УДК 621.3

MIROSHNIK M.A., Doctor of Technical Sciences,
KOVALENKO M.A., graduate student (Ukrainian State Academy of Railway Transport)

Uses of programmable logic integrated circuits for implementations of data encryption standard and its experimental linear cryptanalysis

This paper presents two original programmable logic integrated circuits (FPGA) implementations of a DES encryption/decryption core. This implementations are the fastest ones known nowadays. In design, the plaintext, the key, and the mode (encryption/decryption) can be changed with no dead cycles. The resulting design is deployed on eight FPGAs and allows us to find (12 + 1) key bits in about 2.3 hours.

Key words: *Cryptography, Data Encryption Standard (DES), linear cryptanalysis, FPGA, efficient implementations.*

1 INTRODUCTION

The rapid growth of secure transmission is a critical point nowadays. We have to exchange data securely at very high data rates. Efficient solutions have to be hardware implemented and flexible in order to evolve with the permanent changes in norms. FPGA (Field Programmable Gate Array) implementations of the triple-Data Encryption Standard (triple-DES) efficiently meet these constraints. Triple-DES is based on three consecutive DES (without intermediate IP and IP-1 permutations). DES is very well suited for FPGA solutions.

Some high-speed DES hardware implementations have been published in the literature. These designs unroll the 16 DES rounds and pipeline them. Patterson made a key-dependent data path for encryption in an FPGA which produces a bitstream of about 12 Gbps. Nevertheless, the latency to change keys is tenth of milliseconds. A DES implementation is also downloadable from FreelP [12] and encrypts at 3.05 Gbps. Last known implementations were announced by Xilinx company, including FPGA implementations of a complete unrolled and pipelined DES encryptor/decryptor. The 16-stage and 48-stage pipelined ores could achieve data rates of, respectively, 8.4 Gbps and 2 Gbps (these results were obtained with VIRTEX E technology). It also allowed changing the plaintext, the key, and the encryption/decryption mode on a cycle-by-cycle basis [1-3].

In this paper, author proposes new mathematical descriptions to implement and optimize DES in an FPGA. Author obtains two original designs. Both permit different pipeline levels and encrypt with data rates of 14.5 Gbps and 21.3 Gbps with, respectively, 21 and 37 cycles of latency (these results were obtained with VIRTEX II technology).

In the second part, this paper deals with linear cryptanalysis. Linear cryptanalysis is a cryptanalytic technique that takes advantage of possible input-output correlations over a cipher. By evaluating the linear approximation for a sufficient number of plaintext / ciphertext pairs, it is possible to recover some bits of the key faster than an exhaustive search.

In 1993, Matsui [4], [5] proposed a known-plaintext linear attack against a full DES. It typically requires 243 known plaintext/ciphertext pairs to recover (12 + 1) secret key bits.

Recently, Knudsen and Mathiassen proposed three chosen-plaintext attacks [1], the third one becoming the best chosen-plaintext attack against DES. Their first attack, which turns out to be less efficient from a theoretical point of view, gives birth to a very fast hardware implementation [6]. In fact, this attack requires only 212 chosen-plaintext/ciphertext pairs, but recovers only seven key bits. This attack allowed us [6] to recover the full key in less than two hours with eight FPGAs used in parallel. We denote it Knudsen's attack.

Although Matsui's linear cryptanalysis is the best known-plaintext attack known against DES nowadays, this attack still had a "theoretical" flavor, in the sense that very few experimental applications have actually been performed: A single known-plaintext experimentation for a full DES cipher has been performed in [5] and, until recently, remained the only practical test, to our knowledge.

However, recent technological advances have made the required computing power reachable, as is witnessed by a set of 21 experiments for Matsui's approximation [2], [3], using the idle time of 18 Intel Pentium III MMX, capable of performing an attack in 4.32 days.

Based on our fast DES implementation, we propose an FPGA implementation of Matsui's attack. It recovers 12 + 1 key bits in about 2.3 hours working with eight FPGAs.

In terms of computation time, Knudsen's attack is better than Matsui's. Nevertheless, according to the number of plaintext / ciphertext pairs needed and the number of secret key bits found, Matsui's attack gives better results. In addition, Matsui's is a more realistic attack compared to Knudsen's attack due to the known-plaintext context. Our solution is very useful to perform practical tests, allowing a comparison with theoretical estimations. We believe that our implementations are the fastest implementations of Matsui's linear cryptanalysis known so far.

2 THE DES ALGORITHM

In 1977, the Data Encryption Standard (DES) algorithm was adopted as a Federal Information Processing Standard for unclassified government communication. It is still largely in use. DES [10-11] encrypts 64-bit blocks with a 64-bit key, only 56 bits of which are used. The other 8 bits are parity bits for each byte. The algorithm has 16 rounds.

