CONFIGURABLE HARDWARE IMPLEMENTATION OF TRIPLE-DES ENCRYPTION ALGORITHM FOR WIRELESS LOCAL AREA NETWORK

Panu Hämäläinen, Marko Hännikäinen, Timo Hämäläinen, and Jukka Saarinen

Digital and Computer Systems Laboratory, Tampere University of Technology
Hermiankatu 3 A, FIN-33720 Tampere, FINLAND
Tel: +358 3 365 2111, Fax: +358 3 365 4575
E-mail: panuh@cs.tut.fi, markoh@cs.tut.fi, timoh@cs.tut.fi, jukkas@cs.tut.fi

ABSTRACT

This paper presents three implementations of Triple Data Encryption Standard (3DES) algorithm on a configurable platform. Implementations are aimed at Medium Access Control (MAC) protocol of a multimedia-capable Wireless Local Area Network (WLAN). For this reason, very strict timing constraints as well as demands for area-efficiency are present. The MAC processing is handled by a Digital Signal Processor (DSP) and a Xilinx Virtex Field Programmable Gate Array (FPGA) chip. The latter one is also used for the presented encryption implementations. As a result of the study, 3DES implementations with small area and reasonable throughput and, on the contrary, with large area and very high throughput are realized. Even though 3DES turns out to be quite large and resource-demanding, the implementations still leave enough chip area for the other MAC functions. Consequently, the set requirements are met and the cipher can be integrated into the system.

1. INTRODUCTION

Wireless Local Area Network (WLAN) technology is very promising for various types of wireless indoor and limited outdoor communications. Applications range from electronic mail to wireless multimedia. WLANs provide network solutions when wired networks become impossible or inconvenient, for example, when movement of users is required, or when a network is set up on a temporary basis.

Although wireless connections make networks very flexible and convenient, they are also far easier to eavesdrop than traditional cable networks. At the simplest level, data is available to anyone within the range of a transmitting device. As a result, these networks are very sensitive to security violations and need powerful encryption. On the other hand, because of the real-time service requirements and limited processing capabilities of embedded terminals (e.g. on-chip memory), the encryption implementations must be efficient.

This paper presents an encryption implementation purposed for the Medium Access Control (MAC) protocol of Tampere University of Technology WLAN (TUTWLAN) system [10]. TUTWLAN is a proprietary, multimedia-capable WLAN implemented at the Digital and Computer Systems Laboratory of TUT.

One of the novelties in TUTWLAN is the tightly integrated encryption in the MAC protocol. Thus, a transparently secured transmission channel is provided for the upper layer protocols. In addition, TUTWLAN uses configurable hardware and Digital Signal Processor (DSP) for the MAC implementation, which enables implementation of a large range of different protocol features on the same platform. For example, it is possible to flexibly change the encryption method, and thereby the provided security level, independent of the application [11].

The current TUTWLAN demonstrator platform [9] contains a DSP and a Xilinx Virtex Field Programmable Gate Array (FPGA) chip [12], into which the presented implementations are targeted. In addition to the encryption, the FPGA also includes the functions for interfacing the host computer (PC), the DSP, and a radio sub-system module. Because several different functions are included into the FPGA, the cipher implementation should be area-efficient.

In this paper, three Triple Data Encryption Standard (3DES) [3] implementation alternatives are presented in order to study its suitability for MAC-level encryption on a reconfigurable platform. DES has already been implemented on various platforms and thoroughly analyzed [1][4][7][8]. Especially several *single*-DES dedicated hardware implementations have been presented, but none for 3DES on a configurable platform according to the authors' knowledge. The penalty of mapping to an FPGA compared to a full-custom design is that on an FPGA there exists only limited amount of resources and possible routing paths. A full-custom chip does not have this kind of limitations. Therefore, configurable logic designs cannot usually be as efficient as full-custom designs.

The paper is organized as follows. The first section summarizes the 3DES algorithm. Next, the realized implementations and the obtained results are presented. The basic implementation design is covered first, followed by more optimized designs with removed redundancy. Next, a trade-off analysis between area and throughput is given. Discussion of the results is aroused in the concluding section.

