SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2014; **7**:2225–2236
Published online 12 October 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.651

SPECIAL ISSUE PAPER

FPGA implementation of AES algorithm for high throughput using folded parallel architecture

K. Rahimunnisa[1]*, P. Karthigaikumar[1]*, Soumiya Rasheed[1], J. Jayakumar[2] and S. SureshKumar[3]

[1] Department of Electronics and Communication Engineering, Karunya University, Coimbatore, India
[2] Department of Electrical and Electronics Engineering, Karunya University, Coimbatore, India
[3] Department of Electronics and Communication Engineering, Dr. NGP Institute of Technology, Coimbatore, India

ABSTRACT

This paper presents high throughput architecture for the hardware implementation of Advanced Encryption Standard algorithm. Advanced Encryption Standard is the industry standard crypto algorithm for encryption and is used for protecting secret information. This work is mainly targeted for low-cost embedded applications. This paper introduces parallel operation in the folded architecture to obtain better throughput. The design is coded in Very High-speed Integrated Circuit Hardware Description Language. Timing simulation is performed to verify the functionality of the designed circuit. The proposed structure is implemented in Virtex-6 XC6VLX75T FPGA device. This work gives a high throughput of 37.1 Gb/s with a maximum frequency of 505.5 MHz, which is 20% higher than the maximum throughput reported in the literature. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS
AES; cryptography; Field Programmable Gate Array (FPGA); throughput

***Correspondence**
K. Rahimunnisa and P. Karthigaikumar, Department of Electronics and Communication Engineering, Karunya University, Coimbatore, India.
E-mail: krahimunnisa@gmail.com; karthi_kumar_p@rediffmail.com

1. INTRODUCTION

Secure transmission of information is very important in the field of communication. Nowadays, the increased use of computer and communication systems by industry has increased the risk of theft of information. Information security will be the primary concern of every communication system. Transmission of sensitive data over the communication channel has emphasized the need for fast and secure digital communication networks to achieve the requirements such as secrecy, integrity, and nonrepudiation of exchanged information, so every network should be equipped with a system to fulfill the demands of user for secure data transmission over insecure channels.

Cryptography is probably the most important aspect of communications security, and it is a basic building block of computer security. It enables us to store sensitive information or transmit it across insecure networks, so that unauthorized persons cannot read it. The urgency of secure exchange of digital data resulted in large quantities of different crypto algorithms that are evaluated on the basis of throughput, speed of operation, and area requirements. Every algorithm performs various substitutions and transformation on original data by using a key. Key is a value independent of original data. Depending upon the key, there are mainly two types of cryptographic algorithms: symmetric and asymmetric algorithms. Symmetric systems such as Data Encryption Standard (DES), 3DES, and AES use an identical key for to both encrypt the message text and decrypt the cipher text (encrypted plaintext). Asymmetric systems such as Rivest–Shamir–Adelman and Elliptic Curve Cryptosystem use different keys for encryption and decryption. Symmetric cryptosystems are more suitable to encrypt large amount of data with high speed.

The National Institute of Standards and Technology (NIST) invited proposals for Advanced Encryption Standard algorithm to replace the old DES algorithm in September 12 of 1997. Many algorithms were presented originally with researches from 12 different nations. On October 2. 2000, NIST [1] has announced the Rijndael algorithm to be the best in security, performance, efficiency, implementation ability, and flexibility. From then, the AES has been used in many applications from internet routers, virtual private networks, mobile phone applications, and electronic financial transactions.

The AES algorithm can be efficiently implemented both in hardware and software. Software implementations take the smallest resources, but they offer only limited physical security. Because of the growing requirements for high-speed, high-volume secure communications combined with physical security, hardware implementation of cryptographic algorithms becomes essential. The proposed AES is implemented on Field Programmable Gate Array (FPGA) technology. Hardware implementation is useful in wireless security such as military communication and mobile telephony where there is a greater emphasis on the speed of communication.

2. RELATED WORKS

After the approval of AES by NIST in 2001, most of the research was towards the efficient hardware implementations of AES [2] algorithm. The earlier designs mainly focused on intensively pipelined high-speed implementations, and the more recent works have concentrated on compact and low-power architectures considering low-cost devices and feedback modes of operation. Most of the recent researches now concentrate on increasing the speed of operation. Few of these are discussed in the following, and many of their results are compared in the Section 6.

A compact architecture for AES implementation is proposed by Chodowiec *et al*. [3]. A folded register concept is presented to reduce the area. Also, a dual port RAM-based and shift register-based implementation is specified, but the speed of operation is very slow compared with other structures. The throughput of 1.3 Gb/s can be achieved by using Spartan II device.

Granado-Criado *et al*. [4] introduced a new methodology to implement the AES algorithm by using partial and dynamic reconfiguration. It deals with the pipelined and parallel implementation. The achieved throughput is 24.922 Gb/s.

A low area and low-power architecture is introduced in [5] by Hamalainen *et al*. The high-level architecture of this AES encryption core is depicted in this paper. All the connections and registers in this design are 8 bits wide. The ShiftRows operation and partial storage of State are combined into the byte permutation unit, which helps to reduce the area. The MixColumns multiplier performs a complete operation in 16 cycles in parallel with the rest of the operations of the AES core to reduce the power consumption. The specified architecture that was implemented achieved a throughput of 0.121 Gb/s.

