Dean&Francis
ISSN 2959-6157

The Hardware Implementation of a Register-configurable SPI Interface

Zefeng Liu

Department of Electronic Information Engineering, Beijng University of Posts and Telecommunications, Beijing, China

Corresponding author: jp2022213543@qmul.ac.uk

Abstract:

The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol commonly used in embedded systems, facilitating the connection between microcontrollers and peripheral devices such as sensors, displays, and memory. Traditional SPI hardware implementations often lack flexibility in clock frequency, clock polarity (CPOL), and clock phase (CPHA), limiting their adaptability in various applications and increasing development time and cost. To address these issues, this study designs and implements a configurable SPI module that can dynamically adjust parameters such as clock frequency, data bit width, CPOL, and CPHA through register control. The core of this design adopts a modular architecture that allows flexible configuration via register control, ensuring compatibility with different SPI modes and peripheral devices. Simulation tests have verified that the SPI module achieves the expected dynamic flexibility across multiple operation modes while ensuring communication accuracy and system stability. The configurable SPI module enhances the adaptability of embedded system communication protocols, reducing the need for hardware redesigns for specific applications. This register-based dynamic SPI design holds significant practical value in engineering applications and helps improve hardware development efficiency.

Keywords: SPI; embedded systems; dynamic configuration; register control.

1. Introduction

SPI (Serial Peripheral Interface) is a synchronous serial communication protocol widely used in embedded systems to enable fast data transmission between master and slave devices.

Embedded system applications are becoming more complex. This increases the demand for greater configurability and flexibility in SPI modules. Design optimizations focus on multi-mode switching, adjustable data transmission rates, and bit widths. In existing research, Shingare and Patil introduced an FPGA-based SPI implementation in their study. [1] This design realized a modular SPI circuit using Verilog hardware description language and employed a state machine to control SPI bus communication. However, this research primarily focused on the ba-

ISSN 2959-6157

sic functionality of SPI and did not explore the dynamic configuration capabilities of the SPI module in depth, especially in terms of adjusting clock modes and bit widths dynamically, where there is still room for further optimization.

In addition, Yang Jiang et al. designed a configurable SPI interface based on the APB bus. [2] This solution enabled master-slave mode switching, clock mode switching, and MSB/LSB communication mode switching via the APB bus, supporting variable transmission lengths from 1 to 32 bits. The design adopted a finite state machine to control the transmission timing of the SPI interface. Through RTL simulation and FPGA verification, the results showed that the design performed well in terms of functionality and data transmission stability.

Based on the above research background, this study aims to improve the design of existing SPI modules by combining the advantages of state machine methods and configurable register. The study implements dynamic adjustments of clock frequency, data bit width, and transmission mode through register configuration. It aims to provide an efficient, reliable, and flexible SPI communication solution. This solution meets the needs of different embedded system application scenarios.

2. Research of Objectives

2.1 Support for Multiple SPI Communication Modes

Achieve full support for the four SPI protocol modes (the four combinations of CPOL/CPHA), enabling the module to be compatible with different slave devices and improving system versatility. Automatically adjust the data sampling and transmission timing based on CPOL and CPHA settings to ensure the accuracy and reliability of data communication.

2.2 Dynamic Adjustment of SPI Clock Frequency

Enable dynamic adjustment of the SPI clock frequency through register settings, supporting multiple frequency levels to accommodate communication needs at different rates. Design a clock divider module that can generate the required SPI clock signal based on the set division factor.

2.3 Adjustable Data Transmission Bit Width

Support flexible configuration of data transmission bit width, allowing the selection of different data lengths such as 8-bit, 16-bit, 24-bit, or 32-bit according to application requirements. Design shift registers and control logic to ensure accurate data transmission and reception at different bit widths.

3. System Design and Module Implementation

This study designed the SPI module using Verilog HDL, combined with ModelSim for simulation and verification. Although VHDL is widely used in FPGA design, Verilog also performs excellently in digital circuit design due to its simplicity and efficiency. Leal-del Río et al. adopted a similar approach in their research, implementing SPI and I2C protocols using Verilog and demonstrating its advantages in terms of data transmission speed and hardware resource utilization[3][4].

