6.78 Gigabits per Second Implementation of the IDEA Cryptographic Algorithm

Antti Hämäläinen, Matti Tommiska, and Jorma Skyttä

Signal Processing Laboratory
Helsinki University of Technology
Otakaari 5 A
FIN-02150, Finland
{Antti.Hamalainen, Matti.Tommiska, Jorma.Skytta}@hut.fi

Abstract. IDEA (International Data Encryption Algorithm) is one of the strongest secret-key block ciphers. The algorithm processes data in 16-bit subblocks and can be fully pipelined. The implementation of a fully pipelined IDEA algorithm achieves a clock rate of 105.9 MHz on Xilinx' XCV1000E-6BG560 FPGA of the Virtex-E device family. The implementation uses 18105 logic cells and achieves a throughput of 6.78 Gbps with a latency of 132 clock cycles.

1 Introduction

Cryptography is the science of keeping communication secure, so that eavesdroppers cannot decipher the transmitted messages. The transmission speeds of core networks require hardware-based cryptographic modules, since software-based cryptography cannot meet the required throughput requirements.

Field programmable gate arrays (FPGAs) are ideal components for fast cryptographic algorithms. The large capacities of FPGAs enable the fitting of fully pipelined algorithms on a single chip. The reprogrammability of FPGAs enables using the same hardware platform as a cryptographic engine for a multitude of communications protocols.

Cryptographic algorithms are divided into public-key and secret-key algorithms. In public-key algorithms both public and private keys are used, with the private key computed from the public key. Secret-key algorithms rely on secure distribution and management of the session key, which is used for encrypting and decrypting all messages. When it comes to both software- and hardware-based implementations, secret-key algorithms are 100 to 1000 times faster than public-key algorithms. For this reason, dual-key sessions use a secret-key algorithm for the bulk of communication, whereas the session specific secret keys are agreed on and distributed with a public key algorithm.

The International Data Encryption Algorithm (IDEA) was introduced by Lai and Massay in 1990 [1], and modified the following year [2]. IDEA has been patented in the U.S. and several European countries, but the non-commercial use of IDEA is free everywhere. The patent holder was originally Ascom AG,

M. Glesner, P. Zipf, and M. Renovell (Eds.): FPL 2002, LNCS 2438, pp. 760–769, 2002.

but in 1999 the intellectual property rights were transferred to Mediacrypt AG. Part of the fame of IDEA is due to its usage in Pretty Good Privacy (PGP).

IDEA is considered highly secure, and it has resisted all forms of attack tried by the academic community. No published attack (with the exception of attacks on weak keys) is better than exhaustive search on the 128-bit key space, which is computationally infeasible. The security of IDEA appears bounded only by the weaknesses arising from the relatively small (compared to its keylength) blocklength of 64 bits. [3]

Unlike many other cryptographic algorithms, IDEA can easily be implemented on 16-bit microcontrollers, since the algorithm operates on 16-bit subblocks. In 1999, a software-based implementation of four parallel IDEA algorithms (4-way IDEA) achieved a throughput of the order of 72 megabits per second (Mbps) on a 166 MHz MMX Pentium processor [4]. If this result is scaled to modern 2.533 GHz Pentium 4 processors, a software-based implementation of a 4-way IDEA achieves a throughput of 1.1 gigabits per second (1.1 Gbps). This sets a reference point for hardware-based implementations.

There have been several reported hardware implementations of IDEA in the published literature. Mediacrypt AG sells two hardware-based IDEA solutions, the IDEACrypt Coprocessor and the IDEACrypt Kernel. The IDEACrypt kernel is faster of these two and implements the IDEA algorithm with a three-stage pipeline. A 0.25 micron implementation has a throughput of 720 Mbps at a clock rate of 100 MHz. [5]

The Improved IDEA chip by Salomão et al. [6] achieved a throughput of 809 Mbps at a 100 MHz clock rate. If eight of these devices are connected in series, an estimated throughput of 6.5 Gbps can be achieved.

