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. (We carried out our experiments on VIRTEX1000 bg560-4.)

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.[1] 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.

The paper is organized as follows: Section 2 describes the FPGA technology used, the synthesis/implementation tools, and our FPGA board; Section 3 recalls the Data Encryption Standard (DES); Section 4 refers to the best previous known implementations by Xilinx; Section 5 describes our two proposals to improve FPGA implementations and compares them with previous designs; before explaining our linear cryptanalysis design, Section 6 recalls the basic principles of Matsui's linear cryptanalysis; Section 7 describes our FPGA implementation of Matsui's linear cryptanalysis; finally, Section 8 summarizes the results we obtained on a set of 71 practical attacks.

2 Hardware Description

In this section, we briefly describe the structure of a VIRTEX FPGA as well as the synthesis and implementation tools that were used to obtain our results.

2.1 Configurable Logic Blocks (CLBs)

The basic building block of the VIRTEX logic block is the logic cell (LC). An LC includes a 4-input function generator, carry logic, and a storage element. The output from the function generator in each LC drives both the CLB output and the D input of the flip-flop. Each VIRTEX CLB contains four LC's, organized in two similar slices. Fig. 1 shows a detailed view of a single slice. Virtex function generators are implemented as 4-input look-up tables (LUTs). Besides its operation as a function generator, each LUT can provide a 16 $\times$ 1-bit synchronous RAM. Furthermore, the two LUTs within a slice can be combined to create a 16 $\times$ 2-bit or 32 $\times$ 1-bit synchronous RAM or a 16 $\times$ 1-bit dual port synchronous RAM. The VIRTEX LUT can also provide a 16-bit shift register.

1. We do not speak about the third Knudsen's chosen-plaintext attack in [1], which is the actual best found chosen-plaintext attack.

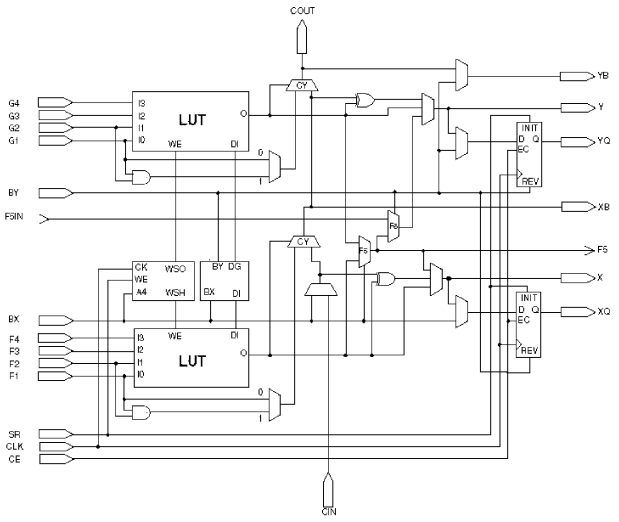


Fig. 1. The VIRTEX slice.

The storage elements in the VIRTEX slice can be configured either as edge-triggered D-type flip-flops or as level-sensitive latches. The D inputs can be driven either by the function generators within the slice or directly from slice inputs, bypassing function generators.

The F5 multiplexer in each slice combines the function generator outputs. This combination provides either a function generator that can implement any 5-input function, a 4:1 multiplexer, or selected functions of up to nine bits. Similarly, the F6 multiplexer combines the outputs of all four function generators in the CLB by selecting one of the F5-multiplexer outputs. This permits the implementation of any 6-input function, an 8:1 multiplexer, or selected functions up to 19 bits. The arithmetic logic also includes an XOR gate that allows a 1-bit full adder to be implemented within an LC. In addition, a dedicated AND gate improves the efficiency of multiplier implementations.

Finally, VIRTEX FPGAs incorporate several large RAM blocks. These complement the distributed LUT implementations of RAMs. Every block is a fully synchronous dual-ported 4,096-bit RAM with independent control signals for each port. The data widths of the two ports can be configured independently.

2.2 Hardware Targets

For our implementations, we used VIRTEX, VIRTEXE, and VIRTEXII technologies. We chose these technologies in order to allow relevant comparisons with the best-known FPGA implementations of DES. In this paper, we compare the number of LUTs, registers, and slices used. We also evaluate the delays and frequencies thanks to our implementation tools (post place-and-route frequencies). The synthesis was performed with FPGA Express 3.6.1 (SYNOPSYS) and the implementation with XILINX ISE-4. Our circuits were described using VHDL.

