# DESIGN AND IMPLEMENTATION OF I2C BUS PROTOCOL ON FPGA USING VERILOG FOR EEPROM

**ABHINAV BODDUPALLI**

IV$^{th}$ year, B.TECH, EEE, NIT BHOPAL
E-mail: ramaabhinav@gmail.com

**Abstract** - The I2C or Inter-Integrated Circuit protocol is a serial communication protocol designed by Philip semiconductors now termed as NXP semiconductors. This paradigm has been proliferating its use in serial communication. I2C is a bidirectional 2-wire bus designed to enhance hardware efficiency and increase simplicity of the circuit. This protocol bolsters multiple masters (which is a limitation with SPI communication) and multiple slaves and also allows communication between faster and slower devices by a serial data bus (SDA) without data loss. The other line is Serial clock line (SCL) which transfers the data according to a synchronized clock which is a limitation for UART communication. Other types of communication protocols like USB, RS-422, RS-485, CAN etc. require more pin connections and signals for communication. The postulates of I2C over UART, SPI, USB and other protocols indicates I2C's significance in its use as communication protocol. This paper makes use of Verilog language in designing and Implementing I2C bus on FPGA (XC3S100E of SPATAN-3E) which acts as master, for interfacing with EEPROM (24C02) which acts as slave. This design makes use of Xilinx 14.2 version for design and Implementation.

**Keywords** - Verilog, I2C, SDA, SCL, FPGA, Master, Slave, HDL.

## I. INTRODUCTION

I2C also termed as Inter-Integrated circuit is a multi-master bus which implies that more than one device which has capability of controlling the bus can be connected to it. As there is no data loss or affect from environment factors, transferring data on I2C bus increases the device performance. Even though there exists other types of serial communication such as SPI (Serial Peripheral Interface) and UART (Universal Asynchronous Receiver Transmitter**)**, I2C surpass these buses with its advantages such as its variable baud rate characteristic which is a limitation in UART, allowing multiple masters in communication which is a limitation in SPI, thus it is used in many applications such as PDA (Personal Digital Assistants), DVD's, ADCs, DACs, Microcontrollers, LCDs, memory devices and so on. I2C is an example of Intra Bus communication similar to SPI in which the data transfer occurs between two devices which belong to the same system, whereas USB, UART etc. belong to Inter bus communication class type in which the data transfer occurs between two devices of different systems. Also, in I2c each device connected to the bus have different addresses from each other which is used for identification of the device for communication.

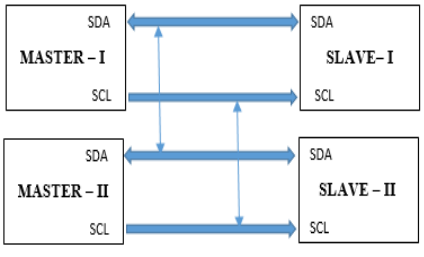


**Fig.1: Block Diagram of I2C bus with multi-master and multi-slave devices**

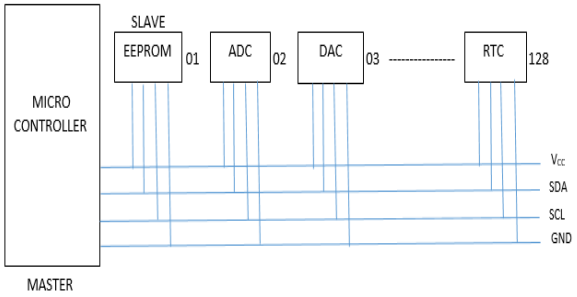


**Fig.2. I2C Bus configuration**

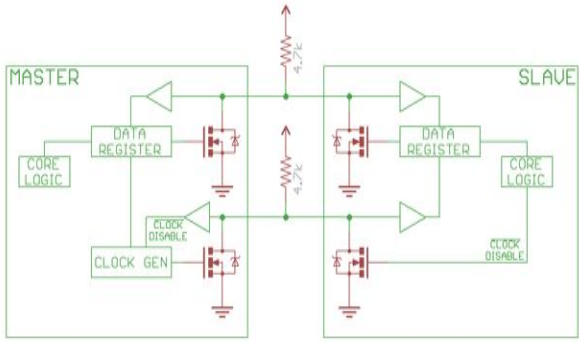


**Fig-3. I2C circuit diagram with pull up resistors each of 4.7 kΩ**

## II. FEATURES OF I2C

- I2C requires only two buses for data communication namely Serial Data Line (SDA) and Serial Clock Line (SCL). The SDA allows communication between Master and Slave devices whereas SCL synchronizes the data transfer with clock between master and slave devices. Because it is synchronized this is an advantage over UART communication in I2C.
- It is a multi-master bus consisting of collision and arbitration which prevents control of more

than one master on the bus thus avoiding data corruption.