Cheung et al. [7] have investigated the tradeoffs in parallel and serial implementations of the IDEA algorithm. The parallel implementation achieved a throughput of 1.17 Gbps on a Xilinx Virtex XCV300-6 at a clock rate of 82 MHz, whereas the serial implementation achieved a throughput of 600 Mbps on the same device at a clock rate of 150 MHz. When two rounds of the IDEA algorithm were implemented, the parallel implementation required 79.56 per cent of the logic resources of an XCV300-6. The serial implementation of a single round required 93.68 per cent of the logic resources of the same device. It was estimated, that by utilizing linear scaling of the area requirements, a fully pipelined parallel implementation of the IDEA algorithm would fit into a XCV1000 Virtex device with a device utilization of 94.42 per cent. This would correspond to a throughput of 5.25 Gbps, if the clock rate remained unchanged.

2 Description of the IDEA Algorithm

IDEA encrypts 64-bit plaintext blocks into 64-bit ciphertext blocks using a 128-bit input key K. The algorithm consists of eight identical rounds followed by an output transformation. Each round uses six 16-bit subkeys $K_i^{(r)}$, $1 \leq i \leq 6$, to transform a 64-bit input X into an output of four 16-bit blocks, which are then input to the next round. All subkeys are derived from the 128-bit input

key K. The subkey derivation process is different in decryption mode from the encryption mode, but otherwise encryption and decryption are performed with identical hardware.

IDEA uses only three operations on 16-bit sub-blocks a and b: bitwise XOR denoted by $\oplus$, unsigned addition mod (2^{16}) denoted by $\boxplus$ and modulo $(2^{16} + 1)$ multiplication, denoted by $\odot$. All these three operations are derived from different algebraic groups of 2^{16} elements, which is crucial to the algorithmic strength of IDEA. Of the three arithmetic operations, bitwise XOR and unsigned addition mod (2^{16}) are trivial to implement, whereas a both area-efficient and fast implementation of modulo $(2^{16} + 1)$ multiplication requires careful design and bit-level optimisation. The IDEA computation path is described in Fig. 1.

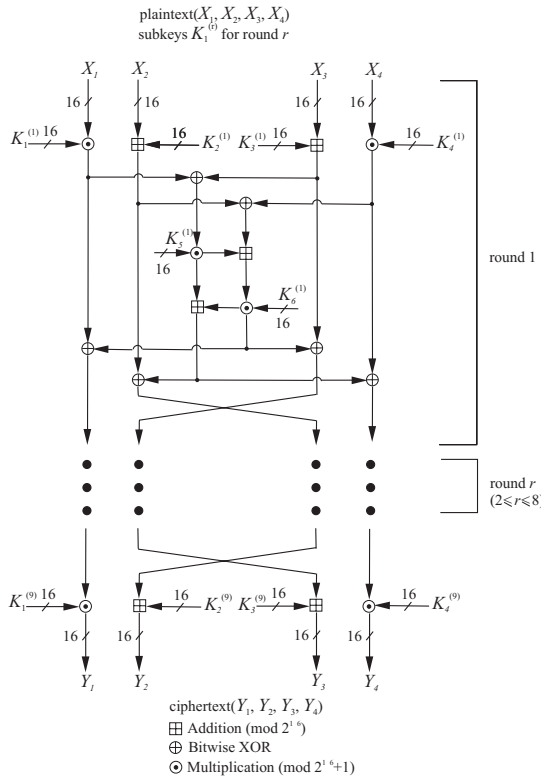

Fig. 1. The IDEA cryptographic algorithm

Except for key scheduling, the IDEA algorithm is defined as follows: [3]

INPUT: 64-bit plaintext $M = m_1...m_{64}$; 128-bit key $K = k_1...k_{128}$.
OUTPUT: 64-bit ciphertext block $Y = (Y_1, Y_2, Y_3, Y_4)$.

1. (Key schedule) Compute 16-bit subkeys $K_1^{(r)}, ..., K_6^{(r)}$ for rounds $1 \leq r \leq 8$, and $K_1^{(9)}, \ldots, K_4^{(9)}$ for the output transformation.

