VLSI ARCHITECTURE AND FPGA IMPLEMENTATION OF ICE ENCRYPTION ALGORITHM

Mrs.Laxmi Tiwari laxmi.nanhoriya@gmail.com
Mrs.Babita Verma babitaself@rediffmail.com
Mr.Saurabh Singh saurabh_singh1983@rediffmail.com
Lecturer IT/ CSE
BIT,Durg India

ABSTRACT

In modern security, the need for safe cryptographic algorithms that are hardware implemental is great. A hardware architecture is proposed in this paper, for the implementation of the ICE encryption algorithm. Since this cipher is optimized for use on software, a hardware implementation of that algorithm that achieves good performance results has much interest. The proposed implementation can be used for both encryption and decryption process. It is a folded architecture using feedback logic, designed for small chip covered area and high speed performance. The proposed architecture was implemented by using an FPGA device. The achieved throughput is equal to

116 Mbit/sec, using a system clock with frequency up to 29.1 MHz.

1. INTRODUCTION

Security is a primary requirement of any wireless cryptographic protocol. In order to find a solution to this always up to date problem, cryptographic algorithms are constructed to provide secure communication applications. However, the clever design of an algorithm is essential if the security of an application is to be maintained. Although there are many good algorithms with different usages and characteristics, not all of them can be characterized fully secure [1]. Many works from different research groups have been published, analyzing cryptographic methods for finding holes in the security strength of today's encryption algorithm. Thus new encryption algorithms are needed that do not have such security holes. That however might have the side effect of high complexity, which can make the implementation of

an algorithm very difficult if not impossible.

One of the major problems of modern computer security is the design of cryptographic algorithms that have as little vulnerabilities as possible while maintaining their low implementation complexity. Many algorithms that are cryptographically secure are not easily implemented in computer applications especially in hardware. Thus the need for hardware implementations of secure algorithms becomes even greater.

Matthew Kwan [2] in order to solve the need for secure encryption algorithms proposed a new algorithm similar to DES. This algorithm is called ICE, which stands for *Information Concealment Engine,* and it has the interface of DES thus maintaining full compatibility with that algorithm. It can act as a substitute in existing applications. It is based on the idea of Data Dependence Rotation (DDR) since it uses Controlled permutation (CP) in order to maintain its cryptographic security, like CIKS-1 and SPECTR-H64 algorithms [3, 4]. The ICE algorithm was designed for use in software applications. Those applications however are slow due to the use of modular arithmetic [2]. So the need for faster implementations is great. That can be achieved through hardware implementations.

The ICE algorithm has not been implemented in hardware so there is a certain interest as to if that is possible and if the results are good enough for the hardware to be usable. Considering the fact that hardware implementations are generally faster and more reliable than software implementations the outcome of a hardware design is even more interesting.

In this paper, an architecture and the VLSI implementation of the ICE encryption algorithm are proposed. The system operates for the both encryption and decryption processes and has been optimized for low hardware resources and for high–speed performance. The proposed architecture has very encouraging performance result in terms of speed and throughput. This makes the design very useful in current applications that use DES as the base of a cryptographic protocol.

With the proposed architecture we focus on proving that the ICE algorithm can easily be implemented in hardware.

The paper is organized as follows: In Section 2 the ICE algorithm is described. In Section 3, a thorough analysis of the architecture and the VLSI implementation
is made. Performance analysis of the architectures is made in Section 4 and finally some conclusions are presented in Section 5.

2. THE ICE ENCRYPTION ALGORITHM

ICE is a standard Feistel block cipher [2], with a structure similar to DES. It takes a 64-bit plaintext, splits it in two 32-bit halves and mixes them with the key in a
fairly simple process. The right half and a 60-bit subkey are fed into the function F. Then the output is XORed with the left part of the key and the halves are swapped. This is the Transformation Round of the ICE algorithm.

This process is repeated for 16 rounds [2]. However the final round, before the ciphertext production, is different.

The final swap is omitted. The decryption process is the same, except that the subkeys are used in reverse order.

