Implementation of High-Speed Secure Communication between Multiple FPGA Systems Using RTOS

G. Snehalatha

Assistance Professor, (ECE), M. TECH (VLSI System Design)
CVR College of Engineering, India

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).The proposed architecture is simulated using Modelsim and synthesized using Xilinx ISE 13.2 and it will be implemented on XC3S500e Spartan 3E FPGA board for hardware implementation and testing. The Xilinx Chip scope tool will be used to test the FPGA inside results while the logic running on FPGA .The necessary software for this design is written using the feature-rich c/c++ code editor and compilation environment provided within the SDK.

Keywords:
High-Speed Secure, Multiple FPGA Systems, RTOS, XilinxChip

1. Introduction

Reconfigurable System like FPGA platform has the potential to provide the performance benefit of ASIC and the flexibility of processors. FPGA is a collection of programmable gates embedded in a flexible interconnect network that can contain hard or soft microprocessors. FPGAs combine programmability of processors with the performance of custom hardware. As FPGAs can provide a useful balance between performance and flexibility, they becomes the primary source of computation in critical embedded systems. A few research works [1, 2, 3], shows 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 our paper lies in the implementation of cryptographic algorithm utilizing threads, run 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 [4] and RSA [5] as our cryptographic algorithm to be implemented, which is quite popular in the public key cryptography domain. Now in comparison to symmetric key cryptography [6] there are few advantages in the public key cryptography regarding scalable communication, trusted identification, and simpler key management [6]. Choosing an algorithm to get implemented in FPGA hardware, concerned facts are processing power, execution time and resource utilization. Among existing implementations of RSA algorithm [5], here comes few solutions like Montgomery Algorithm for Modular Multiplication [7,8]. Several Montgomery designs have been proposed for ASIC and FPGA platform on limited resource availability to satisfy the execution time burden [2, 9, 10]. In work [1] the execution time depends on the value of exponent term (public key, and private key) where as the encryption and decryption time of our chosen algorithm [11] (Binary method) does not depend upon the key values but does depend on the number of modular multiplications. Very of late AES [12] is one of the famous crypto algorithms available over the internet traffic due to its high security and parallel nature of processing [13]. Parallel computing gives the highest flexibility in this domain, but in case of parallel computing though there is an increase of processing speed, the heat dissipation and area is increased which

Manuscript received April 5, 2026
Manuscript revised April 20, 2026
https://doi.org/10.22937/IJCSNS.2026.26.4.6

makes it difficult to implement it in a constrained system with smaller resources, such as real time embedded systems. This is one of the reasons to adopt RSA in our proposal. Our contribution in this paper can be briefed as follows

- Establishing reliable communication between multiple FPGA devices is an essential component for developing complex real time systems used for applications like real time data acquisition and processing [1]. · Our paper proposes the major issue that two threads running separately on each board can communicate with other via RS232 communication link.
- The algorithm is implemented in hardware by introducing the concept of thread execution by RTOS and its hardware utilization proves that our implementation is better with respect to the existing work [1,2,3].
- Large value of secret key (exponent term) increases Security of the encryption process, where as the execution time of the binary method [11] that we used for RSA process is invariant to the value of exponent term.

2. Preliminaries

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. Cryptography prior to the modern age was effectively synonymous with encryption, the conversion of information from a readable state to apparent nonsense. The originator of an encrypted message shared the decoding technique needed to recover the original information only with intended recipients, thereby precluding unwanted persons to do the same. Since World War I and the advent of the computer, the methods used to carry out cryptology have become increasingly complex and its application more widespread.

Until modern times cryptography referred almost exclusively to encryption, which is the process of converting ordinary information (called plaintext) into unintelligible text (called cipher text).Decryption is the reverse, in other words, moving from the unintelligible cipher text 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 cipher text. A "cryptosystem" is the ordered list of elements of finite possible plaintexts, finite possible cyphertexts, finite possible keys, and the encryption and decryption algorithms which correspond to each key. Keys are important, as ciphers without variable keys can be trivially broken with only the knowledge of the cipher used and are therefore useless (or even counter-productive) for most purposes. Historically, ciphers were often used directly for encryption or decryption without additional procedures such as authentication or integrity checks.

