

# Performance Comparisons, Design, and Implementation of RC5 Symmetric Encryption Core using Reconfigurable Hardware

Omar Elkeelany<sup>1</sup>

Tennessee Technological University/Electrical and Computer Engineering, Cookeville, TN  
Email: oelkeelany@tntech.edu

Adegoke Olabisi

Tennessee Technological University/Electrical and Computer Engineering, Cookeville, TN  
Email: aoolabisi21@tntech.edu

**Abstract**—With the wireless communications coming to homes and offices, the need to have secure data transmission is of utmost importance. Today, it is important that information is sent confidentially over the network without fear of hackers or unauthorized access to it. This makes security implementation in networks a crucial demand. Symmetric Encryption Cores provide data protection via the use of secret key only known to the encryption and decryption ends of the communication path. In this paper, first, an overview of two well known symmetric encryption cores is presented, namely the 3DES and RC5. Then a performance evaluation of their computer based implementation is compared to demonstrate the RC5 superior performance. The conventional hardware architecture of the RC5 core is presented and investigated. A hardware system design is proposed to improve its performance. The proposed architecture achieved with three stage pipeline technique an increased encryption throughput as compared to related work.

By exploiting modern features in Field Programmable Gate Arrays (FPGA), which allow the modeling of a System-on-Programmable-Chip (SoPC), this paper proposes a model for symmetric encryption algorithms (e.g., RC5). Structural System analysis of the proposed model shows that it offers extra security against single-site physical access attack that other implementations are vulnerable to. By evaluating the performance of this proposed SoPC model, one finds that it raises the encryption throughput to 300 Mbps. Hence, we report over 80% increase in the encryption throughput as compared to related work. Moreover, our work lowers the implementation cost due to the integration of all system parts into one chip.

**Index Terms**—Cryptography, Systems-on-Programmable Chips, analysis and simulation, Hardware Description Language

## I. INTRODUCTION

The efficiency and success of e-commerce and business was fueled by the underlying growth of available network bandwidth. Over the past few years,

internet-enabled business or e-business has drastically improved revenue and efficiency of large scale organizations. It has enabled organizations to lower operating costs and improve customers' satisfaction. Such applications require networks which accommodate voice, video and protected data [1, 2]. Obviously, privacy must be protected, to maintain the growth of these applications.

However, as today's networks make more and more application to users, they become vulnerable to a wider range of security threats. To prevent those threats and ensure that networks are not compromised, security and privacy must be integrated into network backbone (i.e. switches, routers, servers, etc.).

There is a dire need for integrity and confidentiality of information passed across large scale networks. Encryption algorithms play a great role in achieving these needs [1]. Cryptosystems are of two types: symmetric and asymmetric. Symmetric cryptosystems [2-4] use the same key (the secret key) to encrypt and decrypt a message, and Asymmetric cryptosystems use one key (the public key) to encrypt a message and a different key (the private key) to decrypt it. Asymmetric cryptosystems are also called "public key" cryptosystems. It has however been proven that asymmetric systems are slow to support bulk data encryption [5-7]. Wireless Application Protocol (WAP) forum specifies RC5 [8-11] as its encryption algorithm for its Wireless Transport Level Security (WTLS) clients and servers [12].

The work of [13] proposed an area optimized hardware architecture to the RC5 core into a Field Programmable Gate Array (FPGA) device with fewer resources than the conventional one. But the encryption throughput was found *less* than the conventional architecture, and it did not propose any modifications to the conventional system architecture. This paper presents integrated RC5 encryption and decryption cores with memory elements, and key-expansion units into one chip. It presents a case study and undertakes a performance comparison between RC5 and 3DES as a low end for symmetric encryption family of algorithms. Various design methodologies will be presented to achieve the

<sup>1</sup>: Corresponding author

optimal utilization of the target chip. Simulation and prototype synthesis results are summarized and discussed in detail.

Hardware Description Language is used to model both conventional and proposed architectures. On top of the security add-in that is inherited by the avoidance of external memory use, an important test measure is the speed of operation. Simulation tools are used to verify this speed improvement.

