Kimmo Järvinen, Matti Tommiska and Jorma Skyttä, Comparative Survey of High-Performance Cryptographic Algorithm Implementations on FPGAs, IEE Proceedings - Information Security, vol. 152, no. 1, Oct. 2005, pp. 3-12.

Comparative survey of high-performance cryptographic algorithm implementations on FPGAs

K. Järvinen, M. Tommiska and J. Skyttä

Abstract: The authors present a comparative survey of private-key cryptographic algorithm implementations on field programmable gate arrays (FPGAs). The performance and flexibility of FPGAs make them almost ideal implementation platforms for cryptographic algorithms, and therefore the FPGA-based implementation of cryptographic algorithms has been widely studied during the past few years. However, a complete analysis of published implementations has not been presented previously. The authors analyse FPGA-based implementations of certain widely used cryptographic algorithms in terms of speed, area and implementation techniques. The algorithms studied in this article include the private-key cryptographic algorithms advanced encryption standard and international data encryption algorithm and certain hash algorithms. These algorithm implementations provide a good overview of the field of private-key cryptographic algorithm implementation.

1 Introduction

This article presents a thorough study of the state of private-key cryptographic algorithm implementation on field programmable gate arrays (FPGAs). As cryptographic algorithms become more widely used, the need for high-speed implementations of the algorithms increases. Software-based implementations of cryptographic algorithms fall short in performance in many applications, e.g. on heavily loaded servers. Therefore, an obvious need for high-speed implementations exists.

Reprogrammable hardware is almost ideal for cryptographic implementations because high speed can be achieved without significant reduction in flexibility. Flexibility, meaning that the design can be easily changed or modified, is of especially great importance in cryptographic implementations for the following reasons. First, a cryptographic algorithm can be considered secure only until proven otherwise. If a severe flaw in an algorithm is found, the algorithm must be replaced with a more secure one. Second, in many applications, a large variety of different algorithms are in use, and, therefore, it must be easy to change from one algorithm to another.

This article concentrates on implementations of the advanced encryption standard (AES), the international data encryption algorithm (IDEA) and certain hash algorithms. These algorithms represent both private-key cryptographic algorithms and hash algorithms, thus giving a good overview of the state of cryptographic algorithm implementation. All these algorithms have been implemented by the authors of this article in the Signal Processing Laboratory at Helsinki University of Technology, and these designs are here referred to as the SIG designs, e.g. SIG-AES.

Although FPGA-based cryptographic algorithm implementation has been widely studied during the past few years, a thorough comparative study of published implementations has not been presented, at least to the authors' knowledge. The article by Wollinger *et al.* [1] included a review of implementations, but otherwise the article concentrated more on security questions of FPGAs as implementation platforms. In this article, implementations of cryptographic algorithms are compared in terms of speed, area and implementation techniques. Finally, certain conclusions on cryptographic algorithm implementation on FPGAs are presented.

2 Private-key cryptographic algorithms

Implementation of private-key cryptographic algorithms on reprogrammable hardware has been widely studied for several years. Block ciphers are well suited to hardware implementation, because parallelisation, unrolling and pipelining can usually be efficiently exploited.

Throughput can be increased by pipelining an unrolled design and then calculating a different encryption in each pipelined stage. Pipelining, unfortunately, restricts the use of feedback cipher modes which require the value of the previous ciphertext in the generation of the next one, e.g. cipher block chaining mode [2].

The data encryption standard (DES) and its variant 3DES were the most popular block ciphers for decades. Some recent papers on DES implementation have been published, e.g. [3] and [4]. However DES is currently being replaced by AES, and the role of DES will be marginal in the future. Thus, DES is not considered further.

IEE Proceedings online no. 20055004

doi:10.1049/ip-ifs: 20055004

Paper received 23 June 2005

The authors are with the Signal Processing Laboratory, Helsinki University of Technology, Otakaari 5A, 02150 Finland

E-mail: kimmo.jarvinen@hut.fi

2.1 Advanced encryption standard (AES)

AES is a NIST (National Institute of Standards and Technology) standard introduced in 2001 [5]. AES was developed by two Belgians, Vincent Rijmen and Joan Daemen, and it was originally named Rijndael. AES processes a 128-bit data block with a key of either 128, 192 or 256 bits. The different versions of AES are here referred to as AES-128, AES-192 and AES-256. The 128-bit data block is represented as 4×4 rectangular array of bytes called the State. Depending on the key size, AES consists of either 10, 12 or 14 rounds which include four transformations: SubBytes, ShiftRows, MixColumns and AddRoundKey (the last round does not include MixColumns), where the rows and columns refer to the rows and columns of the State. Keys for every round are derived from the original cipher key using the KeyExpansion routine [5].

The SubBytes operation is the most crucial for both the speed and area requirements of an AES implementation [6]. It operates independently on each byte of the State, and it consists of finding a multiplicative inverse in the Galois field $GF(2^8)$ followed by an affine transformation [5]. Traditionally, these operations are combined and implemented as a single 256×8-bit look-up table (LUT) called the S-box. The inverse transformation of SubBytes, called InvSubBytes, is utilised in the AES decryption. It consists of an inverse affine transformation followed by an inversion in $GF(2^8)$ [5].

The first FPGA-based implementations of AES (at that time known only as Rijndael) were published during the selection process of AES. In the last phase of the selection, there were five finalist algorithms: Mars, RC6, Rijndael, Serpent and Twofish. Because all the algorithms were considered secure, hardware efficiency was given great importance in selecting Rijndael as the winning algorithm [7].

During the selection process Dandalis *et al*. [8], Elbirt *et al*. [9], Fischer [10], Gaj and Chodowiec [11] and Mroczkowski [12] published FPGA implementations on both Altera and Xilinx devices. Their studies concluded that Rijndael and Serpent had the highest throughputs [8, 9, 11], while Twofish and RC6 provided compact implementations with medium speed [11]. Mars clearly had the worst hardware characteristics [11].

Since the selection of Rijndael as AES, an enormous number of FPGA-based implementations have been published. Certain trends in these publications are considered next.

Several publications have presented studies of unrolling and pipelining, e.g. [6, 13–19]. The previously mentioned AES finalist algorithm implementations also considered unrolling and pipelining [9, 11]. Very high throughput can be achieved by pipelining unrolled rounds of the algorithm, but, as mentioned, pipelining cannot be efficiently used in feedback modes. To the authors' knowledge, the fastest published FPGA-based implementation of AES was presented by Zambreno *et al*. in [19]. They used aggressive pipelining and achieved throughput of 23.57 Gbps on a Xilinx Virtex-II XC2V4000.

SubBytes can be implemented as an S-box (LUT) which includes precalculated values of the transformation. One 256×8-bit S-box is required for each byte of the State, and therefore 16 parallel S-boxes are required if SubBytes is performed for the entire State at once. Thus, the total number of S-boxes, without KeyExpansion, is 16 times the number of unrolled rounds, and a fully unrolled AES-128 (10 rounds) requires 160 S-boxes. KeyExpansion requires four additional S-boxes per round, and therefore a total number of 200 S-boxes are required in a fully unrolled key agile AES-128 implementation. If S-boxes are implemented on Xilinx FPGAs using BlockRAMs, 100 BlockRAMs are needed, because one dual-port BlockRAM can implement two S-boxes. BlockRAM-based S-boxes have been used in many publications, e.g. [13, 15, 17–23].

Different S-boxes are used in SubBytes, and InvSubBytes, which makes the combination of encryption and decryption difficult without doubling the BlockRAM need. McLoone and McCanny presented in [20] AES-128, AES-192 and AES-256 implementations combining encryption and decryption. They introduced two ROMs including S-box values for encryption and decryption. The BlockRAMs implementing SubBytes and InvSubBytes were programmed using values from ROMs every time encryption was changed to decryption or vice versa [20]. Another solution was presented by Rodríquez-Henríquez *et al*. in [16]. They combined SubBytes and InvSubBytes so that the inversion in $GF(2^8)$ (utilised by both) was implemented in BlockRAMs, but the affine transformations were implemented with logic. This allowed the same BlockRAMs to be utilised in encryption and decryption.

