

# A High-speed DES Implementation for Network Applications

Hans Eberle

Digital Equipment Corporation, Systems Research Center,  
130 Lytton Ave, Palo Alto CA 94301, USA

**Abstract.** A high-speed data encryption chip implementing the Data Encryption Standard (DES) has been developed. The DES modes of operation supported are Electronic Code Book and Cipher Block Chaining. The chip is based on a gallium arsenide (GaAs) gate array containing 50K transistors. At a clock frequency of 250 MHz, data can be encrypted or decrypted at a rate of 1 GBit/second, making this the fastest single-chip implementation reported to date. High performance and high density have been achieved by using custom-designed circuits to implement the core of the DES algorithm. These circuits employ precharged logic, a methodology novel to the design of GaAs devices. A pipelined flow-through architecture and an efficient key exchange mechanism make this chip suitable for low-latency network controllers.

## 1 Introduction

Networking and secure distributed systems are major research areas at the Digital Equipment Corporation's Systems Research Center. A prototype network called Autonet with 100 MBit/s links has been in service there since early 1990 [14]. We are currently working on a follow-on network with link data rates of 1 GBit/s.

The work described here was motivated by the need for data encryption hardware for this new high-speed network. Secure transmission over a network requires encryption hardware that operates at link speed. Encryption will become an integral part of future high-speed networks.

We have chosen the Data Encryption Standard (DES) since it is widely used in commercial applications and allows for efficient hardware implementations. Several single-chip implementations of the DES algorithm exist or have been announced. Commercial products include the AmZ8068/Am9518 [1] with an encryption rate of 14 MBit/s and the recently announced VM007 with a throughput of 192 MBit/s [18].

An encryption rate of 1 GBit/s can be achieved by using a fast VLSI technology. Possible candidates are GaAs direct-coupled field-effect transistor logic (DCFL) and silicon emitter-coupled logic (ECL). As a semiconductor material GaAs is attractive because of the high electron mobility which makes GaAs circuits twice as fast as silicon circuits. In addition, electrons reach maximum velocity in GaAs at a lower voltage than in silicon, allowing for lower internal operating voltages, which decreases power consumption. These properties position

GaAs favorably with respect to silicon in particular for high speed applications. The disadvantage of GaAs technology is its immaturity compared with silicon technology. GaAs has been recognized as a possible alternative to silicon for over twenty years, but only recently have the difficulties with manufacturing been overcome. GaAs is becoming a viable contender for VLSI designs [8, 10] and motivated us to explore the feasibility of GaAs for our design.

In this paper, we will describe a new implementation of the DES algorithm with a GaAs gate array. We will show how high performance can be obtained even with the limited flexibility of a semi-custom design. Our approach was to use custom-designed circuits to implement the core of the DES algorithm and an unconventional chip layout that optimizes the data paths. Further, we will describe how encryption can be incorporated into network controllers without compromising network throughput or latency. We will show that low latency can be achieved with a fully pipelined DES chip architecture and hardware support for a key exchange mechanism that allows for selecting the key on the fly.

Section 2 of this paper outlines the DES algorithm. Section 3 describes the GaAs gate array that we used for implementing the DES algorithm. Section 4 provides a detailed description of our DES implementation. Section 5 shows how the chip can be used for network applications and the features that make it suitable for building low-latency network controllers. This section also includes a short analysis of the economics of breaking DES enciphered data. Finally, section 6 contains some concluding remarks.

## 2 DES Algorithm

The DES algorithm was issued by the National Bureau of Standards (NBS) in 1977. A detailed description of the algorithm can be found in [11, 13]. The DES algorithm enciphers 64-bit data blocks using a 56-bit secret key (not including parity bits which are part of the 64-bit key block). The algorithm employs three different types of operations: permutations, rotations, and substitutions. The exact choices for these transformations, i.e. the permutation and substitution tables are not important to this paper. They are described in [11]. As shown in Fig. 1, a block to be enciphered is first subjected to an initial permutation (IP), then to 16 iterations, or rounds, of a complex key-dependent computation, and finally to the inverse initial permutation ( $IP^{-1}$ ). The key schedule transforms the 56-bit key into sixteen 48-bit partial keys by using each of the key bits several times.

Figure 2(a) shows an expanded version of the 16 DES iterations for encryption. The inputs to the 16 rounds are the output of IP and sixteen 48-bit keys  $K_{1..16}$  that are derived from the supplied 56-bit key. First, the 64-bit output data block of IP is divided into two halves  $L_0$  and  $R_0$  each consisting of 32 bits. The outputs  $L_n$  and  $R_n$  of an iteration are defined by:

$$\begin{aligned} L_n &= R_{n-1} \\ R_n &= L_{n-1} \text{ XOR } f(R_{n-1}, K_n) \end{aligned}$$

