2011 21st International Conference on Field Programmable Logic and Applications

COMPACT CLEFIA IMPLEMENTATION ON FPGAS

Paulo Proença and Ricardo Chaves

Instituto Superior Técnico / INESC-ID
Rua Alves Redol 9, 1000-029 Lisbon Portugal
pproenca@sips.inesc-id.pt, ricardo.chaves@inesc-id.pt

ABSTRACT

In this paper two compact hardware structures for the computation of the CLEFIA encryption algorithm are presented. One structure based on the existing state of the art and a novel structure with a more compact organization. This paper shows that, with the use of the existing embedded FPGA components and a careful scheduling, throughputs above 1Gbit/s can be achieved with a resource usage as low as 86 LUTs and 3 BRAMs on a VIRTEX 5 FPGA. Implementation results suggest that a LUT reduction up to 67% can be achieved at a performance cost of 17% on a VIRTEX 4 FPGA, resulting in Throughput/Slice efficiency gains up to 2.5 times, when compared with the related state of the art.

1. INTRODUCTION

In the current digital communication world, digital data is constantly being transmitted through public open channels, whether it is an internet network access or a through the air communication, like in wireless or mobile phone networks. In order to have privacy and access management to that same media, ciphering mechanisms need to be employed when sending sensitive information through these public channels. Ciphering algorithms have been in use for a long time, but the growing processing capabilities of digital equipment and the growing bandwidth for digital communication channels impose the need for more dedicated and secure algorithms. These algorithms can be divided in two classes, asymmetric and symmetric. While the first ones are based on complex mathematical problems, thus having long processing times, the second ones are implemented using operations such as byte substitution, bit permutation and basic arithmetic operations, and can process large amounts of data in small amounts of time.

One of such algorithms is the CLEFIA encryption algorithm, the novel symmetrical block ciphering algorithm proposed and developed by SONY Corporation focused for Digital Rights Management (DRM) purposes [1]. This algorithm improves the security of encryption with the use of techniques such as Diffusion Switch Mechanisms, consisting of multiple diffusion matrices in a predetermined order, to ensure immunity against differential and linear attacks [2,3,4], and the use of Whitening Keys, combining data with portions of the Key before the first round and after the last round. In this research work, FPGAs are selected as the target technology for their advantages in computation adaptability, time to market, development costs, and deployment time of dedicated solutions [5,6].

Two structures for the computation of the CLEFIA symmetrical encryption algorithm are presented in this paper. These structures use the FPGA's embedded BRAMs allowing for a more compact and high throughput hardware implementation. The first structure computes one CLEFIA round per clock cycle, and is based on the topology presented in [7] for an ASIC technology, and adapted in this paper to FPGA technologies. The second structure, herein proposed, further optimizes the area resources by exploring the symmetries of the round computation in this algorithm. This second structure allows to obtain a more compact topology by reusing hardware components, while achieving similar throughputs due to the addition of a pipeline stage. Both the presented structures allow for the computation of the CLEFIA algorithm with all the Key sizes defined in the standard. The related CLEFIA state of the art on FPGAs presented in [8] is also considered. This structure performs the CLEFIA computation on a fully unrolled topology, achieving higher throughputs at the expense of area resources and low flexibility.

While few papers proposing the CLEFIA implementation have been published, and mainly for ASIC technologies, the presented structures are compared with the existing related art. The present analysis suggests improvements in the Throughput per Slice efficiency metric of 1.5 to 2.5 times on several FPGA technologies. Hardware resource reductions up to 67%, at the expense of a throughput reduction of 17% on a VIRTEX 4 FPGA are suggested by the experimental results, for the presented compact structures. Considering the fully unrolled structure proposed in [8], area gains of 48 times can be achieved at a cost of a throughput reduction of 20 times. The structures herein proposed are able to achieve throughputs above 1Gbit/s with a low FPGA resource occupation.

