Asian Journal of Applied Science and Technology (AJAST)
(Open Access Quarterly International Journal) Volume 2, Issue 2, Pages 172-180, April-June 2018

Interfacing of I2C Master Bus Controller with FPGA

M.Mohankumar[1], N.Birundha[2] and T.Suganya[3]

[1]Assistant Professor, [2]UG Student, [3]UG Student
Department of Electronics and Communication Engineering, Sri Eshwar College of Engineering, Coimbatore, Tamilnadu, India.

Article Received: 24 January 2018 *Article Accepted: 27 February 2018* *Article Published: 08 April 2018*

ABSTRACT

This paper focuses on the design of I2C (Inter-Integrated Circuits) single master which consists of a bidirectional data line i.e. serial data line (SDA) and serial clock line (SCL).I2C is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. It is used for faster devices to communicate with slower devices and each other without data loss. It requires only two lines for communication with two or more protocols and can control a network of device chips with just two general purpose I/O pins whereas, other bus protocols require more pins and signals to connect devices. The complete module is designed in VHDL and simulated using Xilinx ISE. The I2C Master controller's operation is analyzed to communicate with a real time clock (Maxim DS1307), the Maxim DS1307 which is connected to I2C bus as a slave. Using the I2C bus as the communication channel, the master controller will be able to send messages and receive data from the slave.

Keywords: I2C, MAXIM DS1307, FPGA, VHDL, On-chip, low-speed peripherals, Xilinx.

1. INTRODUCTION

In the field of serial data communication there are some protocols like RS-232, RS-422, RS-485, SPI (Serial peripheral interface), Micro wire for interfacing high speed and low speed peripherals. These protocols require more pin connection in the IC(Integrated Circuit) for serial data communication to take place, as the physical size of IC have decreased over the years, we require less amount of pin connection for serial data transfer. USB/SPI/Micro wire and mostly UARTS are all just 'point to point' data transfer bus systems.

They use multiplexing of the data path and forwarding of messages to service multiple devices. To overcome this problem, the I2C protocol was introduced by Phillips which requires only two lines for communication with two or more chips and can control a network of device chips with just a two general purpose I/O pins whereas, other bus protocols require more pins and signals to connect devices. In this project, we are implementing I2C bus protocol for interfacing low speed peripheral devices on FPGA. It is also the best bus for the control applications, where devices may have to be added or removed from the system. I2C protocol can also be used for communication between multiple circuit boards in equipment's with or without using a shielded cable depending on the distance and speed of data transfer.

The I2C bus is a medium for communication where master controller is used to send and receive data to and from the slave. The low speed peripheral, is interfaced with I2C master bus and synthesized on Spartan 3E. The I2C bus system with the I2C master controller is implemented on a FPGA and the real time clock device acting as the slave.

A. NOC

The regions are interconnected using a NOC. There are several reasons to base the future FPGA on NOC. The NOC is much more scalable than all other interconnect solutions, such as point-to-point wires, buses, etc. Another reason is spatial reuse, which allows for scalable power cost compared to the increased routing matrix that is used in traditional FPGA. The FPGAs are often used as prototypes for ASIC. If the ASIC is migrating to NOC, the FPGA architecture should support NOC. In our architecture, the NOC is a uniform mesh and each region is connected to a router via a local router interface. Like other NOCs, we use wormhole routing. The router structure is very similar to previous NOC designs and it includes support for multiple QOS classes, similar to QNOC. It is also provides direct communication wires between adjacent CRs not via the NOC, we choose to avoid such communication in order to simplify the design and the inter-region communication methodology.

2. EXISTING WORK

The proposed system of SPI protocol consist of the master and slave in whom we have introduced a pipelined buffer in the bus MOSI and MISO. The 1 bit address is used as an input to the system. The buffer stores the data transferred from the master .Due to which there is no data lose in between the data transfer. As the delay of the system gates reduced the speed of the data transfer or Transfer rate of the system increase. Below is the proposed design for SPI Protocol.

