International Journal of Computer Applications (0975 - 8887)
Volume 174 - No.5, September 2017

Low-footprint CLEFIA FPGA Implementations with Full-key Expansion

João Carlos Bittencourt
Centro de Ciências Exatas e Tecnológicas
Universidade Federal do Recôncavo da Bahia
Cruz das Almas, Brazil

Wagner Luiz de Oliveira
Departamento de Engenharia Elétrica
Universidade Federal da Bahia
Salvador, Brazil

Ricardo Chaves
INESC-ID, IST
Universidade de Lisboa
Lisbon, Portugal

ABSTRACT

In this paper two compact and high throughput hardware structures are proposed allowing for the computation of the 128-bit CLEFIA encryption algorithm and its associated key expansion processes. Given the needed modification to the CLEFIA Fiestel network, herein we show that with a small area and low performance impact, the CLEFIA key expansion for 128, 192 and 256-bit key can be deployed. This is achieved by using embedded components available in modern FPGAs and with an adaptable scheduling, allowing to compute the 4 and 8 branch CLEFIA Feistel network within the same structure. The obtained experimental results on a Xilinx Virtex 5 FPGA suggest that throughputs above 1Gbps can be achieved with a resource usage of 200 Slices and 3 BRAMs, achieving a throughput/Slice efficiency metric 50% higher when compared with limited state of the art.

General Terms

Computer systems organization FPGA

Keywords

CLEFIA, Encryption, Cipher, Key Expansion, FPGA

1. INTRODUCTION

The market of embedded systems has experienced substantial growth in the last decades. Currently, the use of mobile and embedded systems, also know as smart devices, already exceeds the use of personal computer systems. Many of such new systems already affect the daily lives of people all over world. As a consequence, the need for security and privacy services has also increased, and to provide them, cryptographic primitives are used. Towards this, efficient and compact implementations of such primitives are needed. One such primitive is the CLEFIA symmetrical 128-bit block cipher, proposed and developed by SONY Corporation [9]. CLEFIA algorithm supports 128, 192, and 256-bit keys and provides improved cryptographic security through the use of Diffusion Switch Mechanisms and Whitening Keys among others, in order to ensure immunity against differential and linear attacks [8].

In 2012, CLEFIA was declared an International Standard in ISO/IEC 29192-2 in lightweight cryptography, and also a Candidate Recommended Cipher by the Japanese Cryptographic Research and Evaluation Committee (CRYPTREC), in the 2013 revision. Recent works on CLEFIA have highlighted its performance, particularly in hardware implementations for both ASIC and FPGA technologies. Many of these approaches strive for compact structures while maintaining high performance, leading to the optimization of the computational resources and the exploitation of possible parallelism between operations.

Because of the need for a complex 8-branch Feistel network when computing the key expansion for 192 and 256-bit keys, most existing lightweight structures that provide key expansion only do so for 128-bit keys, which uses the same 4-branch structure used by the data processing [11], [6], [1]. Note, however, that even though not supporting the key expansion for all key sizes, or none at all, the main encryption computation can generally be performed, since the only change in the core computation is the number of computation rounds. Particularly in FPGA designs, where embedded memories can be used to permanently store the round keys, the major trade-off related to introduce a dedicated hardware for the key scheduling process states to the increase of overall logic elements occupation. Such assertion is potentially aggravated when considered 192 and 256-bit key expansion.

The main goal of the work herein presented is to show that a CLEFIA ciphering structure, capable of supporting the computation of both 4 and 8-branch of CLEFIA Feistel networks, can be designed within the same hardware architecture at very low added resource cost and without performance penalty. To validate this, two fully functional compact hardware structures are proposed, supporting both the encryption/decryption computation of the CLEFIA algorithm and the respective key expansion for all key sizes. Both architectures were designed concerning the FPGAs internal architecture, embedded elements and device primitives, towards a fully optimized solution in both area and speed. In order to provide a proof of concept, a prototype implementation is herein presented supported on an FPGA technology, given its increasing deployment in embedded systems, flexibility and easy prototyping.

The remainder of this paper is presented as follows. Section 2 introduces the CLEFIA algorithm. The most relevant related works are discussed in Section 3. The proposed hardware structures and particular implementation details are presented in Section 4. Section 5 depicts the experimental results and presents the analysis of these results in comparison with the limited state of the art. Concluding remarks and future works are presented in Section 6.

2. CLEFIA

The CLEFIA cipher is an 128-bit symmetrical block ciphering algorithm supporting cipher key sizes of 128, 192, and 256bits. This algorithm is based on the well known and commonly used Feistel Network structure [3]. As in most block ciphers, the input data is processed over several rounds, adding confusion and diffusion with the input key. In this particular algorithm the data and key are processed over 18, 22, or 26 rounds depending on the cipher key size. The round computation is exactly the same for each iteration, only varying the respective round key obtained from the key expansion itself.

The round computation includes state of the art design techniques, such as Whitening Key and Diffusion Switch Mechanism. The Whitening Key is a technique used to improve security of iterated block ciphers, consisting in steps that combine the plain text and the resulting cipher text with portions of the cipher key, before the first and after the last rounds. On the other hand, the Diffusion Switch Mechanism consists of multiple diffusion matrices organized in a predetermined order, to ensure immunity against differential and linear attacks [8].

