International Journal of Scientific Research in Computer Science, Engineering and Information Technology

ISSN : 2456-3307 OPEN ACCESS

Available Online at : www.ijsrcseit.com

doi : https://doi.org/10.32628/CSEIT25112759

Design and Simulation of SPI Multi Master and Multi Slave Communication Using Arbitration Algorithm

Tharun R*, S Kishore, Neha Hosamath, Keerthi Kulkarni, Vrunda Kusanur

Department of Electronics and Communication Engineering(ECE),B.N.M Institute of Technology(BNMIT), Bangalore, Karnataka, India

ARTICLEINFO

Article History:

Accepted : 29 March 2025

Published: 01 April 2025

Publication Issue

Volume 11, Issue 2

March-April-2025

Page Number

2939-2947

ABSTRACT

In order to effectively coordinate numerous devices, ensure scalability, stability, and prevent overload on a single master, multi-master and multi-slave communication is crucial for controlling big, complex systems.Numerous techniques are currently established using I2C protocols, spi single master slave, master and multi slave communication, and others; this could cause a delay in overcoming what I'm going to build.A multi-master, multi-slave communication system is designed and simulated using verilog in Cadence tool. The architecture prioritizes synchronization and arbitration algorithm while utilizing protocols such as SPI to facilitate effective data exchange. The simulation environment provided by Cadence verifies performance indicators like power consumption, latency, and throughput. The system is scalable and reliable, as evidenced by the results, which makes it appropriate for applications involving multi-core processors, the Internet of Things and automobiles. The efficiency with which Cadence optimizes intricate communication designs is demonstrated in this paper.

Keywords: SPI, Multi-master, Multi-slave, Cadence, Efficiency.

Introduction

In Multi-master and multi-slave communication using SPI (Serial Peripheral Interface) allows various controllers (masters) to communicate with several peripherals (slaves) on the same bus. This configuration is helpful in complicated systems that need redundancy and shared resource access, such as industrial automation, automotive, and the Internet of Things. But it creates problems like clock synchronization and bus contention, necessitating techniques for software or hardware arbitration. The flexibility, scalability, and fault tolerance of the system are improved by multi-master SPI, despite its complexity.

In order to avoid conflicts, multi-master SPI allows many controllers to share the SPI bus, necessitating the use of software protocols or hardware arbiters as arbitration methods. Clock synchronization, bus

contention, and dependable communication are among the difficulties. For distributed control and redundancy, it is helpful in industrial, automotive, and Internet of Things systems.

A single master can connect with several peripherals (slaves) via multi-slave SPI by employing separate Chip Select (CS) lines for each slave. To choose the target slave and provide isolated and regulated communication, the master turns on one CS line at a time. Applications such as sensor networks, memory systems, and display interfaces frequently use this configuration.

The major objectives of designing and modeling an effective multi-master and multi-slave SPI system in Cadence with Verilog are throughput maximization and latency minimization. To lessen conflict and guarantee equitable access across masters, the architecture integrates efficient bus arbitration logic, such as priority-based or round-robin systems. While masters share MOSI, MISO, and SCLK lines with low signal propagation delay, slaves are each given their own Chip Select (CS) line. To improve data transfer efficiency, pipeline and parallel processing techniques are incorporated into the architecture of the SPI modules in Verilog. To verify low-latency performance, simulation in Cadence entails stress-testing the system under high-frequency clock speeds and concurrent master requests. A reliable, fast SPI system with effective arbitration and low communication overhead is the intent.

LITERATURE SURVEY

A review of the literature on SPI communication shows that single-master, multi-slave implementations have been extensively studied. These implementations are commonly used in embedded systems for memory access, display control, and sensor interfacing. Research on master SPI focuses on hardware methods to control bus contention and arbitration techniques including priority-based access and token-passing. High-speed SPI enhancements, such as dual/quad-SIP modes and DMA integration for increased throughput, are also being investigated[1]. Low-power SPI designs for wearable and the Internet of Things, along with security improvements for encrypted communication, are the focus of recent study. With continuous improvements addressing scalability, efficiency, and reliability in master, slave systems, SPI is still a flexible protocol overall.