II. OVERVIEW OF TWO SYMETRIC ENCRYPTION ALGORITHMS: 3DES & RC5

Symmetric Encryption cores provide security to data by using a secret key both for encryption and decryption processes. Historically, Triple Data Encryption Standard (3DES) in Cipher Block Chaining (CBC) mode was proposed in IPsec ESP network encryption [14] as a symmetric encryption algorithm. Encryption using DES algorithm is the most time consuming process. DES uses a 56-bit short key, and block sizes of 64 bits. The algorithm has 19 distinct steps as shown if figure 1. The first step is a key independent transposition on the 64-bit input block, using a fixed 64-bit permutation table to change bit locations in the input block. The last step is the exact inverse of this transposition. In the pre-output 18<sup>th</sup> step, a 32 bit SWAP operation is performed. The remaining steps (2 to 17) are functionally identical and are dependent on different portions of the input key. Each of these 16 steps (or 16 rounds) takes two 32 bit inputs, and produces two 32 bit outputs. The left output is a COPY of right input. The right output is from an exclusive-or operation (XOR) of left input and a function *f* of right input and the step key ( $K_i, i=1:16$ ). The function *f* consists of 4 operations. First, a 32:48 bit transposition/expansion of the right input is applied; second a 48 bit XOR of the output with the step key. Then a group mapping is performed to reduce the output size from 48 to 32 bits using two dimensional look-up table of 4 rows and 16 columns for all possible 64 inputs. Lookup tables ( $S_i, i=1:8$ ) are indexed using row and column numbers given by 2 and 4-bits of the 6-bit input respectively. The table entry is a 4-bit value. Also, a 32-bit transposition is performed.

Each of the 16 steps has a special key ( $K_i$ ), which is derived from the '56-bit' secrete key using a special

TABLE I. DES BASIC OPERATIONS

Basic Operation	Equivalent simple operations		
	Type	# Times needed	Space needed
b bit transposition	One dimensional table look-up	b	b
Two dimensional table map (for 6:4 bit map)	Multiply	1	
	Add	1	
	One dimensional table look-up	1	4 rows x 16 cols.

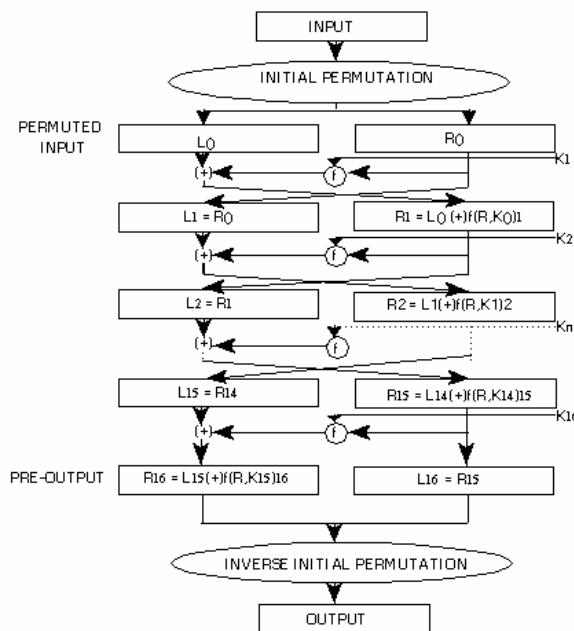


Figure 1: DES encryption steps

function. In this function, an initial 56-bit transposition is performed. Before each step, the key is divided into two 28-bit sub-keys, each of which is rotated left using LEFT SHIFT operation. Finally, a 56:48 bit transposition/reduction is performed.

Table 1 lists all DES basic operations along with a set of equivalent simplified instructions for further analysis, and comparison. These simplified instructions are derived from a virtual instruction set of a 32-bit machine capable of performing simple two operand instructions. We assume all instructions of this machine execute in one cycle (i.e. all instructions are of similar complexity).

More recently, RC5 algorithm was developed by Ronald Rivest in 1995 [2] as a parameterized symmetric encryption core. RC stands for "Rivest Cipher", or alternatively, "Ron's Code". RC5 parameters are; a variable block size (*w*), a variable number of rounds (*r*)

TABLE II. DES OPERATIONS IN ONE BLOCK ENCRYPTION

Step #	Operation	# Times	Equiv. Total*	Notes
1,19	64 bit transposition	2	64x2	
2-17	32 bit COPY	16	16	16 steps
2-17	32 bit XOR	16	16	
2-17	48 bit transposition	16	48x16	<i>f</i> function
2-17	48 bit XOR	16	16	
2-17	6:4 bit Two dimensional Table Mapping	8x16	3x128	
2-17	32 bit transposition	16	32x16	
1	56 bit transposition	1	56	<i>KS</i> function
2-17	28 bit LEFT SHIFT	2x16	32	
2-17	48 bit transposition	16	48x16	
18	32 bit SWAP	1	1	Pre-output
			2697	DES Total
			8091	3DES Total

\* Using equivalent simple substitutions from table II

and a variable key size (k). Allowable choices for the block size (w) are 32, 64 and 128 bits. The number of rounds can range from 0 to 255, while the key size can range from 0 bits to 2040 bits in size. RC5 has three modules: key-expansion, encryption and decryption units. Relatively, RC5 is more secure than RC4 [15] but is slower in operation. Generally, implementing ciphers in software is not efficient based on its speed in terms of computation and hence the use of hardware devices is an alternative [16, 17].

The RC5 algorithm uses three primitive operations and their inverses.

- (1) Addition/subtraction of words modulo  $2^w$ , where w is the word size.
- (2) Bit-wise exclusive-or denoted by XOR.
- (3) Rotation: the rotation of word x left by y bits is denoted by  $x \lll y$ . The inverse operation is the rotation of word x right by y bits, denoted by  $x \ggg y$ .