For the enciphering calculation, the plaintext is first permuted by a fixed permutation IP. The result is next split into the 32 left bits and the 32 right bits, respectively, L and R. The R part is expanded to 48 bits with the E box by doubling some R bits. Then, it performs a bitwise modulo 2 sum of the expanded R part and the 48-bit subkey K_i. The result of the XOR function is sent to eight nonlinear S-boxes (S). Each of them has six inputs bits and four outputs. The result is then permuted in the box P. Finally, to obtain the R part of the next round, a new modulo 2 sum is performed between the P output and the R part of previous round (the L part of current round). In the last round, no interchange of the 16-round R and L is performed; the ciphertext is calculated by applying the inverse of the initial permutation IP to the result of the 16th round.

The secret key is expanded by the key schedule. The key schedule calculation is first based on the 56-bit permutation PC-1 whose output is split into 28-bit blocks C and D. Then, C and D are left (or right for decryption) shifted once or twice, depending on the index of the round (for decryption, no right shift is performed in the first round). The 48-bit subkey is obtained by a second permutation, denoted PC-2. The DES algorithm is detailed in Fig. 1

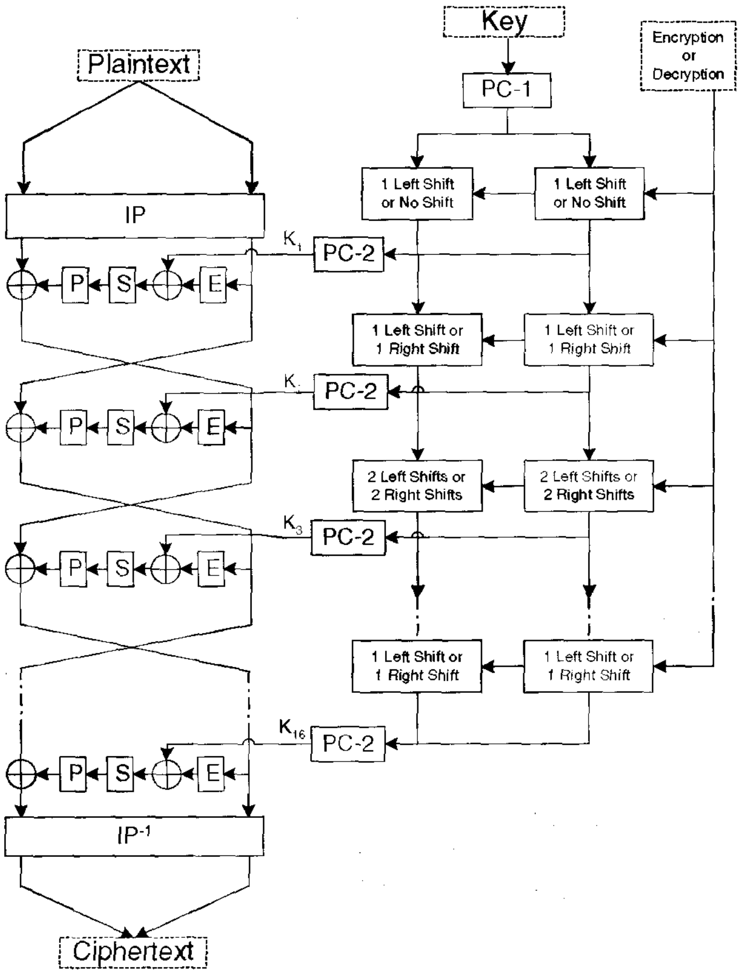
Key
Encryption or Decryption
Plaintext
PC-1
1 Left Shift or No Shift
1 Left Shift or No Shift
IP
K1
PC-2
P
S
E
1 Left Shift or 1 Right Shift
1 Left Shift or 1 Right Shift
K2
PC-2
2 Left Shifts or 2 Right Shifts
2 Left Shifts or 2 Right Shifts
K3
PC-2
1 Left Shift or 1 Right Shift
1 Left Shift or 1 Right Shift
K16
PC-2
IP-1
Ciphertext

Fig. 1. The DES algorithm

3 XILINX IMPLEMENTATIONS

The first proposed solution is a full unrolled and pipelined DES implementation. It pipelines the data through 16 stages, putting registers after each enciphering/key round. This increases the data rate hugely, but also the logic requirement compared to a sequential design.

According to Fig. 4 and [14], the critical path through the round is quite long. First, a multiplexer selects the correct key bits depending on the encryptor/decryptor mode. The selected key bits are XORed with the R part. The resulting 6-bit fields are used to address the S-boxes whose critical path is one LUT followed by two multiplexer functions (F5 and F6). Finally, output bits from the S-boxes are XORed with the L part. Fig. 2 details the critical path.

