SPI slave with single port RAM

- This project involves designing and implementing the SPI (Serial Peripheral Interface) protocol with a single port RAM using Verilog. The SPI protocol is a crucial communication interface for short-distance data transfer in embedded systems, while the single port RAM provides efficient memory storage.
- SPI communication protocol is very popular and most usage due to 1) low cost, 2) simple implementation, 3) ease of use, 4) use less hardware.

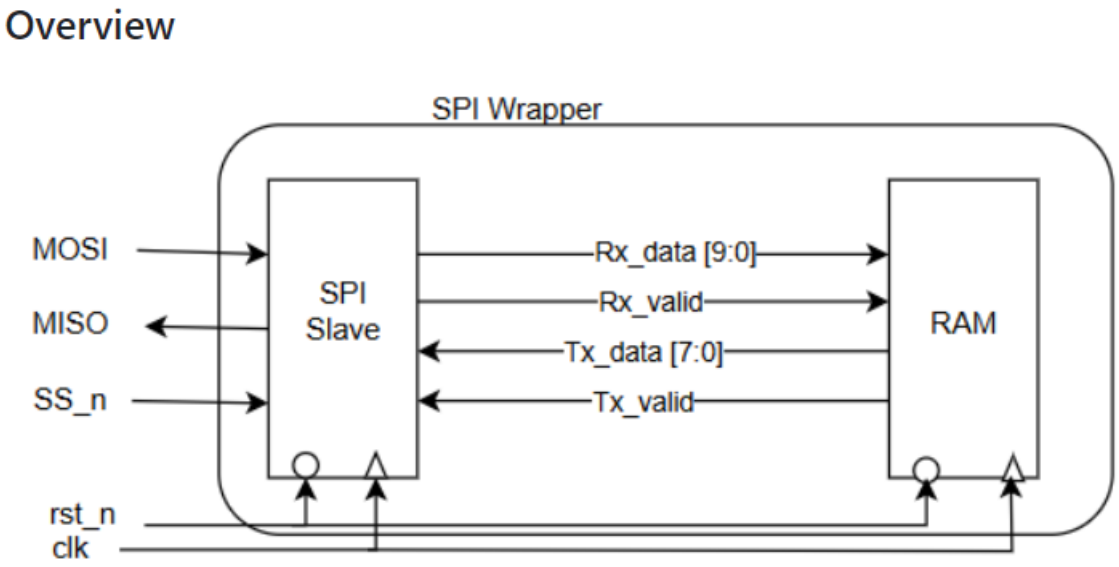
Data Ports

Module	Port Name	Direction	Data Width	Description
Master-Slave Interface	MOSI	Input	1 bit	Master Out Slave In - Data input into slave
	MISO	Output	1 bit	Master In Slave Out - Data output from slave
	clk	Input	1 but	Clock signal for synchronization
	SS_n	Input	1 bit	Slave Select - To start the serial communication
	rst_n	Input	1 bit	Reset signal for initializing the SPI slave
Single Port RAM	din	Input	10 bits	Data input for writing to memory
	dout	Output	8 bits	Data output for reading from memory
	clk	Input	1 bit	Clock input for synchronizing memory operations
	rst_n	Input	1 bit	Reset signal for initializing the RAM

Internal Configuration

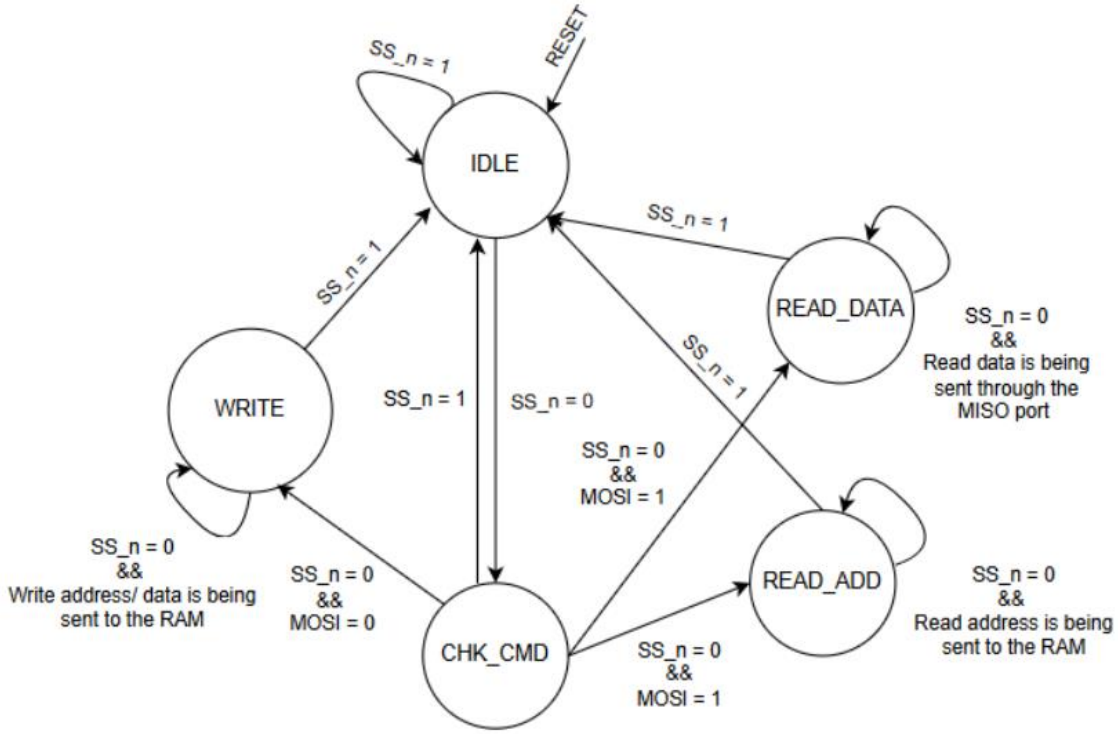
Module	Port Name	Direction	Data Width	Description
Slave-RAM link	rx_data	Output	10 bit	Transferred data/address/control from slave to RAM
	rx_valid	Output	1 bit	A flag to make the RAM accept the data on rx bus
	tx_data	Input	8 bit	The data retreived from the RAM
	tx_valid	Input	1 bit	A flag to make the slave convert the tx data to the MISO port