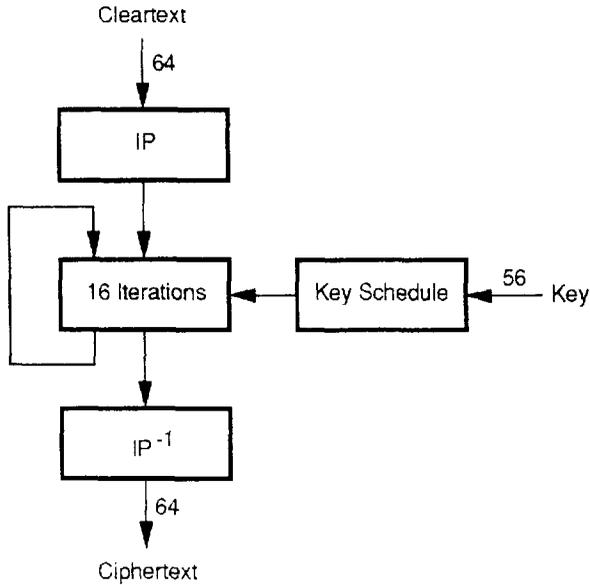


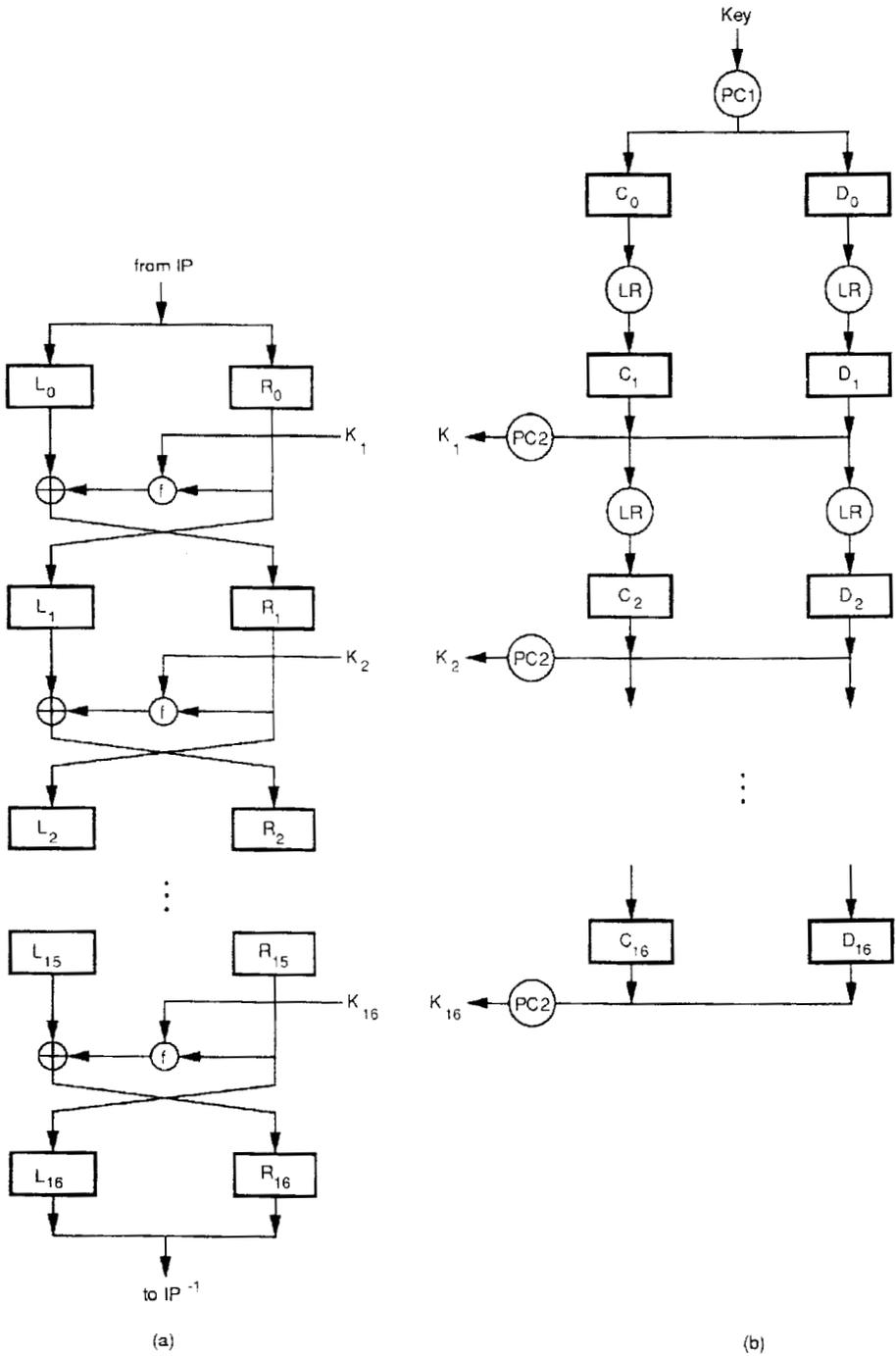
Fig. 1. Overview of the Data Encryption Standard

where  $n$  is in the range from 1 to 16. At the completion of the 16 iterations the two 32-bit words  $L_{16}$  and  $R_{16}$  are put together into a 64-bit block and used as the input to  $IP^{-1}$ .

Figure 2(b) represents the key scheduling algorithm for encryption. The 56-bit key first undergoes permuted choice 1 (PC1). The resulting 56 bits are divided into two 28-bit entities  $C_0$  and  $D_0$ . The outputs of an iteration  $C_n$  and  $D_n$  are obtained by rotating  $C_{n-1}$  and  $D_{n-1}$  by one or two positions to the left, where  $n$  is in the range from 1 to 16. The number of left shifts at each iteration is a fixed part of the algorithm. After 16 rounds the two halves of the 56-bit key will have been shifted by 28 positions, i.e.  $C_{16}$  equals  $C_0$  and  $D_{16}$  equals  $D_0$ . The key value  $K_n$  is obtained from  $C_n$  and  $D_n$  by choosing 48 bits of the available 56 bits according to permuted choice 2 (PC2).

Decryption and encryption use the same data path, and differ only in the order in which the key bits are presented to function  $f$ . That is, for decryption  $K_{16}$  is used in the first iteration,  $K_{15}$  in the second, and so on, with  $K_1$  used in the 16th iteration. The order is reversed simply by changing the direction of the rotate operation performed on  $C_{0..15}$  and  $D_{0..15}$ , that is,  $C_{0..15}$  and  $D_{0..15}$  are rotated to the left during encryption and rotated to the right during decryption.

