MAVEN SILICON

TITLE OF THE PROJECT: SPI DESIGN

NAME: S SAI CHARAN

3rd Year ECE

VIT UNIVERSTITY VELLORE

Introduction:

SPI PROTOCOL

SPI stands for Serial Peripheral Interface.

Serial peripheral interface (SPI) is one of the most widely used interfaces between microcontroller and peripheral ICs such as sensors, ADCs, DACs, shift registers, SRAM, and others.

SPI is a synchronous protocol that allows a master device to initiate communication with a slave device.

Data is exchanged simultaneously between these devices in a serial manner.

The SPI Master core is compatible with both above-mentioned protocols as master with some additional functionality. At the hosts side, the core acts like a WISHBONE compliant slave device.

Features:

- **Full duplex synchronous serial data transfer**
- **Variable length of transfer word up to 128 bits**
- **MSB or LSB first data transfer**
- **Rx and Tx on both rising or falling edge of serial clock independently**
- **8 slave select lines**
- **Fully static synchronous design with one clock domain**
- **Technology independent Verilog**
- **Fully synthesizable**

2.1 WISHBONE interface signals

Port	Width	Direction	Description
wb_clk_i	1	Input	Master clock
wb_rst_i	1	Input	Synchronous reset, active high
wb_adr_i	5	Input	Lower address bits
wb_dat_i	32	Input	Data towards the core
wb_dat_o	32	Output	Data from the core
wb_sel_i	4	Input	Byte select signals
wb_we_i	1	Input	Write enable input
wb_stb_i	1	Input	Strobe signal/Core select input
wb_cyc_i	1	Input	Valid bus cycle input
wb_ack_o	1	Output	Bus cycle acknowledge output
wb_err_o	1	Output	Bus cycle error output
wb_int_o	1	Output	Interrupt signal output

Table 1: Wishbone interface signals

All output WISHBONE signals are registered and driven on the rising edge of wb_clk_i. All input WISHBONE signals are latched on the rising edge of wb_clk_i.

2.2 SPI external connections

Port	Width	Direction	Description
/ss_pad_o	8	Output	Slave select output signals
sclk_pad_o	1	Output	Serial clock output
mosi_pad_o	1	Output	Master out slave in data signal output
miso_pad_i	1	Input	Master in slave out data signal input

Table 2: SPI external connections

3.1 Core Registers list

Name	Address	Width	Access	Description
Rx0	0x00	32	R	Data receive register 0
Rx1	0x04	32	R	Data receive register 1
Rx2	0x08	32	R	Data receive register 2
Rx3	0x0c	32	R	Data receive register 3
Tx0	0x00	32	R/W	Data transmit register 0
Tx1	0x04	32	R/W	Data transmit register 1
Tx2	0x08	32	R/W	Data transmit register 2
Tx3	0x0c	32	R/W	Data transmit register 3
CTRL	0x10	32	R/W	Control and status register
DIVIDER	0x14	32	R/W	Clock divider register
SS	0x18	32	R/W	Slave select register

Table 3: List of core registers

All registers are 32-bit wide and accessible only with 32 bits (all wb_sel_i signals must be active).

3.2 Data receive registers[RxX]

Bit #	31:0
Access	R
Name	Rx

Table 4: Data Receive register

Reset Value: 0x00000000

3.3 Data transmit register [TxX]

Bit #	31:0
Access	R/W
Name	Tx

Table 5: Data Transmit register

Reset Value: 0x00000000

3.4 Control and status register [CTRL]

Bit #	31:14	13	12	11	10	9	8	7	6:0
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Name	Reserved	ASS	IE	LSB	Tx_NEG	Rx_NEG	GO_BSY	Reserved	CHAR_LEN