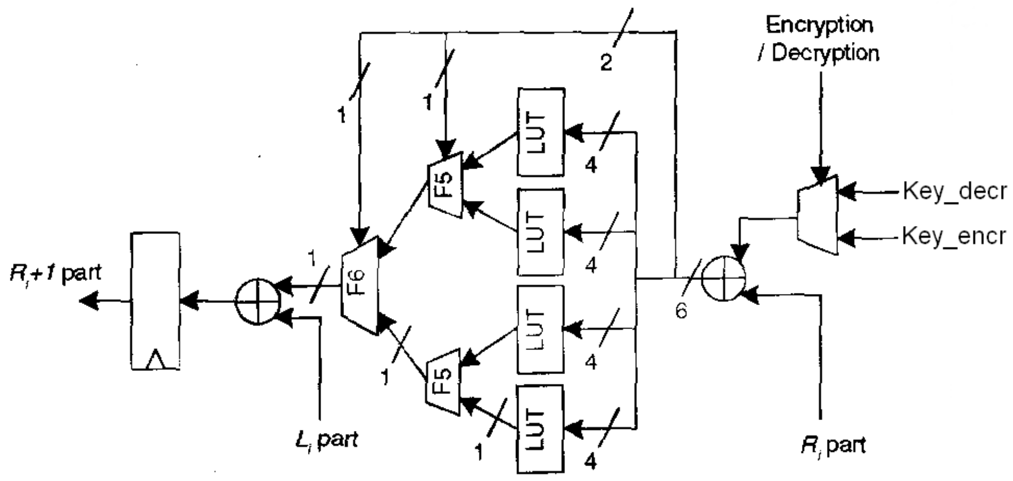

Fig. 2. Critical part of the DES design

The first proposed way to reduce this critical path is to combine the F6 function with the final XOR operator. The resulting 4-bit input logic function that fits in an LUT and eliminates the F6 delay. Another improvement is to decouple the key from the enciphering calculation. This is done with a precomputation of the key schedule. So, the multiplexer selecting the key can be removed from the critical path, putting registers after this multiplexer.

Xilinx also proposes a second implementation. To reach higher data rates, one inserts a pipelined stage, respectively, after the key XOR and after F5 functions. It results in a 3-stage pipeline per round and a 48-stage pipeline ovd cipher.

Nevertheless, after checking and simulating their aj able source code on the web, we found two errors, are they forgot to put a 1-stage pipeline after the XOR beta the key and R part. Actually, Xilinx implemented in 1-stage pipeline, but sent the XOR directly between that and R part into S-boxes, in place of the correspond registered value. They also forgot to register the key before the XOR function. Therefore, their critical pal quite a bit longer. Finally, their solutions do not imply a correct DES that can encrypt every cycle.

4 PROPOSED FPGA DESIGNS

To be speed efficient, we propose designs that unroll the 16 DES rounds and pipeline them. In addition, we implemented solutions that allow us to change the plaintext, the key, and the encryption/decryption mode on a cycle-by-cycle basis, with no dead cycle. As a result, we can achieve very high data rates of encryption/decryption with exactly the same interface as Xilinx.

All of our implementations are first based on new mathematical representations of the DES algorithm. Indeed, the original description of DES is not optimized for FPGA implementation regarding the speed performance and the number of LUTs used. An FPGA is based on slice composed of two 4-bit LUTs (Look Up Tables) and two 1-bit registers. Therefore, an optimal way to reduce the LUTs used is to regroup all the logical operations in order to obtain a minimum number of blocks that take 4-bit inputs and give 1-bit outputs. In addition, we have to note that permutation and expansion operations (typically, P, E, IP, IP-1, PC-1, and PC-2) do not require additional LUTs, but only wire crossings.

4.1 First Solution

In [15], equivalent mathematical descriptions for DES are proposed. Based on these transformations, we propose new representations. First, we transform the round function of the enciphering computation. This transformation has no impact on the computed result of the round.

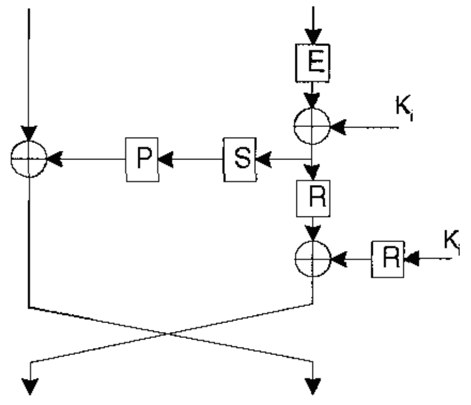

Fig. 3. Modified representation of one DES-round

Fig. 3 shows a modified round representation, where we move the E box and the XOR operation. This involves the definition of a new function denoted R (like reduction):

$$R=E^{-1};$$
$$\forall x,\ R(E(x))=x; \qquad (1)$$
$$\exists y | E(R(y)) \neq y.$$

Now, if we change all the enciphering parts of DES (see Fig. 1) with this modified round function and if we combine the E and XOR block with XOR block of the previous round, we get the architecture detailed in Fig. 4.

In this new arrangement of the DES structure, the first and last rounds are quite different from intermediate ones. Therefore, we obtain an irregular architecture. In addition, we increase the number of E and R blocks, which does not alter the number of LUTs consumed. We also keep exactly the same number of S-boxes, which is the expensive part of the architecture. Finally, the number of modulo two sum operators is slightly increased by 32 additional 2-bit XOR operators. We can directly conclude that this design consumes more logic than Xilinx implementations.

