Security and Performance Optimization of a New DES Data Encryption Chip

INGRID VERBAUWHEDE, FRANK HOORNAERT, JOOS VANDEWALLE,
SENIOR MEMBER, IEEE, AND HUGO J. DE MAN, FELLOW, IEEE

Abstract—**Cryptographical applications demand both high speed and high security. This paper presents the implementation of a new high-performance Data Encryption Standard (DES) data encryption chip. It is the result of close cooperation between cryptographers and chip designers. At the system design level, cryptographical optimizations and equivalence transformations lead to a very efficient floor plan with minimal routing, which otherwise would present a serious problem for data scrambling algorithms. These optimizations, which do not compromise the DES algorithm nor the security, are combined with a highly structured design and layout strategy. Novel CAD tools are used at different steps in the design process. The result is a single chip of 25 mm^2 in 3-μm double-metal CMOS. Functionality tests show that a clock of 16.7 MHz can be applied, which means that a 32-Mbit/s data rate can be achieved for all eight byte modes. This is the fastest DES chip reported yet, allowing equally fast execution of all four DES modes of operation due to an original pipeline architecture.**

I. Introduction

UNTIL the last decade, cryptography was the domain of the diplomatic and military world [3]. Due to the microelectronics (r)evolution a need for *commercial cryptography* has emerged. The ever cheaper and higher performance digital circuits have caused a rapid expansion of international telephone communications, computer networks, etc. Electronic mail, electronic funds transfer, and recently smart cards are part of daily life. But this evolution may create security threats from eavesdropping to theft. However, the same digital circuits allow for small and elegant cryptographical solutions to replace their rigid and heavy mechanical counterparts.

A classical algorithm, like the Data Encryption Standard (DES) [1], protects data in two ways. First, *privacy* is protected. After encryption, the sender can be sure that the message, sent over an insecure communication channel such as electronic mail, is only read by the intended receiver. A second and often more important demand is that of *authentication*. After decryption, the receiver can be sure that the message he received came from the original sender and no one else. Both sender and receiver want to be sure that the integrity of the message is guaranteed, i.e., that an opponent did not change, insert, or delete parts of the message.

As electronic communications and networks grow, there is not only a need for *compatibility* on communication protocols but also for common methods for data protection. To set up a secure communication between different parties of different organizations all over the world *standards* have to be defined [1], [2]. Although there are several DES chips available on the market, there is still a need for higher performance and more secure DES chips. None of the existing chips combines high speed with all requirements for high security and user-friendliness.

The chip presented here has a *unique combination of standards*. First, it implements the Data Encryption Standard, one of the most widely used standards for enciphering computation [1]. Second, it implements all operation modes specified in the standard, "DES Modes of Operation" [2]. The reason for this is that the strength of a cryptographical system depends heavily on its use. These modes give the user the opportunity to include feedback and time dependency in his ciphertext. In this way the output at a certain moment not only depends on the input but also on previous inputs or ciphertext outputs. A special approach allows the implementation of all modes without speed reduction and with a small overhead in silicon. Third, one can easily use the chip for banking-related standards (for message authentication and financial key management), as in the generation of message authentication codes (MAC's) and triple encryption.

Key management and key security both demand multiple key registers on chip. As a practical balance between user convenience of key management and silicon area, four key registers are implemented, e.g., for two master keys and two session keys. This consumes about 30 percent of the data-path area. The security of a publicly known algorithm, like the DES, depends only on the security of the keys. If the keys are compromised, the whole system is. This requires that, once entered, keys may not leave the chip. Hereto special key registers are implemented which cannot be scanned out.

More generally, from the system performance point of view, a cryptographic device should be easy to use and

Manuscript received September 28, 1987; revised February 1, 1988. This work was performed in collaboration with the Cryptech company.
I. Verbauwhede and H. J. De Man are with the IMEC Laboratory, B-3030 Heverlee, Belgium and with the ESAT Laboratory, Katholieke Universiteit Leuven, B-3030 Heverlee, Belgium.
F. Hoornaert is with Cryptech, B-1050 Brussels, Belgium.
J. Vandewalle is with the ESAT Laboratory, Katholieke Universiteit Leuven, B-3030 Heverlee, Belgium.
IEEE Log Number 8820717.

0018-9200/88/0600-0647$01.00

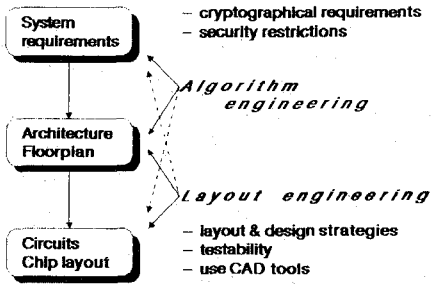


Fig. 1. General design methodology.

fast, e.g., for bulk encryptions, satellite communications, or ISDN. The insertion of an encryption device may not slow down the overall performance of a system and must also be flexible for use in a large number of environments. If someone tries to tamper, a general reset and especially a disabling of all key registers must occur.

The general design methodology of an ASIC can also be applied in this particular case, as shown in Fig. 1. All cryptographical requirements that are implemented in this chip have been described briefly. They can be found in more detail in [4]. There exist several DES chips [12]–[14]. Most of these can compute only the DES algorithm [12]. The fastest DES chip reported yet [13], with a maximum speed of 14 Mbit/s, can calculate only three limited modes and has a synchronous I/O interface which makes it difficult to communicate with other devices in a system. The Data Encryption Processor (DEP) [14] is very flexible to use and is realized as a programmable microprocessor. But this is at the cost of a much slower speed, maximum 4.7 Mbit/s, and at the cost of an overhead on silicon area. None of these chips implements with the same speed and the same ease MAC's or triple encryption or can handle multiple keys. It is unique to have all of these cryptographical requirements *combined in one device*. This has been achieved by applying a modular design strategy, the result of which is explained in Section II.

A modular and efficient floor plan could only be obtained by applying some algorithmic equivalences. The algorithm engineering can only be efficiently done if some layout knowledge and layout restrictions are known (Fig. 1). In Section III the choice of algorithmic equivalences leading to a minimum routing and silicon area are discussed.