In addition to (Inv)SubBytes, the (Inv)MixColumns transformation can also be performed using the LUT approach. An LUT combining (Inv)SubBytes and (Inv)MixColumns is called the T-box. Fischer and Drutarovský studied implementation techniques based on S-boxes and T-boxes on an Altera FPGA in [24]. They concluded that slightly faster performance was attained with the T-box approach, but the memory need increased [24]. In [25], McLoone and McCanny presented an AES-128 encryption implementation utilising T-boxes. Their implementation had high throughput and occupied only a small number of slices, but it required a very large number of BlockRAMs. A device with a large amount of embedded memory, e.g. the Virtex-E Extended Memory [26], is therefore required.

In the above implementations, SubBytes was implemented as an LUT. Another approach is to calculate the multiplicative inverse and the affine transformation using combinatorial logic. Inversion in $GF(2^8)$ can be reduced into an inversion in $GF(2^4)$ or in $GF(2^2)$ accompanied by Galois field additions and multiplications. That is, the problem is mapped from $GF(2^8)$ to another representation of the field, which in these cases is either $GF((2^4)^2)$ or $GF(((2^2)^2)^2)$. This approach is here referred to as combinatorial implementation of SubBytes, and certain implementations using such methods are considered next.

In [27], we presented a design called SIG-AES which implements SubBytes combinatorially as suggested in [28]. Because this approach requires mappings from $GF(2^8)$ to $GF((2^4)^2)$ and vice versa, other transformations were also mapped to $GF((2^4)^2)$ in order to reduce area and latency. Hodjat and Verbauwhede explored the optimal number of pipelined stages in the combinatorially implemented SubBytes in [15], and they compared combinatorial implementation with implementation using BlockRAMs. Zhang and Parhi presented a careful analysis of the combinatorial implementation of SubBytes and introduced highly optimised implementations, one of which exceeds 20 Gbps on a Virtex-E FPGA [29]. The high efficiency was attained through detailed analysis and careful implementation using combinatorial SubBytes in $GF((2^4)^2)$.

The largest benefit of the combinatorial implementation is that the SubBytes can be pipelined and thus higher throughput can be attained. This does, however, increase the latency of the implementation. The slice requirements also increase compared with BlockRAM-based implementations, because SubBytes is implemented with logic.

In many applications, it is more important to minimise area than to maximise throughput. Therefore, several implementations with small logic requirements have been published. Pramstaller and Wolkerstorfer presented a compact implementation of AES encryption and decryption with all key lengths using a novel State representation, which solves the problem of accessing both rows and columns of the State [30]. A very compact implementation was presented by Chodowiec and Gaj in [31]. They efficiently exploited the structure of FPGA and were able to fit AES-128 encryption and decryption into 222 slices and three BlockRAMs on a low-cost Xilinx Spartan-II XC2S30-5/6. The design achieved a throughput of 166 Mbps on an XC2S30-6 [31]. At least to the authors' knowledge, the most compact AES implementation published in the literature so far was presented by Rouvroy *et al*. in [32]. They were able to fit AES-128 with KeyExpansion into only 163 slices and three BlockRAMs on a Xilinx Spartan-3 XC3S50-4 and achieved a throughput of 208 Mbps. They also implemented the same design on a Virtex-II, and it has been included in the comparison below. The key to lower area consumption compared with [31] was the combination of SubBytes and MixColumns transformations [32]. The throughputs of the designs are not comparable because different FPGAs were used. Even Gbps-level throughputs can be achieved with small logic requirements as was shown by Standaert *et al*. in [33], where a throughput of 2.085 Gbps was achieved with only 1769 slices. Other compact designs targeting resource-limited FPGAs include [34] and [35]. Also many of the above-mentioned papers include implementations requiring a low area.

The comparison of different AES implementations is hard for many reasons. First, the large variety of different target devices makes a fair comparison difficult. Second, many authors do not specify their devices well enough to ensure easy comparison, e.g. the size or the speed grade of the device has not been provided. Third, comparison of area requirements is difficult because both slices and embedded memory, i.e. BlockRAMs in the Xilinx devices, are used.

Xilinx Virtex-family FPGAs (i.e. the Virtex [36], Virtex-E [37], Virtex-E Extended Memory (Virtex-EM)[26] and Virtex-II [38]) are clearly the most used implementation platforms for the published designs. Therefore, this comparison concentrates on designs implemented on these devices. Performances on different devices should not be compared, because the device greatly determines the performance of an implementation. Devices are therefore clearly differentiated in the tables and figures in this paper.

A summary of open-literature FPGA-based AES implementations on Xilinx Virtex-family devices is presented in Table 1, and it includes implementations published in [6, 8, 9, 11, 13–23, 25, 27, 29, 30, 32, 33, 39, 40]. In order to compare the area requirements of BlockRAM-based implementations with those of implementations which use only slices, a method introduced in [17] is used. Because a dual-port 256×8-bit BlockRAM can be replaced by distributed memory consisting of 256 LUTs, one BlockRAM can be replaced by 128 slices [17]. Thus, the area value in Table 1 was calculated using the following formula:

$$\text{area} = \text{slices} + 128 \times \text{BlockRAMs} \quad (1)$$

The performance–area relationship is studied using two different metrics. The first one is the traditional throughput per slice (TPS) value [9]. The other, which also takes into account the BlockRAM utilisation, is called the throughput per area (TPA) value and is calculated using the area value obtained using (1). TPA offers a better impression of the performance–area relationship than TPS, which neglects the usage of BlockRAMs. An extreme example of this is the implementation presented in [25], which attains an extremely high TPS value but nonetheless requires a large target device because of the large number of BlockRAMs. However, in such extreme cases, the TPA method yields estimates that are too pessimistic and, therefore, TPA should be used together with TPS to ensure fair comparison.

Throughput-slice and throughput-area scatters are presented in Figs 1 and 2, respectively. There exists only little correlation between slice usage and throughput in Fig. 1. In Fig. 2, however, there is a significantly higher correlation, which again validates the use of TPA.

As stated earlier, the fastest reported AES implementation achieves a throughput of 23.57 Gbps with 16 938 slices on a Virtex-II XC2V4000 [19]. Although it is the fastest implementation, it is not the most efficient of the high-throughput implementations, if TPS and TPA are considered as the efficiency metrics. As can be seen from Fig. 2, implementations by Hodjat and Verbauwhede [15] and Zhang and Parhi [29] achieve almost the same level of throughput with fewer logic resources. Considering the high TPS and TPA values as well as the slower Virtex-E device compared with the Virtex-II devices used in [15] and [19], Zhang's design can be considered the most efficient fully unrolled and pipelined AES-128 implementation published so far.

Combinatorial implementation of SubBytes results in higher TPA values than LUT-based implementation. This is due to the fact that SubBytes can be pipelined, and therefore very high throughput can be achieved. Combinatorial SubBytes also results in moderate area requirements. This can be seen in Fig. 2, where the combinatorial implementations, i.e. [15, 27, 29], are situated in the upper left corner.

If embedded memory can be used, a considerable reduction in slice requirements can be achieved by using BlockRAM-based S-boxes. Also the latency of S-boxes is shorter than that of the combinatorial SubBytes. The T-box approach seems infeasible if TPA is considered, but a very high TPS can be achieved. T-boxes are therefore very inviting if the slice requirement needs to be minimised in a high-throughput design. The approach presented in [25], however, requires a Virtex-EM XCV812E FPGA because, with the exception of the new Virtex-4 FPGAs [41], no other device in the Xilinx Virtex family contains enough BlockRAMs (244) [26, 36–38].

