Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 7, July : 2025

# Design and Implementation of a Configurable N-bit SPI to I2C Protocol Converter using Verilog

*[1]KONINTI PRAVEENA, [2]T. MOHAN DAS,*
*[1]M. Tech, VLSI System Design, Student, Department of ECE,*
*Email: praveenakoninti@gmail.com,*
*[2]Assistant Professor(C), Department of ECE, Email: mohandas.nitdgp@gmail.com,*
*JNTUH University College of Engineering Sultanpur, Sangareddy, Telangana, India*

***Abstract: In embedded system design, serial communication protocols such as SPI (Serial Peripheral Interface) and I2C (Inter-Integrated Circuit) are commonly employed to connect various peripheral devices. However, direct communication between components using different protocols is problematic due to their inherent incompatibility. This work introduces the development of a hardware- based protocol converter designed in Verilog HDL to bridge SPI and I2C communication. The proposed system captures data from an SPI master and reformats it to be compatible with I2C slave devices, allowing efficient interaction across protocol boundaries. The architecture consists of an SPI slave interface, an I2C master interface, and a control logic unit that manages data handling and ensures proper timing coordination. The entire design was synthesized and functionally verified on an FPGA platform to ensure accuracy and timing reliability. This converter offers a reliable and cost-efficient hardware solution for integrating mixed-protocol peripherals in VLSI-based embedded systems.***

***Keywords: SPI, I2C, Verilog, Protocol, Vivado.***

## *I.* INTRODUCTION

Embedded systems play a critical role in modern electronics, powering everything from consumer gadgets to industrial automation solutions. These systems rely heavily on communication protocols to facilitate data exchange between microcontrollers and peripheral devices. Among the most widely used protocols in this domain are SPI (Serial Peripheral Interface) and I2C (Inter-Integrated Circuit), both known for their simplicity, efficiency, and effectiveness in short-distance, intra-board communication. Despite their popularity, SPI and I2C are fundamentally different in their architectures and operation. SPI is a full-duplex, master-slave communication protocol that utilizes separate lines for data transmission and reception, offering high-speed data transfer. In contrast, I2C is a half-duplex, multi-master protocol that operates over two lines: a serial data line (SDA) and a serial clock line (SCL), supporting multiple devices on the same bus through addressing. This difference in design creates a significant challenge when a system requires communication between components using these two incompatible protocols. Integrating devices that use SPI and I2C without a proper interface can lead to data corruption, timing issues, and overall system instability. Therefore, there is a growing need for a mechanism that can enable seamless communication between SPI and I2C devices in mixed-protocol environments.

To overcome this interoperability issue, a protocol converter becomes essential. A protocol converter functions as a bridge between two different communication standards, translating the format and timing of one protocol into that of another. In this context, a converter that can receive data from an SPI master and relay it accurately to an I2C slave offers a highly practical solution in embedded system design. This project aims to design and implement such a converter using Verilog Hardware Description Language (HDL), which is a standard tool in VLSI and digital design. Verilog allows for precise control over hardware behavior at the register-transfer level, making it an ideal choice for implementing custom digital circuits like protocol converters.The proposed converter is developed by creating separate functional modules for SPI and I2C within a single design framework. The SPI slave module captures data from an external SPI master device. This data is temporarily stored in internal registers or buffers before being forwarded by the I2C master module to an I2C-compliant peripheral. A central control unit is responsible for managing the timing and control signals that ensure correct sequencing of data capture, conversion, and transmission. This control logic is critical because SPI and I2C operate with different timing models and data formats. The converter must not only translate the data but also synchronize the operations across two distinct clock domains. Data integrity is a crucial aspect of the converter's design. To maintain reliability, the system incorporates error-checking mechanisms and adheres to the timing constraints specified in both SPI and I2C protocols. This ensures that communication is accurate and consistent across various operating conditions.

