Security Considerations in the Design and Implementation of a new DES chip [1]

Ingrid Verbauwhede[1,2] Frank Hoornaert[3] Joos Vandewalle[2] Hugo De Man[1,2]

IMEC v.z.w.[1]	ESAT, K.U.Leuven[2]	CRYPTECH n.v.[3]
Kapeldreef 75	K. Mercierlaan 94	Lloyd George Av. 6
B-3030 Heverlee	B-3030 Heverlee	B-1050 Brussel
Belgium	Belgium	Belgium
Tel: 32-(0)16-281211	Tel: 32-(0)16-110931	Tel: 32-(0)2-6425931

1 Introduction

This paper describes the impact of cryptographic requirements on the design of a new highly performant DES chip implementation. Actual cryptographical applications demand for both high security and high speed. It is the aim of this contribution to show how both can be combined.

High security is obtained, because not only the DES algorithm but also all "DES Operation Modes" are implemented together with other extra security features, like triple encryption, MAC's and key security. Some cryptographical optimizations and equivalence transformations, which do not compromise the DES nor the security, were choosen for their very efficient hardware implementation with minimal routing, which otherwise would present a serious problem for any data scrambling algorithm like DES.

High speed is achieved, because these optimizations could be combined with typical chip design principles. So the datarate through the chip can be enhanced and a very compact realisation is possible.

The result is a *single* chip implementation in a classic standard CMOS process. Functionality tests show that a clock of 16.7 Mhz can be applied, which means that a *32 Mbit/sec datarate* can be achieved for all 8 bytes modes. This is the fastest DES chip known today.

Although there are several DES chips available on the market, e.g.[8], there is still a need for more performant and more secure DES chips. None of the existing chips combine high speed with all requirements for high security and userfriendliness. In section 2, all cryptographical requirements that are implemented are drawn up. For each demand, the corresponding hardware solution is explained. It is unique to have all these demands *combined in one device.* Therefore, it is necessary to understand the basic hardware design principles and constraints. This

[1] Research performed in collaboration with the company Cryptech n.v.

D. Chaum and W.L. Price (Eds.): Advances in Cryptology - EUROCRYPT '87, LNCS 304, pp. 287-300, 1988.

is explained briefly in section 3. There are lots of alternative DES representations possible, but only these that overcome the hardware limitations are selected. In section 4 it is explained which ones are selected and how this resulted in an efficient realization. At first sight, testability seems to be in conflict with security requirements : one may not be able to read out keys or intermediate ciphertext while testing the chip. How security is combined with testability is discussed in section 5. Finally section 6 contains the implementation and test results of the first prototype and some predictions are made for the second version, which is now being realized.

2 Demands of cryptographers and the hardware solution

The chip presented here is built in a modular way. This made it possible to handle all requirements separately and to provide a hardware solution corresponding to each problem. The different hardware modules are placed together on the layout afterwards.

Below follows an enumeration of the cryptographical demands. For each demand, the corresponding hardware solution is explained and indicated on the global floorplan, Fig. 2, and layout, Fig. 3.

1. *DES and DES-like algorithms.*
   Since there are questions around the safety of DES, the first objective is to implement DES as well as DES like algorithms. The first module contains hardware for one DES round, as shown in Fig. 1. On the layout in Fig. 3 one can easily see at the bottom 8 PLA's for 8 substitution functions. On top of the PLA's the hardware routing for the permutation P is placed. Above the permutation, the left and right registers, the 32 and 48 exors and the expansion E are placed.

   To realize *DES-like algorithms* one can simply cut out the S-boxes or the permutation P and replace them by others. Or one can enhance the number of rounds to increase the calculation effort of an exhaustive search attack. On the layout, this corresponds to setting the counter of the local controller to a higher number.

2. *Key management and security.*
   The *Key scheduling scheme* of the DES algorithm is implemented in a second module, combined with the safe key storage. The key part is placed in the middle of the layout as shown in Fig. 2 and Fig. 3. One can see the key permutation and selection PC2 and the left right shifting hardware LR. Most area is needed for the implementation of the four key registers, KR1 to KR4, e.g. for two master keys and two session keys.

