Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für Integrierte Systeme
Integrated Systems Laboratory

A 177 Mbit/s VLSI Implementation of the International Data Encryption Algorithm

R. Zimmermann, A. Curiger, H. Bonnenberg, H. Kaeslin, N. Felber, and W. Fichtner

December 1993

Abstract

A VLSI implementation of the International Data Encryption Algorithm is presented. Security considerations led to novel system concepts in chip design including protection of sensitive information and on-line failure detection capabilities. BIST was instrumental for reconciling contradicting requirements of VLSI testability and cryptographic security. The VLSI chip implements data encryption and decryption in a single hardware unit. All important standardized modes of operation of block ciphers, such as ECB, CBC, CFB, OFB and MAC, are supported. In addition, new modes are proposed and implemented to fully exploit the algorithm's inherent parallelism. With a system clock frequency of 25 MHz the device permits a data conversion rate of more than 177 Mbit/s. Therefore, the chip can be applied to on-line encryption in high-speed networking protocols like ATM or FDDI.

This paper was published in the *IEEE Journal of Solic-State Circuits* Special Issue on the 1993 Custom Integrated Circuits Conference, vol. 29, no. 3, March 1994.

1 Introduction

1.1 Cryptography

Encryption is the conversion of information, the *plaintext* or *message*, into code, the *ciphertext*, which is intelligible only for an authorized receiver. Data encryption is applied to ensure secrecy of a message transferred. Historically, cryptographic techniques have been developed for diplomatic or military applications; today they can be found everywhere in private and public sectors where confidential information has to be processed, transferred, and stored. An important characteristic of modern cryptography is the use of publicly known, i. e. published, algorithms [1]. The secrecy is therefore not kept in the algorithm itself, but only in a small additional piece of information shared between sender and receiver, the *key*.

1.2 Secret-Key Block Encryption

Secret-key block encryption divides the plaintext into a series of blocks of equal length, typically 64 or 128 bits. These blocks are then processed sequentially using a key known to the sender and receiver exclusively. A well-known and widespread algorithm implementing secret-key block encryption is the Data Encryption Standard (DES) which has been adopted for inter-government use by the American National Bureau of Standards [2] and for the commercial sector by the American National Standards Institute (ANSI, 1982). However, serious concerns arise about long-term security because of DES's relatively short key word length of a mere 56 bit, and, more recently, from the cryptanalysis attack of Biham and Shamir [3].

1.3 The Data Encryption Algorithm IDEA

In this context, a new block encryption algorithm called IDEA™ (International Data Encryption Algorithm), that overcomes the problems mentioned above, has been developed and published by Lai and Massey [4, 5]. In this cipher, the plaintext and the ciphertext are processed in blocks of 64 bits, while the key is 128 bits long. The cipher relies on combining operations from three algebraic groups. The three group operations on vectors of length 16 are:

1. Bitwise addition modulo 2 (XOR) of two 16-bit subblocks,
2. Addition of integers modulo 2^{16}, and
3. Multiplication modulo the Fermat prime $p = 2^{16} + 1$, where the value '0' is never used, and 2^{16} is represented by the all-zero vector 0..00.

Figure 1 gives an overview of the encryption process. The cipher is constructed in such a way that the deciphering process is the same as the enciphering process, the only difference being that different key subblocks are used. This property is called *similarity* of encryption and decryption. As becomes clear from Fig. 1, eight identical rounds, followed by an output transformation, are cascaded to form the complete cipher.

The 64-bit plaintext block is partitioned into four 16-bit subblocks, the i-th of which is denoted as X_i in Fig. 1. The four plaintext subblocks are then transformed into four 16-bit ciphertext subblocks, Y_1, Y_2, Y_3, and Y_4 using fifty-two key subblocks of 16 bits each, where the six key

