Fast DES Implementations for FPGAs and Its Application to a Universal Key-Search Machine *

Jens-Peter Kaps** and Christof Paar

Electrical and Computer Engineering Department
Worcester Polytechnic Institute
Worcester, MA 01605-2280
{kaps,christof}@ece.wpi.edu

Abstract. Most modern security protocols and security applications are defined to be *algorithm independent*, that is, they allow a choice from a set of cryptographic algorithms for the same function. Although an algorithm switch is rather difficult with traditional hardware, i.e., ASIC, implementations, Field Programmable Gate Arrays (FPGAs) offer a promising solution. Similarly, an ASIC-based key search machine is in general only applicable to one specific encryption algorithm. However, a key-search machine based on FPGAs can also be algorithm independent and thus be applicable to a wide variety of ciphers. We researched the feasibility of a universal key-search machine using the Data Encryption Standard (DES) as an example algorithm.
We designed, implemented and compared various architecture options of DES with strong emphasis on high-speed performance. Techniques like *pipelining* and *loop unrolling* were used and their effectiveness for DES on FPGAs investigated. The most interesting result is that we could achieve encryption rates beyond 400 Mbit/s using a standard Xilinx FPGA. This result is by a factor of about 30 faster than software implementations while we are still maintaining flexibility. A DES cracker chip based on this design could search 6.29 million keys per second.

1 Introduction

Most modern security protocols are defined algorithm independent and the protocol standards support a variety of algorithms. Although it is fairly easy to switch cryptographic algorithms in software, it is often painfully difficult in hardware. On the other hand, hardware solutions provide better performance and higher physical security. One answer to this problem is reconfigurable hardware, based on modern Field Programmable Gate Array, or FPGA, devices. Since FPGAs can switch algorithms they can be used to build *algorithm agile* applications. Although at a given time only one algorithm is configured, the FPGA can be reconfigured with a different algorithm. The following lists the main advantages of cryptographic algorithms on FPGAs

* This research was partially sponsored through NFS CAREER award CCR-9733246
** currently with GTE CyberTrust Solutions Incorporated

S. Tavares and H. Meijer (Eds.): SAC'98, LNCS 1556, pp. 234–247, 1999.

- **Algorithm agility**, the same FPGA can be reprogrammed at run time to support different algorithms,
- **Scalable security**, through different versions of the same algorithm (e.g., Data Encryption Standard (DES) and triple-DES),
- **Alterable architecture parameters**, e.g., desirable features such as variable S-boxes, variable number of rounds, or different modes of operation can easily be realized,

Another interesting cryptographic application of FPGAs are key-search machines. Building a key-search machine in hardware is a major investment. Such a machine can be used to retrieve secrete keys for only one algorithm. Therefore it might be interesting to have a key-search machine which is also defined algorithm independent, supporting a variety of algorithms.

We designed a universal key-search machine and used a high speed DES implementation for FPGAs as an algorithm. DES is currently the most widely used private-key algorithm and it is also part of many other standards, e.g., IPSec protocols, ATM cell encryption, the Secure Socket Layer (SSL) protocol, and for various ANSI banking standards. Even though it is expected that DES will not be reapproved as a federal US standard this year, it is still important and will continue to play a major role for several years to come.

In Sect.2 we summarize previous relevant work. Section 3 explores different architecture options for DES like loop unrolling and pipelining and presents the architecture versions we decided to implement. In Sect. 4 we provide an overview of the projected universal key-search machine. Section 5 describes the design and implementation cycle and gives an overview of the hardware and software tools we used for our research. In Sect. 6 we present and compare the results of our implementations of the different architectures and extrapolate cost and speed of the proposed universal key-search machine running a DES Cracker chip.