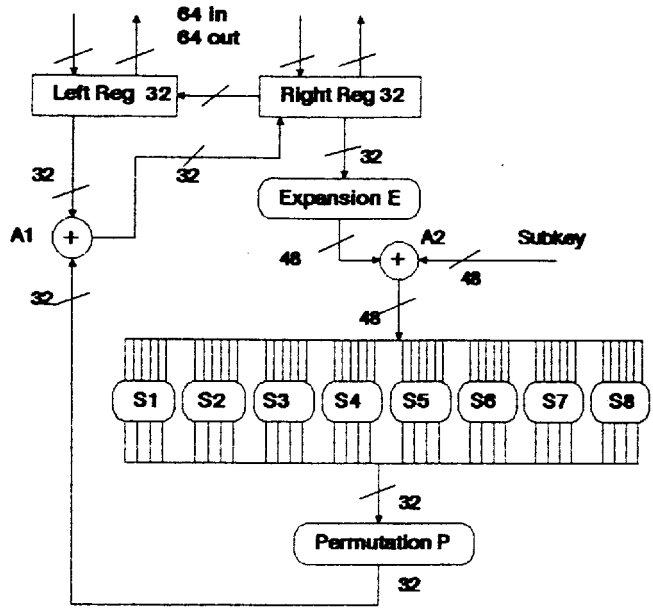


Figure 1: Architecture for one DES round.

The key scheduling is combined with the keyregisters because it makes it more easy to change it for DES-like algorithms. e.g. One can ask for another left right shifting sequence or for the use of two keys (128 bits) or more during one encryption.

For the safety of the keys, it is impossible to read out keys, once they entered the chip, [3]. It is also made impossible to use partial keys. (e.g. a half new and a half old key.) If a new key is entered in a register, the old key is disabled and can not be used anymore for encryption. When someone tries to tamper, a general reset occurs and all keys are disabled. Also key parity is checked on chip and keyexoring of incoming keyparts is possible.

3. Remark that besides the 16 DES rounds and the keyscheduling scheme, the DES algorithm contains also the initial and inverse initial permutation, IP and IP-1, and the initial key permutation PC1. These are realized in a different way using the structure observed in it. This allows considerable savings in silicon area as will be explained in section 4.

4. *All Modes ON chip.*
Another very important demand was the realization of *all modes without speed degeneration.* We have provided all modes as described in [2] in 8 bytes form (ECB, CBC, CFB, OFB) and in 1 byte form (CFB, OFB). Our technique of mode calculation does not influence the speed, all modes are equally fast. The datarate depends only on the number of bytes. The throughput for 8 bytes is 8 times faster as the throughput for 1 byte. This is easy to understand while,

independent of the fact that one is encrypting for 1 byte or 8 bytes, the DES calculation itself is always on 8 bytes (or 64 bits).

A separate module is provided for the mode calculation and the internal transport. It is placed in between the IO part at one side and the DES part and keypart at the other side. It is built of registers, multiplexers and exors. The particular configuration for one or another mode depends on the control bits that are installed in the mode control register and this is software programmable.

5. *Userfriendliness : single commands and powerful IO interface.*
   It is clear that the internal performance of a chip may not be completely available to the outside world. Therefore, for this device, large attention is made to implement *single powerful commands.* One can e.g. start continuous encryption in an arbitrary mode on one command.

   To enhance the secure use of the chip one can execute *triple encryption* in combination with all modes on the same simple commands as single encryption. Moreover for banking applications a command is provided to generate *MAC's* (Message Authentication Codes).

   These requirements for a powerful convenient use of the chip do not create extra datapath hardware. But the translation from high level compact commands to low level instruction bits puts high effort on the design and implementation of the *controller.*

   A second topic which influences the utility of a chip is a powerful *input output interface.* From the system performance point of view a cryptographic device should be *fast*, e.g. for bulk encryptions or satellite communications or ISDN. The insertion of an encryption device may not slow down the overall performance of a system. But it must also be *flexible* to be usable in a large number of environments. This DES device can be used as a fast stand alone device in communication with a microprocessor or DMA. But it is also very compact and small. So in the future, it can also be placed as a small encryption corner on a large digital VLSI design.