Table 6: Control and Status register

Reset Value: 0x00000000

3.5 Divider register [DIVIDER]

Bit #	31:16	15:0
Access	R	R/W
Name	Reserved	DIVIDER

Table 7: Divider register

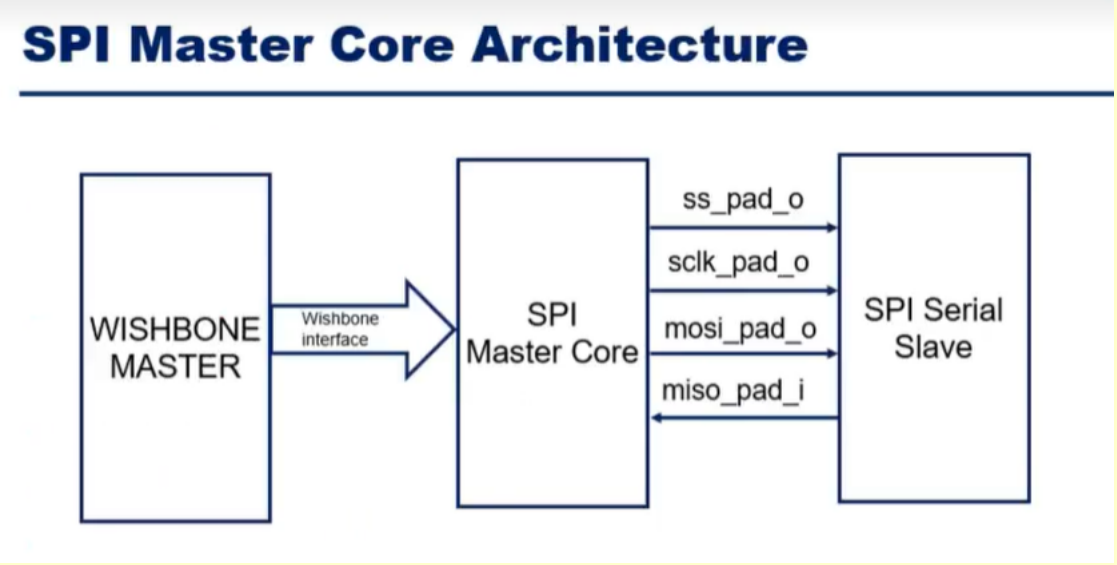
Reset Value: 0x0000ffff

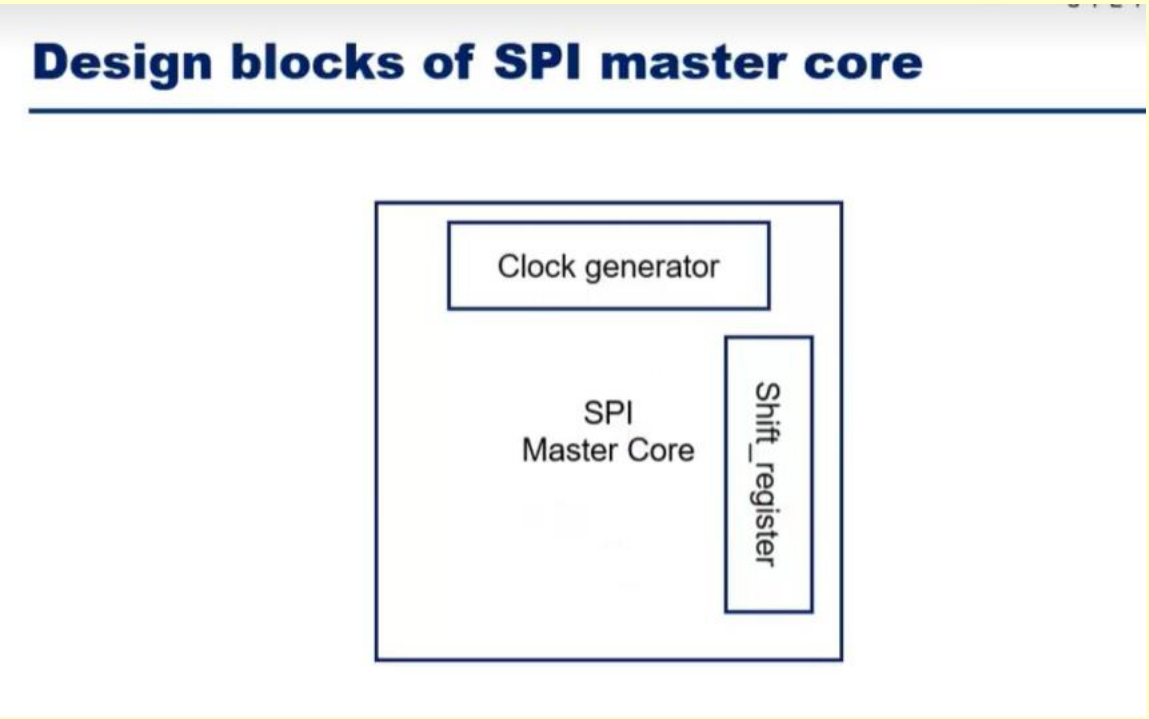
DIVIDER

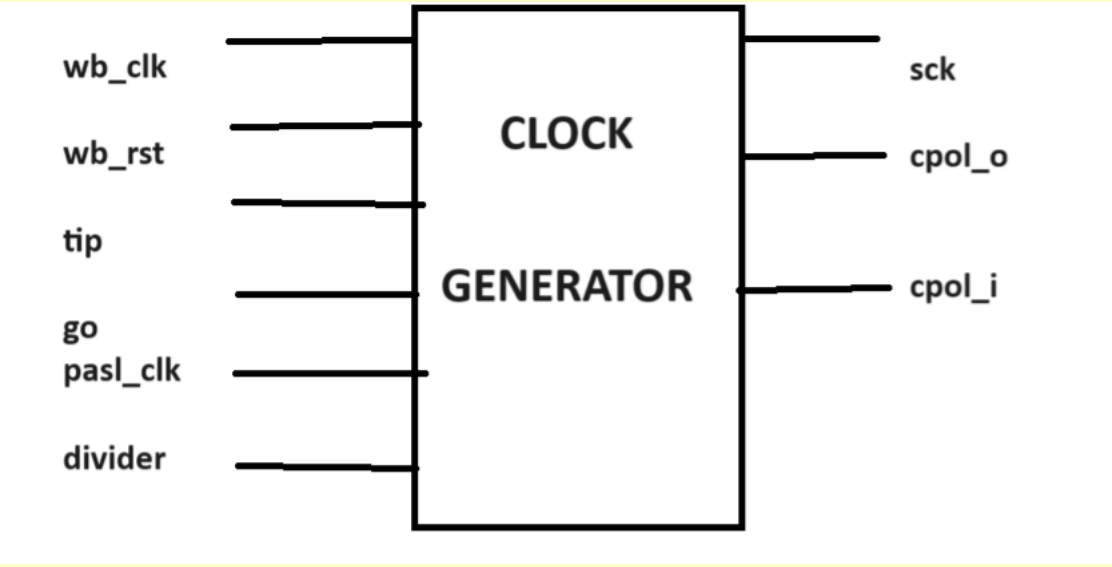
The value in this field is the frequency divider of the system clock wb_clk_i to generate the serial clock on the output sclk_pad_o. The desired frequency is obtained according to the following equation:

$$f_{sclk} = \frac{f_{wb\_clk}}{(DIVIDER+1)*2}$$

Block diagram:

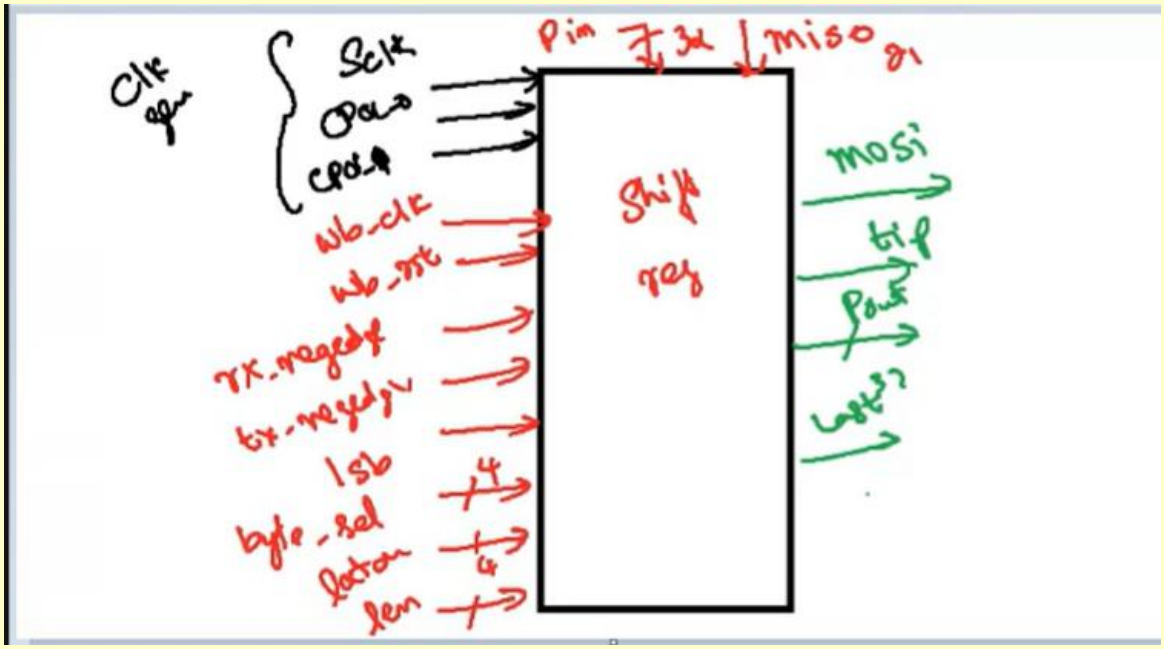






- **Sck will generate the serial clock from the clock generator**
- **Cpol_1 and Cpol_0 will indicate the edges of the clock**
- **Wb_clk,wb_rst,tip,go,last clk,divider,go are the inputs**
- **The serial transmission that needs to be done happens only after the G0_BUSY of CTRL register becomes 1**

Divider will tell us by what value the wishbone clock has to be divided which uses lower 16 bits for serial transmission

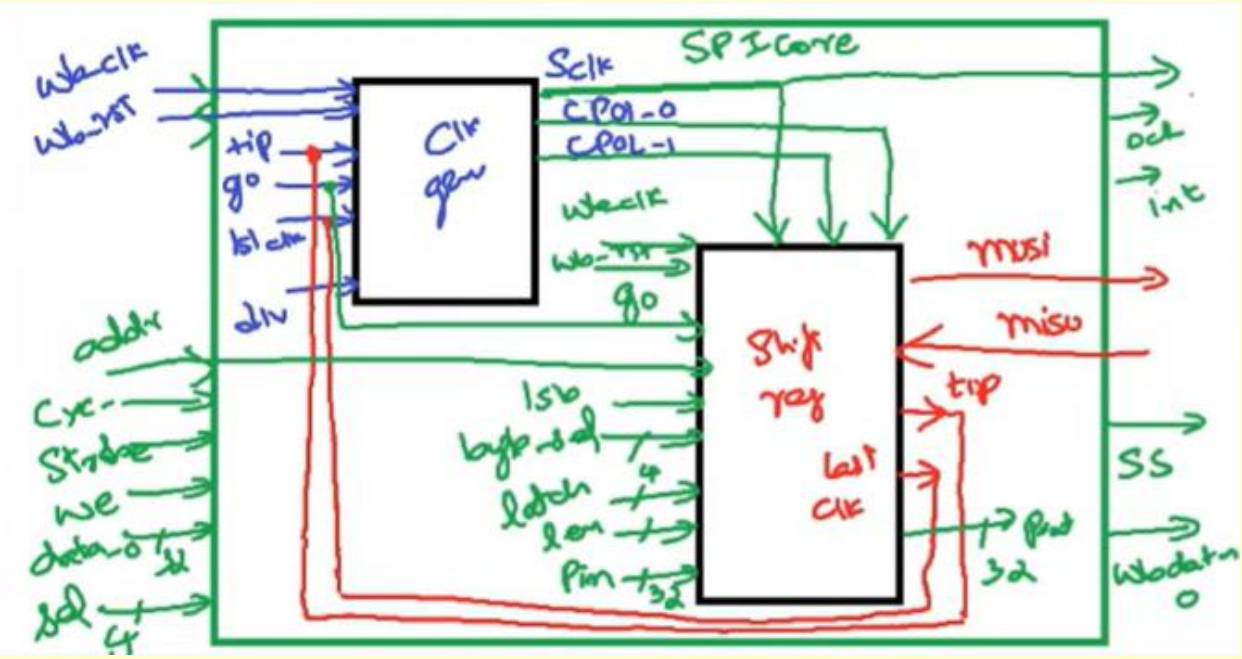


- **If LSB=1, Transmission (Tx) and Receiving (Rx) will happen from index [0]**
- **If LSB=0,Transmission (Tx) and Receiving (Rx) will happen from MSB index**
- **Byte_sel will allow us to select the registers**
- **Latch input will decide which byte is accessed among the 4 bytes within the register**
- **Len specifies the number of characters that are to be transmitted**
- **Pin is the parallel input that specifies the data coming from the wishbone master**
- **Miso(MasterinSlaveout)- slaves sends information that is received by the master and this is also an input for shift register**
- **Sclk, cpol_0, cpol_1 are inputs to shift register coming from output of clock generator**
- **If Rx_negedge=1, receiving information will happen on falling edge**

- If Rx_negedge=0, receiving information will happen on rising edge
- If Tx_negedge=1, receiving information will happen on rising edge
- If Tx_negedge=0, receiving information will happen on falling edge