2 Previous Work

Already one year after the Data Encryption Standard was released in 1976, Whitfield Diffie and Martin Hellman published a paper which describes in detail a key-search machine for DES [2]. They estimated that for $20 million a key-search machine could be built which recoveres a DES key within 12 hours. Two papers with quite different results appeared in 1993 [6] and [13]. Both papers describe a custom chip design and develop from there a key-search machine. Reference [13] calculates 3.5 hours time to break DES at $1 million. Their chip can test 50 million keys per second. The design shown in [6] uses 32 DES breaker (DESB) chips and breaks DES in 1.2 days, each DESB chip can test 5333 million keys per second. This number appears to be very high. The estimated cost including overhead is only $2500 which appears to be very low to us.

Modern custom hardware implementations of DES can achieve data rates of 1 Gbit/sec and beyond. References [4,3] were the first report of a custom chip employing modern GaAs technology to achieve 1 Gbit/sec. This design as well as

many commercially available devices do not support a key change at full speed; i.e., a different key for every block of plain text.

The first paper to show an implementation of DES on FPGAs is [8]. Their approach generates key-specific circuitry for the Xilinx FPGAs. Therefore a binary image (bit-stream) for each key has to be precomputed before it can be used in the device. As for a key search machine the key has to be changed after every encryption, this chip is not suitable for this task. Their fastest implementation without decryption and adjusted to one key achieves a data rate of 26 Mbit/sec.

A group of cryptographers analyzed in 1996 the security of the key lengths of symetric ciphers based on current technology. In their report [1] employing FPGAs is highlited as an efficient approach for a brute-force attack against cryptographic systems. FPGAs are inexpensive, fast, and they need less inital investment than Application-Specific Integrated Circuits (ASIC). If we just take the plain FPGA cost for our design into account we would neet to invest three times as much money as [1] to recover keys at the same speed.

3 Architecture Options for the DES Algorithm

A high speed implementation of DES is crucial for an efficient DES cracker. But not only speed is an important factor, the design also must support a fast key change.

3.1 Basic DES

The DES algorithm possesses an iterative structure [10]. Data is passed through the Feistel network 16 times, each time with a different sub-key from the key transformation. This structure leads itself naturally to the block diagram shown in Fig. 1. The incoming data and key are passed through initial permutations. Then the data passes 16 times through the Feistel network and also 16 sub-keys are generated simultaneously. Both, the Feistel network operation and the sub-key generation is denoted in the block diagram as *Combinatorial Logic* (CLU, combinatorial logic unit). In order to be able to loop the output back to the input of the combinatorial logic unit we need *Registers* and *Multiplexers*. The multiplexer switches the input of the combinatorial logic unit between data from the previous round and new input data and key. The registers store the results of each loop and pass them on to the multiplexer. The output of the data register passes through the *Final Permutation*.

3.2 Loop Unrolling

Loop unrolling is the concatenation of combinatorial units in order to reduce the number of iterations if, for instance, two loops are unrolled then two rounds of DES will be calculated with one clock cycle. Figure 2 shows the block diagram. This block diagram differs from Fig. 1 only in the 2nd combinatorial logic unit.

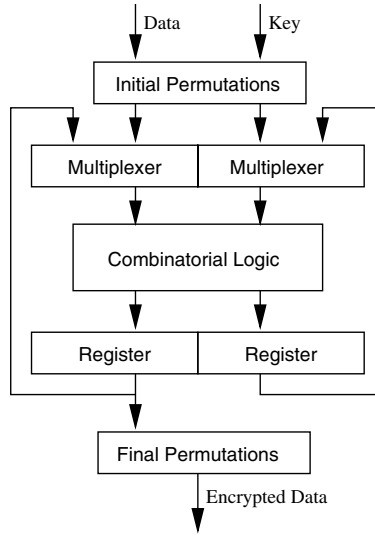

Fig. 1. DES block diagram

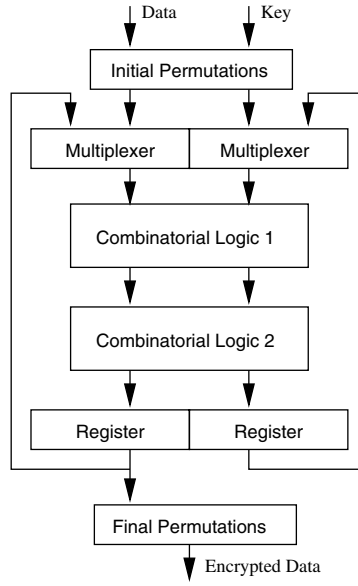

Fig. 2. Block diagram of DES with two unrolled loops

The initial and final permutations as well as the registers and multiplexers are the same.