- There are modes in which Serial, 8-bit, bidirectional data transfer can be made. One is Standard mode in which allows only 7-bit addresses and 100 kHz communications.
- The Second mode is fast mode in which address space was expanded to 10-bit and allows 400 kHz communications. Other modes that are specified are fast mode plus at 1MHz speed, high-speed mode at 3.4MHz and ultra-fast mode at 5 MHz communications.
- On-chip filtering preserves data integration by eliminating the spikes on the bus data line.
- Both master and slave can act as either Transmitter or Receiver and depending upon on this there are four modes for the bus operation which are master transmit, master receive, slave transmit and slave receive.
- The number of ICs that can be connected to same bus segment are limited by a maximum bus capacitive loading of 400pF. The variation of pull-up resistors with bus capacitance is as shown in the graph in fig.4. $V_{cc}$is the Bus-ground Voltage as shown in fig.2.

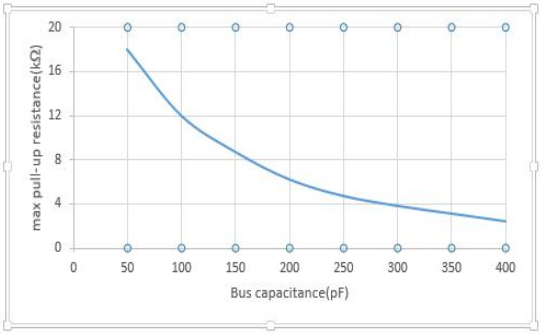


**Fig.4. Dependence of pull-up resistance on bus capacitance with $V_{cc}$ of 5V**

## III. I2C PROTOCOL WORKING

The working of the I2C protocol which is bidirectional, open-drain and pulled up by resistors depends on whether the master device reads/write data and these operations are decided by R/$\overline{W}$ bit. The working for these two conditions is as follows:

- In write operation by the master the mode of operation is master transmitter-slave receiver mode, in which master-transmitter sends START condition followed by the unique address of slave-receiver and then writes/sends data to it. Then master terminates the operation with STOP condition.
- In read operation (master receiver-slave transmitter), master-receiver sends START condition followed by unique address of slave-transmitter and further sends requested register to read to slave. Now, the master-receiver reads from slave-transmitter and terminates the data transfer with STOP condition.
- The START and STOP conditions are always generated by the master. The I2C bus begin or terminate the data transfer operation on SDA line only when SCL line is high (level triggering). That is, a START condition occurs only when there is a high-to-low transition on SDA line when SCL is high and for STOP condition to occur a low-to-high transition must happen on SDA line when SCL is high.
- The data transfer on the I2C bus can be separated into four states which are START, Address frame, Data frame and STOP conditions. After initiation of START by master, the 7-bit address starts its flow with MSB being transferred at first followed by R/$\overline{W}$ bit ($8^{th}$ bit). These 8-bits transfer is also known as the byte format. The next bit after byte format is the ACK/NACK bit which signals the transmitter that the byte was successfully received and another byte may be sent. After transfer of this address frame of 9-bits the SCL is made low (which is optional in fact) and this phenomenon is also known as "clock stretching" as this delay provides more time to slave to store the transferred byte or to get ready for another byte. After the stretch, data frame transfer is initiated in which 8-bit data is transferred along with an ACK/NACK bit ($9^{th}$ bit) that functions same as that of above bit in address frame. This whole data transfer is terminated by the master by generating STOP condition.

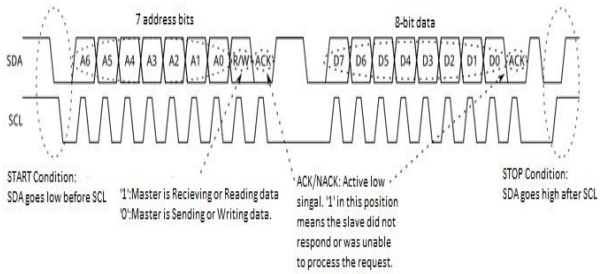


**Fig.5. I2C data transfer for a 7-bit address protocol**

## IV. FPGA AND ITS FEAURES

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. The one feature that distinguishes FPGA from ASICs (Application specified Integrated Circuits) is that FPGA can be reprogrammed to desired application or functionality requirements whereas on the other hand ASIC are manufactured for specific tasks. Due to this reprogrammable nature, FPGA is used in day-to-day applications such as ASIC prototyping, Consumer Electronics, High performance Computing and data storage, Medical, Video and Image processing, Wired and Wireless Communications and so on. The type of

