Perspectives in Communication, Embedded-Systems and Signal-Processing (PiCES) – An International Journal
ISSN: 2566-932X, Vol. 4, Issue 9, December 2020
Part of the Proceedings of the 1st All India Paper writing Competition on Emerging Research - PaCER 2020

Design of I²C Protocol in Verilog-A New Approach

Smitha A
MTech [VLSI & ES], Department of ECE, BNM Institute of Technology, Bangalore, India,
smithaarun19@gmail.com

P A Vijaya
Professor and HoD, Department of ECE, BNM Institute of Technology, Bengaluru, India,
pavmkv@gmail.com

Abstract: Serial communication is a technique where the transfer of data happens serially bitwise. Parallel communication is a technique where the transfer of data happens parallelly simultaneously bitwise. In many applications serial communication is preferred. Comparison between I²C, SPI and CAN serial communication protocol can be seen. I²C communication protocol is preferred over SPI and CAN protocols. I²C is a synchronous serial communication which can be implemented in Verilog.

Keywords: Serial communication, I²C, SPI, CAN, I²C Master and I²C slave.

I. INTRODUCTION

Data-Transmission is the transmission of data over point-to-point or point-to-multi-point communication channel. The communication is very well-known terminology which involves the exchange of information between two or more mediums. The author of article [3] quoted the definition of communication protocol as "The communication in embedded systems means exchanging data in the form of bits between two microcontrollers. This data bits exchange in microcontroller is done through some set of defined rules known as communication protocols". If the data is sent in series i.e. one after the other, then the communication protocol is known as Serial Communication Protocol. If the data is sent in parallel i.e., simultaneously all bits are sent at one time then the communication protocol is known as Parallel Communication protocol.

According to [2], the definition of serial communication in telecommunication is the communication technique in which data is transmitted, one bit at a time, through a communication channel or a device bus. Digital Serial transmissions are bit sent over a single optical path, frequency or wire sequentially. Parallel communication is a communication technique wherein the data transfer occurs by transmitting data, all bits at a single time simultaneously in a sequential order over a communication channels or computer buses. It requires more buses since all bits are transmitted at a single time.

Parallel communication transfers multiple bits at the same time. They usually require separate buses for data - transmission. Serial communication transfers their data, one single bit at a time This technique of communication will work on as little as one wire, usually no more than four. A serial connection requires fewer interconnecting cables (e.g. wires / fibers) and thus requires less space to occupy. Whereas in Parallel communication requires more buses for transmission of data simultaneously. Hence it consumes more space.

Examples of Parallel Communication Protocols are IEEE-488, ATA, SCSI, ISA, and PCI. Similarly, the examples of Serial Communication Protocols such as I²C, SPI, CAN, RS232, USB, SATA, ETHERNET and 1-Wire etc.,

Serial communication is used in many approaches, since implementation is cheaper. Many ICs have serial interfaces, rather than parallel interfaces, so that they have fewer pins and are therefore less expensive. The more bits are transmitted in parallel, the more opportunity there is for things to go wrong. The wider the bus, the less room there is for error. With serial transmission, the data rate can be increased. In seral communication, no clock required in case of asynchronous type, requires less space since it transmits one bit at one time, no cross talks and low cost due to these reasons the serial communication technique is used in many approaches.

In serial communication, there are two forms i.e. Asynchronous and Synchronous. Asynchronous means No clock is used by the system i.e. No common clock has been used by the endpoints. Synchronous means the system uses a common clock for the communication. Serial communication protocols are CAN, SPI, I²C, USB, eSPI Protocols. And the popular RS standards based on Serial communication are RS232, RS422, RS485.

Factors	**Serial Communication Protocol Comparison**		
	I²C	*CAN*	*SPI*
Full form	Inter-Integrated Circuit	Controller Area Network	Serial Peripheral Interface
Invented by	Philips semi-conductor	Robert Bosch	Motorola

	(NXP semi-conductor)		
Invented in year	1982	1983	Mid 1980s
Signals/ wire	2 wire	2 wire	4 wire
Signal names	1.SDA 2.SCL	1. CAN H+ 2. CAN H-	1. MOSI 2. MISO 3. SCLK 4. SS
Clock	synchronous	synchronous	synchronous
Mode	Half duplex	Full duplex	Full duplex
Type	Multi-master protocol	Multi-master protocol	Single master protocol
Applications	Accessing low speed DACs and ADCs	Agricultural equipment, elevators, Escalators	Sensors, control devices, camera lenses
Usage	Within the circuit board	Within the two circuit boards	Within the circuit board

Table 1. Comparison table of I2C, SPI and CAN Protocols

I^2C (inter integrated circuit) is a synchronous serial communication protocol. It uses two lines i.e., SDA- serial data and SCL- serial clock. CAN (Controller Area Network) is a message-based protocol. It is Robust bus which allows the communication between devices with each other's application without the host computer. SPI (serial peripheral interface) is also a synchronous serial communication protocol having 4 wire i.e., MOSI-Master out slave in, MISO- master in slave out, SCLK- serial clock, SS-slave select.