2. 3DES ALGORITHM INTRODUCTION

3DES was designed to encrypt 64-bit blocks of data under the control of three unrelated 64-bit keys. Each key is used as an input to a DES block. The actual utilized key size is three times 56 bits because every eighth bit is used for parity checking and will be ignored afterwards. The cipher is symmetric. Therefore, decryption is accomplished by using the same keys and the same algorithm as in encryption. The only difference is that the key schedule is reversed. Fig. 1 presents the high level structure of 3DES. [2]

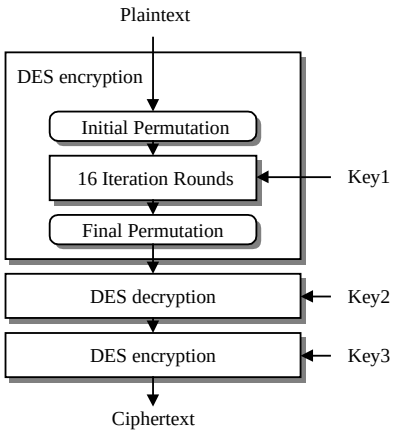

Fig. 1. High level structure of 3DES.

Commonly, 3DES is regarded as a rather weighty cipher, especially for software implementations. In software even a simple bitwise permutation is relatively tricky and therefore leads to several lines of code (at least in C/C++). On the contrary, in hardware permutations can be implemented as hard-wired connections, which are easy to accomplish and do not produce any additional delay. In addition, the 3DES algorithm's most essential parts are the substitution boxes (S-boxes), which are most effectively implemented in hardware. An S-box contains 64 4-bit digits, which are used for substituting the S-box input bits. More detailed information on the algorithm can be found, e.g., in [3] and [2].

Table 1 presents some software performance ratings for 3DES to enable the comparison of software and the FPGA implementations. The used C source code was obtained from Phil Karn of Qualcomm Incorporated[1]. The code was compiled for MS-DOS using *djgpp* compiler and for HP-UX using GNU C-compiler.

Table 1. Software performance of 3DES on different platforms.

Processor type	*Speed*	*Operating System*	*Encryption throughput Mbytes/s*
Intel Pentium	166 MHz	MS-DOS	0.55
Intel Pentium II	400 MHz	MS-DOS	2.26
Intel Pentium III	600 MHz	MS-DOS	3.39
HP C3000	400 MHz	HP-UX	2.50

3. 3DES IMPLEMENTATIONS

This section introduces the realized implementation alternatives, optimizations, and achieved results. The presented implementations were made in Very High-speed integrated circuit Hardware Description Language (VHDL) [5] and the used development software was Xilinx Foundation F2.1i [6]. The designs were targeted to a Xilinx Virtex family FPGA chip [12]. Virtex is a new Xilinx FPGA that has been designed especially for large and time-critical implementations.

[1] The source code is available at http://people.qualcomm.com/karn/code/des/index.html (visited January 26, 2001).

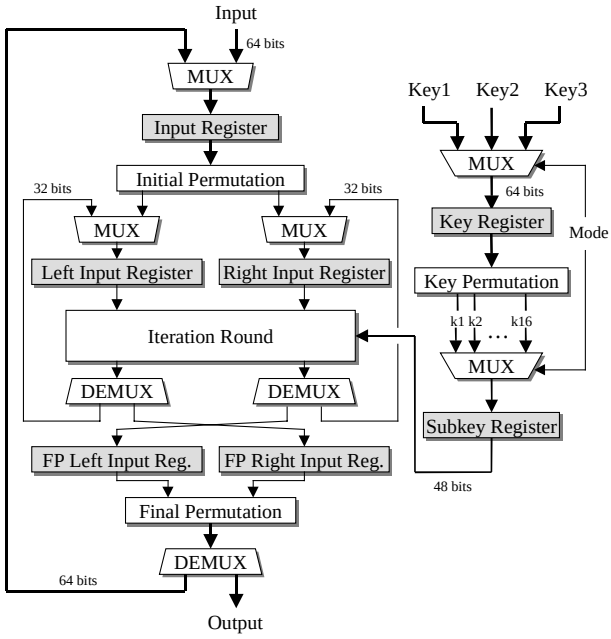

Fig. 2. Full 3DES implementation.

