CryptoBooster:
A Reconfigurable and Modular Cryptographic Coprocessor

Emeka Mosanya[1], Christof Teuscher[1], Héctor Fabio Restrepo[1], Patrick Galley[2], and Eduardo Sanchez[1]

[1] Logic Systems Laboratory, Swiss Federal Institute of Technology
CH-1015 Lausanne, Switzerland.
http://lslwww.epfl.ch
E-mail: {name.surname}@epfl.ch,

[2] LIGHTNING Instrumentation SA
Av. des Boveresses 50, CH-1010 Lausanne, Switzerland.
http://www.lightning.ch,
E-mail: Patrick.Galley@lightning.ch

Abstract. The CryptoBooster is a modular and reconfigurable cryptographic coprocessor that takes full advantage of current high-performance reconfigurable circuits (FPGAs) and their partial reconfigurability. The CryptoBooster works as a coprocessor with a host system in order to accelerate cryptographic operations. A series of cryptographic modules for different encryption algorithms are planned. The first module we implemented is IDEACore, an encryption core for the International Data Encryption Algorithm ($IDEA^{TM}$).

Keywords: Cryptography, Coprocessor, Reconfiguration, FPGA, IDEA.

1 Introduction

In this paper we describe a novel cryptographic coprocessor, the CryptoBooster, optimized for reconfigurable computing devices (e.g., Field Programmable Gate Arrays, or FPGAs). Our implementation is modular, scalable, and helps to resolve the trade-off between device size and data throughput. The implementation is designed to support a large number of different encryption algorithms and includes appropriate session management.

The first CryptoBooster implementation we propose implements $IDEA^{TM1}$, a symmetric-key block cipher algorithm [7]. IDEACore is the first of a series of cryptographic modules for the CryptoBooster coprocessor. A simple reconfiguration of the reconfigurable computing device will suffice to replace IDEACore by another block cipher module (e.g., DES). The other modules of the CryptoBooster generally remain unchanged.

---

[1] IDEA is patented in Europe and the United States [11, 12]. The patent is held by Ascom Systec Ltd., http://www.ascom.ch/systec.

In section 2 we describe the CryptoBooster architecture. Section 3 gives a short introduction to the $IDEA^{TM}$ encryption algorithm and reviews existing implementations of this algorithm in hardware. Section 4 describes the implementation of IDEACore, the first cryptographic module for the CryptoBooster. We conclude in section 5 the paper with the synthesis and performance results of our implementation.

2   CryptoBooster

CryptoBooster is a modular coprocessor dedicated to cryptography. It is designed to be implemented in Field Programmable Gate Arrays (FPGAs) and to take advantage of their partial reconfiguration features. The CryptoBooster works as a coprocessor with a host system in order to accelerate cryptographic operations. It is connected to a session memory responsible for storing session information (Figure 1). A session is characterized by a set of parameters describing the cryptographic packets, the algorithm used, the key(s), the initial vector(s) for block chaining, and other information.
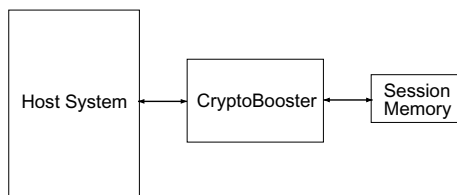
Fig. 1. The CryptoBooster works as a coprocessor together with a host system. Typically, the host system is a PC. The CryptoBooster needs additional memory to store session information.

Our design is motivated by the following objectives: (1) The main goal is to have maximum data throughput to provide a design able to cope with ever-increasing network speed. This justifies hardware implementation in place of a software solution. Physical security may, however, also be an argument for hardware implementation. (2) Our requirements include the ability to easily configure different algorithm sub-blocks and the associated subkey generation. We clearly need a highly modular architecture, allowing us to easily change building blocks. The modularity has been pushed far enough to allow partial reconfiguration of the coprocessor. Partial reconfiguration allows to offer several algorithms for one and the same physical chip with limited resources.
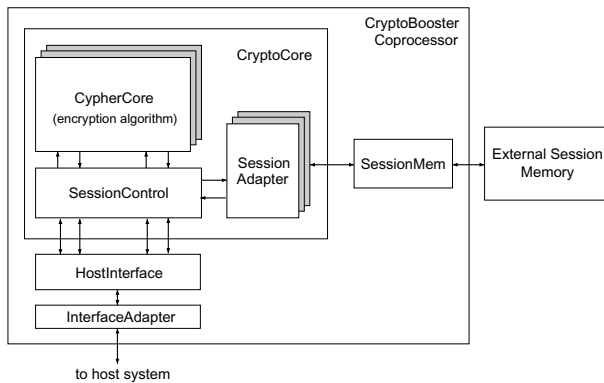
2.1   Modular Architecture

