KLE Technological University

Creating Value
Leveraging Knowledge

School of
Electronics and Communication Engineering

Mini Project Report

on

DESIGN AND IMPLEMENTATION OF AN INTERFACING PROTOCOL BETWEEN APB AND I2C FOR AN SOC

By:

1. **NIDHI DARAK** 01FE20BEC004
2. **AVINASH V V** 01FE20BEC018
3. **PAVAN MYAGERIMATH** 01FE20BEC199
4. **HARSHA JANGAMANI** 01FE20BEC202

Semester: V, 2022-2023

Under the Guidance of

Prof. Shraddha Hiremath

K.L.E SOCIETY'S
KLE Technological University,
HUBBALLI-580031
2022-2023

SCHOOL OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that project entitled **" Design and implementation of an interfacing protocol between APB and I2C for an SOC "** is a bonafide work carried out by the student team of **" Nidhi Darak(01fe20bec004), Avinash V V (01fe20bec018), Pavan Myagerimath(01fe20bec199), Harsha Jangamani(01fe20bec202)"**. The project report has been approved as it satisfies the requirements with respect to the mini project work prescribed by the university curriculum for BE (V Semester) in School of Electronics and Communication Engineering of KLE Technological University for the academic year 2022-2023.

Prof. Shraddha Hiremath	**Dr. Nalini C Iyer**	**Dr. Basavaraj S Anami**
Guide	**Head of School**	**Registrar**

External Viva:

Name of Examiners **Signature with date**

1.

2.

ACKNOWLEDGMENT

The sense of fulfillment that comes with having completed the "Design and implementation of an interfacing protocol between APB and I2C for an SOC" would be incomplete if we didn't mention the names of the people who helped us complete it because of their clear guidance, support, and motivation.

We are thankful to our esteemed institute KLE Technological University, Hubballi has provided us an opportunity to fulfill the most cherished desire to reach our goal. We would like to thank Dr. Nalini Iyer , HOD of School of Electronics and Communication for providing us an opportunity to enhance our thinking abilities and cultivating constructive skills to help complete our project successfully.

We sincerely thank our staff in-charge Prof. Shraddha Hiremath for her consistent support and suggestions. We also thank the complete ADLD team for their guidance and support.

We would also like to acknowledge our friends and seniors for helping us with the resources and for being the source of blessing and aspiration.

Finally, we would like to thank all those who either directly or indirectly made a difference in the project.

-Nidhi Darak, Avinash V V, Pavan Myagerimath, Harsha Jangamani

ABSTRACT

In a SOC, for effective operation different modules need to be in sync with each other. Different blocks adhere to different protocol and each block has unique bit rate or baud rate for data transfer that can be either asynchronous or synchronous. Therefore there is a need for an interface that bridges different protocols. In this project we have designed and implemented an interface that interconnects apb slave and i2c controller that has been synthesizable using verilog HDL. We have used Arty-7 family device xc7a100t – CSG324(device package) in Xilinx 14.7 version. Inter-integrated circuit also known as I2C, is an 8 bit serial communication protocol that needs two bus wires(SDA and SCL) works on 100KHz to 1MHz. Advanced Peripheral Bus (APB) is 32 bit protocol that is a part of the Advanced Microcontroller Bus Architecture (AMBA) protocol family works on 30MHz(in SOC). The data is exchanged through APB convention to I2C convention utilizing apb slave.

Contents

1 Introduction **9**
- 1.1 I2C Protocol . . . 9
- 1.2 APB Protocol . . . 12
- 1.3 Motivation . . . 14
- 1.4 Objectives . . . 14
- 1.5 Literature survey . . . 15
- 1.6 Problem statement . . . 20
- 1.7 Application in Societal Context . . . 20
- 1.8 Project Planning and bill of materials . . . 20
- 1.9 Organization of the report . . . 21

2 System design **22**
- 2.1 Functional Block Diagram . . . 22
- 2.2 Design alternatives . . . 23
- 2.3 Final design . . . 24

3 Implementation details **26**
- 3.1 Specifications and final system architecture . . . 26
- 3.2 Algorithm . . . 27
- 3.3 Flowchart . . . 28

4 Optimization **31**
- 4.1 Introduction to optimization . . . 31
- 4.2 Types of Optimization . . . 31
- 4.3 Selection and justification of optimization method . . . 32

5 Results and discussions **33**
- 5.1 Result Analysis . . . 33

6 Conclusions and future scope **37**
- 6.1 Conclusion . . . 37
- 6.2 Future scope . . . 37

References **37**

Appendix **37**

Appendix **37**

List of Tables

1.1 Bill Of Materials 20

List of Figures

1.1 I2C Data Frame 10
1.2 Start and stop condition 12
1.3 Data validity 12
1.4 Write Operation 13
1.5 Read Operation 14
1.6 Gantt Chart 21

2.1 FUNCTIONAL BLOCK DIAGRAM 23
2.2 Best Design 23
2.3 Alternative Design 24
2.4 Final Design 24

3.1 Architecture 26
3.2 Flowchart 28
3.3 FSM for I2C 29
3.4 FSM for APB 30

5.1 Configuring control registers 33
5.2 start bit 33
5.3 stop bit 34
5.4 Data validity 34
5.5 FPGA Implementation 35
5.6 Write operation to Arduino UNO 35
5.7 Read operation to Arduino UNO 36

Chapter 1

Introduction

This chapter covers insights about introductory part of I2C protocol, APB protocol, motivation, objects, literature survey, problem statement of the project and further more bill of the materials used in this project.

In a multiprocessor system, there are processors with high performance and processors with low power consumption to accommodate a variety of applications. High extreme performance processors are used for complicated processing applications, whereas low power consuming processors are employed for other basic applications. Power is therefore conserved.

Different Processors use different protocols for communication with peripherals. This is drawback as we require peripheral clones for different processor that simply differ in protocol. For instance, if 10 peripherals are used, a dual processor environment requires a total of 20 peripherals. Given that the area is doubled, this is quite ineffective. We can utilise 10 peripherals that support one common protocol and an interface that changes the other protocol to the common protocol in place of 10*2 peripherals.

1.1 I2C Protocol

Inter-Integrated circuit developed 2 decades ago by Philips Semiconductor is used for serial communication of data between various IC's present in the core. Under 100kHz communications, it is the first verbal description which is provided for seven bit addresses with 112 divides on the bus. In 1992, the primary public specification was discovered, i.e. 400kHz fast-mode along with d 10-bit address block. There are three more modes, fast mode which works at 1MHz, high-speed mode working at 3.4MHz and ultra-fast mode working at 5MHz.

Peripheral devices in embedded systems are connected to the MCU, which is mapped to memory input and output devices. The analysis con-

ducted by Philips Labs in the Netherlands) to beat the issues related to I2c complexity resulted in a 2-wire communication bus known as the I2C bus. It's name explains its purpose i.e. It produces a communication link between the integrated circuits.

I2C combines the most effective options of SPI and UARTs. In I2C,a master van be connected to multiple slaves and we can connect multiple masters to a single slave or multiple slaves. I2C has two wires like UART, they are SDA and SCL. Serial data line is the road for the master and slave to send and receive data. Serial clock line is the road that carries the clock signal. In I2C, data is transferred bit by bit on the data line. It is synchronous in nature, i.e. the output bits are synchronous to the sampling of bits of the clock signal which is shared between the master and the slave. The clock signal is usually controlled by the master.

