Hardware performance of the AES finalists - survey and analysis of results

Kris Gaj and Pawel Chodowiec
George Mason University

1. Introduction

The primary criteria used by NIST to evaluate candidates for the new *Advanced Encryption Standard* (AES) include: security, efficiency in hardware and software, and flexibility. Among these four parameters, the efficiency in hardware appeared to be a particularly important factor used to differentiate among competing algorithms because

- the comparison was based on a set of objective and commonly accepted measures;
- there existed large differences among AES candidates;
- there was a good agreement among results reported by several independent groups.

The above conditions were not fulfilled to the same degree by other evaluation criteria.

The initial results regarding the hardware efficiency of the AES candidates were reported during the Third AES Candidate Conference [DPR00a, EYCP00, GaCh00a, IKM00, WeWa00, WBRF00a], and/or submitted as official AES comments [BoCz00, Fis00, Mro00]. Despite the similar approaches taken by these groups, the reported results and rankings of the AES candidates were far from uniform. Additional problem with matching and comparing the results was caused by a different terminology used by various groups.

In this paper, we have made an attempt to summarize and contrast results obtained by various research groups, and to determine and explain the possible sources of differences among these results. To make this comparison more readable, we have suggested the consistent terminology that could be used to describe various architectures and performance parameters of secret-key block ciphers.

The scope of this paper is limited to presenting results that were independently obtained by at least two research teams. No attempt was made to include *all* results reported in the literature. Instead, we have focused on comparing values of parameters we consider the most important for a fair comparison of the AES candidates, and for today's and future implementations.

All opinions and claims presented in this paper are of authors themselves, and are not necessarily shared by members of other teams participating in the AES hardware efficiency analysis.

We hope that our paper will supplement the NIST Final Round 2 Report by providing the more detailed analysis of differences among results of various research teams. We also hope that this paper will stimulate further discussion among the members of all teams involved in the analysis, and within the entire cryptographic community, and will contribute to the development of new architectures, and new fast implementations of the AES standard.

2. Choice of technology

Cryptographic transformations can be implemented in both software and hardware. Software implementations are designed and coded in programming languages, such as C, C++, Java, and assembly language, to be executed, among the other, on general-purpose microprocessors, digital signal processors, and smart cards. Hardware implementations are designed and coded in hardware description languages, such as VHDL and Verilog HDL, and are intended to be realized using two major implementation approaches: *Application Specific Integrated Circuits* (ASICs) and *Field Programmable Gate Arrays* (FPGA).

Application Specific Integrated Circuits (ASICs) are designed all the way from the behavioral description to the physical layout, and then sent for an expensive and time-consuming fabrication. *Field Programmable Gate Array* (FPGA) can be bought off the shelf and reconfigured by designers themselves. With each reconfiguration, which takes only a fraction of a second, an integrated circuit can perform a completely different function. FPGA consists of thousands of universal building blocks, known as *Configurable Logic Blocks* (*CLBs*), connected using programmable interconnects, as shown in Fig. 1a. Additionally, Virtex FPGAs contain dedicated memory blocks called *Embedded Array Blocks* (EABs). Reconfiguration can change a function of each CLB and connections among them, leading to a functionally new digital circuit.

A simplified internal structure of a CLB in the Xilinx XC4000 FPGA family, and a *CLB slice* (1/2 of a CLB) in the Xilinx Virtex FPGA family is shown in Fig. 1bc. In the logic mode (Fig. 1b), each of these elementary units contains a small block of combinational logic, implemented using programmable look-up tables, and two one-bit registers. In the memory mode (Fig. 1c), combinational logic is replaced by two small memories.

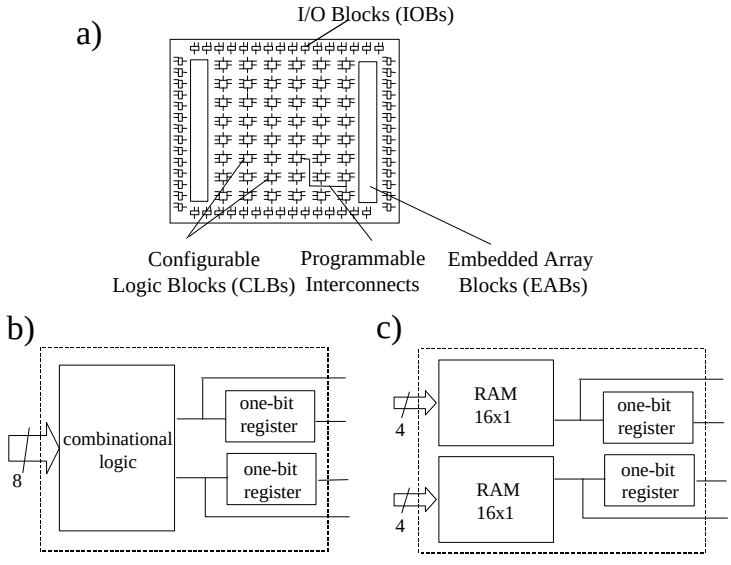

Fig. 1 FPGA device. a) General structure and main components. b) Internal structure of a CLB configured in the logic mode. c) Internal structure of a CLB configured in the memory mode.

In Table I, we collect and contrast features of implementations of cryptographic transformations based on ASICs, FPGAs, and software. The *performance characteristics* of ASICs and FPGAs is almost identical, as demonstrated by the first group of features, and substantially different from the performance characteristics of general-purpose microprocessors. Both ASICs and FPGAs can make a full use of parallel processing and pipelining, and operate on arbitrary size words. In general-purpose microprocessors, parallel processing and pipelining are limited by the number and internal structure of the processor functional units and by the instruction level parallelism. Additionally, all functional units operate on the fixed-size arguments only. The primary difference between ASICs and FPGAs in terms of the performance characteristics is a smaller speed of FPGAs caused by the delays introduced by the circuitry required for reconfiguration. As a result of this speed penalty, any digital circuit implemented in an FPGA is typically slower than the same circuit implemented in an ASIC, assuming that both integrated circuits are fabricated using the same semiconductor technology (in particular, using the same transistor size).

	ASICs (semi-custom and full-custom)	**FPGAs**	**Software (general-purpose microprocessors)**
Performance characteristics			
Parallel processing of data	yes	yes	limited
Pipelining	yes	yes	limited
Word size	variable	variable	fixed
Speed	very fast	fast	moderately fast
Functionality			
Algorithm agility	no	yes	yes
Tamper resistance	strong	limited	weak
Access control to keys	strong	moderate	weak
Development process			
Description languages	VHDL, Verilog HDL	VHDL, Verilog HDL	C, C++, Java, assembly language
Design cycle	long	moderately long	short
Design tools	very expensive	moderately expensive	inexpensive
Testing	expensive	moderately expensive	inexpensive
Maintenance and upgrades	expensive	inexpensive	inexpensive

Table I Characteristic features of implementations of cryptographic transformations in ASICs, FPGAs, and software.

Based on the above characteristics, it should be expected that the relative ranking of the AES algorithms in terms of the hardware efficiency should be the same or similar for ASICs and FPGAs, and can be substantially different for general-purpose microprocessors and other software environments.

The common features of FPGAs and software concern mostly functionality, and do not affect performance. Both general-purpose microprocessors and FPGAs can be easily reconfigured in real time to perform a different algorithm. The disadvantage of this feature is a limited tamper resistance; the contents of an FPGA can be, at least in in theory, modified by an unauthorized user. In practice, the contents of an FPGA is typically downloaded during the initialization from the read-only memory, such as EPROM, which cannot be easily tampared with, at least remotely. The access control to cryptographic keys in FPGAs is also stronger than in software, but weaker than in ASICs.

The development process in both hardware implementation approaches is very similar. In both FPGAs and ASICs, the circuit is described using a hardware description language, verified using a digital circuit simulator, and then tested using a prototyping board. The primary difference between FPGAs and ASICs is that FPGAs do not require the physical design (layout), fabrication, and testing for physical defects. As a result, the design cycle is significantly shorter, and the design tools and testing much less expensive. The interesting similarity between FPGAs and software is a possibility of remote maintenance and upgrading, based on electronic patches.

3. Parameters of hardware implementations

Hardware implementations of secret-key ciphers can be characterized using several parameters. Below we provide our definitions of major parameters and point out the differences in naming conventions used by different authors. We also derive formulas that demonstrate the mutual dependence among these parameters.

3.1 Speed

Encryption (decryption) throughput is defined as the number of bits encrypted (decrypted) in a unit of time. Typically, the encryption and decryption throughputs are equal, and therefore only one parameter is reported. The throughput unit is Mbit/s (megabit per second).

In [WeWa99], a similar parameter is introduced, and called *bandwidth*. Bandwidth is defined as the number of cipher blocks encrypted or decrypted in a unit of time. The unit of bandwidth is 1/s. The bandwidth is related to the throughput by

$$throughput = block\_size \cdot bandwitdh. \quad (1)$$

Encryption (decryption) latency is defined as the time necessary to encrypt (decrypt) a single block of plaintext (ciphertext). The unit of latency is ns (nanosecond). As shown in Table II, there is no common name for this parameter used consistently in the literature. The encryption (decryption) latency and throughput are related by:

$$throughput = block\_size \cdot number\_of\_blocks\_processed\_simultaneously \,/\, latency \quad (2)$$

.

This paper	**NSA** [WBRF00a, WBRF00b]	**WPI** [EYCP00]	**USC** [DPR00a, DPR00b]	**Berkeley** [WeWa00]	**GMU** [GaCh00a]
Encryption (decryption) throughput	Throughput	Throughput	Throughput	Bandwidth (see equation (1))	Speed, Throughput
Encryption (decryption) latency	Time to encrypt (decrypt) one block	-	-	Latency	-
Area	Area [μm^2]	CLB slice count	Area [CLB slices]	Area [4-LUTs & block RAMs]	Area [CLB slices]

Table II Names of the encryption/decryption parameters used in publications of various research groups.

In applications where the large amounts of data are encrypted or decrypted, throughput determines the total encryption/decryption time, and thus is the best measure of the cipher speed. In applications where a small number of plaintext (ciphertext) blocks is processed, the total encryption/decryption time depends on both throughput and latency.

3.2 Area

The area required for the cipher implementation is an important parameter for the following reasons:

a. *Cost*

   The area of an integrated circuit is a primary factor determining its *cost*. It is traditionally assumed that the cost of an integrated circuit is directly proportional to the circuit area. This dependence is not always accurate, especially, taking into account the cost of a package, which is determined by the number of the circuit inputs and outputs.

b. Limit on the maximum area

   In certain hardware environments, there exists a *limit on the maximum area* of the cryptographic unit. This limit may be imposed by the cost, available fabrication technology, power consumption or any combination of these factors. For example, in smart cards and microcontrollers, both cost and power consumption limit the area of the embedded encryption units; in FPGAs, the area is limited by the available fabrication technology and the cost of a programmable device.

In *ASIC implementations*, the area required by the cryptographic unit is typically expressed in μm^2. Two related measures are the transistor count and the logic gate count. Values of all three measures are closely correlated, but not necessarily strictly proportional to each other. All three measures are reported by the tools used for the automated logic synthesis of ASICs. In the semi-custom design methodology, these values are a function of the standard cell library used during logic synthesis.

In *FPGA implementations*, the only circuit size measures reported by the CAD tools are the number of basic configurable logic blocks and the number of equivalent logic gates. It is commonly believed that out of these two measures, the number of basic

configurable logic blocks approximates the circuit area more accurately. Measuring and comparing circuit area in FPGAs is additionally complicated by the existence of dedicated memory blocks. The specification of the FPGA devices does not provide any information about the ratio of the areas used by dedicated memory blocks and basic configurable logic blocks.

4. Modes of operation

Symmetric-key block ciphers are used in several operating modes. From the point of view of hardware implementations, these modes can be divided into two major categories:

a. *Non-feedback modes*, such as Electronic Code Book mode (ECB) and counter mode.
b. *Feedback modes*, such as Cipher Block Chaining mode (CBC), Cipher Feedback Mode (CFB), and Output Feedback Mode (OFB).

In the non-feedback modes, encryption of each subsequent block of data can be performed independently from processing other blocks. In particular, all blocks can be encrypted in parallel. In the feedback modes, it is not possible to start encrypting the next block of data until encryption of the previous block is completed. As a result, all blocks must be encrypted sequentially, with no capability for parallel processing.

The limitation imposed by the feedback modes does not concern decryption, which can be performed on several blocks of ciphertext in parallel for both feedback and non-feedback operating modes.

According to current security standards, the encryption of data is performed primarily using feedback modes, such as CBC and CFB. Non-feedback modes, such as ECB, are used primarily to encrypt session keys during key distribution. As a result, using current standards does not permit to fully utilize the performance advantage of the hardware implementations of secret key cryptosystems, based on parallel processing of multiple blocks of data.

The situation could be remedied by including in the AES standard interleaved modes of operation. In these modes, *N* streams of plaintext blocks, each composed of blocks separated by *N* positions, are encrypted independently, using classical feedback modes

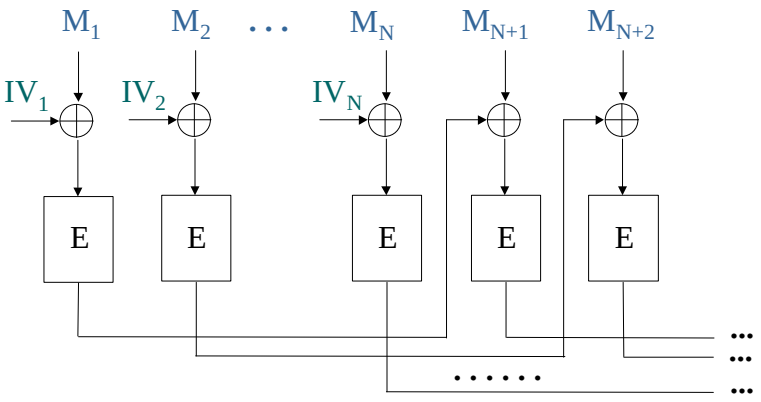

Fig. 2 The interleaved CBC mode.

with N different initialization vectors. The encryption of the next block of data can start as soon as the block N positions earlier has been encrypted. An example of such mode is the interleaved CBC mode shown in Fig. 2 and defined as follows:

$$C_i = \mathrm{AES}(M_i \oplus IV_i) \text{ for } i=1 \text{ to } N, \text{ and } C_i = \mathrm{AES}(M_i \oplus C_{i-N}) \text{ for } i>N\,. \qquad (3)$$

The standard should support arbitrary values of the interleaving factor N, smaller than a certain maximum. The interleaved modes of operation guarantee the speed of non-feedback cipher modes combined with the security of feedback cipher modes.

5. Basic architecture

The top-level block diagram of the symmetric-key block cipher implementation is shown in Fig. 3. Two major functional blocks are: encryption/decryption unit and key scheduling unit.

The interpretation of the function to be performed by the encryption/decryption unit is not uniform among various design groups involved in the comparison of the AES candidates. In the most complete and universal design, this unit is capable of performing both encryption and decryption. Only one operational mode (either encryption or decryption) can be active at a time, and an attempt is made to share as many resources as possible between the encryption and decryption logic. In a simplified design, only one mode (typically encryption) is implemented. The argument behind such implementation is that either two separate chips can be used for each major mode, or the same FPGA chip can be entirely or partially reconfigured to switch between encryption and decryption. Taking into account times required for the complete reconfiguration of a circuit in the current generation of FPGA devices, the latter use is not practical for a large number of applications that require an almost instantaneous switching between encryption and decryption.

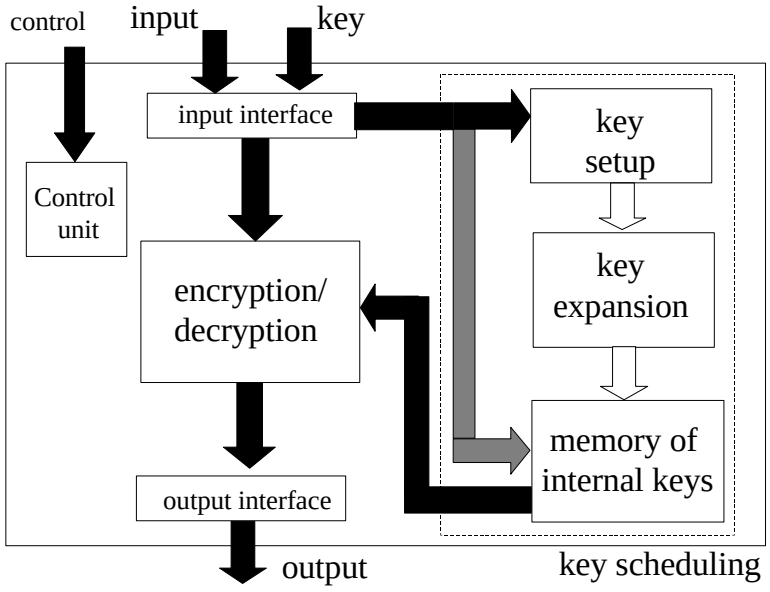

Fig. 3 Top level block diagram of the AES cipher implementation.

The key scheduling unit is in general case composed of three parts: key setup unit, key expansion unit, and memory of internal (round) keys. The key setup unit is

at least three different key sizes: 128, 192, and 256 bits. The output from the key setup unit is than iteratively processed by the key expansion unit in such a way that in every

internal dual-port memory. In each iteration of the encryption /decryption unit, only a small subset of the round keys (typically one or two) are used. After all internal keys are

inactive, till a new key is loaded to the system. The key setup and key expansion operations can be applied to this new key at the same time when the last block of data is

The first simplification of the key scheduling unit, used by majority of research groups, was to limit the key size to only one value, 128 bits. This simplification can

expansion units. The second, more radical simplification was to eliminate the key setup and key expansion units completely, and limit the key scheduling unit to the memory of

memory using input interface.

6. Comparison of the AES candidates in feedback cipher modes

Choice of an architecture

Basic iterative architecture

The basic hardware architecture used to implement an encryption unit of a typical secret-key block cipher is shown in Fig. 4a. One round of the cipher is implemented as a combinational logic, and supplemented with a single register and a multiplexer. In the first clock cycle, input block of data is fed to the circuit through the multiplexer, and stored in the register. In each subsequent clock cycle, one round of the cipher is evaluated, the result is fed back to the circuit through the multiplexer, and stored in the register. The two characteristic features of this architecture are:

- Only one block of data is encrypted at a time.
- The number of clock cycles necessary to encrypt a single block of data is equal to the number of cipher rounds, *#rounds*.

