

LOW POWER DESIGN OF SPI AND I2C PROTOCOL IN VERILOG HDL

Mohanapriya K.^{*}, Mohankumar M., and Suganya L.

Received: December 17, 2024; Revised: February 05, 2025; Accepted: February 26, 2025

Abstract

A method for sharing data between a slave and a master is the SPI (Serial Peripheral Interface). The result of this paper explores a high-speed SPI configuration that utilizes clock gating, drawing inspiration from Motorola's manual and is implemented in Xilinx's Virtex 5 FPGA using Verilog 2001. It covers design strategies for managing the SPI bus, including handling multiple slave devices. The document also explores an I2C (Inter-Integrated Circuit) single master with a bidirectional data line, supporting multiple masters and efficient communication between devices. The I2C master can transfer numerous bytes of data, controlled through a start signal and slave address. The design is in Verilog, simulated with Xilinx Vivado, and is parameterized for flexibility. The goal is to provide a deep understanding of SPI and I2C protocols, their applications, and implementation in FPGA devices, along with insights into their design and simulation process.

Keywords: Clock Gating Technique; I2C; Internal Protocols; SPI

Introduction

The synchronous serial interface technology known as SPI (Serial Peripheral Interface) was first presented by Motorola. A module called a Synchronous Serial Port is present in an MCU that is coupled to a peripheral microcontroller (PIC micro) that directly implements the SPI bus (Palnitkar, 2003). This module enables the MCU to communicate with a variety of peripheral devices at high speeds using full-duplex synchronous serial mode. When compared to other popular serial buses like I2C, CAN, USB, and others, SPI offers several benefits. For instance, SPI buses may transmit data up to numerous Mbps, which is faster than other serial buses (Qiang *et al.*, 2020). EEPROM, Flash, Analog to Digital Converter (ADC), Digital Signal

Processor (DSP), Real Time Clock (RTC), and Decoders are the key devices that take advantage of SPI. The construction and operation of the SPI communication bus are presented in this study, along with an analysis of its timing architecture and four operating modes. This study presents an innovative design methodology that integrates advanced clock gating techniques to minimize power usage while ensuring efficient data transfer. Unlike conventional implementations, our approach employs synthesis-based clock gating to dynamically adjust power consumption based on system requirements. The use of state machine techniques allows SPI bus communication to be implemented on FPGA. This study implements SPI

Department of ECE, Sri Eshwar College of Engineering, Coimbatore, Tamil Nadu 641202, India.
E-mail: mohanapriyak011@gmail.com^{*}; mail2mohanphd@gmail.com; l.suganyavikas2001@gmail.com

^{*} Corresponding author

DOI: <https://doi.org/10.55766/sujst7904>

Suranaree J. Sci. Technol. 32(1):010355(1-10)

and I2C designs using Verilog HDL, validates their performance using Xilinx Vivado, and conducts power analysis to quantify efficiency improvements.

SPI Protocol

A common serial communication protocol used for data transfers within and between devices, the Serial Peripheral Interface (SPI) is particularly well-suited for low to medium-speed applications. It is frequently employed for communication among a microcontroller and other devices. In terms of communication methods, SPI is commonly referred to be a small protocol, primarily designed for communication between integrated circuits, contrasting with broader communication protocols like system-wide communications and data exchanges.

SPI has a master-slave architecture in which the slave device coordinates data exchange with the master device by use of a clock signal. Data transfer occurs only after clock manipulation; the master controls the serial clock line. Since the SPI protocol does not allow any device to be solely a sender or recipient, the device must be able to send and receive data.

Both the slave and the master SPI modules use an eight-bit serial shift register. Data is kept in the slave's shift register, while the master's shift register holds the data that will be sent. Eight clock pulses are created in the slave's shift register and sent to it from the master's shift register via the MOSI line. On the other hand, by transmitting the contents of its shift register back to the master over the MISO line, the slave effectively exchanges the contents of the two shift registers (Anand and Joseph, 2014).

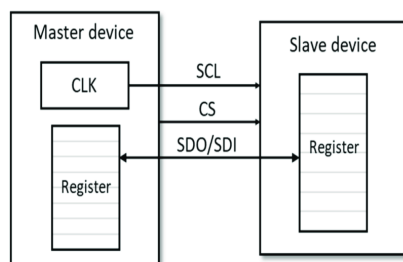


Figure 1. 3-wire SPI protocol Block Diagram

Considering Figure 1, the data lines that permit data transfer and reception between the master device and the slave device are referred to as SDO and SDI, respectively. Given data line sharing, it is desirable to have a way to distinguish between the device to which the data line is transmitting data and

the direction in which it is working (Chen *et al.*, 2020). Various signals are used by SPI to transmit data over its connections:

- Serial Clock (SCL)
- Chip Select (CS)
- Serial Data In (SDI)
- Serial Data Out (SDO)

The essential components of the SPI system are an eight-bit shift register and a data register. A small amount of data is moved out of the SPI during a transfer of SPI, and then serially moved into the master's data register from the slave's data register.

