Available online at www.sciencedirect.com

SCIENCE DIRECT®

Microprocessors and Microsystems 28 (2004) 561–571

MICROPROCESSORS AND MICROSYSTEMS

www.elsevier.com/locate/micpro

Design-space exploration of the most widely used cryptography algorithms

I. Papaefstathiou*, V. Papaefstathiou, C. Sotiriou

Foundation for Research and Technology—Hellas (ICS-FORTH), Vassilika Vouton, 71110 Heraklion, Crete, Greece

Available online 11 September 2004

Abstract

Network data are, 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 the two most widely used encryption algorithms, DES and AES, both in software and in hardware. We present software implementations of the algorithms running on two of the state-of-the-art Intel IXP Network Processors and 11 hardware realizations based on a standard-cell library. In particular, five of our hardware realizations are conventional flip-flop based clocked designs, whereas the other six are either asynchronous, or latch-based synchronous designs. We demonstrate that the most efficient realization of the DES algorithm is one of the proposed asynchronous hardware implementations, whereas for the AES algorithm the latch-based design presented seems to be optimal. 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 and 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.

Keywords: Security protocols; AES; DES; Asynchronous circuits

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 [9], 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 [11].

In this paper we perform a design space exploration of the most widely used cryptography algorithms, namely the DES and the AES [24]. We have implemented both algorithms in both software and hardware and report on the performance and power consumption (and area for the hardware implementations) for each. This design exploration includes, for completeness, the DES software results and a number of the DES implementations is also presented in Ref. [20]. Moreover, in this paper we investigate a number of additional implementations of the DES

* Corresponding author. Tel.: +30 6944277722; fax: +30 2810391609.
E-mail addresses: ygp@ics.forth.gr (I. Papaefstathiou), papaef@ics.forth.gr (V. Papaefstathiou), sotiriou@ics.forth.gr (C. Sotiriou).

0141-9331/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2004.08.009

algorithm, we present new, and very interesting as Section 3 shows, results regarding the placement and routing of all the DES implementations and, more importantly, we also explore all the different approaches for the realization of the AES algorithm, both in software and in hardware. Therefore, we believe that this paper presents a complete review for all the currently, widely used encryption and decryption frameworks.

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. Our hardware realizations consist of 11 different hardware implementations; five flip-flop based conventional clocked implementations, four unclocked (asynchronous) implementations and two latch-based synchronous ones. All of those implementations have not only been synthesized, but also placed and 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 an asynchronous one, while for the AES the most effective implementation seems to be a latch based synchronous 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 and 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 a 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 and AES) in either software, 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 Mb [2,16]. When moving to the FPGA implementations, the fastest DES ciphers presented can provide 12 Gb/s of useful bandwidth [15], but with key setup latency in the order of milliseconds, or even 21 Gb/s [17] 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/s [10,18] even though they are implemented on relatively old CMOS processes (at 0.6 and 0.35 μm). 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 40 Gb/s.

The new AES algorithm has been given, in the last couple of years, much attention. However, even quite 'exotic' software implementations for AES [1,7] exhibit a relatively low throughput of less than 250 Mb/s. Moving to the FPGA implementations, the most complex designs (with a hardware complexity significantly higher than those presented in this paper) can achieve throughputs as high as 3.65 Gb/s [4,19]. On the ASIC side there are some impressive implementations that claim to support up to 70 Gb/s. An IBM core [14] fabricated in the next generation cooper 0.13 μm technology, supports 10 Gb/s but with an exotic S-Box architecture and a clock of 780 Mhz while, probably, consuming more than 1 W of power. More recently, a processor that supports 30–70 Gb/s with minimum area cost, in 0.18 μm technology [8], has been presented. What probably makes this core hardly practical is the fact that the key scheduling is done offline. On our implementations we used a commodity synthesis and backend flow for semi-custom design in a standard 0.18 μm technology. Our goal, for the AES hardware implementations, was to achieve relatively high throughput with very low power consumption; as a result the best of the presented designs can support (after placement and routing) up to 2.5 Gb/s with a power consumption of about 100 mW. Moreover, our designs can easily be pipelined, which will, as Section 4 demonstrates, results in a bandwidth up to 10 times higher (and obviously a similar increase in the power consumption).

3. DES and AES software implementations

In this section we explore the performance of the two encryption algorithms implemented in software running on two Network Processors, a commodity and a state-of-the-art one. The commodity network processor of choice is the widely used Intel IXP-1200 [9]. 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 1 Gb/s.