Alaoui Ismaili *et al*. [6] proposed a self partial and dynamic reconfiguration implementation for AES algorithm. This implementation consists of an FPGA that is partially reconfigured at run-time to provide countermeasures against physical attacks. The static part is only configured upon system reset. By using reconfiguration concept, performance of the implemented circuit can be increased. The algorithm was implemented in Virtex-II device with the throughput of 1.6 Gb/s.

A compact and efficient encryption/decryption module architecture is presented in [7] by Rouvroy *et al*. This structure is used for small embedded applications. Because the key schedule is carried out with precomputation, this part does not work simultaneously with the encryption/decryption process. It is therefore possible to share resources between both circuits. This design provided a throughput of 0.358 Gb/s for the Virtex-II device.

An efficient implementation of AES structure in reconfigurable hardware is proposed by Standaert *et al*. [8]. It proposed heuristics to evaluate hardware efficiency at different steps of the design process and also defined an optimal pipeline that takes place and route constraints into account. Resulting circuits significantly improved the throughput up to 18.5 Gb/s.

A fully pipelined AES encryption architecture was introduced by Hodjat *et al*. [9]. By using loop unrolling and inner-round and outer-round pipelining techniques, a maximum throughput of 21.54 Gbits/s was achieved in Virtex-II Pro FPGA.

Fan *et al*. [10] proposed a high throughput and fully pipelined architecture for AES implementation. An efficient low-cost AddRoundKey architecture is used for real-time key generations. This design achieved an increased throughput of 28.4 Gb/s.

A hardware-efficient design increasing throughput for the AES algorithm using a high-speed parallel pipelined architecture was proposed by Yoo *et al*. [11]. By using an efficient inter-round and intraround pipeline design, this implementation achieved a high throughput of 29.77 Gb/s.

Zhang [12] proposed high-speed VLSI architecture for AES algorithm. The design employs combinational logic for implementing S-box. Subpipelining concept was introduced to increase the speed of operation. Furthermore, composite field arithmetic is employed to reduce the area requirements. The architecture implemented in Xilinx XCV1000 e-8 device with a throughput of 21.566 Gb/s.

Liberatori *et al*. introduced in [13] an architecture for low area 128-bit AES cipher. This paper presents an 8-bit FPGA implementation of the 128-bit block and 128-bit key AES cipher. There is no key scheduling unit in the structure. A memory is used for storing the internal keys, and the circuitry necessary to distribute these keys is included in the encryption/decryption unit. The circuit uses one Embedded Array Block for key and subkey storage. The FPGA family selected for this implementation is Altera Flex 10K, in particular EPF10K20. The reported throughput is 1.1 Gb/s for this architecture.

Finite state machine-based AES implementation is proposed in [14] by Chittu *et al*. This implementation supports only 128-bit key size for 128-bit data. This circuit has the capability to handle encryption/decryption and fitted in one FPGA taking approximately 84% of the area. The experimental procedure demonstrated that the total encryption and decryption throughput of 0.739 Gb/s can be achieved by using Virtex-II XC2V100-4 device.

3. OVERVIEW OF AES STRUCTURE

The AES Algorithm is a symmetric key cipher, in which both the sender and the receiver use a single key for encryption and decryption. The data block length is fixed to be 128 bits, whereas the key length can be 128, 192, or 256 bits. In addition, the AES algorithm is an iterative algorithm. Each iteration can be called a round, and the total number of rounds is dependent on the key length. For each round, 128 bit input data and 128/192/256 bit key is required. The number of bits in key will determine the number of rounds required for encryption. For the AES algorithm, the length of the Cipher Key, *K*, can be 128, 192, or 256 bits. The key length is represented by $Nk = 4$, 6, or 8, which reflects the number of 32-bit words (number of columns) in the Cipher Key. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by *Nr*, where $Nr = 10$ when $Nk = 4$, $Nr = 12$ when $Nk = 6$, and $Nr = 14$ when $Nk = 8$ [1]. This means that it needs four words of 32 bit each, for each round, so the input key has to be expanded to the required number of words for each round of operation. The output of each round serves as input of next stage. In AES system, same secret key is used for both encryption and decryption, so it provides simplicity in design. For this work, 128 bit key is chosen, which requires 10 rounds of encryption.

The 128-bit data block is divided into 16 bytes. These bytes are mapped to a 4×4 array called the State, and all the internal operations of the AES algorithm are performed on the State. In the encryption of the AES algorithm, each round except the final round consists of four transformations:

i. SubBytes: operates in each byte of the State independently. Each byte is substituted by corresponding byte in the S-box (Substitution box).
   a. S-box is one of the basic components of any symmetric key algorithm, which exhibits the property of confusion. This property is provided to increase the difficulty in finding the key from the known cipher text. S-box takes *m* inputs and transforms them to give *n* bits at the output. Fixed S-boxes are used in AES algorithm, which are designed using multiplicative inverse over GF (2^8) and combining the inverse function with an invertible affine transformation. These properties make it efficient over cryptanalysis by providing nonlinear properties.
ii. ShiftRow (Shift): cyclically shifts the rows of the State over different offsets.
iii. MixColumn (MixCol): in this operation, the column of the State are considered as polynomials over GF (2^8) and are multiplied with a fixed polynomial. The MixColumn component does not operate in the last round of the encryption algorithm.
iv. AddRoundKey (Add-R-Key): involves bit-wise XOR operation with key.

The transformations in the decryption process perform the inverse of the corresponding transformations in the encryption process. The decryption structure can be derived by inverting the encryption structure directly. Figure 1 explains the combined encryption and decryption operation.

