# I²C INTERFACE CONTROLLER FOR TEMPERATURE DATA LOGGER

Mohd Shahrul Azree Bin Remly
Bachelor of Electronic Engineering (Hons.)
Faculty of Electrical Engineering
Universiti Teknologi Mara
40450 Shah Alam
Email:mohdshahrulazree@yahoo.com

***Abstract:*** **- This project covers two important parts which are I²C (Inter-Integrated Circuit) and Temperature Data Logger system. This report summarizes the design of I²C interface controller for communication between the temperature sensor and processor in the data logger system. I²C (Inter-Integrated Circuit) is commonly used in serial protocols for data transfers and its interface controller is developed in HDL Verilog, and implemented on Xilinx's Spartan FPGA Development Board. It will be integrated with temperature sensor to perform as a Temperature Data Logger.**

***Keywords*— I²C (Inter-Integrated Circuit); Data Logger; Serial Protocols, Xilinx's FPGA;**

## I. INTRODUCTION

.

A data logger is used to record data over time or in relation to location either with a built in instrument or sensor or via external instruments and sensors. Temperature data logger is one type of data logger and it is needed in scientific, medical and industrial applications [1]. The data loggers are based on a digital processor (or computer) who utilizes software to activate the system and analyze the collected data [2]. The interfacing between the sensor and computer can be in any protocols and it depends on the input and output circuit. There are several protocol drivers for a processor to communicate with any peripheral or devices such as Parallel Peripheral Interface (PPI), Two wire interface(TWI) which is also called I²C, Serial Peripheral Interface (SPI), Serial Port (SPORT) and Universal Asynchronous Receiver/Transmitter (UART). Among these protocols, I²C requires only two I/O pins while others require more pins and signals to connect devices. If the applications consider simplicity and low manufacturing cost more important than speed, then I²C is appropriate for this kind of applications [3]. I²C is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. The I²C system uses a serial data line (SDA) and a serial clock line (SCL) for data transfers.

This paper provides the design of I²C interface controller which can be used in implementing the temperature data logger using Xilinx Spartan Development Board. The I²C kit is not available on Xilinx's board, thus it is a necessary if the data logger system to be designed, requires I²C as its way of communication between the sensor and a processor

The paper describes temperature data logger system in section II, the proposed work in section III, results and discussion in section IV and finally, the conclusion in section V.

## II. TEMPERATURE DATA LOGGER

The purpose of this project is to implement the temperature data logger on Xilinx's Spartan-3E Board. The system will be implemented as shown in Figure 1 where it consists of the National Semiconductor's LM75 as the sensor circuit for the system, the FPGA as the processor or controller and Character LCD is used as a display. This configuration is chosen to minimize the hardware requirements and to overcome the constant calibration need of the analog temperature sensors. Even though the LM75 doesn't have the greatest resolution or accuracy, it is perfect for part-to-part replacement.

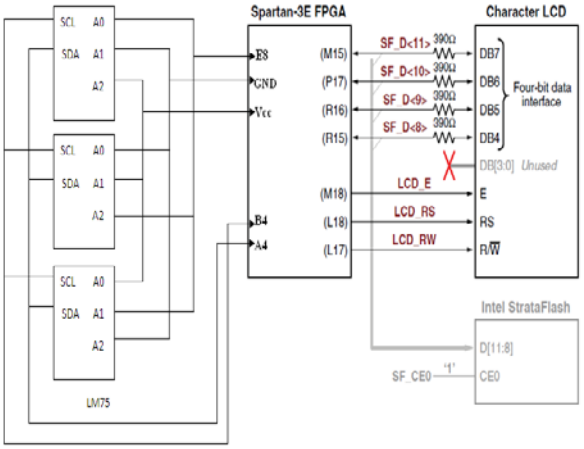


Figure 1: The Block Diagram of the Temperature Data Logger

### A. *LM75:Digital Temperature Sensor and Thermal Watchdog with I²C*

The LM75 is a temperature sensor, Delta-Sigma analog-to digital converter, and digital over-temperature detector with I²C interface [5]. The block diagram of LM75 is shown in Figure 2 and its pin description is tabulated in Figure 3.