In the key expansion module, the password key K is expanded to a much larger size using an expansion table (S). The size of table S is  $2(r+1)$ , where r is the number of rounds [8]. The key-expansion process must be performed before encryption or decryption processes.

The encryption process takes a plain text input and produces a cipher text as the output. The decryption process takes a cipher text as the input and produces a plain text as the output. In general, the same plaintext block will always encrypt to the same ciphertext when using the same key in a block cipher whereas the same plaintext will encrypt to different ciphertext in a stream cipher [18]. Both processes use the expanded key along with segments of the input message to produce their outputs.

The conventional architecture of RC5, shown in figure 2, performs the encryption and decryption processes in two separate cores. As shown in figure 2, the RC5 Core needs to read the expanded key, in a sequential way, in order to encrypt the plain text. This gives a chance for unauthorized users, if they have access to the system, to tap and record the memory contents. On another site, they can use their recordings to decrypt the ciphered text. What is worse, if they have a physical access to the system they will also have access to the user-secret key. A basic way to avoid such attack is to physically secure the system, which might not be enough, especially if the authorized user does not intend to change the secret key

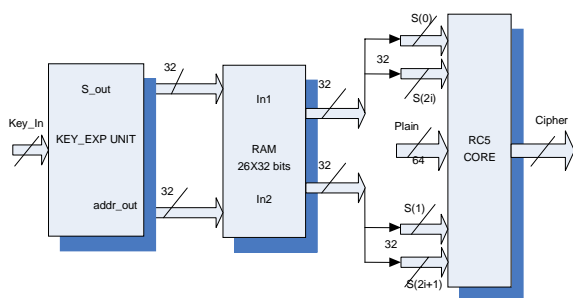


Figure 2: Conventional Architecture of RC5 Encryption System

very often. The proposed system architecture avoids the damages caused by this type of attack and also gives an improved performance in terms of throughput by making use of a three stage pipelining technique in its design.

### III. PERFORMANCE EVALUATIONS OF 3DES AND EXISTING RC5 MODELS

Although 3DES has a short key (56-bits) it is used here as a low end security algorithm for the sake of comparison. The RC5 with its parameterized feature and variable key lengths is more secure and efficient. It is desirable to compare the performance of these two algorithms, to understand how much effort is needed to achieve the higher levels of security offered by RC5. This section presents performance comparison of the two algorithms using a virtual instruction set of a 32-bits that consists of one cycle, homogenous, two-operand based, reduced simple instructions.

Consider, for example, replacing DES with its RC5 equivalent. One reasonable choice of parameters is RC5-32/16/7 for such a replacement. The input/output blocks will be 32-bits long just as in DES. The number of rounds is also the same (16) although each RC5 round is more like two DES rounds since all data registers rather than just half of them are updated in one RC5 round. Obviously, DES and RC5-32/16/7 each have 56 bit (7-byte) secret keys.

Unlike DES which has no parameterization and hence no flexibility, RC5 permits security upgrades as necessary. For example one can upgrade the above choice for a DES replacement to a 128-bit key by upgrading to RC5-32/16/16. Extra bits in the RC5 secret key make it less vulnerable to exhaustive search attacks.

#### A. Triple DES Performance Evaluation

Table 2 lists the DES and 3DES computations of total number of simple operations needed. 3DES is the chained form of DES with a chain size of 3. For simplicity, 3DES is assumed to have only 3 times number of operations as DES has. More accurately, 3DES actually requires an extra random Initialization Vector of 8 bytes, which is omitted here. For complete details of DES and 3DES algorithms see [19].

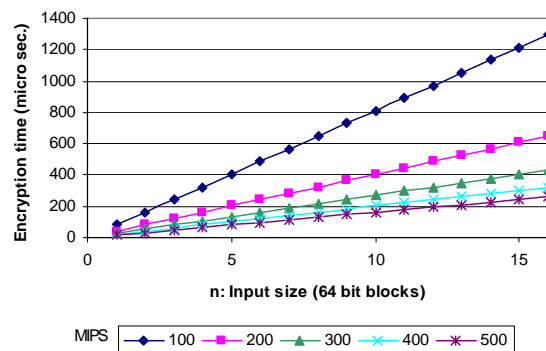


Figure 3: 3DES Encryption time vs. # of input blocks and processing power

As shown in table 2, the total number of operations per 64-bit block is 8091 operations. Given a message size of N bits then the number of blocks (n) is given by

$$n = \left\lceil \frac{N}{64} \right\rceil \tag{1}$$

Where  $\lceil \cdot \rceil$  denotes the smallest integer bigger than or equal to the operand. 3DES time complexity is linear and function of (n), (i.e.  $O(n)$ ).