To have compact size and high throughput, a folded register concept with parallel architecture is proposed in this paper and is discussed in Sections 4 and 5.

4. IMPLEMENTATION OF A FOLDED REGISTER CONCEPT IN AES

The need for secure electronic data exchange will become increasingly more important in the years to come. Therefore, the AES must be extended to low-end customer products, such as PDAs, wireless devices, and many other embedded applications. To achieve this goal, the AES implementations must also become inexpensive.

Most of the low-end applications do not require high encryption speeds. Implementing security protocols, even for those low network speeds, significantly increases the requirements for computational power. For small embedded applications and other security purposes, the low area implementation of algorithm is very necessary. All the devices such as Spartan and Virtex are provided with specified area constraints and other parameters. To use these devices, it is necessary to meet these parameters properly. By reducing the required area, the cost of implementation can be reduced and can be used for many commercial applications that are affordable by low-end customers.

The basic structure of AES implementation requires large resources in terms of registers, slices, LUTs, BRAMs, and so forth. Because its resource utilization exceeds the available resources, the cost effective implementation is not possible, so it is necessary to design a structure that consumes lesser area. By introducing a folded concept [3] in the basic architecture [2], it is possible to obtain the significant reduction in area.

To create a folded architecture, arrange the data bytes into rows as shown in Figure 2 [3]. This data arrangement is consistent with a state that is used in the basic architecture. The operation of folded register architecture includes the following steps.

Step 1: read input bytes: 0, 5, A, F; execute SubBytes, MixColumn, and AddRoundKey on them; write results to the output at the locations: 0, 1, 2, 3.
Step 2: repeat the aforementioned operations for input bytes: 4, 9, E, 3; write results at output locations: 4, 5, 6, 7.
Step 3: repeat the aforementioned operations for input bytes: 8, D, 2, 7; write results at output locations: 8, 9, A, B.
Step 4: repeat the aforementioned operations for input bytes: C, 1, 6, B; write results at output locations: C, D, E, F.

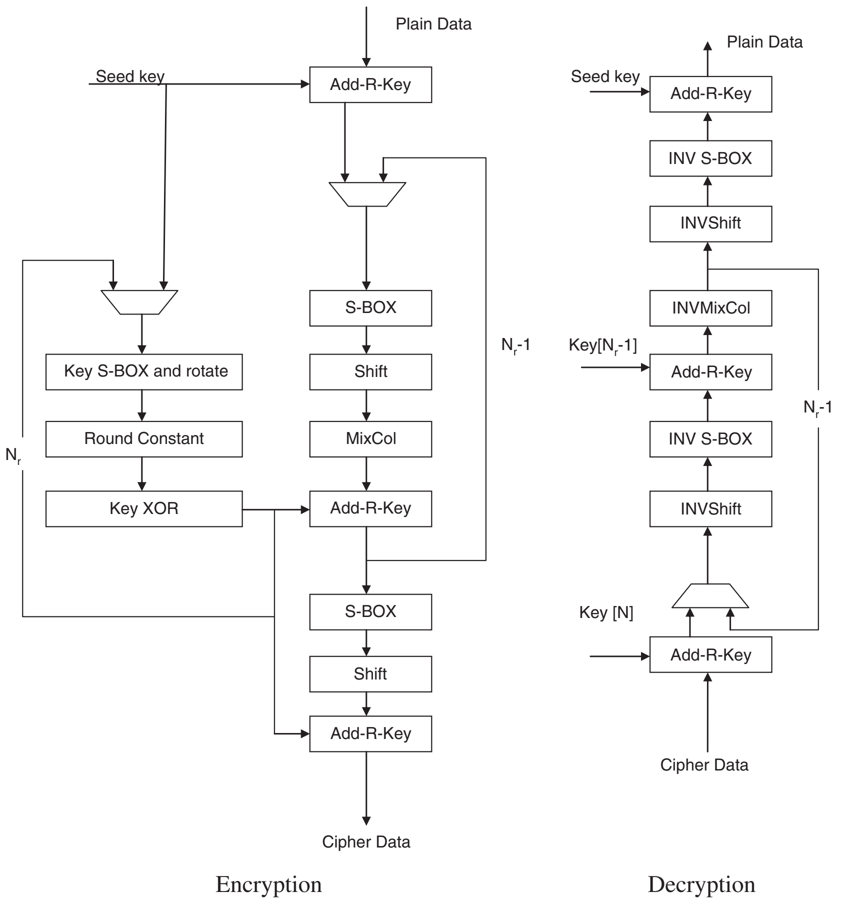


Figure 1. Combined encryption/decryption part.

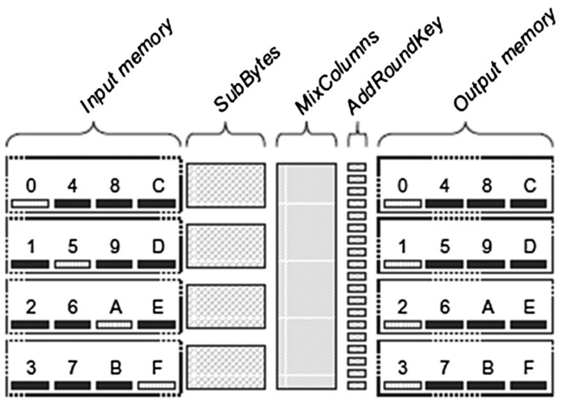


Figure 2. Folded register.

The entire AES round was executed including ShiftRows operation in those four steps. At each step, only one byte is read from each input row, and one byte is written to each output row. Each row can be viewed as an addressable 8-bit memory.