To verify the functionality of the protocol converter, the design is synthesized and simulated using FPGA development tools. Simulation provides insights into the timing behavior and logic correctness, allowing designers to catch potential errors before hardware implementation. The FPGA environment also serves as a platform for real-time testing and debugging. Once verified in simulation, the design is implemented on a physical FPGA board. Hardware-level testing includes sending known data sequences from an SPI master to the converter and confirming the correct reception by the I2C slave. Timing analyzers and logic probes help validate that data transfer occurs without glitches or violations.

## *II.* RELATED WORKS

***B. Jose and J. S. Immanuel, "Design of BIST(Built-In-SelfTest)Embedded Master-Slave communication using SPI Protocol," 2021 3rd International Conference on Signal Processing and Communication (ICPSC), Coimbatore, India, 2021, pp. 581-585***

Focuses on developing a system that integrates a built-in self-test mechanism within a master-slave communication framework based on the SPI (Serial Peripheral Interface) protocol. The design aims to enhance the reliability and fault detection capabilities of SPI-based communication by embedding self-testing features that can automatically verify the functionality of the communication interface without external testing equipment. This approach improves system robustness and reduces maintenance overhead, making it suitable for applications requiring dependable data exchange in embedded systems.

***D. Trivedi, A. Khade, K. Jain and R. Jadhav, "SPI to I2C Protocol Conversion Using Verilog," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-4***

The SPI to I2C Protocol Conversion using Verilog project successfully implements a digital bridge that enables communication between SPI and I2C devices by translating the differing protocols at the hardware level. Utilizing Verilog HDL, the design manages the distinct signaling, timing, and control requirements of both interfaces, ensuring accurate and synchronized data exchange. This conversion facilitates interoperability in systems where components with different communication standards need to interface, enhancing flexibility and integration in embedded system designs. The result is a reliable and efficient protocol converter suitable for FPGA or ASIC implementation.

***S. Mehrotra and N. Charaya, "Design of I2C Single Master Using Verilog," in International Journal of Science and Research, vol. 4, no. 1, pp. 1897-1900, Jan. 2015***

The design of an I2C Single Master using Verilog successfully creates a hardware implementation of the I2C communication protocol with a single master device controlling the data flow. The Verilog-based design manages the essential I2C features such as start and stop conditions, address recognition, data read/write operations, and clock synchronization on the shared bus. This implementation enables reliable communication with multiple slave devices on the bus while maintaining proper timing and protocol compliance. The outcome is an efficient, reusable module suitable for integration into larger digital systems requiring I2C master functionality

***L. M. Kappaganthu, M. D. Prakash and A. Yadlapati, "I2C protocol and its clock stretching verification using system verilog and UVM," 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 2017, pp. 478-480***

The design of an I2C Single Master using Verilog successfully creates a hardware implementation of the I2C communication protocol with a single master device controlling the data flow. The Verilog-based design manages the essential I2C features such as start and stop conditions, address recognition, data read/write operations, and clock synchronization on the shared bus. This implementation enables reliable communication with multiple slave devices on the bus while maintaining proper timing and protocol compliance. The outcome is an efficient, reusable module suitable for integration into larger digital systems requiring I2C master functionality.

***L. Li, Y. Wang, X. Chen and X. Ren, "Research on Improvement of Configurable I2C controller IP Core," 2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT), Jilin, China, 2023, pp. 484-489***

The research focuses on enhancing the performance and flexibility of a configurable I2C controller IP core by optimizing its architecture and adding advanced features such as dynamic clock control, multi-master support, and error detection mechanisms. Through improved configurability, the controller can adapt to a wide range of system requirements, making it suitable for diverse embedded applications. The upgraded IP core not only supports standard I2C operations but also allows developers to adjust parameters like data rate and addressing modes, improving integration efficiency and resource utilization in FPGA or ASIC designs. The outcome is a more versatile and robust I2C controller that meets the evolving demands of modern communication systems.

## *III.* EXISTING METHOD

In hardware system design, interfacing different communication protocols is a common requirement. A typical example involves integrating SPI (Serial Peripheral Interface) with I2C (Inter-Integrated Circuit), which are widely used for short-distance data exchange between microcontrollers and peripherals. In most Verilog-based designs, this integration is realized by combining two separate protocol modules: an SPI slave and an I2C master. These modules are coordinated by a centralized controller that manages the transition and flow of data between them. The SPI module functions as the input channel in this architecture. It receives commands or data from an external SPI master using standardized lines such as SCLK, MOSI, MISO, and SS.

