

FPGA implementations of the ICEBERG block cipher

F.-X. Standaert*, G. Piret, G. Rouvroy, J.-J. Quisquater

UCL Crypto Group, Place du Levant, 3, B-1348 Louvain-La-Neuve, Belgium

Abstract

This paper presents field programmable gate array (FPGA) implementations of ICEBERG, a block cipher designed for reconfigurable hardware implementations and presented at FSE 2004. All its components are involutational and allow very efficient combinations of encryption/decryption. The implementations proposed also allow changing the key and encrypt/decrypt (E/D) mode for every plain text, without any performance loss. In comparison with other recent block ciphers, the implementation results of ICEBERG show a significant improvement of hardware efficiency. Moreover, the key and E/D agility allow considering new encryption modes to counteract certain side-channel attacks.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Block ciphers; Hardware implementations; FPGAs

1. Introduction

In October 2000, National Institute of Standards and Technology (NIST) selected Rijndael as the new Advanced Encryption Standard. The selection process included performance evaluation on both software and hardware platforms. However, as implementation versatility was a criteria for the selection of the AES, it appeared that Rijndael was not optimal for reconfigurable hardware implementations. For example, its highly expensive substitution boxes may be a bottleneck in certain contexts. The combination of encryption and decryption in hardware is usually another critical point (e.g. see [14–16,21]).

ICEBERG is a block cipher designed for efficient reconfigurable hardware implementations. It is based on an involutational structure so that the forward and inverse operation of the cipher may be performed with exactly the same hardware. All its components easily fit into the 4-bit input lookup tables of FPGAs, and its key scheduling allows the round keys to be derived “on the fly” in encryption and decryption mode. In addition to hardware efficiency, the key and E/D agility allow considering new encryption modes to

counteract certain side-channel attacks. In practice, very low-cost hardware crypto-processors and high-throughput data encryption are potential applications of ICEBERG.

This paper presents FPGA implementations of ICEBERG and compares their performances with the ones of recent block ciphers (e.g. AES and NESSIE candidates). Although ICEBERG implementations offer features that most block ciphers do not provide (e.g. key and E/D agility), its implementation results exhibit a significant improvement of hardware efficiency. For this purpose, we investigated various contexts (loop and unrolled implementations, with or without feedback) on the recent Xilinx Virtex-II[®] FPGAs.

The paper is structured as follows. Section 2 briefly presents the specifications of ICEBERG and Section 3 describes our FPGA design methodology. Section 4 lists the combinatorial cost of the block cipher components. The implementation results for various architectures are in Section 5 and comparisons with other block ciphers are in Section 6. Resistance against side-channel analysis is briefly discussed in Section 7. Finally, conclusions are in Section 8.

2. Specifications

2.1. Block and key size

ICEBERG operates on 64-bit blocks and uses a 128-bit key. It is an involutational iterative block cipher based on the

*Corresponding author.

E-mail addresses: standaert@dice.ucl.ac.be (F.-X. Standaert),
piret@dice.ucl.ac.be (G. Piret), rouvroy@dice.ucl.ac.be (G. Rouvroy),
quisquater@dice.ucl.ac.be (J.-J. Quisquater).

repetition of 16 identical key-dependent round functions. In the next subsections, we briefly present the algorithm. A more detailed description can be found in the original paper [1].

2.2. The round function

The round function is pictured in Fig. 1, where we distinguish a non-linear layer and a linear diffusion layer.

The *non-linear layer* is built from the parallel application of 8×8 substitution boxes to the cipher state. For efficiency purposes, these boxes are constructed from smaller 4×4 S-boxes $S0$, $S1$ and bit permutations $P8$ (i.e. 8-bit wire crossings).

The *linear diffusion layer* is built from bit permutations $P64$ (i.e. 64-bit wire crossings), bit permutations $P4$ (i.e. 4-bit wire crossings), bitwise key additions (denoted as $\oplus$ in the figure) and small 4×4 diffusion boxes D . These boxes perform a simple multiplication:

$$\begin{pmatrix} y_3 \\ y_2 \\ y_1 \\ y_0 \end{pmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \times \begin{pmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix},$$

where every output bit is a $\oplus$ operation between three input bits. It is therefore efficiently combined with the key addition inside a single 4-input LUT.

2.3. The key schedule

The key scheduling process consists of key expansion and key selection.

The *key expansion* expands the cipher key K into a sequence of keys $K^0, K^1, \dots, K^{16}$. We set the initial key $K^0 = K$. The following keys are obtained by a keyround function so that: $K^{i+1} = \text{keyround}(K^i)$.

The *keyround* is pictured in Fig. 2, where we distinguish a conditional shift layer, bit permutations $P128$ (i.e. 128-bit wire crossings) and S-boxes $S0$. The conditional shift operation depends on a round constant C that will be discussed further.