In basic AES architecture, each round is composed of sixteen 8-bit S-boxes, computing SubBytes, and four 32-bit MixColumns operations, working independent of each other. The only operation that spans throughout the entire 128-bit block is ShiftRows, but in the case of folded architecture, the ShiftRow operation is eliminated by orderly selecting the input data bytes from the state matrix. The order of selection is such a way that the data bytes are automatically shifted in correct order.

In this architecture, only 4 bytes are processed at a time, so each round includes four 8-bit SubByte operations, one 32-bit MixColumn operation, and four 8-bit AddRoundKey operations. That means the entire basic structure is folded by 4. This design enables reduction in area considerably, but the throughput of the design is poorer than the basic structure as it takes more time to complete one round of operation. Most of the low-end applications do not require high-speed operation, but it must be cost effective and should be implemented in less area; hence, this structure with folded architecture can be implemented on low-end devices with a particular application.

5. PARALLEL OPERATION

The folded architecture is suitable for reducing the area of implementation, but this leads to increase in the overall delay of the design and thus reducing the speed. Every real-time application needs high speed of operation and throughput. To obtain a tradeoff between speed and area constraints, it is necessary to incorporate parallel operation with the folded architecture.

Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel"). There are several different forms of parallel computing: bit level, instruction level, data, and task parallelism. Parallelism has been employed for many years, mainly in high-performance computing.

In folded architecture, only one word is processed at a time, that is, the operations for the second word starts only after the completion of the first word and so on. It takes more time for completing the entire operation. Consequently, the throughput of the design becomes very low, which is not suitable for real-world applications. The parallel computing [15–19] concept solves this problem. Instead of processing one word at a time, it performs the operations in parallel. Figure 3 explains the concept of parallel operation. That is, all 16 bytes (words 1–4) operations are simultaneously processed. Initially, word 1 is the input bytes 0, 5, A, F; word 2 is 4, 9, E, 3, word 3 is 8, D, 2, 7; and the last word is C, 1, 6, B, and these are used as inputs as shown in Section 4. These inputs are then processed in parallel as mentioned in folded architecture, to give the cipher text output of same length. The next set of inputs is processed, after the cipher text is obtained. This means that the time taken to process one word is now sufficient to process four words, at the additional cost of area used. This enables the proposed structure to improve the performance with desired throughput and area constraints. The speed of operation is well suited for both low-end applications and communication purposes. The LUT-based S-box and Inv S-box [20] are used in this architecture.

6. RESULTS AND DISCUSSION

6.1. Simulation results

ModelSimXE II 5.7 g simulator is used for simulating the designed structure. The entire design is synchronized with clock signals. The obtained waveform explains the flow of operation.

In the folded architecture, the entire 128 bit input data is divided into four equal words, that is, 32 bits each, and performs all the operations sequentially. This structure is not so much efficient because it provides less speed than the basic structure, so parallel computing concept is introduced, along with the folded architecture to increase the speed of operation. In this, four input words are processed at same time, and four output words are obtained simultaneously.

The simulated waveform for combined encryption/decryption is given in Figure 4. A control signal mode can combine the encryption and decryption part that gives a flexible implementation. Also, the control signals rst and ce_i are used for port mapping of key expansion unit. Because AES is a symmetric algorithm, same key is used for both encryption and decryption, so only one key expansion unit is necessary for the entire structure. This reduces the area occupied considerably.

Initially, the mode signal is set to "1" for performing the encryption operation. Besides four words of input signals, 128 bit key is also given as input. The key is expanded along with the encryption process and stored in a memory to use it for decryption. Eight clock cycles are required for the encryption. Then, set the mode signal as '0' for decryption operation. It takes only five clock cycles to complete the decryption operation, so the entire design takes only 13 clock cycles.

6.2. Synthesis results

The Xilinx Synthesis Technology synthesizer converts HDL (VHDL/Verilog) code into a gate-level netlist (represented