A block diagram of the CryptoBooster architecture is shown in Figure 2. The InterfaceAdapter module is a technology-dependent interface to the host system. Typically, this is a PCI or a VME interface, but one may also imagine a networking interface like Ethernet. The HostInterface is the software interface to the host system. It offers read/write registers and interruptions to configure and control the coprocessor. The SessionMem module allows to interface different types and configurations of physical

memories. A separate session memory has been chosen mainly to limit the communication between host and coprocessor and in order to change rapidly between different sessions.

The CryptoCore module itself is subdivided into three parts:

– *CypherCore*: encryption algorithm,
– *SessionAdapter*: session parameter management (specific to each CypherCore),
– *SessionControl*: central controller for session management.

The CypherCore and SessionAdapter modules are intelligent modules and can be queried by the SessionControl module. They respond with the implemented features available. It is therefore possible to exchange these modules without changing the control mechanism. All these modules communicate together and with the modules outside the CryptoCore using unidirectional point-to-point links called *CoreLink*. These links are designed to transmit control or data packets. This homogeneity at the interconnection level strongly enforces the modularity of the system.

Fig. 2. Block diagram of the CryptoBooster architecture. The architecture is highly modular and uses standardized interconnections.

2.2   Advantages of an FPGA-Based Implementation and Reconfiguration Features

An FPGA circuit is an array of logic cells placed in an infrastructure of interconnections [14]. Each cell is a universal function or a functionally complete logic device, which can be programmed to realize a certain function. Interconnections between the cells are also programmable. The versatility allowed by logic blocks and the flexibility of the interconnections provide high freedom of design during the utilization of FPGAs.

The CryptoBooster is implemented using the VHDL hardware description language. The design can thus be synthesized without major problems for FPGAs as well as for VLSI technology. A VLSI solution results in general in higher performance than an FPGA implementation but the latter has several important advantages: (1) Reconfigurability of the FPGA allows the developer to easily provide specific solutions to

the customer as it is often needed in cryptography; (2) A VLSI multi-algorithm co-
processor requires all corresponding CypherCores to be implemented in the chip which
demands a huge amount of transistors; (3) On the other hand, an FPGA only contains
one encryption algorithm at a given time. Other algorithms are available in the form
of a configuration bitstream. Thus the maximum area required corresponds to the area
used by the largest algorithm.

One can distinguish between full and partial FPGA reconfiguration. Full reconfigu-
ration is the common method currently used. The configuration is replaced by a new
one each time the algorithm is changed (Figure 3). Partial reconfiguration allows to
reconfigure parts of the FPGA, i.e., only the part where the algorithm is implemented
on the FPGA has to be reconfigured. This normally results in a much shorter interrupt
of service compared to full reconfiguration. The CryptoBooster is designed to take full
advantage of partial reconfiguration.
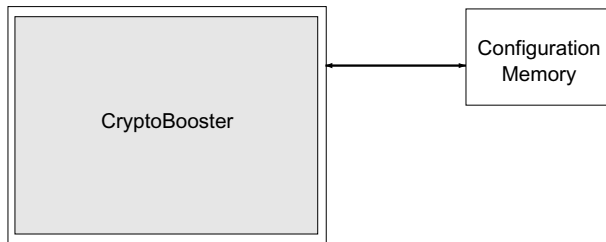
Fig. 3. Full reconfiguration requires to reprogram all the chip including common parts.
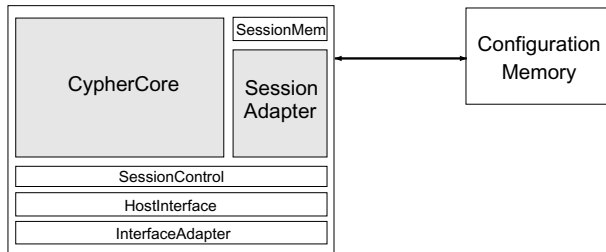
Fig. 4. Partial reconfiguration requires only to reprogram the parts specific to a par-
ticular algorithm.