There are two time parameters used in SPI data transmission: clock polarity (COPL) and clock phase (CPHA). How the SS, SCLK, MOSI, and MISO signals time with one another depends on the timing settings. The SPI control register's (SPCR) two bits are responsible for configuration. One can set the clock polarity (COPL) control bit to either 0 or 1. The SCLK is in its high idle state when the clock polarity is set to 1. The clock phase (CPHA) control bit is another that has two possible values: 0 and 1. On the leading edge of SCLK, data is latching while clock phase is zero, and on the trailing edge, data is changing.Data is captured on the trailing edge of SCLK and shifts on the leading edge when the clock phase is 1[1].

The constraints of parallel and serial communication protocols are addressed in this study through the design and analysis of an FPGA- based Single Master Multiple Slave (SPI) system. With support for both single-slave and multi-slave configurations, the Verilog- designed SPI modules were successfully implemented on an FPGA after being simulated in Model sim. The system showed effective data transfer and synchronization with full-duplex communication for single-slave configurations and half-duplex communication for multi-slave settings. The design's promise for applications needing dependable and scalable peripheral connection was demonstrated by the FPGA implementation, which confirmed the design's functionality[2].According to Motorola's SPI requirements, the document offers general-purpose SPI Master/Slave IP cores made for FPGA and ASIC applications. With thorough RTL implementation and verification in Verilog, it achieves transfer rates of 71–

75 MBPS and emphasizes high flexibility, full-duplex communication, and connection with the OPB bus[3]. In order to manage uncertainties, disturbances, and delays and guarantee reliable and stable performance, this work uses non-singular fast terminal sliding mode control (NFTSMC) and its adaptive counterpart (ANFTSMC) for adaptive finite-time control for master-slave manipulators with time-varying delays[4].With an emphasis on data transfer, error detection, and stability, the research presents a master-slave control system communication strategy for cascaded inverters utilizing FPGA. This ensures coordinated control and real-time monitoring of inverter modules[5].

With a focus on data synchronization, addressing, and read/write operations to guarantee effective communication with a master device, this paper introduces an I2C slave interface utilizing Verilog HDL[6]. Master-slave management for parallel inverters has been studied in the past, with an emphasis on load sharing and communication delays. Performance and stability consequences of wireless communication are also investigated[7]. While BIST research enhanced fault detection and reduced testing costs, earlier studies optimized SPI for effective data transport[8].

Prior research indicates that UVM facilitates effective and reusable SPI design verification. Automated stimulus generation guarantees design reliability and improves functional coverage[9].Prior research has investigated System Verilog- based verification for SPI interfaces using functional coverage, limited random testing, and OOP. To increase efficiency, these works prioritize assertion-based verification and reusable test benches[10].an SPI Master-Slave core with 100% code coverage in Model Sim and complete duplex data transfer is designed and functionally verified using System Verilog. The design has eight slave select lines, programmable bit transfers, and compatibility for SPI and Micro wire/plus protocols[11-12].

The study suggests a master-slave design for a TCP/UDP telecommunication system that uses micro controllers as slaves and a PC as the master to facilitate effective data transfer across a local area network. Because of its real-time communication, minimal installation costs, and ease of use, the system is appropriate for real-world network applications[14].

METHODOLOGY OF PROPOSED MODEL

In general, To manage bus access and avoid conflicts, use a strong arbitration mechanism such as priority-based or token-passing when designing and simulating a multi-master and multi-slave system in Cadence Verilog. This will ensure great efficiency and minimal delay. In order to manage interactions effectively and reduce latency, use state machines, clock synchronization, and data buffering. Utilize error detection (like CRC) and reliability acknowledgments to optimize the data frame structure. In order to detect and fix bottlenecks and guarantee that the system satisfies performance objectives, simulate in Cadence using timing analysis.

