Basics of

SPI PROTOCOL

UMMIDI CHANDRIKA

Serial Peripheral Interface (SPI)

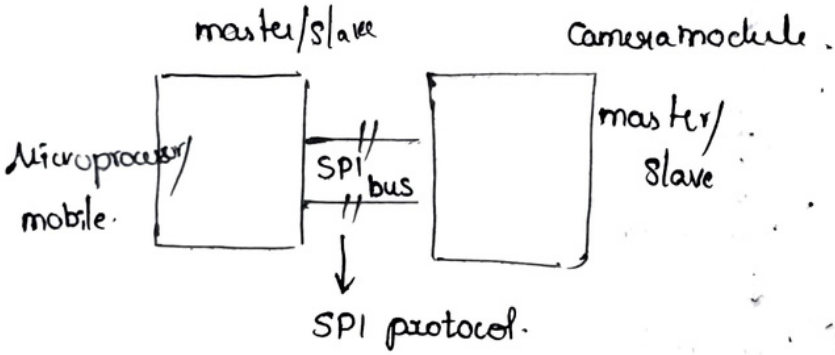
→ developed by Motorola (low cost and simple interface, ease of use, useless hardware and system resources).

→ SPI bus is used to send data between microcontrollers and peripherals like EEPROM, ADC, DAC, RTC (Real time clk) Sensors, SD card, LCD, RFID card module, wireless Tx/Rx.

→ it is a serial communication protocol.

→ it is a synchronous data bus as master or slave is synchronized on edge of clk

→ max speed go over 10mbps.

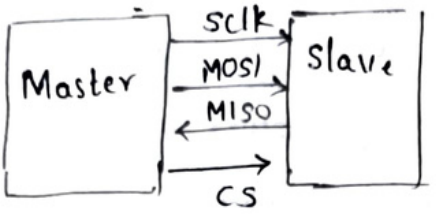


→ it is simple bus having 4 wires for data communication.

- SCLK (serial clock used for all devices)
- MOSI (Master out Slave in o/p dataline from master)
- MISO (Master in Slave out i/p dataline for master coming from Slave)
- SS/CS (Slave select / Chip select used to select a slave).

→ it's a simple master communication protocol where one device initiates communication with slaves

→ it is full duplex (both master and slave can send data at same time through MOSI and MISO lines respectively).



→ it is normally used for short distance communication means communication within same ckt board or b/w devices on same PCB.

→ it supports single master means here one device considered master (μC or processor) & all other devices (peripherals or other μC) are considered as slaves. Slave takes instruction from master.

→ For every slave, there is one $\overline{SS}$ (slave select) lines so it occupies space when more slaves are added. it is disadvantage.

→ Here one master but can have many slaves.

→ it is preferably where large amounts of high speed data is not transferred.

→ Here, any no. of bit can be sent or rcvd in continous stream. it is adv. (some protocol support fixed data width)

→ it doesn't have any start/stop bit. So simple communication. So no extra ckt required

→ In SPI master-slave both share the same clock and clock is produced by master.

→ SPI works in full duplex mode, which means it can receive and send data at a time.

→ In SPI, data is shifted out from master and shifted into master through shift register.

→ SPI is a primitive protocol without an acknowledgment mechanism for checking received or sent data

SPI - Data Transmission:

→ To begin SPI communication, master must send clock signal and select slave by enabling SS signal.

→ Usually chip select is an active low signal, hence master must send logic 0 on this signal to select the slave.

→ During SPI communication, data is simultaneously transmitted (shifted out serially onto MOSI/SDO bus) and received (data on bus (MISO/SDI) is sampled or read in)

→ The serial clock edge synchronizes shifting & sampling of data.

→ The SPI interface provides user with flexibility to select raising or falling edge of clock to sample and/or shift data.