Figure 3 describes the calculation of function  $f$ . First, the 32 bits of the right half  $R$  are permuted and expanded to 48 bits by the  $E$  bit-selection table ( $E$ ). The expansion is achieved by repeating certain bits. The 48-bit result is then XORed with a 48-bit key value  $K$  obtained from the key schedule. Next, the 48-bit output of the XOR operation is split into blocks of 6 bits and delivered to eight substitution boxes  $S_{1..8}$ . Each  $S$  box implements a different nonlinear



**Fig. 2.** Expanded Version of the 16 Iterations (a) and the Key Schedule (b) for Encryption

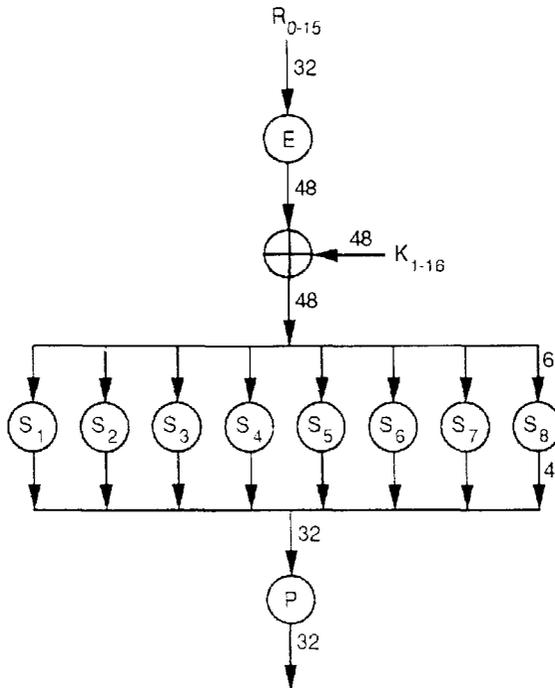


Fig. 3. Expanded Version of Function  $f$

function yielding a 4-bit output block. Finally, the 32 bits produced by the S boxes undergo one more permutation function (P).

For enciphering data streams that are longer than 64 bits the obvious solution is to cut the stream into 64-bit blocks and encipher each of them independently. This method is known as Electronic Code Book (ECB) mode [12]. Since for a given key and a given plaintext block the resulting ciphertext block will always be the same, frequency analysis could be used to retrieve the original data. There exist alternatives to the ECB mode that use the concept of diffusion so that each ciphertext block depends on all previous plaintext blocks. These modes are called Cipher Block Chaining (CBC) mode, Cipher Feedback (CFB) mode, and Output Feedback (OFB) mode [12].

Our implementation complies with the NBS DES and supports ECB mode and CBC mode. We did not implement CFB and OFB modes because they are less useful in network applications. Figure 4 illustrates CBC mode. The plaintext  $p$  is split into 64-bit blocks  $p = p_1 p_2 \dots p_n$ . The ciphertext block  $c_i$  is computed as:

$$c_i = DES_k(p_i \text{ XOR } c_{i-1}) .$$

The resulting ciphertext is  $c = c_1 c_2 \dots c_n$ . Knowing key  $k$  and  $c_0$ , which is also known as the initialization vector, the ciphertext can be deciphered by computing the plaintext block  $p_i$  as:

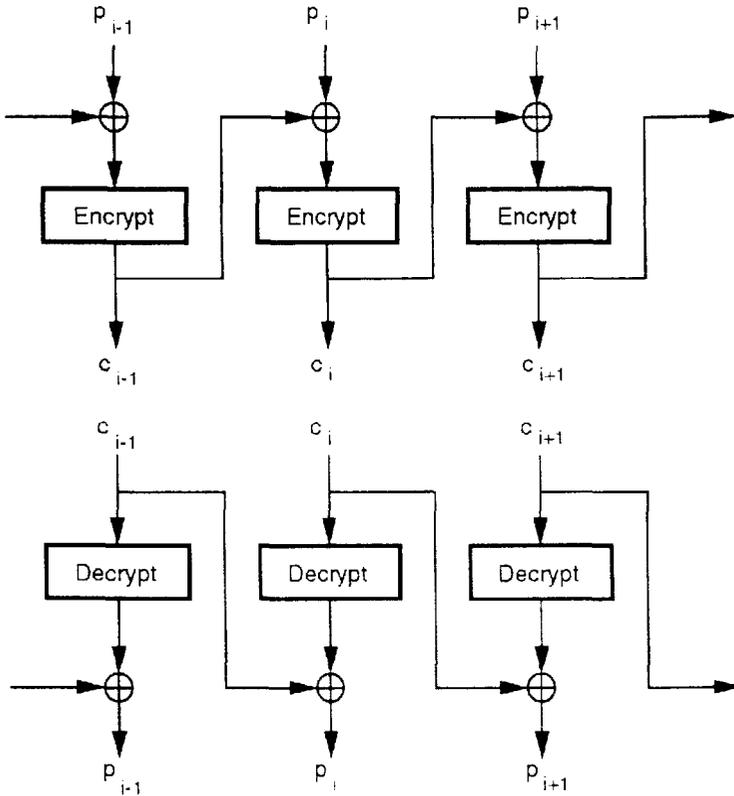


Fig. 4. Cipher Block Chaining

$$p_i = DES_k^{-1}(c_i) \text{ XOR } c_{i-1} .$$

### 3 GaAs Gate Array

The DES chip is based on a FURY VSC15K gate array from Vitesse Semiconductor [16]. It uses a  $0.8 \mu\text{m}$  GaAs enhancement/depletion mode metal-semiconductor field-effect transistor (E/D-mode MESFET) process [17]. The array contains 50K transistors on a 8.1 mm by 7.1 mm die and can implement up to 15K unbuffered DCFL 2-input NOR gates. Of more interest to real applications, the array has the capacity for 4,000 buffered 2-input NOR gates or 1,500 D-flipflops.

Compared with silicon technologies, GaAs DCFL offers higher density than silicon ECL, which is the highest-performance bipolar silicon technology, but cannot yet compete with silicon CMOS, the densest silicon technology. Presently, the densest cell-based GaAs gate arrays offer up to 200K raw gates, while CMOS arrays can integrate up to 800K raw gates. It is worth noting that the density

of GaAs DCFL is currently increasing more rapidly than the density of silicon CMOS. GaAs competes favorably with ECL in that it offers comparable speed, but consumes only about half to a third of the power. It remains to be seen how well GaAs competes with CMOS. Compared with CMOS, GaAs is faster by a factor of two to three at the gate level while power consumption favors GaAs only at clock frequencies higher than 100 MHz.

## 4 DES Chip Implementation

This section describes how we implemented the DES algorithm.

