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

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

Mukthi S. L.[1], Dr. A. R. Aswatha[2]
[1]Department of Electrical and Electronics Engineering Jain University, Bangalore, Karnataka, India
[2]Department of Tele Communication Engineering, DSCE, Bangalore, Karnataka, India

Abstract**—** *SOC devices and circuits will consist of sub-circuits which will use different protocols, some will use serial data transmitting protocols and some will use parallel communication protocols so in this paper we are presenting how to build communication between two different block which use different protocols. To make communication between these two protocols we developed bridge which consists of APB slave and I2C master blocks. I2C or I2C stands for "Inter-integrated circuit," or "inter-IC," and is a simple, 8-bit, serial communication bus protocol that uses just two bus wires; a serial data line (SDA) and a serial clock line (SCL). I2C is integrated into many ICs and allows devices to communicate directly with each other, avoiding CPU cycles. I2C operates on a master-slave basis, and all devices on an I2C bus have a unique address. I2C is used both internally in integrated chips to communicate between areas in the chip-based circuit, as well as externally, from chip to chip. This paper presents implementation I2C protocol that interfaces peripheral devices which operates on low speed.*

Keywords**—** *ARM Processor, I2C protocol, APB peripherals, AMBA protocol.*

I. INTRODUCTION

A multiprocessor environment consists of high performance processors and also low power consuming processors to support many different applications. High performance processors are used for sophisticated and complex processing applications, for other simple applications low power consuming processors are used. Thus power is saved. However, different processors use different protocols to communicate with peripherals. This is a disadvantage as we need peripheral clones for different processors that only differ in protocol. For example if there are 20 peripherals used, totally 40 peripherals are needed to support a dual processor environment. This is very ineffective as area is increased twice. Instead of using 20*2 peripherals we can use 20 peripherals that support one common protocol and an interface that converts the other protocol to common protocol. This saves the chip area which is one of the most important requirements at present scenario.

One such case is when one processor is using AMBA protocol (ARM processor) and other is using I2C protocol. AMBA uses APB peripherals and I2C uses I2C supported peripherals to communicate with external environment. Our goal is to use only APB peripherals and use an interface that converts the I2C protocol to APB protocol so that I2C processors can use the same APB peripherals to communicate. The area in multiprocessor design which uses ARM processor for high performance and an I2C processor for low power can be reduced by good logical design. This involves designing an interface between the I2C protocol and APB protocol.

II. SPECIFICATIONS

To establish the specifications of an embedded system, requirements must be defined first. Requirements in designing an I2C to APB Communication Bridge are:

- Data is sent and received by I2C processor on 2 lines, SDA and SCL. SCL is clock for synchronization whereas SDA provides serial data. Data is sent and received on separate parallel lines, PRDATA and PWDATA respectively in APB slaves using PCLK for synchronization.
- Therefore serial data must be converted to some common parallel data and then it has to be converted into an APB format for sending data from I2C master to APB slave and vice versa.
- The clocks used SCL, PCLK has different frequencies. That must be considered.

From the above requirements specification are defined as follows:

- An I2C interface is designed to convert serial data into common parallel format.
- Then an APB interface is designed to convert the common parallel format to required APB format.
- I2C interface takes and sends back data to I2C master at SCL clock speed and sends and receives data at very high speed to APB interface
- APB interface sends and receives data at very high speed from I2C interface and sends and receives data to/from APB slave at PCLK clock rate

III. INTERFACING BETWEEN I2C AND APB PROTOCOL

Interfacing between I2C and APB Protocol consists of two major parts. They are I2C Slave and APB Master as given in figure 1. These two blocks bridges the communication I2C Master and APB Slave. I2C Slave receives the data from I2C Master in respective format and provides it to APB Slave through APB Master.

In the designed architecture the data communication happens in two stages. They are:

1. Write Operation

- Whenever I2C Master needs to send data to APB Slave it would be done via I2C Slave.
- I2C Slave will assert Data Valid and Address Valid signals to logic high.
- The designed APB Master checks for availability of its memory and initiates the APB write operation

- Four bytes of data from I2C will be stored serially at four consecutive locations of APB Memory.
- After transfer of each byte APB Master keeps a check on count whether all four memory locations are updated successfully.
- As soon as the data at Memory updated successfully APB Master transfers the same 32-bit data to APB Slave paralaly.

