Efficient Hardware Implementations for the DES Family

C. MANIFAVAS
Technological Educational Institute
Applied Informatics & Multimedia Dept.
P.O. Box 1939

I. PAPAEFSTATHIOU, C. SOTIRIOU
Institute of Computer Science
Foundation of Research & Technology
Vassilika Vouton, P.O. Box 1385

Heraklion, Crete, GREECE

Abstract: - Network data is, currently, often encrypted at a low level. In addition, as it is widely supported, the majority of future networks will use low-layer (IP level) encryption. Moreover, current trends imply that future networks are likely to be dominated by mobile terminals, thus, the power consumption and electromagnetic emissions aspects of encryption devices will be critical. This paper presents several realizations of one of the most widely used encryption algorithm, the DES/TripleDES, both in software and in hardware. We present software implementations of the algorithm running on two of the state-of-the-art Intel IXP network processors and several hardware realizations based on a standard-cell library. The hardware platforms presented appear to be optimal. Moreover, by placing and routing those designs, we have also realized that the commercial ASIC synthesis tools cannot accurately predict the area and the performance of the placed & routed final netlist in such designs, since the ASIC implementations of the encrypted algorithms include a very large number of wires and a limited number of logic CMOS cells.

Key-Words: - Cryptography, 3DES, AES, Software Implementation, Hardware Implementation

1 Introduction

Low-level data encryption is becoming one of the essential applications that network devices must implement. One of the characteristics of data encryption algorithms is that, when implemented on a general-purpose CPU, they exhibit poor performance, even when running on a sophisticated, state-of-the-art architecture.

To tackle this problem, special-purpose hardware encryption cores, such as the one in Intel's IXP-2850 network processor [6], are becoming more and more ubiquitous in state-of-the-art networks for providing high-speed encryption. Although, such special-purpose hardware devices may satisfy bandwidth requirements, they seem to have high power requirements and cause high ElectroMagnetic Emissions (EME).

Further on, power consumption is becoming more and more critical as network technology is shifting from today's wired devices to the mobile terminals of tomorrow. In addition, EME levels are constantly increasing by the ever-increasing clock frequencies, however acceptable EME levels are decreasing due to the close proximity of IP cores in contemporary SOC designs. EME may also be used as a means of attacking security devices, as they may reveal critical information about the nature of the encryption algorithm [8].

In this paper we perform a design space exploration of one of the most widely used cryptography algorithms, namely the DES. This design can be used as a building block in the construction of 3DES [18] (the modern incarnation of DES) modules. We have implemented DES in both software and hardware and report on the performance and power consumption (and area for the hardware implementations) for each. In this paper we investigate a number of additional implementations of the DES algorithm, and we present innovative, and very interesting as the Section 3 shows, results regarding the placement & routing of all the DES implementations.

Our software implementations are based on two of Intel's IXP network platforms, for which we wrote assembly core implementing both algorithms and trying to utilize as many as possible of its various programming features. All of our hardware implementations have not only been synthesized, but also placed & routed.

In the next sections we present our experimental results and demonstrate that both commodity and high-end special-purpose network processors exhibit a significant amount of power consumption at very modest performance, when executing the encryption software. We also demonstrate that the most efficient realization of the DES algorithm, when taking into account the data bandwidth supported and the power consumed, is a hardware one. Finally, we show that the results produced by the synthesis tools for this class of algorithms, differ significantly

in both area and performance from the reported results by the placement & routing tools. This discrepancy, which we believe it is an important aspect for anyone implementing encryption algorithms in hardware, is probably due to the following fact: the implementations of those encryption algorithms include a very large number of wires and limited amount of CMOS logic, and as it is widely known, the synthesis tools are not very well in calculating the area and the delays of the wires.

2 Related Work

There is considerable amount of work done in both industry and academia related to cryptographic algorithms and their performance evaluation. In the literature, there are several implementations of the two most widely used cryptographic algorithms (DES [14] and AES[15]) in either software (e.g. [1, 5]), or hardware; the hardware approaches are tailored to both ASICs and FPGAs.

Regarding the software implementations of DES, on various platforms, the maximum throughput achieved is around 100 Mbits [2,10]. When moving to the FPGA implementations, the fastest DES ciphers presented can provide 12 Gb/sec of useful bandwidth [9], but with key set up latency in the order of milliseconds, or even 21 Gb/sec [11] with a highly pipelined design, which is much more complicated, in terms of hardware, than all the designs presented in this paper. As far as the ASIC implementations are concerned, there are fabricated single chip approaches that support up to 9.6 Gb/sec [7, 12] even though they are implemented on relatively old CMOS processes (at 0.6μm and 0.35μm). In this paper, regarding the DES algorithm, we fill the gap of modern software implementations by measuring the performance of both a commodity and a state-of-the-art Network Processor when executing this encryption algorithm; more importantly we present a number of low-cost, low-power implementations for ASICs that achieve throughputs close to 40Gb/sec.