Practical experiments were carried out on eight Virtex1000BG560-4 boards that we developed within DICE (the UCL Microelectronics Laboratory, http://www.dice.ucl.ac.be). One board is composed of a control FPGA (FLEX

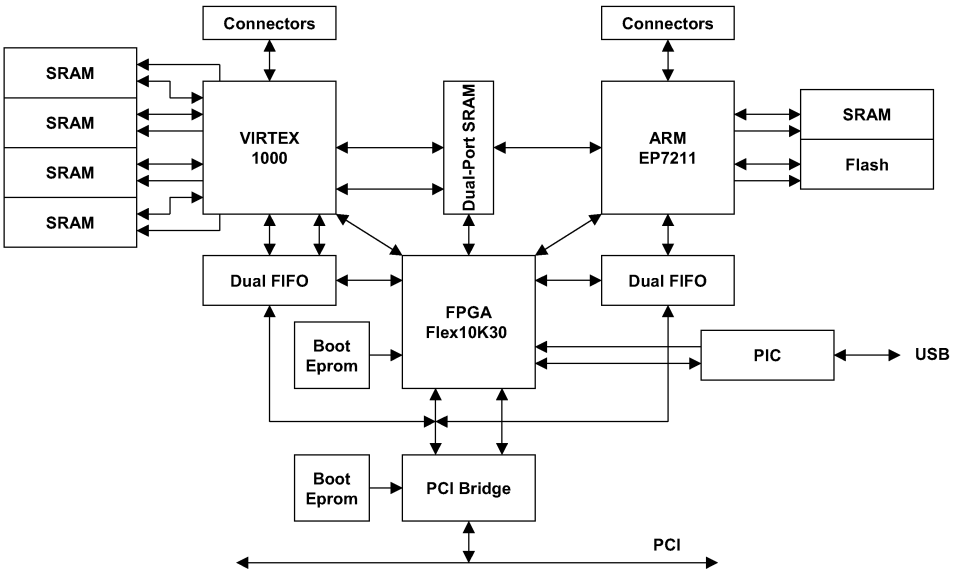


Fig. 2. The block scheme of the board.

10K30) and a VIRTEX1000 FPGA (counts 6,144 CLBs and 32 dual access RAMs of 4,096 bits) associated with several processors (ARM and PIC) and fast extern memories. The board has multiple compatible PC interfaces (PCI, USB). Our simulations used a PCI communication. The cost of one FPGA board is roughly $3,500. Fig. 2 represents the block scheme of the board and Fig. 3 is a picture of it.

3 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 [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. 4.

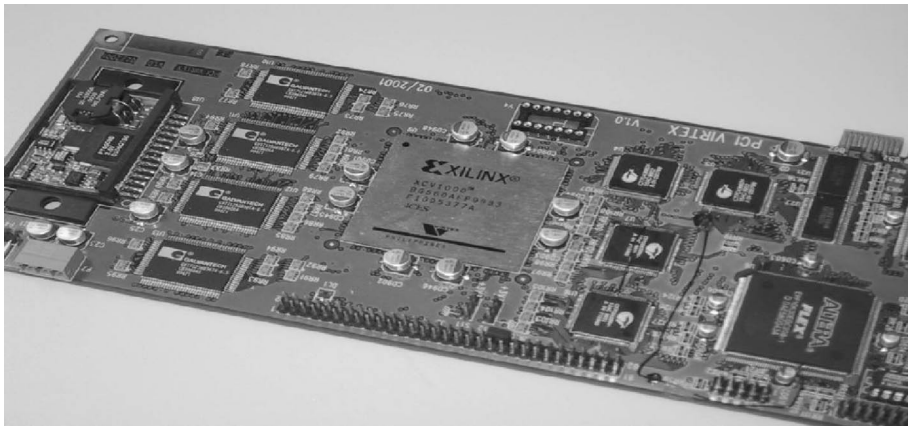


Fig. 3. Our final VIRTEX board.