Many implementations with an area in the range 1600–2000 have been published [17–19, 33, 40]. These compact implementations achieve relatively high throughputs of >1 Gbps on Virtex-E and Virtex-II FPGAs. The implementation by Standaert *et al*. [33] has the highest TPA and throughput among these implementations, as it achieves throughput of 2.085 Gbps with only 1769 slices and no BlockRAMs

Table 1: AES implementations on Xilinx Virtex-family FPGAs

Authors	Key	Device	Slices	BRAM	Area	Throughput (Gbps)	TPS (Mbps/slice)	TPA (Mbps/area)
Chodowiec and coworkers [13, 14]	[Cho]	Virtex 1000-6	12 600	80	22 840	12.16	0.965	0.532
Chodowiec *et al.* [13]	[Cho]	Virtex 1000-6	2 057	8	3 081	1.265	0.615	0.411
Chodowiec and coworkers [13, 14]	[Cho]	Virtex 1000-6	2 507	0	2 507	0.414	0.165	0.165
Dandalis *et al.* [8]	[Dan]	Virtex -6	5 673	0	5 673	0.353	0.062	0.062
Elbirt *et al.* [6, 9]	[Elb]	Virtex 1000-4	10 992	0	10 992	1.938	0.176	0.176
Elbirt *et al.* [6, 9]	[Elb]	Virtex 1000-4	4 871	0	4 871	0.949	0.195	0.195
Gaj and Chodowiec [11]	[Gaj]	Virtex 1000-6	2 902	0	2 902	0.332	0.114	0.114
Hodjat and Verbauwhede [15]	[Hod]	Virtex-II VP20-7	9 446	0	9 446	21.64	2.291	2.291
Hodjat and Verbauwhede [15]	[Hod]	Virtex-II VP20-7	5 177	84	15 929	21.54	4.161	1.352
Järvinen *et al.* [27]	[Jär]	Virtex-II 2000-5	10 750	0	10 750	17.8	1.656	1.656
Järvinen *et al.* [27]	[Jär]	Virtex-E 1000-8	11 719	0	11 719	16.54	1.411	1.411
Labbé and Pérez [39]	[Lab]	Virtex 1000-4	2 151	4	2 663	0.394	0.183	0.148
Labbé and Pérez [39]	[Lab]	Virtex 1000-4	3 543	4	4 055	0.796	0.225	0.196
Labbé and Pérez [39]	[Lab]	Virtex 1000-4	8 767	4	9 279	1.911	0.218	0.206
McLoone and McCanny [20]	[ML1]	Virtex-E 3200-8	2 222	100	15 022	6.956	3.131	0.463
McLoone and McCanny [25]	[ML2]	Virtex-EM 812-8	2 000	244	33 232	12.02	6.010	0.362
McLoone and McCanny [21]	[ML3]	Virtex-EM 812-8	2 679	82	13 175	6.956	2.596	0.528
Pramstaller and Wolkerstrofer [30]	[Pra]	Virtex-E 1000-8	1 125	0	1 125	0.215	0.191	0.191
Rodríquez-H *et al.* [16]	[Rod]	Virtex-E 2600	5 677	80	15 917	4.121	0.726	0.259
Rouvroy *et al.* [32]	[Rou]	Virtex-II 40-6	146	3	530	0.358	2.452	0.675
Saggese *et al.* [17]	[Sag]	Virtex-E 2000-8	2 778	100	15 578	8.9	3.204	0.571
Saggese *et al.* [17]	[Sag]	Virtex-E 2000-8	446	10	1 726	1	2.242	0.579
Saggese *et al.* [17]	[Sag]	Virtex-E 2000-8	5 810	100	18 610	20.3	3.494	1.091
Saggese *et al.* [17]	[Sag]	Virtex-E 2000-8	648	10	1 928	1.82	2.809	0.944
Saqib *et al.* [22]	[Saq]	Virtex-EM 812	2 744	0	2 744	0.259	0.094	0.094
Saqib *et al.* [22]	[Saq]	Virtex-EM 812	2 136	100	14 936	2.868	1.343	0.192
Standaert *et al.* [18]	[St1]	Virtex 1000-6	2 257	0	2 257	1.563	0.693	0.693
Standaert *et al.* [18]	[St1]	Virtex-E 3200-8	2 784	100	15 584	11.776	4.230	0.756
Standaert *et al.* [18]	[St1]	Virtex-E 3200-8	542	10	1 822	1.45	2.675	0.796
Standaert *et al.* [33]	[St2]	Virtex-E 3200-8	1 769	0	1 769	2.085	1.179	1.179
Standaert *et al.* [33]	[St2]	Virtex-E 3200-8	15 112	0	15 112	18.560	1.228	1.228
Wang and Ni [40]	[Wan]	Virtex-E 1000-8	1 857	0	1 857	1.604	0.864	0.864
Weaver and Wawrzynek [23]	[Wea]	Virtex-E 600-8	770	10	2 050	1.75	2.273	0.854
Zambreno *et al.* [19]	[Zam]	Virtex-II 4000	1 254	20	3 814	4.44	3.541	1.164
Zambreno *et al.* [19]	[Zam]	Virtex-II 4000	16 938	0	16 938	23.57	1.392	1.392
Zambreno *et al.* [19]	[Zam]	Virtex-II 4000	2 206	50	8 606	10.88	4.932	1.264
Zambreno *et al.* [19]	[Zam]	Virtex-II 4000	3 766	100	16 566	22.93	6.089	1.384
Zambreno *et al.* [19]	[Zam]	Virtex-II 4000	387	10	1 667	1.41	3.643	0.846
Zhang and Parhi [29]	[Zha]	Virtex 1000-6	11 014	0	11 014	16.032	1.456	1.456
Zhang and Parhi [29]	[Zha]	Virtex 800-6	9 406	0	9 406	9.184	0.976	0.976
Zhang and Parhi [29]	[Zha]	Virtex-E 1000-8	11 022	0	11 022	21.556	1.956	1.956
Zhang and Parhi [29]	[Zha]	Virtex-EM 812-8	9 406	0	9 406	11.965	1.272	1.272

The authors have selected the most relevant implementations (in their opinion) from those publications which include several different implementations. Keys are used in Figs 1 and 2.

on a Virtex-E XCV3200E-8, and thus has a TPA (and TPS) of 1.179 Mbps/area. If even smaller area consumption is required, one most tolerate a large slowdown in throughput. The smallest implementations have throughputs measured in hundreds of Mbps; e.g. Rouvroy *et al.* achieved 358 Mbps with only 146 slices and three BlockRAMs (area 530) on a Virtex-II XC2V40-6 [32].

This survey has shown that various different methods have been presented to implement AES. It is impossible to point out the absolutely best method, because all methods have their advantages and disadvantages. As a conclusions to the AES study it is stated that AES can be efficiently implemented on FPGAs for applications with various requirements. Both very high performance and low area requirements can be efficiently achieved using the methods presented in the literature.

2.2 International Data Encryption Algorithm

IDEA was introduced by Lai and Massay in 1990 [42] and modified the following year [43]. IDEA is considered highly secure, and no published attack