2 AES-DES S/W Implementations

In this section we explore the performance of the DES encryption/decryption algorithm implemented in software running on two network processors, a commodity and a state-of-the-art one. Moreover, in order to demonstrate the inefficiency of the software approach when executing cryptographic algorithms, we also measured the performance of the other standard cryptographic algorithm, the AES. The commodity network processor of choice is the widely used Intel IXP-1200 [6]. The IXP-1200 is a powerful, multiprocessor system, composed of six 32-bit RISC processing ``microengines" and a single general-purpose StrongARM CPU. To enable fast on-chip processing the IXP contains a 4K on-chip SRAM ``scratch" memory. For certain applications its performance may support processing rates of up to 1Gb/sec.

To evaluate the performance of the IXP processor when running both algorithms we hand-crafted assembly code implementing the algorithm. In addition, we developed several software realizations of our encryption algorithms, in order to be able to exploit one or more of the six IXP processing ``microengines".

Tables 1 and 2 show the performance of the IXP processor running our hand-crafted assembly code on only one of the six microengines. Three different data blocks were encrypted for these experiments using the same key. Thus, key setup was run only once, then each of the three blocks were encrypted and then decrypted. Table 1 shows the latency, instruction count (IC) and throughput for the encryption of the three blocks, whereas Table 2 shows the same data for their decryption.

In our next experiment we attempted to use the Strong-ARM CPU instead of a single microengine, however we discovered that the StrongARM performance was lower than that of a single microengine. This can be attributed to the fact that the on-chip scratch memory exhibits long latency with respect to the StrongARM processor (as opposed to the microengines which communicate through an interconnection network), thus creating a bottleneck and producing poor performance.

Next, we distributed six copies of the encryption/decryption code onto the six IXP microengines with the aim of achieving even better performance by utilizing all of the IXP's available resources. This experiment required a significant amount of effort for packing the DES and AES code into a minimum number of instructions in order to fit the code together with the data onto the small SRAM scratch memory. It was essential that the scratch memory was used, since, in any other case, accesses to an external SRAM caused the performance of this experiment to be only twice as high as that of the single microengine one.

Tables 3 and 4 show the performance of both algorithms using the same experimental setup, but running on all six of the IXP's microengines. The figures demonstrate a 5.5 times improvement in throughput when all the IXP's microengines are

used. Based on this fact we claim that the latency of the IXP microengine interconnection network does not significantly reduce the performance of the device. However, even this higher throughput when executing the fastest from the two algorithms (i.e. AES), is insufficient for contemporary commodity network architectures, *e.g.* Fast-Ethernet, which runs at 100 Mbits/sec, or Wireless LAN, 802.11a running at 52 Mbits/sec.

We have also, measured the performance of the encryption algorithms in the state-of-the art, very recently introduced, IXP processors [6] (the IXP-2xxx family) that has just been manufactured in Intel's 0.13 μm technology.

The results of our experiments are shown in Table 5 and 6 where all the microengines of IXP-2400 are employed. Those results are the ones we could achieve after running our handcrafted pieces of software realizations of the two encryption algorithms running on one to eight microengines. A first remark is that the instruction count in the IXP2400 is slightly lower than that on the IXP1200, mainly due to the fact that
the IXP2400 has a different instruction set, and it supports some special "powerful" instructions utilizing some of the unique hardware features provided by it (for example the 16 entry CAM which is associated with each Microengine).

As it can be seen from those tables, even in those high-end devices, the maximum bandwidth supported is not more than 80Mb/sec for the DES and 120 Mb/sec for the AES. As a result, we believe that, the software implementation of those encryption algorithms cannot, support the state-of-the-art LAN speeds, even if those pieces of software are run on a high-end network multi-processor. Similar results regarding the software performance in various, non-network specific, platforms, can be found in [17]. The applicability of this argument in the world of network processing elements, in general, is also shown by the fact that, as it was mentioned earlier, Intel's high-end family of Network Processors include a model with an embedded hardware DES block (IXP2850).