B. MICROBLAZE overview

In terms of its instruction-set architecture, Micro Blaze is very similar to the RISC-based DLX architecture described in a popular computer architecture book by Patterson and Hennessy. With few exceptions, the Micro Blaze can issue a new instruction every cycle, maintaining single-cycle throughput under most circumstances.

The Micro Blaze 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 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.
Many aspects of the MicroBlaze can be user configured: cache size, pipeline depth (3-stage or 5-stage), embedded peripherals, memory management unit, and bus-interfaces can be customized. The area-optimized version of MicroBlaze, which uses a 3-stage pipeline, sacrifices clock-frequency for reduced logic-area. The performance-optimized version expands the execution-pipeline to 5-stages, allowing top speeds of 210 MHz (*on Virtex-5 FPGA family.) Also, key processor instructions which are rarely used but more expensive to

implement in hardware can be selectively added/removed (i.e. multiply, divide, and floating-point ops.) This customization enables a developer to make the appropriate design tradeoffs for a specific set of host hardware and application software requirements. With the memory management unit, MicroBlaze is capable of hosting operating systems requiring hardware-based paging and protection, such as the Linux kernel. Otherwise it is limited to operating systems with a simplified protection and virtual memory-model: e.g. Free RTOS or Linux without MMU support. Micro Blaze's overall throughput is substantially less than a comparable hardened CPU-core (such as the PowerPC440 in the Virtex-5.)

C. EDK(Embedded development kit):

Xilinx's EDK (Embedded Development Kit) is the development package for building Micro Blaze (and PowerPC) embedded processor systems in Xilinx FPGAs. Hosted in the Eclipse IDE, the project manager consists of two separate environments: XPS and SDK.

Designers use XPS (Xilinx Platform Studio) to configure and build the hardware specification of their embedded system (processor core, memory-controller, I/O peripherals, etc.) The XPS converts the designer's platform specification into a synthesizable RTL description (Verilog or VHDL), and writes a set of scripts to automate the implementation of the embedded system (from RTL to the bit stream-file.) For the Micro Blaze core, the EDK normally generates an encrypted (non human-readable) net list, but the processor description (written in VHDL) can be purchased from Xilinx. The SDK handles the software that will execute on the embedded system. Powered by the GNU tool chain (GNU Compiler Collection, GNU Debugger), the SDK enables programmers to write, compile, and debug C/C++ applications for their embedded system. Xilinx includes a cycle-accurate instruction set simulator (ISS), giving programmers the choice of testing their software in simulation, or using a suitable FPGA-board to download and execute on the actual system.

3. Design Methodology

A. Tiny encryption and decryption algorithm:

The Encryption Algorithm is a block cipher notable for its simplicity of description and implementation, typically a few lines of code. It was designed by David Wheeler and Roger Needham of the Cambridge; it was first presented at the Fast Software Encryption workshop in Leuven in 1994. This Algorithm operates on 64-bit blocks and uses a 128-bit key. It has a Feistel structure with a suggested 64 rounds, typically implemented in pairs termed cycles. It has an extremely simple key schedule, mixing all of the key material in exactly the same way for each cycle. Different multiples of a magic constant are used to prevent simple attacks based on the symmetry of the rounds. The magic constant, 2654435769 or 9E3779B916 is chosen to be 232/ϕ, where ϕ is the golden ratio.

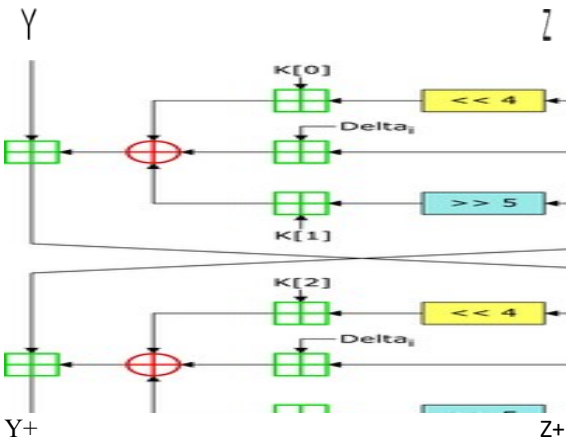


Figure 1 Tiny encryption and decryption algorithm:

sum += delta;
y += ((z << 4)+K(0)) ^ z+sum ^ ((z >> 5)+K(1));
z += ((y << 4)+K(2) ^ y+sum ^ ((y >> 5)+K(3);

Encryption Routine:

As shown in Figure 1 and 2, the inputs to the encryption algorithm are a plaintext block and a key K .The plaintext is P = (Left[0], Right[0]) and the cipher text is C = (Left[64], Right[64]). The plaintext block is split into two halves, Left[0] and Right[0]. Each half is used to encrypt the other half over 32 rounds of processing and then combine to produce the cipher text block. Each round i has inputs Left[i-1] and Right[i-1], derived from the previous round, as well as a sub key K[i] derived from the 128 bit overall K. The sub keys K[i] are different from K and from each other.