A control register is another component of the SPI system that provides initial control over the SPI operation. To regulate the SPI's data transfer, this register can be used to enable the SPI, set it in master or slave mode, configure clock polarity, set data sampling, and more.

The SPI baud rate register, which provides a range of possibilities for baud rate creation with divisors from 2 to 128 bits, selects the bus clock division value. The baud rate generator has to be in master mode for the SPI to function. The procedure of generating the baud rate involves a series of divider operations.

By adjusting the relevant define directives in the source file, the SPI master core can be customized to satisfy certain system needs and space limitations. The adaptable serial communication protocol known as the Serial Peripheral Interface is well-suited for low to medium-speed data transfer rates, with speeds ranging over 10 MHz, based on the particular implementation. SPI is engineered to accommodate various transfer rates, which users can configure through the SPI master's baud rate settings. Additionally, the SPI module is designed to support multi-slave operation, enabling it to interact with multiple slave devices at a similar time.

The SPI module's control register allows users to configure how the master and slave devices exchange data. It is possible to configure the SPI module to function in either slave or master mode by appropriately programming the control register (Anand and Joseph, 2014). The status register offers insights into the present stage of the data transfer execution, allowing users to identify whether the transfer has been completed.

Verilog HDL is used to implement the SPI module's design, while Xilinx Vivado is used for validation. Logic synthesis is then carried out in Xilinx ISE 13.2. The SPI interface signals from the slave and master are shown in the simulation results. The Virtex-6 FPGA from Xilinx is then used to implement the design.

The analysis of a clock-gated D flip-flop, where a 16-bit register was created, is also covered in the paper. The Virtex-6 FPGA's dynamic power consumption at various frequencies was investigated. Two strategies for lowering dynamic power were highlighted: The clock-gated ALU module based on flip-flops and the clock-gated ALU module with a negative latch. These methods primarily limit clock signal transitions, leading to reduced power usage.

As shown in Figure 2, In a typical SPI connection, slaves are connected to a master via the Slave Select (SS), Master Out Slave In (MOSI), Master In Slave Out (MISO), and serial clock (SCK) lines. Slaves can share the SCK, MOSI, and MISO signals, but each slave has its own SS line.

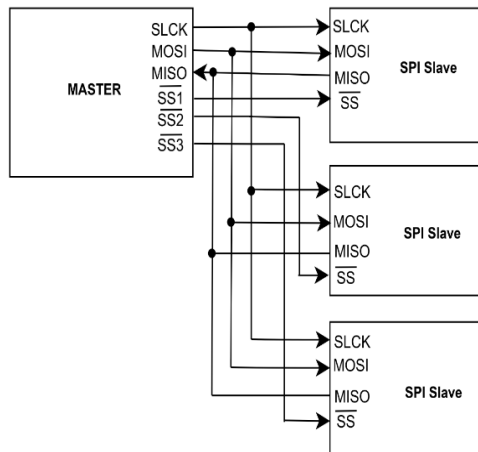


Figure 2. SPI Master Slave Block diagram

Several operations were carried out in the ALU such as AND, OR, NAND, NOR, XNOR, the complements of 1s and 2s, addition, subtraction, multiplication, and division. The clock-gated ALU with a negative latch showed lower dynamic power usage than the one with a flip-flop.

Operations including OR, AND, NOR, NAND, XOR, XNOR, shift right, shift left and complements were all carried out in the ALU. For every clock-gated approach, dynamic power was examined at various frequencies. With the ALU, the clock-gated method based on synthesis showed reduced dynamic power dissipation (Vatte, 2021).

I2C Protocol

Multiple-device interaction is made easier by the I2C protocol, a bidirectional serial communication standard that uses two wires. Its affordability and simple design are the main reasons for its popularity. The I2C bus is regarded as inactive when both the Serial Clock Line (SCL) and

the Serial Data Line (SDA) are at a logic level high. The master device starts a data transfer by carrying out a start sequence that involves the SCL line remains high while the SDA line moves from high to low. The master sends the slave address following the start sequence. The slave that matches the address sends an acknowledgment (ACK) bit and pulls the SDA line low in response.