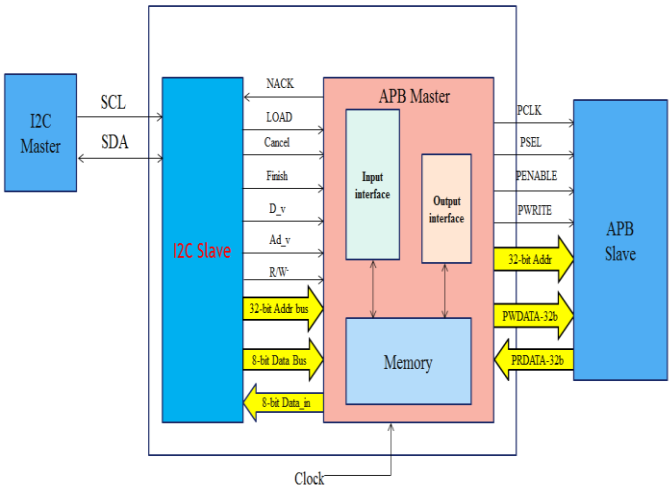

Fig. 1. Block diagram of proposed communication bridge.

2. Read Operation

- Here again when I2C need to read data from the APB Slave, communication will take place via APB Master to I2C Slave to I2C master.
- I2C Slave fetches the data which is transmitted by APB slave on to APB Master internal memory.

IV. DESIGNED INTERFACE

The designed I2C slave part as in figure 2 gets the data and address from the given I2C inputs SCL, SDA. It takes NACK as an input from the APB master side of design and sends acknowledgement signal for every 8 bits to I2C master as it is required by I2C protocol. Acknowledgement is sent whenever NACK is high and acknowledgement is not sent when NACK is low.

This part also provides control signals required by APB master architecture design. The data (8 bit) and address of register (32 bit) obtained are given as input for designed APB master for further use. APB master actually uses these data, address and control signals to send data to APB slave following APB protocol. When I2C master requires data from APB slave, designed I2C slave interface asserts the LOAD signal to start APB read operation and takes the data from the APB master interface during I2C read cycle and writes the data to I2C master via SCL and SDA according to I2C protocol.

The designed APB master consists of three blocks:

 Input interface: The input interface in APB master saves the address in a register when ad_v=1, to access it whenever APB slave is ready for data communication. It also saves the data from 8-bit data bus into a memory. Input interface is also used while reading data from APB slave, it setups the number of read operations to receive data from APB slave whenever I2C master initiates read operation. It sends NACK signal to the I2C slave whenever the APB slave is busy or memory is already full and cannot store data anymore.

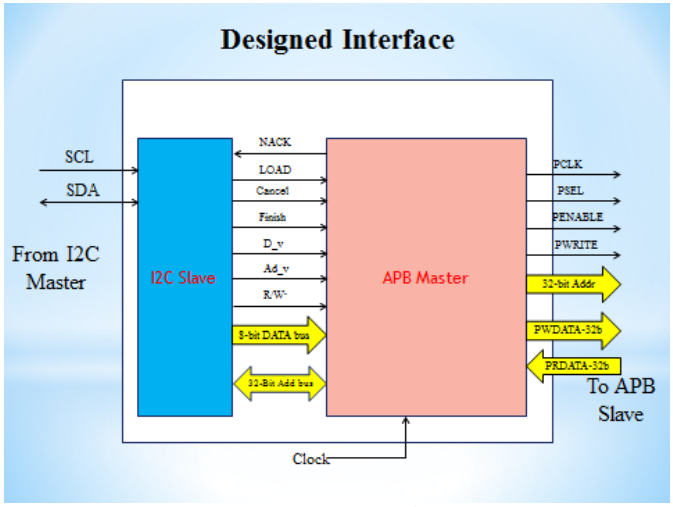

Fig. 2. Designed interface.

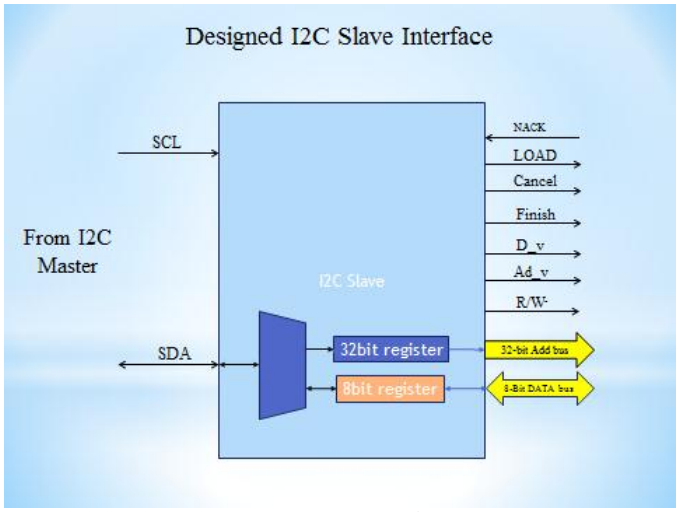

Fig. 3. I2C slave interface.