Loop unrolling leads potentially to speed improvements for the following reason. In the not unrolled version, one iteration of DES has the following simple timing model: $T_{\text{mux}} + T_{\text{cl}} + T_{\text{reg}}$, where T_{mux} denotes the time a signal needs to pass through a multiplexer, T_{cl} the delay introduced by the combinatorial logic, and T_{reg} the delay introduced by the register. Wiring delays are assumed to be included in the specified times for the elements. So for the whole 16 rounds this sums up to: $16 * T_{\text{mux}} + 16 * T_{\text{cl}} + 16 * T_{\text{reg}}$.

The equation for one loop of the structure in Fig. 2 is thus: $T_{\text{mux}} + 2 * T_{\text{cl}} + T_{\text{reg}}$. This has to be executed 8 times, so that the over-all delay is now: $8 * T_{\text{mux}} + 16 * T_{\text{cl}} + 8 * T_{\text{reg}}$. The same principle can be applied to four unrolled DES rounds resulting in: $4 * T_{\text{mux}} + 16 * T_{\text{cl}} + 4 * T_{\text{reg}}$.

Obviously we can not reduce the delay introduced by the combinatorial logic units but we reduced the runs through the multiplexers and buffers. There is also another motivation for speed increase if modern design methods are applied. It is possible that the synthesis tools can optimize an unrolled design better, due to boundary optimization. Loop unrolling works thus well with a modern design process which can reduce the logic complexity and delay of the design.

3.3 Pipelining

Pipelining tries to achieve a speed improvement in a different way. Instead of processing one block of data at a time, a pipelined design can process two or more data blocks simultaneously. A design with two pipelines is shown in Fig. 3. The block diagram in Fig. 3 is very similar to the one with the two unrolled loops (Fig. 2). The only difference is the additional buffer between the combinatorial logic units.

The first block of data x_1 and the associated key k_1 are loaded and passed through the initial permutations and the multiplexer. The 1st combinatorial logic unit computes $x_{1,1}$ and $k_{1,1}$ which are stored into the 1st register block. On the next clock cycle $x_{1,1}$ and $k_{1,1}$ leave the 1st registers and the 2nd combinatorial logic unit computes $x_{1,2}$ and $k_{1,2}$ which are put into the 2nd register block. At the same time the second block of data x_2 and the key k_2 are loaded and passed through the initial permutations, and the multiplexer, and the 1st combinatorial unit computes $x_{2,1}$ and $k_{2,1}$ which are moved into the 1st register block.

Now the pipeline is filled and with each clock cycle another iteration for two pairs of data and key are computed. The data which has entered the pipeline first, will also exit it first. At that time the next data and key pair can be loaded.

The advantage of this design is that two or more data–key pairs can be worked upon at the same time. As there is still only one instance of the initial permutations, the multiplexer and the final permutation, the cost in terms of resources on the chip will not be twice as high as if we implemented two full non-pipelined DES designs. Also there has to be only one control logic which is just slightly more complicated than for a non-pipelined DES design. The maximum clock speed should be roughly the same since during one clock cycle the same amount of logic resources has to be traversed as in the non-pipelined design. It

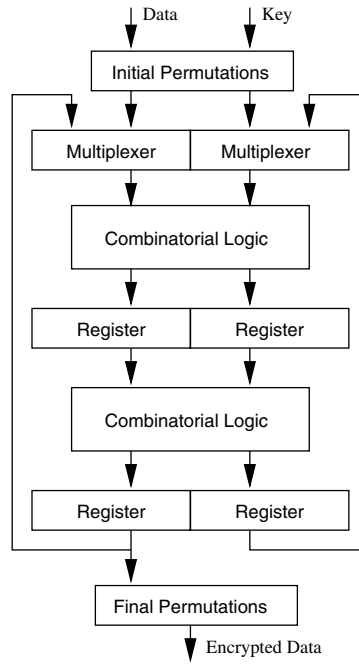

Fig. 3. Block diagram of DES with two pipeline stages

is also straightforward to design pipelines with more than two stages, e.g., with four.

