ISSN:0975-9646

Study of Hummingbird Cryptographic Algorithms based on FPGA Implementation.

Reena Bhatia
M.Tech-CSE (Shri Ram College of Engineering & Mgt.)
Palwal,Haryana,India.

***Abstract*— Cryptographic algorithms are ubiquitous in modern communication systems where they have a central role in ensuring information security. This thesis studies efficient implementation of certain widely-used cryptographic algorithms. Cryptographic algorithms are computationally demanding and software-based implementations are often too slow or power consuming which yields a need for hardware implementation. Field Programmable Gate Arrays (FPGAs) are programmable logic devices which have proven to be highly feasible implementation platforms for cryptographic algorithms because they provide both speed and programmability.The most Important and the advanced one out of FPGA implementations is the Hummingbird Algorithm.Hummingbird is a novel ultra-lightweight cryptographic algorithm aiming at resource-constrained devices. In this work, an enhanced hardware implementation of the Hummingbird cryptographic algorithm for low-cost Spartan-3 FPGA family is described.**

***Keywords*— Hummingbird Algo, PGA,Encryption,Decryption.**

I. INTRODUCTION

Cryptography, the art and science of keeping messages ancient Egypt some 4000 year ago.Undoubtedly, cryptanalysis, the art and science of revealing messages secure, has a long history which can be traced back to hidden by means of cryptography, has an equally long history. Together cryptography and cryptanalysis are called cryptology. When history of cryptology is discussed, it must be emphasized that historical cryptology has little common with modern cryptology which started in the 20th century, and which is the topic of this thesis.First, some fundamental terminology is introduced. The message, which is to be kept in secret, is referred to as plain text. The process of hiding its content is called encryption and the encrypted message is referred to as cipher text. The process of receiving the content of plain text back from cipher text is decryption. A cryptographic algorithm is the mathematical function used for encrypting and decrypting messages. A modern cryptographic algorithm always includes a key.

A cryptographic algorithm, plain texts, cipher texts, and keys are referred to as cryptosystem.

The techniques used were Secret-Key Cryptography,Public-Key Cryptography and Elliptic Curve cryptography which were based on ASIC and used General purpose processors for the task.Then came FPGA which worked by combining them.

II. ROLE OF FPGA

Reconfigurable logic includes a wide scope of programmable devices including PLA (Programmable Logic Array), PAL (Programmable Array Logic),CPLD (Complex Programmable Logic Device), and FPGA. However, FPGAs are the only ones which have any major interest from the cryptography point-of-view and, henceforth, the terms reconfigurable logic and device always refer to FPGAs. FPGAs consists of reconfigurable functional units, reconfigurable interconnections,
and flexible interface. Reconfigurable functional units are used for implementing the logic needed in a design and they are connected with the reconfigurable interconnections. Interfacing is used for communication with the rest of a system. As a rule of thumb, the more flexible a reconfigurable architecture is the slower it is and the more power it consumes. Modern FPGAs typically contain hardwired logic units specialized for certaincommonly used tasks, such as DSP. These units provide improvements in performance and power consumption in those specialized tasks but they are useless in many applications where these operations are not directly needed . Virtex-II has hardwired 18_18-bit multipliers and Stratix II has more flexible DSP blocks which are useful in various DSP-related tasks.

III. WHAT IS HUMMINGBIRD TECHNOLOGY?

A novel ultra-lightweight cryptographic algorithm, referred to as Hummingbird, is proposed for resource-constrained devices. The design of the Hummingbird cryptographic algorithm is motivated by the well-known Enigma machine taking into account both security and efficiency.Block cipher and Stream cipher combines to make hybrid structure of Hummingbird and it has been shown to be resistant to the most common attacks to block ciphers and stream ciphers including birthday attack, differential and linear cryptanalysis etc. Cheap smart devices like RFID tags and smart cards are becoming important in our daily life. Well known applications include electronic passports, contactless payments, product tracking, access control and supply-chain management just to name a few.A considerable body of research has been focused on providing RFID tags with cryptographic functionality usually known as *lightweight cryptography* which has to deal with the trade-off among security, cost, and performance .

