The CSI Journal on
Computer Science and Engineering
Vol. 5, No. 2&4 (b), 2007
Pages 9-15
Regular Paper

A Reconfigurable Architecture for Implementing Multiple Cipher Algorithms

Ali Valizadeh[1] Morteza Saheb Zamani[2] Babak Sadeghiyan[2] Farhad Mehdipour[3]

[1]Department of Computer Engineering, Islamic Azad University, Shahryar-Shahr-e-Qods Branch, Tehran, Iran
[2]Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran
[3]Faculty of Information Science and Electrical Engineering, Department of Informatics, Kyushu University, Fukuoka, Japan

Abstract

Reconfigurable computing offers advantages over traditional hardware and software implementation of computational algorithms. It is based on using reprogrammable devices, which can be reconfigured after fabrication to implement the desired algorithm. Reconfigurable computing systems can take advantage of hardware but still maintains the flexibility of software. Particular applications, including encryption, are specifically well suited to these systems. In this paper, we implemented multiple cryptographic algorithms, namely DES, LOKI, DESX, Biham-DES, and S^nDES on a reconfigurable hardware so that each algorithm could be replaced by another with low reconfiguration overhead time. The hardware reconfiguration time is about 2.6 ms, which is comparable with the other systems overhead. The main reason for the lower reconfiguration time is that our architecture is partially reconfigurable. Our implementation resulted in a high flexibility but comparable ciphering rate in comparison with previous implementations on field programmable gate arrays (FPGAs). We achieved the ciphering rate of 14080 Mbps on Xilinx 2V6000-5 FPGA.

Keywords: Reconfigurable Hardware, FPGA, Cryptography.

1. Introduction

The importance of cryptosystems has been continuously increased in data communications. In a cryptosystem, a plaintext is encrypted to guarantee confidentiality. Many different algorithms have been proposed for cryptography. Some examples are DES [1], FEAL [2], and LOKI [3]. For some applications, which need high data rate, it is necessary to use high-speed cryptographic algorithms implemented in hardware. Implementations of cryptographic algorithms in hardware have several advantages over software implementations, including speedup and security. Typical hardware implementations are application specific integrated circuits (ASIC) and full-custom. In either case, the hardware is fixed at the time of manufacturing.

Although traditional hardware implementations have no flexibility in the algorithm and parameter switching, reconfigurable hardware devices offer a promising alternative for the implementation of block ciphers. One of the potential advantages of block ciphers implemented in reconfigurable hardware is the capability of switching between cryptographic algorithms during operation. The majority of modern security protocols, such as Secure Sockets Layer (SSL) and IPsec, allow for multiple encryption algorithms which are selected in the negotiation phase on a per-session basis. Although switching between algorithms can be very costly with traditional hardware, using reconfigurable hardware appears to be an attractive alternative [10], [14], [15].

Moreover, some applications require modification of a standardized algorithm, e.g. by using proprietary S-Boxes or permutations. Such modifications can easily be made by reconfigurable hardware. Although reconfigurable hardware may typically be slower than ASIC implementations, it has the potential of running substantially faster than software implementations. On the other hand, the time and costs for developing a reconfigurable implementation of a given algorithm are much less than an ASIC implementation. Although for high-volume applications, ASIC solutions may become the more cost-efficient choice.

Reconfigurable computing systems (RCS) were proposed in the late 1960s at UCLA [18]. Advances in the programmable hardware capacity and architecture have resulted in the introduction of reconfigurable computing systems [18], [20]. Such systems usually consist of a host processor connected to a reconfigurable hardware like field programmable gate arrays [19], [20]. RCS has demonstrated the potential to achieve high performance implementations in a wide range of applications, such as image processing, cryptography, target recognition and digital signal processing because of the availability of high-density devices with flexible and relatively fast architectures [21], [23].

One of the important features for enabling the implementation of the RCS is the availability of high-density VLSI devices that use programmable switches to implement flexible hardware architectures. These devices contain memory cells that hold both reconfiguration information for the programmable switches and state information for active computations. In an RCS, downloading the bitstream for changing the configuration of the programmable hardware can be done either statically or dynamically. In the static version, the configurations are generated as bitstreams and saved in an external memory and then are loaded one by one during execution. On the other hand, in the dynamic or online version, the configuration of the programmable hardware changes on the fly. In this version, reconfiguration overhead is a critical factor which affects the overall performance of the system.