The throughput and latency of the basic iterative architecture, $throughput_{bi}$ and $latency_{bi}$, are given by

$$throughput_{bi} = block\_size / \#rounds \cdot clock\_period. \quad (4)$$

$$latency_{bi} = \#rounds \cdot clock\_period. \quad (5)$$

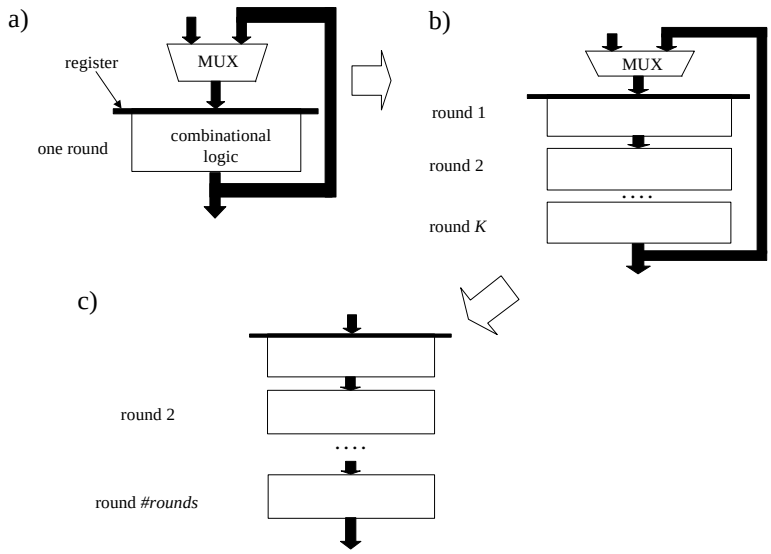

Fig. 4 Alternative architectures of the encryption/decryption unit of a block cipher suitable for feedback cipher modes: a) basic iterative architecture, b) partial loop unrolling, c) full loop unrolling.

This paper	**NSA** [WBRF00a, WBRF00b]	**WPI** [EYCP00]	**USC** [DPR00a, DPR00b]	**Berkeley** [WeWa00]	**GMU** [GaCh00a]
Basic iterative architecture	Iterative architecture	Iterative looping	Single-round based	Single iterative round	Basic architecture
Partial loop unrolling	-	Loop unrolling, LU-*K*	-	-	*K*-round loop unrolling
Full loop unrolling	-	Loop unrolling, LU-*N*	-	-	-

Table III. Names used in publications of various research groups for alternative architectures suitable for feedback cipher modes.

6.1.2 Loop unrolling

Architecture with *partial loop unrolling* is shown in Fig. 4b. The only difference compared to the basic iterative architecture is that the combinational part of the circuit implements *K* rounds of the cipher, instead of a single round. *K* must be a divisor of the total number of rounds, *#rounds*.

The number of clock cycles necessary to encrypt a single block of data decreases by a factor of *K*. At the same time the minimum clock period increases by a factor slightly smaller than *K*, leading to an overall relatively small increase in the encryption throughput and decrease in the encryption latency, given by

$$throughput_{plu}/throughput_{bi} = latency_{bi}/latency_{plu} = (1 + \tau)/(1+\tau/K), \quad (6)$$

where τ is the ratio of the sum of the multiplexer delay, the register delay and the register setup time to the delay of a single cipher round. This increase in speed is obtained at the cost of the circuit area. Because the combinational part of the circuit constitutes the majority of the circuit area, the total area of the encryption/decryption unit increases almost proportionally to the number of unrolled rounds, *K*. Additionally, the number of internal keys used in a single clock cycle increases by a factor of *K*, which in FPGA implementations typically implies the almost proportional growth in the number of CLBs used to store internal keys.

Architecture with full loop unrolling is shown in Fig. 4c. The input multiplexer and the feedback loop are no longer necessary, leading to a small increase in the cipher speed and decrease in the circuit area compared to the partial loop unrolling with the same number of rounds unrolled.

In summary, loop unrolling enables increasing the circuit speed in both feedback and non-feedback operating modes. Nevertheless this increase is relatively small, and incurs a large area penalty. As a result, choosing this architecture can be justified only for feedback cipher modes, where none other architecture offers speed greater than the basic iterative architecture, and only for implementations where large increase in the circuit area can be tolerated.

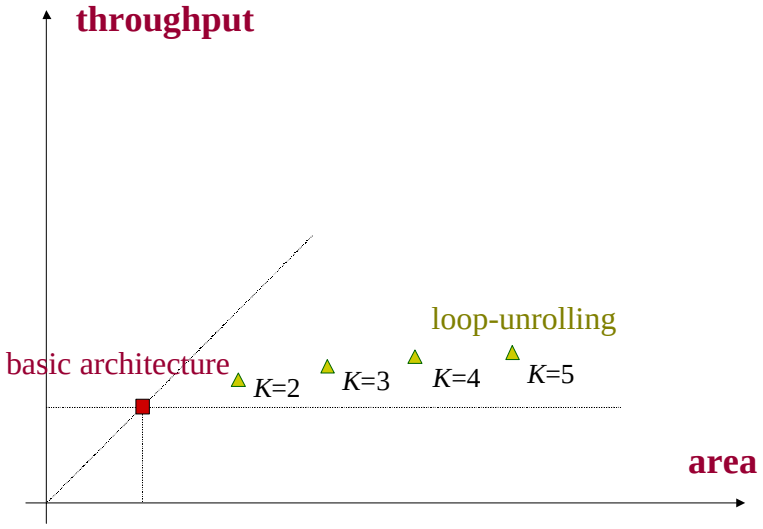

Fig. 5 Throughput vs. area dependence of the basic iterative architecture and architectures with partial loop unrolling.

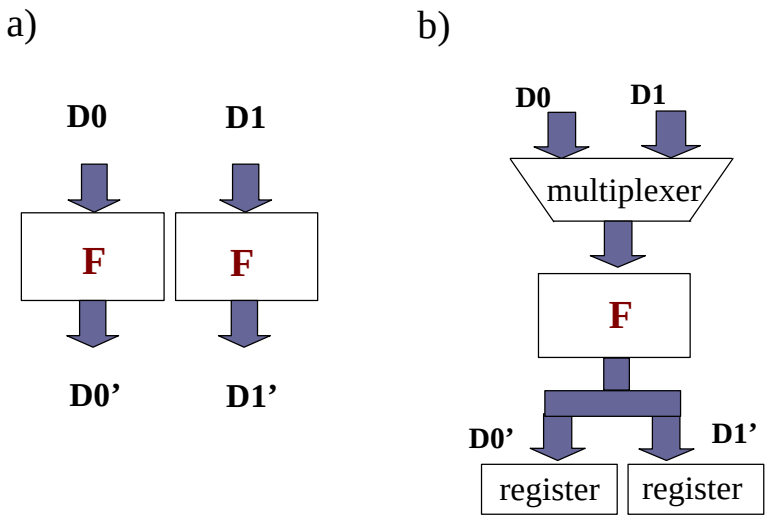

Fig. 6 An example of the architecture with resource sharing. a) basic architecture without resource sharing, with two functional units F running in parallel, b) modified architecture with resource sharing, with one functional unit F used to process two streams of data in two subsequent clock cycles.

6.1.3 Resource sharing

For some ciphers, it is possible to further decrease the circuit area by time sharing of certain resources (e.g., function *h* in Twofish, 4x4 S-boxes in Serpent, 8x32 S-boxes S0, S1 in the mixing transformation of Mars, multiplication units in RC6). This is accomplished by using the same functional unit to process two (or more) parts of the data block in different clock cycles, as shown in Fig. 6b. In Fig. 6a, two parts of the data block, D0 and D1, are processed in parallel, using two independent functional units F. In Fig. 6b, a single unit F is used to process two parts of the data block sequentially, during two subsequent clock cycles.

The use of resource sharing in real life implementations is expected to be limited, because

- Gain in the circuit area is always smaller than the loss in the circuit speed.
- The amount of area used by a basic iterative implementation of a symmetric cipher is typically already quite small.

6.1.4 Deviations from the basic iterative architecture

Three final AES candidates, Twofish, RC6, and Rijndael, can be implemented using exactly the basic iterative architecture shown in Fig. 4a. This is possible because all rounds of these ciphers perform exactly the same operation. For the remaining two ciphers, Serpent and Mars, this condition is not fulfilled, and as a result, the basic iterative architecture can be defined in several different ways.

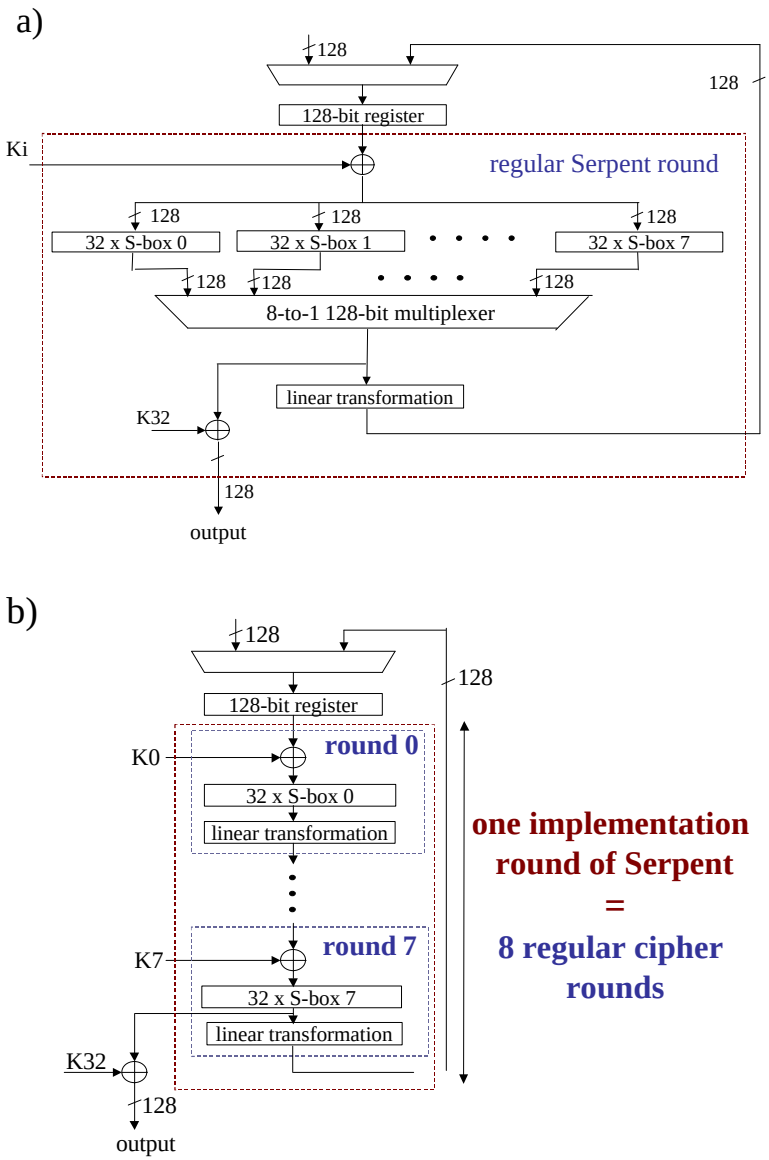

Fig. 7 Two alternative basic iterative architectures of Serpent. a) Serpent I1, with only one regular cipher round executed in every clock cycle and eight sets of S-boxes multiplexed depending on the number of the round, b) Serpent I8, with eight regular cipher rounds treated as a single implementation round, and no need for multiplexing.

Serpent consists of 8 different kinds of rounds. Each round consists of three elementary operations. Two of these operations, key mixing and linear transformation are identical for all rounds; the third operation, S-Boxes, is different for each of the eight consecutive rounds.

Two possible ways of defining the basic iterative architecture of Serpent are shown in Fig. 7. In the first architecture, shown in Fig. 7a, the combinational part of the circuit performs a single regular cipher round. To enable switching between 8 different types of rounds, the combinational part includes 8 sets of S-boxes, each fed by the output from

key mixing. Based on the current round number, the output of only one of the eight S-boxes is chosen using the multiplexer to feed the input of the linear transformation. In this architecture, Serpent is treated literally as a cipher with 32 rounds.

In the second architecture, shown in Fig. 7b, eight regular cipher rounds are treated as a single *implementation round*, and implemented one after the other using a combinational logic. The implementation round needs to be computed only four times, to implement all 32 regular cipher rounds. Thus, in this architecture, Serpent is treated as a cipher with four extended cipher rounds.

Both conventions have their advantages and disadvantages.

The first architecture takes less area (especially taking into account the area required for key scheduling and/or key storage). The second architecture is significantly faster.

If the first architecture, shown in Fig. 7a, is treated as the basic iterative architecture, then the second architecture, shown in Fig. 7b, can be treated as an architecture with the partial loop unrolling (8 rounds unrolled). The disadvantage of this approach is that the dependence between the speed and area of the basic iterative architecture and the architecture with loop unrolling is significantly different for Serpent than for any other cipher.

On the other hand, if the second architecture, shown in Fig. 7b, is treated as a basic iterative architecture, then the first architecture can be treated as an architecture with resource sharing, where linear transformation and key mixing are shared among eight consecutive rounds. The primary advantage of this approach is that the dependence between the basic iterative architecture, architecture with resource sharing, and architecture with loop unrolling remains the same for Serpent and the other AES candidates (see Fig. 5). In particular, the second architecture guarantees the maximum speed/area ratio typical for the basic iterative architecture in other ciphers.

In Mars, there exist four different kinds of rounds, each repeated 8 times: forward mixing, forward keyed transformation, backwards keyed transformation, and backwards mixing. It is possible to implement forward and backwards mixing using the same functional unit; the same holds for the forward and backwards keyed transformation. The structure of the mixing transformation and the keyed transformation are significantly different, and as a result they must be implemented using separate units.

The simplest architecture used to implement Mars is shown in Fig. 8a. The keyed transformations and the mixing transformations are performed using two separate combinational units. The output of one these transformations is chosen using a multiplexer, depending on the number of the round, and fed back to a single register.

An alternative architecture, suggested in [DPR00a] and [DPR00b], is shown in Fig. 8b. In this architecture, each combinational block has its own register. In feedback cipher modes, this architecture offers almost identical throughput and latency as the first, simplest architecture. Only in non-feedback cipher modes, the double increase in the encryption throughput is possible by parallel processing two blocks of data, one in the mixing transformation unit, and the second in the keyed transformation unit.

The disadvantage of both architectures is that the delay of one of the combinational units determines the time used to perform the second operation. This feature may be disadvantageous for Mars, in which the keyed transformation unit has typically a longer delay than the mixing transformation unit. As a result, the encryption latency is equal to 32 delays required to perform the keyed transformation.

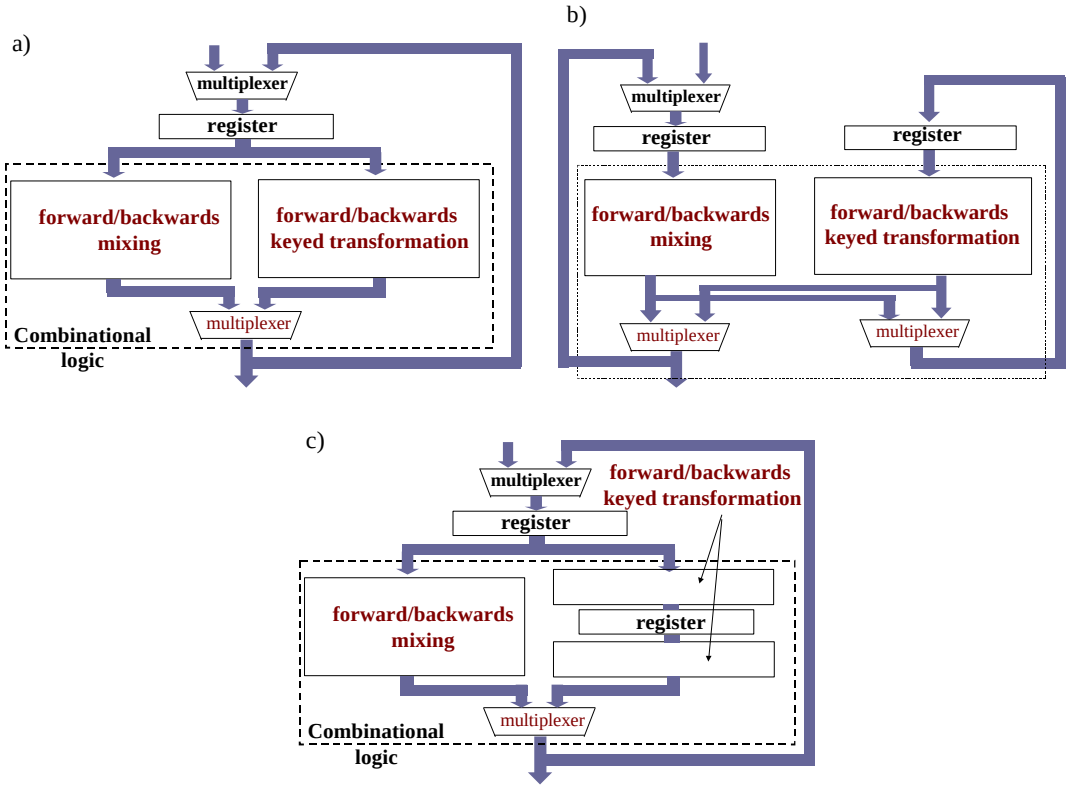

Fig. 8 Three alternative basic iterative architectures of Mars.

The latency and throughput of the circuit can be improved in the third architecture, suggested in [BCD+98] and shown in Fig. 8c. In this architecture, two clock cycles are required to perform the keyed transformation, and only one cycle to perform the mixing transformation. If the delay of the mixing transformation is not greater than one half of the delay of the keyed transformation, the total encryption latency is reduced to 24 delays required to perform the keyed transformation, i.e., by 25% compared to both previous architectures.

Only the first two architectures were used in the reported FPGA and ASIC implementations described in this paper.

6.2 Results and discussion

6.2.1 Results for the basic iterative architecture

Basic iterative architecture was investigated by three groups using Virtex FPGA devices, two groups using Altera FPGA devices, and one group using semi-custom ASICs based on the MOSIS library of standard cells, as summarized in Table IV. Only groups that developed actual implementations for at least three AES candidates are included in this survey.

