International Journal of Engineering Research and Science & Technology

IJERST

ISSN : 2319-5991



www.ijerst.com

Email: editor@ijerst.com or editor.ijerst@gmail.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Vol. 21, Issue 2, 2025

Design of High Speed I2C Interface Protocol ForEnabling Rapid Data Exchange

Ande Meena[1], Hadnoor Mahipal Reddy [2], Gangadhar Aravind Sai[3], Mr.K.Balasubramanyam [4]
[1,2,3] UG Scholar, Dept. of ECE, St. Martin's Engineering College, Secunderabad, Telangana, India, 500100
[4]Assistant Professor, Dept. of ECE, St. Martin's Engineering College, Secunderabad, Telangana, India, 500100
meena.ande9@gmail.com

Abstract:
The increasing demand for high-speed data communication in embedded systems has led to a focus on efficient protocols, with the I2C protocol emerging as a preferred solution. Traditional UART (Universal Asynchronous Receiver-Transmitter) communication often struggles with limitations in speed and the complexity of wiring, making it unsuitable for applications requiring rapid data exchange. This work proposes a novel design for a high-speed I2C interface protocol aimed at enabling efficient data exchange in modern electronic systems. The design emphasizes improved communication speed and reduced complexity through a single master-multiple slaves configuration and optimized data handling techniques. The proposed I2C protocol enhances system performance by allowing for simultaneous communication with multiple devices while maintaining data integrity and reliability. Key advantages of this approach include reduced pin count, increased data throughput, and enhanced scalability for future applications in complex systems.

Keywords: I2C protocol, UART communication, data exchange, single master-multiple slaves, optimized data handling, data integrity, reliability, pin count reduction, data throughput.

1. INTRODUCTION

The I2C, a popular serial communication protocol, facilitates communication between multiple devices on the same bus. It is widely used in applications such as sensors, memory devices, and other integrated circuits. The standard I2C speed modes include Standard Mode (up to 100 kHz), Fast Mode (up to 400 kHz), and High-Speed Mode (up to 3.4 MHz). To enable rapid data exchange, the focus is primarily on optimizing the design parameters for both hardware and software.I2C Protocol is a synchronous protocol that allows a master to initiate communication with a slave device. I2C protocol is a master-slave communication where the master provides the clock which becomes the data transfer rate or clock frequency. It is a bi-directional bus meaning the master can write to the slave and read from the slave it is a serial bus which means data is a clock and it is shifted bit by bit and there are two bus lines serial clock (SCL) and serial data (SDA).

As u see in the diagram three slaves are connected to the same I2C Master and there is two pull-up resistor which is required for the I2C device to communicate properly this is because the I2C protocol works on the SCL and SDA bus lines which are an open drain or open collector the transmitting device just lets go of the I2C bus to create logic one and pulls or drives the line the ground to create logic 0.
In I2C Protocol there are 5-speed categories including standard mode, fast mode, fast plus mode, high-speed mode, and ultra speed mode these speed categories range from 100 kHz to 5MHz, where standard mode is 100KHz, Fast mode is 400KHz, Fast mode plus is 1MHz, High-Speed Mode is 3.4MHz and Ultra-fast mode is 5MHz. 100KHz up to 1MHz are very similar, while 3.4 MHz needs some special consideration, and 5MHz being unidirectional requires even more special attention. But the most common speed categories for I2C Protocol are Standard mode, Fast Mode, and Fast mode plus these modes are easy to implement.

Designing a high-speed I2C interface protocol for enabling rapid data exchange requires a holistic approach, addressing both hardware and software considerations. Achieving high-speed communication within the constraints of the I2C protocol involves optimizing clock frequencies, minimizing bus length, managing impedance, implementing noise filtering, and ensuring clock synchronization. Thorough testing and validation are crucial to guarantee reliable performance under various conditions. Balancing speed with power consumption considerations and maintaining clear documentation contribute to a robust and efficient high-speed I2C design. I²C communication starts with the **master** generating a Start condition, followed by sending a 7-bit or 10-bit address with a Read/Write bit. The addressed slave responds with an acknowledgment (ACK), and data is transferred in 8-bit packets, each followed by an acknowledgment. The communication ends with a Stop condition, signaling the release of the bus. A write operation involves the master sending the slave address, followed by the register address and data, while a read operation consists of the master requesting data from the slave, which then transmits it.