Unlike SPI, I2C doesn't have a slave select line. Therefore, it uses addressing method to communicate with a particular slave. The address block is the block that always comes first after the start condition in a message frame. Here, the master will send address of slave with which it wants to communicate, then all the slaves connected to the master will compare their address to the one sent by the master and if the address matches, the data line will switch from high voltage to low voltage. The slave will acknowledge the master using acknowledgement bit.

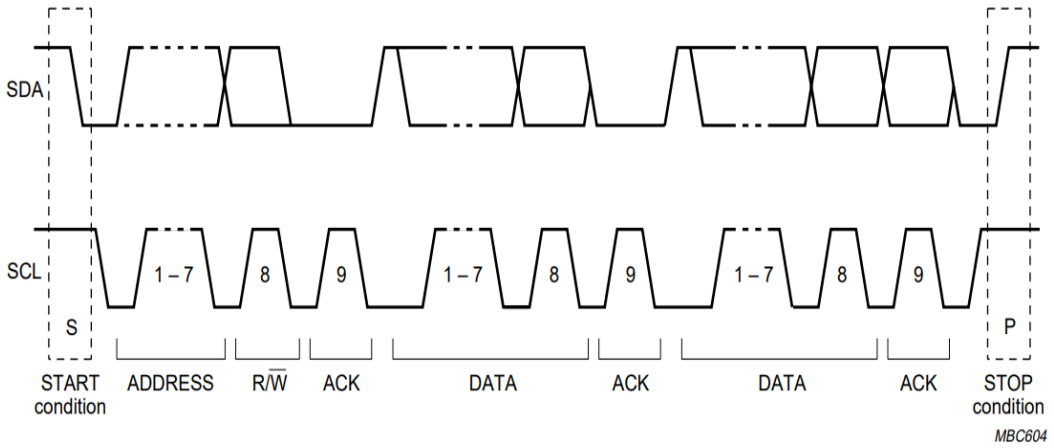

Figure 1.1: I2C Data Frame

Procedure for a master to access a slave device is that the following:

1. Suppose a master needs to send information to a slave:
   - Master-transmitter sends a begin condition and addresses the slave-receiver
   - Master-transmitter sends information to slave-receiver
   - Master-transmitter terminates the transfer with a STOP condition
2. If a master needs to receive/read information from a slave:
   - Master-receiver sends a begin condition and addresses the slave-transmitter
   - Master-receiver sends the requested register to scan to slave-transmitter
   - Master-receiver receives information from the slave-transmitter
   - Master-receiver terminates the transfer with a STOP condition

Write Condition :

To write on the I2C bus, the master will send a begin condition on the bus with the slave's address, as well as the last bit (the R/W bit) set to zero, that signifies a write. Once the slave sends the acknowledge bit, the master can then send the register address of the register it desires to jot down to. The slave can acknowledge again, rental the master comprehend it is prepared. After this, the master can begin causing the register information to the slave, till the master has sent all the information it has to (sometimes this is often solely one byte), and the master can terminate the transmission with a STOP condition.

Read Condition :

Reading from a slave is like writing, however with some further steps. to scan from a slave, the master should 1st instruct the slave that register it desires to scan from. this is often done by the master starting off the transmission in a very similar fashion because the write, by causation the address with the R/W bit equal to zero (signifying a write), followed by the register address it desires to scan from. Once the slave acknowledges this register address, the master can send a begin condition once more, followed by the slave address with the R/W bit set to one (signifying a read). This time, the slave can acknowledge the scan request, and also the master releases the SDA bus, however can continue activity the clock to the slave. During this a part of the group action, the master can become the master-receiver, and also the slave can become the slave-transmitter.The master can continue causation out the clock pulses, however can unharness the SDA line, in order that the slave will transmit information. At the top of each computer memory unit of information, the master can send AN ACK to the slave, rental the slave know that it's prepared for a lot of information. Once the master has received the amount of bytes it's expecting, it will send a NACK, signal to the slave to halt communications and unharness the bus. The master can follow this up with a STOP condition.

Today, the I2C bus is employed in several alternative application fields than simply audio and television equipment. The bus is usually accepted within the trade as a de-facto customary. The I2C bus has been adopted by many leading chip making industries like ST electronics, Infineon Technologies, , Lone-Star State Instruments, Maxim, Atmel, Analog Devices etc.

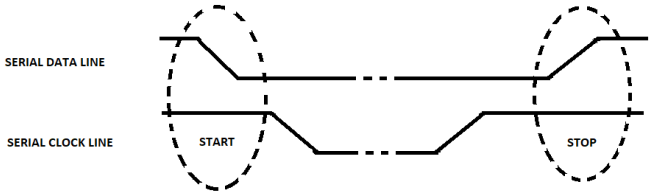

Figure 1.2: Start and stop condition

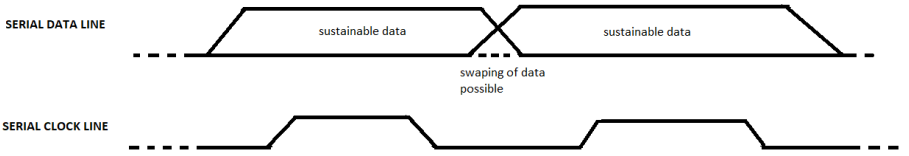

Figure 1.3: Data validity

1.2 APB Protocol

The Advanced Microcontroller Bus Architecture (AMBA) protocol family includes the Advanced Peripheral Bus (APB). It defines a low-cost interface that is designed to use the least amount of power and have a simple interface.

Use the APB protocol to connect to low-bandwidth peripherals that do not need the high performance of the AXI protocol because it is not pipelined. To make it easier to integrate APB peripherals into any design flow, the APB protocol links a signal transition to the rising edge of the clock. It requires at least two cycles for each transfer.

APB advance peripheral bus have signal PCLK, PADDR, PWRITE, PSEL, PENABLE, PWREAD, PWDATA,PWDATA and PREADY before the

operating states the PCLK must be activated and PSEL the slave must be selected by the process once the slave has been selected the PENABLE signal must be at a high state to be able to perform the operations

- Write operation

If the PWRITE signal is high write operation is performed during this state no other signals can switch as the data is sent to the I2C bridge, if any of the signals fall the PSLVERR signal must be high so that any unwanted data that has been read by APB must remove or ignore the signal or data

- Read operation

If the PWRITE signal is low the read operation is performed during this state the PWRITE signal must be switched after the PREADY signal so that the transfer from the buffer is read by the PWREAD signal and sent back to the processor if any of the signals prior to this signal switches the PSLVERR must much high Error response PSLVERR is used to indicate whether the transfer has any errors while sending or receiving the data this only happens while the read operation is occurring the signals switches which might cause the PWDATA or PWREAD to receive unwanted signals which can be reduced by using PSLVERR signal