Finally, the *key selection* first performs a simple compression function that selects 64 bytes of K^i having odd indices. Thereafter, a 4×4 key selection box is applied in parallel to every 4-bit key-block. It performs the following boolean operation:

$$\begin{aligned} y(0) &= (x(0) \oplus x(1) \oplus x(2)) \cdot \text{sel} \vee (x(0) \oplus x(1)) \cdot \overline{\text{sel}}, \\ y(1) &= (x(1) \oplus x(2)) \cdot \text{sel} \vee x(1) \cdot \overline{\text{sel}}, \\ y(2) &= (x(2) \oplus x(3) \oplus x(0)) \cdot \text{sel} \vee (x(2) \oplus x(3)) \cdot \overline{\text{sel}}, \\ y(3) &= (x(3) \oplus x(0)) \cdot \text{sel} \vee x(3) \cdot \overline{\text{sel}}. \end{aligned}$$

Depending on the value of a selection bit sel , we obtain the round key RK_0^i or RK_1^i for round i .

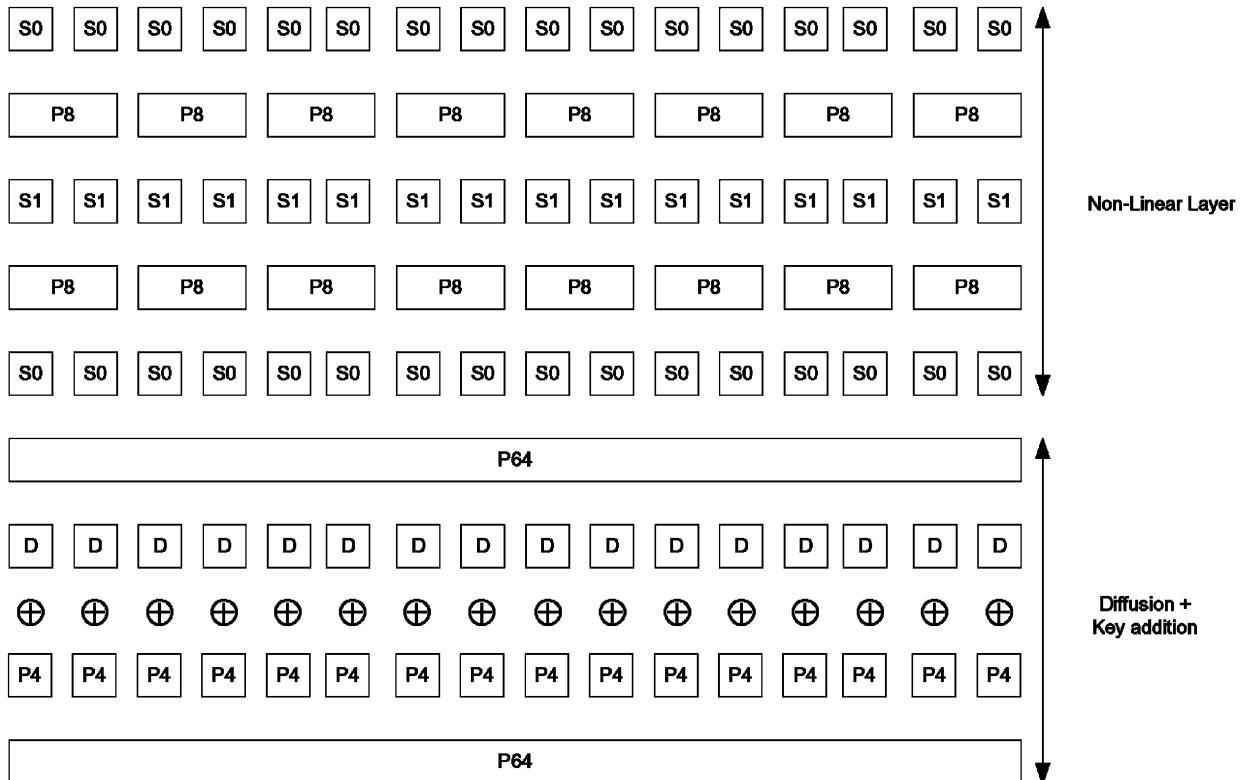


Fig. 1. The round function.