The paper is organized as follows. Section 2 presents a brief description of the CLEFIA algorithm. Section 3 describes the proposed structures and respective implementations on FPGA technologies. Performance evaluation and comparison with the related art are presented in section 4. Section 5 concludes this paper with some final remarks.

2. CLEFIA ENCRYPTION ALGORITHM

The CLEFIA algorithm is a 128 bit block symmetrical

978-0-7695-4529-5/11 $26.00 © 2011 IEEE
DOI 10.1109/FPL.2011.101

encryption algorithm with a Key size varying from 128, 192, to 256 bits. As most current block ciphers, it consists of a Key Scheduling Part and a Data Path computed in multiple rounds, allowing it to be easily implemented in platforms with limited resources [9].

In CLEFIA, state of the art design techniques, present in other ciphering algorithms can also be found, namely: Whitening Keys, a technique used to improve security of iterated block ciphers, consisting in steps to combine data with portions of the key, before the first round and after the last round; Feistel Structures that are the most widely used and the best studied structures for the design of block ciphers, initially proposed by H. Feistel in the early 70's and adopted by the well-known block cipher DES; a Diffusion Switch Mechanism consisting in the usage of multiple diffusion matrices in a predetermined order, to ensure immunity against differential and linear attacks [2,10].

CLEFIAs Data Path uses a 4-branch Feistel structure, an extended version of the traditional 2-branch Feistel structure. It uses two different F-Functions per round, each one having 32 bit input/output data path, as depicted in Figure 1. F-Functions F_0 and F_1, have different Diffusion Matrices, providing CLEFIA with a diffusion switch mechanism. Additional robustness was added to this algorithm with the addition of two Whitening Keys, one added before the main computation round and the other at the end of round operations. The different Key sizes that can be used in CLEFIA (128, 192, or 256 bits) directly influence the number of computed rounds, 18, 22, or 26, respectively [1].

Like in most ciphering algorithms, operations on data consist of byte swapping, byte substitution, and arithmetic operations over finite fields GF(2^8) . The following describes the main operations performed in the CLEFIA algorithm.

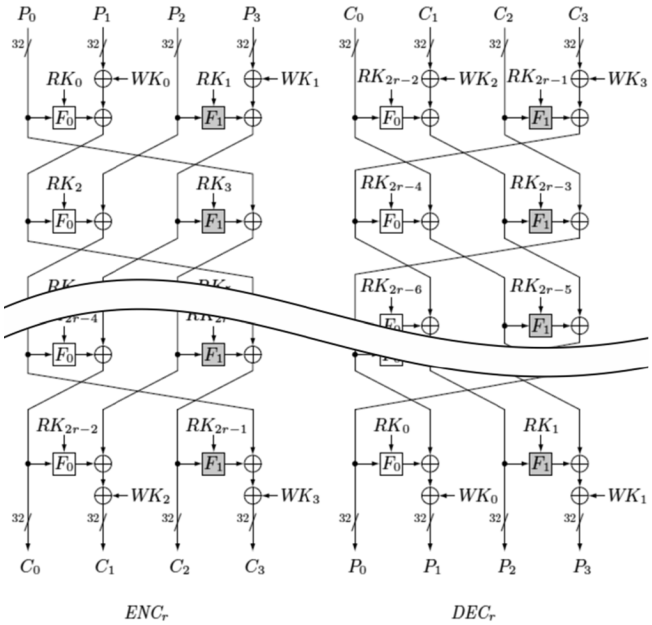

Figure 1. CLEFIA Datapath

2.1. F-Functions

Two different F-Functions, F_0 and F_1 are employed in each round, used for data randomization. These F-Functions consist of additions in GF(2^8) between the round data and the Round Keys; Substitution Boxes S_0 and S_1, and Diffusion Matrices M_0 and M_1, one for each F-Function (F_0 and F_1), as depicted in Figure 2.

Two different types of 8 bit S-boxes are used in each F-Function, S_0 and S_1 [1].