3.4 Combination of Pipelining and Loop Unrolling

It is possible to combine both architecture acceleration techniques that we described above. For instance, each pipeline stage can contain two unrolled loops. The resulting block diagram looks similar to Fig. 3 except that each combinatorial logic unit is duplicated. During one clock cycle two iterations of two data–key pairs are computed: $x_{1,4}$ and $k_{1,4}$ are computed from $x_{1,2}$ and $k_{1,2}$, and $x_{2,2}$ and $k_{2,2}$ are computed from x_2 and k_2. An extension to four unrolled loops per pipeline stage is also possible.

3.5 Design Decisions

One major objective of this research was to obtain a realistic comparison of the different acceleration methods (loop unrolling, pipelining, combination of both) for FPGAs. Table 1 shows the architecture versions we decided to implement, results of which are described in Section 6.

Table 1. Implemented DES architectures

Name	Description
DES_ED16	standard DES (16 iterations)
DES_ED8	DES with 2 unrolled loops (8 iterations)
DES_ED4	DES with 4 unrolled loops (4 iterations)
DES_ED16x2	DES with 2 pipeline stages
DES_ED16x4	DES with 4 pipeline stages
DES_ED8x2	DES with 2 pipeline stages each containing 2 unrolled loops

4 Universal Key-Search Machine

The design of the Universal Key-Search Machine has to be algorithm independent. Therefore no assumption on the length of the plain text, cipher text, or key is made. The system architecture is described in the following subsection and the modifications necessary for our DES FPGA [7] to meet the I/O requirements for this architecture in the next subsection.

4.1 System Architecture

Our biggest concern is scalability. Therefore one design goal is to have a minimum amount of wiring and additional external logic (glue logic). A bus topology and an optimized addressing scheme was developed to meet this requirement. The addressing scheme requires no address decoding logic and there is no logical limit on the number of key-search chips connected to the bus.

Figure 4 gives an overview of the system architecture. When more key-search chips are attached additional buffers are needed to drive the bus. These bus drivers are left out of Fig. 4 for simplicity. The cracker-bus comprises an bidirectional 16 bit wide data bus, the control signals, **IE** (input enable) and **OE** (output enable), four chip select signals, **CS** (chip select), **CSI** (chip select in), **CSO** (chip select out), and **CSCLK** (chip select clock), and an indicator **done**. The function of the control chip is to interface the cracker-bus with the ISA bus of a PC. We use DESC, the DES Cracker chip, as an example.

The system supports sequential as well as parallel access. When *CS* is active all chips are selected and listening to the bus. The plain text and the cipher text can be send with *IE* to every chip at the same time. A different start key has to be sent to each chip. For this purpose we implemented an sequential addressing scheme. The signals *CSI*, *CSO*, and *CSCLK* work with the chips as a long shift register. The control chip sends a logic '1' to the first chip. With each clock *CSCLK* this '1' is passed from *CSO* to *CSI* of the next chip; the next chip is selected and the start key can be send with *IE*. The number of clock cycles *CSCLK* it takes for the '1' to come back to the control chip is equal to the number of key-search chips put in. With *OE* the current key of the selected chip is clocked out. With this function the state of the key-search machine can be saved or the final key can be retrieved. The *done* signal is a tristated signal

