Implementation of SPI Protocol with Adaptive Baud Rate

Archana B S[1], Dr. Kiran Bailey[2]

[1]*PG Scholar, Department of ECE, BMS College of Engineering, Bengaluru-560004, India*

[2]*Assistant Professor, Department of ECE, BMS College of Engineering, Bengaluru-560004, India*

***Abstract*— The Serial Peripheral Interface (SPI) protocol is a widely used synchronous communication protocol that enables high-speed data exchange between microcontrollers and peripheral devices. This project focuses on the implementation of the SPI protocol with an adaptive baud rate using Verilog. The adaptive baud rate functionality ensures flexibility by dynamically adjusting the communication speed based on the requirements of the connected devices, enhancing performance and compatibility. The Verilog-based SPI implementation includes modules for Master-Slave configuration, clock generation, data transmission, and reception, as well as an adaptive baud rate controller. The adaptive baud rate module dynamically adjusts the SPI clock frequency using programmable settings, ensuring seamless communication across devices with different speed requirements.**

***Index Terms*— SPI protocol, Adaptive baud rate, Master-slave, clock generation, Verilog, data transmission.**

I. INTRODUCTION

The Serial Peripheral Interface (SPI) protocol is one of the most common communication methods used in embedded systems, especially for quickly moving data between a main device and one or more connected devices. This protocol works in full-duplex mode, which means data can be sent and received at the same time. That makes it great for real-time uses where fast and dependable communication is needed. Usually, SPI uses a fixed speed for sending data, but changing the speed depending on what the system needs can greatly improve how well the system works and how flexible it is.

Using SPI with a speed that can change on the fly has some big benefits.

It helps when the system needs to adjust its communication speed based on what's required, like saving power or sending more data faster. By letting the speed change during communication, the system can run better without needing new hardware or someone to manually change settings. This is especially helpful in systems with many devices that need different speeds or for apps that switch between fast and slow modes while they're running.

In this setup, the SPI master creates the clock (SCLK) signal using a control module that changes the baud rate by adjusting the clock speed based on specific conditions. The baud rate can be changed either through signals from outside the system or based on what the system needs, which lets the system switch between different data speeds smoothly without stopping communication. The system also has ways to keep the data accurate, making sure the clock and data lines stay in sync all the time. The design includes a detailed control system that handles when data is sent and received, keeping the master and slave devices properly connected. It also allows for multiple slave devices to be connected, making the system flexible and able to be used in many different embedded applications.

An adaptive baud rate system helps make data transfer more efficient. In many embedded systems, there are situations where the system can use a slower baud rate when it's not busy or when fast data transfer isn't needed. However, when quick communication is necessary, the system can change to a faster baud rate, which makes sending data quicker and improves overall performance.

The protocol also allows for customizable data order, which means data can be sent either starting with the most significant bit (MSB) or the least significant bit (LSB). This makes the SPI more flexible for different devices and data formats, helping engineers design better embedded systems.

In systems where using a lot of power is a big issue, SPI is a good option because it uses very little power. It only uses power when it's actually sending or receiving data, which makes it great for devices that run on batteries or have limited energy. SPI is also simple to use, which makes it perfect for teaching and testing new ideas. Because of this, engineers and developers can easily set up and work with SPI systems, making it a popular choice for learning and building new projects.

II. BLOCK DIAGRAM

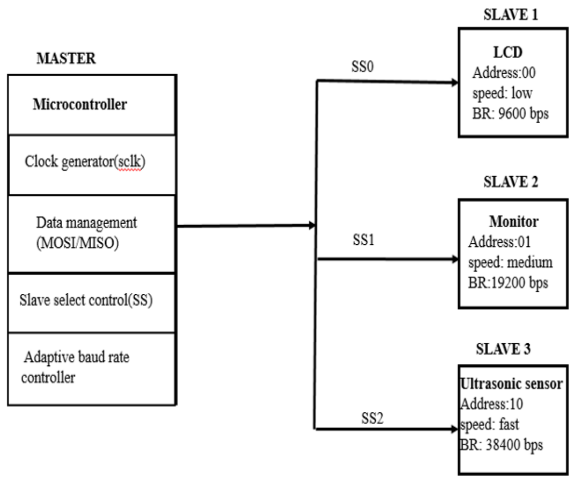

Fig.1.Block Diagram

The master module is the main control part for SPI communication and has several important parts. It includes a microcontroller that manages data sending and receiving with the connected slave devices. There's a clock generator (SCLK) that creates the SPI clock signal and can change the speed to match different data transfer rates. A data management part handles sending and receiving data at the same time through the MOSI and MISO lines. A slave select control part turns on the needed slave by using the SS0, SS1, or SS2 lines so only one slave is active at a time. An adaptive baud rate controller changes the clock speed by adjusting the clock division based on certain conditions, allowing the system to change data transfer speeds quickly for better performance and lower power use. The system connects three SPI slave devices, each with a unique address and different data transfer speed needs.