2. LITERATURE SURVEY

Mohamad Khairi et.al [1] presented the design of the inter-integrated circuit (I2C) protocol with different types of features, such as combined messages, addressing modes, different data patterns and start addresses, clock frequencies, and types of modes between the field-programmable gate array (FPGA) and test card. Moreover, all these features was randomized and run for long hours. The FPGA and the test card respectively acted as master and slave. The design architecture comprised master and slave. The master generated a START condition, in which the serial data would transact between high to low levels and the serial clock would remain high. Then, the master also generated the STOP condition. Additionally, a few types of messaging modes, such as PIO read, PIO write, PIO write–read, and PIO read–write, were available. By contrast, the master also transferred and received data to or from slave devices by using different addressing modes. The implemented addressing modes were 7 and 10 bits. This paper also focused on randomizing the sent data byte and the start address. Particularly, data sending, reading, and writing operations were conducted and stimulated by capturing the signal using a logic analyzer.A stress test was performed by randomizing all the features and running for long hours (4 h). The stress test aimed to stress the IP and ensure the health of IP.

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

Malviya, Utsav Kumar et.al [2] developed the Mobile Industry Protocol Interface (MIPI) used the Camera Serial Interface (CSI-2) protocol for fast serial interface between the camera device and the mobile main processor. MIPI suggested that unidirectional transmission of images taken from the camera to the memory of the baseband processor using CSI-2 was faster than bidirectional. Hence, in MIPI, control commands between the camera sensor and the baseband processor were exchanged using another bidirectional interface known as the Camera Control Interface (CCI).The literature presented a new Tiny Interconnect Integrated Circuits (TI2C) protocol for exchanging control commands between the camera sensor and the mobile baseband processor. TI2C supported the new Camera Command Set (CCS), which was developed for multiple advanced camera sensor interfacings. On the other hand, CCI supported the Standard Mobile Imaging Architecture (SMIA) command register, which was developed according to single-camera interfacing. CCI used the I2C core protocol; however, TI2C used only necessary modules of the I2C core for camera control commands communication. This study showed the necessary elements required for command communication and additional components available in the I2C core that was omitted. TI2C was more area-optimized in comparison with CCI because of its application-specific design. This study discussed different methods used by researchers for the improvement of CSI-2 performance.

Nagaraju, C. H et.al [3] suggested the IIC protocol (Inter-Integrated Circuit) was a communication bus protocol that allowed many masters and slaves to communicate with each other. The I2C bus was a well- known and widely used communication technology in the electronics industry that was found in a wide range of electronic devices. Circuit complexity increased as new technologies were created at a rapid pace. These circuits had to be capable of self-testing. Built-In Self-Test (BIST) was a self-testing method that was utilized instead of expensive testing equipment. The design and creation of an Inter-Integrated Circuit (I2C) protocol that self-test were presented in this work. The I2C used the Verilog HDL language to achieve data transfer that was small, stable, and reliable.

Nayana, D. K. et.al [4] developed today's world reached a goal in which a complete module was instigated on a single chip called SOC (System on Chip). Protocols were required to combine these components. One such simple and emerging development protocol was the I2C protocol. I2C used only two initial fundamentals bidirectional buses of pull-up resistors configuration, i.e., Serial Clock Line (SCA) and Serial Data line (SDL). This protocol provided an efficient and simple method of data transaction among the devices and also supported multiple masters, a bi-directional serial bus used for quicker connections between devices with no loss of data.System Verilog and its verification were used for designing the real-time I2C controller with the help of the mentor graphics tool by creating a test bench environment. Code coverage and functional coverages were the two verification coverage matrices in this environment. The DUT achieved a 93.3% code coverage, and 100% functional coverage was achieved for the data and address parameters. The benefit of this protocol was that, by using the Ultra-Fast mode technique, data transfer rates was improved, and it had low wiring. This paper explained the design of an I2C protocol between a slave and master and its verification using the System Verilog language.

Chaithanya, A. Sainath et.al [5] developed,one of the foremost, well-liked, less sophisticated Serial communication standards, I2C; a bus protocol familiarly meant for the exchange of information among the peripherals residing on the constant circuit card, housed two-wires, i.e., data and clock, supporting duplex communication between multiple masters and slaves, considered as prominent and efficient in data transmission. The present work emphasized the I2C controller desiged for interfacing with slaves, a simple control register of I2C switches/cards where the data is written or scanned from.