I^2C is better than SPI and CAN. I^2C is used in many applications. Why?

I^2C is a Synchronous Serial Communication protocol like SPI and CAN. It is Half-duplex protocol i.e. only one end can transfer the data at a time. When one device sends the data, the other device cannot send the data back it can only receive the data. Transferring of data happens from one direction at a time. When master sends the data to slave, slave can only receive the data, it cannot send the data back to master at that time. After receiving the data when master has stopped sending only then slave can send the data to master. Both blocks can communicate but not simultaneously at a time. Half duplex nature can be very useful because it will have same frequency but transmits and receives at different times. This method is commonly used in cost-effective wireless bridging. So, half duplex nature can be very useful. I^2C allows many masters to involve in the communication.

I^2C is a multi-master, multi-slave serial communication. I^2C is used in many applications with its advantages such allowing multi masters which is not there in SPI which is a drawback in SPI. Half-duplex systems conserves bandwidth as only one communication channel is required, which is shared between the two directions alternately at different timing, which is not there in CAN protocol since it is a Full-duplex communication protocol. In full duplex, there are chances of data collision or data loss, since more than one channel is used for communication, both the devices can send and receive the data at the same time. So, data loss can occur in full-duplex communication systems Half-duplex nature is present in I^2C protocol, so it is better than SPI and CAN communication protocols.

In other terms, I^2C is the combination of SPI and CAN. Since I^2C protocol has both the qualities of CAN i.e., allowing Multi-masters and SPI i.e., Usage within the circuit board. So, in many applications I^2C protocols is used due it is simplicity in architecture, half duplex nature, allowing multi-masters and variable baud rates. So I^2C is used in many applications like DVD's, ADCs, DACs, microcontrollers, LCDs, etc.

II. EXISTING METHODOLOGY

I^2C termed as Inter-Integrated Circuit is a synchronous serial communication protocol. It has two signals/logics i.e. SDA (Serial Data) and SCL (Serial clock). It performs two operations i.e. Write operation and Read operation. SDA (Serial data) is a signal through which the master sends the data to slave. I^2C is a multi-master, multi-slave protocol. In I^2C protocol one end is the Master which starts the communication and other end is Slave. Here master has the ability to start the communication and to terminate the communication. Master also has the ability to specify to which slave it wants to communicate with, and master also tells whether it wants to write the data to slave or to read the data from the slave. Here in I^2C, slave only has the ability to send the acknowledgment to master. SCL (serial clock) is always controlled by the master.

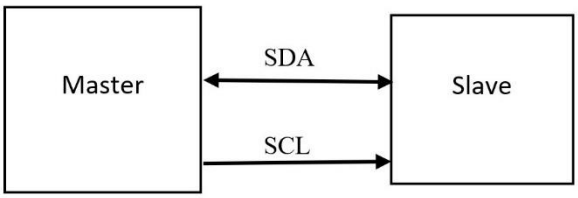

Fig 1. I^2C Communication between Master and Slave

I^2C protocol is standard protocol, since it is a communication protocol not much changes can be done. In I^2C, data is transferred in messages. Each message has a Start condition, slave address, R/W bit then acknowledgment bit, Data (8 bit or 10 bit), ACK bit, Stop condition.

In I^2C, there are modes at which the communication can occur for an 8-bit serial protocol. Those are:

START	Slave Address	R/W	ACK	DATA (8 bits)	ACK	STOP

Fig 2. I^2C protocol Interface Message format

START	Slave Address	R/W	ACK	DATA (8 bits)	ACK	DATA (8 bits)	ACK	DATA (8 bits)	ACK	STOP

Fig 3. I^2C protocol interface message format 16 bits

1. Normal Mode/ Standard Mode - 7-bit addresses and 100KHz communication.
2. Fast Mode - 10-bit addresses and 400kHz communication
3. High Speed Mode/ Fast mode plus - 3.4Mhz Communication
4. Ultra-Fast Mode – 5MHz Communication

I^2C has two signals which is active high, to start the communication master has to pull SDA line from High to low when SCL is still high such a condition is called as START condition. Later Master has to specify to which slave it wants to communicate with. It is done by sending the Slave address then master has to tell whether it wants to write or read the data i.e. Write=0 and Read=1. Each slave in the communication will have a unique slave address, after receiving slave will verify by sending the acknowledgment to the master from slave. After getting the ACK bit, Master will send the 8 bit Data to master if write operation(R/W=0) else master will receive the 8 bit Data from slave if Read operation(R/W=1) then ACK bit and STOP condition by pulling the SDA line from low to high when SCL is high.