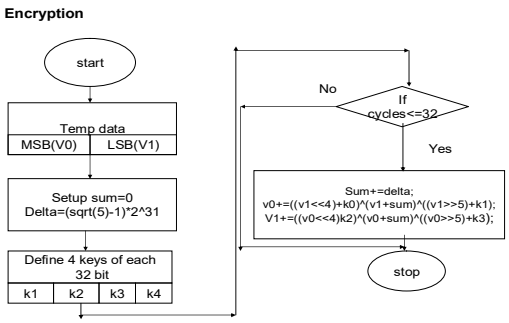


Figure 2 Tiny encryption algorithm:

The constant delta=√(5-1)*231=9E3779B9h is derived from the golden number ratio to ensure that the sub keys are distinct and its precise value has no cryptographic significance. The round function differs slightly from a classical Fiestel cipher structure in that integer addition modulo 2^{32} is used instead of exclusive-or as the combining operators.

```
void code(long* v, long* k)  {
unsigned long y = v[0], z = v[1], sum = 0, /* set up */
delta = 0x9e3779b9, n = 32 ;  /* a key schedule constant */
while (n-->0) {         /* basic cycle start */
 sum += delta ;
 y +=  (z<<4)+k[0] ^ z+sum ^ (z>>5)+k[1] ;
 z +=  (y<<4)+k[2] ^ y+sum ^ (y>>5)+k[3] ;  /* end cycle */
     }
 v[0] = y ; v[1] = z ; }
```

Figure 3 Decryption Routine:

As Figure 4, the decryption is essentially the same as the encryption process; in the decode routine the cipher text is used as input to the algorithm, but the sub keys K[*i*] are used in the reverse order. The intermediate value of the decryption process is equal to the corresponding value of the encryption process with the two halves of the value swapped

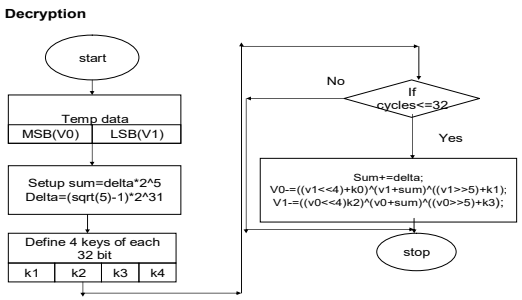


Figure 4 Decryption Routine Flow

B. Real Time Operating System (Xi/kernel)

Real time embedded systems are typically designed for various purposes such as to control or to process Data, meeting certain deadlines at the right time. To achieve this purpose, real-time operating systems (R TOS) are often used. RTOS can be defmed as: "a program that schedules execution in a timely manner, manages system resources, and provides a consistent foundation for developing application code. "[18J, more specifically An RTOS is a piece of software with a set of APIs for users to develop applications RTOS are typically differentiated from generic OSes regarding the following criteria i.e. Preemptive or priority-based scheduling, Predictability in task synchronization, deterministic behaviors [19]. Multitasking, other key features of RTOS which usually means that the software is divided into tasks, or smaller subsets of the total problem and at run-time, creating an environment that provides each task with its own processor [20]. There are various RTOS are available for microcontroller as well as for FPGA based design ie VxWorks, QNX, eCos, LynxOS, and RTLinux[2I] Here we have chosen Xilkernel as our RTOS which is provided by Xilinx. Xilkernel is a small, robust, and modular kernel. It is highly integrated with the Platform Studio Frame .It allows a very high degree of customization [22]. It supports the core features required in a lightweight embedded kernel, with a POSIX API. Xilkernel works on both the Micro Blaze and PowerPC 405 processors. Xilkernel has very low memory footprint, it uses 7-16 kb of BRAM in a multi threaded program [4], which is much smaller than the RTOS used in microcontroller [19]. Now scheduling is also a major issue in the environment of multitasking, Xilkernel supports priority driven, preemptive scheduling with time slicing (SCHED]RIO) or simple round-robin scheduling (SCHED_RR)[4]. Xilkernel is structured as a library. The user application source files must link with Xilkernel to access Xilkernel functionality. In Xilkernel a thread is the unit of execution and is analogous to a process. Threads are coded like functions. At least one thread is required to spawn from the system at kernel start [4].