2.1 Data Processing

The CLEFIA encryption process, depicted in Figure 1, takes a 128-bit input data block $P = P_0|P_1|P_2|P_3$, four 32-bit whitening keys $WK = WK_0|WK_1|WK_2|WK_3$, and a set of 32-bit round keys RK_i as data inputs. The resulting outputted cipher text, $C = C_0|C_1|C_2|C_3$, is a 128-bit cryptogram. The first step of the encryption process is to XOR the second and fourth words of the plain text (P_1 and P_3) with the first and second 32 bits of the original key (WK_0 and WK_1), performing the first key whitening procedure. After this operation the rounds are executed.

Each round computation is composed of a 4-branch, Type-2, generalized Feistel iterative structure, defined by $GFN_{4,n}$ where n is the number of rounds to be computed [9]. The round computation contains two parallel non-linear F Functions per round, in which a copy of the first and third words, and two round keys, are their respective inputs. The output result of each of these functions is XORed with the second and fourth words, respectively. The first and third words are not submitted to any process and are swapped with the other two processed words by left-round shifting all four of them, as illustrated in Figure 1. In the final round, instead of performing the Feistel swap, the final output values are directly obtained by XORing the second and fourth final words with the last two whitening keys.

Besides the round keys addition, the F_0 and F_1 functions employ two different types of 8-bit S-Boxes (S_0 and S_1) and two distinct diffusion matrices (M_0 and M_1) [9], as depicted at the rightmost of Figure 1. Each of the four 8-bit input lines is multiplied by the values in one column of the matrix, and additions are made lastly in order to perform a matrix multiplication. Note that these computations are performed over GF(2^8), meaning that they are composed of XOR operations.

Given the Feistel Network based structure of this algorithm, the decryption process is identical to the encryption one, using the same computational units, differing only in the order that the operations are performed, and by feeding the round and whitening keys in the inverse order [9].

2.2 Key Scheduling

As stated above, on each round two 32-bit round keys are used. Thus, a set of 36, 44, or 52 round keys (depending on the key

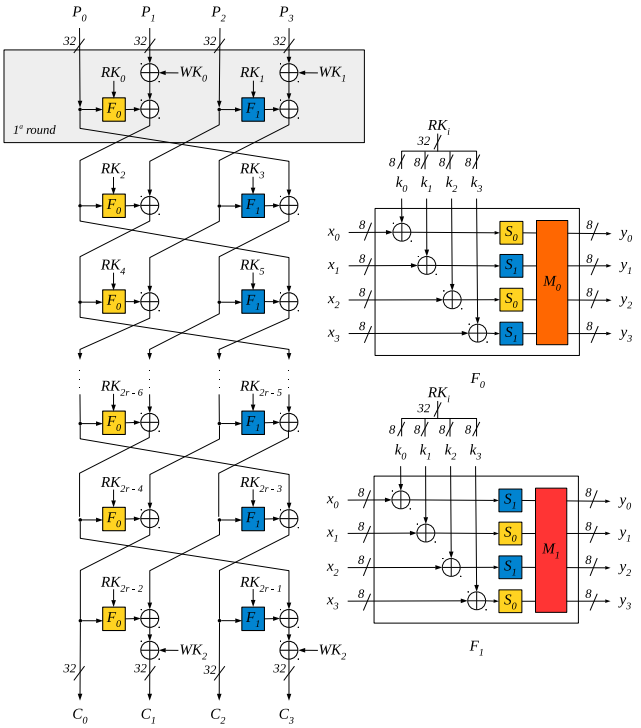

Fig. 1. CLEFIA encryption data path.

size) are used, plus 4 additional 32-bit whitening keys [9]. In order to obtain these multiple keys, the 128 to 256-bit input cipher key needs to be expanded. Such an expansion is performed using the key schedule algorithm specified in [9].

The whitening key (WK) generation is accomplished according to the key size. For a 128-bit input key, the four 32-bit whitening keys are obtained directly from the input key by:

$$WK_0|WK_1|WK_2|WK_3 \leftarrow K. \quad (1)$$

For the 192 or 256-bit input keys, the four 32-bit whitening keys are no longer obtained directly. The input key becomes two 128-bit blocks, KL and KR, and a bitwise XOR operation is applied between the two, resulting in the 128 bits of the whitening key. The KL and KR are composed as shown in the following, accordingly to the cipher key: if it is 192 bits long (2) or 256 bits long (3).

$$KL||KR \leftarrow K_0|K_1|K_2|K_3 \;||\; K_4|K_5|\overline{K_0}|\overline{K_1} : K^{192} \quad (2)$$

$$KL||KR \leftarrow K_0|K_1|K_2|K_3 \;||\; K_4|K_5|K_6|K_7 : K^{256} \quad (3)$$

For both 192 and 256-bit keys, the corresponding whitening key is computed by:

$$WK = KL \oplus KR. \quad (4)$$

In order to provide a preprocessing structure, the key expansion of a 128-bit key uses the same 4-branch GFN network used for the CLEFIA data processing to obtain an intermediate L key. The differences in the 128-bit key expansion remains that the input data of the GFN structure is now the input key itself. However, when considering the key scheduling process for the 192 or 256-bit keys, the GFN network becomes an 8-branch structure ($GFN_{8,n}$), as depicted in Figure 2. In this case, the input value is a combination

of $K = KL|KR$, resulting in a 256-bit input data block. The round keys used on the $GFNs$ are replaced by a different set of predefined constant values CON_i. Note that no whitening keys are used when performing the key expansion.

