International Journal of VLSI System Design and Communication Systems

ISSN 2322-0929
Vol.02,Issue.02,
February-2014,
Pages:0141-0146

ww.semargroup.org
www.ijvdcs.org

Communication between Two FPGA Systems using RTOS and Microblaze

MANDADA VINUSH[1], VIDATALA RAGHAVENDRA[2]

[1]Assoc Prof, Dept of ECE, DRK Institute of Science & Technology, Hyderabad, A.P, India, E-mail: vinu.mandava@gmail.com.
[2]PG Scholar, Dept of ECE, DRK Institute of Science & Technology, Hyderabad, A.P, India, E-mail: vraghavendray@gmail.com.

Abstract: Reconfigurable system like FPGA platform has the potential to provide the performance benefits of ASICs and the flexibility of processors. FPGA based embedded system have become a platform for the implementation of cryptographic algorithms. In this project we are going to implement cryptographic algorithm utilizing threads run by an RTOS [Real time operating systems] on FPGA systems. Since RTOS is an efficient tool to optimize the software runtime as the code complexity grows, by distributing the tasks into multiple threads. As an RTOS we have chosen Xilkernel and SEA [Scalable Encryption Algorithm] for implementing Cryptographic algorithms. Our project mainly focus on the major issue that two threads running separately on each board can communicate with each other via RS232 communication link. The system that is used for establishing the serial communication between the multiple FPGA systems is UART (universal Asynchronous Receiver Transmitter).

Keywords: RTOS, XTEA, RS232 Communication Link, Chip Scope Tool.

I. INTRODUCTION

Reconfigurable System like FPGA platform has the potential to provide the performance benefits of ASICs and the flexibility of processors. An FPGA is a collection of programmable gates embedded in a flexible interconnect network that can contain hard or soft microprocessors. FPGAs combine the programmability of processors with the performance of custom hardware. As FPGAs provide a useful balance between performance and flexibility, they become the primary source of computation in many critical embedded systems. FPGA based embedded system have become a platform for the implementation of cryptographic algorithms, which needs large number of bit-level operations, and that can be done efficiently on FPGA. The merit of paper lies in the implementation of cryptographic algorithm utilizing threads, run by an RTOS on FPGA systems, which is quite challenging in this reconfigurable architecture domain.

The usage of RTOS is advantageous in many respects, as RTOS is an efficient tool to optimize the software runtime as the code complexity grows, by distributing the task into multiple threads. Better and safer synchronization and resource establishing reliable communication between multiple FPGA devices is an essential component for developing complex real time systems used for manage-ment are also major advantages of an RTOS. As an RTOS, we have chosen Xilkernel and XTEA as our cryptographic algorithm to be implemented, which is quite popular.

- Applications like real time data acquisition and processing [1].
- The paper proposes the major issue that two threads running separately on each board can communicate with each other via RS232 communication link.
- The algorithm is implemented in hardware by introducing the concept of thread execution by an RTOS and its hardware utilization proves that our implementation is efficient.

A. Cryptography

Cryptography is the practice and study of techniques for secure communication in the presence of third parties (called adversaries).More generally, it is about constructing and analyzing protocols that overcome the influence of adversaries and which are related to various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation. Modern cryptography intersects the disciplines of mathematics, computer science, and electrical engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce.

B. Terminology

Until modern times cryptography referred almost exclusively to encryption, which is the process of converting ordinary information (called plaintext) into unintelligible text (called ciphertext). Decryption is the reverse, in other words, moving from the unintelligible ciphertext back to plaintext. A cipher (or cypher) is a pair of algorithms that create the encryption and the reversing decryption. The detailed operation of a cipher is controlled both by the algorithm and in each instance by a "key". This is a secret (ideally known only to the communicants)

usually a short string of characters, which is needed to decrypt the ciphertext.