Two different diffusion matrices, M_0 and M_1, are an integral part of the diffusion mechanism present in CLEFIA providing the algorithm with resistance to differential attacks. Each one of the four 8 bit input lines are multiplied by the values in each line of the matrix and additions are performed at the end to finish the operations on these matrices. The constant values used on these matrices suggest some simplifications for the operations needed in these diffusion matrices, as proposed in [7].

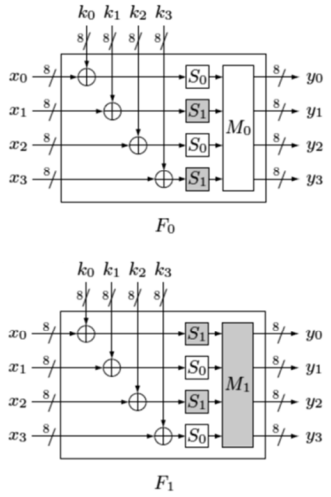

Figure 2. CLEFIAs F-Functions

2.2. Data Processing

The ciphering process in CLEFIA is performed in a sequence of rounds, composed by the mentioned F-Functions, and XOR additions. Four 32 bit Whitening Keys are also added, two of them before the round computations start and two more added after all the rounds are computed, as depicted in Figure 1 as WK_i. Given the Feistel Network based structure of this algorithm the decryption process is identical to the encryption one, using the same computational units, only differing in the order these operations are performed, as also depicted in Figure 1. This inverse computation is achieved by feeding the round key in the inverse order, allowing for the same computational structure to be used [1].

Two 32 bit Round Keys are employed in each round. These Round Keys are obtained from the original Key, as are the Whitening Keys. These Round Keys are added in the F-Functions computation.

2.3. Key Scheduling

In order to obtain the several needed Rounds Keys and Whitening Keys, the ciphering Key needs to be expanded. This expansion is realized by the Key Scheduling part of the CLEFIA algorithm. The Whitening Keys are obtained directly from the Key, depending of the key size [1]. The calculation of the round keys is performed by passing the initial key value through a processing network (GFN) identical to the one used to cipher the data. This GFN network can be a 4 branch structure, similar to the one depicted in Figure 1, used for a 128 bit input Key, or an 8 branch GFN, for 192 and 256 bit input Key sizes [1]. After GFN calculation is completed, the result is expanded using a double swap function (a simple bitwise permutation) and additional constants are added. The resulting values are the needed Round Keys, used in the ciphering data path.

3. PROPOSED CLEFIA STRUCTURES

The main goal on this research work was to provide a compact hardware CLEFIA structure, while still being able to achieve implementations with adequate throughput and performance, even on low cost devices. Two hardware structures are herein presented, one being the derivation of the structure proposed in [7] for ASIC technologies and a second one herein proposed that further optimizes the data path. Both these structures allow for the cipher and decipher computations with all three Key sizes specified in the algorithm.

As described above, the CLEFIA algorithm computation is divided into the Key Scheduling computation and the ciphering computation itself. While the ciphering computation needs to be performed for every 128 bit data block, the key scheduling computation only needs to be computed once for the same ciphering Key value. Moreover, the Key Scheduling computation changes according to the Key size used of 128, 192, or 256 bits, implying additional hardware costs. This lead to the decision to perform the Key Scheduling computation in software and transfer the resulting expanded Key, already computed, to the hardware core during the initialization procedure. Apart from receiving and storing the expanded Keys, the hardware core is also responsible for the transfer and computation of the data to be encrypted or decrypted.

As suggested in [10] and validated by the structures proposed in [7] and [8], faster implementation of CLEFIA can be achieved with the usage of T-boxes. T-Boxes merge the computation of the S-box operations with the linear transformation layers, compressing the resulting structure into a lookup table, also resulting on a reduction of the critical path [11].