C. Hardware Architectural Design

This work is implemented using the Xilinx EDK 11.1 (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. 5

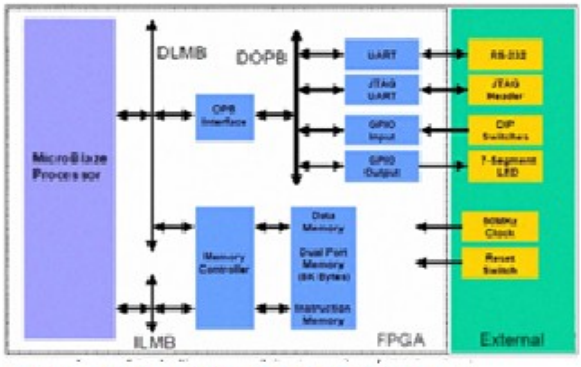


Fig 1: Architecture of each FPGA system

Figure 5 Architecture of each FPGA system

D. Serial communication between FPGAs

The system that is used for establishing the serial communication between the multiple FPGA systems is UART (Universal Asynchronous Receiver Transmitter). Here "BRG" stands for "Baud Rate Generator" which controls the speed of the data communication in RS232 channel. Both receiver and sender side must work in the same band ratio otherwise data will be lost. BRG control the received data store initially at received FIFO and the transmit Data FIFO transfer the data through the transmitter Module (TX Module). Micro blaze Processor Local Bus Interface module is used here for connecting and accessing UART module to the Processor as shown in the Fig.1.

4. Implementation

The proposed architecture was synthesized using Xilinx ISE Il.l [23] and was implemented on XC3S500e Spartan 3E FPGA Board [23]. The necessary software for this design is written using the feature-rich C/C++ code editor and compilation environment provided within the SDK (Xilinx Software Development Kit). The SDK provides an environment for creating software platforms and applications targeted for Xilinx embedded processor (Micro blaze)[23].We have tested the real time execution of the program in the hardware in 4 ways, which are described in fig 2.

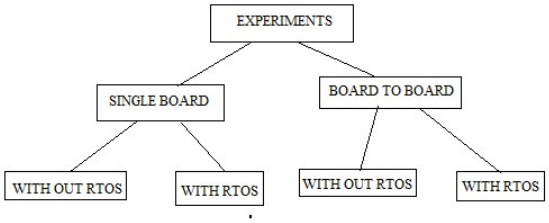


Fig 2

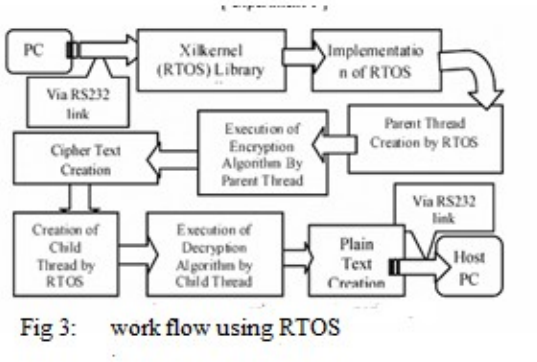


Fig 3: work flow using RTOS

Figure 6 Work Flow using RTOS

A real time data is received from the key board using the hyper-terminal application of a host computer which is then being sent to the board using RS232(9600bps,no parity bit)serial cable in the DCE port on the board, receiving the data the encryption engine execute the encryption algorithm and creates the cipher text. Again this cipher text converts to the plain text by decryption program by decryption engine. The plain text is transferred to the PC hyper terminal via theRS232 DCE port (9600 bps, no parity bit) .Fig. 3 shows the Hyper-terminal output of experiment. The encryption and decryption algorithm is processed in two different threads run by the RTOS. The kernel starts with the executing of a main thread which is executing the encryption algorithm now we can refer this thread as parent thread because later this thread is creating a child thread and which is executing the decryption algorithm