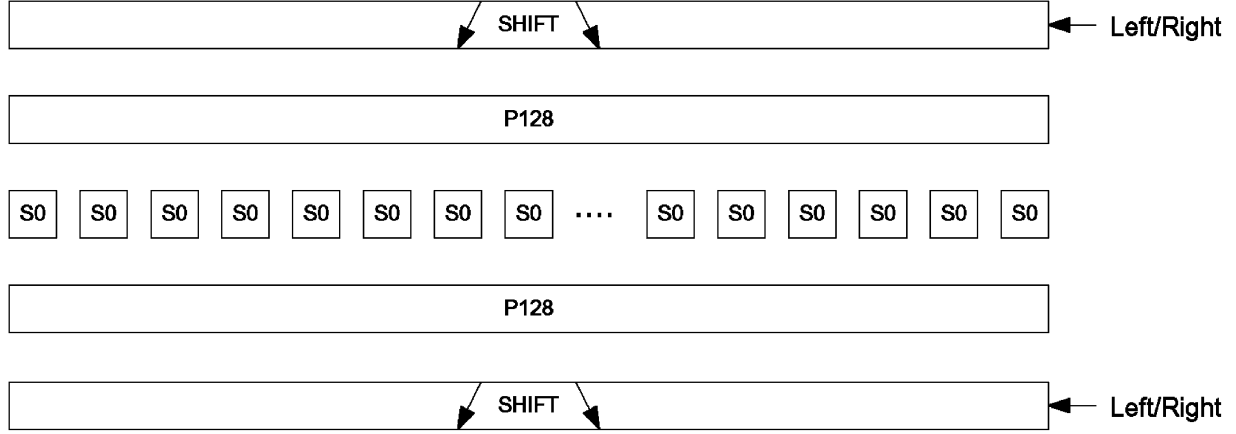


Fig. 2. The key round.

2.4. Encryption/decryption process

The complete cipher consists of an initial round key addition, 15 rounds and a final transform. Due to the involutonal structure of every single component of ICEBERG, the E/D mode is fixed with the selection bit only: $sel = 1$ in encryption and $sel = 0$ in decryption. Let us denote the key addition under a key K as σ_K , the round non-linear layer as γ and the round linear layer (including the key addition) as ε_K . Then, ICEBERG is defined for the cipher key K , as the transformation $\text{ICEBERG}[K] = \alpha_R[RK_1^0, RK_1^1, \dots, RK_0^R]$ applied to the plain text where:

$$\alpha_R[RK_1^0, RK_1^1, \dots, RK_0^R] = \sigma_{RK_0^R} \circ \gamma \circ \left(\bigcirc_{r=1}^{R-1} \rho_{RK_1^r} \right) \circ \sigma_{RK_1^0}$$

The standard number of rounds is $R = 16$. Similarly, for decryption, we have

$$\alpha_R^{-1}[RK_1^0, RK_0^1, \dots, RK_0^R] = \sigma_{RK_1^0} \circ \gamma \circ \left(\bigcirc_{r=R-1}^1 \varepsilon_{RK_0^r} \circ \gamma \right) \circ \sigma_{RK_0^R}$$

The key sequence $RK_1^0, RK_0^1, \dots, RK_0^R$ is obtained by the execution of R key rounds and key selection layers. Finally, the round constants are: $C = 0$ until round 8, $C = 1$ thereafter. A particular structure of the expanded key is therefore obtained:

$$\begin{aligned} K^0 &= K^R, \\ K^1 &= K^{R-1}, \\ K^2 &= K^{R-2}, \\ &\dots \end{aligned}$$

As a consequence, ICEBERG allows the encryption/decryption with exactly the same hardware (only the selection bit has to be changed) and the expanded key may be derived “on the fly” in encryption and decryption (the storage of round keys is not necessary). More details about this particular structure are available in the paper of FSE 2004.

3. Design methodology

Present reconfigurable components like FPGAs are usually made of reconfigurable logic blocks combined with fast access memories (RAM blocks) and high-speed arithmetic circuits [2,3]. Basic logic blocks of FPGAs include a 4-input function generator (called lookup table, LUT) and a storage element. In addition, most FPGA manufacturers provide users with fast carry logic and particular structures of the logic blocks to efficiently implement distributed memories, shift registers, ... A complete description of these components can be found in the Xilinx data sheets [2].

As reconfigurable components are divided into logic elements and storage elements, an efficient implementation will be the result of a better compromise between combinatorial logic used, sequential logic used and resulting performances. These observations lead to different definitions of implementation efficiency:

- (1) In terms of performances, let the efficiency of a block cipher be the ratio *Throughput (Mbits/s)/Area (LUTs, RAM blocks)*.
- (2) In terms of resources, the efficiency is easily tested by computing the ratio *Nbr of registers/Nbr of LUTs*: it should be close to one.

ICEBERG was designed in order to allow very efficient FPGA implementations and our architectures are defined in order to maximize these notions of hardware efficiency. It practically results in the pipelining of the round and keyround functions. Pipelining increases the encryption speed by processing multiple blocks of data simultaneously. It is achieved by inserting rows of registers among combinatorial logic. Parts of logic between two consecutive registers form pipeline stages and we define the *maximum pipeline* as the pipeline of which the number of stages implies that the ratio *Nbr of registers/Nbr of LUTs* is the closest to one (and lower than one).