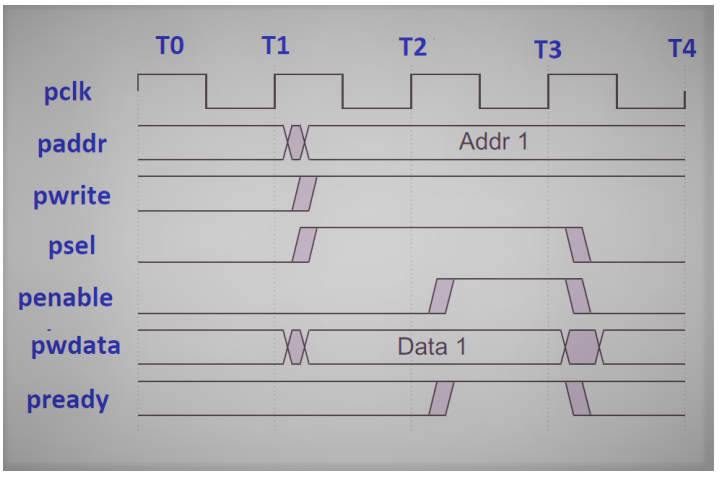

Figure 1.4: Write Operation

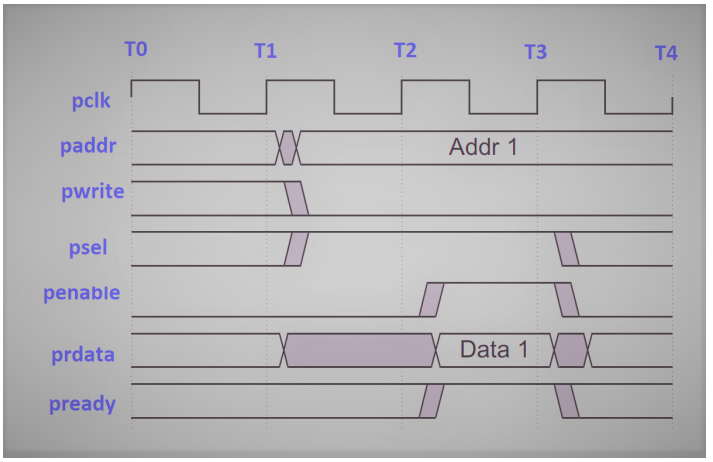

Figure 1.5: Read Operation

1.3 Motivation

In a processor or SOC, for short distance communication within the board there is need for low cost communication protocol. In some applications like inter device communication multiple masters must be used. Hence there is a need for a protocol that supports multi-masters and has simple circuit. I2C supports multimaster unlike costlier SPI, UART protocols. SPI needs separate select lines for each slave and also lack of flow control makes it less reliable. I2C protocol uses only two wires (SDA and SCL), were both masters and slave share same data and clock lines. I2C supports acknowledge feature which makes error correction effortless.

1.4 Objectives

- Understand the system overveiw and usage of I2C.
- Arrive at the specification of the I2C.
- Recongnise the fundamental working and design of the I2C.
- Arrive at the functional block representation of the communication betwen APB and I2C.
- Design and implementation of I2C protocol and its communicatiob to APB.

1.5 Literature survey

References:

I2C:

[1]The focal point of this research paper is efficient designing and implementing the Inter-Integrated Circuit (I2C) using Verilog language, which consists of single master and bidirectional data line i.e., serial data line and serial clock line. This protocol can support multiple masters. I2C is a two-wire, bi-directional serial bus which provides a simple and efficient way to exchange data between devices and it is used when faster devices need to communicate with slower devices without any data loss. It is equipped to ensure no data loss. The protocol concept comes into the picture for reducing the hardware complexity and power consumption. It has different types of features like addressing mode, different data's pattern, clock frequency, and operation of FPGA. All these features will help our I2C to run for longer hours. [2]Since our architecture consists of master and slave, the master helps in generating a START condition at which the serial data line will switch from high voltage level to low voltage level and the serial clock line will remain high. Besides that, master also generates STOP condition. At stop condition, clock line remains high and serial data line will switch from low voltage level to high voltage level. Additionally, there are different types of messaging modes like read transaction, write transaction, write-read transaction and read-write transaction. The design follows I2C specifications for address sending and data transfer operations. On the other hand, master also transfer and receive data to or from slave devices using different addressing mode. The addressing mode that is implemented consists of 7 bits. The data read and write operations are carried out between master and slave using address block, ACK/NACK bits, data clock, start and stop conditions. Whenever the master and slave transactions are a success, they will be acknowledged using ACK bit. Whenever the address sent by the master is matched with a particular slave, it will acknowledge the master. Then the master will send data to the slave and after every eight bits, the mater will be acknowledged by the slave. This process is repeated, and the termination of the process will be done using stop bit. The complete module is designed in Verilog and results are observed using Xilinx tool which provides the efficient

data transfer and synchronization by using acknowledgement. Any low-speed peripheral devices can be interfaced using I2C bus protocol as a master This project can be further extended to design for multiple multi-master protocol where two and more masters are used to communicate to support the external devices.

APB:

[3]The APB (Advanced peripheral bus) protocol is a part of AMBA (advanced microcontroller bus architecture) family. It establishes the communication between master and slave. APB is a low bandwidth, low cost and minimal power consumption and is used to connect many devices such as Timer, Keypad to the bus architecture. Different signals involved in APB transactions are PCLK, PADDR, PWRITE, PSEL, PENABLE, PWDATA and PREADY. PWDATA and PREADY before the operating states the PCLK must be activated and PSEL the slave must be selected by the process once the slave has been selected the PENABLE signal must be at a high state to be able to perform the operations Write operation If the PWRITE signal is high write operation is performed during this state no other signals can switch as the data is sent to the I2C bridge, if any of the signals fall the PSLVERR signal must be high so that any unwanted data that has been read by APB must remove or ignore the signal or data Read operation If the PWRITE signal is low the read operation is performed during this state the PWRITE signal must be switched after the PREADY signal so that the transfer from the buffer is read by the PWREAD signal and sent back to the processor if any of the signals prior to this signal switches the PSLVERR must much high Error response PSLVERR is used to indicate whether the transfer has any errors while sending or receiving the data this only happens while the read operation is occurring the signals switches which might cause the PWDATA or PWREAD to receive unwanted signals which can be reduced by using PSLVERR signal [4] The operation of APB is done by using finite state machine. There are totally three states namely, IDLE, SETUP and ACCESS states. In APB every transfer takes at least two cycles (SETUP phase and ACCESS phase). The first transfer will take three clock cycles, but the following transactions are completed in two clock cycles. In Idle state also known as default state no operation is performed. Setup state is active when the transfer is required which indicates beginning of transfer. The bus will enter setup phase only

when the transfer is needed, otherwise bus will remain in the idle phase. On the upcoming positive clock or rising edge of clock, the bus will automatically move to ACCESS state. This state is mainly used to tell that the transaction is completed. If further transactions are needed, then bus will move to SETUP phase otherwise it moves to the default state.

Bridge:

[5] This work describes APB to I2C interface. Data is sent and received by I2C on SDA. SCL is clock for synchronization. APB uses PCLK for synchronization. Data is sent and received on separate parallel lines, PWDATA and PRDATA respectively. Therefore parallel data must be converted to serial data. SCL and PCLk has different frequencies that must be considered. An interface is designed using FIFO to receive APB parallel data and store it. Then I2C interface is designed to take data from FIFO. Interfacing between I2C and APB Protocol consists of two major parts. They are APB slave and I2C Master. These two blocks bridges the communication I2C Master and APB Slave. APB Slave receives the data from APB Master in respective format and provides it to I2C master through FIFO.