Subsequently, I2C core implementation on Spartan 3E FPGA, where one of its on-chip peripherals, in this case, LCD, was treated as a slave for performing data transactions. The entire module was designed in Verilog HDL, functional checking was accomplished with the ISIM 10.0b simulator, followed by the design synthesis using Xilinx ISE14.4 tool.

Ghuse, Satish M. et.al [6] developed,I2C and SPI were the serial communication protocols commonly used for both intra-chip and inter-chip low/medium bandwidth data transfer. They support bidirectional data transfers at up to 100 Kbit/s in the standard mode, up to 400 Kbit/s in the Fast-mode, up to 1 M bit/s in the Fast-mode plus, or up to 3.4 M bit/s in the High-speed mode. These protocols had preferable speed and power consumption capabilities when implemented with different devices, but their speed was low when used with BIST or checksums.In the earlier systems, speed and delay were not taken into consideration, and the protocol was implemented as it was in the standard mode. Efforts were made to implement the I2C and SPI protocols efficiently so that the speed of data transfer increased, and the delay was reduced. To achieve this, a design was proposed in which a pipelined buffer was used between the Master and Slave to reduce the delay and introduce synchronization, resulting in an increase in the speed of data transfer as the delay decreased. The pipelined buffer led to speed enhancement for the critical paths.

Khelif, Mohamed Amine et.al [7] developed,in smartphones, and more generally in IoT devices, manufacturers focused their efforts on securing communications with the outside world that were more exposed to attacks while considering communications between secure components. By doing this, it resulted in internal communication buses with little or no security against attackers. I2C was the most used internal communication bus in IoT devices to communicate with sensors and memories. It was also used in recent smartphones to connect the Trusted Execution Environments (ARM TrustZone, Apple SEP, or Google Titan M) to a dedicated EEPROM memory that contained secret information such as encryption keys, anti-replay counter, or the boot ROM. In this paper, we proposed a non-invasive attack through a hardware trojan on the I2C bus, which allowed us to perform two attack scenarios: a heart-bleeding type attack which allowed retrieving additional information at each memory read, and a buffer overflow attack which allowed writing additional data in the memory at each write, resulting in modifying secret information such as passwords or counters. These attacks was performed on any device using the I2C bus. In smartphones, these attacks would allow the extraction of sensitive information stored in the secure EEPROM memory.

3. PROPOSED METHODOLOGY

I2C stands for Inter-Integrated Circuit. It is a bus interface connection protocol incorporated into devices for serial communication. It was originally designed by Philips Semiconductor in 1982. Recently, it is a widely used protocol for short-distance communication. It is also known as Two Wired Interface(TWI).I2C combines the best features of SPI and UARTs. With I2C, you can connect multiple slaves to a single master (like SPI) and you can have multiple masters controlling

International Journal of Engineering Research and Science & Technology
Vol. 21, Issue 2, 2025
ISSN 2319-5991 www.ijerst.com

single, or multiple slaves. Figure 1 shows the proposed i2c protocol architecture diagram. The detailed operation given as follows

Step 1: Consider the data input, size N bits: When transmitting a data payload of size N bits over I2C, the master device initiates communication with a start condition and addresses the intended slave device. The data payload, broken into bytes, is then transferred from the master to the slave, with acknowledgment after each byte. Once the transmission is complete, the master sends a stop condition to signal the end. Throughout the process, synchronization and error checking mechanisms ensure reliable communication between thedevices.

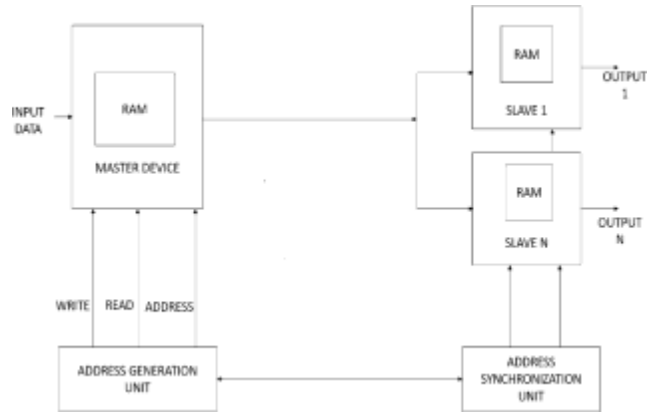


Figure 1: Proposed I2C Protocol.