The left part of Fig. 5 illustrates how the critical path, in our solution, is hugely decreased. We only keep one S-box operator and one XOR function. With this solution, we obtain a 1-stage pipeline per round. Due to the irregular structure of our design, we have to add an additional stage in the first round. To be speed efficient for implementation constraints, we also put a 2-stage pipeline, respectively, in the input and in the output. As mentioned in the figure, first and last registers are packed into IOBs. Therefore, we obtain a 21-stage pipeline.

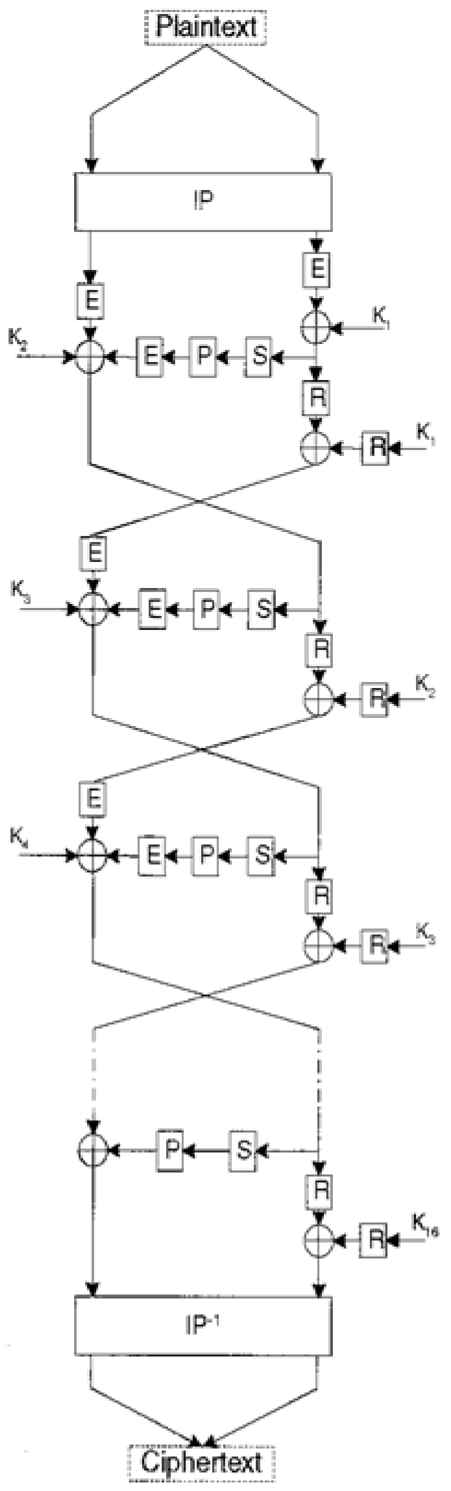

Fig. 4. First modified representation of the DES algorithm

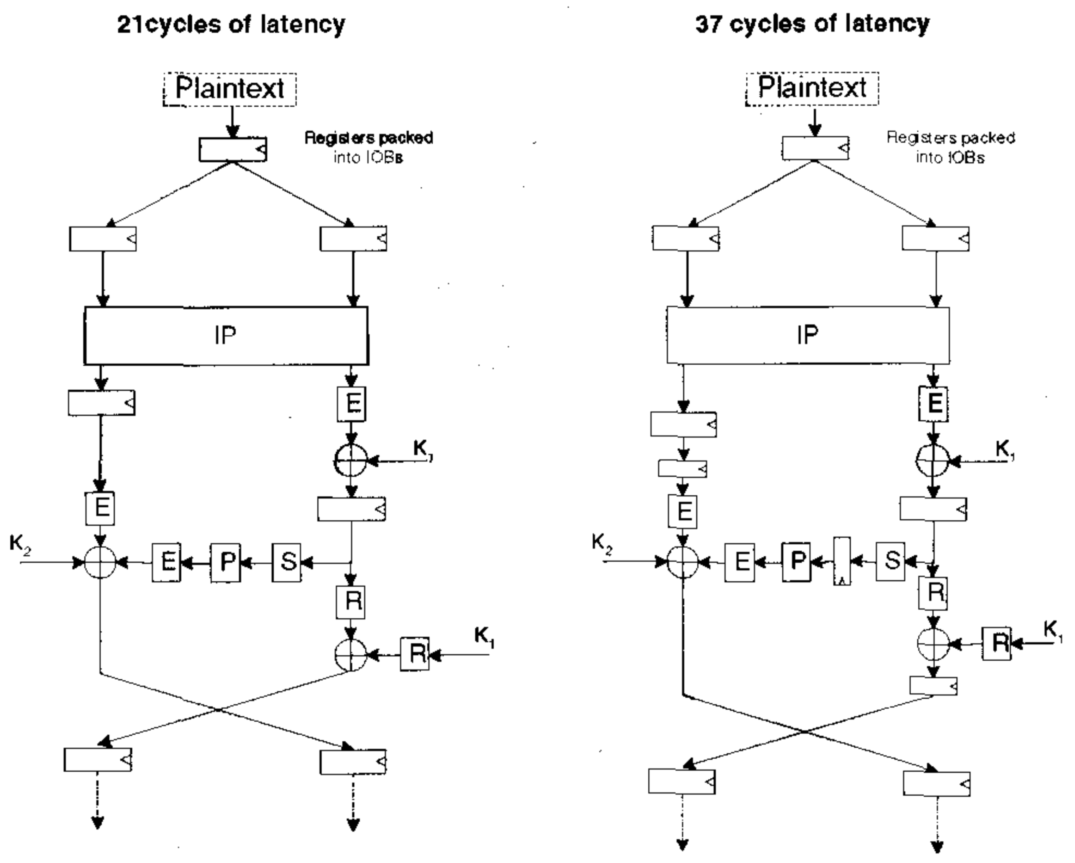

Fig. 5. Pipelining our first solutions.

In the right part of Fig. 5, we put an extra pipelined stage in ach round in order to limit the critical path to only one S-box. As a consequence, we get a 37-stage pipelined design.

4.2 Second Solution

Another solution is to move the R and XOR of the right part of the round into the left XOR operator of the previous round. As a result, we obtain the architecture shown in Fig. 6.

As Fig. 6 underlines, we again obtain an irregular architecture. First and last rounds are quite different from intermediate rounds. We also keep exactly the same number of S-boxes as our precedent design. But, we really decrease the number of modulo two sum operators. We spare 15 x 32 2-bit XOR and can directly conclude that this design consumes less logic than Xilinx implementations.