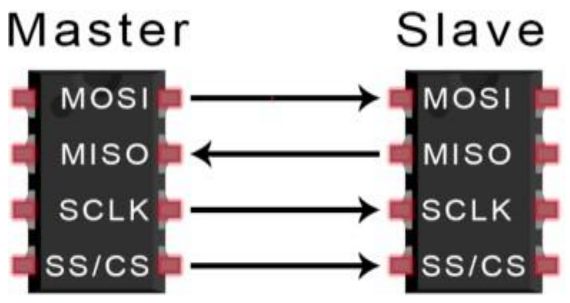


Fig.1: SPI (master and slave) protocol block diagram

As the SPI slave captures serial data bit-by-bit, it employs a shift register to sequentially store the incoming data. Once a complete byte or word is received, it is converted into a parallel format for further processing. This converted data is usually stored in a register or buffer, allowing temporary retention before it is forwarded to the I2C module. This intermediate storage is crucial for maintaining data integrity, especially when dealing with protocol speed mismatches. The handoff between the SPI and I2C modules is managed by control logic, which is often implemented as a finite state machine (FSM). The FSM monitors flags or signals from the SPI module and triggers appropriate sequences for I2C transmission. When the control unit detects that valid data has been received from SPI, it prepares the I2C master to initiate communication. The FSM generates control signals for start conditions, addressing, and data transmission according to the I2C protocol.

The I2C module begins the transfer by sending a start condition, followed by the slave address and a read/write bit. This is done using the SDA (data) and SCL (clock) lines, which are open-drain and require synchronization. Since SPI uses separate lines for data in and out and operates at higher frequencies, while I2C uses a shared, slower bus, the two protocols differ significantly in timing characteristics. Careful synchronization is necessary to avoid data corruption.

Timing issues are handled by introducing buffering and control mechanisms. These may include wait states or ready flags that ensure each module only proceeds when the other is prepared, especially when data arrives faster than it can be sent out. FIFO (First-In, First-Out) buffers are commonly used between SPI and I2C components. These allow continuous SPI data reception while the I2C bus is busy transmitting earlier data, preventing overflows and ensuring seamless operation. These FIFO structures act as timing equalizers and decouplers between the two protocols, bridging the speed gap and allowing each protocol to operate at its optimal rate without causing delays or loss. Error handling is generally basic in such systems. While the I2C protocol includes an ACK/NACK mechanism to confirm successful data reception, SPI assumes successful communication unless custom error detection logic is added. For enhanced reliability, some implementations may incorporate basic status flags or error indicators that are monitored by the control logic. However, full-fledged error correction techniques are rare in basic SPI-to-I2C bridges.

The entire system is typically designed using Verilog HDL (Hardware Description Language). This allows designers to define precise timing, logic behaviour, and signal interactions at the register-transfer level (RTL).Before physical implementation, simulation is conducted using tools such as Xilinx Vivado or ModelSim. These simulations validate the logic flow, timing constraints, and data integrity across all protocol layers. After successful simulation and synthesis, the design is deployed onto a target FPGA board. This allows real-time verification of the interface's behaviour and provides an opportunity to test various edge cases in hardware. Developers often include debug registers or test modes in the Verilog design, which aid in diagnosing problems during hardware testing. These may include loopback tests or status outputs to observe system performance. Power and resource usage are also considered, especially when the SPI-I2C Bridge is implemented in embedded systems or IoT devices. Lightweight design and low clock domains are favoured to reduce overhead.