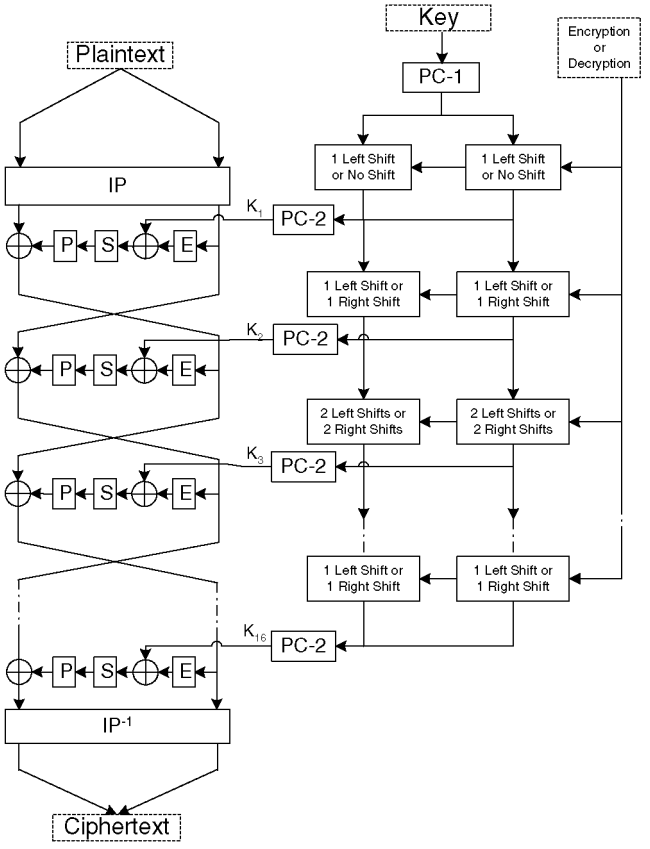


Fig. 4. The DES algorithm.

4 Xilinx Implementations

This section briefly summarizes the previous implementations of Xilinx and gives their final results. For thorough information, please refer to [9], [14].

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. 5 details the critical path.

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 over the cipher.

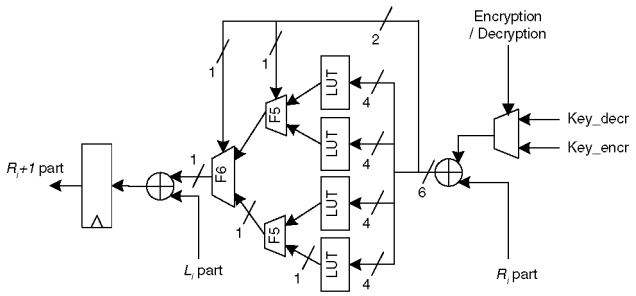


Fig. 5. Critical path of the DES design.

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

Table 1 summarizes their two pipelined designs where we modify their mistakes and, therefore, their results.

5 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 slices 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 all permutation and expansion operations (typically, *P*, *E*, *IP*,

TABLE 1
Final Results of Xilinx Pipelined Implementations

Pipeline	16-stage	48-stage
LUT's used	4225	4225
Registers used	1943	5822
Frequency in XCV300-6	100 MHz	158 MHz
Data rate in XCV300-6	6.4 Gbps	10.1 Gbps
Frequency in XCV300E-6	132 MHz	189 MHz
Data rate in XCV300E-6	8.4 Gbps	12.0 Gbps
Frequency in XC2V1000-5	/	237 MHz
Data rate in XC2V1000-5	/	15.1 Gbps

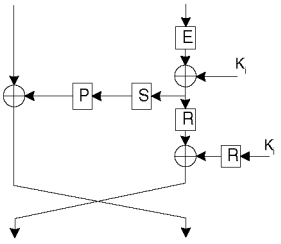


Fig. 6. Modified representation of one DES-round.

IP-1, *PC-1*, and *PC-2*) do not require additional LUTs, but only wire crossings and fanouts (pure routing). The next subsections show how to optimize the logic used or how to reduce the critical path.

5.1 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. 6 shows a modified round representation, where we move the *E* box and the *XOR* operation. This involves the definition of a new function (like reduction) denoted *R*:

$$\begin{aligned} R &= E^{-1}; \\ \forall x, R(E(x)) &= x; \\ \exists y \mid E(R(y)) &\neq y. \end{aligned} \tag{1}$$

Now, if we change all the enciphering parts of DES (see Fig. 4) 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. 7.

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.[2] We can directly conclude that this design consumes more logic than Xilinx implementations.

The left part of Fig. 8 illustrates how the critical path, in our solution, is hugely decreased. We only keep one S-box operator and one *XOR* function.[3] 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.