The SPI module design in this study adopts a modular and parameterized architecture to enhance system flexibility and scalability. The overall design mainly includes functional units such as the clock generation module, state machine control module, data transmission module, register configuration module, and control signal generation module. These modules work together to achieve the dynamic configuration and stable transmission of SPI communication. The state machine control module plays a key role in this design, controlling the entire process of data transmission through state transitions. The state machine method proposed by Qiang Jiayi et al. (2020) has been successfully applied to FPGA implementations of SPI buses, proving its effectiveness in synchronous serial communication [5].

3.1 Key Module Design

First, the clock generation module is one of the core components of the system. It generates the SPI clock signal by setting the division factor via registers. This module can dynamically adjust the clock frequency according to system requirements to support different communication rates. Additionally, the module configures the initial clock level based on the CPOL (clock polarity) parameter in the register, ensuring the correct execution of the SPI communication protocol.

Second, the state machine control module serves as the control center for the SPI communication process. The state machine ensures that the SPI module transitions sequentially between different states, including initialization, idle, start, data transmission, stop, and error handling. Each state has a clearly defined function. For example, in the INIT state, the SPI module initializes relevant parameters based on the register settings; in the TRANSFER state, the module performs data transmission and reception operations based on the CPOL and CPHA (clock phase) configurations; and in the STOP state, the system completes communication and returns to the initial idle state.

Next, the data transmission module consists of two shift registers for transmission and reception, respectively used for MOSI (Master Out Slave In) data transmission from

master to slave and MISO (Master In Slave Out) data reception from slave to master. During transmission, the shift registers shift data in or out bit by bit on the rising or falling edge of the clock signal, enabling synchronous data transmission. This module supports different data transmission widths, allowing the selection of 8-bit, 16-bit, 24-bit, or 32-bit data lengths based on register configurations, ensuring system flexibility and compatibility.

The register configuration module is crucial for realizing the reconfigurability of the SPI module. This module adjusts SPI parameters such as clock frequency, data bit width, CPOL, and CPHA by setting registers, enabling dynamic adjustments for SPI communication. Controlled by the state machine, the configuration values in the registers are loaded into the control module during the system initialization phase, allowing the SPI module to be adjusted in real-time according to application requirements, thus improving system adaptability.

3.2 Detailed Explanation of State Machine Design

The state machine is the core control unit of the SPI module, used to control the data transmission process and the operating states of the SPI module. Through state transitions, the SPI module executes the complete communication process, from initialization to data transmission and then to the end, based on different operation commands and status signals, ensuring system stability and reliability. The entire state machine design is divided into several key states: initialization (INIT), idle (IDLE), start (START), data transmission (TRANSFER), stop (STOP), and error handling (ERROR). The function and role of each state are as follows:

3.2.1 Initialization state (INIT)

The initialization state is the starting state of the system. In this state, the SPI module sets various initial parameters of the system, including clock polarity (CPOL), clock phase (CPHA), clock division factor (div_factor), and other parameters. These parameters are read from the registers and configured into the corresponding control modules. Meanwhile, the chip select signal (CS) is pulled high, indicating that the SPI module is in an inactive state. After completing all necessary initialization operations, the state machine switches to the idle state (IDLE).

3.2.2 Idle state (IDLE)

When the SPI module is in the idle state, the system is in standby mode, waiting for an external start signal (start) to arrive. In this state, the clock signal maintains stable output, and the data registers and other control parameters remain unchanged. Once the start signal is detected, the state machine immediately switches from the IDLE state to the start state (START), preparing for data transmission.

3.2.3 Start state (START)

The SPI module prepares to enter data transmission mode in the start state. First, the chip select signal (CS) is pulled low, indicating the start of SPI communication. Next, the data registers load the data to be transmitted, and the data shifting operation is prepared based on the configured bit width and clock settings. Once the data and control signals are ready, the state machine transitions to the data transmission state (TRANSFER).

3.2.4 Data transmission state (TRANSFER)

The data transmission state is the main working state of the SPI module. In this state, data is sent from the master device to the slave device via the MOSI (Master Out Slave In) line, or received from the slave device via the MISO (Master In Slave Out) line. The shifting of data is performed on the rising or falling edge of the clock signal, determined by the CPOL and CPHA settings. In each clock cycle, data is shifted out or in bit by bit until all data is transmitted. After the transmission is complete, the state machine switches to the stop state (STOP).