Message format Sequence is as follows:

1. START condition
2. Slave Address
3. Read/Write bit, W=0 & R=1
4. Acknowledgement bit
5. DATA (8 bit) – if write, then Master sends to slave. If read, then Master Receives the data from slave.
6. ACK bit
7. STOP condition

Since I^2C is an 8-bit communication protocol. So, at a time only 8 bits can be sent. It can be increased to many bytes each followed by an ACK bit.

I^2C protocol is an 8-bit communication protocol. As many numbers of data bits can be sent but in the form of 8-bit data followed by n ACK bit. After sending the Start condition, slave address, R/W bit to the slave, master can receive or send as many data as it wants but in the format of 8-bit data each followed by an ACK bit.

III. PROPOSED METHODOLOGY

I^2C is a synchronous serial communication protocol which was invented by Philips semiconductor or NXP semiconductor in 1982. It has two signals i.e., SDA (serial Data) and SCL (serial clock). I^2C protocol is used for shorter distance communication, inter-board communication, for communicating between lower speed Peripheral ICs to Processors and Microcontrollers, etc.I^2C protocol is a standard protocol having SDA and SCL.

I^2C protocol has two lines/signals i.e., SDA and SCL. SDA (serial Data) is a bidirectional signal which sends and receives the data between I^2C master and I^2C Slave as shown. SCL (serial clock) is controlled by the I^2C master. It is the line that carries the clock signal from master to slave.

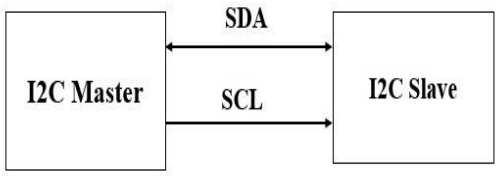

Fig 4. I^2C protocol

I^2C Master will send the start condition, slave address, R/W bit then depending on write or read, it sends or receives the data. Then slave will send the ACK bit after receiving the slave address and after receiving the data through SDA line. So, SDA line has to be controlled by both I^2C Master and I^2C Slave as shown in Fig. 4, which becomes a problem while implementing I^2C in Verilog. Instead of using same SDA line by both, that SDA line can be split into two i.e., SDA_out & SDA_in for I^2C master block and I^2C Slave block respectively as shown in Fig. 5

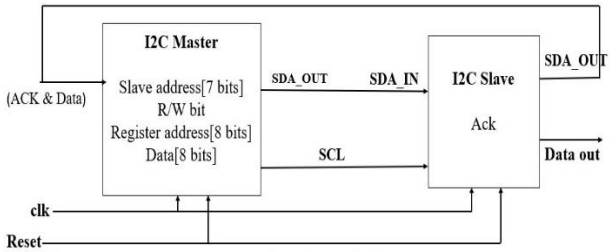

Fig 5. Proposed Block Diagram for I^2C protocol

In the proposed methodology SDA is spilt into two as shown. I²C Master will have SDA_out and SDA_in and I²C_Slave will have SDA_out and SDA_in as shown in Fig. 5. I²C Master will send the START condition, slave address, R/W bit through SDA_out signal to I²C slave. Here, SDA_out of I²C master will become SDA_in of I²C Slave. if write condition then I²C master will send the 8-bit data through SDA_out to I²C slave. I²C slave will send the ACK bit through SDA_out of slave to master which is SDA_in of master. Here, SDA_out of I²C slave will become SDA_in of master. If Read operation, then SDA_out of slave will send the 8-bit Data to I²C master.

