Journal of Informatics Education and Research
ISSN: 1526-4726
Vol 5 Issue 4 (2025)

Simulation and Synthesis Model of SPI Bus using Verilog HDL

Ravi Payal
CDAC Noida
ravipayal@cdac.in

K Bhagirath
CDAC Noida,

kbhagirath@cdac.in

***Abstract*—** The SPI Protocol is very widely used serial protocol used in various applications. This paper discuss about Implementation of SPI Protocol using Verilog HDL and targeting the FPGA technology. The First two section of the Paper talks about the SPI working and past work done using this protocol. The paper also deals with SPI working and various modes of operation and finally deals with desiging and implementation. The whole design code for simulation, synthesis and FPGA implementation is used by Verilog HDL.

Keywords*—SPI , FPGA, Verilog HDL, State Machine, Moore Machine. Protocol.*

I. Introduction (*Heading 1*)

SPI protocol used in many devices for communication purposes. This protocol was invented by Motorola in the year 1979. Even after 4 decades of its invention this protocol still widely used for many application in embedded system. It is used for various applications like for temperature and pressure measuring, ADC, DAC, in video games, in SDC card etc. While in most of the protocols data is passed through in packets but in SPI data (in terms of bits) transferred continuously from one point to another without any interruption. This data transfer without any interruption is main advantage with SPI protocol.FPGA devices are popular now a days due to its reconfiguration capability concept. Researches are solving lot of new communication problems with the help of growing capabilities of FPGA. FSM State Machine approach gives RTL level coding gives designers a flexibility to do modification at a entry level and fast modifications according to need.

II. Literature Review

Lot of work is being done by researchers in past.

With the increasing demand for high-speed data communication, enhancements to traditional SPI implementations become necessary. Khan, Nisar, and Aftab (2018) explored a high-speed SPI interface implementation on an FPGA. Their work showed that by optimizing the clock frequency and using FIFO (First In, First Out) structures, significant improvements in data rates could be achieved.[1] .

In addition to speed, ensuring secure communication has become crucial. Selimis et al. (2011) proposed an FPGA-based SPI protocol that incorporated AES encryption for secure data transfer. Their approach showcased how FPGAs can be efficiently used for real-time data encryption in SPI communication.[2]

Several research works have presented the basic design and implementation of the SPI protocol on FPGA platforms. Ahmed, Shah, and Hussain (2015) proposed a design of SPI master-slave configuration using Verilog and implemented it on an Altera FPGA. Their work involved developing separate state machines for the master and the slave, with a focus on efficient data transmission and synchronization.[3]

While FPGAs offer many advantages for SPI implementation, some challenges like power consumption, real-time reconfiguration, and multi-master configurations need to be addressed (Loo, 2020). The trend seems to be moving towards integrating more intelligence and capabilities directly into the SPI hardware implementation rather than depending solely on software-level implementations.[4]

With the increasing complexity of communication systems, multi-master SPI configurations are gaining traction. Lee et al. (2017) present a Verilog model that supports multi-master configurations, detailing the challenges and solutions for arbitration and synchronization in such systems.[5]

III. SPI WORKING MODEL AS A STATEMACHINE

SPI is a communication protocol which is serial in nature and generally used to transfer the data between various controllers and different peripheral devices [6]. The protocol was developed by Motorola and has become a widely used standard in the industry. The SPI bus is a serial protocol where 8-Bit byte data is shifted bit by bit in a cycle [7].SPI is communicating with other serial devices by using some interface..

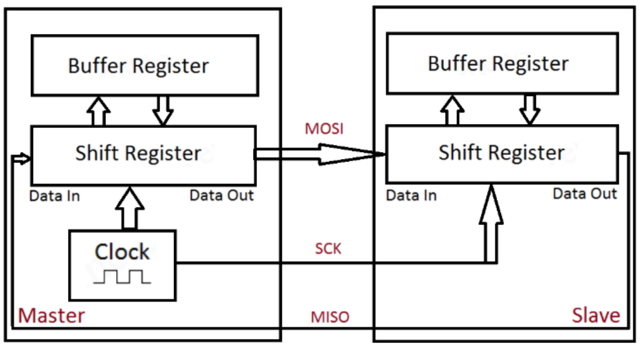

Figure1. Block Diagram of SPI Protocol

In the above block diagram:

•MOSI: Master Output Slave Input line. Here master is used to send data to device.

•MISO: Master Input Slave Output. Here slave is used send data to the device acting as a master.

•SCLK: Serial Clock. This line is used to synchronize the data transfer between master and slave.

•SSEL: Slave Select. line is used by the master for searching the slave with whom it wants to talk.