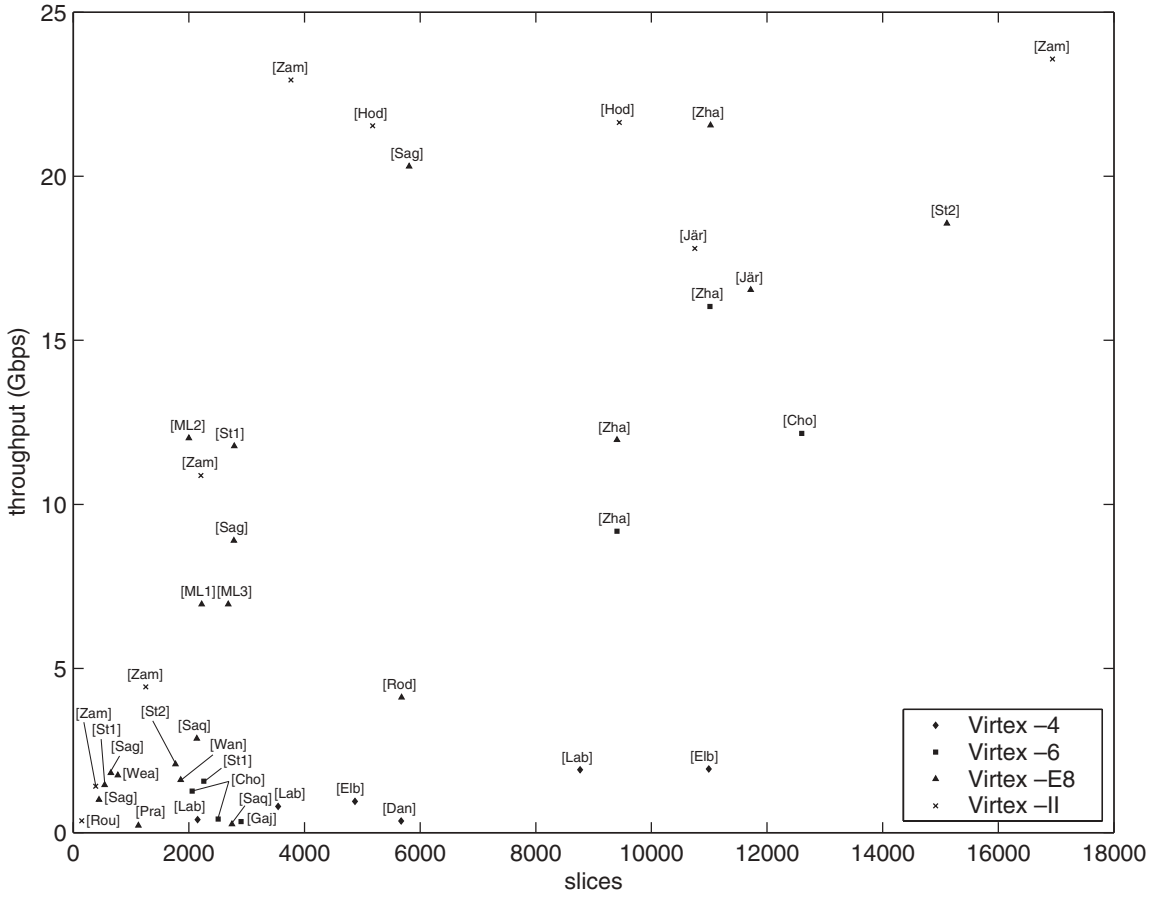

Fig. 1 *Throughput–slice chart of FPGA-based AES implementations*

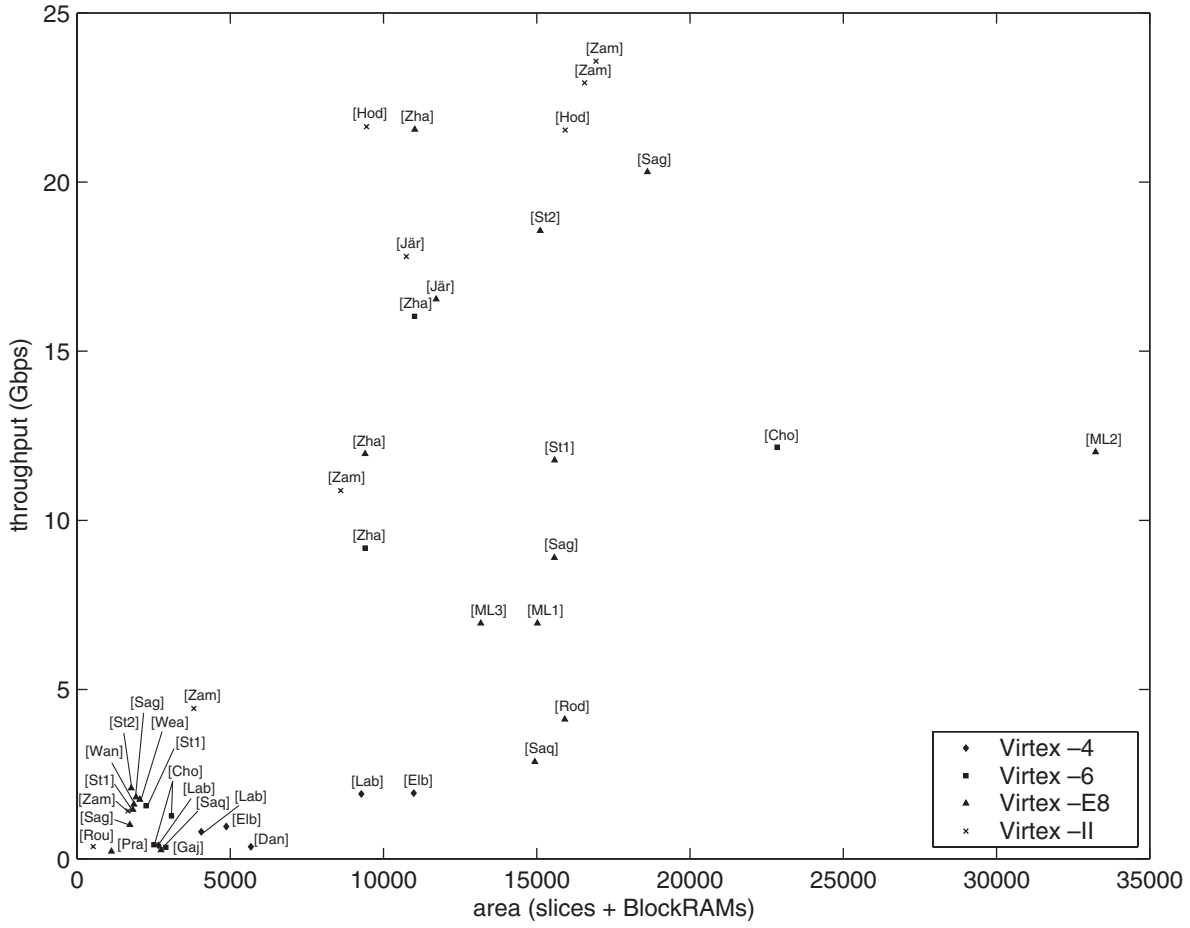

Fig. 2 *Throughput–area chart of FPGA-based AES implementations*

(with the exception of attacks on weak keys) is better than an exhaustive search on the 128-bit key space, which is computationally infeasible. The security of IDEA appears to be bounded only by the weaknesses arising from the relatively small (compared with its keylength) blocklength of 64 bits [44]. It has been stated that before the introduction of the AES, IDEA may have been the most secure private-key cryptographic algorithm available to the public [2].

IDEA encrypts 64-bit plaintext blocks into 64-bit ciphertext blocks using a 128-bit input key K. The algorithm consists of eight identical rounds followed by an output transformation. Each round uses six 16-bit subkeys $K_i^{(r)}$, $1 \leqslant i \leqslant 6$, to transform a 64-bit input X into an output of four 16-bit blocks, which are then input to the next round. All subkeys are derived from the 128-bit input key K. The subkey derivation process is different in decryption mode from the encryption mode, but otherwise encryption and decryption are performed using identical hardware.

IDEA uses only three operations on 16-bit sub-blocks a and b: bitwise XOR, unsigned addition mod (2^{16}) and modulo $(2^{16}+1)$ multiplication. All three operations are derived from different algebraic groups of 2^{16} elements, which is crucial to the algorithmic strength of IDEA. Of the three arithmetic operations, bitwise XOR and unsigned addition mod (2^{16}) are trivial to implement, whereas an implementation of modulo $(2^{16}+1)$ multiplication that is both area efficient and fast requires careful design and bit-level optimisation.

Two early FPGA-based IDEA implementations were published by Mencer *et al.* [45] and Mosanya *et al.* [46] in 1998 and 1999, respectively. Mencer *et al.* studied the benefits and limitations of FPGA systems compared with processors and application-specific integrated circuits (ASICs) using IDEA encryption as a benchmark. Their IDEA implementation had a throughput of 528 Mbps, and it covered four Xilinx XC4020 FPGAs (3200 CLBs) [45]. Mosanya *et al.* presented a reconfigurable cryptoprocessor called CryptoBooster in [46]. They did not present any exact performance figures for their implementation but they estimated that throughputs of 200–1500 Mbps could be achieved on a state-of-the-art FPGA of that time [46].

Representative compact and high-speed FPGA-based implementations of IDEA include a bit-serial implementation described by Leong *et al.* in [47], with a throughput of 500 Mbps on a Xilinx Virtex XCV300-6, and the bit-parallel implementation described by Cheung *et al.* in [48], which achieved throughput of 5.25 Gbps on a Virtex XCV1000-6. The bit-parallel implementation also included bespoke software to customise the FPGA reprogramming bitstream for different key schedules.