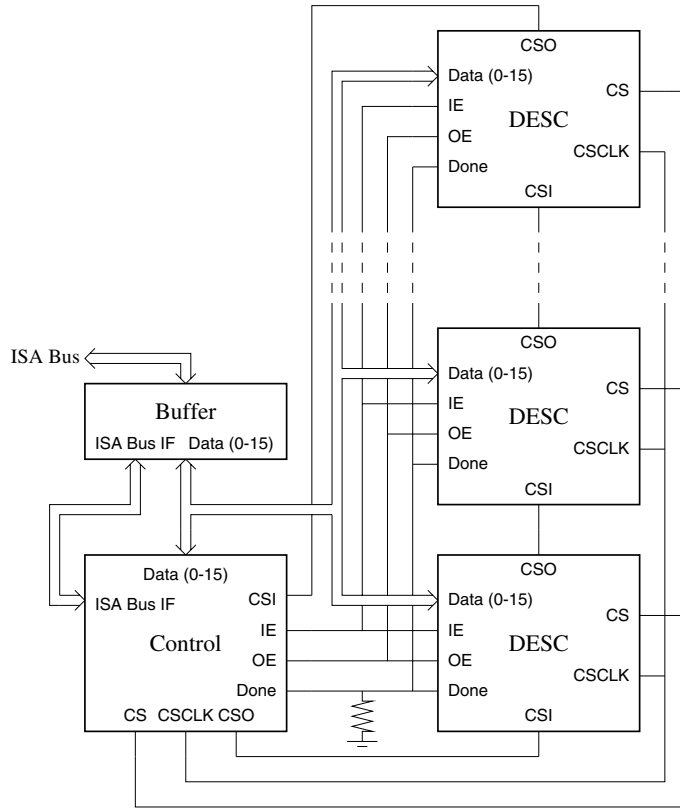

Fig. 4. System architecture

pulled to logic '0' with a resistor. When the key is found this signal goes to '1'. Now the control chip addresses sequentially the chips. The chip which found the key takes the '1' off as soon as it is selected. Through this scheme the controller can identify the chip.

4.2 DES Cracker-Chip

The DES cracker chip (see Fig. 5) is based on our DES FPGA [7] denoted as *DES Core* in the Figure. Each of the implemented DES architectures shown in Table 1 can be used as a DES Core. Our DES FPGA is not optimized for key-search but it supports a key change for every block of plain text at full speed. The buffers for the cipher text and the plain text are 64 bits wide, the buffer for the start key 56 bits. The bus can transfer 16 bits at a time, therefore the buffers are split up in parts of maximum 16 bits. They are addressed through a shift register (not shown in Fig. 5). When the chip is selected the signal *IE* cycles through the input buffers. The 4×16 bits output register for the current

key is addressed via a 4 bits shift register, which in turn is operated through the *OE* signal when the chip is selected.

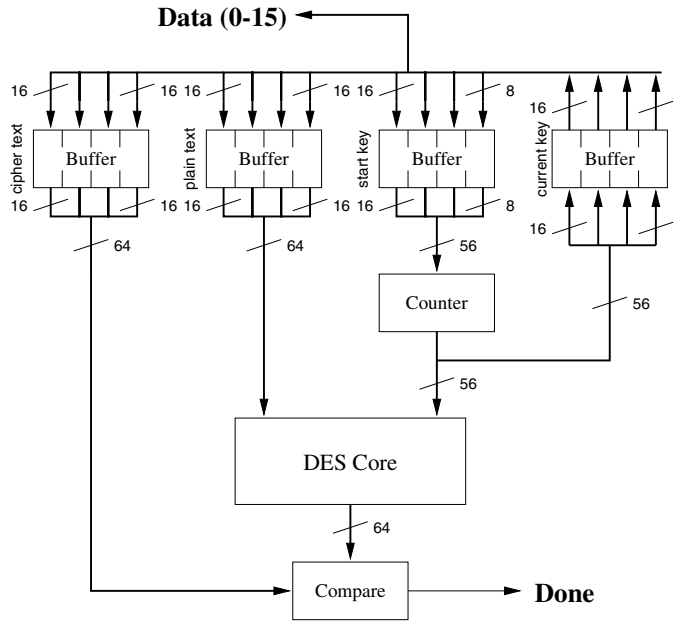

Fig. 5. DES Cracker-Chip

A 56 bit counter takes care of incrementing the key for each new encryption. A 56 bit compare unit compares the encrypted plain text with the cipher text. If both are the same the *done* signal goes high. The chip select circuitry and the control logic is not shown in the Figure.

