doi.org/10.24191/jeesr.v19i1.011

Design and Simulation of Serial Peripheral Interface Core with APB Interfacing

Muhammad Hafeez Sabparie, Emillia Noorsal, Suhana Sulaiman, and Azilah Saparon

***Abstract*— The model and design of the IP core of Serial Peripheral Interface (SPI) with Advanced Peripheral Bus (APB) interfacing is presented in this paper. SPI is known as a kind of serial protocol for serial communication bus developed by Motorola and has become de facto standard. There is possibility for a system to have more than one slaves for the integrated circuit, however the master can only be one at any given time. Hence, in this work, the SPI was modelled by Verilog code and it was simulated and synthesized using ModelSim and Quartus Prime Lite Edition 16.0 for earlier stage of design. While Synopsys Tools *i.e.* Design Compiler was used as the primary synthesis for the design. The SPI interface was designed to send or receive data from a single slave and efficient APB-SPI controller with flexible data width and frequency is proven for maximum frequency of 16 MHz. The modes of SPI also play its role in this work where this protocol can run through four modes that corresponds to four possible clocking configurations. The results showed that the core of SPI was successfully modelled for mode 0, 1, 2 and 3. In additon, the modes were simulated with maximum operating frequency of 16 MHz and flexibility in all four clocking modes. The ASIC design of this work consumed 27750 µm^2 and 47.12µW using Silterra 0.18µm CMOS process.**

***Index Terms*— SPI, IP, APB-SPI controller, single slave.**

I. INTRODUCTION

THERE are many types of protocols in communication and it is very clear that these protocols are very important and each of the protocols is different and has its own methods. The commonly used and popular protocols is Serial Peripheral interface (SPI), Inter-Integrated Circuit (I^{2}C) and wireless for serial communications. In 1979, SPI was introduced and was defined as external microcontroller bus which connected the four wires with microcontroller peripherals. Meanwhile in 1982, the I^{2}C was developed. The purpose was to provide a connection between peripheral chips in television set to a CPU in an easy way. For I^{2}C protocols, all the peripherals need only two wires to be connected to a microcontroller. SPI is a serial communication bus developed by Motorola. It is a full-duplex protocol that functions on a master-slave paradigm that is ideally suited to data stream application [1].

Communication between the two processors is handled via the serial peripheral interface (SPI). Every SPI system consists of one master and one or more slaves, where a master is defined as the microcomputer that provides the SPI clock, and a slave is any integrated circuit (IC) that receives the SPI clock from the master [2]. It is possible to have a system where more than one IC can be master, but there can only be one master at any given time. In this issue, SPI is preferable for high-speed and it is full duplex. This is because SPI can achieve higher data rates which are limited to 400KHz and compatible interfaces often range into 10 MHz but normally it serves several slaves. Furthermore, SPI does not define any speed limit, the implementation often performs over 10Mb/s [3-4]. Though, I^{2}C is also serial interface and it can have multi-master, but the speed is limited to 1Mb/s in fast mode and 3.4Mb/s in high-speed mode.

Hence, the motivation of this study is to model the SPI unit using HDL. Although there are many literatures that describe the design of SPI using HDL, their designs are aimed at different application and the communication protocol involved is between the SPI master and slave only [3-5]. Since Advanced Microcontroller Bus Architecture (AMBA) APB Bus Interface is widely used in any processors [6], this is another motivation for this work which is to develop SPI model which can interface with AMBA-APB Bus. Furthermore, the model system can improve the effectiveness of transferring data accordingly. Also, the model is aimed to transmit data and receive data with four types of modes that corresponds to four possible clocking configurations in SPI itself. Therefore, the SPI with an efficient controller can allow only the master to operate with flexible data width and multi selective frequencies and clocking configurations due to the applications. The model and design of the SPI unit can be utilized to interface with the system main data bus such as Advanced Microcontroller Bus Architecture (AMBA) APB Bus Interface Core and it is challenging to control the SPI unit through the AMBA Bus. Although there are designs for this interface, the available works only focus on one mode and there is no available paper that describes the implementation in ASIC [7-10]. Hence, this paper shows the methods and results of the design which would be beneficial to IC design researchers.

This manuscript is submitted on 26 February 2021 and accepted on 24 June 2021. The work is partly the MSc tesis of Muhammad Hafeez Sabparie.
Muhammad Hafeez Sabparie, Suhana Sulaiman and Azilah Saparon are with the School of Electrical Engineering, College of Engineering, Universiti Teknologi MARA, 40450 Shah Alam, Selangor (email: muhd27hafeez@gmail.com, suhana832@uitm.edu.my, azilah574@uitm.edu.my)
Emilia Noorsal is with School of Electrical Engineering, Universiti Teknologi MARA, Penang, Malaysia (emilia.noorsal@uitm.edu.my).