Table 1
Combinatorial cost of ICEBERG components

Round components	HW cost (LUTs)	Keyround components	HW cost (LUTs)
S_0, S_1 layers	64	Shift layer	128
Non-linear layer	$64 \times 3 = 192$	S_0 layer	128
Linear diffusion layer	64	Keyround	384
Round	256	Selection layer	64

Finally, depending on the optimization criteria, different architectures can be employed. Optimization for maximum speed can be achieved by a fully pipelined unrolled architecture. In the applications requiring minimum area, a loop architecture with only one round implemented seems to be the best choice. For both cases, we tried to maximize the previously defined efficiency. In addition, we provide results of non-pipelined implementations that are useful when a block cipher is used in feedback modes. In the next sections, we present the results of various FPGA implementations of ICEBERG.

4. Combinatorial cost of ICEBERG

As all components perfectly fit into 4-input LUTs, we can directly evaluate their combinatorial cost in the Xilinx Virtex-II[®] family of devices (Table 1).

Remark that if the maximum pipeline is not inserted, the shift layers can be efficiently implemented inside the Virtex slice, using additional multiplexers $F5$ and $F6$ available next to the LUT [2].

In the next section, we investigate the practical implementation of different architectures for ICEBERG.

5. Implementation results

All the architectures proposed in this section allow the choice of the key and E/D mode for every plain text. The area and frequency estimations presented are provided after implementation with Xilinx ISE[®] 6.1 on the Xilinx Virtex-II[®] technology. The timing constraints were applied to the inner clock and we used the input–output (IO) registers embedded into the FPGA IOBs¹ in order to take the interface constraints into account. It is important to note that the limiting factor of our work frequencies was always the IO management. As an illustration, the internal clock of the fully pipelined unrolled implementation without IO registers is near to the maximum (380 MHz), but if IO registers are considered, it decreases to 297 MHz, what we believe to be a fair frequency estimation.

5.1. Unrolled architectures

For high-throughput applications, we propose an unrolled implementation with the 16 rounds implemented

and we applied two pipelining strategies. If a maximum throughput is required, a full pipe implementation is provided, with the maximum pipeline inserted. However, for large designs, the implementation (and specially the routing task) may become the bottleneck, with routing delays larger than logic delays. Therefore, for an optimized efficiency, we propose the half pipe architecture. In addition to a better tradeoff between logic and routing delays, it also allows an efficient implementation of the shift layer, using the additional multiplexers available inside the Virtex slice. Both architectures are pictured in Fig. 3. Finally, if the half pipe architecture is considered, we can also implement the round S-box inside the FPGA RAM blocks. The implementation results for these three proposals are in Table 2.

5.2. Loop architectures

In the applications requiring minimum area, we propose a loop architecture with only one round implemented. In order to decrease the area requirements, we only considered the half pipe strategy. In addition to the efficiency advantages already mentioned, half pipe structures are specially convenient for loop architectures because they allow the combination of the loop multiplexer with the round and keyround logic. Our proposal is pictured in Fig. 3, where we share the initial and final key addition. As for unrolled architectures, it is possible to use the FPGA RAM blocks to implement the round S-box. The implementation results for these loop architectures are provided in Table 3.

5.3. Feedback modes

As soon as a feedback mode is used, pipelining techniques are not relevant for block cipher implementations. This is due to the fact that multiple blocks of data cannot be managed in parallel because encrypting one block of data requires the result of the previously encrypted block. Although we do not recommend the use of feedback modes in FPGA implementations of block ciphers (they do not allow us to take advantage of hardware efficiency), we propose the following designs for comparison purposes. An unrolled architecture without pipelining and a minimum latency loop architecture are represented in Fig. 4. The implementation results of these designs are in Table 4.

¹IOBs: Input–output blocks.

