Design and Analysis of Various Models of RC5-192 Embedded Information Security Algorithm

Omar Elkeelany

***Abstract*—**This article presents the design and analysis of various hardware reconfigurable models of RC5 Encryption algorithm. The original contribution herein is to determine the effects of loop-unrolling design concept on improving the encryption performance. We show how we determined the optimal design value of the number of unrolled loops to implement the RC5 algorithm using 192-bit encryption key. The various models tested were based on single-custom processor with no-loop-unrolling and with various sizes of loop unrolled implementations. In this research, various performance measures were considered. Namely, these were; the maximum frequency of operation, circuit size, throughput and energy consumption. To achieve proper comparison results, all models were implemented in the same hardware reconfigurable chip, a Field Programmable Gate Array (FPGA). The performance metrics of each model were evaluated to determine the best hardware model. Verilog hardware description language was used to model and test all implementations.

Results revealed that while no-loop-unrolling provided the least circuit size, the 3-loop-unrolled approach provided the highest encryption throughput. Further, a throughput speed up of 24% was achieved as compared to a reference system implemented in a similar target device using a Xilinx FPGA family. Comparing our implementations on the same Altera FPGA family, a maximum throughput speed up of 50% was achieved.

These results provide a much better ground for applications involving high performance embedded data security, such as in military communications, nuclear digital instrumentation and control, and portable biomedical devices.

***Keywords*—** Cryptography, FPGA design and analysis, RC5 encryption, Loop-unrolling

I. Introduction

Secure data transmission over unreliable medium is continuously gaining higher importance. It demands improvements in the performance of existing encryption algorithms. This is particularly needed in a wide range of applications including: virtual enterprise security [1] [2], portable network devices [2], and visual cryptography [3]. Even though the software implementation of encryption algorithms has the advantages of portability, flexibility, and ease of use, it provides a limited physical security and agility compared to hardware implementations. Major advantages that lead to the hardware implementation include less power consumption, small circuit size, hardware reconfigurability, cost efficiency, high operating speed and security.

Manuscript received October 30, 2008: Revised version received December 16, 2008. This work was supported in part by the Center of Manufacturing Research and the Department of Electrical and Computer Engineering at Tennessee Technological University.

Omar Elkeelany is Assistant Professor of Electrical and Computer Engineering at Tennessee Technological University, Cookeville, TN 38505 USA (phone: 931-372-3677; fax: 931-372-3436; e-mail: OElkeelany@ TnTech.edu).

Symmetric cryptosystems are based on algorithms in which identical keys are used for encryption and decryption [4]. The secret key used for encryption/decryption should be known only to the legitimate senders and receivers in order to protect data. Symmetric key algorithms can be further divided into block ciphers for fixed transformations on plain-text data, and stream ciphers for time varying transformations.

Block ciphers are the most basic type of ciphers and operate on the principle of encrypting/decrypting fixed size blocks. The size of the block is algorithm specific. For example, the Advanced Encryption System (AES) operates on a block size of 128 bits, Data Encryption Standard (DES) [5] works on a block size of 64 bits. But, Rivest Cipher 5 (RC5) does not have a fixed block size; it can be 32, 64, or 128 bits. That's why the Wireless Application Protocol (WAP) forum for example, specifies RC5 as its encryption algorithm for its Wireless Transport Level Security (WTLS) clients and servers [6].

Any particular RC5 algorithm is represented with the notation of RC5-w/r/b, where w/r/b are reconfigurable parameters. W is the word size in bits, r signifies the number of rounds and, b signifies the number of bytes in the secret key. The parameters w/r/b are configured such that the algorithm gives maximum security [7].

RC5 was originally developed for the software implementation, but fits hardware implementation as well [8]. In general ciphers implemented in software are not efficient when compared to their implementations in hardware [9]. The hardware implementation of the RC5 algorithm can be done using different approaches.

Various research efforts exploited the System on Chip (SoC) implementation or hardware implementation of RC5 algorithm to provide enhanced security architecture when compared with the conventional architecture [10] [11]. Enhanced performance measures include: less power consumption, allocation of resources, re-configurability, architecture efficiency and cost efficiency [12].

The potential features of Field Programmable Gate Arrays (FPGA) implementation is that it allows SoC modeling. The performance evaluation of different hardware models of RC5 algorithm will be done with FPGA as the target technology.

As stated before, research has exploited RC5 hardware implementations for many advantages such as less power consumption, reduced circuit size, reconfigurability, cost efficiency, improvement in operating speed, and gaining extra security [13], [14]. The work of Skalvos [8], [15], [9] proposed hardware architecture of the RC5 core into a FPGA device with fewer resources than the conventional implementation by introducing the use of shared resources. However, the encryption throughput it reported was found less than that of the conventional architecture [8]. In cases where speed is more desirable than circuit size, more research is needed to investigate design choices to achieve the best possible encryption rate without demanding a huge increase in the circuit size. Specifically, it is not obvious how loop unrolling design choice affects the performance of the algorithm as well as the effect on circuit size.

This paper also studies the effect of the loop unrolling technique, on implementing RC5 on reconfigurable hardware. Loop unrolling is a used generally in system design to improve throughput and optimize critical parts of the system by duplicating hardware components [16].

The rest of the paper is organized as follows. Section 2 provides an overview of the RC5 encryption algorithm. Section 3 presents design methodologies with focus on loop-unrolling technique. Section 4 provides synthesis results and analysis. The paper then concludes in section 5, with insight to future work.