II. RESEARCH METHODOLOGY

This section describes the SPI design flow and details the process based on the specification and functionality of each sub-modules. There are two parts of explanation, *i.e.* Design Specification and Serial Peripheral Interface (SPI) Architecture, respectively.

Fig. 1 shows the overall process flow for designing IP core of SPI with APB interfacing design. Initially, the SPI was remodeled and simulated with multi slave using several simulation solutions such as ModelSim, Synopsys VCS and Quartus Lite 16 for the flexibility of data width of SPI. Further investigation on flexibility of data width of SPI is conducted thru data analysis to ensure the model fulfil the design specification successfully. Once, all design specifications are fulfilled, the task is continued for the layout and post simulation and synthesizing using Design Compiler and Astro.

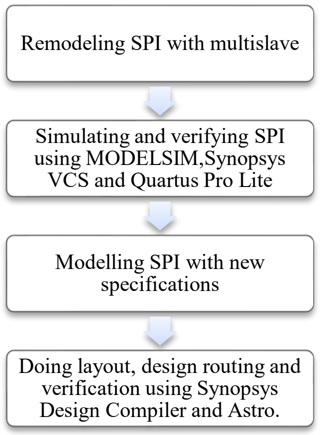

Remodeling SPI with multislave
Simulating and verifying SPI using MODELSIM,Synopsys VCS and Quartus Pro Lite
Modelling SPI with new specifications
Doing layout, design routing and verification using Synopsys Design Compiler and Astro.

Fig. 1. Overall Process flow for the design of IP Core of Serial Peripheral Interface (SPI) with AMBA APB Interface

A. *Design Specification*

In this work, the design specification was aimed to design the SPI with an efficient controller for the master to operate with flexible data-width and multi selective frequencies so that it can fulfil many applications such as Nordic wireless microcontroller unit.

Hence to fulfill the application, the design of SPI must satisfy certain constraint such as its maximum clock frequency i.e. 16Mhz and other constraints such as:

i. Full duplex, synchronous, 8-bit serial data transfer
ii. Master or slave mode
iii. Fully Synchronous design with one clock domain
iv. Simple interface allows easy connection to microcontrollers (IO port)
v. APB interface

B. *Modelling and Simulation of Serial Peripheral Interface (SPI)*

This section focuses on the system architecture design of the SPI protocol. There are three type of functional modules: clock generator, serial interface or top module, and the wishbone interface.

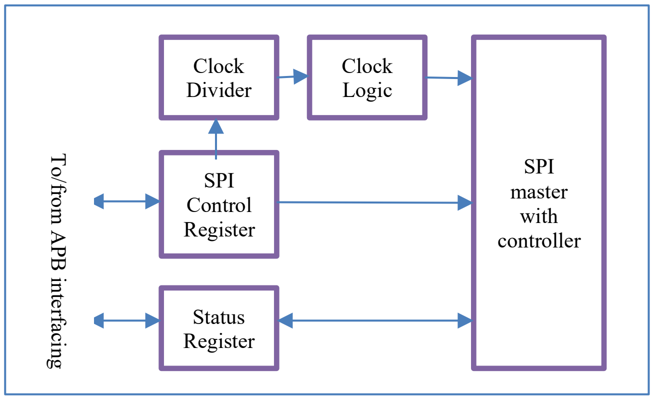


Fig. 1: Internal design sub-modules of top module SPI []

Finally, the top module design is to ensure the sub module of the SPI working smoothly and accordingly as planned. With all these have been done, a test bench was design to act as input or in other word to be act as AMBA bus data transmission. This test bench was design using Verilog language to check on the input data transmission of the SPI and the output data that carry out from the SPI to the Slave.

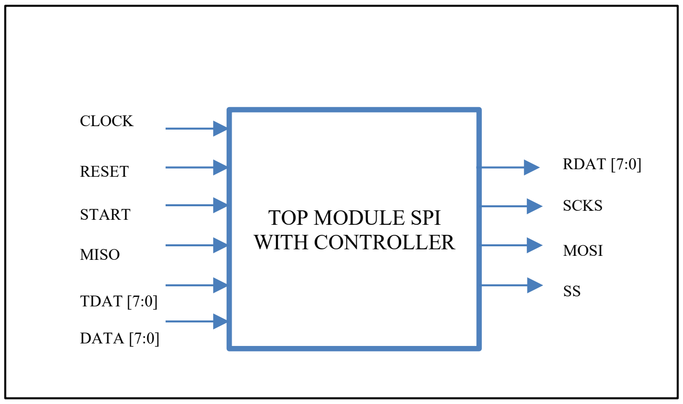


Fig. 2. Block diagram of SPI top module

Fig. 2 refers to the block diagram of SPI top module.The top module is divided into 4 main modules: SPI control register module, SPI clock divider module, SPI clock logic module, SPI status Register and SPI master controller module. In addition, each one of the modules are designed using Verilog HDL [11].