Figure 3 shows the computed encryption time in micro seconds as function of (n) in blocks of 64 bits. The results are plotted with a range of operation frequency from 100 to 500 MHz. Since the machine executes one instruction per cycle, the operation frequency is equivalent to million of instructions per seconds (MIPS). One may use this general equation of encryption throughput:

$$Th(Mbps) = \frac{b}{I * C} = \frac{b * MIPS}{I} \tag{2}$$

where b is the block size in bits, I is the number of virtual machine instructions needed per block and C is the cycle time (in micro seconds). Consequently, one finds that a 3DES algorithm running on this virtual machine will have an encryption throughput of only 4 Mbps at 500 MHz operation frequency.

**B. RC5 Performance Evaluation of Existing Models**

Table 3 lists all RC5 basic operations and their equivalent simple ones, using the same virtual machine used before. In this table, one finds that RC5-32/16/16 needs only 160 instructions to run 16 rounds. Figure 4 shows the computed encryption time in micro seconds as function of input packet size in blocks of 64 bits. Using eq(2), this predicts a throughput of 200Mbps ( $Th=64*500/160$ ). Hence, the RC5 algorithm is much faster than 3DES, and it is simple to implement both in software and in hardware. In section 4, we will present a hardware implementation of RC5, and show that it delivers a better throughput of 300Mbps using much slower processing rates.

TABLE III  
RC5 OPERATIONS PER BLOCK

Step #	Block Operation	# of rounds	Equivalent Operations#	
			per round	Total
1&4	32-bit XOR	16	2	32
2	32-bit data dependent shift left	16	2	32
3&6	32-bit modulo Addition	16	2	32
5	32-bit data dependent shift right	16	2	32
3&6	32-bit one dimensional table lookup	16	2	32
Total (T)				160

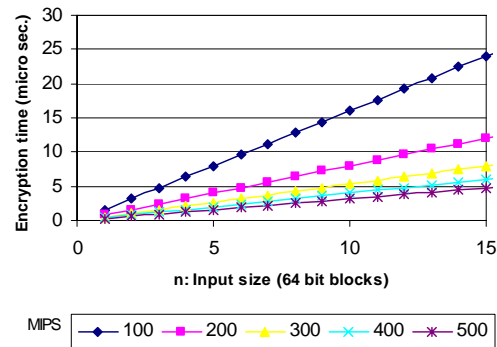


Figure 4: RC5 Encryption time vs. # of input blocks and processing power

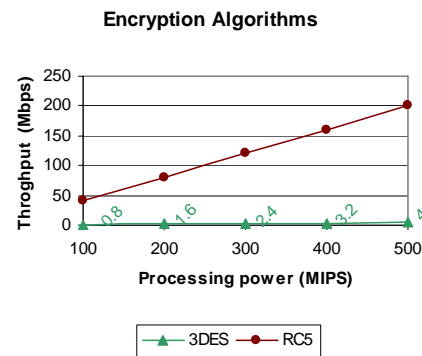


Figure 5: Encryption Algorithm comparison; Throughput vs. processing power

It is clear from figures 3 and 4 that the computation time for 3DES is much larger than that of RC5, for same input block size. For example, in 16 input blocks, comparison shows that RC5 takes only 2% of the computation time consumed by 3DES. This is further illustrated in figure 5. The conventional RC5 algorithm implemented in hardware reported 247 Mbps while an area optimized hardware realization of RC5 [13] reported a net 207 Mbps. It is well known that the key size of 3DES is too short and there is no easy way to increase it, and hence it is only presented here as a benchmark of comparison.

**IV. PROPOSED RC5 SYSTEM-ON-CHIP MODEL**

The proposed architecture, namely the System-on-Chip (SoPC) Model has three main components: the key\_exp unit, where the user secret key is recorded and expanded in size; the internal memory unit where the expanded keys are stored; and the RC5 Core, where the encryption and decryption processes are performed. In this architecture, the memory is part of the chip to avoid memory tapping attacks. This is made feasible using modern FPGA technology, with on-chip memories. This not only has the advantage of added security but also increases speed of operation as it decreases the data transaction latency between the core and the memory.

The proposed system uses the RC5-32/15/16 parameters. This means two 32-bit word inputs and outputs, 15 rounds and 16- byte (128-bit) secret key.

The proposed model is made flexible by providing these parameters as inputs to the circuit. Hence they can be modified to suit whatever ones goal is. The choice of  $r$ , for instance, affects both encryption speed and security. For some applications, high speed may be the most critical requirement and one could thus choose a small value of  $r$ . In other applications, such as credit card transaction, security is the primary concern and speed is relatively unimportant and one could thus go for a larger value of  $r$ . This makes the algorithm flexible.

The word size also affects speed and security. For example, choosing a larger value of word size,  $w$  larger than the register size of the CPU can degrade the encryption speed. It is also unusual and risky to have a fixed set of parameters.

Figure 6 below shows the proposed architecture of RC5-SoPC. This architecture requires a registration of the user secret key into the system at a particular timing of a SetKey signal. This registration is done with a sequence of short pulses. However, one step registration may be possible, assuming enough parallel lines for providing the secret key. In any case, the secret key does not stay as input and can change only at the presence of the SetKey pulse. In regular (i.e. encrypt) operation, only Plaintext input blocks are processed, and the RC5 core reads the internal memory logic module for the successful generation of the Ciphertext.