3.1. Basic Implementation

Since 3DES consists of 48 consecutive similar iterations, it was possible to carry out the basic implementation with only one iteration round block, a subkey generator, and a state machine. The iteration block performs enciphering/deciphering, the key generator creates the needed subkeys, and the state machine takes care of feeding the right inputs and subkeys to the iteration block. The same approach was also used in [7] and [8] for DES.

Fig. 2 illustrates the high level structure of the entire implementation. The state machine controls the configuration of the multiplexors and demultiplexors presented in the figure. The external *mode* signal determines whether the cipher is performing encryption or decryption. If the decryption mode is chosen, the subkey order is reversed.

The logic operates as follows:

1. On the first clock cycle the input is stored in the initial permutation input register.
2. Next the plaintext/ciphertext input proceeds through the initial permutation to the left and right input registers of the iteration round block. In the beginning of the next clock cycle both registers contain a 32-bit half of the permuted input.
3. During the following 16 cycles the contents of the input registers and the subkey register are repeatedly fed through the iteration round block, and the result is written back to the input registers. After the 16 rounds the result is permuted in the final permutation block to finish the DES iteration.
4. Steps 1-3 are repeated three times to realize the full 3DES encryption/decryption. For the first 16 iteration rounds the used encryption key is Key1, then Key2 for the next 16 rounds, and Key3 for the final 16 iterations.

5. After the last iteration the undermost demultiplexor directs the output of the final permutation to the chip output, and the encryption/decryption is completed.

The execution of the entire flow takes 55 clock cycles. In order to find the best alternative for S-box implementation, two different methods were tested. The first one uses only combinatorial logic to perform the substitutions, and the other takes advantage of the FPGA's internal Read Only Memory (ROM). In the latter alternative each memory slot contains an output value corresponding to the input. Table 2 shows the detailed implementation results of the both, logic and ROM, designs on Virtex-V800FG676-6. In the table *user I/O* means the chip input/output pins reserved by the designer, *slice* is a Virtex configurable logic cell, and *gate count* refers to the equivalent amount of gates if the design was implemented with logical gates only. A slice contains logical gates and look up tables (LUTs), which can be configured to function as ROM. As can be seen, the results of the implementation with ROM S-boxes are far better than those of the logic implementation: the number of reserved slices is 22% smaller and the maximum clock frequency given by the Foundation is 47% higher.

Table 2. Implementation results of the full 3DES.

Design	LOGIC	ROM
User I/O	298 / 444	298 / 444
Virtex Slices	1,059 / 9,408	825 / 9,408
4-input LUTs	1,949	1,359
16x1 ROMs	0	128
Gate Count	14,622	15,370
Max. Clock Frequency	27.566 MHz	40.532 MHz
Throughput with Maximum Clock	4.01 Mbytes/s	5.90 Mbytes/s

3.2. Optimizations and Trade-offs

In order to improve the efficiency and the execution time of the cipher, three additional implementations are designed. The first one removes redundancy from the previous implementation, and the latter ones attempt to shorten the encryption time. In the last implementation 3DES is fully pipelined in order to achieve maximum throughput and to test Virtex's capacity.

Because the initial and final permutations are each other's inverses [3], performing final permutation to a message permuted with the initial permutation leads to the original message as described in [4]. Therefore, it is possible to leave out the final permutations of the first and the second DES and the initial permutations of the second and third DES. This way four clock cycles per message block are saved (resulting execution time is 51 clocks). In addition, excluding the intermediate initial and final permutations makes the outline of the cipher simpler as data path from final permutation to initial permutation can be removed.

Table 3. Implementation results of the optimized 3DES.