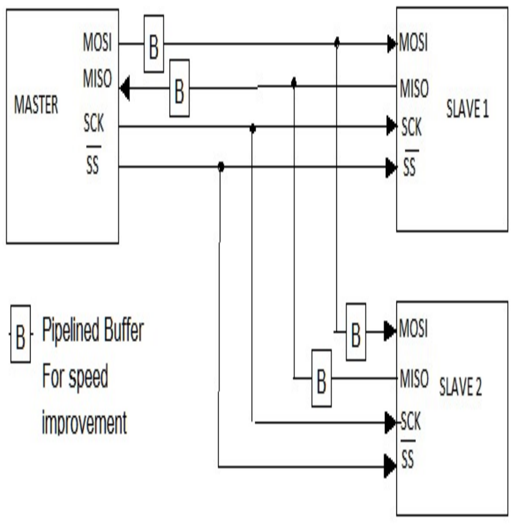


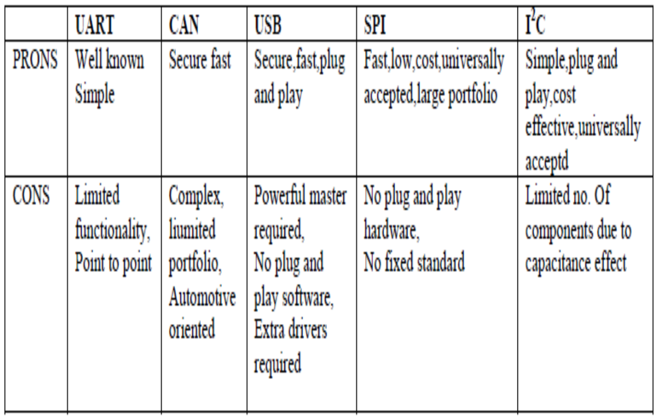
Figure 1: Proposed design for SPI Protocol

.

3. LITERATURE SURVEY

There are many reasons for using serial interface design much more important application includes serial communication like sensors communication with personal computer. Many common embedded system peripherals, such as analog-to-digital and digital-to-analog convertors, LCDs, and temperature sensors, support serial interfaces. Serial interface allow processors to communicate without the need for shared memory and the problems they can create. There are Serial communication protocols like UART, CAN, USB, SPI and Inter IC. USB, SPI and UARTS are all just one type to point type protocol. USB uses multiplexer to communicate with other devices. Only I2C and CAN protocol uses software addressing. But only I2C is very simple to design and easy to maintain [1].

The controller is connected to a microprocessor or computer and reads 8 bit instructions following I2C protocol. 32 bit register is designed to send data serially as per SPI instructions. The complete module is designed in VHDL and simulated in Xilinx. This concept is widely applicable where a microprocessor wants to communicate with SPI device [2]. The bus protocol which manages the communication between the ICs within a system and between the systems is called the Inter-IC bus or I2C bus. In the world of multiple application based product, it is mandatory to have multiple devices connected to a system, this includes peripherals following different communication protocols as well. This requirement gives rise to the need for an intermediate system which can act as a bridge between two or more devices. This is where I2C master protocol design is very useful. Today a system is connected to a number of devices and make the communication smooth and fast, I2C bus protocol is considered as one of the best serial communication protocol [3].



	UART	CAN	USB	SPI	I^2C
PRONS	Well known Simple	Secure fast	Secure,fast,plug and play	Fast,low,cost,universally accepted,large portfolio	Simple,plug and play,cost effective,universally acceptd
CONS	Limited functionality, Point to point	Complex, liumited portfolio, Automotive oriented	Powerful master required, No plug and play software, Extra drivers required	No plug and play hardware, No fixed standard	Limited no. Of components due to capacitance effect