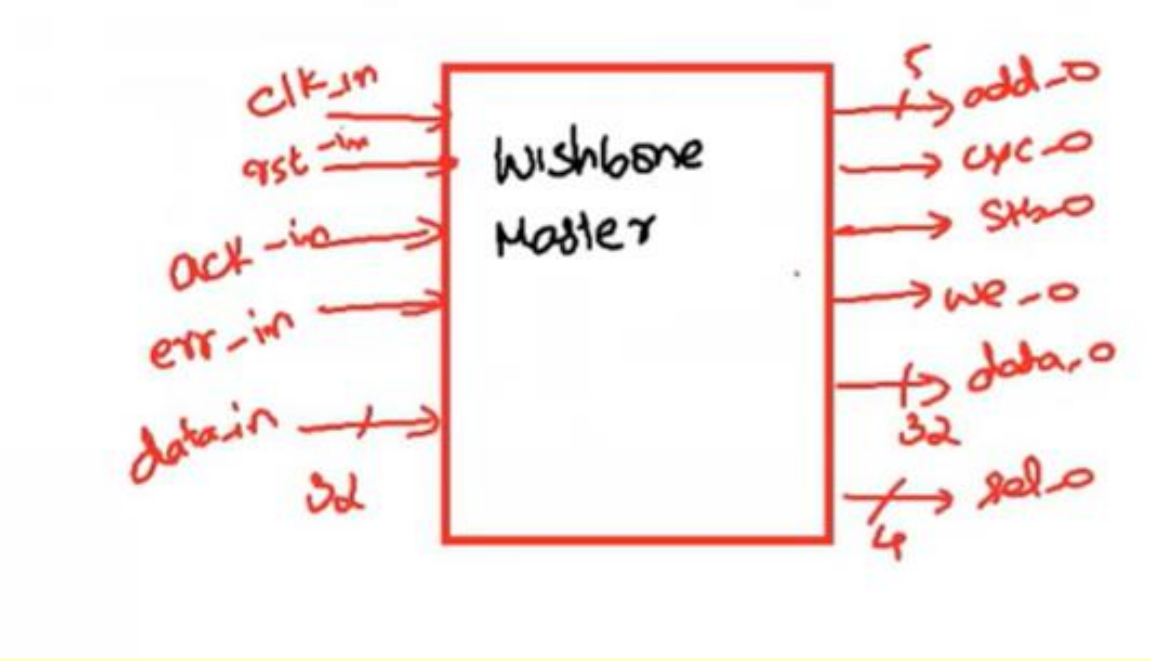
- Mosi(MasteroutSlavein),TIP,Pout,Last are the outputs of shift register
- Last clk indicates completion of serial communication

Top module:



- Outputs of wishbone master are inputs to SPI master core
- We have
- Slave select signal
- wb_dataout
- Acknowledgement signal
- Interrupt enable signal as outputs to SPI Core

Sub-level



- **Data Inputs are driven only when the acknowledgement is high**
- **The data bus is driven high to perform serial communication as read/write operation from wishbone master**
- **Shift register gets serial clock from clock generator by connecting the top module**
- **If slave is acting as receiver,master will act as transmitting device**
- **If master is acting as receiver,slave will be acting as transmitting device**

Functionality:

SPI is a widely used synchronous serial communication protocol that allows multiple peripheral devices to communicate with a microcontroller or microprocessor.

Here are the key functionalities of an SPI core:

- **Serial Communication: The SPI core is responsible for enabling serial communication between a microcontroller or microprocessor and one or more peripheral devices. It allows for full-duplex communication, meaning data can be sent and received simultaneously.**

- **Master/Slave Operation: SPI can operate in master-slave mode. The SPI core can be configured as either a master or a slave device, depending on its role in the system. The master initiates data transfers, while the slave responds to requests from the master.**

- **Clock Generation: SPI uses a clock signal (SCK) to synchronize data transfer. The SPI core generates and manages the clock signal, ensuring that both the master and slave devices are synchronized.**

- **Data Transmission: The SPI core handles the transmission of data bits, typically in 8-bit or 16-bit chunks. It serializes and deserializes the data for transmission.**

- **Data Framing: SPI allows for flexible data framing. The core may support different data frame formats, including various numbers of bits per frame and different clock polarity and phase settings.**

- **Chip Select (CS) Control: SPI uses chip-select signals to enable or disable specific peripheral devices. The SPI core manages the chip-select lines to select the target device for communication.**

- **Full-Duplex Communication: SPI supports full-duplex communication, which means data can be sent and received simultaneously. The SPI core handles the bidirectional data flow.**

- **Data Rate Configuration: The SPI core often allows the configuration of the data rate or clock speed, enabling communication at different speeds to suit the requirements of the connected devices.**

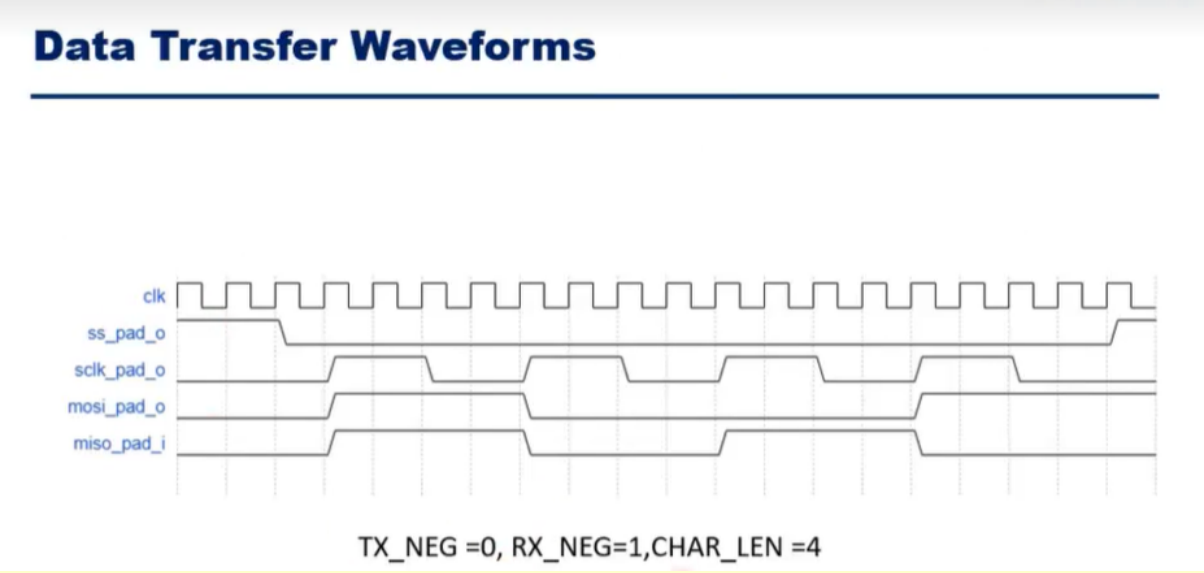
- **Error Handling: The SPI core may include error detection and handling mechanisms to ensure the integrity of the data transfer.**