5 Methodology

This section describes the design cycle and the tools used for our research of the DES core. The design procedure can roughly be broken down into the following stages:

1. Creating VHDL[1] descriptions of the DES design employing different architecture options and verifying each version.
2. Synthesis and logic optimization.

[1] VHDL is the VHSIC Hardware Description Language. VHSIC is an abbreviation for Very High Speed Integrated Circuit.

3. Place and Route for a specific device and back-annotated verification of the design.

Early in the design we decided upon Xilinx as FPGA vendor and a device family. That decision was based mainly on our previous work described in [5]. The entire design was implemented using VHDL and vendor specific macros. We used Synopsys version 1997.08 for the synthesis, logic optimization, design verification, and timing analysis.

Xilinx Alliance Series version M1.3.7 provided macro functions (LogiBLOX) and was used for the place and route. The Xilinx tools also perform a timing analysis after place and route which shows the **minimum clock period** for the given design. This clock period is guaranteed by Xilinx for the design and therefore is to be seen as rather pessimistic. We are using this timing result for our speed calculations.

6 Results

We implemented multiple versions of each architecture option listed in Table 1 in order to evaluate their effectiveness. In the following sections we compare the different designs. In most cases the designs are compared to the design *DES_ED16* which serves as our reference model.

The unit **CLB** stands for *combinatorial logic block* which is employed by Xilinx to measure the amount of logic resources on a device. We are using it here to compare the amount of logic resources used by a given design. The abbreviation **CLU** stands for *combinatorial logic unit* (see Chap. 3), i.e., one Feistel network round.

6.1 Loop Unrolling

We implemented two loop unrolled versions: *DES_ED8* and *DES_ED4*. The design *DES_ED8* contains two combinatorial logic units (*CLU*, see Sect. 3.2) and therefore encrypts or decrypts one data block in 8 clock cycles. The design *DES_ED4* contains four CLUs and provides the result after 4 clock cycles. Both designs are compared with the design *DES_ED16* in Table 2.

Table 2. Comparison of loop unrolled architectures

Design	**Chip**	**CLBs** used	**CLBs** per CLU	**Min CLK** in ns	**Data Rate** per CLU in Mbit/s	**Data Rate** in Mbit/s	rel.
DES_ED16	XC4008E-3-PG223	262	262	40.4	99.1	**99.1**	1.00
DES_ED8	XC4013E-3-PG223	443	222	54.0	74.1	**148.2**	1.50
DES_ED4	XC4028EX-3-PG299	722	241	86.7	46.1	**184.5**	1.86

Our standard DES design *DES_ED16* allowed already for data rates of 99.1 Mbit/sec[2]. The design *DES_ED8* is with 148.2 Mbit/sec 50% faster than *DES_ED16* whereas the resource consumption (in CLBs) increases by 69%. The design *DES_ED4* is with 184.5 Mbit/sec only 25% faster than *DES_ED8*, the speed increase is only half as much as from the first unrolling. The resource consumption increases by 63%.

The number of CLBs divided by the number of CLUs indicates that the amount of logic resources consumed per unrolled CLU is almost constant. The speed divided by the number of CLUs shows that the speed for one CLU in the design *DES_ED4* is less then half the speed of *DES_ED16*. From this we can see that loop unrolling shows a non-linear speed improvement. It seems unlikely that a fourth loop unrolling would yield significantly higher performance.

6.2 Pipelining

We implemented two pipelined designs, *DES_ED16x2* and *DES_ED16x4*. The design *DES_ED16x2* contains two CLUs and therefore 2 pipeline stages and the design *DES_ED16x4* contains four CLUs and therefore 4 pipeline stages. The encryption or decryption of one block of data takes in both cases 16 clock cycles. Table 3 compares both designs with the design *DES_ED16*.