3.2.5 Stop state (STOP)

After the data transmission is completed, the system enters the stop state. In this state, the chip select signal (CS) is pulled high again, indicating the end of data transmission. Meanwhile, the SPI module sets the transmission completion flags (tx_done and rx_done) to indicate that the current transmission process has been successfully completed. At this point, the system returns to the idle state (IDLE), waiting for the next transmission request.

4. Simulation and Verification

4.1 Basic Functionality Verification

4.1.1 Set the rst signal to reset the module and check the initialization state (INIT).

ISSN 2959-6157

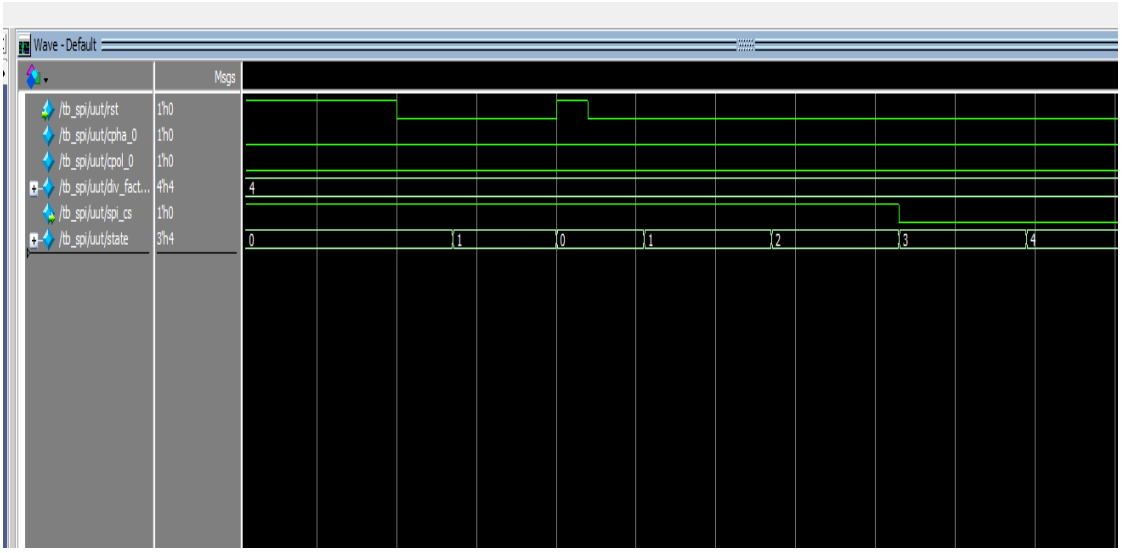

Fig. 1 INIT state

As shown in Figure 1, when the rst signal is pulled high, the state enters the INIT state, and information such as cpol, cpha, and div_factor is written into the registers. The chip select signal is pulled high, waiting for the start state, at which point it is pulled low to trigger the communication.