Table 1
Encryption results using one IXP1200 microengine

1-IXP	DES			AES		
	Latency (ns)	IC	Throughput (Mb/s)	Latency (ns)	IC	Throughput (Mb/s)
Key setup	412,405	93,000	N/A	92,460	21,000	N/A
First block encryption	428,038	3808	3.74	266,662	2394	5.98
Second block encryption	444,032	3808	3.84	265,237	2394	6.12
Third block encryption	460,026	3808	3.88	271,802	2394	5.83

Table 2
Decryption results using one IXP1200 microengine

1-IXP	DES			AES		
	Latency (ns)	IC	Throughput (Mb/s)	Latency (ns)	IC	Throughput (Mb/s)
Key setup	410,955	93,000	N/A	185,964	42,110	N/A
First block decryption	427,669	3970	3.62	264,320	2453	5.86
Second block decryption	444,334	3968	3.71	252,942	2453	5.93
Third block decryption	461,000	3968	3.69	252,272	2453	5.95

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, and 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 StrongARM 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 Mb/s, or Wireless LAN, 802.11a running at 52 Mb/s.

We have also, measured the performance of the encryption algorithms in the state-of-the art, very recently introduced, IXP processors [9] (the IXP-2xxx family) that has just been manufactured in Intel’s 0.13 μm technology.

Table 3
Encryption results using 6 IXP1200 microengines

6-IXP	DES			AES		
	Latency (ns)	IC	Throughput (Mb/s)	Latency (ns)	IC	Throughput (Mb/s)
Key setup	412,405	93,000	N/A	92,460	21,000	N/A
First 6-block encryption	430,613	4407	20.23	271,028 ns	2713	32.43 Mb/sec
Second 6-block encryption	446,731	4407	20.26	270,159 ns	2713	32.56 Mb/sec
Third 6-block encryption	463,425	4407	20.26	268,225 ns	2713	32.89 Mb/sec

Table 4
Decryption results using 6 IXP1200 microengines

6-IXP	DES			AES		
	Latency (ns)	IC	Throughput (Mb/s)	Latency (ns)	IC	Throughput (Mb/s)
Key setup	410,955	93,000	N/A	185,964	42,110	N/A
First 6-block decryption	431,421	4582	19.35	280,692	2890	30.82
Second 6-block decryption	447,968	4582	19.46	284,945	2890	30.13
Third 6-block decryption	464,793	4582	19.39	277,832	2890	31.05

Table 5
Encryption results using all eight IXP1200 microengine

8-IXP2400	DES			AES		
	Latency (ns)	IC	Throughput (Mb/s)	Latency (ns)	IC	Throughput (Mb/s)
Key setup	157,241	91,213	N/A	37,493 ns	21,000	N/A
First 8-block encryption	164,894	4230	72.22	42,148 ns	2698	112.09 Mb/s
Second 8-block encryption	174,713	4230	72.22	46,658 ns	2698	112.09 Mb/s
Third 8-block encryption	182,423	4230	72.22	51,280 ns	2698	112.09 Mb/s

The results of our experiments are shown in Tables 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 IC 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 80 Mb/s for the DES and 120 Mb/s 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 Ref. [26]. 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 W and a typical power consumption of approximately 2.4 W. 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.

4. DES synchronous hardware implementations

In this section we present several hardware implementations of the DES algorithm. We implemented two types of

Table 6
Decryption results using all eight IXP2400 microengines

8-IXP2400	DES			AES		
	Latency (ns)	IC	Throughput (Mb/s)	Latency (ns)	IC	Throughput (Mb/s)
Key setup	157,241	91,213	N/A	73,242	41,732	N/A
First 8-block decryption	164,642	4379	68.27	78,042	2864	116.32
Second 8-block decryption	171,964	4379	68.27	82,142	2864	116.32
Third 8-block decryption	179,087	4379	68.27	86,965	2864	116.32

Table 7
IXP core power consumption

SW design	IXP core power (W)
DES or AES on one microengine in IXP1200	1.8
DES or AES on six microengines in IXP1200	3.2

hardware implementations, synchronous (clocked) and asynchronous (unclocked). All designs were synthesized, using Synopsys [25]; placed and routed, using Cadence's Silicon Enseble [3], and targeted to the 0.18 μm VST-UMC [5] technology library. We also present, separately, the results produced by the synthesis tools and those produced by the placement and 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.

4.1. Synchronous hardware implementations

We implemented three synchronous implementations of the DES algorithm. Two of these 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 Section 4.2. 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

Table 8
DES synchronous designs: synthesis results

HW design	Latency (ns)	Th/put (Gb/s)	Power (mW)	Area (Kμm^2)
AO SDES(PERM)	65.6	0.97	31.74	37
PO SDES(PERM)	28.8	35.7	661.69	5952
PO SDES(OD)	27.28	32.27	331.4	675

Key. AO SDES(PERM), Area-Optimized synchronous DES based on permutations; PO DES(PERM), Performance-Optimized synchronous DES based on permutations; PO SDES(OD), Performance-Optimized synchronous DES, our design.

Table 9
DES synchronous designs: placed and routed results

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

Key. AO SDES(PERM), Area-Optimized synchronous DES based on permutations; PO SDES(PERM), Performance-Optimized synchronous DES based on permutations; PO SDES(OD), Performance-Optimized synchronous DES, our design.