## V. DESIGN IMPLEMENTATIONS

We used Verilog HDL to implement both conventional (for comparison) and the proposed architectures in FPGA. The result obtained after modeling the algorithm tallies with the result obtained from the original code in C, which models the conventional architecture.

Various code implementations were tested using Xilinx design tools. The target hardware is a Xilinx Virtex-II -1000 FPGA. The resource utilizations of the various design methodologies are summarized in Table 4.

The conventional iterative RC5 “C-based” code was transformed into a Verilog HDL code. However, with the selected target FPGA, the design required 10,603 slices of the FPGA, where only 5120 are available. In order to minimize FPGA resource utilization, various design optimizations are researched. Some of the main methodologies followed are listed below:

### A. Fixed Key Synchronous Encryption

Here, each round is performed via a pulse given at the Start signal. Since the number of rounds is 15, fifteen Start pulses were given. The synthesis results have shown that this design fitted into the Virtex-II FPGA. Although the number of registers increased, the number of Adders/Subtractors and logic shifters decreased considerably. The number of required slices of the FPGA was 609. The use of the Start pulse also decreased the synthesis time.

### B. Variable Secret Key Asynchronous Encryption

Since the use of a fixed secret key does not allow for flexibility, the code is modified to accept a variable secret key. The signal Setkey is introduced to the design. At the positive edge of the Setkey signal, 8 bits of the user secret key are registered. Since the secret key is 128 bits (16 bytes), sixteen pulses of the Setkey signal were needed to fully register the keys. Once these pulses were provided, the encryption process follows, provided that an active high level of the Start signal is present. The total number of required slices of the FPGA was 31,791, and hence the design could not be synthesized as it was 620% over mapped. An increase in the number of logic components resulted from making the key a variable input as opposed

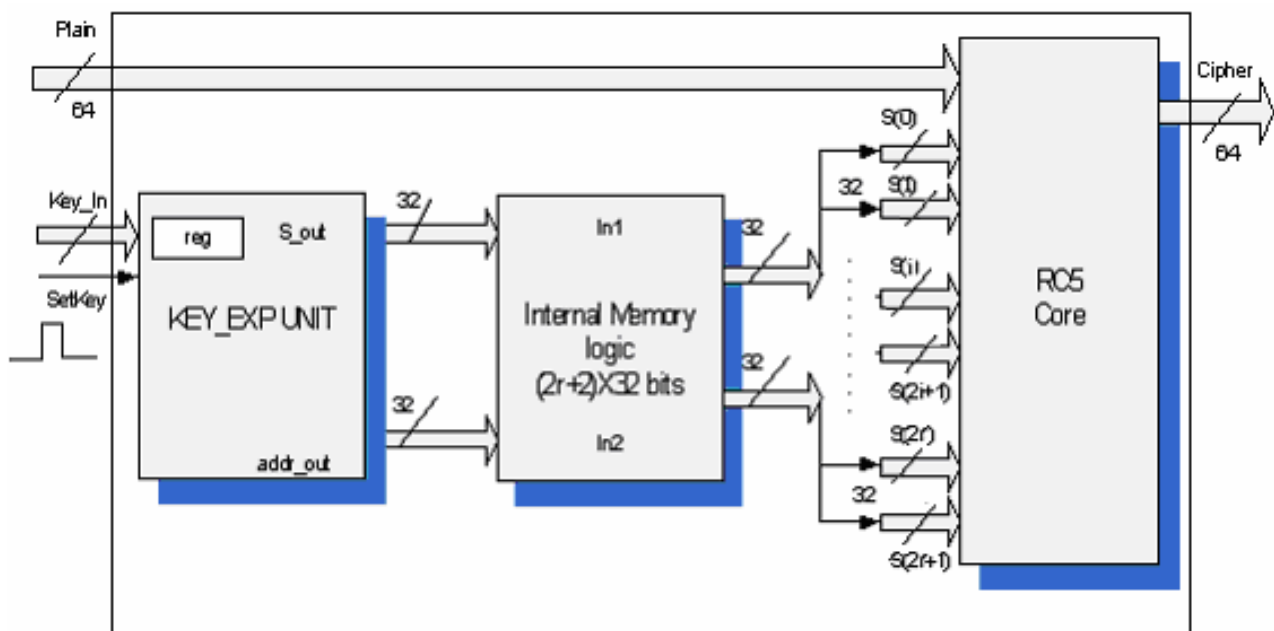


Figure 6: Integrating Key expansion and memory logic in one chip with the encryption core

to the fixed key code. Obviously, further code optimizations were needed to reduce the drastic increase in the number of required components.