FIFO:

[6]This paper proposes design of asynchronous FIFO on FPGA for the purpose of high-speed, steady data transmission between asynchronous clock domains. The FIFO constructed is dual-port RAM. In the design, the difficult is to generate the empty and full flag, because the flag of input control is due to output, similarly the flag of output control is usually generated by input. This logic takes pointer as input and generates the required condition. If pointer=0, empty condition is generated and if pointer =15 full condition is generated. Similarly almost full, almost empty, half full conditions are generated for required values. Since pointer differences is calculated with respect to both read and write clocks, FIFO status assertion is immediate with zero clock delay. Control blocks decide the enabling of read and write once the empty and full conditions are asserted. After the assertion of empty flag read pointer should not increment unless and until there is at least one data is written. Thus once empty flag is asserted next read control logic looks into the write domain for any write activity. It disables read next signal till it finds that data has been written to

FIFO. Immediately after the write operation (i.e. posedge of write clk) control logic enables read next signals.

[7] This paper proposes the idea regarding the generation of I2C clock and the register useful for designing the I2C Master. There is a counter in the clock generator. By manipulating the divisor registers, the user can use the counter to generate the I2C clock and a divide signal as needed. The operational registers are used to hold status and programme the controller. The registers, which the I2C controller uses to operate the I2C interface, are configured using the APB interface block.

Verilog:

[8]Value Set Verilog supports four values to model the practicality of real hardware. In Verilog we tend to use Nets and wires. Nets represent connections between hardware components. Nets are declared primarily with the keyword wire. Note that web isn't a keyword however represents a category of information varieties like wire, wand, wor, tri, triand, trior, trireg, etc.

Registers: Registers are used for storing knowledge. Registers retain price till another price is placed onto them Register knowledge varieties ar normally declared by the keyword reg. Vectors Nets or reg knowledge varieties will be declared as vectors with multiple bit breadth.

Integers: Integers are general purpose register knowledge kind used for manipulating quantities. Integers are declared by the keyword number. Here we tend to conjointly use real numbers and these constants and real register knowledge varieties ar declared with the keyword real.

Time knowledge types: In Verilog language, simulation is finished with regard to simulation time. A special time register knowledge kind is employed in Verilog to store simulation time. A time variable is asserted with the keyword time. The breadth for time registers knowledge varieties is implementation specific however is a minimum of sixty four bits.

Arrays: we tend to conjointly use Arrays that are allowed in Verilog for reg, integer, time, and vector register knowledge varieties. Arrays aren't allowed for real variables.

Parameters: Parameters Verilog permits constants to be outlined during a module by the keyword parameter. Parameters can't be used as variables.

Strings: Strings will be hold on in reg. The breadth of the register variables should be giant enough to carry the string. every character within the string takes up eight bits (1 computer memory unit

Special character: Special characters serve a special purpose in displaying strings, like newline, tabs and displaying argument values. Special characters will be displayed in strings only if they're preceded by escape characters

[9] Arty A7 development board is AN Artix-7 FPGA Development Board for manufacturers and Hobbyists. it had been designed specifically to be used as a MicroBlaze Soft process System. once utilized in this context, pretentious becomes AN improbably versatile process platform, capable of adapting to no matter your project needs. in contrast to different single board computers, pretentious is not guaranteed to one set of process peripherals; one moment it is a communication powerhouse full of UARTs, SPIs, IICs, ANd an local area network waterproof, and therefore the next it is a meticulous timekeeper with a dozen 32-bit timers. pretentious can become the foremost variable tool for our project tool box. Features: • Arty A7-100T options the larger Xilinx XC7A100TCSG324-1 IC • Internal clock speeds olympian 450MHz • On-chip analogue-to-digital device (XADC) • Programmable over JTAG and Quad-SPI Flash • 256MB DDR3L with a 16-bit bus @ 667MHz • 16MB quad-SPI flash • USB-JTAG programming electronic equipment (Micro USB cable not included) • Powered from USB or any 7V-15V supply • System connectivity-10/100 Mbps local area network, USB-UART Bridge • 4 Pmod connections and Arduino/chipKIT protect connector

1.6 Problem statement

Design and Implementation of an Interfacing Protocol between APB and I2C for an SOC

1.7 Application in Societal Context

- When several devices needed to be connected to a microcontroller, the I2C bus was created to lessen congestion, take up less space, and use less power.
- Due to the growing demand for high-speed, high-performance systems, parallel buses were used by microcontrollers. This led to the requirement for serial bus interface and parallel bus interfaces.
- This led to the requirement for serial bus interface and parallel bus interfaces. By combining the I2C and other interfaces with the APB AMBA bus, was able to reduce the chip space.

1.8 Project Planning and bill of materials

SL.No	Name of the Component	Quantity	Cost in Ruppes
1	FPGA board	1	2708
2	Bread board	1	60
3	Jumper wires	5	20
4	Arduino UNO cable	1	60
5	Mobile adapter with B cable	1	70
6	Arduino UNO	1	700

Table 1.1: Bill Of Materials

	PROJECT NAME	PROJECT DURATION	PROJECT START DATE	PROJECT END DATE										
	INTER-INTEGRATED CIRCUIT	25 Hours	06-09-22	15-12-22										
TASK	TASK DESCRIPTION	TASK DURATION (Hours)	START DATE	END DATE	06-9-22	16-9-22	26-9-22	6-10-22	16-10-22	5-11-22	15-11-22	25-11-22	5-12-22	15-12-22
1	Generating Problem Statement	2	06-9-22	11-09-22										
2	Initialization	2	12-09-22	22-09-22										
3	Literature Survey	3	23-09-22	07-10-22										
4	Analysis of multiple solution	3	08-10-22	16-10-22										
5	Project Planning	3	17-10-22	26-10-22										
6	Functional Specification	2	03-11-22	12-11-22										
7	Design	2	13-11-22	23-11-22										
8	Simulation	3	24-11-22	8-12-22										
9	Implementation	3	09-12-22	14-12-22										
10	Closure	2	15-12-22	15-12-22										

Figure 1.6: Gantt Chart

1.9 Organization of the report

- Chapter2: In this chapter we given overview of the model which conveys functionality, design architecture and justification for selected design.
- Chapter3: In this chapter we have discussed flow of the code using algorithms and flowchart
- Chapter4: In this chapter we have discussed some effective optimization techniques to model fully functional system.
- Chapter5: Here we obtained results for our model and verified the results.
- Chapter6: In this chapter we have concluded our report by mentioning future scope.

Chapter 2

System design

This chapter covers the basics of functional block diagram of APB slave, I2C Bridge, I2C Master and I2C slave. Alternative designs of communication between I2C and APB is also discussed

2.1 Functional Block Diagram

Functional blocks consist of APB slave connected to I2C controller through I2C bridge. PCLK is clock connected to APB slave by APB master. Presetn is active low asynchronous reset. PAddr is an address bus from master to slave to specify location where the data must be written to, it can be upto 32 bit wide. PWDATA is a write data bus from master to slave. PRDATA is a read bus slave to master used to send the read data to master. PWRITE signal indicates read or write operation. PREADY-slave uses this signals to include wait states in the transfer. In other words, anytime a slave is not prepared to finish a transaction, it will de-assert the PREADY and ask the master for some further time. PENABLE - Denotes the second and its next transfer cycles. The ACCESS phase begins when PENABLE is asserted. APB slave block consists of memory, control registers. In the write operation from the APB master data is written into memory of the APB slave by configuring control registers. I2C bridge acts as bridge to exchange data between I2C controller and APB slave. It can be a fifo or some data registers.

In the write operation from the APB master data is written into memory of the APB slave by configuring control registers. I2C bridge acts as bridge to exchange data between I2C controller and APB slave. It can be a fifo or some data registers. The last block denotes I2C controller which acts I2C master facilitates I2C Protocol. Converts parallel data in I2C bridge to serial data.