C. Symmetric-key cryptography

Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key (or, less commonly, in which their keys are different, but related in an easily computable way). This was the only kind of encryption publicly known until June 1976. Symmetric key ciphers are implemented as either block ciphers or stream ciphers. A block cipher enciphers input in blocks of plaintext as opposed to individual characters, the input form used by a stream cipher. The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are block cipher designs which have been designated cryptography standards by the US government (though DES's designation was finally withdrawn after the AES was adopted). Despite its deprecation as an official standard, DES (especially its still-approved and much more secure triple-DES variant) remains quite popular; it is used across a wide range of applications, from ATM encryption to e-mail privacy and secure remote access. Many other block ciphers have been designed and released, with considerable variation in quality. Many have been thoroughly broken, such as FEAL.

D. Public-key cryptography

Symmetric-key cryptosystems use the same key for encryption and decryption of a message, though a message or group of messages may have a different key than others. A significant disadvantage of symmetric ciphers is the key management necessary to use them securely. Each distinct pair of communicating parties must, ideally, share a different key, and perhaps each ciphertext exchanged as well. The number of keys required increases as the square of the number of network members, which very quickly requires complex key management schemes to keep them all straight and secret. The difficulty of securely establishing a secret key between two communicating parties, when a secure channel does not already exist between them, also presents a chicken-and-egg problem which is a considerable practical obstacle for cryptography users in the real world.

E. Microblaze Overview

In terms of its instruction-set architecture, MicroBlaze is very similar to the RISC-based DLX architecture described in a popular computer architecture book by Patterson and Hennessy. With few exceptions, the MicroBlaze can issue a new instruction every cycle, maintaining single-cycle throughput under most circumstances. The MicroBlaze has a versatile interconnect system to support a variety of embedded applications. MicroBlaze's primary I/O bus, the Core Connect PLB bus, is a traditional system-memory mapped transaction bus with master/slave capability. A newer version of the MicroBlaze, supported in both Spartan-6 and Virtex-6 implementations, as well as the 7-Series, supports the AXI specification. The majority of vendor-supplied and third-party IP interface to PLB directly (or through a PLB to OPB bus bridge). For access to local-memory (FPGA BRAM), MicroBlaze uses a dedicated LMB bus, which reduces loading on the other buses. User-defined coprocessors are supported through a dedicated FIFO-style connection called FSL (Fast Simplex Link). The coprocessor(s) interface can accelerate computationally intensive algorithms by offloading parts or the entirety of the computation to a user-designed hardware module.

II. DESIGN IMPLEMTATION

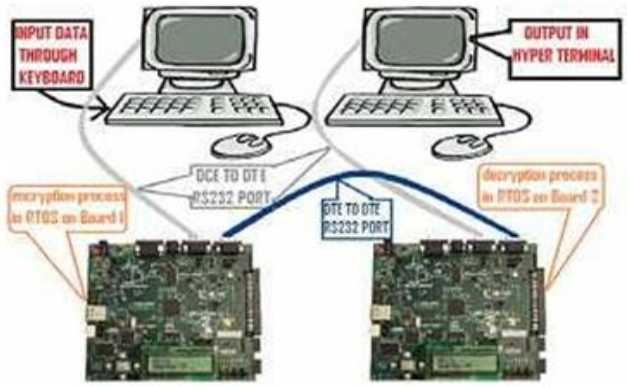

Fig1.Communication between two FPGA's using RTOS

The main theme of the project is to implement Real time Communication between two FPGA's for cryptographic applications. Data from FPGA 1 is not directly transmitted to the FPGA-2 rather than that we are applying cryptographic mechanism. According to cryptography, the input data is being transmitted from the transmitter to receiver by converting the text into a specific format which cannot be read by the unauthorized (third person) person. This text is called as cipher text. The process of transforming the plain text into cipher text is called as encryption. This cipher text is transmitted to the receiver end where the cipher text is again retransformed back to plain text. This process is called as Decryption. In this paper we are implementing this cryptography between two configurable devices (FPGA's) for effective real time communication. Presently FPGAs are considered as a major platform for high performance embedded applications as it provides the opportunity for reconfiguration as well as good clock speed and design resources. In present day application scenarios most embedded systems have real-time requirements that demand the use of Real-time operating systems (RTOS), which creates a suitable environment for real time applications to be designed and expanded easily. In an RTOS the design process is simplified by splitting the application code into separate task and then the scheduler executes them according to a specific schedule, meeting the real-time deadline.