Group	Publications and presentations	Technology	Family of FPGA devices/ ASIC library	Device numbers	Implemented AES candidates
GMU - George Mason University	[GaCh00a, GaCh00b]	FPGA, CMOS, 0.22 μm	Xilinx Virtex	XCV 1000	5
			Xilinx 4000	XC 4085 XL	3 (Serpent, RC6, Twofish)
	[ChGa00]	FPGA, CMOS, 0.35 μm / 0.22 μm	Altera FLEX	FLEX 10K250A	5
				FLEX 10K130E	4 (all except Serpent)
USC - University of Southern California	[DPR00a, DPR00b]	FPGA, CMOS, 0.22 μm	Xilinx Virtex	XCV 1000	5
WPI - Worcester Polytechnic Institute	[EYCP00]	FPGA, CMOS, 0.22 μm	Xilinx Virtex	XCV 1000	4 (all except Mars)
MICRONIC	[Fis00]	FPGA, CMOS, 0.22 μm	Altera FLEX	FLEX 10K130E	3 (Rijndael, Serpent, Twofish)
NSA - National Security Agency	[WBRF00a, WBRF00b]	ASIC, CMOS, 0.5 μm	MOSIS		5

Table IV Research groups comparing hardware performance of the AES candidates using basic iterative architecture.

6.2.1.1 Speed

Speed, and specifically the encryption throughput, was chosen by all five groups as a primary optimization criterion. Therefore, it should be expected that the agreement among the results of various groups would be the best for this implementation parameter.

We will first collect together corresponding results obtained by various research groups, and discuss them separately for each particular family of FPGA and ASIC devices. We will then make a cross-technology comparison by combining together the best results obtained for each technology.

The best sample of results exists for the Xilinx Virtex family of FPGA devices. These devices were used for implementing AES finalists by three research groups: GMU, USC, and WPI. Three ciphers, RC6, Rijndael, and Twofish, were implemented by all three research groups. Mars, and each of the two possible basic iterative architectures for

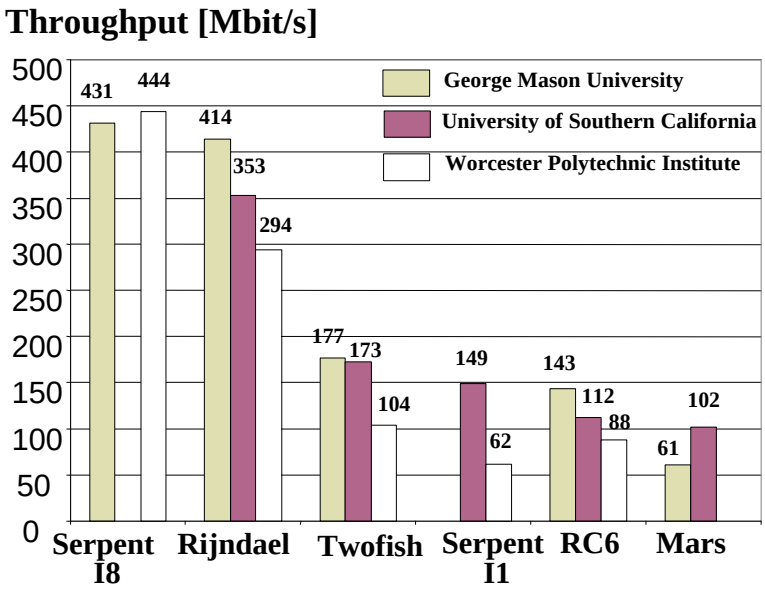

Fig. 9 Comparison of throughputs of all AES candidates implemented by three independent university groups using Xilinx Virtex XCV-1000 FPGA devices.

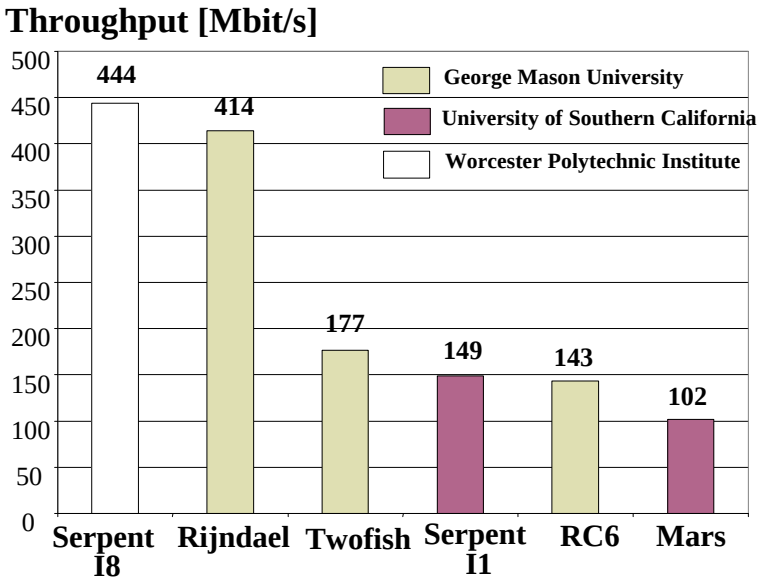

Fig. 10 Throughputs of the fastest reported implementations of the AES candidates using Xilinx Virtex XCV-1000 FPGA devices.

Serpent (denoted as Serpent I1 and Serpent I8) were implemented by two research groups. The results are put together in Fig. 9.

The design choices made by all three research groups were very similar in the following areas:

1. *FPGA device* - all groups chose to use Xilinx Virtex XCV-1000.

2. *Synthesis tool* - all groups used Xilinx Foundation Series 2.1.
3. *Optimization criteria* - all groups set the optimization mode of the synthesis tool to optimization for maximum speed (vs. optimization for minimum area).

The following differences among implementations developed by various groups exist or cannot be excluded based on the available reports [DPR00b, EYCP00, GaCh00a]:

1. *resource sharing between encryption and decryption*.
   The GMU group assumed that the circuit resources are shared between encryption and decryption units, as long as this sharing does not significantly affect the encryption and decryption throughput. The WPI and USC groups made an assumption that the encryption and decryption circuits are completely independent, and are never present in the FPGA device at the same time. In the GMU designs, switching between encryption and decryption is instantaneous, and is controlled by a single logic signal. In the WPI and USC designs, switching between encryption and decryption requires the entire FPGA device to be reprogrammed, which takes about 10-15 ms. At least in theory, the WPI and USC assumptions should lead to slightly higher speeds compared to the GMU designs, with the largest effect in case of RC6, and no effect in case of Serpent [GaCh00a].
2. *options of the Xilinx synthesis and implementation tools*
   The exact choice of options of the synthesis and implementation tools is not reported by any group. The different choice of these options might have resulted in small differences in speed and area of the implemented circuits. For example, choosing the option *Preserve hierarchy* vs. *Eliminate hierarchy* in Xilinx implementation tools results typically in a higher speed at the cost of a larger circuit area.
3. *speed grade of the Virtex devices*
   The speed grades of the Virtex devices that were chosen by particular groups are as follows: WPI: -4, GMU and USC: -6. The differences in speed resulting from the different speed grades typically do not exceed 5%, with the grade -6 faster than the grade -4.
4. *design styles*
   Since VHDL codes were developed independently by each group, the individual design styles of authors are likely to have the biggest influence on differences in absolute results.

In Fig. 9, it is shown that the order of the AES algorithms in terms of the encryption and decryption throughput is identical in reports of all research groups. Serpent in architecture I8 (see Fig. 7b) and Rijndael are over twice as fast as Twofish and RC6. Mars is the slowest of all candidates. The second basic architecture of Serpent - architecture I8 (see Fig. 7b) is significantly faster than the first architecture - I1 (Fig. 7a), and should be used in cipher feedback modes whenever the speed is a primary concern, and the area limit is not exceeded. The architecture I1 of Serpent is slower than Rijndael and Twofish, and faster than Mars. The relation to RC6 is different for USC (Serpent faster than RC6) and WPI (Serpent slower than RC6).

Implementations based on Altera FLEX FPGA devices were developed by two groups: MICRONIC and GMU. Additionally, the implementations of a single AES candidate using the same Altera devices were reported by two teams from the Military

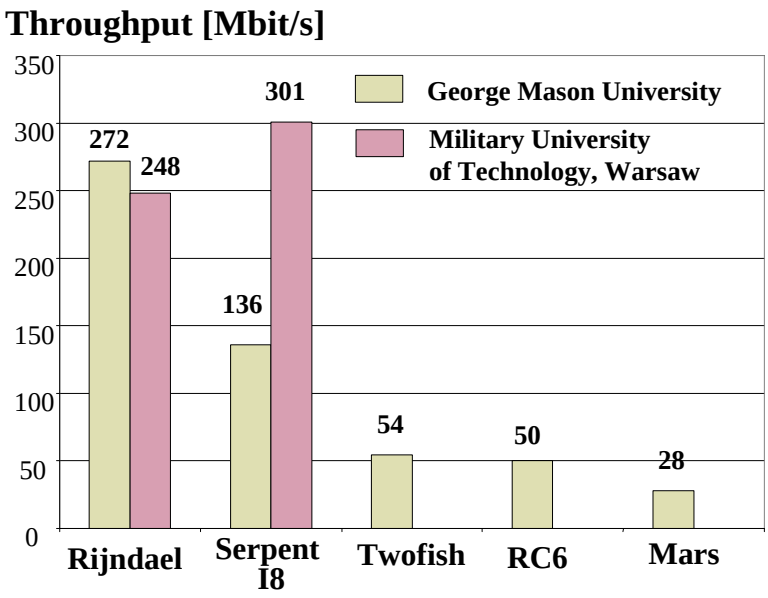

Fig. 11 Comparison of implementations of the AES candidates by two independent groups using Altera FLEX 10K250A FPGA devices.

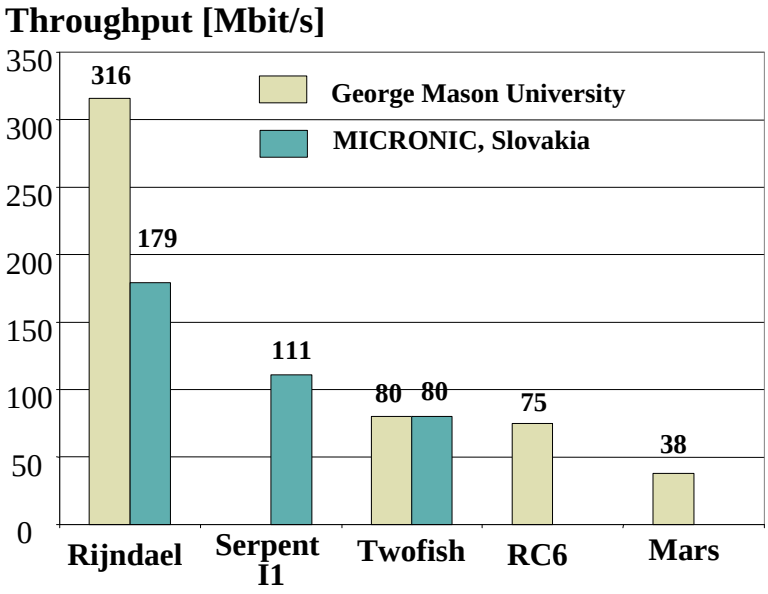

Fig. 12 Comparison of implementations of the AES candidates by two independent groups using Altera FLEX 10K130E FPGA devices.

University of Technology in Warsaw, Poland [BoCz00, Mro00]. The results of implementing AES candidates using 10K130E and FLEX 10K250A devices are summarized in Fig. 11 and Fig. 12.

For FLEX 10K250A (see Fig. 11), in the GMU implementations, Rijndael is over 50% faster than Serpent I8; Twofish and RC6 are both more than two times slower than Serpent, and Mars is almost twice as slow as Twofish and RC6. Compared to the results

for Xilinx Virtex devices, the primary change is the first position of Rijndael and its big speed superiority over Serpent. Additionally, the advantage of Twofish over RC6 is only marginal for Altera devices. An agreement between the results of the GMU group and results of teams from the Warsaw Military University of Technology is good for Rijndael, and poor for Serpent I8.

The large change of the Rijndael/Serpent throughput ratio can be explained by the use of block RAMs in Altera implementations. These block RAMs are ideal for implementing large S-boxes, such as 8x8 S-boxes used in Rijndael. Similar dedicated RAM memories present in the Xilinx Virtex devices were not used in the Xilinx implementations. The only other cipher that can benefit from using dedicated RAMs is Mars, which contains large 9x32 bit S-box. Nevertheless, this S-box is not located in the critical path of Mars, and therefore no improvement in the relative speed of Mars can be observed.

For FLEX 10K130E, only two ciphers have been implemented by both of the two research groups, MICRONIC and GMU, as shown in Fig. 12. The agreement between the results of both groups is very good for Twofish, but rather poor for Rijndael (GMU's result superior by about 75%). In terms of the order of algorithms, Rijndael is the fastest, followed by Serpent I1 (Serpent I8 requires the amount of resources unavailable on the chip), Twofish and RC6 come next with the comparable throughput, and the implementation of Mars does not fit on the integrated circuit. In general, the order of algorithms is the same as for FLEX 10K250A.

The only results, regarding the comparison of the AES candidates using the basic iterative architecture implemented in ASICs, come from the NSA group [WBRF00a, WBRF00b]. The primary difference between the NSA top level architecture, and the top level architectures used by GMU and WPI is the implementation of key scheduling. In the GMU and WPI architectures, key setup and key expansion units are not implemented.

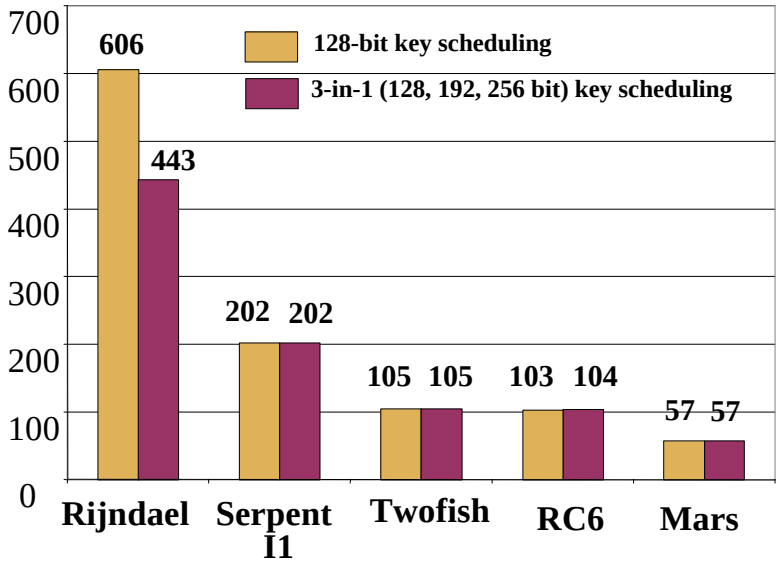

Fig. 13 Results of the NSA implementations of the basic iterative architecture in 0.5 μm CMOS ASICs, for two different designs of the key scheduling unit: a) 128-bit key design, b) universal 3-in-1 design capable of processing keys of the size 128, 192, and 256 bits.

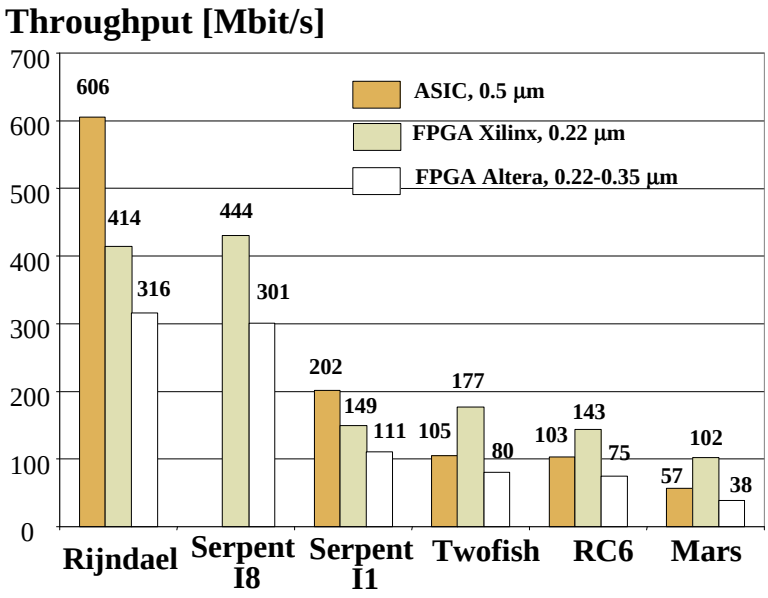

Fig. 14 Comparison of the best results obtained using three different technologies and devices types: a) 0.5 µm CMOS standard-cell ASIC technology, b) 0.22 µm CMOS FPGA technology, Virtex XCV1000 device, c) 0.22 µm and 0.35 µm CMOS FPGA technology, Altera FLEX 10K130E and FLEX 10K250A devices.

Instead, internal keys are generated off-the-chip and downloaded to the internal memory using input interface. In the NSA designs, the key scheduling unit consists of all three parts. Two types of this unit are being implemented: one, called 128-bit, is able to process only 128-bit keys, the second, called 3-in-1, can process all three AES key sizes, 128, 192, and 256-bits. In Fig. 13 the encryption throughput of all AES candidates is shown for the two different ways of implementing the key scheduling unit and the key size of 128 bits. For the majority of ciphers, the encryption/decryption unit, rather than the key scheduling unit, limits the encryption throughput. Only the encryption throughput of Rijndael depends strongly on the type of the key scheduling unit, and decreases for a more universal and complex 3-in-1 key scheduling unit.

In Fig. 14, we compare the best encryption throughputs obtained for semi-custom ASICs, Xilinx Virtex FPGAs, and Altera FLEX FPGAs. In comparing the absolute speeds of the ASIC vs. FPGA implementations, it should be noted that the two primary differences in the fabrication technology tend to cancel each other:

- Assuming the same fabrication process, ASIC implementations are typically several times faster than the corresponding FPGA implementations because of the speed and area penalty caused by the reconfiguration capabilities of FPGAs.
- The ASIC library of standard cells, used by the NSA group, is based on 0.5 µm CMOS fabrication process, which is two generations older than the 0.22 µm process used to fabricate Virtex and Altera FPGA devices. In each generation, the speed of the ASICs improves by a factor of 2-3 [WBRF00a].

The order of the algorithms is almost identical for all three types of implementations. The only exceptions are the dependence between the throughput of Rijndael and Serpent I8, and the position of Serpent I1.

For three ciphers, Twofish, RC6, and Mars, the best FPGA implementations give the higher throughput than the NSA ASIC implementations. For Rijndael and Serpent I1, ASIC implementations are superior. Serpent I8 was not implemented in ASICs.

Rijndael and Serpent seem to benefit most from switching from FPGA to ASIC. In case of Rijndael, the performance advantage of ASICs can be explained by the superior implementation of large S-boxes in the ASIC technology. Large S-boxes are implemented in ASICs using fast multiple-port ROMs; in Xilinx Virtex FPGAs using slower distributed RAM memory (composed of multiple 16x1 memory units), and in Altera FLEX FPGAs using medium-speed dedicated block RAMs.