4.1.2 Configure div_factor, cpol, and cpha, and observe the generation of the SPI clock.

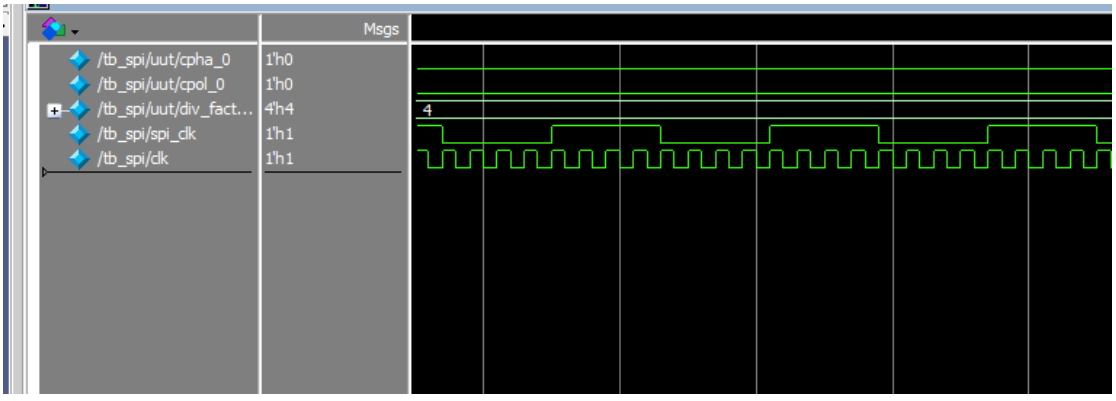

Fig. 2 Configurable parameters in the INIT state

As shown in Figure 2, when the division factor is set to 4 and both cpol and cpha are set to 0, the SPI clock signal is generated as illustrated in the figure.

4.1.3 Activate the start signal and verify the transition from the IDLE state to the START state.

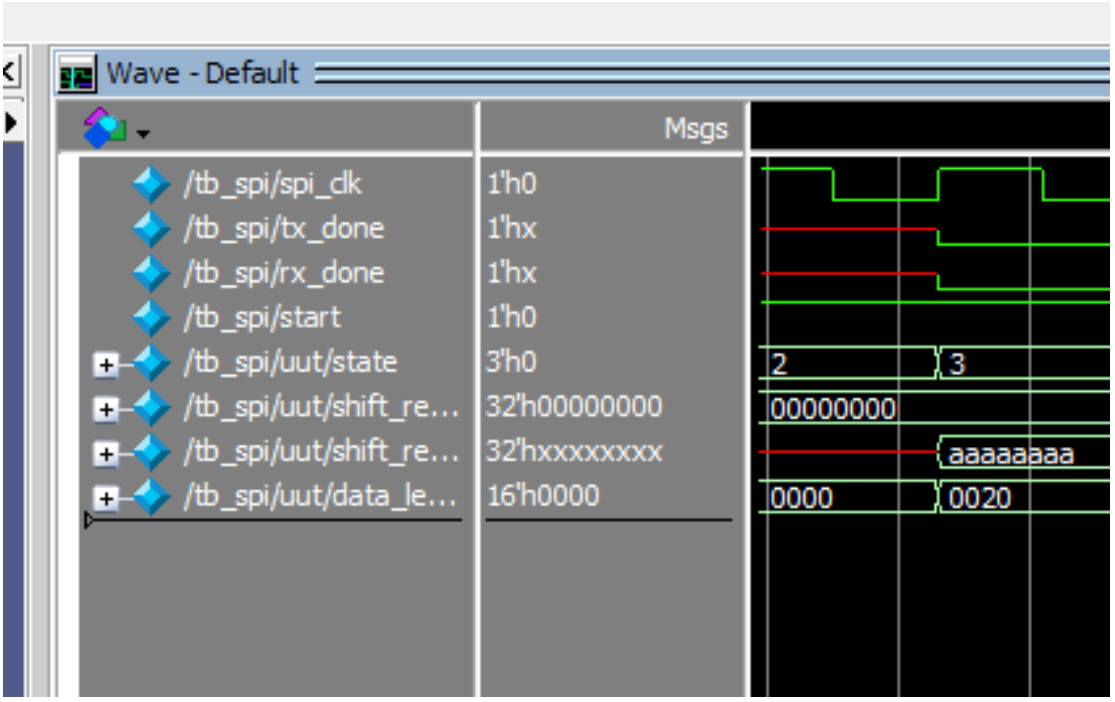

Fig. 3 START state

As shown in Figure 3, after entering the IDLE state, the state switches to START upon detecting that the start signal is pulled high after one clock cycle. In this state, the shift register is initialized, the transmission data is written into the shift register, and the length of the transmission data is written into the register. The tx_done and rx_done signals are pulled low, indicating that the transmission is not yet complete.

