International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-1, November 2019

Area Optimization using Structural Modeling for Gate Level Implementation of SPI for Microcontroller

Amrut Anilrao Purohit, Mohammed Riyaz Ahmed, R. Venkata Siva Reddy

Abstract*: The need for miniaturization has been the driving force in chip manufacturing. The proliferation of IoT, robotics, consumer electronics and medical instruments pose unprecedented demands on the embedded system design. The area optimization can be achieved either by reducing the size of transistors or by optimizing (reducing) the circuit at the gate level. The first solution has attracted many researchers while the later has not been explored to its full potential. The aim is to design a System on Chip (SoC) to satisfy the dynamic requirements of disruptive technologies while occupying the lesser area. The design and testing of communication interfaces such as Serial Peripheral Interface (SPI), Inter-IC Communication (I2C), Universal Asynchronous Receiver and Transmitter (UART) are very crucial in the area optimization of microcontroller design. Since SPI being an important communication protocol, this work reports the preliminary research carried in the design and verification of it. In this work, Verilog is used for the design and verification of the SPI module. The results show that there is a drastic reduction in the number of Look-Up-Tables (LUTs) and slices required to build the circuit. We conclude that sophisticated optimization techniques of the circuit at the gate level has the potential to reduce the area by half.

Keywords: Area Optimization, Communication Protocol, Serial Peripheral Interface, Structural Modeling.

I. INTRODUCTION

The ever-increasing proliferation of IoT devices has posed demand on processing devices (can be of 5 different categories namely Microprocessor, Microcontroller, Digital Signal Processor, Field Programmable Gate Array, Application Specific Integrated Circuits) to accommodate the varied type of peripheral devices such as sensors and actuators [1] and [2]. Though microcontrollers are less power hungry and work with lower clock frequency their massive adoption in the automation industry can be owed to their ability to interface (to a peripheral device) with minimal hardware resources. The presence of various modules (Analog-to-Digital Converter, memories, communication modules, timers/counters, Digital-to-Analog Converter, etc.) within the microcontroller will subdue the need to have additional hardware for interfacing [3], [4] and [5]. A typical master-slave arrangement between a microcontroller and peripheral device is depicted in Fig.1.

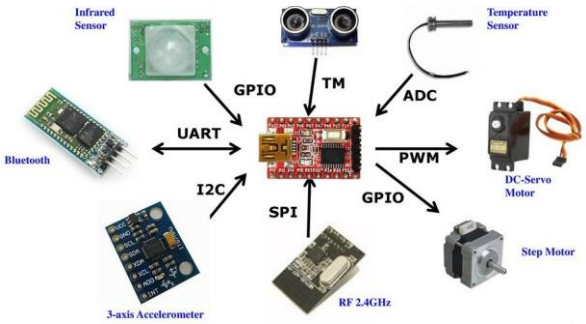

Fig. 1. Microcontroller with various peripherals including SPI, I2C, UART

Designing an efficient communication protocol which governs the communication between the processing device and the peripheral(s) is the crux of the matter. The seamless communication happens either in the synchronous (master-slave) fashion or asynchronous fashion. Serial Peripheral Interface (SPI), Inter-IC Communication (I2C), and Universal Asynchronous Receive Transmit (UART) are quite a bit slower than protocols like Universal Serial Bus (USB), Ethernet, Bluetooth, and Wi-Fi, but they are a lot simpler and manage with less hardware and system resources [6] and [7]. For low-data rates, SPI, I2C, and UART are ideal for communication between microcontrollers or microcontroller and sensors/actuators [8], [9] and [10].

Fig. 2. Trends graph showing the dominance of SPI among all interface communication protocols: A comparison since 2004, obtained from Google trends.

SPI has been investigated either for reducing the power consumption or for increasing the speed. One more domain which has got equal attention is of area reduction.

Revised Manuscript Received on November 30, 2019.
***** Correspondence Author
Amrut Anilrao Purohit*, Research Scholar, Department of Electronics and Communication Engineering, VTU, Belagavi, India.
Assistant Professor, School of Electronics and Communication Engineering, REVA University, Kattigenahalli, Yelahanka, Bengaluru, KTK India-560064.
Mohammed Riyaz Ahmed, and **R Venkata Siva Reddy**, are with School of Electronics and Communication Engineering, REVA University, Kattigenahalli, Yelahanka, Bengaluru, KTK India-560064.