Figure 2: Comparison of different protocols

The I2C and SPI are the most commonly used serial protocols for both inter-chip and intra-chip low/medium bandwidth data-transfers. This paper contrasts and compares physical implementation aspects of the two protocols through a number of recent Xilinx FPGA families, showing up which protocol features are responsible of substantial area overhead. This valuable information helps designers to make careful and tightly tailored architecture decisions. For a comprehensive comparative study, both protocols are implemented as general purpose IP solutions, incorporating all necessary features required by modern ASIC/SOC applications according to a recent market investigation of an important number of commercial I2C and SPI devices. The RTL code is technology independent, inducing around 25% are overhead for I2C over SPI, and almost the same delays for both designs[4]

A. I2C Protocol

The I2C is a two wire, bidirectional serial bus that provides effective data communication between two devices. I2C bus supports many devices and each device is recognized by its unique address. The data-in and address-in are the 8 bit address given as an input. Clock and reset are the input lines used to initiate the bus controller process. The R/W signal is given as an input to indicate whether master or slave acts as a transmitter in the data transmission. The physical I2C bus consists of two wires, called SCL and SDA. SCL is the clock line. It is used to synchronize all data transfers over the I2C bus. SDA is the data line. The SCL and SDA lines are connected to all devices on the I2C bus. As both SCL and SDA lines are "open drain" drivers they are pulled up using pull up resistors. The I2C bus is said to be idle when both SCL and SDA are at logic-1 level. When the master (controller) wishes to transmit data to a slave (DS1307) it begins by issuing a start sequence on the I2C bus, which is a high to low transition on the SDA line while the SCL line is high as shown below. The bus is considered to be busy after the START condition.

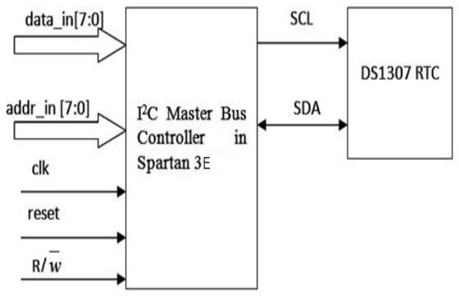
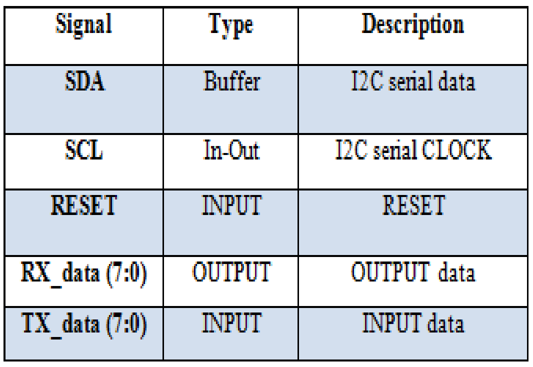


Figure 3: I/O diagram of I2C master controller interfaced with DS1307 RTC slave device.



Signal	Type	Description
SDA	Buffer	I2C serial data
SCL	In-Out	I2C serial CLOCK
RESET	INPUT	RESET
RX_data (7:0)	OUTPUT	OUTPUT data
TX_data (7:0)	INPUT	INPUT data

Figure 4: I2C description table

After the START condition, slave address is sent by the master. The slave device whose address matches the address that is being sent out by the master will respond with an acknowledgement bit on the SDA line by pulling the SDA line low. Data is transferred in sequences of 8 bits.

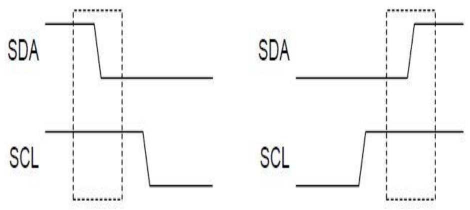


Figure 4(a) "START" (b) "STOP"
sequence sequence

The bits are placed on the SDA line starting with the MSB (Most Significant Bit). For every 8 bits transferred, the slave device receiving the data sends back an acknowledge bit, so there are actually 9 SCL clock pulses to transfer each 8 bit byte of data. If the receiving device sends back a low ACK bit, then it has received the data and is ready to accept another byte. If it sends back a high then it is indicating it cannot accept any further data and the master should terminate the transfer by sending a STOP sequence. In Fig-4(b) which shows the STOP sequence, where the SDA line is driven low while SCL line is high.

