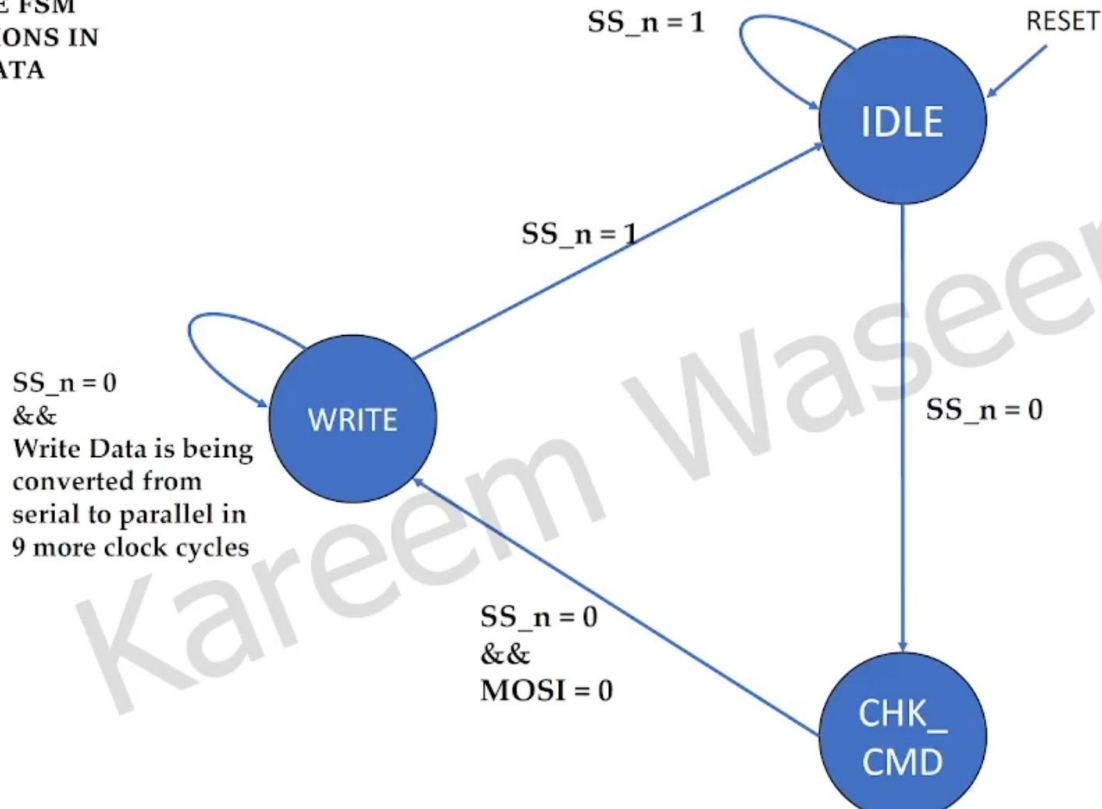**IDLE** — SS_n = 1 (self loop), RESET

SS_n = 1 (IDLE → WRITE)

WRITE — SS_n = 0 && Write address is being converted from serial to parallel in 9 more clock cycles (self loop)

SS_n = 0 (IDLE → CHK_CMD)

SS_n = 0 && MOSI = 0 (CHK_CMD → WRITE)

CHK_CMD

RAM Write Command – Write Data

1. Master will continue the write command by sending the write data value, rx_data[9:8] = din[9:8] = **2'b01**

2. SS_n = 0 to tell the SPI Slave that the master will begin communication

3. SPI Slave check the first received bit on MOSI port '0' which is a control bit to let the slave determine which operation will take place "write in this case". SPI Slave then expects to receive 10 more bits, the first 2 bits are "01" on two clock cycles and then the wr_**data** will be sent on 8 more clock cycles

4. Now the data is converted from serial "MOSI" to parallel after writing the rx_data[9:0] bus

5. rx_valid will be HIGH to inform the RAM that it should expect data on din bus

6. din takes the value of rx_data

7. RAM checks on din[9:8] and find that they hold **"01"**

8. RAM stores din[7:0] in the RAM with wr_address previously held
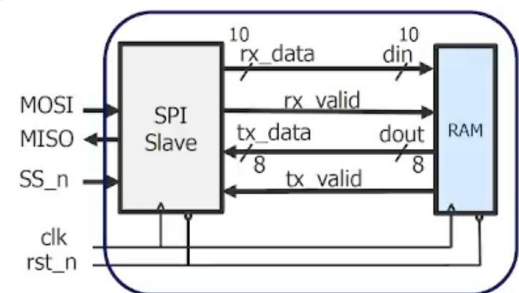
9. SS_n = 1 to end communication from Master side

SPI SLAVE FSM TRANSITIONS IN WRITE DATA

IDLE — RESET

SS_n = 1 (IDLE self loop)

SS_n = 1 (WRITE → IDLE)

SS_n = 0
&&
Write Data is being converted from serial to parallel in 9 more clock cycles (WRITE self loop)