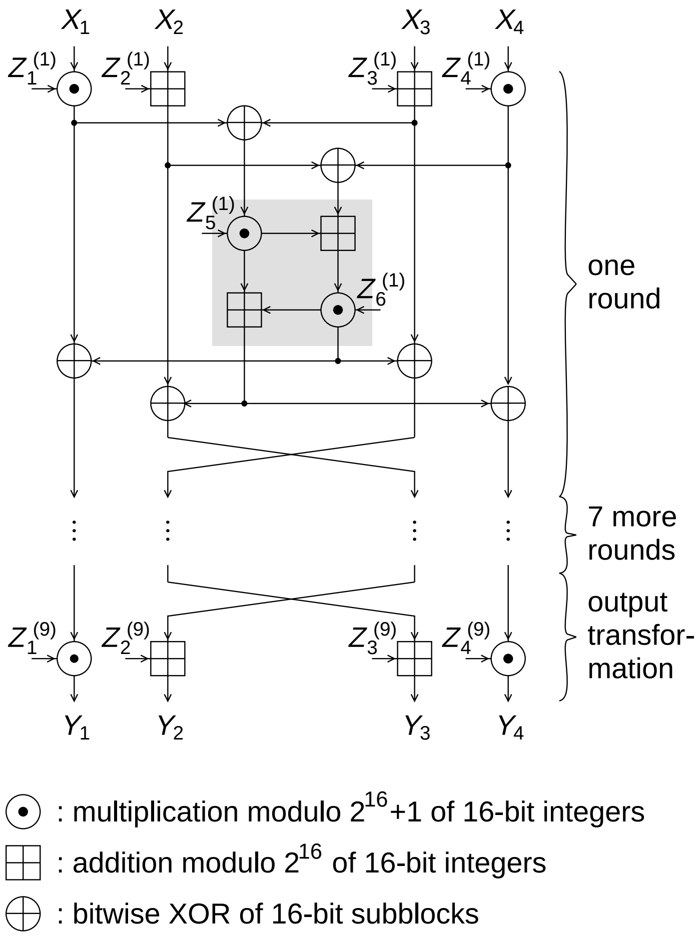

Figure 1: Data dependency graph for encryption/decryption.

subblocks used in the r-th round ($r = 1, \ldots, 8$) are denoted as $Z_1^{(r)}, \ldots, Z_6^{(r)}$, and where the four key subblocks used in the output transformation are denoted as $Z_1^{(9)}, \ldots, Z_4^{(9)}$. The fifty-two key subblocks used in the encryption or decryption process are generated from the 128 user-selected key bits according to a key schedule [5].

The transformation at the heart of a round, consisting of two multiplications mod $2^{16} + 1$ and two additions mod 2^{16} (structure on grey background in Fig. 1), provides the desired statistical properties of the cipher. This transformation is called the *multiplication-addition* (MA) *structure*.

2 VLSI Implementation

A software implementation of IDEA on a Sun SPARCstation 2 performs data encryption at a processing speed of 400 kBit/s. A first prototype VLSI circuit, developed and successfully tested two years ago [6], essentially served to speed up cryptographical tests and statistical analyses of the algorithm. Its throughput of 44 Mbit/s exceeded that of the software version by a factor of more than 100.

Specifications for the present implementation aimed at a data rate in excess of 155 Mbit/s, thus demanding a more sophisticated design as was realized in the new VLSI implementation called VINCI described in this paper.

The stringent speed requirement essentially posed three problems, namely, to design efficient basic building blocks, to find a high-throughput datapath architecture, and to come up with a chip interface capable of handling the resulting off-chip data traffic.

2.1 Basic Building Blocks

Whereas the implementation of the XOR and addition operations was obvious, the structure of a modulo $(2^{16}+1)$ multiplier turned out to be challenging. The realization of the required on-chip buffer memories was the subject of careful investigation as well.