II. RC5 Algorithm Overview and Conventional Architecture

As mentioned briefly in section 1, RC5 is a parameterized symmetric encryption algorithm. RC stands for “Rivest Cipher”, or alternatively, “Ron’s Code” [7]. RC5 parameters are; a variable block size, a variable key size and a variable number of rounds. Allowable choices for the block size 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. It is the latest in a family of secret key cryptographic methods; RC5 is more secure than RC4 [17] but is slower. 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 [9].

The RC5 algorithm uses three primitive operations and their inverses. These are:

(1) Addition/subtraction of words modulo 2W, where w is the word size.

(2) Bit-wise exclusive-OR of words denoted by XOR.

(3) Rotation: the rotation of word x left by y bits is denoted x<<<y. The inverse operation is the rotation of word x right by y bits, denoted by x>>>y.

In the key expansion module, the password key K is expanded to a much larger size in a key table S.

The size of table S is 2(r+1), where r is the number of rounds [7].

The encryption process takes a plaintext input and produces a cipher-text as the output. The key-expansion process must have already been performed before this process. The decryption process takes a cipher-text as the input and produces a plaintext as the output. In general, the same plaintext block will always encrypt to the same cipher-text when using the same key in a block cipher whereas the same plaintext will encrypt to different cipher-text 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 fig. 1, performs the encryption and decryption tasks in two separate cores. The RC5 Core, shown in the figure, needs to read the expanded key, in a sequential way, in order to encrypt the plain text given to the core.

As shown in the figure, 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 un-authorized users, if they have access to the system to tap and record the memory contents. Later 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 user does not intend to change the secret key very often. Our 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 loop-unrolling technique in its design as discussed in section 3.

One can use the following equation to calculate encryption throughput:

$$Th(Mbps) = \frac{b}{I * C} = \frac{b * MIPS}{I} \quad , \qquad (1)$$

where b is the block size in bits, I is the number of software instructions needed per block and C is the cycle time (in micro seconds). It is noteworthy that using this equation, one finds that other encryption algorithm (i.e. 3DES algorithm) may have an encryption throughput of 4 Mbps at 500 MIPS

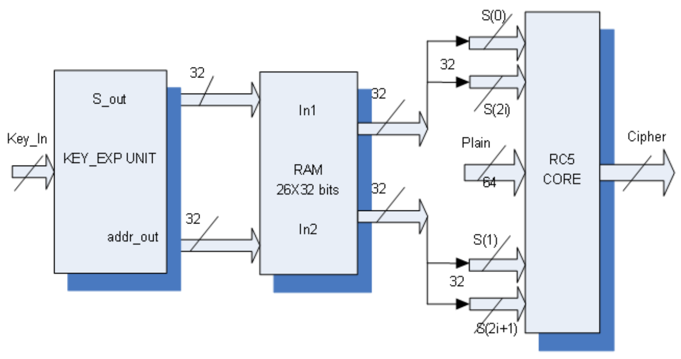


Fig. 1 The Conventional Architecture of the RC5

(million instructions per second) rate [19].

Table 1 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(1), this yeilds a throughput of 200 Mbps (Th=64*500/160). It is clear from this comparison that RC5 outperforms more complicated encryption algorithms (of equal key length) operating at the software level. In section 3, we will show that use of a three loop-unrolling technique at the hardware level on RC5 can improve this throughput at much slower processing rates.

III. Design Methodologies

Schematic based design methodology is a conventional hardware design approach that is been recently replaced by hardware description language based (HDL) methods. As the architectural complexity increases, schematic design methodology becomes no longer a feasible technique [12]. Coming to the language based design tools, which are solely dependent on the synthesis tools, a rapid improvement in synthesis tools has resulted in choosing the HDL based design methodology. In this research, Verilog HDL is used to model the various RC5 design methods.

Herein, the two investigated hardware models are:
1. Soft-core general purpose processor based and
2. Single-custom processor based

The soft-core general-purpose processor based model yields a programmable device that can be used in a variety of applications. Hence, it offers high flexibility, but typically at the cost of design size, and power consumption.

The single-custom processor based model, on the other hand, executes a single program, or has a custom hardware to perform a single function. It is used to achieve low circuit size, low power, and high performance designs. It does not have a programming memory since its function is inherently

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

Table 1 RC5 operations per block

integrated in the design. Hence, called single functioned.

In order to evaluate the performance of these two hardware models under investigation, there must be certain parameters to be measure in each model. The parameters that best suit to evaluate the performance of these hardware models are: maximum frequency of operation, resource utilization, power/energy consumption, encryption throughput, and cost efficiency.

After acquiring all the parameters, the two models were compared to determine the best hardware implementation of the RC5 algorithm.

A. Design Implementations

The first model is designed by using a soft-core general purpose processor. The Nios II soft processor is selected due to its ample features and flexibility; and also it is the most popular soft microprocessor available in the market today [20] [21]. As Nios II is a soft processor defined in HDL, it can be implemented in the FPGA by using the Quartus II design tool [21]. The System-on-a-programmable-chip (SOPC) Builder software [22] was used to add the necessary functional units such as memories, I/O interfaces and timers to the Nios II processor. It is well known that a general purpose processor by itself won't be a useful system [22].