The merit of this paper lies in the implementation of cryptographic algorithm utilizing threads, run by an RTOS on FPGA systems, which is quite challenging in this reconfigurable architecture domain. The usage of RTOS is advantageous in many respects, as RTOS is an efficient tool to optimize the software runtime as the code complexity grows, by distributing the tasks into multiple threads. Better and safer synchronization and resource management are also major advantages of an RTOS. As an RTOS, we have chosen Xilkernel. This work is implemented using the Xilinx EDK 13.2 (version) and Xilinx Spartan 3E FPGA prototyping board has been used for the hardware implementation and testing. A soft core 32-bit RISC processor Micro Blaze has been used as a CPU for this embedded computing unit and all the required soft core peripherals are UART 1 (used for RS232 DCE (Data Circuit-Terminal Equipment) port), UART 2(used for RS232 DTE (Data Terminal Equipment) port). The blocks used to build up the FPGA based embedded computing unit is shown in Fig. 2. A soft microprocessor (also called softcore microprocessor or a soft processor) is a microprocessor core that can be wholly implemented using logic synthesis. It can be implemented via different semiconductor devices containing programmable logic (e.g., ASIC, FPGA, CPLD), including both high-end and commodity variations.

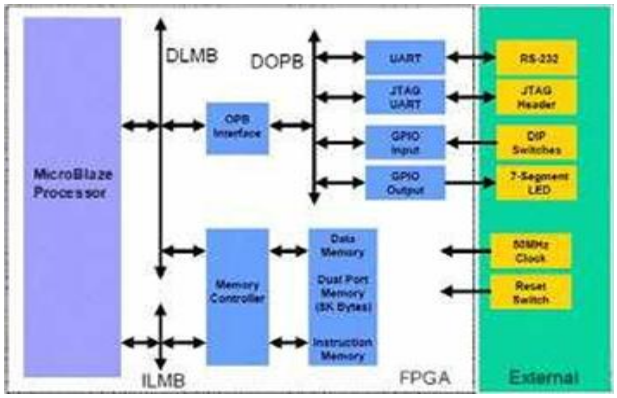

Fig2. Block Diagram Architecture of FPGA system.

The MicroBlaze is a soft processor core designed for Xilinx FPGAs from Xilinx. As a soft-core processor, MicroBlaze is implemented entirely in the general-purpose memory and logic fabric of Xilinx FPGAs. The MicroBlaze has a versatile interconnect system to support a variety of embedded applications. MicroBlaze's primary I/O bus, the CoreConnect PLB bus, is a traditional system-memory mapped transaction bus with master/slave capability. A newer version of the MicroBlaze, supported in both Spartan-6 and Virtex-6 implementations, as well as the 7-Series, supports the AXI specification. The majority of vendor-supplied and third-party IP interface to PLB directly (or through an PLB to OPB bus bridge.) For access to local-memory (FPGA BRAM), MicroBlaze uses a dedicated LMB bus, which reduces loading on the other buses. User-defined coprocessors are supported through a dedicated FIFO-style connection called FSL (Fast Simplex Link). The coprocessor(s) interface can accelerate computationally intensive algorithms by offloading parts or the entirety of the computation to a user-designed hardware module.

III. MODULES INVOLVED

A. UART:

UART is a type of "asynchronous receiver /transmitter", a piece of computer hardware that translates data between parallel and serial interfaces.

- UARTs are commonly used in conjunction with other communication standards such as EIA RS-232.
- A UART is usually an individual (or part of an) integrated circuit used for serial communications over a computer or peripheral device serial port.