2.1.1 Multiplication Modulo $2^{16}+1$

Combinational delay and area consumption of the multiplication modulo $(2^{16}+1)$ unit are crucial to the entire chip architecture and to the maximum attainable data throughput rate. Various methods of implementing such a dedicated multiplication unit were investigated and compared [7]. Table-lookup based solutions and an ordinary multiplication with subsequent modulo correction both resulted in overly high computation time and area. Therefore a custom-layout realization was chosen using a computation scheme with stepwise modulo reduction. A modified Booth recoding multiplication and fast carry-select additions for the final modulo correction form the two stages of the multiplier pipeline structure. Additional test circuitry has been added to increase fault coverage. The small size of the resulting layout block allowed the placement of four multiplication units instead of only two, which together with the massively reduced computational delay increased the throughput of the encryption/decryption datapath by a factor of four compared to the scheme implemented in the prototype VLSI circuit [6]. Technical data of the multiplication unit are summarized in Table 1.

process technology:	double-metal CMOS 1.2 μm
area:	1.51 mm^2
complexity:	12 028 transistors
latency:	2 clock cycles
throughput:	31 M Mult/s

Table 1: Technical data of mod-$(2^{16}+1)$ multiplication unit.

2.1.2 On-chip Buffer Memory

On-chip storage elements can be classified as delay buffers, input and output (I/O) buffers, key memory, and pipeline registers and their compensation registers on parallel feed-forward branches of the data dependency graph.

To implement the different modes of operation two eight-cycle 64-bit *delay buffers* were required, which were implemented as (8×64)-bit shift registers. There speed was not critical but, however, area efficiency was very important. Different implementation variants using automatically generated layout were compared, but none of them met the stringent area requirements. A very area-efficient storage block, for example, was the (64×8) - SRAM, which in turn was not suited for our needs because of its small word size. Consequently, a custom-layout realization was chosen, which was also encouraged by the high regularity of the shift register structure. Table 2 compares area and transistor counts of various alternative (8×64)-bit shift register implementations.

	area (mm^2)	rel. area	# trans.	# trans./mm^2
full-custom	0.90	1.0	8 960	9 956
standard-cell	3.84	4.3	18 222	4 745
macrocell register file [1]	1.94	2.2	8 318	4 288
macrocell shift register	2.90	3.2	18 880	6 510
(16×64) - SRAM [1 2]	2.36	2.6	not avail.	not avail.
(64×8) - SRAM [1]	0.52	0.6	not avail.	not avail.

[1] needs additional control logic (not included)

[2] smallest available SRAM with appropriate word length

Table 2: Comparison between different shift register implementations.

Input and output buffers are implemented in ping-pong mode, i. e., both buffers are divided into two parts. During encryption or decryption, each part alternatingly receives and releases eight 64-bit data blocks. The (8×64)-bit shift registers described in the previous paragraph were reused for that purpose.

Finally, the *key memory* is implemented as a (256×16)-bit RAM to store all subkeys required for encryption and decryption. Two different keys can be stored in parallel, which allows the usage of an additional session key. Recall that a total of 104 subkeys forms the complete set of encryption and decryption subkeys.

2.2 Datapath Architecture

2.2.1 Encryption and Decryption

Encryption and decryption are the speed-critical data processing operations that must be carried out in real-time, as opposed to subkey computation, for instance. The main objective in designing the VINCI datapath architecture has therefore been to meet the 155 Mbit/s throughput requirement at reasonable costs. The fact that the cascaded rounds in Fig. 1 are identical, except for the key bits, suggested recycling data through a single datapath. Note that the output transform in this context is computationally identical to the first section of a regular round.

This single datapath was then chosen to be isomorphic to one round, which is to say that it includes a hardware unit for each computation and that all units operate in parallel. In order to further increase clock rate and throughput, an eight-stage pipelining scheme was incorporated. The final datapath structure is shown in Fig. 2.