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 Section 4.2[1]. 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 design 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 [23]. 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 Fig. 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.

4.2. DES asynchronous hardware implementations

The reasons for exploring asynchronous implementations of encryption algorithms, and not only the conventional synchronous designs, are the following:

[1] This 'Coarse Grain' asynchronous design comes from a synchronous one to which we have applied the 'desynchronization' method, also presented in Section 4.2.

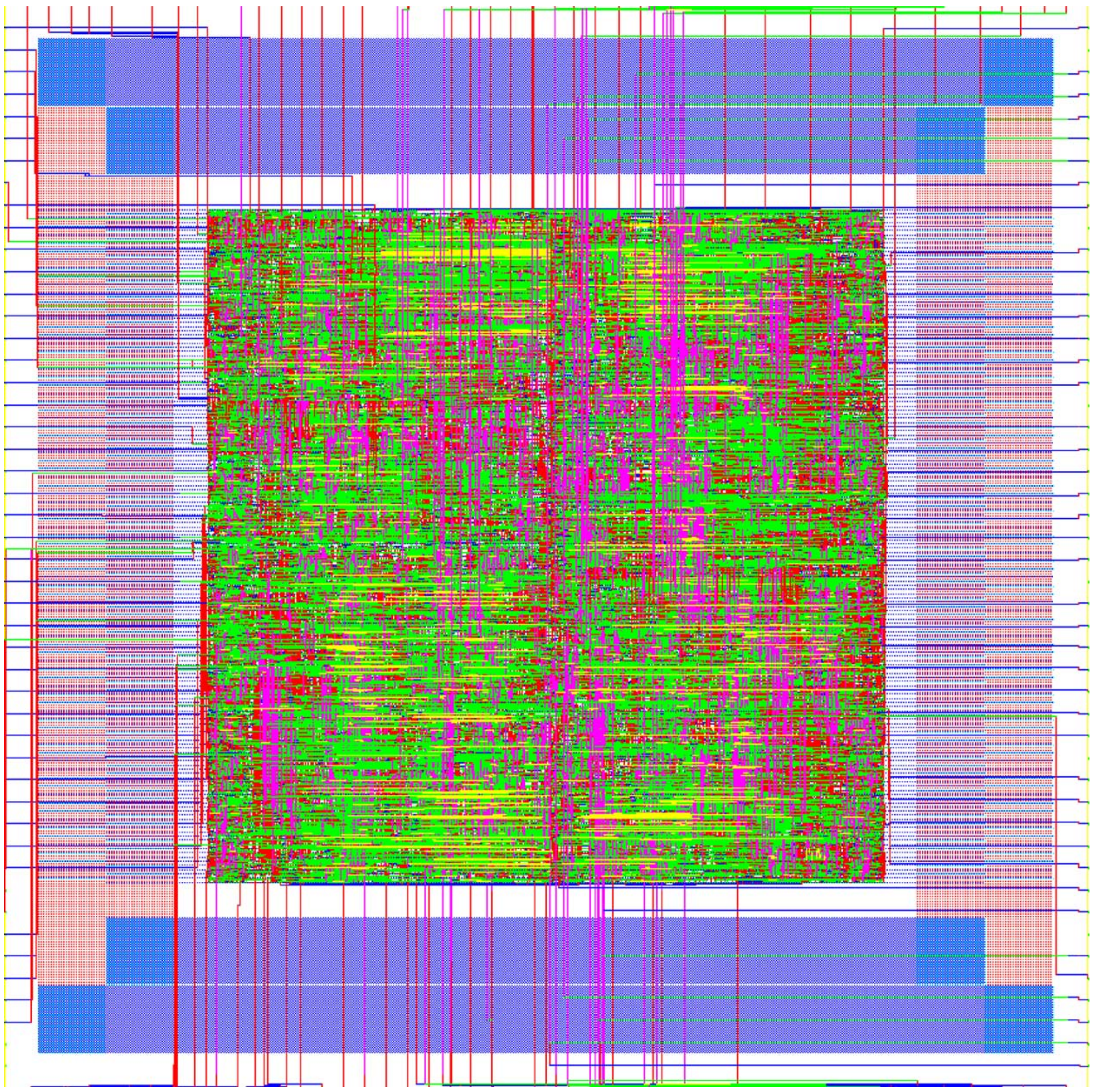
Fig. 1. Layout of our Performance-Optimized synchronous DES implementation.

asynchronous circuits have several properties, that make them desirable in mobile and security applications. The absence of a central synchronization mechanism relieves the designer from the need to distribute one or more clocks, with negligible skew, to every sequential elements of the circuit. This results in a considerable saving of power, since low-skew drivers are extremely power-hungry. Asynchronous circuits also exhibit dramatic improvements in terms of EME. This is due to the fact that flip-flops no longer switch in phase, thus reducing noise power. The absence of a reference point also makes it harder to attack a secure device, such as a public-key encryptor or decryptor, in order to identify its secret keys, by analyzing current absorption at specific points during the clock cycle. All these advantages are offset by a traditionally harder design cycle.