3 The $IDEA^{TM}$ Block Encryption Algorithm

The International Data Encryption Algorithm ($IDEA^{TM}$) was developed by Xuejia Lai
and James Massey of the Swiss Federal Institute of Technology in Zurich. The original
version—called PES (Proposed Encryption Standard)—was first published in 1990 [8].

The following year, after Biham and Shamir's demonstrated differential cryptanalysis, the authors strengthened their cipher against the attack and called the new algorithm IPES (Improved Proposed Encryption Standard) [9]. IPES changed its name to IDEA in 1992 [7].

$IDEA^{TM}$ is one of a number of conventional encryption algorithms that have been proposed in recent years to replace DES. However, there has been no rush to adopt it as a replacement to DES, partly because it is patented and must be licensed for commercial applications, and partly because people are still waiting to see how well the algorithm fares during the upcoming years of cryptanalysis.

$IDEA^{TM}$ is a 64-bit block cipher that uses a 128-bit key to encrypt data (DES also uses 64-bit blocks but only a 56-bit key). The same algorithm is used for encryption and decryption. It consists of 8 computationally identical rounds followed by an output transformation. Round r uses six 16-bit subkeys $K_i^{(r)}$, 1 ¡ i ¡ 6, to transform a 64-bit input X = (X_1, X_2, X_3, X_4) into an output of four 16-bit blocks, which are input to the next round. The round 8 output enters the output transformation, employing four additional subkeys $K_i^{(9)}$, 1 ¡ i ¡ 4 to produce the final ciphertext Y = (Y_1, Y_2, Y_3, Y_4). All 52 16-bit subkeys are derived from the 128-bit master key K. The key is long enough to withstand from exhaustive key searches well into the future.

$IDEA^{TM}$ is easy to implement in both software and hardware, even in embedded systems. A typical software implementation of $IDEA^{TM}$ in C (using the Ascom Systec source code) performs data encryption at 16 Mbit/s on a PentiumPro 180 MHz machine. It is clear that a hardware implementation of the algorithm may essentially speed up the throughput. To the best of our knowledge, the first VLSI implementation was developed at the Swiss Federal Institute of Technology in Zurich by Bonnenberg *et al.* [1] and reached a throughput of 44 Mbit/s at 25 MHz.

Current hardware implementations have stressed the importance of the combinatorial delay and area consumption of the multiplication modulo $(2^{16} + 1)$ units which are crucial to the entire system. These units are the limiting factor to obtain high data throughput. Various methods of implementing such a multiplication are investigated in [16, 5, 15, 10, 3]. The VINCI implementation uses a modified Booth recording multiplication and fast carry select additions for the final modulo correction [17]. In general, for larger words, ROM-based solutions using lookup-tables require large ROMs. In a recent paper, Zimmermann [16] presents an efficient VLSI implementation of the modulo $(2^{16} + 1)$ multiplication. Several different hardware implementations of the $IDEA^{TM}$ algorithm are given below.

3.1   The VINCI VLSI–Implementation

A new VLSI implementation—called VINCI [4,17]—was developed in 1993 at ETH Zurich. The chip consists of 250,000 transistors on a total area of $107.8mm^2$ and attained 177.8 Mbit/s @ 25 MHz. The data path was optimized using an eight-stage pipeline and full-custom modulo $(2^{16} + 1)$ multipliers. The VINCI chip was the first chip that could be used for on-line encryption in high-speed networking like ATM or FDDI. Figure 5 shows the pipeline of the VINCI data path for one round. The multiplication modulo $(2^{16} + 1)$ operation is distributed in two pipeline stages to reduce the critical path.

Note that the output round is identical to the first section (3 stages) of a regular round as shown in Figure 5.
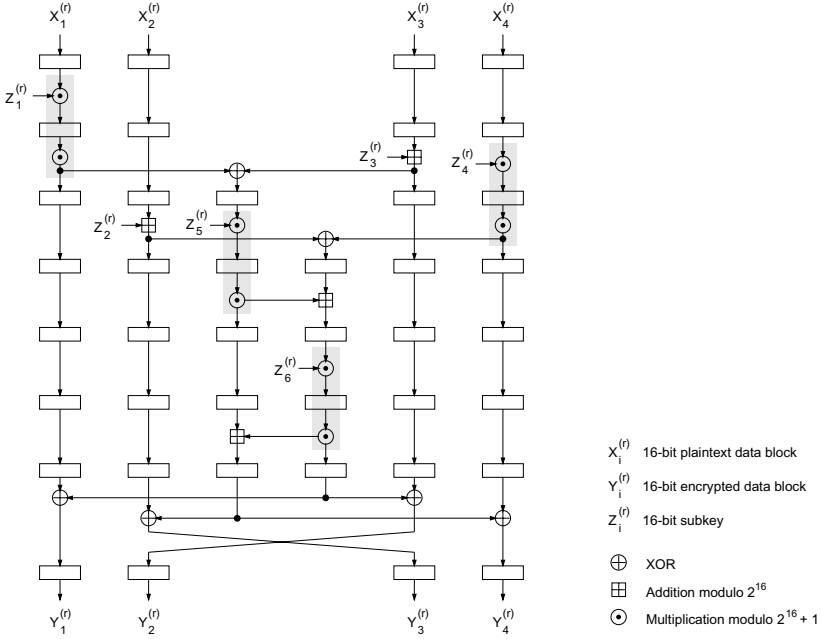
Fig. 5. Pipeline of the VINCI [17] data path for one round.