2. $(X_1, X_2, X_3, X_4) \leftarrow (m_1 \ldots m_{16}, m_{17} \ldots m_{32}, m_{33} \ldots m_{48}, m_{49} \ldots m_{64})$, where X_i is a 16-bit data store.
3. For round r from 1 to 8 do:
   (a) $X_1 \leftarrow X_1 \odot K_1^{(r)}$, $X_4 \leftarrow X_4 \odot K_4^{(r)}$, $X_2 \leftarrow X_2 \boxplus K_2^{(r)}$, $X_3 \leftarrow X_3 \boxplus K_3^{(r)}$.
   (b) $t_0 \leftarrow K_5^{(r)} \odot (X_1 \oplus X_3), t_1 \leftarrow K_6^{(r)} \odot (t_0 \boxplus (X_2 \oplus X_4)), t_2 \leftarrow t_0 \boxplus t_1$.
   (c) $X_1 \leftarrow X_1 \oplus t_1, X_4 \leftarrow X_4 \oplus t_2, a \leftarrow X_2 \oplus t_2, X_2 \leftarrow X_3 \oplus t_1, X_3 \leftarrow a$.
4. (Output transformation) $Y_1 \leftarrow X_1 \odot K_1^{(9)}, Y_4 \leftarrow X_4 \odot K_4^{(9)}, Y_2 \leftarrow X_3 \boxplus K_2^{(9)}, Y_3 \leftarrow X_2 \boxplus K_3^{(9)}$.

In IDEA, $a \odot b$ corresponds to modulo $(2^{16}+1)$ multiplication of two unsigned 16-bit integers a and b, where $0 \in \mathbb{Z}_{2^{16}}$ is associated with $2^{16} \in \mathbb{Z}_{2^{16}+1}$as follows: if $a = 0$ or $b = 0$, replace it by 2^{16} (which is $\equiv -1 \mod (2^{16}+1)$) prior to modular multiplication; and if the result is 2^{16}, replace this by 0. Decryption is achieved with the ciphertext Y provided as input M. Key scheduling is described in standard textbooks on cryptography [3] [8], and its hardware requirements are negligible when compared to modulo $(2^{16}+1)$ multipliers.

2.1 Diminished-One Number Representation

The diminished-one number representation is often used in arithmetic modulo $(2^n + 1)$ [9]. In diminished-one number system the number A is represented by $A' = A - 1$ and the value 0 is represented by 2^n. In IDEA, $n = 16$, and consequentially, the value 0x0000 as a 16-bit unsigned integer is represented by 0x10000 in diminished-one representation.

The usage of diminished-one number system is advantageous in the implementation of modulo $(2^{16}+1)$ multipliers. There are also downsides in using the diminished-one number system: additional logic is required in adding up partial products and conversions are required to/from ordinary 16-bit unsigned integers. However, the advantages of using the diminished-one number system outweigh the disadvantages in the design described in this paper.

3 Design and Implementation

With the introduction of million-gate FPGAs, the implementation of fully unrolled secret-key cryptographic algorithms became feasible. If the entire algorithm with full inner and outer loop pipelining fits on a single FPGA, the limiting factor for throughput is the achieved clock rate as follows: [10]

$$Throughput = block\_size \times clock\_rate \tag{1}$$

Since the block size of IDEA is fixed at 64 bits, a 100 MHz clock rate implies a throughput of 6.4 Gbps. Clock rates above 100 MHz can be achieved in modern FPGAs by carefully analysing the algorithm, partitioning the design into stages and pipelining the entire system. The disadvantage is increased latency

measured in clock cycles, but this can usually be tolerated, since throughput is the dominant design factor in high-speed applications. The high-level block diagram of the fully pipelined IDEA implementation is described in Fig. 2

