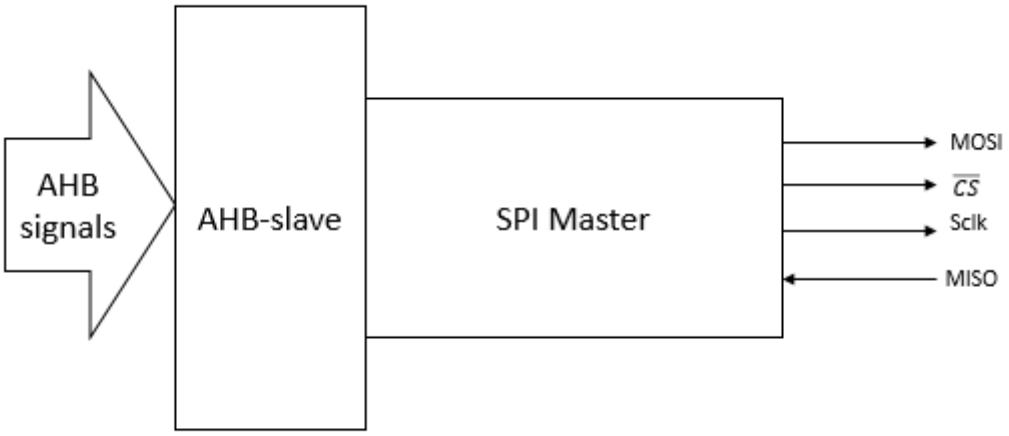
Features

In this code the hardware is designed which acts as a master device on a SPI bus and slave device on AHB-Lite bus. This design acts as a peripheral device with ARM Cortex-M0 processor on one side and sensor ADXL362 accelerometer on the other side. This SPI master receives the AHB signals and converts it into the SPI signals which will help to communicate the SPI devices. Along with transmission and reception of the serial signals, it also generates the Sclk signal. This Sclk signal provides the frequency in which SPI device will receive the signal.

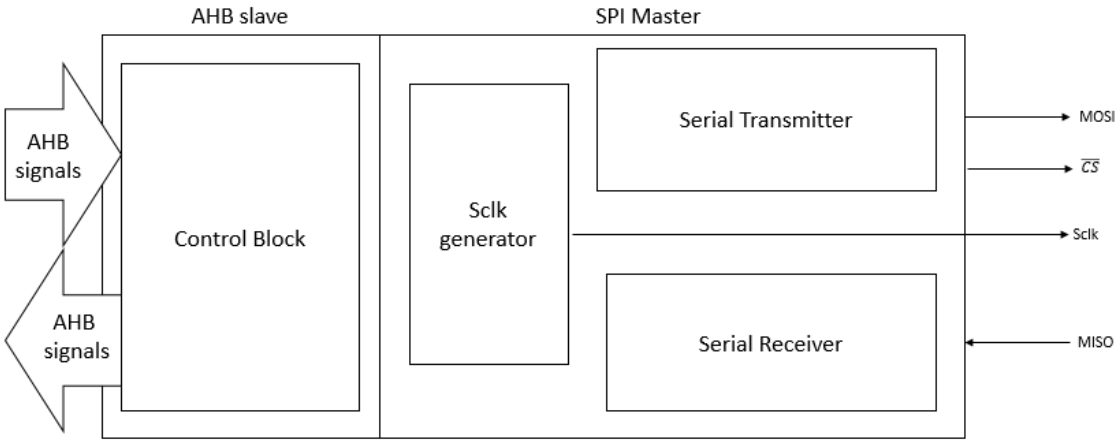
High-Level Design

In this assignment there are two task,

1) **AHB Slave**
2) **SPI master**



The AHB slave consist of control block which receives the AHB signal and sends out the signal back to processor through AHB lite bus. The SPI master lock consist of three major block Serial transmitter block, Serial receiver block and Sclk generator. The detailed block diagram is shown below:



From above block diagram, it is evident that for this hardware design majorly 4 blocks need to be designed.

1) **Control block**:-
   This block act as an interface between the AHB lite bus and SPI master. It will collect the data from the bus and convert the data into the suitable format for the SPI master also it will convert the data received from the SPI device so that it can be transmitted over the AHB bus.

2) **Sclk Generator**:-
   This block will convert the Hclk (AHB signal) which is 50 MHz the operating frequency of AHB lite bus into the operating frequency of the SPI device within the range of 1 to 8 MHz.