A. Design of Serial Peripheral Interface:

The master and slave module, which are seen in Fig 1, make up the majority of SPI's internal architecture. Between the MCU and peripheral devices, the SPI module enables full duplex, synchronous, serial communication. To enable it, set the SPI Control Register's SPI enable (SPE) bit. A synchronization clock, two data lines, and a control line are used by the SPI to communicate[1]:

1. Shift Register: Manages serial data transfer, for 8-bit data transmission, bits are labeled D7 to D0.
2. Slave Enable: This feature, which is produced by the Slave Generator, permits communication with the chosen slave device.
3. SPI Master: Manages data flow across MOSI (Master Out Slave In) and MISO (Master In Slave Out) lines and houses the Shift Register.
4. SPI Slave: In sync with the master's clock, it receives and sends data via the Shift Register.

5. Control Signals: Output Enable and Shift Enable regulate how the data output and shift register function.
6. Lines of Communication: MISO for slave-to-master data transfer and MOSI for master-to-slave data transfer.
7. Synchronization: The Clock Generator makes sure that the slave and master devices exchange data in sync.

confusing the reader.

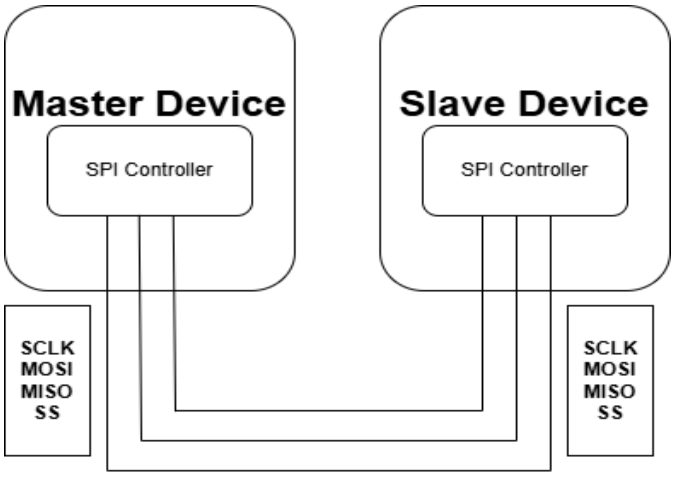

Figure 1. SPI Block Diagram with Master-Slave

The SPI system creates a 16-bit distributed register by connecting the 8-bit data registers in the master and slave via MOSI and MISO. The register shifts serially during transfer, allowing the slave and master to exchange data. The slave's input is reflected in the data read from the master, and the slave's output is the data sent to the master. SPI functions in two modes: slave (responds) and master (initiates).

B. Master and Slave transfer operation:

By creating the clock signal (SCLK) and choosing the slave via the Slave Select (SS) line, the master module in the SPI system starts communication. The MOSI (Master Out Slave In) line is used by the master to send data to the slave, and the MISO (Master In Slave Out) line is used to receive data from the slave. The slave module processes incoming data from MOSI and transmits data back through MISO in response to directives from the master. Shift registers are used by both the slave and the master to serially send and receive data while keeping track of the master's clock. As a result, the master and slave can exchange data simultaneously thanks to full-duplex communication[1-8].

C. Master Design of Single master-Slave flowchart:

The flowchart representation of the SPI communication protocol in Figure 2. To begin communication, Chip Select (CS) must be asserted first. Next, data must be initialized in the Shift Registers. Serial clocks, or SCLK, are created to synchronize data transport. Data is simultaneously moved out and sampled in during the operation. The communication is terminated by deasserting Chip Select (CS) and setting the Done Signal to 1 after 16 bits have been sent. An SPI system's master and slave devices transmit data in a coordinated and effective manner thanks to this method.