### 4.1 Organization

There are two ways to improve an algorithm's performance. One can choose a dense but slow technology such as silicon CMOS and increase performance by parallelizing the algorithm or flattening the logic. Alternatively, one can choose a fast but low-density technology such as silicon ECL or GaAs DCFL. The DES algorithm imposes limits on the former approach. As was shown in Fig. 4, the CBC mode of operation combines the result obtained by encrypting a block with the next input block. Since the result has to be available before the next block can be processed, it is impossible to parallelize the algorithm and operate on more than one block at a time. It is, however, possible to unroll the 16 rounds of Fig. 1 and implement all 16 iterations in sequence. Flattening the design in this manner will save the time needed to latch the intermediate results in a register on every iteration. Even though the density of CMOS chips is sufficient for doing this, the speed requirements of a 1 GBit/s CMOS implementation might still be challenging.

Since we wanted to use GaAs technology, we had to choose a different approach. The limited density of GaAs gate arrays forced us to implement only one of the 16 rounds and reuse it for all 16 iterations. Even without unrolling the 16 rounds, fitting the implementation into the available space and meeting the speed requirements was a major challenge. In order to achieve a data rate of 1 GBit/s, each block has to be processed in 64 ns, which corresponds to 4 ns per iteration or a clock rate of 250 MHz.

The register-level block diagrams for encryption and decryption are shown in Figures 5 and 6. The DES chip realizes a rigid 3-stage pipeline, that is, a block is first written into the input register I, is then moved into register LR, where it undergoes the 16 iterations of the cipher function  $f$ , and finally is written into the output register O.

The key schedule is formed by the master key register MK, which holds the encryption or decryption key, and a shift register CD, which supplies a different key value for each of the 16 iterations. Registers MK and CD can be written but not read by external circuitry. This is important since the security of a secret key system depends on the security of the keys. If the keys are compromised,



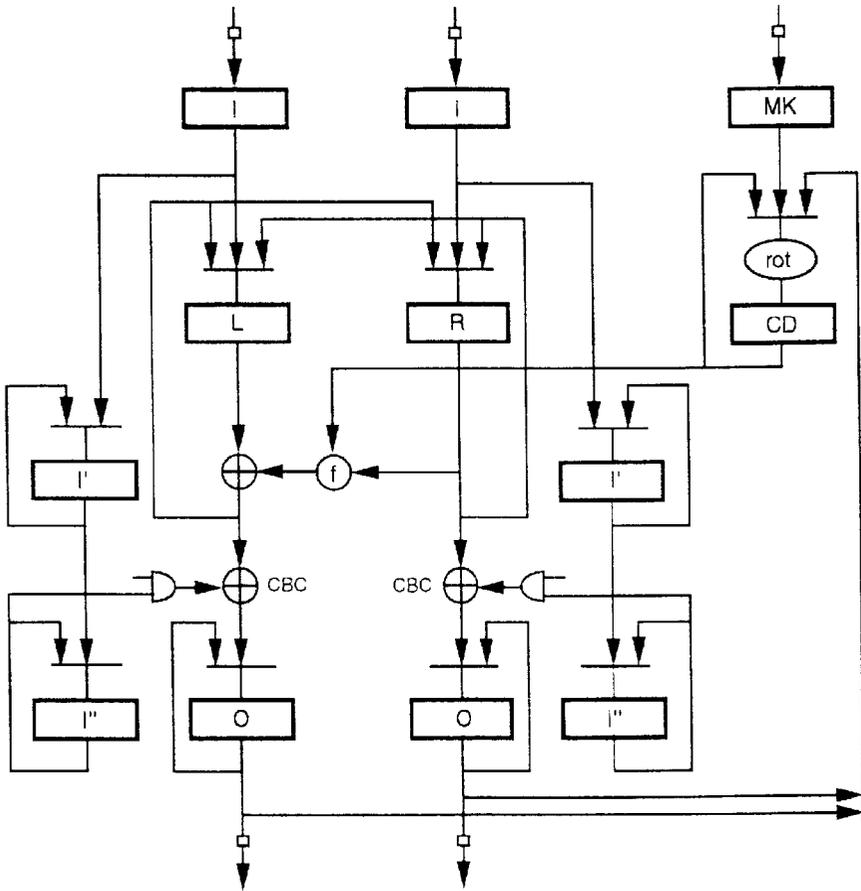


Fig. 6. Decryption

## 4.2 Implementation Characteristics

The implementation of the DES chip contains 480 flipflops, 2580 gates, and 8 PLAs. There are up to ten logic levels that have to be passed during the 4 ns clock period. The chip uses 84% of the transistors available in the VSC15K gate array. The high utilization is the result of a fully manual placement. Timing constraints further forced us to lay out signal wires partially by hand.

The chip's interface is completely asynchronous. The data ports are 8, 16, or 32 bits wide. A separate 7-bit wide port is available for loading the master key. Of the 211 available pins, 144 are used for signals and 45 are used for power and ground. With the exception of the 250 MHz clock, which is ECL compatible, all input and output signals are TTL compatible.

The chip requires power supply voltages of -2 V for the GaAs logic and 5 V for the TTL-compatible output drivers. The maximum power consumption is 8 W.

### 4.3 Asynchronous Interface

Asynchronous ports are provided in order to avoid synchronization with the 250 MHz clock. The data input and output registers are controlled by two-way handshake signals which determine when the registers can be written or read. The data ports are 8, 16, or 32 bits wide. The variable width allows for reducing the width of the external data path at lower operating speeds. With the 32-bit wide port, a new data word must be loaded every 32 ns in order to achieve an encryption rate of 1 GBit/s. The master key register is loaded through a separate, also fully asynchronous 7-bit wide port. Our implementation does not check the byte parity bits included in the 64-bit key. The low speed of the data and key ports makes it possible to use TTL-levels for all signals except for the 250 MHz clock which is a differential ECL-compatible signal.

Thanks to the fully asynchronous chip interface, the chip manufacturer was able to do at-speed testing even without being able to supply test vectors at full speed. For this purpose, the 250 MHz clock was generated by a separate generator, while the test vectors were supplied asynchronously by a tester running at only 40 MHz. At-speed testing was essential particularly in testing the precharged logic which will be described in the following section.

Due to the high chip utilization there was no room for test structures like scan-paths [9]. A special test mode, however, allows for single-stepping through the iterations of the cipher function and reading out intermediate results and the state of the control logic after each DES round. Combined with the possibility of at-speed testing this technique can provide valuable information about the chip internals.