From the above description of ICE algorithm it is clear that its strength is centred in the F function. In ICE the 32-bit plaintext, using a function E, is expanded in
four 10-bit values according to the manner:

$E1 = P_1\ P_0\ P_{31}\ P_{30}\ P_{29}\ P_{28}\ P_{27}\ P_{26}\ P_{25}\ P_{24}$

$E2 = P_{25}\ P_{24}\ P_{23}\ P_{22}\ P_{21}\ P_{20}\ P_{19}\ P_{18}\ P_{17}\ P_{16}$

$E3 = P_{17}\ P_{16}\ P_{15}\ P_{14}\ P_{13}\ P_{12}\ P_{11}\ P_{10}\ P_9\ P_8$

$E4 = P_9\ P_8\ P_7\ P_6\ P_5\ P_4\ P_3\ P_2\ P_1\ P_0$

One of the differences from DES is that after the
expansion function E, key permutation is used [2]. A 20- bit subkey, called permutation key is used to swap E1 with E3 and E2 with E4. When the odd bits of the
permutation key are set they swap E1 relative bits with E3 bits else they swap E2 relative bits with E4 bits. The outcome is XORed with a 40-bit subkey and the fed in the S-boxes.

The S-boxes of ICE use Galois Field exponentiation. Each S-box takes a 10-bit input X. Bits X_9 and X_0 are concatenated and form the row selector R while bits X_8 to X_1 concatenated form the 8-bit column selector C. For each row, there is a XOR offset value O_R and a Galois Field prime P_R. The output of the S-box is an 8bit value which is given by $(C \text{ xor } O_R)^7 mod\ P_R$. In Table 1 the values of the XOR offset and the Galois Field primes can be seen for all four S-boxes.

Table 1. The S-box XOR offset values and the S-box Galois Field prime values

S-box	O0	O1	O2	O3	P0
	P1	P2	P3		
S1	131	133	155	205	333
	313	505	369		
S2	204	167	173	65	379
	375	319	391		
S3	75	46	212	51	361
	445	451	397		
S4	234	205	46	4	397
	425	395	505		

The four 8-bit outputs of the S-boxes are combined using a permutation function P in a 32-bit value which is the result of the F function [2].

3. PROPOSED ARCHITECTURE

The proposed Feedback Architecture is shown in Fig. 1. The proposed Feedback Architecture performs both encryption and decryption with input plaintext block and key vector equal to 64 bits. It uses an input and an output register. Each of them stores the values of the left and right part of every round and swaps the two parts if needed (according to the algorithm in the final round there is no swap). Also a 16x60-bit RAM is needed to load and store the round keys.

The Key Expansion Unit creates the round keys, using the 64-bit Input Key following the specifications of ICE. The Keys are stored inside the RAM for every round.

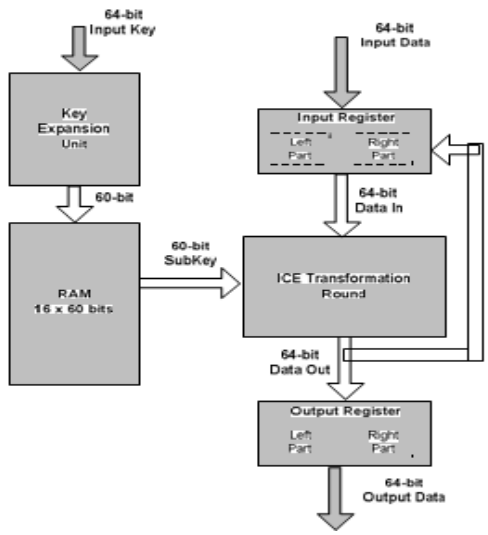


Fig.1 The Proposed feedback architecture

The encryption process is fairly simple. In each clock cycle, the data stored in the input register are inserted in the ICE Transformation Round along with the subkeys stored in the RAM. This process is repeated for 16 rounds. However at the final round, the Output Register is used in order to swap the left and right part of the ICE Transformation Round Output. That value of the Output Register is the cipher text. The decryption process follows the same process. However the subkeys are used in reverse order.

From the analysis of ICE, it is clearly seen that the main design interest lies in the ICE Transformation round of the algorithm, shown in Figure 2. Especially, in the implementation of the F function. The F function has four parts. The Expansion function E, the key permutation, the S-boxes and the Permutation function P.

Key permutation can easily be implemented using two multiplexers 2-1, while the Expansion and Permutation functions are just a rearrangement of wires.

So the highest implementation cost of the F function lies in the design of the S-boxes.

