PROJECT REPORT ON

“SERIAL PERIPHERAL INTERFACE DESIGN”

Submitted by :

ASIM ANAND
21BEC1519

INTRODUCTION

Serial Peripheral Interface (SPI) is a communication link between microcontrollers and peripheral ICs such as ADCs, DACs, SRAM, sensors etc.
It is a synchronous protocol that allows a master device to initiate communication with a slave device, with data being exchanged them in serial, fully duplex manner.
It is basically a way electronics devices communicate with each other.
In SPI protocol, there can be only one master but many slave devices.

Many IC manufacturers produce components that are compatible with SPI and Microwire/plus.
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

PORTS

The devices associated with SPI protocol, they connect using multiple wires.

i. MOSI - (Master Out Slave In)
Signal output line
Device A uses to send data to device B
1 Bit

ii. MISO - (Master In Slave Out)
Signal input line
Device B uses to send data back to device A
1 Bit

iii. SCLK - (Serial Clock)
Output line
1 Bit

iv. SS - (Slave Select)
Slave select output signals
8 Bits

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

Other than these, Wishbone interface has 12 input/output ports. All output WISHBONE signals are registered and driven on the rising edge of 'Master Clock'. All input WISHBONE signals are latched on the rising edge of 'Master Clock'.

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

REGISTERS

1. CORE REGISTERS

It has 11 core registers, in which 4 Data receive registers, 4 Data transmit registers, Control and status register, clock divider register and a slave select register.
All registers are 32-bit wide and operational and except data receive register all are both read and write access.

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

2. DATA RECEIVE REGISTER

The data received registers hold the value of received data of the last executed transfer. These are read-only registers.

Bit #	31:0
Access	R
Name	Rx

Table 4: Data Receive register

3. DATA TRANSMIT REGISTER

The data received registers hold the data to be transmitted in the next transfer.

Bit #	31:0
Access	R/W
Name	Tx

Table 5: Data Transmit register

4. CONTROL AND STATUS REGISTER

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

5. DIVIDER REGISTER

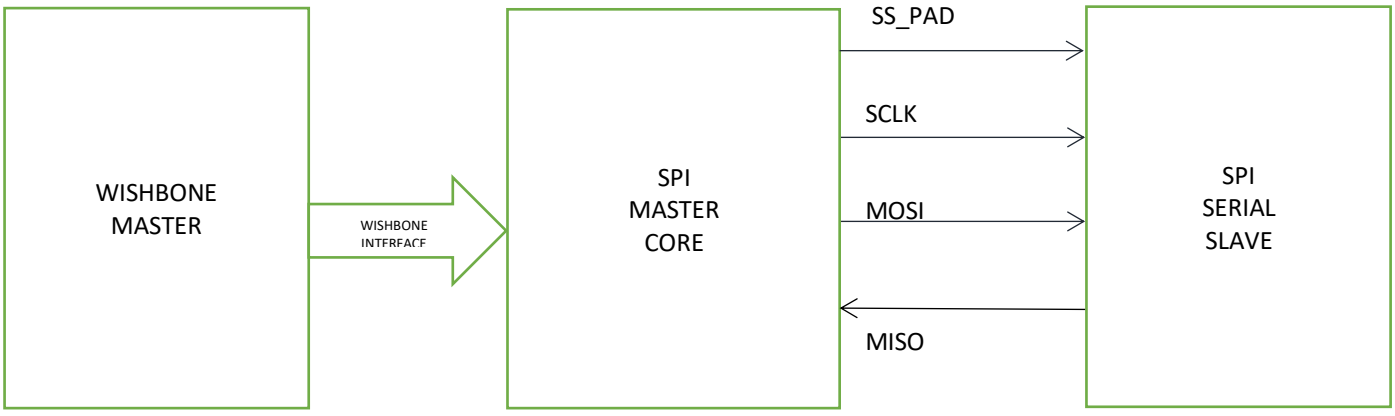
The value in this field is the frequency divider of the system clock to generate the serial clock on the output serial clock. It has 32 bits in total in which 16 bits are already configured and 16 bits are reserved. The desired frequency is obtained according to the following equation:-

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

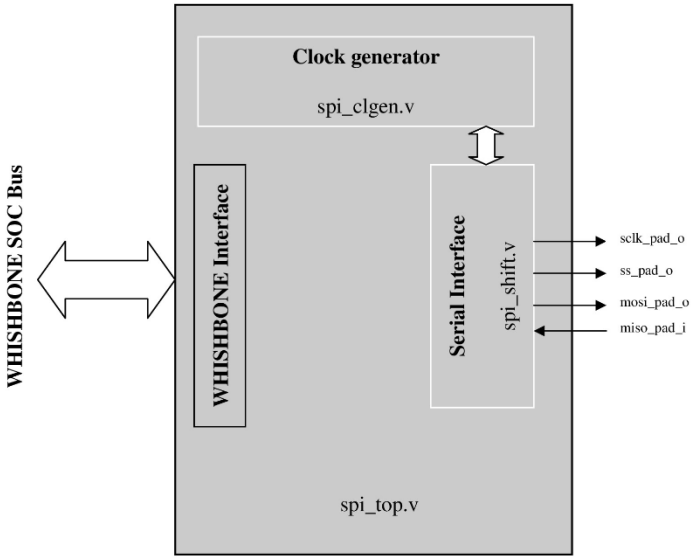
Bit #	31:16	15:0
Access	R	R/W
Name	Reserved	DIVIDER