In this paper, we propose a reconfigurable architecture to implement multiple cipher algorithms. The cryptographic algorithms are implemented on a reconfigurable hardware so that each algorithm can be replaced by another with the goal of reducing reconfiguration overhead time. In our architecture, the similarities between the algorithms is detected and implemented in the hardware. The common parts are fixed in the hardware and only the different parts are reconfigured. Therefore, our architecture is partially reconfigurable and that is why it has a small reconfiguration overhead time.

For our architecture, we selected DES, LOKI, DESX, Biham-DES, and S^{n}DES cipher algorithms. The proposed architecture was implemented on Xilinx 2V6000-5 FPGA device which resulted in a flexible hardware with a low reconfiguration overhead time. Moreover, the ciphering rate of the proposed hardware is comparable with other implementation of DES algorithm on FPGA. It also takes benefits of the partially reconfigurability.

To introduce our architecture, we explain some previous implementations in Section 2. In Sec. 3, the selected algorithms are described and in Sec. 4, our reconfigurable architecture is proposed. Our experimental results and the conclusion are presented in Sec. 5 and Sec. 6, respectively.

2. Previous Work

In this section, we review some previous hardware implementations of the selected algorithms. Since DESX, S^{n}DES, and Biham-DES are similar to DES algorithm, In Section 3, we will show that the throughput obtained for DES in the previous implementations can give an estimation for the throughput of the above mentioned algorithms.

The first published implementation of DES on an FPGA achieved a throughput of 26.4 Mbps [16]. However, this implementation required generation of key-specific circuitry for the target Xilinx XC4013-4, requiring recompilation of the implementation for each key.

Another implementation of DES on an FPGA achieved a throughput of 402.7 Mbps when operating in Electronic Code Book (ECB) mode using a Xilinx XC4028EX-3 FPGA [17]. This implementation employed pipeline design techniques to maximize the system clock frequency at the cost of pipeline latency cycles.

With the advent of run-time reconfiguration and more technologically advanced FPGAs, implementations with throughputs in the range of ASIC performance have been achieved. One of these implementations employing run-time reconfiguration achieved a throughput of 10.752 Gbps when operating in ECB mode using a Xilinx Virtex XCV150-6 FPGA [10].

Using Xilinx run-time reconfiguration software application, called JBits, allowed for real-time key-specific compilation of the bit-stream to program the FPGA, resulting in a smaller and faster design (which operated at 168 MHz) as compared to the design in [17] (which operated at 25.19 MHz).

Another implementation on FPGA reported a throughout of 12 Gbps when operating in ECB mode using a Xilinx Virtex-E XCV300E-8 FPGA [24] but it did not make use of run-time reconfiguration.

Xilinx performed another implementation [11] which unrolled and pipelined DES. The 48-stage pipelined version was able to get a throughput of 15.1 Gbps on Xilinx Virtex-II XC2V1000-5 device. It also allowed for changing the plaintext, the key and the Encryptor/Decryptor mode on a cycle by cycle basis.

In [12], a new mathematical DES description was presented that allows us to achieve an optimized implementation in terms of *throughput*/*area* ratio. The implementation unrolls DES algorithm and implements two versions of the algorithm: a 21-stage and a 37-stage one. With the 37-stage implementation, the throughput was 21.3 Gbps (333 MHz) on Xilinx XC2V1000-5. The implementation achieved throughput/area ratio of 7.18 compared with [11] that resulted in 3.87 for this ratio.

In an attempt to implement multiple block ciphers on a reconfigurable architecture, Elbirt and Paar developed COBRA architecture [15]. They studied a wide range of block ciphers to develop an understanding of the functional requirements of each algorithm. In their implementation, it was determined that a reconfigurable cryptographic processor core might support a 32-bit datapath replicated at least four times to allow for the implementation of algorithms requiring either 64-bit or 128-bit block length. Although this architecture was able to implement a wide range of block ciphers that meet the 622 Mbps ATM network encryption throughput requirements, there was one problem

with COBRA, i.e. it was not able to implement DES algorithm.

To the best of our knowledge, there is only one reported hardware implementation of LOKI91 algorithm. The implementation reported in [13] achieved the ciphering rate of 127 Mbps on Xilinx Virtex-E.