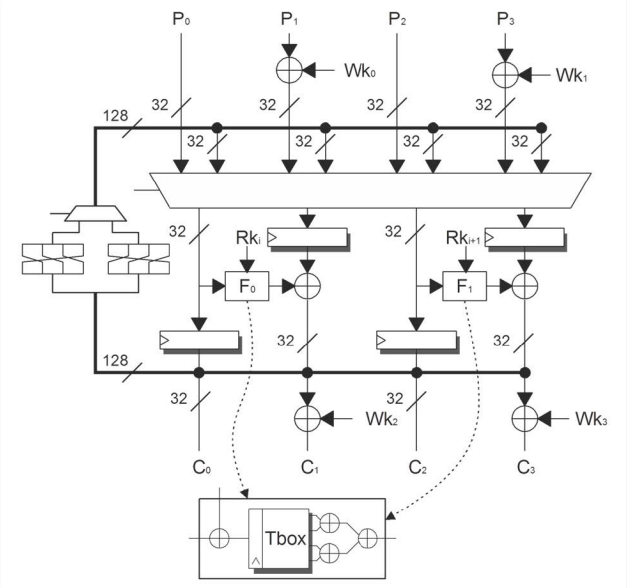

Figure 3. Type-I CLEFIA Structure

In the CLEFIAs F-Functions operation, T-Boxes can be used to replace S_0, S_1, M_0 and M_1, by the lookup operations depicted by (1), followed by XOR operations ($GF(2^8)$ additions) [7], as depicted at the bottom of Figure 3.

$$\begin{aligned}
T_{00} &= (S_0, 02\times S_0, 04\times S_0, 06\times S_0)\\
T_{01} &= (02\times S_1, S_1, 06\times S_1, 04\times S_1)\\
T_{02} &= (04\times S_0, 06\times S_0, S_0, 02\times S_0)\\
T_{03} &= (06\times S_1, 04\times S_1, 02\times S_1, S_1)\\
T_{10} &= (S_1, 08\times S_1, 02\times S_1, 0A\times S_1)\\
T_{11} &= (08\times S_0, S_0, 0A\times S_0, 02\times S_0)\\
T_{12} &= (02\times S_1, 0A\times S_1, S_1, 08\times S_1)\\
T_{13} &= (0A\times S_0, 02\times S_0, 08\times S_0, S_0)
\end{aligned} \quad (1)$$

The resulting T-Boxes have an 8 bit input bus and 32 bit data output. These lookup tables can be implemented in two ways: i) using logic gates (or LUT in FPGAs) [8]; ii) or using dedicated memory blocks. Given that most of the current reconfigurable devices, in particular FPGAs, have dedicated embedded memory blocks designated as BRAMs, the T-Box implementation can be efficiently realized by these components. This allows to achieve faster and less LUT demanding solutions [5]. Further optimizations can be accomplished in terms of resource requirements taking into account that these tables perform identical calculations. Actually, T_{00} and T_{02}, depicted in (1), perform the exact same lookup operation, given the same input, only differing in a 16 bits shift of the output. The same applies to T_{01}/T_{03}, T_{10}/T_{12} and T_{11}/T_{13}. Given this and due to the existence of dual port BRAMs in most FPGA devices, two of these lookup operations can be realized in a single BRAM component. The additional shift operations can be implemented by hardwired routing, without additional area overhead. The remaining hardware required to perform the round computations is composed by a tree of XOR operations

(additions over GF(2^8)) [7].

Apart from the round computation, the addition of the four 32-bit Whitening Keys also need to be performed, two at the beginning and two more in the end of the final round computation. The resulting structure, depicted in Figure 3 is similar to the one proposed in [7], and herein designated as Type-I CLEFIA structure.

In order to obtain an even more compact structure for the CLEFIA implementation, the symmetry between the F_0 and F_1 functions is further explored. The main difference between F_0 and F_1 resides in the M_0 and M_1 tables, as depicted in Figure 2. A more compact structure can be derived by merging the computation of these two tables into a single lookup table. Combining the resulting table for both M_0 and M_1 and taking into consideration the computation structures of the F-functions, a single merged structure able to compute both F_0 and F_1 can be derived.