Writing software for this processor is similar to other micro-controller family. The Nios II integrated development environment (IDE) interface is used to write the software for this processor. The hardware abstraction layer (HAL) of the Nios II software serves as a board-support package for Nios II processor systems. The tight integration between SOPC Builder and Nios II IDE allows the HAL system library to be generated automatically. The HAL system library provides a hosted C runtime environment based on the ANSI C standard libraries. It also provides generic I/O devices, allowing user to write programs that access hardware using the C standard library routines. After SOPC Builder generates a hardware system, the Nios II IDE can generate a custom HAL system library to match the hardware configuration [21]. RC5 algorithm that is available in C was used to implement the same in the Nios II and modifications were made to suit for the Nios II processor environment.

The second model is implemented using a bottom-up design methodology in Verilog HDL with Quartus II as the synthesis tool. The model was exhaustively tested using ModelSim simulations. The results obtained after the simulation are compared with the results of RC5 algorithm written in C language.

For both the models the registration of the secret key into the system was done with a sequence of short pulses or in parallel feed depending on the device support.

B. RC5 Loop Unrolling Discussion in Single Custom Processor

As mentioned earlier, loop unrolling is a technique used in system design to improve the throughput and optimize its critical parts. In computer programming languages, loop unrolling mechanism is done to instructions that are called in multiple iterations of the loop by combining them into a single iteration [16].

We propose the use of this technique to improve the throughput of the RC5 core. Specifically, we use this technique to unroll the r rounds of encryption task into duplicate hardware components. In no-unrolling approach every block of data requires a number of (Cc) clock cycles to complete the entire encryption process.

If a round of encryption task is done for every clock cycle, so to perform r rounds of encryption task, a circuit needs r clock cycles. Instead of performing the encryption task once per clock cycle we can use loop-unrolling approach to cut short the number of cycles required. The unrolling of r rounds of encryption task can be done in multiple ways by considering the integer factors of r. When r=15, we can have a total of 5 cycles with 3 loops-unrolled and performed per clock cycle. Or, we can have 3 cycles with 5 loops unrolled per cycle. Finally, we can have 15 loops of encryption done in a single clock cycle. When compared to no-loop-unrolling approach, the first approach requires only five clock cycles, which means cutting the number of clock cycles required to one third. Similarly second approach requires five times less clock cycles and the third approach requires r times less clock cycles for encrypting a block of data. Fig. 2 shows the pictorial view of the loop unrolling technique, for number of rounds, r=15. The penalty for this approach will be the increase in chip size. This is because, when a loop unrolling is implemented in hardware, the unrolled loop of the task is duplicated as extra hardware on the chip. So with the increase in number of unrolled loops, the size of hardware increases.

Loop-unrolling requires more logic elements compared to the no-loop-unrolling. This will be seen in detail in the next section.

Cycle #														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Single Custom Processor Approach - One Round Per Cycle Round #														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3 Loops Unrolled Approach - 3 Rounds Per Cycle Round #														
1	4	7	10	13	1	4	7	10	13	1	4	7	10	13
2	5	8	11	14	2	5	8	11	14	2	5	8	11	14
3	6	9	12	15	3	6	9	12	15	3	6	9	12	15
5 Loops Unrolled Approach - 3 Rounds Per Cycle Round #														
1	6	11	1	6	11	1	6	11	1	6	11	1	6	11
2	7	12	2	7	12	2	7	12	2	7	12	2	7	12
3	8	13	3	8	13	3	8	13	3	8	13	3	8	13
4	9	14	4	9	14	4	9	14	4	9	14	4	9	14
5	10	15	5	10	15	5	10	15	5	10	15	5	10	15

Fig. 2 Mechanism of Loop Unrolling (no unrolling, 3 loops unrolled, 5 loops unrolled). Total number of loops =15.

IV. Verification, Synthesis and analysis results

Both investigated models were designed, simulated and synthesized using Altera Quartus II development tool. The simulation results were observed for the functional correctness of the algorithm. Fig. 3 shows the simulation result of Single custom model of RC5. After the functional verification is done the code was synthesized, placed and routed, and re-simulated to check whether the implementation is successful or not. The two models were implemented using the same hardware target [12].

The selection of a target device depended on several requirements like available clock speeds, number of I/O pins, on chip memory, display interfaces (LEDs, LCD), etc. The DE2 development board, which has the Cyclone II EP2C35F672C6 FPGA and EPCS16 serial configuration device, is selected as the target device. The Cyclone II device family has the features of high-density architecture with 4,608 to 68,416 logic elements, embedded multipliers, advanced I/O support, flexible clock management circuitry, etc [23].

We used Verilog HDL to implement both conventional (for comparison) and the proposed architectures. Various code implementations were synthesized using Altera FPGA development tools. The result obtained after modeling the algorithm tallies with the result obtained from the original code in C, which models the conventional architecture. Fig. 4 shows the circuit used for providing plain-test input, and Fig. 5 shows the target FPGA board with input test circuit. Fig. 6 shows two samples of in-circuit verification for plain-text input 0x98721827-BE7B1E6F provided to RC5-32/15/16. An output is verified on the seven-segment display to show valid values of 0xD56280F6 as first word and 0xE836330E on the second.