Data is transmitted in 8-bit segments, beginning with the MSB, data is sent in 8-bits, and the slave transmits an acknowledgment bit. When the ACK bit is low, it indicates that the slave has successfully received the data and is prepared for the subsequent byte (Mehrotra and Charaya, 2015). Conversely, a high ACK bit indicates that the slave can't take any more information, therefore the master sends a stop sequence to terminate the transfer.

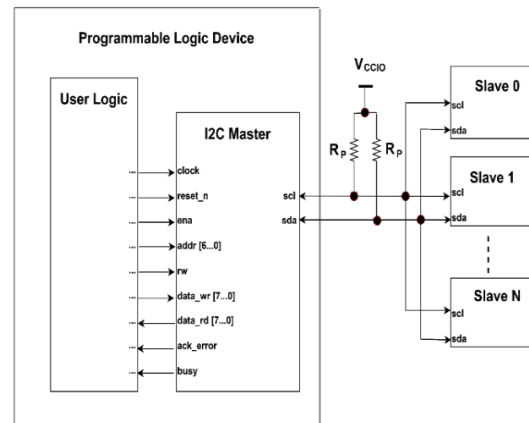


Figure 3. Block Diagram of I2C Protocol

In essence, the I2C protocol serves as a dependable and economical approach for serial communication among multiple standalone integrated circuits (ICs). Its complexity is deliberate, facilitating a broad spectrum of communication modes and enabling multiple devices to share the bus. The I2C single master architecture is seen in Figure 3. It is made up of a bidirectional data line, or serial data line (SDA), and serial clock line (SCL), to which several slaves are linked.

Earlier work

Numerous synchronous serial communication interfaces, including 1-Wire, SPI, and I2C, are utilized for on-board communication. Among all of them, the SPI interface is the one that is most commonly utilized for data exchange. Applications of SPI are found in many different fields, including:

1. Sensor data transfer, such as pressure and temperature sensors.
2. IEEE 802.11, Ethernet, and USB communication.
3. Data transfer between EEPROM and Flash.
4. LCD interfacing.

The four-wire (SPI) Serial Peripheral Interface was first created by Motorola with subsequent developments by teams like Cadence, Digital Core Design, and Texas Instrument (Oruganti and Yellampalli, 2014). These organizations have adhered to the basic SPI specifications while incorporating unique features into their devices. It discusses a unique SPI design that incorporates special features present in current SPI variants such as:

1. The implementation of a power-saving stop state, which is absent from Digital Core Design and Microchip products.
2. The addition of a double-buffered register to stop data overflow, which isn't seen in products made by Cadence, Texas Instruments, Digital Core Design, and Microchip.
3. A broad variety of clock frequencies from $\text{clk}/2$ to $\text{clk}/2048$, a feature not found in Motorola, Microchip, Digital Core Design, and Texas Instruments designs.

Efforts are made to minimize SPI consumption of power through clock gating techniques, including the clock gating technique based on D flip-flops for dynamic power reduction and the leakage control transistor technique for static power reduction in the AND gate following the D flip-flop. These techniques are applied to the PIPO. The main goal of the study is to reduce dynamic power in a 16-bit ALU by using various clock-gating approaches, such as latch-free, latch-based, flip-flop based, and synthesis-based clock-gating techniques (Khanna and Mishra, 2018). The ALU is used to perform

operations including addition, subtraction, increment, decrement, OR, XNOR, XOR, NAND, AND, NOR, complements, shift left, and shift right. The synthesis-based clock-gated approach with the ALU results in less dynamic power dissipation.

We present an implementation of the SPI and I2C protocols in different FPGA devices, helping designers to choose the right architecture for their systems. To do this, they designed the code in Verilog HDL (according to each protocol) for the slave SPI and I2C to the different FPGA devices, considering similarities in their functionality in response times and clock settings, concluding that logic can predict certain behaviors for master devices from the results of the slaves (Oudjida *et al.*, 2009).

Gating Method for Latch-Free Clocks

This method employs basic two-input gates, such as AND or NOR for clock gating as seen in Figure 4. A significant limitation of this approach is that if the enable signal fails to function in conjunction with the clock, it can result in glitches in the gated clock signal.

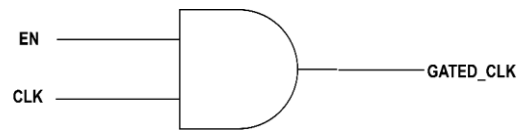


Figure 4. Gating Method for Latch-Free Clocks

The activation signal is recorded during the system clock cycle's high phase in positive latching. Only when the system clock and latch output are both at high levels does the gated clock signal (Srivatsava and Singh, 2015) stay active; otherwise, it deactivates to zero. This setup guarantees that the gated clock signal cannot be triggered while the system clock is in its sleep mode. The overview of various clock gating models are tabulated in Table 1.