Slave 1 (LCD, Address 00) works at a slow speed of 9600 bps for basic display updates. Slave 2 (Monitor, Address 01) runs at a medium speed of 19200 bps for moderate data transfer. Slave 3 (Ultrasonic Sensor, Address 10) uses a fast speed of 38400 bps for quick sensing. Each slave listens for the master's clock signal and only responds when its slave select line is active, making data transfer reliable and in sync with the clock.

The baud rate control part is important because it creates clock signals that control the SCLK speed.

It uses counters and special logic to divide the main clock, and it can adjust the data transfer speed as needed depending on which slave is active or the system's power and speed needs. The clock management part splits the main clock into different speeds and picks the correct one based on signals from the baud rate controller, allowing the system to switch between transfer speeds without stopping communication.

When the master needs to talk to a slave, it activates the right SS line, sets the clock speed to match the slave's needs, and manages sending and receiving data through the MOSI and MISO lines.

Once communication is done, the SS line is turned off, letting the master choose another slave with a different speed if needed. This setup allows the system to handle various data transfer needs efficiently and flexibly.

III. ADAPTIVE BAUD RATE CONTROLLER

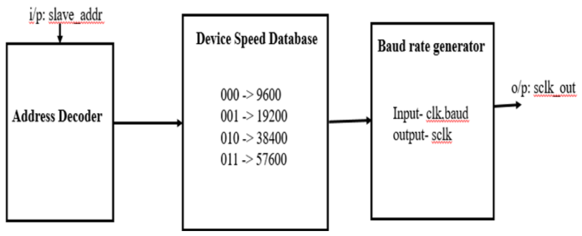

Fig.2.Adaptive baud rate controller.

The system starts with an Address Decoder that takes the slave device address (slave_addr) as input. This address input helps choose which connected slave device will communicate. The decoder checks the incoming address against set codes to find out which slave is active. It then sends this decoded address to the Device Speed Database block.

The Device Speed Database connects each slave address to its needed communication speed. For instance, address 000 uses 9600 bps, 001 uses 19200 bps, 010 uses 38400 bps, and 011 uses 57600 bps. This setup lets the system pick the right speed for each slave.

Once the speed is set, the information goes to the Baud Rate Generator block. This block uses the system clock (clk_band) and the set baud rate to create the correct SPI clock (sclk). It changes the base clock to match the required speed for the selected slave. The output from the Baud Rate Generator is the sclk_out signal used for SPI communication. This ensures each slave talks at its best speed without needing manual clock setup.

By using an address-based lookup and automatic clock generation, the system can change baud rates automatically. This design makes the system more efficient and flexible for multi-slave SPI setups. It also allows smooth switching between devices that use different speeds. This method is all hardware-based and doesn't need software changes while the system is running. This type of setup is helpful in embedded systems where various peripherals need different communication speeds.

IV. STATE DIAGRAM

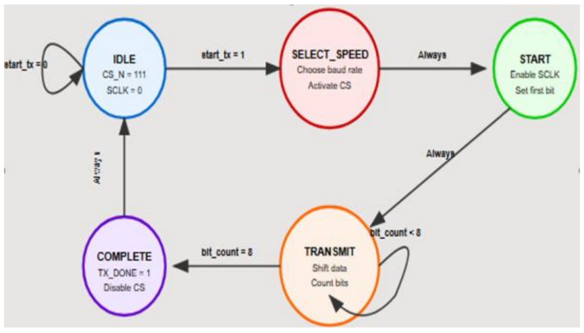

Fig.3.SPI master module for adaptive baud rate.

This state diagram shows how an SPI master works with a system that can change the speed of data transfer. The system starts in the IDLE state, where all the chip select lines are not active (CS_N = 111), the clock signal (SCLK) is turned off, and no data is being sent. When the master is told to start sending data (start_tx = 1), it goes to the SELECT_SPEED state. In this state, it reads the address of the slave device, figures out the right speed for communication, and turns on the correct chip select line.