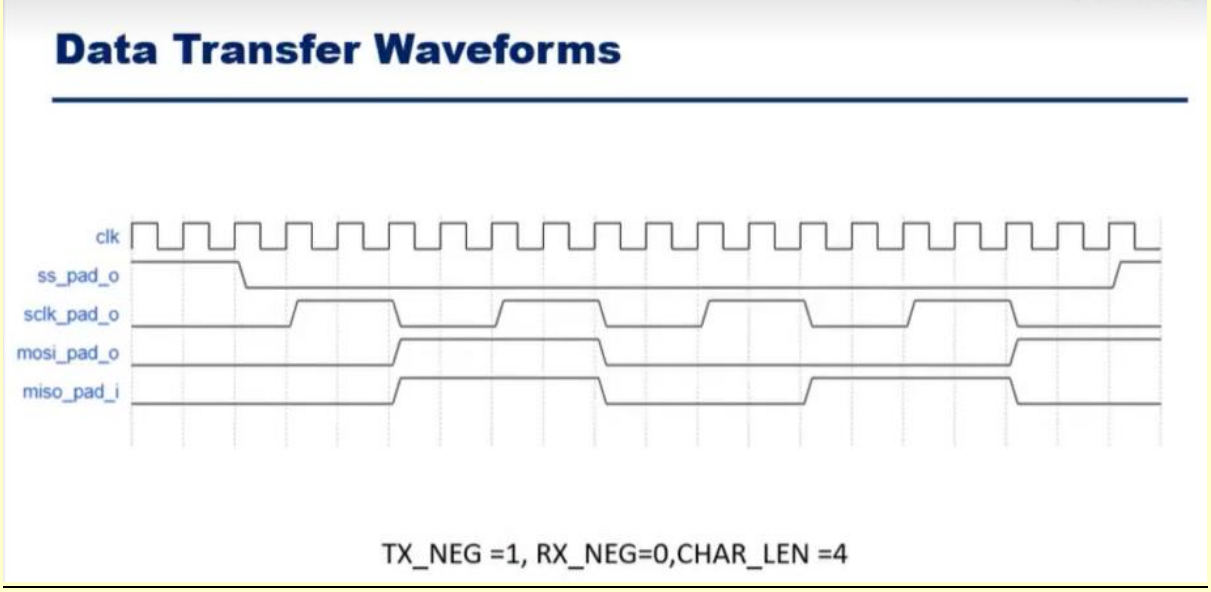
- **Buffering: To optimize data transfer and minimize CPU overhead, the SPI core may include buffer memory to temporarily store data before transmission or after reception.**

- **Interrupts or DMA: Many SPI cores provide interrupt-driven or Direct Memory Access (DMA) capabilities to offload the CPU from actively managing data transfer, allowing it to handle other tasks.**

- **Configuration Registers: SPI cores often come with a set of configuration registers that allow the user to specify the operational parameters, such as clock polarity, clock phase, and data frame format**

Waveform:





Functionality of SPI CLOCK GENERATOR:

The clock generator generates the clock signal (often called SCK or SCLK) that synchronizes data transmission between the devices. Here's an overview of the functionality of an SPI clock generator:

- **Clock Signal Generation: The primary function of an SPI clock generator is to produce a clock signal. This clock signal determines the timing for data transmission on the SPI bus. The clock signal is typically a square wave with a specific frequency and duty cycle.**

- **Frequency Control: SPI clock generators allow you to set the clock frequency. The clock frequency is important because it determines how fast data can be transmitted between devices. You can configure the clock generator to generate clock signals with different frequencies to meet the requirements of your SPI communication.**

- **Duty Cycle Control: In some SPI implementations, you can also control the duty cycle of the clock signal. The duty cycle represents the ratio of time the clock signal is high (active) to the total clock period. The typical duty cycle for SPI is 50%, but some applications may require variations.**

- **Phase and Polarity Control: Some SPI clock generators allow you to control the phase and polarity of the clock signal. This can be useful for compatibility with different SPI devices that might have specific requirements regarding clock signal timing.**

- **Synchronization: The clock generator ensures that both the master and slave devices are synchronized to the same clock signal. This synchronization is essential for accurate data transmission and reception.**

