FOX Algorithm Implementation: a hardware design approach

Colm O'Keeffe * Emanuel Popovici *

Abstract — **Encryption algorithms are becoming more necessary to ensure data is securely transmitted over insecure communication channels. FOX is a recently developed algorithm and its structure is based on the already proven IDEA (International Data Encryption Algorithm) cipher.**

FOX is a symmetric (private key) block cipher. Its top-level structure uses the Lai-Massey scheme and the round functions used in the scheme are substitution permutation networks (SPN). Its flexibility lies in the fact that it can be efficiently implemented in hardware and software. We report some of the first results of implementing the cipher on an FPGA.

1 INTRODUCTION

Over the last number of decades there has been rapid development in communication technology and in computer processing power, which in turn has lead to a huge increase in data passing between companies, individuals and organizations. Encryption algorithms are used to provide the secure transmission of this data across insecure channels, such as telephone lines and ISDN, while maintaining its integrity. Attacks on transmitted data by unwanted third parties exploit weakly designed algorithms and these attacks can be either passive (listening on transmissions) or active (modifying the transmitted data).

In communications the original message before encryption is referred to as the plaintext (P) and is usually of a fixed length. Encrypting a message requires the algorithm to operate upon the plaintext using an encryption key (Ke) to produce the ciphertext (C). After transmission the original plaintext (P) is reconstructed at the receiving side from the ciphertext using the decryption algorithm and a decryption key (Kd). Ke and Kd are pseudo-random data patterns that are generated by a separate key-scheduling algorithm.

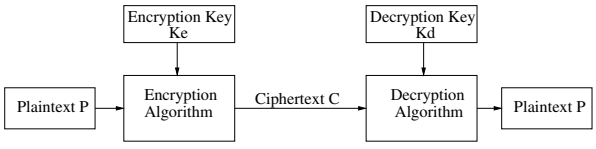

Figure 1: Cipher Process

*Department of Microelectronic Engineering, University College Cork, Ireland, email: colm.okeeffe@mars.ucc.ie, e.popovici@ucc.ie

Ciphers refer to the encryption/decryption process and there are two possible types. One cipher type is called a symmetric cipher (private-key), in which Ke and Kd are the same and are known only to the sender and receiver. The second type is an asymmetric cipher (public-key) in which different keys are used for the encryption and decryption process [2]. Private key algorithms can be either block ciphers, where the plaintext is a fixed length block, or stream ciphers, in which case the plaintext is broken down into individual bits. This paper will investigate the implementation on hardware of a new private-key block cipher called FOX.

Section 2 overviews FOX's construction. Section 3 goes into detail about the high-level structure of FOX and the primitive components used in its round functions. Section 4 explains the finite field arithmetic used in the round functions and finally section 5 discuss some of the architectures used and the synthesized results.

2 FOX Algorithm

FOX is a symmetric block cipher that was developed to have a high security level and large implementation possibilities [1,7]. A new key-scheduling algorithm was developed to increase this security level.

FOX's top level scheme is based on the Lai-Massey scheme which is the same used for the IDEA cipher [3,4]. Previous ciphers such as DES, Triple DES and Blowfish were based on the Feistel scheme. This scheme proved that if the round functions are random, then a 3-round Feistel cipher will look random to any chosen plaintext attack. For the Lai-Massey scheme it was proved that a similar result could be obtained if an orthomorphism function was added. The orthomorphism used is a Feistel scheme with an identity function as its round function [3].