A. Hummingbird algorithm:

Hummingbird is neither a block cipher nor a stream cipher,but a *rotor machine* equipped with novel rotor-stepping rules. The design of Hummingbird is based on an combination of a block cipher and stream cipher with 16-bit block size, 256-bit key size, and 80-bit internal state.The steps performed are:

1) Initialization Process: Figure 1. Shows the overall structure of Hummingbird Initialisation Algorithm.When using Hummingbird in practice, four 16-bit random nonces NONCE*i* are first chosen to initialize the four internal state registers *RSi* (*i* = 1*;* 2*;* 3*;* 4), respectively, followed by four consecutive encryptions on the message *RS*1 _ *RS*3 by Hummingbird running in initialization mode . The final 16-bit ciphertext *TV* is used to initialize the LFSR. Moreover, the 13th bit of the LFSR is always set to prevent a zero register. The LFSR is also stepped once before it is used to update the internal state register *RS*3.

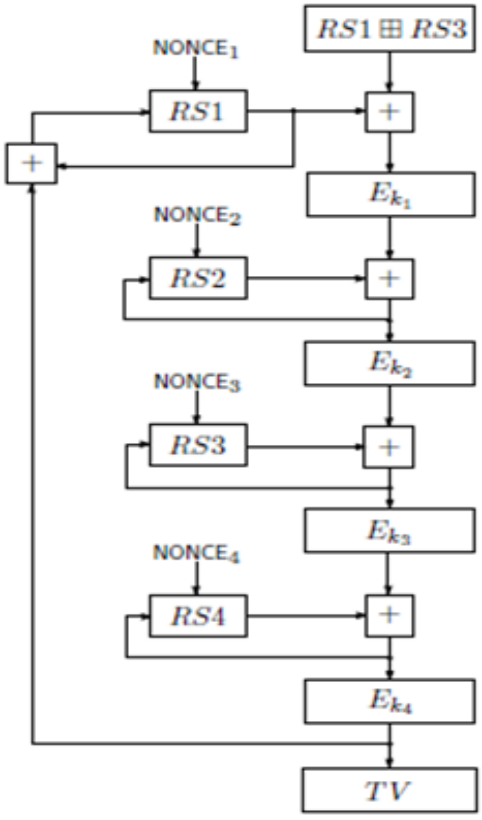

Fig. 1 Initialization Process

2) Encryption Process: The overall structure of the Hummingbird encryption algorithm is depicted in Figure 1(b).After a system initialization process, a 16-bit plaintext block *PTi* is encrypted by first executing a modulo 216 addition of *PTi* and the content of the first internal state register *RS*1.The result of the addition is then encrypted by the first block cipher *Ek*1 . This procedure is repeated in a similar manner for another three times and the output of *Ek*4 is the corresponding ciphertext *CTi*. Furthermore, the states of the four internal state registers will also be updated in an unpredictable way based on their current states, the outputs of the first three block ciphers, and the state of the LFSR.

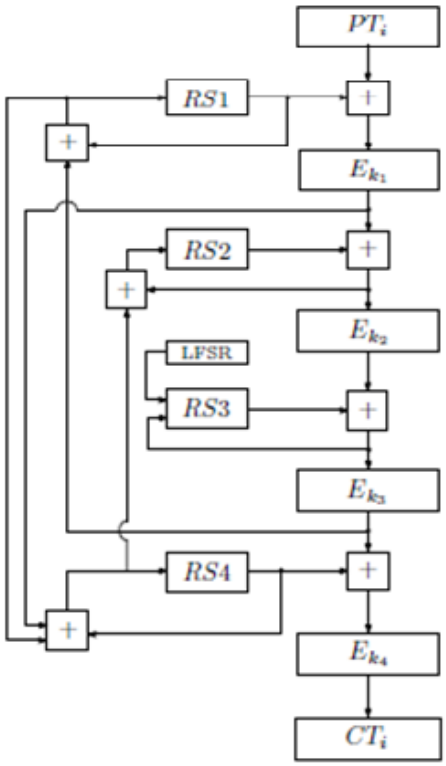

Fig. 2 Encryption Process

3) Decryption Process: The overall structure of the Hummingbird decryption algorithm is illustrated in Figure below.The decryption process follows the similar pattern as the encryption.