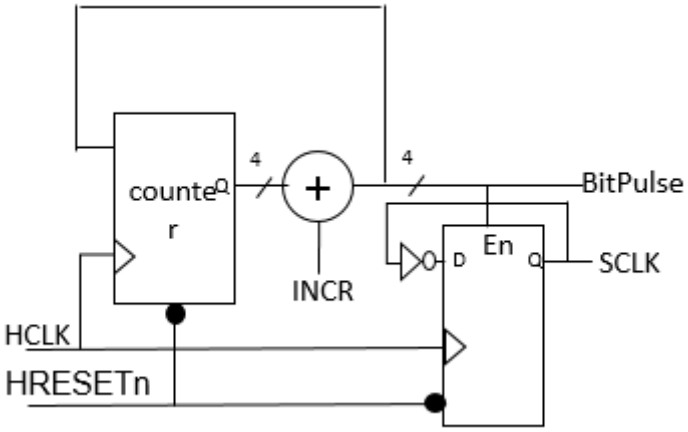
3) **Serial Transmitter** :-
   This block will receive the data from the control block and it will send the 8 bits serially (MSB first) to SPI device through signal MOSI.

4) **Serial Receiver**:-
   This block will receive the data from SPI device through MISO signal. It will collect the 8 bit data transmitted by the SPI device and store it into the 8 bit register.

Block Design

RTL diagram for

1. **Sclk Generator**



The Sclk generator is used to generate Bitpulse and SCLK. The detail description of the each signal generation is explained below:

i. **BitPulse Genration**:-

The bit pulse is generated using a counter and adder. For each HCLK cycle the value of the increment is added to register. This process will continue till the value overflows. After this the MSB of adder output will be 1 for one clock cycle. This MSB will be the bit pulse signal.

For k bit register the value will over flow after $\frac{HCLK}{2^k}$ cycle. In this design the value 4 bit register is selected with the input HCLK frequency of 50 MHz .Therefore the output frequency will

$$\frac{50}{2^4} = \frac{50}{16} = 3.125MHz$$

This is the frequency of the BitPulse signal. The value in the INCR is set as integer 2. This will decide the rate of overflow. For INCR value 2 the number of bits transmitted are

$$2 \times 3.12 = 6.25Mbps$$

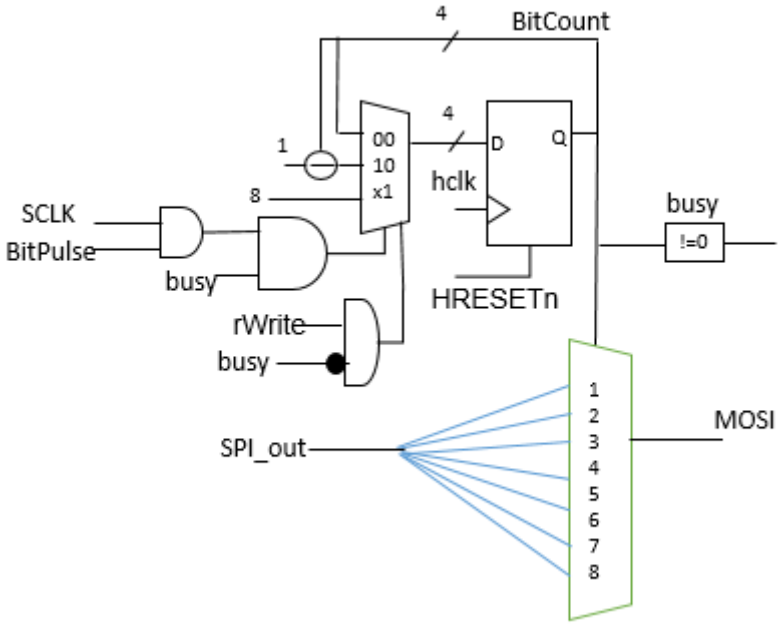
ii. **SCLK Generation**:-

The Bitpulse signal is used as enable signal for the shift register. For each high Bitpulse signal the shift register will get enable. The output of the signal is inverted with the not gate at each enable operation. For high SCLK one Bitpulse is required and for low SCLK one Bitpulse required. So the output will be twice the frequency of the Bitpulse.