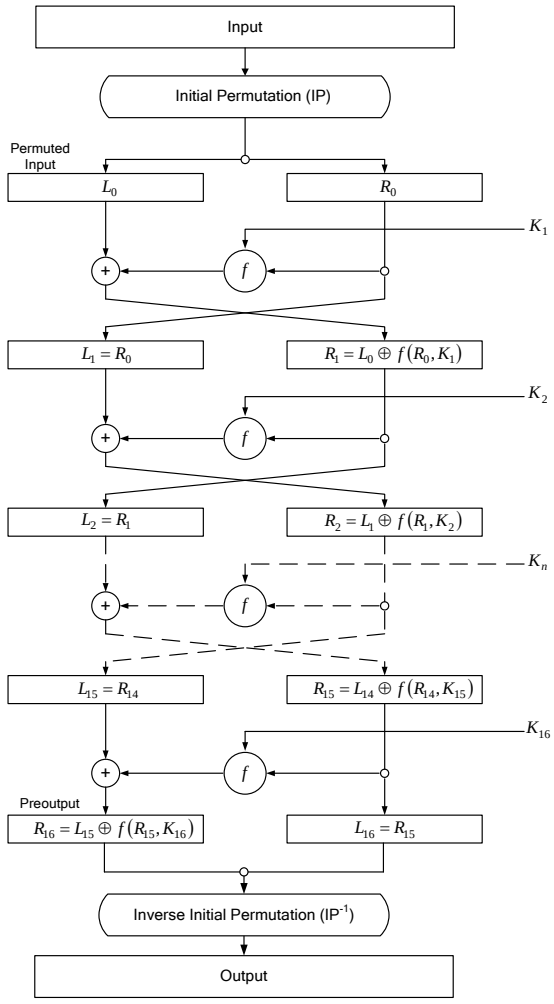

Figure 1. DES algorithm overview

3. Cryptography Algorithms

In this section, our selected cryptography algorithms are explained. DES (Data Encryption Standard) is a private and symmetric key algorithm, in which the same key is used for both encryption and decryption. A public key algorithm such as RSA uses different keys for encryption and decryption. Private key algorithms are generally much faster than public key algorithms. DES encryption algorithm was proposed in 1977. It encrypts 64-bit blocks with a 56-bit key in 16 rounds of iterations. Each round uses a different subset of key bits and consists of swapping the left and the right 32-bits, as well as permutation, substitution and XOR operations. The overall structure of DES is shown in Figure 1.

The F-function of DES algorithm is shown in Figure 2. In this figure, *E* and *P* represent expansion and permutation, respectively. S_1 to S_8 in this figure represent S-boxes, which are tables of constant values and have 6 inputs and 4 outputs. DES decryption process is identical to encryption but the sub-keys are generated in reverse order.

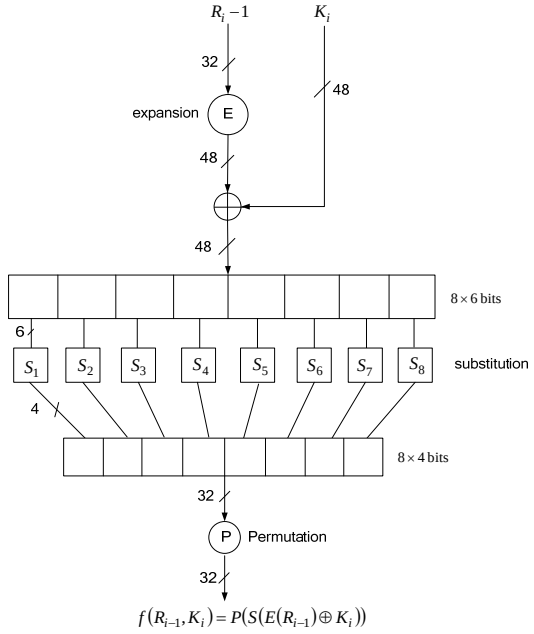

Figure 2. The F-function of DES algorithm

LOKI91 was proposed in [3]. The ciphers from the LOKI family are DES-like iterated block ciphers based on iterating a function, called F-function. The block and key size are 64-bits. The input to each round is divided into two halves. The right half is fed into the F-function together with a 32-bit round key derived from a key schedule algorithm. The output of the F-function is added (modulo 2) to the left half of the input and, as in DES, the two halves are interchanged except for the last round. The LOKI cipher runs 16 rounds. The input to the F-function is the result of XOR of a 32-bit input text and a 32-bit round key. The 32 bits of the result are expanded to 48 bits and divided into blocks of 12 bits. The 12-bit blocks are applied to the four S-boxes, each of which produces an 8-bit output. The 32 bits from the S-boxes are permuted to make the output of the F-function. As in DES, the decryption process is identical to encryption but the sub-keys are generated in reverse order.