4.2.1. De-synchronization

We implemented asynchronous versions of the DES algorithm by exploiting the methodology of de-synchronization. De-synchronization is a design technique that replaces the clock distribution tree of a traditional synchronous circuit is replaced by a local synchronization mechanism, built out of very simple standard handshaking circuits [22]. This idea has been discussed in the past in Ref. [13], where it was suggested to replace each gate (or combinational logic block in an FPGA-based implementation) of a synchronous circuit with a complex sequential circuit. Similarly [12] proposed a design flow that used synchronous tools for synthesis, and then replaced each combinational gate in the optimized circuit with a sequential majority-gate-based sub-circuit.

De-synchronization works at the level of combinational logic and multi-bit registers. Any synchronous circuit may be de-synchronized by doing the following steps. Firstly, control part and datapath are separated. Next, the datapath is implemented by employing the de-synchronization approach (i.e. each register is clocked individually at the right time by an asynchronous control circuit). Timing the combinational logic delay required to produce data between pairs of registers may be implemented using delay elements. Lastly, the asynchronous control circuitry is designed using an asynchronous design approach, in our case direct-mapped AFSMs [21].

4.2.2. Asynchronous DES designs

We implemented four asynchronous DES versions, an Area-Optimized non-pipelined design, a Fine-Grain pipelined design and two versions of a Coarse-Grain (or Performance-Optimized) pipelined design, one flip-flop based and one latch based. All four designs were implemented bottom-up in VHDL. Each showed a different trade-off between the complexity of the control part and the size of the datapath. This resulted in using handshaking-based local clocking at different levels of granularity. Those designs are totally different from the synchronous ones.

The Area-Optimized version aims to achieve minimum area for a DES algorithm implementation. The asynchronous Area-Optimized DES is composed of three datapath modules, RL, KEY and F and a DES control unit, as shown in Fig. 2.

In our Area-Optimized DES, the algorithm is implemented iteratively with modules RL and KEY internally feeding back their results. The KEY module uses the result of the current iteration to produce the next DES key. The RL module uses the result of the current iteration and the output of the F module to produce the next values for variables R and L, the right and left parts of the encrypted data. According to the DES algorithm, the encrypted output data are produced after 16 iterations by joining R and L. The control unit was designed using AFSMGEN as a complex AFSM, that counts the RL and KEY module handshakes (by sampling the reqout_RL and reqout_K signals, registering the count and then outputing the new reqout_RL_c and reqout_K_c signals) and after registering 16 iterations, it de-asserts the feedback signal, outputs a done signal and generates the output request signal.

The Fine-Grain pipelined DES is derived from the Area-Optimized version by replicating its three datapath blocks (i.e. RL, KEY and F 16 times), as in Fig. 3. This produces a Fine-Grain pipeline (of smaller grain than a DES iteration). By unrolling the datapath, the need for a separate control unit is eliminated and the only control signals are the handshakes between the RL, KEY and F stages.

The Performance-Optimized or Coarse-Grain version of the asynchronous DES squashes the logic and handshakes of the RL, KEY and F modules of the Fine-Grain version into a single stage of logic and a single register, thus removing internal handshaking overhead. There are two versions of the Performance-Optimized asynchronous DES; a version implemented using edge-triggered flip-flops and one using level-sensitive latches. The high-level structure of

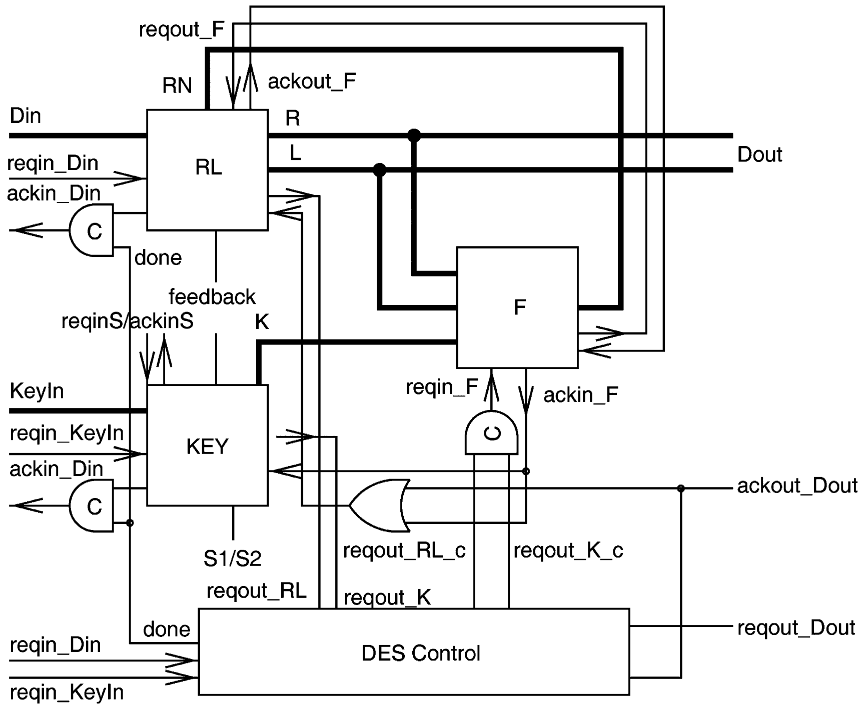

Fig. 2. Area-Optimized asynchronous DES.