B. Serial Data Communication

In Serial Data Communication the address byte contains 7 bit DS1307 address, which is 1101000, followed by the direction bit (R/ w). After receiving and decoding the address byte the device outputs acknowledge on the SDA line. After the DS1307 acknowledges the slave address and write bit, the master transmits a register address to the DS1307 this will set the register pointer on the DS1307. The master will then begin transmitting each byte of data with the DS1307 acknowledging each byte received. The master will generate a stop condition to terminate the data write. In master receiver mode, the first byte is received and handled as in the master transmission mode. In this mode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted on SDA by the DS1307 while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial data transfer. The address byte is the first byte received after the start condition is generated by the master. The DS1307 begins to transmit data starting with the register address pointed by the register pointer. If the register pointer is not written before the initiation of a read mode, the first address is the last one stored in the register pointer. The DS1307 must receive a “not acknowledged” to end a read.

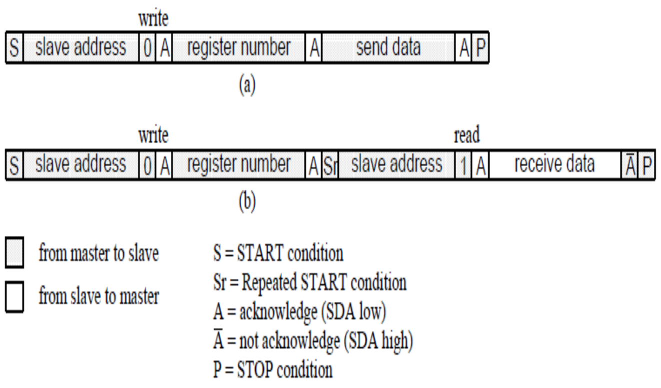


Figure 5: Master Receiver Mode

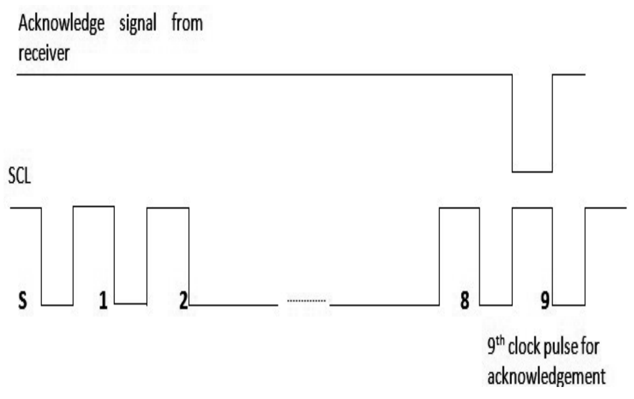

Acknowledge signal from receiver
SCL
S
1
2
8
9
9th clock pulse for acknowledgement

Figure 6: Acknowledgement on the I2C Bus

C. MAXIM DS1307

The DS1307 supports a bi-directional, 2- wire bus and data transmission protocol. The pin assignment of DS1307 is given in Fig-6. The DS1307 operates as a slave on the 2- wire bus. Fig-7 shows the interface connection of I2C bus in Spartan 3E with the DS1307 RTC chip.