$$2 \times 3.125MHz = 6.25MHz$$

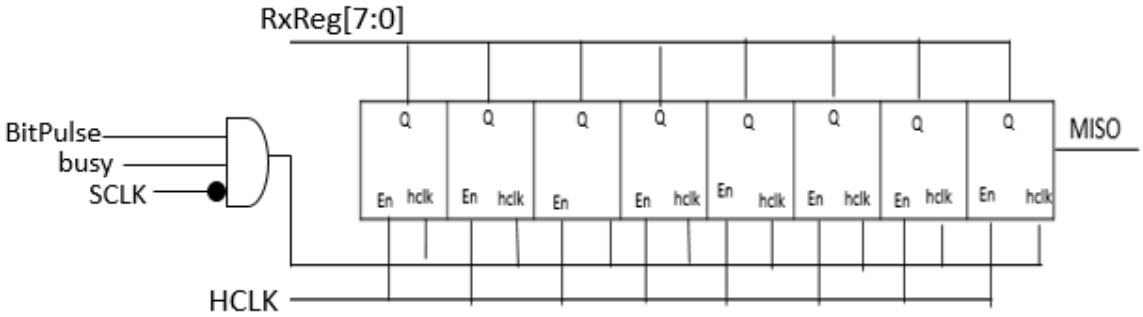
The SCLK frequency is $6.25MHz$ which is within the range of SPI device (1 to 8 MHz).

2. Serial Transmitter



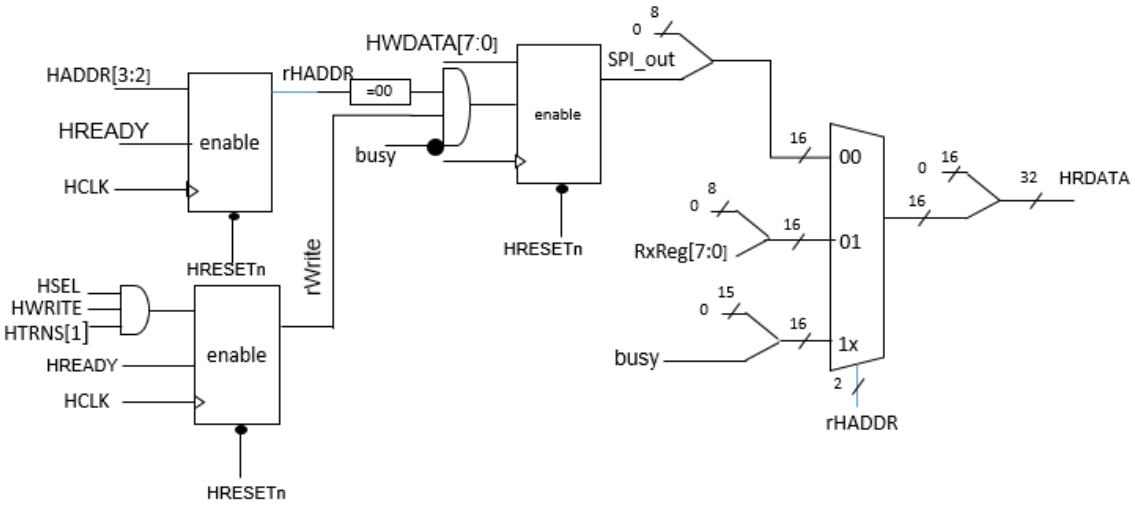
The Serial is transmitted on the positive edge of the SCLK through MOSI. In order the achieve this , the multiplexer with input of feedback, subtractor for count -1 and value that need to be load in this case it is 8 is used. This multiplex is controlled by 2 bits. The lower bit is the output of the AND gate with rWrite signal which will be obtained from control block and negation of the busy signal. The higher bit is the output of the AND gate with SCLK, BitPulse and busy signal. When the data is available to write rWrite will become 1 and the 8 value is loaded through the Multiplexer. This will make busy signal high and after that for every positive edge of the SCLK and Bitpulse the value in the register is decremented by 1. This decremented value will select the bit of the 8 bit data that need to be transmitted through MOSI. This data is available as an input for 8:1 Multiplexer. As per the count value it will transmit the data. As the MSB bit need to be transmitted first the value of the count will start from 8 and transmit till 1. For 0 the output will be zero.

3. **Serial Receiver**:-



In the serial Receiver design, 8 serial in parallel out register are used. As the data from MISO signal is serial data the serial in shift register are used. These shift register are operated on HCLK cycle and the enable signal. As the data is received on the negative edge of the clock cycle, the enable signal is the output of the AND gate with Bitpulse, busy and negation of SCLK signal. The output of 8 register is stored in RxReg which 8 bit and sent to control block for HRDATA.