The clock generation design module is to ensure the timing reliability of the SPI protocol design. This is because, in SPI protocol design there will be no response mechanism. With this clock generation design module, the SPI can generate reliable serial clock of transmission.

The interface module design of the SPI is essential because it has to follow the SPI protocol which is the data to be transferred from parallel data in into serially in and serially out data to be transferred parallel out if needed. It can bring benefit to the design of the SPI such as increasing the overall rate of data transmission.

Table 1 displays the input pins of the top SPI module as well as the pins respective function.

TABLE I
INPUT PINS OF TOP SPI MODULE

Pins (Input Top SPI)	Function
CLOCK	The clock input defines the bit-rate of the serial communication (16MHz max frequency)
RESET	Resets the SPI state machine to the idle state
START	Start data transmission (e.g.: start =1)
MISO	The MISO input carries the master input
Tdat [7:0]	Data transmitted by the APB
Data [7:0]	Master input data

As depicted in Table 2, there are four output pins to be transferred to slave and each pin has its own functionality.

TABLE II
OUTPUT PINS OF TOP SPI MODULE

Pins (Output Top SPI)	Function
MOSI	The clock input defines the bit-rate of the serial communication (16MHz max frequency)
SCKS	Resets the SPI state machine to the idle state
RDAT [7:0]	Start data transmission (e.g.: start =1)
SS	The miso input carries the master input

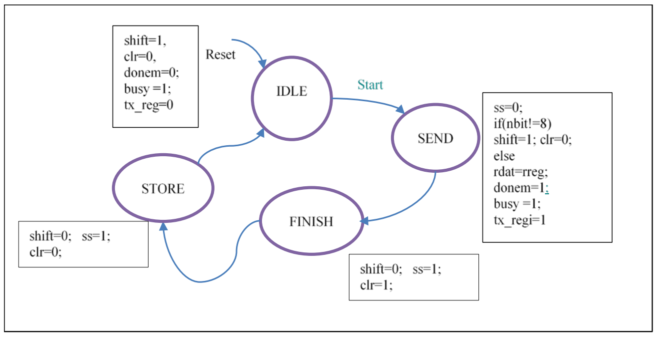


Fig. 3: State diagram of SPI controller

Fig. 3 shows the state diagram of the SPI controller. The Finite State Machine (FSM) of the controller determines the value of the control signal. These signals are initialized at IDLE state whenever the Reset signal is asserted. Signal shift is set to logic 1 while signal *clr* and *donem* are set to logic 0. The *clr* signal clears all registers. At SEND state, the *shift* and *clr* are still maintained at 1 and 0, respectively. This is to allow the master to transmit the data and save them in the register, *rdat*. When all data have been transmitted, signal *donem* is asserted HIGH. Then, the system goes to state FINISH and stop shifting the data and clear the counter register, *nbit*. Lastly, the status signals are is stored in the status register.

The status register holds status flags, and these values are notification from the SPI master that certain process is still in progress or completed. Table 3 describes the function and number of bits used for the status signal.

TABLE III
FUNCTION OF STATUS REGISTER

Bit(s)	Pin Name	Function
7	enable	Indicates that Core SPI is enabled. If this bit is set to logic 1, Core SPI is currently enabled
6:4	Unused	unused
3	busy	Indicates that the Master is busy. If this bit is at logic 1, the Core SPI Master is currently transferring data. This status bit is used to check if the SPI Master is busy before disabling it.
2	tx_register_empty	New data for transmission can be written to the Transmit Data Register when this bit is at logic 1
1	rx_data_ready	When this bit is at logic 1, the RX Data Register must be read before the next character is received
0	error	If this bit is at logic 1, it indicates that a character has been received before the previous character has been read from the RX Data Register.

As mentioned previously, there are three particular tools are used for the simulation process: ModelSim, VCS and Quartus Lite 16. ModelSim and VCS are used to collect data or the waveform of the simulation. Both tools are used to see the simulation results of the SPI where the Quartus Lite 16 is used to synthesize and validate the condition of the SPI protocol. For the simulation process, there are few simulations to observe the maximum frequency of the SPI, the input and output of the SPI, and 4 different modes of SPI.

Table 4 presents the SPI modes definitions for the simulation waveform. As depicted in Table 4, the manipulative variable is the clock polarity and clock phase. The SPI interface defines no protocol for data exchange overhead limitation and will allow high speed data to stream. Clock polarity (CPOL) and clock phase (CPHA) will be specified as ‘0’ and ‘1’ to form four different and unique modes that is to provide flexibility to communicate between master and slave. The Mode 0 is the most common mode in SPI bus Slave communication.

