DESIGN OF HIGH SPEED 12C INTERFACE PROTOCOL DESIGN FOR ENABLING RAPID DATA EXCHANGE

P.V.Krishnarao[1], Pasupuleti Yasaswini[1], Samiti Divya[1], Pusala Geethika[1], Peddineni Asritha[1]
[1]Department of Electronics and Communication Engineering, Geethanjali Institute of Science and Technology, Nellore.

Abstract: The rapid exchange of data has become increasingly essential in today's interconnected world, and the I2C (Inter-Integrated Circuit) protocol is a popular choice for achieving this goal in various applications. This paper presents the design of a high-speed I2C interface protocol aimed at enabling swift data exchange. The conventional I2C system is known for its reliability and simplicity, but it has inherent limitations in terms of data transfer rates. These drawbacks, such as low speed and limited scalability, hinder its effectiveness in modern applications requiring rapid data exchange. In response to these limitations, our proposed system leverages innovative techniques to enhance the I2C protocol's performance significantly. By optimizing signal integrity, implementing clock stretching and arbitration mechanisms, and supporting higher clock frequencies, our design enables high-speed data exchange without compromising the protocol's core characteristics. We demonstrate the effectiveness of the proposed high-speed I2C interface through simulation and real-world testing, showcasing its potential to revolutionize data exchange in a wide range of applications, including IoT devices, sensors, and embedded systems. The proposed design addresses the limitations of the conventional I2C system and offers a reliable and efficient solution for achieving rapid data exchange in the modern era.

Keywords: Inter-Integrated Circuit protocol, High speed, XILINX, Data transfer

1.Introduction

The Inter-Integrated Circuit (I2C) bus protocol is a widely used synchronous serial communication protocol that enables communication between multiple integrated circuits (ICs) or devices on a shared bus. It was developed by Philips (now NXP Semiconductors) in the early 1980s and has since become a de facto standard for interconnecting various components in embedded systems, such as microcontrollers, sensors, EEPROMs, and other peripherals. The I2C follows a master-slave architecture, where one device acts as the master, controlling the communication on the bus, and one or more devices serve as slaves, responding to commands from the master. I2C uses only two lines for communication: a Serial Clock (SCL) line and a Serial Data (SDA) line. The SCL line carries the clock signal generated by the master, and the SDA line carries the actual data. The SDA line is bidirectional, allowing both the master and the slave devices to transmit and receive data. This bidirectional nature simplifies the hardware requirements for devices on the bus.

Each slave device on the I2C bus is assigned a unique 7-bit or 10-bit address. The master initiates communication by addressing a specific slave device before transmitting or receiving data. Communication on the I2C bus begins with a start condition (S) and ends with a stop condition (P). The start condition indicates the beginning of a data transfer, and the stop condition signifies the end. Slaves can hold the SCL line low to slow down the master's clock, a feature known as clock stretching. This allows slower devices to participate in the communication without overwhelming the bus. After receiving a byte of data, the receiving party (master or

slave) sends an acknowledgment bit (ACK) to indicate successful reception. Lack of acknowledgment can signal the end of communication or an error condition. The I2C supports multiple masters on the same bus, allowing for more complex system configurations and increased flexibility. The I2C protocol is versatile, efficient, and widely adopted in diverse applications, making it a fundamental communication standard in the world of embedded systems and integrated circuits. It is important to consult the datasheets of specific devices and components for accurate timing requirements and protocol details when implementing I2C communication in a particular application.

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 could be 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. 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).

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 could be found in a wide range of electronic devices. These circuits had to be capable of self-testing. Built-In Self-Test (BIST) was a self-testing method that could be utilized instead of expensive testing equipment. The design and creation of an Inter-Integrated Circuit (I2C) protocol that could 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 could be 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.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 could be 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.

I2C controller designed 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 could 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. 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. Albalooshi, Amina et.al [8] proposed, Despite the reliability concerns that are associated with the I2C bus, it is still one of the most popular on-board data busses to be used in nanosatellite missions. This paper provides a detailed fault analysis for the I2C bus in the context of nanosatellite missions, and consequently investigates and proposes potential mitigation techniques. By conducting experimental testing using the appropriate hardware and software to construct a comprehensive list of I2C bus requirements, characteristics, and failures. Based on the experimental testing possible mitigation approaches are proposed and finally a qualitative risk analysis is delivered to measure the impact of the methods on the overall mission success. The study shows high influence of the I2C bus on the CubeSat health and mission success, thus emphasizing on the importance of design considerations to reduce missions' risk level, as well as counting for runtime failures that can occur during mission operation.

Sajjanar, Sumanth et.al [9] developed, this literature reviewed I2C (Inter-Integrated Circuit) and SPI (Serial-Peripheral Interface) protocols with Built-in Self-Test (BIST) mode. Both protocols were used to connect two devices for exchanging data with each other quickly without any kind of data losses. Chauhan, Deepak et.al [10] developed, the ARM Advanced Microcontroller Bus Architecture (AMBA) was an open-standard interconnect protocol for connecting and managing functional blocks in a system on a chip design. With this bus architecture, it was easier to construct multiprocessor architectures with a high number of controllers and components. Nagaraju, C. H., P. L. Mounika, K. Rohini, et.al [11] developed, 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 could be found in a wide range of electronic devices. Shilaskar, Swati, Anup Behare, et.al [12] proposed, the paper described the importance of Post Silicon Validation in the VLSI design flow and its challenges. It revolved around validating the expected functionality of devices, their compliance with the respective protocol requirements on the tape outs by using the knowledge of C programming, embedded systems, debugger tools

like Lauterbach Trace 32, Corel was debugger for I2C, etc., before the product release in the market. Albalooshi, Amina, Abdul-Halim, et.al [14] suggested, despite the reliability concerns associated with the I2C bus, it was still one of the most popular on-board data buses used in nanosatellite missions. This paper provided a detailed fault analysis for the I2C bus in the context of nanosatellite missions and consequently investigated and proposed potential mitigation techniques. Thumma, Rajesh, and Pilli Prashanth, et.al [15] implemented serial peripheral interface (SPI) transferred the data between electronic devices like microcontrollers and other peripherals. SPI consisted of two control lines: select signal and clock signal, and two data lines: input and output.