Step 2: Start the environment, write enable will be active high. Random data counter starts generating address sequences: In the enabled environment, activate the write enable signal, which operates in an active-high state. Initiate the random data counter to commence generating address sequences. Ensure that the address sequences adhere to the protocol specifications for seamless communication. Monitor the data counter's output to synchronize with the transmission process effectively. Maintain compatibility with the system's overall architecture for smooth operation.

Step 3: Perform writing operation into SRAM memory with respect to address data will be stored: Initiate communication by sending a start condition and addressing the SRAM device on the I2C bus. Transmit the memory address where the data will be stored, followed by sending the actual data to be written into the specified memory address. Verify acknowledgment from the SRAM device after each byte transmission to ensure successful data storage. Conclude the operation by sending a stop condition to finalize the write process and complete the data transfer into the SRAM memory.

Step 4: Disable write enable, and enable read signal: Disable the write enable signal to prevent further write operations, ensuring data integrity. Activate the read signal to enable reading from the SRAM memory. Adjust the control signals to maintain proper synchronization with the memory access process. Validate the timing and signal transitions to ensure accurate data retrieval. Confirm compatibility with the system's operational requirements for seamless memory access.

Step 5: Generate acknowledgment(ack) signal to initialize receiver operation: The acknowledgment (ACK) signal is generated by the receiver to indicate the status of data reception. When the receiver successfully receives data, it sends an ACK signal, indicating that the data transmission was successful. Conversely, if the receiver does not receive any data or encounters an error, it sends a No ACK signal, denoting that no data was received or that an error occurred during transmission. This bidirectional signaling mechanism ensures proper synchronization between the transmitter and receiver, facilitating reliable communication over the I2C bus.

- No Ack=1(no data received)
- Ack=1(data received)

Step 6: select the slave devices based on acknowledgment signal: When the master sends the address of a slave device, it awaits an acknowledgment signal from the addressed slave. If the slave device acknowledges the address, it indicates readiness for communication, and the master proceeds with data transfer. Conversely, if no acknowledgment is received, the master attempt communication with another slave device or handle the lack of acknowledgment according to the communication protocol. This acknowledgment-based selection process ensures effective communication with responsive slave devices on the I2C bus.

Step 7: Start the data reception and store the received data frame into SRAM with address synchronization mechanism: Initiate data reception by configuring the master device to receive data from the designated slave device on the I2C bus. Implement a mechanism to synchronize the received data frame with the desired memory address in the SRAM. Continuously monitor the incoming data stream and store each received data frame into the SRAM memory, ensuring proper alignment with the specified memory addresses. Validate the integrity of the received data through error-checking mechanisms during the reception process. Conclude the data reception and storage operation once all data frames have been successfully stored in the SRAM memory.

Step 8: Repeat the process for all the slaves: Iterate through each slave device connected to the I2C bus. For each slave, repeat the data transmission and reception process as previously outlined, addressing and communicating with one slave at a time. Ensure proper synchronization and error handling to maintain reliable communication with each slave device.

Step 9: Initialize stop signal, Data is perfectly transmitted, received: Generate a stop signal to conclude the data transmission process on the I2C bus. Confirm that the transmitted data has been successfully received by the designated recipient without errors. Validate the integrity of the received data to ensure accurate communication between the devices.

Applications:

The high-speed I2C interface protocol can be utilized in a variety of applications that demand efficient and rapid data exchange, including:

- Embedded Systems: Enhancing communication between microcontrollers and peripherals in real-time applications.
- Medical Devices: Facilitating high-speed data transfer in medical imaging and monitoring devices.
- Automotive Electronics: Enabling rapid data exchange in automotive sensor networks, including ADAS (Advanced Driver Assistance Systems) and ECU (Electronic Control Unit) communications.

Advantages:

The high-speed I2C protocol introduces several enhancements over the standard I2C, making it an optimal solution for modern high-

International Journal of Engineering Research and Science & Technology
Vol. 21, Issue 2, 2025
ISSN 2319-5991 www.ijerst.com

performance applications:

- Faster Data Transfer: The protocol supports significantly higher clock speeds, reducing latency and improving system performance.
- Lower Power Consumption: Optimized signal integrity and efficient communication reduce energy consumption, making it ideal for battery-powered devices.
- Enhanced Noise Immunity: Advanced error detection and correction mechanisms improve reliability in high-interference environments.
- Increased Scalability: The protocol supports multiple slave devices, allowing seamless expansion in complex systems.
- Backward Compatibility: High-speed I2C retains compatibility with standard and fast-mode I2C, ensuring smooth integration with existing hardware.
- Improved Reliability: The protocol includes robust arbitration and clock stretching mechanisms, preventing data collisions and ensuring smooth communication.
- Efficient Multi-Master Communication: It enables multiple master devices to communicate efficiently, making it suitable for distributed processing applications.
- Optimized Signal Integrity: The protocol leverages advanced techniques like dynamic pull-up resistors to maintain signal quality at high speeds.