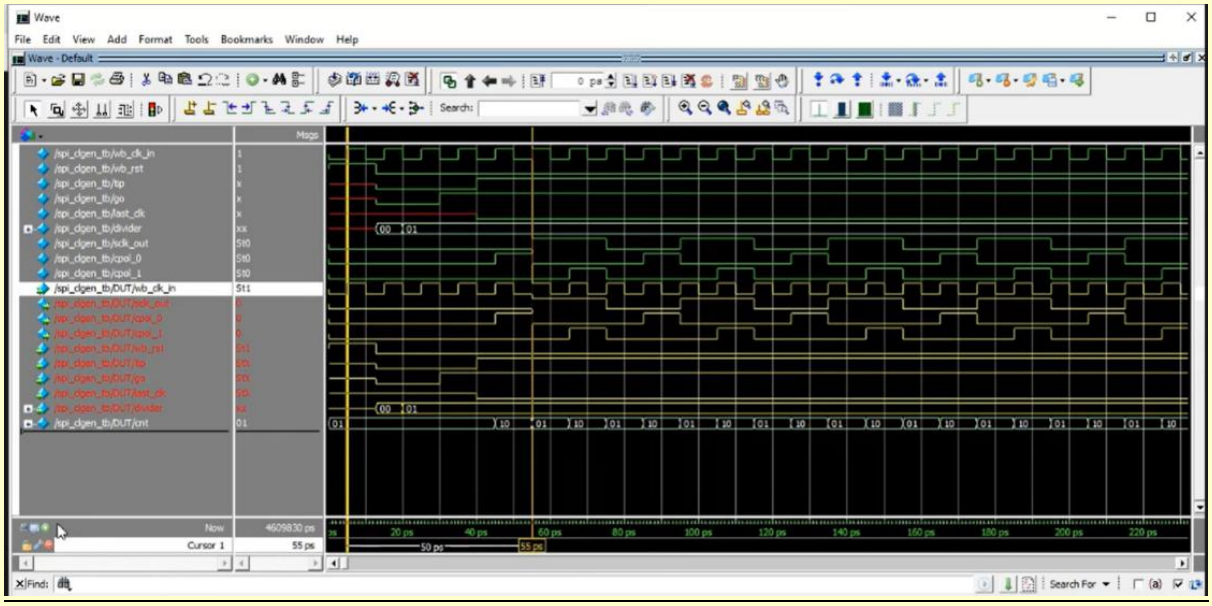
- **Clock Source Selection: In some systems, you can choose the source of the clock signal, which may be an external crystal oscillator, an internal oscillator, or another reference source.**

- **Error Handling: Some clock generators include error detection and correction features to ensure that the clock signal remains stable and reliable during communication. This is important for minimizing data errors.**

- **Power Management: Some clock generators provide power management features, allowing you to adjust the power consumption based on the operational requirements of your system.**

- **Configurability: SPI clock generators are often configurable via software or hardware settings, allowing you to adapt the clock signal to your specific application.**

Clock generator:



- 4 cycles of wishbone clock is equal 1 cycle of serial clock
- For one clock cycle, the cpol_0 should be 1
- GO_BUSY and TIP should be made high before starting the serial communication
- CPOL_1 detects the falling edge of serial clock
- CPOL_0 is detects the rising edge of serial clock
- Last clk tells us the last serial clock generated
- Wishbone clk is the clock input
- Whenever the count is incrementing the wishbone clock and if its value becomes equal to (Divider+1), then the count variable will go back to the initial value.
- To make CPOL_0 HIGH:
- wb_clk_in = 1
- wb_rst=0
- divider=01
- last_clk=0
- tip=1
- go=1

Functionality of Shift Register:

The shift register is a crucial part of the SPI bus, facilitating the serial transfer of data between devices. Here's an overview of its functionality:

- **Serial Data Transfer: The shift register serves as a shift buffer for data transfer. Data is shifted into and out of the shift register bit by bit. It can be considered a line of memory cells that store bits of data.**

- **Synchronous Communication: SPI communication is synchronous, meaning that data is transferred based on a clock signal (SCK or CLK). The shift register operates in sync with this clock, transferring data bits in parallel across all connected devices.**

- **Parallel-to-Serial Conversion: When you send data from a microcontroller to a device via SPI, the data is parallel within the microcontroller (in bytes) but needs to be serialized to send over the SPI bus. The shift register helps convert parallel data into a serial stream for transmission.**

- **Serial-to-Parallel Conversion: When receiving data, the shift register receives serial data and converts it back to parallel data. This is crucial for correctly interpreting data received from an SPI device.**

- **Shift Direction: Depending on the SPI mode and configuration, the shift register can shift data in either a left or right direction. This is determined by the Clock Polarity (CPOL) and Clock Phase (CPHA) settings in the SPI mode.**

- **Buffering: The shift register can temporarily store data being transmitted or received. This is important because the SPI protocol typically involves full-duplex communication**

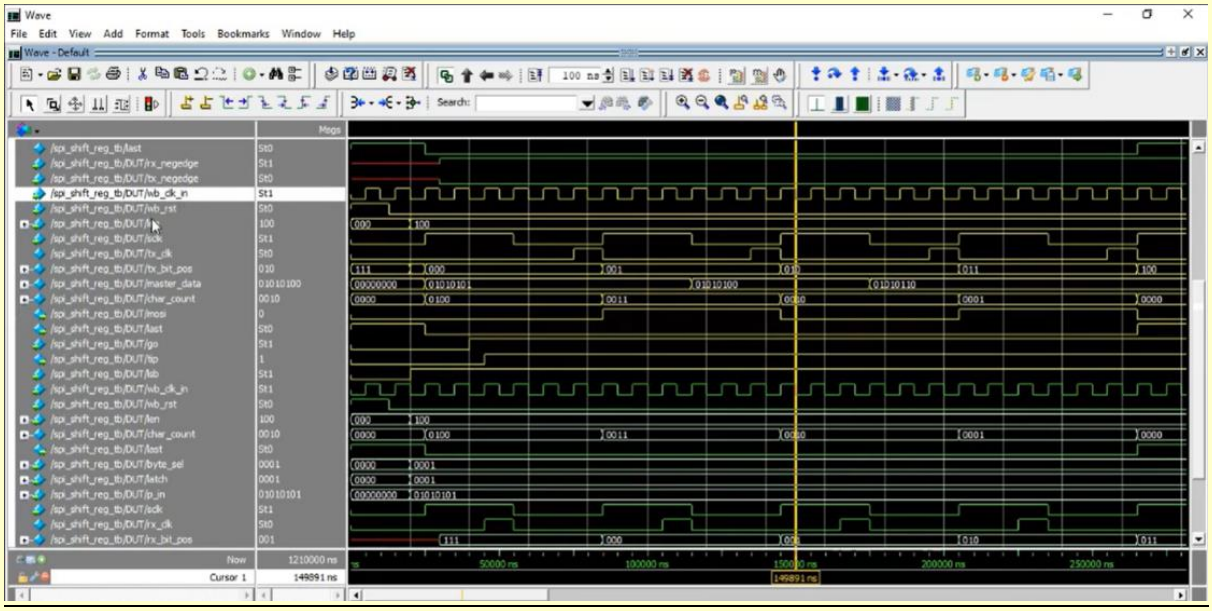
(transmitting and receiving simultaneously), and the shift register helps buffer data in transit.