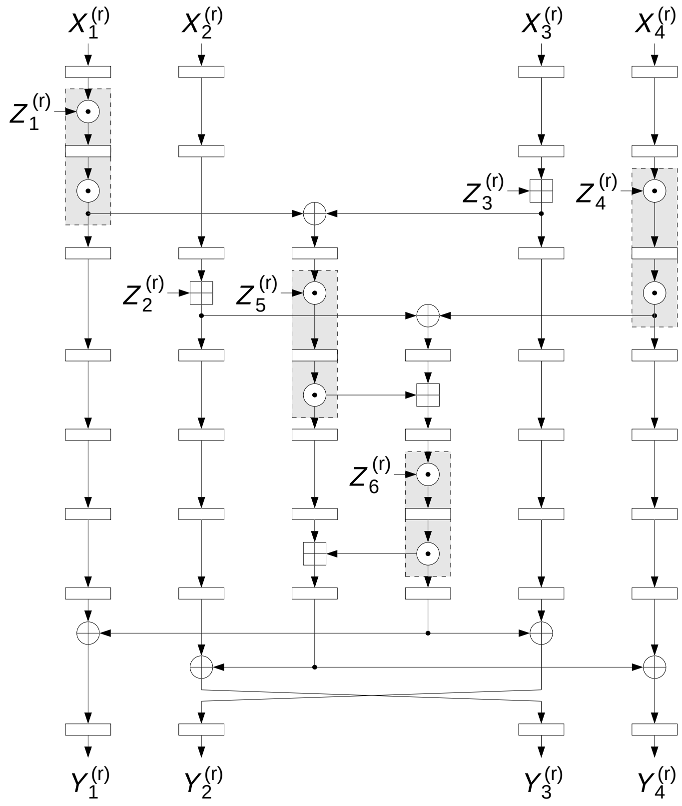

Figure 2: Basic pipeline architecture of the VINCI datapath.

2.2.2 Exploiting Pipelining: New Modes of Operation of a Block Cipher

The chip implements the five standardized modes of operation of block ciphers [9] which are ECB (Electronic Code Book), CBC (Cipher Block Chaining), CFB (Cipher Feedback), OFB (Output Feedback), and MAC (Message Authentication Code).

All operation modes which include feedback, i. e., CBC, CFB, OFB, and MAC, require first order recursive computation. However, this type of feedback scheme cannot accommodate any latency caused by pipelining in its computational units without changing the input-to-output relationship. The highly nonlinear nature of the ciphering algorithm also precludes any unfolding of the recursive loop.

In our eight-stage pipeline this dilemma can be resolved in two ways. First, the original dataflow can be preserved by refraining from pipelining within the recursive computations, thus reducing throughput to one eighth of its potential. Second, accepting the decomposition of the data stream into eight separately encrypted chains makes it possible to run the pipeline at full speed. Here the l-th block is combined with the $(l + 8)$-th (see Fig. 3), resulting in eight different and independent data streams as depicted in Fig. 4. For CBC, this new mode is called *CBC 8*, where CBC 1 denotes the standard scheme. Both solutions are implemented in VINCI for all operation modes that imply recursion.

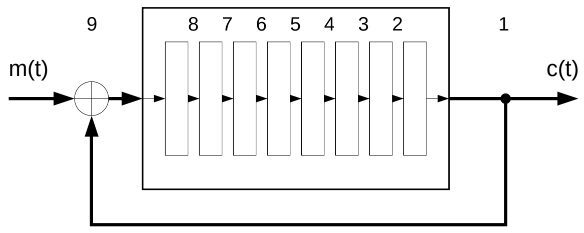

Figure 3: Encryption in CBC mode.

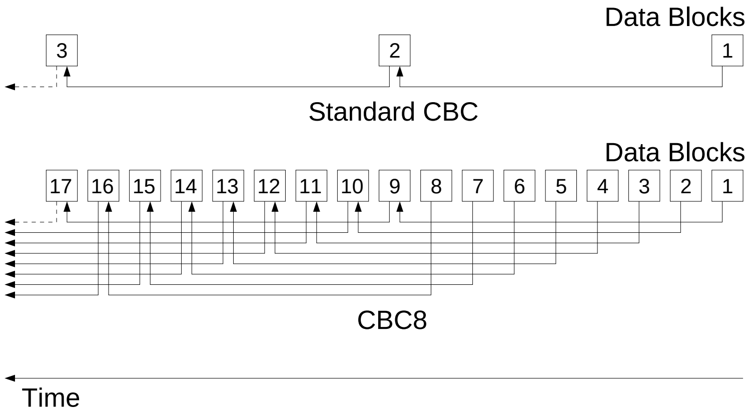

Figure 4: Data dependencies in CBC 1 and CBC 8 mode.