Two timing parameters used in SPI data transfer are clock polarity (COPL) and clock phase (CPHA). The SS,SCLK, MOSI and MISO signals time connection is

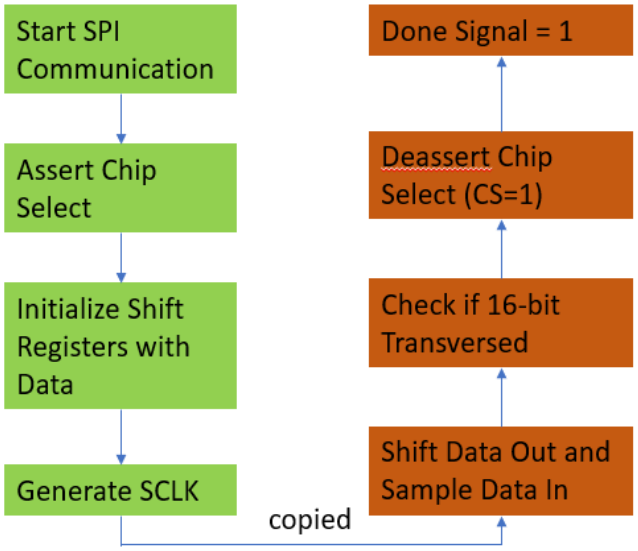

Figure 2. Single SPI Master-Slave flowchart

influenced by the timing parameters. The SPI control register has two bits that perform the configuration. The clock polarity control bit has two possible values: 0 and 1. SCLK is in a low idle state when clock polarity is set to 0. SCLK is in its high idle state when

clock polarity is set to 1[4]. Also, there are two possible values for the clock phase control bit: 0 and 1. Data changes on the following edge of SCLK and is latching on the leading edge when the clock phase is 0.Data is captured on the SCLK trailing edge and changes on the leading edge when the clock phase is 1[1-9].

D. Design of Multi master and Multi slave Communication:

In order to manage bus access, the device implements a multi-master SPI system with two masters, two slaves, and an arbiter. The SPI master module processes data transmission and reception by generating the clock (sclk), sending and receiving data via mosi and miso, and controlling the slave select (ss). The SPI slave module receives data from the master via mosi, sends and receives data via miso, and processes data when selected by ss.

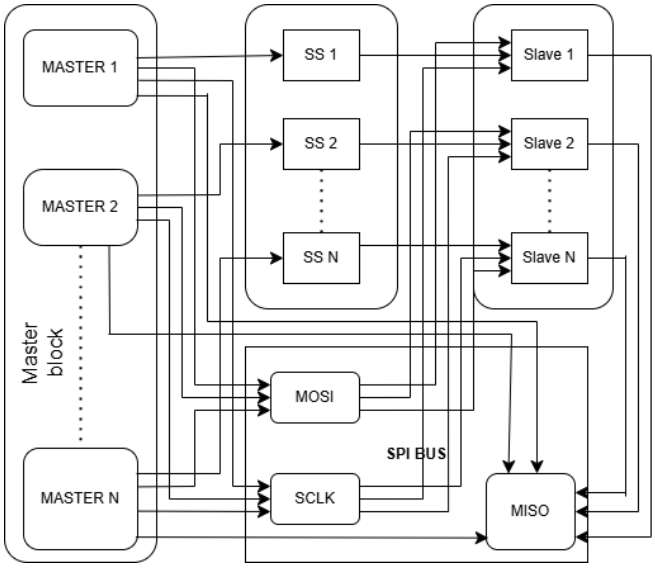

Figure 3. Block design of SPI multi Master and multi Slave communication

A multi-master and multi-slave SPI communication system is depicted in Fig. 3. It consists of three slaves (S1, S2, S3) connected by the SPI bus and three masters (M1, M2, M3). Master Out Slave In (MOSI), Master In Slave Out (MISO), Serial Clock (SCK), and Slave Select (SS) are the four primary lines that make up the SPI bus.Data is sent via MOSI (Master Out Slave In) and received via MISO (Master In Slave Out), with chip select (CS) lines used to address individual slaves. The system enables simultaneous communication between masters and their respective slaves.The arbiter module controls access to the SPI bus by granting access to either Master 1 or Master 2 based on their requests, making sure that only one master communicates at a time.