DESX is a 64-bit block cipher with a 2*64+56 = 184-bit key and is a simple extension of DES [4]. It was proposed to overcome the problem of short 56-bit key size which made the cipher vulnerable to exhaustive key search attack. The idea is only to XOR a private 64-bit key K1 with the input of DES and to XOR another 64-bit private key K2 with the output of DES: $C = K2 \oplus DES_K(P \oplus K1)$

Comparing DES and DESX architectures shows that the implementation of these two algorithms only differs at the start and stop points of the algorithms. Therefore, for implementing DESX only a few xor-gates should be added to DES implementation. Therefore the critical paths of these two algorithms are approximately the same especially in pipelined implementations.

Some modifications have been proposed for increasing the robustness of DES against known attacks. One of the important considerations in DES is changing the contents of S-boxes. S^nDES is a variant of DES, which uses best possible alternative S-boxes [5], [6], [7], [8].

Since DES and S^nDES architectures, only differ in their S-boxes, the implementations of these two algorithms have

the same critical path. Hence, the previous hardware implementation results of DES can be supposed for S^{n}DES as well.

Another modification on DES is to changing the order of S-boxes. Biham and Birykov [9] suggested strengthening DES against exhaustive attacks by using extra key bits to modify F-function of DES slightly. One of their modifications uses 5 key bits to select from 32 possible reorderings of the 8 DES S-boxes.

The difference of Biham-DES and DES is in their S-boxes, but to this difference, we need multiplexers of size 32 to 1 to select between 32 reorderings of S-boxes in hardware which makes the critical path length longer. Therefore, the previous implementations of DES have rough approximations of Biham-DES. Although in our architecture, we use memories instead of S-boxes and program the memories in configuration time. Thus, in our implementation, DES and Biham-DES have the same critical path.

4. Our Reconfigurable Architecture

Before presenting our reconfigurable architecture, we look at the selected algorithms in more details. We pinpoint the similarities and differences between the algorithms and then propose our architecture.

All of the above algorithms have a Feistel structure with 16 rounds and their input block size of plaintext is 64-bit. In addition, the block size of the cipher key is 64-bit. First, the 64-bit plaintext is divided into two 32-bit blocks (L0, R0) and then, the encryption is performed by the F-function and round keys which are extracted from cipher key. The Feistel network algorithms may have different subkey generation methods and F-functions. We may assume that the subkey generation is performed by software. In this case, these algorithms are different only in their F-functions. In our proposed architecture, we maintained the Feistel structure and took advantage of reconfigurability of hardware for the F-functions.

Therefore, when our hardware runs an algorithm, e.g. DES, changing the encryption algorithm, e.g. to LOKI, can be done by only changing the F-function. In the following, we explain the details of F-function of each algorithm.

The F-function of DESX, S^{n}DES, and Biham-DES are rather similar to DES. For example, in S^{n}DES and Biham-DES algorithms, only the contents of S-boxes are different from DES S-boxes. Also, in DESX algorithm, there are only some extra XOR gates before the first round and after the final round. These algorithms, which are very similar to DES, are called DES-like algorithms.

One round of DES and LOKI algorithms are shown in Figure 3 and Figure 4, respectively. As shown in Figure 2, there are a number of XOR gates in the F-function of DES. For a given subkey, generated by a key-generation software, the XOR in the F-function can be eliminated and the effect can be applied to the S-box contents [10].

Similarly, the XORs, located before the first round of DESX, can also be eliminated and the effects can be applied to the first round S-boxes. Such an elimination and consideration of effects can also be done for the XOR's after the final round. In this way, switching between DES-like algorithms is performed by only replacing the S-box contents, according to the selected algorithm.