6.2.1.2 Area

Comparing AES candidates in terms of the circuit area is much more difficult than comparing them in terms of the speed.

The primary difficulties are as follows:

a. *area was not a primary optimization criterion*
All research groups, considered in this survey, made speed the primary optimization criterion. As a result, in at least several cases, the design decisions have been made that substantially increase circuit area, in exchange for even a relatively small increase in the encryption throughput.

b. *different top level units are included in the area count*
The following units may be included in the area count (see Fig. 3): encryption/decryption unit, key setup unit, key expansion unit, memory of internal keys, control unit, and i/o interface. The GMU, WPI, and MICRONIC groups did not implement the full key scheduling unit, and therefore their area count includes only encryption/decryption unit, memory of internal keys, control unit, and i/o interface. The relative percentage of the area used for particular units has not been reported. From our experience, the control unit and i/o interface take typically less than 5% of the circuit area and are identical for all ciphers. Therefore, the major contribution and the source of major differences among the circuit areas come from the encryption/decryption unit and the memory of internal keys. Additionally, in the MICRONIC implementation, internal keys are stored in the block RAMs, and as a result, the memory of internal keys is not included in the total area expressed in the number of basic configurable logic blocks (logic elements).

The USC and NSA designs include key scheduling. The USC group reports separately the amount of area taken by the cryptographic core (encryption/decryption unit, control unit, and i/o interface), and area taken by key scheduling. The NSA group reports the total amount of area [WBRF00b], and percentage of the area used by all individual units [WBRF00c].

Taking into account these major differences, the fairest approach would be to include in the comparison:

- the *total area* reported by the GMU and WPI groups, which includes the encryption/decryption unit, memory of internal keys, control unit, and i/o interface;
- the *total area* reported by MICRONIC, which includes the encryption/decryption unit, control unit, and i/o interface;
- the area of the *cryptographic core* reported by USC, which includes the encryption/decryption unit, control unit, and i/o interface;
- the *percentage of the total area* reported by NSA, attributed to encryption/decryption unit, control unit, and i/o interface.

The consequence of this approach is that the areas reported by GMU and WPI are expected to be larger than areas reported by USC, MICRONIC, and NSA, because they include the area used by the memory of internal keys.

c. different assumptions regarding the key storage

The GMU and WPI groups are the only groups that implement the memory of internal keys using basic configurable logic blocks. Still, the approaches taken by both groups are considerably different. In the WPI designs, the internal keys are stored in the CLB flip-flops. Since each Virtex CLB slice includes two flip-flops, only *2 bits* of a key can be stored in each CLB slice. In the GMU designs, the internal keys are stored in the distributed RAMs included in each CLB. Since, each Virtex CLB slice includes two 16x1 RAMs, up to *32 bits* of a key can be stored in each CLB slice.

As a result, the area used by the memory of internal keys contributes to about 40-70% of the total area in the WPI designs, and only about 15-25% in the GMU designs [EYCP00, GaCh00a]. In our opinion, the area of the WPI designs could be significantly reduced by using CLB RAMs, without any significant influence on the encryption throughput.

d. sharing resources between encryption and decryption

The NSA, GMU, and MICRONIC groups assumed that the hardware resources are shared between encryption and decryption. The WPI and USC designs perform only encryption. The relative amount of additional logic required to perform decryption on top of encryption is different for each AES candidate. It was estimated that adding decryption capabilities to the encryption circuit increases the total amount of area by 3% in case of Mars, 6% in case of Twofish, 20% for RC6, 55% for Rijndael, and 100% for Serpent [GaCh00a]. Therefore, to make the comparison fair, the areas reported by WPI and USC should be multiplied by the appropriate factor, different for each cipher.

e. dedicated block RAMs

Comparing the circuit area used by various ciphers is additionally complicated in the Altera FPGA implementations by the use of dedicated block RAMs.

Since

- implementations of only two out of five AES candidates include block RAMs, and
- it is not known how many basic configurable logic blocks take the same amount of area as a single block RAM,

it is not possible to put all Altera FPGA implementations in order according to the circuit area.

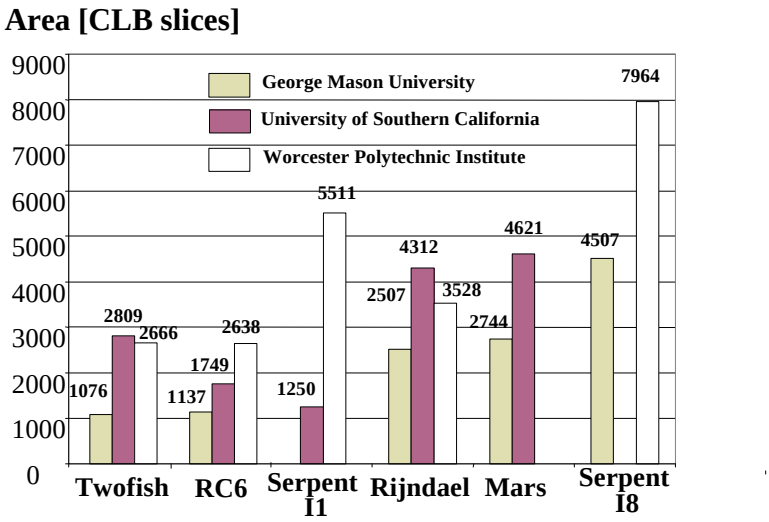

Fig. 15 Comparison of the circuit areas for AES candidates implemented by three independent university groups using Xilinx Virtex FPGA XCV 1000 device.

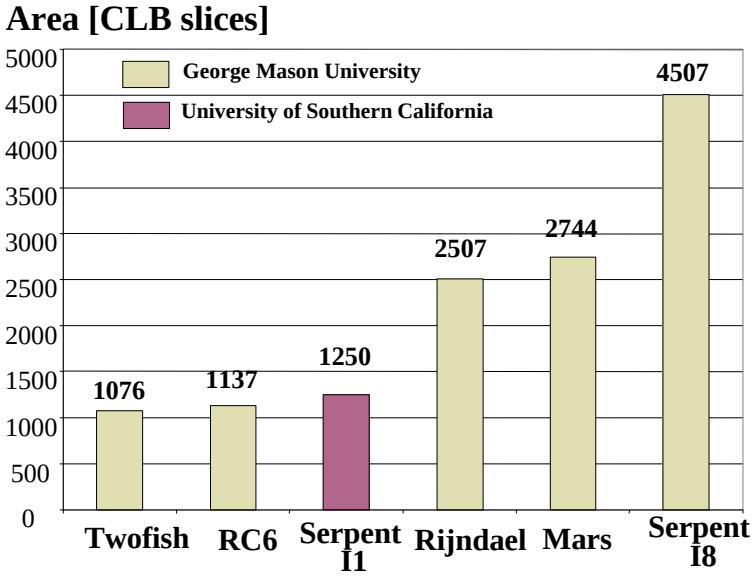

Fig. 16 Minimum areas reported for implementations of the AES candidates using Xilinx Virtex FPGA XCV 1000 devices.

Despite these interpretation difficulties, the analysis of results presented in Fig. 15 and Fig. 16 leads to relatively consistent conclusions. All ciphers can be divided into three major groups, depending on the area taken by all units except for key scheduling:

- Twofish and RC6 require the smallest amount of area;
- Rijndael and Mars require the medium amount of area (at least 50% more than Twofish and RC6)

- Serpent I8 requires the largest amount of area (at least 60% more than Rijndael and Mars).

Serpent I1 belongs to the first group according to USC, and to the second group according to WPI. Based on the discussion presented in points b. and c. above, the USC interpretation seems to be more appropriate.

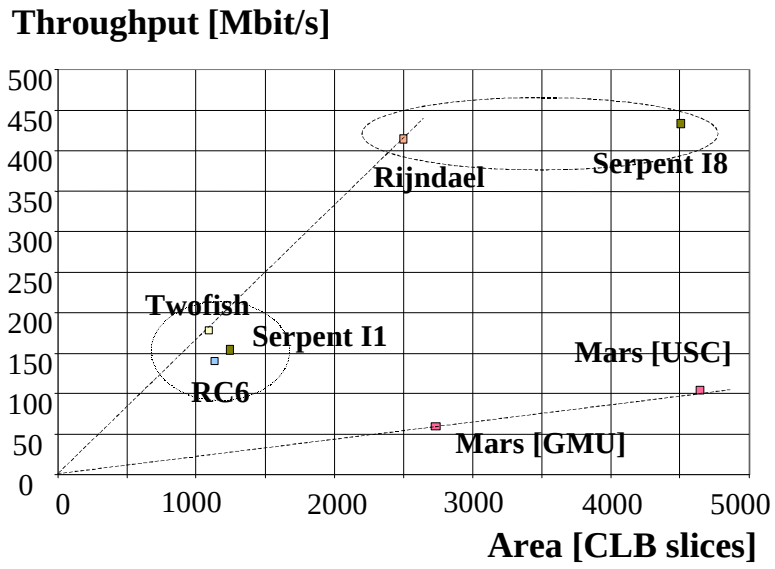

Fig. 17 Throughput vs. area diagram for Xilinx Virtex FPGA implementations. For each cipher, only implementations with the best throughput/area ratio are shown in the diagram.

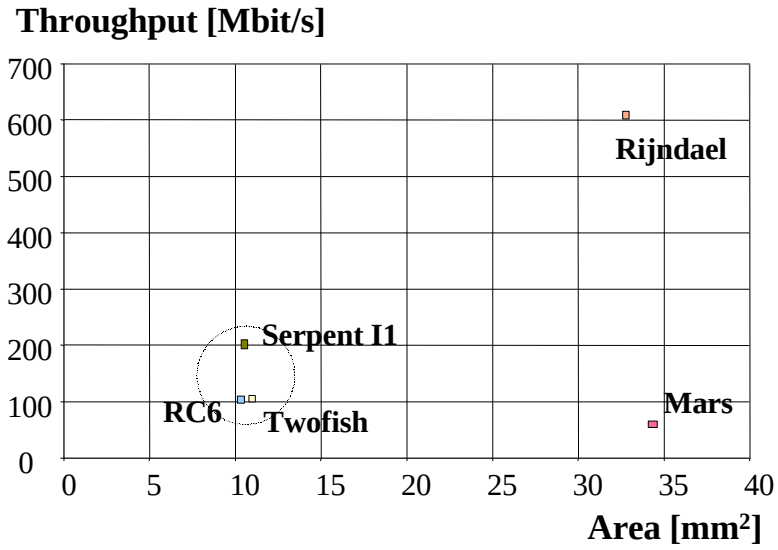

Fig. 18 Throughput vs. area diagram for the ASIC implementations of the basic iterative architecture developed by NSA.

The NSA results for ASICs confirm this classification (see Fig. 18)

- Twofish, RC6, and Serpent I1 require approximately the same amount of area;
- Rijndael and Mars require about twice as much area as ciphers from the first group.

6.2.1.3 Speed vs. area

The overall features of all AES candidates can be the best presented using a two-dimensional diagram showing the relationship between the encryption/decryption throughput and the circuit area. In Fig. 17, we collect the results for the Xilinx Virtex FPGA implementations, and in Fig. 18 for the ASIC implementations.

For FPGA implementations, we include for each cipher the parameters of a single implementation with the best throughput/area ratio. For Rijndael, Twofish, RC6, and Serpent I8 the best implementations were obtained by the GMU group; for Serpent I1 by the USC group, and for Mars, the USC and GMU implementations offer almost identical value of the throughput/area ratio.

Comparing diagrams shown in Fig. 17 and Fig. 18, reveals that the speed/area characteristics of the AES candidates is almost identical for the FPGA and ASIC implementations. The primary difference between the two diagrams comes from the absence of the ASIC implementation of Serpent I8 in the NSA report [WBRF00b].
All ciphers can be divided into three distinct groups:

- Rijndael and Serpent I8 offer the highest speed at the expense of the relatively large area;
- Twofish, RC6, and Serpent I1 offer medium speed combined with a very small area;
- Mars is the slowest of all AES candidates and last or second to last in terms of the circuit area.

Looking at this diagram, one may ask which of the two parameters: speed or area should be weighted more during the comparison? The definitive answer is *speed*. The primary reason for this choice is that in the feedback cipher modes it is not possible to substantially increase encryption throughput even at the cost of a very substantial increase in the circuit area. On the other hand, by using resource sharing described in section 6.1.3, the designer can substantially decrease circuit area at the cost of a proportional (or higher) decrease in the encryption throughput. Therefore, Rijndael and Serpent can be implemented using almost the same amount of area as Twofish and RC6; but Twofish and RC6 can never reach the speeds of the fastest implementations of Rijndael and Serpent I8.

6.2.2 Loop unrolling - results

The only architecture capable of improving cipher speed in the feedback cipher modes, over the speed obtained using the basic iterative architecture is loop unrolling. The only two groups practically investigating this architecture are WPI and Mitsubishi. The WPI group used the Xilinx Virtex FPGA devices, Mitsubishi group used semi-custom ASICs based on the 0.35 μm Mitsubishi Electric's CMOS ASIC library.

The WPI group implemented partial loop unrolling for four ciphers. The maximum number of unrolled rounds was equal to one half of the total number of rounds for RC6,

Group	Publications and presentations	Technology	Family of FPGA devices/ ASIC library	Device numbers	Implemented AES candidates
WPI - Worcester Polytechnic Institute	[EYCP00]	FPGA, CMOS, 0.22 μm	Xilinx Virtex	XCV 1000	4 (all except Mars)
Mitsubishi	[IKM00]	ASIC, CMOS, 0.35 μm	Mitsubishi Electric's library		5 (plus DES and Triple DES)

Table V Research groups comparing the hardware performance of the AES candidates using architectures with loop unrolling.

Rijndael and Twofish; and to all 32 rounds for Serpent. This limitation was imposed by the size of the FPGA device used for comparison.

In Fig. 19, we show the speed of the WPI implementations with the maximum number of rounds unrolled, and compare them with the speed of the basic iterative architecture. These results clearly demonstrate that loop unrolling offers only marginal speed-up over the basic iterative architecture, which does not exceed 10% for any of the AES candidates. Actually, the WPI group reports that the loop unrolling can even decrease the encryption throughput for Rijndael and Serpent I8. In our opinion, this

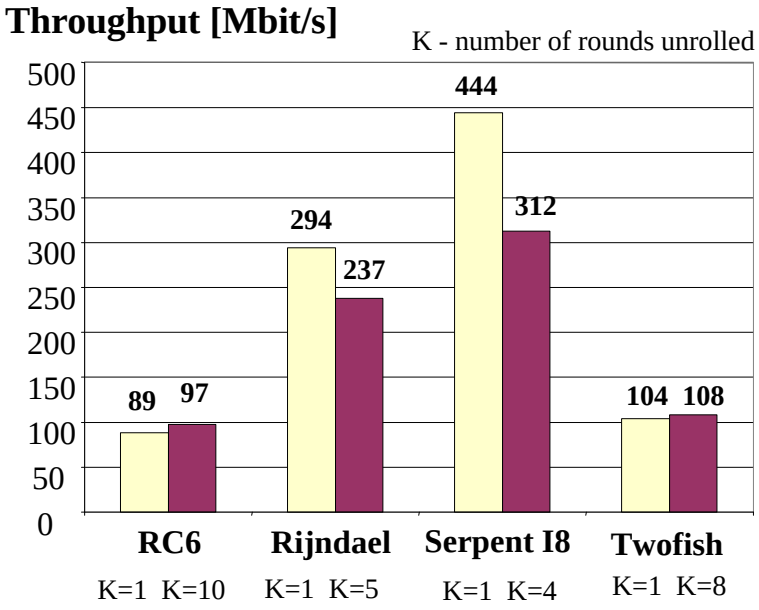

Fig. 19 The effect of loop unrolling on the throughput of the cipher. The WPI implementations using Xilinx Virtex FPGA - XCV 1000.

decrease could have been avoided, for Rijndael by the appropriate implementation of the last cipher round; and for Serpent, by effective buffering of signals with large fanouts.

The Mitsubishi group chose to implement full loop unrolling for each of the AES candidates, using the semi-custom ASIC technology based on the 0.35 μm Mitsubishi CMOS library of standard cells. The encryption throughputs reported by this group and shown in Fig. 20 are the highest among all results reported for implementations of the AES candidates in feedback cipher modes.

These high throughputs should be attributed primarily to the combination of the ASIC technology, intrinsically faster than the FPGA technology, and a superior 0.35 μm fabrication process, one generation ahead of the 0.5 μm fabrication process chosen by NSA. The speed-up resulting from choosing full-loop unrolling over the basic iterative architecture has not been reported, but based on the theoretical estimations and the WPI simulations, it is likely to be relatively small. The only AES candidate that benefits substantially from using the full loop unrolling is Mars. In any of the two basic iterative architectures of Mars shown in Fig. 8, the encryption latency is given by

$$32 \cdot \max\{\text{delay}_{\text{keyed\_transformation}}, \text{delay}_{\text{mixing\_transformation}}\}, \quad (7)$$

and thus, it is determined by the longer of the two critical paths through the mixing transformation and the keyed-transformation.

In the architecture based on full loop unrolling, the encryption latency is given by

$$8 \cdot \text{delay}_{\text{forward\_mixing\_transformation}} + 8 \cdot \text{delay}_{\text{forward\_keyed\_transformation}} + 8 \cdot \text{delay}_{\text{backwards\_keyed\_transformation}} + 8 \cdot \text{delay}_{\text{backwards\_mixing\_transformation}} \quad (8)$$

and thus, it is simply the sum of critical paths through all necessary transformations.

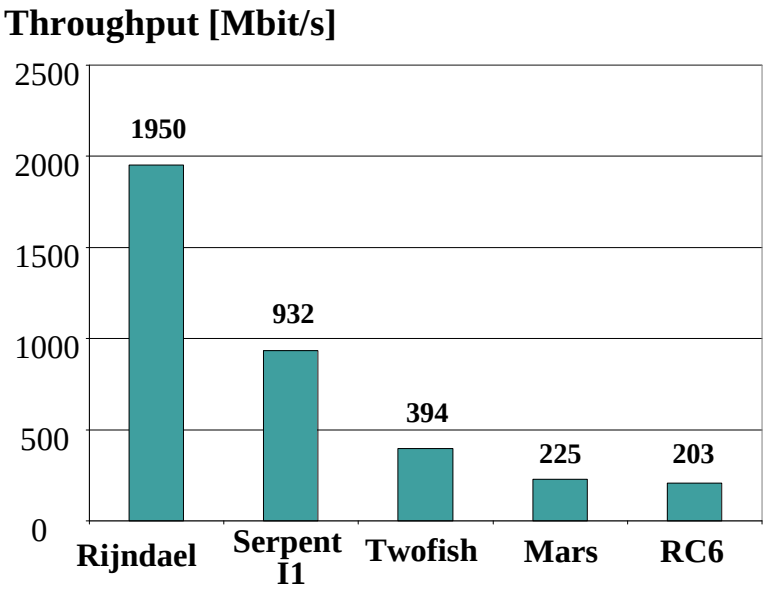

Fig. 20 Comparison of implementations of full loop unrolling using 0.35 μm Mitsubishi ASIC process.