Design	LOGIC	ROM
User I/O	298 / 444	298 / 444
Virtex Slices	1,107 / 9,408	740 / 9,408
4-input LUTs	1,949	1,110
16x1 ROMs	0	128
Gate Count	14,622	13,876
Max. Clock Frequency	43.906 MHz	35.791 MHz
Throughput with Maximum Clock	6.89 Mbytes/s	5.61 Mbytes/s

Table 3 presents the detailed results of the simplified implementations on the Virtex FPGA. The table shows a significant improvement in the efficiency of the ROM implementation. The number of used slices has dropped almost by a hundred from the full implementation. Surprisingly, despite of the achieved smaller area, the maximum clock frequency is decreased. Even though the logic implementation is much larger than the ROM implementation, the maximum clock frequency is still better. The design tool managed to optimize this logic implementation markedly better than the basic implementation (the maximum clock is almost doubled). In addition, as it can be seen in the maximum clock frequencies, the design tool also succeeded better in routing for the logic implementation than for the ROM implementation.

In the next optimization state it was tested whether the execution time of the cipher could be shortened. The design adds one iteration round to the optimized data path of Fig. 2. The result is that the loop from the demultiplexors to the iteration round input registers needs to be executed only 24 times instead of 48. The idea of loop unrolling for DES was originally presented in [1]. However, in [7] it was never taken advantage of. The altered part is depicted in Fig. 3.

Table 4 presents the results of this two-round implementation. The method cut the execution time down to 27 clock cycles and proved to be very cost-effective as well, especially as a ROM version. The results show that even though the maximum clock frequency on the chip was somewhat decreased, the data throughput was increased. It is possible to execute even more iterations on a clock cycle. However, since the maximum clock frequency in two rounds was already decreased, adding more rounds was not assumed to increase the overall performance.

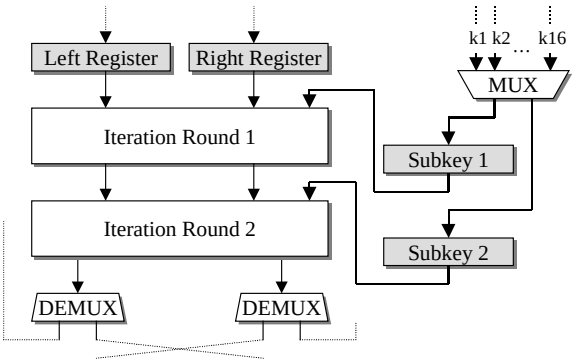

Fig. 3. 3DES with two iterations per clock cycle.

Table 4. Implementation results of the two-round 3DES.

Design	LOGIC	ROM
User I/O	298 / 444	298 / 444
Virtex Slices	1,257 / 9,408	833 / 9,408
4-input LUTs	2,280	1,175
16x1 ROMs	0	256
Gate Count	16,984	18,930
Max. Clock Frequency	25.092 MHz	29.044 MHz
Throughput with Maximum Clock	7.43 Mbytes/s	8.61 Mbytes/s

In order to maximize throughput, pipelining was utilized in the last version of the cipher. According to the results of the previous implementations, the most effective way is to execute two iteration rounds per clock cycle. Therefore, the design consists of 24 consecutive double-iteration rounds with registers in between. As a result, a new plaintext block can be fed to the logic on every clock cycle. Since every iteration round is followed by a register, the output latency is 24 clock cycles. However, with this design the throughput increases remarkably. After the first 24 clock cycles, a new 64-bit ciphertext block is received on every cycle. The implementation covers both encryption and decryption.

Table 5 shows the results for the design. The design was only implemented as a ROM version since it had already been verified to be the more effective one. As can be seen, pipelining led to quite a large implementation. However, it proved to be very effective when examining the throughput, which was almost 370 Mbytes/s. Furthermore, even though the implementation requires 12 kilobytes of ROM, it only consumes 71% of the chip's resources. This shows that Virtex is suitable for very large implementations.

Table 5. Implementation results of the pipelined 3DES.