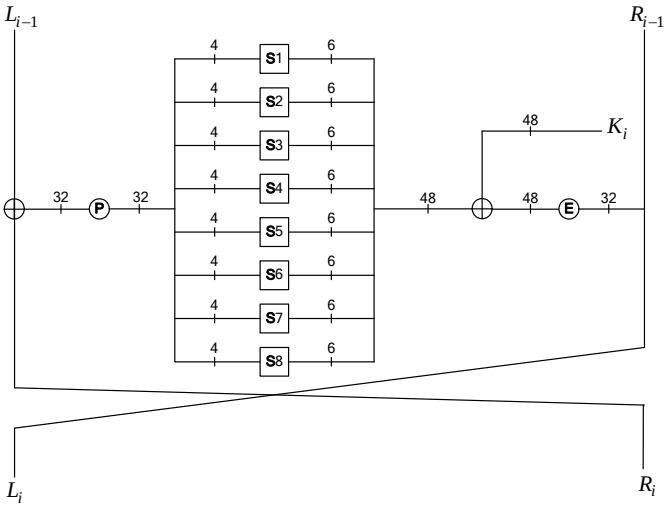

Figure 3. One round of DES algorithm

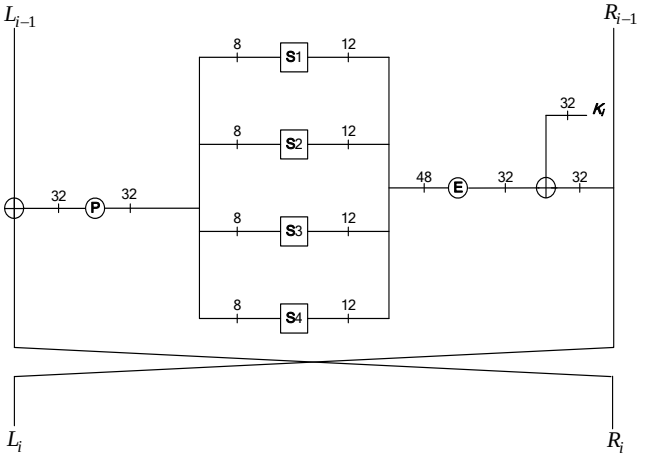

Figure 4. One round of LOKI algorithm

Our goal is to presenting a reconfigurable unified architecture for LOKI and DES. This architecture covers other DES-like algorithms as well. LOKI algorithm is more different than other algorithms. According to Figure 3 and Figure 4, the main differences between the F-functions of DES and LOKI are:

- subkeys in DES are 48 bits, but they are 32 bits in LOKI.
- DES S-boxes are 6x4, but they are 12x8 in LOKI.
- Expansion (E) and Permutation (P) functions are completely different.

The first difference between the two F-functions is not important because, as mentioned above, the subkeys are generated in software and consequently, XOR gates in the F-function can be eliminated in DES and LOKI. In addition, 12x8 S-box table can be considered an extension of 6x4 S-box table. In this case, the remaining $(2^{12}-2*2^6)*8$ bits of 12x8 S-box tables must be filled with appropriate values when they are used for 6x4 S-boxes. In each round, we have to use four 12x8 S-box tables to support both of DES and LOKI algorithms.

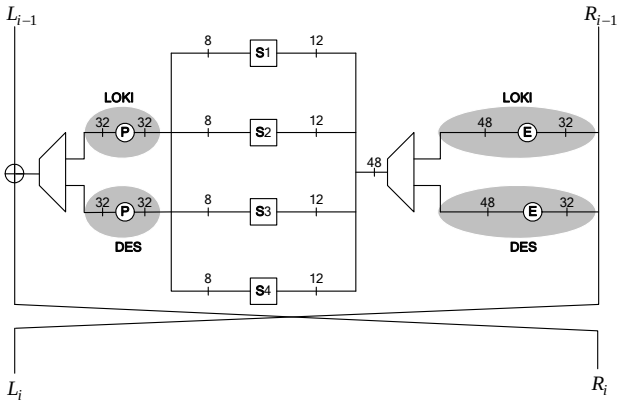

Figure 5. Selecting between DES and LOKI

The difference between E and P functions is the most considerable one. We propose using multiplexers, where both E and P functions for DES and LOKI are implemented in hardware, and the multiplexers select proper output (Figure 5). In this approach, only 32 or 48 multiplexers (2 to 1) are used and have a one-stage delay. As E and P functions can be implemented with wiring in hardware, there is no gate overhead for them.

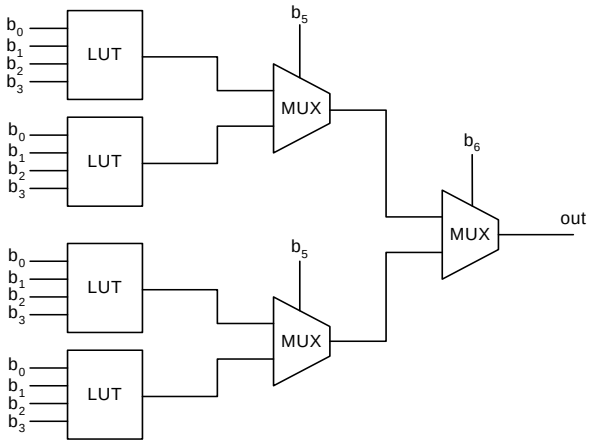

Figure 6. Implementation of a 6 to 1 S-box in Virtex FPGAs

4.1. S-Box Design

The design of S-boxes is another important part of our implementation. As mentioned earlier, we use 12x8 S-boxes in this architecture. While most FPGAs use 4 to 1 LUTs, the implementation of 12x8 S-boxes need a large number of LUTs. Figure 6 shows the implementation of a 6x1 S-box in a Xilinx Virtex FPGA, as an example. Therefore, large S-boxes implemented in FPGA have large delay and low performance.