 Memory: 2 memory blocks are available one is 8-bit wide 256 block depth and another is 32-bit wide 4 block depth. Data is stored in FIFO (first in first out) manner. It is used as a buffer to store 8-bit data in blocks.

Every time it completely receives 4 blocks of 8-bit data from same I2C master it increments the count value. Count value indicates the total number of remaining blocks of 32-bit data. This value is used to know the number of write setup operation required.

Two memory blocks are designed one block is used for I2C read operation (I2C reads data from APB slave) which is 32-bit wide 4 block depth and while another is used for I2C write operation which is 8-bit wide 256 block depth.

 Output interface: This block communicates with APB slave as per APB protocol, it uses the address stored in register and

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

data stored in memory by input interface during I2C write operation and sends PCLK, PADDR, PENABLE, PWRITE, PWDATA, PREADY, PRDATA, PSLVERR etc. APB signals according to protocol requirement.

It also sends data to I2C slave during I2C read operation which is collected by input interface from APB slave. Output interface uses the count value given by memory during write operation to know the blocks of 32-bit data still to be delivered and setups the number of write operation accordingly.

Designed Interface Intermediate Signals

The I2C slave and APB master in designed interface communicates with the intermediate signals.

TABLE 1. Interface Intermediate signal description

Signals	Description
32 Bit Address bus	The register address of APB slave from which the data is read or written is sent serially via SDA from or by I2C master. This serial 32 bit address is converted into parallel data and given as output to APB master.
8 Bit Data bus	This is a bidirectional bus, as an output bus it carries 8-bit data which is to be written on APB slave and as an input bus it carries 8-bit data to be read by I2C master
R/W-Signal	Indicates the read or write operation to APB master so that it knows whether to read the data from I2C slave or write the date to I2C slave
Address Valid	ad_v, It indicates 32-bit address bus has valid address or not.
Data Valid	d_v, It indicates 8-bit data bus has valid data or not.
NACK	This signal indicates that APB slave is busy at the moment and cannot the read the data from I2C master via I2C slave; it is used by I2C slave to send NACK signal to the I2C master.
Finish	Used to indicate that current transfer from I2C is finished.
Cancel	This signal is used to indicate that I2C master cancelled the data transfer.

Flowcharts

The two flowcharts shown below for two designed sub-modules gives a summarized view of source code.

Figure 4 gives an idea about I2C slave interface part, it takes SCL, SDA, NACK and Data-in as inputs and Load, Read, Finish, Cancel, Data valid, Address valid and R/W- gives as output signals.

Figure 5 shows the dataflow of designed APB master interface, the outputs provided from designed I2C slave, PRDATA and PREADY are inputs to APB master part. Data_in and other APB slave signals are output from this designed interface.