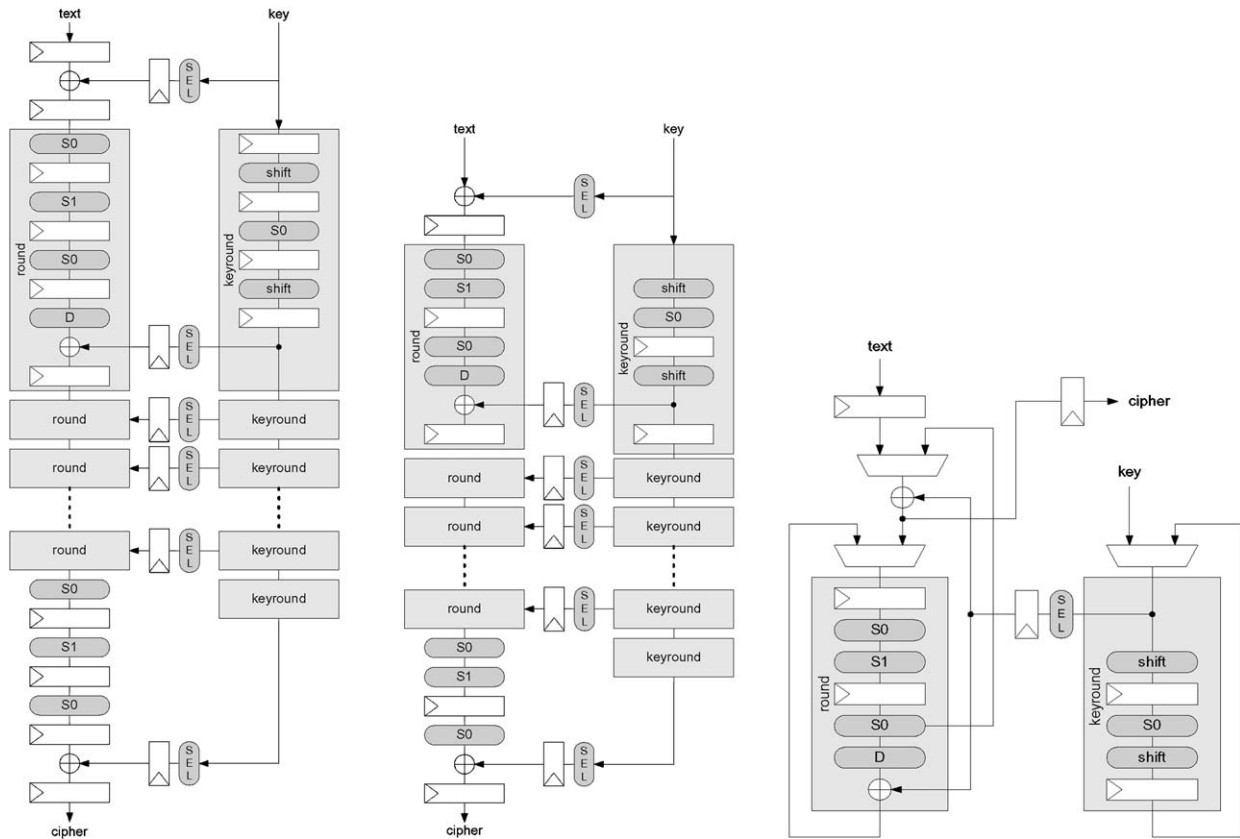


Fig. 3. Full pipe unrolled, half pipe unrolled and loop architectures.

Table 2
Unrolled architectures results on Virtex-II®

Type	# of slices	# of RAMBs	Latency (cycles)	Out. every (cycles)	Frequency (MHz)	Throughput (Mbits/s)
Full pipe	6808	0	66	1	297	19008
Half pipe	4946	0	33	1	271	17344
RAM	3132	64	33	1	210	13440

Table 3
Loop architecture results on Virtex-II®

Type	# of slices	# of RAMBs	Latency (cycles)	Out. every (cycles)	Frequency (MHz)	Throughput (Mbits/s)
Loop	631	0	34	2/32	254	1016
RAM	526	4	34	2/32	227	908

6. Comparisons with other block ciphers

Comparing the performances of block cipher hardware implementations is generally a delicate task. This is due to the high dependency of these implementation results on the design methodology, but also to the various commercial FPGAs that may be chosen for evaluation. In the case of ICEBERG, it is even more critical as our implementations provide key *and* *E/D* agility: two properties that are never combined in other block cipher implementations (excepted in the Triple-DES). The

following considerations must therefore be taken with care and should be considered as general guidelines more than as a strict comparison. We tried to find the best results for various block ciphers in non-feedback modes, if possible in the most recent technology (Virtex-II®). Then, we provide the area and throughput results. If no RAMBs are used, the ratio throughput/area is given in order to estimate the hardware efficiency. We also specify the architecture used (loop or unrolled) and its basic features (encryption only, encryption/decryption, key agility) (Tables 5 and 6).