A. *Throughput Analysis of Embedded General Purpose Processor*

For throughput analysis, we started with a general purpose processor model embedded on the FPGA and implemented the RC5 in a high level programming language (C code). An embedded processor (NIOS II/s) system was used to execute the C code, which was almost same for both RC5-32/15/16 and RC5-32/15/24 versions. We then inserted timing calculation functions to measure throughput. A 50 MHz clock was the on-board system clock, so the maximum operating frequency could not exceed 50 MHz. Table 2 summarizes the resource utilization of the general purpose processor model. This model implementation utilizes multipliers, random access memory (RAM), and PLL blocks. The embedded multipliers are used to speed up multiplier intensive applications. The processor need for memory is satisfied by employing embedded memories, which consists of M4K (4Kbit) blocks that can be configured to provide various memory functions like RAM, first-in-first-out (FIFO), or read only memory. In addition, potential timing problem can rise from clock skew,

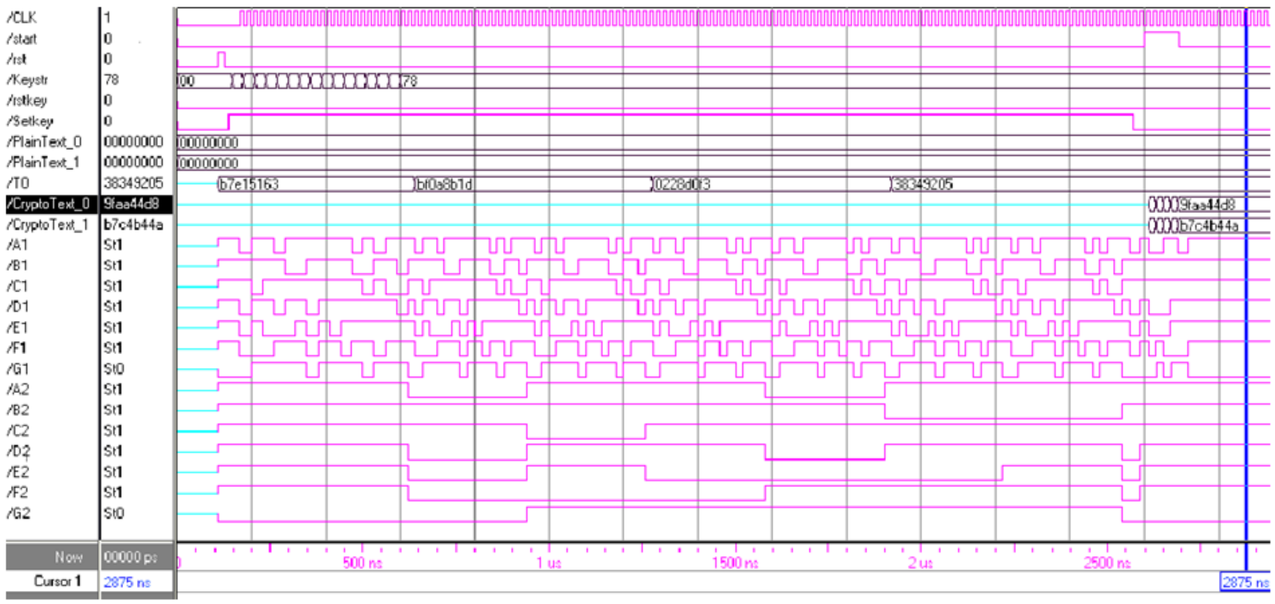


Fig. 3: RC5 Simulation waveforms using blank plaintext for known correct cipher output of 0xb7c4b44a-9faa44d8

which leaves external memory (i.e., SDRAM) inaccessible. Hence, Phase Lock Loop module (PLL) was used.

Throughput calculation of this model involves the calculation of number of cycles required to perform the encryption operation. High resolution Altera timing functions were used. Of these functions available, *alt_timestamp_start()* function can be used start the counter and a call to *alt_timestamp()* will provide the value of timestamp counter. These two functions were used to obtain the number of clock pulses required to complete the encryption process.

The number of cycles required was found to be 1529 for both the versions of RC5. Using Fmax of 50 Mhz, Throughput was calculated using:

$$Throughput = Fmax * (64/Cc). \qquad (2)$$

Fmax is the maximum frequency of operation, 64 is the block size, and Cc is the number of clock cycles required to encrypt one block. So,