Starting from the architecture and floor plan, the layout engineer wants to generate in a minimal design time layout that is fast, compact, and highly efficient. Therefore advanced CAD tools are helpful. The whole design process and the related CAD tools are explained in Section IV. Even for a cryptographical chip, the design conforms to a conventional design and layout strategy, as explained in Section V. However, testability is something special and at first sight seems to be in conflict with security requirements: one must be unable to scan out keys or intermediate ciphertext. The solution to this dilemma is discussed in Section VI. Finally, Section VII contains the implementation and test results of the first prototype.

The algorithm engineering design task, which consists of the translation of the cryptographical and security requirements in a floor plan and architecture, is reported in full detail in [4]. The layout engineering task is the topic of this paper. Only the important aspects of the high level design, necessary to understand the context, are repeated here.

II. Implementing a Multimode DES into a Compact Floor Plan

The aim of this work is not a straightforward implementation of the DES algorithm but to provide a more general cryptographical device of which the DES computation is only a part. In order to deal with all cryptographical requirements, the global problem has been partitioned into smaller subproblems. This results in a data path consisting of a DES part, a key management part, an input/output part, and a mode calculation part. For each module, the functionality and its implementation are discussed below.

A. The DES Algorithm and its Hardware

As shown in Fig. 2 a single DES calculation [1] is a sequence of a 64-bit initial permutation, a consecutive calculation of 16 rounds, and a 64-bit inverse initial permutation. A round can be compared with an iteration step, whereby the number of iterations is fixed at 16. The DES part, shown in Fig. 3, contains hardware for one DES round. It consists of 32- and 48-bit modulo 2 adders (XOR's $A1$ and $A2$), eight nonlinear substitution functions with six inputs and four outputs (also called S boxes), an expansion E, a permutation P, and two registers of 32 bits.

In the floor plan and layout the S boxes are realized as the eight PLA's (see the bottoms of Figs. 15 and 16). PLA's are the best way of implementing these nonlinear functions, which are defined in table format, since these substitutions are supposed to be as random as possible. PLA's are well suited to implement, in a compact and structured way, nonlinear arbitrary functions. We found that these PLA's cannot be minimized because the ZERO's and ONE's are randomly scattered. Indeed, if a reduction were possible, some structure would have to be present in the tables and that would constitute a weakness of the security of the DES algorithm. On top of the PLA's the 32-bit hardware routing for the permutation P is placed, which at the same time makes a connection between the PLA's and the 48 bit slices of the rest of the DES part. In contrast, the large 64-bit permutations at the beginning and the end could be realized without the use of routing by using the inherent structure. In this way a considerable amount of silicon area has been saved, as will be explained in Section III.

B. The Keys

The key module implements two tasks: 1) the safe storage of the keys, and 2) the key schedule calculation. For each DES round, a subkey of 48 bits has to be

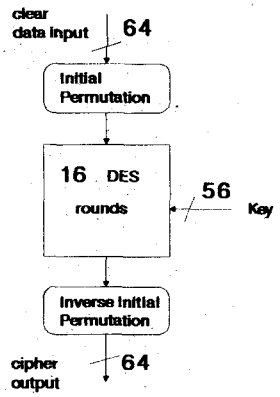


Fig. 2. The main parts of the DES algorithm.

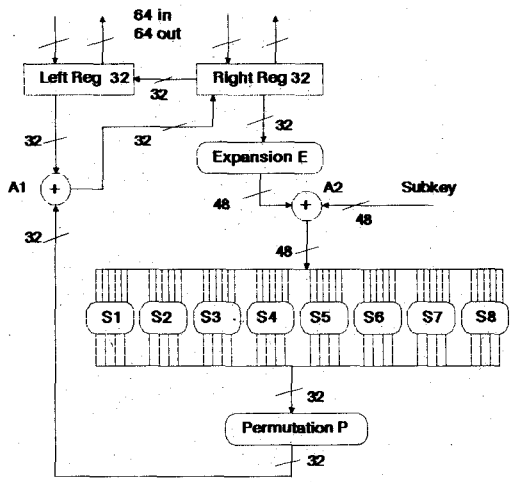


Fig. 3. Architecture for one DES round.

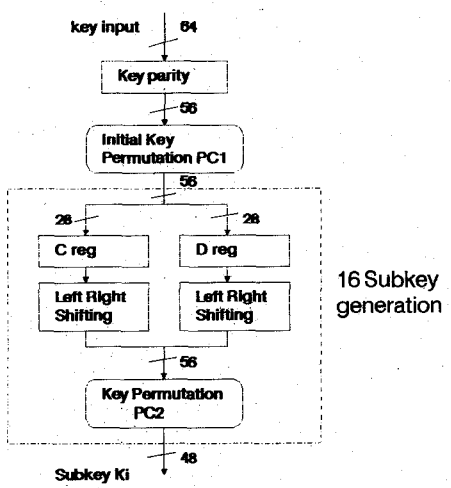


Fig. 4. The key schedule calculation.

generated, as shown in Fig. 4. The input key is also 64 bit, but 8 bits are used for parity checking [1]. After an initial key permutation, $PC1$, the 16 subkeys, one for each round, are derived from the 56-bit key selected for encryption. One subkey is obtained after some left or right shifting and after a 56- to 48-bit permutation and selection, $PC2$ [1].

The key hardware is placed in the middle of the layout (see Figs. 15 and 16). At the bottom, the key permutation

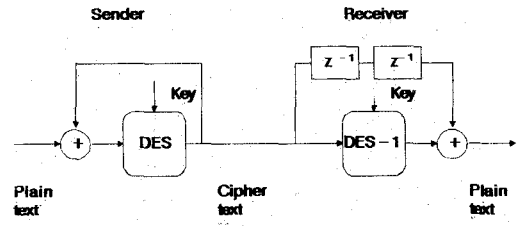


Fig. 5. Definition of the CBC mode, encryption and decryption.

and selection $PC2$ is located, which consists of two 28-bit parts placed next to each other. This enables an easy transition between the 56 bit slices of the keypart and the 32 bit slices of the DES part. Above this routing, the hardware for left right shifting (LR) is implemented. Most area is needed for the implementation of the four key registers, $KR1-KR4$. As shown in Fig. 4 an incoming key has 64 bits. The key parity is checked only once, when the key enters the chip, so 8 bits are dropped. A hardware crisscross routing for the initial key permutation $PC1$ could be saved by using the inherent structure. Therefore the initial key permutation $PC1$ is also computed only once when the key enters the chip and the keys are stored in a permuted fashion. The algorithmic equivalent to save this routing is explained in Section III.