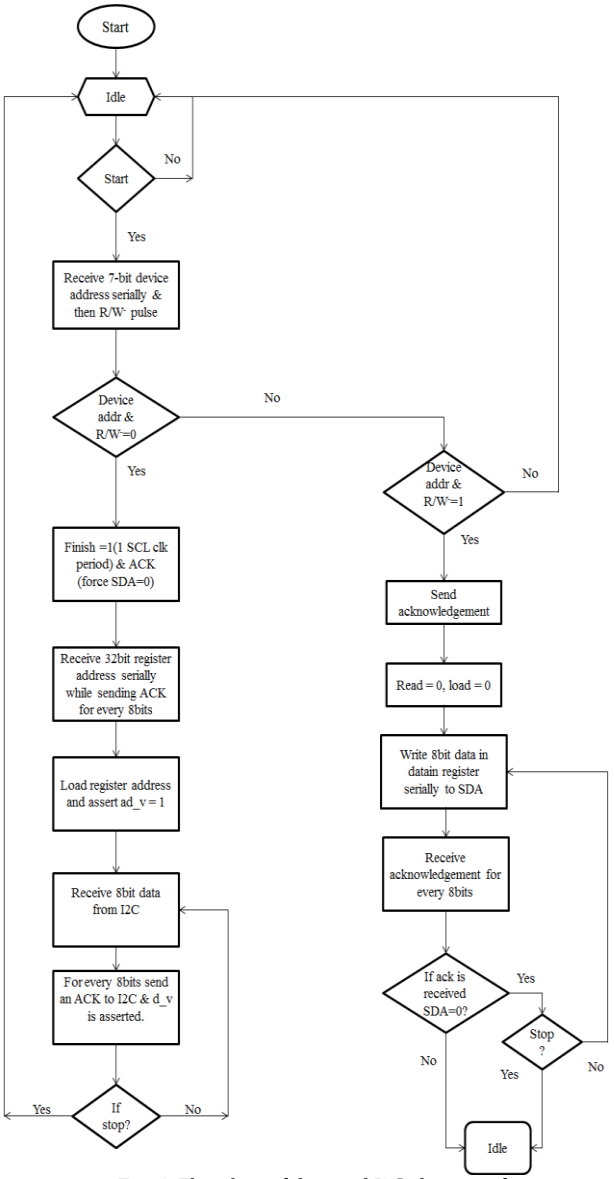

Fig. 4. Flowchart of designed I2C slave interface.

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

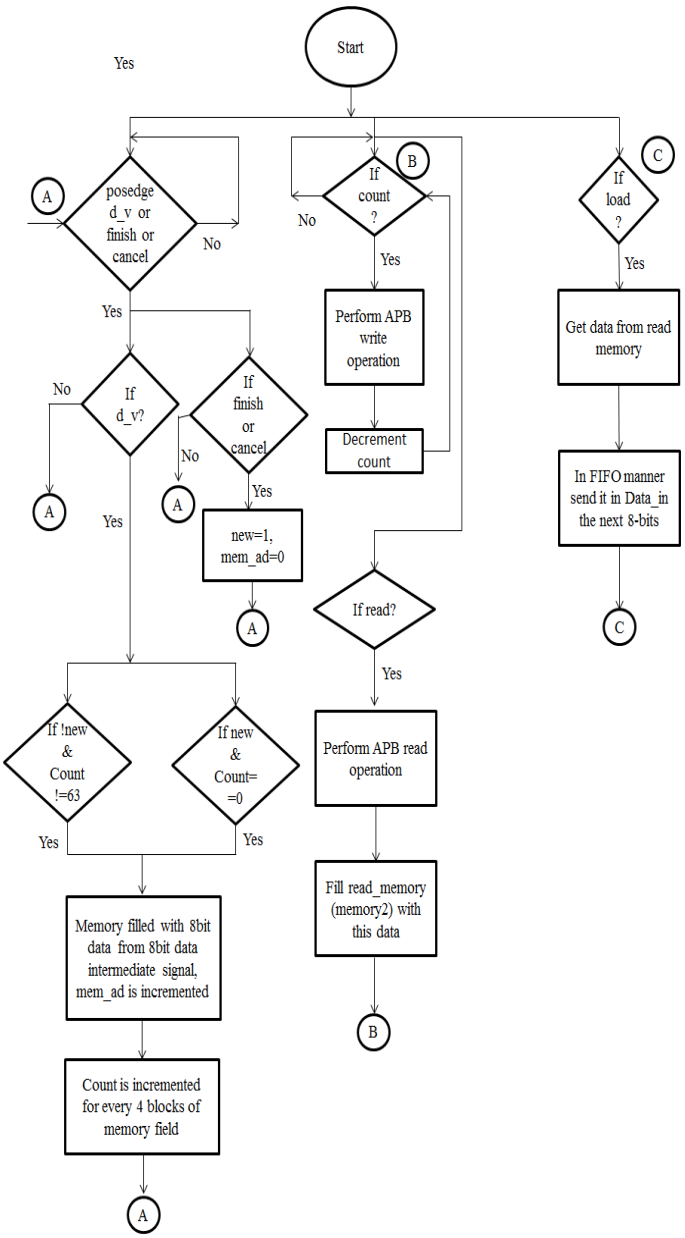

Fig. 5. Flowchart of Designed APB master interface.

V. Simulation Results

Write Operation:

After start condition The bridge receives the 8bits on SDA in sync using SCL clock. First 7 bits are I2C slave address and 8th bit is read/write bit.