In the right part of Fig. 8, we put an extra pipelined stage in each round in order to limit the critical path to only one S-box. As a consequence, we get a 37-stage pipelined design.

2. The design exactly counts 17×32 2-bit XOR, 15×48 3-bit XOR, and 1×48 2-bit XOR.

3. *E* and *R* operators do not increase the critical path.

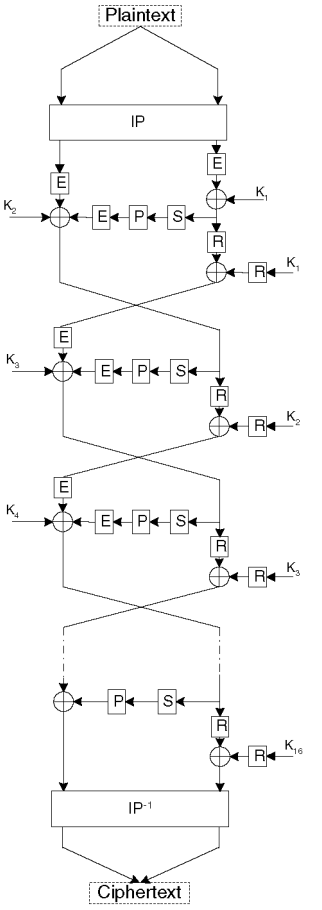


Fig. 7. First modified representation of the DES algorithm.

Table 2 shows our 21-stage and 37-stage pipelined results. Comparing to Table 1, our 21-stage gives better results in terms of speed, but consumes slightly more logical resources and registers. Concerning the 37-stage pipeline, we again use more LUTs, but reduce the number of registers needed. This is due to the fact that we only have a 2-stage pipeline per round. In addition, this design uses shift registers for the key schedule calculation. In Virtex FPGAs, *SRL16* cores can directly implement a 16-bit shift register into one LUT. So, we finally use 892 extra LUTs for shift registers.[4]

The reason why we have better speed results for the 37-stage pipeline is quite strange. Obviously, in its design, Xilinx does not put registers into IOBs and an additional pipelined stage before and after encryption. Without such registers, the critical path is in the input and output paths. In addition, to generate a data ready output signal, its proposed design is a solution with a critical path corresponding to two LUTs. It's why they get a small work frequency.

5.2 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. 9.

4. The corrected Xilinx implementation uses 903 LUTs for shift registers. No accurate values are given in [9], [14].

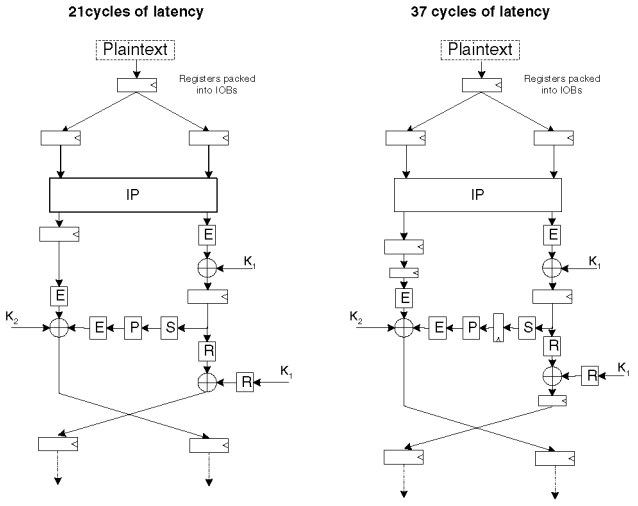


Fig. 8. Pipelining our first solutions.

As Fig. 9 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 15×32 2-bit XOR[5] and can directly conclude that this design consumes less logic than Xilinx implementations.

Fig. 10 gives more details about the initial round of our design.

Table 3 summarizes our 21-stage and 37-stage pipelined results. Comparing to Table 1, our 21-stage gives better results in terms of speed and logical resources. Nevertheless, it consumes slightly more registers. For the 37-stage pipeline, we again use fewer LUTs and also reduce the number of registers needed. This is due to the fact that we only have a 2-stage pipeline per round. In addition, this design uses shift registers for the key schedule calculation. So, we finally use 996 extra LUT's for shift registers.[6]

To conclude, we propose efficient and different solutions in terms of space and data rate for the hardware implemenation of DES. Depending on environment constraints, we really believe that one of our designs should be well appropriate. Especially, our second proposal is very interesting in terms of speed, LUTs used, and registers. For high ratio $Throughput/Area$, the second 37-stage pipelined solution is very efficient. Table 4 compares ratio $Throughput/Area$ between 48-stage Xilinx implementation and our second 37-stage implementation. We directly see a significant improvement: Our design is almost two times better than Xilinx one in term of ratio $Throughput/Area$.