C. The Modes: Combining Feedback and Pipelining

For a given fixed key and given fixed data input, the DES cipher output is always the same. An opponent could thus use frequency analysis to retrieve the original text [3]. To avoid this, several DES modes of operation are defined [2]. These provide feedback and block chaining. As an example, in Fig. 5 the cipher block chaining (CBC) is defined. In this mode a given fixed input and a given fixed key will produce different cipher outputs, because they do not depend only on the input but also on previous inputs.

From the implementation point of view, a *pipeline* is introduced to increase the data rate through the chip. The inherently slow on–off chip communication is done in parallel with the DES calculation. The previous cipher text is written out while the actual plain text is being enciphered and the next one is read in. At first sight this seems to be in conflict with the feedback modes. For example, when the DES device is configured in the CBC mode the new input for the following DES calculation is the bit-wise modulo 2 sum of the previous cipher output and the new data input.

This conflict has been resolved by placing the mode calculation strategically between the DES hardware and the input/output hardware and by combining mode calculation with the internal data transport on chip. How the feedback loop can be combined with the pipeline is explained in Fig. 6 for the same CBC mode (encryption and decryption). The newly entered data are XORed with the enciphered data while they are transferred from the I/O to the DES part or vice-versa [4], [5]. This mode calculation is done on the fly between every two pipeline stages as shown in Fig. 7. This technique of mode calculation does

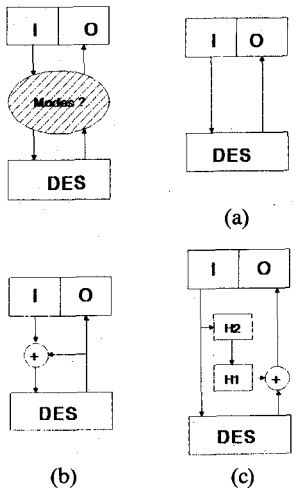


Fig. 6. Mode calculation combined with internal transport: (a) ECB, (b) CBCenc, and (c) CBCdec.

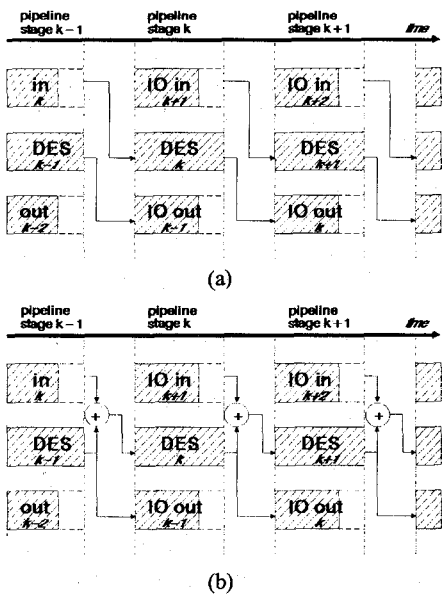


Fig. 7. Fast internal mode calculation in between two pipeline stages: (a) ECB, and (b) CBCenc.

not decrease the speed of the device. The same can be shown for all other modes. Hence a unique feature of the design, not found in other implementations, is that *all four modes are equally fast.*

D. The Input/Output (IO) Part

The input/output module realizes off-chip communication. On the floor plan of the prototype, this is still very limited. Therefore, in a second implementation an independent I/O interface (I/O frame) is being designed, which is divided into two main parts. The internal DES hardware communicates with this I/O frame in a synchronous way and performs mainly a redirection of the data or the keys. The second part can read or write data to and from the outside world in a asynchronous or semisynchronous fashion. It can combine or separate data for buses of 8, 16, or 32 bits and it contains the necessary glue logic for

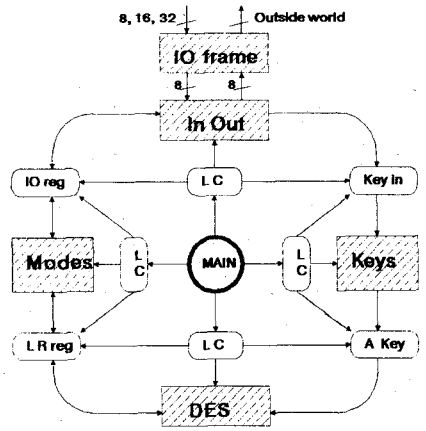


Fig. 8. Hierarchy: one head controller, four local controllers (LC's), and four data-path modules (cross-hatched).

communications with a microprocessor or for DMA. For most applications it is important that the chip can be placed in a number of different systems with minimal modifications to the original system and without decreasing the communication speed.

Moreover, if in the future this DES hardware module is used as a small encryption corner on a large VLSI digital device, it is only this I/O frame that has to be stripped off or replaced by a smaller one, e.g., for on-chip interprocessor communication.

E. The Controller Unit

In addition to the data path a controller structure is needed. Extra features like MAC's, triple encryption, etc. can be computed with the same data path but impose strict requirements on this controller. As shown in Fig. 8 the controller has also been designed hierarchically. Each module has its own local controller, which communicates only with the main controller. This central controller decodes incoming commands and decides which part of the chip has to be activated.

The *local controllers* can execute a limited number of long sequences, e.g., the local DES controller can generate sequences for DES or inverse DES (DES-1) calculations. The local controller for the modes decides the configuration of the modes instruction register for the correct internal transport and mode calculation.

The *main controller* synchronizes the local controllers for parallel execution as, e.g., for the pipeline execution of Fig. 7. It monitors the local controllers until every module has finished. The execution times can be of varying length, e.g., depending on the speed of the external I/O communication or depending on single or triple encryption.

The communication between two modules is restricted to *one common register* between every pair of modules, which is strictly monitored, as shown in Fig. 8. Neighboring local controllers can give instructions to these registers but at different time instances. For example, the I/O register is shared by the I/O part and modes part. The I/O part fills or clears it, and the modes part uses it for internal transport. The local controllers use the common

registers but only the main controller decides *when* they are allowed to use it. For example, if the I/O part is filling the I/O register, the modes part is not allowed to work. The main controller starts alternatively the DES and/or I/O controller and then the key and/or modes controller. To maximize the data rate through the chip, the key and modes sequences in between different pipeline sections (cf. Fig. 7) should be minimal. At present, the longest modes sequence takes only nine clock cycles and the longest key sequence takes three clock cycles.