Fig. 6 gives more details about the initial round of our design.

5 LINEAR CRYPTANALYSIS

This section is a brief reminder of Matsui's linear cryptanalysis [4], [5], [6] before explaining the resulting VHDL design. Linear cryptanalysis is an attack based on the existence of some unbalanced linear relationship between inputs and outputs of a reduced-round version of the target encryption scheme. In the case of DES, Matsui used the relationship:

$$P_L[15]\oplus P_H[7,18,24,29]\oplus C_L[7,18,24]=K_1[22]\oplus K_3[22]\oplus K_4[44]\oplus K_5[22]\oplus K_7[22]\oplus K_8[44]\oplus K_9[22]\oplus K_{11}[22]\oplus K_{12}[44]\oplus K_{13}[22], \qquad (2)$$

where $X[7,18,24,29] := X[7]\oplus X[18]\oplus X[24]\oplus X[29]$

Basically, this relationship means that the exclusive-or of some well-chosen bits of the plaintext (namely, the seventh, 18th, 24th, 29th bits of its high-order part) and some well-chosen bits of the ciphertext are equal to the exclusive-or of some well-chosen secret bits of the key with probability different from ½.

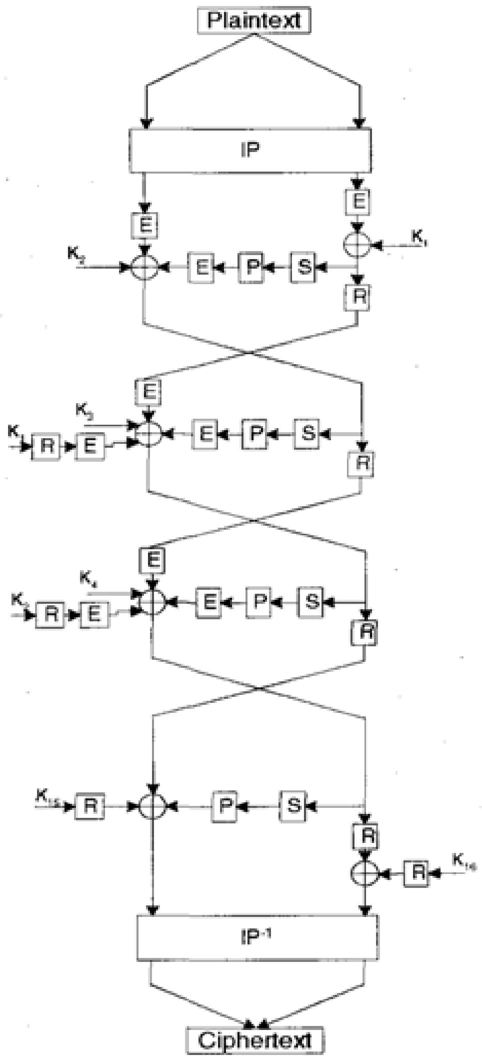

Fig. 6. Second modified representation of the DES algorithm

We can easily calculate its dual, obtained by reversing the expression

$$P_L[7,18,24]\oplus C_L[15]\oplus C_H[7,18,24,29]=K_2[22]\oplus K_3[44]\oplus K_4[22]\oplus K_6[22]\oplus K_7[44]\oplus K_8[22]\oplus K_{10}[22]\oplus K_{11}[44]\oplus K_{12}[22]\oplus K_{14}[22]. \quad (3)$$

Those characteristics are the best linear approximations of 14-round DES cipher. They are satisfied with probability $p=\frac{1}{2}-1.9\times$.