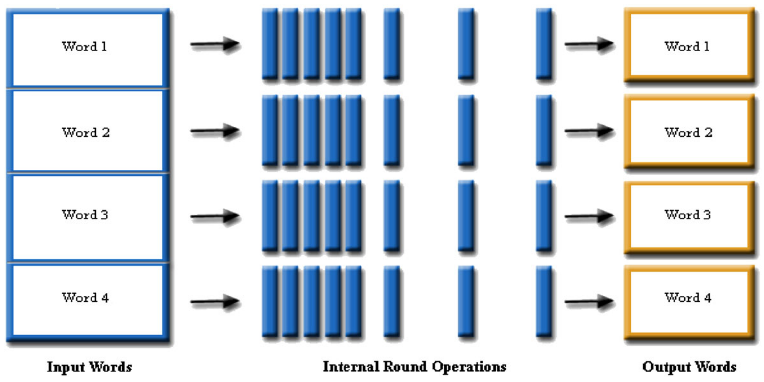
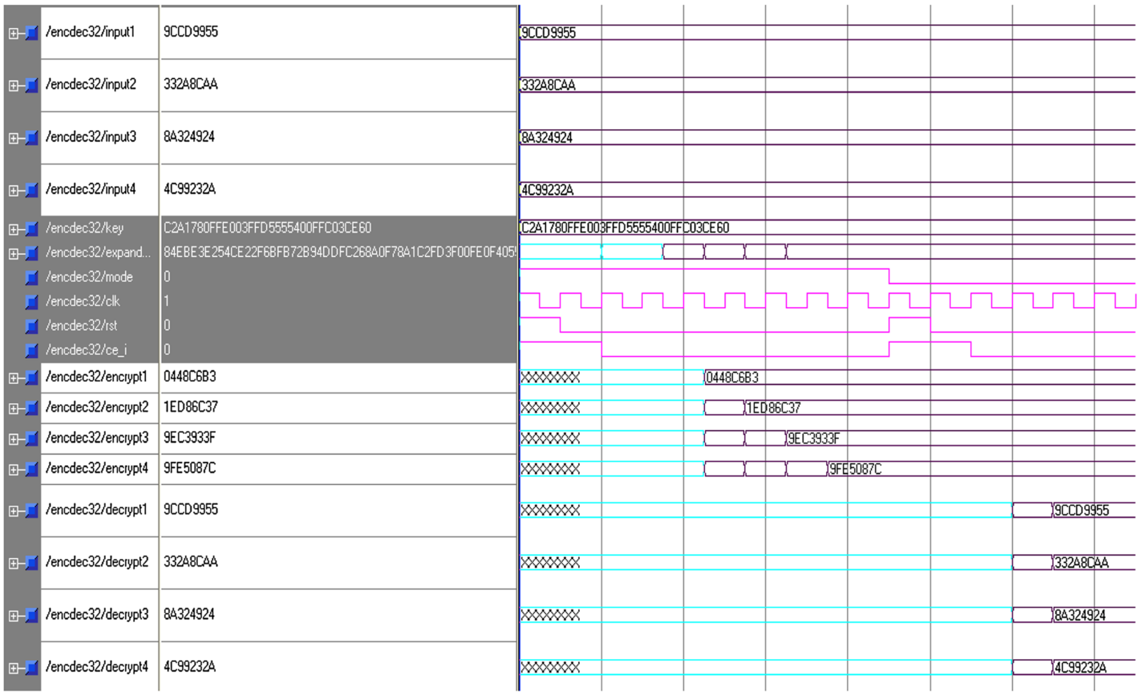


Figure 3. Parallel computing.



```
/encdec32/input1     9CCD9955
/encdec32/input2     332A8CAA
/encdec32/input3     8A324924
/encdec32/input4     4C99232A
/encdec32/key        C2A1780FFE003FFD5555400FFC03CE60
/encdec32/expand...  84EBE3E254CE22F6BFB72B94DDFC268A0F78A1C2FD3F00FE0F405
/encdec32/mode       0
/encdec32/clk        1
/encdec32/rst        0
/encdec32/ce_i       0
/encdec32/encrypt1   0448C6B3
/encdec32/encrypt2   1ED86C37
/encdec32/encrypt3   9EC3933F
/encdec32/encrypt4   9FE5087C
/encdec32/decrypt1   9CCD9955
/encdec32/decrypt2   332A8CAA
/encdec32/decrypt3   8A324924
/encdec32/decrypt4   4C99232A
```

Figure 4. Simulated waveform for combined encryption/decryption unit.

in the terms of the UNISIM component library, a Xilinx library containing basic primitives). By default, Xilinx ISE uses built-in synthesizer Xilinx Synthesis Technology. Other synthesizers can also be used. In this work, Xilinx 12.1 version is used to synthesis the entire design. The synthesis report allows viewing the results of the netlist generation synthesis process. This report contains the results from the synthesis run, including area and timing estimation. The targeted device is Virtex-6 XC6VLX75T as it is the recent technology.

Device utilization summary is available after the compiler completes the synthesizing step of the compilation process. This report contains a summary of the FPGA utilization as estimated during the synthesis of the FPGA. Timing report is available after the compile server completes the mapping step of the compilation process. This report contains a summary of the FPGA clocks, as estimated during the mapping of the FPGA. Table I shows the design summary for entire structure.

From Table I, it is observed that the proposed design occupies only 5% of available LUT in the FPGA. The maximum frequency reported is 505.5 MHz for the entire design. The throughput calculation is performed on the basis of the total time taken for the completion of the entire process. The throughput is determined by the following formula [21].

$$\text{Throughput} = \frac{128 \ * \ \text{Number of blocks/cycle}}{\text{Clock Period}}$$

128 indicates the block size of the input data.

Table I. Device utilization summary of folded parallel architecture of AES for Enc/Dec.

Number of slice registers	2056 out of 138,240 (1%)
Number of slice LUTs	3788 out of 138,240 (2%)
Number of fully used LUT–FF pairs	1792 out of 4,052 (44%)
Number of DSP48Es	2 out of 128 (1%)
BRAM	48
Minimum period	1.978 ns
Maximum frequency	505.5 MHz

Because the specified architecture provides higher speed than the basic structure, it gives a large throughput value. For the Virtex-6 XC6VLX75T FPGA device, the obtained throughput is 37.1 Gb/s, which is much higher than the throughput available in the literature so far.

The block diagram for the entire AES architecture is shown in Figure 5, and the detailed internal structure of AES is given in Figure 6.

6.3. Register Transfer Language schematic diagram

The detailed internal structure for the four transformation of the proposed AES algorithm is obtained from Register Transfer Language schematic representation and is shown in Figure 7.

6.4. Performance analysis

The comparison with existing architectures is very important for evaluating the efficiency of the proposed design. The comparison is performed on the basis of area requirements, throughput, operating speed, and so forth.