In addition to these lines, the SPI interface may have other pins for power and ground connections, as well as for configuring the communication parameters. However, these are not shown in the block diagram above.

The main features of SPI Protocols are

Master-slave architecture: SPI data transfer communication totally depends on the Master-Slave architecture design process where the master starts the data transfer process and controls all the slaves.

Data transfer: Data is transferred serially using a four-wire interface. These include a clock signal (SCK), a master output/slave input data line (MOSI), a master input/slave output data line (MISO), and a slave select signal (SS).

Clock signal: The master device generates a clock signal that synchronizes data transfer between the master and slave devices. The clock signal is used to determine the timing and rate of data transfer.

Data transfer modes: SPI supports different data transfer modes, such as full-duplex (where data can be sent and received simultaneously) or half-duplex (where data can only be sent in one direction at a time).

Data format: The data format can be configured to support different data sizes, such as 8-bit or 16-bit, and the data can be transmitted in various formats such as LSB (Least Significant Bit) or MSB (Most Significant Bit) first.

Chip select signal: Each slave device is assigned a unique chip select signal (SS) and this signal is used by the master device for selecting the device to whom it wants to talk.

Multiple slave devices: SPI protocol establish the data transfer with different slave devices, and each device is connected to the same clock, data, and select lines.

Simple protocol: SPI is a simple protocol with no packet structure, and the communication is based on a simple handshake mechanism between the master and slave devices.

At first between master and slave, no data transfer takes place. SCK is in control of Host and output of CS is in inactive state. At this moment SPI bus is in initial state. During data transfer state, host will control the CS output and SPI bus will operate. Here Master and Slave devices will transmit the data through MOSI and MISO and on the next clock both the devices will receive the data at the same time. When transfer of data is complete, SCK comes to idle state. When during the condition where FPGA acts as a host then CS signal and SCK signal always be initiated by FPGA.

SPI bus during transmission of Data signal also transmits the Clock signal.

The SPI bus transmission has four modes of operation which are defined in the below table

MODE	CPOL	CPHA	WORK
Zero	Low	Low	SCK clock line idle is low, on positive end of SCK data is sampled and data is switched on negative end SCK .
One	Low	High	SCK clock line idle is low, on negative end of SCK data is sampled and data is switched on positive end SCK .
Two	High	Low	SCK clock line idle is high, on negative end of SCK data is sampled and data is switched on positive end SCK .
Three	High	High	SCK clock line idle is high, on positive end of SCK data is sampled and data is switched on negative edge of SCK .

Table1. Modes of SPI bus transmission

Concept of Moore machine approach was used for SPI bus transfer. FSM finite state machine which controls SCK, MOSI and CS signals was used. Simple concept of State-machine approach was used . The process is divided into three state machines.

State1:(Initialization state)

State2:(Transmission State)

State3: (End State)

During state1 (Initialization state) all variables must be in idle state. SCK bus clock will be in low state and most information bit in low and data is continuously transferred. SCLK=0; CS=1;MOSI=0;done =1.

During transmission state(State2) CS signal is set to low. when the communication started then SCK clock is enabled and data transmission towards MOSI is started. A counter is established which will verify the transferred data.

In the last state (Finish state) SPI bus is disabled by putting the CS in high state and putting the vectors and counters at initialization values. Here before going to initial state all the variables are places the done bit to high state signal which indicated the finish of data communication.

IV. RESULTS

The design was simulated in Questa Sim (Version 10.2) Simulator software platform and verified using Verilog HDL language. The simulation was running perfectly fine. The figure (2) is a waveform of successful data transmission.

The Design was synthesized on Xilinx Zynq board(device number 7Z010ICLG225). The RTL and Technology view was generated and shown in figure (3) and (4).