OPERATIONS

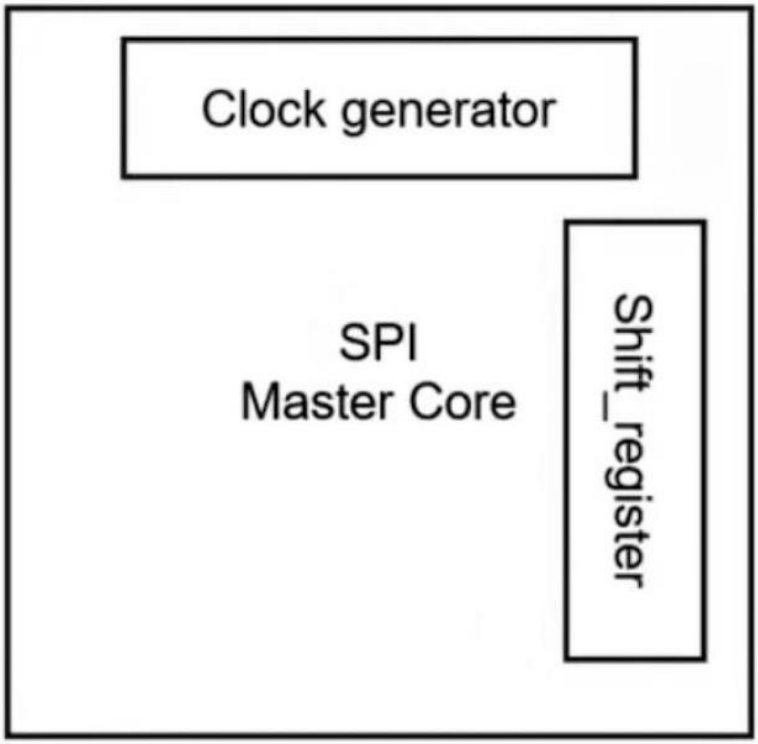
During an SPI transfer, data is simultaneously transmitted and received. The SPI core suppose has a five 32-bit registers through WISHBONE interface. A serial clock line synchronizes shifting and sampling of information on the two serial data lines. Data is transferred between them using the wishbone interface in a synchronized and organized manner. This allows for reliable and efficient communication, making it easier for the host and the slave device to exchange data and control signals seamlessly.



ARCHITECTURE



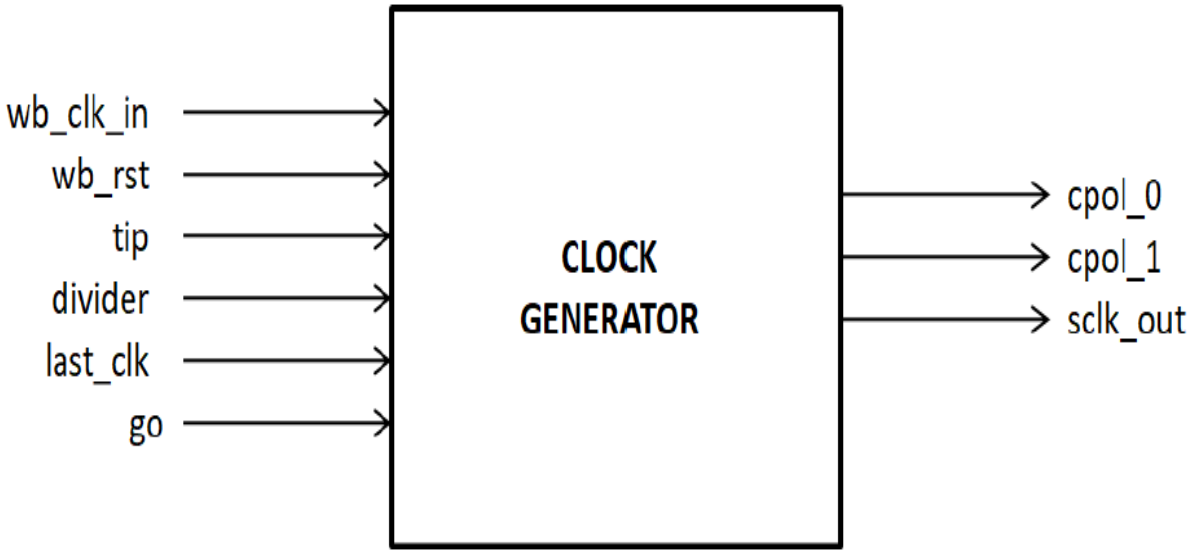
The SPI master core consists a combination of clock generator and shift register blocks.



CLOCK GENERATOR BLOCK

The clock generator block is basically responsible for generating the clock signal used to synchronize data transfer between the master and slave devices. The clock signal's frequency and timing are essential for communication in SPI.

BLOCK DIAGRAM



INPUTS

- wb_clk_in : Input clock signal
- wb_rst : Reset Signal
- go : Signal indicating whether the start of the transfer
- tip : Signal indication the last clock edge of the transfer
- divider : Input value determining the clock divider for generating the SPI clock.