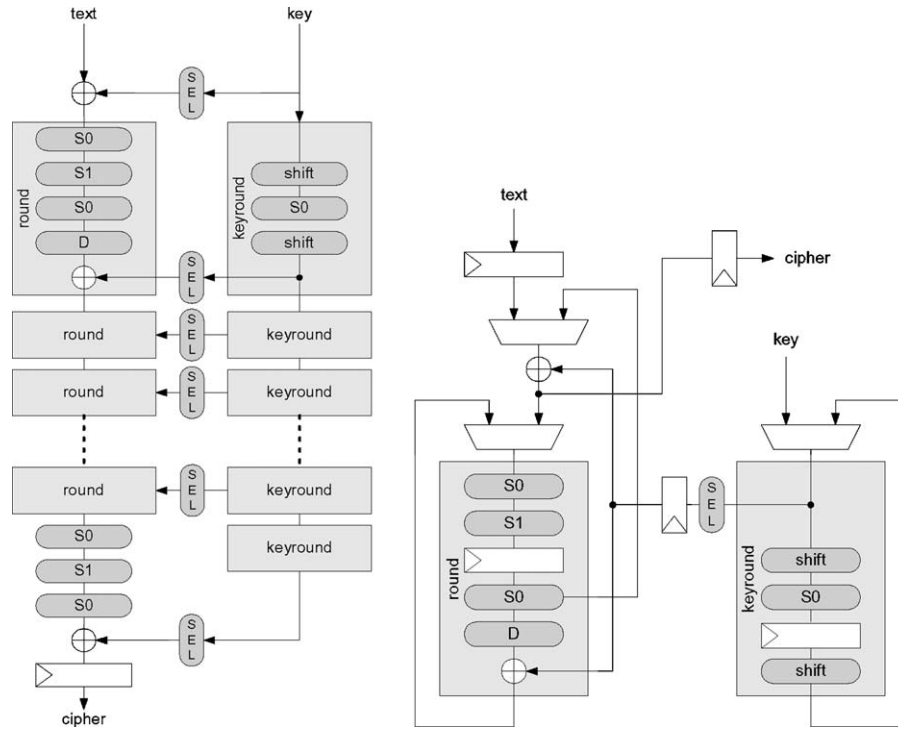


Fig. 4. Feedback mode architectures.

Table 4
Feedback mode results on Virtex-II®

Type	# of slices	# of RAMBs	Latency (cycles)	Out. every (cycles)	Frequency (MHz)	Throughput (Mbits/s)
Unrolled	3174	0	1	1	14	896
Loop	571	0	17	1/16	147	588
RAM	467	4	17	1/16	145	580

In general, ICEBERG implementations exhibit a significant improvement of the hardware efficiency, even if we compare them with encryption only designs. It is clear that the most relevant implementation schemes for ICEBERG do not use RAMBs because they considerably increase the S-box memory requirements (The ICEBERG S-box memory requirements are: $(2^4 \times 4) \times 6 = 384$ bits). If RAMBs are used, it becomes $2^8 \times 8 = 2048$ bits). LUT-only implementations are also the best estimators for ASIC performances and underline the excellent potentialities of ICEBERG for hardware implementations in general. More specifically, only Rijndael [12] and the 3DES have an efficiency comparable to ICEBERG with an E/D structure. However, the specified Rijndael implementation does not provide key and E/D agility, uses RAM blocks and shares resources between the round and keyround. For 3DES, it is well known that it allows very efficient implementation opportunities and having a comparable efficiency is probably an excellent result for ICEBERG.

7. Resistance against side-channel attacks

Although cryptosystem designers frequently assume that secret parameters will be manipulated in closed reliable computing environments, Kocher et al. [17] stressed in 1998 that actual computers and microchips leak information correlated to the data handled. Side-channel attacks based on time, power and electromagnetic measurements were successfully applied to smart card implementations of block ciphers. Protecting implementations against side-channel attacks is usually difficult and expensive. Masking all the data with random boolean values is suggested in several papers [18,19] and the use of small substitution tables allows this to be efficiently implemented, although it is still an expensive solution (the additional cost of masking a 2^n -bit table is another 2^{2n} -bit table).

The key agility provided by ICEBERG (changing the key at every plain text block is for free) also offers interesting opportunities to prevent certain side-channel attacks by defining new encryption modes where the key is changed

Table 5
Basic features of compared block ciphers

Algorithm	Device	Enc.	Dec.	Key ag.	Loop/Unr.
0.22 μ m					
Twofish [4]	Virtex [®]	•		•	U
Serpent [4]	Virtex [®]	•		•	U
0.18 μ m					
Rijndael [5]	Virtex-E [®]	•		•	U
Camelia [6]	Virtex-E [®]	•		•	U
Khazad [7]	Virtex-E [®]	•		•	U
Misty1 [7]	Virtex-E [®]	•		•	U
Rijndael [5]	Virtex-E [®]	•		•	
0.15 μ m					
RC6 [8]	Virtex-II [®]	•		•	U
IDEA [9]	Virtex-II [®]	•		•	U
SHACAL-1 [10]	Virtex-II [®]	•		•	U
3DES [11]	Virtex-II [®]	•	•	•	U
ICEBERG	Virtex-II [®]	•	•	•	U
3DES [11]	Virtex-II [®]	•	•	•	L
ICEBERG	Virtex-II [®]	•	•	•	L
0.15 μ m + RAMBs					
Rijndael [12]	Virtex-II [®]	•	•		L
ICEBERG	Virtex-II [®]	•	•	•	L
Rijndael [13]	Virtex-II [®]	•	•		L
ICEBERG	Virtex-II [®]	•	•	•	U