Additionally, the critical path through each of the four different types of rounds is shorter than in the basic iterative architecture because the multiplexers used to switch between the forward and backward transformations are no longer necessary.

7. Comparing AES candidates in non-feedback cipher modes

7.1 Choice of an architecture

7.1.1 Alternative architectures

Six alternative hardware architectures that can be used for fast implementations of secret-key block ciphers operating in non-feedback cipher modes are shown in Fig. 21abc and Fig. 22bcd. Captions to these figures provide names of these architectures used in this article. Table VI give the correspondence between our names and names used in publications of various research groups.

Six alternative architectures can be divided into two major groups, with members of the first group shown in Fig. 21 and members of the second group shown in Fig. 22. Each group consist of a sequence of several architectures arranged in such a way that

- the next architecture in each sequence requires more area, but gives the higher throughput than the previous architecture in the sequence;
- there exist a simple transformation that leads to the subsequent architecture in each sequence.

Both sequences of architectures start from the well known basic iterative architecture, discussed in section 6.1.1, and chosen as an optimum architecture for feedback cipher modes.

The first sequence of architectures is used by the traditional design methodology for pipelined implementations of secret-key ciphers, known for years, and followed among the other in [WPRW99]. The second sequence of architectures is used by the new design methodology proposed independently by several research groups and described in [GaCh00b].

The traditional methodology for design of high-performance implementations of secret-key block ciphers, operating in non-feedback cipher modes is summarized in Fig. 21. The basic iterative architecture, shown in Fig. 21a is implemented first, and its speed and area determined. Based on these estimations, the number of rounds K that can be unrolled without exceeding the available circuit area is found. The number of unrolled rounds, K, must be a divisor of the total number of cipher rounds, *#rounds*. If the available circuit area is not large enough to fit all cipher rounds, architecture with partial outer-round pipelining, shown in Fig. 21b, is applied. The difference between this architecture and the architecture with partial loop unrolling, shown in Fig. 4b, is the presence of registers inside of the combinational logic on the boundaries between any two subsequent cipher rounds. As a result, K blocks of data can be processed by the circuit at the same time, with each of these blocks stored in a different register at the end of a clock cycle. This technique of paralell processing multiple streams of data by the same circuit is called pipelining. The throughput and area of the circuit with partial outer-round pipelining increase proportionally to the value of K, as shown in Fig. 23, the encryption/decryption latency remains the same as in the basic iterative architecture, as

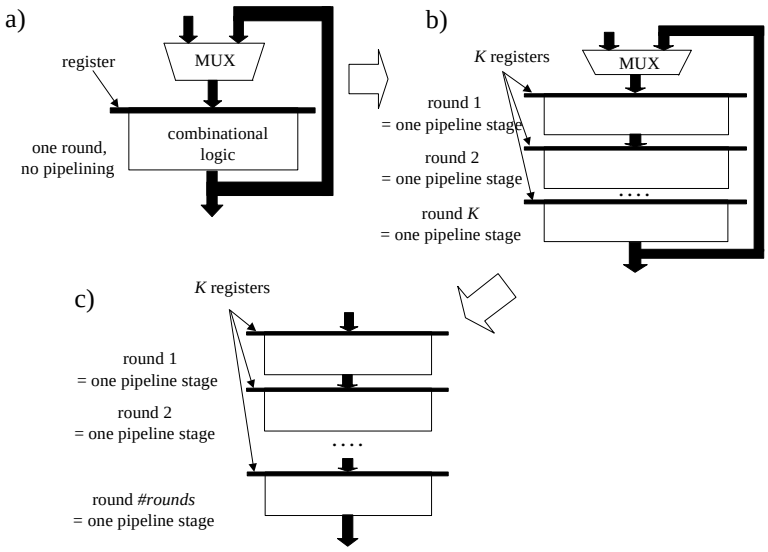

Fig. 21 Three architectures used traditionally to implement non-feedback cipher modes: a) basic iterative architecture, b) partial outer-round pipelining, c) full outer-round pipelining.

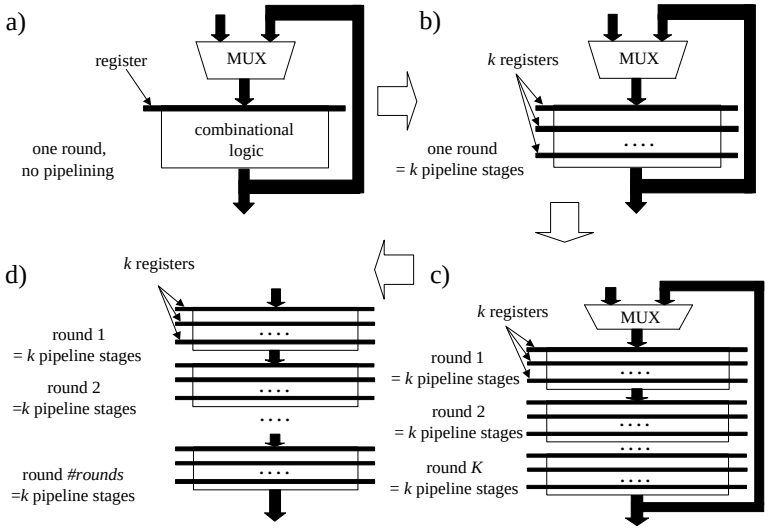

Fig. 22 Our architectures used to implement non-feedback cipher modes, a) basic iterative architecture, b) inner-round pipelining, c) partial mixed inner- and outer-round pipelining, d) full mixed inner- and outer-round pipelining.

shown in Fig. 24. If the available area is large enough to fit all cipher rounds, the feedback loop is not longer necessary, and full outer-round pipelining, shown in Fig. 21c, can be applied.

The new methodology for implementing secret-key ciphers in non-feedback cipher modes is shown in Fig. 22. The primary difference is that before loop unrolling, the optimum number of pipeline registers is inserted inside of a cipher round, as shown in Fig. 22b. The entire round, including internal pipeline registers is then repeated *K* times

(see Fig. 22c). The number of unrolled rounds K depends on the maximum available area or the maximum required throughput.

The primary advantage of the new methodology is shown in Fig. 23. Inserting registers inside of a cipher round significantly increases cipher throughput at the cost of only marginal increase in the circuit area. As a result, the throughput to area ratio increases until the number of internal pipeline stages reaches its optimum value k_{opt}. Inserting additional registers may still increase the circuit throughput, but the throughput to area ratio will deteriorate. The throughput to area ratio remains unchanged during the subsequent loop unrolling. The throughput of the circuit is given by

$$Throughput\,(K, k) = K \cdot block\_size \,/\, \#rounds \cdot \mathrm{T}_{\mathrm{CLKinner\_round}}(k) \qquad (9)$$

where k is the number of inner-round pipeline stages, K is the number of outer-round pipeline stages, and $\mathrm{T}_{\mathrm{CLKinner\_round}}(k)$ is the clock period in the architecture with the k-stage inner-round pipelining.

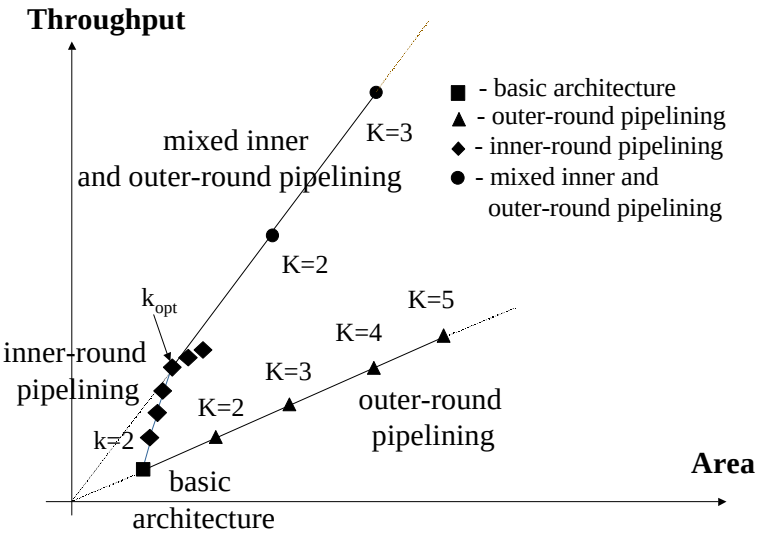

Fig. 23 Throughput vs. area characteristics of alternative architectures working in feedback cipher modes.

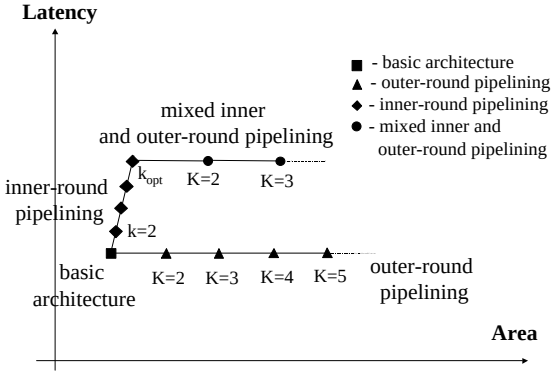

Fig. 24 Latency vs. area characteristics of alternative architectures working in feedback cipher modes.

This paper	NSA [WBRF00a, WBRF00b]	WPI [EYCP00]	USC [DPR00a, DPR00b]	Berkeley [WeWa00]	GMU [GaCh00a]
Partial outer-round pipelining	-	Partial pipelining, PP-*K*	Multiple-round based	Unrolled pipeline	*K*-stage outer-round pipelining
Full outer-round pipelining	Pipelined architecture	Full pipelining, PP-*N*	-	-	*N*-stage outer-round pipelining
Inner-round pipelining	-	Sub-Pipelining, SP-1-*y* (*y*=*k*-1)	-	*C*-slow single-round pipeline (*C*=*k*)	*k*-stage inner-round pipelining
Partial mixed inner- and outer-round pipelining	-	Partial Pipelining with Sub-Pipelining SP-*x*-*y* (*x*=*K*, *y*=*k*-1)	-	*C*-slow multiple-round pipeline (*C*=*k*)	-
Full mixed inner- and outer-round pipelining	-	Full pipelining with Sub-Pipelining	-	-	-

Table VI Names used in publications of various research groups for architectures suitable for non-feedback cipher modes.

For a given limit in the circuit area, mixed inner- and outer-round pipelining shown in Fig. 22c offers significantly higher throughput compared to the pure outer-round pipelining (see Fig. 23). When the limit on the circuit area is large enough, all rounds of the cipher can be unrolled, leading to the architecture with full mixed inner- and outer-round pipelining shown in Fig. 22d. The throughput of this architecture is given by

$$Throughput\ (\#rounds, k_{opt}) = block\_size \,/\, T_{CLKinner\_round}(k_{opt}) \quad (10)$$

where k_{opt} is the number of inner-round pipeline stages optimum from the point of view of the throughput to area ratio.

The only side effect of the new methodology is an increase in the encryption/decryption latency. This latency is given by

$$Latency(K, k) = \#rounds \cdot k \cdot T_{CLKinner\_round}(k) \quad (11)$$

It does not depend on the number of rounds unrolled, *K*.

When the combinational portion of a cipher round is divided into *k* equal pipeline stages, the increase in latency is given by

$$DLatency(K, k) = \#rounds \cdot (k\text{-}1)(t_P + t_{su}) \quad (12)$$

where t_P and t_{su} denote the propagation delay and the setup time of a register, respectively.

The increase in the encryption/decryption latency, typically in the range of single microseconds, usually does not have any major influence on the operation of the high-volume cryptographic system optimized for maximum throughput. This is particularly

true for applications with a human operator present on at least one end of the secure communication channel.

7.1.2 Choice of an architecture by various research groups

Three research teams comparing AES candidates using the pipelined architectures suitable for non-feedback cipher modes are introduced in Table VII. Each of these three teams decided to use a different architecture and design methodology. Their designs choices are summarized below using terminology introduced in this article in the previous section. These choices were made independently by every design team, and no effort was ever made to agree on a common architecture or methodology.

1. The NSA team chose to use the full outer-round pipelining shown in Fig. 21c.
2. The GMU group chose to use the full mixed inner- and outer-round pipelining shown in Fig. 22d, with the number of inner-round pipeline stages k optimized for the maximum throughput to area ratio (K=#rounds, $k=k_{opt}$).
3. The WPI team chose to follow the methodology summarized in Fig. 22 with the following additional assumptions:
   - The area of the circuit was limited in such a way that the entire circuit could be implemented using a single current-generation FPGA device (Virtex XCV-1000).
   - The number of the inner-round pipeline stages k was chosen depending on the complexity of the cipher round, and was set to k=1 for Serpent, k=2 for Twofish and Rijndael, and k=3 for RC6.

   The above assumptions led to the use of the full mixed inner- and outer-round pipelining (see Fig. 22d) in case of Serpent, and partial mixed inner- and outer-round pipelining (see Fig. 22c) in case of Rijndael, Twofish, and RC6, with the number of rounds unrolled, K= #rounds/2 for all three ciphers.

Group	Publications and presentations	Technology	Family of FPGA devices/ ASIC library	Device numbers	Implemented AES candidates
NSA - National Security Agency	[WBRF00a, WBRF00b]	ASIC, CMOS, 0.5 μm	MOSIS		5
GMU - George Mason University	[GaCh00b]	FPGA, CMOS, 0.22 μm	Xilinx Virtex	XCV 1000	4 (all except Mars)
WPI - Worcester Polytechnic Institute	[EYCP00]	FPGA, CMOS, 0.22 μm	Xilinx Virtex	XCV 1000	4 (all except Mars)

Table VII Research groups comparing hardware performance of the AES candidates using pipelined architectures suitable for non-feedback cipher modes.

Below we discuss the advantages and disadvantages of all three design choices. It should be understood that the opinions expressed below are of authors themselves, and are not necessarily shared by members of other research teams. The purpose of this summary is to stimulate further discussion, design, and research effort in this area.

In our opinion, a fair methodology for comparing hardware performance of the AES candidates should fulfill the following requirements.

a) It should be based on the architecture that is likely to be used in practical implementations, because of the superior throughput/area ratio.
b) It should not favor any group of ciphers or a specific internal structure of a cipher.

For feedback cipher modes, both conditions are very well fulfilled by the basic iterative architecture, and this architecture was commonly used for comparison. For non-feedback cipher modes, each group listed in Table VII used a different architecture, and no consensus regarding the optimum choice of the architecture was achieved.

7.1.2.1 NSA team's choice

The NSA team chose to use the full outer-round pipelining [WBRF00a, WBRF00b].

The primary advantage of this choice is the simplicity of the design process. The number of the pipeline stages is determined uniquely by the specification of each cipher, and no additional assumptions and decisions regarding the choice of the architecture must be made. Additionally, this architecture offers very high throughput with no increase in the encryption latency compared to the basic iterative architecture.

In our opinion, this choice has also the following disadvantages:

First, as shown in Fig. 23, the outer-round pipelining offers significantly worse throughput to area ratio compared to the architecture with the mixed inner- and outer-round pipelining. Therefore, the use of this architecture may lead to suboptimum designs, which are not likely to be used in practice.

Secondly, the choice of the outer-round pipelining favors ciphers with a short and simple cipher round, such as Serpent and Rijndael. The AES candidates with more complex internal rounds, such as Mars, RC6, and Twofish, are adversely affected.

This unequal treatment of AES candidates can be explained by looking at the formulas describing encryption throughput in full outer-round pipelining (13) and full mixed inner- and outer-round pipelining (14). The encryption throughput in the full outer round pipelining is given by

$$Throughput_{\text{full\_outer\_round}} = block\_size \,/ T_{\text{CLKbasic}} \qquad (13)$$

where T_{CLKbasic} is a delay of a single round.

The throughput does not depend any longer on the number of cipher rounds, but is inversely proportional to the delay of a single round. Ciphers with the large number of simple rounds are favored over ciphers with the small number of complex rounds.

On the other hand, the throughput in the full mixed inner and outer-round pipelining is given by

$$Throughput_{\text{full\_mixed}} = block\_size \,/ T_{\text{CLKinner\_round}}(k_{\text{opt}}) \qquad (14)$$

where $T_{\text{CLKinner\_round}}(k_{\text{opt}})$ is the delay of a single pipeline stage for the optimum number of registers introduced inside of a single round.

In FPGA implementations, this delay is determined by the delay of a single CLB slice and delays of interconnects between CLBs. In semi-custom ASIC implementations, this

delay is likely to be determined by the delay of a single logic gate (or a more complex standard cell) and by interconnects between logically adjacent gates. In both technologies, the throughput does not depend on the complexity of a cipher round and tend to be similar for all AES candidates.

7.1.2.2 GMU's team choice

The primary advantage of this choice is the use of the architecture with the maximum throughput to area ratio, and the maximum overall throughput, which is likely to be used in practical implementations. Secondly, the chosen architecture does not favor any specific type or internal structure of a secret-key block cipher. In particular, ciphers with complex internal rounds can achieve the same throughput as ciphers with much simpler rounds.

The primary disadvantage of this architecture is the difficulty of the design process. The design includes choosing the optimum number of the inner-round pipeline stages, and then choosing the optimum location for each of the pipeline registers. This process can be hard to automate using the current generation of the computer-aided design tools.

The design is simplified in case of FPGAs, where the optimum number of the inner-round pipeline stages is likely to be equal to the number of CLB levels in the critical path of the basic iterative architecture. Additionally, the position of the pipeline registers is determined by mapping of the round operations into CLB slices. Nevertheless, the result may depend on the choice of the family of FPGA devices. Choosing the optimum number of the inner-round pipeline stages for semi-custom ASICs may be more difficult, and the result may depend on the available standard-cell library.

The second disadvantage of this architecture is the significant increase of the encryption/decryption latency compared to the basic iterative architecture, as shown in Fig. 24 and Fig. 31. This increase is particularly large for ciphers with the relatively complex cipher round. Nevertheless, the time necessary to encrypt a long stream of data is primarily a function of the encryption throughput, and is almost independent of the encryption latency. Since a primary function of the high-speed hardware cryptographic devices is to encrypt long streams of data, the increase in latency might not be critical in practical applications. This matter should be further investigated taking into account the properties of today's high-speed networks.