6 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

5. The design exactly counts 14×48 4-bit XOR, 1×48 3-bit XOR, 1×48 2-bit XOR, 1×32 3-bit XOR, and 1×32 2-bit XOR.

6. The corrected Xilinx implementation uses 903 LUTs for shift registers. No accurate values are given in [14], [9].

TABLE 2
Final Results of Our First Implementations

Pipeline	21-stage	37-stage
LUT's used	4255	4255
Registers used	2424	4128
Frequency in XCV300-6	127 MHz	175 MHz
Data rate in XCV300-6	8.1 Gbps	11.2 Gbps
Frequency in XCV300E-6	176 MHz	258 MHz
Data rate in XCV300E-6	11.2 Gbps	16.5 Gbps
Frequency in XC2V1000-5	227 MHz	333 MHz
Data rate in XC2V1000-6	14.5 Gbps	21.3 Gbps

inputs and outputs of a reduced-round version of the target encryption scheme. In the case of DES, Matsui used the relationship

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

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 $\frac{1}{2}$.

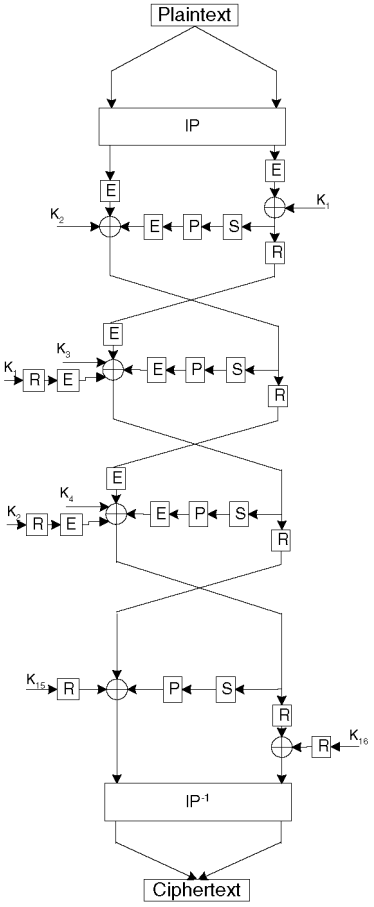


Fig. 9. Second modified representation of the DES algorithm.

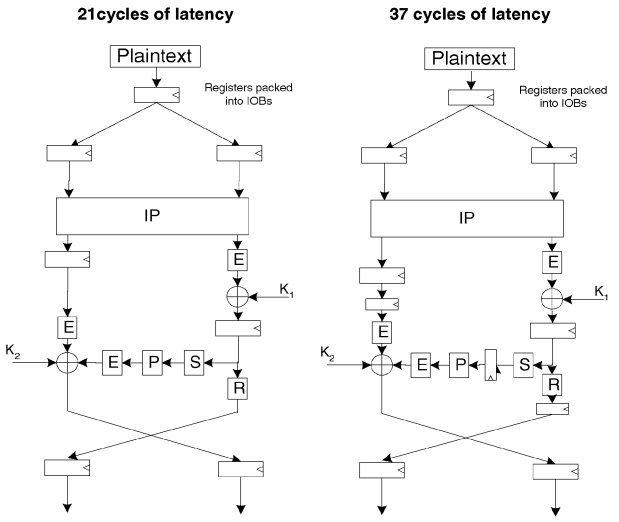


Fig. 10. Pipelining our second solutions.

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

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

Those characteristics are the best linear approximations of 14-round DES cipher. They are satisfied with probability $p = \frac{1}{2} - 1.19 \times 2^{-21}$.

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

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

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.

.0. Algorithm

1. For each candidate $(K_1^{(i)}|K_{16}^{(j)})$ $(i = 1, 2, \ldots 64,$ $j = 1, 2, \ldots 64)$ 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.