### 4.4 Precharged S box

The core of the DES algorithm consists of eight substitution boxes (S boxes) which are part of the cipher function  $f$  in Fig. 2(a). Each S box computes a different boolean function with 6 inputs and 4 outputs. The most challenging and interesting part of the DES chip is to design and implement S boxes that are both fast and space-efficient.

The obvious implementation structure for the S boxes is a programmable logic array (PLA). In order to meet space and timing constraints, a precharged design using custom macros was chosen.

Precharging is a well-known design technique for silicon nMOS [5]. It offers the density of unbuffered gates and the speed of buffered gates. For FURY gate arrays, the difference in cell count between buffered logic versus unbuffered logic typically is a factor of four. The goal of precharged logic is to overcome the slow rise time of unbuffered gates that must drive large capacitive loads. The rise time of an unbuffered gate can be as much as ten times the fall time when driving a significant amount of metal because of the weak pullup transistors used in DCFL.

Figure 7(a) shows the basic building block of precharged NOR-NOR logic. The first-level gates have an extra input for the precharge signal, while the

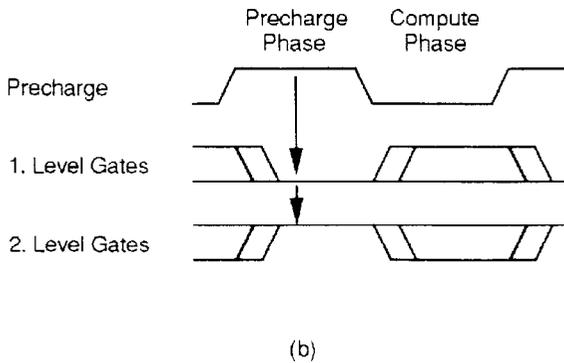
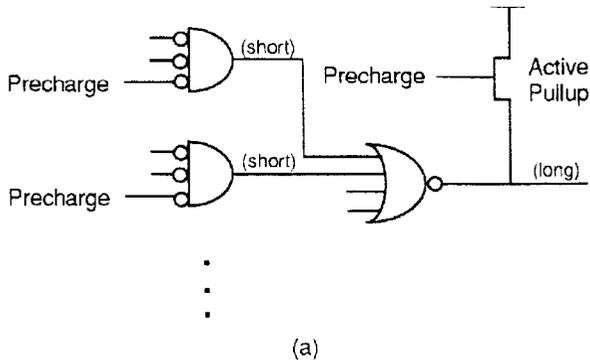


Fig. 7. Precharged NOR-NOR Logic (a) and Timing (b)

second-level gates have an active pullup connected to the output. As shown in Fig. 7(b), precharged logic operates in two phases: a precharge phase and a compute phase. During precharge, when the precharge signal is high, the outputs of the first-level gates are forced to a low level, while the active pullups will force the outputs of the second-level gates to a high level. During the compute phase, when the precharge signal is low, the outputs of the first-level gates stay low or go high while the outputs of the second-level gates stay high or go low. The first-level gates are placed adjacent to the second-level gates to make the rising edges of the first-level gates fast. The second-level gates are equipped with an active pullup to drive large capacitive loads. In a typical application several basic blocks are chained together. Notice that the slow low-to-high transitions for the second-level gates will occur in parallel during the precharge phase. During the compute phase, the long wires of the logic chain propagate only falling edges, which are fast. The penalty of this design technique is the time required for precharging. The precharge phase has to be long enough to charge the worst-case capacitance driven by any second-level gate. Therefore, the more levels of logic, the bigger the gain in performance.

The S box implementation shown in Fig. 8 contains two levels of precharged

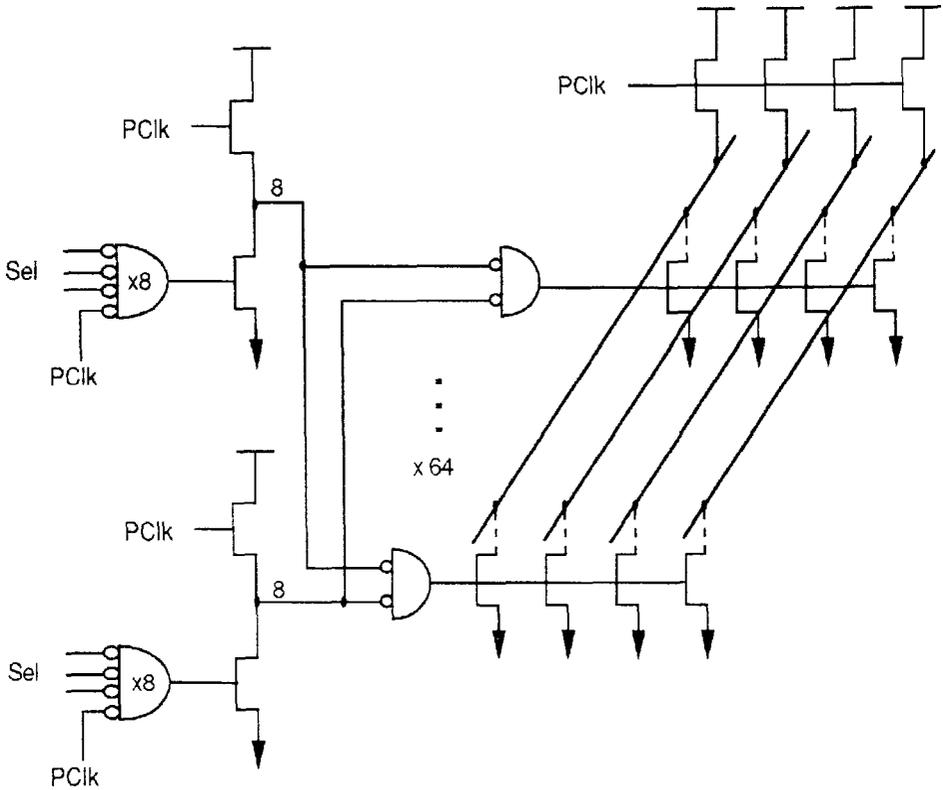


Fig. 8. Precharged S box

NOR-NOR logic: a 4-input NOR gate driving an inverter followed by a 2-input NOR gate driving from zero to four pull-down transistors. The row decoder uses two 3:8 decoders in order to save space. By using precharged logic, the S boxes occupy less than 10% of the die area. If standard macros were chosen, the S box implementation would require 5.5 times as many cells. An implementation with available macros would not have fit into the chosen gate array.