4.1.4 Send data via MOSI output and check the data shifting operation in the TRANSFER state.

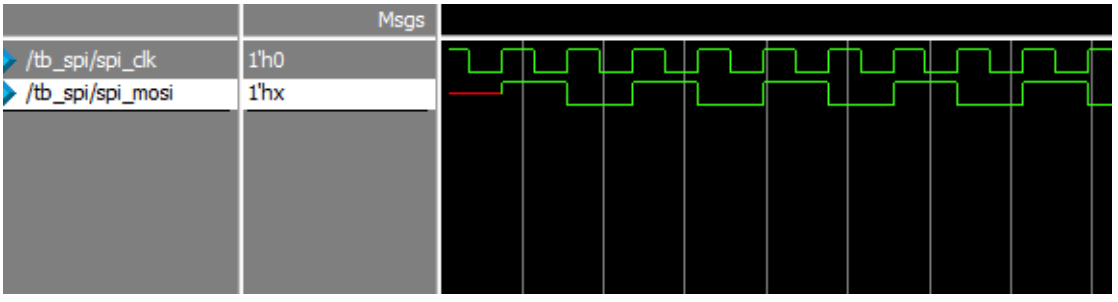

Fig. 4 MOSI signal in the TRANSFER state

As shown in Figure 4, with both cpol and cpha set to 0, 32 bits of data, “10101010 10101010 10101010 10101010,” are transmitted on the rising edge of the clock.

4.1.5 Receive data via MISO input and verify that the data is correctly shifted into the data_out register.

ISSN 2959-6157

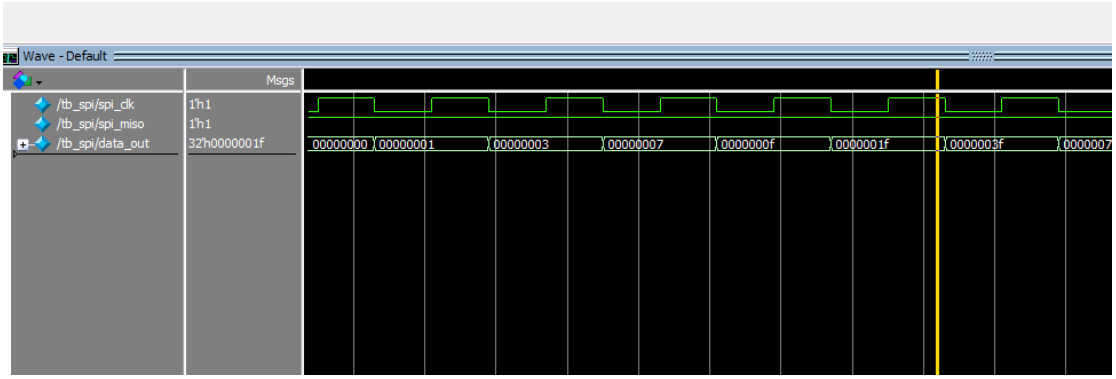

Fig. 5 MISO signal in the TRANSFER state

As shown in Figure 5, with both cpol and cpha set to 0, 32 bits of data are received on the MISO line along the falling edge of the clock.