Table 1. An overview of different Techniques of clock gating

Clock Gating Techniques	Basic Building Block	Advantages	Disadvantages
Latch free	Basic AND Gate	Simple gated Logic	Glitches
Latch-based clock gating	Level sensitive latch with AND Gate	Glitch free, easy to implement	Not testable
Flip flop based	Flip flop with AND gate	Testability is easily implemented	More power
Synthesis based	Combination of gates with latch	Minimum power consumption among all	Limitations on area

Clock Gating Method on Flip-Flop

Flip-flop designs are frequently preferred over latch-based designs in a wide range of applications. When a flip-flop is divided, it can be seen as two latches, based on the master-slave principle. This technique employs an AND gate in conjunction with a D flip-flop (Chindhu and Shanmugasundaram, 2018).

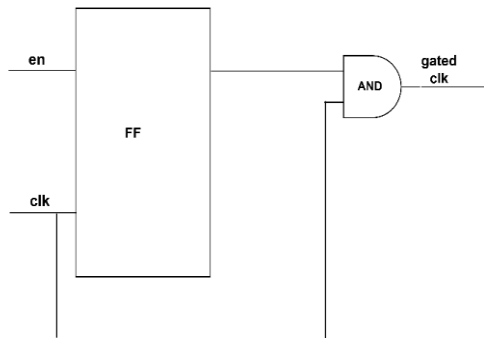


Figure 5. Clock Gating Method on Flip-Flop

As seen in Figure 5, the clock-gated signal only becomes active when the clock and flip-flop output are both high. In contrast, if either the clock or the flip-flop output is weak, the gated clock signal is not active. Therefore, there is no gated clock signal when the system clock is in sleep mode.

Method of Clock Gating Based on Synthesis

This method employs a gated clock, which is created by combining whether beneficial or adverse latch with AND, OR, X-NOR, or X-OR logic gates, as depicted in Figure 6.

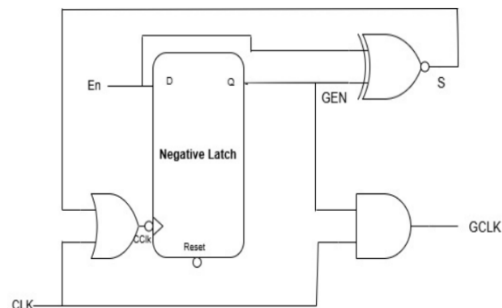


Figure 6. Using a negative latch, synthesis-based clock gating technique

When the enable signal is constant, the X signal may be one, which results in a high controlled

clock when employing the clock gating method based on synthesis using a negative latch. This situation prevents the negative latch from operating, thus the output keeps its previous value. However, when the enable signal changes, the value of X becomes zero, and the controlled clock operates the negative latch circuit, producing various output values. The clock signals AND gate and the negative latch's output signal are combined to create the gated clock signal (Vatte, 2021). We implemented SPI with a latch-based clock gating technique. This approach presents the SPI mode definitions for the simulation waveform, showcasing the manipulative variables of clock phase (CPHA) and clock polarity (CPOL). The SPI interface does not impose a protocol for data exchange overhead limitation, allowing for high-speed data streaming. To provide four separate and distinctive modes that allow for flexibility in communication between the slave and the master, the CPHA is given as one and the CPOL as zero. The most common method for SPI bus slave communication is mode 0.

Methods and Implementation

I2C

The Inter-Integrated Circuit protocol is a two-wire, bidirectional serial communication standard that enables multiple devices to interact with each other. It employs two wires: Data transmission via Serial Data Line (SDA) and synchronization of data transfers via Serial Clock Line (SCL). When the Serial Clock Line and Serial Data Line are both set to a high logic level, the I2C bus goes into a state of inactivity. To begin a data transfer, the master device initiates a start sequence by lowering the SDA line when keeping the Serial Clock Line as high. Following this, the master device transmits the slave address. If a slave matches the address, it acknowledges by lowering the SDA line.