OUTPUTS

- sclk_out : Output SPI clock signal
- cpol_0 : Output signal indicating the pulse marking the positive edge of the SPI clock.
- cpol_1 : Output signal indicating the pulse marking the negative edge of the SPI clock.

FUNCTION

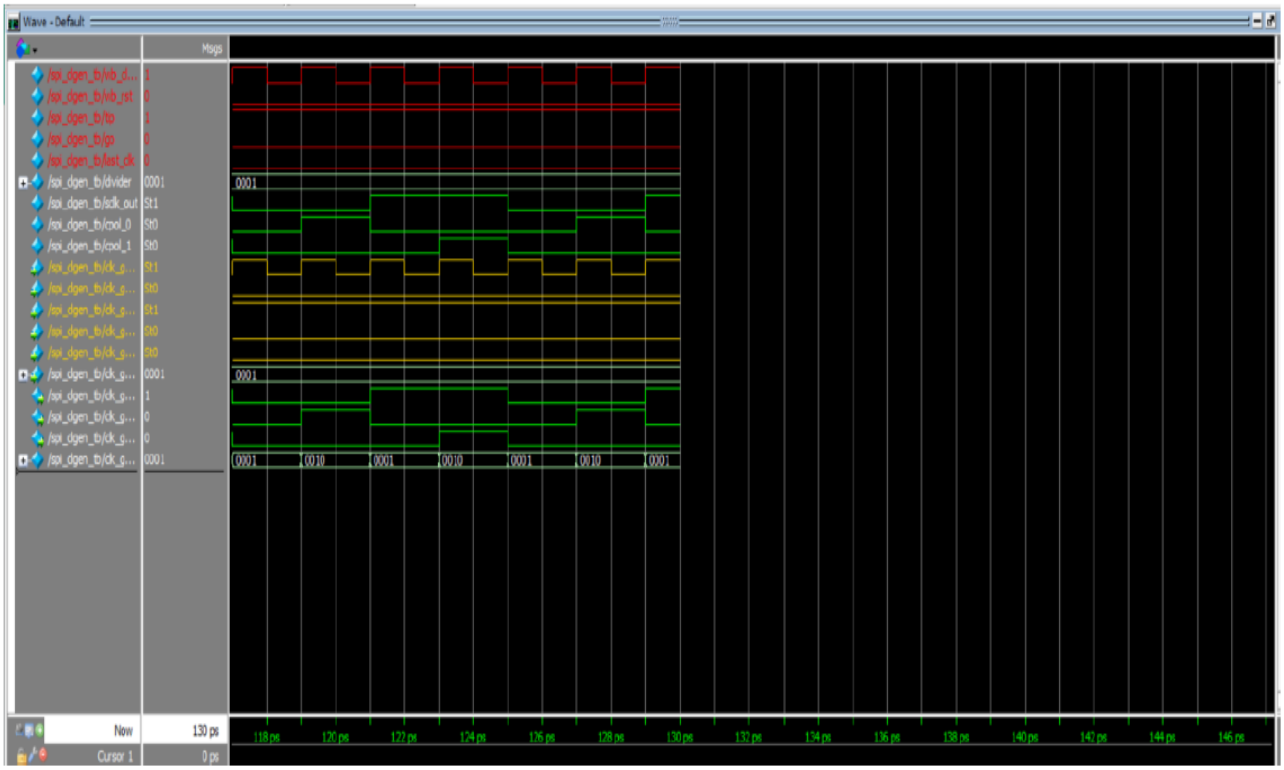
The clock signal generates two clock signals, namely rx_clk and tx_clk. These clocks basically regulates the receiving and transmitting processes of the SPI shift register. The generation of these clock signals depend on several input signals, including rx_negedge, tx_negedge, last, sclk, cpol_0, and cpol_1.
To determine the output clock frequency on sclk_pad_o, a frequency divider field is set using the wb_clk_I system clock as a reference. The desired frequency is computed according to this specific formula.

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

The rx_clk signal is formed by combining the rx_negedge input signal with the logical OR operation involving last and sclk. This rx_clk signal serves as the clock enable for receiving data. On the other hand, the tx_clk signal is generated by combining the tx_negedge input signal with the logical AND operation of last and the selected clock polarity (either cpol_0 or cpol_1). The tx_xlk is used as the clock enable for transmitting data.

The clock generator block is designed to ensure that the clock signals are generated correctly based on the specified clock polarities and the transfer status (last). This, in turn, guarantees the appropriate synchronization and precise timing control required for the SPI shift register module during both data reception and transmission.