B. RS-232 Interface

The RS-232 interface is the Electronic Industries Association (EIA) standard for the interchange of serial binary data between two devices. It was initially developed by the EIA to standardize the connection of computers with telephone line modems. The standard allows as many as 20 signals to be defined, but gives complete freedom to the user. Three wires are sufficient: send data, receive data, and signal ground. The remaining lines can be hardwired on or off permanently. The signal transmission is bipolar, requiring two voltages, from 5 to 25 volts, of opposite polarity. Communication Standards: The industry custom is to use an asynchronous word consisting of: a start bit, seven or eight data bits, an optional parity bit and one or two stop bits. The baud rate at which the word sent is device-dependent. The baud rate is usually 150 times an integer power of 2, ranging from 0 to 7 (150, 300, 600,...., 19,200). Below 150 baud, many system-unique rates are used. The standard RS 232-C connector has 25 pins, 21 pins which are used in the complete standard. Many of the modem signals are not needed when a computer terminal is connected directly to a computer, and Figure 1 illustrates how some of the "spare" pins should be linked if not needed.

C. MAX 232

When communicating with various microprocessors one needs to convert the RS232 levels down to lower levels, typically 3.3 or 5.0 volts. A serial RS-232 communication works with voltages -15v to -15v for high and low. On the other hand TTL logic operates between 0v and +5v. Modern low power consumption logic operates in the range of 0v and +3.3v or even lower. Thus the RS-232 signal levels are too far too high TTL electronics, and the negative RS-232 voltage for high can't be handeld at all by computer logic. To receive serial data from an Rs-232 interface the voltage has to be reduced also the low and

high voltage level has to be inverted. This level converter uses a Max 232 and five capacitors. The max232 is quite cheap(less than 5 dollars).

D. Data circuit-terminating equipment

Data circuit-terminating equipment (DCE) is a device that sits between the data terminal equipment (DTE) and a data transmission circuit. It is also called data communication equipment and data carrier equipment. Usually, the DTE device is the terminal (or computer), and the DCE is a modem. In a data station, the DCE performs functions such as signal conversion, coding, and line clocking and may be a part of the DTE or intermediate equipment. Interfacing equipment may be required to couple the data terminal equipment (DTE) into a transmission circuit or channel and from a transmission circuit or channel into the DTE.

IV. HARDWARE RESULTS AND DISCUSSION

For this I attached one results document to u with this file so insert those pictures here the above figures 3 to 6 show the hardware configuration for communication between two fpgas. In this communication we require two pc's two fpgas and three rs-232 cables. RS232 (DTE to DCE cable) is used for communication between fpga and pc. RS232 (DTE to DTE cable) is used for communication between two fpga's . It is used to transfer data from one fpga to another fpga. The two different cables and its connections with fpga are shown in the above figures.

V. SIMULATION RESULTS

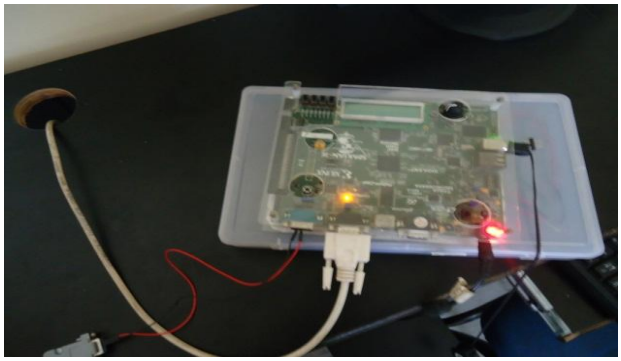

Fig3. Transmitter side FPGA

Fig4. Receiver side FPGA.