4. **Control Block**:-



As per the given requirement of the BUS interface

Register	Relative Address	Access	HADDR (Hexadecimal)	rHADDR (Binary)
TXdata	0x0	R/W	0	00
RXdata	0x4	R	4	01
Status	0x8	R	8	10

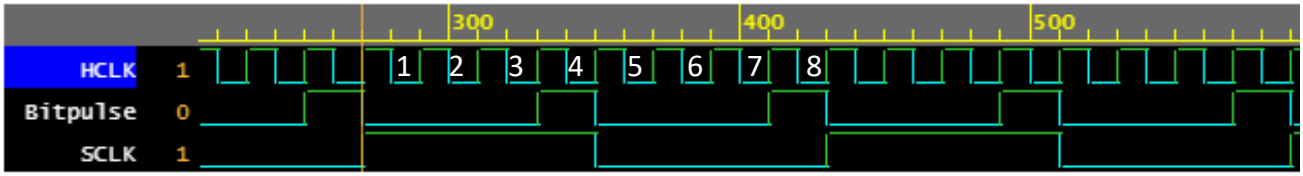
In the above table the register and their address are mentioned. To access particular register the required value in HADDR and rHADDR is given in the table.The control block mostly deals with the AHB signals. The HADDR is the 32 bit system adders Bus .In this 32 bits only 3rd and 2nd bit is used to control the multiplexer which transmits the data to the HRDATA. The first register enables on high HREADY signal and positive edge of HCLK and transmits 3rd and 2nd bit to the 2 bit signal called rHADDR. Second enable block will decide the operation as it is write operation or read operation. The signal HSEL is separate for every slave signal and will be enable only when the master wants to send data to particular slave. The HTRANS shows the transition state of the bus. The HWRITE signal is high for write operation and low for read operation .The combination of HWRITE, HSEL and HTRANS forms the rwrite signal. This rwrite signal is used to load the value of count in the transmission block.

For write operation
When the write command is sent through the AHB master the rWrite becomes high and the 3rd and 2nd bit of the HADDR will be 00 .The value of the rHADDR is compared with 00 and if it is same then this will enable the register to transmit the lower 8 bit of the HWDATA into SPI_out
For Read operation
The 3rd and 2nd bit of the HADDR is 01 this will select the second input from the multiplexer which is data in RxReg with 8 zeros. As we need to transmit 32 bit to the HRDATA the 24 zeros and 8 bit Rxreg value is transmitted.
For Status read operation

The 3rd and 2nd bit of the HADDR is 10 this will select the third input from the multiplexer which is busy with 15 zeros. As we need to transmit 32 bit to the HRDATA the 31 zeros and 1 bit busy signal value is transmitted.

Verification

1. **Verification of Sclk Generator block**

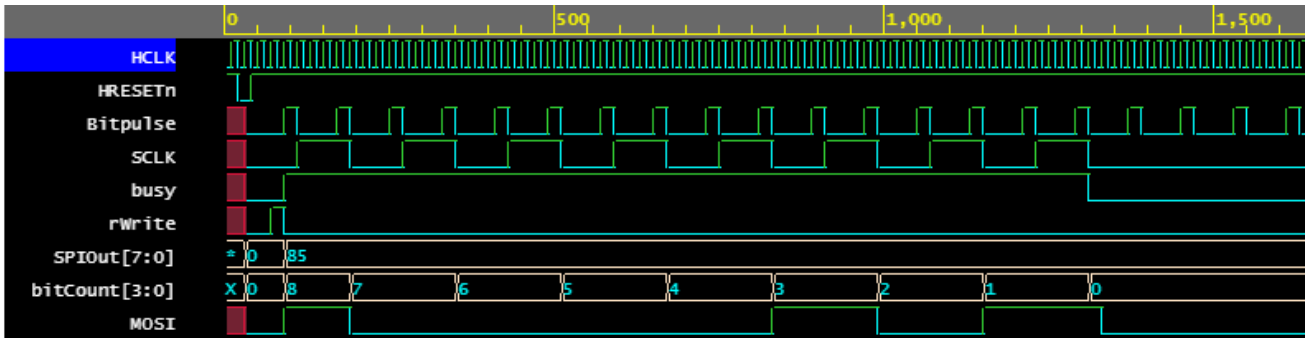


The HCLK is generated by inverting its value after every 10 nsec. It generates the cycle of 20 nsec which is of 50 MHz frequency. From the above waveform it can be clearly seen that a bitpulse is generated after every 4 HCLK cycles. This gives the Bitpulse frequency of 12.5 MHz. For every Bitpulse the value of the SCLK is inverted. For two Bitpulse cycle only one SCLK cycle is generated. For the generation of one SCLK cycle 8 cycles of HCLK and 2 cycles of Bitpulse is required. This gives the SCLK a frequency of

$$\frac{50}{8} = \mathbf{6.25}\boldsymbol{MHz}$$

This frequency is in the range of required SPI device range i.e. 1 to 8 MHz.