Most modern ciphers are based on Claude Shannon's principles of diffusion and confusion. Confusion is the obscuring of the relationship between the plaintext and the ciphertext and can be achieved through substitution. FOX uses a non-linear 8-bit to 8-bit mapping using constructs called s-boxes. However confusion is not enough to provide security for a cipher and must be coupled with diffu-

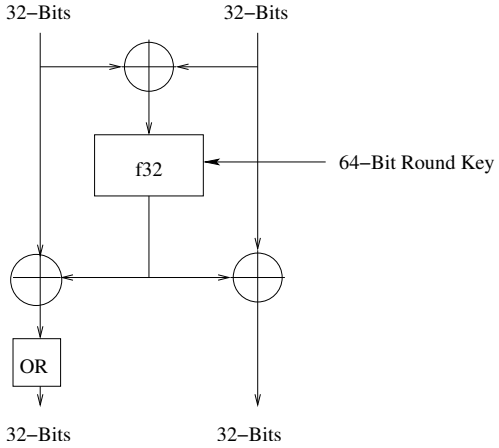

Figure 2: Imor64 Scheme

sion. To achieve diffusion, patterns that occur in the plaintext must be spread out so that they are undetectable in the ciphertext. For diffusion to occur a change in a single bit of the plaintext should result in changing the value of many ciphertext bits. Linear multipermutations are used in FOX to employ diffusion.

3 FOX Construction

There are four members of the FOX family as shown in Table 1. FOX64 and FOX128 are the generic members while FOX64/k/r and FOX128/k/r are variants of the cipher in which the key size k can be of lengtht 0 to 256 bits and r, the number of encryption/decryption rounds, can be any where from 12 to 255 iterations. The standard number of rounds used in the generic version is sixteen but twelve is the minimum for acceptable security levels.

Name	Block	Key	Round No
FOX64	64-bits	128-bits	16
FOX128	128-bits	256-bits	16
FOX64/k/r	64-bits	k	r
FOX128/k/r	128-bits	k	r

Table 1: FOX family members

3.1 Top Level Structure

FOX64 and its variant is based on the top-level Lai-Massey scheme shown in Figure 2. This scheme called Imor64 is used in the encryption process and contains the orthomorphism function called *OR* that is based on a Feistel structure. For decryption the Imio64 scheme is used, but the orthomorphism used is the inverse of the *OR* function called *IO* (Inverse *OR*).For encryption and decryption, Imor64 and Imio64 are used for (r-1) iterations, where r is the number of rounds used. On the final round the function Imid64 is used. The difference between this and the previous functions is that it contains no orthomorphism function. All three schemes use the same SPN called *f32*.

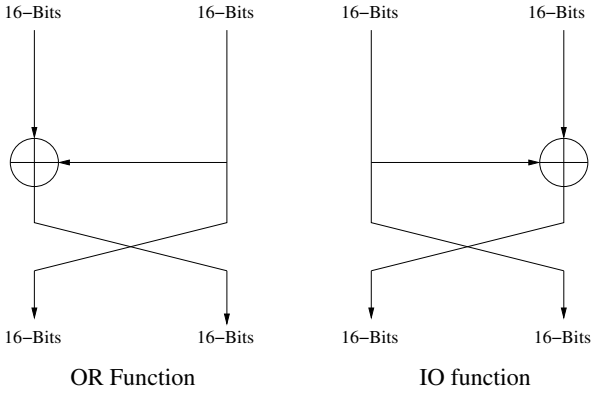

Figure 3: One-round Feistel Schemes

FOX128 uses an Extended Lai-Massey scheme with the SPN *f64*. Elmor128 and Elmio128 are the (r-1) iterated encryption and decryption schemes respectively. They use the same *OR* and *IO* orthomorphism as FOX64. Elmid128 is used for the final round and contains no orthomorphism.

3.2 Substitution Permutation Network

Two SPN's are used named *f32* and *f64*. The input to the SPN's are broken into equivalent byte values to be operated upon. The FOX128 SPN, *f64*, is structurally the same as FOX64's *f32* except an extended version. Eight byte values are operated upon in *f64*, while *f32* has four byte operands. For each round a different key called a round-key (RK) is used. These round-keys have been put in place by the key-scheduling algorithm.