Considering that each S–box uses modular exponentiation in order to calculate its output, a VLSI architecture based on Montgomery Multiplication algorithm is proposed. This component is specially designed to do the mathematical function $A^7 \bmod P$. The architecture of this component is pipelined, with 6 stages, and it is based on the following algorithm:

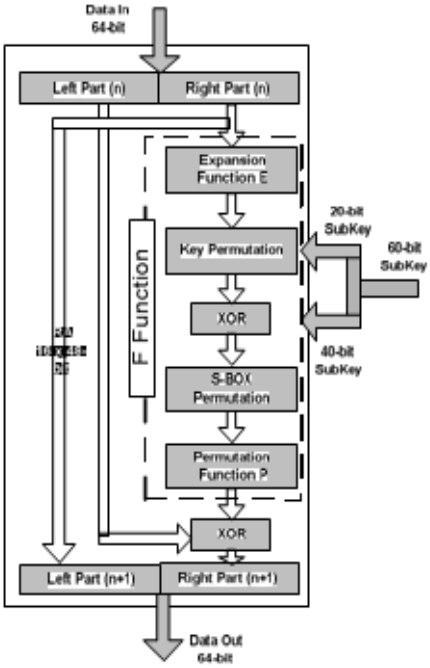
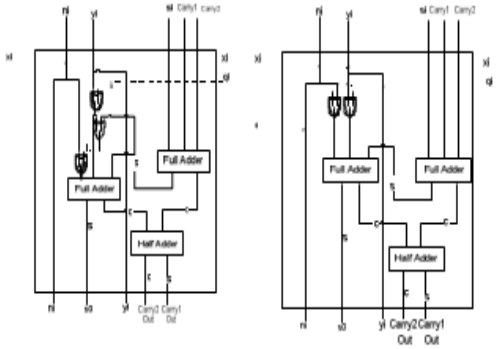


Fig.2 The ICE Transformation Round

Function A^7 mod P (X, P)

1. A=MM(X, R', P)
2. B=MM(A, A, P)
3. C=MM(B, A, P)
4. D=MM(C, C, P)
5. E=MM(D, A, P)
6. Out=MM(E, 1, P)

MM is the Montgomery Multiplication function and $R'=R^2$ modP is a pre calculated, fixed number. Step 1 is needed to transform the input value X into Montgomery format and step 6 to change the Montgomery formatted result E into a normal number value. The Montgomery Multiplication algorithm was implemented using a systolic architecture, shown in Figure 3, based on the following algorithm [5-8]:



(b) Q-calc Processing Element (c) The Basic Processing Element

Function MM (X, Y, N)

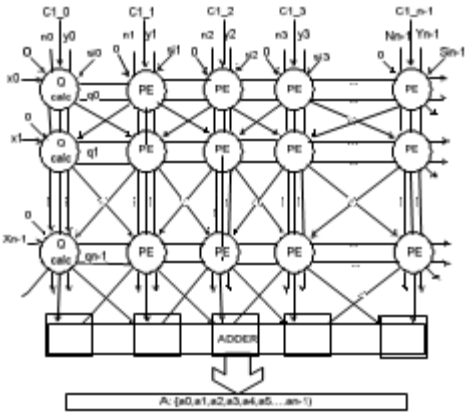
1. A=0
2. **For** k=0 to n-1 **do begin**
3. $q=(a_0 + x_k y_0) \bmod b$
4. $A=A+x_k Y+qN$
5. A=A/b

   End
6. **Return** A

This algorithm is a modified version of the original Montgomery multiplication algorithm [6]. The base b is considered Radix 2 (b=2) and $R=2^n$ where n is the bit length of the value N.

The architecture of the Montgomery multiplication, as seen in Figure 3, is an array of Processing Elements (Figure 3(c)). The elements on the first row, however, have a XOR gate more than the basic Elements PE. This gate is used for the calculation of the q value. Those elements are called Q-calc Processing Elements (Figure 3(b)). The output of the array is produced in a Carry

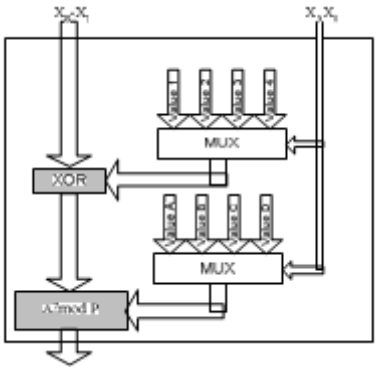
Save format so an adder is needed in order to get the final result. For that function an adder was implemented using Carry look ahead logic.



(a) The Systolic Architecture of Montgomery Multiplication

(b) The Hardware Architecture of the MM Function

The input X of the S-box is XORed with the appropriate value from Table 1. The output is fed to the A7mod P function where P is the value taken from Table



1. The appropriate values for the S-box are chosen from X_9X_0 bits of the input using multiplexers 4-1. The result of the A^7mod P function is the output of the S-box. The structure of an S-box is shown in Figure 4.

Fig.4. The Hardware architecture of the MM Function

4 PERFORMANCE

The proposed architecture has been captured by using VHDL. All the internal components of the design were synthesized placed and routed using XILINX FPGAdevices. The VLSI synthesis results are shown in Table2. The throughput reaches the value of 116 Mbit/sec for the encryption and decryption process.