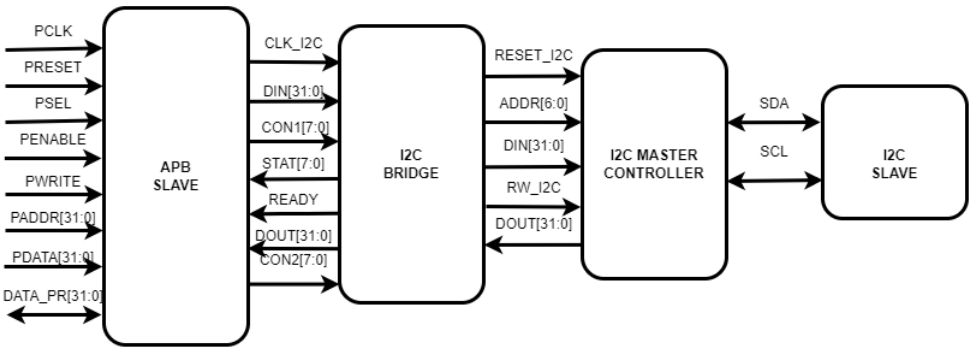

Figure 2.1: FUNCTIONAL BLOCK DIAGRAM

2.2 Design alternatives

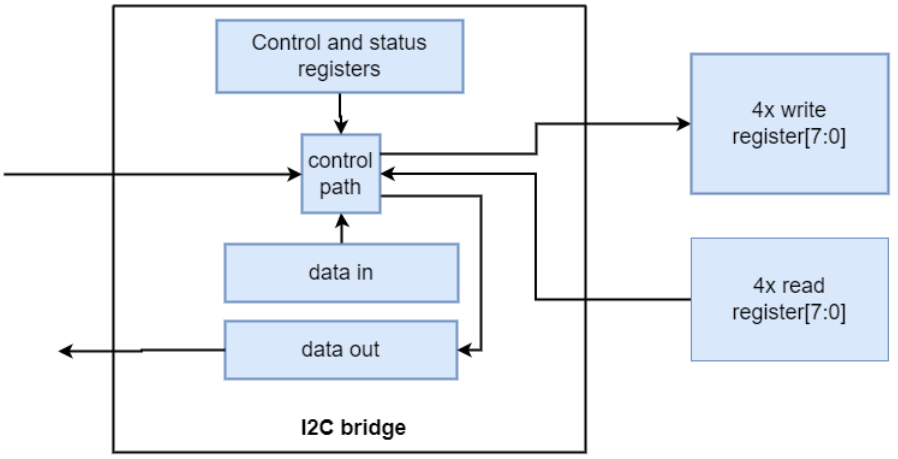

Figure 2.2: Best Design

In design 1 I2C bridge block performs exchange of data from I2C controller and APB slave block using controller registers and some data in and data out registers of size 32 bit. Control registers provides information such as busy/idle state count of the byte sent , it also allows to operate I2C for different bit rate by configuring it for different prescalar values for different frequencies. Control registers also hold the slave address and specify read/write information to I2C controller.

In design 2 I2C bridge block is implemented by using a fifo. Initially 32 bit APB slave Data is written into fifo memory. Fifo has width of 32 bit

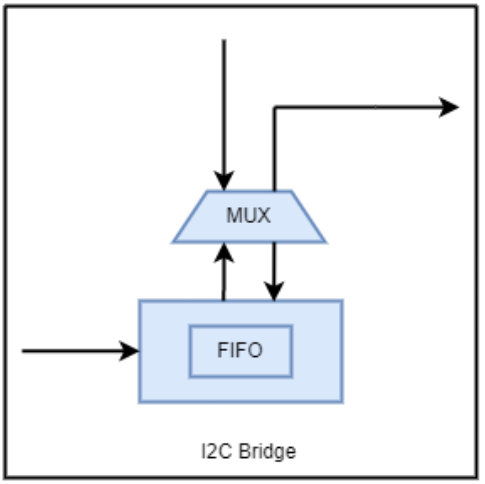

Figure 2.3: Alternative Design

to hold 32 bit of data written from APB. If fifo is full I2C controller takes first 8 bit of data for transmission. This is repeated until fifo is empty. Similarly for read operation first 8 bit of data is written in fifo data read registers, this is repeated until fifo is full. When the data read register in fifo is full. APB slave carries the read operation.

2.3 Final design

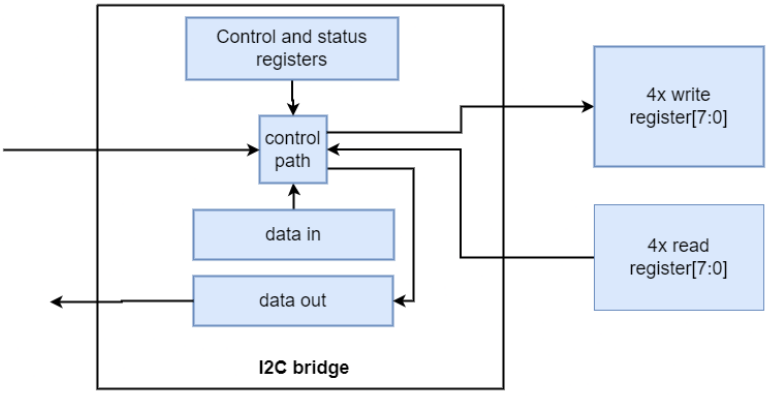

Figure 2.4: Final Design

The selected design is optimal and best architecture for the implementation. It supports different modes of operation which enables us to operate in different data transfer rates(100kHz/400Hz/3MHz/5MHz). If the I2C master is busy in read or write operation it acknowledges the APB slave so that processor can complete other operations until I2C is available for use. It also supports different size of data transfer(32bit/24bit/16bit/8bit). The design includes two registers of 32 bit each which is simple rather than having FIFO which is complex to implement.

Chapter 3

Implementation details

This chapter covers the implementation details of I2C and APB communication, Algorithm of architecture used and flowchart of I2C and APB.

3.1 Specifications and final system architecture

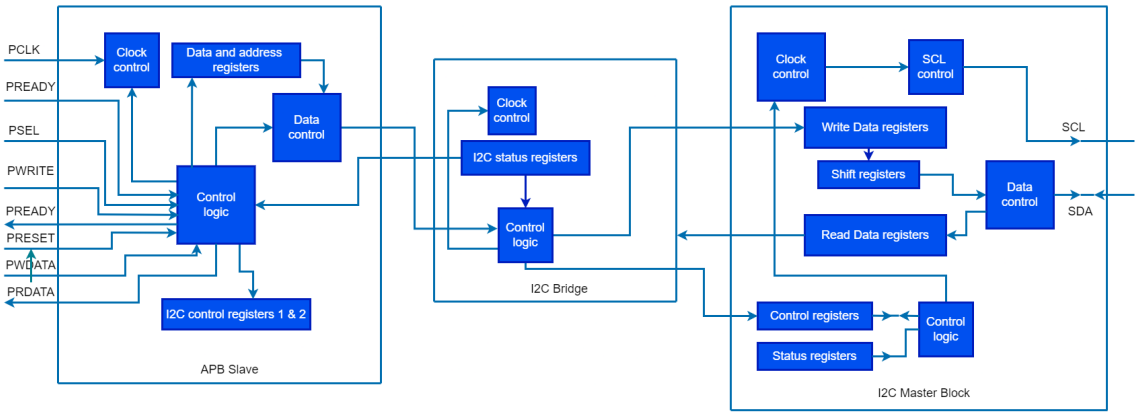

Figure 3.1: Architecture

- APB clock frequency- 30MHz.
- Write and read operation for different data size(8 bit, 16 bit, 24 bit, 32 bit).
- Repeated start condition.
- Different mode of operation to work on different frequency of operation(100KHz, 400KHz, 3MHz, 5MHz).