Though many approaches have been reported to date, structural modeling remains unexplored [11]. This work proposes to explore and exploit structural modeling of the SPI module with an aim to reduce the number of Look-Up-Tables (LUTs) occupied, thereby reducing the area on chip in the analog front-end design.

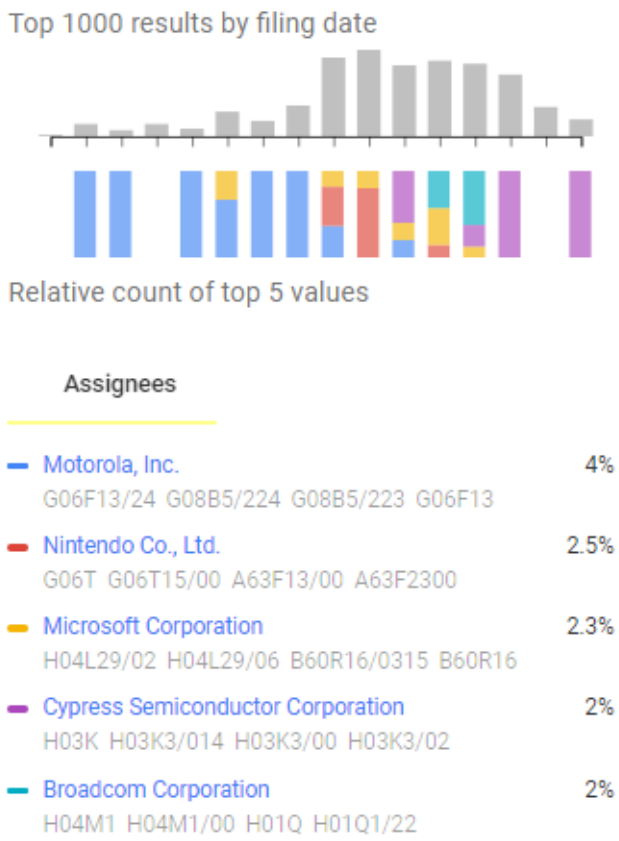

Fig. 3. Evolution of patents in the field of SPI along with the share of top 5 organizations, Google patents since 1974

Trends graph showing the dominance of SPI among all interface communication protocols: A comparison since 2004, obtained from Google trends is shown Fig.2 Evolution of patents in the field of SPI along with the share of top 5 organizations, Google patents since 1974 is shown in Fig. 3.

The remainder of the paper is arranged as follows: Section 2 provides a brief background of the SPI communication protocol while outlining its history and evolution. It also has a comparative study along with recommendations in Table 1. Section 3 describes the proposed SPI protocol architecture, along with experimental set-up. Section 4 presents the results along with the discussion. The obtained results are compared with the previous approaches and validated here. Finally, the paper concludes in section 5. To reach to a broader audience, the paper also provides research gaps and future scope in this section.

II. BACKGROUND

The interfacing of various digital and analog circuits on to a single board plays a vital role in the Embedded System Design. While communicating between the 2 or more devices, the protocol standards are to be decided and maintained. The various and most common interfaces used are SPI, I2C, UART, USB, Ethernet etc.

An interface is a connecting point between two entities, while protocols refer to the soft rules followed to have an exchange of information via that interface. Interface protocol specifies the way data is delivered and interpreted. Interface (communication) protocols facilitate chip-to-chip (or board-to-board) connectivity in system designs. A protocol is ranked based on the data throughput, latency, redundancy, interoperability, or combination of them.

The communication can happen within the board (via intra-communication protocols viz. SPI, I2C, and CAN), or between two systems (via inter-communication protocols viz. UART, VGA, HDMI, and USB). The board-to-board and box-to-box communication happen in serial fashion while on-chip communication can be serial or parallel type. Serial communication happens bit-by-bit hence is slower compared to parallel which communicates in byte(s) at a time. Serial communication is preferred in the industry since it requires fewer lines and is used for long-distance communication, unlike parallel communication where fast communication is possible at the cost of multifold infrastructure.