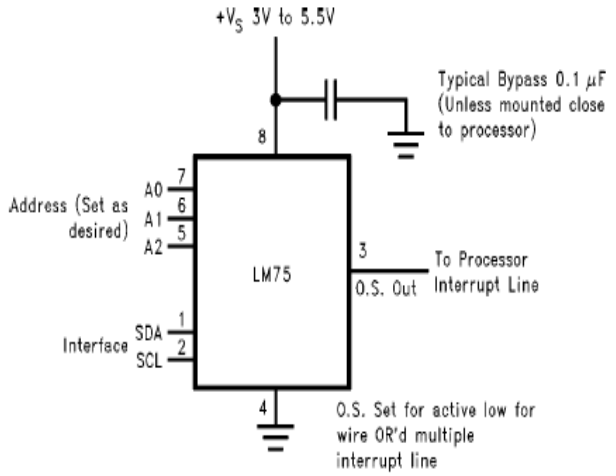


Figure 2: Typical Application of LM75

| Label | Pin | Function |
|---|---|---|
| SDA | 1 | I²C Serial Bi-Directional Data Line |
| SCL | 2 | I²C Clock Input |
| O.S | 3 | Over Temperature Shutdown Open Collector Output |
| GND | 4 | Power Supply Ground |
| +Vs | 8 | Positive Supply Voltage Input |
| A0-A2 | 7,6,5 | User Set I²C Address Input |

Figure 3: Pin Description

The LM75 sensors are connected to the I²C-bus with the SDA, SCL and GND pins and to a power supply with the +VS and GND pins. The address pins A2, A1 and A0 have to be connected to +VS or GND, setting the address of the sensor. Each sensor on the bus needs a different address for proper operation. The temperature data output of the LM75 is available at all times via the I²C bus. If a conversion is in progress, it will be stopped and restarted after the read. The LM75 operates as a slave on the I²C bus, so the SCL line is an input (no clock is generated by the LM75) and the SDA line is a bi-directional serial data path. According to I²C bus specifications, the LM75 has a 7-bit slave address. The four most significant bits of the slave address are hard wired inside the LM75 and are ``1001''. The three least significant bits of the address are assigned to pins A2-A0, and are set by connecting these pins to ground for a low, (0); or to +VS for a high, (1).[5]

### B. *Temperature Data Format*

Temperature data can be read from the Temperature, TOS Set Point, and THYST Set Point registers; and written to the TOS Set Point, and THYST Set Point registers. Temperature data is represented by a 9-bit, two's complement word with an LSB (Least Significant Bit) equal to 0.5˚C:[5]

| Temperature | Digital Output | |
|---|---|---|
| | Binary | Hex |
| +125˚C | 0 1111 1010 | 0FAh |
| +25˚C | 0 0011 0010 | 032h |
| +0.5˚C | 0 0000 0001 | 001h |
| 0˚C | 0 0000 0000 | 000h |
| -0.5˚C | 1 1111 1111 | 1FFh |
| -25˚C | 1 1100 1110 | 1CEh |
| -125˚C | 1 10010010 | 192h |

Table 1: Example of Temperature Data Format

### C. *I²C (Inter-Integrated Circuit)*

I²C uses only two bidirectional open-drain lines, Serial Data Line (SDA) and Serial Clock (SCL), pulled up with resistors. Typical voltages used are +5 V or +3.3 V although systems with other voltages are permitted. The I²C reference design has a 7-bit address space with 16 reserved addresses, so a maximum of 112 nodes can communicate on the same bus. Figure 4 shows an example of I²C configuration as a master or slaves in any interfacing circuits.

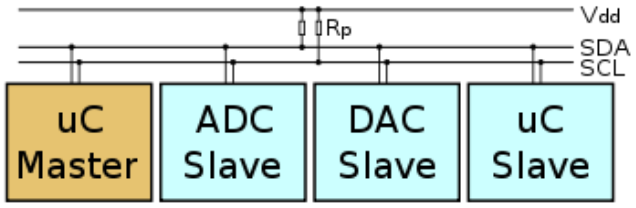


Figure 4: A sample schematic with one master (a microcontroller), three slave nodes (an ADC, a DAC, and a microcontroller), and pull-up resistors Rp.