## *IV.* PROPOSED METHOD:

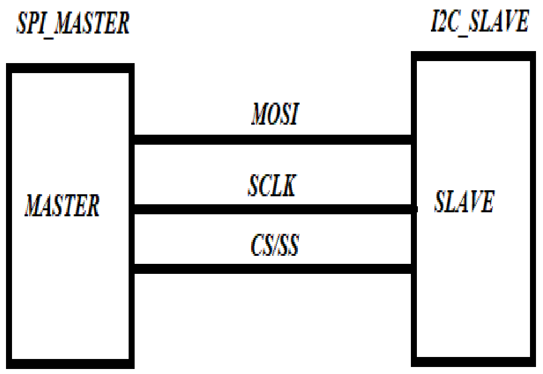


Fig: 2. PROPOSED BLOCK DIAGRAM

The proposed architecture offers a dynamic and customizable solution for converting SPI input into I2C output using Verilog hardware description language. This system emphasizes flexibility, making it suitable for embedded applications with varying protocol requirements. Unlike rigid, fixed-function converters, this design introduces parameters that allow developers to adapt the system to different data widths, clock speeds, and I2C addressing schemes. This results in a more scalable and reusable communication bridge. At its core, the architecture is modular and broken down into four essential components: an SPI slave unit, an I2C master unit, a control unit based on a finite state machine (FSM), and a data buffer that connects the two. Each of these modules is written to support parameterization, so users can change features during synthesis or runtime through control registers. This gives significant flexibility during integration into different systems. The SPI slave component receives serial data from an SPI master. It is designed to be highly configurable, accepting 8-bit, 16-bit, or wider data words based on the defined data width parameter N.A shift register captures the incoming bits serially, with its length determined by the value of N. When the specified number of bits is received, the shift register's content is moved to a parallel data buffer. A signal is generated internally to indicate that a complete data word has been received. This triggers the control unit to prepare for data transmission to the I2C bus. The SPI interface also monitors the Slave Select (SS) signal to recognize when a new transmission is about to start. It resets its internal state and readies itself to receive data accordingly.

Additionally, built-in validation logic within the SPI module ensures proper clocking and data alignment. It checks for invalid clock sequences or incomplete transmissions and flags any detected issues. Once data is safely captured, it is stored temporarily in a register or FIFO buffer. This buffer plays a crucial role in isolating the SPI and I2C modules, which typically operate at different speeds. The FSM-based controller manages the transition from SPI reception to I2C transmission. It shifts between states like idle, receive, prepare, transmit, wait for ACK, and error handling. Before initiating I2C transmission, the FSM processes the data to fit the I2C protocol format. This includes breaking the N-bit SPI data into 8-bit chunks, if necessary.

A 7-bit slave address and a control bit (read or write) are appended to structure the I2C packet properly. These values are programmable to support different I2C peripherals. The I2C master unit in this design is not hardcoded but programmable. A prescaler inside it controls the I2C clock (SCL), supporting various standard modes like 100 kHz and 400 kHz. The I2C communication starts with a START condition, which signals the bus is busy and prevents other devices from transmitting. The slave address is sent out first, followed by the data chunks one by one. Each chunk is placed on the SDA line in synchronization with the SCL line. After each 8-bit segment, the system checks for an acknowledgment (ACK) from the targeted I2C device. This ensures the device is responding and accepting data.

If no ACK is received, the FSM enters a separate state to handle this failure. Depending on configuration, it can retry the transmission or raise an error interrupt to the system. After all data chunks are successfully acknowledged, the I2C controller issues a STOP condition, releasing the bus for other devices. The FSM transitions back to the idle state, completing the cycle and waiting for the next SPI command to arrive. Due to the asynchronous nature of SPI and I2C, a timing bridge is required. This is usually implemented using a FIFO buffer that operates on two different clocks. The FIFO helps maintain a steady flow of data and prevents loss due to mismatches in transmission rates. It acts as a safeguard for timing and synchronization. Developers can configure protocol parameters like SPI clock polarity (CPOL), clock phase (CPHA), and I2C timing through either Verilog parameters or mapped control registers. The system can be expanded to support repeated START conditions in I2C. This allows the master to communicate with multiple devices on the same bus without issuing a STOP in between.