To communicate the start state to each connected slave, the master first switches the Serial Data Line wire from high to low before switching the Serial Clock Line from high to low. The read/write bit and the 7-bit or 10-bit address of the slave it wishes to connect, are sent to each slave by the master. Every slave verifies its address against the master's. The slave will provide an ACK signal if the address matches by pulling the SDA line low for one bit. If the personal address of the slave differs from the master's, the slave leaves the SDA line high. Both the master and the data frame are sent and received. Following the transmission of each data frame, the receiving device provides the sender with an additional ACK bit to verify successful receipt. The slave receives a stop

condition from the master to end data transfer when SCL is set high before SDA is turned high. The I2C bus waveform while in transmission mode is displayed in Figure 7. The start condition merely causes SDA to transition from high to low, after which it serially transmits the address and data. Both lines are initially high, the next sections describe the serial transmission of addresses and data. Only when SCL is high does SDA at the halt condition go from low to high.

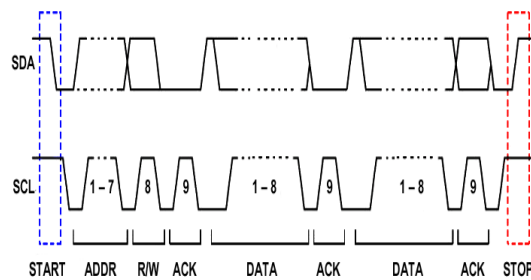


Figure 7. Waveform Of I2C

In the proposed approach, the number of bytes to be sent as well as received is determined by a parameter N. This argument is essential for specifying the size of the counter (byte_counter) and the data array (data[N-1:0]). The data array is utilized for storing the data being transmitted or received, while the byte_counter is used to monitor the current byte being processed.

When the master device plans to transmit the data, it populates the data array with the intended data, and the I2C controller transmits the data sequentially. Conversely, when the master device aims to receive data, it adjusts the byte_counter to reflect the number of bytes expected, and the I2C controller acquires the data sequentially.

The I2C controller employs a state machine to manage the data transfer process. This state machine includes distinct states for various operations, such as initiating a start condition, transmitting an address, transmitting data, and terminating with a stop condition. The state machine transitions among these states based on the ongoing operation and the responses from the slave device. This technique enables the I2C controller to transmit and receive a variable number of bytes, offering flexibility and adaptability for various applications.

SPI:

The Serial Peripheral Interface enables communication between microcontrollers and peripherals through a synchronous, full-duplex

protocol. This technique uses separate data lines and clock, as well as a selection line to identify the communication device. The Slave Select (SS), Serial Clock (SCK), Master In Slave Out (MISO), and Master Out Slave In (MOSI) lines connect a master device to many slave devices in a standard SPI arrangement. While the MOSI, MISO and SCK signals can be used jointly, each slave has its own distinct Slave Select line. The SPI bus waveform while in transmission mode is displayed in Figure 8.

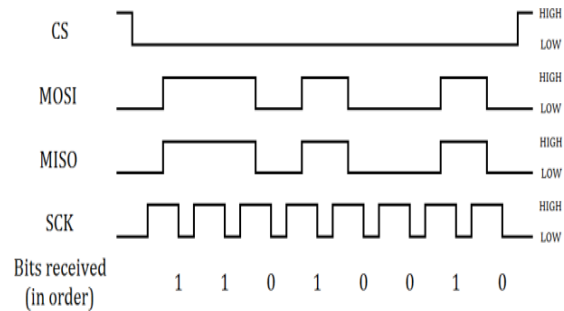


Figure 8. Waveform of SPI

The suggested approach employs a parameter to determine the quantity of bytes intended for transmission and reception. This parameter is instrumental in defining the dimensions of the data array and the counter. The data array serves as the storage for the data being transmitted or received, while the counter is employed to monitor the current byte being processed. The SPI controller sequentially gathers the data after the master device modifies the counter to match the number of bytes expected.

The SPI controller operates a state machine to manage the data transfer process. This state machine includes distinct states for various operations, such as data transmission. The state machine transitions among these states based on the ongoing operation and the responses from the slave device. This technique enables the SPI controller to transmit and receive a variable number of bytes, offering flexibility and adaptability for various applications.

Serial Peripheral Interface (SPI) and Inter-Integrated Circuits (I2C) are the two worldwide standards now in use at the low end of communication protocols. With onboard peripherals, both protocols are well-suited for low to medium-speed data transfer between integrated circuits. SPI and I2C are complementary for this type of communication. The SPI and I2C protocols have clear standards. The comparison between SPI and I2C is tabulated in Table 2 (Ghuse *et al.*, 2016).