The related works show that different architectures are introduced for AES to obtain sufficient area requirements, throughput, and so forth, which are suitable for various applications. This paper introduces a folded architecture with parallel operation in AES. The combined encoding and decoding operations are performed with the help of a control signal.

In the basic structure of AES implemented here, the required area, in terms of slices and number of registers, is high. The reported throughput is also insufficient for real-time applications. A compact architecture concept is introduced in the basic structure for reducing the area of utilization. It helps to reduce the overall area by a factor of 4, but it adversely affects the speed of operation, that is, the throughput or performance becomes very poor. The parallel computing helps to resolve this problem by performing parallel operations for four words separately. It increases the speed of operation and throughput of the architecture without much utilization of available resources.

Table II shows the comparison of the proposed architecture with previous works. The highest throughput reported in the literature is 29.77 Gb/s with maximum frequency of 125.3 MHz. This proposed work gives higher throughput of 37.1 Gb/s with a maximum frequency of 505.5 MHz.

The proposed work gives 20% increase in throughput compared with the highest throughput reported in the literature [11,26] because of parallel processing. The area requirement is also reduced compared with basic architecture and less compared with architectures presented in [4,8–10,12,14,23–26,28,30–36,39] because of folded structure. We also compared the throughput, area of folded parallel architecture with folded structure, and it is reported in Table II. In Table II, it is proved that the throughput of the folded with parallel architecture is 46.52% higher than the folded structure at the cost of slight increase in area.

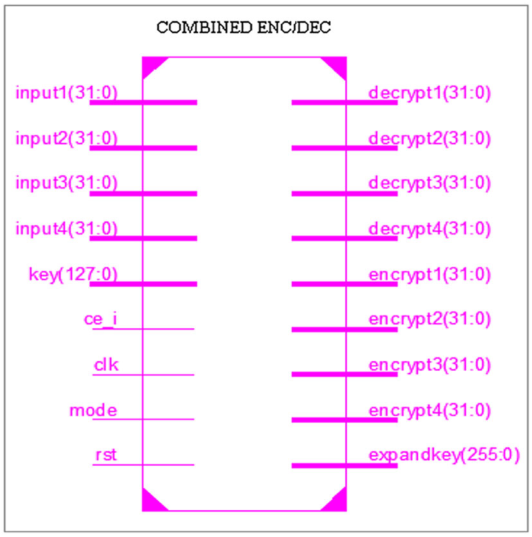


Figure 5. Block diagram of encryption/decryption unit.

7. CONCLUSION

This paper presents the FPGA implementation of low area, high throughput encryption and decryption by using AES algorithm. This design uses a 128-bit key for both encryption and decryption. This encoder makes use of 128-bit key, so the entire design includes the design of a key expansion unit. The folded concept is introduced in the basic architecture to reduce the area of utilization. Also, parallel operation is performed to obtain higher throughput than basic structure. The Virtex-6 XC6VLX75T is the targeted device for FPGA implementation.