3.2   Ascom IDEACrypt Coprocessor

Ascom Systec Ltd., holder of the $IDEA^{TM}$ patents, offers a high-speed implementation of the $IDEA^{TM}$ algorithm as an embedded ASIC core. The IDEACrypt kernel implements data encryption and decryption in all common operating modes for block ciphers (ECB, CBC, CFB, OFB) [6]. IDEACrypt provides flexible key management for both master-session key and asymmetric key and stores the keys in a RAM which may be either on-chip or off-chip. Therefore, with a RAM, a bus interface, and a global controller, a complete $IDEA^{TM}$ cipher may be implemented. The entire IDEACrypt coprocessor is implemented in synthesizable VHDL code. Ascom Systec lists a complexity of approximatively 35k gates.

With 0.25 micron technology using a 3-stage pipeline, the chip provides a throughput of 300 Mbit/sec (@ 40 MHz) in ECB mode and a throughput of 100 Mbit/sec in the other modes. At 100 MHz, the throughput goes up to 720 Mbit/sec in ECB mode.

## 3.3   Existing FPGA-Implementations

Caspi and Weaver [2] proposed IDEA as a benchmark for reconfigurable computing. Their implementation on Xilinx XC4005 achieves a throughput of 0.477 Mbit/s. A high space-conserving design was used because of the limited resources of this FPGA.

Mencer et al. [13] compared different implementations of the $IDEA^{TM}$ algorithm. They compared a Digital Signal Processor (DSP) from Texas Instruments to Xilinx XC4000 series FPGAs and pointed out the benefits and limitations of FPGA and DSP technologies for $IDEA^{TM}$. The FPGA implementation has a throughput rate of 528

Mbit/s @ 33 MHz using a fully pipelined version of $IDEA^{TM}$ distributed on four XC4000XL FPGAs. Taking into account the powerful programming capabilities of the FPGA technology and the performance of the new families (e.g., Xilinx Virtex), this kind of circuit is becoming an excellent option to implement $IDEA^{TM}$.

4   The First CypherCore: IDEACore Encryption Module

The first CypherCore module—called IDEACore—implements the $IDEA^{TM}$ algorithm. It is composed of a scalable pipeline and an associated block chaining module (Figure 6).
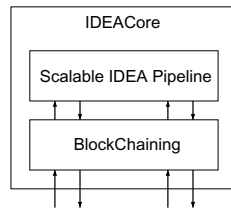
Fig. 6. The IDEACore module is composed of a scalable pipeline and a block-chaining module.

4.1   A Scalable $IDEA^{TM}$ Pipeline

We adopted a highly scalable solution for our $IDEA^{TM}$ pipeline: the length of the pipeline can be chosen at compilation time. The regular round is inspired by the VINCI datapath (see Figure 5). Figure 7 shows the regular round consisting of seven pipeline stages. The first three stages of a regular round simply form the output round (Fig. 8).

As Figure 9 shows, the minimal pipeline length is one regular round followed by one output round. Data has to be fed eight times through the regular round before passing through the output round. The longest pipeline (full-length pipeline) consists of eight rounds and one output round. In each configuration, the data needs 59 clock cycles to pass through the pipeline. A longer pipeline has a smaller latency and thus a higher throughput.

We use a fully self-controlled pipeline with a control pipeline associated in parallel to the data pipeline. The control pipeline addresses the key memories attached to each stage that needs an encryption key. Every 64-bit data block has an associated counter that indicates the current round. Data is automatically feed to the output stage if it was sent the correct number of times through the regular rounds or it is fed back through the block of regular rounds. Pipeline bubbles (data marked by a non-valid bit) are automatically inserted into the pipeline if the module preceding the pipeline (block chaining) is not able to deliver new data packets. This mechanism allows us to avoid pipeline stalls.