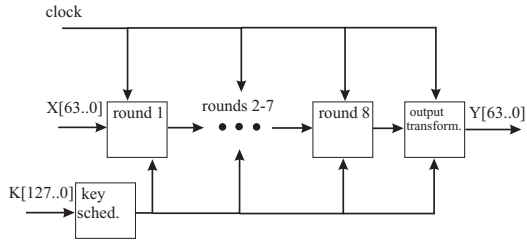

Fig. 2. Fully pipelined IDEA algorithm

A fully unrolled round of the IDEA algorithm is described in Fig. 3, which can be compared with Fig. 1. There are seven subrounds in one round, and the modulo $(2^{16}+1)$ multiplier is further divided into four sub-subrounds (See 3.2). A single round is calculated in 16 clock cycles, since three subrounds (1, 3, and 5) with modulo $(2^{16}+1)$ multipliers each require four clock cycles and four subrounds (2, 4, 6 and 7) are executed in a single clock cycle. After the 8th round, $8 \times 16 = 128$ clock cycles have been used. The final output transformation requires additional four clock cycles, since two modulo $(2^{16}+1)$ multiplication operations are executed in parallel. This adds up to a total latency of 132 clock cycles, which corresponds to 1.246 μs with the maximum clock rate of 105.9 MHz (See also Table 3).

3.1 Design Flows

The design was initially going to be implemented only in Handel-C [11], a high-level hardware description language with built-in directives for implementing parallelism in hardware. The design flow consisted of Celoxica's DK1 Design Suite v1.0 SP1, which produced a structural VHDL output file for logic synthesis with Synplicity's Synplify Pro 7.0 and subsequent place and route with Xilinx' ISE Foundation Series 4.1i.

A single round of the IDEA algorithm was coded in Handel-C, with the clock rate exceeding 80 MHz. However, as individual rounds were connected together, the performance of the overall design decelerated and the area requirements of the entire design increased in a non-linear manner for unknown reasons. Furthermore, compilation times exceeded ten hours, which made it impractical to improve the design within a single working day.

To achieve a clock rate of at least 100 MHz, it was decided to recode the entire design in synthesisable VHDL with Synplify Pro 7.0. Investigating the critical path revealed that the carry-save adder (CSA) structure (See 3.2) [12] was the most time-consuming part. The CSA structure was replaced with a simple three-stage adder tree, which both reduced area and increased the clock rate over the

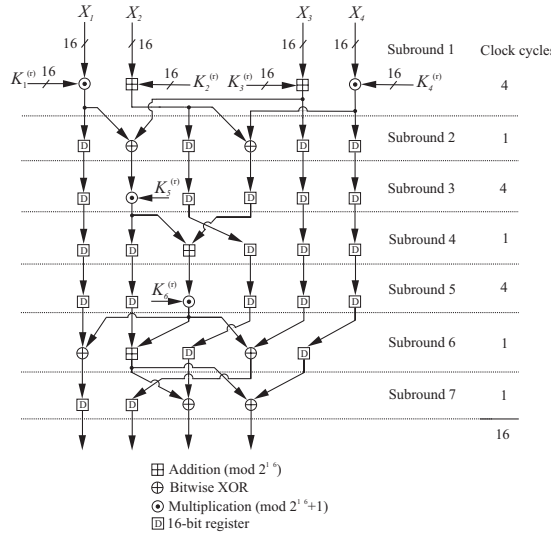

Fig. 3. A fully unrolled round of the IDEA algorithm

targeted 100 MHz. When the design was targeted for XCV1000E-6BG560, a clock rate of 105.9 MHz was achieved with 18105 logic cells (LCs).

The design flows are compared in Table 1 with further details in Table 3.

Table 1. A comparison of the two design flows. Note that the implementation of modulo $(2^{16} + 1)$ multiplier was coded in different manner in Handel-C and VHDL.