2.2.3 Computing Subkeys

Key management is performed entirely on-chip. Only the 128-bit wide master key has to be loaded whereupon all subkeys are generated internally. This generation process includes shifting of the master key and computing additive and multiplicative inverse elements. The method for computing a multiplicative inverse modulo $2^{16} + 1$ is based on Fermat's theorem and implemented using modular exponentiation. Modular exponentiation is carried out as a sequence of modulo-squaring and modulo-multiplication operations,

$$\begin{aligned} X^{-1} \bmod p &= X^{p-2} \bmod p \\ &= X((\ldots((X(X^2 \bmod p) \bmod p)^2 \bmod p)\ldots)^2 \bmod p) \bmod p \end{aligned} \tag{1}$$

for which the on-chip modulo-multipliers are used.

2.3 Interfacing the VLSI Circuit

The overall architecture of VINCI is depicted in Fig. 5. With consideration for high-speed applications, the data interfaces are designed as unidirectional input and output ports. The two 16-bit wide ports of VINCI are designed such that data can be transferred continuously and independently from the internal chip clock.

All parameters required to set up VINCI for a specific conversion can be loaded and the actual status watched via an 8-bit wide bidirectional control port. Seven control registers are implemented for that purpose. The master key is also loaded via this port.

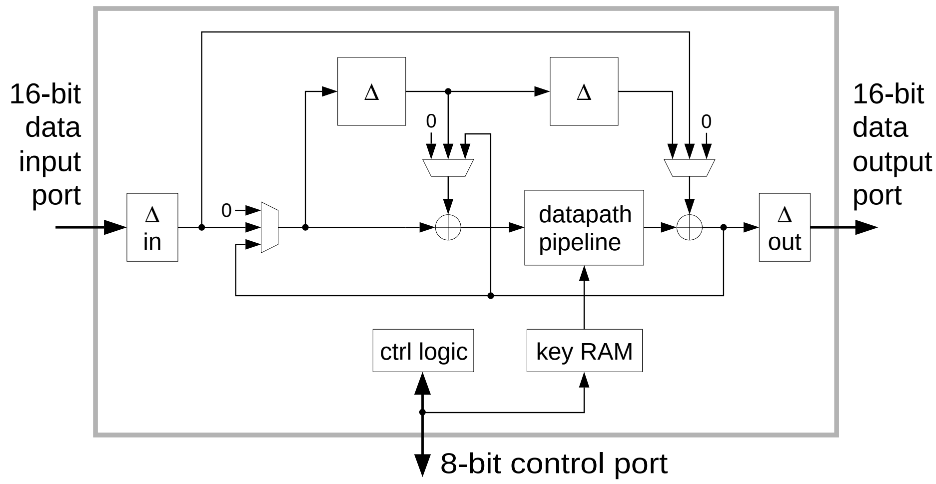

Figure 5: Overall architecture of VINCI.

3 Verification and Testing

3.1 Off-line Built-in Self-test

Security standards for cryptographic equipment (see [10]) require physical protection of secret keys and unencrypted or partially encrypted data. On the other hand, a sufficient testability of complex modern electronic devices is a must that is usually achieved by easing physical access to internal signals (observability and controllability). Such contradictory requirements can be reconciled by introducing a *built-in self-test* (BIST). BIST includes stimulus generation and response analysis on-chip and thus makes accessibility to potential sensitive internal nodes unnecessary. It has been shown in [11] that ciphers like IDEA which are based on different algebraic group operations are well testable using pseudo-random patterns. Therefore a VLSI implementation of such a cipher is also well-suited to pseudo-random BIST.

The implemented off-line self-test scheme starts with a test of the subkey RAM using checkerboard and retention tests. A pseudo-random pattern generator based on a linear feedback shift register provides data for the subsequent subkey generation process and a number of encryptions and decryptions in different modes of operation. A 21-bit signature is generated out of the results and compared to a precomputed and hard-wired value. This off-line BIST scheme covers nearly the complete on-chip circuitry. It achieves a gate-level fault coverage of 93.6 % using the stuck-at fault model, and it results in an area overhead of only 2.0 %. Note that a comparable device, [12], achieves a fault coverage of 94.7 % with an area overhead of 9.4 %.