The controller and the four main datapath modules, the DES part, the keypart, the modes part and the IO part are arranged together on one global floorplan and layout, as shown in Fig. 2 and Fig. 3.

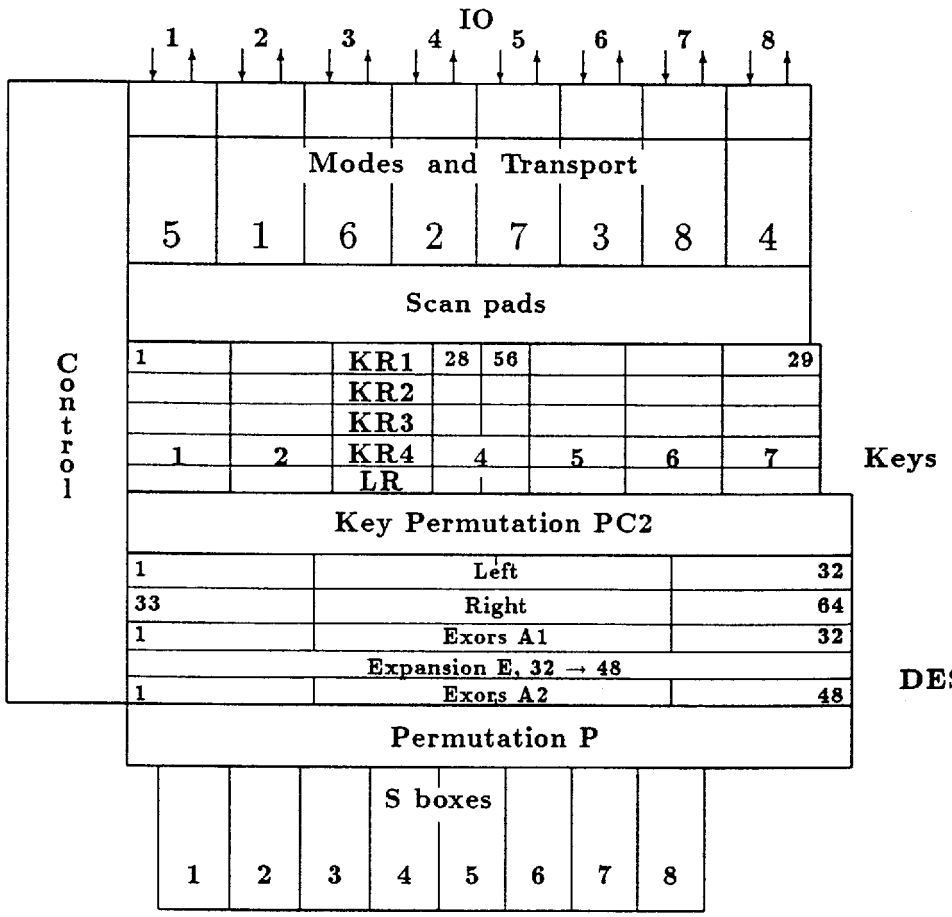


Figure 2: General Floorplan of the first prototype.