Since LUTs are similar to memory cells and using them reduces the delay of combinational logic, the implementation of large S-boxes in memory reduces the logic and routing resources. Figure 7 shows one stage of our proposed reconfigurable architecture employing memories. Memories used for S-box implementation are single port and use two isolated data buses for inputs and outputs. A separate address and data line, which are common for all memories, are used for programming the chip. Programming a specific S-box (memory) can be performed by activating its *we* (write enable) line.

The proposed reconfigurable system has two different operational modes: reconfiguration mode and normal mode. In the reconfiguration mode, the contents of S-boxes (memories) are reconfigured according to the selected algorithm and in the normal mode, the text is encrypted/decrypted.

5. Experimental Results

The recent hardware implementations of DES in FPGAs, studied in Section 2, are given in Table 1. The last entry in the table represents our implementation. The table shows that these implementations have used different FPGA devices and have obtained different clock rates and throughputs. As DESX, S^{n}DES, and Biham-DES are rather similar to DES algorithm, the throughput obtained for DES in the previous implementations may give an estimate for the mentioned algorithms too (as aforementioned in Section 3).

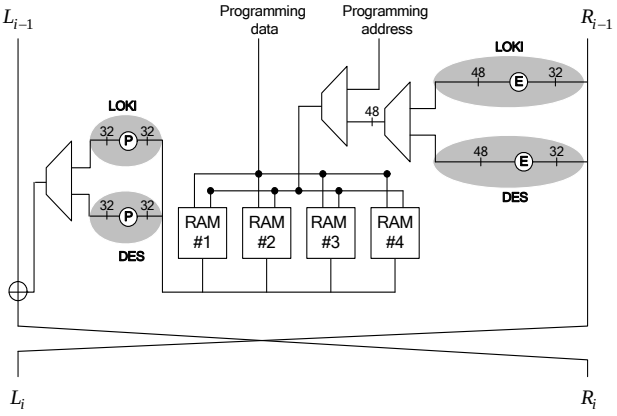

Figure 7. One round of the proposed reconfigurable architecture

The proposed architecture in Section 5 was implemented in Xilinx 2V6000-5 FPGA. We achieved ciphering rate of 200MHz or 14080 Mbps, which is comparable to the recent implementations reported in Table 1. On the other hand, our architecture supports five algorithms and therefore, it has more flexibility than other implementations. A hardware implementation of LOKI91 algorithm by the authors of this paper reported in [13], achieved the ciphering rate of 127 Mbps on Xilinx Virtex-E. The implementation presented in [13] did not reach this new reconfigurable implementation speed because of large S-boxes of LOKI. The implementation in [13] did not use lookup tables for 12x8 S-boxes; instead it implemented the S-box in hardware gates. However, in our reconfigurable architecture, memory was used as a large S-box. The hardware presented in this paper implements all above algorithms by a unique reconfigurable scheme that shares the resources for the common parts of the algorithms. The different parts are implemented in memories and content of the memories are changed according to the running algorithm. Since in our architecture, there is a similar critical path for all algorithms, therefore, LOKI91 algorithm has the same speedup of DES.

Table 1. DES algorithm implementations

	Year Reported	Integrated circuit(s) used	Throughput (Mbits/sec)	Reference
1	1997	Xilinx 4013-4	26.4	[16]
2	1998	Xilinx 4028EX-3	402.7	[17]
3	2000	Xilinx V150-6	10752	[10]
4	2000	XCV300E-8	12000	[24]
5	2003	Xilinx 2V1000-5	15168	[11]
6	2003	Xilinx 2V1000-5	21312	[12]
7	2006	Xilinx 2V6000-5	14080	Ours

A simple control unit is built together with our reconfigurable architecture to program its memories, so the requirements of using PC to upload reconfiguration data can be eliminated and our hardware can be dynamically reconfigured. In addition, an onboard external memory to retain the S-box's data for all selected algorithms is required. Our hardware reconfiguration time is the time our internal memories take to be programmed. As four RAM blocks of size 2^{12} * 8 bits exist in each 16 rounds of the architecture, the configuration time can be calculated as:

Configuration time: $16 \times (4 \times 2^{12}) \times Access\ time$

When our external memory can work at 100MHz clocking rate then our reconfiguration time will be 2621440 ns or 2.6ms.