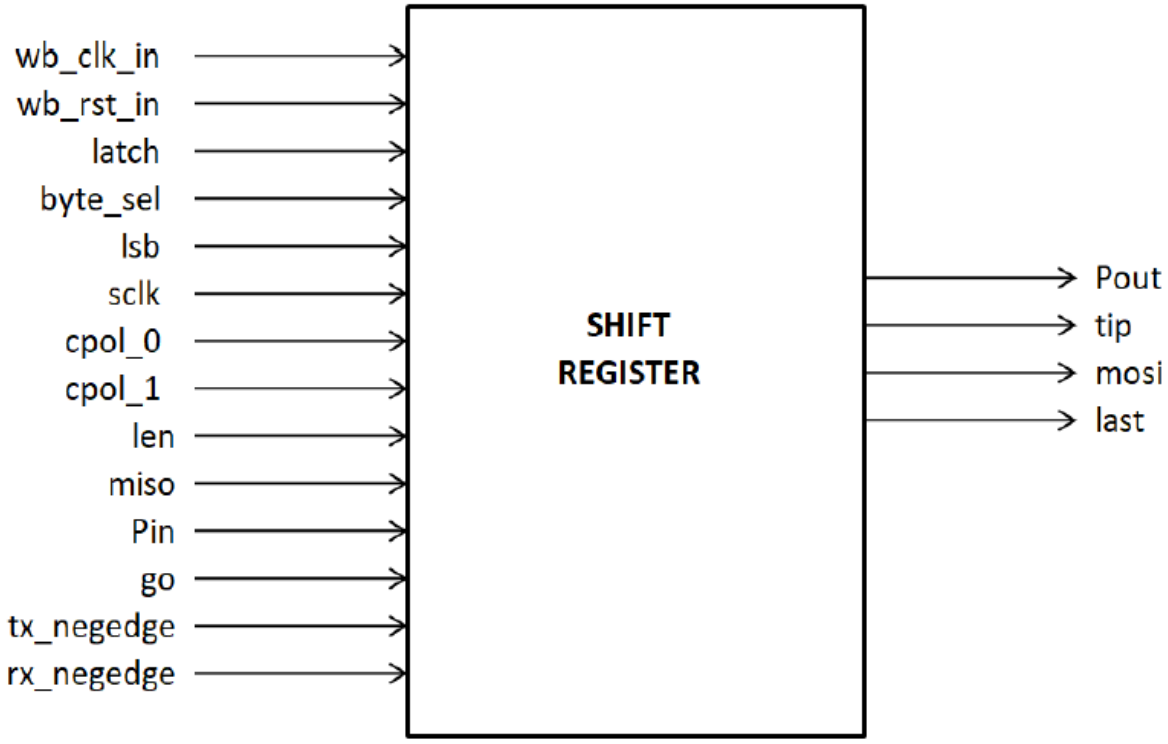
CLOCK GENERATOR MODULE WAVEFORM



SHIFT REGISTER BLOCK

The shift register block handles the serial data transfer between the SPI master and the SPI slave devices. It is responsible for the conversion of parallel data to a serial format for transmission and the conversion of incoming serial data back to parallel format.

BLOCK DIAGRAM



INPUTS

- rx_negedge : Input signal, indicates the negative edge of the receive clock.
- tx_negedge : Input signal, indicates the neagtive edge of the transmit clock.
- byte_sel : 4-bit input signal used for byte selection
- latch : Input signal used for latch control
- len: 32-bit input signal representing the length of the character.
- p_in: 32-bit input signal for the incoming data.
- wb_clk_in: Input clock signal (system clock).
- wb_rst: Input reset signal.
- go: Input signal to initiate the transfer.
- miso: Input signal for the master input slave output.
- lsb: Input signal indicating the least significant bit.
- cpol_0: Input signal for the pulse marking the positive edge of thesclk_out.
- cpol_1: Input signal for the pulse marking the negative edge of the sclk_out.

OUTPUTS

- p_out: Output signal representing the shifted data from the shift register. It is
- SPI_MAX_CHAR bits wide.
- Last: Output signal indicating whether the last character is being transmitted or received.
- mosi: Output signal representing the serial output data from the shift
- register.
- tip: Output signal indicating if a transfer is in progress or not.
- rx_clk: Output signal representing the receive clock enable.
- tx_clk: Output signal representing the transmit clock enable.