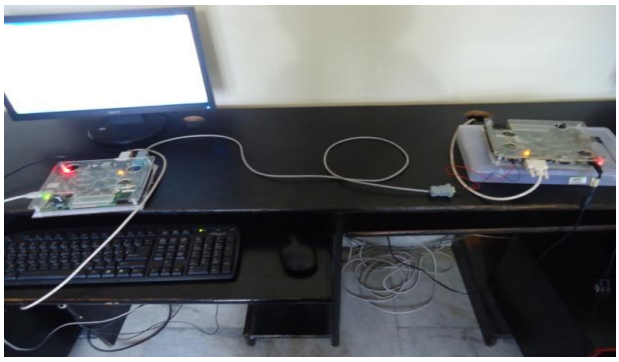

Fig5. Final Result.

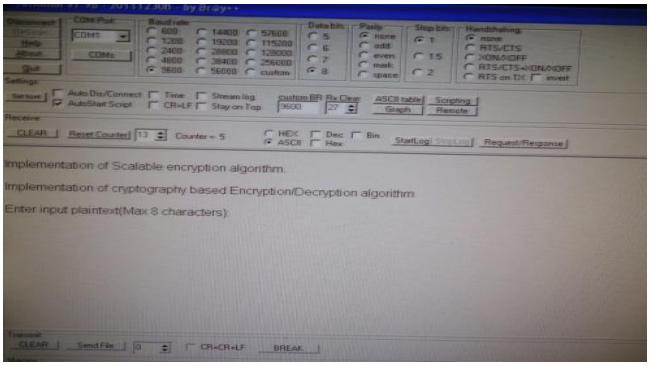

Fig6. Transmitter side terminal is ready to enter message.