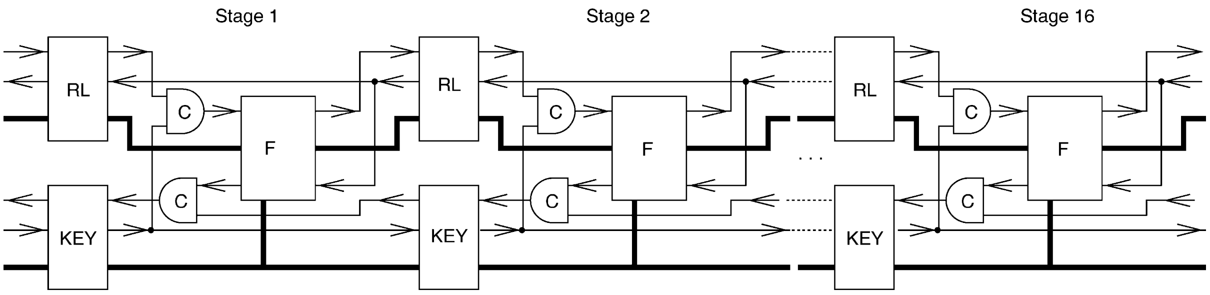

Fig. 3. Fine-Grain pipelined asynchronous DES.

the Performance-Optimized asynchronous DES is shown in Fig. 4.

Table 10 contrasts the characteristics of these four asynchronous DES designs. Note that the power consumption numbers shown have been produced by measuring the actual circuit consumption after placement and routing.

The additional asynchronous control required to implement the Area-Optimized and Fine-Grain pipelined ADES designs implies a significantly high cost in performance (and control complexity is indeed higher than the datapath complexity, which is only a few levels of logic). Thus, the performance of these versions is control-bound. This is somewhat similar to the software implementations.

However, results are particularly striking for the Performance Optimize (PO) versions. The asynchronous versions have comparable performance and the flip-flop version exhibits a significant power improvement. Note that even by lowering the voltage of the synchronous version in order to match its performance to the asynchronous power-optimized ADES, the power advantage of the latter remains significant. The difference is essentially due to the power consumed by the clock tree. The latch design is faster as latches have a shorter propagation delay, however, it consumes more power than the equivalent edge-triggered versions because in our implementation latches are normally open and close when data arrive, so arriving inputs can potentially cause a large number of transitions in the asynchronous pipeline.

When comparing those results with the synchronous ones, we see that the flip-flop based Performance-Optimized asynchronous design has a higher performance to power ratio than the best synchronous one by about 12%, while they cover about the same silicon area.

The asynchronous versions, as opposed to their synchronous counterparts, consume almost no power when idle (as expected), whereas for the synchronous DES designs presented earlier in the paper, power consumption does not drop significantly. For example, we performed an experiment where we exercised our synchronous Area-Optimized DES with a 50% throughput (16 busy followed by 16 idle cycles); the power consumption actually measured after P&R dropped to 171.38 mW; for the asynchronous it dropped to 12.78 mW.

It should be noted that in this section we do not present the results of the synthesis tool since the commercial synthesizers cannot accurately predict the performance of the final design due to the fact that their predictions are mainly based on the notion of one or a small number of global clocks and in our designs we do not have such clocks.

As all the results, for both the synchronous and asynchronous designs, clearly demonstrate, the fastest implementation of the DES algorithm can service up to 38.78 Gb/s of network data. Since, according to Ref. [6], more than 5%, in average, of the network traffic should not be encrypted (i.e. the headers of packets or the minimum size ‘ack’ packets in TCP/IP networks), we claim that our device can fully utilize even the state-of-the-art network speeds of 40 Gb/s (0C-768), while consuming much less than half a Watt of power.

4.3. AES hardware implementations

In this section we present four different implementations of the AES algorithm. Two of them are conventional flip-flop based synchronous designs; one is optimized for

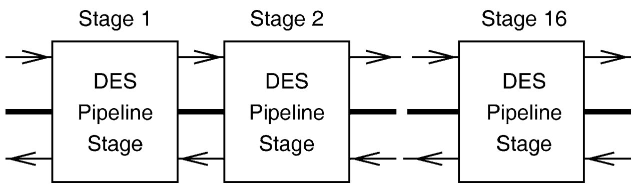

Fig. 4. Coarse-Grain pipelined asynchronous DES.