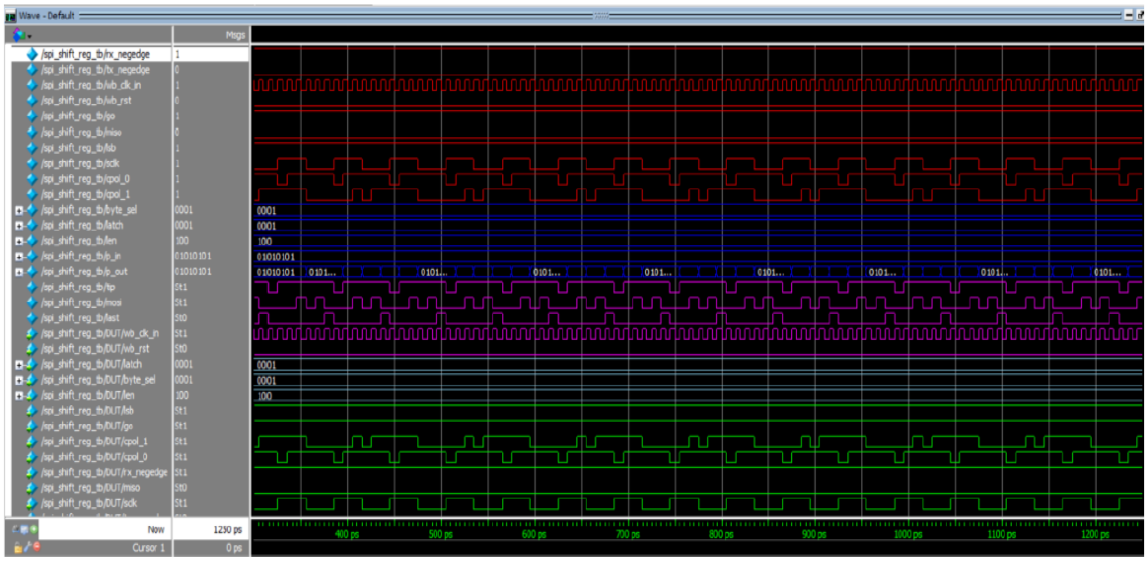
INTERNAL REGISTERS

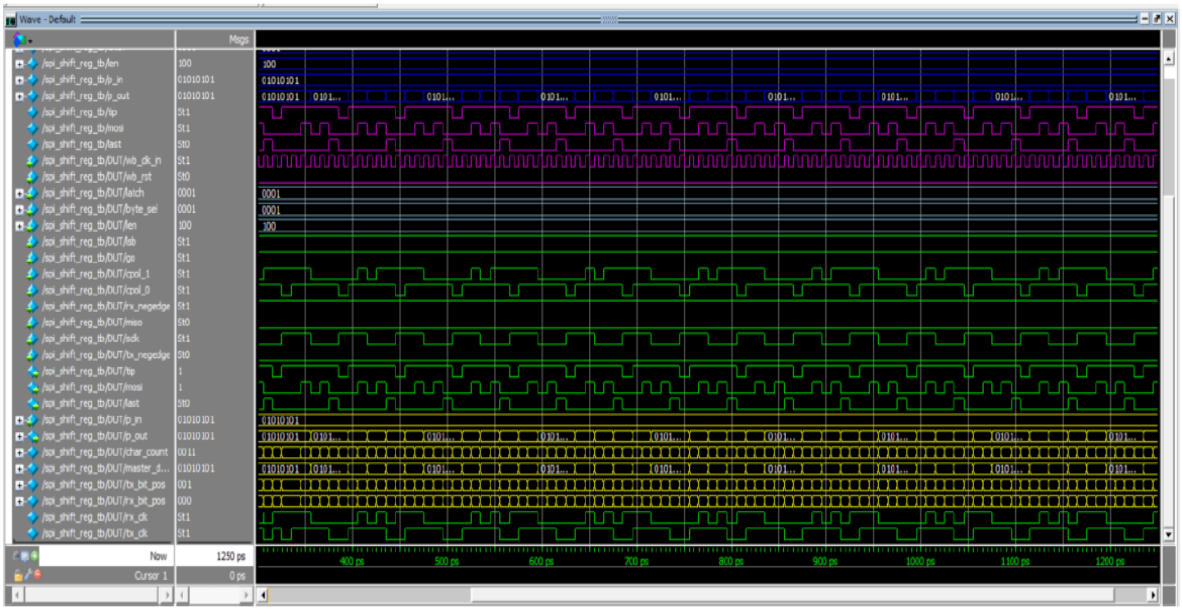
- char_count: Register used for counting the number of bits in a character.
- master_data: Register representing the shift register holding the data.
- tx_bit_pos: Register indicating the next bit position for transmission.
- rx_bit_pos: Register indicating the next bit position for reception.

FUNCTION

The shift register component receives a range of inputs, including clock signals, data selection, data input, and control signals. It incorporates internal registers for data storage, positions for shifting data bits, and a counter to keep track of character bits.
It computes the status of the ongoing data transfer and identifies the final character in the process. Additionally, it determines the serial output by considering the clock signals and the current bit location.
This module manages the computation of bit positions for both sending and receiving data according to the configuration. The result is the data that has been shifted within the shift register. Furthermore, it offers support for data latching.

SHIFT REGISTER MODULE WAVEFORMS





SPI TOP BLOCK

It interfaces with a Wishbone bus and provides communication with a master device using the SPI protocol.

INPUTS

- wb_clk_in: Clock input for the Wishbone bus.
- wb_rst_in: Reset input for the Wishbone bus.
- wb_adr_in: Address input for the Wishbone bus.
- wb_dat_in: Data input for the Wishbone bus.
- wb_sel_in: Select input for the Wishbone bus.
- wb_we_in: Write enable input for the Wishbone bus.
- wb_stb_in: Strobe input for the Wishbone bus.
- wb_cyc_in: Cycle input for the Wishbone bus.
- miso: Master Input Slave Output (MISO) signal

OUTPUTS

- wb_dat_o: Data output for the Wishbone bus.
- wb_ack_out: Acknowledge output for the Wishbone bus.
- wb_int_o: Interrupt output for the Wishbone bus.
- sclk_out: Clock output for the SPI interface.
- mosi: Master Output Slave Input (MOSI) signal.
- ss_pad_o: Slave select output for the SPI interface

The code instantiates two sub-modules: "spi_clgen" and "spi_shift_reg".
- "spi_clgen" generates the SPI clock signal based on the input clock and control signals.
- "spi_shift_reg" handles the data shifting and synchronization for SPI communication.

It decodes addresses to read from specific registers and provides output data and acknowledgment signals. It also supports interrupt generation and slave device selection. Overall, it serves as a complete SPI controller for data transfer between a master and multiple slave devices.