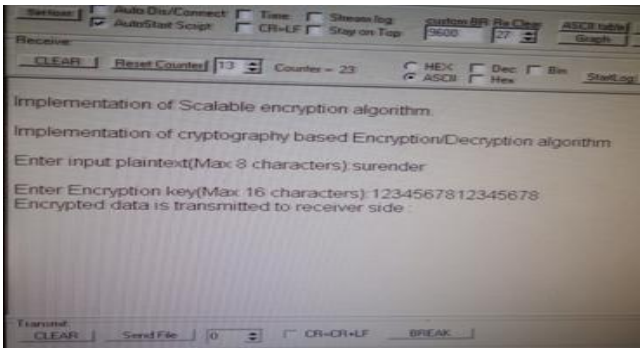

Fig7. Transmitter side terminal with message and key

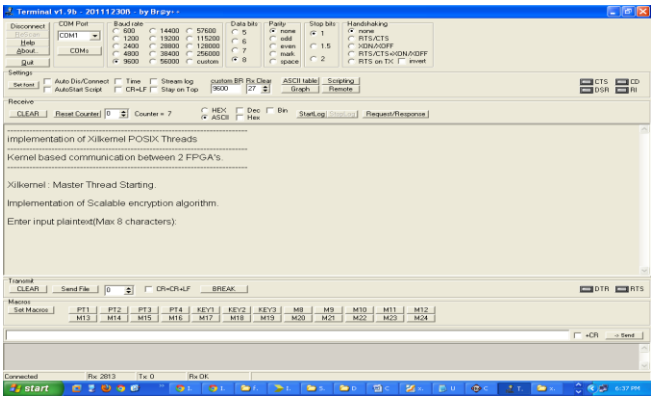

Fig8. Transmitter side terminal is ready to enter message

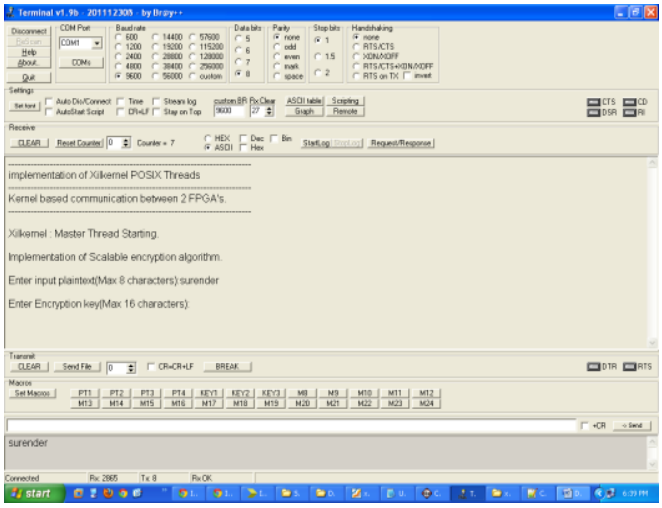

Fig9. Transmitter side terminal with message

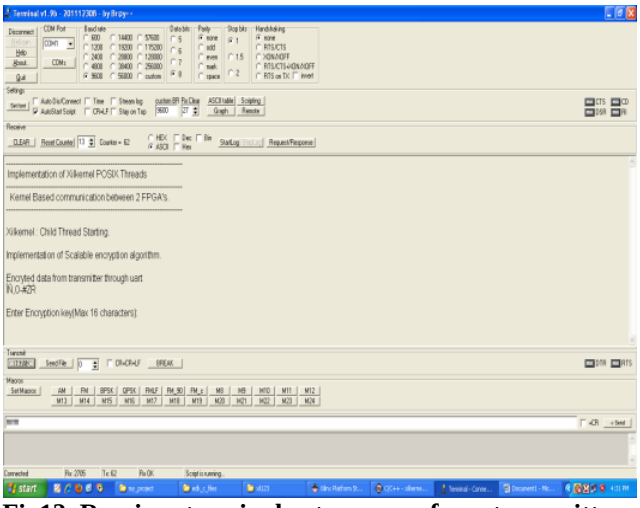

Fig12. Receiver terminal get message from transmitter

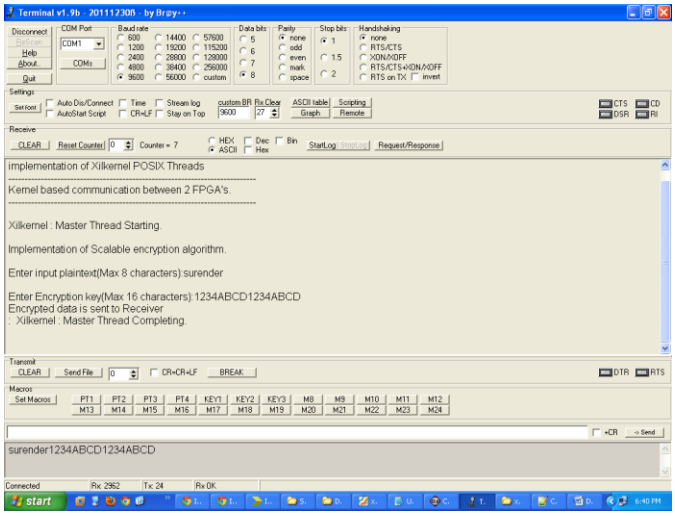

Fig10. Transmitter side terminal with key and data is sanded to receiver.