The 8-branch Feistel structure uses two non-linear F functions twice per round and processes eight input words on each round. In such an 8-branch network, just as copies of the first and third words are fed into F_0 and F_1 respectively, copies of the fifth and seventh words follow a similar pattern, as depicted in Figure 2. The result of each F function is XORed with the second, fourth, sixth, and eighth words. The resulting eight words are then swapped. In the final round, the output LL and LR values are directly obtained, as depicted in Figure 2. Instead of a ciphered text, the output of the CLEFIA GFN structures in the key expansion process is either a 128-bit block (L), for 128-bit input keys, or two 128-bit blocks (LL and LR), for the remaining key sizes.

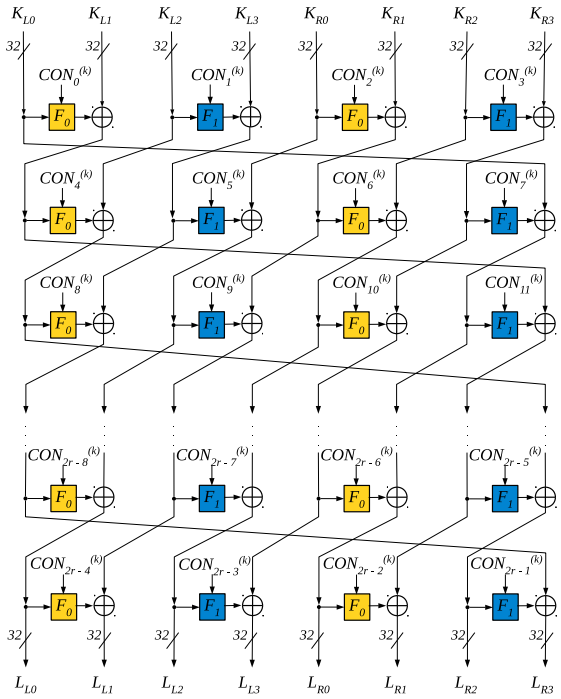

Fig. 2. CLEFIA $GFN_{8,n}$ structure.

After the GFN computation is completed, the result (L or LL and LR) is expanded in an iterative way using a Double Swap (Σ) function. This function simply swaps several bits of its 128-bit input and returns another equally sized, as specified by:

$$\Sigma(X) = X[7-63]|X[121-127]|X[0-6]|X[64-120] \quad (5)$$

After performing the Σ function, the 32-bit round keys are obtained by adding alternately the L, K, and $\Sigma(X)$ values with another predefined set of constants [9].

3. RELATED WORKS

When implementing dedicated structures for block ciphers on hardware, the efficiency improvements are mainly achieved by: round folding, with unrolled or rolled structures; the type of components used for the implementation of the round operations, in particular the substitution operation; and with operation rescheduling. Particularly in unrolled structures, multiple rounds of an algorithm can be processed independently and in parallel through a pipelined computation, imposing higher area demands. On the other hand, this approach allows for higher throughput and working frequencies.

Considering the existing state of the art, one of the first implementations of CLEFIA, targeting ASIC technologies, was proposed in [11], and included the CLEFIA encryption/decryption and a 128-bit key expansion, not supporting 192 and 256-bit keys. The authors proposes two hardware structures analyzed from the point of view of the non-linear F functions implementation. Three approaches which considered: (i) a hardwired implementation of both S-boxes and diffusion matrices; (ii) implementing two lookup tables for the S-Boxes; and (iii) merging both S-boxes and diffusion matrices operations into T-boxes. The authors then demonstrates the possibility of merging both F functions with no area increase an still achieving high throughput.

The first FPGA based CLEFIA implementation was presented in [5] consisting of a pipelined unrolled architecture with three different key expansion units for all possible key sizes. Although unrolled structures commonly result in a higher throughput, it comes with the cost of very high area requirements and the lack of flexibility. Furthermore, each unrolled structure can only target a particular key size, thus increasing the occupation overhead in systems which targets multiple key sizes.

In [6] it is considered a folded full-round compact structure and folded half-round counterpart. These architectures are based on a 128-bit and 32-bit data paths and explore the FPGA embedded memory, to store the T-Boxes. Rather than implementing a key expansion circuit, the authors load the already expanded keys into another dedicated embedded memory, achieving a more compact design with throughputs above 1 Gbps.

An extended version of the work of [6] is presented in [1], which proposes a lightweight key expansion structure for 128-bit keys. Such a key expansion structure also explores the embedded memories of FPGA devices to store the generated round keys as well as the CLEFIA constants used for both obtain the L keys and the ones used in the round key computation.

In [4] two CLEFIA encryption structures where evaluated on FPGA devices. The results show that ASIC targeted structures are also feasible in an FPGA design environment. The considered implementations used FPGA LUT primitives to store the S-boxes lookup tables, instead of a memory-based design, resulting in a high resource demanding structure. Besides that, throughput rates are bellow the ones achieved by so far state of the art CLEFIA FPGA implementations.

4. PROPOSED CLEFIA ARCHITECTURES

Recent contributions on FPGA based CLEFIA implementations provides a reasonable perspective of commonly approaches used to deploy symmetric cipher algorithms in such devices. Unrolled round designs provides a faster cipher implementation to non-feedback modes in which input block operates independently. On the other hand, rolled round structures results in compact hardware architectures in feedback mode, although it requires a more accurate design tuning in order to satisfy performance requirements.

Regarding block ciphers, two methods can be applied for describing the non-linear transformations: (i) hardwired logic; and (ii) addressable lookup tables, such as S-box or T-box. FPGA

devices provide a small amount of on-chip programmable memory blocks which can be used for that purpose.