- Satisfy data validity condition.
- Control and status registers to configure different modes.
- Status registers which provides I2C controller status information.

3.2 Algorithm

- Step 1: Start
- Step 2: Check for Control and Status Registers
- Step 3: Wait Until I2C Master is Idle
- Step 4: Configure Control Registers-Set frequency of operation and byte count
- Step 5: Set RW bit=0 for write operation and 1 for write operation
- Step 6: Write operation- send start bit, followed by 7 bit of address and 1 RW bit.
- Step 7: Slave acknowledges if address is matched.
- Step 8: First byte is sent serially.
- Step 9: Acknowledgement is received from slave. Similarly other 3 byte if data is sent
- Step 10: Read operation-send start bit, followed by 7 bit of address and 1 RW bit.
- Step 11: Slave acknowledges if address is matched.
- Step 12: 8 bit of serial data is received from slave and gets stored in read data registers in i2c controller block.
- Step 13: Acknowledgement is sent by master after receiving data. Similarly other 3 bytes if data is received.
- Step 14: Send Stop bit
- Step 15: 32 bit read data is will be stored in data out register and read by APB slave in the read cycle.
- Step 15: Stop

3.3 Flowchart

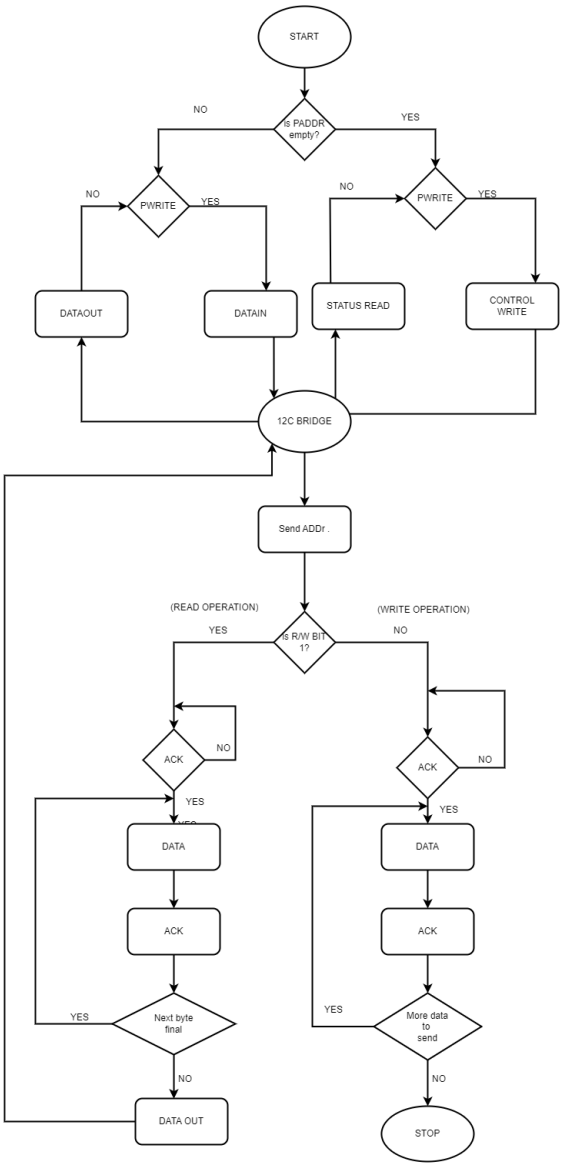

Figure 3.2: Flowchart

- Finite state machine of I2C

Write Condition:

To write on the I2C bus, the master will send a begin condition on the bus with the slave's address, as well as the last bit i.e., the R/W bit

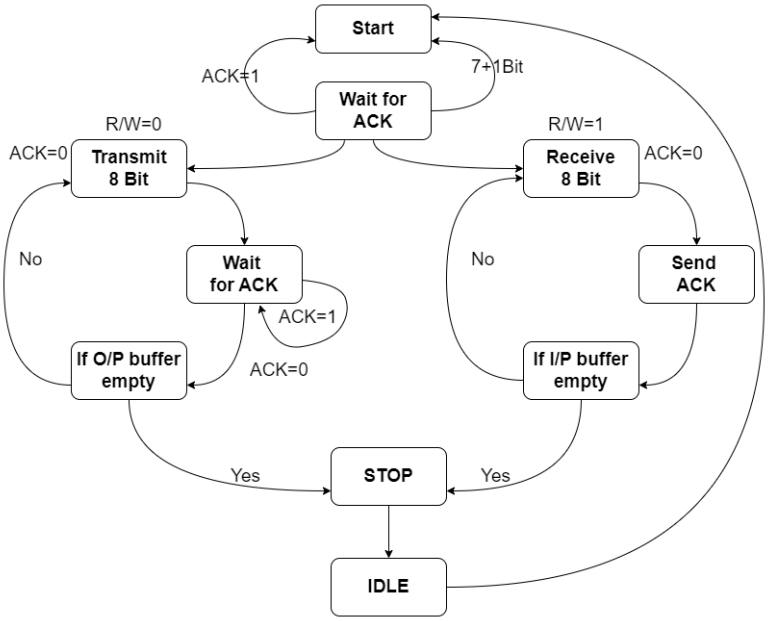

Figure 3.3: FSM for I2C

is set to zero, which signifies write condition. Once the slave address is matched it sends the acknowledge bit to the master. The serial data line will be switched from low voltage level to high voltage level. The clock line will remain high. Now the master can begin sending the data to the slave and after every eight bits master will be acknowledged with an acknowledgement bit. After sending all the information(data), the master can terminate the transmission with a STOP condition.

Read Condition:

Reading from a slave is same as writing operation with few extra steps. Then the R/W bit equals to one(signifying a read), followed by the register address it desires to scan from. The master should first send the address of the slave from which it wants to read the data. This is often done by the master before starting the transaction in a very similar fashion of write operation. Once the slave acknowledges the master, the master can read the data from the slave.This time, the slave can acknowledge the scan request, and also the master releases the SDA bus, however can continue activity the clock to the slave. During this a part of the group action, the master can become the master-receiver, and also the slave can become the slave-transmitter.

- Finite state machine of APB

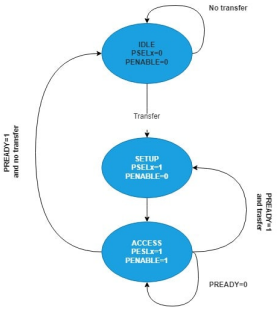

Figure 3.4: FSM for APB

The state machine operates through the following states:

- IDLE: The APB is currently in this default state.
- SETUP: The bus enters the SETUP state, where the proper choose signal, PSELx, is asserted, when a transfer is necessary. The bus always transitions to the ACCESS state on the following rising edge of the clock after spending just one clock cycle in the SETUP state.
- ACCESS: In the ACCESS state, the enable signal, PENABLE, is asserted. When switching from the SETUP to the ACCESS state, the address, write, select, and write data signals need to be steady.