Contrary to the results obtained by analog simulations of the S box, the first implementation exhibited a discharge problem, which caused the chips to fail at high temperature. The discharge problem affected the last stage of the PLA structure in Fig. 8, which corresponds to a 32-bit wide NOR gate. The models of the pull-down transistors provided by the chip manufacturer basically ignored leakage currents. This caused the output of the PLA to drop from a high to a low level before the compute phase was over. Since leakage is proportional to temperature, the discharge problem was even worse at higher temperatures. The problem can be eliminated by lowering the voltage of the low level of the gates driving the 32-input NOR gate and thereby turning off the 32-input NOR gate harder. This requires a major change of the driving circuitry. Due to space constraints, we decided to improve the drop rate by simply changing the precharge

pullup of the 32-input NOR gate. A current source in the form of a D-mode FET was added to the existing active pullup transistor in order to compensate for the leakage current.

#### 4.5 Floorplan

The usual choices when laying out a pipelined design are to partition the logic either into register slices or bit slices. The various permutations of the data paths contained in the DES algorithm complicate this task. The permutation tables employed by the DES algorithm are the so-called initial permutation (IP), the E bit-selection table (E), the permutation function (P), and a pair of permuted-choice tables (PC1, PC2). Some of the tables not only permute the input bits but also duplicate or omit input bits and, thereby, expand or shrink the input string. The wiring of the data paths, however, is not as badly scrambled as one might fear. IP,  $IP^{-1}$ , and PC1 affect the wiring of the input and output pads only, not the wiring of the critical path, the iteration feedback loop. Fig. 9 shows one DES iteration. The wires belonging to the critical path are highlighted. This feedback loop contains two permuted data paths: permutations E and P.

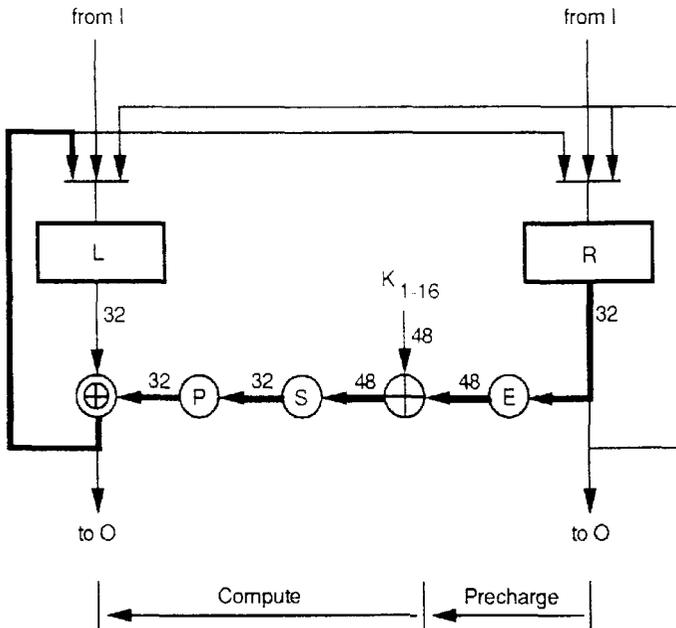


Fig. 9. One DES Iteration

While previous implementations have chosen a register-sliced layout [7, 15], we preferred a mixed strategy. As shown in Fig. 10, we first divide the design into

blocks corresponding to the eight S boxes. We further subdivide each block into four bit slices each containing one bit of the left and the right half of registers I, I', I'', LR, and O. The register bits are laid out so that the wires connecting the outputs of the S boxes and the inputs of LR are as short as possible. Referring to Fig. 9, the only scrambled data path is permutation E which connects the outputs of R with the inputs of the XOR gate. These wires potentially have to go all the way across the chip. In our implementation, the longest of these wires is 6 mm long. The time to drive these wires is significant. However, driving these long wires happens at the beginning of a clock cycle and, therefore, coincides with the precharge phase. Thus, there is no data path with long wires that would contribute to the cycle time of the critical path.