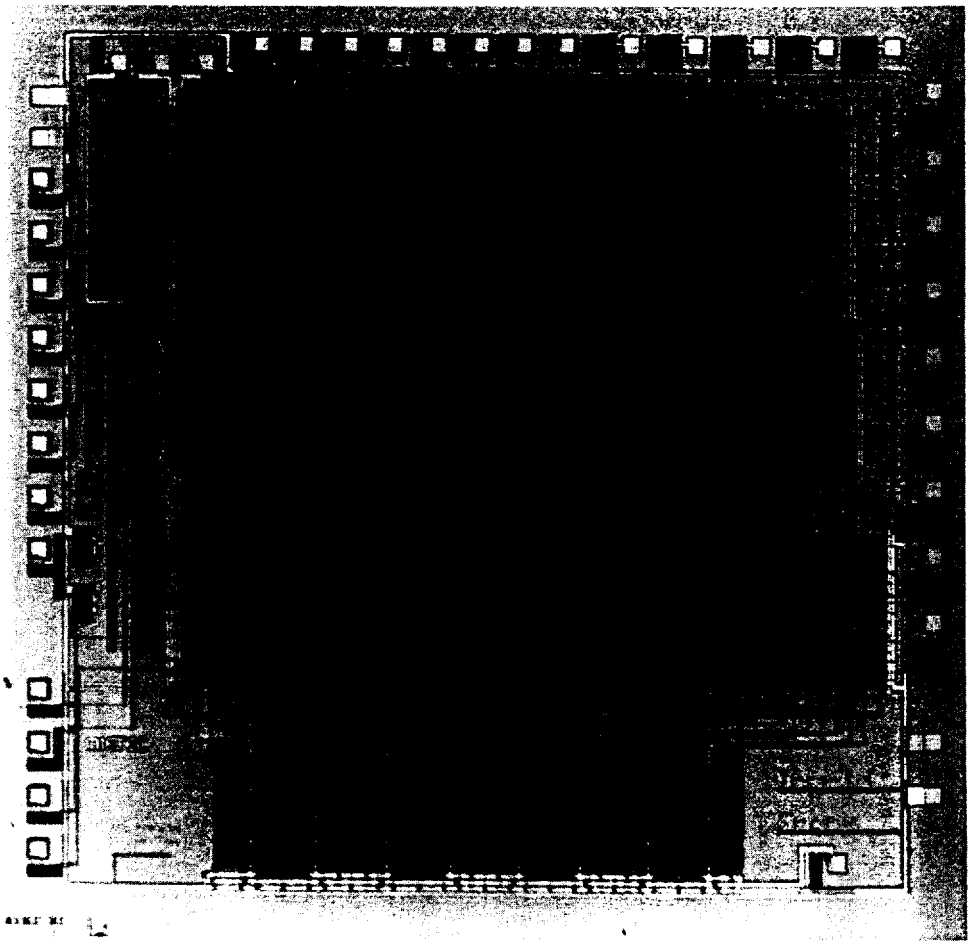


Figure 3: Chip photograph of the first prototype.

3 Hardware design principles

In order to implement an algorithm on chip, first of all one asks for a correct implementation. Moreover, for economical reasons, the chip should be the fastest on the smallest silicon area and should be realized in a minimal design time. But from the design point of view an optimal trade-off has to be made between algorithm, speed and chip area, while keeping the chip testable.

To enhance the *speed,* first the *off chip communication* should be minimized. This means that more tasks are performed on chip. Powerful commands are necessary so that the number of commands for one execution is limited. Second *pipelining* raises speed. Hereby one calculates independent things in parallel as much as possible. The parts on chip that require the largest number of execution cycles, i.e. those that are the slowest, are to be kept busy. Their idle time must be as small as possible. e.g., On this chip the time consuming parts are the DES part and the input-output part. The DES part takes much time because one has to calculate the 16 rounds one after another. And the IO part is slow because going on-off chip is inherently slow. Therefore, the previous ciphertext is written out while the actual plaintext is being enciphered and the next one is read in. At first sight this seems to be in conflict with the feedback modes where one needs previous output to calculate next inputs. But this is not the case, as explained further in section 4.

Area is expensive and should be kept minimal. So the routing must be reduced and a good floorplan is essential. A floorplan for a chip designer is what a floorplan of a house is for an architect. Also the architecture design consists in deciding which building blocks are necessary in order to implement the algorithm. e.g., To save area the 16 DES rounds are calculated in sequence on the same hardware part instead of repeating this hardware 16 times. This would be area inefficient and it would be impossible to keep all 16 DES hardware parts busy, because pipelining on this level is in conflict with the feedback modes.

One way to reduce design time is to apply the principle of divide and conquer. *Modularity* is fully exploited in this design. It is apparent as well on the system design level as on the floorplan and layout. The four parts of the floorplan are designed independently, work independently and communicate only with each other through one common register which is strictly watched. Each part has its own local controller. A modular design is *easily modified.* A change has only local influence. e.g., One can easily cut the actual S-boxes and replace these by others.

This modularity is also reflected in the floorplan. So the routing between the blocks remains minimal.