In order to integrate two masters, two slaves, and an arbiter, the SPI top module connects shared SPI signals (sclk, mosi, miso), as well as separate ss lines for each slave. The test bench generates a 100 MHz clock, initializes the system, and sequentially activates Master 1 and Master 2 to demonstrate communication. This design can be expanded for additional masters and slaves by changing the arbiter and adding ss lines, making it appropriate for applications such as sensor networks or multi-device communication[9].

E. SPI Multi Master and Multi Slave using Arbitration Algorithm:

In order to gather current information on SPI communication protocols, their drawbacks, and FPGA implementations,are determined based on this review. Verilog is then used in Quartos to construct the SPI master and slave modules[6]. High-priority masters are guaranteed low latency with fixed-priority arbitration, whereas low-priority masters run the danger of starvation. Access is granted according to predetermined master priorities. Dynamic-priority arbitration strikes a balance between efficiency and fairness by modifying priorities according to variables like waiting time or urgency, but its implementation is more complicated.

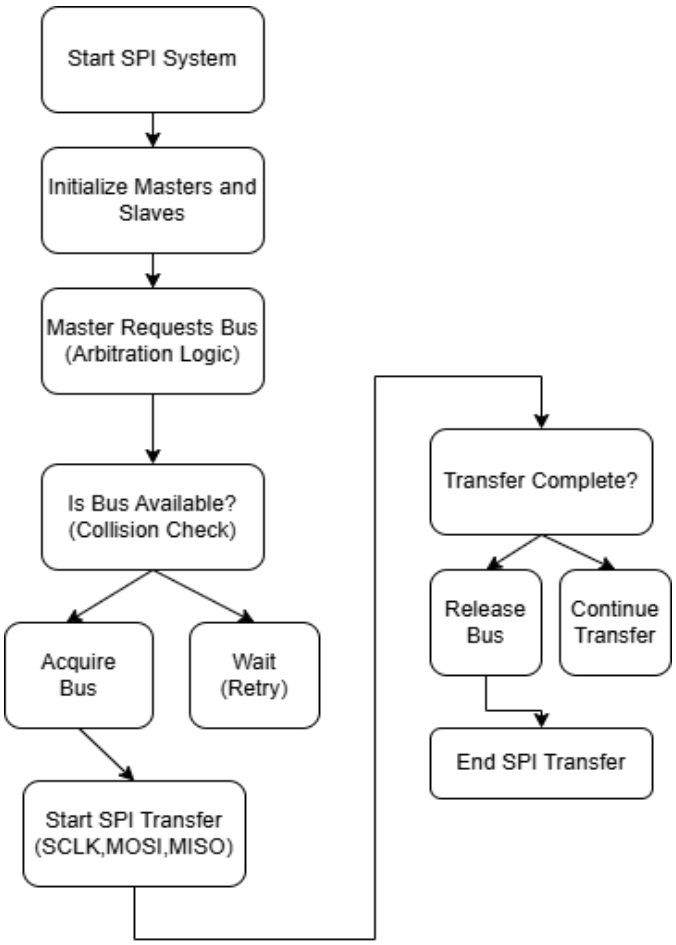

Figure 4. SPI Multi Master and Multi Slave operation flow chart

The Serial Peripheral Interface (SPI) communication procedure is described in the flowchart. Initializing Masters and Slaves is the first step. Arbitration logic verifies the availability of bus access after the Master asks it. The Master takes the bus if it is available; if not, it waits and tries again. Following acquisition, the SPI transfer starts with MISO (Master In Slave Out), MOSI (Master Out Slave In), and SCLK (clock) lines. The bus is released by the Master after the transfer is finished. The transfer continues if it is not completed. By effectively controlling bus contention and data transmission, the procedure guarantees device-to-device orderly communication.