TABLE IV
SPI MODES DEFINITIONS

MODE	Clock Polarity (CPOL)	Clock Phase (CPHA)
0	0	0
1	0	1
2	1	0
3	1	1

To verify the functionality of the SPI, a Verilog model of SPI testbench is developed. The testbench forces stimulus into the Device Under Test (DUT) *i.e.* SPI Core and monitoring the output of the SPI and the content of registers. Fig. 4 shows the snippet of the stimulus in the testbench.

```
%Enabling SPI and assertions
start = 1; miso = 1;
ctr_data = 8'b0_0_1_00_001;
tdat = 8'b10101010;
```

Fig. 4: Stimulus for SPI input

The control register holds the 8-bit data sent from APB which the data represents control signals to SPI master. The *ctr_data* represents interrupt, mode of SPI, data order, clock phase, clock polarity, clock divider and in this case, the interrupt is set to 0 which it disables the interrupt signal. The mode of SPI is in master mode and order of sending the data is MSB instead of LSB. The transmission mode is set to Mode 0 which the clock polarity (CPOL) and clock phase (CPHA) are set to '0' and then a data of *8'b10101010* is pushed in by the test bench to observe the input and output for this mode. This part of the code is changed to test for other modes which are Mode 1 : CPOL is '0' and CPHA is '1', Mode 2 : CPOL is '1' and CPHA is '0' and Mode 3 : CPOL is '1' and CPHA is '1'

When all the modes that have been tested, the SPI Verilog model is processed using Synopsys tools for the ASIC design. Synopsys Design Compiler (DC) and Astro analysis are performed for the layout and post simulation as well as synthesizing.

III. RESULTS & DISCUSSION

This section exhibits the findings of Register Transfer Level (RTL) Netlist Viewer Block Diagram of SPI using Quartus Prime 16.0, Design Compiler (DC) Analysis for SPI and Simulation Results using ModelSim and Silterra Astro.

A. Simultation Results for Each Modes of SPI using MoldelSim and VCS

Referring to work in [1], the simulation and synthesized SPI design, the RTL Block diagram and also signals for mode 0 has been presented. Hence, the following is the subsequent results of the timing waveform for MODE1, 2, 3 respectively.

1) Simulation Result of Timing Waveform for MODE 1

Fig. 5 displays the result of timing waveform for SPI for MODE 1. In this mode, clock polarity is 0. This indicates that the idle state of the clock is low. However, the clock phase of this mode is 1. This implies that the data for this mode was latched and sampled on the falling edge/ negative edge of the scks (clock signals) and shifted on the positive edge/ rising edge of the scks (clock signals). In short, MODE 1 was formed with a clock polarity that was non-inverted (i.e., the clock is at logic low when slave select transitions to logic low).

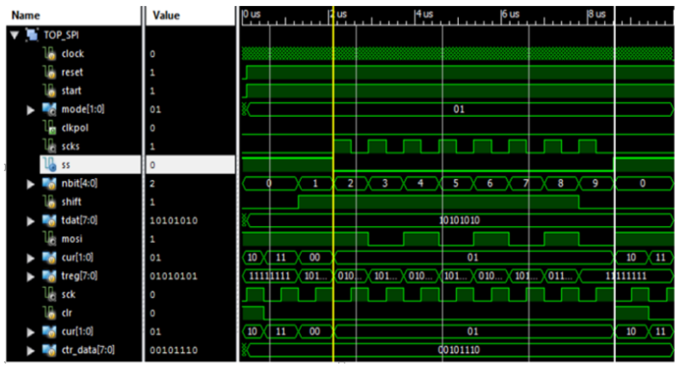
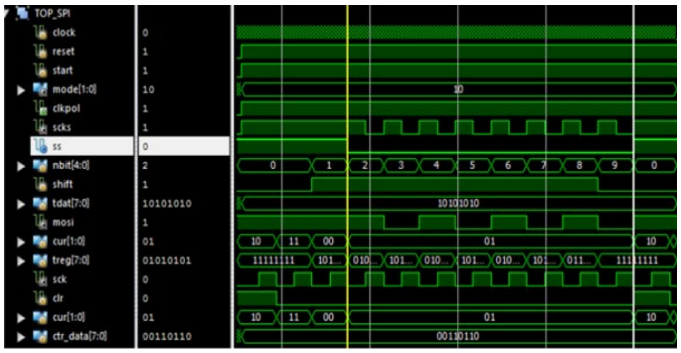


Fig. 5: Timing waveform for SPI of MODE 1

2) Simulation Result of Timing Waveform for MODE 2

Fig. 6 shows the simulation result of timing waveform for SPI of MODE 2. In this mode, clock polarity is 1. This indicates that the idle state of the clock signal is high and the clock phase of is 0. This deduces that the data for this mode was latched and sampled on the falling edge/ negative edge of the *scks* (clock signals) and shifted on the positive edge/rising edge of the *scks* (clock signals). Briefly, Mode 2 was formed with a clock polarity that was inverted (i.e., the clock is at logic high when slave select transitions to logic low).