An SPN in FOX consists of three stages, the round key addition part, a substitution part and a diffusion part. The round key addition is simply the bit-wise addition of the round key with the data at various stages through the network. This bit-wise addition is performed using an XOR operation. The substitution phase is a bijective 8-bit to 8-bit mapping with each different input mapping to a different output. FOX uses three small s-box's to achieve this. These s-boxes are arranged into a three round Lai-Massey scheme. The "small" s-boxes themselves are 4-bit to 4-bit mappings but when arranged in the scheme result in the 8-bit to 8-bit mapping. Diffusion is achieved through

a linear-multipermutation. This multipermutation multiplies the data by predefined matrices. For diffusion in FOX64, referred to as mu4, a 4x4 matrix is used due to the fact that there are four 8-bit operands. Each operand is multiplied by a matrix column. For FOX128 an 8x8 matrix is used for the eight byte operands used in *f64*.

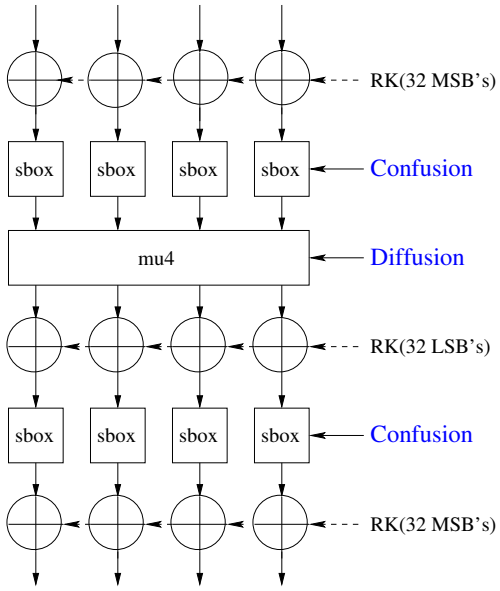

Figure 4: The FOX Substitution Permutation Network

4 Finite Field Arithmetic

Multiplication of the matrix used for diffusion is performed using finite field arithmetic [5]. The field used is GF(2^m). In this case m is 8 which gives a field containing 256 byte elements. Operations in a finite field are beneficial to a hardware design because the result of all calculations of the field will be represented using a constant number of bits (for our implementation it will be 8 bits). Elements can be represented as polynomials with the highest degree of (m-1) with coefficients in GF(2).

Multiplication in GF(2^8) corresponds to the multiplication of polynomials modulo an irreducible polynomial of degree 8. In FOX the irreducible polynomial is

$$P(x) = x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + 1 \qquad (1)$$

For the hardware design a Mastrovito multiplier is used, which is easily implemented using a network of AND gates and XOR gates [5].

5 Architectures

5.1 Design Implementation

Two architectural versions of both the generic FOX64 and FOX128 were implemented. The first is a pipeline architecture that uses registers between each round to pipeline the data through the design as efficiently as possible. A one-shot purely combinational design is the second version. The pipeline design is the most efficient for data throughput. Power consumption is minimized in the one-shot design as no clock is used which reduces switching.

One entity can be used for both the encryption and decryption process as both use the same structure except for the orthomorphism function. A encrypt/decrypt signal is used to select the required orthomorphism function depending on the process. The sixteen round-keys are held in on-chip registers and provided for each round.

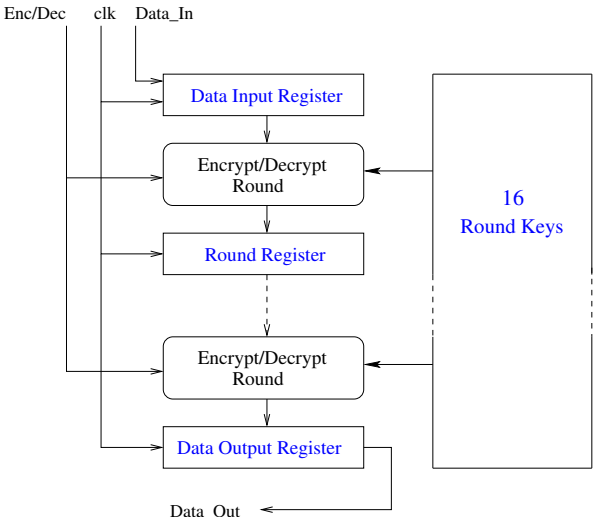

Figure 5: Pipeline Architecture

5.2 Design Results

These designs were synthesized for Xilinx VirtexII Pro technology and some results of implementation are given in Table 2.