6. Conclusion

In this paper, a single reconfigurable architecture was proposed to implement five cryptography algorithms. In this architecture, DES, LOKI, DESX, Biham-DES, and S^nDES algorithms share the same resources. The implementation of this architecture brings about similar throughput comparing with other implementations of single algorithms. The most important feature of this implementation is high flexibility of the hardware. Switching between the selected algorithms can be done by replacing the S-box contents implemented in memories.

On the other hand, one can implement all of our selected algorithms in hardware and choose between them using multiplexers. This is an alternative version but it would need approximately 5 times more area than ours. In the FPGA chips such as Xilinx 2V6000 there are not enough resources to implement all five encryption algorithms.

References

[1] National Bureau of Standards. FIPS PUB 46: Data Encryption Standard, January 1997.

[2] A. Shimizu, and S. Miyaguchi, "Fast Data Encipherment Algorithm FEAL," In David Chaum and Wyn L. Price, editors, *Advances in Cryptology-Euro-crypt'87*, Vol. 304 of LNCS, pp. 267-280, Springer-Verlag, 1998.

[3] L. Brown, M. Kwan, J. Pieprzyk, and J. Seberry, "Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI," *Proceedings of AsiaCrypt'91*, Vol. 739 of LNCS, pp. 36-50, 1993.

[4] J. Kilian, and P. Rogaway, "How to Protect Against Exhaustive Key Search," *Advances in Cryptology – CRYPTO'96*, Vol. 1109 of LNCS, pp. 252–267, Springer-Verlag, 1996.

[5] K. Kim, "Construction of DES-like S-boxes based on Boolean Functions Satisfying the SAC," *Proc. of ASIACRYPT'91*, 1991.

[6] K. Kim, S. Park, and S. Lee, "Reconstruction of S^2DES S-boxes and their Immunity to Differential Cryptanalysis," *Joint Workshop on Information Security and Cryptology*, pp. 59-72, 1993.

[7] K. Kim, S. Lee, S. Park, and D. Lee, "How to Strengthen DES against Two Robust Attacks," *Joint Workshop on Information Security and Cryptology*, 1995.

[8] K. Kim, S. Lee, S. Park, and D. Lee, "How to Strengthen DES against Three Robust Attacks," *Electronics and Telecommunications Research Institute*, 1995.

[9] E. Biham, and A. Biryukov, "How to Strengthen DES Using Existing Hardware," *Proc. of the 4th International Conference on the Theory and Applications of Cryptology: Advances in Cryptology*, pp. 398-412, Springer-Verlag, 1994.

[10] C. Patterson, "High Performance DES Encryption in Virtex FPGAs using JBits," *In Proc. of FCCM 2000*, IEEE Computer Society, 2000.

[11] V. Pasham, High-Speed DES and Triple DES Encryptor/Decryptor, http://www.xilinx.com/bvdocs/appnotes/xapp270.pdf.

[12] G. Rouvroy, F-X. Standaert, J.-J. Quisquater, and J.-D. Legat, "Design Strategies and Modifed Descriptions to Optimize Cipher FPGA Implementations: Fast and Compact Results for DES and Triple-DES," *Proceedings of FPL 2003*, Vol. 2778 of LNCS, pp. 181-193, Springer-Verlag, 2003.

[13] A. Valizadeh, M. Saheb Zamani, B. Sadeghian, and F. Mehdipour, "Hardware Implemenation of LOKI Block Cipher," *Proceedings of the Tenth International Symposium on Integrated Circuit*, September 2004.

[14] C. Patterson, "A Dynamic Implementation of the Serpent Block Cipher," *Proc. of Workshop on Cryptographic Hardware and Embedded Systems*, Vol. 1965 of LNCS, pp. 142–155, Springer-Verlag, 2000.

[15] A. J. Elbirt, and C. Paar, "Instruction-Level Distributed Processing for Symmetric-Key Cryptography," *Proceedings of the Seventeenth International Parallel and Distributed Processing Symposium*, 2003.

[16] J. Leonard, and W. H. Mangione-Smith, A case study of partially evaluated hardware circuits: Key-specific DES, *Proc. of Seventh International Workshop on Field-Programmable Logic and Applications*, pp. 151-160, Vol. 1304 of LNCS, 1997.

[17] J.-P. Kaps, and C. Paar, "DES auf FPGAs (DES on FPGAs) ," in German, *Datenschutz und Datensicherheit*, Vol. 23, No. 10, pp. 565–569, 1999.