In the first prototype the controller is very small, as the objective was to test the functionality of the data path. In the actually completed second implementation, the performance of the controller has been optimized to maximize data rate through the chip. The challenge for the main controller is on the one hand to provide the external user with simple powerful high-level commands, but on the other hand to keep all parallel parts of the data path calculating without losing time in between subtasks. If this were not possible, the internal calculation power of the data path would not be available to the outside world.

F. The Overall Floor Plan

For the first prototype the floor plan and layout are illustrated in Figs. 15 and 16. The relative placement shown leads to minimum wire and bus length. Each module is bit-slice designed and has the same width, though they have different word lengths. Thus modules are abutted and routing is avoided.

III. Algorithmic Equivalences for an Efficient Floor Plan

The global layout (Fig. 16) contains only two hardwired permutations, one of 56 bits (which is in fact two times a 28-bit routing) and one of 32 bits. However, the DES algorithm as defined in the standard contains three more large permutations, a 64-bit initial permutation *IP*, a 64-bit final permutation *IP*-1, and a 56-bit initial key permutation *PC*1. These permutations are *not* realized with dedicated crisscross routing but by using a shift technique and an optimal arrangement of the floor plan [4]. The three most important algorithmic equivalences which yield this hardware saving will be briefly explained.

A. Byte-Oriented IP and IP-1 Realization

For illustration, the *IP* table [1] as defined in the standard is given as Table I. It reads as follows: bit 58 of the input is the first, most significant bit of the permuted input, bit 50 is the second and so on, and bit 7 is the last, least significant bit. The corresponding straightforward 64-bit channel routing, which is not implemented, is shown in Fig. 9 to compare it with the actual data path. If the width is taken the same as for all other modules, the height is 500 μm and this only for *IP*.

TABLE I
The Initial Permutation *IP*

IP = [	58	50	42	34	26	18	10	2
	60	52	44	36	28	20	12	4
	62	54	46	38	30	22	14	6
	64	56	48	40	32	24	16	8
	57	49	41	33	25	17	9	1
	59	51	43	35	27	19	11	3
	61	53	45	37	29	21	13	5
	63	55	47	39	31	23	15	7]

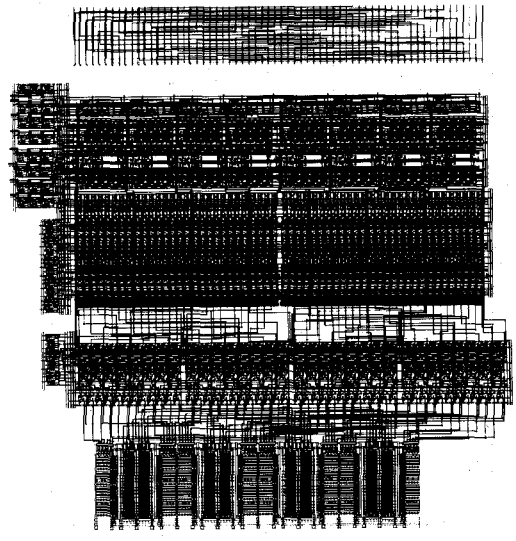


Fig. 9. Straightforward hardware routing for *IP* which is *not* implemented compared with the data path.

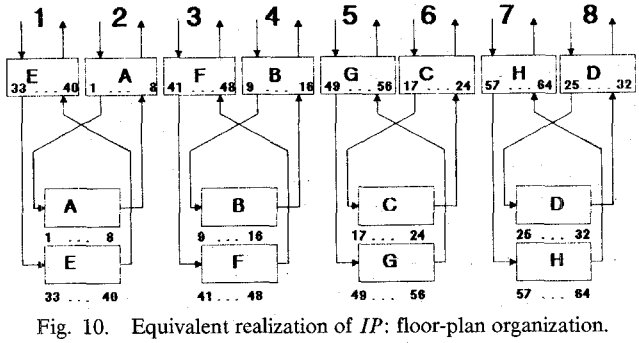


Fig. 10. Equivalent realization of *IP*: floor-plan organization.

However, *IP* is byte oriented and so can be realized with a shift technique [4]. *IP* is calculated when going on–off chip. Instead of two 64-bit complex channel routings, one for *IP* and one for *IP*-1, the 64-bit I/O register is arranged as a concatenation of eight shift registers of 8 bits. One can prove [5] that this shift-register concatenation together with a small floor-plan reorganization corresponds to the initial permutation *IP*. As can be seen in Fig. 10, no scrambling routing remains. Shifting the result back after the DES round calculations and reading it out byte per byte performs *IP*-1.

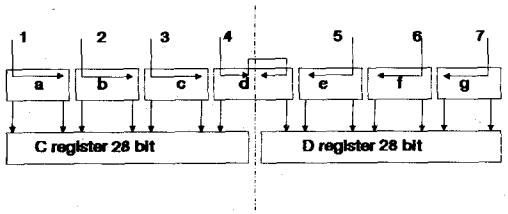


Fig. 11. Equivalent realization of *PC*1: floor-plan organization.

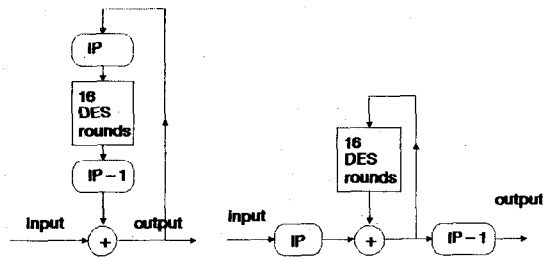


Fig. 12. Equivalent realization: *IP* and *IP*-1 outside the feedback loops of the modes.

B. Equivalent PC1 Realization

A similar shift technique with seven registers can be applied to realize the initial key permutation *PC*1. The corresponding floor plan is shown in Fig. 11. There is *no* routing of the incoming bits. The seven shift registers are combined into two 28-bit registers, necessary for key schedule calculation. Again, similar to the *IP* realization, it is the *switching from serial shift to parallel use* that performs the permutation.