Communication Setup

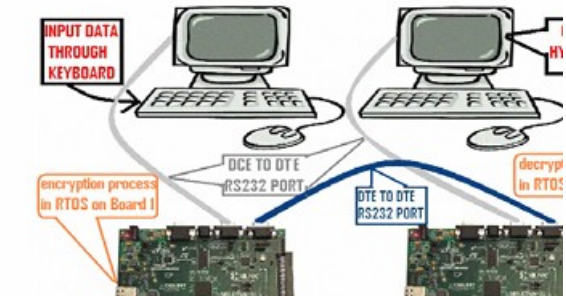


Figure 7 Communication Setup Flow

This is the most challenging and interesting part of our paper, where two FPGA board can communicate. Successfully. The real time data are taken from Key Board into the board 1 through RS232 (DTE to DCE cable) port where the encryption process is going on. The encrypted plain text called cipher text is being sent to the board 2 usingRS232 (DTE to DTE cable) port. After receiving the cipher text, board 2 decrypts the encrypted text and converts to the plaintext again. This plain text is being send to the HyperTerminal of next PC through RS232 (DCE to DTE cable). Shows the Hyper-terminal output of experiments. This proposed architecture is tested for both cases, without RTOS and with RTOS.

5..RESULTS

A.HARDWARE RESULTS:

The below figures 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 below figures.