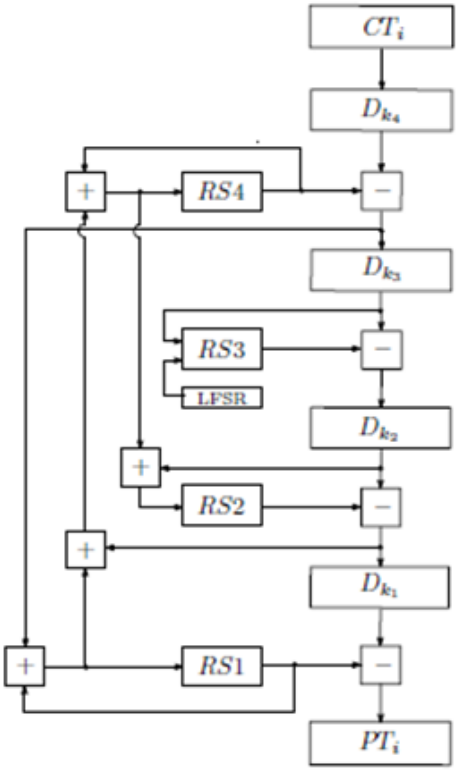

Fig. 3 Decryption Process

4)16-Bit Block Cipher : Hummingbird employs four identical block ciphers *Eki* ( )(*i* = 1*;* 2*;* 3*;* 4) in a consecutive manner, each of which isa typical substitution-permutation (SP) network with 16-bitblock size and 64-bit key as shown in the following.The block cipher consists of four regular rounds and a final round. The 64-bit subkey *ki* is split into four 16-bit round keys keys *K*(*i*)5 and *K*(*i*)6 directly derived from the four round keys. While each regular round comprises of a key mixing step,a substitution layer, and a permutation layer,the final round only includes the key mixing and the S-box substitution steps. The key mixing step is implemented using a simple exclusive-OR operation,whereas the substitution layer is composed of four S-boxes with 4-bit inputs and 4-bit outputs.The selected four S-boxes, denoted by *Si*(*x*) : F42→F42*;* *i* = 1*;* 2*;* 3*;* 4, are Serpent type S-boxes [1] with additional properties which can ensure that the 16-bit block cipher is resistant to linear and differential attacks as well as interpolation attack.