In 2002, the SIG-IDEA implementation was described in [49], and, at 6.78 Gbps, its throughput represented the fastest published FPGA-based implementation of IDEA at that time. Other contributions of SIG-IDEA include implementing a fully pipelined algorithm with both inner and outer loop pipelining on a single Xilinx Virtex-E XCV1000E-6 device, the efficient usage of the diminished-one number system and an area-efficient implementation of the modulo (2^{16}) multiplication.

Currently, the fastest FPGA-based implementation of IDEA is probably [50], where Gonzalez *et al.* achieved a throughput of 8.3 Gbps on a Xilinx Virtex XCV600-6 device. The key to high throughput was replacing all the operational units involving the key with its constant-operand equivalents by partial reconfiguration, the overhead for which was 4 ms. However, only a few devices support partial reconfiguration, and the scheme requires a controlling microprocessor. Another recent IDEA implementation by Pan *et al.* achieved a throughput of 6 Gbps by utilising the embedded multipliers for the modulo $(2^{16}+1)$ multiplication algorithm [51].

The FPGA-based IDEA implementations mentioned above are summarised in Table 2.

3 Hash algorithms

Commonly used hash algorithms, e.g. MD5 [52] and the secure hash algorithm (SHA) [53], are not as well suited to high-speed hardware implementations as most of the private-key or public-key algorithms, mainly because parallelisation cannot be used as efficiently. Hash algorithms can be implemented efficiently on software as they use common modulo (2^{32}) additions, which are easy and fast to perform with traditional microprocessors. However, significant accelerations from 25 to 31 times for SHA-1 and SHA-512 have been reported [54].

There are certain applications which greatly benefit from hardware acceleration. For example, if a cryptographic scheme requiring hash calculations, e.g. the digital signature algorithm [55], is implemented on an FPGA, it is well-grounded to implement a hash module on the chip too. Certain very demanding hash calculations, e.g. long chains of hash rounds, benefit from hardware acceleration [56]. Hash algorithm implementations presented in [54, 56–69] are considered here.

A throughput of several hundred Mbps can be achieved with small logic requirements using a basic iterative architecture [54, 56, 57, 63, 65, 67, 69]. However, the throughput and efficiency of an implementation can be increased considerably by partially unrolling the algorithm rounds [54, 58, 66]. This is because the structure of hash algorithms favours

Table 2: IDEA implementations on Xilinx FPGAs

Authors	Device	Slices	Throughput (Gbps)	TPS (Mbps/slice)
Cheung *et al.* [48]	Virtex 1000-6	11 602	5.24	0.452
Gonzalez *et al.* [50]	Virtex 600-6	6078	8.3	1.366
Hämäläinen *et al.* [49]	Virtex-E 1000-6	9855[a]	6.78	0.688
Leong *et al.* [47]	Virtex 300-6	2801	0.5	0.179
Mencer *et al.* [45]	XC4000	n.a. (3200 CLBs)	0.528	n.a.
Pan *et al.* [51]	Virtex-II 1000-6	4221	6.0	1.421

[a] The value is not available in the original publication. Received from the design files.

unrolling; i.e. if k rounds are unrolled, the critical path increases for fewer than k times [54]. Pipelining, however, cannot be used for increasing throughput as efficiently as for block ciphers [56]. Unrolling was used in the fastest published SHA-1 implementations, where Lien *et al*. reported a throughput of 1024 Mbps on a Virtex XCV1000-6 [54] and Sklavos *et al*. achieved a throughput of 1339 Mbps on a Virtex-II XC2V500 with their combined SHA-1 and RIPEMD design [66].

Because many of the commonly used hash algorithms share resources and have a similar kind of structure, many implementations combining several hash algorithms have been proposed. Dominikus presented a general hash processor architecture which can be used for MD5, SHA-1, SHA-256 and RIPEMD calculations in [59]. A general processor architecture naturally achieves slower performance than algorithm-specific implementations, such as [54, 56–58, 67], but algorithm flexibility may be an essential feature in certain applications. A sufficient balance between speed and algorithm flexibility may be achieved by implementing an algorithm-specific design of two or more commonly used algorithms. Implementations combining certain algorithms have been published; e.g. MD5 and SHA-1 were combined in [61, 62, 68], different SHA algorithms were combined in [63] and MD5 was combined with RIPEMD in [64]. Algorithm support of published FPGA-based implementations is presented in Table 3.

The performance and area requirements of FPGA-based open-literature hash algorithm implementations are presented in Table 4. Hash algorithm implementations achieving throughputs of several hundred Mbps require only a minimal amount of logic resources. Hash algorithms with different cryptographic strengths, e.g. SHA-1 and SHA-512, have almost similar throughputs, but stronger algorithms have larger logic requirements [54, 60].

Increasing throughput to several Gbps is difficult because of the structures of commonly used hash algorithms. In high-speed implementations of the AES, for example, aggressive parallelisation, unrolling and pipelining can be used efficiently, whereas the structures of hash algorithms usually make the efficient use of such methods difficult [56]. Even for hash algorithms, parallel hash blocks or unrolling and pipelining can be used for increasing throughput, as we have shown in [56]. However, very high-speed implementations of hash algorithms require considerably more area than implementations of block ciphers, e.g. the AES, of the same speed [56]. This can be verified, for example, by comparing the SIG-MD5 implementation of four parallel MD5 blocks [56] with the AES implementation by Standaert *et al*. [33]. Both have a similar level of throughput (2395 and 2085 Mbps), but the MD5 implementation consumes a lot more area (5732 slices) than the AES implementation (1769 slices).

MD5, RIPEMD and SHA-1 were recently compromised so that finding collisions is possible with much less effort than exhaustive searching [70, 71]; therefore these algorithms can no longer be considered secure. Although finding collisions is not a problem in every application using hash algorithms, e.g. HMAC (the keyed-hash message authentication code), these algorithms will certainly be replaced with stronger ones in the future. Hardware implementation of hash algorithms will probably be studied actively when MD5 and SHA-1 are replaced with new algorithms.

4 Conclusions and future work

We have shown that FPGAs can be used very efficiently for high-speed implementations of cryptographic algorithms. The field has been studied extensively for the past few years and very efficient implementations have been presented regardless of the implemented algorithm.

Similar design methodologies apply to all algorithms studied in this survey. The key to a high-speed implementation is to identify the critical operation, e.g. SubBytes in the AES, and implement it efficiently. In general, operations, or at least the critical operation,

Table 3: Algorithm support of published FPGA-based implementations of hash algorithms

Authors	MD5	SHA-1	SHA-256	SHA-384	SHA-512	RIPEMD	HAS-160
Deepakumara *et al*. [57]	✓						
Diez *et al*. [58]	✓						
Diez *et al*. [58]		✓					
Dominikus [59]	✓	✓	✓			✓	
Grembowski *et al*. [60]		✓					
Grembowski *et al*. [60]					✓		
Järvinen *et al*. [56]	✓						
Järvinen *et al*. [61]	✓	✓					
Kang *et al*. [62]	✓	✓					✓
Lien *et al*. [54]		✓					
Lien *et al*. [54]					✓		
McLoone and McCanny [63]				✓	✓		
Ng *et al*. [64]	✓					✓	
Selimis *et al*. [65]		✓					
Sklavos *et al*. [66]		✓				✓	
Ting *et al*. [67]			✓				
Wang *et al*. [68]	✓	✓					
Zibin and Ning [69]		✓					

Table 4: Performance and area requirements of published FPGA-based implementations of hash algorithms