Expression (2) is then extended to the full 16 rounds by adding two nonlinear round functions, respectively, in the first and 16th rounds (we will leave the second relationship aside in this discussion since it is the first one's dual):

$$P_L[7,18,24,29]\oplus P_H[15]\oplus F_1(P_L,K_1)[15]\oplus C_H[7,18,24]F_{16}(C_L,K_{16})[7,18,24]=K_2[22]\oplus K_4[22]\oplus K_5[44]\oplus K_6[22]\oplus K_8[22]\oplus K_9[44]\oplus K_{10}[22]\oplus K_{12}[22]\oplus K_{13}[44]\oplus K_{14}[22], \quad (4)$$

where $F_1(P_L,K_1)$ denotes the first round function. This relationship keeps exactly the same probability as (2). In fact, only 6 bits of K_1 (resp. K_{16}) influence the value of $F_1(P_L,K_1)$ [15] (resp. $F_{16}(C_L,K_{16})$ [7,18,24]).

If we compute this equation for all 4,096 possibilities of the key (K_1 and K_{16}), a large number of plaintexts, knowing that only one of these 4,096 keys is correct, we will find one significative probability corresponding to the 12 correct key bits. The following algorithm summarizes this idea.

Algorithm

1. For each candidate $(K_1^{(i)}|K_{16}^{(j)})$ (i = 1,2,.. .64, j = 1,2, ...64) of $(K_1|K_{16})$, let $T_{(I,j)}$ be the number of plaintexts such that the left side of the (4) is equal to zero.
2. Let $T_{(max\ i,max\ j)}$ be the maximal value, $T_{(min\ i,min\ j)}$ the minimal value of all $T_{(I,j)}$s, and N the number of plaintexts / ciphertexts.

If $|T_{(max\ i,max\ j)}-N/2|>|T_{(min\ i,min\ j)}-N/2|$, then adopt the key candidate corresponding to $T_{(max\ i,max\ j)}$.

If $|T_{(max\ i,max\ j)}-N/2|<|T_{(min\ i,min\ j)}-N/2|$, then adopt the key candidate corresponding to $T_{(min\ i,min\ j)}$.

An extra bit can be found thanks to (4). Indeed, 12 key bits of K_1 and K_{16} were found thanks to the previous algorithm and we can derive the value of

$$K_2[22]\oplus K_4[22]\oplus K_5[44]\oplus K_6[22]\oplus K_8[22]\oplus K_9[44]\oplus K_{10}[22]\oplus K_{12}[22]\oplus K_{13}[44]\oplus K_{14}[22]$$

from the same experiments. It is therefore possible to recover 12 + 1 bits of the key. The same treatment can be applied to the dual equation (4), thus yielding a total of 26 bits. The remaining 30 unknown key bits have to be searched exhaustively.

Let us have a look at the success rate of Matsui's attack. In [4], the following lemmas are proposed:

Lemma 1. Let N be the number of given random plaintexts and p be the probability that (4) holds and assume |p is sufficiently small. Then, the success rate of the algorithm depends on the bits involved in the equation and √N|p only.

Generally speaking, it is not easy to calculate numerically the accurate probability above. However, under a condition, it can be possible as follows: In this case, we rewrite it for Matsui's attack on a full DES.

Lemma 2. With the same hypotheses as Lemma 1, let $q^{(i,j)}$ be the probability that the following equation holds for subkeys $(K_1^{(i)}|K_{16}^{(j)})$ and random variables X, Y:

$$F_1(X,R_1)[15]\oplus F_{16}(Y,R_{16})[7,18,24]=F_1(X,K_1^{(j)})[15]\oplus F_{16}(Y,K_{16}^{(j)})[7,18,24], \quad (5)$$

where K_1 and K_{16} are the correct subkeys.

Then, if $q^{(i,j)}$ are independent, the success rate of the algorithm is

$$f_x = \frac{1}{\sqrt{2\Pi}} e^{\frac{-x^2}{2}}, f_y = \frac{1}{\sqrt{2\Pi}} e^{\frac{-y^2}{2}},$$

$$\int_{x=-2\sqrt{N}|p-\frac{1}{2}|}^{\infty} \left(\prod_{(i,j)} \int_{-x-4\sqrt{N}(p-\frac{1}{2})q^{(i,j)}}^{x+4\sqrt{N}(p-\frac{1}{2})(1-q^{(i,j)})} f_y dy \right) f_x dx, \qquad (6)$$

where the product is taken over all subkey candidstes except ($K_1|K_{16}$)

We compute (6) to show the theoretical success probability of Matsui's 14-round attack. (Due to the large (4,095) number of factors involved, the equation could not be computed exactly; therefore, we used an approximation.) Results are shown in Table 1.