Table 2. Comparison of I2C and SPI Protocol

Parameter	I2C	SPI
Full Form	Inter-Integrated Circuit	Serial Peripheral Interface
Communication Type	Synchronous-Clock signal present	Synchronous-Clock signal present
Addressing	Software programmable addressing 7-bit and 10-bit addressing formats	Hardware addressing. There is a dedicated pin for each slave.
Data Lines	Only 2 data lines (SDA and SCL)	Four data lines (MOSI, MISO, SS and SCLK)
Data Transfer	It can communicate in both directions and enables half-duplex.	It allows half-duplex communication and is bi-directional.
Design Cost	I2C is less expensive to implement.	SPI protocol is more expensive to implement.
Number of Devices	It takes up to 127 devices. But, it becomes complex with the addition of devices.	Can take numerous devices but it becomes more complex.
Input/Output Constraints	Pull-up resistors and an open drain	There are no constraints
Power Consumption	The power consumption is low. But, compared to SPI, it consumes more power because it has two pull-up resistors as well.	Lower
Speed	There are four different modes. 1000Hz, 400Hz, 1MHz, 3.4MHz	25MHz higher as compared to I2C
Applications	Frequently utilized for devices with slow speeds.	Frequently utilized for devices with high speeds.
Example Devices	EEPROM, Sensors.	Flash memory.

Table 3. Comparison of SPI and I2C protocols with other low-power design approaches

Feature	SPI	I2C	UART	GPIO-Based Communication	Low-Power Wireless
Type	Synchronous serial communication	Synchronous serial communication	Asynchronous serial communication	Digital communication	Wireless communication
Number of Wires	4 (MISO, MOSI, SCLK, SS)	2 (SDA, SCL)	2 (TX, RX)	Varies (depends on configuration)	No wires (wireless)
Speed	Very high (up to tens of Mbps)	Moderate (up to 3.4 Mbps in fast mode)	Moderate (up to 4 Mbps in some designs)	Depends on the microcontroller's capability	Moderate (up to 2 Mbps for BLE)
Power Consumption	Moderate to high	Low to moderate	Moderate	Low (if properly managed)	Very low (optimized for low-power use)
Complexity	Simple to implement	Higher due to addressing and protocol layers	Simple	Very simple (basic toggling of pins)	High (requires wireless protocol stack)
Device Addressing	No device addressing; dedicated SS lines	Supports multiple devices with addressing	Point-to-point communication only	No addressing; manual control	Unique IDs or pairing mechanisms
Number of Devices Supported	Limited (1 master, multiple slaves with SS)	Supports multiple masters and slaves	Typically point-to-point	Limited by GPIO availability	Many devices can coexist in a range
Power-Saving Features	Non-inherent	Clock stretching for power saving	No inherent features	Controlled by software	Sleep modes, low TX power options
Reliability	High (short distances, low noise)	Moderate (more prone to noise)	High (depends on configuration)	Low (manual timing can cause errors)	Moderate to high (depends on interference)
Distance Supported	Short (<1 meter)	Short (<1 meter)	Short to medium (up to 15 meters)	Short (depends on voltage drop limitations)	Medium (up to 100 meters or more)
Use Cases	High-speed sensors, ADCs, DACs, memory chips	Low-speed sensors, EEPROMs, RTCs	Debugging, GPS modules, simple sensors	Custom low-power toggling, LEDs, etc.	Wearables, IoT, wireless communication

Comparison

A comprehensive review of power, complexity, speed, and use cases is given in Table 3, which also sheds light on how well-suited each is for low-power design situations. It measures design complexity in terms of hardware, algorithms, and integration and emphasizes power consumption efficiency, which is important for battery-operated devices. Additionally, the table evaluates speed, emphasizing performance in time-sensitive applications such as real-time data processing. Technical features are connected to real-world use cases that demonstrate how each solution fits certain requirements, such as wearable technology that prioritizes efficiency or edge computing that requires high speed. By weighing power, complexity, and performance for certain applications, this comparison aids in the identification of the best options (Chen and Huang, 2023).

Result

SPI PROTOCOL

Utilization Factor: The performance parameters like LUT, FF and IOs are analyzed and found that the number of LUTs used is 147 out of 53200, the number of FFs used is 162 out of 106400 and the number of IOs used is 70 out of 200. The utilization percentage of LUT and FF are less than 1% and the utilization of IO is 35% and these utilization factors are shown in Figure 9.

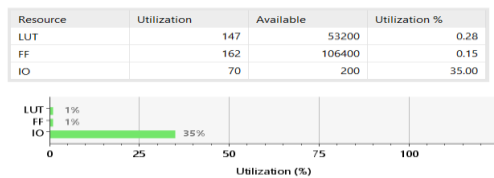


Figure 9. Utilization factors of SPI

Type	Total Endpoints
Max Delay	388
Min Delay	388

Figure 10. Delay of SPI

Delay: The delay factor of the SPI protocol is shown in Figure 10. The maximum delay and minimum delay of the SPI protocol is 388.