Authors	Device	Algorithm	Slices	BlockRAMs	Throughput (Mbps)
Deepakumara *et al.* [57]	Virtex 1000-6	MD5	880	2	165
Deepakumara *et al.* [57]	Virtex 1000-6	MD5	4763	0	354
Diez *et al.* [58]	Virtex-II 3000	MD5	1369	0	467.3
Diez *et al.* [58]	Virtex-II 3000	SHA-1	1550	0	899.8
Dominikus [59]	Virtex-E 300	MD5	1004	0	146
Dominikus [59]	Virtex-E 300	RIPEMD	1004	0	89
Dominikus [59]	Virtex-E 300	SHA-1	1004	0	119
Dominikus [59]	Virtex-E 300	SHA-256	1004	0	77
Grembowski *et al.* [60]	Virtex 1000-6	SHA-1	1475[a]	0[a]	462
Grembowski *et al.* [60]	Virtex 1000-6	SHA-512	2826[a]	2[a]	616
Järvinen *et al.* [56]	Virtex-II 4000-6	MD5	1325	0	607
Järvinen *et al.* [56]	Virtex-II 4000-6	MD5	5732	0	2395
Järvinen *et al.* [56]	Virtex-II 4000-6	MD5	11498	10	5857
Järvinen *et al.* [61]	Virtex-II 2000-6	MD5	1882	0	602
Järvinen *et al.* [61]	Virtex-II 2000-6	SHA-1	1882	0	485
Kang *et al.* [62]	Apex 20K 1000-3	MD5	10573 (LE)	0	142
Kang *et al.* [62]	Apex 20K 1000-3	SHA-1	10573 (LE)	0	114
Kang *et al.* [62]	Apex 20K 1000-3	HAS-160	10573 (LE)	0	160
Lien *et al.* [54]	Virtex 1000-6	SHA-1	480	0	544
Lien *et al.* [54]	Virtex 1000-6	SHA-1	1480	0	1024
Lien *et al.* [54]	Virtex 1000-6	SHA-512	2384	0	717
Lien *et al.* [54]	Virtex 1000-6	SHA-512	3521	0	929
McLoone and McCanny [63]	Virtex-E 600-8	SHA-384/512	2914	2	479
Ng *et al.* [64]	Flex 50-1	MD5	1964 (LE)	0	206
Ng *et al.* [64]	Flex 50-1	RIPEMD	1964 (LE)	0	84
Selimis *et al.* [65]	Virtex 150	SHA-1	518	0	518
Sklavos *et al.* [66]	Virtex-II 500	SHA-1	2245	0	1339
Sklavos *et al.* [66]	Virtex-II 500	RIPEMD	2245	0	1656
Ting *et al.* [67]	Virtex-E 300-8	SHA-256	1261	0	693
Wang *et al.* [68]	Apex 20K 1000-3	MD5	3040 (LE)	1 (ESB)	178.6
Wang *et al.* [68]	Apex 20K 1000-3	SHA-1	3040 (LE)	1 (ESB)	143.3
Zibin and Ning [69]	Acex 100-1	SHA-1	1622 (LE)	0	268.99

Notice that 1 slice $\approx$ 2 logic elements (LEs), 1 BlockRAM = 4096 bits [36, 37, 26] and 1 embedded system block (ESB) = 2048 bits [72].
[a] Calculated from the percentages presented in the paper.

should be implemented on as low a level as possible in order to guarantee maximum performance with minimum resources.

Although all the designs considered here were implemented on FPGAs, only a small number of them specifically target FPGAs as they are merely general hardware implementations which could be implemented on ASICs as well. Exploiting the special properties of FPGAs has not yet been thoroughly studied, with the exception of embedded memory usage. However, partial reconfigurability was used in an IDEA implementation [50] as discussed in Section 2.2. Certain implementations which have been optimised especially for the slice structure have been published, e.g. [18].

Many cryptographic algorithms use similar kinds of operations, so it may be possible to combine several algorithms into a single design efficiently by exploiting these similarities. Such combinations of cryptographic algorithms have not been studied extensively, except in the case of hash algorithms.

There exist dedicated embedded blocks for certain commonly used operations in modern FPGAs, e.g. the Altera Stratix-II architecture includes dedicated blocks for digital signal processing (DSP) [73]. Dedicated blocks for cryptography do not yet exist on any device, but if such blocks were implemented they could speed up the performance of cryptographic algorithms substantially. The question of how these blocks should be arranged and which operations should be implemented is an open research problem.

Based on our observations, certain possible research topics in the future include

- architectures for constrained environments
- an increase of generality, general cryptographic architectures implementing several cryptographic algorithms in a single design (cf. MD5/SHA-1 implementations)
- dedicated blocks for cryptographic operations into FPGAs (cf. DSP blocks)
- efficient utilisation of the special abilities of FPGAs (e.g. partial reconfiguration)
- efficient implementations of strong hash algorithms.

It was concluded that regardless of the algorithm, very efficient implementations have been published in terms of both speed and logic requirements. Although cryptographic algorithm implementation has been widely studied, certain open problems remain.

5 Acknowledgments

This paper was written as part of the GO-SEC project at Helsinki University of Technology. GO-SEC is financed by the National Technology Agency of Finland and several Finnish telecommunications companies. Finally, we would like to thank the anonymous reviewers for their helpful comments and proposals for improvement.