Data transfer is initiated with the START bit (S) when SDA is pulled low while SCL stays high. Then, SDA sets the transferred bit while SCL is low (blue) and the data is sampled (received) when SCL rises (green). When the transfer is complete, a STOP bit (P) is sent by releasing the data line to allow it to be pulled up while SCL is constantly high.

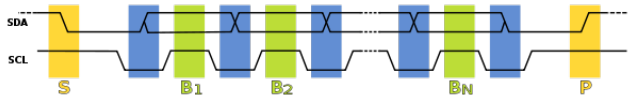


Figure 5: Timing Diagram

### D. *Xilinx FPGA*

A Field-programmable Gate Array (FPGA) is an integrated circuit designed to be configured by the customer or designer after manufacturing—hence "field-

programmable"[4]. The FPGA configuration is typically specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare). FPGAs can be used to implement any logical function that an ASIC could perform. A recent trend has been to take the coarse-grained architectural approach a step further by combining the logic blocks and interconnects of traditional FPGAs with embedded microprocessors and related peripherals to form a complete "system on a programmable chip". An alternate approach to using hard-macro processors is to make use of soft processor cores that are implemented within the FPGA logic. The relatively low cost and easiness of implementation and reprogramming of FF'GA's in comparison with the custom VLSI technology offer attractive features for the designer.

## III. I²C INTERFACE CONTROLLER

I²C Protocols can be comprehended by knowing the signals used in the interfacing technique as described below:

### *A. START signal*

When the bus is free/idle, meaning no master device is engaging the bus (both SCL and SDA lines are high), a master can initiate a transfer by sending a START signal. A START signal, usually referred to as the S-bit, is defined as a high-to-low transition of SDA while SCL is high. The START signal denotes the beginning of a new data transfer.

A repeated START is a START signal without first generating a STOP signal. The master uses this method to communicate with another slave or the same slave in a different transfer direction (e.g. from writing to a device to reading from a device) without releasing the bus. The core generates a START signal when the STA-bit in the Command Register is set and the RD or WR bits are set. Depending on the current status of the SCL line, a START or Repeated START is generated.

### *B. Slave Address Transfer*

The first byte of data transferred by the master immediately after the START signal is the slave address. This is a seven-bits calling address followed by a RW bit. The RW bit signals the slave the data transfer direction. No two slaves in the system can have the same address. Only the slave with an address that matches the one transmitted by the master will respond by returning an acknowledge bit by pulling the SDA low at the 9$^{th}$ SCL clock cycle.

### *C. Data Transfer*

Once successful slave addressing has been achieved, the data transfer can proceed on a byte-by-byte basis in the direction specified by the RW bit sent by the master. Each transferred byte is followed by an acknowledge bit on the 9th SCL clock cycle. If the slave signals a No Acknowledge, the master can generate a STOP signal to abort the data transfer or generate a Repeated START signal and start a new transfer cycle.

If the master, as the receiving device, does not acknowledge the slave, the slave releases the SDA line for the master to generate a STOP or Repeated START signal.

To write data to a slave, store the data to be transmitted in the Transmit Register and set the WR bit. To read data from a slave, set the RD bit. During a transfer the core set the

TIP flag, indicating that a Transfer is In Progress. When the transfer is done the TIP flag is reset, the IF flag set and, when enabled, an interrupt generated. The Receive Register contains valid data after the IF flag has been set. The user may issue a new write or read command when the TIP flag is reset.

### *D. STOP signal*

The master can terminate the communication by generating a STOP signal. A STOP signal, usually referred to as the P-bit, is defined as a low-to-high transition of SDA while SCL is at logical '1'.[7]

### *E. Finite State Machine (FSM)*

The controller which is the FSM is best described by its state diagram. There are two type of command controller in the system and they are known as byte or bit command controller. The FSM for these controllers are shown in Figure 6 and Figure 7, respectively.