The resulting merged T-Boxes, capable of computing both the F_0 and F_1, use a 9-bit input divided in two parts, 8 bits for the data and the other one for F-Function selection. As in the Type-I CLEFIA structure, a 32-bit value is outputted by this T-Box. However, for the implementation of these T-Boxes the BRAMs need to store twice the data. While in Type-I the T-Box blocks require 256x32bits=8kbits, in the Type-II structure, the memory block needs 512x32bits=16kbits to store the lookup values. Most FPGA devices have 18Kbit BRAMs units, meaning that for these FPGAs the resulting T-Box blocks for the Type-II structure will occupy the entire BRAM unit, while in Type-I, only half of each used BRAM is occupied.

In the T-Box of Type-II structure, the selection of which function is to be computed within the T-Box is performed by a single bit value at the most significant bit of the address bus of the BRAM, as depicted in Figure 4 by the T_0/T_1 selector in the BRAM.

Being able to perform the lookup operation of the F-functions within a single component, an additional level of folding can be applied, performing the computation of F_0 and F_1 in the same hardware structure. With this technique, approximately half of the hardware resources are needed, apart from the additional selection logic. Consequently the computation of each round will now require two clock cycles, twice as much as in the Type-I structure.

Note that, even though a data dependency exists between the data of each round, with a careful scheduling of the round operations and data storage, round $i+1$ can start its computation before round i has completely finished its computation. With this in mind a pipeline stage can be added to Type-II CLEFIA structure dividing the computation into two stages. Table 1 depicts the proposed scheduling for this computation, where P_i refers to 32-bits of the 128-bit input and P^j refers to the respective round being computed.

In this improved structure the computation of each round is performed in two clock cycles. In the first stage, one of the F-Function is computed by the T-Box structures. In the second stage, the remaining data and Round Key additions are performed. With the proposed schedule, and considering the resulting hardware structure within the FPGA fabric, a pipeline stage can be placed in such a way that stage one and stage two are relatively balanced. The real gain in this structure comes from the fact that while one stage computes one-half of the CLEFIA algorithm, the other stage computes the other half of the CLEFIA algorithm.

Table 1. Type-II Structure Pipeline Scheduling

	First Stage	Second Stage	Output
1	$T_0(P_0^0 + RK_0)$	-	-
2	$T_1(P_2^0 + RK_1)$	$(T_{00} + T_{01} + T_{02} + T_{03}) + WK_0 + P_1^0$	P_0^1
3	$T_0(P_0^1 + RK_2)$	$(T_{10} + T_{11} + T_{12} + T_{13}) + WK_1 + P_3^0$	P_2^1
4	$T_1(P_2^1 + RK_3)$	$(T_{00} + T_{01} + T_{02} + T_{03}) + P_1^1$	P_0^2
5	$T_0(P_0^2 + RK_4)$	$(T_{10} + T_{11} + T_{12} + T_{13}) + P_3^1$	P_2^2
6	$T_1(P_2^2 + RK_5)$	$(T_{00} + T_{01} + T_{02} + T_{03}) + P_1^2$	P_0^3
7	$T_0(P_0^3 + RK_6)$	$(T_{10} + T_{11} + T_{12} + T_{13}) + P_3^2$	P_2^3
…	…	…	…
34	$T_1(P_2^{16} + RK_{33})$	$(T_{00} + T_{01} + T_{02} + T_{03}) + P_1^{16}$	$P_0^{17}=C_0$
35	$T_0(P_0^{17} + RK_{34})$	$(T_{10} + T_{11} + T_{12} + T_{13}) + P_3^{16}$	$P_2^{17}=C_2$
36	$T_1(P_2^{17} + RK_{35})$	$(T_{00} + T_{01} + T_{02} + T_{03}) + P_1^{17} + WK_2$	C_1
37	-	$(T_{10} + T_{11} + T_{12} + T_{13}) + P_3^{17} + WK_3$	C_3