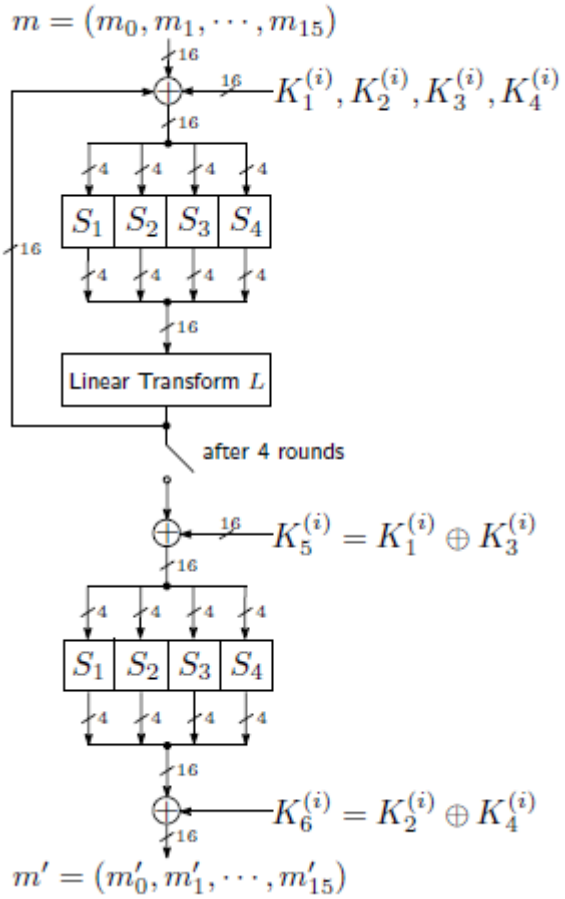

Fig. 4 16-bit Block Cipher

IV FPGA IMPLEMENTATIONS OF HUMMINGBIRD CIPHER

In this section efficient FPGA implementations of a standalone Hummingbird component are described. We implement an encryption only core and an encryption/decryption core on the low-cost Xilinx FPGA series Spartan-3 and compare our results with other reported (ultra-)lightweightblock cipher implementations on the same series.A speed-optimized and an area-optimized hardware architectures are also described in this section. Note that the choice of different kinds of I/O interfaces has a significant influence onthe performance of hardware implementation and is highly application specific. Therefore, we do not implement any specific I/O logic in order to obtain the accurate performance profile of a plain Hummingbird encryption/decryption core as well as provide enough flexibility for various applications.

A. Target Platform and Design Tools

FPGAs are composed of configurable logic blocks (CLB) and a programmable interconnection network. We implement both encryption and decryption modules in VHDL for the low-cost Spartan-3 XC3S200 (Package FT256 with speed grade -5) FPGA device from Xilinx. We use the integrated FPGA development environment Aldec Active-HDL 8.2sp1 for writing, debugging and simulating VHDL codes.Furthermore, Synopsys Synplify Pro C-2009.06-SP1 and Xilinx ISE Design Suite v11.1 are employed for the design synthesis and implementation, respectively.

B. Selection of a "Hardware-Friendly" S-Box

A "hardware-friendly" S-box is the S-box that can be efficiently implemented in the target hardware platform with a small area requirement. Four 4 × 4 S-boxes *Si*(*x*) : F42→ F42(*i* = 1*;* 2*;* 3*;* 4) have been carefully selected in Hummingbird according to certain security criteria (see Section II-D). To implement the compact version of Hummingbird, we need to choose a "hardware-friendly" S-box from four S-boxes l. Let *x* = (*x*3*//x*2*//x*1*//x*0) be the 4-bit input to the S-box and let *Si*(*x*) = (*S*(3)*i* (*x*)*//S*(2)*i* (*x*)*//S*(1)*i* (*x*)*//S*(0) *i* (*x*)) denote the 4-bit output of the *i*-th S-box (*i* = 1*;* 2*;* 3*;* 4). By using the Boolean minimization tool *Espresso* we can obtain the following minimal Boolean function representations (BFR) for the four S-boxes in Hummingbird where *xi* denotes the inversion of bit *xi*, denotes a logical AND and + denotes a logical OR. Note that each S-box can be implemented in hardware by using either a look-up table (LUT) or the Boolean function representations (i.e., combinatorial logic). The exact efficiency of the above two approaches significantly depends on specific hardware platforms and synthesis tools.

C. Speed Optimized Hardware Architecture

In this subsection we present a speed-optimized hardware architecture for Hummingbird encryption/decryption cores,where the encryption or decryption can be performed with four clock cycles. The main goal of the design is to achieve a high speed and throughput. To this end, we first propose a loop-unrolled architecture for the 16-bit block cipher, followedby a detailed description of the data path architectures of encryption/decryption cores.