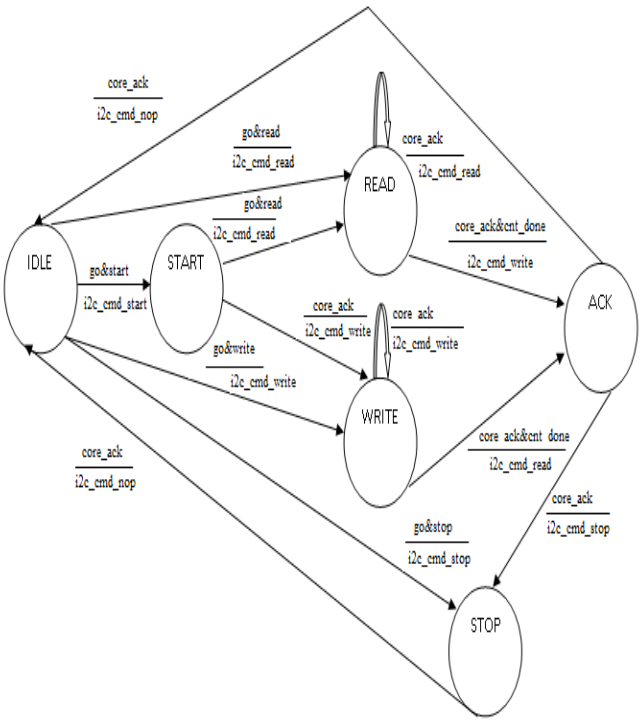


Figure 6: state diagram for byte-controller

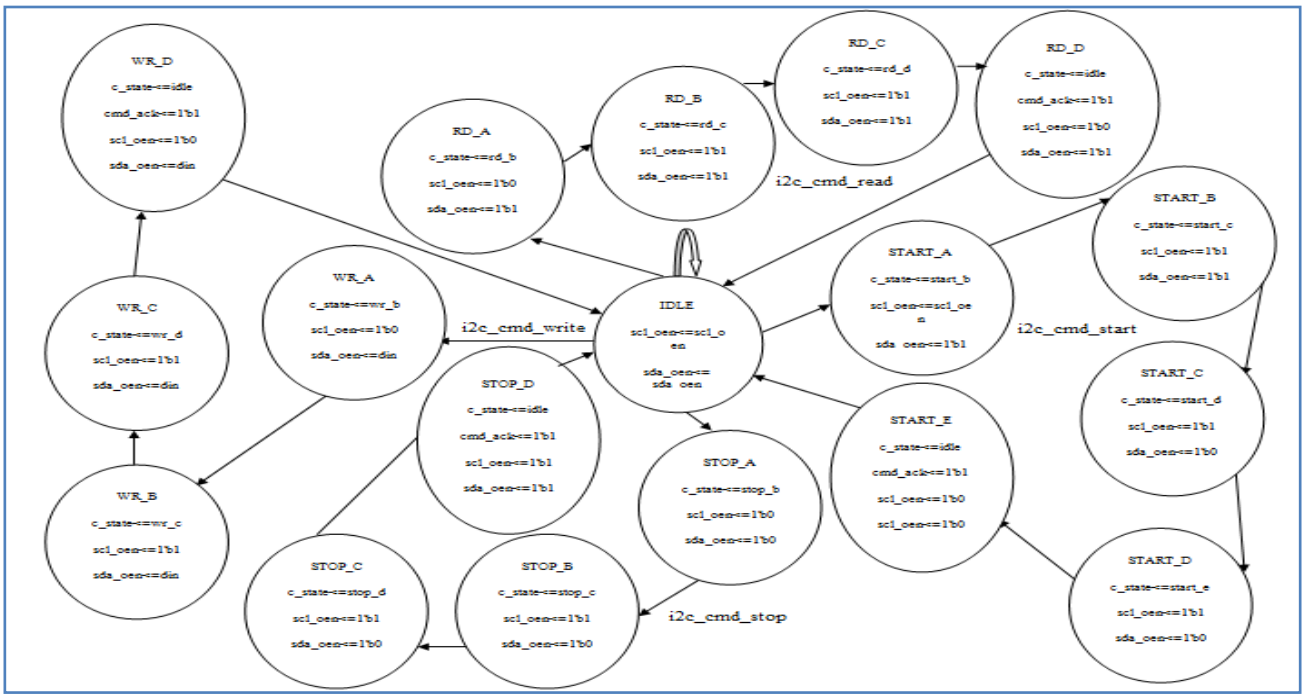


Figure 7: state diagram for bit-controller

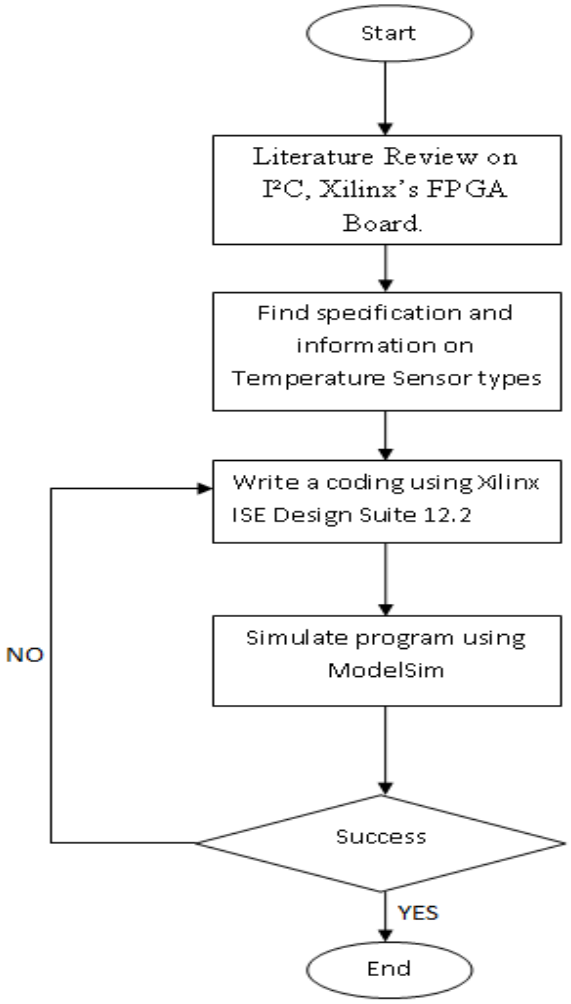


Figure 8: Flowchart of whole process of project

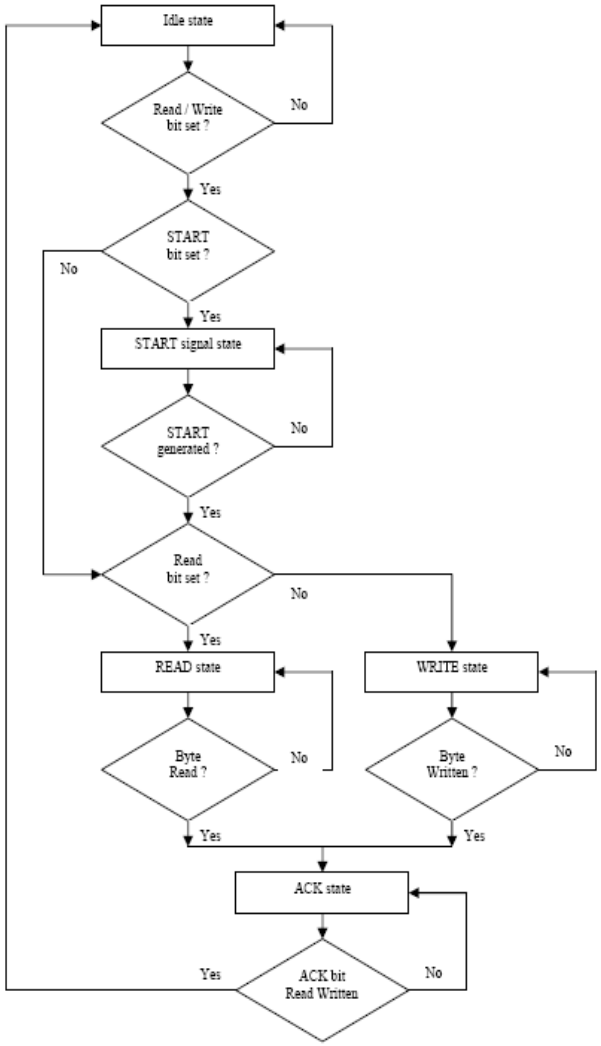


Figure 9: Flowchart for byte-controller

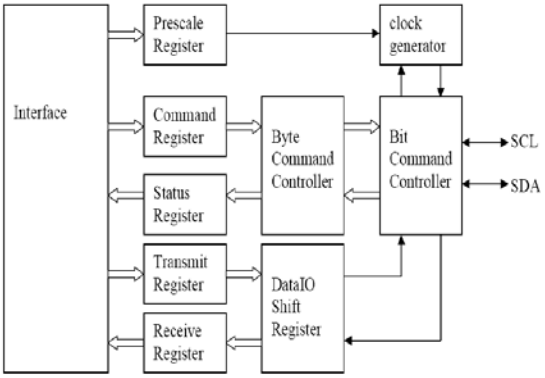


Figure 10: I²C Controller

The internal structure of the I2C controller is shown in Figure 8 where there are several registers and two command controllers.

## IV. RESULTS SIMULATION AND DISCUSSION

The FSM and the datapaths are designed using Verilog code and to check for its functionality, a test bench with fixed parameters is chosen and the parameters are circled in Figure 11 and 12.

Figure 11: Simulation of I²C Controller

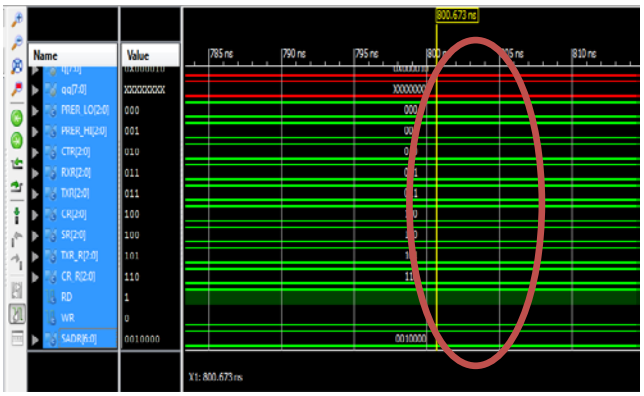

Figure 12: Simulation of I²C Controller

