Design and Implementation of I²C Single Master on FPGA Using Verilog

[1] Dr. G. Bhoopal Rao, G. Soumya[2], M. Nagul Meer[3], K. Sai Kumar[4]
[1]*Associate Professor, ECE Dept, Nalla Malla Reddy Engineering College,Hyderabad, India,500088*
[2,3,4] *Student of ECE at Nalla Malla Reddy Engineering College, Hyderabad, India,500088*

Abstract: Design of the I²C single master, consists of a bidirectional data line, serial data line (SDA), and serial clock line. The protocol presented may accommodate several masters. The I²C serial bus, which has two wires and is bidirectional, enables rapid device-to-device communication without compromising data integrity. It is a quick and efficient method of sending data between devices. Other bus protocols require extra pins and signals to link devices, whereas it can operate a network of device chips using just two general-purpose I/O pins. It only requires two lines for communication with two or more chips. Model SIM is used to duplicate the entire Verilog-written module in a smaller number of I²C devices. The strength of the I²C protocol is in its innate ability to employ chip addressing, which makes it simple to add new components to the bus.

Keywords: I2C, Serial Communication, SDA, SCL, UART

INTRODUCTION

The act of transmitting data through a point-to-point or point-to-multi-point communication channel is known as data-transmission. The term "communication" is often used to describe the transfer of information across two or more media. The definition of communication protocol, according to the article's author, is "The exchange of data between two microcontrollers in the form of bits in embedded systems. The exchange of data bits in microcontrollers is carried out in accordance with a set of established guidelines called communication protocols. Serial Communication Protocol is the name of the communication protocol used when data is transmitted in series, or one after the other. When data is transferred simultaneously and in parallel, or when all bits are sent at once, the communication protocol is referred to as a parallel communication protocol. Serial communication is described as a communication strategy in telecommunications where data is sent one bit at a time across a communication channel or device bus. Bits are transmitted serially via a single optical channel, frequency, or cable in digital communications. When using parallel communication, data is sent by sending each piece of information across a communication channel or computer bus one at a time, concurrently, in a sequential manner. Since every bit is sent at once, more buses are needed.

Multiple bits are transferred simultaneously during parallel transmission. Typically, they require separate buses for the transfer of data. One bit at a time is sent during serial transmission. This kind of communication can be used with as little as one wire and often no more than four. A serial connection takes up less space since it requires fewer interconnecting connections (such as wires or fibres). In contrast, parallel communication needs multiple buses to allow for simultaneous data transfer. It takes up more space as a result. The IEEE-488, ATA, SCSI, ISA, and PCI parallel communication protocols are a few examples. In a similar vein, the Serial communication is utilised in various techniques because it is more cost-effective to implement than other communication protocols as I²C, SPI, CAN, RS232, USB, SATA, ETHERNET, and 1-Wire. Because they have fewer pins and are thus less costly, many ICs use serial interfaces rather than parallel interfaces. The possibility of something going wrong increases as more bits are transferred in parallel. There is less space for error the broader the bus. Data rate may be enhanced via serial transmission. The serial communication technique is utilized in various ways because it uses less space than asynchronous communication, transmits one bit at a time, eliminates crosstalk, and is inexpensive. There are two types of communication: asynchronous and synchronous. Asynchronous denotes that the system does not employ a clock, i.e., the endpoints have not

shared a common clock. Synchronous refers to a system that bases its communication on a single clock. CAN, SPI, I²C, USB, and ESPI Protocols are examples of serial communication protocols. And the widely used Serial communication-based RS standards include RS232, RS422, and RS485.

LITERATURE SURVEY

A serial communication protocol called I²C, or Inter-Integrated Circuit, was created by Philips Semiconductors, now known as NXP Semiconductors. Serial communication has been using this paradigm more and more. I²C is a two-wire, bidirectional bus created to boost circuit simplicity and hardware efficiency. This protocol supports many masters and slaves (which is a restriction with SPI communication) and permits data loss-free serial data bus (SDA) connection between faster and slower devices. The other line is known as the Serial Clock Line (SCL), which restricts UART connection by transferring data in accordance with a synchronized clock. In order to communicate, other communication protocols like USB, RS-422, RS-485, CAN, etc. need extra pin connections and signals. The premises I²C 's importance as a communication protocol is demonstrated by the use of I²C over UART, SPI, USB, and other protocols. Verilog is used in this study to design and implement the I2C bus on the FPGA, which serves as the master, and to interface with the EEPROM (24C02), which serves as the slave [1].