Table 3. Comparison of pipelined architectures

Design	**Chip**	**CLBs** used	**CLBs** per CLU	**Min CLK** in ns	**Data Rate** per CLU in Mbit/s	**Data Rate** in Mbit/s	rel.
DES_ED16	XC4008E-3-PG223	262	262	40.4	99.1	**99.1**	1.00
DES_ED16x2	XC4013E-3-PG223	433	217	43.5	91.9	**183.8**	1.86
DES_ED16x4	XC4028EX-3-PG299	741	185	39.7	100.7	**402.7**	4.06

The design *DES_ED16x2* with two pipeline stages is with 183.8 Mbit/sec almost twice as fast as our reference design *DES_ED16*. And our design *DES_ED16x4* achieves 402.7 Mbit/sec which is four times faster. The speed divided by the number of CLUs shows that the speed per CLU stays almost constant for all designs. The lower speed for the design *DES_ED16x2* is caused by the lack of wiring resources on the device which results in a less efficient design.

The amount of logic resources consumed per implemented CLB is decreasing if we create more pipelines. This is due to the fact that the control unit does not become more complicated if we implement more pipelines. Also the multiplexers are implemented only once.

It is interesting to compare the pipelined designs with the loop unrolled designs. It can be seen that *DES_ED16x2* is both faster and smaller than the loop

[2] Mbit= 10^6bit

unrolled *DES_ED8*. The difference is even more dramatically if the *DES_ED16x4* is compared with the *DES_ED4*. *DES_ED16* is more than twice as fast as *DES_ED4* and utilizes almost the same amount of CLBs.

6.3 Combination of Pipelining and Loop Unrolling

A design that contains loop unrolling as well as pipelining is in the simplest version already as large as the largest designs we have implemented so far which were *DES_ED16x4* and *DES_ED4*. Therefore we implemented only the design *DES_ED8x2* which contains 4 CLUs; 2 in each of the 2 pipeline stages. Table 4 compares this design with *DES_ED16x2* and *DES_ED8*.

Table 4. Comparison of a combined architecture with others

Design	**Chip**	**CLBs**	**Min CLK** in ns	**Data Rate** p. pipeline in Mbit/s	**Data Rate** in Mbit/s
DES_ED8x2	XC4028EX-3-PG299	733	48.0	166.5	**333.0**
DES_ED16x2	XC4013E-3-PG223	433	43.5	91.9	**183.8**
DES_ED8	XC4013E-3-PG223	443	54.0	148.2	**148.2**

It is not easy to compare this mixed design with the two other designs. The minimum clock period shows that the time it takes for two CLUs (loop unrolled) to execute in the design *DES_ED8x2* is faster than in the design *DES_ED8*. It is of course slower, but surprisingly not much, than one CLU in the design *DES_16x2*.

6.4 Extrapolation for Cracker

Our fastest design is DES_ED16x4 with a data rate of 402 Mbit/sec. This design could be the *DES Core* for our *DES Cracker-Chip* (DESC) as shown in Sect. 4.2. After the plain text, cipher text and start-key are loaded into the buffers the DES Core can start working at full speed. We expect that we can achieve the same data rate for the DESC chip as we achieve with DES_ED16x4. The data rate of 402 Mbit/sec corresponds to a key-search rate of 6.29 million keys per second. This means that a brute force attack on DES with 2^{56} keys would take 182 years on average. A fully unrolled design would theoretically search 25 million keys per second which is about half the speed of the key-search ASIC introduced in [13].

A cracker box comprising one control board, 16 boards each with four DESC chips (using DES_ED16x4 as the DES core) connected to a PC could search the entire key space of DES with a 40-bit key in one hour. The FPGA costs alone for such a box are about $18.000[3]; when we include 50% overhead for the other

[3] Based on actual current (May 1998) pricing supplied by a vendor.

required circuits, boards, racks and an additional $1000 for the PC (a low end PC is fast enough) this cracker box would amount to cost $28.000.