Finally, unrolling all cipher rounds might impose large area requirements, which is especially critical in case of the low-cost FPGA devices and embedded ASICs. This problem can be dealt with by using multi-chip modules or switching to the partial mixed inner- and outer-round pipelining with the optimum number of the inner-round pipeline stages (see Fig. 22c and Fig. 23).

7.1.2.3 WPI's team choice

The WPI approach has several important advantages. In several respects, it is a middle ground between an aggressive GMU approach and a conservative NSA approach. The design procedure is simpler than in the GMU approach, because the number of the inner-round pipeline stages k is chosen to be small, ranging from 1 to 3 depending on the complexity of a cipher round. An increase in the encryption latency is also significantly

smaller than in the GMU approach, nevertheless greater than in the NSA approach. Additionally, the WPI team assumes a realistic limit on the maximum area of the AES implementation, which is chosen based on the size of one of the largest currently available Xilinx Virtex FPGA devices. This approach has two advantages: first, it limits the AES candidate comparison to one parameter - the encryption throughput (vs. two parameters, throughput and area, that must be taken into account in other approaches); secondly it demonstrates the realistic capabilities of each algorithm in the current-generation of FPGA devices.

At the same time the WPI approach and its implementation has the following disadvantages. Choosing a small number of the inner round pipeline stages leads to the designs suboptimum from the point of view of the encryption throughput. This feature is demonstrated in Fig. 25, where the speed-ups of the pipelined architectures over the basic iterative architectures are demonstrated for each of the three design approaches. It can be seen that the speed-ups obtained using the WPI approach are from 3.4 to 5.7 times smaller than the speed-ups demonstrated using the GMU approach. If the limit on the circuit area was not used in the WPI approach, the speed-ups for Twofish, RC6, and Rijndael would increase by a factor of two. Nevertheless, these speed-ups would still be smaller than the GMU speed-ups by a factor ranging from 1.7 for RC6 to 3.5 for Serpent I8.

Additionally, the implementation of the WPI approach described in [EYCP00] has the following drawback. The number of the inner-round pipeline stages is not chosen proportionally to the length of the critical path through a cipher round. For example, the ratio of the critical paths of RC6 and Twofish is 0.94, and the ratio of the number of inner-round pipeline stages is 3/2, which gives RC6 a 50% advantage over Twofish. Based on the clock frequencies of the basic iterative architecture reported in [EYCP00]

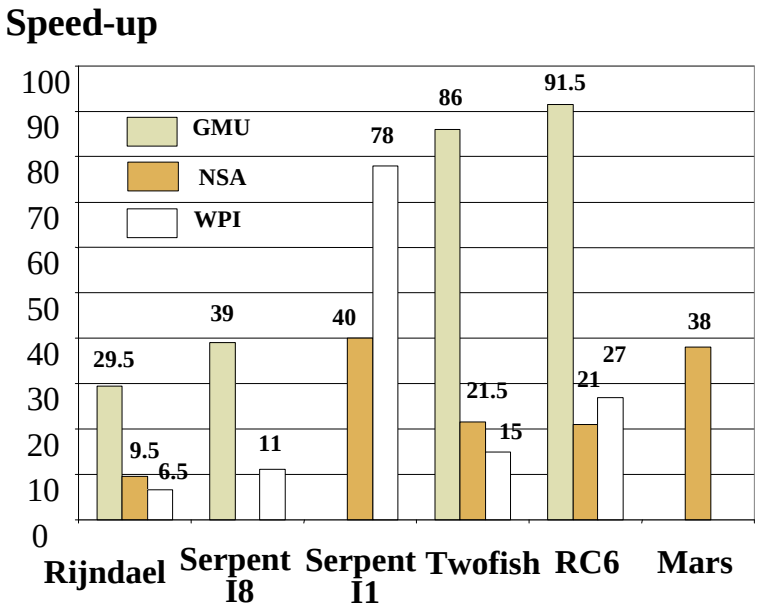

Fig. 25 Speed-up compared to the basic iterative architecture obtained by three independent research groups by applying a pipelined architecture of their choice.

(25.6 MHz for Rijndael, 13.0 MHz for Twofish, and 13.8 MHz for RC6), the much fairer choice of the number of inner round pipeline stages would be 1 (instead of 2) for Rijndael, 2 for Twofish, and 2 (instead of 3) for RC6.

Introducing a limit on the maximum circuit area, apart from its advantages described above, has also the following disadvantages.

First, any limit on the circuit area applies only to the current generation of FPGA devices. This limit is likely to substantially increase in the future, which will strongly affect the results of comparison. For example if the maximum circuit area increased twice, the encryption throughputs of Rijndael, Twofish, and RC6 would also increase twice, while the encryption throughput of Serpent would remain the same. These changes would substantially change the relationships among throughputs of Serpent and other ciphers, eliminating the big advantage of Serpent reported in [EYCP00] and shown in Fig. 28 (WPI results).

Secondly, even if only the current-generation FPGA devices are in use, implementing the more advanced architectures, and achieving the higher throughputs is possible by using multiple FPGA devices driven by the global external system clock.

Finally, the assumption about implementing encryption and decryption circuits separately, and switching between these two circuits through the FPGA reconfiguration has two disadvantages. First, the results cannot be directly compared to ASICs, where the reconfiguration is not feasible. Secondly, the current generation of FPGA devices has a reconfiguration time in the range of several milliseconds, which might be prohibitive for many applications.

7.2 Results

Below we discuss and compare results obtained by all three design teams.

7.2.1 GMU approach

The results of the GMU implementations of four AES candidates using full mixed inner- and outer-round pipelining and Virtex XCV-1000 FPGA devices are summarized in Fig. 26, Fig. 30, and Fig. 31.

The deviations in the values of the AES candidate throughputs in full mixed inner- and outer-round pipelining do not exceed 20% of their mean value. All critical paths contain only a single level of CLBs and differ only in delays of programmable interconnects. For example, the relatively smaller throughput of Rijndael is the result of relatively long interconnects between Embedded Array Blocks (EABs) and logically adjacent CLBs. The differences in interconnect delays could be probably further reduced by a second-level optimization. Taking into account already small spread of the AES candidate throughputs and potential for further optimizations, we conclude that the demonstrated differences in throughput are not sufficient to favor any of the AES algorithms over the others. As a result, circuit area should be the primary criterion of comparison in this architecture.

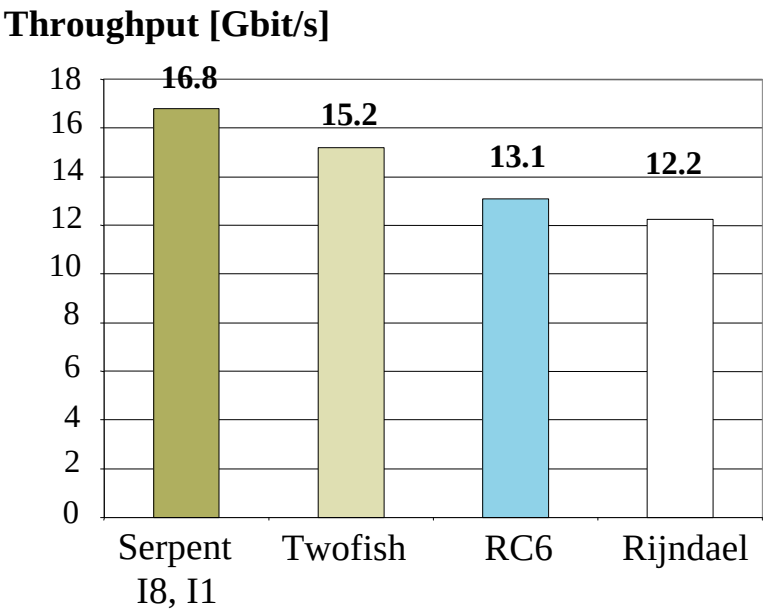

Fig. 26 Full mixed inner- and outer-round pipelining throughput for Virtex XCV-1000, GMU results.

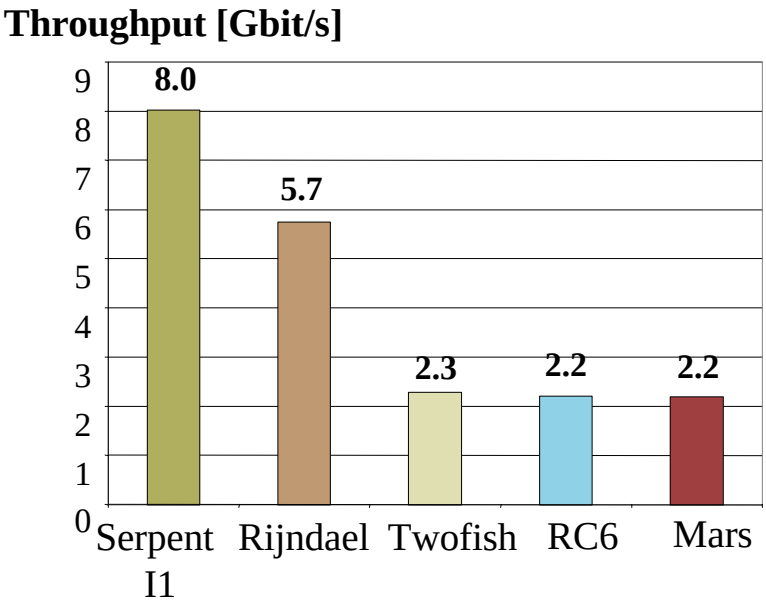

Fig. 27 Full outer-round pipelining, throughput for 0.5 μm CMOS standard-cell ASICs, NSA results.

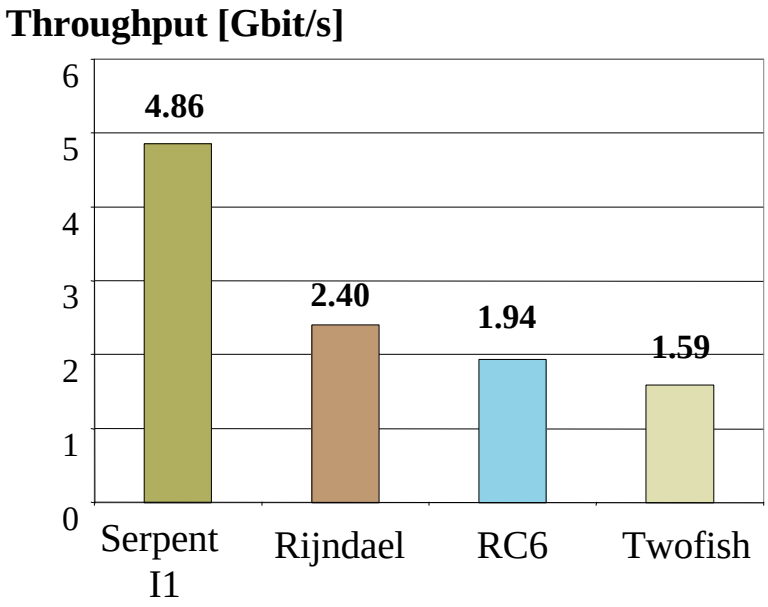

Fig. 28 Partial mixed inner- and outer-round pipelining throughput for Virtex XCV-1000, WPI results.

As shown in Fig. 30, Serpent and Twofish require almost identical area for their implementations based on full mixed inner- and outer-round pipelining. RC6 imposes over twice as large area requirements. Comparison of the area of Rijndael and other ciphers is made difficult by the use of dedicated memory blocks, Embedded Array Blocks (EABs), to implement large S-boxes. EABs are not used in implementations of any of the remaining AES candidates, and we are not aware of any formula for expressing the area of EABs in terms of the area used by CLB slices. Nevertheless, we have estimated that a function of each EAB in our implementation of Rijndael could be implemented with about 150 CLB slices [GaCh00b]. Therefore, an equivalent implementation of Rijndael, composed of CLBs only, would take about 24,600 CLBs, which is only 17 to 25 percent more than the implementations of Twofish and Serpent.

Additionally, Serpent, Twofish, and Rijndael all can be implemented using two FPGA devices XCV 1000; while RC6 requires four such devices. It should be noted that in the GMU designs, all implemented circuits perform both encryption and decryption. This is in contrast with the designs reported by WPI in [EYCP00], where only encryption logic is implemented, and therefore a fully pipelined implementation of Serpent can be included in one FPGA device.

Connecting two or more Virtex FPGA devices into a multi-chip module working with the same clock frequency is possible because the FPGA system level clock can achieve rates up to 200 MHz [Xili99], and the highest internal clock frequency required by the pipelined AES candidate implementation is 131 MHz for Serpent. New devices of the Virtex family, scheduled to be released in 2001, are likely to be capable of including full implementations of Serpent, Twofish, and Rijndael on a single integrated circuit.

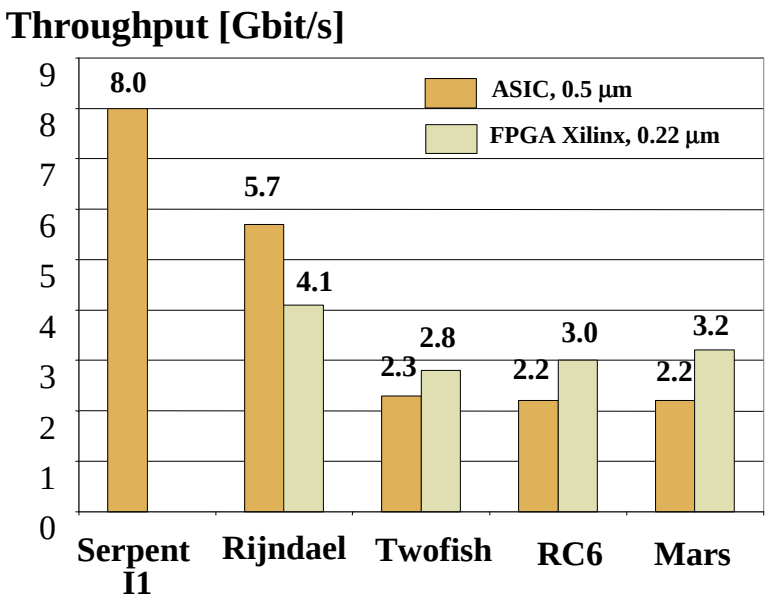

Fig. 29 Full outer-round pipelining. Comparison between the simulation results of the NSA group for ASICs and estimations based on the best results for the FPGA implementations of the basic iterative architecture, used as an input in formula (13).

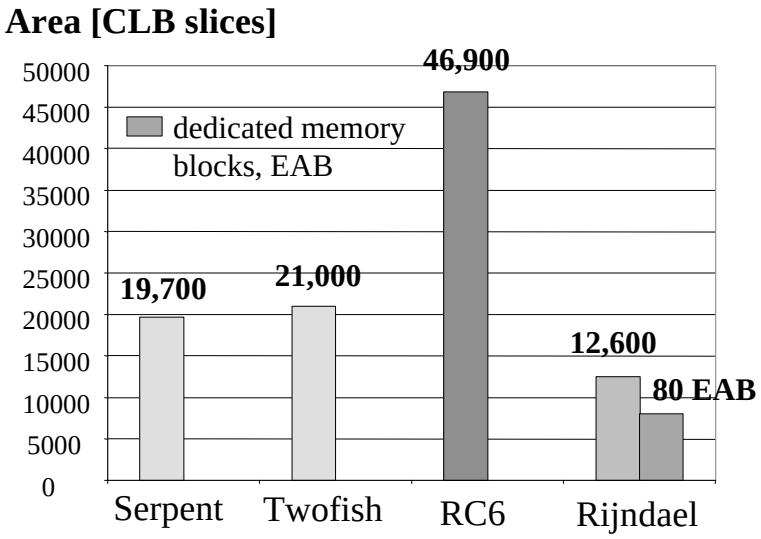

Fig. 30 Full mixed inner- and outer-round pipelining, area for Virtex XCV-1000, GMU results. Multiple XCV-1000 devices used when necessary.

In Fig. 31, we report the increase in the encryption/decryption latency resulting from using the inner-round pipelining with the number of stages optimum from the point of view of the throughput/area ratio. The latency increases by a factor of about 2.5 for Serpent and Rijndael, ciphers with a relatively simple cipher round, and by factors 4.3 and 6.1 respectively for Twofish and RC6, ciphers with more complex cipher rounds. The absolute value of the latency in architecture with full mixed inner- and outer-round

pipelining is below 1µs for Serpent and Rijndael, about 3 µs for Twofish, and below 6 µs for RC6. In majority of applications that require hardware-based high-speed encryption, the encryption/decryption throughput is a primary performance measure, and the aforementioned values of latency are fully acceptable. Therefore, in this type of applications, the only parameter that truly differentiates AES candidates, working in non-feedback cipher modes, is the area, and thus the cost, of implementations. As a result, in

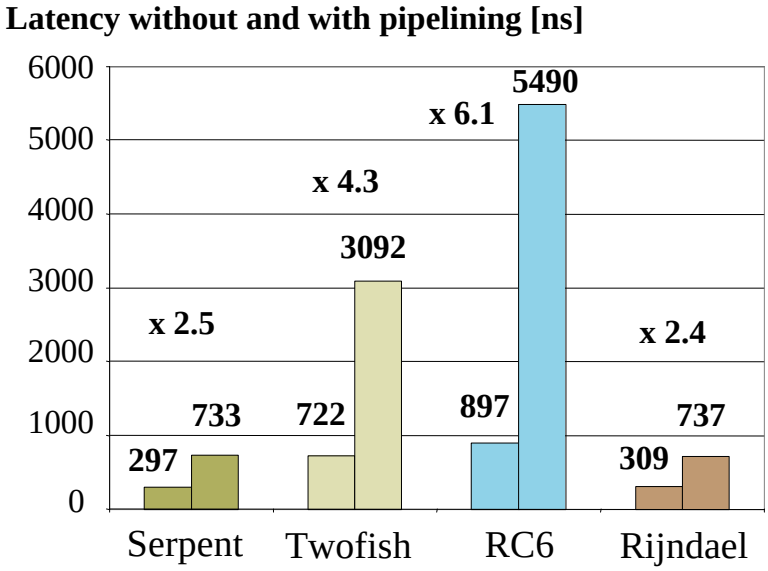

Fig. 31 Increase in the encryption/decryption latency as a result of moving from the basic iterative architecture to full mixed inner- and outer-round pipelining, GMU results. The upper number (after 'x') shows the ratio of latencies.