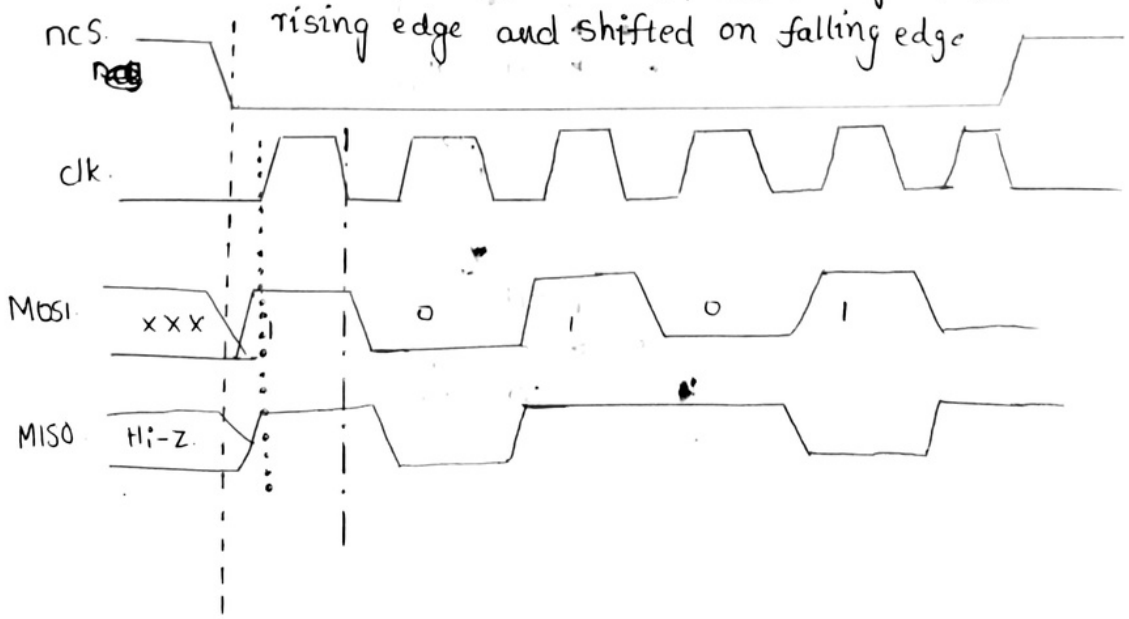
Clock polarity and clock phase:

SPI communication data is given in 4 modes by combination of CPOL & CPHA.

- clock polarity (CPOL) - designates default value (high or low) of SCLK when bus is idle.
- Clock phase (CPHA) - Determine which edge of clock data is sampled (rising / falling).
- CPOL & CPHA has to match with SPA slaves for proper data transfer.

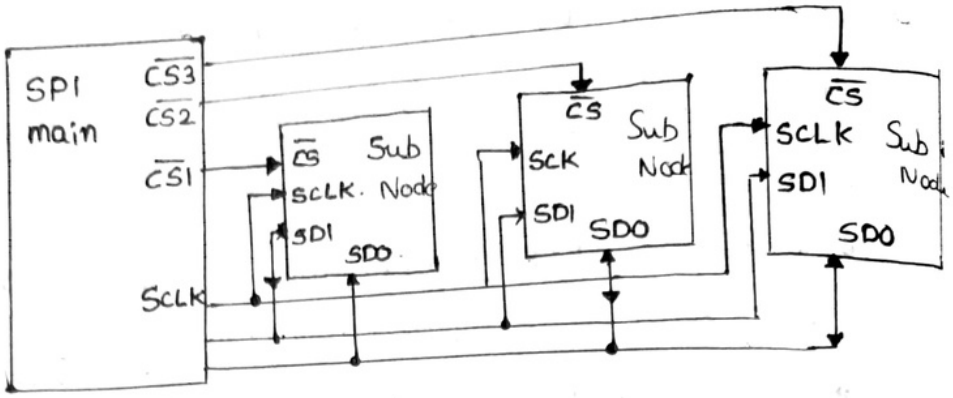
SPI modes	CPOL	CPHA	clock polarity in idle state	clock phase
0	0	0	Low	data sampled on rising edge and shifted out on falling edge
1	0	1	Low	data sampled on falling edge & shifted out on rising edge
2	1	0	High	data sampled on falling edge & shifted out on rising edge
3	1	1	High	data sampled on rising edge and shifted on falling edge.

i) SPI mode = 0, CPOL = 0, CPHA = 0, data sampled on rising edge and shifted on falling edge