The RTL representation of the design is transformed into a gate-level net list for FPGA implementation using Quartos. To simulate and validate the SPI modules' performance and functionality, test benches are built in Model sim. In order to test the physical implementation of the SPI modules, the gate-level net list is loaded onto the FPGA and configured if the simulations are successful[2].

F. Comparison of single master-slave and multi master and multi slave:

In generally recognized, Single Master-Slave SPI is straightforward, economical, and perfect for low-speed applications involving a single master and a single slave[1]. Multi-Master and Multi-Slave SPI requires arbitration for bus access and is more complicated, scalable, and appropriate for high-speed systems with several masters and slaves.

TABLE I: DIFFERENCE BETWEEN SINGLE MASTER-SLAVE AND MULTI MASTER AND MULTI SLAVE OPERATION.

Aspect	Single Master-Slave	Multi Master and Multi Slave
Complexity	Simple	Complex
Bus Access	No Contention	Requires arbitration
Scalability	Limited	Highly Scalable
Data Throughput	Lower	Higher
Resource utilization	Minimal	Higher
Applications	Simple,low-speed system	Complex,high-speed system
Fault Tolerance	Less fault-tolerant	More fault-tolerant
Cost	Lower	Higher

As demonstrated in above Table 1 , the single master-slave SPI system is straightforward, free of bus contention, inexpensive, and uses little resources, which makes it appropriate for low-speed applications[5-8]. It is less fault-tolerant, has a lower data throughput, and is not as scalable. Even though a multi-master multi-slave SPI system is more complicated and requires arbitration for bus access, it

is perfect for high-speed applications due to its high scalability, better data throughput, and greater fault tolerance.

RESULTS AND DISCUSSION

The vesilog design and simulation is demonstrated effective arbitration and data transport with bus contention by simulating a multi-master and multi-slave SPI communication system. While the arbiter made sure that correct bus access was maintained by allocating control to one master at a time based on priority, Masters 1 and 2 spoke with Slaves 1 and 2, respectively. The simulation results verified that the data was sent and received correctly. Additionally, it is possible to decrease delays and improve the effectiveness of single master-slave communication. This confirmed that the system could manage communication between multiple masters and slaves, and we obtained some simulated outputs, which are displayed below.

A. Simulation results of single master-slave operation:

The simulation results are shown in the corresponding Fig.5 and 6. The Cadence tool has been successfully used to synthesize the data transmission from the slave to the master and from the slave to the master. The communication system's operation is confirmed by these results, which show that data is transferred accurately between the master and slave devices. The figures validate the efficacy of the architecture in managing SPI communication by showing the timing and integrity of the data exchange.

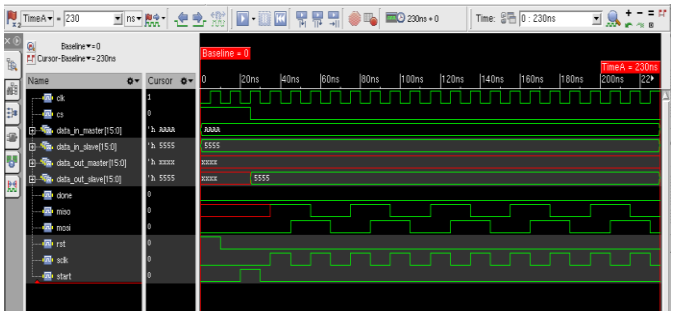

Figure 5. Simulation of spi single master and slave communication.