The I²C bus is still one of the most often utilized on-board data buses for nanosatellite missions despite the dependability issues that are connected to it. In the context of nanosatellite missions, this study presents a thorough fault analysis for the I²C bus, and as a result, it looks at and suggests viable mitigation measures. Most CubeSat missions must consider the danger of the I²C bus failure since associated bus failures can result in certain catastrophic failures. The features of the I²C bus, the software and hardware requirements, and the major causes of I²C bus failure are therefore examined in this paper. I²C bus specifications, characteristics, and failures are compiled by doing experimental testing using the required hardware and software. Possible mitigation strategies are suggested based on the experimental testing, and then a qualitative risk analysis is provided to assess the influence of the methods on the overall mission success. The study demonstrates the significant impact of the I²C bus on the health of the CubeSat and the success of the mission, highlighting the significance of design considerations to lower the risk level of missions as well as accounting for runtime faults that may occur during mission operation.[2]

Inter Integrated Circuit is referred to as I²C. Philips Semiconductor developed the serial bus protocol. I²C bus is well-liked since it's easy to use. Low speed devices are communicated with by CPUs using the I²C protocol. It enables quicker devices to communicate with slower ones without sacrificing any data. The I²C bus controller connects the slave and master components. MEMS motion sensor (1TG MPU 6050) serves as the slave to FPGA, which serves as the master. Data from MEMS motion sensors is sent over the I²C protocol to the FPGA. The signals from the accelerometer are digitalized using 16-bit ADCs in MEMS motion sensors. Because it only needs two wires and fewer pin connections than other buses, the I²C bus on FPGA offers more simplicity. Due of this, this paper we are going to design an I²C bus protocol using Verilog code which interfaces FPGA board with MEMS motion sensor. Noisy data from the MEMS motion sensor is denoised by using Haar wavelet coefficients.[3]

This paper presents, hardware implementation of I²C and UART controller on FPGA for interfacing ambient light sensor BH1750FVI and transmitting data serially to Simulink respectively. The objective is to establish and synchronize communication between light sensor and Simulink through I²C-to-UART Bridge implemented on FPGA. VHDL hardware description language is used to implement I²C and UART controller through finite state machine (FSM) structure. Functional verification of I²C controller implemented on FPGA is done through hardware I²C analyzer. After successful acquisition of data from light sensor, UART controller is used to send data serially via COM port into Matlab-Simulink for monitoring and processing of data from ambient light sensor.[4]

In this research, a Protocol Conversion Unit (PCU) enabling smooth communication between the two commonly used serial communication protocols SPI and I²C is designed and simulated. The architecture provided in this article enables the transmission of data from an SPI-enabled sender device to an I²C -enabled receiver device, which would not otherwise be

feasible. Unlike I²C, which only offers half-duplex communication, SPI supports full duplex. SPI is also quicker than I²C. I²C, in contrast, only uses two wires since, unlike SPI, it does not include a separate Slave Select line. I²C instead uses the Address and Acknowledgement protocol to speak with slaves. In places where the controlling device must communicate, as a result with a lot of peripheral devices, it is essential for the controller to send commands and data to the concerned peripheral device quickly using high speed of SPI and at the same time save on dedicated pins for each peripheral device using a rather simple Two Wire Interface of I²C, a design capable of providing conversion between SPI and I²C formats becomes essential. In this paper support for just one peripheral device is given. The design in this paper can be upgraded to support large no of peripherals by providing a First In First Out (FIFO) Queue for storing commands and data along with corresponding addresses of peripheral devices in the Protocol Conversion Unit (PCU).[5]