- **Shift Register Size:** The size of the shift register can vary depending on the specific microcontroller or device. It can be 8, 16, 32, or more bits wide, depending on the system's requirements.

- **Cascade Configuration:** In some cases, multiple shift registers can be cascaded together to increase the number of bits that can be transferred in a single operation.

- **Master-Slave Operation:** In SPI communication, one device (usually a microcontroller) acts as the master, and the other device(s) act as slaves. The shift register is often a part of the slave device, receiving data from the master and sending data back.

Shift register:



Each time the transmitting and receiving is done,character count decrements and while reaching the end it increments and stops indicating the serial communication has completed.

Here,1 clock cycle of serial clock is equal to 6 clock cycles of wishbone clock

Tx_clk and Rx_clk will select CPOL_0 and CPOL_1

Internal variable tx_bit_pos is used for transmission from master to slave

At index[0]- value is 1, transmission at rising edge and receiving at falling edge done through mosi

Functionality of SPI Top Level module:

A top-level module typically represents the highest-level component in a digital design hierarchy, responsible for coordinating and managing various sub-modules and external interfaces.

The functionality of an SPI top-level module would generally include the following aspects:

SPI Configuration: The module allows the user to configure various parameters of the SPI communication, such as clock frequency, data order (MSB or LSB first), and clock polarity/phase.

Data Exchange: It manages the transmission and reception of data over the SPI bus, ensuring proper timing and synchronization between the master and slave devices.

Slave Select (SS) Management: The module often controls the activation and deactivation of slave devices by managing the slave select (SS) signals. This involves enabling the appropriate SS signal for the target slave device during data transfer.

Clock Generation: The module generates the clock signal (SCK) for the SPI communication, adhering to the specified clock settings.

Data Buffering: It typically includes data buffers or FIFOs to store data for transmission and reception.

Error Handling: The module may implement error detection and handling mechanisms, such as checking for framing errors, overrun errors, and other potential issues that can occur during SPI communication.

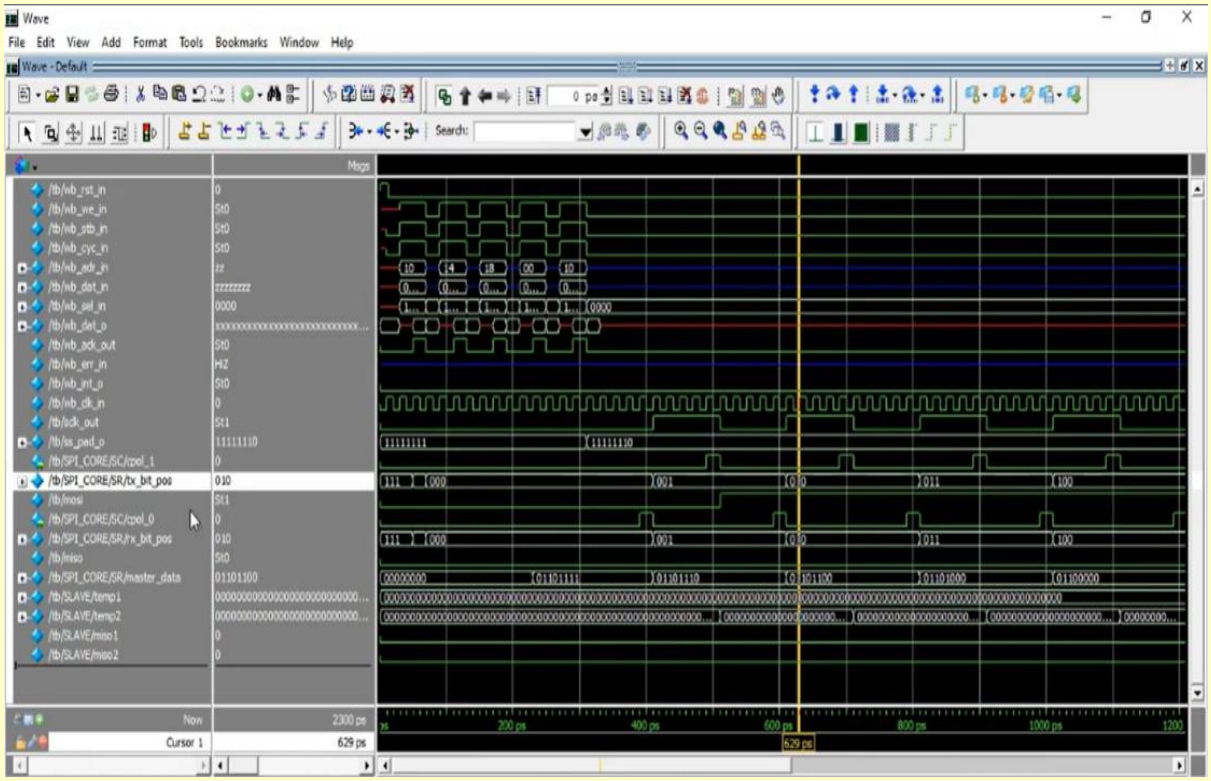
Status and Control Registers: It provides registers to allow the user to monitor the status of ongoing SPI transactions and configure various aspects of the communication.

Interrupt Handling: The module can generate interrupts to signal events like the completion of a data transfer, errors, or buffer status.

Clock Domain Synchronization: If the SPI module operates in a different clock domain from the rest of the system, it may include logic for clock domain crossing to ensure proper data transfer.

Interface to the Microcontroller or CPU: The SPI top-level module typically interfaces with the microcontroller or CPU to send and receive data, control the SPI communication, and manage the configuration.