Figure 11 and 12 show the simulation result of $I^2C$ controller with the parameter. Adr[31:] is a address bus, used to select an internal register of the device for writing to slave or reading from slave. For example, Figure 11 shows the value of adr[31:0] is '100' which refers to the set address for Command register and Status register. If '1', it will be read from the slave and if '0', it will be written to slave. Dat_i is data received from the host processor and dat_o is a data to be sent to the host processor (Valid when inta is asserted). The value of dat_i is '000010' (Figure 11) because these values present slaves memory address. Value of '000000' and 'xxxxxx' is representing address to send the data to the host processor. For now, the controller does not have a data to send to the host processor because the controller is not interface with sensor, so that's why it displays this value at the simulation.

Write enable signal (we) are used to indicate whether the current local bus cycle is a Read or Write cycle, '0' is read and '1' is write. Stb (Strobe signal), are asserted when indicates the start of a valid data transfer cycle. The value of strobe signal is depends on value of cycle signal, valid when HIGH (1). The cycle signal (cyc) is asserted when indicates the start of a valid cycle. Valid when '1' (HIGH). Standard device acknowledgement signal (ack), when this signal goes HIGH '1', the Controller (Master Slave) has finished execution of the requested action and the current bus cycle is terminated. Interrupt Signal (inta), this line is taken High if the input enable bit in the Control register is set (CONTROL.1) and the interrupt request bit in the Status register (STATUS.0) becomes set. The latter is set by the Controller whenever it completes its current operation.