$$Throughput = 50x10^6 * (64/1529)$$
$$=2.1\ Mbps$$

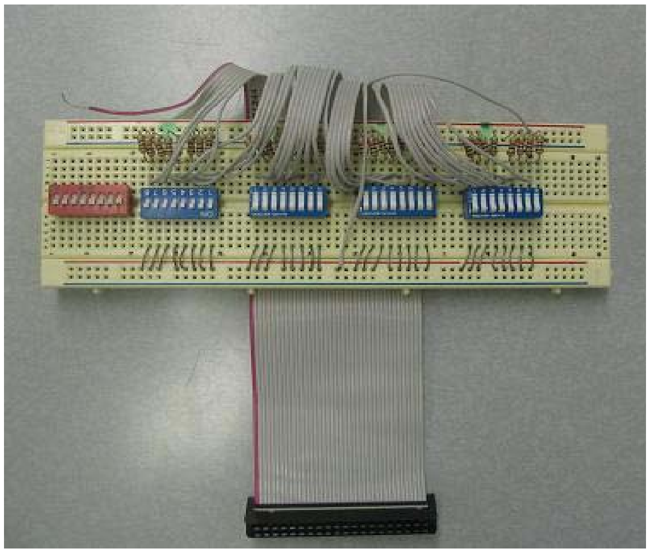


Fig. 4: Parallel input test circuit for in-circuit verification

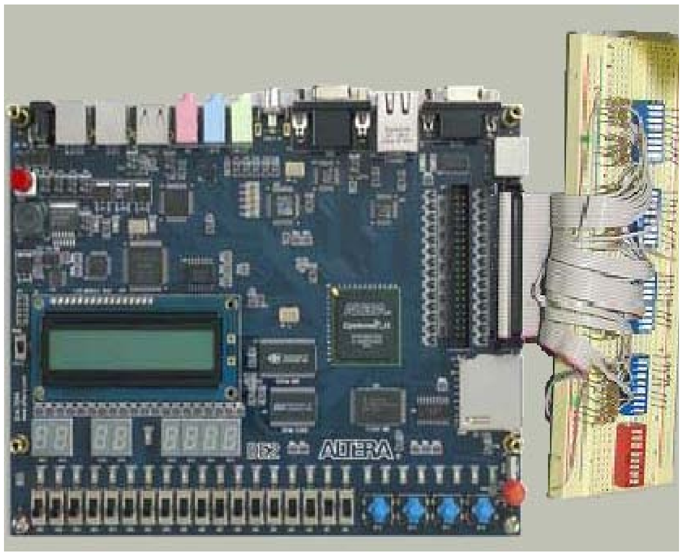
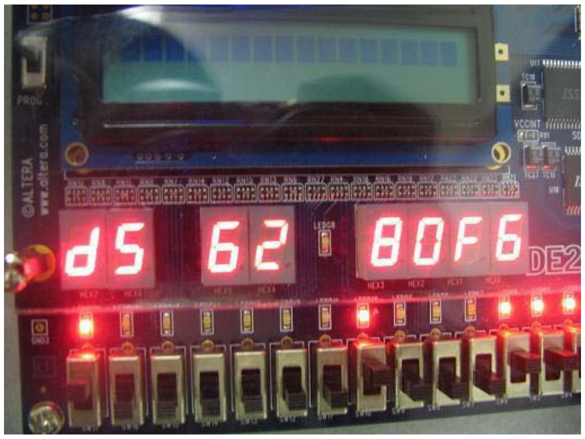
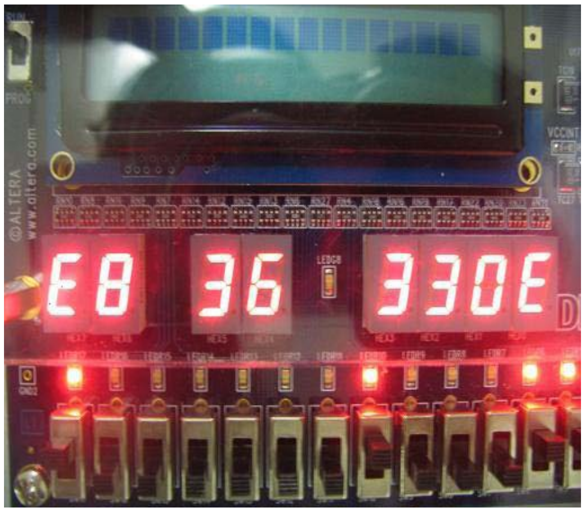


Fig 5: Target FPGA board, with input test circuit



(a)



(b)

Fig 6: Correct in-circuit verification for plain-text input 0x98721827-BE7B1E6F provided to RC5-32/15/16. An output is displayed sequentially on (a) first word (b) second

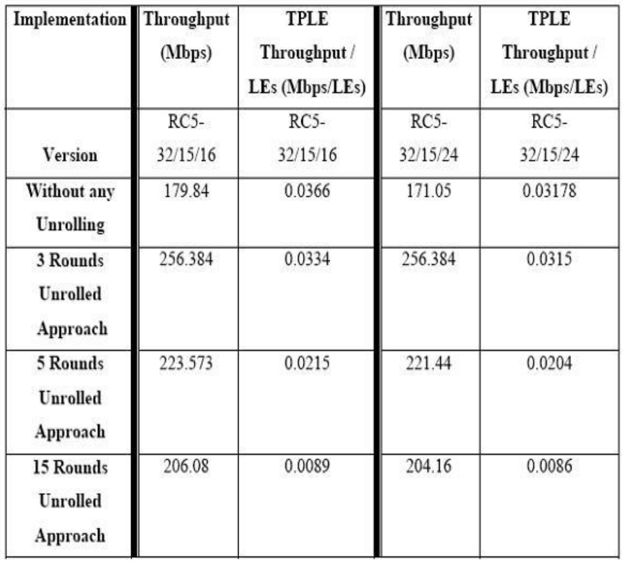
FPGA Resource	Utilization
Logic elements	3,331 / 33,216 (10 %)
Registers	1768
Combinational Elements	1237
Logic array blocks	319 / 2,076
I/O pins	365 / 475
Clock pins	8 / 8
Total fan-out	19121
Average fan-out	3.34
Embedded multipliers	4 / 70
PLLs	1 / 4
M4Ks	15 / 105 (14 %)
Total RAM block bits	69,120 / 483,840

Table 2: Resource utilization for general-purpose processor

When we compare this result with discussion presented in section 2, we find that this embedded processor runs on roughly 5 MIPS, which is so slow to perform high performance. However, this solution offer fastest time to market design, and minimal design cost. The extra resources like multipliers, RAM and PLL are not major contributors to cost, especially in bulk production. The bottom line is; the performance is poor at the advantage of low cost.