Main Design Tool	Language	Other Tools	Performance of one round	Entire IDEA algorithm	Additional information
DK1 Design Suite	Handel-C	Synplify Pro 7.0 ISE 4.1i ModelSim	87.3 MHz 2902 LCs XCV1000E-6	13h compilation did not fit into an XCV2000E-6	CSA Array (See 3.2)
Synplify Pro 7.0	VHDL	ISE 4.1i ModelSim	105.9 MHz 2122 LCs XCV1000E-6	105.9 MHz 18105 LCs XCV1000E-6	Three-stage adder tree (See 3.2)

3.2 Modulo $(2^{16} + 1)$ Multiplication

The critical part of an efficient hardware implementation of IDEA is the modulo $(2^{16} + 1)$ multiplication operator. There has been a lot of academic activity in researching an optimum implementation of the modulo $(2^{16} + 1)$ multiplier [12], [13], [14], but the research has been limited to full-custom design.

The partial product generation proposed by Ma [12] was used. The inputs to the partial product generation logic are 16-bit unsigned integers a and d in

diminished-one representation. Since the second input d is always a subkey, it can be converted in advance in the key scheduling block. After simplifications, the eight partial products $p_0, \ldots, p_7$ are generated by a set of eight 8-to-1 multiplexers, whose control input is a 3-bit wide sequence from the subkey. Additional combinational logic is required for cyclic modulo left shifts, but the design fits into little over 200 logic cells. An outline of the multiplexer bank is presented in Fig. 4.

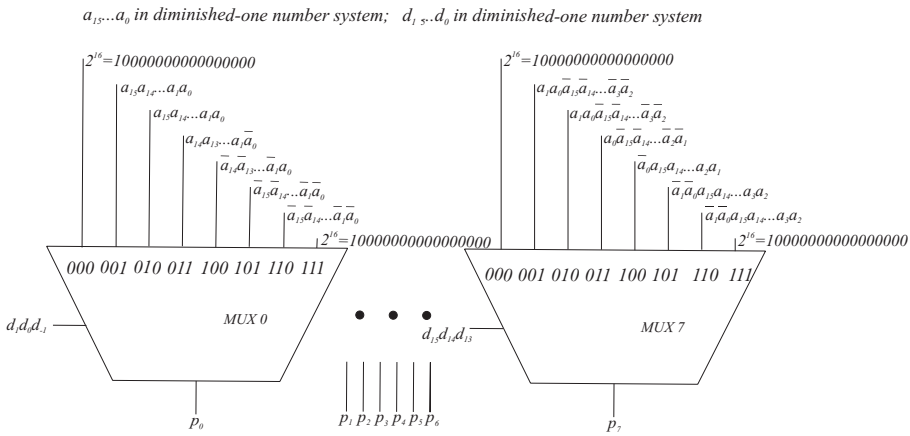

Fig. 4. An outline of the multiplexer bank. The outputs are eight partial products $p_0 \ldots p_7$ in diminished-one notation.

The produced partial products $p_0, \ldots, p_7$ are 17 bits wide, since zero is represented by 0x10000 in diminished-one number representation. To obtain the multiplication result, the partial products have to be summed together.

The CSA structure in [12] was coded in Handel-C, whereas a simpler three-stage adder tree was coded in synthesisable VHDL (See 3.1). It was noted, that the three-stage adder tree required fewer logic resources in the targeted devices of the Virtex-E family. CSA structures do not save area resources in FPGAs, but require more area resources than a straightforward implementation of the partial products summation. This is due to the efficient implementation of fast look-ahead carry logic chains in modern FPGAs, which leaves no practical room for optimisation of adder structures.

When adding two numbers a and b represented in diminished-one notation, attention must be given to the special case of zero. If zero is not used, addition in diminished-one number system looks as follows: [13]

$$(a+b+1) \bmod (2^n+1) = \begin{cases} (a+b) \bmod 2^n, & \text{if } a+b \geq 2^n \\ a+b+1, & \text{otherwise} \end{cases} \tag{2}$$

Modulo $(2^n + 1)$ addition can be realised by an end-around-carry adder, where the carry-out is inverted and fed back into the carry-in, i.e. $c_{in} = \overline{c_{out}}$.