4 Algorithmic equivalences for an efficient floorplan

The off chip communication and the internal datatransport are *byte oriented* to reduce the number of IO pads and the width of the internal and external busses. Busses of 8 or 16 bits are a good trade-off between area and speed. To enhance the speed, mode calculation is done on chip in combination with the internal transport.

It has been proven, [4], that the initial and inverse initial permutations IP, IP-1 and the initial key permutation PC1 are also byte oriented. Therefore these permutations are *not* realized with an area consuming hardware routing but with a shift technique and an optimal placement on the floorplan. The most important algorithmic equivalences to save this hardware will be explained below.

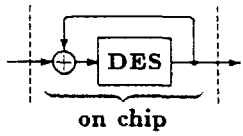
4.1 Mode calculation on chip : combining pipeline and feedback

The mode part is placed strategically between the DES hardware and the Input Output part. When both time consuming tasks are fulfilled, the external IO of the previous output and the next input and the DES calculation of the actual ciphertext, then the internal datatransport with the mode calculation is done. On Fig. 4 this is explained for the Cipher Block Chaining Mode. The newly entered data is exored with the enciphered data while they are carried from the IO to the DES part or vice versa, [4]. This mode calculation on the fly is done between every two pipeline stages as shown in Fig. 4c. This technique of mode calculation does not decrease the speed of the device, while data transport is inherently present on chip. The same can be proven for all other modes. Hence a unique feature of the design, not found in other implementations, is that all four modes are equally fast.

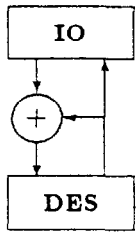
4.2 Byte oriented IP and IP-1 realization

The straightforward 64 bit hardware routing only for IP is given in Fig. 5a. If one assumes the same width as for all other modules, the height is about 500 μm and this only for IP. In reality however, IP is byte oriented and can be realized with a shift technique, [4]. IP is calculated when going on-off chip. Instead of two 64 bit hardware routings, one for IP and one for IP-1, one uses 8 shift registers of 8 bits and two small 8 bit routings, Fig. 5b. Indeed, one can prove that when shifting byte per byte into a shift register followed by a concatenation of these shift registers, a permutation is performed [4]. 8 times a small permutation followed by shifting in the incoming bytes corresponds to the initial permutation IP.

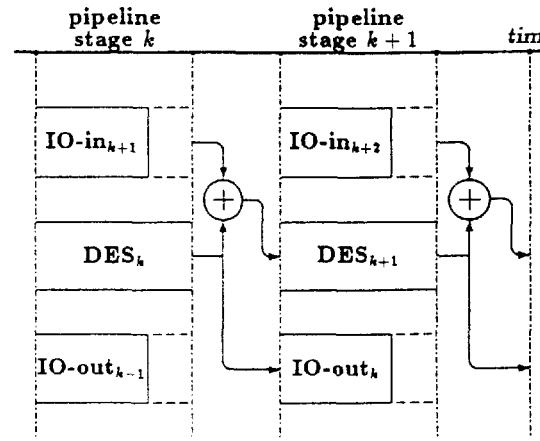
But even these 8 to 8 bit hardware routings, Fig. 5c, could be saved. Instead of placing the 8 shift registers, numbered from A to H, next to each other on the floorplan and routing the input and output busses according to the small permutations, we have reordened the shiftregisters as in Fig. 6. The internal transport from the Input–Output register to the left and the right register of the



a. DES in CBC mode



b. Hardware flow of the data



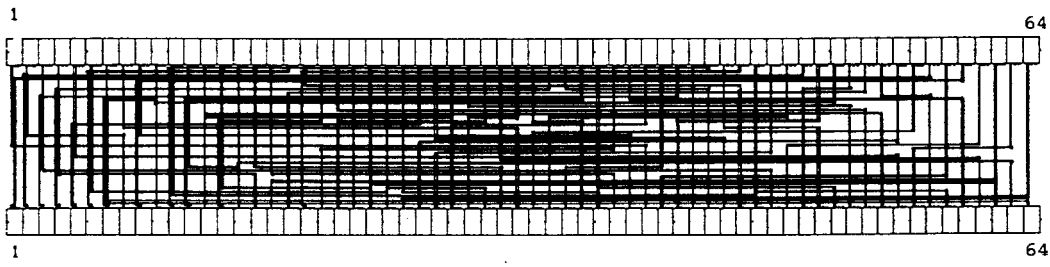
c. Evolution in time : fast internal transport between every two consecutive pipelinestages.