RTL Schematic: The RTL schematic of the SPI protocol is shown in Figure 11, which demonstrates how the MOSI (Master Out Slave In) and MISO (Master In Slave Out) units interact.

The SPI full-duplex communication method can be better understood with the help of this block diagram.

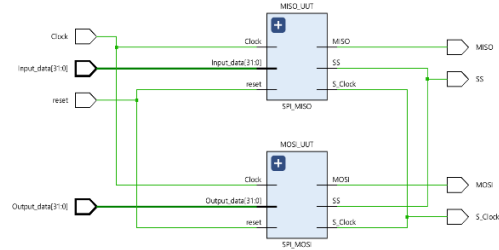


Figure 11. RTL Schematic of SPI

Simulation Waveform: The simulation waveform of the SPI protocol is done by using the Xilinx Vivado 2018.3 and is shown in Figure 12. The parameters that are obtained in the waveforms are clock, reset, transmitter and receiver enable, input data, chip select, MISO, MOSI and SCK.

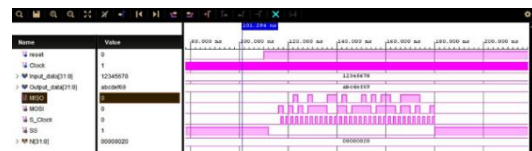


Figure 12. Waveform of SPI Protocol

I2C PROTOCOL

Utilization factor: The parameters like LUT, LUTRAM, FF and IOs are analyzed and found that the number of LUTs used is 150 out of 53200, number of LUTRAM used is 16 out of 17400, the number of FFs used is 120 out of 106400 and the number of IOs used are 69 out of 200. The utilization percentage of LUT, LUTRAM and FF are less than 1% and the utilization percentage of IO is 34% which is shown in Figure 13.

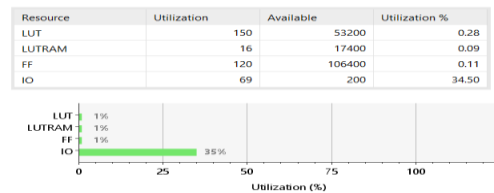


Figure 13. Utilization factors of I2C

Delay: The delay of the I2C protocol is shown in Figure 14. The maximum delay is 460 and the minimum delay is also 460.

RTL Schematic of I2C: The RTL schematic of I2C is shown in Figure 15 and the schematic represents an I2C which consists of modules handling I2C read and write operations and a multiplexer (MUX) for data selection.

Type	Total Endpoints
Max Delay	460
Min Delay	460

Figure 14. Delay of I2C Protocol

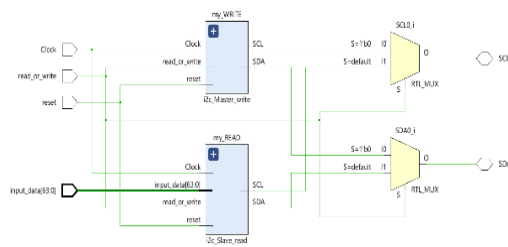


Figure 15. RTL Schematic of I2C Protocol

Simulation Waveform of I2C: The simulation waveform of the I2C protocol is done by using the Xilinx Vivado 2018.3 and is shown in Figure 16 and Figure 17.



Figure 16. Waveform of I2C Slave Read Protocol

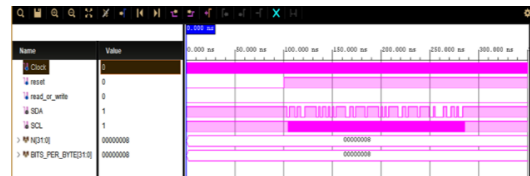


Figure 17. Waveform of I2C Master Write Protocol

Analysis

The analysis of SPI and I2C protocols in this study demonstrates the effectiveness of clock-gating techniques in reducing power consumption while maintaining operational efficiency. These results align with previous studies, such as those by (Khanna and Mishra, 2018), which highlight the advantages of synthesis-based clock gating in minimizing dynamic power dissipation in FPGA designs.

Compared to traditional implementations, our approach further improves power efficiency by optimizing the design complexity and operational speed of SPI and I2C protocols. For instance, our findings regarding the performance of the clock-

gated ALU align closely with the results presented by (Vatte, 2021), who noted significant power savings using negative latches in similar architectures.