When considering the key expansion on FPGA devices, one can say that it can be implemented in tree different ways: (i) precompute the round keys on the system software and store it into a internal storage structure (Distributed LUT or BRAM, for example); (ii) compute it at run time; or (iii) provide a precomputation hardware and store the resulting keys into the device storage elements. Software implementations commonly leads to a more compact and high throughput hardware structure, due to the absence of a key expansion unit. Although it requires a general purpose unit to perform such operation. In fact the coexistence of key expansion mechanisms in the hardware structure increases the area demands, particularly when considered 192 and 256-bit keys, posing a design challenge to provide a fully compatible solution performing at high throughput rates.

The main goal of the work herein proposed is to design two compact structures capable of both computing the CLEFIA encryption and the key scheduling for all possible key sizes without interfering the overall process efficiency. In the following sections we present two types of hardware structures for that purpose.

In Type-A structure, a rolled round structure based on the work presented in [6] is proposed. Such an implementation considers a 32-bit half-round register chain pipeline structure and two T-boxes for the F functions in order to provide a hybrid $GFN_{4/8,n}$ network. Regarding the key expansion structure of Type-A starts from the expansion structure proposed in [1] thus providing a fully featured CLEFIA key expansion unit.

By exploring the addressable shift logic operation mode present in Xilinx FPGA devices, Type-B architecture reduces the circuit area requirement by packing the register chain into LUTs primitives settled as SRL logic blocks. By doing this, it is also possible to reduce the number of multiplexers (MUXes) with no performance penalty.

Addressable shift logic operation is a commonly resource used in FPGA and thus in block cipher implementations, such as AES. It consists of a LUT commonly configured as 1 bit wide per 16 to 32 bits depth. Therefore a SRL can be grouped in order to provide arbitrary word width shift registers. The main advantage of the SRL is the capability of address a specific data to the output. Such a structure can be explored in several ways and typically leads to a more compact structure.

Additionally, in both architectures, we consider that the data is fed into the structures through 32-bit data buses, taking into particular account that external data buses are typically smaller than 128-bits. Although compatible with any FPGA internal configuration, the following implementations was specially designed for targeting devices with six or more input LUTs, thus each highlighted block in the following represents an FPGA combinational logic level.

4.1 Type-A Architecture

In Type-A architecture, the CLEFIA word swapping needed to implement the Feistel network is performed by a multiplexed chain of registers, which stores the intermediate values and schedules them when needed for the computation. Such a $GFN_{8/4,n}$ Feistel network has a larger data path, due to the need of storing and multiplex additional intermediate values. This storage and multiplexing can be performed by extra registers and wider multiplexers.

Figure 3 shows the CLEFIA data path of Type-A architecture, where the width of data lines is of 32-bit except for those properly indicated. The data processing input data is fed through the multiplexers depicted in the left most of Figure 3 by stages Ⓐ and Ⓑ, which selects 32-bit words at a time. For the $GFN_{8,n}$ computation, the input is divided into two 128-bit serial data streams represented by $P_i = P_1|P_3|P_5|P_7$ and $P_j = P_0|P_2|P_4|P_6$. Similarly, to compute the $GFN_{4,n}$ network the 128-bit input is represented by two 64-bit data blocks $P_i = P_1|P_3$ and $P_j = P_0|P_2$.

Since the original computation can still be accomplished, Type-A structure can perform both $GFN_{4,n}$ and $GFN_{8,n}$ depending on its control signals. When settled to cipher or 128-bit key expansion (4-branch mode), the structure computes a $GFN_{4,n}$, using 2 clock cycles per round. For the 192 or 256-bit key expansion (8-branch mode), the structure schedules two pairs of F functions, thus requiring 4 clock cycles to compute each round. In order to provide a proper scheduling, the two registers in the left most of the GFN data path are used only when performing the intermediate L key computation in 8-branch mode.

Considering FPGAs as the target technology, embedded RAM blocks are considered for the T-box implementation as depicted in [6], [1]. Since a 32-bit data path structure is being herein designed, two dual-port BRAMs are required. In each input port of each BRAM, the first 8 bits of address are used by the respective byte input. An additional address bit of the BRAM is used to switch between F_0/F_1. Given the 32-bit output word of each BRAM port, a total of 16 Kb per BRAM is required.

The whitening key is inputed in block Ⓓ in order to take advantage of the 6-input LUTs present in most modern FPGA devices, thus reducing the overall LUT occupation and critical path. As it will be presented in the following, the whitening key input can be settled to zero when needed by the key scheduling control mechanism. Block Ⓓ also implements the final addition over $GF(2^8)$ following the T-box.

The data processing is started once the first 64 bit block is introduced into the data path, thus requiring 1 or 2 additional clock cycles in 4 and 8-branch modes, respectively. Since that there is no word rotation in the last round, the 128-bit resulting output is obtained through the unswap stage Ⓔ which selects the 32-bit blocks in a proper order for both cipher or plain text to the C_i output. This process then requires 2 additional clock cycles before a new block may be processed. Therefore, Type-A architecture takes 40, 48 or 59 cycles for complete the CLEFIA data ciphering for 128, 192 or 256-bit cipher key, respectively.

The intermediate L key is obtained through two 128-bit independent data buses composed by the proper combination of outputs ① to ⑥. The proposed architecture takes advantage of the data processing scheduling and the inherent shifting operations to use only four to six output signals to provide the proper 128 or 256-bit output key in one or two clock cycles, respectively. The resulting two internal data buses are then connected to the key expansion unit.