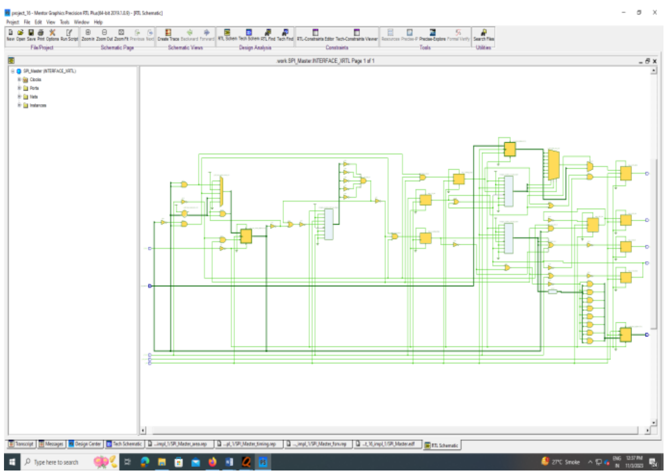

Figure3.RTL view

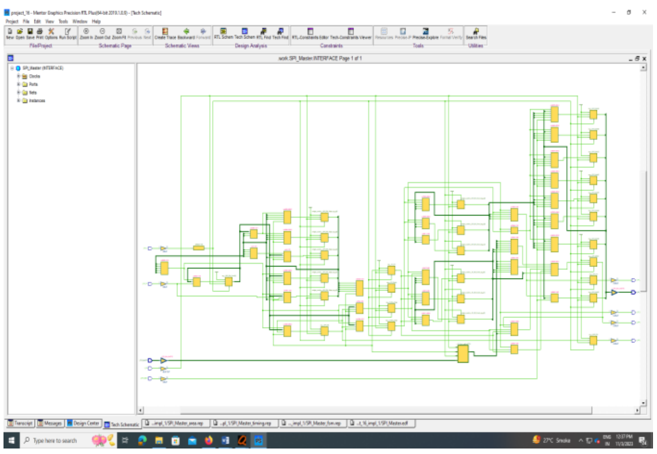

Figure4. Technology View

The generated Technology view consist of various cells from the target technology library. The Table (2) consist of Area report of SPI RTL.

Resource	**Used**	**Avail**	**Utilization**
IOs	24	54	44.44%
Global Buffers	1	32	3.13%
LUTs	35	17600	0.20%
CLB Slices	5	4400	0.11%
Dffs or Latches	36	35200	0.10%
Block RAMs	0	60	0.00%
DSP48E1s	0	80	0.00%

Table (2): Resource Utilization Report

Table(2) explains the usage of various cells from a technology library of Xilinx Zynq- 7Z010ICLG225.

For calculating the slack of design we put required time 2.00 ns whereas after synthesis the Data arrival time came around 0.445 ns. As slack is the difference between required time and arrival time. So Net slack came 1.55 ns. As we know that whole timing of the design is considered around critical path. Figure(5) shows the Critical path of the design.

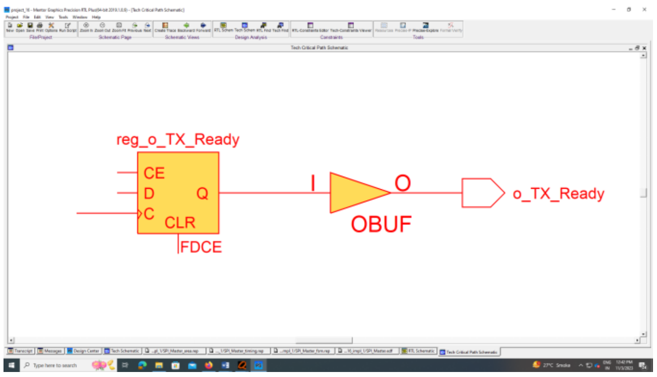

Figure5. Critical Path of the design

V. CONCLUSION

SPI protocol design was implemented successfully using verilog HDL. The Approach of FSM machine specially Moore model was used in the RTL level Simulation.The SPI model was successfully synthesized on target FPGA library (ZYnnQ) of Xilinx with device number 7Z010ICLG225 .

This paper initially describe the SPI bus with verilog HDL,

designes a SPI host model and then Performs a functional simulation. Due to the usage of SPI bus in many application areas and reconfigurability property of FPGA ,the designed model will be further elongated with respect to various different applications.

REFERENCES

[1] 1: Khan, M., Nisar, W., & Aftab, M. (2018). High-speed SPI protocol for FPGA based embedded systems. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 26(2), 319-327

[2] Selimis, G., Konijnenburg, M., Ashouei, M., Huisken, J., De Groot, H., & Van Der Leij, L. (2011). A self-calibrating, sub-vt, variation-tolerant security aware SPI protocol for ultra-low-power sensor nodes in wireless body area networks. IEEE Transactions on Biomedical Circuits and Systems, 5(4), 349-358

[3] Ahmed, R., Shah, M. A., & Hussain, S. (2015). Design and Implementation of SPI Master Slave Using FPGA. International Journal of Computer Applications, 113(1), 21-25..

[4] Loo, Z. H. (2020). FPGA-based systems design: Trends and challenges. Journal of Computer Science and Technology, 35(1), 6-14.

[5] Lee, D., Kim, Y., & Lee, J. (2017). Multi-master SPI design and verification using Verilog. IEEE Transactions on Circuits and Systems II: Express Briefs, 64(7), 751-755.

[6] Shingare, T.D. and Patil, R.T., 2013. SPI implementation on FPGA. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 2(2), pp.7-9.

[7] https://ww1.microchip.com/downloads/en/DeviceDoc/70272A.pdf