Design	Clock	Speed	LUT	Slices
F64 Pipe	29.8ns	2.1Gbs	13929	7173
F128 Pipe	33.9ns	3.7Gbs	30393	15668
F64 1Shot	421.8ns	152Mbs	9562	5343
F128 1Shot	487ns	263Mbs	25087	14107

Table 2: Synthesis Results

From the results, the pipelined architectures are the most efficient, with FOX128 and FOX64 having

a throughput of 3.76 Gbits/s and 2.15 Gbits/s respectively. The clock for both one-shot implementations is quite large which is to be expected as it is a fully combinational design. This in turn has the effect of reducing throughput to 152 Mbits/s and 263 Mbits/s for FOX64 and FOX128 respectively. An increase in design area is a trade-off for the increased throughput in these designs. The pipelined design uses a greater amount of LUT's and slices in comparison to the one-shot designs. Also the amount of flip-flops used is increased for pipelining due to the use of registers. Pipelined FOX128 uses 4119 slice flip-flops while the pipelined FOX64 utilizes 2067 flip-flops. Registers in FOX128 are twice the width of FOX64 and is the reason behind the greater number of flip-flops in FOX128. The results compare well with an FPGA implementation involving AES [6].

5.3 Hardware versus Software

Encryption algorithms can be implemented in both hardware and software. FOX was developed with this flexibility in mind. It can be easily implemented on 32 or 64-bit architectures. Round functions can be implemented very efficiently in software using a combination of look-up tables and XOR operations [1,7]. On hardware round functions use memory blocks for s-box mappings and finite field arithmetic can be done using XOR and AND gates.

Although software encryption is becoming more prevalent today, hardware is still the embodiment of choice for military and computationally intensive commercial applications [2]. The main advantage of using hardware is speed as these algorithms use a lot of processing power for computation. Hardware is also quite secure in relation to software. An algorithm would need to be embedded deep in an operating system to be secure in software whereas hardware can be manufactured in tamper-proof devices. Software has the disadvantage of being slower but its benefits include portability and flexibility.

6 Conclusion

Aspects which are crucial to a good design are performance, size, cost and reusability. A fully parameterised hardware cryptographic design can save both time and cost when implemented correctly. In this paper different architectures are examined and implemented for the generic versions of FOX. The results showed that the FOX algorithm is suitable for FPGA implementation and some preliminary results are reported in this paper. Migration to other FPGA technologies is possible and the design is open to implementation of the key scheduling algorithm on-chip.

References

[1] P. Junod, S. Vaudenay, "FOX Specifications Version 1.1", EPFL Technical Report IC/2004/75, 24-11-2004.

[2] B. Schneier, "Applied Cryptography", Second Edition, John Wiley and Sons, Inc, 1996.

[3] S. Vaudenay, "On the Lai-Massey Scheme", In the Advances in Cryptology - ASIA - CRYPT'99, Springer-Verlag, volume 1716 of LNCS, pages 49-61, 2000.

[4] N. Sklavos, O. Koufopavlou, "Asynchronous Low Power VLSI Implementation of the IDEA Algorithm", Proceedings of IEEE International Conference on Electronics, Circuits and Systems, pages 1425-1428, 2001.

[5] E.D. Mastrovito, "VLSI Architectures for Computation in Galois Fields", PhD thesis, Linkping University, Dept. of Electrical Eng., Linkping, Sweden, 1991.

[6] M. McLoone, J.V. McCanny "Generic architectures and semiconductor intellectual property cores for advanced encryption standard cryptography", Computers and Digital Techniques, IEEE Proceedings, Volume: 150, pages 239-244, 18 July 2003.

[7] P. Junod, S. Vaudenay, "FOX: a New Family of Block Ciphers", Lecture Notes in Computer Science, Springer-Verlag, Volume 3357, pages 114-129, 2004.