SS_n = 0 (IDLE → CHK_CMD)

SS_n = 0
&&
MOSI = 0 (CHK_CMD → WRITE)
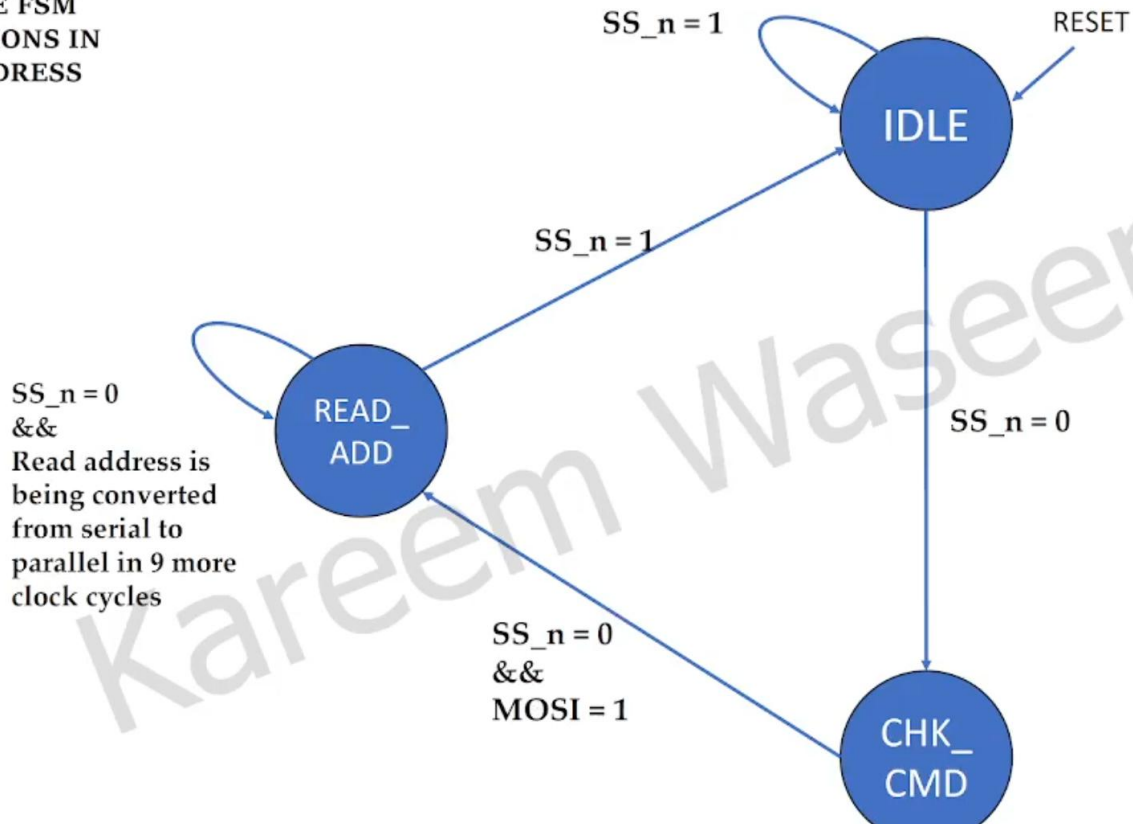
RAM Read Command – Read Address

1. Master will start the write command by sending the read address value, rx_data[9:8] = din[9:8] = 2'b10

2. SS_n = 0 to tell the SPI Slave that the master will begin communication

3. SPI Slave check the first received bit on MOSI port '1' which is a control bit to let the slave determine which operation will take place "read in this case". SPI Slave then expects to receive 10 more bits, the first 2 bits are "10" on two clock cycles and then the rd_address will be sent on 8 more clock cycles

4. Now the data is converted from serial "MOSI" to parallel after writing the rx_data[9:0] bus

5. rx_valid will be HIGH to inform the RAM that it should expect data on din bus

6. din takes the value of rx_data

7. RAM checks on din[9:8] and find that they hold "10"

8. RAM stores din[7:0] in the internal read address bus
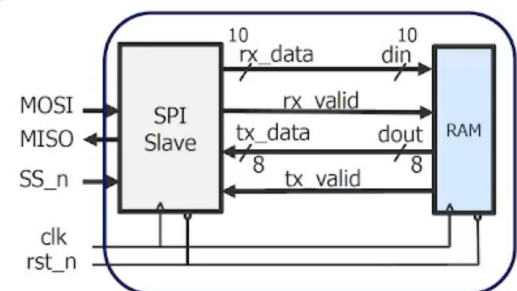
9. SS_n = 1 to end communication from Master side