Table 7 shows the mean power consumption actually *measured* in the development board, when the IXP1200 was running our experiments. These figures show only the power consumed by the processor core and not by the processor's interface, which operates at a different voltage level. The power consumption was about the same no matter which of the two cryptography algorithms the core was executing. These measurements are in line with the IXP's datasheet, which states a maximum power consumption of 4 Watts and a typical power consumption of approximately 2.4 Watts. In these experiments we have not come close to the maximum power consumption, by not using the StrongARM core.

The power figures demonstrate that even a dedicated, state-of-the-art, low-power CPU, certainly exceeds the power requirements of mobile network terminals, when executing either of the two most widely used cryptography algorithms.

2 DES Hardware Implementations

In this section we present three hardware implementations of the DES algorithm. All designs were synthesized, using Synopsys [16]; placed and routed, using Cadence's Silicon Ensemble [3], and targeted to the 0.18μm VST-UMC [4] technology library. We also present, separately, the results produced by the synthesis tools and those produced by the placement & routing tools, that make the final silicon masks. As it is demonstrated in the next sections, those results differ by a relatively large amount, probably, due to the fact that there is a large number of wires in any DES implementation.

Two of the hardware implementations were based on just permutations of bits and a final XOR of them. The third was our own design optimized for low-power, and it is based on three hardware modules the RL, F and Key, just as the software implementation of the DES, which comprises of those three subtasks. This design was also used as the basis for the asynchronous designs described in the next subsection. Table 8 contrasts the characteristics of the three synchronous DES designs.

The data presented in Table 8 were obtained by post-synthesis simulation. The power consumption figures were obtained by performing switching activity annotation of the circuit during simulation. The area figures are cell totals.

The Area-Optimized version aims to achieve minimum area for a DES algorithm implementation, by performing the 16 steps of the DES algorithm iteratively and with limited CMOS resources, whereas the Performance-Optimized version aims at maximum throughput by employing a 16-stage pipeline, comprising of 16 identical high-speed DES modules. Our own design, has 16 pipeline stages as well, and each stage is optimized mainly for low power by trying to reduce both the number of the CMOS cells utilized at a given time and the signal transitions to a minimum. The actual architecture is very similar to the `` Coarse-grain" asynchronous one described in detail in the next section. This

"Coarse grain" asynchronous design comes from a synchronous one to which we have applied the "desynchronization" method, also presented in this next section}. As those results demonstrate, our design succeeds in its target since it has much lower power consumption than the Permutation-Based performance optimized one, while its speed is not significantly lower. In other words, the performance to power ratio is about 30% better in our design than in the widely used permutation based one.

We have also placed and routed (P&R) all those designs using Cadence's Silicon Ensemble flat P&R tool. Post P&R results demonstrated a significant increase in the latency of the designs as Table 9 clearly shows. This increase in latency is much higher than the 20% factor used as a rule of thumb in the majority of the general-purpose hardware modules [13]. This discrepancy, seems to mainly come from the fact that in the implementations of those encryption algorithms the fraction of wires over logic cells is very high (much larger than in other more regular devices, such as processors for example). Additionally, as the flat layout of our own Performance Optimized DES core shows in Figure 1, the interconnected basic blocks, within an implementation, have great variations in terms of the their complexity, and therefore they cannot easily "pitch-matched" (at least automatically by the tools and without any human interaction).

Moreover, the power consumption figures actually *measured* after P&R, differ significantly from the ones produced by switching annotation of the post synthesis circuit. This is probably due to the facts that a) the maximum working frequency after P & R is different than that reported by the synthesis tools, and b) the synthesis tools cannot accurately predict the capacitance of the wires or the exact impact of factors such as the actual applied voltage.

4 Conclusions

This paper explored different implementation choices for implementing one of the most widely used cryptography algorithm, namely the DES, and as a consequence 3DES, the modern incarnation of DES. We presented software implementations of the algorithms on both a commodity and a the state-of-the-art Intel's IXP network processor and demonstrated that both the performance and the power consumption of those realizations are inadequate for mobile, or high-speed network terminals. We also presented a set of three possible hardware implementations.

We have demonstrated that the most efficient implementation of the DES algorithm in terms of data bandwidth serviced and power consumed, is indeed a hardware one. This design can sustain 21Gb/s of throughput at a very modest power consumption of about 400mW.