3.2 Concurrent Self-test

A security standard [10] requires the immediate detection of failure during operation to prevent the release of faulty data blocks. Hence, an on-line BIST has been added, by which 93 % of the pipeline circuitry is concurrently checked during the encryption/decryption phase. An on-line controller continuously monitors all self-test results and immediately triggers an alarm in case of non-conformance. The underlying self-test scheme is based on the following concepts:

- *Hardware redundancy:* A fifth multiplier observes on-line the four pipeline multipliers alternatingly, and an additional adder checks for correct adder operation.
- *Involution property of the XOR operation:* Data blocks which are XORed and delayed on different paths can be restored by another XOR operation and then be compared to the original value, thus checking most parts of the pipeline's XOR and delay circuitry.
- *Computation of an invariant:* For all multiplicative and additive subkeys their inverse is present in the RAM. Thus pairwise multiplication or addition, respectively, of a subkey and its inverse yields the identity elements '1' or '0', which validate correctness of the key processing and storing circuitry. The redundant multiplier and adder are also used for this purpose.
- *Concurrent controller check:* Concurrent self-test of the pipeline controller can efficiently be performed by merging controller and signature analyzer function into one single circuit.

The entire self-testing scheme is described in detail in [13].

The area requirements for off-line and concurrent self-test are illustrated in Table 3. Note that nearly half of the area needed for off-line self-test is occupied by RAM test circuitry.

3.3 Fault Localization

A BIST scheme with only one signature allows detection but not localization of faults. Therefore VINCI has been provided with additional monitoring mechanisms that allow identification of failures without releasing sensitive data.

Unit	Subunit	Area (mm^2)	in % of chip
chip		107.80	100
off-line self-test	RAM test	0.50	
	retention	0.43	
	scheduler	1.24	
total		2.17	2.01
concurrent self-test	circuitry	7.50	
	controller	0.35	
total		7.85	7.28

Table 3: Area consumption of self-test hardware.

3.4 Boundary Scan

A boundary-scan scheme was implemented for system test requirements. However, sensitive I/O signals are excluded from boundary scan for security reasons.

4 Technical Data

The technical data of the VINCI chip are shown in Table 4. The chip was fabricated in CMN12, a 1.2 μm double-metal n-well CMOS process by VLSI Technology, Inc. Fig. 6 illustrates the floorplan and the corresponding photomicrograph of the chip.

process technology:	CMN12
complexity:	251 000 transistors
test structures:	boundary scan path, built-in off-line self-test, built-in concurrent self-test
chip area:	107.8 mm^2
package:	PGA 132
system clock frequency:	25 MHz
power consumption:	1.25 W (@ 25 MHz, 5.0 V, 25° C)
max. encryption throughput:	177.8 MBit/s (@ 25 MHz)

Table 4: Technical data of VINCI.

Tools by COMPASS Design Automation were used throughout for design, simulation, verification, and logic synthesis. Total design effort for the VLSI circuit was approximately 18 man-month.