1)Loop-Unrolled Architecture of 16-bit Block Cipher: The loop-unrolled architecture for the 16-bit block cipher is.In this architecture, only one 16-bit block of data is processed at a time. However, five rounds are cascaded and the whole encryption can be performed in a single clock cycle. The loop-unrolled architecture consists of 8 XORs, 20 S-boxes,

and 4 permutation layers for the datapath.After the given 16-bit block is XORed with the first round key,the obtained result is split into four 4-bit chunks and each of them is then processed by a 4-bit S-box in parallel. The linear transform *L* performs a permutation on the output of the Sbox layer for each of four regular rounds. The final round only includes the S-box layer and four XOR operations and the output ciphertext is stored into a 16-bit flip-flop (FF).

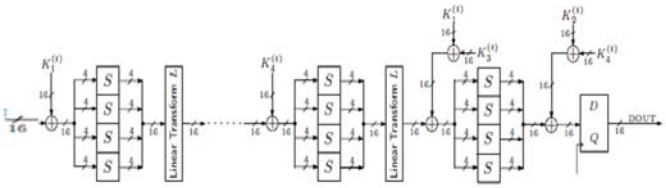

Fig. 5 Loop Uncontrolled Architecture of 16-bit Block Cipher

2)Speed Optimized Hummingbird Encryption Core: The top-level description of a speed optimized Hummingbird encryption core is illustrated as: After the chipenable signal changes from '0' to '1', the initialization process begins and four rotors *RSi* (*i* = 1*;* 2*;* 3*;*4)
are first initialized by four 16-bit random nonce through the interface *RSi* within four clock cycles. From the fifth clock cycle, the core starts encrypting *RS*1_*RS*3 for four times and each iteration requires four clock cycles to finish encryptionsby four 16-bit block ciphers as well as the internal state updating. During the above procedure, the 64-bit subkeys *ki* (*i* = 1*;* 2*;* 3*;* 4) are read from an external register under the control of a key selection signal. Moreover, depending on the value of a round counter, the multiplexer *M*5 chooses the correct computation results to update four rotors and other multiplexers select appropriate inputs to feed the 16-bit block cipher. Once the initialization process is done after 20 clock cycles, the first 16-bit plaintext block is read from an external register for encryption. With another four clock cycles, the corresponding ciphertext is output from the encryption core. Therefore, the proposed speed optimized Hummingbird encryption core can encrypt one 16-bit plaintext block per 4 clock cycles, after an initialization process of 20 clock cycles.

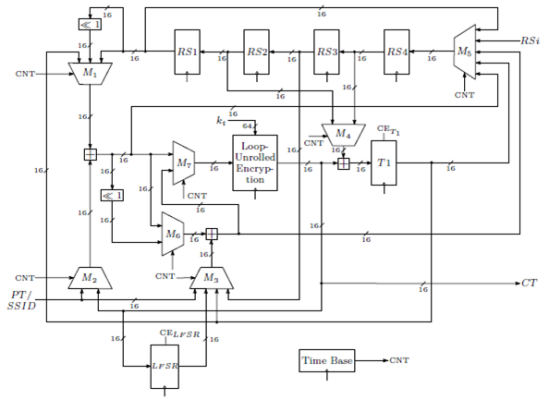

Fig. 6 Speed Optimized Hummingbird Encryption Core

3)Speed Optimized Hummingbird Encryption/Decryption Core: We depict the top-level architecture of a speed optimized Hummingbird encryption/decryption core in the following Figure. The Hummingbird encryption/decryption core supports the following four operation modes: i) encryption only; ii) decryption only; iii) encryption followed by decryption;and iv) decryption followed by encryption. Both encryption and decryption routines share the same initialization procedure that first takes 4 clock cycles to load four random nonce into rotors through multiplexers *M*5 and *M*11, followed by 16 clock cycles for four iterations. The architecture of the encryption/decryption core is quite similar to that of the encryption only core except the following several aspects. Firstly, the rotor *RS*2 completes the update when encrypting two successive plaintext blocks in the encryption-only core, whereas all rotors are fully updated each time a plaintext block is encrypted or decrypted in order to support the four operation modes in the encryption/decryption core. For this purpose, two multiplexers *M*10 and *M*11 are introduced to fully update the rotor *RS*2after each encryption/decryption. Secondly, an adder that can perform both modulo 216 addition and subtraction is included,which executes the corresponding arithmetic according to the operation modes of the core. Thirdly, two multiplexers *M*7 and *M*8 are used to feed correct values to the encryption and decryption routines of the 16-bit block cipher, respectively. Finally, all the other multiplexers select appropriate inputs based on the value of a round counter as well as the operation modes.