Figure 4: The Mode calculation on chip, e.g. CBC.

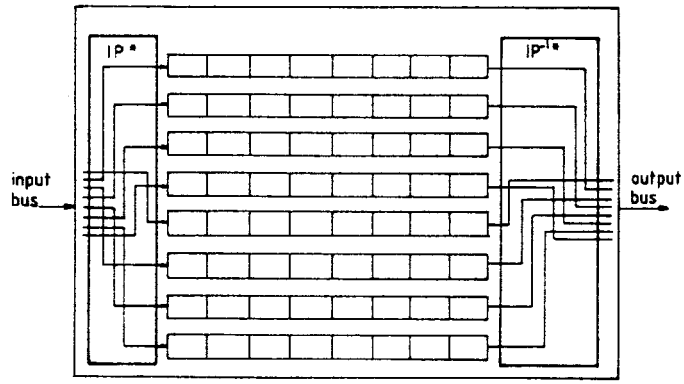
DES part is also done byte oriented by shifting. The left register corresponds to the concatenation of *A, B, C, D* and the right register to *E, F, G, H.* Once the data is entered in the left and the right register, these registers are switched to full parallel operation to calculate the 16 DES rounds. As can be seen in Fig. 6 *no* scrambling routing remained. Shifting the result of the left and right register back into the Input Output register, after the DES round calculations, and reading it out byte per byte, corresponds to IP-1.

4.3 Equivalent PC1 realization

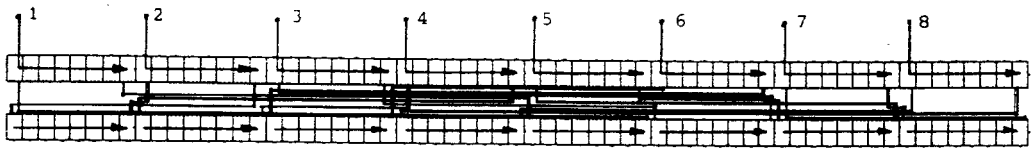
A similar shift technique can be applied to realize the initial key permutation PC1. Instead of 8 shift registers only 7 are used. The corresponding floorplan is shown in Fig. 7. There is *no* routing of the incoming bits, the only irregularity is one connection in shiftregister D, over the central mirror line. On the left part the bits are shifted from left to right and on the right side form right to left. The 7 shift registers are combined to two 28 bit registers, registers C and D, for the key schedule calculation. The mirroring of the right part, corresponding to register D, is included into the key permutation PC2. In PC2 no regularity could be found. By mirroring the right part, this hardware routing did not grow nor shrink. Again, comparable to the IP realization, it is the *switching from serial shift to parallel use* that realizes the permutation.



a. Straightforward hardware routing for IP which is NOT implemented.



b. Equivalent realization of IP, with two 8 bit routings and 8 shift registers.



c. Hardware routing for the two 8 bit routings, equivalent to IP and IP-1.

Figure 5: Equivalent IP realization with shift technique.