B. Throughput Analysis of Single Custom Processor

As explained in section 2, RC5 is a block cipher that works on two-word input and two-word output blocks. The number of clock cycles required for encryption is dependent on the parameter r. No-loop unrolling (conventional) implementation needs 15 clock cycles for encrypting a data block of 64 bits (two words). The throughput is calculated by (2). As the number of loops unrolled increase, it is intuitive that the longest path delay of the circuit increase as more components are connected in sequence, and fewer resources are shared iteratively. Consequently, the maximum clock frequency of operation decreases. At the same time, circuit size increases, since more components are needed. But, as the maximum clocking frequency decreases, the total number of cycles needed is also reduced. Hence, it is not obvious, how does the size of unrolling (the number of unrolled loops) affect the overall throughput of the system. There exists an optimal value which achieves the highest rate of encryption (i.e., the highest throughput). In this section we study the effect of loop-unrolling on RC5 by developing multiple Verilog HDL implementations to a same target, Cyclone II- EP2C35F672C6 FPGA device (for fair comparisons). Fig. 7 summarizes the throughput for RC5 implementations using no-unrolling and the loop-unrolled implementations and is used in calculating Throughput per Logic Elements (TPLE). TPLE is calculated by dividing the throughput with its respective number of Logic Elements (LEs) required for each implementation. The total cost of implementation is proportional to the number of LEs. Because, if the number of LEs required increases then the total area occupied by the circuit will increase, resulting in increase in production costs. So we have to select an implementation that gives optimum performance and requires least number of LEs. TPLE is a measure of the circuits cost. If the TPLE is more, it indicates that the logic elements of the implementation are used efficiently to increase the throughput. Table 3 compares the FPGA resource utilizations of both RC5 models based on single-custom processor and general-purpose processor.



Implementation	Throughput (Mbps)	TPLE Throughput / LEs (Mbps/LEs)	Throughput (Mbps)	TPLE Throughput / LEs (Mbps/LEs)
Version	RC5-32/15/16	RC5-32/15/16	RC5-32/15/24	RC5-32/15/24
Without any Unrolling	179.84	0.0366	171.05	0.03178
3 Rounds Unrolled Approach	256.384	0.0334	256.384	0.0315
5 Rounds Unrolled Approach	223.573	0.0215	221.44	0.0204
15 Rounds Unrolled Approach	206.08	0.0089	204.16	0.0086

Fig. 7 Throughput per Logic Elements for various RC5 implementations

In other words, the higher the TPLE the higher the cost efficiency will be. Even though the circuit is cost efficient there is no guarantee that performance of the circuit will be good. So, we should consider the performance as well as the cost efficiency into consideration when choosing any approach to implement on hardware. Fig 7 indicates that 15-loops unrolled approach has the least TPLE, which means that its cost will be high, whereas the no-unrolling approach and 3-rounds unrolled approach indicate less cost among the listed implementations. Obviously, the cost of manufacturing no-unrolling is less due to its small circuit size. However, no-unrolling implementation provides lower throughput when compared with the 3-loops unrolled approach. So, 3-loops unrolled approach provides a better alternative as it meets the throughput requirements at an affordable cost. Fig. 8 shows the throughput vs. the number of loops unrolled for RC5 16 and 24 byte key sizes. As shown from the figure, the best throughput (highest) is associated with the rightmost horizontal side of the graph. It is clearly shown that 3-loops unrolled outperforms 5 and 15-loops unrolled. The reason behind that is the extra increase in the propagation delay of higher unrolled loops, downgraded the value of Fmax by a rate grater than the improvement of reduced value of required clock cycles *Cc*, see (2). To illustrate how this was in effect, table 4 shows the longest path delay and the respective value of Fmax on the four loop-unrolling models for both versions of RC5 encryption 32/15/16 and 32/15/24. The value of *Cc* parameter required for these models are 15, 5,3 and 1 respectively.

In addition, the achieved encryption throughput of 3-loops unrolled is higher than that reported of the related work of 207 Mbps using a Xilinx VirtexII-1000 FPGA device [8]. Hence, 3-loops unrolled offered a maximum throughput improvement of 24% compared to the work related. Comparing implementations on the same Altera Cyclone II-EP2C35F672C6 FPGA, a maximum throughput speed up of 50% is achieved for 192 bit key. (i.e., comparing throughput speedup of 3-loops unrolled to no-unrolling in Fig. 7)

FPGA Resources	**Single-Custom Processor**	**Soft-core Processor**
Registers	1299	1974
PLLs	0	1
Total memory bits	0/483,840	46592/483,840
Logic array blocks	330/2076	319/2076
I/O pins	307/475	365/475
Clock pins	5/8	8/8
Maximum fan-out	1500	1906
Total fan-out	24520	20469
Average fan-out	3.68	3.77
Embedded multipliers	0/70	4/70
Logic elements	5050/33216	3184/33216

Table 3: Resource utilization comparison of single purpose (SCP) and soft-core general-purpose processor (GPP)