Note that the computation in each round can now be performed in approximately half of the time as in Type-I structure. Thus, it is expected that an approximate ciphering throughput can be achieved, given that no pipeline stalling exists. In order to optimize the data path to the used FPGA technology, the pipeline stage register, depicted in dark in Figure 4, can be placed in different parts of the data path. Several realized implementations, suggested that, in the analyzed devices, this register is best placed at the output of the BRAMs. In all the families of analyzed FPGAs, a register is intrinsically located at the beginning of the BRAMs, which

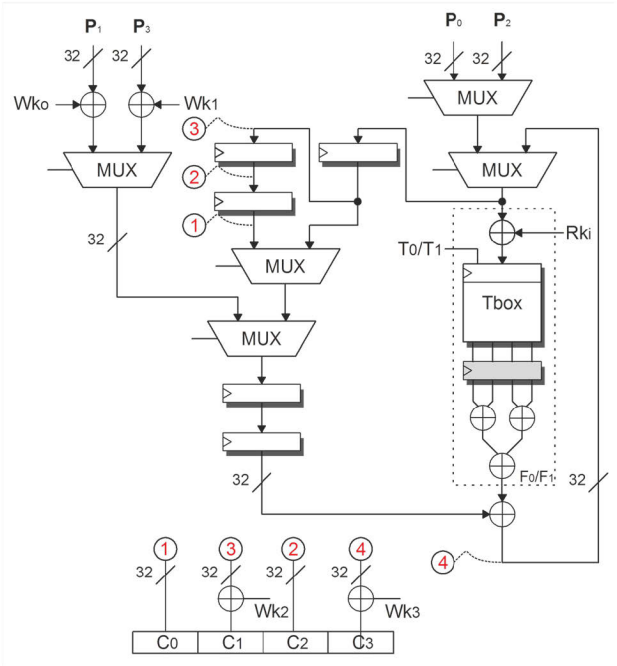

Figure 4. Type-II CLEFIA Structure

cannot be removed, thus defining the frontier of the second pipeline stage.

On the left side of the resulting structure, depicted in Figure 4, a set of registers can be observed. These registers are used to store the temporary round values, needed by the proposed schedule (see Table 1).

4. PERFORMANCE EVALUATION

In this section, experimental results for the proposed structures, on several Xilinx FPGAs technologies, are presented and compared with the work in the existing related art [7,8].

In order to evaluate the presented CLEFIA structures on low cost FPGA devices, the Xilinx Spartan 3E technology was selected. The implementations results, presented in Table 3, suggest that throughputs above 650 Mbit/s can be achieved for both structures at a resource cost of 624 LUT and 5 BRAMs for Type-I structure and 270 LUT and 3 BRAMs for the more compact Type-II structure. Note that in all implementations of the proposed structures an extra BRAM is added. This BRAM is used to store the Round Keys used in the CLEFIA computation, and is herein considered as part of the CLEFIA core.

Results in higher end devices were also obtained, namely for the VIRTEX 4 and VIRTEX 5 technologies. In these FPGAs throughputs between 1.0 and 1.7 Gbit/s are achievable with a relatively low resource cost for both structure types. For the Type-II structure, on a VIRTEX 4, a resource occupation of 205 LUTs and 3 BRAMs is achieved. For the VIRTEX 5 the resource usage is reduced to 86 LUTs and 3 BRAMs. This decrease in LUT usage is due to the fact that the VIRTEX 5 technology has 6 input LUTs while the LUTs in the VIRTEX 4 only have 4. Throughput/Slice efficiency metrics up to 15.1 (Mbps/Slice) are achieved for the proposed Type-II structure. Table 2 presents the obtained throughputs for the two presented CLEFIA structures acording to the Key size. As expected when longer ciphering keys are used the performance efficiency of the ciphering computation decreases.