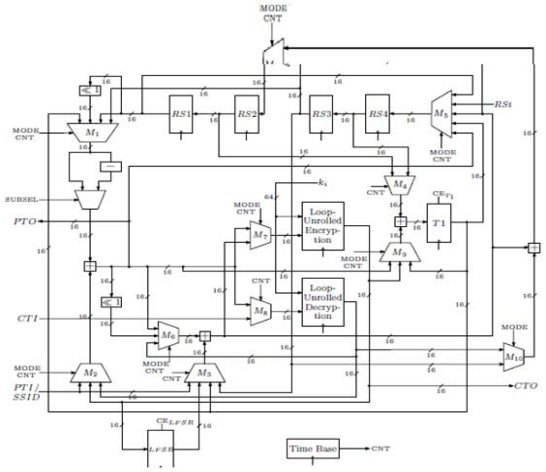

Fig. 7 Speed Optimized Hummingbird Encryption/Decryption Core

V. CONCLUSION AND FUTURE SCOPE.

This work presents the most efficient FPGA implementation of the ultralight weight cryptographic algorithm Hummingbird thanks to the coprocessor approach. The coprocessor approach is enabled due to the fact that FPGAs have dedicated memory blocks. The datapath of the Hummingbird coprocessor is implemented in four stages and the instruction count is reduced via pipelining technique. As the future research, we intend to conduct further cryptanalysis and security evaluations for Hummingbird cipher as well as propose low power ASIC implementations for low-cost RFID tags.

ACKNOWLEDGMENT

We are thankful to our Principal **Dr. S.K Gupta** for providing necessary facilities to wards carrying out this work. We acknowledge the diligent efforts of our head of department Mr**. Dinesh Sorout in** assisting us towards implementations of this idea.

REFERENCES.

[1] Agnew, G.B., Mullin, R.C., Onyszchuk, I.M., and Vanstone, S.A., "An implementation for a fast public-key cryptosystem," Journal of Cryptology, vol. 3, no. 2, Jan. 1991, pp. 63–79.

[2] P.Bulens, F.-X Standaert,J-J. Quisqarter, and P.Pellegrin, "Implementation of AES-128 on Virtex-5 FPGA",*Progress in Cryptography* AFRICRYPT 2008, LNCS 5023.

[3] G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C.Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher", *The 9th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2007*, LNCS 4727, P. Paillier and I. Verbauwhede (eds.), Berlin, Germany: Springer-Verlag, pp. 450-466, 2007.

[4] A.Poschmann, "Lightweight Cryptography-Cryptographic Engineering for Prvasive World". Ph.D. Thesis, Department of Electrical Engineering & Information Sciences, Ruhr Universitaet Bochum,Bochum,2009

[5] T.Eisebarth, S.Kumar, C.Paar,A. Poscmann, and L.Uhsadel. "A Survey of Lightweight Cryptography Implementations", *IEEE Design & Test of Computers*.vol.24.n0. 6.,pp 56.

[6] R.ANDERSON, E.BIHAM and L.KNUDSEN. "Serpent : A proposal for Advanced Encryption Standard." http://www.cl.cam.ac.uk/~rja14/Papers/Serpent.pdf(1999).

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.186.4213&rep=rep1&type=pdf

http://www.share-pdf.com/700915a60eb543b89a628e7f60fdb25f/FPGA%20IMPLEMENTATION%20OF%20HUMMINGBIRD%20CRYPTOGRAPHIC%20ALGORITHM.htm

http://www.computer.org/csdl/proceedings/fpl/2011/4529/00/4529a376-abs.html