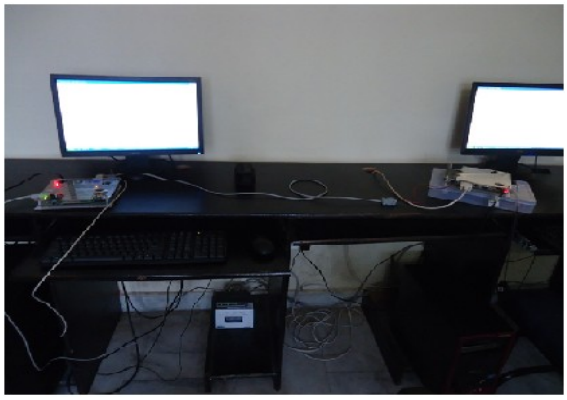


Figure 8 Communication in two FPGA

B.SOFT WARE RESULTS*:*

WITH RTOS

1. Example

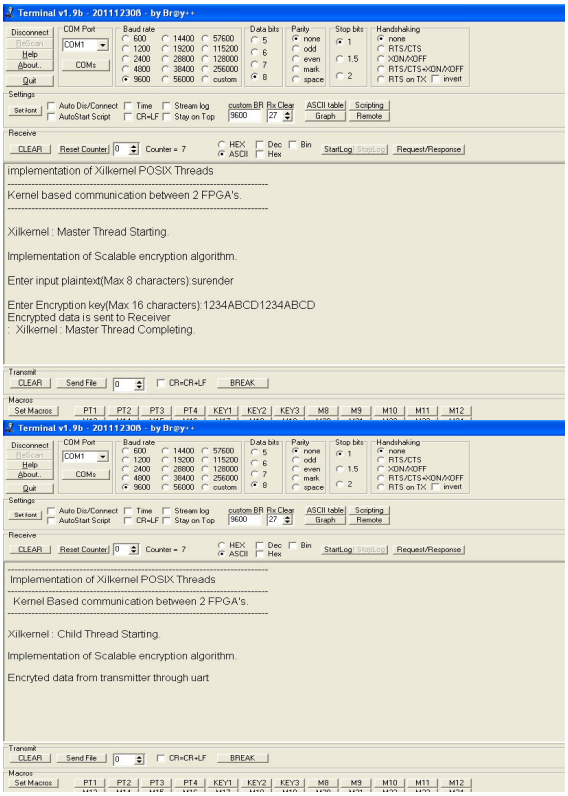


Figure 9 Examples with RTOS

2. Example

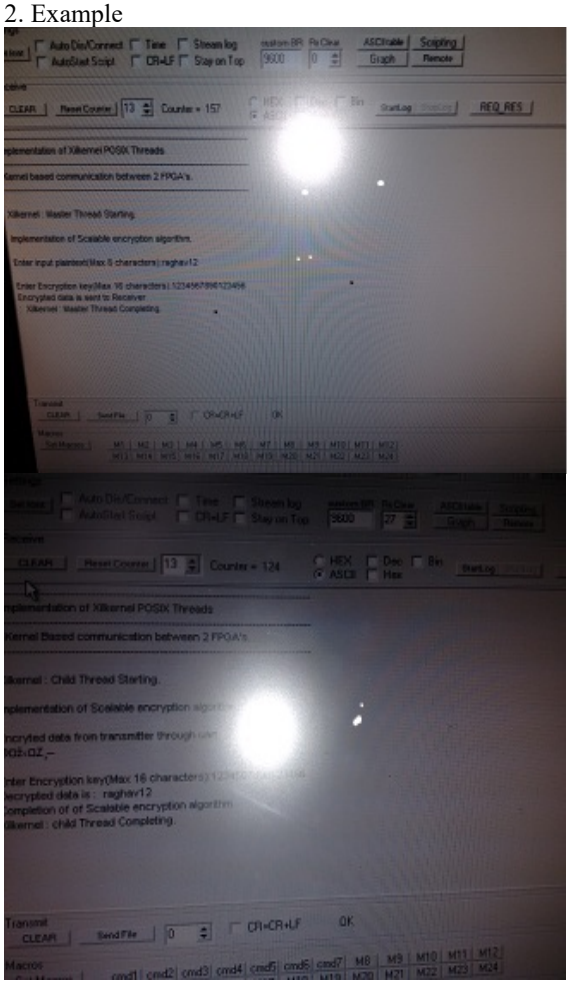
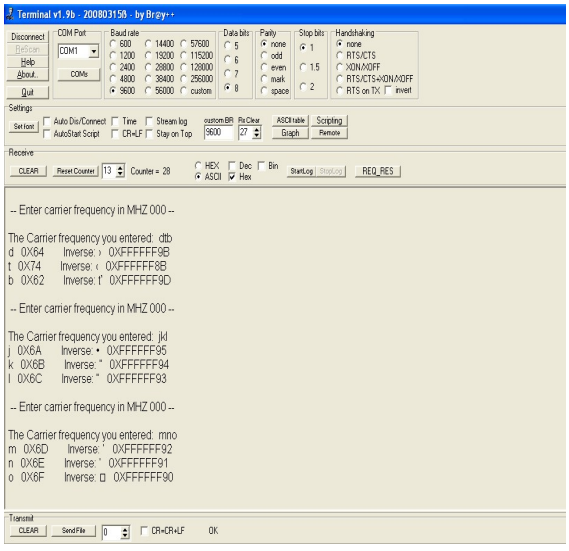
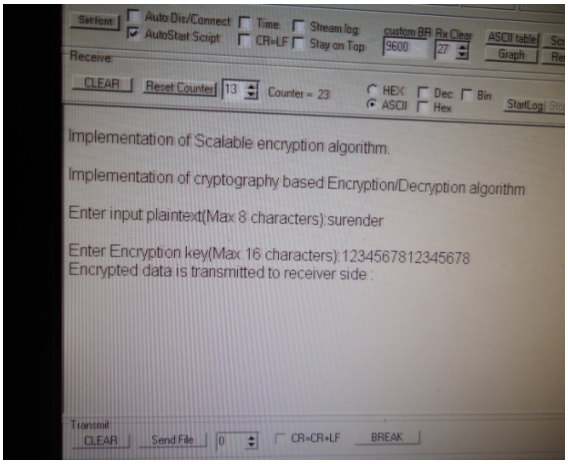


Figure 10 Examples with RTOS

WITHOUT RTOS





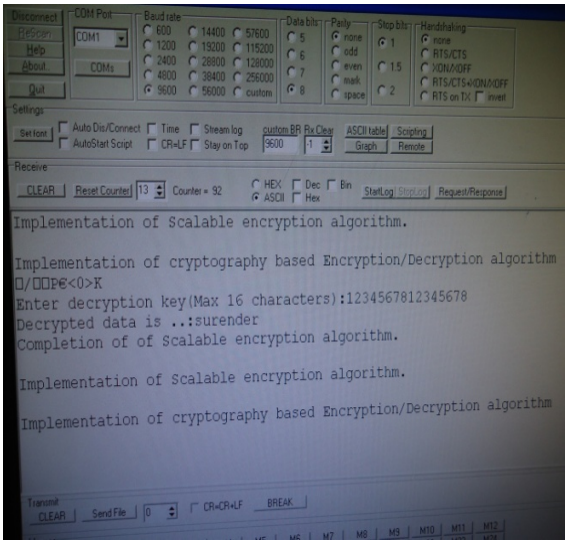


Figure 11 Examples without RTOS

TABLE I

COMPARISION OF EXCUTING SPEED BETWEEN PROPOSED AND EXISTING

clock freq 50Hz	Encryption process		Decryption process	
	clock cycles	Time in (ms)	clock cycles	Time in (ms)
Existing	5176	0.103	1116878	22.34
Proposed	4414	0.088	7146	0.143