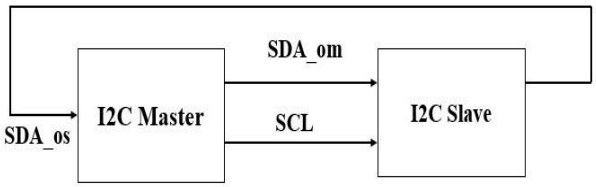

Fig 6. Proposed I²C Communication protocol

I²C master will have SDA_out which is SDA_in for I²C Slave. So, that line can be SDA_om when combined (SDA_out for master and SDA_in for slave). Similarly, I²C slave will have SDA_out which is SDA_in for I²C Master. So, that line can be SDA_os when combined (SDA_out for slave and SDA_in for the master).

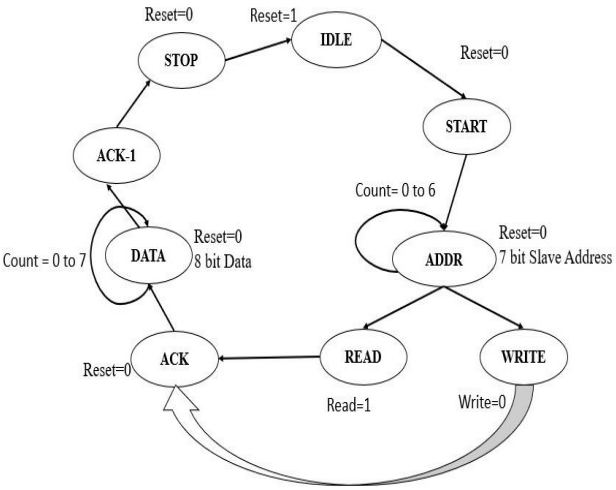

Fig 7. Finite State Machine for the proposed methodology of I²C protocol

There are the following states

1. State 0 = IDLE (reset =1, initiate =1)

   Else when reset =0 and initiate =0 then the following states appear.

2. State 1 = START CONDITION - SDA is pulled from high to low when SCL is still high
3. State 2= SLAVE ADDRESS – Normal mode so, 7-bit slave address is sent.
4. State 3= READ/WRITE BIT- Write is 0 and Read is 1.
5. State 4 = ACK/NACK BIT- SDA line is low if acknowledged and SDA line is high if NACK (not acknowledged)
6. State 5 = DATA- 8-bit data is sent or received at once
7. State 6 = ACK_1 BIT – After receiving the data ACK bit sent i.e. SDA line is pulled low. NACK bit means SDA line is high
8. State 7 = STOP CONDITION- Master will terminate the communication by pulling the SDA line from low to high when SCL line is still high.

FSM (finite state machine) gives the exact flow of the Protocol. Using this FSM, I²C protocol can be implemented in Verilog where SDA line which is a bidirectional one can be taken as SDA_out and SDA_in for master as well as slave later they can be combined as SDA_om and SDA_os respectively.

IV. SIMULATION RESULTS

The Simulation results for I²C Master writing to I²C slave is as shown in Fig. 8 and I²C Master reading to I²C slave is as shown in Fig. 9. Here, Sdaout_m will be the output of I²C Master and Sdain_m will be the input to the I²C Master. If master writes then the data will be stored in dataop_s (data output of slave). If master reads then the data will be stored in dataop_m (data output of master). The output dataop_s and dataop_m is observed here. For write operation, observe dataop_s and for read operation observe dataop_m. If the slave address sent by I²C master does not match the one present in the I²C Slave, then it will send a NACK bit i.e. sdain_m which becomes high as shown in Fig. 10 The simulation results are as shown.