In serial communication protocols, the communication that complies with a clock signal is termed as synchronous. High data rates, high reliability with zero bit-error-rate makes it popular as compared to the random, slower and error-prone asynchronous communication.

UART (an Asynchronous communication protocol) was the earliest serial communication protocol to be introduced in 1971 by Gorden Bell of Digital Equipment Corporation. The year 1979 saw the introduction of the famous synchronous communication protocol SPI by Motorola [12]. This overcame the limitations of asynchronous data transfer in serial communication while enhancing the data rate considerably with better reliability [13]. A typical SPI protocol will have a master (usually microcontroller) which generates the clock for synchronization. MOSI (Master-Out-Slave-In) and MISO (Master-In-Slave-Out) are two data lines via which slave(s) can receive and send data respectively. The fact that the slave can be a simple shift register makes SPI a popular choice when compared to UART.

When Philips introduced I2C in 1982, it grabbed the attention of many researchers due to its reduced interfacing hardware, especially wires [14], [15], and [16]. While a single bidirectional data line of MIMO (Master Input and Master Output) replaced both MOSI and MISO; the style of transmission starting from device address eliminated the necessity of the slave select line. For N slaves an SPI needs to have 3+N lines for communication while 2 wires will suffice in I2C for any number of slaves (up to 127). But, soon I2C lost its sheen due to complex set-up, reduced speed and limited accommodation to slaves. The fact that SPI will have N lines (plus 3) for N slaves makes buffering and isolation easy which is not possible in I2C. Thus until there is a strict constraint to have less number of wires, SPI is the best way to go. Simple set-up, buffering and isolation provision, better data rates, and the transmission style of 'just have to get the clock and data lined up' makes it the most preferred communication protocol in the industry.

Retrieval Number: A4588119119/2019©BEIESP
DOI: 10.35940/ijitee.A4588.119119
Journal Website: www.ijitee.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Table 1. Scrutiny of SPI and I2C with respect to various parameters

.Item	SPI (Serial Peripheral Interface)	I2C (Inter IC Communication)	Recommendation
Origin	Introduced by Motorola in 1979	Introduced by Philips (now NXP) in 1982.	SPI is senior
Transfer type	Full duplex communication protocol with dedicated lines for transmission and reception	Half duplex communication protocol i.e. data bus is bidirectional.	SPI is faster
Interface type	Needs three + N wires for communication, i.e. MOSI, MISO, SCL and N Chip-select pins	Uses only two wires for the communication, i.e. one wire for the data and the other for the clock.	SPI if no constraint on wires
Addressing type	Addresses using hardware pin select, you have to select the slave select pin for the communication.	Is address based bus protocol, you have to send the address of the slave for the communication.	SPI can address infinite devices*
Flow control	Does not have acknowledgement	We get the acknowledgment bit after each byte.	I2C for flow-control
Multi-master	Is not a multi-master communication protocol, hence arbitration is not applicable	Is multi-master communication protocol, has the feature of arbitration	I2C for Plug-&-Play
Clock stretching	Not the feature of SPI.	Suppresses the clock to stop the communication if slave does not respond.	I2C is reliable+
Protocol complexity	Simple, works with shift registers, hence low design cost and low power consumption	Slightly complex due to bidirectional data bus hence, higher design cost and more power hungry.	SPI for simple, economical and green design
I/O constraints	There is no requirement of pull-up resistor.	Works on wire and logic, requires pull-up resistors.	SPI is constraints free
Noise	more susceptible to noise	less susceptible to noise	I2C for long distance
Overhead	does not have a start and stop bits	Has some extra overhead due to start and stop bits	SPI has no overhead

* The number of slave devices, SPI can address is dependent on N select lines available on master.
+ I2C ensures that data sent is received by the slave device.

III. SPI PROTOCOL ARCHITECTURE

Serial Peripheral Interface (SPI) allows full-duplex synchronous data communication between the FPGA and other peripheral devices. In SPI, there is no predefined limit on the maximum data rate and is not confined to any one particular addressing scheme. The communication is similar to the Unit Datagram Packet (UDP) protocol where the data is transferred and the transmitter is least bothered about the reception. This kind of communication does not embrace the acknowledgment mechanism. The SPI master does not know whether a slave exists unless something additional is done outside the SPI protocol.