TABLE II

COMPARISIO OF RESOURCE USAGE WITH EXISTING WORKS

Resource type	Proposed	Existing	Available
4 input LUT	1440	2667	9312

Total: 9.076ns (5.136ns logic, 3.940ns route)

(56.6% logic, 43.4% route)

Total REAL time to Xst completion: 34.00 secs

Total CPU time to Xst completion: 33.80 secs

Total memory usage is 210840 kilobytes

Number of errors : 0 (0 filtered)

5.1 ADVANATGES:

- High speed devices such as serial ADC/DACs can be interfaced to Micro blaze
- Full digital techniques can give room for full reconfigurability of the MODEM for different specifications
- FPGA based implantation will result in extremely high speed and also can provide room for adding additional hardware blocks.
- Low power solution
- High reliability

5.2 APPLICATIONS:

- The Xilinx Micro blaze soft processor and PowerPC hard processor are widely used in FPGA based CSOC (configurable system on chip) applications.
- Mainly in communication systems our proposed solution can be very usefull in terms of speed as well as in real time environment.
- We can use this type application in processor based solutions.
- And we can use at aviation, navigation and other equipment
- Software defined radios for military applications

6. Conclusion

Implementing a microblaze soft core processor on the FPGA.The verification of l communication between two FPGA's(chip to chip communication).And here the communication is done in secured manner by applying tiny encryption algorithm. This is all done on RTOS(realtimeoperatingsystem) environment. Verifying the input and output data.

References

[I] S. Sau , C. Pal and A Chakrabarti "Design and Implementation of Real Time Secured RS232 Link for Multiple FPGA Communication, Proc. Of International Conference on Communication, Computing & Security, 201 I, ISBN - 978-1-4503-0464-1.

[2] C. D. Walter. August 1999. Montgomery's Multiplication Technique: How to Make It Smaller and Faster. Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science, Springer. No. 1717. pp. 80-93.

[3] A Mazzeo, L. Romano, G. P. Suggest and N. Mazzocca. 2003. FPGABased Implementation ofa Serial RSA Processor. Design. Proceedings of the conference on Design, Automation and Test in Europe - Volume I. ISBN:O-7695-1870-2.

[4] Xilkernel_v3.00.pdf on www.xilinx.com.

[5] R. L. Rivets et al. 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM. Vol. 21. pp. 120-126.

[6] Cryptography & Network Security By Behrouz AForouzan.

[7] Montgomery Algorithm for Modular Multiplication Professor Dr. D. J. Guan, August 25, 2003.

[8]RSA & Public Key Cryptography in FPGAs, John Fry, Martin Langhammer Altera Corporation -Europe

[9] A. Tenca, C. Koc. 1999. A Scalable Architecture for Montgomery Multiplication. Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science, No. 1717, pp. 94-108.

[10]. A. Tenca, G. Todorov, C. Koc. May 2001. High-radix design of a scalable modular multiplier. Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science, Springer. No. 2162. pp. 185-201.

[II] High-Speed RSA Implementation, Cetin Kaya Koc, November 1994, Version 2.0, ftp://ftp.rsa.comlpub/pdfs/tr20I.pdf.

[12]] http://csrc.nist.gov/publications/fips/fipsI97Ifips- 197.pdf.

[13] http://www.design-reuse.comlarticlesIl3981 /fpga-implementation-of-aes-encryption n-and-decryptio n. html.

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

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

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

[17] 4.6.3 of D. E. Knuth , The Art of Computer Programming Seminumeritical Algorithm, Volume 2, Reading M.A. : Addison Wesley, Second Edition, 1981.

[18] Qing Li, Caroline Yao "Real-Time Concepts for Embedded Systems".

[19] Tran Nguyen Bao Anh*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.

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

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

[22] Sarat Yoowattana, 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.

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

[24]] M. Ibrahim, 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.