SS_n = 1

RESET

IDLE

SS_n = 1

SS_n = 0

READ_ADD

SS_n = 0
&&
Read address is being converted from serial to parallel in 9 more clock cycles
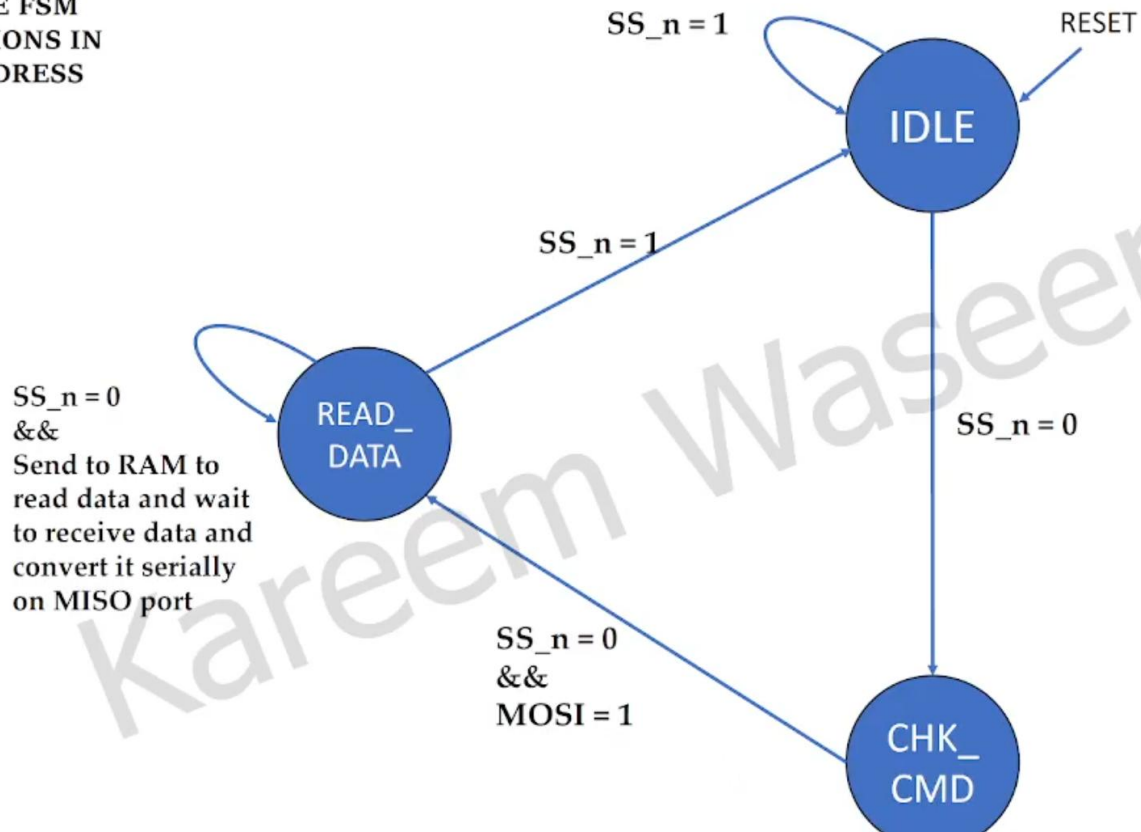
SS_n = 0
&&
MOSI = 1

CHK_CMD

RAM Read Command – Read Data

1. Master will start the write command by sending the read address value, rx_data[9:8] = din[9:8] = 2′b1**1**

2. SS_n = 0 to tell the SPI Slave that the master will begin communication

3. SPI Slave check the first received bit on MOSI port '1' which is a control bit to let the slave determine which operation will take place "read in this case". SPI Slave then expects to receive 10 more bits, the first 2 bits are "11" on two clock cycles and then dummy data will be sent and ignored since the master is waiting for the data to be sent from slave side

4. Now the data is converted from serial "MOSI" to parallel after writing the rx_data[9:0] bus

5. din takes the value of rx_data

6. RAM reads din[9:8] and find that they hold "11"

7. RAM will read from the memory with rd_address previously held

8. RAM will assert tx_valid to inform slave that data out is ready

9. Slave reads tx_data and convert it into serial out data on MISO port

10. SS_n = 1, Master ends communication after receiving data
    "8 clock cycles"

MOSI
MISO
SS_n

SPI Slave

clk
rst_n

10
rx_data

rx_valid

tx_data

8

tx_valid

10
din

dout

8

RAM

SPI SLAVE FSM TRANSITIONS IN READ ADDRESS

SS_n = 1

RESET

IDLE

SS_n = 1

SS_n = 0

SS_n = 0
&&
Send to RAM to
read data and wait
to receive data and
convert it serially
on MISO port

READ_
DATA

SS_n = 0
&&
MOSI = 1

CHK_
CMD

SPI Slave implementation Suggestions

- Split the SPI Slave design into three **always** blocks
  - State Memory always block
  - Next state Logic always block
  - Output Logic always block with posedge clk as the sensitivity

- Extra signals that you will need
  1. Counter for serial to parallel and vice verse Conversion
  2. Internal Signal to allow SPI slave to memorize if the read address is received or not to decide for READ_ADD or READ_DATA transition

whenever
_____
always @