Then the master moves to the START state, where it turns on the SPI clock (SCLK) and gets ready to send the first bit of data.

Next, it enters the TRANSMIT state, where it sends out each bit one after the other and keeps a count of how many bits have been sent. Once all eight bits are sent (bit_count = 8), the master goes to the COMPLETE state. Here, it turns off the chip select line, marks the transmission as done (TX_DONE = 1), and puts the received data into a register (rx_data).

Finally, the master goes back to the IDLE state, ready for the next time it needs to send or receive data.

This process helps make SPI communication reliable and allows the master to adjust the data speed based on which slave device is being used, making data transfer more efficient for devices that need different speeds.

To create a baud rate of 9600 bps for SPI communication, you need a clock divider that works with a 50 MHz system clock.

The formula to calculate the baud rate is:

Baud Rate = System Clock / (2 × Clock Divider)

To get 9600 bps with a 50 MHz clock:

Clock Divider = 50,000,000 / (2 × 9600) = 2604

This means the 50 MHz system clock must be divided by 5208 (which is 2 × 2604) to produce the correct SPI clock frequency.

The actual baud rate is:

Actual Baud Rate = 50,000,000 / (2 × 2604) = 9600 bps

This shows the calculated clock divider gives the right baud rate for reliable SPI communication.

This method can also be used to adjust the baud rate for other speeds by simply changing the clock divider value based on the slave device's requirements.

V. SIMULATION RESULTS

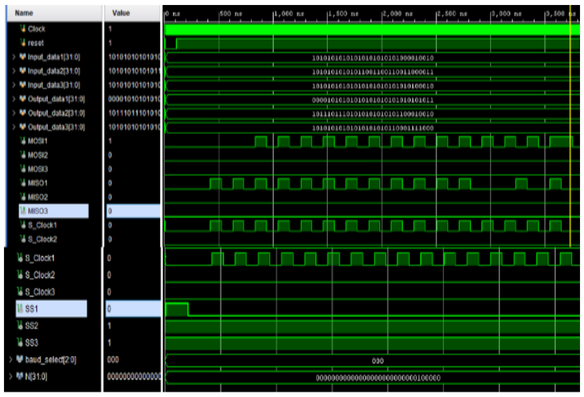

Fig.4. if Baud_select=000,9600 bps, clk_divider=2604

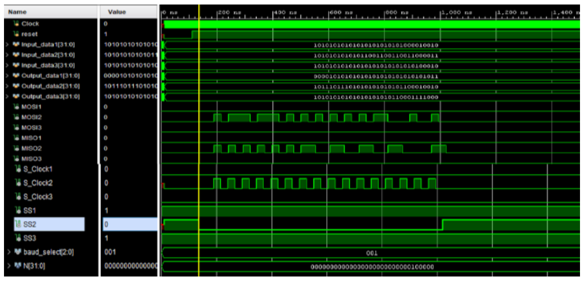

Fig.5. if Baud_select=001,19200bps,clk_divider=1302

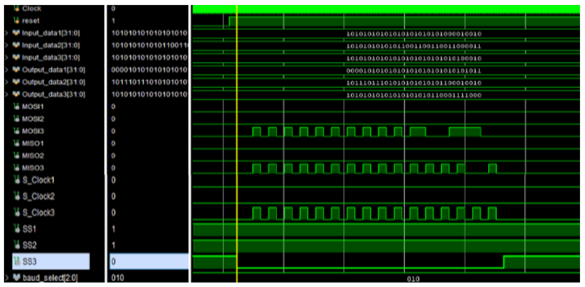

Fig.6. if Baud_select=010,38400 bps, clk_divider=862

Standard mode	9600b/s
Fast mode	19200b/s
High Speed mode	38400b/s
Ultra Speed mode	112500b/s

VI. CONCLUSION

In conclusion, using Verilog to implement the SPI protocol with an adaptive baud rate shows how effective Verilog is in creating efficient communication systems. The system allows for reliable full-duplex communication between one master device and multiple slave devices, with accurate synchronization and data transfer. The adaptive baud rate feature makes the design more flexible, letting the master change the communication speed as needed, which improves performance and saves power. The simulation results confirm that the system works correctly and meets all timing requirements, ensuring data is sent without errors even when the baud rate changes. This design offers a scalable and customizable SPI interface that can be used in many different embedded system applications, proving that Verilog is a good choice for building high-performance and dependable communication protocols.