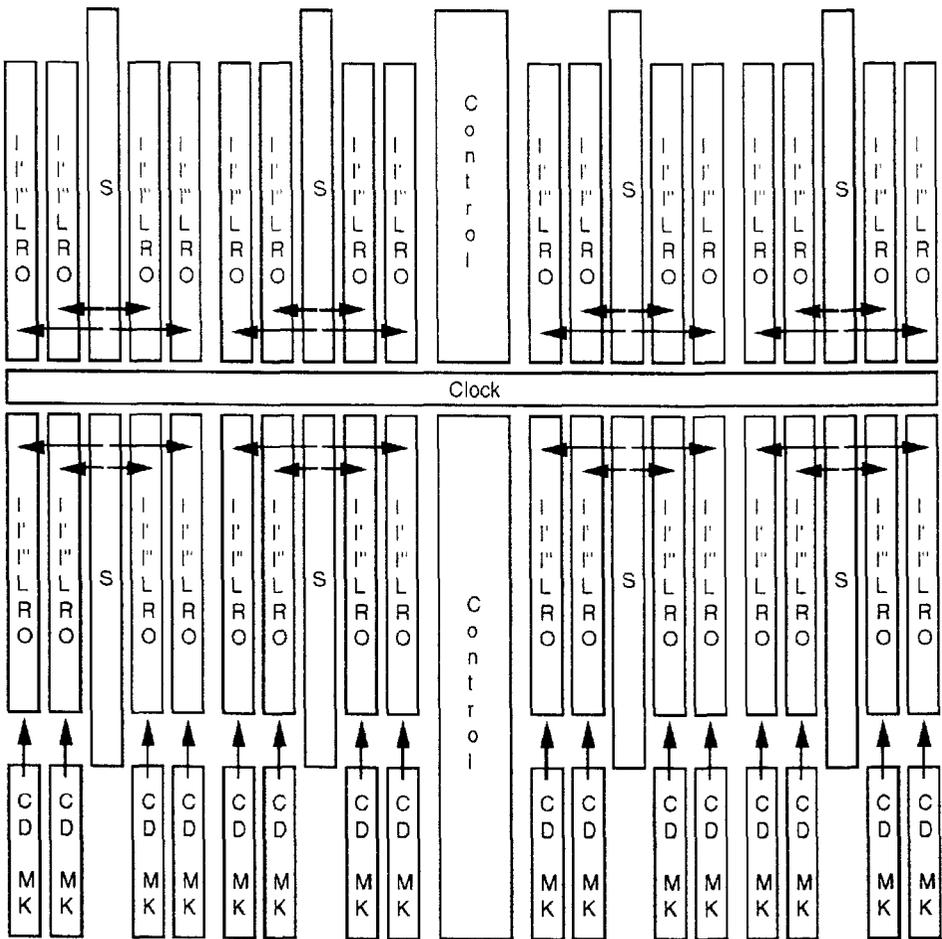


Fig. 10. DES Chip Floorplan

The key bits of register CD are laid out so that the wires connecting CD and the XOR gates are kept as short as possible. This scrambles the wiring of the key schedule (which implements two 28-bit wide registers that can be rotated by one or two bits either to the right or to the left). The timing of these wires is, however, not critical since the only logic this path contains is a multiplexer that implements the rotate function.

The control signals are generated in the middle columns of the chip. Drivers are duplicated; that is, there are separate drivers for each side of the chip in order to reduce the load and wire length and with it the propagation delay.

## 5 Applications

We now discuss applications of the DES chip, which is intended primarily for use in network controllers.

### 5.1 Low-latency Network Controller

Our implementation of the DES algorithm is tailored for high-speed network applications. This requires not only encryption hardware operating at link speed but also support for low-latency controllers. Operating at link data rates of 1 GBit/s requires a completely pipelined controller structure. Low latency can be achieved by buffering data in the controller as little as possible and by avoiding protocol processing in the controller. In this respect, the main features of the DES chip are a pipelined flow-through design and an efficient key exchange mechanism.

As described in the previous section, the chip is implemented as a rigid 3-stage pipeline with separate input and output ports. Each 64-bit data block is entered into the pipeline together with a command word. While the data block flows through the pipeline, the accompanying command instructs the pipeline stages which operations to apply to the data block. On a block-by-block basis it is possible to enable or disable encryption, to choose ECB or CBC mode, and to select the master key in MK or the key in CD. None of these commands causes the pipeline to stall. It is further possible to instruct the pipeline to load a block from the output register O into register CD. Typical usage of this feature is as follows: a data block is decrypted with the master key, is loaded into CD, and is then used for encrypting or decrypting subsequent data blocks. This operation requires a one-cycle delay slot; that is, the new key in CD cannot be applied to the data block immediately following.

The format of packets transmitted over the Autonet network efficiently uses the described architecture allowing for very low-latency controllers. The data flow of a packet transmission is as follows. With the help of a public key algorithm, a sender S and receiver R first exchange a key K that will subsequently be used for encrypting packets. Sender and receiver encrypt this key under their master keys and exchange the resulting values. Both store copies of  $[K]_{MKS}$  and  $[K]_{MKR}$  in their memories. MKS is the master key of S and MKR the master

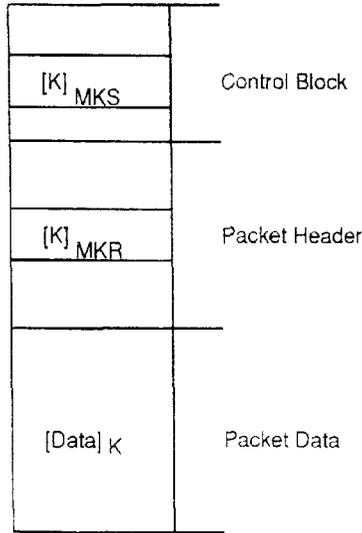


Fig. 11. Packet Format

key of R. Note that a plaintext version of  $K$  is not stored in either memory. The transmission of the actual data can now begin. The data flow through the sender's and receiver's DES chips is as follows.

Figure 11 shows the data that flows through the DES chip in the sender. First, a control block containing the key needed for encrypting the data part of the packet will be read from host memory and be presented to the sender's DES chip. The DES chip will decrypt  $[K]_{MKS}$  and load the resulting key value  $K$  into key register CD. The control block will not be sent to the network since it contains only information required by the sender. Next, the header of the packet containing  $[K]_{MKR}$  will pass through the DES chip without being manipulated, followed by the data, for which encryption and CBC mode are enabled. Both header and encrypted data will be sent over the network to the receiver.

When the header of the packet flows through the receiver's DES chip,  $[K]_{MKR}$  will be picked out of the header, decrypted, and loaded into register CD. When the data part begins, decryption and CBC mode will be enabled. Note that in order to obtain key  $K$ , the receiver did not have to access memory or halt the DES pipeline.

## 5.2 Breaking DES

In 1979, Hellman published a paper with the title 'DES will be totally insecure within ten years' [6]. The controversy comes from the rather short length of the DES key, which could make an exhaustive search of the key space feasible [3, 4].

In 1977, Diffie and Hellman proposed a machine consisting of 1 million processors that would each be able to try 1 million keys per second. At an estimated

cost of \$20M this machine would exhaust the key space in 20 hours [4]. In 1984, Hoornaert, Goubert, and Desmedt proposed a machine consisting of 25,000 devices that would each be able to try 1.13 million keys per second. At an estimated cost of \$1M this machine would exhaust the key space in about 4 weeks [7].

This section compares the length of time taken by our implementation to break DES with the time taken by two other popular implementations [1, 18]. We assume a known-plaintext cryptanalytic attack as described in [4]. The search starts out with one or several corresponding plaintext-ciphertext blocks, all encrypted under the same key. The attack is based on brute force in that key after key of the key space, which contains  $2^{56} = 7.2 \times 10^{17}$  elements, is tried. Once the key is broken, messages can be forged or cryptograms for which the plaintext is not known can be read.