Table 6
Performances of compared block ciphers

Algorithm	# Slices	# RAMBs	Throughput (Mbits/s)	Thr./area (Mbits/s/slices)
0.22 μ m				
Twofish [4]	21 000	0	15 200	0.72
Serpent [4]	19 700	0	16 800	0.85
0.18 μ m				
Rijndael [5]	2784	100	11 776	—
Camelia [6]	9692	0	6750	0.7
Khazad [7]	7175	0	7872	1.10
Misty1 [7]	6322	0	10 176	1.61
Rijndael [5]	2524	0	2085	1.17
0.15 μ m				
RC6 [8]	7456	0	4800	0.64
IDEA [9]	9793	0	6800	0.69
SHACAL-1 [10]	13 729	0	17 021	1.24
3DES [11]	604	0	917	1.51
ICEBERG	4946	0	17 344	3.51
3DES [11]	227	0	326	1.44
ICEBERG	631	0	1016	1.61
0.15 μ m + RAMBs				
Rijndael [12]	146	3	358	—
ICEBERG	526	4	908	—
Rijndael [13]	$\approx$ 1125	18	1408	—
ICEBERG	3132	64	13 440	—

The bold lines correspond to the results of the implementations of the ICEBERG algorithm, i.e., the new results proposed in the paper. The other lines are references to previous works.

sufficiently often. As most side-channel attacks need to collect several leakage traces to remove the noise from useful information, changing the key frequently, even in a well-chosen deterministic way (e.g. LFSR-based), could help to counteract (or at least make more difficult) these attacks. A thorough analysis of side-channel resistance based on re-keying techniques would deserve further research and analysis.

8. Conclusion

We presented FPGA implementations of ICEBERG, a block cipher designed for hardware implementations. In terms of area requirements, throughput and hardware efficiency, ICEBERG exhibits excellent abilities compared to most recent block ciphers. The simplicity of the design is also considerably improved and allows the fast development of an efficient architecture. In practice, an unrolled (resp. loop) architecture has a throughput of 17.3 Gbits/s (resp. 1.0 Gbits/s), using 4946 FPGA slices (resp. 631 FPGA slices) in the Xilinx Virtex-II[®] technology. In addition, ICEBERG allows key and E/D agility. These properties could be used to improve resistance against certain side-channel attacks, although this last point is let as a scope for further research. Due to the simplicity of its component functions, ICEBERG is also likely to exhibit excellent implementation results in hardware in general (not only FPGAs).

References

- [1] F.-X. Standaert, G. Piret, G. Rouvroy, J.-J. Quisquater, J.-D. Legat, ICEBERG: an involutational cipher efficient for block encryption in reconfigurable hardware, in: Proceedings of FSE 2004, The Fast Software Encryption Workshop, New Delhi, February 5–7, 2004, Springer, Berlin.
- [2] Xilinx: Virtex 2 FPGAs Data Sheet, (<http://www.xilinx.com>).
- [3] Altera: Stratix 1.5V FPGAs Data Sheet, (<http://www.altera.com>).
- [4] K. Gaj, P. Chodowicz, Fast implementation and fair comparison of the final candidates for the advanced encryption standard using field programmable gate arrays, in: Proceedings of the RSA Security Conference—Cryptographer's Track, April 8–12, San Francisco, CA, 2001, pp. 84–99.
- [5] F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, J.-D. Legat, Efficient implementation of Rijndael encryption in reconfigurable hardware: improvements and design tradeoffs, in: Proceedings of CHES 2003, Lecture Notes in Computer Science, vol. 2779, Springer, Berlin, 2003, pp. 334–350.
- [6] T. Ichikawa, T. Sorimachi, T. Kasuya, M. Matsui, On the criteria of hardware evaluation of block ciphers, Technical Report of IEICE, ISEC 2001.
- [7] F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, J.-D. Legat, Efficient FPGA implementation of block ciphers Khazad and Misty1, Third NESSIE Workshop, Munich, Germany, November 2002.
- [8] J.-L. Beuchat, FPGA implementation of the RC6 block cipher, in: Proceedings of FPL 2003, Lecture Notes in Computer Science, vol. 2778, Springer, Berlin, 2003, pp. 101–110.