VII. FUTURE SCOPE

The SPI protocol with an adaptive baud rate can be made even better in different ways. Future improvements might include adding multi-master support to make communication more flexible and dynamic slave detection to automatically set up peripherals. Using error detection methods like CRC can help keep data accurate. The adaptive baud rate controller could get smarter with algorithms that look at real-time traffic and power use. A power management system can help save energy in devices that run on batteries. Moving the design to low-power FPGAs or ASICs can make it more practical. Testing the hardware in real situations will make sure it works well in actual use. These changes are meant to make the protocol stronger, more flexible, and better suited for today's embedded systems.

REFERENCE

[1] Shama, Ayman, Manar Lashin, and Ayman Nada. "A State Machine-Based Approach for Implementing SPI Communication on FPGAs." Benha Journal of Applied Sciences, vol 9, issue 5, 2024, pp 127-134.

[2] Kumar, K. Charan, et al. "Design of Low Power SPI Protocol using Clock Gating Techniques." 2024 International Conference on Emerging Technologies in Computer Science for Interdisciplinary Applications (ICETCS). IEEE, 2024.

[3] Naveen, R. S., S. Sanjay, and C. Mukuntharaj. "Implementation Of SPI Protocol with Adaptive Baud Rate Using Verilog.", 7th International Conference on Devices, Circuits and Systems (ICDCS), 2024.

[4] Mayank Trehan, Pradeep Kumar, Nidhi Gaur. “Design and Analysis of Multi-Protocol Conversion Unit for SPI, SPI and UART”. 2nd International Conference on Device Intelligence, Computing and Communication Technologies, 2024.

[5] A. H. Rahimi, A. K. Halim*, A. H. A. Razak, M. F. M. Idros, F.N. Osman, S. A. M. A. Junid, S. L. M. Hassan. “Design and Analysis of Single Master Multiple Slave Serial Peripheral Interface (SPI) on FPGA”. IEEE International Conference

on Applied Electronics and Engineering (ICAEE), 2024.

[6] Nishant Sahay1, Sachin Gajjar2. “Design and UVM based Verification of UART, SPI, and SPI Protocols”. 5th International Conference on Smart Electronics and Communication (ICOSEC 2024), 2024.

[7] Dr. S Sasikumar, B.Aravind Balaji, N.Jaya Krishna, Sidda Reddy. “Design and Implementation of a Protocol Conversion Unit for SPI to SPI Communication”. Proceedings of the 8th International Conference on Electronics, Communication and Aerospace Technology (ICECA 2024), 2024.

[8] Patra, Aritra, and Lalit Mohan Saini. "Analysis of Serial Peripheral Interface." 2023 Second IEEE International Conference on Measurement, Instrumentation, Control and Automation (ICMICA). IEEE, 2024.

[9] Josip Zidar∗, Ivan Aleksi∗, Tomislav Matic∗ “Analysis of energy consumption for SPI and SPI communications in ultra-low power embedded systems”. MIPRO 2023, May 22 - 26, 2023, Opatija, Croatia, 2023.

[10] Ionelia-Bianca Brezeanu, Cătălin Botezatu, Florin Drăghici, Gheorghe Brezeanu. “Improved SPI Controlled, Low-Voltage, High Speed, Multi-Channel Switch”. 2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 2022.

[11] Bitty Jose, J.Samson 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), 2021.

[12] Jiayi Qiang1, Yong Gu2 and Guochu Chen. “FPGA Implementation of SPI Bus Communication Based on State Machine Method.” Journal of Physics: Conference Series,2020.

[13] Jiang Yang, Yile Xiano, Dejian Li, Zheng Li, Zhijie chen, Peiyuan wan. “A configurable SPI interface based on APB bus”. IEEE 14th International Conference on Anti-counterfeiting, security, and identification (ASID), 2020.

[14] Mehmet Burak AYKENAR, Gökhan SOYSAL Murat EFE. “Design and Implementation of a Lightweight SPI Master IP for Low Cost FPGAs”. 28th signal processing and communications Applications conferences (SIU), 2020.

[15] Dvijen Trivedi, Aniruddha Khade, Kashish Jain, Ruchira Jadhav. “SPI to SPI Protocol Conversion using Verilog”. Fourth international conference on computing communication control and automation(ICCUBEA), 2018.