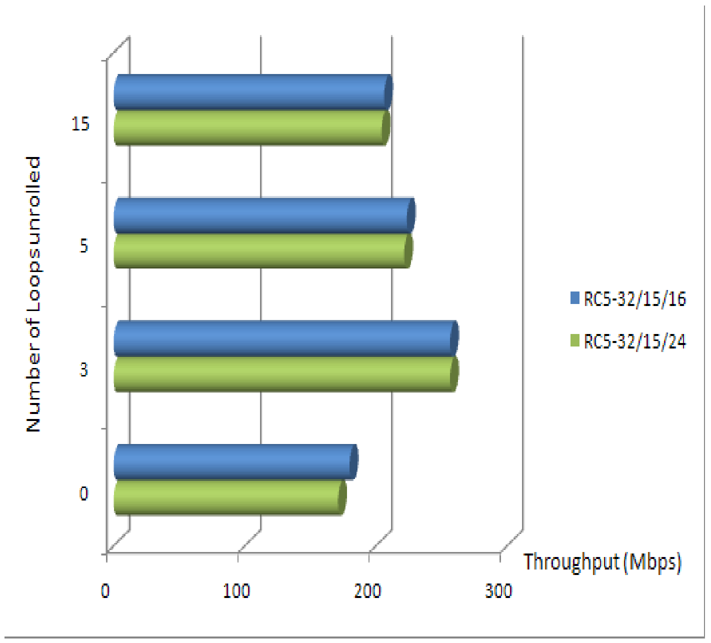


Fig. 8 Throughput vs. Number of Rounds Unrolled in RC5 implementations using 128 and 192 bit key

Parameter	RC5-32/15/16	RC5- 32/15/24
Fmax	42.15 MHz	40.09 MHz
Longest Path Delay (or clock period)	23.72 ns	24.941 ns
Worst Case Setup (t_{su}) and Hold (t_h) times	t_{su} = 8.519 ns t_h = 4.206 ns	t_{su} = 9.605 ns t_h = 4.198 ns

(a)

Parameter	RC5-32/15/16	RC5- 32/15/24
Fmax	20.03 MHz	20.03 MHz
Longest Path Delay (or clock period)	49.92 ns	49.92 ns
Worst Case Setup (t_{su}) and Hold (t_h) times	t_{su} = 9.920 ns t_h = 4.104 ns	t_{su} = 9.920 ns t_h = 4.104 ns

(b)

Parameter	RC5-32/15/16	RC5- 32/15/24
Fmax	10.48 MHz	10.38 MHz
Longest Path Delay (or clock period)	95.41 ns	96.33 ns
Worst Case Setup (t_{su}) and Hold (t_h) times	t_{su} = 9.556 ns t_h = 4.616 ns	t_{su} = 9.110 ns t_h = 4.695 ns

(c)

Parameter	RC5-32/15/16	RC5- 32/15/24
Fmax	3.22 MHz	3.19 MHz
Longest Path Delay (or clock period)	310.55 ns	313.47 ns
Worst Case Setup (t_{su}) and Hold (t_h) times	t_{su} = 13.224 ns t_h = 4.815 ns	t_{su} = 13.424 ns t_h = 4.739 ns

(d)

Table 4 Timing Analysis of (a) no loop unrolling (b) 3, (c) 5, (d) 15 loop unrolling respectively

C. *Energy Evaluations of Single Custom Processor*

An estimate of power consumption can be made with FPGA development tools after the synthesis and place-and-route phases. For Altera FPGA devices, the PowerPlay power analyzer tool performs after synthesis power estimation. It calculates a close estimate for power consumption by using inputs from the resource utilization phase (Fitter report), signal activities from the functional simulation and operating condition of the design (junction temperature and board cooling solution settings). One of these features is the toggle rate, which is how often the output changes with respect to the input clock signal [24]. Table 5 summarizes the power and energy consumption generated with a toggle rate of 25% (assuming that toggle rate is same for all implementations). It is shown from the table that the no-loop-unrolling consumes the most energy and that the 3-loops unrolled consume the least energy. This is due to the fact that it experiences the minimum encryption time per block (or highest throughput).

V. Conclusion

We presented two hardware models of the RC5 algorithm and from the summary of the results we can observe both the models have their advantages and disadvantages.

In this paper, various hardware implementations of RC5 algorithm were presented using variable sizes of loop unrolling technique, and implemented on FPGA. The performance evaluation suite involved calculating maximum frequency of operation, circuit size (in terms of the number of Logic Elements, power consumption, throughput computation, and cost efficiency. With the aid of performance evaluation

Approach	Power (mW)	Energy (nJ)
No-unrolling	138.27	51.728
3 loop-unrolling	140.0	34.944
5 loop-unrolling	141.79	40.975
15 loop-unrolling	145.82	45.71

Table 5 Power and Energy consumption for various RC5 models under test

tools, it is concluded that, of all the implemented models, 3-loops unrolled approach can be selected to achieve highest throughput. However, only if the user requirements are such that the model should fit in least circuit size with moderate throughput, the no-loop-unrolled can be used.

The achieved encryption throughput of 3-loops unrolled is higher than that reported of the related work of 207 Mbps using a Xilinx VirtexII-1000 FPGA device [8]. Hence, 3-loops unrolled offered a maximum throughput improvement of 24% compared to the work related. Comparing implementations on the same Altera Cyclone II- EP2C35F672C6 FPGA, a maximum throughput speed up of 50% is achieved for 192 bit key.

It was shown that, when we integrated the maximum operating frequency (Fmax) into the required number of cycles to operate, the soft-core general-purpose processor was found to be less than 100 times slower than the single custom processor (although it operates at higher frequency). Even though the second model is faster there is price paid for it. It incurs extra time of design, and longer time to market. However, when we compare the resource utilization of the two models, we show that the single custom processor generally uses a lot less resources in terms of multipliers, memory, PLLs, registers etc. So, in high volume production, it can also be cheaper than the first model. Since the resource utilization is less for the second model the manufacturing cost will be less.

In future work, the author would like to investigate the effect of loop-unrolling of other encryption algorithms such as RC5 [25], and AES.