Scl is serial clock input. Scl_0 is serial clock output and Scl_oen is output enable signal for the $I^2C$ clock bidirectional buffer. Serial clock input is high '1' when STOP signal is producing. Serial clock output always low '0' for generate a START signal and polarity for output enable signal for $I^2C$ clock bidirectional buffer is always high '1'.

Serial data input (sda) is high '1' when START signal are generated. For serial data output is low '0' when STOP signal are producing and polarity for output enable signal for $I^2C$ data bidirectional buffer is always high '1'. Master clock (clk) is set always toggle and asynchronous active low reset (rstn) is reset when high '1'.

The expected outputs are tabulated in Figure 13.

| input | | Output | |
|---|---|---|---|
| dat_i[7:0] | 0x000010 | dat_o[7:0] | Xxxxxxxx |
| Scl_i | 1 | scl_o | 0 (always 0) |
| | | scl_oen | 1 (active low) |
| Sda_i | 1 | sda_o | 0 (always 0) |
| | | sda_oen | 1 (active low) |
| q[7:0] | dat_i[7:0] | qq[7:0] | dat_o[7:0] |

Figure 13: The expected outputs

dat_i: data input
dat_o: data ouput
scl_i: serial clock line input
scl_o: serial clock line output
scl_oen: serial clock line output enable
sda_i: serial data line input
sda_o: serial data line output
sda_oen: serial data line output enable
q: input data
qq: output data

The Byte Command Controller handles $I^2C$ traffic at the byte level. It takes data from the Command Register and translates it into sequences based on the transmission of a single byte. By setting the START, STOP, and READ bit in the Command Register, for example, the Byte Command Controller generates a sequence that results in the generation of a START signal, the reading of a byte from the slave device, and the generation of a STOP signal. It does this by dividing each byte operation into separate bit-operations, which are then sent to the Bit Command Controller.

The Bit Command Controller handles the actual transmission of data and the generation of the specific levels for START, Repeated START, and STOP signals by controlling the SCL and SDA lines. The Byte Command Controller tells the Bit Command Controller which operation has to be performed. For a single byte read, the Bit Command Controller receives 8 separate read commands.

The DataIO Shift Register contains the data associated with the current transfer. During a read action, data is shifted in from the SDA line. After a byte has been read the contents are copied into the Receive Register. During a write action, the Transmit Register's contents are copied into the DataIO Shift Register and are then transmitted onto the SDA line.

## V. CONCLUSION

As a conclusion, a controller for $I^2C$ interfacing method can be implemented using Xilinx's FPGA and it can be used as interfacing technique for Temperature Logger System with LM75 temperature sensor.


## ACKNOWLEDGMENT

A special thank to my supervisor Dr. Azilah Saparon and for her advice, support and guidance throughout this research.


## REFERENCES


[1] Kuchta, R.; Stefan, P.; Barton, Z.; Vrba, R.; Sveda, M.; , "Wireless temperature data logger," Sensors and the International Conference on new Techniques in Pharmaceutical and Biomedical Research, 2005 Asian Conference on , pp. 208- 212, 5-7 Sept. 2005.

[2] Petreus, D.; Juhos, Z.; Pitica, D.; "System in Package for Temperature Logging," Electronics Technology, 2006. ISSE '06. 29th International Spring Seminar on, pp.309-312, 10-14 May 2006.

[3] Godi Fisches Sangmok Lee and Michael Obara, "A Programmable Monolithic Temperature Logging Device": Department of Electrical & Computer Engineering University of Rhode Island Kingston, U.S.A, 2002.

[4] Kevin Roebuck, "FPGA Field-Programmable Gate Array: High-impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors" Published by Tebbo, 2011.

[5] LM75 Digital Temperature Sensor and Thermal Watchdog with Two-Wire Interface, National Semiconductor Corporation, June 1996

[6] Spartan-3E Starter Kit Board User Guide,XILINX, 9 March 2006.

[7] Richard Herveille, "I2C-Master Core Specification ", OpenCores, 3 July 2003.