Assume that eight such boxes could be attached to one PC and 160 PCs are connected via a network. This machine comprises 81,920 DESC chips and would cost about $35 million. It could find a DES key of 2^{56} bits length in one day on average. We want to notice that the prices for FPGAs are decreasing at a very high rate and that a key search machine at a considerable lower price might be feasible in the near future.

7 Comparison

We implemented all DES designs based on standard devices from Xilinx with a medium speed grade. With these devices we achieved speeds of up to 402.0 Mbit/sec using four pipelined stages and no loop unrolling. If we compare the reported DES speeds for high speed ASICs (1600 Mbit/sec) [11] and high speed software (13 Mbit/sec) [11], with our best result of 402.0 Mbit/sec we conclude that the speed-up factor from software to FPGAs is about 31, and from FPGAs to ASICs is about 4. It is difficult to provide a fair comparison to this respect but our results clearly show that FPGAs implementations of DES are very attractive for many applications. If we compare our result with the one from [8] we see that even though the same device (although at a slower speed grade) was used, the fastest implementation in the reference is by factor three slower and requires almost twice as much logic resources as our *DES_ED16*. It is to be noted that the fastest implementation in the reference is adjusted to one key.

Using our fastest design as a DES core for the DESC chip we can achieve a key-search rate of 6.29 million keys. For $38 million a machine can be built that breaks DES with 2^{56} keys in one day on average. Although it might very well be possible to build a DES-specific key search machine (much) cheaper, we would like to note that our design allows the application to a wide variety of block ciphers by simply downloading a different algorithm in to the FPGAs. Considering the significant cost of a key-search machine, this can be a major advantage if more than one algorithm is to be attacked.

References

1. M. Blaze, W. Diffie, R.L. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener. Minimal key lengths for symmetric ciphers to provide adequate commercial security. January 1996.
2. W. Diffie and M.E. Hellman. Exhaustive cryptanalysis of the NBS Data Encryption Standard. *Computer*, 10:74–84, June 1977.
3. H. Eberle. A high-speed DES implementation for network applications. In E.F. Brickell, editor, *Advances in Cryptology - CRYPTO '92. 12th Anual International Cryptology Conference Proceedings*, Lecture Notes in Computer Science, pages 521–539, Berlin, Germany, 1993. Springer-Verlag.

4. H. Eberle and C.P. Thacker. A 1 Gbit/second GaAs DES chip. In *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, pages 19.7/1–4, New York, NY, USA, 1992. IEEE, IEEE.
5. G.M. Haskins. Securing Asynchronous Tranfer Mode networks. Masters thesis, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, May 1997.
6. F. Hendessi and M. Aref. A successful attack against the DES. In T.A. Gulliver and N.P. Secord, editors, *Information Theory and Applications : Proceedings Third Canadian Workshop*, volume 793 of *Lecture Notes in Computer Science*, pages 78–90, Berlin, 1994. Springer Verlag.
7. Jens-Peter Kaps. High speed FPGA architectures for the Data Encryption Standard. Masters thesis, Worcester Polytechnic Institute, Worcester, Massachusetts, USA, May 1998.
8. J. Leonard and W.H. Magione-Smith. A case study of partially evaluated hardware circuits: Keyspecific DES. In W. Luk, P.Y.K. Cheung, and M. Glesner, editors, *Field-programmable Logic and Applications. 7th International Workshop, FPL '97*, Berlin, Germany, 1997. Springer-Verlag.
9. A.J. Menezes, S.A. Vanstone, and P.C. Van Oorschot. *Handbook of Applied Cryptography.* Discrete Mathematics and its Application. CRC Press, Florida, USA, 1997.
10. National Bureau of Standards FIPS Publication 46. *DES modes of operation*, 1977.
11. B. Schneier. *Applied Cryptography Second Edition: protocols, algorithms, and source code in C.* Wiley & Sons, New York, USA, 2nd edition, 1996.
12. D.R. Stinson. *Cryptography: Theory and Practice.* Discrete Mathematics and its Applications. CRC Press, Florida, USA, 1995.
13. M.J. Wiener. Efficient DES key search. Crypto '93 Rump Session Presentation, August 1993.