TABLE 3
Final Results of Our Second Implementations

Pipeline	21-stage	37-stage
LUT's used	3775	3775
Registers used	2904	4387
Frequency in XCV300-6	127 MHz	175 MHz
Data rate in XCV300-6	8.1 Gbps	11.2 Gbps
Frequency in XCV300E-6	176 MHz	258 MHz
Data rate in XCV300E-6	11.2 Gbps	16.5 Gbps
Frequency in XC2V1000-5	227 MHz	333 MHz
Data rate in XC2V1000-6	14.5 Gbps	21.3 Gbps

If $|T_{(max_i,max_j)} - \frac{N}{2}| > |T_{(min_i,min_j)} - \frac{N}{2}|$, then adopt the key candidate corresponding to $T_{(max_i,max_j)}$.

If $|T_{(max_i,max_j)} - \frac{N}{2}| < |T_{(min_i,min_j)} - \frac{N}{2}|$, 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

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

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 $|p - \frac{1}{2}|$ is sufficiently small. Then, the success rate of the algorithm depends on the bits involved in the equation and $\sqrt{N}|p - \frac{1}{2}|$ 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:*

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

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

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

TABLE 4
Comparisons with Xilinx Implementation of VIRTEXII

DES Version	Xil. 48-stage	**Our 37-stage**
Slices used	3900	**2965**
Data rate	15.1 Gbps	**21.3 Gbps**
Throughput/Area (Mbps/slices)	3.87	**7.18**

TABLE 5
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%

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

where the product is taken over all subkey candidates 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 5.

For information, we give the complexities of Matsui's linear attack on a full DES predicted by Knudsen. Comparing with Table 6, our theoretical result seems to be too pessimistic.

7 FPGA Implementation of Matsui's Attack

As previously described, Matsui's linear cryptanalysis allows us to find 26 key bits with about 2^{43} known-plaintexts. We propose an FPGA implementation of Matsui's attack that permits recovering 12 + 1 key bits with about 2^{43} 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 are available on our FPGA, the implementation of 2^{12} parallelized counters is much too expensive to be realistic (about 65,000 LUTs). (We have to keep a sufficient bits size, say 16 bits, for the conounters to have an efficient and feasible implementation.)

This section will briefly introduce how we implement Matsui's linear cryptanalysis without 4,096 parallelized 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.) Fig. 11 shows architecture and underlines how we took advantage of RAM blocks available in VIRTEX technology to implement counters.

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

According to Matsui's algorithm (see Section 6), we have to increment a counter (corresponding to $T_{(i,j)}$ for a key candidate $(K_1^{(i)}|K_{16}^{(j)})$) each time the linear approximation ((4)) equals to 0 for this key. A naive application of this scheme is obviously unfeasable (4,096 parallelized counters). Instead of incrementing the parallelized counters every clock cycle, with a value of 0 or 1, we increment the RAM counters every 128 cycles with a value between -128 and 128, corresponding to $(T_{(i,j)} - \frac{2\times 128}{2})$. Therefore, our RAM counters directly represent the bias $(T_{(i,j)} - \frac{2\times N}{2})$. The "2"s, before *times* operations, correspond to the two parallelized DES.

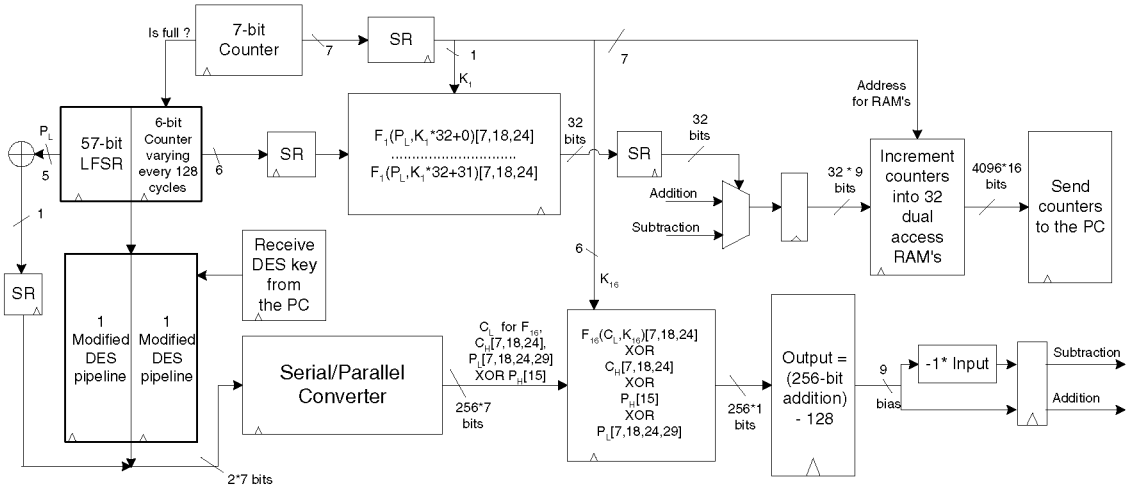


Fig. 11. Architecture of Matsui's linear cryptanalysis.