*C. Variable Secret Key and Synchronous Encryption*

Synchronous encryption feature was added to the variable secret. This in essence means that the user secret key is being retrieved before the encryption, followed by encryption rounds performed at the edges of consecutive pulses of the Start signal. By making the secret key a variable input, and performing all rounds of encryption afterwards the synthesis result have shown that the Virtex-II chip was over utilized and thus the design could not fit into the chip. The number of required slices of the FPGA was 27,368, (i.e. 534% over mapped). The design thus needed further optimizations in order to be implemented to target hardware.

*D. Partial Serial Variable Key Registration (Phase I)*

The setup task of the algorithm was modified to test the effect of partial serial variable key registration. Conventionally, the setup task iterates 96 times in a “for-loop” fashion. The setup task was modified to perform 32 iterations for each pulse of an introduced Setkey signal. Thus, 3 Setkey pulses yields 96 iterations as needed. With this serial un-winding of the for-loop, we noticed a reduction in required FPGA slices, to 9,953, This was still not synthesizable, as it was 194% over mapped. Further optimization was needed to fit the design fit into the target FPGA.

*E. Full Serial Variable Key Registration (Phase II)*

The design was further optimized by performing a full un-winding of the for-loop in setup task, using the Setkey pulse introduced in phase I. Instead of performing 32 iterations for each pulse of the Setkey, 96 pulses were used, each corresponding to a single iteration of the setup task. With this full serial un-winding of for loop, we noticed a reduction in required FPGA slices, to 3,053 which fits the target FPGA, and occupies only 59% of its slices.

*F. Synchronizing the inputs with External Clock, and using Variable Key, and Single stage Encryption*

In order to efficiently perform the RC5 tasks, dedicated input clock signal must be used. For testing purposes, the input clock was divided internally, for slower operation, to be able to visually monitor the progress of the states of the design. Unifying all sequential input pulses into one global input clock and using internal counters to progress the design states properly, reduced the total number of required slices by 4%.

*G. Multiple Stage Pipelines with Synchronous Encryption*

As the target FPGA implementation of previous design was under utilized, the encrypt part of the design was optimized further to achieve faster performance. To the extent that is possible, the previous implementation was improved by taking advantage of CPU architectural advances such as pipelining [20]. A three stage pipeline is

TABLE IV  
RESOURCE UTILIZATION AND MAXIMUM DELAY FOR SOME RC5 DESIGN METHODOLOGIES

#	Design Methodology	FPGA	
		Utilization	* Longest Path Delay
1	Conventional RC5	207%	N.A..
2	A. Fixed Key Synchronous Encryption	11%	21.21 ns
3	E. Full Serial Variable Key Registration (Phase II)	59%	21.087 ns
4	F. Synchronizing the inputs with External Clock, and using Variable Key, and Single stage Encryption	55%	28.539 ns
5	G. Multiple Stage pipelines with Synchronous Encryption	70%	66.303 n

\*not necessarily the critical or dominant path

incorporated in the design, where three rounds of encryption are performed in sequence for each input clock pulse, provided that a Start signal is active. The total number of required slices of the FPGA reached 3,618, which is a 15% increase over the single stage encryption method, (i.e., roughly 7.5% increase per extra stage).

*H. Integrating Design for Test*

The previously presented design, though fits the target FPGA, does not provide means for tracking the number of pulses given to the encryption process. Two 7-Segment display, and internal hexadecimal counters, and encoders were incorporated to the design to improve its testing capability. A slight increase in the FPGA slices was required (i.e. 41 slices, less than extra 1%). This design was tested successfully in simulation before placing and routing it to the target FPGA prototype.

VI. IMPLEMENTATION AND EVALUATION

Shown in figure 7, simulation results of null plain text, with non-zero secret key (keyst), and a pre-known and properly matching cipher text of “B7C4B44A-9FAA44D8” after the 15 rounds (highlighted in the figure is the first word of the cipher text output). Also shown is the duration of the Start signal, which allows for only 5 changes for the cipher text. This is due to the fact that 3 rounds were executed per clock edge (in the 3 stage pipeline). Also shown in the figure the test signals which drive the seven-segments, which illustrate internal state modifications of the design.

The final design was implemented and mapped to the Virtex-II 1000 FPGA device (XC2V1000-4FG456C) in the 456 fine-pitch ball grid array package, see figure 8. The power consumption for the final design was found to be 351.3 mW of power.

The maximum delay for some of the designs which fitted into the Virtex-II -1000 FPGA is shown in Table 4. The operating frequency is 24 MHz which is provided by the Virtex-II prototype board (a clock period of about 42 ns). This clock period is suitable since the longest path delay of the final design methodology in Table 4 is

66.303 ns. However, 99% of the paths are less than half the clock period. Again, in the final design, one block cipher is produced every 5 clock cycles, while the Start signal is active. Since the block size is 64 bits, we can calculate the encryption throughput as in (3):

$$Throughput = \frac{(64)bits/block}{(5)cycles/block \times (\frac{1}{24 \times 10^6 cycles/sec})} \approx 300Mbps \quad (3)$$

This encryption throughput is higher than conventional and the related work in [3]. It was reported that 207 to 247 Mbps were achievable using the same FPGA family [3, 4]. Here, faster encryption throughput is believed to happen due to the integration of memory logic as part of the design, and the use of internal three stage pipelines. Doing away with external interfaces, improves overall performance.