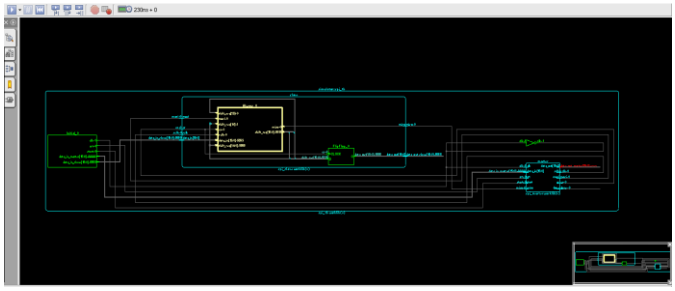

Figure 6. Block design of spi single master and slave communication.

A test bench was used to resemble the SPI master and slave modules, as seen in Fig. 5 and 6. The slave responded with 16'h5555 and the master transmitted 16'hAAAA. The slave successfully shifted in the master's data and sent its response, while the master succeeded in creating the clock (sclk), sending and receiving data via mosi and miso. The `done` flag signified that the transmission was finished, and the `cs` signal managed the communication. The SPI communication system's operation was validated by the simulation, which verified precise 16-bit data interchange between the master and slave.

B. Simulation results of multi master-multi slave operation:

The results and simulation of spi multi master and multi slave are displayed below.

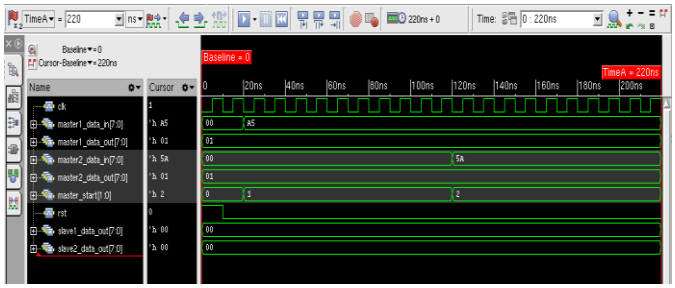

Figure 7. Simulation of spi multi master and multi slave communication.

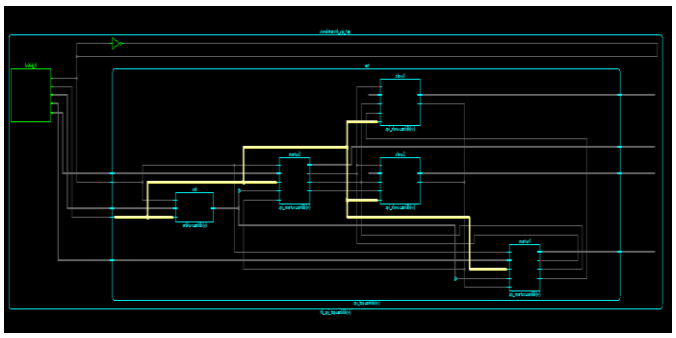

Figure 8. Block design of spi multi master and multi slave communication.

As illustrated in Figures 7 and 8, Two masters, two slaves, and an arbiter to control bus access make up the SPI system. In order to prevent conflicts, the arbiter allows access to one master at a time in response to requests. Slaves process data when chosen by `ss`, and masters transmit data to slaves by `mosi` and get responses via `miso`. Master 1 is activated to send `8'hA5` and receive `8'hAA`, and Master 2 is activated to send `8'h5A` and receive `8'h55` after the test bench initializes the system. The simulation validates the multi-master multi-slave SPI communication by confirming proper data transmission and reception. and successfully replicated in the cadence tool by effectively.

CONCLUSION

A multi-master, multi-slave communication system is designed and simulated using Cadence to show off its scalability and efficiency. Strong arbitration and error handling are features of protocols like SPI and I2C that provide dependable data exchange. According to simulation studies, it has low latency, high throughput, and optimum power, which makes it appropriate for multi-core, automotive, and Internet of Things applications. Cadence's aptitude for creating cutting-edge communication systems for upcoming digital designs is demonstrated in Future.