Fig. 6: Timing waveform for SPI of MODE 2



3) Simulation Result of Timing Waveform for MODE 3

Simulation result of timing waveform for SPI of MODE 3 is presented in Fig. 7. In this mode, clock polarity is 1 which indicates that the idle state of the clock signal is high. In addition, the clock phase of this mode is 1. The result suggests that the data for this mode was latched and sampled on the rising edge/ positive edge of the *scks* (clock signals) and shifted on the negative edge/falling edge of the *scks* (clock signals). In short, MODE 3 was formed with a clock polarity that was inverted (i.e., the clock is at logic high when slave select transitions to logic low).

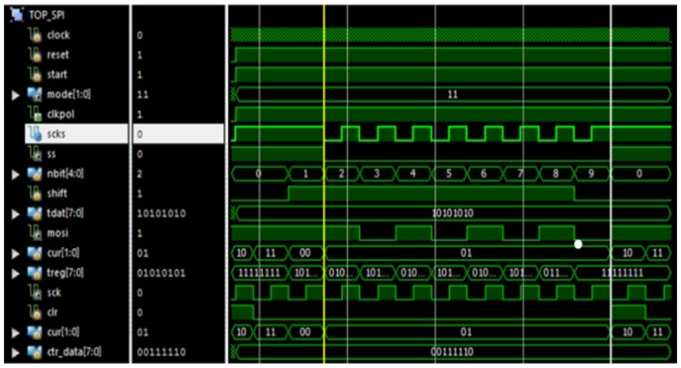


Fig. 7: Timing waveform for SPI of MODE 3

In summary, these results show that the clock polarity and clock phase can be set and varied according to the applications determined for the SPI. In this study, this is accomplished in the master control unit where the master controller will determine which mode the SPI will make use of. Prior studies that have noted the importance of other SPI will lock on one mode and the mode was MODE 0. What is interesting about our SPI is the design SPI able to be flexible by using all four modes that has been set up in the master. These differences can be explained whereby the idle state was defined as the period when slave select (*SS*) was high and transitioning at low at the start of their transmission and when slave select *SS* is low and transitioning to high at the end of the transmission. As shown in Fig. 7, the clock phase (CPHA) bit select the clock phase. The rising or falling clock edge was used to sample and / or shift the data depending on the CPHA bit. The master must select both clock polarity and clock phase as per requirement of the slave. This combination of findings provides some support for the SPI to provide all four modes for the slave that intend to relate to it depending on the CPOL and CPHA bit selection in the master.

B. Register Transfer Level (RTL) Netlist Viewer Block Diagram of SPI using Quartus Prime 16.0

Fig. 8 exhibits the overall RTL master block diagram for the SPI.

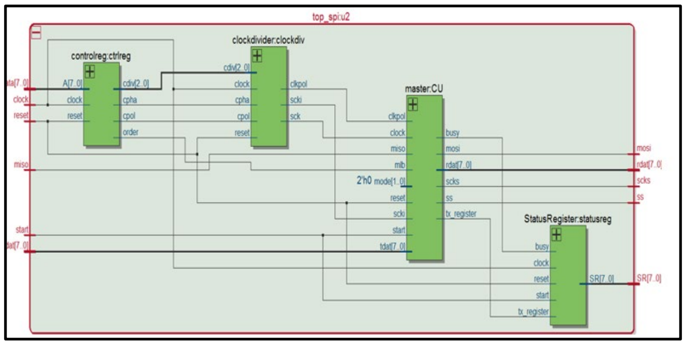


Fig. 8 : Full Master block diagram for SPI from RTL NETLIST VIEWER

As shown in Fig. 8, there are four main block diagram that was constructed in the SPI. The blocks are control register, clock divider, master control unit and status register. The description for each block diagrams has been discussed in [12].

To summarize, the SPI control register module was designed to function like a CPU. A CPU can Read from and Write to the 8-bit. Where Data [7:0] is Master input data to the Control Register module. The outputs of the control register module will be the inputs of the clock divider modules.

The clock divider module will provide the clock for the master controller. The clock divider can define the data transfer rate and it can select the serial data clock frequency of the *sck*. In general, period T is defined as inverse of frequency f *i.e*, T= 1/f. In this design, the clock is divided when the master is operating based on the 3 bits selection of divider. This will select the serial data clock frequency of the *scks* signal which will also define the data transfer rate. Table 5 illustrates the division frequency for chosen selection bits.