This can be realized with two adders to prevent a combinational loop. The carry-out inversion logic does not work when both summands equal zero. Therefore an additional AND gate has to be added to produce the control input for a multiplexer, which selects the correct output $a + b = 0x10000$, when $a = b = 0x10000$. There are seven modulo $(2^n + 1)$ adders to sum up the eight partial products. This is described in detail in Fig. 5

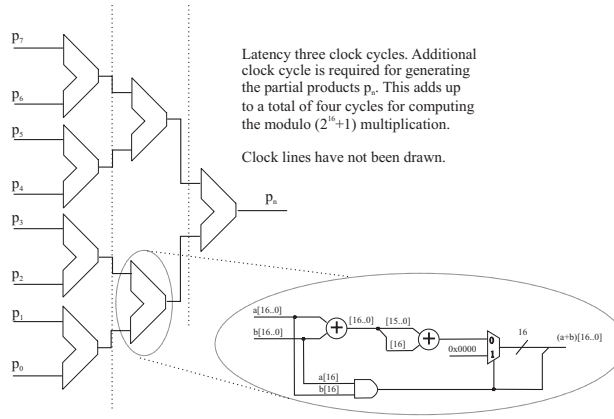

Fig. 5. The three-stage diminished-one adder tree used in IDEA implementation. The inner structure of a diminished-one adder includes extra control logic for the output multiplexer.

3.3 Results

The logic cell (LC) requirements of the seven subrounds (See Fig. 3) are summarized in Table 2. About 93 per cent of the area requirements of a single round are caused by modulo $(2^{16} + 1)$ multipliers implemented with a simple three-stage adder tree (See Fig. 5).

As mentioned, the two implementations of the modulo $(2^{16} + 1)$ multiplier differ in their area requirements, because a straightforward implementation of the three-level adder tree fits into a much smaller number of LCs than a more sophisticated CSA design (See also 3.2). A single modulo $(2^{16} + 1)$ multiplier implemented in Handel-C utilizing the CSA scheme [12] required 919 LCs compared to 463 LCs for the simpler VHDL-based multiplier.

To compare the results with those projected in [7], the VHDL-based design was targeted for an XCV1000E-6 device, and the area and timing characteristics are presented in Table 3. For comparison purposes, the design was compiled for certain Altera FPGAs, but these results must be viewed with caution, since no device-specific optimisations were made.

To verify the design in hardware, the IDEA algorithm was targeted for Xilinx' XCV2000E-6BG560 device on an RC1000 PCI card [15]. Logic synthesis with

Table 2. The area requirements of a single IDEA round in an XCV1000E-6.

Subround	LCs	Additional Information
1	969	2 multipliers, 2 adders
2	32	2 XORs
3	533	1 multiplier
4	16	1 adder
5	533	1 multiplier
6	48	2 XORs, 1 adder
7	32	2 XORs
Total	2163	4 multipliers, 4 adders, 6 XORs
After minimisation	2125	

Table 3. Timing and area characteristics of the entire IDEA algorithm.

	XCV1000E-6	XCV2000E-6	APEX20KC-8	EP2A40-8
LCs (Xilinx) ATOMs (Altera)	18105	18233	37289	37413
Device Utilisation	73 %	45 %	71 %	97%
Clock rate	105.9 MHz	105.9 MHz	32.0 MHz	66.2 MHz
Throughput	6.78 Gbps	6.78 Gbps	2.05 Gbps	4.24 Gbps
Latency (132 cycles)	1.25μs	1.25μs	4.13μs	1.99μs

Synplify Pro 7.0 and place and route with Xilinx' ISE Foundation Series 4.1i reported the same maximum clock rate of 105.9 MHz (See Table 3).

The RC1000 card has a programmable clock circuit with a maximum frequency of 100 MHz. The design fit into 45 per cent of the logic resources of an XCV2000E, and the functionality was verified with test vectors and with varying clock rates. Since the maximum clock frequency was 100 MHz, the functionality could not quite be tested at the reported maximum clock rate of 105.9 MHz.

4 Conclusions

The design and implementation of the IDEA algorithm at 6.78 Gbps on a single XCV1000E-6BG560 proves that entire unrolled and pipelined complex cryptographic functions can be implemented on a single FPGA. The limiting factors in achieving maximal throughput are the block size of the algorithm and the clock rate, which can be increased by carefully analysing the algorithm and partitioning the individual operations into subblocks executed in a single clock cycle.