ACKNOWLEDGMENT

The authors sincerely thank the Research Centre, Department of ECE, B.N.M Institute of Technology, Karnataka, India, for the assistance in carrying out the aforementioned work.

References

[1]. Prasad, T. Durga, and B. Ramesh Babu. "Design and Simulation of SPI Master/Slave Using Verilog HDL." Int. J. Sci. Res 3.8 (2014).

[2]. A. H. Rahimi et al., "Design and Analysis of Single Master Multiple Slave Serial Peripheral Interface (SPI) on FPGA," 2024 IEEE International Conference on Applied Electronics and Engineering (ICAEE), Shah Alam, Malaysia, 2024, pp. 1-7, doi: 10.1109/ICAEE62924.2024.10667616.

[3]. A. K. Oudjida, M. L. Berrandjia, A. Liacha, R. Tiar, K. Tahraoui and Y. N. Alhoumays, "Design and test of general-purpose SPI Master/Slave IPs on OPB bus," 2010 7th International Multi-Conference on Systems, Signals and Devices, Amman, Jordan, 2010, pp. 1-6, doi: 10.1109/SSD.2010.5585592.

[4]. A. Ounissi and N. M. Ben Romdhane, "Adaptive finite-time control of master-slave manipulators with time-varying delay," 2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Monastir, Tunisia, 2020, pp. 163-168, doi: 10.1109/STA50679.2020.9329356.

[5]. Z. Wang, T. Zhao and Z. Shu, "Communication Scheme of Master-Slave Control System for Cascaded Inverter," 2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA), Chengdu, China, 2022, pp. 94-99, doi: 10.1109/ICIEA54703.2022.10005935.

[6]. Kaith, D., Patel, J. B., & Gupta, N. (2018). An Implementation of I2C Slave Interface using Verilog HDL. Internatioal Journal of Modern Engineering Research, ISSN, 2249-6645.

[7]. D. Li and C. N. Man Ho, "Master-Slave Control of Parallel-Operated Interfacing Inverters Based

on Wireless Digital Communication," 2018 IEEE Energy Conversion Congress and Exposition (ECCE), Portland, OR, USA, 2018, pp. 1466-1472, doi: 10.1109/ECCE.2018.8557410.

[8]. 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, doi: 10.1109/ICSPC51351.2021.9451702.

[9]. Kumar, KV Ashok, and M. Santosh Krishna. "Design and Functional Verification of A SPI Master Slave Core using UVM." International Journal of Scientific Engineering and Technology Research 4.51 (2015): 11023-11030.

[10]. Z. Zhou, Z. Xie, X. Wang and T. Wang, "Development of verification envioronment for SPI master interface using SystemVerilog," 2012 IEEE 11th International Conference on Signal Processing, Beijing, China, 2012, pp. 2188-2192, doi: 10.1109/ICoSP.2012.6492015.

[11]. 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, doi: 10.1109/ICCUBEA.2018.8697415.

[12]. Aditya, K., et al. "Design and functional verification of a SPI master slave core using system verilog." International Journal of Soft Computing and Engineering 2.2 (2012): 558-563.

[13]. M. B. Aykenar, G. Soysal and M. Efe, "Design and Implementation of a Lightweight SPI Master IP for Low Cost FPGAs," 2020 28th Signal Processing and Communications Applications Conference (SIU), Gaziantep, Turkey, 2020, pp. 1-4, doi: 10.1109/SIU49456.2020.9302434.

[14]. C. Liu, Y. Cao and J. Feng, "Design and Implementation of Master-slave Network Communication Based on TCP and UDP," 2017 International Conference on Computer Technology, Electronics and Communication (ICCTEC), Dalian, China, 2017, pp. 1253-1256, doi: 10.1109/ICCTEC.2017.00273.