Herein the multiplexers in stages Ⓑ and Ⓒ were replicated. This reduces the critical patch in the CLEFIA feedback circuit without increase the LUT usage. Thus, the critical path in data processing remains between the T-box output and the output, regarding stages Ⓓ–Ⓑ–Ⓔ, resulting in three combinational logic levels and a total of 16 inputs.

The CLEFIA key expansion if performed after the $GFN_{4/8,n}$ computation. The proposed structure for Type-A architecture is based on a mixed 128 and 32-bit data path depicted in the right most of Figure 3. The first step is presented in stage Ⓖ, where K or KL and KR are inputed and stored into 128-bit registers by right shifting the 32-bit input blocks. As described in Section 2.2,

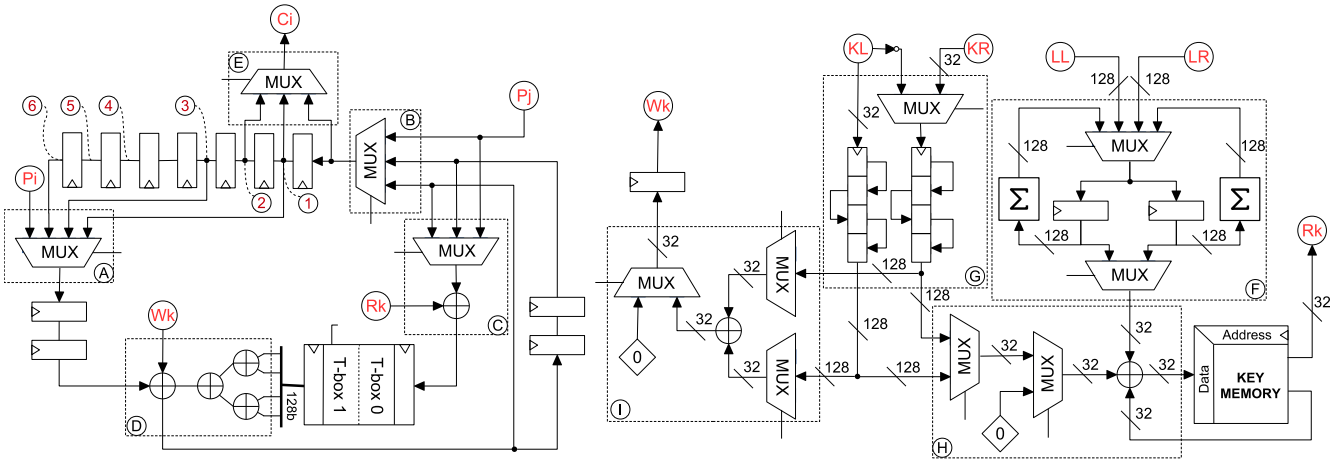

Fig. 3. Proposed data path for the Type-A architecture.

for a 128-bit cipher key, the whitening key is obtained directly from the input key. On the other hand, the whitening keys derived from 192 and 256-bit input keys are obtained by the XORing the lower and higher values of these larger keys. As depicted in stage ① of Figure 3, with a proper scheduling such an operation can be performed in a 32-bit data path, thus providing the whitening keys as they are required. This structure is also capable of send zeros to the whitening key output, thus reducing the data processing area overhead.

The key expansion itself is composed of two Σ function over the L keys (stage ⑧) and the round key computation (stage ⑨). Note that since Σ is a bit permutation function this overhead does not requires any additional logic. The Σ function can be performed in two ways: (i) for 128-bit keys, only the LL value is used; and (ii) for larger key values, both LL and LR values are computed in an iterative way. In order to load the initial LL and LR values, two 128-bit blocks feds the input multiplexer on the top of block Ⓕ in Figure 3, loaded by the combination of the several 32-bit blocks. At each four clock cycles the corresponding register is updated by the Σ. The writing control is managed by two independent clock enable signals. The output of stage Ⓕ is given by an 8-to-1 multiplexer. In the targeted device, such a structure can thus be implemented by a combination of two 6-input LUTs in the same Slice, thus reducing the routing overhead.

For each key size, a predefined set of constant is mapped into a 32-bit key memory block. These constant values (CON_{ij}) and the generated round keys are stored into the same embedded memory. Thus, this memory operates both as RAM, storing and reading the generated round keys, and ROM, where the constant values are stored. The memory address input control can target either the round keys, when data is being ciphered, the "keys" used for the L value computation or the remaining constant values. Given the 32-bit data words, 9 kbit is be necessary to store the keys, which fits into the embedded memory blocks of most common devices.

The last step of the round key expansion is the XORing between the corresponding constant CON_{ij}, the input key K_i and L_i value. This process is accomplished through a 32-bit data path and the resulting value is stored in the upper part of the BRAM. The corresponding key is also fed into block Ⓗ of Figure 3 through a 8-to-1 multiplexer. Since the values are stored into a embedded memory, the key expansion needs to be performed only once for each key change, without the need to recover the original L key.

Since an 8-to-1 multiplexer usually requires two 6-input LUTs to be implemented, a total of 416 LUTs are needed to implement the Type-A data path structure.

4.2 Type-B Architecture

In Type-B architecture the main objective is to reduce the overall area requirements due to the additional registers and larger multiplexers usage in both data processing and key expansion units. One of the main optimizations herein considered, in order to reduce the area overhead due to the $GFN_{4/8,n}$ supporting structure, is related to the CLEFIA word swap. This particular structure imposes a high cost due to the number of registers and multiplexers needed to provide the $GFN_{4/8,n}$ feature.