Multipliers and Modulo $(2^{16} + 1)$ We currently use simple bit-parallel multipliers optimized for FPGAs and the low-high algorithm [7] for the modulo $(2^{16} + 1)$ calculation. As stated in section 3, the combinatorial delay and area consumption of the multiplication modulo $(2^{16} + 1)$ units are crucial to the entire design and are limit the data path.

Bit-parallel multipliers are perhaps not the best choice, but we were surprised by the performance they achieved and the area they used in our FPGA-based implementation. In the near future, we intend to optimize the multiplication modulo $(2^{16} + 1)$ units to achieve yet higher throughput.
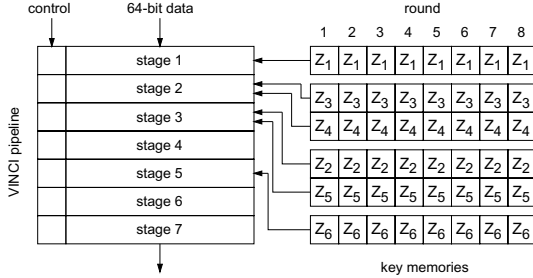
**Fig. 7.** One round with associated encryption key memories of the CryptoBooster pipeline.
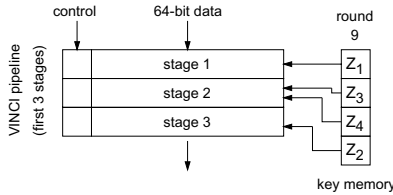
Fig. 8. Output round with associated encryption key memories of the CryptoBooster pipeline.

4.2   Block-Chaining

The block-chaining module implements the commonly used block-chaining algorithms like ECB (Electronic Codebook Mode), CBC (Cipher Block Chaining Mode), CFB (Cipher Feedback Mode), and OFB (Output Feedback Mode). To prevent the pipeline from stalling, the block-chaining module always disposes of enough initial vectors (the number depends on the number of regular rounds used in the pipeline).

As with all other modules in the CryptoBooster architecture, the block-chaining module is connected to the other modules by CoreLink unidirectional point-to-point interconnections.
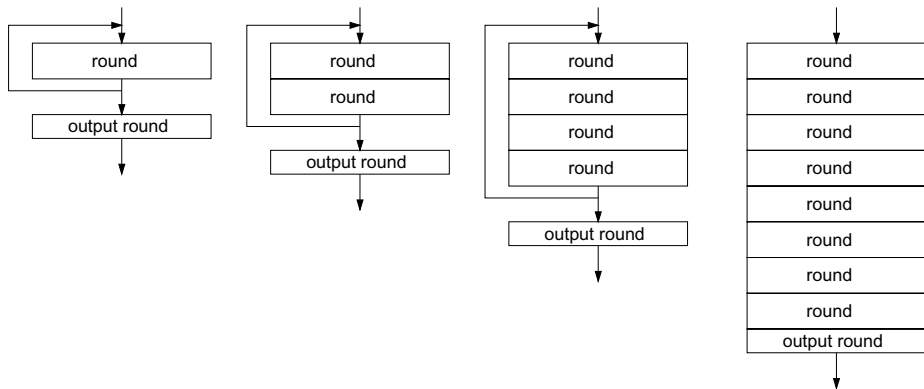
Fig. 9. The four different pipeline configurations: 1+1, 2+1, 4+1, and 8+1 rounds. Data has to be feed 8, 4, 2, and 1 time through the regular rounds.

4.3   Performance of the IDEACore CryptoCore

The CryptoBooster is designed to achieve maximum throughput for a given area in the FPGA. Our current implementation allows pipeline lengths of 1, 2, 4, or 8 regular rounds, followed by one output round. A full-length pipeline consists of 59 (8 regular rounds + 1 output round) stages with a latency of 1 clock cycle when using bit-parallel multipliers (Figure 9).

The peak performance of our current implementation is estimated at 200 Mbit/s for a 1-round pipeline (1 regular round + 1 output round) and it easily fits into a state-of-the-art FPGA. Performance for a full-length pipeline (8 regular rounds + 1 output round) is estimated at more than 1500 Mbit/s. However, the area needed in terms of reconfigurable logic blocks in the FPGA is quite important.