FPGA used as I2C's Master is XC3S100E of family SPARTAN-3E. The features of Spartan-3E family are as follows:

- Very low cost, high-performance logic solution for high-volume, consumer-oriented applications.
- Proven advanced 90-nanometer process technology.
- Multi-voltage, multi-standard Select I/O interface pins:
- Up to 376 I/O pins or 156 differential signal pairs.
- LVCMOS, LVTTL, HSTL, and SSTL single-ended signal standards.
- 3.3V, 2.5V, 1.8V, 1.5V, and 1.2V signaling.
- 622+ Mb/s data transfer rate per I/O.
- True LVDS, RSDS, mini-LVDS, differential HSTL/SSTL differential I/O.
- Enhanced Double Data Rate (DDR) support.
- DDR SDRAM support up to 333 Mb/s.
- Abundant, flexible logic resources:
- Densities up to 33,192 logic cells, including optional shift register or distributed RAM support.
- Efficient wide multiplexers, wide logic.
- Fast look-ahead carry logic.
- Enhanced 18 x 18 multipliers with optional pipeline.
- IEEE 1149.1/1532 JTAG programming/debug port.
- Hierarchical Select RAM memory architecture:
- Up to 648 Kbits of fast block RAM.
- Up to 231 Kbits of efficient distributed RAM.
- Up to eight Digital Clock Managers (DCMs):
- Clock skew elimination (delay locked loop).
- Frequency synthesis, multiplication, division.
- High-resolution phase shifting.
- Wide frequency range (5 MHz to over 300 MHz).
- Eight global clocks plus eight additional clocks per each half of device, plus abundant low-skew routing.
- Configuration interface to industry-standard PROMs:
- Low-cost, space-saving SPI serial Flash PROM.
- •X8 or X8/X16 parallel NOR Flash PROM.
- Low-cost Xilinx Platform Flash with JTAG.
- Complete Xilinx ISE and Web PACK software.
- Micro Blaze and Pico Blaze embedded processor cores.
- Fully compliant 32-/64-bit 33 MHz PCI support (66 MHz in some devices).
- Low-cost QFP and BGA packaging options.
- Common footprints support easy density migration.
- Pb (Lead)-free packaging options
- XA Automotive version available.


**Fig.6. FPGA kit XC3S100E of SPARTAN -3E family.**

## V. EEPROM AND ITS FEAURES

Electrically erasable programmable ROM (EEPROM) is user modified ROM which can be removed and reprogrammed by raising the voltage larger than normal electrical voltage. It is a type of non-volatile memory in which data is stored in small quantities and this data must be saved when power is turned off. For a 7-bit address protocol on I2C bus, when EEPROM is acting as slave the first four address bits are constant which are "1010" and the remaining 3-address bits are programmable which allows to connect a maximum of $2^3$(8) EEPROM'S at a time. All 24CXX EEPROM families have same features and differ only in size. The features of EEPROM (24C02) are as follows:

- Enabled with bidirectional data protocol suitable for I2C protocol standards.
- Partial age writes are allowed and also has the advantage of self-timed write cycle.
- Low voltage (1.8V at 100kbps) and standard voltage (2.7V, 5V at 400kbps) operations are compatible.
- Schmitt trigger, filtered inputs for noise suppression.
- Internally Organized 128 x 8 (1K), 256 x 8 (2K), 512 x 8 (4K), 1024 x 8 (8K) or 2048 x 8 (16K).
- Wire protect pin for hardware data protection.
- Availability of automated devices.
- 8-byte page (1K, 2K), 16-byte page (4K, 8K, 16K) write modes.

## VI. READING AND WRITING FOR EEPROM (24C02)

- **WRITING SEQUENCE:**
  1. Send the START sequence and the 7-bit data address with R/$\overline{W}$ bit low (even address).
  2. Send the Internal register number in which you want to write to.
  3. Send the data byte.
  4. Optionally, send any further bytes.
  5. Send the STOP sequence.