Additionally, this study reinforces insights from prior research on the scalability of SPI and I2C protocols, such as (Zidar *et al.*, 2023), who examined the energy consumption of these.

protocols in ultra-low-power systems. The comparisons underscore that SPI remains advantageous for high-speed, low-latency applications, while I2C is better suited for scenarios requiring multiple device communication with lower power demands.

Conclusion

The I2C and SPI protocols are both used for communication between devices, but they have different characteristics and use cases. In the proposed method for I2C, a parameter is used to select the number of bytes to be sent or received. This allows for flexibility in the number of bytes transferred, and the use of a state machine for data transfer control makes the process adaptable to different applications. The proposed method for SPI also uses a parameter to select the number of bytes to be sent or received. This study presents an optimized low-power design for SPI and I2C protocols using synthesis-based clock gating and parameterized communication frameworks. The proposed approach enhances power efficiency, adaptability, and data integrity while maintaining effective data transfer. The experimental results validate that the clock gating techniques significantly reduce dynamic power consumption compared to traditional designs.

References

- Anand, N., and Joseph, G. (2014). Design and implementation of a high-speed Serial Peripheral Interface. Proceedings of the International Conference on Advances in Electrical Engineering (ICAEE), p. 1-3. <https://doi.org/10.1109/ICAEE.2014.6838431>
- Chen, J., and Huang, S. (2023). Analysis and comparison of UART, SPI, and I2C. Proceedings of the IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, p. 272-276. <https://doi.org/10.1109/EEBDA56825.2023.10090677>
- Chen, S.L., Chi, C.C., Lin, Y.T., and Wu, H.C. (2020). A novel low-power synchronous preamble data line chip design for oscillator control interface. Electronics, 9:1509. <https://doi.org/10.3390/electronics9091509>
- Chindhu, T. S., and Shanmugasundaram, N. (2018). Clock gating techniques: An overview. Proceedings of the Conference on Emerging Devices and Smart Systems (ICEDSS), p. 217-221. <https://doi.org/10.1109/ICEDSS.2018.8544281>

- Ghuse, M., Jadhav, M., Badole, C., and Pathan, A. (2016). FPGA implementation of I2C and SPI protocols using VHDL. *IJSART*, 2(1).
- Khanna, N., and Mishra, D.K. (2018). Clock gated 16-bit ALU design & implementation on FPGA. *Proceedings of the 4th International Conference for Convergence in Technology (I2CT)*, Mangalore, India, p. 1-5. <https://doi.org/10.1109/I2CT42659.2018.9057910>
- Mehrotra, S., and Charaya, N. (2015). Design and implementation of I2C single master on FPGA using Verilog. *Progress in Science and Engineering Research Journal*, 3(1):1-5.
- Oruganti, D.N., and Yellampalli, S.S. (2014). Design of power efficient SPI interface. *Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, p. 2602-2606. <https://doi.org/10.1109/ICACCI.2014.6968350>
- Oudjida, A.K., Berrandjia, M.L., Tiar, R., Liacha, A., and Tahraui, K. (2009). FPGA implementation of I2C and SPI protocols: A comparative study. *IEEE Conference*, p. 1-6. <https://doi.org/10.1109/ICECS.2009.5410881>
- Palnitkar, S. (2003). *Verilog HDL: A guide to digital design and synthesis* (Vol. 1).
- Qiang, J., Jiayi, L., Yuming, H., and Xiaoyang, S. (2020). FPGA implementation of SPI bus communication based on state machine method. *Journal of Physics: Conference Series*, 1449(1). <https://doi.org/10.1088/1742-6596/1449/1/012027>
- Srivatsava, G.S.R., and Singh, P. (2015). Dynamic power reduction through clock gating technique for low power memory applications. *Proceedings of the International Conference on Electrical, Computer, and Communication Technologies (ICECCT)*, p. 1-6. <https://doi.org/10.1109/ICECCT.2015.7226204>
- Vatte, M. (2021). Power optimization in configurable ALU using blend of techniques. *Proceedings of the 2nd International Conference for Emerging Technology (INCET)*, p. 1-5. <https://doi.org/10.1109/INCET51464.2021.9456346>
- Zidar, J., Aleksi, I., and Matić, T. (2023). Analysis of energy consumption for SPI and I2C communications in ultra-low power embedded systems. *Proceedings of the 46th MIPRO ICT and Electronics Convention (MIPRO)*, Opatija, Croatia, p. 213-217. <https://doi.org/10.23919/MIPRO57284.2023.10159889>