TABLE V
SELECTION OF FREQUENCY DETERMINED BY 3-BIT CLOCK DIVIDER

[2:0] SCK	*Frequency*	*[2:0] SCK*	*Frequency*
000:	*fPCLK ÷ 2*	*100:*	*fPCLK ÷ 32*
001:	*fPCLK ÷ 4*	*101:*	*fPCLK ÷ 64*
010:	*fPCLK ÷ 8*	*110:*	*fPCLK ÷ 128*
011:	*fPCLK ÷ 16*	*111:*	*fPCLK ÷ 256*

Full Master block diagram for SPI is Master control unit as shown in Fig. 8. Previously, clock input defined the bit rate that came from the clock divider of the serial communication. In this module, MISO is the master in and slave out pin. This pin is used to receive data when it is configured as Slave and transmit data of the SPI module when it is configured as a Master. Reset act similarly as the other module which will reset the SPI state machine to idle state. While *Tdat* came from the data transmit from the APB and finally the *start* is an indication to start running with the program. As for the output, MOSI pin is used to receive data when it is configured as Master and transmit data out of the SPI module when it is configured as a Slave. While *Scks* is a pin for the output of the clock with either SPI receive data or transfer data clock in case of Slave. SS is a slave select that is used to enable the slave. Lastly, *Rdat* is an 8-bit data that will be transferred out from the master.

The final module of Full Master block diagram for SPI is status register as depicted in Fig. 8. The description for the pins for the output of Top SPI design is referred in [12]. The function of status register is to hold signal flags and send to the APB by indicating what is being executed by the SPI.

Overall, each modules of the RTL constructed in the SPI provide different functions toward the SPI. The differences between modules have been highlighted discretely to show that all modules have related to one another after being synthesized. In addition, the block diagram shows that all modules required the input that applicable and the master controller can produce four output to the slave and one to check the status of the serial peripheral interface, which is MOSI, SS, SCKS, Rdat [7:0], and SR [7:0].

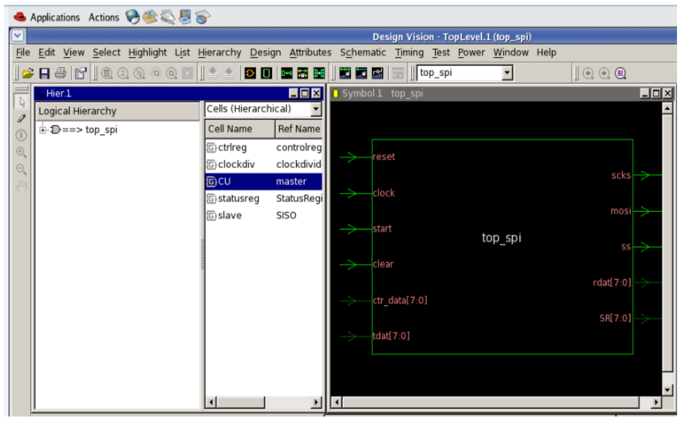
C. Results of Design Compiler (DC) Analysis for SPI

The analysis from Design Compiler mainly focusing on the synthesis part of SPI. The design compiler (DC) analysis results emphasizing the block diagram for the SPI which include the top level of the SPI. Also, the result produced report scripts such as report area, report constraints, report power, report timing maximum and minimum of the SPI.

1) Block Diagram Of SPI Using DC Analysis

Fig. 9 (a) and 9 (b) display the TOP module of the SPI and block diagram of SPI after the synthesis process respectively. The TOP module of the SPI is the similar TOP module attained from Quartus Prime 16.0. This implies that the SPI is correctly constructed with the Verilog code. Fig. 9 (b) represents the details of the TOP SPI after the Synthesis Process from Design Compiler (DC) Analysis. All inputs and output are indispensable as expected in the design. In this simulation, the process was essential to ascertain there is no error in Synopsys Astro analysis where all the floor planning, placement and routing is expected of this work. With the DC, the SPI also undergoes the process of optimization and compilation after adding the constraints to design. Prior to the addition of constraints, the block diagram of the SPI from the design compiler has been monitored cautiously to avoid any leakages and any error in the design such as unconnected wiring between the modules.

By comparing the block diagram with the block diagram in Quartus Prime 16.0, the finding revealed that there is no error in the synthesizing of the SPI. The result deduces that there is no leakage on any modules and there is no unconnected wire. To sum, when both block diagram produce similar actual diagram; this signify that the design should be good enough to proceed to the next process of the research.



Cell Name	Ref Name
ctrlreg	controlreg
clockdiv	clockdivid
CU	master
statusreg	StatusRegi
slave	SISO

Fig. 9 (a): TOP module of SPI

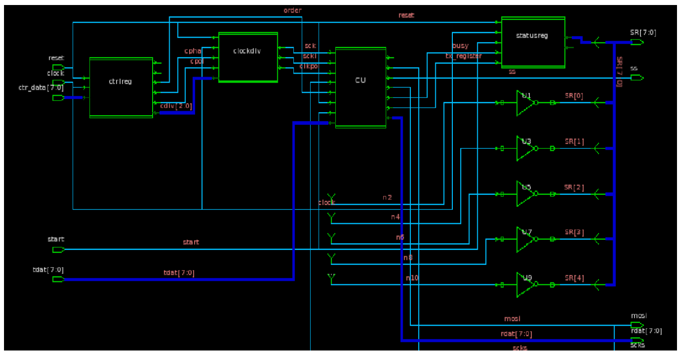


Fig. 9 (b): Block Diagram of SPI after Synthesis Process from Design Compiler (DC) Analysis

i. Report of Constraint for DC Analysis of SPI

As depicted in Table 6 is the constraint result from the DC analysis. The max_area result shows that the area was 27750.93 (VIOLATED) was violated. This result does not imply an error in the design. This is because the set up for the constraints value of the area was written as 0. This indicates the area for the design is 27750.93.