4.1.6 After completing the transmission, check the STOP state and confirm that the tx_done and rx_done flags are set.

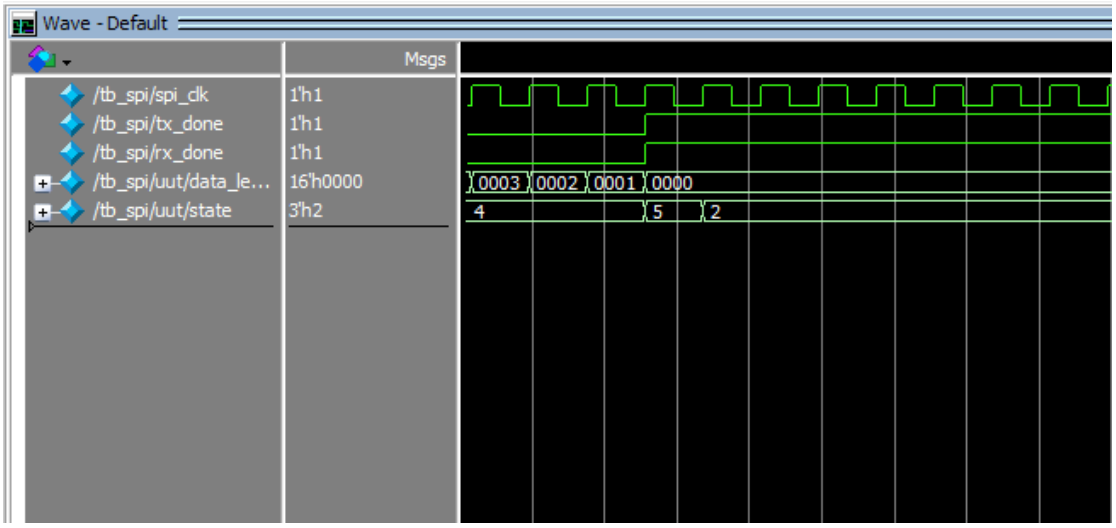

Fig. 6 STOP State

As shown in Figure 6, after the data length decreases to 0, the tx_done and rx_done signals are pulled high, and the state transitions to STOP.

4.2 Multi-mode Testing

Testing Objective: Verify the operation of SPI under different combinations of CPOL and CPHA, ensuring that data is correctly transmitted in all modes

4.2.1 Mode0(cpha=0 cpol=0)

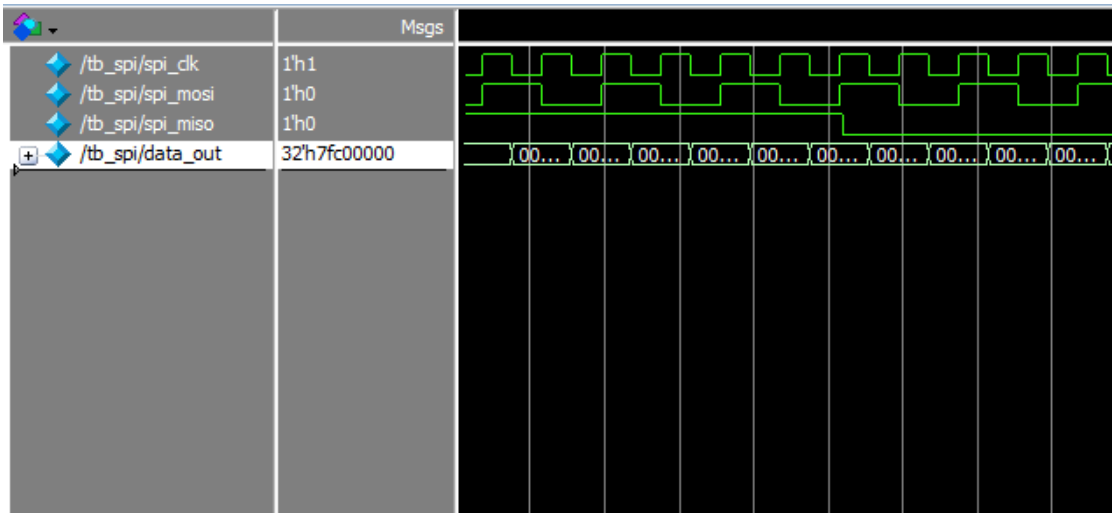

Fig. 7 Mode 0

As shown in Figure 7, in this mode, data is transmitted on the rising edge and received on the falling edge, and the clock signal is low during the idle state.