4. EXPERIMENTAL ANALYSIS

Figure 1 shows the proposed simulation output. Here clk is the clock input,rd is read and wr is the write.

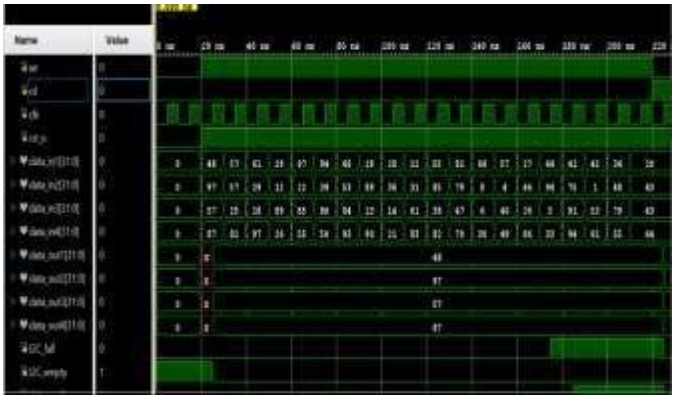


Figure 2: Proposed Simulation Result

Figure 3 shows the proposed area measurement. Here 215 number of lookup tables are used out of available 134600 which consumes 0.16% utilization ,96 number of LUTRAM are used out of available 46200 consumes 0.21% utilization, 48 number of flip flops are used out of 269200 and consumes 0.02% utilization,264 number of IO are used out of 500 consumes 52.80% utilization, 1 number of BUFG is used out of 32 which consumes 3.13% utilization

Resource	Estimation	Available	Utilizatio...
LUT	215	134600	0.16
LUTRAM	96	46200	0.21
FF	48	269200	0.02
IO	264	500	52.80
BUFG	1	32	3.13

Figure 3: Proposed area

Figure 4 shows the proposed power measurement.Here the total power is 35.398W. Static power include PL static is 0.309W and dynamic power includes signals of 2.793W, logic of 0.686W, I/O of 31.609W

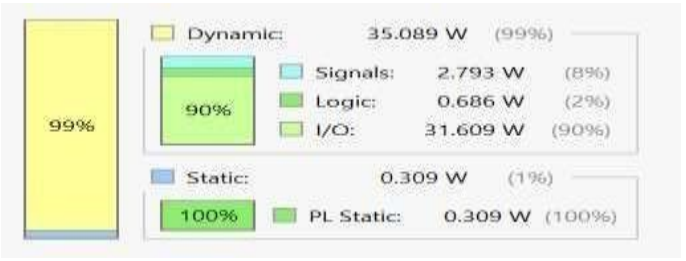


Figure 4: Proposed power

Figure 5 shows the proposed setup delay. Here the total delay is 16.108, Maximum logic delay is 4.263 and maximum net delay is 11.845.

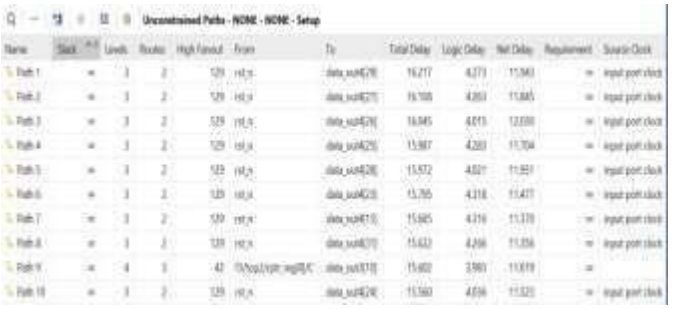


Figure 5: Proposed setup delay

Figure 6 shows the proposed hold delay. Here total delay is 0.449, Maximum logic delay is 0.255, Maximum net delay is 0.194.