C. IP and IP-1 Outside the Feedback Loops of the Modes

IP and *IP*-1 are brought outside the feedback loops while these are calculated when going on or off chip. The modes, however, are calculated on chip. For example, for the 8-byte cipher feedback mode (CFB), this is explained in Fig. 12. The simplification one obtains by bringing *IP* and *IP*-1 outside the feedback loops is based on the fact that *IP* and *IP*-1 are each others inverse, so $IP\text{-}1(IP(\text{Data})) = \text{Data}$.

The combination of both techniques, the equivalent shift technique and bringing *IP* and *IP*-1 outside the feedback loops, allows for very compact realization of the modes hardware. First the hardware routing for three permutations is saved. This would produce an increase of 30 percent of the data path without considering the placement problems. Second, by shifting from the I/O register to the internal register, only eight XOR's, eight multiplexers, and eight bit buses are needed, instead of an area-consuming 64 XOR's, 65 multiplexers, and 64 bit buses such as used in [14].

IV. The Design Process and Use of CAD Tools

The design evolution of the data path, from specifications to layout, is presented in Fig. 13. It can be broken up into five phases.

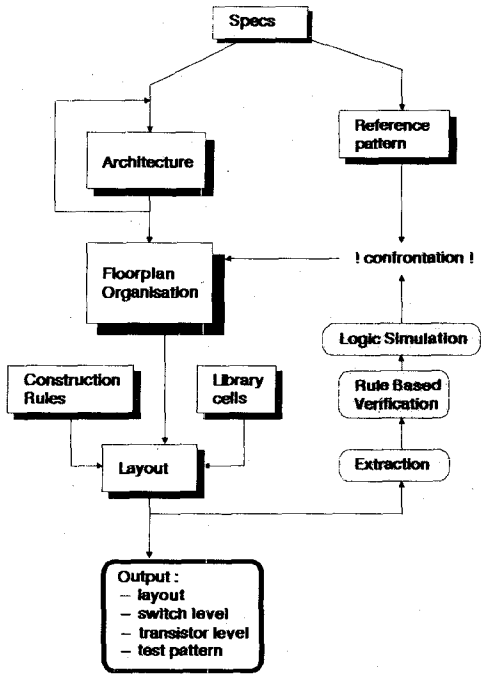


Fig. 13. Design evolution of the data path.

A. The Architecture Synthesis

The DES algorithm, its modes of operation, and the other cryptographical specifications have already been explained above. The feedback loop over architecture represents the choice of the algorithmic equivalences thereby producing a better floor plan.

B. The Floor-Plan Organization and the Cell Library

The result of the first iteration loop is the *floor-plan organization* discussed above on which the optimal relative position of the different elements is established. Based on that floor-plan organization a minimum cell library was defined.

Besides the eight PLA's, registers, multiplexers and XOR's are needed to implement the DES, key, and modes hardware. Except for the PLA's, we have been able to build all of the other logic out of only *four* gates: a static XOR and three types of inverters (with small, medium, or large W/L). This is possible because DES consists essentially of data scrambling and bit-wise modulo 2 arithmetic. Pass transistors are used to implement the clocks and conditions of multiplexers and registers (a maximum of two in series is permitted). The inverters can be provided with, or without, static refresh to implement memory. As an example, a register with a two-input multiplexer at the input is given in Fig. 14. The basic cells have been characterized individually for the transistor dimensions, the driving capability, etc. "Special" cells are the eight PLA's used to realize the substitution functions. These are also characterized separately.

The *cell layouts* are, however, *customized* to the floor plan. For one cell different versions exist, depending on the width of the bit slice. Typical for the DES algorithm is

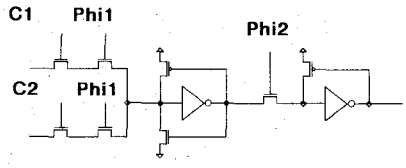


Fig. 14. Register implementation with two-input multiplexer, realized using pass transistors and refresh inverters.

the use of different word lengths, 8×8, 32, 48, and 56. To generate a floor plan on which all modules have the same width, the cells had to be laid out on different pitches. This has been done using the pitch matching capabilities of the symbolic layout editor, CAMELEON [11].

To implement the "glue" logic of the I/O interface and the controller, some more logic gates are necessary. Because they are not critical in generating a compact DES device, they are taken from a standard cell library.

C. Construction Rules and Verification

A novel way of design has been the use of *a priori* defined construction rules and the use of an expert system to check consistent design practice. The basic logic gates and cells are connected in accordance with some *construction rules* including clocking and timing rules, driving capability rules, etc. Some examples are as follows.

1) For correct clocking and control: "*Conditional clocking is only allowed on clock Phi1.*" The reason for this is that it would cause an unfeasible controller implementation later on.

2) For correct memory implementation: "*Conditional clocks always have to be followed by a refreshing inverter.*" The first reason is for level restoration, to restore the threshold-voltage V_t drop after a pass transistor, and second to avoid leakage problems as it is unknown how long a value must be stored in a register. As shown in Fig. 14 the Phi1 conditional clocks are followed by full static refresh; for the Phi2 latch a restoring PMOS transistor is sufficient.

3) For correct logic levels: "*A 'medium' inverter may drive at most two refreshing inverters through a pass-transistor network.*" This is to avoid problems with ratioed logic. Indeed, static refresh constitutes a ratioed load to the preceeding inverter which switches the refreshing inverter.

4) For capacitive loading: "*A 'small' inverter may drive a maximal capacitive load of* 500 *fF.*" These capacitive loads are estimated initially and checked afterwards by extraction programs.

For a large digital chip a rigorous construction strategy must be followed. One cannot allow "fancy tricks" to save gates or transistors. Such local optimization carries a penalty back, when the overall system has to be tested, controlled, or verified. How the construction rules are used for a novel way of verification is discussed next.

For *verification* the construction rules for building the data path are written in the LISP-based language LEXTOC (Language EXpressing TOpology Constraints) of the expert system DIALOG [6]. The formal description results in a knowledge base of 85 rules. The rules are formulated in such a way that everything is forbidden except what is consistent with the rules, instead of accepting everything except what is forbidden by the rules. The reason for this is that otherwise some construction errors can pass undetected.