When considering FPGAs as the target technology, these individual registers can be replaced by shift registers. If these shift registers can be addressed to output particular internal values, then the swap operation itself can be performed by adequately controlling this same address. When considering Xilinx FPGAs, this addressable shift register can be mapped into Look Up Tables (LUTs) operating in either SRL16 or SRL32 LUT mode [12]. Currently Intel FPGA devices does not implement such a primitive. Instead, to obtain the same functionality of the SRL primitives, one can use the SHIFTREG function connected to a 16:1 multiplexer [2].

When configured to SRL mode, each LUT is able to implement a 1-bit wide addressable shift register, capable of storing up to 16 or 32 bits. Given the Type-B architecture with a 32-bit data path, the full storage and swapping operation of CLEFIA word swap can thus be implemented using only 32 LUTs, as depicted in block ② of Figure 4. An additional register is placed outside the shift register in order to optimize and reduce the critical path. Thus, the critical path of the ciphering structure was reduced to only two logic levels composed by nine 32-bit variables and remains between the T-box output and input.

The proposed Type-B architecture considers that the data is fed using a single 32-bit data bus Ⓟ at stage ① depicted in Figure 4. The input Ⓚ in stage ④ is used for both the cipher key and the whitening keys. In both cases we assume that it is possible to set zero on the input during the data processing flow.

After each iteration of a F function within a round (and Feistel word addition), the result is immediately fed back to the beginning of stage ①. This way an encrypting round is always being

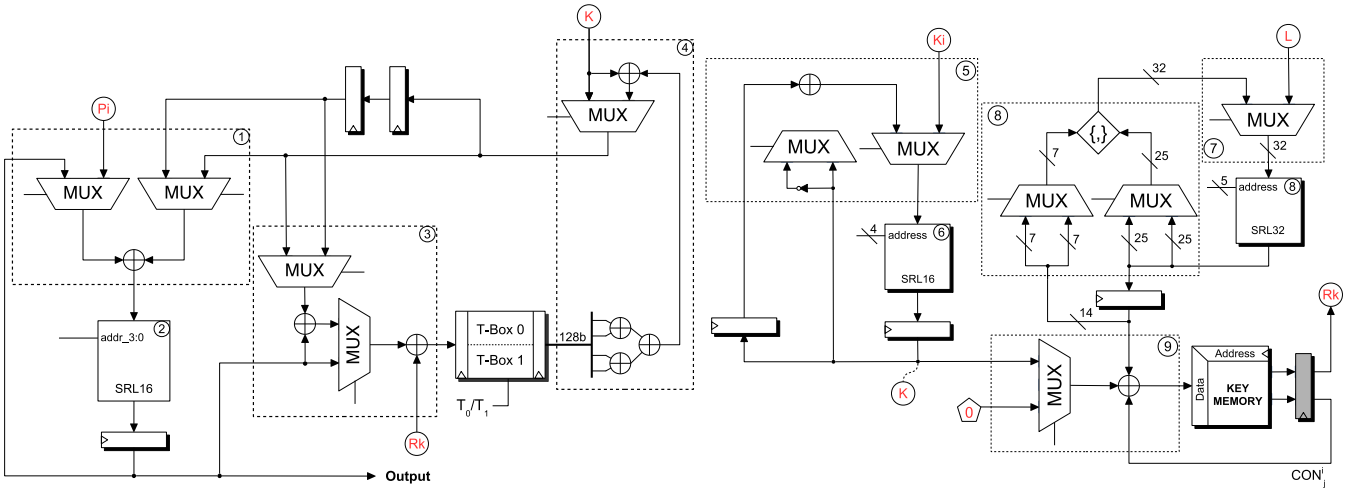

Fig. 4. Proposed data path for the Type-B architecture.

performed. The last operation performed by the CLEFIA algorithm is the XORing between the last two whitening keys in the final round, performed in stage ④. The resulting words are then stored into the SRL, so the output can thus be extracted in order directly to the SRL output bus. In addition, two additional clock cycles are needed to finish the data processing without data hazard.

Although the functional description stated above operates properly for the block cipher process, the key scheduling generation of 128 and 256-bit wide L keys needs to be treated carefully. During a key expansion process of a 128, 192 or 256-bit cipher key, the L value is collected directly from the shift register (after performing a GFN computation), and passed onto the key expansion structure, described below. The consequence of this approach is that 4 or 8 additional clock cycles are needed after the last round computation. This extra latency allows for the possibility to send the unswapped L values directly to the key expansion structure. Besides, since the encryption process can only start once the round keys have been computed, these additional clock cycles, required to feed the generated L key, do not impact the overall encryption performance.

Regarding the key expansion, the optimization goal is to reduce the area overhead imposed by the wider registers and MUXes needed to store and select the keys. In Type-B architecture the key expansion function is based on a 32-bit data path depicted in the right most of Figure 4. The first step, where the cipher key K and the whitening key WK are computed, is presented in stage ⑤. The 128, 192 or 256-bit cipher key is obtained by a 32-bit input bus and stored into the SRL16 LUT ⑥. Once the key is stored, it is sent to the $GFN_{4/8,n}$ structure, for the computation of the intermediate L key, through the Ⓚ connection, located at the SRL output. As stated before, this Ⓚ output is also used to provide the whitening keys to the ciphering structure.