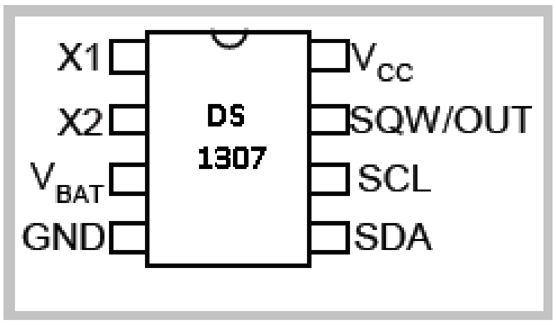


Figure 7: Pin configuration of RTC

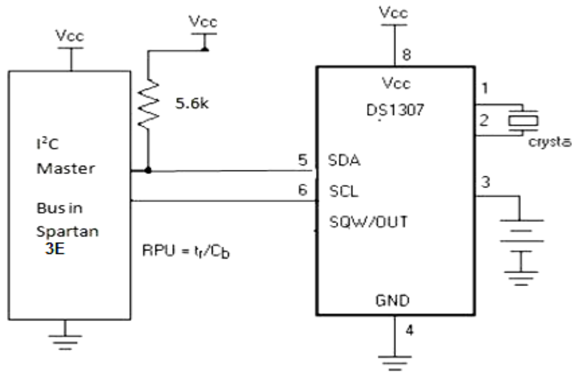


Figure 8: DS1307 connected to two wire data bus

D. Algorithm

State 1: In idle condition, an I2C bus doesn’t perform any operation (SCL and SDA remains high).

State 2: In start condition master initiates data transmission by providing START (SCL is high and SDA is from high to low).

State 3: Slave address - write: master sends the slave address write (11010000) to the slave.

State 4: If the slave address matches with the slave, it sends an acknowledgement bit in response to the master.

State 5: 8 Bit Register Address will be transmitted to the slave. Again acknowledgement is sent to the master by the slave.

State 6: Data to be transmitted is sent to the slave by the master. After receiving the data, slave acknowledges the master.

State 7: Stop condition: Slave sends a stop bit to the master to terminate the communication (SCL is high and SDA is from Low to high). For performing read operation, write operation is performed first and then read operation is done. Slave address for read is 11010001. (State 7 will not be performed for read operation)

State 8: Master transmits slave address for read operation to the slave.

State 9: Master receives the data from the slave and acknowledges the slave.

State 10: Master sends a STOP bit to terminate the connection (SCL is high and SDA is from Low to high).

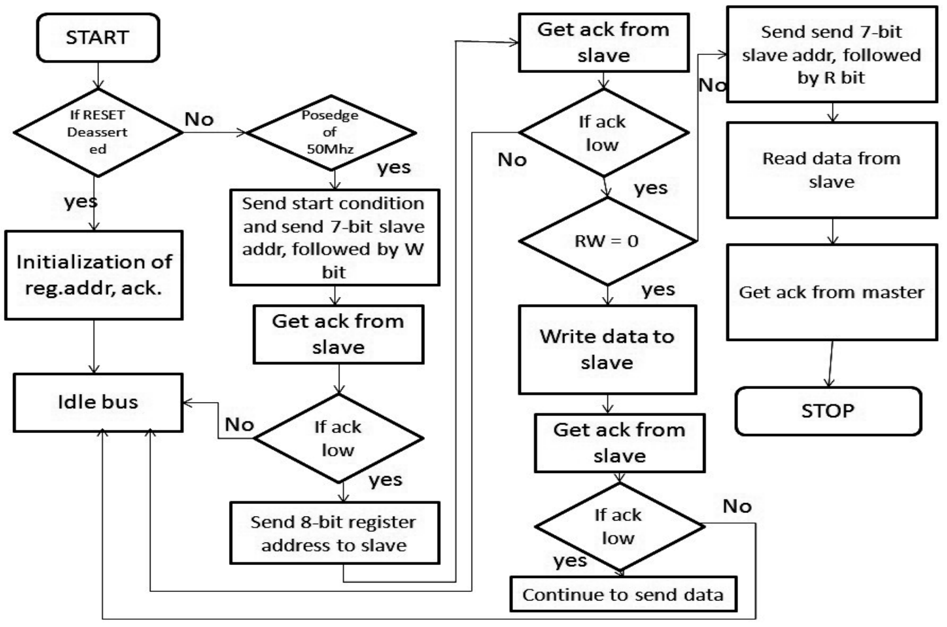
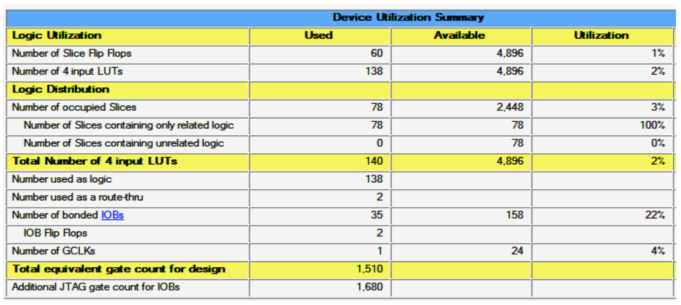


