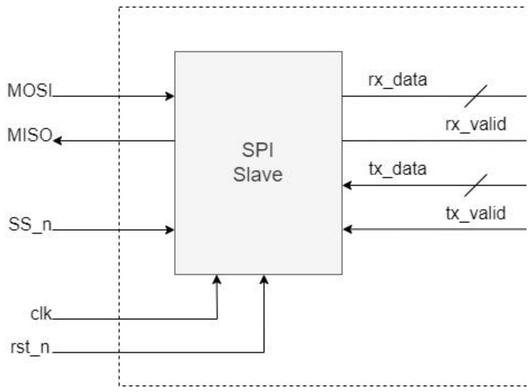
1. SPI Slave

Four Wires

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

MSB first on MISO and MOSI ports



2. Single-port Synchronous RAM

Parameters

- MEM_DEPTH: default: 256
- ADDR_SIZE: default: 8

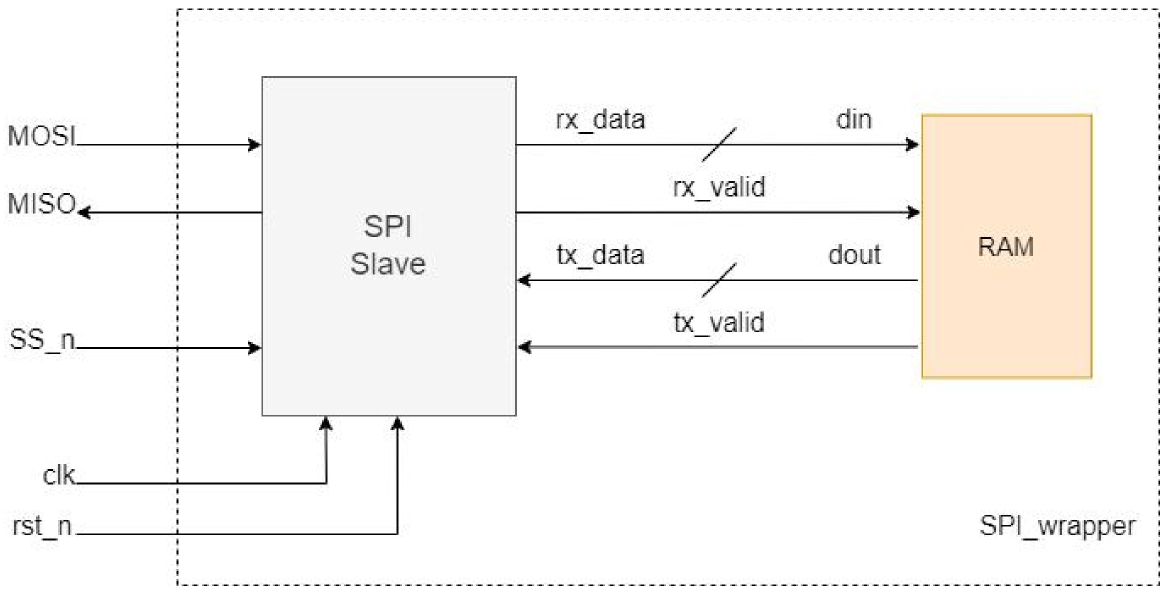
Ports

Port	Size	Direction	Function
din	10-bit	Input	Data Input
clk	1-bit		Clock signal
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	8-bit	Output	Data Output
tx_valid	1-bit		Asserts when the data is valid on dout

MSB of din "din[9:8]" determines if it's 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 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]

3. SPI Wrapper (SPI Slave + RAM)



I. RAM Write Command - Write Address

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

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

- 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

- Now the data is converted from serial “MOSI” to parallel after writing the rx_data[9:0] bus

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

- din takes the value of rx_data

- RAM checks on din[9:8] and find that they hold “00”

- RAM stores din[7:0] in the internal write address bus

- SS_n = 1 to end communication from Master side

II. RAM Write Command - Write Data

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

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

- 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

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

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

- din takes the value of rx_data

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

- RAM stores din[7:0] in the RAM with wr_address previously held

- SS_n = 1 to end communication from Master side

III. RAM Read Command - Read Address

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

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

- 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

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

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

- din takes the value of rx_data

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

- RAM stores din[7:0] in the internal read address bus

- SS_n = 1 to end communication from Master side

IV. RAM Read Command - Read Data

- Master will start the write command by sending the read address value, rx_data[9:8] = din[9:8] = 2'b11

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

- 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

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

- din takes the value of rx_data

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

- RAM will read from the memory with rd_address previously held

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

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

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