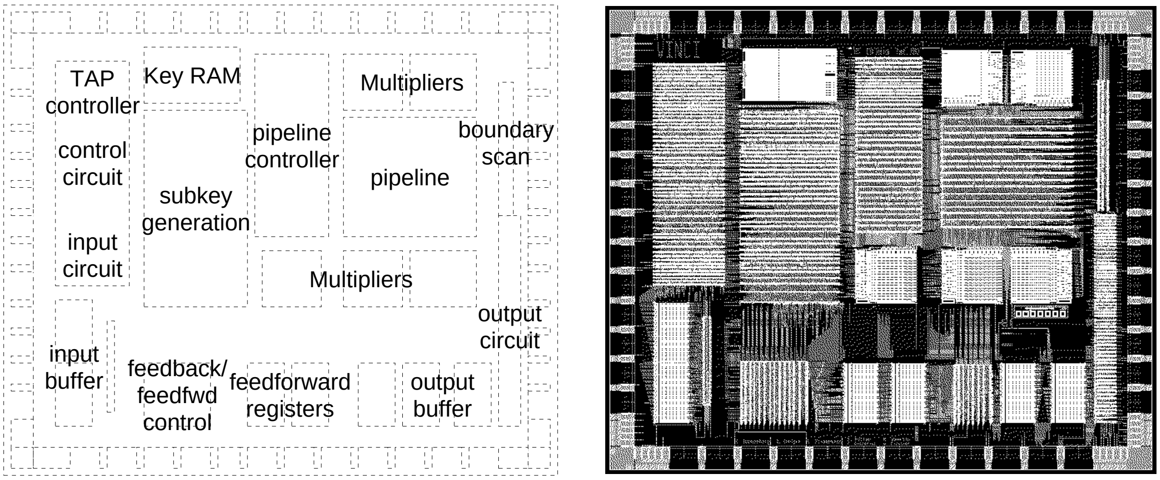

Figure 6: Floorplan and photomicrograph of VINCI.

5 Conclusions

A high-speed VLSI block encryption circuit based on the new cipher IDEA has been presented. VINCI, IDEA's first silicon realization, integrates high-speed encryption and decryption, comprehensive key management functions, and all standard cipher modes of operation in their ordinary and high-speed adapted versions.

High data throughput of 177 Mbit/s has been obtained from pipelining and from incorporating four full-custom modulo $(2^{16}+1)$ multipliers. Two unidirectional high-speed 16-bit data ports allow for continuous operation of the datapath. All controller, pipeline stage registers, and the remaining arithmetic units were realized with standard-cells.

The resulting VLSI circuit achieves data rates significantly higher than the fastest silicon implementation of DES known to the authors. The only known faster single-chip implementation of a block cipher uses GaAs-technology [14]. VINCI is therefore the first silicon block encryption device that can be applied to on-line encryption in high-speed networking protocols like ATM (Asynchronous Transfer Mode) or FDDI (Fiber Distributed Data Interface).

References

[1] J. L. Massey. An introduction to contemporary cryptology. *Proceedings of the IEEE*, 76(5):533–549, May 1988.

[2] National Bureau of Standards NBS. Data Encryption Standard. FIPS PUB 46, January 1977.

[3] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[4] X. Lai and J. L. Massey. A proposal for a new block encryption standard. In *Advances in Cryptology – EUROCRYPT '90*, pages 389–404, Berlin, 1990. Springer Verlag.

[5] X. Lai, J. L. Massey, and S. Murphy. Markov ciphers and differential cryptanalysis. In *Advances in Cryptology – EUROCRYPT '91*, pages 8–13, Berlin, 1991. Springer Verlag.

[6] H. Bonnenberg, A. Curiger, N. Felber, H. Kaeslin, and X. Lai. VLSI implementation of a new block cipher. In *ICCD'91, 1991 IEEE International Conference on Computer Design: VLSI in Computer and Processors*, pages 510–513, Washington, 1991. IEEE Computer Society Press.

[7] 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.

[8] A. D. Booth. A signed binary multiplication technique. *IEEE Quarterly Journal of Mechanics and Applied Mathematics*, 4:236–240, 1951.

[9] International Standards Organization. Information technology – modes of operation for an n-bit block cipher algorithm. ISO/IEC 10116, 1991.

[10] Federal Information Processing Standards Publication. Telecommunications: General security requirements for equipment using the data encryption standard. FIPS PUB 140, 1982.

[11] H. Bonnenberg, A. Curiger, R. Zimmermann, N. Felber, H. Kaeslin, and W. Fichtner. A new and secure selftest scheme for block cipher implementations. In *Proc. of the 3rd European Test Conference (ETC) 1993*, pages 501–502, 1993.

[12] K. Hafner, H. C. Ritter, T. M. Schwair, S. Wallstab, M. Deppermann, J. Gessner, S. Koesters, W.-D. Moeller, and G. Sandweg. Design and test of an integrated cryptochip. *IEEE Design and Test*, pages 6–17, December 1991.

[13] H. Bonnenberg. *Secure Testing of VLSI Cryptographic Equipment*. PhD thesis, ETH Zürich, 1993.

[14] H. Eberle. A high-speed DES implementation for network applications. *DEC Systems Research Center*, 90, 1992.