Table 1

Success rate of Matsui's attack on a full DES

N	2^{37}	2^{38}	2^{39}	2^{40}	2^{41}	2^{42}	2^{43}	2^{44}	2^{45}	2^{46}
Success rate	0,1%	0,1%	0,3%	0,8%	2,5%	9,3%	31,9%	71,8%	95,3%	99,8%

For information, we give the complexities of Matsui's linear attack on a full DES predicted by Knudsen. Comparing with Table 2, our theoretical result seems to be too pessimistic.

Table 2

Knudsen's value of the same attack

N	2^{43}	2^{44}	2^{45}
Success rate	32,5%	77,7%	99,4%

6 FPGA IMPLEMENTATION OF MATSUIS ATTACK

As previously described, Matsui's linear cryptanalysis allows us to find 26 key bits with about 243 known-plaintexts. We propose an FPGA implementation of Matsui's attack that permits recovering 12 + 1 key bits with about 243 known-plaintexts. We did not use the second relation to spare hardware resources and we decided to use our second 21-stage pipelined DES, which is the fewer resources consuming design. In order to increase speed performances, we parallelized two of them so that we got a data rate of two encryptions per cycle. We also modified them in order to gain resources space: The key schedule was simplified and the input and output registers were removed.

Nevertheless, for a hardware implementation, the main problem of this attack is the 2^{12} counters needed to perform the key guess. Knowing that about 24,000 LUTs available on our FPGA, the implementation of 2^{12} parallelized counters is much too expensive to be realistic (at 65,000 LUTs). (We have to keep a sufficient bits size, say 16 bits, for the counters to have an efficient and feasible implementation.)

This section will briefly introduce how we implement Matsui's linear cryptanalysis without 4,096 parallel counters in one FPGA board, keeping our very fast data throughput. We do it with 4,096 RAM-based counters. (We configure all the RAMs to have 8-bit address and 16-bit data.)

In practice, we need to implement 4,096 RAM based counter values, with only 32 parallel access (with real and writing operations; we use dual access RAMs). Therefore, this operation can be performed in 128 clock cycles.

This is practically performed using a large serial/parallel converter making the ciphertext bits involved in Matsui's linear approximation (4) available during 128 cycles.

By choosing the plaintext bits involved in the linear approximation (4) such that they are fixed during the same 128 clock cycles, we avoid the need of a serial/parallel converter for the plaintext bits. We also avoid the use of I0R operators between plaintext and ciphertext parts. Therefore, we spare a lot of hardware resources. We just need an *n*-delay shift register (*SR* block) to synchronize the design.

To generate plaintext bits, we use an LFSR of 57 bits and a 6-bit counter (the remaining bit is used for the two DES parallelization). This counter controls the P_L part used to calculate $F_1[15]$, varying every 128 cycles. Therefore, we obtain 128 successive cycles where the P_L part of $F_1(P_L,K_1)[15]$ is constant.

Knowing 256 parallelized results of $P_L[7,18,24,29] \oplus P_H[15] \oplus C_H[7,18,24] \oplus F_{16}(C_L,K_{16})[7,18,24]$, we have to count the number of bits equal to 0 and subtract 128, thanks to the previous comment (we only store the bias). We obtain 9-bit result, called *bias* in Fig. 6.

Depending on the 32 parallelized values of $F_1(P_L,K_1+i)$ [15], we have to carry out a subtraction or an addition between the 32 RAM values stored (in the correct address) and the *bias* value. (We have i = from 0 to 31 and K_1 equal to 0 or 1.)

Therefore, we get one Matsui's attack implementation that allows us to recover 12 + 1 secret key bits. Our cryptanalysis design is based on a sequentialized access of 4,096 counters, without altering the encryption rate of two DES per cycle. To analyze our experiments, the 4,096 RAMs stored results are sent to the PC when one of them exceeds the 16-bit RAM data size. In addition, the PC can send the secret key to the FPGA board. This allows us to perform very practical tests.

7 EXPERIMENTAL RESULTS

In this section, we give the results we got running Matsui's attack on eight Xilinx FPGAs (VIRTEX1000 bg560-4). We carried out the experiments at a work frequency = 66.6 MHz ($=2^{26}$) (Because of the FPGA heat running at 66 MHz, we do not carry experiments at higher frequency. It is why we use our second 21-stage solution, which is the less resource consuming design.). Therefore, we are able to compute $2x2^{26}$ equations per second. Using eight FPGA boards, 2^{43} evaluations take less than 2.3 hours.

We performed tests with 71 different keys. Table 7 summarizes the experimental success rate of the attack for various amounts N of chosen-plaintext/ciphertext pairs.

These experimental results suggest that Matsui's theoretical analysis is quite good (see Tables 5 and 6). Indeed, our results are very close to mathematical estimations.