Table 10
DES asynchronous designs: placed and routed results

HW design	Latency (ns)	Th/put (Gb/s)	Power (mW)	Area (Kμm^2)
AO DES	138	7.42	23.76	83
FGP DES	69.74	10.42	70.04	882
PO DES(F)	30.83	33.16	156.29	560
PO DES(L)	26.36	38.78	420.8	512

Key. AO DES, Area-Optimized asynchronous DES; FGP DES, Fine-Grain pipelined asynchronous DES; PO DES(F), Performance-Optimized asynchronous DES, flip-flop design. PO DES(L), Performance-Optimized asynchronous DES, latch design.

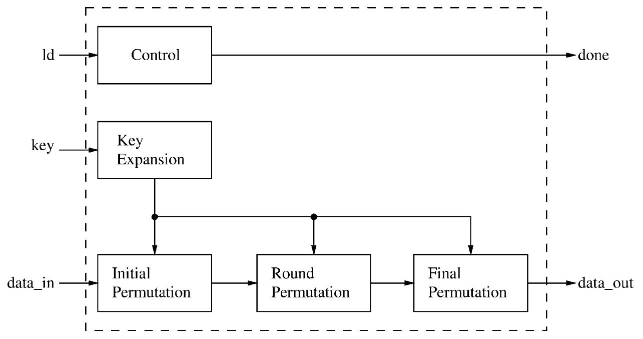

Fig. 5. Encryption AES module.

speed, while the other is optimized for performance. The other two realizations are latch-based designs optimized either for performance or speed.

All those designs are based on the same architecture which is shown in Figs. 5 and 6. As it widely known the AES' encryption and decryption procedures are not symmetrical and therefore different modules are used for encryption and decryption.

The Area-Optimized designs, were produced by slightly altering the initial simple VHDL code implementing the AES, and mainly, by utilizing the most sophisticated features of the Synthesis and placement and routing tools. Several runs of the whole design flow were executed, using different synthesis and P&R parameters, until, what we claim are, the optimal results were produced. In particular, we have set the maximum area in synthesis to 0 and the 'area effort' to the maximum possible, whereas in the P&R process we have tried using different cells from the library so as to achieve the best 'pitch-match' possible, and group, by hand, the library cells in such a way so as to reduce the routing overhead. Similarly for the Performance-Optimized design, we have not altered the VHDL description of the core but instead we have tried to produce the best synthesis and P&R results by altering the different parameters at those levels so as to balance the latency and throughput of the various submodules. In particular, in the synthesis phase, we have allowed the tool to use as much area as needed, while we have also asked for the fastest possible circuit the tool can produce (we set the 'performance effort' to the maximum possible). In the P&R process, we have handcrafted the produced netlist several times, so as to: (a) use the optimal library cells for our hardware organization (for example very 'strong cells' when driving 'important' wires and smaller when driving less 'important' ones), and (b) reduce the length of the interconnection wires in the critical paths. The latter optimization resulted in higher performance by more than 12%.

In other words, in both cases, we have tried to demonstrate the differences in the final netlist's characteristics, triggered when altering the synthesis and P&R parameters. It should be noted that none of the above designs is pipelined, and that is the reason that the throughput shown in this section is much lower than that of the DES implementations. By using the technique of pipelining and/or parallelism, we can increase the bandwidth supported by a factor of 10 quite easily, just as we did in the DES case. For example, for encrypting a single data item, the AES algorithm must run 10 identical iterations so by plugging 10 of those stand-alone AES cores in a pipeline (which will be perfectly balanced), there will be no data or control dependencies between those stages and therefore the data bandwidth supported will be (almost) 10 times higher than the one achieved by the single AES core.

As Table 11 shows, the throughput of the synthesized Performance-Optimized module is more than 100% higher than that of the Area-Optimized module, while the power consumed is less than 40% higher in the Performance-Optimized design. We thus claim that the different synthesis parameters can severely affect the characteristics of the devices implementing such encryption/decryption algorithms.

In Table 12 we demonstrate the results after placement and routing. It is an evident that the throughput of the PO

Table 11
AES designs: synthesis results

HW design	Latency (ns)	Th/put (Gb/s)	Power (mW)	Area (Kμm^2)
AO Enc AES(FF)	57.6	2.2	85.49	245
AO Dec AES(FF)	96.4	1.33	175.21	426
PO Enc AES(FF)	25.2	5.07	121.7	340
PO Dec AES(FF)	64.8	1.97	206.85	526

Key. AO Enc AES(FF), Area-Optimized encryption AES flip-flop design; AO Dec AES(FF), Area-Optimized Decryption AES flip-flop design; PO Enc AES(FF), Performance-Optimized encryption AES flip-flop design; PO Dec AES(FF), Performance-Optimized decryption AES flip-flop design.