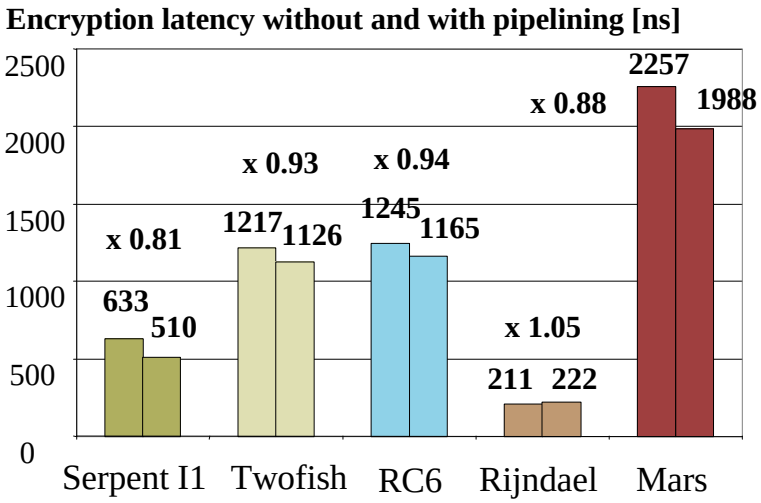

Fig. 32 Change in the encryption latency as a result of moving from the basic iterative architecture to full outer-round pipelining, NSA results. The upper number (after 'x') shows the ratio of latencies.

the GMU's approach, Serpent, Twofish, and Rijndael offer very similar performance characteristics, while RC6 requires over twice as much area and twice as many Virtex XCV-1000 FPGA devices.

7.2.2. NSA results

In Fig. 27, we collect results obtained by the NSA group by implementing all five AES finalists, using full outer-round pipelining and semi-custom ASICs based on the 0.5 μm CMOS MOSIS library [WBRF00b].

In Fig. 25, it can be clearly seen that the speed-up of the full outer-round architecture over the basic iterative architecture is equal approximately to the total number of the cipher rounds, *#rounds*. This dependence is almost perfect for Rijndael and RC6. For Serpent I1, Mars, and Twofish a slightly larger speed-up was achieved. This additional speed-up was possible primarily by a simplification of a cipher round. For example, in the pipelined implementation of Serpent I1, only one set of S-boxes must be implemented per round (vs. 8 sets of S-boxes for the basic iterative architecture shown in Fig. 7a), and the 8-to-1 multiplexer can be eliminated. Additionally, for all ciphers, no feedback loop is necessary, and the input 2-to-1 multiplexer can also be eliminated.

As shown in Fig. 27, Serpent achieves the highest speed, over 3.5 times higher than the speed of Twofish, RC6, and Mars. Rijndael is only 30% slower than Serpent, and outperforms the remaining three ciphers by a factor of over 2.5 times. By comparing Fig. 26 and Fig. 27, it can be clearly seen that using full outer-round pipelining (NSA approach) for comparison of the AES candidates favors ciphers with less complex cipher rounds. For example, Twofish and RC6 are over two times slower than Rindael and Serpent, when full outer-round pipelining is used (NSA approach); and have the throughput greater than Rijndael, and comparable to Serpent, when full mixed inner- and outer-round pipelining is applied (GMU approach).

By using the best results of the FPGA implementations of the AES ciphers in the basic iterative architecture (shown in Fig. 10), and applying formula (13), it is possible to estimate the throughput of the four AES ciphers for the full outer-round pipelining and FPGA implementation. These results are collected in Fig. 29. The relative ranking of algorithms is similar, with the differences among Rijndael and three other ciphers reduced in the FPGA technology.

The area used by the encryption/decryption units of the AES ciphers is not directly reported in [WBRF00b]. Only the total area, including key scheduling unit, is given. The ranking of the algorithms in terms of the area of the encryption/decryption unit itself can be however derived based on the pipelined transistor count listed separately for each unit [WBRF00b, sec. 7.2.9]. Based on this transistor count, Mars and Serpent require almost twice as much area as Twofish, RC6, and Rijndael. Please, note that the area required for RC6 is relatively smaller than in the GMU's approach, and the area required by Serpent is relatively higher than in the GMU's approach. These differences may be partially explained by the fact that the pipelined registers tend to contribute more to the circuit area count in the ASIC technology compared to the FPGA technology.

In terms of the encryption/decryption latency (time to encrypt/decrypt one block), the NSA's pipelined architecture achieves shorter latency than the corresponding basic

iterative architecture by a factor ranging from 6 % for RC6 to 19% for Serpent. Only the latency of Rijndael increases by 6% for encryption and 17% for decryption.

7.2.3 WPI results

The WPI group results are shown in Fig. 28 and Fig. 33. Additionally, the areas of all implementations expressed in the number of CLB slices are approximately equal, and range from 9,000 to 11,000 CLB slices. These areas are limited by the number of CLB slices in the XCV-1000 FPGA device, equal to 12,288.

The WPI results for throughput, shown in Fig. 28, demonstrate that Serpent is over twice as fast as three other candidates. Rijndael, RC6, and Twofish offer a similar performance, with all three throughputs within 20% of their mean value. Mars was not implemented by the WPI group.

These results are a very strong function of assumptions used by the WPI group, summarized in section 7.1.2.3. The big advantage of Serpent over the other AES candidates is the combined effect of two major assumptions: limit on the circuit area and implementation of the encryption part only. Because of the limit on the circuit area, Rijndael, RC6 and Twofish have only half of their rounds unrolled, while Serpent has all its rounds unrolled. If this limitation was not used, Rijndael, RC6, and Twofish would increase their throughput approximately twice, which would make them almost as fast as Serpent. Additionally, the only reason why the implementation of Serpent can have all rounds unrolled within the current limitation on the circuit area is that only encryption function is implemented for all ciphers. This assumption affects Serpent the most, as it is the only AES candidate in which implementation of decryption requires approximately the same amount of area as the implementation of encryption. Adding decryption to the encryption unit increases the overall area by no more than 20% for Twofish and RC6, and by approximately 55% for Rijndael [see GaCh00a, Table II]. Both WPI assumptions are

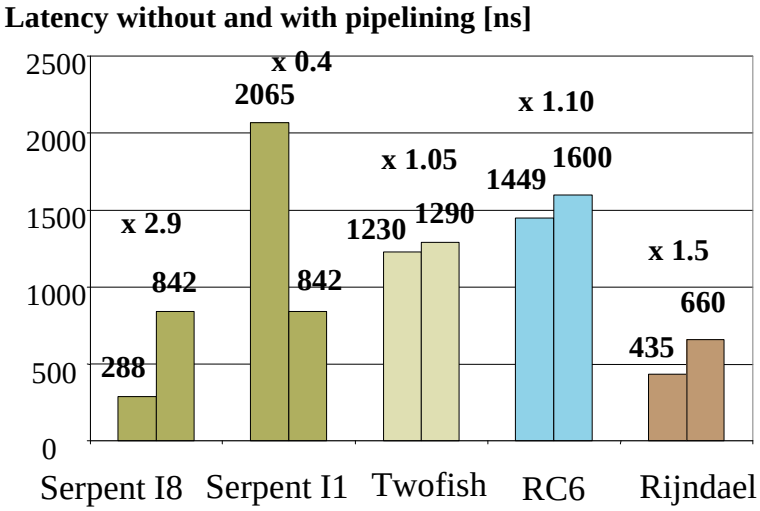

Fig. 33 Increase in the encryption/decryption latency as a result of moving from the basic iterative architecture to mixed inner- and outer-round pipelining, WPI results. The upper number (after 'x') shows the ratio of latencies.

specific to the FPGA technology, and are not applicable to any ASIC technology, where the limit on the circuit area is much larger, and both encryption and decryption must be implemented within the same integrated circuit.

The increase in the encryption latency compared to the iterative architecture is within 10% for Twofish and RC6, and about 50% for Rijndael, as shown in Fig. 33. For Serpent I1, the encryption latency actually decreases because of the elimination of the path through the 8-to-1 128-bit multiplexer (see Fig. 7a) in the pipelined implementation. The absolute values of latencies are slightly larger than in the NSA designs, but smaller than in the GMU designs.

8. Comparing hardware performance of key scheduling

8.1 Definition of parameters

The most important parameter describing the operation of the key scheduling unit is the *encryption (decryption) key setup latency*. This parameter is defined as the amount of time necessary to begin encryption (decryption) after providing the input key [DPR00b, WBRF00b]. Unfortunately, this definition is interpreted differently by various groups. For example, the USC group requires the first internal key to be computed as a part of key setup [DPR00b]. The NSA group assumes that it is sufficient to calculate an input to the key expansion unit of the key scheduling unit [WBRF00b]. Based on this input, all internal keys can be computed in subsequent clock cycles. Still other groups might define the key setup latency as the time necessary to compute all internal keys. In this article, we will define the encryption (decryption) key setup latency as the time necessary to compute key-related variables (such as internal keys) required to perform the *first* round of encryption (decryption). We also will assume that the key expansion circuit must be capable of computing internal keys for each subsequent round in the time shorter than the time necessary to execute the corresponding encryption (decryption) round.

This paper	**NSA** [WBRF00a, WBRF00b]	**WPI** [EYCP00]	**USC** [DPR00a, DPR00b]	**Berkeley** [WeWa00]	**GMU** [GaCh00a, GaCh00b]
Encryption key setup latency	Key setup time (encrypt), Time to switch keys (encrypt)	-	Latency, Key-Setup Latency	Key setup time	-
Decryption key setup latency	Key setup time (decrypt), Time to switch keys (decrypt)	-	-	Key setup time	-

Table VIII Names of the key-scheduling parameters used in publications of various research groups.

Group	Publications and presentations	Technology	Family of FPGA devices/ ASIC library	Device numbers	Implemented AES candidates
NSA - National Security Agency	[WBRF00a, WBRF00b]	ASIC, CMOS, 0.5 μm	MOSIS		5
USC - University of Southern California	[DPR00a, DPR00b]	FPGA, CMOS, 0.22 μm	Xilinx Virtex	XCV 1000	5

Table IX Research groups comparing hardware implementation of key scheduling for all five AES candidates.

8.2 Results

Only two research groups, NSA and USC, has reported their results regarding the comparison of the hardware performance of key scheduling units for final AES candidates (see Table IX). Other groups assumed that the generation of internal keys is performed off-the-chip, and all these keys are downloaded to the internal memory using an input interface [GaCh00a, EYCP00].

The USC group limited their investigation to the encryption key setup latency, the NSA group considered both the encryption and decryption key setup latencies. As a result, the only parameter that was determined by more than one group was the encryption key setup latency.

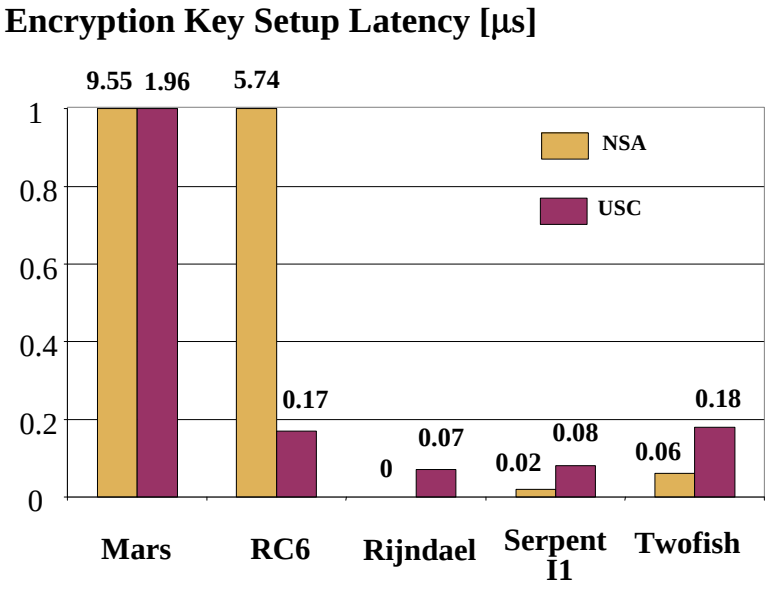

Fig. 34 Comparison of results regarding the encryption key setup latency obtained by the NSA and USC groups.

In Fig. 34, we present the absolute values of this parameter obtained independently by the NSA and USC groups. The differences between the results of both groups are extremely large. The ratio of key setup latencies ranges from 1:4 in favor of the NSA group for Serpent, to 33.8:1 in favor of the USC group for RC6. These differences are even more clear when the number of clock cycles necessary for the key setup operations are compared, as shown in Table X. The ratio of clock cycles ranges from 4:1 in favor of the NSA group for Twofish, to 30.7:1 in favor of the USC group for RC6. Additionally, key setup for Rijndael takes zero clock cycles in the NSA design, and two clock cycles in the USC design.

Apart from the small difference in the definition of the key setup latency, described in section 8.1, the other assumptions seem to be identical for both groups. Both groups assume that the encryption starts as soon as the first key is computed, and the remaining keys are computed in parallel with encryption. Both groups also assume that the internal keys are stored in a dual-port memory with separate data inputs and outputs. As soon as all internal keys are computed, the key scheduling unit is put on hold, until the next external key needs to be processed. Therefore, the large differences in results can only be explained by significant differences in the internal architecture of the key scheduling unit. Unfortunately, the reports of both groups do not describe their respective architectures to the level of detail sufficient to explain any of these differences. The only good explanation of large differences in results exists for Mars, where the USC team decided to skip in their design one of the most time and area consuming operations, string matching [DPR00a].

In spite of relatively large differences in absolute values of the key setup latencies, results from both groups clearly demonstrate that the AES candidates can be divided into two major groups. In the first group, the key setup latency is only a fraction of the time necessary to encrypt a single block of data. As a result, switching to a new external key can be performed on-the-fly, at the time when the last block of data is encrypted using a previous external key. There is no delay, or extra circuitry associated with the key change. This group of ciphers includes Rijndael, Serpent, and Twofish. In all these ciphers, the number of clock cycles necessary to compute the first internal key is no greater than 25% of the number of clock cycles necessary to encrypt a single block of data. In the second group of ciphers, the key setup time is greater than the time necessary to encrypt a single block of data. As a result, changing the key either introduces an additional delay, or requires an extra circuitry to store new internal keys during the time when the previous internal keys are still in use. According to both the USC and NSA results, this second group of ciphers includes Mars. The classification of RC6 is different in reports of both research groups. According to the USC group, RC6 belongs to the first group, with the key setup time taking only 15% of the time necessary to encrypt a single block of data. According to the NSA report, RC6 belongs to the second group, with the key setup time 4.6 times larger than the time necessary to encrypt a single block of data. The classification of RC6 is a major inconsistency between results of both research groups.

The second common result obtained by the NSA group and the USC group is that for all AES candidates, the minimum clock period of the key expansion unit is smaller than the minimum clock period of the encryption/decryption unit implemented using the basic iterative architecture. As a result, the next set of internal keys can always be computed in

Algorithm	USC	NSA	
	Encryption	*Encryption*	*Decryption*
Mars	50	144	144
RC6	3	92	92
Rijndael	2	0	10
Serpent I1	3	1	34
Twofish	4	1	2

Table X Number of clock cycles required for the key setup in the USC and NSA architectures.

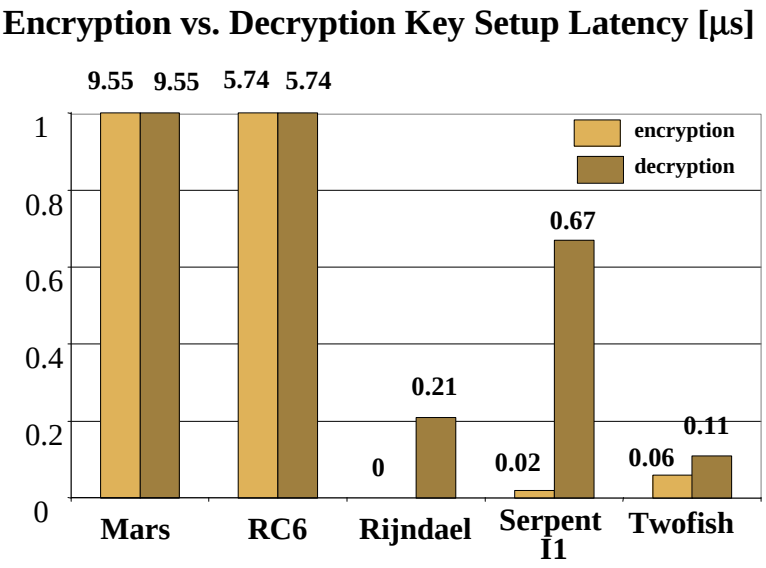

Fig. 35 The differences between the encryption and decryption key setup latencies according to the NSA group.

time for the next round of the encryption and decryption process, and the encryption (decryption) throughput is not affected by the operation of the key scheduling unit.

In Fig. 35, the differences between the encryption and decryption key setup latencies are demonstrated, as reported by the NSA group [WBRF00b]. These differences are the largest for Serpent and Rijndael. In these ciphers, the computation of internal keys cannot be easily performed in the reverse order, therefore, after an external key is changed, all corresponding internal keys have to be calculated before the decryption can begin. The NSA group report states that the decryption key setup latency can be significantly reduced for Serpent, by computing the last internal key directly from the external key. Nevertheless, no details of this procedure are provided in [WBRF00b].

9. Weights of particular parameters

In [Smi00a] Miles Smid, suggested the following measure of the hardware efficiency, h_i:

$$h_i = (\text{HardwareSpeed}_i * \text{HardwareKeyAgility}_i) / \text{HardwareMemory}_i \quad (15)$$

Below we suggest an alternative measure, and discuss the choice and values of parameters and weights used in this measure.
Our formula looks as follow:

$$H_i = (w_{tf} \cdot \text{Throughput\_feedback} + w_{taf} \cdot \text{Throughput\_Area\_Characteristics\_feedback} + w_{kf} \cdot 1/\text{Key\_Setup\_Latency\_feedback} + w_{kaf} \cdot 1/\text{Key\_Scheduling\_Area\_feedback}) + (w_{tnf} \cdot \text{Throughput\_non-feedback} + w_{anf} \cdot \text{Encryption\_Area\_non-feedback} + w_{lnf} \cdot \text{Latency\_non-feedback} + w_{knf} \cdot 1/\text{Key\_Setup\_Latency\_non-feedback} + w_{kanf} \cdot 1/\text{Key\_Scheduling\_Area\_non\_feedback}) \quad (16)$$

The primary difference between measures defined by formulas (15) and (16) are as follows:

- Our measure H_i allows us to clearly distinguish between the behavior of the AES candidates in feedback and non-feedback cipher modes. This distinction is very important because of the
  - large differences in the relative ranking of the AES algorithms in feedback and non-feedback cipher modes;
  - different level of utilization of both types of modes in the existing cryptographic protocols, networks, and products;
  - different amount of effort devoted to the analysis of the hardware performance of feedback and non-feedback modes, and a different level of consistency between results obtained by various groups.
- The weights assigned to particular parameters are identical in h_i (15) and different in H_i (16). Assigning different weights to various parameters is important because
  - not all parameters are equally important in practical applications;
  - a different amount of knowledge was collected about various parameters during the analysis. For example, some parameters were selected as optimization targets, while others were never optimized for.