ACKNOWLEDGMENT

The author would like to acknowledge the Masters students: Suman Nimmagadda and Adegoke Olabsi for their thorough research work related to this paper. The author also acknowledges the Center for Energy Systems Research and The Department of Electrical and Computer Engineering at Tennessee Technological University for supporting this research.

REFERENCES

[1] M. M. Zanjireh, A. Kargarnejad and Tayebi, M. A. “Virtual Enterprise Security: Importance, Challenges and Solutions.” WSEAS Transactions on Information Science & Applications, volume 4, no. 4, pages 879–884, 2007.
[2] Lee, Tsang-Yean and Lee, Huey-Ming. “Encryption and Decryption Algorithm of Data Transmission in Network Security.” WSEAS Transactions on Information Science & Applications, volume 3, no. 12, pages 2557–2562, 2006.
[3] Chao, Kun-Yuan and Lin, Ja-Chen. “Fault-Tolerant and Non-Expanded Visual Cryptography for Color Images.” WSEAS Transactions on Information Science & Applications, volume 3, no. 11, pages 2184–2191, 2006.
[4] Schubert, A. and Anheier, W. “Efficient VLSI Implementation of Modern Symmetric Block Ciphers.” In “The ICECS99,” , 1999.
[5] G. Rouvroy, J. J.-Quisquater, F. X.-Standaert and D.-Legat, J. “Efficient uses of FPGAs for implementations of DES and its experimental liner cryptanalysis.” IEEE Transactions on Computers, volume 52, no. 4, pages 473–482, 2003.
[6] RSA Security. “RSA Security Algorithm.” URL http://www.rsasecurity.com/press release.asp?doc-id=172&id=1034 (Accessed: April 23, 2006)
[7] L.-Rivest, R. “The RC5 Encryption Algorithm.”In “The 1994 Leuven Workshop on Fast Software Encryption (Springer 1995),” pages 86–96, 1994.
[8] N. Sklavos, C. Machas and Koufopavlou, O. “Area Optimized Architecture and VLSI Implementation of RC5 Encryption Algorithm.” In “The IEEE ICECS 2003,” volume 1, December 2003.
[9] Sklavos, N. and Koufopavlou, O. “Mobile Communications World: Security Implementation Aspects- A State of the Art.” Computer Science Journal of Moldova,Institute of Mathematics and Computer Science, volume 11, no. 2, 2003.
[10] Olabisi, A. System on Chip Architecture for RC5 with Enhanced Security. Master’s thesis, May 2006.
[11] N. Sklavos, K. Touliou and Efstathiou, C. “Security & Privacy Architectural Modeles: On the Hardware & Software Integration Platforms.” WSEAS Transactions on Information Science & Applications, volume 3, no. 5, pages 965–971, 2006.
[12] A.J. Elbirt, B. Chetwynd, W. Yip and Paar, C. “An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists.” In “The AES Candidate Conference 2000,” pages 13–27, 2000.
[13] Olabisi, A. and Elkeelany, O. “Integrated design of RC5 algorithm.” In “The IEEE 39th Southeastern Symposium on System Theory,” , 2007.
[14] Nimmagadda, S. and Elkeelany, O. “Performance evaluation of different hardware models of RC5 algorithm.” In “The IEEE 39th Southeastern Symposium on System Theory,” , 2007.
[15] N. Sklavos, A. P.-Fournaris and Koufopavlou, O. “WAP Security: Implementation Cost and Performance Evaluation of a Scalable Architecture for RC5 Parameterized Block Cipher.” In “The IEEE Mediterranean Electrotechnical Conference (IEEE MELECON’04),” , May 2004.
[16] Ken, K. and Randy, A. Optimizing Compilers for Modern Architectures: A Dependence-based Approach- Loop unrolling technique and its advantages and disadvantages. A Morgan Kaufmann, 2001.
[17] F. Scott, M. Itsik and Adi, S. “Weakness in the Key Scheduling Algorithm of RC4.” In “The 8th Annual Workshop on SAC,” , August 2001.
[18] Sessions, J. B. Fast Software Implementations of Block Ciphers. Master’s thesis, 1998.
[19] Elkeelany O., and Olabisi A., "Performance Comparisons, Design and Implementation of RC5 Symmetric Encryption Core." Journal of Computers, no 1, pages 48-55, 2008.
[20] Altera Corporation. “Nios Processor.” URL http://www.altera.com/products/ip/processors/nios2/ni2-index.html (Accessed: April 23, 2006)
[21] Altera Corporation. “Nios II Processor.” URL http://www.altera.com/literature/hb/nios2/n2cpu nii51004.pdf# (Accessed: April 23, 2006)
[22] Altera Corporation. “SOPC Design Tool.” URL http://www.altera.com/education/univ/materials/manual/labs/tut sopc introduction verilog.pdf (Accessed: April 23, 2006)
[23] Altera Corporation. “DE2 user manual.”, 2006.URL http://www.altera.com/
[24] Xilinx Inc. “How to calculate toggle rate.” URL http://www.xilinx.com/ise/powertools/wpt help/app docs/calculating toggle rates.htm (Accessed: April 23, 2006)
[25] Riaz, M. and M.-Heys, H. “The FPGA Implementation of the RC6 and CAST-256 Encryption Algorithms.” In “The IEEE Canadian Conference on Electrical and Computer Engineering,” pages 367–372, 1999.

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 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 Technological University, Cookeville, TN USA 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. In 2008, Dr. Elkeelany was recognized as a lifetime member of The Strathmore's Who's Who.