Figure 9: Flowchart for I2C master bus communication with slave device

Device Utilization Summary



Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	60	4,896	1%
Number of 4 input LUTs	138	4,896	2%
Logic Distribution			
Number of occupied Slices	78	2,448	3%
Number of Slices containing only related logic	78	78	100%
Number of Slices containing unrelated logic	0	78	0%
Total Number of 4 input LUTs	140	4,896	2%
Number used as logic	138		
Number used as a route-thru	2		
Number of bonded IOBs	35	158	22%
IOB Flip Flops	2		
Number of GCLKs	1	24	4%
Total equivalent gate count for design	1,510		
Additional JTAG gate count for IOBs	1,680		

4. RESULT

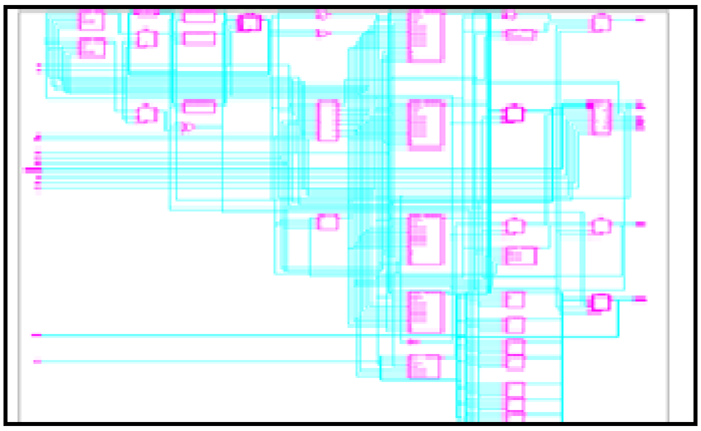


Figure 10: RTL schematic diagram

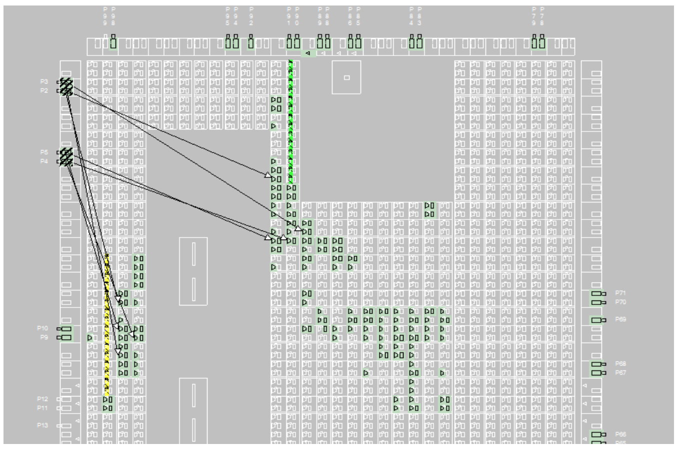


Figure 11: Floor planner

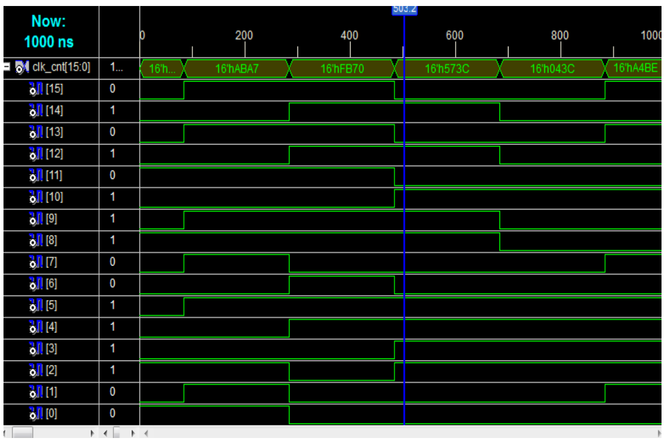


Figure 12: Output

5. CONCLUSION

The result shows that minimal resources are utilized in Spartan 3E. The design process is simplified using VHDL to design the I2C bus protocol, I2C Master transmits and receives data to and from the DS1307 (RTC). So that low speed peripheral devices can be interfaced using I2C bus protocol as master. In future, this can be implemented in networks and embedded system works in single chip.

REFERENCES

[1]. A. S. Tadkal, P. Patil - "Design of serial data communication using dual master and slave I2C bus controller", vol.2 , April – 2014

[2]. Frank Vahid, "Hardware Description Language," in Digital Design with RTL Design, Verilog and VHDL, 2nd Ed. Katie Singleton Ed. Hoboken, New Jersey: VP and Executive publisher, 2010

[3]. Prof. Jai Karan Singh et al - "Design and Implementation of I2C master controller on FPGA using VHDL", Aug-Sep 2012

[4]. Sangepu Baby Dhivya – "implementation of i2c single master multiple slave bus controller on FPGA", vol. 3, January – 2015