The PREADY signal from the slave controls exit from the ACCESS state:

- The peripheral bus remains in the ACCESS state if PREADY is held LOW by the slave, but the IDLE state is entered if no additional transfers are needed if PREADY is driven HIGH by the slave.
- Another transfer comes after, on the other hand, the bus goes straight to the SETUP state.

Chapter 4

Optimization

Optimization is process of choosing best method to enhance the performance of a system interms of time,power,area and size. This chapter covers the basic understanding of Optimization and types of Optimizations.

4.1 Introduction to optimization

Optimization in the area of protocol design includes efficient data transfer, data validity , low power dissipation etc. Power consumption is an important aspect as we move towards more battery dependent technologies. Dynamic power accounts for upto 50 percent of the total power dissipation. Hence various power management techniques are used to reduce power dissipation.

4.2 Types of Optimization

Some of the useful and widely used power optimization include Clock gating and frequency scaling.

- Clock gating method is used when the circuit is not in use. When the circuit is not ON, providing clock signal results in power dissipation. Hence clock gating technique removes the clock when it is not required.
- Another popular optimization method to reduce is frequency scaling whereby operating frequency of the circuit can be changed according required operation.This saves energy when needed and also makes compatible with the slaves working on different data transfer rates.

4.3 Selection and justification of optimization method

Clock gating technique can be implemented at gate level or RTL level. It is more easy and effective to implement it at gate level. Hence we choose frequency scaling optimization method in the RTL code. We have used 4 mode of operation which include different frequency of operation i,e 100kHz , 400kHz , 3MHz and 5MHz.

Chapter 5

Results and discussions

5.1 Result Analysis

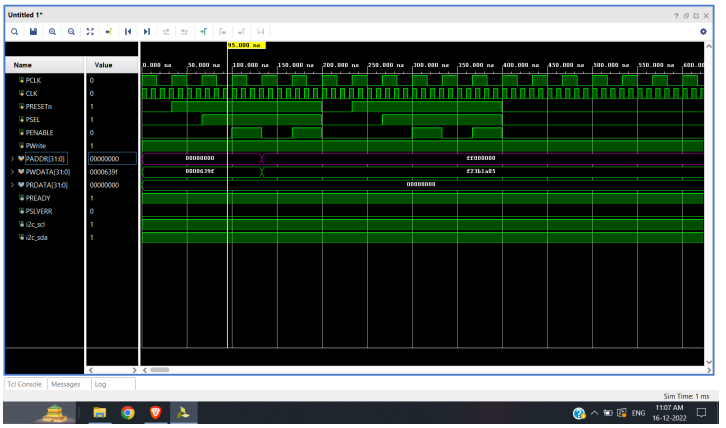

Figure 5.1: Configuring control registers

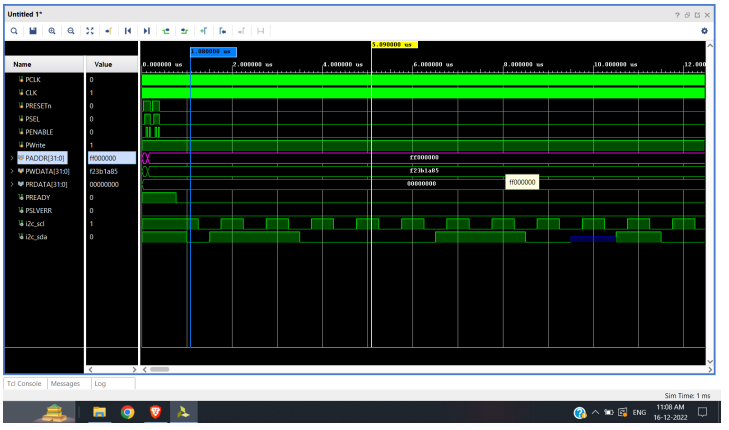

Figure 5.2: start bit

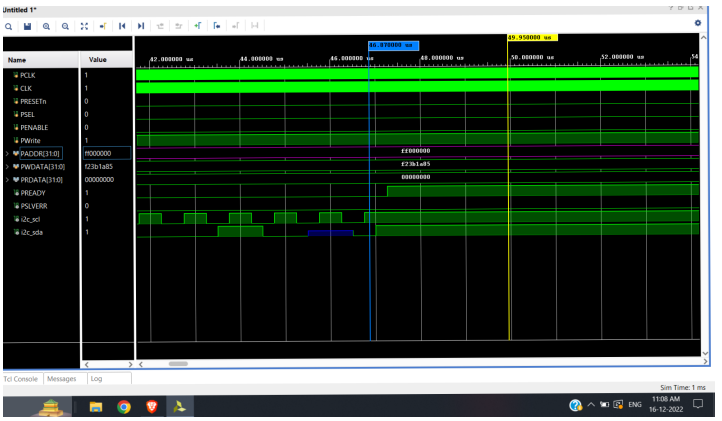

Figure 5.3: stop bit

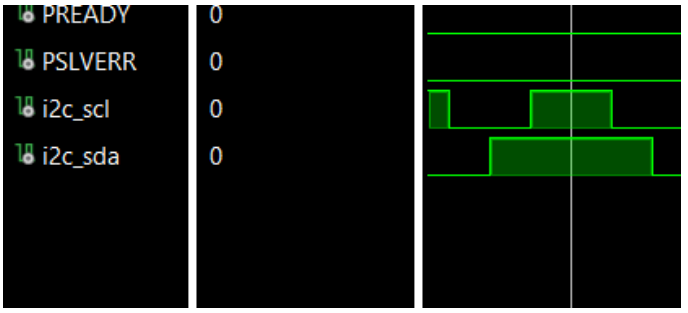

Figure 5.4: Data validity

- As it is seen in the simulation results (figure 5.1), when paddr line is 0, control registers are configured i,e size of data to be sent, repeated start condition, mode of operation for different frequencies.
- Start bit in I2C serial transmission is, when SCL line is high SDA line should go from high to low state. Thus figure 5.2 verifies the start bit.
- In I2C when the SCL line is high SDA should go from low to high state, this is the condition for stop bit and the same is verified in the simulation shown in the figure 5.3 .
- In synchronous circuits if data line changes at the edge of the clock line it can result in data loss. Hence it must ensured that data is changed at the low level or high level of the clock, figure 5.4 shows data validity condition.
- figure 5.5 shows fpga implementation for write and read operations using aruidno uno as slave. The results of read and write operation are seen in below figures.