Considering the related art, the presented structures implemented on FPGA cannot be directly compared with the ones proposed in [7], since these authors focused their work on ASIC technology. Nevertheless, as mentioned above, the presented Type-I structure is similar to the Type-A structure proposed in [7], which suggests the best throughput/area efficiency metric. With this, comparing the presented Type-I structure with the proposed Type-II structure allows us to evaluate the improvements of the proposed modification to the computation structure. Experimental results suggest an area reduction between 50% and 67%, at the expense of a throughput reduction between 14% and 24%, for the SPARTAN 3 and VIRTEX 5 technologies, respectively. These values suggest an improvement of the Throughput/Slice efficiency metric of 1.5 times on the VIRTEX 5 technology, and more significantly of 2 and 2.5 times for the SPARTAN 3E and VIRTEX 4 technologies.

This significant efficiency improvement is due to the component reutilization accomplished by the pipeline and data path rescheduling. Note that, even though the number of cycles needed to cipher a data block doubles, the operating frequency also increases (185 MHz instead of 108 MHz for the SPARTAN 3E), since the computational data path is evenly divided in two.

Note that, while the original structure (Type-A) proposed

Table 2. Summary of obtained Performance Results

		Key Size	Clock Cycles	Mbps	Mbps/Slice
Spartan 3E	Type-I	128	18	768	1.2
		192	22	628	1.0
		256	26	531	0.9
	Type-II	128	36	658	2.4
		192	44	538	2.0
		256	52	455	1.7
Virtex 4	Type-I	128	18	1273	2.0
		192	22	1042	1.7
		256	26	881	1.4
	Type-II	128	36	1045	5.1
		192	44	855	4.2
		256	52	724	3.5
Virtex 5	Type-I	128	18	1707	10.0
		192	22	1396	8.2
		256	26	1181	6.9
	Type-II	128	36	1301	15.1
		192	44	1064	12.4
		256	52	900	10.5

Table 3. Hardware Performance Comparison of CLEFIA Implementations

	Takeshi [7]	**Ours Type-I**	**Ours Type-II**	**Ours Type-I**	**Ours Type-II**	**Kryjak [8]**	**Ours Type-I**	**Ours Type-II**	**Kryjak [8]**
Device	90nm ASIC	XC3S1200-4		XC4LX200-11		XC4 LX200	XC5LX30-3		XC5 LX30
# Slices	21.07*	624	270	625	205	9896	170	86	2612
# BRAMs	n.a	4 + 1	2 + 1	4 + 1	2 + 1	0	4 + 1	2 + 1	0
Max. Frequency (MHz)	746.27	108	185	179	294	167	240	366	167
Latency (cycles)	18	18	36	18	36	18	18	36	18
Throughput (Mbps)	5306	768	658	1273	1045	21376	1707	1301	21376
Throughput/Slice (Mbps/s)	n.a	**1.2**	**2.4**	2.0	**5.1**	**2.1**	10.0	**15.1**	**8.2**

* Kgates not Slices

in [7] performs the Key expansion, the ones herein presented do not. The advantage of our approach is that we are able to cipher with 128, 192, and 256 bit Keys while the one in [7] is only able to cipher using 128 bit Keys. Nevertheless, the above comparison was performed between the two presented structures, which do not perform the Key expansion.

For the implementation of T-Boxes using LUTs, a total of 185 slices would be needed for each T-Box if using a Spartan 3E device. Thus 1480 slices would be needed to implement the 8 required T-Boxes in Type-I structure. Also, the delay imposed by this kind of implementation leads to a longer critical path and consequently to lower frequencies and throughputs. This can be seen by the low working frequencies obtained in [8] when compared to the ones herein proposed.

In [8] a fully unfolded structure is proposed, justifying the extremely high throughput obtained (21Gbit/s). However, this throughput comes at the expense of an excessively high area resource usage as depicted in Table 3. This structure does not allow for the use of encryption modes other than ECB and a key size of 128 bits. On a VIRTEX 4 FPGA a Throughput/Slice efficiency metric of 2.1 is obtained for [8] in comparison with a Throughput/Slice of 5.1 for the proposed Type-II structure. The comparison between the two implementations, suggest a Throughput/Slice of 1.8 to 2.4 times better on a VIRTEX 5 and VIRTEX 4 technologies, respectively. Throughputs significantly above 1Gbit/s were not a target, since most FPGA applications do not require such high throughput values.