The input and output blocks are used to interface with the processor and the peripheral devices. The clock generator and the receiver block is used to generate the clock in the master, sent through the SCK pin. While the clock is received on the SCK during the slave mode of operation. The memory block is used to store the contents of the control register, data registers, and status register.

There are 2 separate data registers one for the transmitting and one for receiving, but both have the same address. Whenever the data is written, it is written into the data output register. While the data is read, it is read from the data input register. The data cannot be read from the data output register and written into the data input register. In slave mode, the interrupt is cleared automatically, once the data is read from the data input register. While in the master mode, the interrupt has to be cleared by activating the spi_flag_clr bit. The SIPO and PISO registers are used to convert the data from serial and parallel and vice versa respectively. The description of proposed design is provided in Table 2.

Table 2.Pin Description of Proposed SPI Module.

Pin	Size	Direction	Description
Din	8-bit	Input	to receive the data from the processing block and further convert it into serial data with the PISO shift register
Clk	1-bit	Input	Clock generated by the processor for the SPI module
Sdi	1-bit	Input	for receiving the data from the peripheral devices.
spi_flg_clr	1-bit	Input	to clear the interrupt flag bit, which is triggered when the data transmission/reception is complete.
spi_l	1-bit	Input	write the 8-bit data into the internal data buffer.
Ss	1-bit	Output	to activate the chip for communication during SPI slave mode operation.

Intr	1-bit	Output	is the interrupt flag bit, which asserts upon the completion of data transmission/reception.
Sdo	1-bit	Output	to transmit the data to the peripheral device serially. This data is received on din line of the slave.
data_out	1-bit	Output	to transmit the data parallelly to the processor, which is received from the peripheral device on the sdi line serially.
spi_baud	8-bit	Input	to set the data rate during the master mode operation.
spi_con1	8-bit	Input	to specify the SPI module control or behavior
spi_status	8-bit	Input	to specify the status of the operation in the SPI module.
Sck	1-bit	In-out	to send the clock out during SPI master mode operation and receive the clock in during the SPI slave mode operation

The SPI module can be enabled or disabled and made to operate in master or slave mode based on the configuration of the control register. Here we have configured the control register to decide the phase and the edge of the clock during which the data communication takes place along with the LSB or MSB being transmitted or received first. The enabling and disabling of the interrupts is controlled through the control register. The structure of Control Register (SPI_CON1) with default bit values is provided in Table 3. The minimum and the maximum baud rates are according to the equations (1) and (2). The Baud Rate Register (BAUD_CON) bit structure is as shown in Table 4. From the SPI status register the information on the data transfer has completed or not can be inferred. The bit mapping of the same is provided in Table 5. The block diagram of the proposed architecture is shown in Fig.4.

Table 3. Control register (SPI_CON1)

Bit	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	*	LSBFE
Default	0	0	0	1	0	0	0	0

* Reserved for future use.

Description of **SPI control register**

SPIE: Receive interrupt enable bit of SPI module.

SPE: Enable SPI module.

SPTIE: Transmission interrupt enable bit of SPI module.

MSTR: Indicates the mode of operation of module i.e. either in Master or in Slave.

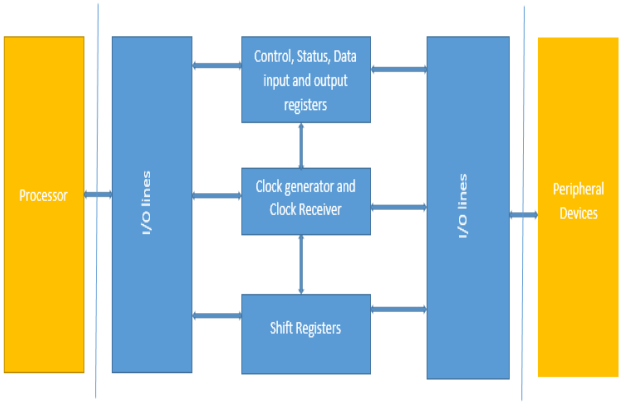

Fig. 4. Block Diagram of the proposed architecture

CPOL: Clock POLarity bit. The clock polarity is inverted for '1'.