The data given in Table 1 illustrates the economics of breaking DES. As expected, the cost per GBit/s of decryption bandwidth and the time required for doing an exhaustive search drop with more recent implementations. The given duration for doing an exhaustive search assumes that one is willing to spend \$1M on DES chips alone. The necessary support circuitry might easily double that figure. The given cost per chip assumes quantities of thousands.

Part	Year	Technology	Data Rate	Cost/Chip	Cost/GBit/s	Exh. Search
Am9518	84	Silicon nMOS	14 MBit/s	\$19	\$1357	72 days
VM007	92	Silicon CMOS	192 MBit/s	\$170	\$885	47 days
GaAs DES	92	GaAs DCFL	1 GBit/s	\$300	\$300	16 days

Table 1. Cost of Breaking DES

For our implementation, it takes 16 days to try  $2^{56}$  keys or an average of 8 days to find the key. With the separate key port our chip would be well suited for breaking DES in that the key could be easily changed every decryption cycle without stalling the pipeline. Moreover, the use of field-programmable gate arrays in our network controllers would easily allow for turning a network of controllers into a distributed machine for breaking DES. We believe that the full decryption bandwidth of 1 GBit/s per chip could be achieved without having to modify existing hardware. Therefore, a network of 10,000 machines each containing two DES chips to encrypt data full duplex at 1 GBit/s would exhaust the key space in 2 days and 16 hours.

Biham and Shamir recently showed that DES can be broken in less than the  $2^{56}$  DES operations required for an exhaustive search [2]. The cryptanalytical attack consists of a data collection phase during which a pool of  $2^{47}$  chosen plaintext blocks are encrypted and a data analysis phase which consists of  $2^{37}$  DES-like operations. The proposed attack will not be further considered here since it cannot make use of existing DES implementations and since the practicability of the data collection phase is questionable.

## 6 Status and Conclusions

We began designing the DES chip in early 1989 and received the first prototypes at the beginning of 1991. The parts were logically functional, but exhibited electrical problems and failed at high temperature. A minor design change fixed this problem. In the fall of 1991, we received 25 fully functional parts that we plan to use in future high-speed network controllers.

With an encryption rate of 1 GBit/s, the design presented in this paper is the fastest DES implementation reported to date. Both ECB and CBC modes of operation are supported at full speed. This data rate is based on a worst case timing analysis and a clock frequency of 250 MHz. The fastest chips we tested run at 350 MHz or 1.4 GBit/s.

We have shown that a high-speed implementation of the DES algorithm is possible even with the limited flexibility of a semi-custom design. An efficient implementation of the S boxes offering both high performance and high density has been achieved with a novel approach to designing PLA structures in GaAs. An unconventional floorplan has been presented that eliminates long wires caused by permuted data bits in the critical path.

The architecture of the DES chip makes it possible to build very low-latency network controllers. A pipelined design together with separate fully asynchronous input and output ports allows for easy integration into controllers with a flow-through architecture. ECL levels are required only for the 250 MHz clock; TTL levels are used for all the data and control pins, thus providing a cost-effective interface even at data rates of 1 GBit/s. The provision of a data path for loading the key from the data stream allows for selecting the encryption or decryption key on the fly. These features make it possible to use encryption hardware for network applications with very little overhead.

## References

1. Advanced Micro Devices: AmZ8068/Am9518 Data Ciphering Processor. Datasheet, July 1984
2. Biham, E., Shamir, A.: Differential Cryptanalysis of the Full 16-round DES. CRYPTO'92, Santa Barbara, August 16-20, 1992
3. Brassard, G.: Modern Cryptology. Lecture Notes in Computer Science, no. 325, Springer-Verlag, 1988
4. Diffie, W., Hellman, M.: Exhaustive cryptanalysis of the NBS Data Encryption Standard. Computer, vol. 10, no. 6, June 1977, pp. 74-84
5. Glasser, L., Dobberpuhl, D.: The Design and Analysis of VLSI Circuits. Addison-Wesley, 1988
6. Hellman, M.: DES will be totally insecure within ten years. IEEE Spectrum, vol. 16, July 1979, pp. 32-39
7. Hoornaert, F., Goubert, J., Desmedt, Y.: Efficient hardware implementation of the DES. Advances in Cryptology: Proceedings of Crypto 84, Springer-Verlag, 1985, pp. 147-173

8. Lee, G., Donckels, B., Grey, A., Deyhimi, I.: A High Density GaAs Gate Array Architecture. CICC 1991: IEEE Custom Integrated Circuits Conference, San Diego, May 13-16, 1991, pp. 14.7.1-14.7.4
9. McCluskey, E.: Logic Design Principles. Prentice-Hall, 1986
10. Milutinovic, V., Fura, D.: Gallium Arsenide Computer Design. Computer Society Press of the IEEE, 1988
11. National Bureau of Standards: Data Encryption Standard. Federal Information Processing Standards Publication FIPS PUB 46-1, January 1988 (supersedes FIPS PUB 46, January 1977)
12. National Bureau of Standards: DES Modes of Operation. Federal Information Processing Standards Publication FIPS PUB 81, December 1980
13. National Bureau of Standards: Guidelines for Implementing and Using the NBS Data Encryption Standard. Federal Information Processing Standards Publication FIPS PUB 74, April 1981
14. Schroeder, M., Birrell, A., Burrows, M., Murray, H., Needham, R., Rodeheffer, T., Satterthwaite, E., Thacker, C.: Autonet: a High-speed, Self-configuring Local Area Network Using Point-to-point Links. Research Report 59, DEC Systems Research Center, Palo Alto, CA, 1990
15. Verbauwhede, I., Hoornaert, F., Vandewalle, J., De Man, H.: Security and Performance Optimization of a New DES Data Encryption Chip. IEEE Journal of Solid-State Circuits, Vol. 23, No. 3, June 1988, pp. 647-656
16. Vitesse Semiconductor Corporation: FURY Series Gate Array Design Manual. Version 3.0, June 1990
17. Vitesse Semiconductor Corporation: GaAs DCFL ASIC Design. Product Data Book: Application Note 7. 1991, pp. 7.30-7.35
18. VLSI Technology: VM007 Data Encryption Processor. Datasheet, October 1991 (Advance Information)