Design	ROM
User I/O	298 / 444
Virtex Slices	6,689 / 9,408
4-input LUTs	3,568
16x1 ROMs	6,184
Gate Count	271,472
Max. Clock Frequency	45.550 MHz
Throughput with Maximum Clock	364 Mbytes/s

4. DISCUSSION

Altogether, 3DES turned out to be quite large and resource demanding. However, the used Virtex chip proved to be well suited for this algorithm. In fact, the target frequency of 40 MHz was exceeded, and the achieved data throughput was high.

The best throughput was achieved by pipelining the encryption process. A new input was fed and one iteration round was executed on every clock cycle. Considering the pipeline, a potential improvement is to combine more iteration rounds to shorten the pipeline. This would improve the output latency. However, the combinatorial path delay would increase resulting in a decrease in maximum clock frequency. For the presented non-pipeline implementations, pipelining two or a few more iteration rounds would increase the throughput. However, this kind of pipeline requires a more complicated state machine, which inevitably leads to a decrease in maximum clock frequency and consumes more chip resources.

Within the presented implementations, 3DES with two iterations per clock cycle was considered the best. The maximum clock frequency was relatively high, and the design could also be fitted on smaller chips. Furthermore, compared to the software performances in Table 1, the two-round hardware was more than 2.5 times faster than the best software implementation. With some re-timing and careful adjustment of synthesis parameters, it should be possible to increase the maximum clock frequency closer to the target of 40 MHz of the TUTWLAN platform.

As a result, the 3DES algorithm was proved to be suitable for hardware implementation. In addition, the presented results showed that the TUTWLAN MAC level requirements were met, and thus it is possible to combine the cipher to the system. Enough chip area was still left for the host and radio interface functions.

5. REFERENCES

[1] A. G. Broscius, J. M. Smith, "Exploiting Parallelism in Hardware Implementation of the DES," CRYPTO'91, Santa Barbara, California, USA, pp. 367-376, 1991.

[2] B. Schneier, "Applied Cryptography: Protocols, Algorithms and Source Code in C," 2nd edition, John Wiley & Sons, Inc., USA, 1996.

[3] "Data Encryption Standard," Federal Information Processing Standards (FIPS) Publication 46-7, National institute of Standards and Technology (NIST), USA, 1999.

[4] D. C. Feldmeier, P. R. Karn, "UNIX Password Security – Ten Years Later," CRYPTO'89, Santa Barbara, California, USA, pp. 44-63, 1989.

[5] D. L. Perry, "VHDL," 2nd edition, McGraw-Hill, Inc., USA, 1994.

[6] "Foundation Series 2.1i Install and Release Document," Xilinx, Inc., USA, 1999.

[7] H. Eberle, "A High-speed DES Implementation for Network Applications," CRYPTO'92, Santa Barbara, California, USA, pp. 521-539, 1992.

[8] J-P. Kaps, C. Paar, "Fast DES Implementations for FPGAs and its Application to a Universal Key-Search Machine," SAC'98, 1998, Kingston, Ontario, Canada, pp. 234-247.

[9] K. Tikkanen, M. Hännikäinen, T. Hämäläinen, J. Saarinen, "Advanced Prototype Platform for a Wireless Multimedia Local Area Network," EUSIPCO 2000, Tampere, Finland, pp. 2309-2312, 2000.

[10] M. Hännikäinen, J. Knuutila, A. Letonsaari, T. Hämäläinen, J. Jokela, J. Ala-Laurila and J. Saarinen, "TUTMAC: A Medium Access Control Protocol for A New Multimedia Wireless Local Area Network," PIMRC'98, Boston, USA, pp. 592-596, 1998.

[11] P. Hämäläinen, M. Hännikäinen, T. Hämäläinen, J. Saarinen, "Hardware Implementation of the Improved WEP and RC4 Encryption Algorithms for Wireless Terminals," EUSIPCO 2000, Tampere, Finland, pp. 2289-2292, 2000.

[12] "The Programmable Logic Data Book," Xilinx, Inc., USA, 1999.