WISHBONE MASTER MODULE INTERFACE

SPI Master acts as a slave to Wishbone Interface. Wishbone master communicates with the host. Bus signals of Wishbone master are as follows:

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

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.
Module Declaration: Defines the module wishbone_master with input and output ports.

INPUTS

- clk_in: The clock input for the module.
- rst_in: The reset input for the module.
- ack_in: The acknowledgment input from the slave device.
- err_in: The error input from the slave device.
- dat_in: The data input for write operations.

OUTPUTS

- adr_o: The address output for the Wishbone master.
- cyc_o: The cycle output for initiating a Wishbone transaction.
- stb_o: The strobe output for indicating the start of a Wishbone transaction.
- we_o: The write enable output for write operations.
- dat_o: The data output for read operations.
- sel_o: The select output for selecting the active byte lanes.

FUNCTION

The module contains two significant tasks: one called "initialize" for setting up internal signals, and another called "single_write" designed to perform single write operations by configuring various internal signals based on provided inputs. The "single_write" task also waits for the acknowledgment signal (ack_in) to become inactive before resetting these internal signals.
Sequential blocks, triggered by the positive edge of the clock signal (clk_in), are used to determine the values of the output signals. The rst_in signal is employed to manage reset conditions for the cyc_o and stb_o signals.

In summary, the wishbone_master module acts as a controller for communicating with a Wishbone slave device, offering the necessary control and data signals for read and write operations. The "initialize" and "single_write" tasks, along with internal signal management, are fundamental aspects of this module's functionality.

SLAVE MODULE INTERFACE

The master can choose which slave it wants to talk to by setting the slave's SS line to a low voltage level. In the idle, non-transmitting state, the slave select line is kept at a high voltage level. Multiple SS pins may be available on the master, which allows for multiple slaves to be wired in parallel. If only one SS pin is present, multiple slaves can be wired to the master by daisy-chaining.
SPI Slave module here just behaves as a "slave" to test for the code results. It is not an actual slave.

Block Diagram

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

INPUTS

- sclk: The serial clock input for synchronization.
- mosi: The input signal representing the data received from the SPI master.
- ss_pad_o: A multi-bit output signal representing the chip select lines for the SPI slave.

OUTPUTS

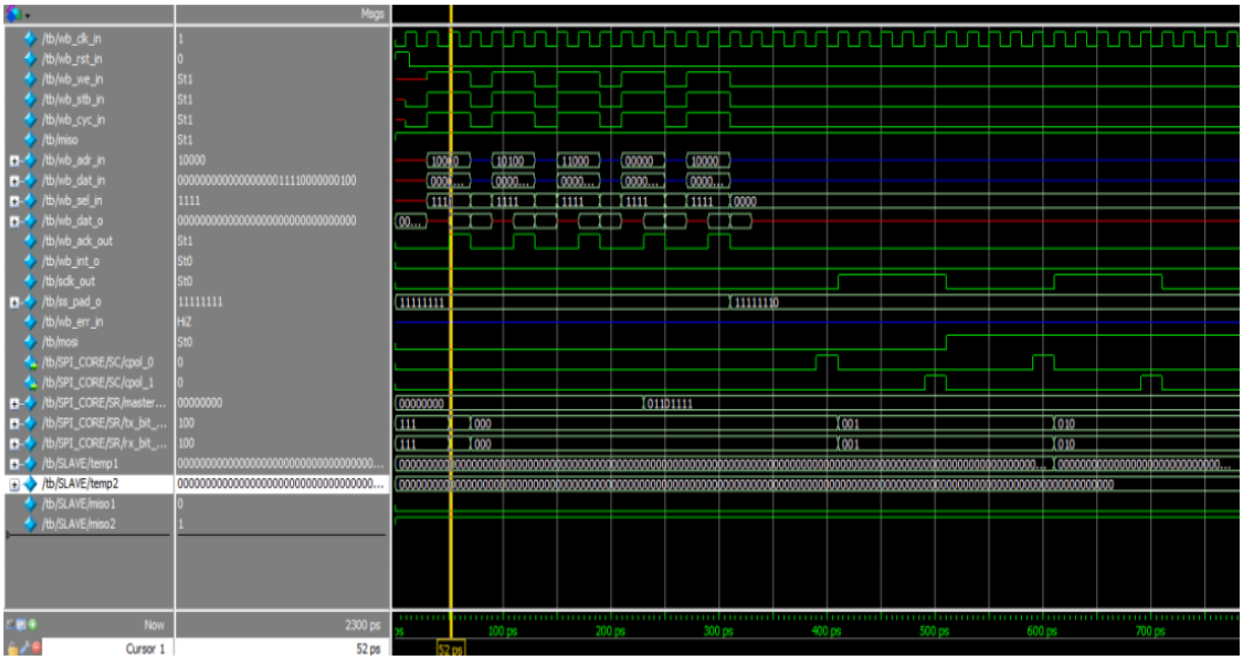
- miso: The output signal representing the data to be transmitted from the SPI slave.