Session initialization and key calculation slightly decrease the overall performance over a complete session. Depending on the block-chaining mode used, performance may, however, significantly decrease.

5   Conclusions

The CryptoBooster is a modular and reconfigurable cryptographic coprocessor taking full advantage of current high-performance reconfigurable circuits (FPGAs). Reconfigurable circuits can be reconfigured within a few milliseconds and they provide speed rates close to ASIC designs. Our main goal is to have maximum data throughput so as to provide a design able to cope with ever-increasing network speed. This justifies hardware implementation in place of a software solution. Physical security may, however, also be an argument for hardware implementation.

As our results show, the throughput of the CryptoBooster allows the coprocessor to be used in today's high-speed networks like ATM, Sonet, and GigaEthernet; moreover, it is competitive with full-custom circuits or DSP implementations. $IDEA^{TM}$ was chosen as a first CryptoCore module. More modules with different algorithms (e.g., DES) are planned.

Acknowledgments

References

1. H Bonnenberg, A. Curiger, N. Felber, H. Kaeslin, and X. Lai. VLSI implementation of a new block cipher. In *Proceeding of the International Conference on Computer Design: VLSI in Computer and Processors*, pages 510–513, Washington, 1991. IEEE, IEEE Computer Society Press.
2. E. Caspi and N. Weaver. IDEA as a benchmark for reconfigurable computing. Technical report, BRASS Research Group, University of Berkeley, December 1996. Report available form http://www.cs.berkeley.edu/projects/brass/projects.html.
3. A. Curiger, H. Bonnenberg, and H. Kaeslin. Regular VLSI-architectures for multiplication modulo $(2^n + 1)$. *IEEE Journal of Solid-State Circuits*, 26(7):990–994, July 1991.
4. A. Curiger, H. Bonnenberg, R. Zimmermann, N. Felber, H. Kaeslin, and W. Fichtner. VINCI: VLSI implementation of the new block cipher IDEA. In *Proceedings of the IEEE CICC'93*, pages 15.5.1–15.5.4, San Diego, CA, May 1993. IEEE.
5. A. Hiasat. New memoryless modulo $(2^n + 1)$ residu multiplier. *Electronic Letters*, 41(3):314–315, January 1992.
6. ISO. Information technology-security techniques-modes of operation for an n-bit block cipher. International standard ISO/IEC 10116:1997(E), ISO/IEC, 15 April 1997.
7. X. Lai. *On the Design and Security of Block Ciphers*. Number 1 in ETH Series in Information Processing. Hartung-Gorre Verlag Konstanz, 1992.
8. X. Lai and J. Massey. A proposal for a new block encryption standard. In *Advances in Cryptology–EUROCRYPT'90*, pages 389–404, Berlin, 1990. Springer-Verlag.
9. X. Lai, J. Massey, and S. Murphy. Markov ciphers and differential cryptanalysis. In *Advances in Cryptology–EUROCRYPT '91*, pages 8–13, Berlin, 1991. Springer-Verlag.
10. Y. Ma. A simplified architecture for modulo $(2^n + 1)$ multiplication. *IEEE Transactions on Computers*, 47(3):333–337, March 1998.
11. J. L. Massey and X. Lai. Device for convertig a digital block and the use thereof, 28 Nov 1991. International Patent PCT/CH91/00117.
12. J. L. Massey and X. Lai. Device for the conversion of a digital block and use of same, 25 May 1993. U.S. Patent #5,214,703.
13. O. Mencer, M. Morf, and M. J. Flynn. Hardware software tri-design of encryption for mobile communication units. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, Seattle, Washington, USA, May 1998.
14. S. Trimberger. *Field-Programmable Gate Array Technology*. Kluwer Academic Publishers, 1994.
15. Z. Wang, G. A. Jullien, and W. C. Miller. An efficient tree architecture for modulo $(2^n + 1)$ multiplication. *Journal of VLSI Signal Processing Systems*, 14(3):241–248, December 1996.

16. R. Zimmermann. Efficient VLSI implementation of modulo $(2^n + 1)$ addition and multiplication. In *Proceedings IEEE Symposium on Computer Arithmetic*, Adeleide, Australia, April 1999. IEEE.
17. R. Zimmermann, A. Curiger, H. Bonnenberg, H. Kaeslin, N. Felber, and W. Fichtner. A 177 Mbit/s VLSI implementation of the international data encryption algorithm. *IEEE Journal of Solid-State Circuits*, 29(3):303–307, March 1994.