The whitening keys derived from 192 and 256-bit keys are obtained by using the 32-bit XOR, depicted in stage ⑤, while the two left bottom registers are used to store the targeted KL_i and KR_i values. The resulting keys are then stored back into the SRL. Additionally, by disabling the SRL clock enable signal, both cipher key and whitening keys are kept into the same position, without affecting the LUT output mapping and also reducing the control complexity. Such an operation imposes eight clock cycles to perform the WK computation of 192 and 256-bit keys, performed while the GFN is being computed. A special care need to be taken regarding the control of the key input Ⓚ, since it is used both for the cipher keys, before the L values computation, and the whitening key feed when ciphering. Since only 12 positions are used, it is possible to send zeros to the output when needed.

The Σ function can be performed in two ways: (i) for 128-bit keys, where only the LL value is used; and (ii) for the larger key values, where both LL and LR values need to be computed. In the Type-B structure, we also aim to reduce the area overhead imposed by the key scheduling block. Thus, the set of 32-bit blocks of the L key are fed from the GFN main output through Ⓛ, and loaded by several 32-bit blocks into a SRL32 LUT. However, instead of performing the complete Σ operation in one cycle, the proposed structure takes four clock cycles to complete the permutation. The Double Swap function Σ is decomposed into an iterative computation targeting the proposed 32-bit data path as following.

$$\begin{aligned} \Sigma &: X \rightarrow Y_{0:3} \\ Y_0 &= X[92-120] \mid X[85-91] \\ Y_2 &= X[121-127] \mid X[39-63] \\ Y_3 &= X[32-38] \mid X[7-31] \\ Y_1 &= X[0-6] \mid X[64-84] \end{aligned} \tag{6}$$

Although four clock cycles are needed to perform the Σ function, only three stalls are imposed. Since the round key computation is performed once for each key, this latency overhead do not affects the overall circuit performance.

5. PERFORMANCE EVALUATION

In this section, experimental results for the proposed structures are presented and compared with the related state of the art. These results were obtained using the Xilinx ISE Design Suite (v14.7), while the design itself was described using VHDL language. The values presented for the proposed designs were obtained after Place & Route processing with software default parameters, namely Synthesis Normal Speed Optimization Effort, High Optimization Effort in Mapping and Place & Route, with no extra effort.

Table 1. Performance comparison of CLEFIA FPGA implementations with key expansion.

Design	Key Length (bits)	Latency (Cycle)	F-Function	Freq. (MHz)	Throughput (Mbps)	LUTs	FFs	Area (SLICE)	Efficiency (Mbps/S)
This Work Type-A	128,192, 256	40	LUT	216.1	691.59	950	1058	511	1.35
			Harwired	166.9	534.22	1033	1058	568	0.9
			T-box	**376.2**	**1,203.91**	**675**	**994**	**427**	**2.8**
This Work Type-B	128,192, 256	42	LUT	242.2	738.10	804	411	303	2.44
			Harwired	127.4	388.38	880	411	342	1.14
			T-box	**380.2**	**1,158.79**	**537**	**347**	**200**	**5.8**
[1] Type-B	128	38	LUT	235.3	792.57	808	449	297	2.7
			Harwired	166.7	561.40	895	449	325	1.7
			T-box	343.6	1,157.53	672	385	303	3.8
[11] Type-B	128	40	LUT	135.6	433.96	981	501	328	1.3
			Harwired	98.9	316.58	1049	502	364	0.9
			T-box	316.5	1,012.66	774	501	295	3.4

The obtained results for the presented structures, and those in the existing state of the art are depicted in Table 1. In order to achieve a more reliable comparison, the structures proposed in [11] and [1] the structures were implemented in VHDL as specified by the authors and mapped into the same technology using the exact same constrains for a "xc5vlx50" Virtex 5 device. In addition, the evaluation also considers the absence of a memory block by consider both LUT-based and hardwired S-boxes implementation. The hardwired structure of S-box S_1 is derived from a inversion over the composite field $GF((2^4)^2)$ such as described in [7]. On the other hand, the S-box S_0 was implemented as specified in [10]. For the proposed CLEFIA core with full key expansion, operating frequencies in the order of 380 MHz are achieved resulting in a throughput of 1.2 Gbps when considering the ciphering process. Type-B architecture achieve a more compact implementation at a cost of 200 Slices and 3 BRAMs. Considering a throughout per Slice efficiency metric, an efficiency of 5.8 can be achieved. Note that the structures herein presented was purposely design to be reused for the key expansion for all possible key sizes. If only the encryption process is required, several resources will remain unused and could be trimmed in order to reduce area.

In regard to the existing state of the art, two main approaches are considered, namely folded or unfolded structures. When considering a folded approach, and thus the most compact one of the two, the Type-A structure is not the most compact one available, but remains with competitive throughput. In [1] and [11], the authors only considered the data ciphering process, and 128-bit key scheduling. The lower complexity of the data path proposed in [1] requires 303 Slices and allows it to achieve a higher efficiency, of 5,8 Mbps per Slice. Although the wider data path, the implementation proposed by [11] requires 295 Slices and can achieve a throughput above 1 Gbps.

Given the achieved throughput per Slice efficiency of the structures herein presented, it can be concluded that the presented modifications to the CLEFIA core in FPGA do not represent a significant performance impact to the ciphering process itself. This means that, with proper care, it is possible to also perform the $GFN_{8,n}$ with minimum cost.