References:

[1] G. Bertoni, L. Breveglieri, P. Fragneto, M. Macchetti, and S. Marchesi, Efficient Software Implementation of AES on 32-Bit Platforms, *In Proceedings of Cryptographic Hardware and Embedded Systems (CHES'02)*, pp. 159--171, 2002

[2] E. Biham, A Fast New DES Implementation in Software, *In Fast Software Encryption, 4th International Workshop (FSE'97)*, Haifa, Israel, January 20-22, 1997, vol. 1267 of *Lecture Notes in Computer Science*, pp. 260--271. Springer, 1997

[3] Cadence Design Systems, *Envisia Silicon Ensemble Place-and-Route Reference*

[4] EUROPRACTICE, UMC 0.18µm CMOS technology documentation

[5] B. Gladman, Implementation Experience with AES Candidate Algorithms, *In Proceedings of Second AES Candidate Conference (AES2)*, 1999

[6] Intel, The Next Generation of Intel IXP Network Processors, *Intel Technology Journal*, 6(3), Aug. 2002

[7] I. Kim, C.S. Steele, and J.G. Koller, A Fully Pipelined 700MBytes/s DES Encryption Core, *In Proceedings of 9th Great Lakes Symposium on VLSI*, page 386, 1999.

[8] M.G. Kuhn, Cipher Instruction Search Attack on the Bus-Encryption Security Microcontroller DS5002FP, *IEEE Transactions on Computers*, 47(10):1153--1157, Oct. 1998

[9] C. Patterson, High performance DES encryption in Virtex FPGAs using JBits, *In Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'00)*, pp. 113--121, 2000

[10] A. Pfitzmann and R. Amann, More efficient software implementations of (generalized) DES, *Computers and Security*, 12(5):477--500, Aug 1993

[11] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat, Efficient Uses of FPGAs for Implementations of DES and Its Experimental Linear Cryptanalysis, *IEEE Transactions on Computers*, 52(4):473--482, Apr 2003

[12] T. Schaffer, A. Glaser, S. Rao, and P. Franzon, A Flip-Chip Implementation of the Data

Encryption Standard (DES), *In Proceedings of IEEE Multi-Chip Module Conference (MCMC'97)*, page 13, 1997

[13] ST Microelectronics, *Notes on Synthesis, Placement and Routing*,

[14] F.I.P. Standard, Data Encryption Standard (DES), National Institute of Standards and Technology (NIST), FIPS 46-3, Oct 1999

[15] F.I.P. Standard, Advanced Encryption Standard (AES), National Institute of Standards and Technology (NIST), 2001

[16] Synopsys, *Design Analyzer Reference Manual*, 2000

[17] J. Worley, B. Worley, T. Christian, and C. Worley, AES Finalists on PA-RISC and IA-64: Implementations and Performance, *In Proceedings of the 3rd Advanced Encryption Standard (AES) Candidate conference*, 2000

[18] TripleDES, ANSI X9.52: Triple Data Encryption Algorithm Modes of Operation. 1998

	DES			AES		
1-IXP	**Latency**	**IC**	**Throughput**	**Latency**	**IC**	**Throughput**
Key Setup	412405ns	93000	N/A	92460	21000	N/A
1st Block Encryption	428038ns	3808	3.74Mb/s	266662ns	2394	5.98Mb/sec
2nd Block Encryption	444032ns	3808	3.84Mb/s	265237ns	2394	6.12Mb/sec
3rd Block Encryption	460026ns	3808	3.88Mb/s	271802ns	2394	5.83Mb/sec

Table 1: Encryption results using one IXP1200 Microengine.

	DES			AES		
1-IXP	**Latency**	**IC**	**Throughput**	**Latency**	**IC**	**Throughput**
Key Setup	410955ns	93000	N/A	185964ns	42110	N/A
1st Block Decryption	427669ns	3970	3.62Mb/s	264320ns	2453	5.86Mb/sec
2nd Block Decryption	444334ns	3968	3.71Mb/s	252942ns	2453	5.93Mb/sec
3rd Block Decryption	461000ns	3968	3.69Mb/s	252272ns	2453	5.95Mb/sec

Table 2: Decryption results using one IXP1200 Microengine.

	DES			AES		
6-IXP	**Latency**	**IC**	**Throughput**			
Key Setup	412405ns	93000	N/A	92460	21000	N/A
1st 6-Block Encryption	430613ns	4407	20.23Mb/s	271028ns	2713	32.43Mb/sec
2nd 6-Block Encryption	446731ns	4407	20.26Mb/s	270159ns	2713	32.56Mb/sec
3rd 6-Block Encryption	463425ns	4407	20.26Mb/s	268225ns	2713	32.89Mb/sec

Table 3: Encryption results using 6 IXP1200 Microengines.