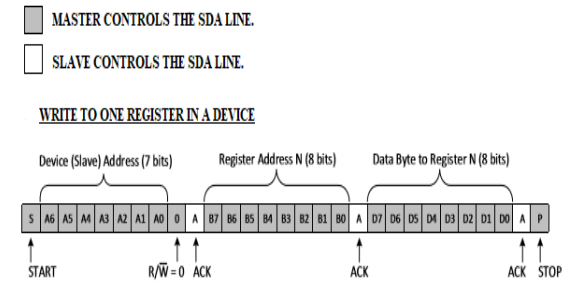


Fig.7. Writing sequence to EEPROM slave.

- **READING SEQUENCE:**
  1. Send the START sequence and the 7-bit address with R/$\overline{W}$ bit low (even address).
  2. Send the Internal address of bearing register.
  3. Send the START sequence again which is termed as "repeated START".
  4. Now again send the 7-bit address with R/$\overline{W}$ bit high (odd address).
  5. Read data byte which is written into the register by the slave.
  6. Send the STOP sequence.

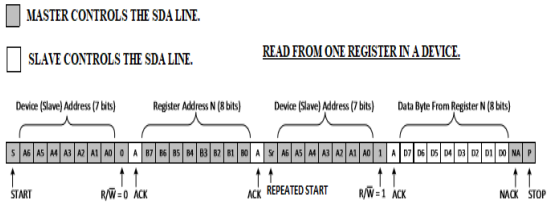


Fig.8. Reading sequence from EEPROM slave.

## VII. FSM AND DESIGN STEPS

The design of I2C protocol is made in Verilog language with the help of states in finite state machine (FSM) of the bus. This FSM represents the data transfer format explained above. The diagram for algorithm of the FSM is shown below. The steps in this algorithm are:

**STATE_IDLE:** In this state the I2C bus does not perform any operation and the bus is idle. Both the SDA and SCL lines remain high in this state. The reset bit remains high during this state and the bus moves to next state only when the reset bit becomes low shifting the bus to start state.

**STATE_START:** After reset becomes low this state is achieved and during this the FPGA master generates START condition for the initiation of data transfer shifting the bus to address frame. If reset becomes high in this state, the bus reverts back its changes and goes to STATE_IDLE. Also, in this state the count for the address frame is decided. Ae were discussing the 7-bit address protocol the count declared here is 6.

**STATE_ADDR:** This state executes the address frame state and if reset becomes high the bus reverts back to idle. The count is decremented by 1 for each of the 7-bit address and until count is zero the bus stays in this state. After annulation of the count the bus is moved to 8$^{th}$ bit state which is R/$\overline{W}$ bit. Is reset is high during this state the bus returns to STATE_IDLE.

**STATE_ RW:** After the address frame, this state decides the operation whether it is read or write depending on this bit. If this bit is high the operation is read otherwise it is write operation and this is acknowledged by the slave.

**STATE_ACK:** This is the first acknowledgment for address frame by the slave and it transfers the bus for initiation of data frame. During this state, the count for data state is decided and as the data is byte format the count is 7.

**STATE_DATA:** In this count is decremented by 1 for each bit of the data and loop continues until count is zero. After that, bus is transferred to second acknowledgement.

**STATE_ACK_2:** This is the 2$^{nd}$ acknowledgment done by the slave for the data frame state.

**STATE_STOP:** The master generates STOP condition and terminates the action of carry by the bus by making both SDA and SCL high again. If reset becomes high after this the bus shifts to STATE_IDLE. If the reset stays zero, the bus makes the slave ready for another byte transfer by shifting to STATE_START.

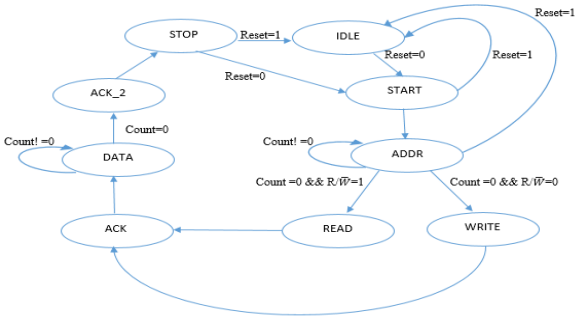


Fig.9. Example of a Finite State Machine diagram for the FPGA master.

## VIII. SIMULATION AND IMPLEMENTATION RESULTS

The Simulation and Implementation Results for Writing and Reading to EEPROM slave by FPGA master using I2C bus are as shown. The results are Simulated and Synthesized using Xilinx ISE 14.2.