❖ Complete diagram of SPI slave with single port RAM



❖ Where the SPI slave module operates on FSM concept with 5 states (**IDLE, CHK_CMD, WRITE, READ_ADD, READ_DATA**).

The serial input from the master side is being handled through a finite state machine elaborated as follows :

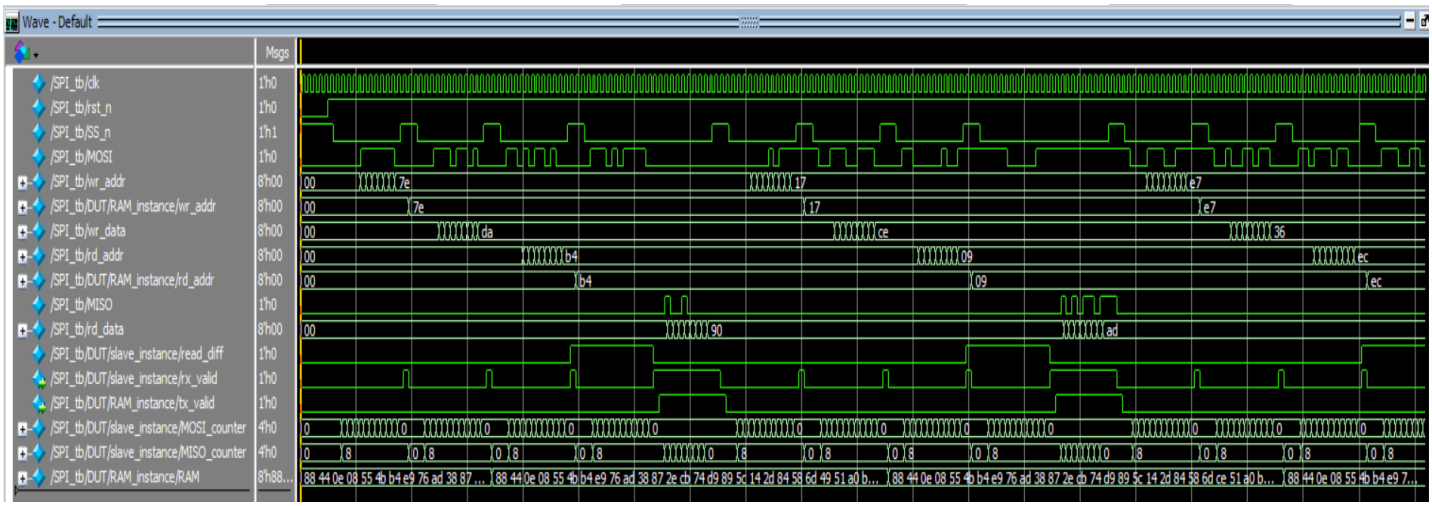


RAM control

The design uses single-port synchronous RAM for memory operations & the serial control is given on the same data bus as follows:

Port	Din[9:8]	Command	Description
din	00	Write	Hold din[7:0] internally as write address
	01	Write	Write din[7:0] in the memory with write address held previously
	10	Read	Hold din[7:0] internally as read address
	11	Read	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]

- Here is the wave form snippet from the Questa Sim



- **<u>Note</u>:** we simulate the design with different FSM "finite state machine" encoding as **binary coding, gray coding** & **one-hot coding** as we found that the gray encoding is the best timing then one-hot which is moderate timing then binary which is the worst timing due to multiple bit transition occur.