CPHA: Clock PHAse bit. Decides the occurrence of latching and shifting, when set to '0' the event occurs on rising edge.

LSBFE: Least Significant Bit First Enable bit. When set to '1' LSB will sent out first.

Table 4. Baud Rate register (BAUD_CON)

Bit	*	SPPR2	SPPR1	SPPR0	*	SPR2	SPR1	SPR0
Default	0	0	0	1	0	0	0	0

* Reserved for future use.

Description of SPI Baud Rate register

SPPR[2:0]: Baud rate Pre-selection bits.

SPR[2:0]: Baud rate selection bits.

There will be a divisor to divide clock as below.

$$BaudRateDivisor = (SPPR + 1) * 2\ (SPR+1) \ldots\ldots\ldots (1)$$

The final baud rate from baud clock generator is

$$BaudRate = BusClock/(SPPR + 1) * 2\ (SPR+1) \ldots\ldots (2)$$

Table 5. Status (SPI_STATUS)

Bit	*	*	SPTEF	*	*	*	*	*
Default	0	0	0	1	0	0	0	0

* Reserved for future use.

Description of SPI Status register

SPTEF: set's the interrupt when the transmit data register is empty. Clearing this bit after writing to data register starts next transmission.

IV. SIMULATION RESULTS

Xilinx 14.2 ISE is used to compile the HDL code in Verilog. The XC5VLX-3FF324 device used in the compilation at a speed of -3. The program is written using Verilog structural modelling. The result is obtained from the design summary with respect to the number of LUT's used, the number of slices used is obtained as in Fig.5.

The bottom-up approach is used to design the entire system to start with the design and testing of all the gate, MUXes and so on. The synthesis design is done at technology level as well as at the RTL level with timing report analysis. The Fig.6(a) shows the RTL schematic and Fig.6(b) depicts the technology schematic of the proposed design.

The proposed design is simulated and verified using ISim for its timing in master and slave modes of operation which is represented in the Fig.7(a) and Fig.7(b) respectively. The Table 6. gives the comparative study or various works along with the proposed work.

spi_top Project Status (08/18/2019 - 21:06:35)			
Project File:	spi_top.xise	**Parser Errors:**	No Errors
Module Name:	spi_top	**Implementation State:**	Placed and Routed
Target Device:	xc5vlx30-3ff324	**•Errors:**	No Errors
Product Version:	ISE 14.2	**•Warnings:**	155 Warnings (155 new)
Design Goal:	Balanced	**•Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**•Timing Constraints:**	
Environment:	System Settings	**•Final Timing Score:**	0 (Timing Report)

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice LUTs	162	19,200	1%	
Number used as logic	162	19,200	1%	
Number using O6 output only	131			
Number using O5 and O6	31			
Number of occupied Slices	87	4,800	1%	
Number of LUT Flip Flop pairs used	162			
Number with an unused Flip Flop	162	162	100%	
Number with an unused LUT	0	162	0%	
Number of fully used LUT-FF pairs	0	162	0%	
Number of slice register sites lost to control set restrictions	0	19,200	0%	
Number of bonded IOBs	37	220	16%	
Average Fanout of Non-Clock Nets	3.73			

Fig. 5. Design summary of the proposed SPI module. Number of LUTs used indicates the amount of area required for fabrication on the chip. Lesser the number of LUTs, smaller the area required. In the proposed design the number of LUTs used is 162. The total input output lines in the proposed design is 37.

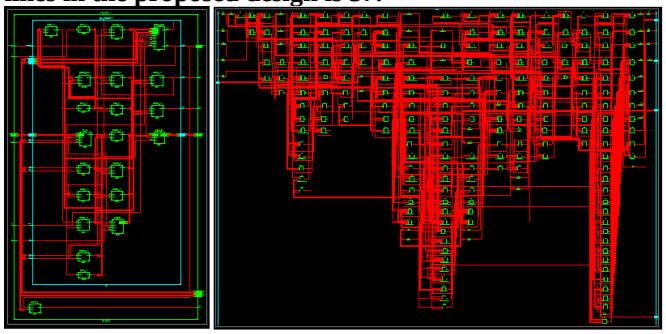

Fig 6(a). RTL Schematic of the proposed design, Fig 6(b). Technology Schematic of the proposed design

Table 6. Comparision of various works along with proposed SPI

Work	Module	LUT	Slices
[5]	SPI	712*	390
[7]	SPI-Master	-	232
[7]	SPI-Slave	-	141
[14]	SPI-Slave	-	141
[17]	SPI	1049*	-
[17]	SPI-Slave	219*	-
Our work	SPI	162#	87
Our work	SPI-Slave	64#	19

* Represents 4-bit LUTs.
\# Represents 6-bit LUTs.