- [9] J.-L. Beuchat, Modular multiplication for FPGA implementation of the IDEA block cipher, in: *Proceedings of ASAP 2003, Application Specific Systems Architectures and Processors*, IEEE, Silverspring, MD, 2003.
- [10] M. McLoone, J.V. McCanny, Very high speed 17 Gbps SHACAL encryption architecture, in: *Proceedings of FPL 2003, Lecture Notes in Computer Science*, vol. 2778, Springer, Berlin, 2003, pp. 111–120.
- [11] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, J.-D. Legat, Optimizing cipher FPGA implementations: DES and triple DES, in: *Proceedings of FPL 2003, Lecture Notes in Computer Science*, vol. 2778, Springer, Berlin, 2003, pp. 181–193.
- [12] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, J.-D. Legat, Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications, in: *Proceedings of ITCC 2004, Las Vegas, April 5–7, 2004*.
- [13] Helion Encryption Cores: (<http://www.heliontech.com/>).
- [14] P. Chodowicz, K. Gaj, Very compact FPGA implementation of the AES, in: *Proceedings of CHES 2003, Lecture Notes in Computer Science*, vol. 2779, Springer, Berlin, 2003, pp. 319–333.
- [15] A.J. Elbirt, W. Yip, B. Chetwynd, C. Paar, An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists, in: *Proceedings of the Third AES Candidate Conference*, April 13–14, 2000, New York, USA.
- [16] B. Weeks, M. Bean, T. Rozyłowicz, C. Ficke, Hardware performance simulations of round 2 advanced encryption standard algorithms, NSA Final Report on AES candidates, available from (<http://csrc.nist.gov/CryptoToolkit/aes/round2/NSA-AESfinalreport.pdf>).
- [17] P. Kocher, J. Jaffe, B. Jun, Differential power analysis, in: *Proceedings of CRYPTO 99, Lecture Notes in Computer Science*, vol. 1666, Springer, Berlin, 1999, pp. 398–412.
- [18] L. Goubin, J. Patarin, DES and differential power analysis: the duplication method, in: *Proceedings of CHES 1999, Lecture Notes in Computer Science*, vol. 1717, Springer, Berlin, 1999, pp. 158–172.
- [19] S. Chari, et al., Towards sound approaches to counteract power-analysis attacks, in: *Proceedings of CRYPTO 1999, Lecture Notes in Computer Science*, vol. 1666, Springer, Berlin, 1999, pp. 398–412.
- [21] J. Wolkenstorfer, E. Oswald, M. Lamberger, An ASIC implementation of the AES S-boxes, in: *Proceedings of CT-RSA 2002, Lecture Notes in Computer Sciences*, vol. 2271, Springer, Berlin, 2002, pp. 67–78.

François-Xavier Standaert was born in Brussels, Belgium in 1978. He received the Electrical Engineering degree and Ph.D. degree from the Université catholique de Louvain, respectively, in June 2001 and June 2004. In 2005, he was a Fulbright visiting researcher at Columbia University (Networks Security Laboratory) and MIT Medialab. He is now a post doctoral researcher funded by the FNRS (Funds for National Scientific Research, Belgium), at the UCL Crypto Group. His research interests include digital design and FPGA's, cryptographic hardware, design of cryptographic primitives and side-channel analysis.

Gilles Piret received the degree of Engineer in Applied Mathematics in 1998, and a 3rd cycle master in Mathematics in 1999, both from the Université Catholique de Louvain (Belgium). Between 1999 and 2005, he worked as a research assistant in the UCL Crypto group. In 2005, he graduated as a Ph.D. His thesis deals with block ciphers, and was carried out under the direction of professor J.-J. Quisquater. Since then, he works as a post-doctoral researcher at the Computer Science Departement of ENS (France). His research interests include all aspects of symmetric cryptography: block ciphers, stream ciphers, provable security, side channel analysis and implementation of these primitives.

Gael Rouvroy was born in Brussels, Belgium, in 1978. He obtained the Electromechanical Engineering degree and the Ph.D. degree from the Université catholique de Louvain in 2001 and 2004. He is currently a post-doctoral researcher at the electrical engineering department of the same university. His research interests include digital design and FPGAs, cryptographic hardware, block ciphers, cryptanalysis, image processing (JPEG 2000) and watermarking.

Jean-Jacques Quisquater is Professor of Cryptography and Multimedia Security at the Department of Electrical Engineering, University of Louvain, Louvain-la-Neuve, Belgium. He is responsible, at least at the scientific level, of many projects related to smart cards (protocols, implementations, side-channels), secure protocols for communications, digital signatures, payTV, protection of copyrights and security tools for electronic commerce. He was the main designer of several coprocessors for powerful smart cards: CORSAIR (Philips) and FAME (Philips). He holds 17 patents in the field of smart cards. He is co-inventor of the so-called GQ cryptographic identification scheme.