	DES			AES		
6-IXP	**Latency**	**IC**	**Throughput**	**Latency**	**IC**	**Throughput**
Key Setup	410955ns	93000	N/A	185964ns	42110	N/A
1st 6-Block Decryption	431421ns	4582	19.35Mb/s	280692ns	2890	30.82Mb/sec
2nd 6-Block Decryption	447968ns	4582	19.46Mb/s	284945ns	2890	30.13Mb/sec
3rd 6-Block Decryption	464793ns	4582	19.39Mb/s	277832ns	2890	31.05Mb/sec

Table 4: Decryption results using 6 IXP1200 Microengines.

	DES			AES		
8-IXP2400	**Latency**	**IC**	**Throughput**			
Key Setup	157241ns	91213	N/A	37493ns	21000	N/A
1st 8-Block Encryption	164894ns	4230	72.22Mb/s	42148ns	2698	112.09Mb/sec
2nd 8-Block Encryption	174713ns	4230	72.22Mb/s	46658ns	2698	112.09Mb/sec
3rd 8-Block Encryption	182423ns	4230	72.22Mb/s	51280ns	2698	112.09Mb/sec

Table 5: Encryption results using all eight IXP2400 Microengines.

	DES			AES		
8-IXP2400	**Latency**	**IC**	**Throughput**	**Latency**	**IC**	**Throughput**
Key Setup	157241ns	91213	N/A	73242ns	41732	N/A
1st 8-Block Decryption	164642ns	4379	68.27Mb/s	78042ns	2864	116.32Mb/sec
2nd 8-Block Decryption	171964ns	4379	68.27Mb/s	82142ns	2864	116.32Mb/sec
3rd 8-Block Decryption	179087ns	4379	68.27Mb/s	86965ns	2864	116.32Mb/sec

Table 6: Decryption results using all eight IXP2400 Microengines.

SW Design	**IXP Core Power**
DES or AES on one Microengine in IXP1200	1.8W
DES or AES on six Microengines in IXP1200	3.2W

Table 7: IXP Core Power Consumption

HW Design	**Lat.**	**Th/put**	**Power**	**Area**
AO SDES(PERM)	65.6ns	0.97Gb/s	31.74mW	37Kμm^2
PO SDES(PERM)	28.8ns	35.7Gb/s	661.69mW	595Kμm^2
PO SDES(OD)	27.28ns	32.27Gb/s	331.4mW	675Kμm^2

Key:
AO SDES(PERM): Area-Optimized Synchronous DES based on Permutations.
PO SDES(PERM): Performance-Optimized Sync. DES based on Permutations
PO SDES(OD): Performance-Optimized Sync. DES; Our Design.

Table 8: DES Synchronous Designs: Synthesis Results

HW Design	**Lat.**	**Th/put**	**Power**	**Area**
AO SDES(PERM)	70.4ns	0.91Gb/s	12.55mW	52Kμm^2
PO SDES(PERM)	41.6ns	24.6Gb/s	553.3mW	842Kμm^2
PO SDES(OD)	49.2ns	21.46Gb/s	412.1mW	912Kμm^2

Key:
AO SDES(PERM): Area-Optimized Synchronous DES based on Permutations.
PO SDES(PERM): Performance-Optimized Sync. DES based on Permutations
PO SDES(OD): Performance-Optimized Sync. DES; Our Design.

Table 9: DES Synchronous Designs: Placed & Routed Results

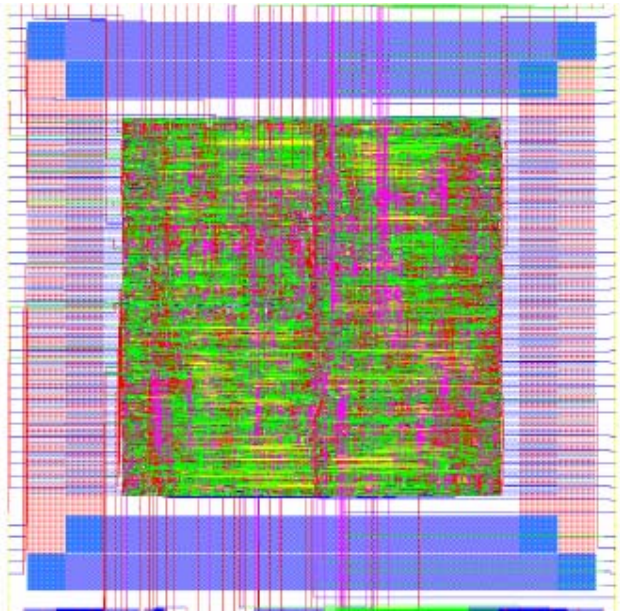

Fig. 1: Layout of our Performance Optimized Synchronous DES Implementation