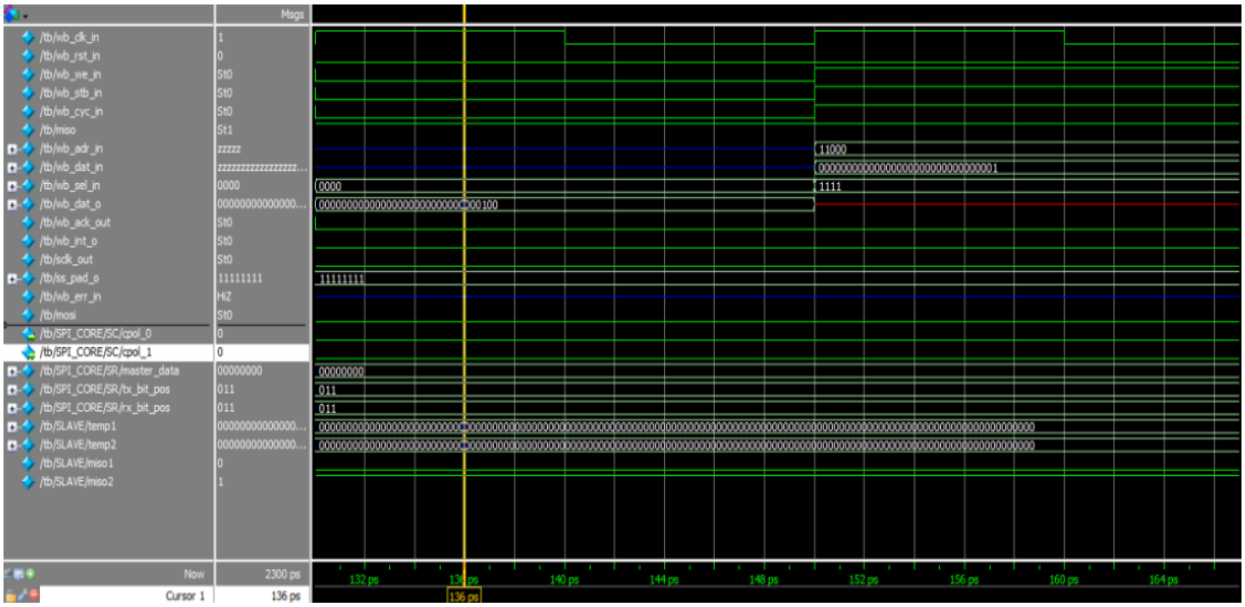
INTERNAL SIGNALS

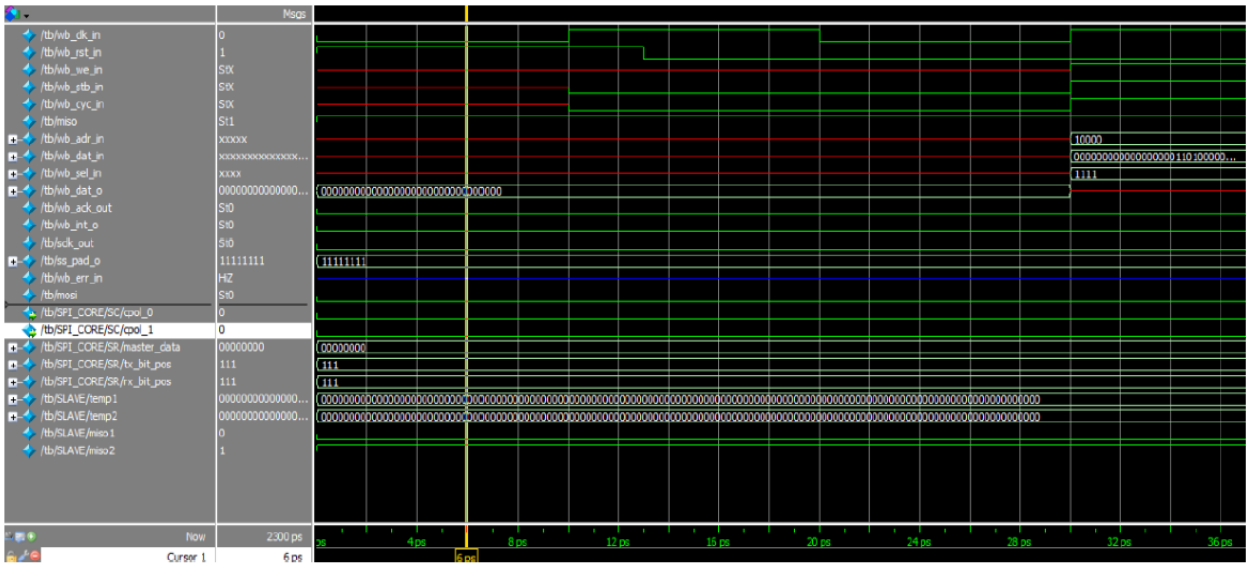
- rx_slave: A register indicating whether the SPI slave is in receive mode.
- tx_slave: A register indicating whether the SPI slave is in transmit mode.
- temp1: A 128-bit register used for temporary storage of received data.
- temp2: A 128-bit register used for temporary storage of received data.