CONCLUSION

This paper deals with two new ideas for FPGA implementations of DES leading to four improved practical appropriate implementations. All of them are very efficient in terms of speed and/or resources needed. Then, this paper presents the first known FPGA implementation of Matsui's linear cryptanalysis. The resulting attack is capable of finding a 13-bit key in less than 2.3 hours, using eight FPGA boards. In addition, it is worth noting that, with the new Xilinx FPGA (Xilinx VIRTEX-II XC2V8000), we would be able to carry out the same attack in about 1 hour, using only one FPGA board. Therefore, in some applications, FPGAs can be used as powerful cryptographic calculation tools.

REFERENCES

1. L.R. Knudsen and J.E. Mathiassen, "A Chosen-Plaintext Linear Attack on DES” Proc. Int'l Symp. Foundations of Software Eng. (FSE '00), B. Schneier, ed., pp. 262-272, 2000.
2. P. Junod, "Linear Cryptanalysis of DES," Master's thesis, Swiss Inst, of Technology, 2000.
3. P. Junod, "On the Complexity of Matsui's Attack," Proc. ACM Symp. Applied Computing (SAC '01), pp. 216-230, 2001.
4. M. Matsui, "Linear Cryptanalysis Method for DES Cipher," Proc. Advances in Cryptology – EuroCrypt '93, T. Helleseth, ed., pp. 386-397, 1993.
5. M. Matsui, "The First Experimental Cryptanalysis of the Data Encryption Standard," Y. Desmedt, ed., Proc. Advances in Cryptology – Crypto '94, pp. 1-11, 1994.
6. F. Koeune, G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, J.-P. David, and J.-D. Legat, "An FPGA Implementation of the Linear Cryptanalysis," Proc. In'tl Conf. Field Programmable Logic and Applications (FPL 4)2), M. Glesner, P. Zipf, M. Renovell, eds., pp. 845-853, 2002.
7. J.M. Rabaey, Digital Integrated Circuits. Prentice Hall, 1996.
8. Xilinx, "Virtex 2.5V Field Programmable Gate Arrays Data Sheet," http://www.xilinx.com, 2002.
9. Xilinx, V. Pasham, and S. Trimberger, "High-Speed DES and Triple DES Encryptor / Decryptor," http://www.xilinx.com/xapp/xapp270.pdf, Aug. 2001.
10. B. Schneier, Applied Cryptography, second ed. John Wiley & Sons, 1996.
11. Nat'l Bureau of Standards, FIPS PUB 46, The Data Encryption Standard, US Dept. of Commerce, Jan. 1977.
12. FreelP, http://www.free-ip.com/DES/index.html, 2000.
13. C. Patterson, "High Performance DES Encryption in Virtex FPGAs Using Jbits," Proc. IEEE Symp. Field-Programmable Custom Computng Machines (FCCM VI), 2000.
14. S. Trimberger, R. Pang, and A. Singh, "A 12 Gbps DES Encryptor/ Decryptor Core in an FPGA," Proc. Cryptographic Hardware and Embedded Systems (CHES '00), pp. 156-163, 2000.
15. M. Davio, Y. Desmedt, M. Fossprez, R. Govaerts, J. Hulsbosch, P. Neutjens, P. Piret, J.J. Quisquater, J. Vandewalle, and P. Wouters, "Analytical Characteristics of the DES," Proc. Advances in Cryptology – Crypto '83, D. Chaum, ed., pp. 171-202, 1983.

Мирошник М.А., Коваленко М.А. Использование ПЛИС для реализаций стандарта шифрования данных. В настоящей статье представлены две оригинальные реализации ПЛИС DES шифрования / дешифрования. Эта реализация являются самой быстродействующей в настоящее время. В схеме текст, ключ и режим (шифрование (кодирование) / расшифровка (декодирование)) могут быть изменены без пассивных циклов. Результирующая схема развернута на восемь и ПЛИС позволяет найти (12 + 1) битов ключа за 2,3 часов.

Ключевые слова: криптография, стандарт шифрования данных, линейный криптоанализ, FPGA, эффективные реализации.

Мірошник М.А., Коваленко М.А. Використання ПЛІС для реалізацій стандарту шифрування даних. У цій статті представлені дві оригінальні реалізації ПЛІС DES шифрування / дешифрування. Ця реалізація є самою швидкодіючою в даний час. У схемі текст, ключ і режим (шифрування (кодування) / розшифровка (декодування)) можуть бути змінені без пасивних циклів. Результуюча схема розгорнута на вісім і ПЛІС дозволяє знайти (12 + 1) бітів ключа за 2,3 годин.

Ключові слова: криптографія, стандарт шифрування даних, лінійний криптоаналіз, FPGA, ефективні реалізації.

Рецензент д.т.н., профессор, профессор кафедры СКС Листровой С.В. (УкрГАЖТ)

Поступила 13.12.2013г.