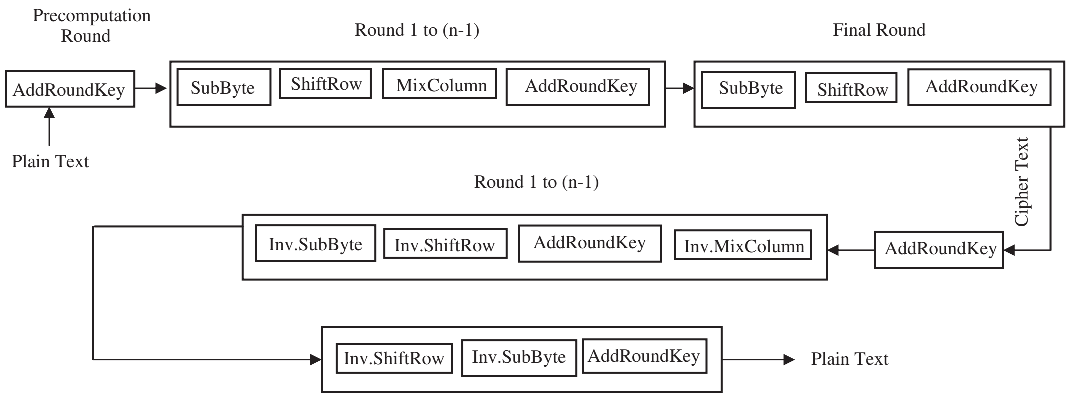
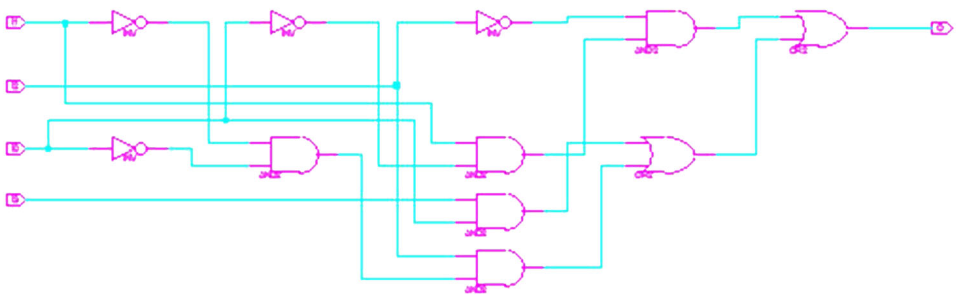
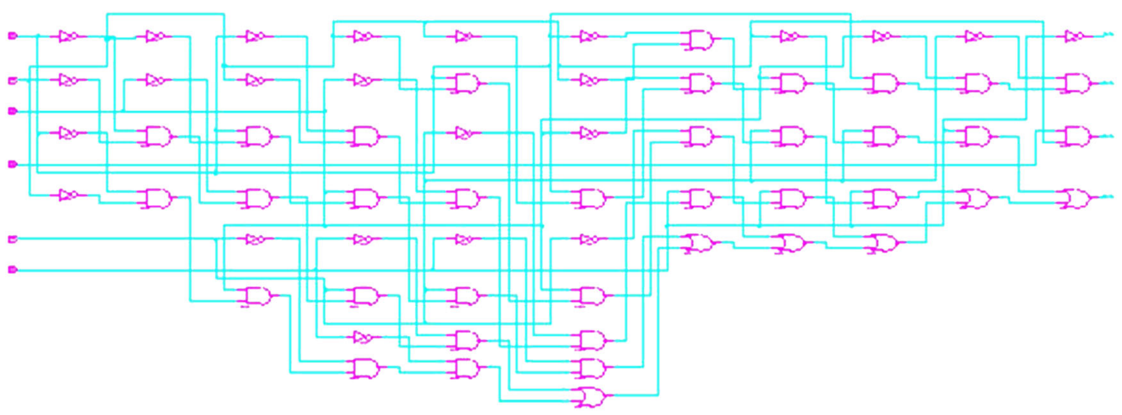
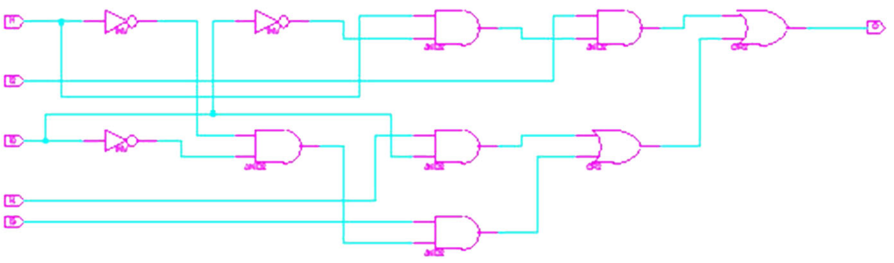
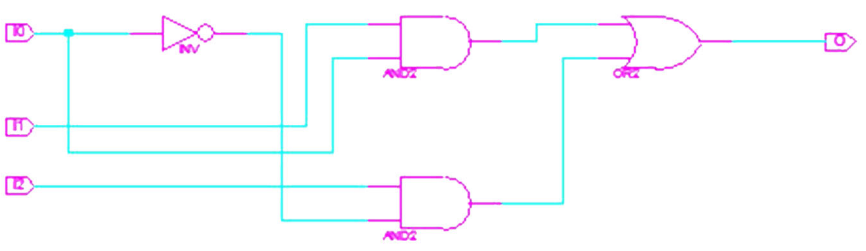


Figure 6. Internal structure of AES encryption/decryption unit.


(a) A Portion of Internal structure of AddRound operation


(b) A Portion of Internal structure of MixColumn operation


(c) A Portion of Internal structure of ShiftRow operation



(d) A Portion of Internal structure of SubByte operation

Figure 7. (a): A portion of internal structure of AddRound operation. (b): A portion of internal structure of MixColumn operation. (c): A portion of internal structure of ShiftRow operation. (d): A portion of internal structure of SubByte operation.

Table II. Comparison of proposed design with existing architectures.

Author	Architecture	FPGA device	Throughput (Gb/s)		Frequency (MHz)	Slices/available
			ENC	ENC/DEC		
This work	*Basic AES*	*Virtex-6 XC6VLX75T*	*11*	*16*	*505.5*	*2053/93120*
This work	*Folded structure*	*Virtex-6 XC6VLX75T*	*NA*	*25.32*	*505.5*	*1586/93120*
This work	*Folded with parallel*	*Virtex-6 XC6VLX75T*	*32*	*37.1*	*505.5*	*1664/93120*
Granado-Criado *et al.* [4]	Partial and dynamic reconfiguration	XC2V6000-6	24.922	NA	NA	3576/33792
Alaoui Ismaili *et al.* [6]	Self partial and dynamic reconfiguration	Spartan II–XC2s200E	0.016	NA	28.7	196/2353
Rais *et al.* [22]	Simple AES	Virtex-5 XC5VLX50	4.34	NA	339.09	399/7200
Fan *et al.* [10]	Fully pipelined	XC2V3000-6	NA	28.4	222.2	139357/14334
Bulens *et al.* [23]	LUT AES	Virtex-5	NA	4.1	350	800/1100
		Virtex-4		2.9	250	700/1220
		Spartan-3		1.7	150	1800/2150
Lemsitzer *et al.* [24]	GCM AES	Virtex-4	NA	17.9	140	18400/1220
Yoo *et al.* [11]	Interpipelining and intrapipelining	XC2VP70-7	29.77	NA	125.3	200/5177
Good *et al.* [25]	LUT-based AES	Virtex-E XCV2000E-8	NA	23.65	184.8	16693/19200
		Spartan-III XC3s2000-5	NA	25.10	196.1	17425/1280
Kotturi *et al.* [26]	Parallel pipelined AES	XC2VP70-7	29.77	NA	232.6	5408/5177
Aziz *et al.* [27]	CCM	Spartan II-XC3s200pq208-5	2.699	NA	231.97	481/120
Rouvroy *et al.* [7]	AES for embedded	Virtex-2	NA	0.358	123	146/256
Hodjat *et al.* [9]	Fully pipelined	XC2VP20-7	21.54	NA	157	5177/5177
Zhang *et al.* [12]	Subpipelining	XCV1000-8	21.57	NA	168.4	11022/1536
Zambreno *et al.* [28]	AES	Virtex-II XC2V4000	23.57	NA	184.1	16938/17021
Farhan *et al.* [29]	Simple AES	Xilinx x2v1000	1.45	NA	119	542/5120
Hodjat *et al.* [30]	Fully pipelined	XC2VP20-7	21.64	NA	169.1	9445/5177
Sever *et al.* [31]	Sequential	XC2V8000-5	NA	0.83	65	8378/46592
Wang *et al.* [32]	Sequential	XCV1000e-8	NA	0.463	75	5150/1536
Standaert *et al.* [8]	Pipelined	XCV3200e-8	18.5	NA	169	2257/8235
Chodowiec *et al.* [3]	Folded	Spartan II-XC2S30	NA	1.3	50	222/54
Jarvinen *et al.* [33]	Fully pipelined	Virtex-E XCV1000e-8	16.5	NA	129.2	11719/1536
Saggesse *et al.* [34]	Unrolling, tiling, and pipelining	Virtex-E XCV2000e-8	20.3	NA	158	5810/19200
Vu *et al.* [35]	CCM	Spartan II-2s200pq208-5	NA	NA	43.34	2035/120
Saqip *et al.* [36]	Sequential	XCV812	0.259	NA	22.41	2744/8544
Standaert *et al.* [37]	Reconfigurable AES	Virtex-5	NA	1.45	119	542/4800
Chittu *et al.* [14]	State machine-based AES	Virtex-II XC2V1000-4	NA	0.739	75	4325/5120
Sklavos *et al.* [17]	Sequential	XCV300BG432	NA	0.259	22	2358/384
Chitu *et al.* [38]	Sequential	XC2V1000-1	NA	0.739	75	4325/5120