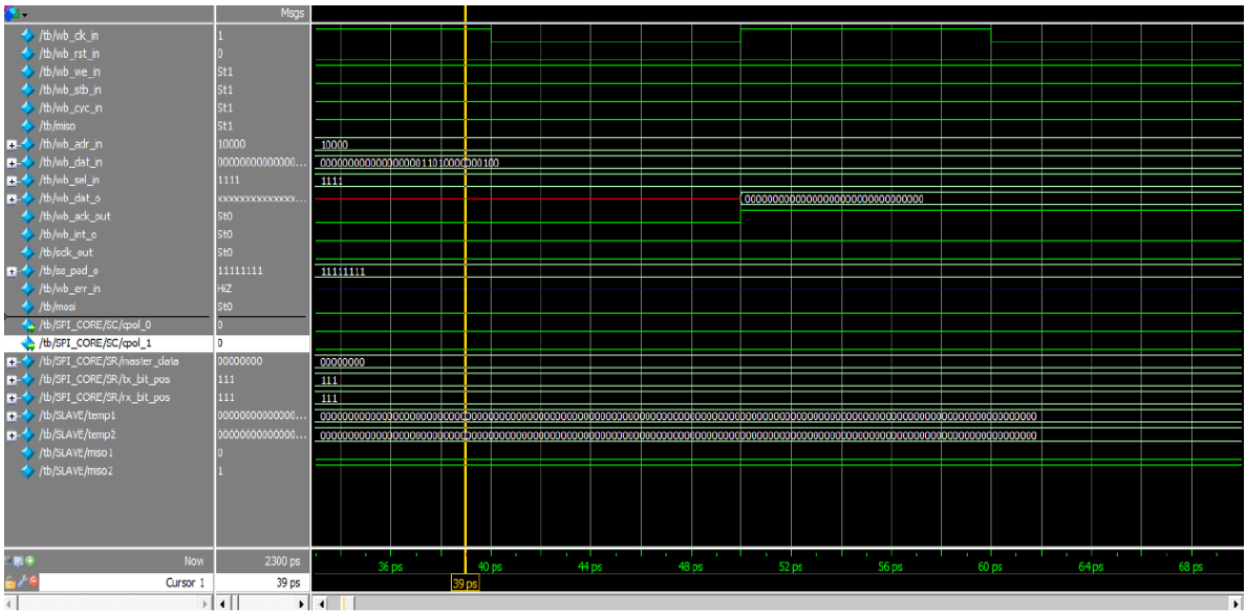
On the posedge of sclk, if the chip select lines are not all high (ss_pad_o != 8'b1111111) and the slave is in receive mode (rx_slave is high) and not in transmit mode (tx_slave is low), the received data mosi is concatenated with the existing data in temp1.
On the negedge of sclk, if the slave is in receive mode (rx_slave is high) and not in transmit mode (tx_slave is low), the value in temp1 is assigned to miso1.
On the posedge of sclk, if the slave is in receive mode (rx_slave is high) and not in transmit mode (tx_slave is low), the value in temp2 is assigned to miso2.
The miso signal is assigned as the logical OR of miso1 and miso2, which represents the data to be transmitted from the SPI slave.
Overall, the spi_slave module receives data from the SPI master on the mosi line, processes and stores it in temp1 and temp2, and provides the output miso for transmitting data back to the SPI master. The chip select lines (ss_pad_o) are used to control the slave operation.

WAVEFORMS






Msgs
/tb/wb_clk_in
/tb/wb_rst_in
/tb/wb_we_in
/tb/wb_stb_in
/tb/wb_cyc_in
/tb/miso
/tb/wb_adr_in
/tb/wb_dat_in
/tb/wb_sel_in
/tb/wb_dat_o
/tb/wb_ack_out
/tb/wb_int_o
/tb/sck_out
/tb/ss_pad_o
/tb/wb_err_in
/tb/mosi
/tb/SPI_CORE/SC/cpol_0
/tb/SPI_CORE/SC/cpol_1
/tb/SPI_CORE/SR/master_data
/tb/SPI_CORE/SR/tx_bit_pos
/tb/SPI_CORE/SR/rx_bit_pos
/tb/SLAVE/temp1
/tb/SLAVE/temp2
/tb/SLAVE/miso1
/tb/SLAVE/miso2
Now
Cursor 1
2300 ps
6 ps
4 ps
8 ps
12 ps
16 ps
20 ps
24 ps
28 ps
32 ps
36 ps


Msgs
/tb/wb_clk_in
/tb/wb_rst_in
/tb/wb_we_in
/tb/wb_stb_in
/tb/wb_cyc_in
/tb/miso
/tb/wb_adr_in
/tb/wb_dat_in
/tb/wb_sel_in
/tb/wb_dat_o
/tb/wb_ack_out
/tb/wb_int_o
/tb/sck_out
/tb/ss_pad_o
/tb/wb_err_in
/tb/mosi
/tb/SPI_CORE/SC/cpol_0
/tb/SPI_CORE/SC/cpol_1
/tb/SPI_CORE/SR/master_data
/tb/SPI_CORE/SR/tx_bit_pos
/tb/SPI_CORE/SR/rx_bit_pos
/tb/SLAVE/temp1
/tb/SLAVE/temp2
/tb/SLAVE/miso1
/tb/SLAVE/miso2
Now
Cursor 1
2300 ps
39 ps
36 ps
40 ps
44 ps
48 ps
52 ps
56 ps
60 ps
64 ps
68 ps