TABLE 6
Knudsen's Values of the Same Attack

N	2^{43}	2^{44}	2^{45}
Success rate	32.5%	77.7%	99.4%

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 convertor for the plaintext bits. We also avoid the use of *XOR* 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. 11. 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.

8 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 2×2^{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.

TABLE 7
Experimental Success Rate (SR) of Matsui's Attack

N	2^{37}	2^{38}	2^{39}	2^{40}	2^{41}	2^{42}	2^{43}
SR	0%	0%	0%	1.4%	0%	9.9%	26.8%

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

9 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 '02),* 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, year?

[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.

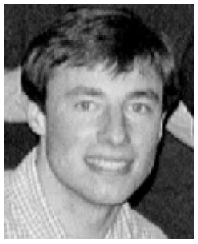
[11] Nat'l Bureau of Standards, *FIPS PUB 46, The Data Encryption Standard,* US Dept. of Commerce, Jan. 1977.

[12] FreeIP, http://www.free-ip.com/DES/index.html, year?

[13] C. Patterson, "High Performance DES Encryption in Virtex FPGAs Using Jbits," *Proc. IEEE Symp. Field-Programmable Custom Computing Machines (FCCM '01),* 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.



Gaël Rouvroy (S'02) began his engineering studies in 1996 at the Université catholique de Louvain (UCL, Belgium). In 2000, he made a 4-month student exchange and joined the University of Virginia. In 2001, he received the MS degree in electronics and mechanics engineering with the highest distinction from the Belgian university. He is currently a PhD candidate at UCL, in the UCL Crypto Group, where he is involved in the research project TACTILS. He is working on the analysis of cryptographic primitives, but also on efficient speed and secure cipher FPGA implementations, under the supervision of Professor Jean-Didier Legat and Jean-Jacques Quisquater. He is a student member of the IEEE.

Francois-Xavier Standaert (S'O2) received the electrical engineering degree with high distinction from the Université catholique de Louvain in 2001. He is currently a research assistant i the Electrical Engineering Department of the same university, where he is working toward the PhD degree. He is a member of the UCL Crypto Group and is involved in the research project TACTILS (Tracage et Acces Conditionnel Temps-reel d'Images par Lecteur Securise). His research interests include digital design and FPGAs, cryptographic hardware, block ciphers and side-channel analysis. He is a student member of the IEEE.

Jean-Jacques Quisquater is a professor of cryptography and multimedia security in the Department of Electrical Engineering, University of Louvain, Louvain-la-Neuve, Belgium, where he is responsible, at least at the scientific level, for many projects related to smart cards (protocols, implementations, side-channels), to secure protocols for communications, digital signatures, payTV, protection of copyrights, and security tools for electronic commerce. He was the main designer for several coprocessors for powerful smart cards: CORSAIR (Philips) and FAME (Philips). He holds 17 patents in the field of smart cards. He is co-inventor of an identification cryptographic scheme, the so-called GQ scheme. He is a member of the IEEE.

Jean-Didier Legat received the engineering and PhD degrees in microelectronics from the Université catholique de Louvain, Louvain-la-Neuve, Belgium, in 1981 and 1987, respectively. From 1987 to 1990, he was with Image Recognition Integrated Systems (I.R.I.S.), a new company specializing in optical character recognition and automatic document processing. He was cofounder and vice-president of I.R.I.S. In October 1990, he returned to the UCL Microelectronics Laboratory. He is currently a professor in the Electrical Engineering Department. His current interests are low power digital circuits, reconfigurable architectures and design of embedded integrated circuits in the area of artificial neural networks, digital signal processing, computer vision, and pattern recognition. He has been an author or coauthor of more than 110 publications in the field of microelectronics and he is a member of the IEEE. He is currently chairman of the Electrical Engineering Department of UCL.