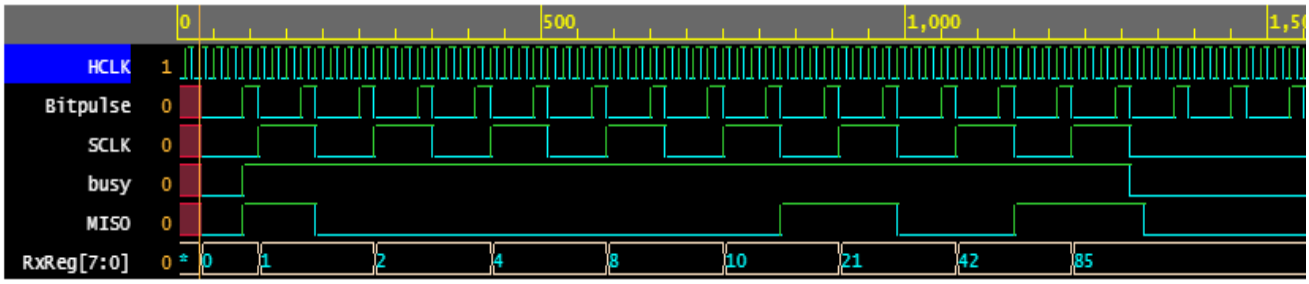
2. **Verification Serial Transmitter Block**



In the serial transmitter block the control signals are rwrite, busy and Bitpulse. When the all the three signals are high the value 8 is loaded in the bitcount. To test this functionality the value of the signal rwrite which is coming from control block is made high. Initially busy signal was low, the combination of low busy and high write loaded 8 value through multiplexer which bitcount signal as 8. This made busy signal high and it remained high

till the value of the bitcount signal becomes 0. The value of the signal bitcount selected the input of the multiplexer and gave the output on MOSI. Form the figure it can be seen that the value of the bit count is decrementing with the SCLK and the output signal on MOSI is loaded with the value of the SPIOut signal with MSB first. In this way the working of the Serial Transmitter block is verified.

3. **Verification Serial Receiver Block**

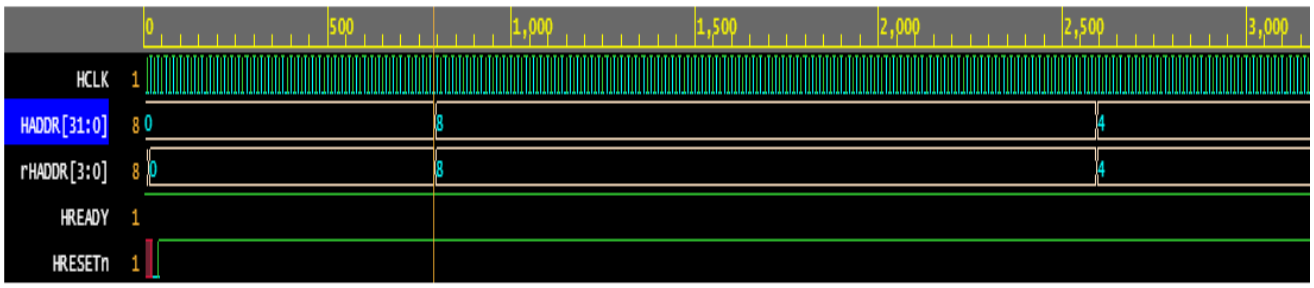


The receiver block is operating on the basis of the high busy and Bipulse signal and low SCLK signal. The receiver block accepts the signal as MISO signal. On every rising edge of the SCLK where SCLK is low and Bitpulse and Busy signal are high, the MISO signal is accepting the value and it is getting added in the parallel collecting signal RxReg. After 8 clock cycle the data is received with MSB first. As the transmitted 8 bit 85 is received in the RxReg register, the operation of the serial receiver block is verified.

4. **Verification Serial Receiver Block**

   The control block is having many components and it makes its verification complex in one single waveform diagram. In order to reduce this complexity each block is verified separately.