- The received 7bit device address is then verified by comparing the received address with bridge's i2c slave address (I2C slave address for this is assumed to be 0000000).
- Then read/write bit is used to determine the type of operation (read or write) I2C master is looking for
- If the device address matches it first sends acknowledgement and then it performs a read operation from APB if read/write is 1 and performs a write operation if read/write is 0. Here since SDA=0 for 8 SCL clk periods the device address is matched and an I2C write operation is started. An acknowledgment is sent by the design as device address is matched.

Next, SCL remains same for this part and SDA is loaded with 32 APB slave register address bits serially for 32 SCL pulses. Between each 8 bits SDA is unforced to receive acknowledgement from the design. In the figure 7, it is loaded with 11111111110011111111111111111111.

- The design receives 32 register bits in the order required while sending acknowledgements for each 8 bits. The received register address is 11111111110011111111111111111111.
- At the end, 32 bit register address is loaded into address, an intermediate signal and address valid (ad_v) is asserted.

After 32 bit register address is loaded, The SDA is forced to provide data (minimum 32 bits) for every SCL clock period. For every 8 bits it is unforced as in previous case to receive acknowledgement from the design block.

- The data input over SDA is 11001111, 11111111, 10001111, and 11111111.
- The design received data is 11111111100011111111111111001111. Data valid is asserted and also acknowledgement is sent to I2C master for every 8 bits.
- PREADY is permanently forced to 1 for APB communication.
- The output of PWDATA is 111111111100011.
- PSEL, PENABLE, PWRITE as shown in figure are set by design at different PCLK periods for the APB write operation.

Read Operation:

APB read operation when I2C master signals read operation

- This case has same inputs as the APB write operation till loading APB register address. The loaded register address is 11111111111001111000001.
- Then SCL and SDA are forced to give a repeated start.
- After repeated start, the device address is sent as 0000000(repeated as APB write part)
- The 8th bit after device address is different from previous cases. Now the SDA is forced to give 1 to signal a APB read operation.
- At the same time PRDATA is forced to give data as input from APB
- After receiving 1st 8 bits as 00000001(device address and read/write) an acknowledgement is sent as an output on SDA as shown in figure 7.5.
- Now the data given on PRDATA is transferred serially 8 bits each time in sync with SCL.
- After every 8 bits SDA is forced to send acknowledgement to the design block.
- If ack is received APB read is continued otherwise it goes back to idle condition.
- Once stop condition is received the operation is finished and the design goes back to idle condition.

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

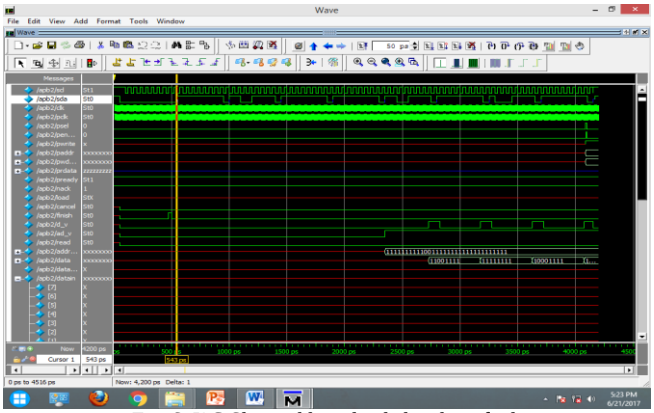

Fig. 6. I2C Slave address loaded and verified.

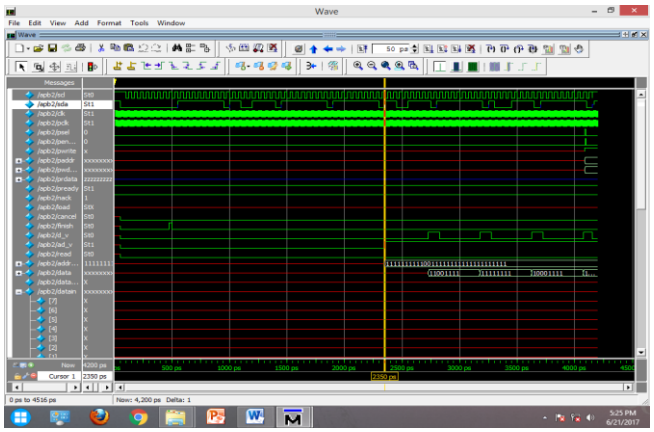

Fig. 7. 32-bit APB register address loaded.