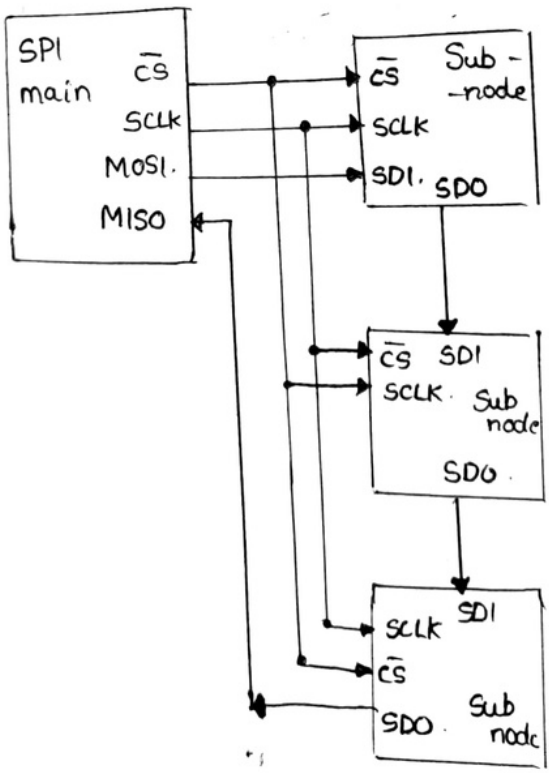
SPI - multi slave Configurations:

- Independent Slave Configuration:



- There are independent slave line (cs1, cs2...) from master to each slave. SPI slave output MISO will be tristate pin, so it will be high impedance state when slave is not selected.
- Slave line cs will make slow and keep rest of them high.

- Daisy chain Configuration:



- SPI can be connected one after another in serial form. In this configuration, a single slave select line is used to select all daisy chain slaves.
- whole chain acts as Communication through shift registers connected in series.
- Each daisy chain slave is supposed to send out exact copy of data received in 1st group of clock cycles during 2nd group of clock cycles

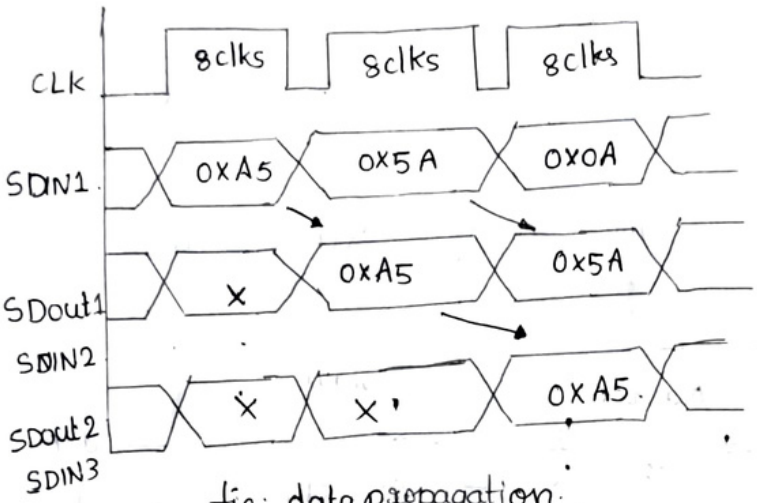
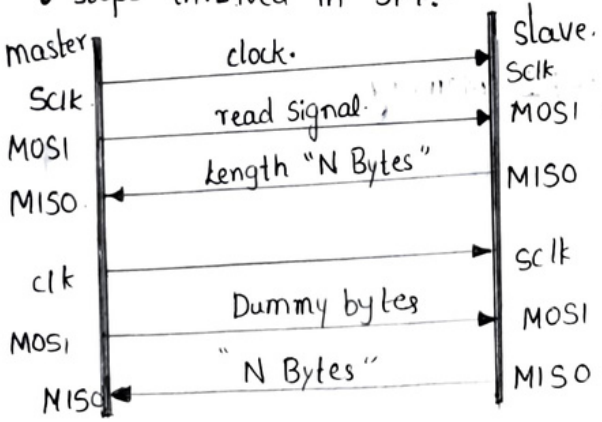


fig: data propagation

• Verilog Implementation of SPI:

- steps involved in SPI:



Let's assume Master wants to read some data collected by the slave which is a sensor.