Regarding the key expansion itself, two main approaches are used: (i) compute them locally with a dedicated hardware; or (ii) compute the round keys off chip and store them into an internal memory during the initialization phase. Although an off chip computation of the key expansion may lead to more compact and efficient designs, several systems such as small embedded devices (that cannot afford software processing units) or systems that require rapid key change, need to have on chip key expansion. When this is required, extra resources are needed to be used in addition to the main CLEFIA encryption core.

The key expansion structures herein proposed has been design to support the key scheduling for all three key sizes. Type-A structure extends the work proposed in [1] by allowing the expansion of 192 and 256-bit keys. However, to support these key sizes, a significant amount of additional logic needs to be added. Since the 192 or 256-bit input keys are now supported, the input value needs to be merged by XORing them, conditionally, according to the key size. Additionally, two 128-bit values are now manipulated, namely the LL, LR, KL and KR values. Particularly, the doubling of the 128-bit registers, and their respective multiplexers, has increased the needed resources.

Regarding the data path resource utilization, both structures were designed to met the Virtex 5 internal structures constrains and functionalities. Each stage of the computation was designed to have no more then 6 inputs, thus being able to be implemented into a single device LUT. In order to reduce the delay impact of implementing a 128-bit data path for key expansion, Type-A architecture also explores the MUXF7 mode of operation. Such a implementation allows to implement an 8-input LUT by merging two 6-input LUTs into the same device Slice.

Obtained results demonstrated that a total of 675 LUTs are needed to implement the Type-A architecture. By exploring the SRL mode the number of LUTs in Type-B architecture was reduced to about 537 LUTs. Besides, the use of the SRL requires less storage elements when compared to the Type-A.

A special care must be taken to the scheduling process of both architectures, since it suports three modes of operation. The key scheduling control unit also imposes a higher complexity, resulting in almost 30% of the needed resources.

6. CONCLUSIONS

In this paper, two compact hardware structures are proposed, targeting FPGA devices for the computation of the CLEFIA block cipher algorithm, capable of performing not just data encryption but also the key expansion for 128, 192, and 256-bit keys. One of the main issues in performing the key scheduling for 192 and 256-bit keys is the need for an 8-branch Feistel network, other than the 4-branch Feistel network need for the encryption process and 128-bit key expansion. Experimental results obtained for a Virtex 5 device suggest that a multiple branch Feistel network can be implemented with a minimum efficiency impact. Additionally, the obtained results also suggests that the area impact of the key expansion itself can be significantly reduced by exploring the inner FPGA devices technology. The performance evaluation shows that, with the addressable shift register, the 4 and 8 branch Feistel networks and the full key expansion can be efficiently implemented on the same structure. Such a implementation comes with a cost of 200 Slices and with an efficiency of 5.8 Mbps/Slice, which is about 50% higher than the related state of the art design references, which do not support full key expansion.
Future work will target the application of the obtained results in an ASIC characterization of the proposed architectures.

7. ACKNOWLEDGEMENTS

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

8. REFERENCES

[1] Ricardo Chaves. Compact CLEFIA Implementation on FPGAs. In Peter Athanas, Dionisios Pnevmatikatos, and Nicolas Sklavos, editors, *Embedded Systems Design with FPGAs*, pages 225–243. Springer New York, New York, NY, 2013.

[2] Altera Corporation. An 307 : Altera design flow for xilinx users quartus ii approach to fpga design. Technical Report March, ALTERA Corporation, 2013.

[3] Horst Feistel. Cryptography and Computer Privacy. *Scientific American*, 228(5):15–23, 1973.

[4] Neil Hanley and Maire ONeill. Hardware Comparison of the ISO/IEC 29192-2 Block Ciphers. In *2012 IEEE Computer Society Annual Symposium on VLSI*, pages 57–62. IEEE, August 2012.

[5] T. Kryjak and M. Gorgón. Pipeline implementation of the 128-bit block cipher CLEFIA in FPGA. In *International Conference on Field Programmable Logic and Applications, 2009.*, pages 373–378, Prague, 2009.

[6] Paulo Proença and Ricardo Chaves. Compact CLEFIA Implementation on FPGAs. In *2011 21st International Conference on Field Programmable Logic and Applications*, pages 512–517. IEEE, September 2011.

[7] Atri Rudra, Pradeep K. Dubey, Charanjit S. Jutla, Vijay Kumar, Josyula R. Rao, and Pankaj Rohatgi. Efficient Rijndael Encryption Implementation with Composite Field Arithmetic. In *Cryptographic Hardware and Embedded Systems*, number April, pages 171–184. Springer Berlin Heidelberg, 2001.

[8] Taizo Shirai and Shibutani Kyoji. On Feistel Structures Using a Diffusion Switching Mechanism. In Matthew Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Luxembourg, Luxembourg, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*, chapter Lecture No, pages pp 41–56. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[9] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-Bit Blockcipher CLEFIA (Extended Abstract). In *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, volume 4593, pages 181–195, 2007.

[10] SONY Corporation. The 128-bit Blockcipher CLEFIA Algorithm Specification. Technical report, 2007.

[11] Takeshi Sugawara, Naofumi Homma, Takafumi Aoki, and Akashi Satoh. High-performance ASIC implementations of the 128-bit block cipher CLEFIA. In *2008 IEEE International Symposium on Circuits and Systems*, pages 2925–2928. IEEE, May 2008.

[12] Inc. Xilinx. Xilinx UG190 Virtex-5 FPGA User Guide. *6.375*, 190:1–385, 2012.