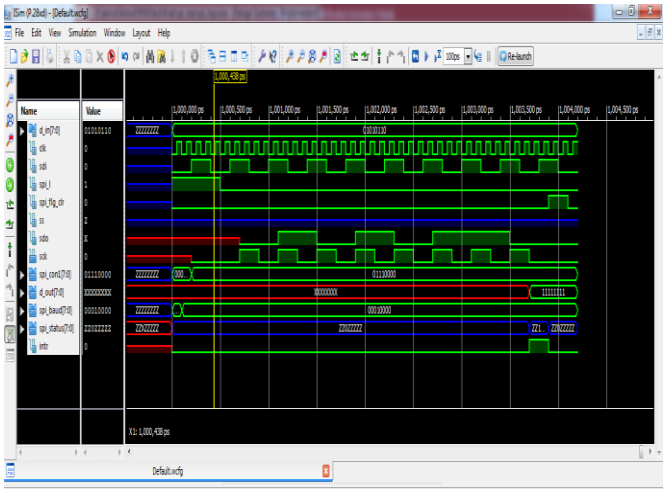

Fig. 7(a) Timing diagram of the master mode operations

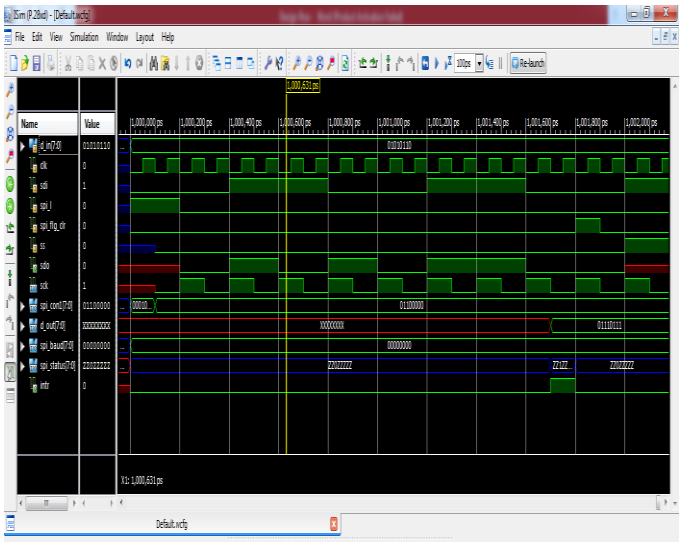

Fig. 7(b) Timing diagram of the slave mode operations

V. CONCLUSION

The disruptive technologies in embedded systems and the varied demands of various use-cases have demanded for unprecedented optimization of various parameters including area. While many options have been explored for transistor level optimization, yet gate level optimization remains less probed. Here we have proposed a new SPI module which achieves area reduction by optimizing the circuit. In our approach the total LUTs used are 162, which is a remarkable achievement as compared to the existing solutions. The gate level implementation shows potential to further reduce other parameters as well. The impact of this approach can also be examined on power and delay in future.

ACKNOWLEDGMENT

The authors would like to thank Rukmini Educational Charitable Trust and REVA University for providing necessary infrastructure to carry out this work.