Using repeated START improves efficiency, especially in sensor-dense environments where data must be fetched from multiple nodes quickly. Fault resilience is enhanced through built-in detection flags for errors such as framing issues, ACK failures, or buffer overflows. The control unit responds appropriately to each scenario. To validate the design, a comprehensive test bench is constructed. It simulates SPI transmissions and observes the corresponding I2C outputs under various conditions. The simulation environment checks boundary conditions, such as maximum data width handling, edge cases like missed ACKs, and invalid clock behaviour. The FSM's behaviour is also verified to ensure all state transitions are accurate and all outputs are correctly timed relative to the clock domains.

Ultimately, this modular, parameterized protocol bridge provides an efficient and robust method of converting SPI data into I2C communication streams in real-time, making it suitable for a wide range of digital systems and applications.

## *V.* RESULTS:

**Simulation waveform:**

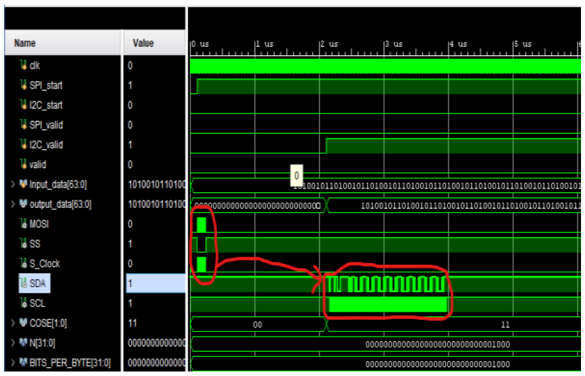
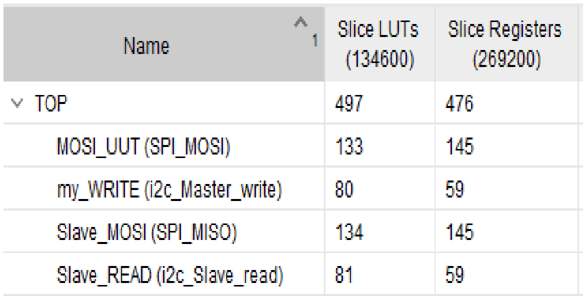


Fig:2. Simulation waveform spi to i2c

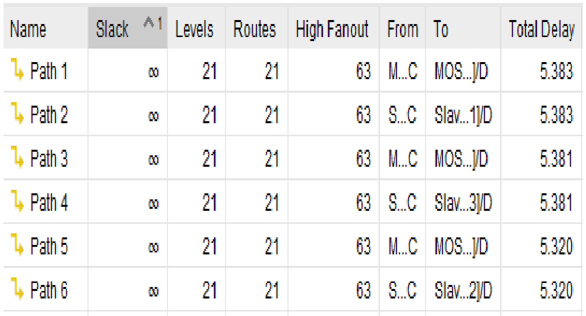
**Area:**



| Name | Slice LUTs (134600) | Slice Registers (269200) |
|---|---|---|
| TOP | 497 | 476 |
| MOSI_UUT (SPI_MOSI) | 133 | 145 |
| my_WRITE (i2c_Master_write) | 80 | 59 |
| Slave_MOSI (SPI_MISO) | 134 | 145 |
| Slave_READ (i2c_Slave_read) | 81 | 59 |

Fig11: AREA

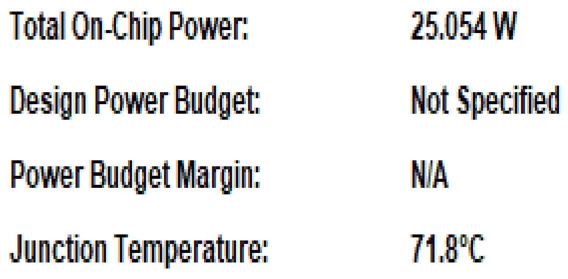
**Delay**:



| Name | Slack | Levels | Routes | High Fanout | From | To | Total Delay |
|---|---|---|---|---|---|---|---|
| Path 1 | ∞ | 21 | 21 | 63 | M...C | MOS...]/D | 5.383 |
| Path 2 | ∞ | 21 | 21 | 63 | S...C | Slav...1]/D | 5.383 |
| Path 3 | ∞ | 21 | 21 | 63 | M...C | MOS...]/D | 5.381 |
| Path 4 | ∞ | 21 | 21 | 63 | S...C | Slav...3]/D | 5.381 |
| Path 5 | ∞ | 21 | 21 | 63 | M...C | MOS...]/D | 5.320 |
| Path 6 | ∞ | 21 | 21 | 63 | S...C | Slav...2]/D | 5.320 |