Table 2. ICE Feedback Architecture Synthesis Results

FPGA DEVICE	XILINX v600efg900	
AREA ALLOCATION	*Used / Available*	*Utilization*
IOs	126 / 512	86 %
Fun. Generators	10661 / 13824	39 %
CLB Slices	5331 / 6912	42 %
Dffs or Latches	288 / 15360	12 %
f (MHz)	29.1	

Table 3. Covered Area and Frequency Comparisons.

Ciphers	Platform	Freq. (MHz)	AREA (CLBs)
AES [9]	Xilinx Virtex	25.9	2902
IDEA [10]	ASIC	25	251 k transistors
CIKS-1 [11]	Xilinx Virtex II	81	907
RSA[12]	0.5 μm SOG	50	156 k gates
Spectr-H64 [11]	Xilinx Virtex II	83	713
ICE (proposed)	Virtex-E v600efg900	29.1	5331

According to our knowledge until now, no other hardware implementation of the ICE cipher has been well known in the technical literature. For this reason the proposed architecture was compared with some good implementations of other widely used encryption algorithms that are based on the DDR logic [9-12]. In Table 3, the implementations are compared in both covered area and operating frequency. ICE performance in operating frequency is better compared to AES [9] and IDEA [10] block ciphers. Furthermore, in order to provide a detailed view of ICE performance, we compared ICE implementation with an RSA implementation [12].

5.CONCLUSIONS

ICE is a symmetric key block cipher specially designed for software applications. An efficient architecture for the VLSI implementation is proposed in this paper. It is designed for high clock speed – performance and minimized area resources. It is proven that the ICE algorithm, used for this architecture, is able to be implemented on hardware applications.

The implementation on FPGA is a system that has an external clock of 29.1 MHz and a throughput of 116Mbits/sec. Compared to other popular encryption algorithms it is concluded that the implementation's performance is better than most block encryption algorithms implementations.

6. REFERENCES

[1] Bruce Schneier, *Applied Cryptography – Protocols, Algorithms and Source Code in C,* John Wiley & Sons, second ed. New York, 1996.

[2] M. Kwan, "The Design of the ICE Encryption Algorithm," in *Proc. of Fast Software Encryption Workshop,*1997.

[3]A. A. Moldovyan and N. A. Moldovyan, "A cipher Based on Data - Dependent Permutations", *Journal of*

Cryptology,
no.15, pp 61 – 72, 2002.

[4] Nick D. Goots, Alexander A. Moldovyan, Nick A. Moldovyan, "Fast Encryption Algorithm SPECTR – H64", *in Proc. Of International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security, (MMM-ACNS'01*), Russia, May 2001, Springer – Verlag, pp. 275-286

[5] D.J. Guan, *Montgomery Algorithm for Modular*
Multiplication, Lecture notes, National Sun Yat Sen University, 2001.

[6] Peter L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation,* vol. 44, no. 170,
pp. 519-521, 1985.

[7] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, The MIT Press, Cambridge, 1990.

[8] Shay Gueron, "Enhanced Montgomery Multiplication," in *Proc. of Workshop on Cryptographic Hardware and Embedded*
Systems, San Francisco, August 13-15 2002.

[9] K. Gaj and P. Chodowiec, "Comparison of the Hardware Performance of the AES Candidates Using Reconfigurable Hardware," *in proc. Of Third Advanced Encryption Standard (AES) Candidate Conf,* Apr. 2000.

10] R. Zimmermann, A. Curiger, H. Bonnenberg, H. Kaeslin, N. Felber, and W.Fichtner, "A 177Mb/s VLSI Implementation of the International Data Encryption Algorithm", *IEEE Journal of Solid-State Circuits*, vol. 29, no. 3, pp 303 – 307, 1994.

[11]N. Sklavos, A. A. Moldovyan, and O. Koufopavlou, "Encryption and Data Dependent Permutations: Implementation Cost and Performance Evaluation",*proceedings of International Workshop on MathematicalMethods, Models and Architectures for Computer Networks Security, (MMM-ACNS'03),* Springer-Verlag, Berlin 2003.

[12]Taek-Won Kwan, Chang-Seok You, Won-Seok Heo, Yong-Kyu Kang, and Jun-Rim Choi, "Two implementation methods of a 1024-bit RSA cryptoprocessor Based on Modified Montgomery Algorithm",*proceedings of 2001 IEEE ISCAS*
'01, May 6 - 9, Sydney, Australia.