[5] T. Thamaraimanalan, Dr. P. Sampath, "An Energy Efficient power optimized 32 bit BCD adder using power gating and multi channel technique Vol 10, International Journal of Applied Engineering Research, 2015.

[6] K Kavitha, T Thamarai Manalan, M Suresh Kumar, "An Optimized Heal Algorithm for Hole Detection and Healing in Wireless Sensor Networks", International Journal of Advanced Engineering Research and Technology (IJAERT), Vol 2, Issue 7, Page: 243-249.

[7] R Santhiya, Mr T Thamaraimanalan, "Power gating based low power 32bit BCD Adder using DVT", Int. J. Sci. Res. Dev, Vol 3, Page: 802-805.

[8] S. Yasotha, V. Gopalakrishnan &M. Mohankumar," Multi-sink Optimal Repositioning for Energy and Power Optimization in Wireless Sensor Networks" in Wireless Personal Communications, Volume 82, Number 3, June(1),2015.

[9] M. Mohankumar, V. Gopalakrishnan and S.Yasotha, "A Vlsi Approach For Distortion Correction In Surveillance Camera Images," in ARPN Journal of Engineering and Applied Sciences, VOL. 10, NO. 9, MAY 2015 ISSN 1819-6608.

[10] M.Mohankumar, R.Gowrimanohari, "A Novel Design Of Current Mode Multiplier/Divider Circuits For Analog Signal Processing," in International Journal of Computer Science and Mobile Computing, Vol. 3, Issue. 10, October 2014, pg.918 – 925.

[11] M.Mohankumar, R.Gowrimanohari, "VLSI Architecture For Barrel Distortion Correction In Surveillance Camera Images," In Journal of Electronics And Computer Sciecne, Vol 2, Issue 5, May 2015 Paper 9.

[12] V.Pavithra, M.Mohankumar,"Secure Network Sharing Nemo based Ad-Hoc," In IJCSMC, Vol 3, Issue.2,Februray 2014, pg.645-652.

[13] K.Kokulavani, M.Mohankumar," High Speed And Lower Hardware Complexity VLSI Architecture For Lifting Based Discrete Wavelet Transform," In IJCSMC, Vol. 3, Issue. 3, March 2014, pg.733 – 739.

[14] M.Mohankumar, T.Thamaraimanalan, N.Sanjeev M,"Distortion Correction Scheme for Multiresolution Camera Images." in Asian Journal of Applied Science and Technology (AJAST).