4.2.2 Mode1(cpha=1 cpol=0)

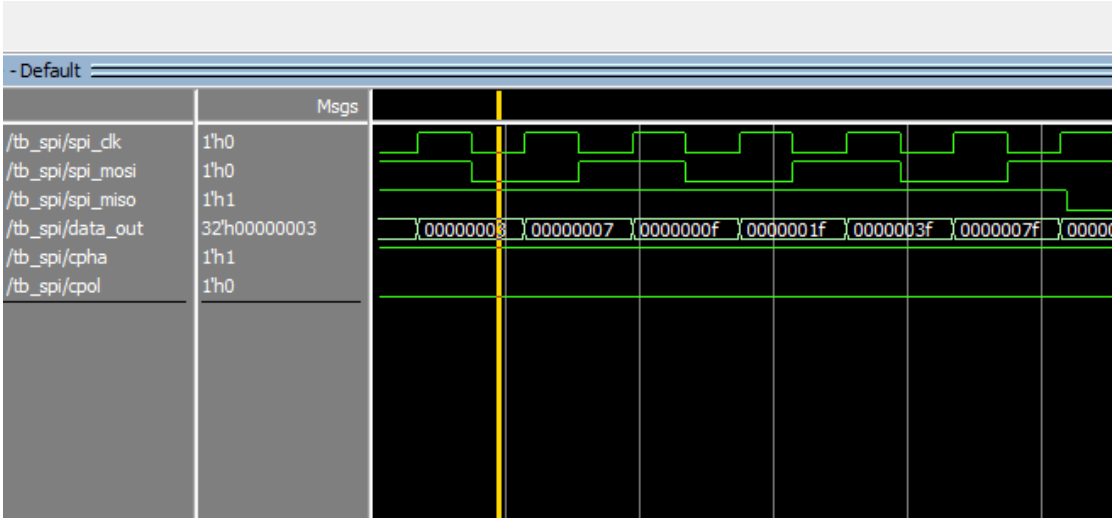

Fig. 8 Mode 1

As shown in Figure 8, in this mode, data is received on the rising edge and transmitted on the falling edge, and the clock signal is low during the idle state.

4.2.3 Mode2(cpha=0 cpol=1)

ISSN 2959-6157

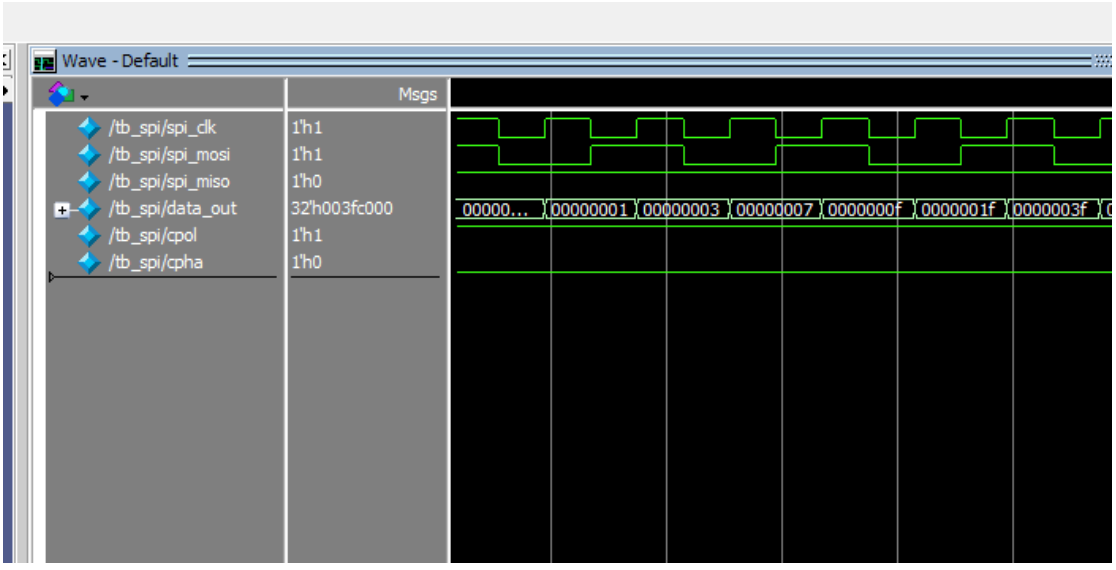

Fig. 9 Mode 2

As shown in Figure 9, in this mode, data is received on the rising edge and transmitted on the falling edge, and the clock signal is high during the idle state.