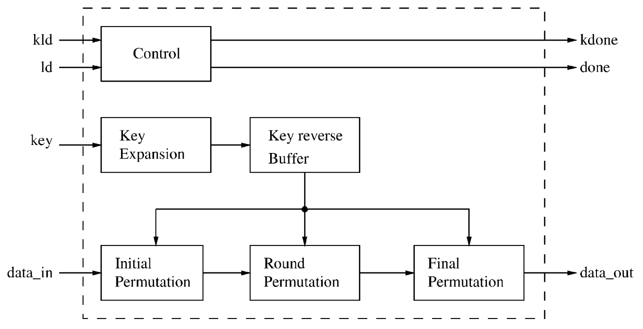

Fig. 6. Decryption AES module.

Table 12
AES designs: placement and routed results

HW design	Latency (ns)	Th/put (Gb/s)	Power (mW)	Area (Kμm^2)
AO Enc AES(FF)	55.2	2.31	79.24	349
AO Dec AES(FF)	110.4	1.15	163.72	602
PO Enc AES(FF)	43.2	2.96	112.48	484
PO Dec AES(FF)	96	1.33	193.45	743

Key. AO Enc AES(FF), Area-Optimized encryption AES flip-flop design; AO Dec AES(FF), Area-Optimized decryption AES flip-flop design; PO Enc AES(FF), Performance-Optimized encryption AES flip-flop design; PO Dec AES(FF), Performance-Optimized decryption AES flip-flop design.

approach was significantly reduced, while the Area-Optimized one was not heavily affected. Moreover, even though the throughput of the PO design was reduced, its power consumption was not. The performance of the PO design after placement and routing does not differ significantly with that of the AO probably due to the fact that in the placed and routed design, the high delay and capacity of the large number of wires were taken into account and since the two designs have similar number/length of wires, their performance and power characteristics were much closer. As a result, even though after synthesis the PO design seems to be the optimal one, in terms of the performance to power ratio, the final netlist (after P&R) shows that the AO design is probably more efficient.

In general, and taken into account the similar results for the DES case, we claim that when implementing an encryption algorithm, the designer should be aware that the placed and routed design, may have performance characteristics that differ by much more than 20%, which is the typical factor used today for the performance/latency discrepancies between the synthesized and the placed and routed netlists. We believe that this is due to the fact that in the implementations of the cryptography algorithms the wire to logic-cells ratio is much higher than in the case of other special or general purpose hardware modules, such as Microprocessors, Memory Management Systems, Schedulers, etc.

Finally, we have implemented two latch-based designs. The reason for producing them was that the latches have lower propagation delay than the flip-flops; therefore the final design would probably be faster. As Tables 13 and 14 show, the latch based designs are indeed up to 10% faster both at the synthesis and the P&R level, than the corresponding flip-flop based designs. Moreover, the power consumed by the Performance-Optimized module is about 7% lower than that of corresponding flip-flop design. This is probably due to the fact that the synthesis tool is using a gated-clock methodology when implementing a module based on latches rather than flip-flops (since it is much easier and safer to 'block' the clock when latches are used), and that certainly reduces the power consumption when a module is idle. Furthermore, the latch-based design can very easily be transformed to an asynchronous one [27] and gain all the advantages of the asynchronous devices such as no power consumption when idle and much lower EME. As a result, we claim that the most efficient implementation of the AES algorithm is this latch-based one.

Table 13
AES designs: synthesis results

HW design	Latency (ns)	Th/put (Gb/s)	Power (mW)	Area (Kμm^2)
AO Enc AES(LA)	52.4	2.37	84.82	269
AO Dec AES(LA)	85.38	1.29	196.47	511
PO Enc AES(LA)	23.8	5.31	129.09	387
PO Dec AES(LA)	63.1	2.03	225.49	641

Key. AO Enc AES(LA), Area-Optimized encryption AES latch design; AO Dec AES(LA), Area-Optimized decryption AES latch design; PO Enc AES(LA), Performance-Optimized encryption AES latch design; PO Dec AES(LA), Performance-Optimized decryption AES latch design.

Table 14
AES designs: placed and routed results

HW design	Latency (ns)	Th/put (Gb/s)	Power (mW)	Area (Kμm^2)
AO Enc AES(LA)	56.2	2.28	84.82	363
AO Dec AES(LA)	92.22	1.39	196.47	656
PO Enc AES(LA)	49.7	2.57	129.09	476
PO Dec AES(LA)	87.6	1.46	225.49	808

Key. AO Enc AES(LA), Area-Optimized encryption AES latch design; AO Dec AES(LA), Area-Optimized decryption AES latch design; PO Enc AES(LA), Performance-Optimized encryption AES latch design; PO Dec AES(LA), Performance-Optimized decryption AES latch design.

5. Conclusions

This paper explored different implementation choices for implementing the most widely used cryptography algorithms, namely the DES and the AES. 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 11 possible hardware implementations, five flip-flop based synchronous, four asynchronous and two latch-based synchronous. The asynchronous designs were designed by employing the method of de-synchronization, whereby we replaced clock signals by asynchronous control.