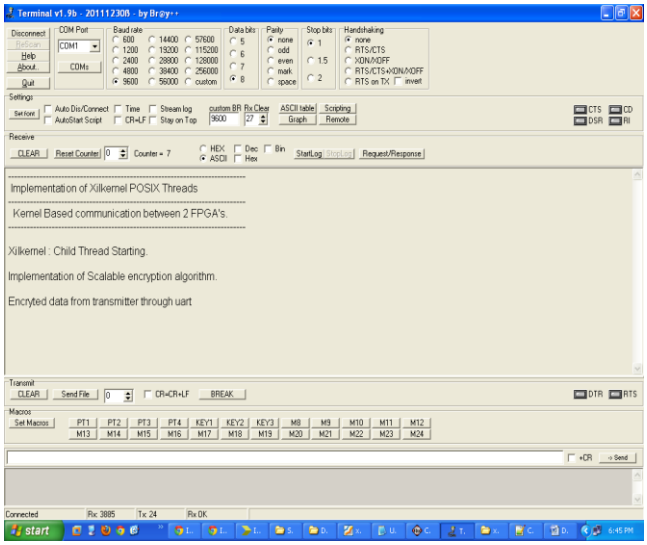

Fig11. Receiver side terminal

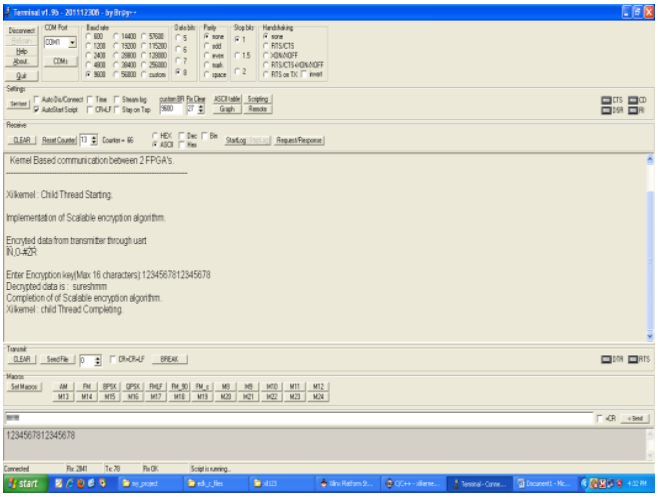

Fig13. Enter key in receiver terminal we get message.

V. CONCLUSION

Implementing a micro blaze soft core processor on the FPGA. The verification of communication between two FPGA's (chip to chip communication). And here the communication is done in secured manner by applying TINY encryption algorithm. Verifying the input and output data is done on RTOS (real time operating system) environment.

VI. REFERENCES

[1] http://www.design-reuse.comlarticlesIl3981 /fpga-implementation-of-aes-encryptio n-and-decryption n. html.

[2] B. Schneider. 1996. Applied Cryptography, Protocols, Algorithms, and Source Code in C, John Wiley and Sons Inc. 2nd Edition. New York, U.S.A.

[3] G.B. Arfken, D.F. Griffing, D.C. Kelly and J priest. University Physics San Diego, CA Harcourt Brace, Jovanovich Publishers, 1989.

[4] San Diego, CA Harcourt Brace, Jovanovich Publishers, 1989.

[5] http://www.techmaish.comlmaximum-internet-speed-available-in-the-world.

[6]4.6.3 of D. E. knuth, The Art of Computer Programming Seminumeritical Algorithm, Volume 2, Reading M.A. Addison Wesley, Second Edition, 1981.

[7]QingLi, Caroline Yao "Real-Time Concepts for Embedded Systems".

[8] Tran Nguyen BaoAnh*t, Su-Lim TantSurvey and performance evaluation of real-time operating systems (RTOS) for small microcontrollers", *Renesas Technology Singapore, Singapore Engineering Centre, Singapore 098632, t School of Computer Engineering, Nanyang Technological University, Singapore 639708.

[9] Awais M. Kamboh, Adithya H. Krishnamurthy and Jaya Krishna K.Vallabhaneni "Demonstration of Multitasking using Thread RTOS on Micro blaze and PowerPC".

[10] Operating system for Xilinx embedded processor" at http://www.em.avnet.com.

[11] SaratYoowattana, Chinnapat Nantajiwakornchai, Manas Sangworasil "A Design of Embedded DMX512 Controller using FPGA and XILKernel" ,2009 IEEE Symposium on Industrial Electronics and Applications (lSIEA 2009), October 4-6,2009, Kuala Lumpur, Malaysia.

[12] http://www.xilinx.com.

[13] M. Ibrahimy, M.B.Reaz, K.Asaduzzaman and S.Hussain. 2007. FPGA Implementation of RSA Encryption Engine with Flexible Key Size. International Journal of Communications. Issue 3. Volume I.