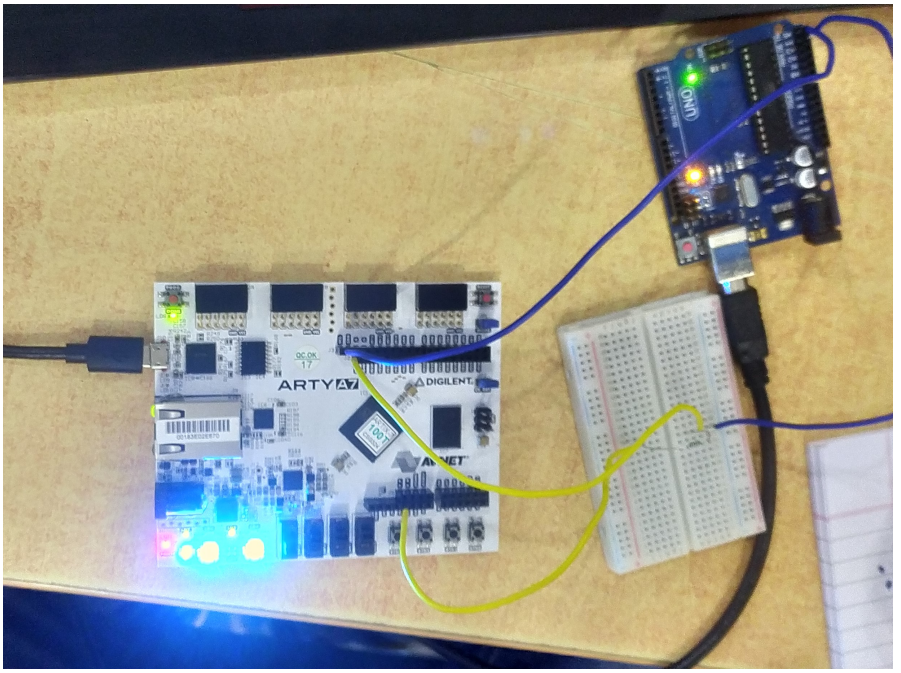

Figure 5.5: FPGA Implementation

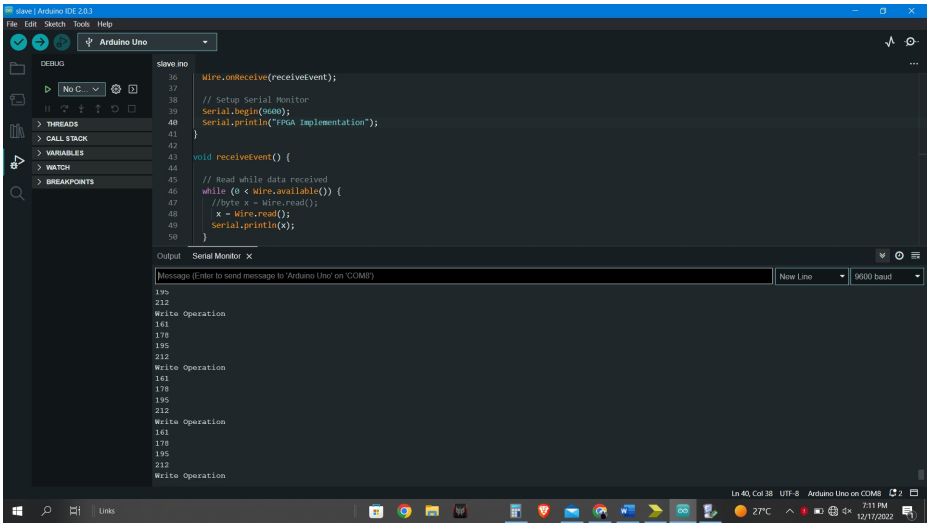

Figure 5.6: Write operation to Arduino UNO

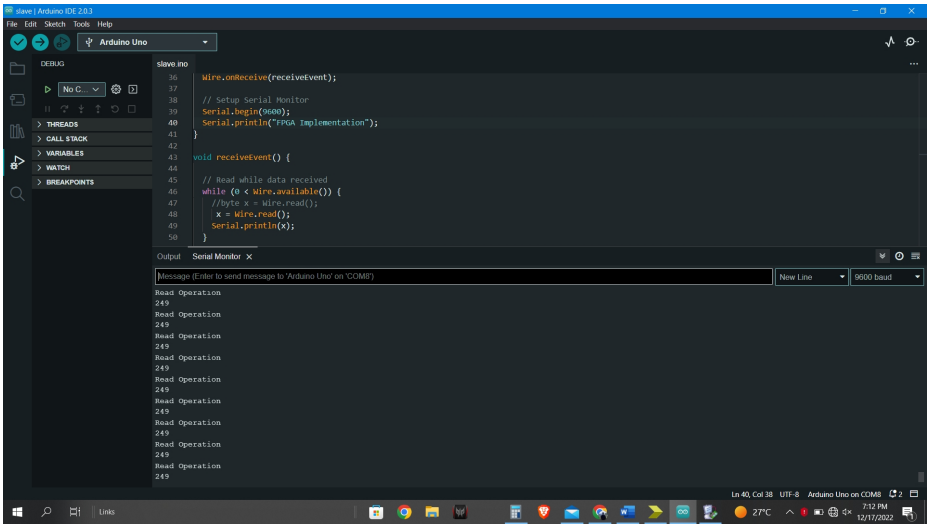

Figure 5.7: Read operation to Arduino UNO

Chapter 6

Conclusions and future scope

6.1 Conclusion

This paper has shown up results of simulation and FPGA implementation of interface between APB and I2C controller by considering Arduino UNO as slave using Verilog HDL. The project demonstrates how APB slave transmits and receives data to and from I2C slave through I2C master. We have used Arty-7 family device xc7a100t – CSG324(device package) in Xilinx 14.7 version. The project has resulted an efficient system that interfaces APB protocol with I2C protocol. Simulation results are verified and data transfer from APB slave to I2C slave can be clearly seen in provided simulation results. Implementation results are verified using arduino uno as slave for read and write operations.

6.2 Future scope

Data collision avoidance-

Clock stretching:I2C devices have the ability to stretch SCL in order to slow down communication: During a SCL low phase, any I2C device on the bus may also hold down SCL in order to prevent it from rising again, allowing it to reduce the SCL clock rate or temporarily halt I2C communication.

Multimaster-

Arbitration: The same I2C bus can support many I2C multi-master connections, all of which can run simultaneously. They can tell whether the bus is idle or not right now by continuously checking SDA and SCL for start and stop conditions. Master delays pending I2C transfers if the bus is busy until a stop condition shows that the bus is free once again.

Bibliography

1.International Journal of Scientific Engineering and Science. Volume 1, Issue 7, pp. 70-74, 2017. ISSN (Online): 2456-7361

2.International Journal of VLSI System Design and Communication Systems. ISSN 2322-0929 Vol.04, Issue.07, July-2016, Pages:0533-0535

3.International Journal of Engineering Sciences and Research Technology Design of I2C-APB Protocol. ISSN: 2277-9655 [Devakrupa* et al., 5(11): November, 2016] Impact Factor: 4.116 IC™ Value: 3.00 CODEN: IJESS7

4.Design, integration and implementation of crypto cores in an SoC environment. Jai Gopal Pandey, Sanskriti Gupta, Abhijit Karmakar Microelectronics International ISSN: 1356-5362

5.I2C protocol and its clock stretching verification using system verilog and UVM. Lakshmi Manasa Kappaganthu, M. Prakash, Avinash Yadlapati Published 1 March 2017 Computer Science 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT).

6.Kommirisetti Bheema Raju. Int. Journal of Engineering Research and Application. ISSN: 2248-9622, Vol. 7, Issue 1, (Part -1) January 2017.

7.International Journal of Engineering Research Technology (IJERT). Vol. 2 Issue 9, September - 2013 IJERT ISSN: 2278-0181

8.IPASJ International Journal of Electronics Communication (IIJEC) Web Site: http://www.ipasj.org/IIJEC/IIJEC.htm A Publisher for Research Motivation..... Email: editoriijec@ipasj.org Volume 3, Issue 1, January 2015 ISSN 2321-5984

9.International Journal of Advanced Research in Engineering and Technology (IJARET) Volume 7, Issue 3, May-June 2016, pp. 103-113.

10.FPGA implementation of I2C protocol: A. K. Oudjida, M. L. Berrandjia, K. Tahraoui Published 1 December 2009 Computer Science 2009 16th IEEE International Conference on Electronics, Circuits and Systems - (ICECS 2009)