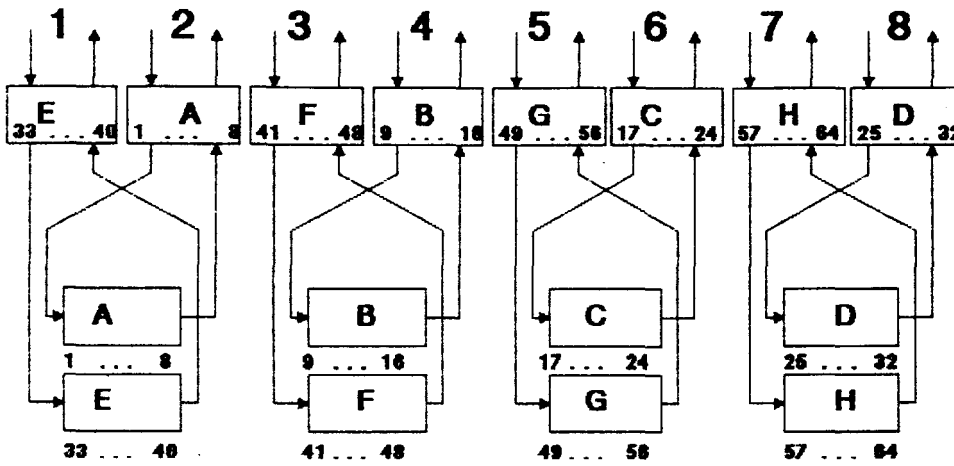


Figure 6: Equivalent realization of IP, floorplan organisation.

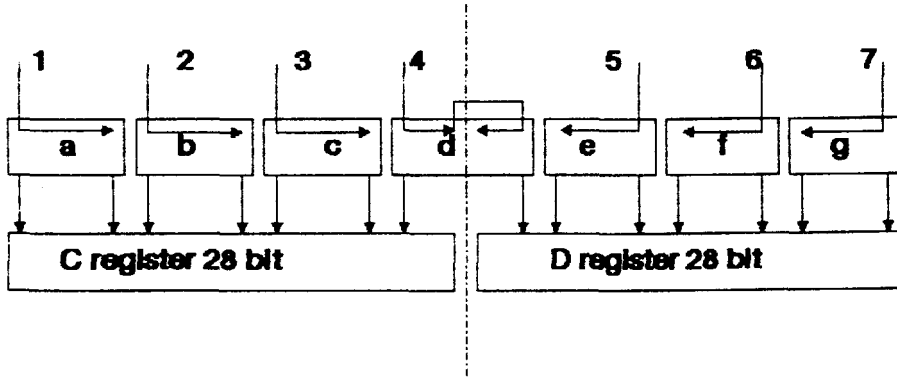


Figure 7: Equivalent realization of PC1, floorplan organisation.

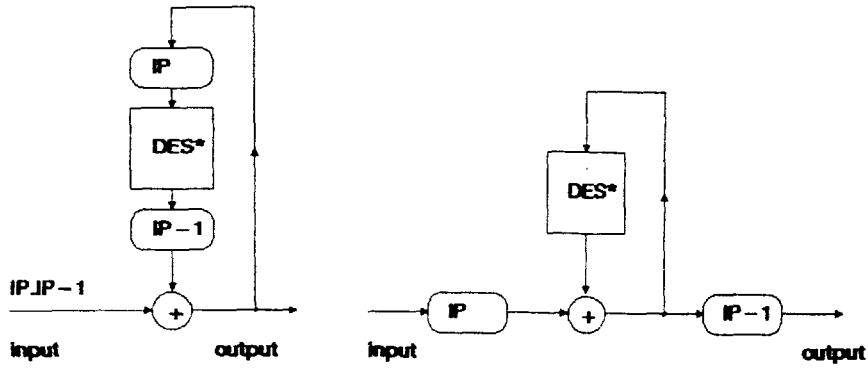


Figure 8: IP and IP-1 outside the feedback loops of the modes.

4.4 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. e.g. for the 8 byte Cipher Feedback mode (CFB) this is explained in Fig. 8. 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-1(IP(Data)) = Data. For *64 bit feedback modes*, or *block modes*, this is very easy to see, while mathematically the following statement is true : "The permutation of two exored data words equals the exoring of the two permuted words", [4].

The feedback modes are however defined feeding K bits back, K going from 1 to 64, [2]. One can prove that bringing the permutations outside the feedback loops is mathematically allowed, [4]. But the interesting simplification follows out of these equivalences for *1 byte modes*, ($K = 8$). This results from the fact that 'by coincidence' IP and IP-1 can be realized byte oriented, applying 8 times the same small permutation. Therefore only 1 byte feedback modes are implemented instead of arbitrary K. Practically however, 1 byte and 8 bytes feedback are mostly used for high speed applications.