TABLE VI
REPORT OF CONSTRAINTS FOR THE SPI

Constraint	Cost
multiport net	0.00 (MET)
max_transition	0.00 (MET)
max_capacitance	0.00 (MET)
max_delay/setup	0.00 (MET)
critical_range	0.00 (MET)
min_delay/hold	0.00 (MET)
max_area	27750.93 (VIOLATED)

Table 7 refers to the power for the SPI. The main purpose of power report is that to show the cell internal power and the net switching power. Both of those values will contribute to the total dynamic power. The cell internal power will give out 42.5780 uW which is 90% of the total dynamic power where the net switching power will give out 4.5494 uW which is 10% of the total dynamic power that was going to be used. Meanwhile cell leakage Power will give out 29.3476 nW. All these values came from the setting that has been preset in Astro software.

TABLE VII
REPORT OF POWER FOR THE SPI

Global Operating Voltage	1.62
Power-specific unit information:	Voltage Units = 1V Capacitance Units = 1.000000pf Time Units = 1ns Dynamic Power Units = 1mW (derived from V,C,T units) Leakage Power Units = 1pW
Cell Internal Power	42.5780 uW (90%)
Net Switching Power	4.5494 uW (10%)
Total Dynamic Power	47.1274 uW (100%)
Cell Leakage Power	29.3476 nW

As tabulated in Table 8, it is the Quality of Result (QoR) report for the SPI. It shows the cell area and the design area of the SPI. From this report, it also shows that there are no nets with violation with the total number of nets which are 232 and required area for the design is about 27,750 μm^2.

TABLE VIII
REPORT OF QOR FOR THE SPI

Timing Path Group 'clock'	Levels of Logic:	3.00
	Critical Path Length:	1.26
	Critical Path Slack:	47.44
	Critical Path Clk Period:	100.00
	Total Negative Slack:	0.00
	No. of Violating Paths:	0.00
	Worst Hold Violation:	0.00
	Total Hold Violation:	0.00
	No. of Hold Violations:	0.00
Cell Count	Hierarchical Cell Count:	6
	Hierarchical Port Count:	90
	Leaf Cell Cou nt:	175
	Buf/Inv Cell Count:	26
	CT Buf/Inv Cell Count:	0
Area	Combinational Area:	1726.401624
	Noncombinationa Area:	4277.750484
	Net Area:	21746.774780
Cell Area:	6004.152109	
Design Area:	27750.926889	
Design Rules	Total Number of Nets:	232
	Nets With Violations:	0
Compile CPU Statistics	Resource Sharing:	0.04
	Logic Optimization:	0.22
	Mapping Optimization:	0.40
Overall Compile Time:	4.67	

Table 9 describes the report of timing for SPI. In this report it also gives out the overall compilation time after gone thru the optimization process for the logic and mapping optimization. Overall, there are no violations and errors in the compilation process.

TABLE IX
REPORT OF TIMING FOR SPI

Maximum timing	data required time	99.50
	data arrival time	-52.06
	slack (MET)	47.44
Minimum Timing	data required time	50.49
	data arrival time	50.55
	slack (MET)	0.06

In summary, these reports met and run according to the constraints that was set to the design. The files has been saved and utilized to the next process - Astro for the floor planning, placement and routing of the design.
big, but it also come with a benefit such as all parts was connected, and it was easy for each part to be placed into the floor plan of this SPI. As shown in figure above, the floor plan includes the rectangular rings which are VDD and VSS. The width and also height of the SPI was actually bigger than it needs to be because to avoid any congestion occurs.
Fig. 11 shows the standard cells of the SPI was placed inside the floor plan that was created for it and also routed accordingly.

V. CONCLUSION

To conclude, the design and simulation of the IP core of APB Interfacing with SPI were successfully simulated and synthesized using ModelSim, QuartusLite 16, design compiler (DC analysis) and Silterra Astro. The findings revelaled that the SPI interfacing to send or receive data from a single slave and efficient APB-SPI controller with flexible data width and frequency is established for maximum frequency of 16 MHz. The modes of SPI have performed four modes that is mode one, mode two, mode three and mode four. Also, the design and simulation of the IP core of APB Interfacing with SPI were completed for the gdsii file.