[18] M. Barr, *A reconfigurable computing primer*, Miller Freeman Inc., 1998.

[19] K. Bazargan, R. Kastner, and M. Sarrafzadeh, "Fast template placement for reconfigurable computing systems", *IEEE Design and Test of Computers*, pp. 68-83, 2000.

[20] K. Bazargan, M. Orgenci, and M. Sarrafzadeh, "Integrating scheduling and physical design into a coherent compilation cycle for reconfigurable computing architectures," *Proc. of Design Automation Conference*, pp. 635-640, 2001.

[21] R. Enzler, The current status of reconfigurable computing, Swiss Institute of Technology, Technical report, July 1999.

[22] R. Maestre, F. J. Kurdahi, N. Bagherzadeh, H. Singh, R. Hermida, and M. Fernandez, "Kernel scheduling in reconfigurable computing," *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, pp. 90-96, 1999.

[23] R. Tessier, and W. Burleson, "Reconfigurable computing for digital signal processing, A survey," *Journal of VLSI Signal Processing*, Vol. 28, pp. 7-27, 2001.

[24] S. Trimberger, R. Pang, and A. Singh, "A 12 Gbps DES Encryptor/ Decryptor Core in an FPGA," *Proc. of Workshop on Cryptographic Hardware and Embedded Systems*, Vol. 1965 of LNCS, pp. 156–163, Springer-Verlag, 2000.

Ali Valizadeh received the B.S. and M.S. degree in Computer Engineering from Amirkabir University of Technology (Tehran Polytechnic) in Iran, in 2001 and 2005 respectively. He Joined Computer Engineering Department of Islamic Azad University Shahryar-Shahr-e-Qods branch in 2005, and has been with the department since then. His research interests are reconfigurable computing systems, Cryptography and Information Security.
E-mail: valizadeh@mail.sana-co.com

Morteza Saheb Zamani received the B.Sc. degree in Computer Engineering from Isfahan University of Technology in 1989, and the M.Eng.Sc. and Ph.D. degrees in Computer Engineering from the University of New South Wales, Australia in 1992 and 1996, respectively. He joined Amirkabir University of Technology in 1996 and He is now an associate professor at Computer Engineering and IT department. His research interests are electronic design automation, quantum computing and reconfguarble computing.
E-mail: szamani@aut.ac.ir

Babak Sadeghiyan was born in 1961. He received his B.Sc. in 1985 in Electronics from Isfahan University of Technology, and his M.Sc. in 1989 in Electronics from Amirkabir University of Technology. He received his Ph.D. in 1993 in Computer Science from University College, University of New South Wales, Australia. Currently, he is an associate professor in Amirkabir University of Technology. His academic fields of interest are on Cryptology and Information Security, where his research areas of interest are design and analysis of cryptographic algorithms and protocols, implementation and analysis of cryptographic hardware system, and intrusion detection systems. He is the co-author of the book “Design of Hashing Algorithm”. He is the designer of Moamagar block cipher, and so far, has published more than 80 journal and conference research papers. He joined Computer Engineering Department of Amirkabir University of Technology in 1993, and has been with the department since then. He had role in the initiation of Information Security M.Sc. program in the department in 2004, and currently takes part in Information Security subgroup in the Department. He also had role in the establishment of Computer Security Incident Response Team laboratory in the university in 2008, known as APA Lab. of Amirkabir University of Technology. He is a founder for Iranian Society of Cryptology in 2000, and a member of Computer Society of Iran since foundation in 1993.
E-mail: basadegh@aut.ac.ir

Farhad Mehdipour received the B.Sc. degree from Sharif University of Technology in 1996, and the M.Sc. and Ph.D. degrees in Computer Systems Architectures from the Amirkabir University of Technology (Tehran Polytechnic) in 1999 and 2006, respectively. He was a visiting researcher at the Kyushu University in Fukuoka, Japan from November 2005 to June 2006. Since December 2006, he joined to the Faculty of Electrical Engineering and Information Science in the Kyushu University as a researcher and is working on high-performance reconfigurable processors. His research interests include reconfigurable computing systems, VLSI physical design automation and reconfigurable embedded processors. He is a member of IEEE.
E-mail: farhad@c.csce.kyushu-u.ac.jp

Paper Handling Data:

Submitted: 13.05.2007
Received in revised form: 24.12.2008
Accepted: 11.04.2009
Corresponding author: Ali Valizadeh,
Department of Computer Engineering, Islamic Azad University, Shahryar-Shahr-e-Qods branch, Tehran, Iran.