4.5 The combination of these equivalences

The combination of both techniques, the equivalent shift technique and bringing IP and IP-1 outside the feedback loops, allow a very *compact mode realization.* First the hardware routing for three permutations is saved. This would correspond to an increase of 30% of the actual datapath without counting the placement problems. Second by shifting from the IO register to the internal register, only 8 exors, 8 multiplexers and 8 bit buses are needed instead of area consuming 64 exors, 64 multiplexers and 64 bit buses.

5 Testability of a cryptographical device

On the floorplan of the first prototype, Fig. 2 one can see that also some *scan registers* are implemented. These are necessary during the development of the chip, while at this stage testability means *controllability* and *observability*, [5]. Controllability implies that one must be able to reach every part on chip. On the other hand if one is able to watch the reaction of every part of the chip then it is observable. This is done with special scan registers which can isolate parts of the chip, activate and watch by scanning out the observed data. This scan technique is very useful for *localising* faults in a design during development.

But it is in conflict with security demands. Scan pads provide a very easy way to scan out keys, initialisation vectors or intermediate ciphertext. Therefore they are not allowed for safe commercial applications. They are only provided during development and they are realized as independent blocks (modularity !) instead of combining them with existing registers. It means overhead on area, but it is very easy to cut it out.

Scan pads are cut out for safe commercial applications. But for validation or maintenance purposes, the chip must still be testable, [6,7]. These tests are based on *signature*. The test signature uses the same propagation principle as the digital signature like in the generation of MAC's. For a Message Authentication Code, which is based on the Cipher Block Chaining Mode, it is the purpose that an unallowed change or an insertion of false data is detected. A small change of the data will completely change the signature. The same propagation principle is used for test signature. A known pair of clear data and key is applied, and the result is compared with a reference cipher text. A small hardware error will result in a completely different test signature. *While in other chip designs, an overhead on logic and chip area is necessary to realize fault propagation, in DES it is inherently present.*

6 Implementation and test results

The first prototype, with scan registers, has been processed in a 3 μm Nwell CMOS process with double metal, as shown on the photograph on Fig. 3. It contains 12.000 transistors and has an area of 25 mm^2. The aim was to test the functional working of the different parts, mainly the DES and the Modes part. Tests have shown full functionality up to 16,7 Mhz clock rate. With this clock frequency, a 20 Mbit/sec datarate on all 8 bytes mode is achieved.

A second implementation is now being developped, which contains also a full controller and microprocessor interface. The scan registers are cut out. Thanks to a highly performant controller, which keeps the DES part calculating, one expects a datarate of 32 Mbit/sec if the same clockfrequency can be applied to this device.

7 Conclusions

A single chip has been designed and implemented which executes DES with a number of unique cryptographical features : all modes, safety, key management, speed, triple encryption, MAC's etc. Due to the modularity the main architecture and many building blocks can be reused in a flexible way for safe commercial applications or for DES like algorithms. The original approach of modes calculation allows pipelining in combination with feedback modes. This implies high speed as well as high security. 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.

References

[1] "Data Encryption Standard," FIPS, Federal Information Processing Standard, Pub no.46, National Bureau of Standards, January 1977

[2] "DES modes of operation," FIPS, Federal Information Processing Standard, Pub no.81, National Bureau of Standards, December 1980

[3] "Financial Institution Keymanagement." Draft American National Standard, Document N216, April 1984.

[4] M. Davio, Y. Desmedt, J. Goubert, F. Hoornaert, and J.-J. Quisquater, "Efficient hardware and software implementations of the DES," Advances in Cryptology Proc. Crypto 84, August 84.

[5] T.W. Williams and K.P. Parker, "Design for Testability - A Survey," IEEE Transactions on Computers, Vol. C-31 No. 1. January 1982.

[6] J. Gait, Computer Science and Technology, " Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard," Special Publication 500-20, U.S. Department of Commerce, National Bureau of Standards

[7] J. Gait, Computer Science and Technology, "Maintenance Testing for the Data Encryption Standard," Special Publication 500-61, U.S. Department of Commerce, National Bureau of Standards

[8] R. C. Fairfield, A. Matusevich, J. Plany, " An LSI Digital Encryption Processor (DEP)." IEEE, Communications Magazine, Vol. 23, No. 7, July 1985.