REFERENCES

[1] M. K. Saxena, E. Bhatnagar, N. Jaiswal, M. Parab, and S. N. Kulkarni, “Reconfigurable architecture for IP peripherals,” *Int. Conf. Signals Electron. Syst. ICSES’10 - Conf. Proceeding*, pp. 347–350, 2010.

[2] C. Mac Donald, E. Kilbride, and S. Philippe Alves, “Serial Peripheral Interface Device Emulation Routine for the MC68340,” 2004.

[3] N. B. Mohd Noor and A. Saparon, “FPGA implementation of high speed serial peripheral interface for motion controller,” ISIEA 2012 - 2012 IEEE Symp. Ind. Electron. Appl., pp. 78–83, 2012.

[4] F. Naqvi, “Design and Implementation of Serial Peripheral Interface Protocol Using Verilog HDL,” *Int. J. Eng. Dev. Res.*, vol. 3, no. 3, pp. 1–5, 2015.

[5] M.-C. Tuan, S.-L. Chen, Y.-K. Lai, C.-C. Chen, and H.-Y. Lee, “A 3-wire SPI Protocol Chip Design with Application-Specific Integrated Circuit (ASIC) and FPGA Verification,” *Proc. 3rd World Congr. Electr. Eng. Comput. Syst. Sci.*, vol. 1, pp. 1–7, 2017.

[6] C. D. Systems, “32-bit APB Serial Peripheral Interface (SPI) IP.”

[7] A. K. Oudjida, M. L. Berrandjia, A. Liacha, R. Tiar, K. Tahraoui, and Y. N. Alhoumays, “Design and test of general-purpose SPI master/slave IPs on OPB bus,” *2010 7th Int. Multi-Conference Syst. Signals Devices, SSD-10*, pp. 0–5, 2010.

[8] M. Sl and A. R. Aswatha, “SPI Slave Controller for AMBA Based SOC,” vol. 3, no. 8, pp. 60–66, 2016.

[9] Z. Zhou, Z. Xie, X. Wang, and T. Wang, “Development of verification envioronment for SPI master interface using SystemVerilog,” *Int. Conf. Signal Process. Proceedings, ICSP*, vol. 3, pp. 2188–2192, 2012.

[10] S. Sharma and S. M. Sakthivel, “Design and verification of AMBA AXI3 protocol,” *Lect. Notes Electr. Eng.*, vol. 469, no. 21, pp. 247–259, 2018.

[11] Muhammad Hafeez Sabparie et.al, "IP Core of Serial Peripheral Interface (SPI) with AMBA APB Interface," 2017 National Symposium on VLSI Technology and System on Chip, 12 - 14 Disember 2017, Penang, Malaysia

[12] M. Hafeez and A. Saparon, “IP core of serial peripheral interface (SPI) with AMBA APB interface,” *ISCAIE 2019 - 2019 IEEE Symp. Comput. Appl. Ind. Electron.*, no. Ic, pp. 55–59, 2019.

Muhammad Hafeez bin Sabparie has completed her BSc in Electrical Engineering from Universiti Teknologi MARA Shah Alam, Selangor andpursuing his MSc in Electrical Engineering.

Emilia Noorsal received her B.Eng. (Hons) and MSc degrees from the Universiti Sains Malaysia, Malaysa, in 1998 and 2005, respectively. In April 2014, she obtained her PhD in Digital Stimulation Control (ASIC) for Biomedical Engineering application from the Institute of Microelectronics, University Ulm, Germany. Currently, she is a senior lecturer in Faculty of Electrical Engineering, UiTM Penang, Pulau Pinang, Malaysia. Her research interests include VLSI digital ASIC, mixed-signal circuit design, FPGA and electronics for biomedical applications (FES).

Suhana Sulaiman received her B.Sc in Electrical Engineering from University of Utah, Salt Lake City, USA in 1996 and M.Sc in Microelectronics from University of Newcastle Upon Tyne, UK in 2002. Her Ph.D. degree in Electrical Engineering from Universiti Teknologi MARA, Shah Alam Selangor, Malaysia in 2013. Currently, she is a senior lecturer in Faculty of Electrical Engineering, UiTM Shah Alam, Selangor. Her research interests include RFIC, microelectronics, electronics circuit and system and microfabrication.

Azilah Saparon, is an Associate Professor at Faculty of Electrical Engineering, UiTM Shah Alam, Selangor, Malaysia. She received her B.E.E from Gannon University, Erie, Pennsylvania, U.S.A in 1989, M.Sc in Electrical Engineering from Washington State University at Pullman,U.S.A in 1995 and a Ph.D in Electronic and Electrical Engineering from Loughborough University, Leicestershire, U.K 2006. Her research interests include reconfigurable system, micro-computer architecture and video coding.