4.2.4 Mode3(cpha=1 cpol=1)

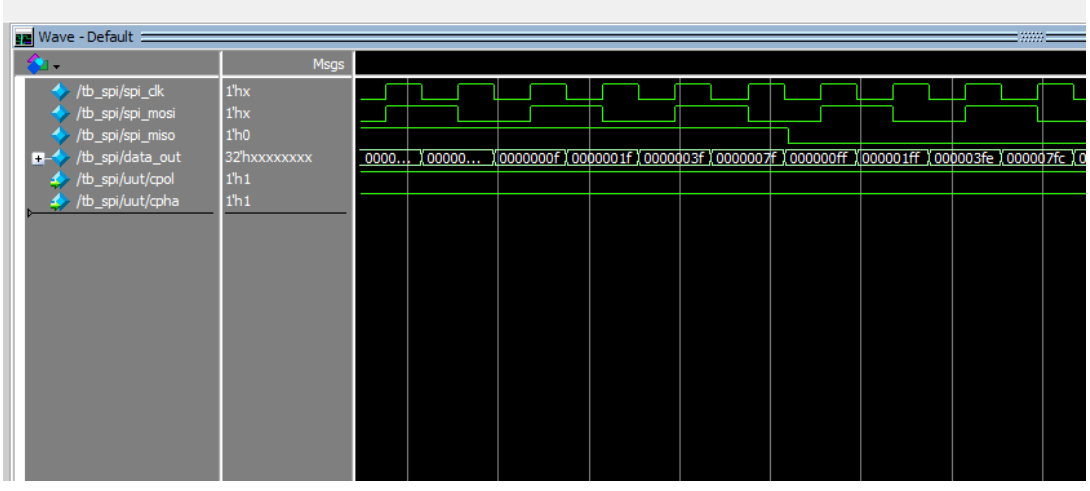

Fig. 10 Mode 2

As shown in Figure 10, in this mode, data is transmitted on the rising edge and received on the falling edge, and the clock signal is high during the idle state.

In conclusion, through simulation and verification, the design of the SPI module fully meets expectations and all functions have been successfully implemented.

5. Discussion

The practical significance of this study lies in the implementation of a configurable SPI module, which enhances the flexibility and adaptability of embedded systems in communication protocol design. It reduces the need for hardware redesigns and improves development efficiency. This module has broad potential for future applications, particularly in fields such as IoT devices, industrial automation, and sensor networks, where it can effectively meet diverse data communication requirements. Furthermore, this design contributes to the advancement of embedded systems towards greater flexibility and higher performance, providing an efficient and reliable solution for complex embedded application scenarios.

6. Conclusion

This report provides a detailed explanation of the design and implementation process of the configurable SPI hardware module, covering the background and significance of the project, system design, module implementation, and simulation and verification. It comprehensively demon-

strates all aspects of the SPI module. Through register configuration, the SPI module achieves dynamic adjustments for clock frequency, data bit width, clock phase (CPHA), and clock polarity (CPOL), significantly improving the adaptability and generality of the hardware.

In the module implementation, this study meticulously designed key modules such as clock signal generation, data transmission, state machine control, and register configuration, ensuring system stability and efficiency. The module exhibited excellent functionality and stability under different operating conditions through simulation and verification of various modes. The simulation results were fully consistent with expectations, confirming the correctness and reliability of the design.

The successful realization of this project not only addresses the fixed configuration limitations of traditional SPI hardware but also provides a flexible solution for the data communication needs in various application scenarios. The module operates stably across multiple frequencies, data bit widths, and communication modes, demonstrating strong potential for engineering applications.

Future work will focus on further optimizing the module's performance, including improving timing accuracy, reducing power consumption, and implementing and testing on actual hardware. Overall, this project provides an innovative and practical solution for SPI communication design in embedded systems, with significant engineering application value and broad prospects for further development.

References

[1] D. Trupti, R. T. ShingarePatil. SPI Implementation on FPGA. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 2013,2: 2278-3075.

[2] Jiayi Qiang, Yong Gu, Guochu Chen. FPGA Implementation of SPI Bus Communication Based on State Machine Method. Journal of Physics: Conference Series, 2020,1449.

[3] A. Maity, P. K. Samanta, B. P. De, S. K. Dash, W. Bhowmik and A. Bakshi, "FPGA Implementation of Serial Peripheral Interface Transceiver Module," 2024 International Conference on Computer, Electrical Communication Engineering (ICCECE), Kolkata, India, 2024:1-5

[4] Tatiana Leal-del Río, Gustavo Juarez-Gracia, L. Noé Oliva-Moreno, "Implementation of the communication protocols SPI and I2C using an FPGA by the HDL-Verilog language," Research in Computing Science, 2014, 75: 31–41.

[5] Yang Jiang, Yile Xiao, Dejian Li, Zheng Li, Zhijie Chen, Peiyuan Wan. A Configurable SPI Interface Based on APB Bus. 2020 IEEE 14th International Conference on Anti-counterfeiting, Security, and Identification (ASID).2020