The DIALOG system verifies first the correctness of the *library topology*. This means that every cell that does not conform to the four basic cells is reported to the designer. Refresh inverters are identified by their weak W/L ratio. PLA's are identified and treated separately.

Second, the system verifies the *library rules*. These include:

a) pass-transistor verification—e.g., the length of a pass network is checked, or it is verified that every pass transistor network is followed by at least a level-restoring inverter;
b) electrical problems—e.g., to detect possible problems with charge sharing, the ratio of the capacitances at the input and output nodes of a pass-transistor network is calculated and problem nodes are identified;
c) timing verification—this is mainly used for the verification of the correct implementation of conditional clocks; and
d) capacitance checks—DIALOG calculates the load of every cell and checks whether it exceeds the maximum value specified in the rules. This load incorporates input (gate) capacitances, routing capacitances and diffusion area capacitances.

The verification of the transistor level description of the whole DES data path (12K transistors and 6K capacitances) took 45-min CPU time on a Symbolics machine. The main advantage of this verification is that it is complementary to simulation. With simulation one detects faults that one is searching for. With this verification mainly connection faults at the *interface* between different modules are detected, particularly construction violations and exceeded capacitive loads.

D. The Layout Generation

The floor plan, the adjusted cell layouts, and the construction rules form one global layout. Placement and routing of the different modules is done by hand.

The hardwired permutations, P and $PC2$, are difficult to route by hand. Therefore the router of the module generator environment (MGE) [10] has been used. Terminal connections can be specified in a textual form using LISP. The resulting routing has a poor density compared with what could be obtained by hand routing. In this case, however, writing connections in a textual form is less sensitive to errors and mistakes are more easily detected because the permutations tables are copied from the standard [1], similar to Table I for *IP*.

E. ERC and Logic Simulation

An electrical rule check is performed on the layout. The output of the extraction is a transistor-level description which can be converted easily to a switch-level description. The latter forms the input for a logic simulation. The first goal is to check the floor plan and in particular the *global routing* of the floor plan, which is part of the algorithm implementation. The whole data path needs to be simulated because routing, e.g., for the permutations, can only be checked in relation to neighboring modules. A second goal of the logic simulation is to generate test patterns for the processed prototypes.

Indeed, a *functional reference pattern* for an adder or a multiplier is easy to check. But it is not obvious that for the hexadecimal data word *567890ab cdeff31f* and the key *01234567 89abcdef*, the first right input is *7c6abae9* and the first *S*-box input is *b48132c41ef7*. Reference patterns are generated by an independent software implementation of the DES algorithm and checked with independently available reference tables.

F. The Result: A Cryptographical Hardware Module

The result of the whole design process is a hardware module for the data path: full layout, transistor- and switch-level descriptions, and test patterns are produced. This module can be used as a black box at a higher abstraction level in system design, as a stand-alone encryption device or as a small encryption corner on a larger VLSI circuit.

V. Structured Design and Layout Strategies

It will be explained in this section that the design fulfills general design and layout strategies. The application of these techniques eases the design effort.

A. A Structured Design Methodology

In general, four techniques [7] are available for reducing the complexity of IC design: hierarchy, modularity, regularity, and locality. These techniques are closely related. The implementation of the DES algorithm itself is only a small part of the problem. The architecture for one DES round is easy to deduce and the control for this hardware is a well-defined sequence. But the realization and interaction of all aspects, the mode calculation, the key handling, and the I/O interface make it a complex problem. It can be said that a more attractive solution is obtained when all four techniques are applied.

For instance, the controller is designed in a hierarchical way. One controller is master and the others are slaves. Each controller is a separate module. Modularity means well-defined interfaces. The terminals and the timing are defined for higher levels and what is inside is transparent. This is the definition of locality. The same is valid for the data path. The communication between two modules is restricted to one communication register between every two modules as explained in Section II. The use of these registers is submitted to very strict timing and communication protocols, but they apply everywhere (regularity). Modularity allows also the reuse of the same modules for *DES-like algorithms*, and if something has to be changed, this will have only local influence.

Regularity is most clearly visible in layout, as shown in the chip photograph in Fig. 16. But regularity also exists in the consistent application of the construction rules, when constructing logic gates, registers, etc., into functional building blocks. The same construction rules, sometimes termed a methodology, can be coded in a verification tool to allow automatic verification.

B. Layout Strategies

Typical for this design is the use of *different word lengths* in different places. The DES part has a combination of 32 and 48 bit slices. The key part has 56 bit slices and the I/O part is byte oriented and has eight blocks of 8 bits. This has several consequences on the general layout strategy.

First, to obtain a more or less square layout, *cell stretching* is applied. Each module, independent of the number of slices, is 4 mm wide. The DES part has a combination of 32 and 48 bit slices. The key part has 56 bit slices and the I/O part is byte oriented and has eight blocks of 8 bits. Most connections between cells inside the modules are made by *abutment*. This means that a bit slice in the key part is only 70 μm wide compared to 120 μm for a bit slice in the DES part. The layout of a register in the DES part will thus be arranged differently from a register in the key part.

Second, the *relative placement* of the routing channels for the permutations also allows the transitions from one word length to another. For example, *PC*2 is placed between the 56 key bits and the 32 data registers, and the expansion *E* is placed between the 32 and 48 data slices. Also, PLA's are difficult to fit nicely on a bit-sliced data path, therefore they are placed at the bottom and the terminal fitting is included in the permutation *P* between the 48 data slices and the eight PLA's.

Third, the variation in word lengths has influence on our *power and ground routing strategy*. A double-metal technology is used in the design. The two metal layers run *perpendicular* to each other on chip. Metal1 runs parallel to the data flow, in the vertical direction, because there are more data contacts to be made (in metal1). Metal2 runs horizontally, parallel with the control flow. Control is mostly supplied in poly to make connections to the gates of (pass) transistors. Long poly wires have *RC* problems. In this case the maximum length amounts to 4 mm. Therefore metal2 is used to bypass these long poly wires.