Moreover, by increasing the clock frequency of the proposed design to 35 MHz, the encryption throughput can reach 450 Mbps using equation in (3). Thus, this work can yield up to an 80% increase over the encryption throughput of conventional architecture and related work.

VII. CONCLUSIONS

We have presented high performance RC5- integrated architecture with variable key registration, enhanced security and improved encryption throughput. The proposed architecture is synthesized to FPGA device similar to the family of related work for comparisons. The proposed architecture shows an improvement in the speed of operation as compared to the conventional architecture and related work.

Moreover, Comparing to conventional architecture, we show that we avoid damages caused by single site physical access attacks.

The deliverable encryption throughput of the proposed RC5-SoPC design is from 300 Mbps to 450 Mbps,

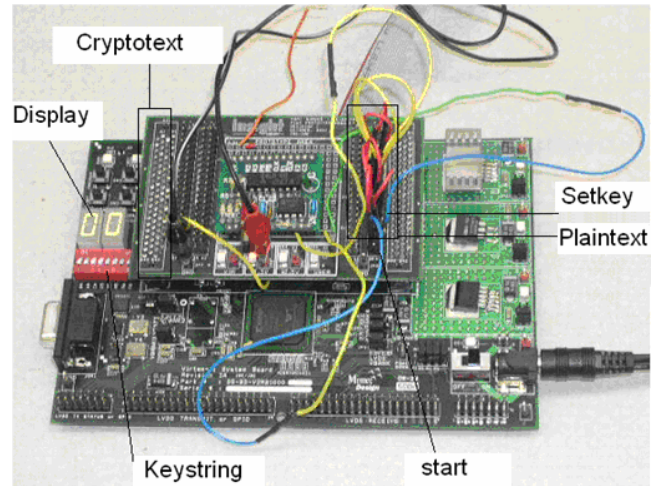


Figure 8: Hardware implementation of RC5 Encryption Algorithm

depending on the choice of the clock frequency (i.e. 24 MHz or 35MHz). This makes it suitable to high-speed networks (e.g. Fast Ethernet). Compared to conventional RC5 encryption throughput, we have shown an 80% increase in the achievable encryption throughput.

REFERENCES

- [1] Andrew Mason, "Network Security and Virtual Private Networks Technologies," Cisco Press, 2004.
- [2] K. Hausman, N. Alston, M. Chapple, Kalani K. Hausman, *Protecting Your Network from Security Threats*, Addison Wesley Professional, Nov. 2005.
- [3] Bassard, G., *Modern Cryptography*, Springer-Verlag, 1988.
- [4] Feistel, H., "Cryptography and Data Security," *Scientific American*, vol. 228, No. 5, pp. 15-23, 1973.
- [5] Coppersmith, D., "Cryptography," *IBM Journal of Research and Development*, vol. 31, pp. 244-248, 1987.