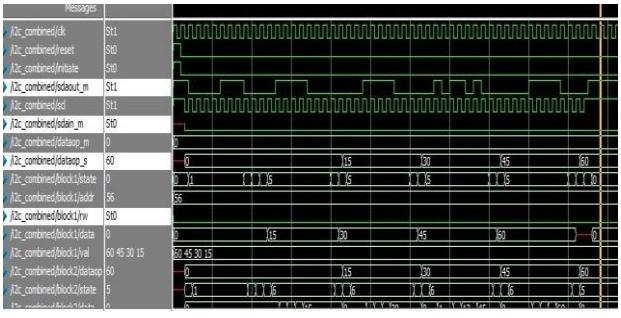

Fig 8. Simulation Result for Write operation 64 bits

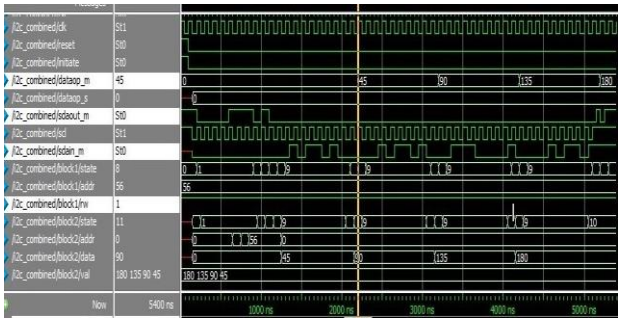

Fig 9. Simulation Results for Read operation 64 bits

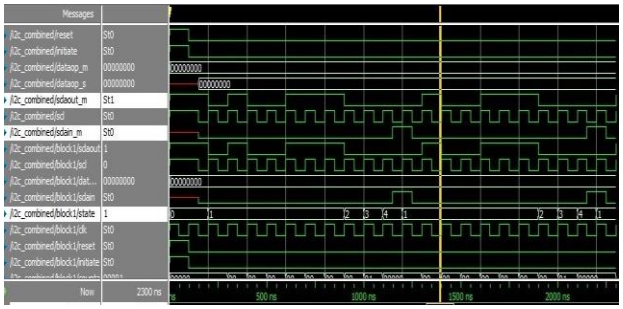

Fig 10. Simulation Results for Wrong Slave Address

V. CONCLUSION

Serial communication is used in many applications due to its low cost, no Cross talks and less space when compared to parallel communication. In serial communication there are Asynchronous and Synchronous modes. Asynchronous means no clock is used while transmitting the bits serially. Synchronous means system uses its clock while transmitting the bits over a communication channel. Some of the Synchronous Serial communication protocols are I^2C, SPI and CAN. Comparison between I^2C, SPI and CAN protocols have been shown. I^2C is used in many applications due to its Half duplex nature, multi-master nature and simplicity in architecture when compared to CAN and SPI Protocols. I^2C protocol can be implemented in Verilog using the finite state machine.

REFERENCES

[1] Abhinav Boddupalli, "Design and Implementation of I2C bus protocol on FPGA using verilog for EEPROM", Proceedings of IEEEFORUM International Conference, 01st October, 2017, Pune, India

[2] Serial Communication, Techopedia, [Online]. Available: https://www.techopedia.com/definition/22010/serial-communication. Accessed on 25st June 2020.

[3] Abhimanyu Pandit, Serial Communication Protocols, Circuit Digest,[Online]. Available: https://circuitdigest.com/tutorial/serial-communication-protocols. Accessed on 25st June 2020.

[4] Jason Fitzpatrick, Why Is Serial Data Transmission Faster Than Parallel Data Transmission?, How to Geek, [Online]. Available: https://www.howtogeek.com/171947/why-is-serial-data-transmission-faster-than-parallel-data-transmission/. Accessed on 25th June 2020

[5] Shaik.fazil ahmed y.murali, "Implementation of I2C multi task and multi slave bus controller using verilog", International journal of engineering and computer science ISSN: 2319-7242 volume 4 issue 8 Aug 2015.

[6] Half Duplex Vs Full Duplex - What is it, and why does it matter?, DigitaLair, [Online]. Available: https://www.digitalairwireless.com/articles/blog/half-duplex-vs-full-duplex-what-it-and-why-does-it-matter. Accessed on 25th June 2020.

[7] Half-Duplex, Network Encyclopedia, [Online]. Available: https://networkencyclopedia.com/half-duplex/. Accessed on 25th June 2020.

[8] Serial Communication, Sparkfun, [Online]. Available: https://learn.sparkfun.com/tutorials/serial-communication. Accessed on 25th June 2020.

[9] Serial Peripheral Interface, Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface. Accessed on 25th June 2020.

[10] CAN Bus, Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/ CAN_bus. Accessed on 25th June 2020.

[11] I^2C, Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/I%C2%B2C. Accessed on 25th June 2020.

[12] Pankaj kumar mehto, Ashish radhuvansi , Sonu lal "Design and modeling of I2C bus controller using VHDL", International journal of innovative research in computer and communication engineering. ISSN(Online): 2320-9801; ISSN (Print): 2320-9798; Volume. 3, issue 2, february 2015.

[13] DVIJEN TRIVEDI, Aniruddha Khade, Kashish Jain, Kashish Jain, Ruchira Jadhav, "SPI to I2C Protocol Conversion using Verilog", 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA).

[14] Bharath.K.B, K. V. Kumaraswamy, Roopa K Swamy, "Design of Arbitrated I2C Protocol with DO-254 compliance", 2016 International Conference on Emerging Technological Trends [ICETT].

[15] Zheng-wei HU, "I2C Protocol Design for Reusability", Third International Symposium on Information Processing, April-2010 IEEE.