Power and ground lines run also horizontally, perpendicular to data flow. Because of the problem of different word lengths power and ground cannot be provided vertically for two reasons. First, crossovers and crossunders would be unavoidable at the interface between

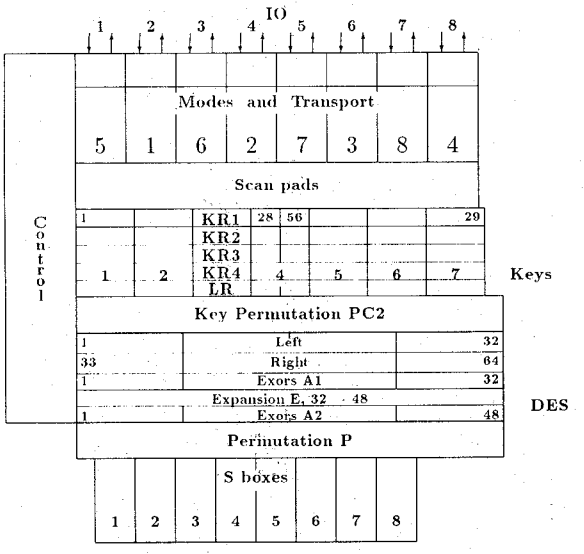


Fig. 15. Floor plan of the data path of the first prototype.

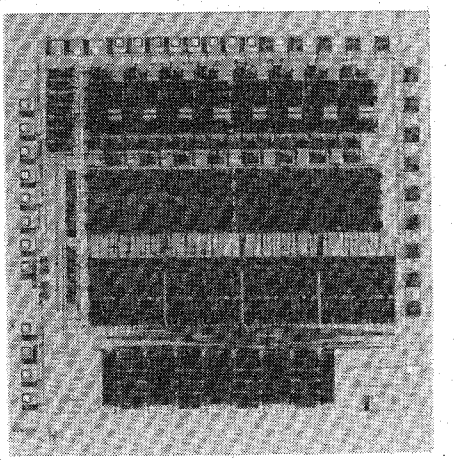


Fig. 16. Chip photograph of the implemented prototype.

different modules. A second problem is caused by the very small pitches in some modules such as the key registers. Even by mirroring the cells, it would be impossible to arrange the wells and to separate the NMOS and PMOS transistors. Now the n-wells are connected by abutment horizontally over the different bit slices. Power and ground lines are used in common between two different cells.

Usually, however, power is distributed along the signal lines to avoid surge currents on the power lines [8]. The problem of surges on the power lines is solved by our typical double-metal power and ground distribution. A classic finger or fork structure which is typical for a single-metal-layer technology has the problem that all current passes through one critical bottleneck, the stem of the fork. In this design, this is solved by providing a double ring of metal1 and metal2 all around the data path. The metal1 ring is the ground ring, and the metal2 ring is the power ring. Each horizontal power or ground line through the cells is connected at both sides to the rings. Via connections between metal1 and metal2 are necessary but they are multiple and the current is more equally distributed along the ring. Thus there is no current concentration and power dips are unlikely to occur.

VI. Testability of a Cryptographical Device

On the floor plan of the first prototype (Fig. 15) it can be seen that some *scan registers* are implemented. This scan technique is very useful for *localizing* faults in a design during development. But this is in conflict with security demands. Scan pads provide a very easy way to scan out keys, initialization vectors or intermediate ciphertext. Therefore they are not allowed for safe commercial applications. They are only provided during development and are realized as independent blocks instead of combining them with existing registers. This means overhead on area, but they are very easy to remove.

Scan pads are removed for safe commercial applications. For validation or maintenance purposes, the chip must still be testable. These tests are based on *signatures*. A test signature uses the same propagation principle as a digital signature as in the generation of MAC's. For a MAC, which is based on the CBC mode, the function is to detect an illegal change or an insertion of false data. A small change of input data will completely change the signature.

The same propagation principle is used for test signatures. A known clear data and key pair is applied, and the result is compared with a reference cipher text after a number of encryptions and decryptions. Complete functionality of the DES will be tested and stuck-at-ONE, stuck-at-ZERO faults will be detected. The typical stuck-open and dynamic faults of static CMOS design are a topic of further research. *While in other chip designs, an overhead on logic and chip area is necessary to realize fault propagation, in DES it is inherently present.*

VII. Implementation and Test Results

The first prototype, with scan registers, has been processed using a 3-μm N-well CMOS process with double metal (see Fig. 16). It contains 12 000 transistors and has an area of 25 mm^2. The aim was to test the functionality of the different parts, mainly the DES and the modes part. Tests have shown full operation up to 16.7-MHz clock rate. With this clock frequency, a 20-Mbit/s data rate on all eight byte modes is achieved.

A second implementation is now being developed, which takes the data path as a hardware module and which also contains the full controller and microprocessor interface. The scan registers have been removed. Thanks to a high-performance controller, a full DES calculation, including the mode calculation, takes only 28 clock cycles. If the same clock frequency can be applied to this device, a data rate of 32 Mbit/s can be expected.

VIII. Conclusions

A single chip has been designed and implemented which executes DES with a number of unique cryptographical features. The original approach of mode calculation allows pipelining in combination with feedback modes. Due to

the structured design strategy, the global rather than local optimization, the regular floor plan, and the use of CAD tools at all design levels, the main architecture and many building blocks can be reused in a flexible way for safe commercial applications or for DES-like algorithms. Checking at different design levels, including simulation, verification and electrical and design rule checks are essential and complement each other. The outcome is a compact design which can be used as a small module in larger digital VLSI circuits, but also as a fast stand-alone device. In short, it has a number of unique features not found in other devices.

Acknowledgment

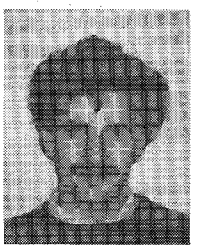
The authors wish to thank I. Bolsens and W. De Rammelaere for their help in using the DIALOG expert system for chip verification, the INVOMEC MPC service chip processing, and the members of the VSDM division for stimulating discussions.