REFERENCES

1. Gal, R., Golda, A., Frankiewicz, M., & Kos, A. (2011, June). FPGA implementation of 8-bit RISC microcontroller for embedded systems. In Proceedings of the 18th International Conference Mixed Design of Integrated Circuits and Systems-MIXDES 2011 (pp. 323-328). IEEE.
2. Sandya, M., & Rajasekhar, K. (2012). Design and verification of serial peripheral interface. Int. J. Eng. Trends Technol, 3(4), 522-524.
3. Anand, N., Joseph, G., Oommen, S. S., & Dhanabal, R. (2014, January). Design and implementation of a high speed Serial Peripheral Interface. In 2014 International Conference on Advances in Electrical Engineering (ICAEE) (pp. 1-3). IEEE.
4. KUMAR, K. A., & KRISHNA, M. S. (2015). Design and Functional Verification of A SPI Master Slave Core using UVM.
5. Chen, R., Huang, S. Z., Lin, W., & Li, L. (2009, December). Design and implementation of a reused interface. In 2009 First International Conference on Information Science and Engineering (pp. 2603-2605). IEEE.
6. Kaneriya, H., & Jagpat, S. (2015). Design of I2C Master with Multiple Slave. IJRSTCC, 3, 2890-2893.
7. Oudjida, A. K., Berrandjia, M. L., Liacha, A., Tiar, R., Tahraoui, K., & Alhoumays, Y. N. (2010, June). Design and test of general-purpose SPI Master/Slave IPs on OPB bus. In 2010 7th International Multi-Conference on Systems, Signals and Devices (pp. 1-6). IEEE.
8. Jain, N., & Jain, P. Implementation of Serial Peripheral Interface Protocol for LCD.
9. Leens, F. (2009). An introduction to I 2 C and SPI protocols. IEEE Instrumentation & Measurement Magazine, 12(1), 8-13.
10. Liu, T., & Wang, Y. (2011, June). IP design of universal multiple devices SPI interface. In 2011 IEEE International Conference on Anti-Counterfeiting, Security and Identification (pp. 169-172). IEEE.
11. Kundu, M., & Kumar, A. (2014). Implementation of Low Power SPI Protocol with Clock Domain Crossing. IJRIT, 1(2), 26-29.
    Mueller, S. (2003). Upgrading and repairing PCs. Que Publishing.
12. Agarwal, V. I. V. E. K., Arya, H. E. M. E. N. D. R. A., & Bhaktavatsala, S. H. I. V. A. R. A. M. (2009). Design and development of a real-time DSP and FPGA-based integrated GPS-INS system for compact and low power applications. IEEE Transactions on Aerospace and Electronic Systems, 45(2), 443-454.
13. Oudjida, A. K., Berrandjia, M. L., Tiar, R., Liacha, A., & Tahraoui, K. (2009, December). Fpga implementation of i 2 c & spi protocols: A comparative study. In 2009 16th IEEE International Conference on Electronics, Circuits and Systems-(ICECS 2009) (pp. 507-510). IEEE.
14. Gopal, N. (2015). Spi controller core: Verification. SSRG International Journal of VLSI & Signal Processing (SSRG-IJVSP), 2(5), 1-7.
15. Eswari, B., Ponmagal, N., Preethi, K., & Sreejeesh, S. G. (2013, April). Implementation of I 2 C master bus controller on FPGA. In 2013 International Conference on Communication and Signal Processing (pp. 1113-1116). IEEE.
16. Mukthi, S. L., and A. R. Aswatha. "Design and Implementation of an Interfacing Protocol between I2C and APB for an AMBA Based SOC."

AUTHORS PROFILE

Amrut Anilrao Purohit is a research scholar in VTU. He completed B. E. in ECE from SVCE, Bangalore and M. Tech in Electronics from Sir. M. V. I. T., Bangalore from VTU, Belagavi IN 2006 and 2011 respectively. He has 12 years of Industry, Teaching, Research and Administrative Experience. His patent applied in year 2017 has been published in the Indian Journal of Patents.

Dr Mohammed Riyaz Ahmed is an Associate Professor in School of ECE. He is a Senior member of IEEE, USA Life member of ISTE, India. He is a recognized (by Texas Instruments) mentor and his student projects explore the areas of Smart Cities (from smart grid to smart transportation), Technology Intervention for Elderly people, Artificial Intelligence, and Analog VLSI design.

Dr. R Venkata Siva Reddy has 27 years of teaching experience. He has graduated from Gulbarga University and done his masters in Power Electronics. His areas of research are Digital System Design, Embedded Systems Design. He is a senior member of IEEE and a valued IEEE member for last 18 years and Fellow of IETE (FIETE) India. He is an active HAM and have established Amateur Radio Club in REVA University with call sign VU3UFF.