In the above discussion the BRAMs needed by the proposed structures were not considered. However, the number of used Slices is within the percentage of used BRAMs, which are available in the FPGA, either we use them or not. In the structure propose in [8] no BRAMs are used.

5. CONCLUSION

In this paper two hardware structures are presented for the implementations of the CLEFIA encryption algorithm. The presented structures were designed having in mind reconfigurable technologies, in particular FPGA with BRAMs, but can be easily targeted to other technologies such as ASIC. While one of the presented structures is similar to the one presented in the related art, and used for comparison, a second structure herein proposed, which collapses the round computation by carefully scheduling the operations with the use of an additional pipeline stage.

The results analysis suggest that with the collapsing of the computation presented in the proposed structure, significant gains in resource usage and in the Throughput/Slice efficiency metric can be achieved. Experimental results suggest that efficiency improvements of 2.5 times can be achieved as well as a reduction in the required LUT up to 67% at a performance cost of 17% on a VIRTEX 4 FPGA.

In conclusion, compact structures for the implementation of the CLEFIA encryption algorithm can be developed and throughputs near 1Git/s can be achieved with low resource usage, even on low cost FPGA devices.

6. ACKNOWLEDGMENTS

This work was supported by the Portuguese Foundation for Science and for Technology (INESC-ID multiannual funding) through the PIDDAC Program funds and by the QREN Programme under contract No. 3487.

7. REFERENCES

[1] T. and Shibutani, K. and Akishita, T. and Moriai, S. and Iwata, T. Shirai, "The 128-Bit Blockcipher CLEFIA (Extended Abstract)" in *Fast Software Encryption*, 2007, pp. 181--195.

[2] Taizo Shirai and Kyoji Shibutani, "On Feistel Structures Using a Difusion Switching Mechanism" in *Fast Software Encryption*, 2006, pp. 41--56.

[3] H. and Wu, W. and Feng, D. Chen, "Differential fault analysis on CLEFIA" in *Proceedings of the 9th international conference on Information and communications security*, 2007, pp. 284--295.

[4] Y. and Tsujihara, E. and Shigeri, M. and Suzaki, T. and Kawabata, T. Tsunoo, "Cryptanalysis of CLEFIA using multiple impossible differentials" , 2009, pp. 1--6.

[5] Francisco Rodriguez-Henriquez, N.A. Saqib, A. Diaz-Perez, and Çetin Kaya Koç, *Cryptographic Algorithms on Reconfigurable Hardware*.: Springer, 2006.

[6] AJ Elbirt, W. Yip, B. Chetwind, and C. Paar, "An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists" in *The Third AES Candidate Conference, printed by the National Institute of Standards and Technology, Gaithersburg, MD*, 2000, pp. 13--27.

[7] T. Sugawara, N. Homma, T. Aoki, and A. Satoh, "High Performance ASIC Implementation of the 128-bit Block Cipher CLEFIA" in *ISCAS 2008. IEEE International Symposium on Circuits and Systems*, 2008, pp. 2925--2928.

[8] Tomasz Kryjak and Marek Gorgon, "Pipeline Implementation ot the 128-bit Block Cipher CLEFIA in FPGA" in *FPL 2009. International Conference on Field Programmable Logic and Applications*, 2009, pp. 373--378.

[9] T. Shirai and A. Mizumo, "A Compact and High-Speed Cipher Suitable for Limited Resources Environment" in *3rd ETSI security wrokshop presentation*, Sophia-Antipolis, France, 2007.

[10] SONY Corporation. (Revision 1.0, June 1, 2007) The 128-bit Block Cipher CLEFIA Security and Performance Evaluations. [Online]. http://www.sony.net/Products/ cryptography/clefia/technical/data/clefia-eval-1.0.pdf

[11] T. and Benaissa, M. Good, "AES on FPGA from the Fastest to the Smallest" *Cryptographic Hardware and Embedded Systems*, pp. 427--440, 2005.