The general rule of selecting weights in (16) would be to assign higher weights to parameters which are

- more important in majority of practical applications,
- the best studied, with consistent results confirmed by at least two independent groups.

We suggest that both weights and parameters should be treated as numbers in the range from 0 to 1. In case of the performance parameters, such as throughput, an algorithm which is the first in the ranking should be assigned the value 1.0. The parameter values for the other algorithms should be normalized accordingly.

Below we discuss the ways of determining values of particular parameters and selecting weights associated with these parameters.
In general, all weights associated with feedback cipher modes should be significantly greater than weights for non-feedback cipher modes. There are two major reasons for this choice. First, today's standards, protocols, and cryptographic products use almost exclusively feedback modes, such as CBC and CFB. Secondly, there are much fewer results regarding the analysis of the AES candidate performance in non-feedback modes, and these results are inconsistent among each other.

Both aforementioned reasons have only temporary value:

Non-feedback modes, such as the counter mode, are already a part of the standardization effort for secure ATM networks. The interleaved modes, such as the interleaved CBC mode shown in Fig. 2, have a potential to offer security of feedback modes combined with the performance of non-feedback modes. Both of these types of modes are likely to be considered by NIST for standardization as future AES operating modes [SKBC00], and become a part of other standardization efforts.

The different results obtained by various research groups for non-feedback cipher modes can be quite easily explained based on the different architectures and assumptions used during analysis, as demonstrated in Sections 7.1.2 and 7.2. Common conclusions can be derived and approved by all research groups. Nevertheless, to date, this consensus has not been achieved, and therefore the results for non-feedback modes cannot be treated with the same level of certainty as results for feedback operating modes.

After considering these two major groups of parameters, below we look at each individual parameter separately:

a. Throughput_feedback - Throughput for feedback modes

The maximum throughput in feedback cipher modes is the most important parameter determining the maximum speed of encryption and decryption in real life applications used for processing of large amounts of data. This throughput has been analyzed by several research groups, as shown in Table IV. All these research groups chose throughput as a major target for optimization. The results obtained independently by these groups led to an identical or almost identical ranking of all algorithms, as shown in Fig. 9 and Fig. 14. In Fig. 14, the best results obtained for various semiconductor technologies are summarized. This figure can be used to determine values of the normalized throughput for all AES candidates. The weight w_{tf}, associated with this parameter, should be the largest among all weights used in equation (16).

b. Throughput_Area_Characteristics_feedback - Throughput vs. Area characteristics for feedback modes

By the *Throughput vs. Area characteristics* we understand the location of the cipher on a two-dimensional diagram showing the dependence between the encryption throughput and the area of the integrated circuit implementing the cipher. Two such diagrams, for the FPGA and ASIC implementations, accordingly, are shown in Fig. 17 and Fig. 18. Both diagrams are in almost perfect agreement. They allow dividing AES candidates into three distinct groups. Ciphers from the same group should be rated similarly. The first group is formed by Rijndael and Serpent, the second group includes RC6 and Twofish, and the third group is composed of Mars itself. The *Throughput vs. Area characteristics* determines the maximum speed of the cipher in environments with the limited amount of circuit area. By applying resource sharing, each implementation can be scaled down to fit within the given area, at the cost of the proportional decrease in the circuit throughput. The weight of this parameter, w_{taf}, should be smaller than the weight of the cipher throughput, w_{tf}, but still very significant because of a very good agreement among various groups, and the role of this parameter in applications with the limited circuit area.

c. 1/Key_Setup_Latency_feedback - inverse of the key setup latency for feedback modes.
d. 1/Key_Scheduling_Area_feedback - inverse of the area used by the key scheduling unit

The encryption and decryption key setup latencies are particularly important in applications where only several blocks of data are encrypted between two consecutive key changes. IPSec and ATM, with small sizes of packets, and consecutive packets encrypted using different keys, are two widespread protocols in which the key setup latencies may play a very important role. From the point of view of the key agility, it is sufficient to divide AES candidates into three major groups. In the first group, both encryption and decryption key setup take less time than processing (encryption or decryption, accordingly) of a single block of data. In the second group of ciphers, the encryption key setup is shorter, but the decryption key setup is longer, than the time required to process a single block of data. In the third group, both key setup latencies are greater than the encryption/decryption latencies. Ciphers from the same group should be rated similarly, even if the absolute values of their latencies differ substantially. Unfortunately, even the division of the AES candidates into these three groups has not been confirmed by any two independent groups. For example, RC6 belongs to the first group according to the USC report, and to the third group according to the NSA report. Because of this uncertainty, and a large number of applications in which the key setup latency does not significantly affect the cipher performance, the weight of this parameter w_{kf}, should be smaller than w_{tf} and w_{taf}.

e. Throughput_non-feedback - Throughput for non-feedback modes
f. Encryption Area_non-feedback - Area of the encryption/decryption unit for non-feedback modes
g. Latency_non-feedback - Latency for non-feedback modes

Because in non-feedback modes, area can be directly traded for encryption throughput, we believe that the throughput and area should have the same or similar weights, $w_{tnf} = w_{anf}$. Encryption latency is only indirectly related to the encryption throughput and circuit area as shown in Fig. 23 and Fig. 24. Additionally, in many applications, the time necessary to encrypt a long string of data is a function of the encryption throughput and is almost independent of the encryption latency. As a result, the weight of the latency w_{lnf} should be smaller than the weights of the throughput and area.

The value of the sum

$$w_{tnf} \cdot \text{Throughput\_non-feedback} + w_{anf} \cdot \text{Area\_non-feedback} + w_{lnf} \cdot \text{Latency\_non-feedback}$$

can be computed independently based on the results of three research groups presented in Section 7.2. Unfortunately, it should be expected that these three sets of results will lead to a substantially different ranking of the AES candidates.

h. 1/Key_Setup_Latency_non-feedback - inverse of the key setup latency for non-feedback modes

i. 1/Key_Scheduling_Area_non_feedback - inverse of the area used by the key scheduling unit for non-feedback modes

The results of the key setup latency for non-feedback modes were reported by only one group [WBRF00b], and were never confirmed by any other team. As a result, we would suggest to assign a relatively small weight to this parameter.

10. Summary

The comparison of the AES candidates should be based on measures that are important from the point of view of practical applications and have been studied independently by several research groups.

Out of two major types of cipher modes, feedback modes, such as CBC and CFB, are predominantly used in today's secure network protocols. These modes are required among the other in the three most popular Internet security protocols: SSL, S-MIME, and IPSec. In hardware implementations, feedback modes impose a serious limitation on the internal architecture of the encryption unit by preventing applying parallel processing and pipelining to encrypting blocks of a single message. This limitation does not apply to decryption, and can be relaxed for encryption if several messages or packets encrypted using different initialization vectors are available for processing at the same time. Several architectures have been proposed to implement AES candidates in feedback modes. Among them, the basic iterative architecture guarantees the highest throughput to area ratio and close to the highest throughput for encryption, assuming that only one message or packet is available at a time. In this architecture, decryption can be typically performed with the same throughput as encryption.

Five research groups reported results of their practical implementations of the basic iterative architecture for at least three AES candidates (see Table IV). These groups used two families of FPGA devices, Xilinx Virtex and Altera FLEX, and a semi-custom ASIC technology based on MOSIS library of standard cells. The primary optimization criterion used by all groups was speed. For comparisons in terms of the encryption throughput, the ranking of the AES candidates was almost identical for all five groups and three different implementation approaches (see Figs. 9-14). Serpent and Rijndael were at least twice as fast as all remaining candidates; Twofish and RC6 offered similar medium speed, and Mars was the slowest in all rankings. Serpent was slightly faster than Rijndael if architecture Serpent I8 (with eight regular cipher rounds treated as a single implementation round) was used, and significantly slower if architecture Serpent I1 (with an implementation round the same as a regular cipher round) was applied (see Fig. 7). Twofish was faster than RC6 in all rankings by factors ranging from 1% to 54%.

The second most important factor used for comparison was the throughput vs. area characteristics of all candidates. This characteristics is particularly important for applications with the restricted circuit area. A very good agreement between the two-dimensional throughput vs. area diagrams was demonstrated for the FPGA and ASIC implementations selected based on the best throughput to area ratio (see Figs. 17 and 18). Based on these diagrams, AES candidates can be divided into three major groups. Rijndael and Serpent I8 offer superior speed but at the same time require the largest amount of area; Twofish, RC6, and Serpent I1 are the most area-efficient, but provide

only medium speed; Mars is the least efficient of all candidates in terms of both speed and area. The areas required by all ciphers can be reduced using resource sharing at the cost of at least proportional decrease in the cipher speed (see Fig. 6). At the same time, the speed of Twofish and RC6 cannot be easily improved even at the cost of a substantial increase in the circuit area (see Fig. 5).

Key scheduling for the basic iterative architecture was implemented by only two groups, using Xilinx Virtex FPGAs and semi-custom ASICs, respectively (see Table IX). The agreement between the results of both groups was not very good in either absolute values or the number of clock cycles of the key setup latency (see Fig. 34 and Table X). These differences can be partially explained by a slightly different definition of the key setup latency used by both groups (see Section 8.1), and simplifications in the key scheduling of Mars introduced by one of the groups. Nevertheless, despite these differences the general conclusions reached by both groups were similar. The candidates were divided into two groups. For the first group, which includes Rijndael, Serpent, and Twofish, the key setup latency is only a fraction of the time required to encrypt a single block of data. As a result, the key can be changed on the fly, without affecting the cipher throughput. For the second group of ciphers, which includes Mars, the key setup latency is several times bigger than the time necessary to encrypt a single block of data. As a result, switching to the new key either introduces an extra delay or requires an additional circuitry to store the new internal keys at the time when the previous keys are still in use. The only major inconsistency between the results of both research teams is the classification of RC6 to one of the defined above groups, which is clearly different in reports of both teams. Additionally, only one group reported results regarding the key setup latency for decryption.

Although non-feedback cipher modes, such as ECB and counter mode, are relatively rarely used in today's security protocols, the importance of these modes and the corresponding hardware architectures is likely to increase in the near future. The counter mode has already been proposed to be used in secure protocols for high-speed networks, such as ATM networks. The interleaved modes, such as interleaved CBC, seem to offer the same security as feedback modes, and can be implemented using pipelined architectures, identical to those used for non-feedback modes. The pipelined implementations of non-feedback modes can be used for *decryption* in all regular feedback modes. Additionally, the parallel and pipelined hardware architectures suitable for non-feedback modes can be relatively easily extended to handle *encryption* in regular feedback modes assuming that several packets using different initialization vectors are available for processing at the same time. This condition is fulfilled for example in the implementations of IPSec, one of the most widely implemented Internet security protocols.

Three research groups implemented pipelined architectures suitable for non-feedback cipher modes (see Table VII). Each of these groups implemented at least four ciphers. The results obtained by each of these groups were considerably different (see Figs. 26-33). According to the GMU group, all four implemented cipher (Mars not included in the comparison) offered approximately the same throughput, and differed only in terms of the circuit area and encryption latency (see Fig. 26). Serpent, Twofish, and Rijndael required approximately the same area; RC6 had twice as high area requirements (Fig. 30). According to the NSA group, Serpent was the fastest of all ciphers, followed closely

by Rijndael. Both these ciphers outperformed three remaining candidates in terms of the encryption throughput by factors of 3.5 and 2.5, respectively (Fig. 27). The area requirements for Mars and Serpent were twice as high as for remaining three candidates. According to the WPI group, which assumed the same area for each cipher, Serpent achieved over twice as high throughput as Rijndael, RC6, and Twofish (Mars was not implemented) (see Fig. 28).

The main reason for so large inconsistencies among the three aforementioned reports was that each research group chose to implement a different pipelined architecture. The main difference among these architectures was the number and location of the pipeline registers. All three designs have their advantages and disadvantages as described in Section 7.1.2. Nevertheless, the specific architectures and assumptions selected by two research groups (NSA and WPI) seem to favor AES candidates with the large number of simple rounds, namely Serpent and Rijndael (see sections 7.1.2 and 7.2 for details). Taking into account these large differences, and the possible strong influence of assumptions on results of analyses, we recommend caution and encourage further discussion regarding the efficiency of the AES candidates in the architectures suitable for non-feedback cipher modes. Additionally, it should be pointed out that there is very little known about the parameters of the key scheduling unit in non-feedback cipher modes. The only results in this area come from the NSA group.

Each of the measures taken into account during the overall analysis of the hardware efficiency of the AES candidates should be assigned its specific weight. A value of each weight should be a function of the importance of the given measure in the majority of today's and future applications. It should also depend on the amount and consistency of the evidence supporting the ranking of the AES candidates according to the given measure.

In Section 9, we proposed a set of performance measures to be taken into account during the evaluation, and gave our recommendation regarding the relative values of particular weights. We left choosing exact values of each weight and the values of performance measures for each candidate to the discretion of the reader. Nevertheless, we can predict that if these values are chosen anywhere close to our guidelines the ranking of the AES candidates is likely to be as follows: Rijndael and Serpent close first, followed in order by Twofish, RC6, and Mars.

References

[AES] "Advanced Encryption Standard Development Effort," http://www.nist.gov/aes.

[AES3] "Third AES Candidate Conference," http://csrc.nist.gov/encryption/aes/round2/conf3/aes3conf.htm.

[BCD+98] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas, L. O'Connor, M. Peyravian, D. Safford, and N. Zunic, "Mars - A Candidate Cipher for AES," NIST AES Proposal, June 1998, available at [AES].

[BoCz00] P. Bora and T. Czajka, "Implementation of the Serpent Algorithm Using Altera FPGA Devices," Public Comments on AES Candidate Algorithms - Round 2, available at http://csrc.nist.gov/encryption/aes/round2/pubcmnts.htm.

[DPR00a] A. Dandalis, V. K. Prasanna, J. D. Rolim, "A Comparative Study of Performance of AES Final Candidates Using FPGAs," *Rump Session of the 3rd Advanced Encryption Standard (AES) Candidate Conference*, New York, April 13-14, 2000, text available at [AES3].

[DPR00b] A. Dandalis, V. K. Prasanna, J. D. Rolim, "A Comparative Study of Performance of AES Final Candidates Using FPGAs," *Proc. Cryptographic Hardware and Embedded Systems Workshop*, CHES 2000, Worcester, MA, Aug 17-18, 2000.

[ElPa00] A.J. Elbirt and C. Paar, "An FPGA Implementation and Performance Evaluation of the Serpent Block Cipher," *Eighth ACM International Symposium on Field-Programmable Gate Arrays*, Monterey, California, February 10-11, 2000. Preprint available at http://ece.wpi.edu/Research/crypt/publications/index.html.

[EYCP00] A. J. Elbirt, W. Yip, B. Chetwynd, C. Paar, "An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists," *Proc. 3rd Advanced Encryption Standard (AES) Candidate Conference*, New York, April 13-14, 2000.

[Fis00] V. Fischer, "Realization of the Round 2 AES Candidates using Altera FPGA," submitted for *3rd Advanced Encryption Standard (AES) Candidate Conference*, New York, April 13-14, 2000; available at http://csrc.nist.gov/encryption/aes/round2/conf3/aes3papers.html

[GaCh00a] K. Gaj and P. Chodowiec, "Comparison of the hardware performance of the AES candidates using reconfigurable hardware," *Proc. 3rd Advanced Encryption Standard (AES) Candidate Conference*, New York, April 13-14, 2000.

[GaCh00b] K. Gaj and P. Chodowiec, "Fast implementation and fair comparison of the final candidates for Advanced Encryption Standard using Field Programmable Gate Arrays," available at http://ece.gmu.edu/crypto/publications.htm

[IKM00] T. Ichikawa, T. Kasuya, M. Matsui, "Hardware Evaluation of the AES Finalists," *Proc. 3rd Advanced Encryption Standard (AES) Candidate Conference*, New York, April 13-14, 2000.

[Mro00] P. Mroczkowski, "Implementation of the block cipher Rijndael using Altera FPGA," Public Comments on AES Candidate Algorithms - Round 2, available at http://csrc.nist.gov/encryption/aes/round2/pubcmnts.htm.

[SKBC00] *Symmetric Key Block Cipher Modes of Operation Workshop*, Oct. 20, 2000, Baltimore, MD.

[Smi00b] M. Smid, "A Strategy for Analyzing Public Comments and Preparing the Round 2 Status Report," available at http://csrc.nist.gov/encryption/aes/round2/pubcmnts.htm

[Smi00b] M. Smid, "AES Issues," available at http://csrc.nist.gov/encryption/aes/round2/pubcmnts.htm

[WBRF00a] B. Weeks, M. Bean, T. Rozylowicz, C. Ficke, "Hardware performance simulations of Round 2 Advanced Encryption Standard algorithms," *Proc. 3rd Advanced Encryption Standard (AES) Candidate Conference*, New York, April 13-14, 2000.

[WBRF00b] B. Weeks, M. Bean, T. Rozylowicz, C. Ficke, "Hardware performance simulations of Round 2 Advanced Encryption Standard algorithms," NSA's final report on hardware evaluations published May 15, 2000, available at http://csrc.nist.gov/encryption/aes/round2/r2anlsys.htm#NSA

[WBRF00c] B. Weeks, M. Bean, T. Rozylowicz, C. Ficke, "Hardware performance simulations of Round 2 Advanced Encryption Standard algorithms," slides from the presentation at the *3rd Advanced Encryption Standard (AES) Candidate Conference*, New York, April 13-14, 2000, available at http://csrc.nist.gov/encryption/aes/round2/conf3/aes3agenda.html.

[WeWa00] N. Weaver, J. Wawrzynek, "A comparison of the AES candidates amenability to FPGA Implementation," Proc. 3rd Advanced Encryption Standard (AES) Candidate Conference, New York, April 13-14, 2000.

[WPRW99] D. C. Wilcox, L. G. Pierson, P. J. Robertson, E. L. Witzke, and K. Gass, "A DES ASIC suitable for network encryption at 10 Gbps and beyond," Proc. 1st Int. Workshop on Cryptographic Hardware and Embedded Systems, CHES'99.

[Xili99] Xilinx, Inc., "Virtex 2.5 V Field Programmable Gate Arrays," available at www.xilinx.com.