BLOCK DIAGRAM OF THE PROJECT

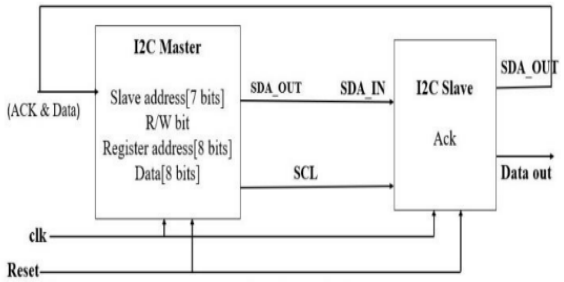


Fig:1 Block diagram of I²C

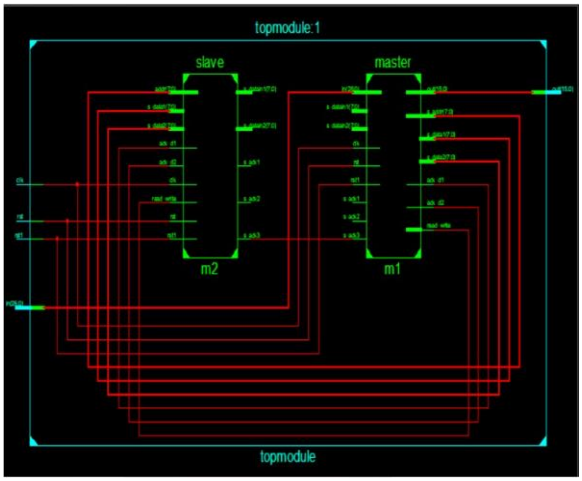


Fig:2 Schematic of I²C in Xlinix ISE

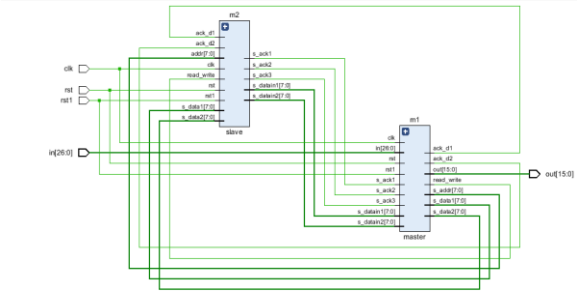


Fig:3 Schematic of I²C in Xlinix vivado.

OPERATION OF THE PROJECT:

- In this I²C controller, data connection between master and slave. Considering address bits can be either 7 or 10 bits, we chose to utilize 7 bits for the address.
- In order for the draw-down clock to activate after we begin the operation, we must pull down the SDA line. When the clock reaches high, operation will start.

Step 1 - start
Step 2 - enter the slave address ID 7bits
Step 3 - the ACK bit will come from slave
Step 4 - have to give write or read operation
Step 5 - enter the data 1 8bits
Step 6 - ACK bit will come from slave
Step 7 - enter the data 2 8bits
Step 8 - ACK bit will come from slave
Step 9 - Stop

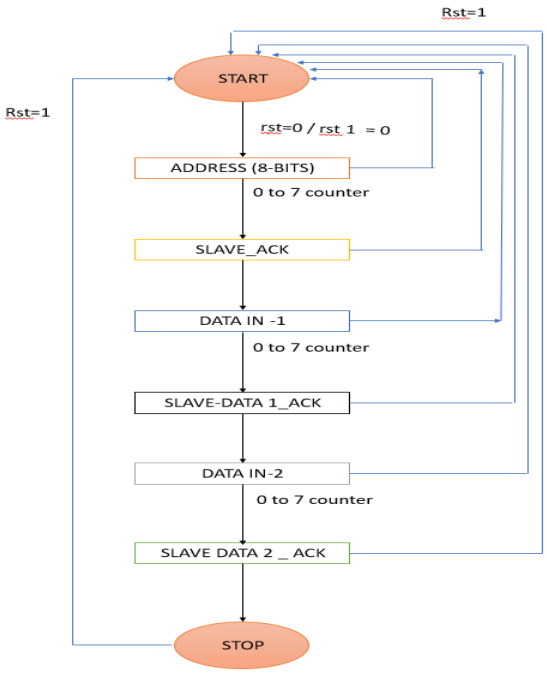


Fig:4 Flow chart for designing I²C protocol

- The clock will be produced by SCL during the entire process. The input is 27 bits, the output is s_data1, s_data2, and the slave responds with 3 ACK bits when data is sent from the master to the slave. 27 bits, clk, rst, and rst 1 make up the inputs.
- When the master gives us data, the slave will reply with an ACK in s_ack1, s_ack2, and s_ack3 whenever we get data from a slave.
- The original source code The slave's address ID will be checked at a stage when the slave gives the ACK after matching the same address ID.
- Data 1 will be written if the operation is a write one; it will be received if the operation is a read one.
- Data 2 is written if it's a write operation, and it's received if it's a read action.
- Data from the slave will show up in s_datain1 and s_datain2 if we get it. The master will learn information about the slave from the outside world and use it in the slave code. If data has to be transferred, it must be placed with values of 1 and 0, or it can be offered as a number between 0-255.
- The difference between the master and slave is that when we provide data to the slave, the slave will respond with an ACK; however, when we request data from the slave, we transmit data in code as 7 bits of 1 and 0 so that we will receive the same output.