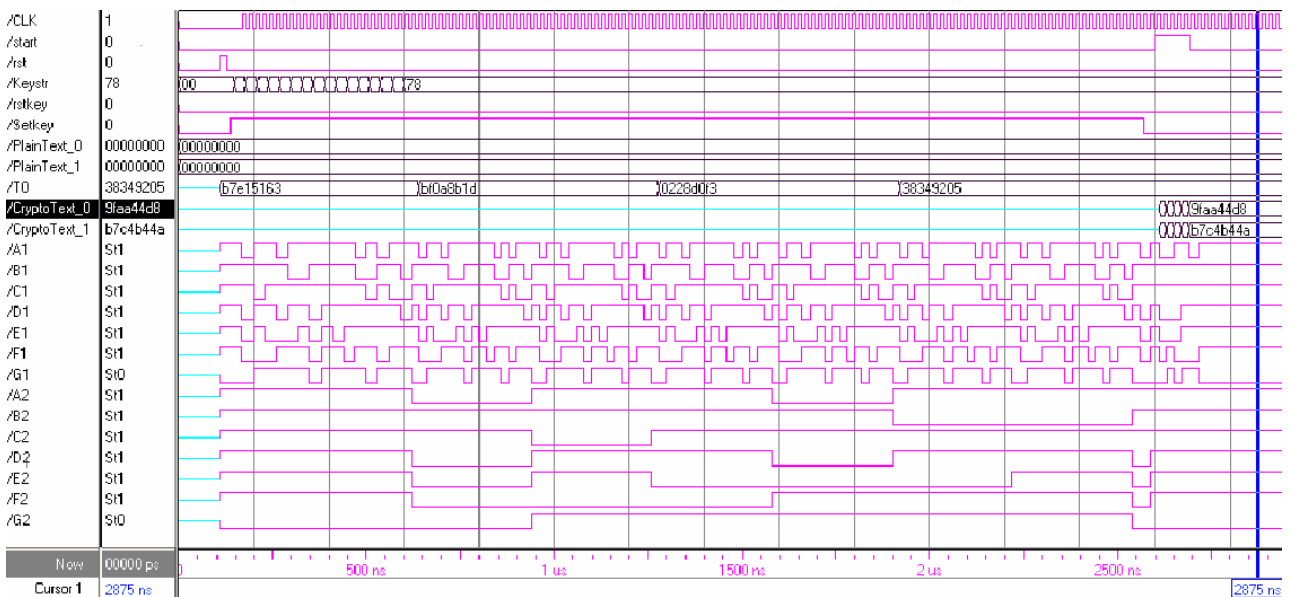


Figure 7: Simulation waveforms using blank plaintext for correct cipher output

- [6] Garry C. Kessler, "An overview of Cryptography," <http://www.garykessler.net/library/crypto.html#intro>, 2006
- [7] C. Meyer and S. Matyas, *Cryptography: A New Dimension in Computer Data Security*, John Wiley, New York, 1982.
- [8] Ronald L. Rivest, "The RC5 Encryption Algorithm", Proceedings of the 1994 Leuven Workshop on Fast Software Encryption (Springer 1995), pages 86-96.
- [9] Ronald L. Rivest, "A Description of the RC2 Encryption Algorithm," RFC 2268, 1998.
- [10] Burton S. Kalinski Jr., Yiqun Lisa Yin, "On the security of the RC5 Encryption Algorithms," RSA Laboratories Technical Report, September 1998.
- [11] R. Baldwin and R. Rivest, "The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms," RFC 2040, October 1996.
- [12] A. Schubert and W. Anheier, "Efficient VLSI Implementation of Modern Symmetric Block Ciphers", In the Proceedings of *THE ICECS'99*, Cyprus 1999. [http://www.rsasecurity.com/press\\_release.asp?doc\\_id=172&id=1034](http://www.rsasecurity.com/press_release.asp?doc_id=172&id=1034) (accessed 04/23/06)
- [13] N. Sklavos, C. Machas, O. Koufopavlou, "Area Optimized Architecture and VLSI Implementation of RC5 Encryption Algorithm", In Proceedings of the *IEEE ICECS '2003*, vol. 1, Dec. 2003.
- [14] C. Madson, and N. Doraswamy, "The ESP DES-CBC Cipher Algorithm with Explicit IV," RFC 2405, 1998.
- [15] F. Scott, M. Itsik, S. Adi "Weakness in the Key Scheduling Algorithm Of RC4", In Proceedings of *The 8th Annual Workshop on SAC*, August, 2001.
- [16] N. Sklavos and O.Koufopavlou, "Mobile Communications World: Security Implementation Aspects- A State of the Art", *Computer Science Journal of Moldova*, Institute of Mathematics & Computer Science, vol. 11 (2), 2003.
- [17] Menezes, A., Van Oorschot, P.C., Vanstone, S.A., *Handbook of Applied Cryptography*, CRC Press, 1997.
- [18] J. B. Sessions, "Fast Software Implementations of Block Ciphers," M.S. Thesis, Department of Electrical & Computer Engineering, Oregon State University, 1998.
- [19] US. National Bureau of Standards, "Data Encryption Standard," Federal Information Processing Standard (FIPS) publication 46-2, December 1993 <http://www.itl.nist.gov/fipspubs/fip46-2.htm>
- [20] Patterson D., and Hennessy J., *Computer Organization and Design: The Hardware/Software Interface*, Morgan Kaufmann Publishers, 1994.

1998 respectively. In 2004, he received the Ph.D. degree from the University of Missouri-Kansas City (UMKC) in Engineering and Networking disciplines.

While being at UMKC, he served as an adjunct faculty of electrical engineering department. In May 2004, after he received the Ph.D. degree, he joined the research team of Wideband Corporation, where he worked in the design and development of layer 3 network routers. In August 2005, he joined Tennessee Tech University as an Assistant Professor.

Dr. Elkeelany is a member of the Institute of Electronic and Electrical Engineers (IEEE), and the Eta Kappa Nu honorary Society. He has a distinguished educational record being the recipient of the UMKC Outstanding Doctoral Interdisciplinary Ph.D. Student Award in 2004, the UMKC Chancellor's Interdisciplinary Ph.D. Merit Award in 2001-2002 and the UMKC Outstanding Graduate Student Award from the School of Engineering, during 1999, 2000 and 2002. He received his B.Sc. degree with Distinction and degree of honor. In May 2005, Dr. Elkeelany received the Doctor of Research degree from the International Institute of Science and Technology.



**Omar S. Elkeelany** received the B.Sc. and M.Sc. degrees in Computer Science and Automatic Control from the University of Alexandria, Egypt 1992 and