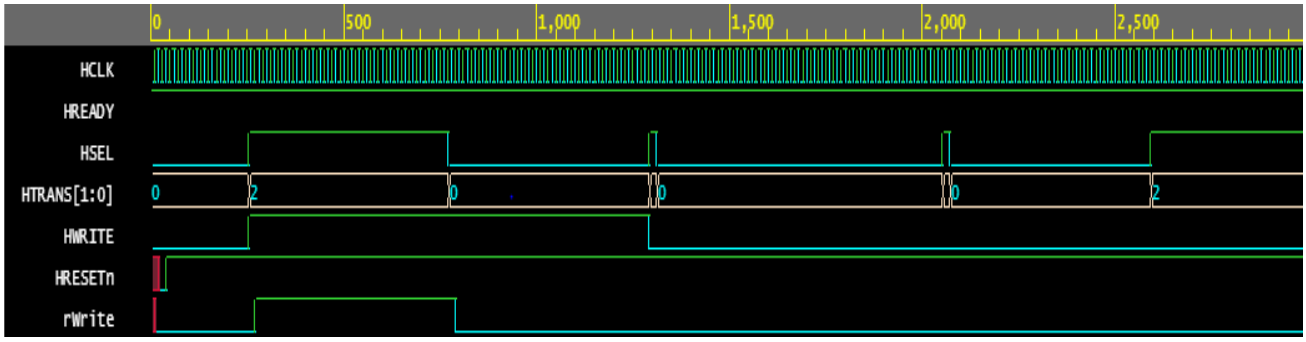
I. **Verification of rHADDR generation**.



In this block the data present on the AHB signal HADDR a 32 bit signal is transferred to rHADDR signal which 4 bit. The value is transferred on the HCLK signal and the block is enabled when the HREADY signal is high. In order to test this , the value of the HADDR is changed first 0 , 8 and then 4 .In the above waveform it can be seen that as

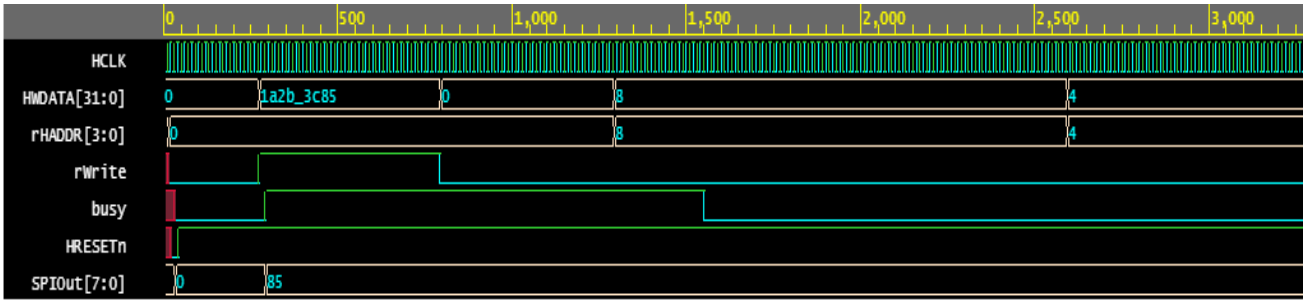
HREADY signal and HRESETn signal is always high, The value is on HADDR is transferred to rHADDR whenever it changes.

II. Verification of rwrite generation



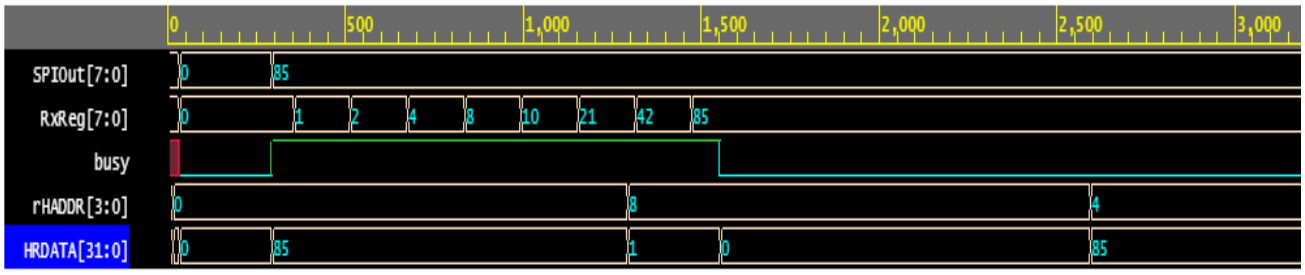
The rwrite signal is generated when the value of AHB signals HWRITE, HTRANS and HSEL is high. To test this the for write operation the HWRITE, HSEL is made high for some duration and the value of 2 is loaded for HTRANS. For the duration of HSEL is high the rwrite signal becomes high this signal will be used to convey all other blocks that processor is requested for write operation.