Top level module:



We add CPOL_0 and CPOL_1 from clock generator

We add tx_bit_pos and rx_bit_pos from shift register

We add temp1,temp2,miso1 and miso2 from slave

Here 4 characters are transmitted in each so 4 serial clocks are generated

After first rising edge index[0] has become 0 which indicates receiving has completed

After first falling edge index[1] has become 1 which indicates transmitting has completed

At falling edge MOSI becomes high

At rising edge MISO becomes high

While reaching the end of serial bit transmission, the last output of shift register will go high and stops indicating the serial communication has completed.

When tx_neg =1 and rx_neg=0, transmission will happen on falling edge and receiving will happen on rising edge

Functionality of SPI Sub blocks:

SPI typically consists of several "sub-blocks" or components that work together to facilitate data transfer. These sub-blocks include:

Master and Slave Devices: In an SPI communication setup, there are typically one or more master devices and one or more slave devices. The master device initiates and controls data transfers, while slave devices respond to commands and send or receive data.

Serial Data Lines (MISO and MOSI):

MISO (Master In Slave Out): This is the data line through which the slave sends data to the master.

MOSI (Master Out Slave In): This is the data line through which the master sends data to the slave.

Serial Clock (SCK): The clock signal generated by the master device that synchronizes data transmission. It dictates the timing of data bits being transferred.

Chip Select (CS/SS): A control line that the master uses to select a specific slave device with which it wants to communicate. When the CS/SS line is asserted (pulled low), it indicates which slave is active for the data transfer.

Data Frame Format: The data frame format specifies how the data is organized for transmission. This includes parameters like the number of bits per word, the data format (e.g., MSB or LSB first), and whether the data is transmitted in a specific order (e.g., CPHA and CPOL settings).

Clock Phase (CPHA): Determines the edge of the clock signal at which data is sampled. It can be on the leading or trailing edge of the clock.

Clock Polarity (CPOL): Defines the idle state of the clock signal. It can be high or low when not actively transferring data.

Data Buffer/Register: The data buffer or register in the master and slave devices stores the data to be transmitted or received. Data is shifted in and out of these buffers during communication.

Shift Register: A shift register is used to serialize and deserialize data, converting between parallel and serial data formats.

State Machine: The SPI communication process typically involves a state machine or controller that manages the flow of data between the master and slave devices. This controller ensures proper synchronization and data handling.

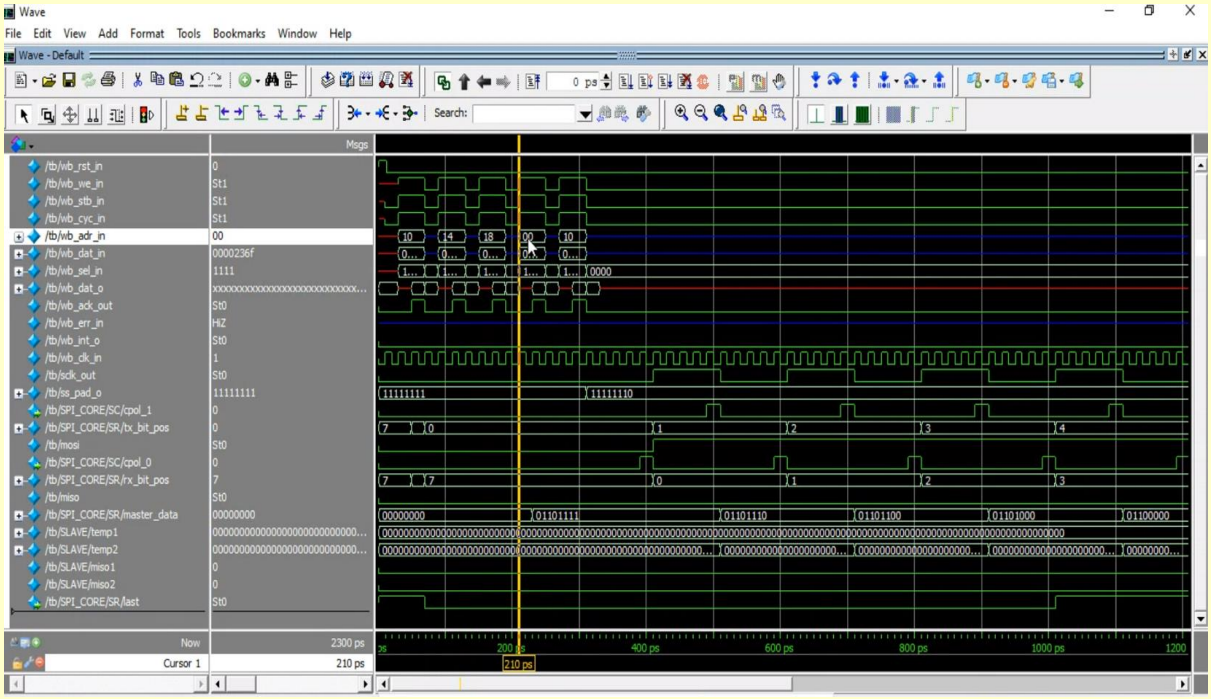
Control Logic: The control logic in the master device manages the overall operation of the SPI interface, including selecting slave

devices, initiating data transfers, and handling various configuration settings.

Interrupts and Flags: SPI often provides interrupt capabilities and status flags that allow the microcontroller or device to detect when data transfer is complete or when errors occur.

Clock Generation: In the master device, there is a clock generation unit responsible for generating the clock signal (SCK) at the desired frequency and phase.

Sub level



We have address 10 for control register initially

We have divider register,slave select register

There are four transmitting registers and we have selected tx0 register and loaded with the value 236f

By changing the divider value, the clock division is going to change

Rising edge is used for transmission

Falling edge is used for receiving

Master data content is same as tx0 content

From the shift register,we add transmitting bit position,receiving bit position and master data

From the clock generator, we add CPOL_0

Inference & Conclusion:

We have understood the working of various registers of SPI protocol and waveforms that can be generated using top level and sub level modules

We have successfully demonstrated the waveforms using different type of registers based on their working

The project clearly demonstrates our approach to SPI design which has the potential to revolutionize the technology industry, making it more sustainable, accessible, and efficient while positively affecting society and the environment.

S SAI CHARAN

21BEC0225

VIT VELLORE