We have demonstrated that the most efficient implementation of the DES algorithm in terms of data bandwidth serviced and power consumed, is indeed an asynchronous one, albeit with a limited amount of control circuitry, i.e. a 16-stage asynchronous flip-flop-based pipeline. This design can sustain 33.16 Gb/s of throughput at a very modest power consumption of less than 160 mW. For the AES the optimal realization seems to be a latch-based one, which although is optimized for high performance, it consumes a modest amount of power.

References

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

[2] E. Biham, A Fast New DES Implementation in Software, in: Fast Software Encryption, Fourth International Workshop, FSE'97, Haifa, Israel, January 20–22, Proceedings, Volume 1267 of Lecture Notes in Computer Science, Springer, 1997, pp. 260–271.

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

[4] A. Elbirt, W. Yip, B. Chetwynd, C. Paar, An FPGA based performance evaluation of the AES block cipher candidate algorithm finalists, IEEE Transactions on VLSI Systems 9 (4) (2001) 545–557.

[5] EUROPRACTICE. UMC 0.18 μm CMOS technology documentation.

[6] M. Fomenkov, K. Keys, D. Moore, K. Claffy, Longitudinal Study of Internet Traffic from 1998–2003, in: CAIDA Technical Report, September 2003.

[7] B. Gladman, Implementation Experience with AES Candidate Algorithms, in: Proceedings of Second AES Candidate Conference (AES2), 1999.

[8] A. Hodjat, I. Verbauwhede, Minimum Area Cost for a 30 to 70 Gbits/s AES Processor, in: Proceedings of IEEE Computer Society Annual Symposium on VLSI Emerging Trends in VLSI Systems Design (ISVLSI'04), 2004, p. 83.

[9] Intel, The next generation of intel IXP network processors, Intel Technology Journal 6 (3) (2002).

[10] I. Kim, C.S. Steele, J.G.A. Koller, Fully Pipelined 700 MBytes/s DES Encryption Core, in: Proceedings of Ninth Great Lakes Symposium on VLSI, 1999, p. 386.

[11] M.G. Kuhn, Cipher instruction search attack on the bus-encryption security microcontroller DS5002FP, IEEE Transactions on Computers 47 (10) (1998) 1153–1157.

[12] M. Ligthart, K. Fant, R. Smith, A. Taubin, A. Kondratyev, Asynchronous Design Using Commercial HDL Synthesis Tools, in: Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems, IEEE Computer Society Press, April 2000, pp. 114–125.

[13] D.H. Linder, J.C. Harden, Phased logic supporting the synchronous design paradigm with delay-insensitive circuitry, IEEE Transactions on Computers 45 (9) (1996) 1031–1044.

[14] S. Morioka, A. Satoh, A 10 Gbps Full-AES Crypto Design with a Twisted-BDD S-Box Architecture, in: Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'02), 2002, p. 98.

[15] C. Patterson, High Performance DES Encryption in Virtex FPGAs Using JBits, in: Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'00), 2000, p. 113–121.

[16] A. Pfitzmann, R. Amann, More efficient software implementations of (generalized) DES, Computers and Security 12 (5) (1993) 477–500.

[17] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, J.-D. Legat, Efficient uses of FPGAs for implementations of DES and its experimental linear cryptanalysis, IEEE Transactions on Computers 52 (4) (2003) 473–482.

[18] T. Schaffer, A. Glaser, S. Rao, P. Franzon, A Flip-Chip Implementation of the Data Encryption Standard (DES), in: Proceedings of IEEE Multi-Chip Module Conference (MCMC'97), 1997, p. 13.

[19] N. Sklavos, O. Koufopavlou, Architectures and VLSI implementations of the AES-proposal Rijndael, IEEE Transactions on Computers 51 (12) (2002) 1454–1459.

[20] C. Sotiriou, I. Papaefstathiou, A Design-Space Exploration of Alternative Des Implementations, in: Proceedings of 10th IEEE International Conference on Electronics, Circuits and Systems, (ICECS2003), December 2003, pp. 14–17.

[21] C.P. Sotiriou, Implementing Asynchronous Circuits Using a Conventional EDA Tool-Flow, in: Proceedings of the 39th Design Automation Conference, The Association for Computer Machinery, June 2002, pp. 415–418.

[22] C.P. Sotiriou, L. Lavagno, Dc-Synchronization: Asynchronous Circuits from Synchronous Specifications, in: Proceedings of the International IEEE SOC Conference, September 2003.

[23] ST Microelectronics. Notes on Synthesis, Placement and Routing.

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

[25] Synopsys, Design Analyzer Reference Manual (2000).

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

[27] K.Y. Yun, P.A. Beerel, J. Arceo, High-Performance Asynchronous Pipeline Circuits, in: Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems, IEEE Computer Society Press, 1996.