III. Verification of SPIOut generation



This is the generation of the SPIOut signal which will be transmitted to SPI device through the multiplexer in serial transmitter block.In order to generate this the HEX 85 is loaded on HWDATA with write operation indicated by rwrite signal high in the waveform. The lower 8 of the HWDATA hex 85 can be seen on the SPIout signal. Some changed data can be seen on HWDATA signal but as rwirte signal is low, this data is not loaded on the SPIOut.

IV. Verification of HRDATA generation



The HRDATA is 32 bit signal which will be sent back to the AHB master from the AHB slave. The value of the HRDATA data depends upon the operation expected. To verify

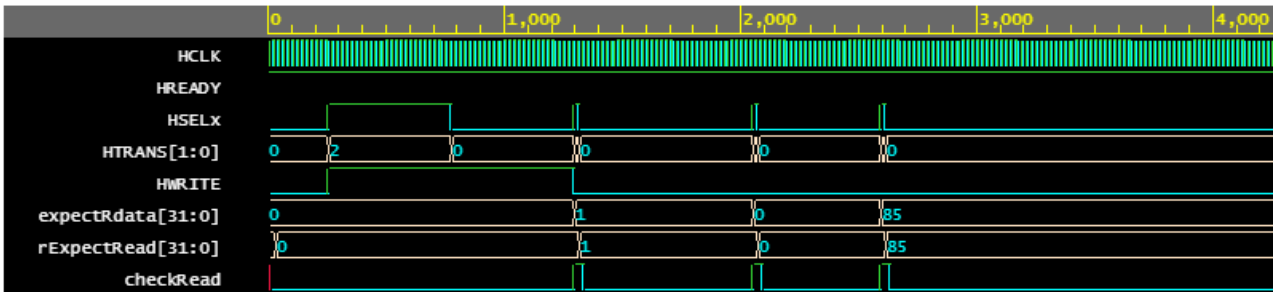
this the current values of SPIOut, Rxreg and busy signal is shown in the waveform. The 0 value of rHADDR signal is expecting write operation with the value in SPIOut register. The 8 value of rHADDR is expecting read operation with the status of busy signal and 4 value expects the read operation with the value in RxReg.

This can be verified from the waveform that when the value of the rHADDR is 0 HRADDR is updated with the value of SPIOut when it is loaded with 85 hex value.

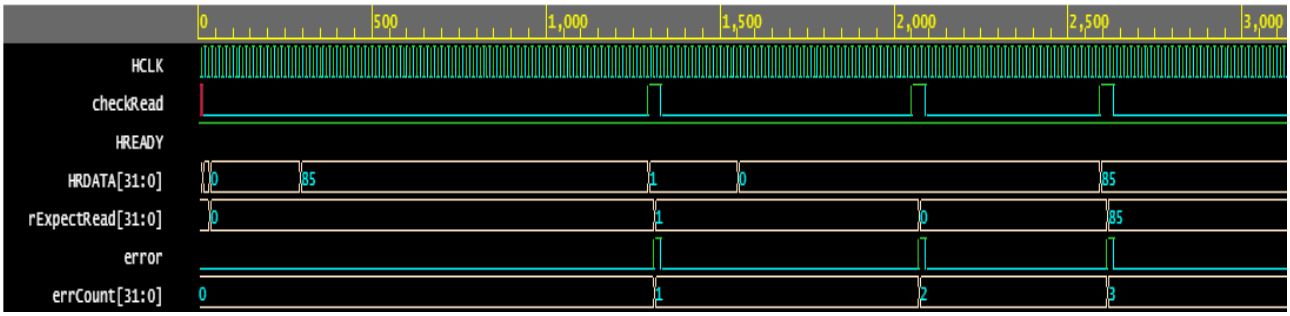
After that rHADDR changes to 4 and busy is high that's why the value of HRDATA is updated to 1 and when busy signal goes low HRDATA becomes 0. At last when rHADDR becomes 4 the value of HRDATA is updated with the value present in the RxReg.