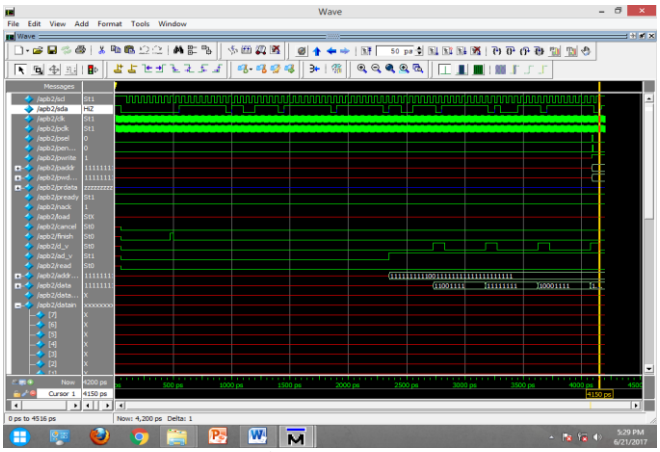

Fig. 8. 32-bit received data from I2C Master is loaded into PWDATA.

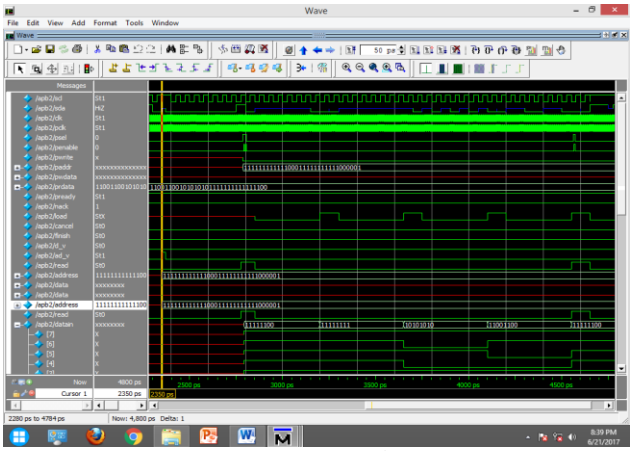

Fig. 9. I2C Master reading data from APB Slave.

VI. CONCLUSION

This paper Implementation of I2C Protocol Interface which build the bridge between I2C and APB is proposed. Implementation of the proposed design is done in ModelSim-Altera Starter Edition 6.5e, using Verilog HDL.I2C Bus protocol was designed based on NXP Semiconductors standards. Read and Write operation of I2c master is verified and simulation results are discussed clearly. Data flow from I2C master to I2C slave to APB master to APB Slave is shown in simulation results.

Future work

We designed the interface considering that an APB slave cannot initiate the read or write operation. Both read and write operations are assumed to be initiated by I2C master only. If, in future APB slaves come with write and read operation initiation capability, future work is needed to modify the interface to support this capability. This capability can also be used to increase the allowed delay for APB slave to send or receive data. Further the interface could be improved to support multiple I2C masters accessing APB slaves using different techniques like polling, token passing etc.

REFERENCES

[1] J. J. Patel and B.H. Soni, "Design and implementation of I2C bus controller using verilog," *Journal of Information, Knowledge and Research in Electronics and Communication Engineering*, ISSN: 0975 – 6779, vol. 02, issue 02, pp. 520-522.

[2] Philips Semiconductors: PCF 8584, I2C bus controller datasheet,http://www.semiconductors.philips.com/acrobat/datasheets/PCF8584_4.

[3] Samir Palnitkar, *Verilog HDL*, second Edition.

[4] J. K. Singh, M. Tiwari, and V. Sharma, "Design and implementation of I2C master controller on FPGAusing VHDL," *International Journal of Engineering and Technology (IJET)*, vol. 4, no. 4, pp. 162-166, 2012.

[5] A. K. Oudjida, M. L. Berrandjia, R. Tiar, A. Liacha, and K. Tahraoui, "FPGA implementation of I^2C& SPI protocols: A comparative study," *IEEE*, 2009.

[6] S. Devakrupa and D. Desai, "Design of I2C-APB protocol," *International Journal of Engineering Sciences & Research Technology*, vol. 5, issue 11, pp. 330-335, 2016.

[7] J. Chhikara, R. Sinha, and S. Kala, "Designing communication bridge between I2C and APB," *IEEE International Conference on Computational Intelligence & Communication Technology*, 2015.