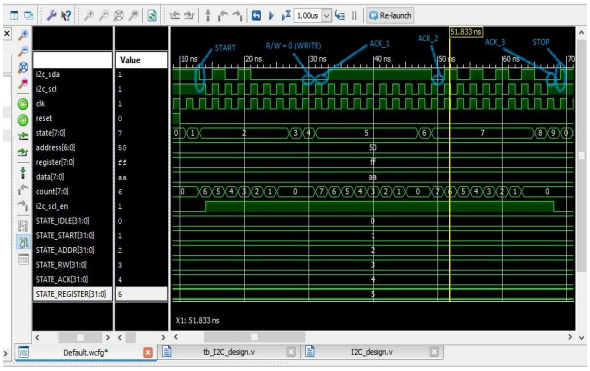

**Fig.10. Simulation results for writing data to EEPROM (24C02)**

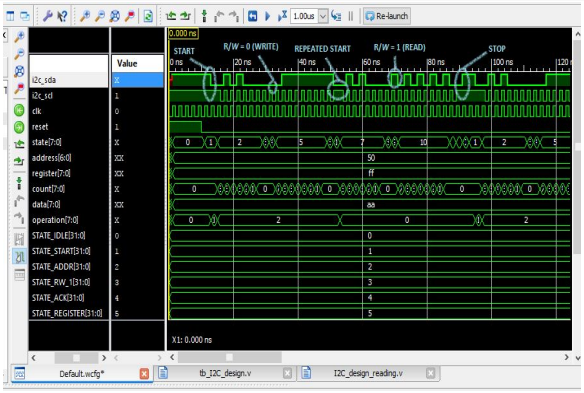

**Fig.11. Simulation results for reading data from EEPROM (24C02)**

**Fig.12. Implementation results for writing data to EEPROM (24C02)**

**Fig.13. Implementation results for reading data from EEPROM (24C02).**

## CONCLUSION AND FUTURE SCOPE

In conclusion, the Design and Implementation of I2C bus protocol with FPGA (Spartan-3E) as Master and EEPROM (24C02) as slave is done and the results are presented using Verilog HDL. The protocol is implemented in Xilinx ISE 14.2 platform and the simulations and synthesis are made. The FPGA master controls the I2C bus while writing operation and gives control of the I2C bus to EEPROM while reading operation. Because of its simplicity to implement, ability to interface fast and slow devices and other advantages over UART, SPI, etc. makes I2C the most popular communication to allow serial transfer between two devices. As the parallel operation of devices continue to increase the significance of I2C bus is enhanced. So, this project can be further extended into interfacing between multiple masters and slaves.

## REFERENCES


[1] Shiva Mehotra, Nisha Charaya., Journal on *Design and Implementation of I2C single master on FPGA using Verilog*, PISER 18, Vol.3,2006, ISSN 2347-6680(E).

[2] Philips Semiconductor "*I2C Bus Specification*", April 1995.

[3] Radha R C , RavuriAneesh Kumar, Journal on "*Design and Implementation of I2C Communication Protocol on FPGA for EEPROM*", International Journal of Scientific & Engineering Research, Volume 5, Issue 3, March-2014, ISSN 2229-5518.

[4] M.Morris Mano, "*Digital Design*" EBSCO publishing. Inc., 2002.

[5] Jonathan Valdez, Jared Becker, Application report on "*Understanding the I2C bus*", Texas Instruments, SLVA704, June 2015.

[6] Philips Semiconductors, "*The I2C Bus Specification*", version 2.1, January 2000.

[7] Philips Semiconductors, "*I2C Bus Manual*", AN10216-01, March 24 2003.

[8] I2C Tutorial "Using the I2C Bus", http:// www.robotelectronics.co.uk/acatalog/ I2C_Tutorial.html.

[9] I2C tutorials at https://learn.sparkfun.com.

[10] Deepa Kaith, Dr. Janankkumar, B. Patel, Mr. Neeraj Gupta, Journal on "*An Implementation of I2C Slave Interface using Verilog HDL*", IJMER, Vol.5, Issue.3, March 2015, ISSN: 2249-6645.

[11] O. Romain, T.Cuenin&P.Garda: "Design &modeling of an I2c Bus Controller", FDL 0'3, Frankfurt, Deutschland, Sept 23-26, 2003.

[12] Xilinx "*SPARTAN-3E FPGA family data sheet*", product specification, DS 312, July 19, 2013.

[13] Atmel Corporation. "*AT24C02 Data Sheet*" PDF document, (2007).

[14] alearningroom's "*i2c bus protocol tutorial*" https://www.youtube.com.

[15] UM10204, "*I2C-bus specification and user manual*", Rev.6, 4th April 2014.


★★★