Figure 6: Proposed hold delay

5. CONCLUSION

In conclusion, the design of a high-speed I2C interface protocol and the implementation of rapid data exchange present a compelling narrative of innovation and efficiency in communication technology. Through meticulous planning, optimization, and rigorous testing, the development of a high-speed I2C interface protocol has enabled seamless communication between devices, facilitating faster data transfer rates and enhanced system performance. By prioritizing speed, reliability, and efficiency, these mechanisms ensure swift and efficient exchange of data, enabling real-time processing, analysis, and decision-making in various applications. In both endeavors, collaboration, and interdisciplinary. This collaborative effort led to the creation of high-performance communication solutions that drive innovation and propel technological advancements forward. The future holds exciting possibilities for further optimization and refinement of high-speed communication protocols and data exchange mechanisms.

International Journal of Engineering Research and Science & Technology
Vol. 21, Issue 2, 2025
ISSN 2319-5991 www.ijerst.com

As technology continues to evolve, there will be a growing need for faster, more efficient. The design of high-speed I2C interface protocols and rapid data exchange mechanisms represents a milestone in communication technology, paving the way for enhanced connectivity, improved system performance, and transformative innovations in the digital age. Future advancements may focus on reducing latency, minimizing power consumption, and integrating advanced error correction techniques to further improve data integrity and reliability. Additionally, the application of artificial intelligence and machine learning in optimizing communication protocols could open new possibilities for adaptive and intelligent data transfer. These innovations will continue to shape the future of high-speed communication, fostering seamless interactions across a broad spectrum of industries and applications.

REFERENCES

[1] Ishak, Mohamad Khairi, and Meenal Pradeep Kumar. "Design and Implementation of I2C Bus Protocol on Master and Slave Data Transfer Based on FPGA." *Makara Journal of Technology* 26, no. 1: 5.

[2] Malviya, Utsav Kumar, and Gopal Kumar. "Tiny I2C protocol for camera command exchange in CSI-2: a review." In 2020 International Conference on Inventive Computation Technologies (ICICT), pp. 149-154. IEEE, 2020.

[3] Nagaraju, C. H., P. L. Mounika, K. Rohini, T. Naga Yaswanth, and A. Maheswar Reddy. "Design and Implementation of Built-In Self-Test (BIST) Master Slave Communication Using I2C Protocol." In International Conference on Communications and Cyber Physical Engineering 2018, pp. 65-73. Singapore: Springer Nature Singapore, 2023.

[4] Nayana, D. K. "Design and verification for i2c interface protocol using system verilog." Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12, no. 12 (2021): 2380-2384.

[5] Chaithanya, A. Sainath, D. Sindhuja, D. Bhavana, and P. Vennela. "Design and Interfacing of I2C Master with Register and LCD Slaves."

[6] Ghuse, Satish M., and Surendra K. Waghmare. "FPGA Implementation of I2C and SPI Protocols using VHDL."

[7] Khelif, Mohamed Amine, Jordane Lorandel, and Olivier Romain. "Non-invasive i2c hardware trojan attack vector." In 2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 1-6. IEEE, 2021.

[8] Albalooshi, Amina, Abdul-Halim M. Jallad, and Prashanth R. Marpu. "Fault Analysis and Mitigation Techniques of the I2C Bus for Nanosatellite Missions." IEEE Access 11 (2023): 34709-34717.

[9] Sajjanar, Sumanth, and Mrs Deepika Prabhakar. "A Review on BIST Embedded I2C and SPI using Verilog HDL." (2020).

[10] Chauhan, Deepak. "STUDY AND FPGA BASED DESIGN OF BUS PROTOCOLS." PhD diss., 2021.

[11] Nagaraju, C. H., P. L. Mounika, K. Rohini, T. Naga Yaswanth, and A. Maheswar Reddy. "Design and Implementation of Built-In Self-Test (BIST) Master Slave Communication Using I2C Protocol." In International Conference on Communications and Cyber Physical Engineering 2018, pp. 65-73. Singapore: Springer Nature Singapore, 2023.

[12] Shilaskar, Swati, Anup Behare, Ketki Sonawane, and Shripad Bhatlawande. "Post Silicon Validation for I2C (SMBUS) Peripheral." In 2023 36th International Conference on VLSI Design and 2023 22nd International Conference on Embedded Systems (VLSID), pp. 313-318. IEEE, 2023.

[13] Chauhan, Shreyash Naresh, and Ganesh K. Andurkar. "Development of UVM Testbench for I3C protocol." In 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1-4. IEEE, 2023.

[14] Albalooshi, Amina, Abdul-Halim M. Jallad, and Prashanth R. Marpu. "Fault Analysis and Mitigation Techniques of the I2C Bus for Nanosatellite Missions." IEEE Access 11 (2023): 34709-34717