V. Verification of the working of all the blocks combine.

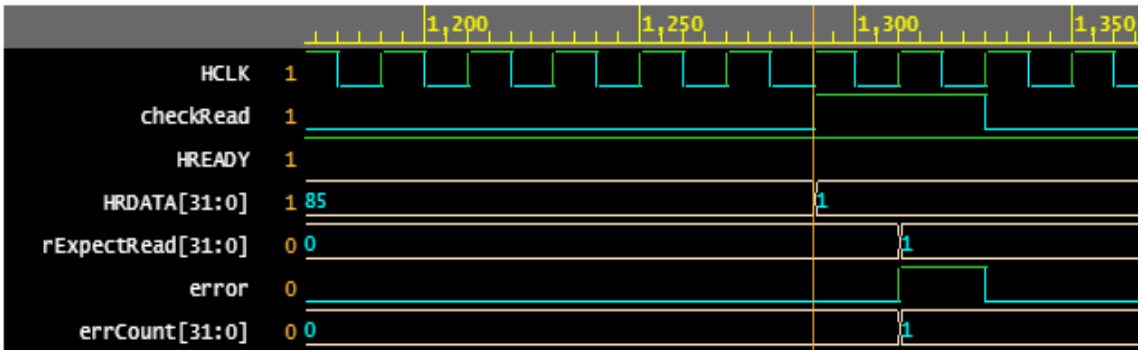
To test if all the blocks working properly, the checkread flag is made high for each read operation. The expectRdata variable stores the data which is expected from the receiver and then it is loaded in rExpectRead .In this case, the sequence is first write operation then two read operation for status read of busy signal followed by the read operation of the received data in serial receiver. As there are 3 read operation the checkRead flag become high 3 times.



In the next block, the data present in the HRDATA is compared with the data in rExpectRead. If the data is different than the error flag becomes 1 and error count will get increment by 1. In the following waveform it can be seen the error flag is getting 1 three time and the final value of error count is 3. This error occurred because there is delay of 1 HCLK cycle. It is shown in another waveform below.



Because of this delay the error flag becomes 1 and error is captured.



Conclusion

In this code, I understood the end to end flow of hardware design process. I started with developing the block diagram according to the requirement list. This block diagram shows top level picture of the hardware. Each block is characterized by different functionalities that needs to be implemented. For example, in this assignment serial transmitter block transmits signal serially and SCLK generation block generates clock signal with required frequency. Then I converted each block into RTL diagram. Through RTL diagrams, I implemented the functionalities of all the blocks using logical gates, multiplexers and shift registers.

On the basis of RTL diagrams, I developed the Verilog code for software simulation which gives waveforms as output. Further, for the verification of each block, I developed a Test bench in Verilog covering all the scenarios. The generated waveforms helped me understand the changing behaviour of input and output signals from each block. Since this assignment covers design as well as implementation of hardware, it gave me an insight on how to convert theoretical design into a working model.

References:

For RTL Diagrams

[1]*DES08_HardwareDesignReview*. Prof.Brian Mulkeen, 2020 [Online]. Available: https://brightspace.ucd.ie/d2l/le/content/57714/viewContent/844936/View. [Accessed: 01- May- 2020]

[2]*DES07_AMBA_Buses*. Prof.BrianMulkeen,2020 [Online].Available:https://brightspace.ucd.ie/d2l/le/content/57714/viewContent/831989/View.[Accessed:01-May-2020]

[3]AHB-GPIO Prof.BrianMulkeen2020 [Online].Available:https://brightspace.ucd.ie/d2l/le/content/57714/viewContent/759580/View. [Accessed: 01- May- 2020]

[4]AHB-UART Prof.BrianMulkeen,2020 2020[Online].Available:https://brightspace.ucd.ie/d2l/le/content/57714/viewContent/759579/View. [Accessed: 01- May- 2020]

For Verilog code

[5]AHBgpio.v Prof.BrianMulkeen,2020 [Online].Available:https://brightspace.ucd.ie/d2l/le/content/57714/viewContent/871102/View. [Accessed: 01- May- 2020]

[6]serialTX.v Prof.BrianMulkeen,2020 2020[Online].Available:https://brightspace.ucd.ie/d2l/le/content/57714/viewContent/866670/View. [Accessed: 01- May- 2020]

[7]*TB_serialTX.v* Prof.BrianMulkeen,2020 [Online].Available:https://brightspace.ucd.ie/d2l/le/content/57714/viewContent/873843/View. [Accessed: 01- May- 2020]

[8]TB_AHBgpio Prof.BrianMulkeen,2020 [Online].Available:https://brightspace.ucd.ie/d2l/le/content/57714/viewContent/878943/View. [Accessed: 01- May- 2020]