- To start communication bus master configures clock signal by using frequency supported by slave.
- The master then pulls respective slave select pin to select slave it wishes to communicate with.
- The master then sends ~~data~~ read signal via MOSI which is acknowledged by slave by sending data length on MISO.
- depending on size of data, master keeps read signal high for required clock signals.
- The data transmitted back to master via MISO by the slave.
- other slaves must disregard SCLK and MOSI s/l during above process.

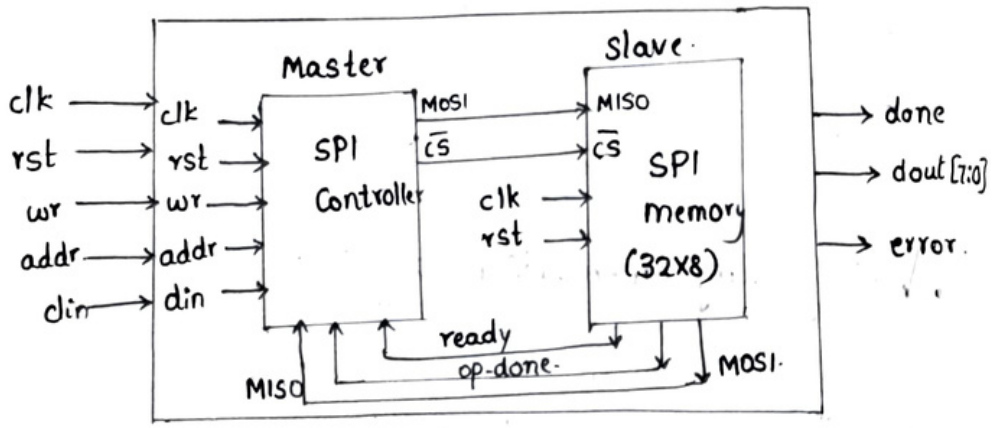


fig: SPI Memory.

- We need a SPI controller where we will be applying data that we want to write to memory or read from memory.
- SPI controller will convert the data into SPI signals, and that will be going to memory which will then respond to our request

Top module:

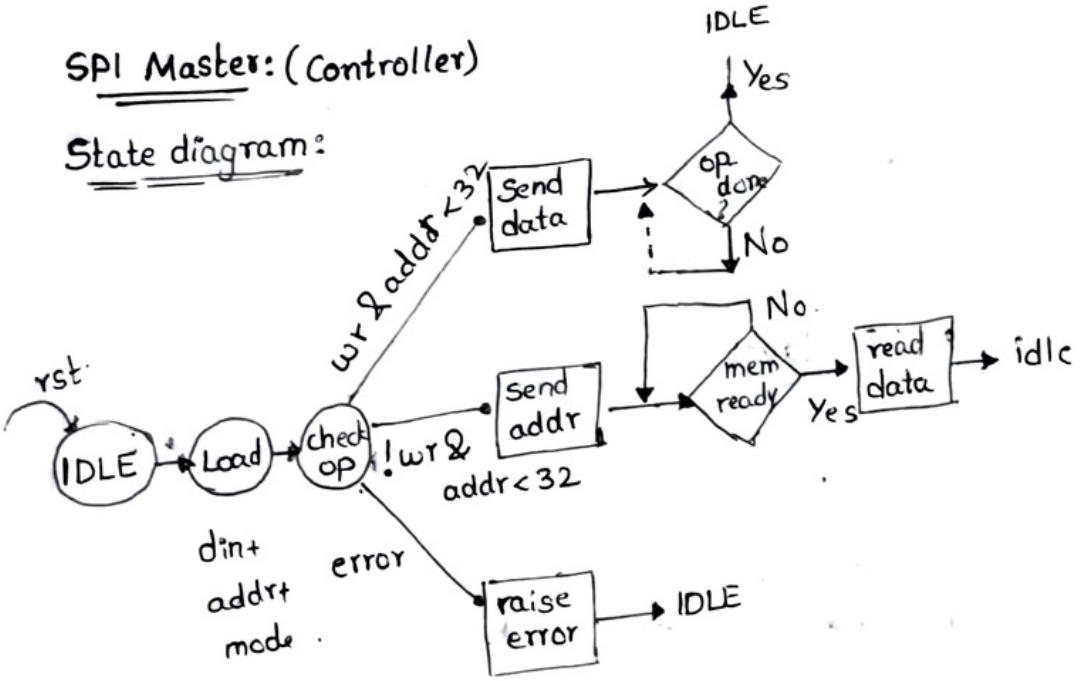
--> Signals include clock, reset, write which signify type of operation that we want to perform with the memory

if write = 1, writing data into memory
write = 0, reading data from memory address bus