RESULTS AND DISCUSSION

Fig: 5 Simulation Results

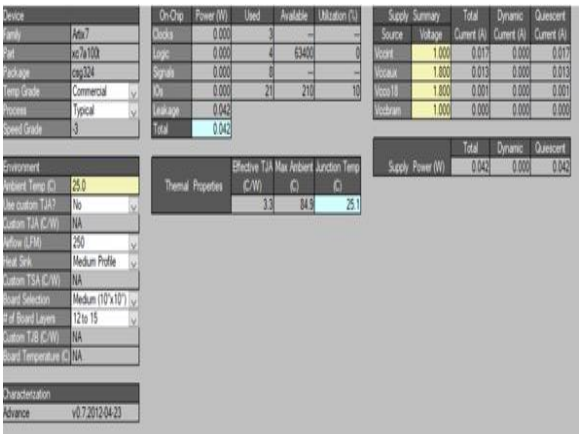
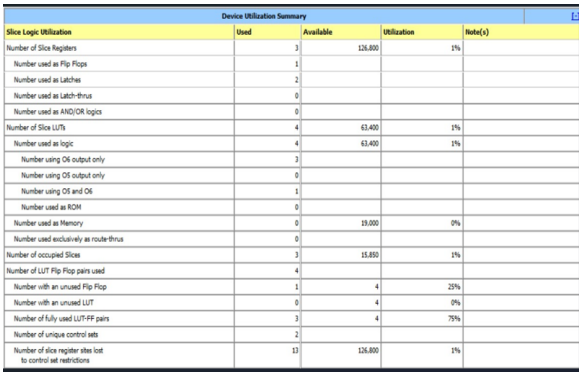


Fig:6 Power analysis



Device Utilization Summary				
Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	3	126,800	1%	
Number used as Flip Flops	1			
Number used as Latches	2			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	4	63,400	1%	
Number used as logic	4	63,400	1%	
Number using O6 output only	3			
Number using O5 output only	0			
Number using O5 and O6	1			
Number used as ROM	0			
Number used as Memory	0	19,000	0%	
Number used exclusively as route-thrus	0			
Number of occupied Slices	3	15,850	1%	
Number of LUT Flip Flop pairs used	4			
Number with an unused Flip Flop	1	4	25%	
Number with an unused LUT	0	4	0%	
Number of fully used LUT-FF pairs	3	4	75%	
Number of unique control sets	2			
Number of slice register sites lost to control set restrictions	13	126,800	1%	

Fig:7 Area (no of LUT's)

CONCLUSION

The primary focus of this study (SCL) is on the design of the I^2C single master, which consists of a bi-directional data line, serial data line (SDA), and serial clock line. This protocol may accommodate several masters. The I^2C serial bus, which has two wires and is bidirectional, enables rapid device-to-device communication without compromising data integrity. It is a quick and efficient method of sending data between devices. Other bus protocols require additional pins and signals to connect devices, whereas it can control a network of device chips using just two general-purpose I/O pins. For communication with two or more chips, it simply needs two lines. The complete Verilog-written module is replicated using Model SIM. a lesser quantity in I^2C. Due to its half-duplex nature, multi-master nature, and simplicity in architecture as compared to CAN and SPI Protocols, I^2C is employed in numerous applications. The finite state machine in Verilog can be used to create the I^2C protocol.

REFERENCE

[1] Abhinav Boddupalli, "Design and Implementation of I^2C bus protocol on FPGA using verilog for EEPROM", Proceedings of IEEEFORUM International Conference, 01st October, 2017, Pune, India

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

[3] Shaik.fazil ahmed y.murali, "Implementation of I^2C 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.

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

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

[6] Pankaj kumar mehto, Ashish radhuvansi , Sonu lal "Design and modeling of I^2C 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.

[7] DVIJEN TRIVEDI, Aniruddha Khade, Kashish Jain, Kashish Jain, Ruchira Jadhav, "SPI to I^2C Protocol Conversion using Verilog", 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA).

[8] Bharath.K.B, K. V. Kumaraswamy, Roopa K Swamy, "Design of Arbitrated I^2C Protocol with DO-254 compliance", 2016 International Conference on Emerging Technological Trends [ICETT].