An FPGA-based cryptographic module is a strong candidate, when high-performance cryptography is required. Applications include Virtual Private Networks (VPNs), satellite communications and hardware accelerators for encrypting huge files or entire disks.

An interesting research area is partial reprogrammability applied to cryptography. This is especially true for algorithms, which process data in small-sized blocks, whose other input is directly computed from the session key. IDEA is an

example of this kind of an algorithm. Xilinx' Virtex FPGAs support partial runtime reconfiguration [16], and the area requirements of IDEA could be reduced by precalculating the optimum implementation for modulo $(2^{16}+1)$ multiplication for every 65536 subkeys. When a session key is changed, the device would be partially reconfigured with optimal modular multipliers.

References

1. X. Lai and J. L. Massey. "A proposal for a new block encryption standard", *Advances in Cryptology – EUROCRYPT 90*, volume 473 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 389–404.
2. X. Lai, J. L. Massey and S. Murphy "Markov ciphers and differential cryptanalysis", *Advances in Cryptology – EUROCRYPT 91,* pp. 17–38.
3. A.J. Menezes, PC. van Oorschot, S.A. Vanstone: *"Handbook of Applied Cryptography"*, CRC Press Ltd., 1997, pp. 263–266.
4. H Lipmaa. "IDEA: A Cipher for Multimedia Architectures?" *Selected Areas in Cryptography '98*, volume 1556 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 253–268.
5. IDEACrypt Kernel, http://www.mediacrypt.com/engl/Content/cryptkernel.htm
6. S.L.C. Salomão, J.M.S. Alcântara, V.C. Alves, F.M.G. Franca: "Improved IDEA", *Proceedings of the 13th Brazilian Symposium on Integrated Circuits and System Design - SBCCI 2000,* September 2000, Manaus, Brazil, pp. 47–52.
7. O.Y.H. Cheung, K.H. Tsoi, P.H.W. Leong, M.P. Leong, "Tradeoffs in Parallel and Serial Implementations of the International Data Encryption Algorithm IDEA", *Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems,* May 14–16, 2001, Paris, France, pp. 333–347.
8. B. Schneier: *"Applied Cryptography"*, John Wiley & Sons, 1996, pp. 319–324.
9. L.M. Leibowitz: "A Simplified Binary Arithmetic for the Fermat Number Transform", *IEEE Transactions on Acoustics, Speech and Signal Processing,* Vol. 24, No. 5, October 1976, pp. 356–359.
10. P. Chodowiec, P. Khuon, K. Gaj: "Fast Implementations of Secret-Key Block Ciphers Using Mixed Inner- and Outer-Round Pipelining", *Proceedings of the ACM/SIGDA Ninth International Symposium on Field Programmable Gate Arrays*, February 11-13, 2001, Monterey, California, USA, pp. 94–102.
11. *"Handel-C Language Reference Manual, Version 2.1"*, Celoxica Ltd., 2001
12. Y. Ma: "A Simplified Architecture for Modulo $(2^n + 1)$ Multiplication", *IEEE Transactions on Computers*, Vol. 47, No. 3, March 1998, pp. 333–337
13. R. Zimmermann: "Efficient VLSI Implementation of Modulo $(2^n \pm 1)$ Addition and Multiplication", *Proceedings of the 14th IEEE Symposium on Computer Arithmetic,* April 1999, Adelaide, Australia, pp. 158–167.
14. A. Curiger, H. Bonnenberg, H. Kaeslin: "Regular VLSI Architectures for Multiplication Modulo (2^n+1)", *IEEE Journal of Solid-state Circuits,* Vol. 26, No. 7, July 1991, pp. 990–994.
15. RC1000 Data Sheet, Celoxica RC1000, http://www.celoxica.com/products /technical_papers/datasheets/DATRHD002_0.pdf
16. Virtex Series Configuration Architecture User Guide, XAPP151 (v1.5), Xilinx Inc.