References

[1] *Data Encryption Standard*, Federal Information Processing Standard (FIPS) 46, Nat. Bur. Stand., Jan. 1977.
[2] *DES Modes of Operation*, Federal Information Processing Standard (FIPS) 81, Nat. Bur. Stand., Dec. 1980.
[3] W. Diffie and M. E. Hellman, "Privacy and authentication: An introduction to cryptography," *Proc. IEEE*, vol. 67, no. 3, pp. 397–427, Mar. 1979.
[4] I. Verbauwhede, F. Hoornaert, J. Vandewalle, H. De Man, and R. Govaerts, "Security considerations in the design and implementation of a new DES chip," to be published in *Proc. Eurocrypt 87 Conf.*, Springer-Verlag.
[5] M. Davio, Y. Desmedt, J. Goubert, F. Hoornaert, and J.-J. Quisquater, "Efficient hardware and software implementations of the DES," in *Advances in Cryptology, Proc. Crypto 84*, Aug. 1984.
[6] H. De Man, I. Bolsens, E. Vanden Meersch, and J. Van Cleynenbreughel, "DIALOG: An expert debugging system for MOSVLSI design," *IEEE Trans. Computer-Aided Des.*, vol. CAD-4, no. 3, pp. 303–311, July 1985.
[7] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design, A Systems Perspective*. Reading, MA: Addison-Wesley, 1984, pp. 238–240.
[8] L. A. Glasser and D. W. Dopperpuhl, *The Design and Analysis of VLSI Circuits*. Reading, MA: Addison-Wesley, 1985, pp. 316–317.
[9] M. Bartholomeus, L. Reynders, M. Pauwels, and H. De Man, "PLASCO: A procedural silicon compiler for PLA based systems," in *Proc. IEEE 1985 Custom Integrated Circuits Conf.*, pp. 226–229.
[10] H. De Man, J. Rabaey, P. Six, and L. Claesen, "Cathedral-II: A silicon compiler for digital signal processing," *IEEE Design&Test*, pp. 13–25, Dec. 1986.
[11] K. Croes, H. J. De Man, and P. Six, "CAMELEON: A process-tolerant symbolic layout system," *IEEE J. Solid-State Circuits*, vol. 23, no. 3, pp. 705–713, June 1988.
[12] R. H. Cushman, "Data-encryption chips provide security—or is it false security?," *EDN*, vol. 27, pp. 39–45, Feb. 17, 1982.
[13] D. MacMillan, "Single chip encrypts data at 14 Mb/s," *Electronics*, vol. 54, pp. 161–165, June 16, 1981.
[14] R. C. Fairfield, A. Matusevich, and J. Plany, "An LSI digital encryption processor (DEP)," *IEEE Commun. Mag.*, vol. 23, no. 7, pp. 30–41, July 1985.

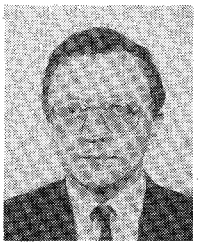
Ingrid Verbauwhede was born in Leuven, Belgium, on October 24, 1961. She received the electrical engineering degree from the Katholieke Universiteit Leuven, Heverlee, Belgium, in 1984. Her M.S. thesis was on analog design: the study and design of a micropower instrumentation amplifier based on switched-capacitor techniques.

Since September 1984 she has been at the VSDM (VLSI Systems Design Methodologies) division of the IMEC Laboratory, Heverlee, Belgium, working towards the Ph.D. degree in electrical engineering. She is a member of the research group on applications and architectural design strategies, under the direction of Dr. F. Catthoor, Prof. H. De Man, and Prof. J. Vandewalle. Her research interest is in the design and integration of cryptography applications and currently of algebraic data processing applications. In particular, the formalization of the architectural methodologies is envisioned.



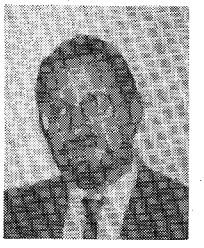
Frank Hoornaert was born on October 20, 1961 in Belgium. He received his education from the Klein Seminarie Roeselare and from the Katholieke Universiteit Leuven, Heverlee, Belgium.

From October 1984 to September 1986 he was a Chip Designer at IMEC, Heverlee, Belgium. He worked on hardware design and development of security systems at the Katholieke Universiteit Leuven from October 1986 until February 1987. Since March 1987 he has been a Security and Development Manager at Cryptech n.v., Brussels, Belgium. His hardware and software knowledge includes gate array and full-custom design experience, digital design with PP and TTL components, and design of cryptographic hardware.



Joos Vandewalle (S'71–M'79–SM'82) was born in Kortrijk, Belgium, on August 31, 1948. He received the engineering degree and a doctorate in applied sciences, both from the Katholieke Universiteit Leuven, Heverlee, Belgium, in 1971 and 1976, respectively.

From 1976 to 1978 he was Research Associate and from July 1978 to July 1979 he was Visiting Assistant Professor, both at the University of California, Berkeley. Since July 1979 he has been at the ESAT Laboratory of the Katholieke Universiteit Leuven, where he is currently Professor. His research interests are mainly in mathematical system theory and its applications in circuit theory, control, signal processing, and cryptography. He has authored or coauthored about 70 papers in these areas.



Hugo J. De Man (M'81–SM'81–F'86) was born in Boom, Belgium, on September 19, 1940. He received the electrical engineering degree and the Ph.D. degree in applied sciences from the Katholieke Universiteit Leuven, Heverlee, Belgium, in 1964 and 1968, respectively.

In 1968 he became a member of the staff of the Laboratory for Physics and Electronics of Semiconductors at the University of Leuven, working on device physics and integrated circuit technology. From 1969 to 1971 he was at the Electronic Research Laboratory, University of California, Berkeley, as an ESRO-NASA Postdoctoral Research Fellow, working on computer-aided device and circuit design. In 1971 he returned to the University of Leuven as a Research Associate of the NFWO (Belgian National Science Foundation). In 1974 he became a Professor at the University of Leuven. During the winter quarter of 1974–1975 he was a Visiting Associate Professor at the University of California, Berkeley. Since 1984 he has been Vice-President of the VLSI systems design group at the IMEC Laboratory, Leuven, Belgium.

Dr. De Man was an Associate Editor for the IEEE JOURNAL OF SOLID-STATE CIRCUITS from 1975 to 1980 and was European Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN from 1982 to 1985. He received a Best Paper Award at the ISSCC of 1973 on Bipolar Device Simulation and at the 1981 ESSCIRC conference for work on an integrated CAD system. In 1986 he became Fellow of the IEEE. His actual field of research is the design of integrated circuits and computer-aided design.