6 References

1 Wollinger, T., Guajardo, J., and Paar, C.: 'Security on FPGAs: state of the art implementations and attacks', *ACM Trans. Embed. Comput. Syst.*, 2004, **3**, pp. 534–574
2 Schneier, B.: 'Applied cryptography' (John Wiley & Sons, 2nd ed. 1996)
3 McLoone, M., and McCanny, J.V.: 'High-performance FPGA implementation of DES using novel method for implementing the key schedule', *IEE Proc. Circ. Dev. Syst.*, 2003, **150** (5), pp. 373–378
4 Rouvroy, G., Standaert, F.-X., Quisquater, J.-J., and Legat, J.-D.: 'Efficient uses of FPGAs for implementations of DES and its experimental linear cryptanalysis', *IEEE Trans. Comput.*, 2003, **52** (4), pp. 473–482
5 National Institute of Standards and Technology.: 'Advanced Encryption Standard (AES)'. Federal Information Processing Standards Publication (FIPS PUB) 197, 26 November 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, accessed June 2005
6 Elbirt, A.J., Yip, W., Chetwynd, B., and Paar, C.: 'An FPGA-based performance evaluation of the AES block cipher candidate algorithm finalists', *IEEE Trans. VLSI Syst.*, 2001, **9** (4), pp. 545–557
7 Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., and Roback, E.: 'Report on the development of the Advanced Encryption Standard (AES)', 2 October 2000, http://csrc.nist.gov/Cryptotoolkit/aes/round2/r2report.pdf, accessed June 2005
8 Dandalis, A., Prasanna, V.K, and Rolim, J.D.P.: 'A comparative study of performance of AES final candidates using FPGAs'. Proc. Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000, Worcester, MA, USA, August 2000, pp. 125–140
9 Elbirt, A.J., Yip, W., Chetwynd, B., and Paar, C.: 'An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists'. Proc. Third Advanced Encryption Conf., AES3, New York, NY, USA, April 2000, pp. 13–27
10 Fischer, V.: 'Realization of the round 2 AES candidates using Altera FPGA.' Proc. 3rd Advanced Encryption Standard Candidate Conf., AES3, New York, NY, USA, April 2000, http://csrc.nist. gov/CryptoToolkit/aes/round2/conf3/papers/24-vfischer.pdf, accessed June 2005
11 Gaj, K., and Chodowiec, P.: 'Comparison of the hardware performance of the AES candidates using reconfigurable hardware'. Proc. 3rd Advanced Encryption Standard Candidate Conf., AES3, New York, NY, USA, April 2000, pp. 40–54, http://csrc.nist.gov/CryptoToolkit/aes/round2/conf3/papers/AES3Proceedings.pdf accessed June 2005
12 Mroczkowski, P.: 'Implementation of the block cipher Rijndael using altera FPGA'. Public Comments on AES Candidate Algorithms—Round 2, May 2000, http://csrc.nist.gov/CryptoToolkit/aes/round2/pubcmnts.htm, accessed June 2005
13 Chodowiec, P., Khuon, P., and Gaj, K.: 'Fast implementation of secret-key block ciphers using mixed inner- and outer-round pipelining'. Proc. 2001 ACM/SIGDA 9th Int. Symp. on Field Programmable Gate Arrays, FPGA 2001, Monterey CA, USA, February 2001, pp. 94–102
14 Gaj, K., and Chodowiec, P.: 'Fast implementation and fair comparison of the final candidates for advanced encryption standard using field programmable gate arrays'. Proc. Topics in Cryptology—CT-RSA 2001, The Cryptographer's Track at RSA Conf. 2001, San Francisco, CA, USA, April 2001 pp. 84–99
15 Hodjat, A., and Verbauwhede, I.: 'A 21.54 Gbits/s fully pipelined AES processor on FPGA'. Proc. 12th Annual IEEE Symp. Field-Programmable Custom Computing Machines, FCCM'04, Napa, CA, USA, April 2004, pp. 308–309
16 Rodríquez-Henríquez, F., Saqib, N.A., and Díaz-Pérez, A.: '4.2 Gbit/s single-chip FPGA implementation of AES algorithm', *Electr. Lett.*, 2003, **39** (15), pp. 1115–1116
17 Saggese, G.P., Mazzeo, A., Mazzocca, N., and Strollo, A.G.M.: 'An FPGA-based performance analysis of the unrolling, tiling, and pipelining of the AES algorithm'. Proc. 13th Int. Conf. Field Programmable Logic and Applications, FPL 2003, Lisbon, Portugal, September 2003, pp. 292–302
18 Standaert, F.-X., Rouvroy, G., Quisquater, J.-J., and Legat, J.-D.: 'A methodology to implement block ciphers in reconfigurable hardware and its application to fast and compact AES RIJNDAEL'. Proc. ACM/SIGDA 11th ACM Int. Symp. Field-Programmable Gate Arrays, FPGA 2003, Monterey, CA, USA, February 2003, pp. 216–224
19 Zambreno, J., Nguyen, D., and Choudhary, A.: 'Exploring area/delay tradeoffs in an AES FPGA implementation'. Proc. 14th Int. Conf. Field-Programmable Logic and its Applications, FPL 2004, Antwerp, Belgium, August–September 2004, pp. 575–585
20 McLoone, M., and McCanny, J.V.: 'High performance single-chip FPGA Rijndael algorithm implementation'. Proc. Workshop on Cryptographic Hardware and Embedded Systems, CHES 2001, Paris, France, May 2001, pp. 65–76
21 McLoone, M., and McCanny, J.V.: 'Single-chip FPGA implementation of the advanced encryption standard algorithm'. Proc. 11th Int. Conf. Field-Programmable Logic and Applications, FPL 2001, Belfast, Northern Ireland, UK, August 2001, pp. 152–161
22 Saqib, N.A., Rodríquez-Henríquez, F., and Díaz-Pérez, A.: 'AES algorithm implementation—an efficient approach for sequential and pipeline architectures'. Proc. 4th Mexican Int. Computer Science, ENC 2003, Tlaxcala, Mexico, September 2003, pp. 126–130
23 Weaver, N., and Wawrzynek, J.: 'High performance, compact AES implementations in Xilinx FPGAs', 27 September 2002, http://www.cs.berkeley.edu/nweaver/sfra/rijndael.pdf, accessed June 2005
24 Fischer, V., and Drutarovský, M.: 'Two methods of Rijndael implementation in reconfigurable hardware'. Proc. Workshop on Cryptographic Hardware and Embedded Systems, CHES 2001, Paris, France, May 2001, pp. 77–92
25 McLoone, M., and McCanny, J.V.: 'Rijndael FPGA implementation utilizing look-up tables'. Proc. 2001 IEEE Workshop on Signal Processing Systems, SIPS'01, Antwerp, Belgium, September 2001, pp. 349–360
26 Xilinx, Inc.: 'Virtex-E 1.8 V extended memory field programmable gate arrays', 17 July 2002, http://www.xilinx.com/bvdocs/publications/ds025.pdf, accessed June 2005
27 Järvinen, K., Tommiska, M., and Skyttä, J.: 'A fully pipelined memoryless 17.8 Gbps AES-128 encryptor'. Proc. ACM/SIGDA 11th ACM Int. Symp. on Field-Programmable Gate Arrays, FPGA 2003, Monterey, CA, USA, February 2003, pp. 207–215
28 Daemen, J., and Rijmen, V.: 'The design of Rijndael' (Springer-Verlag, 2002)
29 Zhang, X., and Parhi, K.K.: 'High-Speed VLSI architectures for the AES algorithm', *IEEE Trans. VLSI Syst.*, 2004, **12** (9), pp. 957–967
30 Pramstaller, N., and Wolkerstorfer, J.: 'A universal and efficient AES co-processor for field programmable logic arrays'. Proc. 14th Int. Conf. Field-Programmable Logic and its Applications, FPL 2004, Antwerp, Belgium, August–September 2004, pp. 565–574
31 Chodowiec, P., and Gaj, K.: 'Very compact FPGA implementation of the AES algorithm'. Proc. Workshop on Cryptographic Hardware and Embedded Systems, CHES 2003, Cologne, Germany, September 2003, pp. 319–333
32 Rouvroy, G., Standaert, F.-X., Quisquater, J.-J., and Legat, J.-D.: 'Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications'. Proc. Int. Conf. Information Technology: Coding and Computing, ITCC'04, Las Vegas, NV, USA, April 2004, Vol. 2, pp. 583–587
33 Standaert, F.-X., Rouvroy, G., Quisquater, J.-J., and Legat, J.-D.: 'Efficient implementation of Rijndael encryption in reconfigurable hardware: improvements and design tradeoffs'. Proc. Workshop on Cryptographic Hardware and Embedded Systems, CHES 2003, Cologne, Germany, September 2003, pp. 334–350
34 Caltagirone, C., and Anantha, K.: 'High throughput, parallelized 128-bit AES encryption in a resource-limited FPGA'. Proc. 15th Annual ACM Symp. Parallel Algorithms and Architectures, SPAA'03, San Diego, CA, USA, June 2003, pp. 240–241
35 Zigiotto, A.C., and d'Amore, R.: 'A low-cost FPGA implementation of the Advanced Encryption Standard algorithm'. Proc. 15th Symp. Integrated Circuits and Systems Design, SBCCI'02, Porto Alegre, Brazil, September 2002, pp. 181–186