- We have data bus and output of the SPI system includes done which indicates the completion of an operation.
- We also have an error signal which indicate whenever user add an address which is beyond capability of memory going through each of the system.

SPI Master: (Controller)

State diagram:



IDLE state: when reset is applied.

Load state: Sampling data applied by the user on the top module.
(din + addr + mode) are sampled.

3) check_op : check type of operation user want to perform. This depends on write pin.

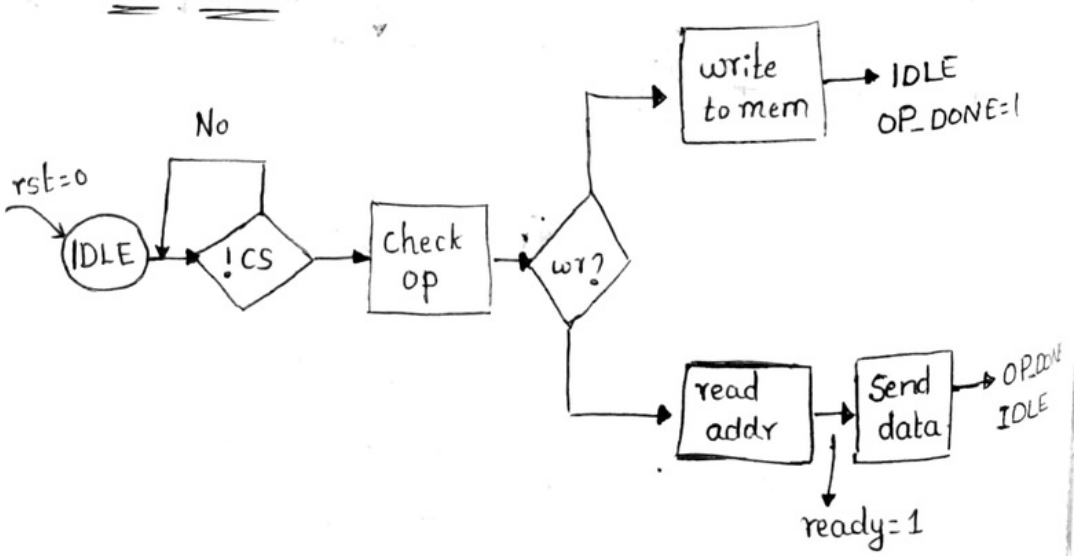
Case:1 → if wr = 1 & address is less than 32 as we are preferring 32X8 SPI memory. we send data along with address to SPI memory. Then we wait for memory to complete its write operation and after complete of write, it will raise done s/l. & that will be given as i/p to master as op_done = 1

Case:2 wr=0 & addr<32, it indicates read operation First, we send address to slave, then we wait till slave is ready to send data.

- To indicate that we have a handshaking signal called "ready", when ready = 1 data will be read.

Case 3: when addr > 32 & independent of write then error signal will be 1.

- **SPI Slave:**



chip Select = 0, then check_op state.

wr = 1, write data.

wr = 0, read addr & send data.

after read/write operations are completed and op_done will be high.

Advantages of SPI

1. It is simple protocol hence it doesn't require processing overheads. Designers need to understand respective read/write timing diagrams. of microcontrollers/ EEPROM inorder to use it.
2) it supports full duplex communication.
3). SPI pushes push-pull configuration & hence high data rates are supported.
4) Data rate in terms of 10's of MHZ.

Disadvantages of SPI:

1. It uses one CS line per slave & hence hardware complexity increases if more slaves are in design.
2. Inorder to add slave device, software needs to be changed. & extra CS line required.
3. Handshaking is needed.
4. Support only single master.