Fig12: DELAY

**Power:**



| | |
|---|---|
| Total On-Chip Power: | 25.054 W |
| Design Power Budget: | Not Specified |
| Power Budget Margin: | N/A |
| Junction Temperature: | 71.8°C |

Fig13: POWER

## *VI.* CONCLUSION:

The custom-designed N-bit SPI to I2C protocol converter, developed using Verilog, effectively connects two popular communication interfaces. Its support for variable data widths and precise handling of timing and control signals specific to each protocol ensures seamless and dependable data transfer. Implemented directly in hardware, the design delivers low-latency performance while minimizing resource usage, making it well-suited for FPGA-based systems. This converter improves compatibility across devices, eliminates the need for extra interfacing components, and offers a scalable solution tailored for a wide range of embedded system applications. In essence, the project presents a robust and efficient method for enabling communication between SPI and I2C environments.

## *VII.* REFERENCES

1. B. Jose and J. S. Immanuel, "Design of BIST(Built-In-Self Test)Embedded Master-Slave communication using SPI Protocol," 2021 3rd International Conference on Signal Processing and Communication (ICPSC), Coimbatore, India, 2021, pp. 581-585
2. A. Kulkarni and S. M. Sakthivel, “UVM methodology based functional Verification of SPI Protocol,” in Journal of Physics Conference Series, vol. 1716, no. 1, pp. 012035, Dec. 2021
3. Y. Guo et al., "A SPI Interface Module Verification Method Based on UVM," 2020 IEEE International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), Chongqing, China, 2020, pp. 1219-1223
4. D. Trivedi, A. Khade, K. Jain and R. Jadhav, "SPI to I2C Protocol Conversion Using Verilog," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-4
5. S. Mehrotra and N. Charaya, “Design of I2C Single Master Using Verilog,” in International Journal of Science and Research, vol. 4, no. 1, pp. 1897-1900, Jan. 2015
6. L. M. Kappaganthu, M. D. Prakash and A. Yadlapati, "I2C protocol and its clock stretching verification using system verilog and UVM," 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 2017, pp. 478-480
7. L. Li, Y. Wang, X. Chen and X. Ren, "Research on Improvement of Configurable I2C controller IP Core," 2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT), Jilin, China, 2023, pp. 484-489
8. B. Jeevan, P. Sahithi, P. Samskruthi and K. Sivani, "Simulation and synthesis of UART through FPGA Zedboard for IoT applications," 2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 2022, pp. 1-7
9. F. Leens, (2009) An Introduction to I2C and SPI Protocols. In: IEEE Instrumentation & Measurement Magazine. Beijing. pp. 8-13
10. S. Wang, (2010) Design and Implementation of Serial Peripheral Interface SPI Based on FPGA. Control & Automation, pp.117-119.
11. W.L.Li, D.P. Y. (2007) Implementation of Asynchronous Serial Port and Synchronous Serial Port Conversion Using FPGA. Electronic Engineer, pp.52-53.
12. S.Zhang, W.Li. (2014) Modular design method of FPGA. Journal of Electronic Measurement and Instrument, pp. 560-565.
13. F.J.Sun, C.X.Yu.(2005) Verilog Implementation of SPI Serial Bus Interface. Modern Electronic Technique, pp. 105-106,109.
14. W.Mai, W.Liu. (2007) Design and Implementation of SPI Interface Based on FPGA and MSP430. Instrumentation Users, pp. 100-102.
15. W.C.Zhu, S. ,Zhang, H.L.Jiang. (2017) Design of high speed data communication interface based on ARM and FPGA. Journal of Guilin University of Electronic Technology, pp. 293-297