36 Xilinx, Inc.: 'Virtex 2.5 V field programmable gate arrays', 2 April 2001, http://www.xilinx.com/bvdocs/publications/ds003.pdf, accessed June, 2005
37 Xilinx, Inc.: 'Virtex-E 1.8 V field programmable gate arrays', 17 July 2002, http://www.xilinx.com/bvdocs/publications/ds022.pdf, accessed June 2005
38 Xilinx, Inc.: 'Virtex-II platform FPGAs: complete data sheet', 1 March 2005, http://www.xilinx.com/bvdocs/publications/ds031.pdf, accessed June 2005
39 Labbé, A., and Pérez, A.: 'AES implementation on FPGA: time—flexibility tradeoff'. Proc. 12th Int. Conf. Field-Programmable Logic and its Applications, FPL 2002, Montpellier, France, September 2002, pp. 836–844
40 Wang, S.-S., and Ni, W.-S.: 'An efficient FPGA implementation of Advanced Encryption Standard algorithm'. Proc. 2004 IEEE Int. Symp. on Circuits and Systems, ISCAS'04, Vancouver, British Columbia, Canada, May 2004, pp. 597–600
41 Xilinx, Inc.: 'Virtex-4 family overview', 17 June 2005, http://www.xilinx.com/bvdocs/publications/ds112.pdf, accessed June 2005
42 Lai, X., and Massey, J.L.: 'A proposal for a new block encryption standard'. Proc. Advances in Cryptology—EUROCRYPT 90 pp. 389–404
43 Lai, X., Massey, J.L., and Murphy, S.: 'Markov ciphers and differential cryptanalysis'. Proc. Advances in Cryptology—EUROCRYPT 91, pp. 17–38
44 Menezes, A.J., van Oorschot, P.C., and Vanstone, S.A.: 'Handbook of applied cryptography' (CRC Press Ltd., 1997)
45 Mencer, O., Morf, M., and Flynn, M.J.: 'Hardware software tri-design of encryption for mobile communication units'. Proc. 1998 IEEE Int. Acoustics, Speech, and Signal Processing, ICASSP '98, Seattle, WA, USA, May 1998, Vol. 5, pp. 3045–3048
46 Mosanya, E., Teuscher, C., Restrepo, H.F., Galley, P., and Sanchez, E.: 'CryptoBooster: a reconfigurable and modular cryptographic coprocessor'. Proc. Workshop on Cryptographic Hardware and Embedded Systems, CHES 1999, Worcester, MA, USA, August 1999, pp. 246–256
47 Leong, M.P., Cheung, O.Y.H., Tsoi, K.H., and Leong, P.H.W.: 'A bit-serial implementation of the international data encryption algorithm IDEA'. Proc. IEEE Symp. Field-Programmable Custom Computing Machines (FCCM'00), Napa Valley, CA, USA, April 2000, pp. 122–131
48 Cheung, O.Y.H., Tsoi, K.H., Wai Leong, P.H., and Leong, M.P.: 'Tradeoffs in parallel and serial implementations of the international data encryption algorithm IDEA'. Proc. Third Int. Workshop on Cryptographic Hardware and Embedded Systems, CHES 2001, Paris, France, May 2001, pp. 333–347
49 Hämäläinen, A., Tommiska, M., and Skyttä, J.: '8 Gigabits per second implementation of the IDEA cryptographic algorithm'. Proc. 12th Int. Conf. Field-Programmable Logic and its Applications, FPL 2002, Montpellier, France, September 2002, pp. 760–769
50 Gonzalez, I., López-Buedo, S., Gómez, F.J., and Martínez, J.: 'Using partial reconfiguration in cryptographic applications: an implementation of the IDEA algorithm'. Proc. 13th International Workshop on Field-Programmable Logic and Applications (FPL'03), Lisbon, Portugal, September 2003, pp. 194–203
51 Pan, Z., Venkateshwaran, S., Gurumani, S.T., and Wells, B.E.: 'Exploiting fine-grain parallelism of IDEA using Xilinx FPGA'. Proc. 16th Int. Conf. Parallel and Distributed Computing Systems (PDCS-2003), Reno, NV, USA, August 2003, pp. 122–131
52 Rivest, R.L.: 'The MD5 message-digest algorithm', RFC 1321 (MIT Laboratory for Computer Science and RSA Data Security, Inc., 1992)
53 National Institute of Standards and Technology.: 'Secure hash standard'. Federal Information Processing Standards Publication (FIPS PUB) 180-2, 1 August 2002, with changes, 25 February 2004, http://www.csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf, accessed June 2005
54 Lien, R., Grembowski, T., and Gaj, K.: 'A 1 Gbit/s partially unrolled architecture of hash functions SHA-1 and SHA-512'. Proc. Topics in Cryptology, CT-RSA 2004, The Cryptographers' Track at the RSA Conf. 2004, San Francisco, CA, USA, February 2004, pp. 324–338
55 National Institute of Standards and Technology.: 'Digital signature standard (DSS), 'Federal Information Processing Standards Publication (FIPS PUB) 186-2', 27 January 2000, http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf, accessed June2005
56 Järvinen, K., Tommiska, M., and Skyttä, J.: 'Hardware implementation analysis of the MD5 hash algorithm'. Proc. 38th Hawai'i Int. Conf. System Sciences HICSS-38, Big Island, HI, USA, January 2005, p. 298 (abstract)
57 Deepakumara, J., Heys, H.M., and Venkatesan, R.: 'FPGA implementation of MD5 hash algorithm'. Proc. Canadian Conf. Electrical and Computer Engineering, CCECE 2001, Toronto, Canada, May 2001, Vol. 2, pp. 919–924
58 Diez, J.M., Bojanić, S., Stanimirovicć, Lj., Carreras, C., and Nieto-Taladriz, O.: 'Hash algorithms for cryptographic protocols: FPGA implementations'. Proc. 10th Telecommunications Forum, TELFOR'2002, Belgrade, Yugoslavia, November 2002,
59 Dominikus, S.: 'A hardware implementation of MD4-family hash algorithms'. Proc. 9th IEEE Int. Conf. Electronics, Circuits and Systems, ICECS 2002, Dubrovnik, Croatia, September 2002, Vol. 3, pp. 1143–1146
60 Grembowski, T., Lien, R., Gaj, K., Nguyen, N., Bellows, P., Flidr, J., Lehman, T., and Schott, B.: 'Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512'. Proc. 5th Int. Conf. Information Security, ISC 2002, Sao Paulo, Brazil, September–October 2002, pp. 75–89
61 Järvinen, K., Tommiska, M., and Skyttä, J.: 'A compact MD5 and SHA-1 co-implementation utilizing algorithm similarities'. Proc. Int. Conf. Engineering of Reconfigurable Systems and Algorithms, ERSA'05, Las Vegas, NV, USA, June 2005, pp. 48–54
62 Kang, Y.K., Kim, D.W., Kwon, T.W., and Choi, J.R.: 'An efficient implementation of hash function processor for IPSEC'. Proc. IEEE Asia-Pacific Conf. on ASIC, AP-ASIC 2002, Taipei, Taiwan, August 2002, pp. 93–96
63 McLoone, M., and McCanny, J.V.: 'Efficient single-chip implementation of SHA-384 and SHA-512'. Proc. 2002 Int. Conf. Field-Programmable Technology, FPT 2002, Hong Kong, China, December 2002, pp. 311–314
64 Ng, C.-W., Ng, T.-S., and Yip, K.-W.: 'A uninified architecture of MD5 and RIPEMD-160 hash algorithms'. Proc. 2004 IEEE Int. Symp. on Circuits and Systems, ISCAS'04, Vancouver, British Columbia, Canada, May 2004, Vol. 2, pp. 889–892
65 Selimis, G., Sklavos, N., and Koufopavlou, O.: 'VLSI implementation of the keyed-hash message authentication code for the wireless application protocol'. Proc. 2003 10th IEEE Int. Conf. Electronics, Circuits and Systems, ICECS 2003, Sharjah, United Arab Emirates, December 2003, Vol. 1, pp. 24–27
66 Sklavos, N., Dimitroulakos, G., and Koufopavlou, O.: 'An ultra high speed architecture for VLSI implementation of hash functions'. Proc. 2003 10th IEEE Int. Conf. Electronics, Circuits and Systems, ICECS 2003, Sharjah, United Arab Emirates, December 2003, Vol. 3, pp. 990–993
67 Ting, K.K., Yuen, S.C.L., Lee, K.H., and Leong, P.H.W.: 'An FPGA based SHA-256 processor'. Proc. 12th Int. Conf. Field-Programmable Logic and its Applications, FPL 2002, Montpellier, France, September 2002, pp. 577–585
68 Wang, M.-Y., Su, C.-P., Huang, C.-T., and Wu, C.-W.: 'An HMAC processor with integrated SHA-1 and MD5 algorithms'. Proc. Asia and South Pacific Design Automation Conf. 2004, Yokohama, Japan, January 2004, pp. 456–458
69 Zibin, D., and Ning, Z.: 'FPGA Implementation of SHA-1 algorithm'. Proc. 2003 5th Int. Conf. ASIC, ASICON 2003, Beijing, China, October 2003, Vol. 2, pp. 1321–1324
70 Wang, X., Yin, Y.L., and Yu, H.: 'Collision search attacks on SHA1', 13 February 2005, http://theory.csail.mit.edu/~yiqun/shanote.pdf, accessed June 2005
71 Wang, X., and Yu, H.: 'How to break MD5 and other hash functions'. Proc. Advances in Cryptology—EUROCRYPT 2005: 24th Annual Int. Conf. the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 2005, pp. 19–35
72 Altera Corporation: 'APEX 20K Programmable logic device family datasheet', http://www.altera.com/literature/ds/apex.pdf, March 2004 accessed June 2005
73 Altera Corporation: 'Stratix II device handbook, volume 2', http://www.altera.com/literature/hb/stx2/stratix2_handbook.pdf, May 2005 accessed June 2005