3.Proposed Methodology

Figure 1 shows the proposed system architecture.

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 the devices.

Step 2: Start the environment, write enable will be active high. Random data counter starts generate 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.

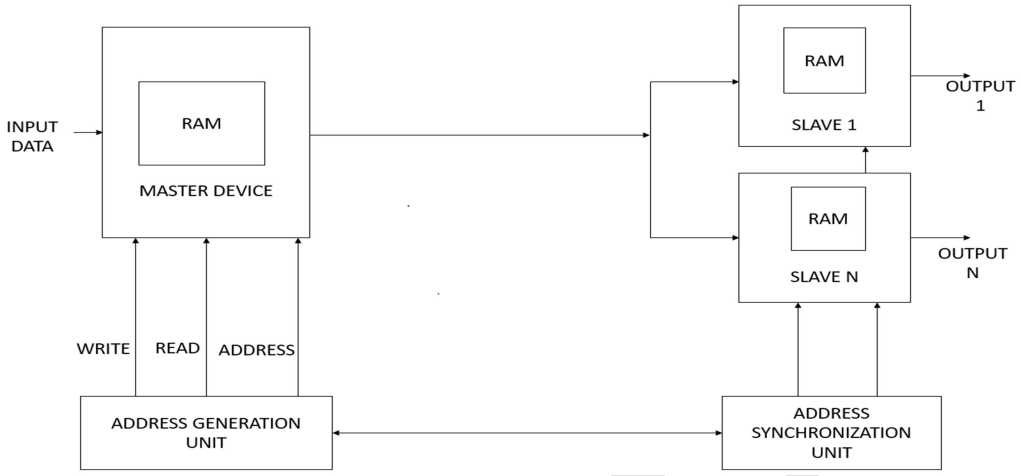

Figure 1. Proposed system architecture.

Step 5: Generate acknowledgment(ack) signal to initialize receiver operation.

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

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.

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 may 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.

4.Result and Discussion

Figure 2 likely shows the result of simulating proposed implementations of the I2C protocol. Figure 3 provides a summary of the design aspects of proposed implementations of the I2C protocol. Table 1 presents a comparison between the existing and proposed implementations of the I2C protocol across various metrics or parameters. The comparison may include metrics such as the number of Look-Up Tables (LUTs), Look-Up Table RAMs (LUTRAMs), Flip-Flops (FFs), Input/Output (IO) ports, and other relevant resources utilized by each implementation. The comparison highlights the differences and improvements achieved by the proposed implementation compared to the existing one, such as reductions in resource utilization.

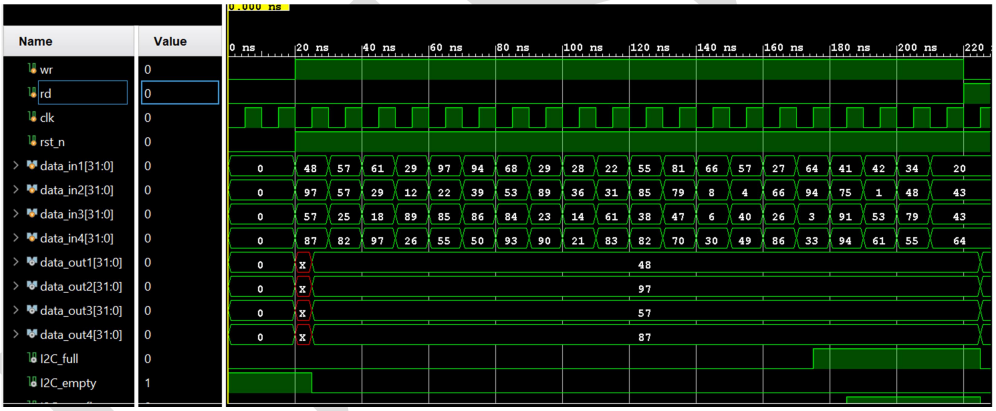

Figure 2. Simulation outcome.

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. Design Summary

Table 1. comparison Table

Metric	Existing	Proposed

LUT	2190	215
LUTRAM	96	96
FF	272	48
IO	264	252
BUFFER	1	1

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. 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.

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] U. Penchalaiah and V. S. Kumar, "Design and Implementation of Low Power and Area Efficient Architecture for High Performance ALU", Parallel Processing Letters., vol. 32, no. 01n02, pp. 2150017, 2022.

[14] 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.

[15] 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.

[16] Thumma, Rajesh, and Pilli Prashanth. "Design and verification of daisy chain serial peripheral interface using system Verilog and universal verification methodology." TELKOMNIKA (Telecommunication Computing Electronics and Control) 21, no. 1 (2023): 168-177.