(*Continues*)

Table II. (Continued)

Author	Architecture	FPGA device	Throughput (Gb/s)		Frequency (MHz)	Slices/available
			ENC	ENC/DEC		
Helion [39]	AES	Virtex-5	NA	4.1	350	349/4800
Manavski [40]	AES	NVIDIA GeForce8800GTXGPU	8.28	NA	NA	NA
Harrison *et al.* [41]	AES	NVIDIA GeForce7900GTGPU	0.87	NA	NA	NA
Wollinger *et al.* [42]	AES	TMS320C6x DSP	0.14	NA	NA	NA

From the analysis, it is proved that the proposed design is achieved 20% higher throughput than the maximum throughput reported in the literature and 46.52% higher than folded architecture.

REFERENCES

1. Federal Information Processing Standards Publication 197 November 26, Announcing the Advanced Encryption Standard (AES), 2001.
2. Mali M, Novak F, Biasizzo A. Hardware implementation of AES algorithm. *Journal of Electrical Engineering* 2005; **56**(9-10):265–269.
3. Chodowiec P, Gaj K. Very compact FPGA implementation of the AES algorithm. *Proc. of Cryptographic Hardware and Embedded System Workshop*, 2003; 319–333.
4. Granado-Criado JM. A new methodology to implement the AES algorithm using partial and dynamic reconfiguration. *Integration, the VLSI Journal* 2010; **43**(1):72–80.
5. Hamalainen P, Alho T, Hannikainen M. Design and implementation of low-area and low-power AES encryption hardware core. *9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools*, 2006; 577–583.
6. Alaoui Ismaili ZEA, Moussa A. Self-partial and dynamic reconfiguration implementation for AES using FPGA. *IJCSI International Journal of Computer Science Issues* 2009; **2**:33–40.
7. Rouvroy G, Standaert F-X. Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications. *Proceeding of International Conference on Information Technology: Coding and Computing*, Vol. 2, 2004; 583–587.
8. Standaert F-X, Rouvroy G. *Efficient implementation of Rijndael encryption in reconfigurable hardware: improvements and design tradeoffs*. Lecture Notes in Computer Science, Vol. 2779/2003, Cologne, Germany: Springer, 2003; 334–350.
9. Hodjat A, Verbauwhede I. A 21.54 Gbits/s fully pipelined AES processor on FPGA. *Proceeding of 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2004; 308–309.
10. Fan C-P, Hwang J-K. FPGA implementations of high throughput sequential and fully pipelined AES algorithm. *International Journal of Electrical Engineering* 2008; **15**(6):447–455.
11. Yoo SM, Kotturi D, Pan DW, Blizzard J. An AES crypto chip using a high-speed parallel pipelined architecture. *Microprocessors and Microsystems* 2005; **29**(7):317–326.

12. Zhang X, Parthi KK. High-speed VLSI architecture for AES algorithm. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 2004; **12**(9): 957–967.
13. Liberatori MC, Bonadero JC. AES-128 cipher minimum area, low cost FPGA implementation. *Lattin American Applied Research* 2007; **37**:71–77.
14. Chittu C, Chien D, Chien C, Verbauwhede I, Chang F. A hardware implementation in FPGA of the Rijndael algorithm. *Proceeding of 45th Midwest Symposium on Circuits and Systems*, Vol. 1, 2002; 507–510.
15. Fischer V, Drutarovsky M. Two methods of Rijndael implementation in reconfigurable hardware. *Proc. CHES 2001*, Paris, France, May 2001; 77–92.
16. Rodriguez Henriquez F, Saquib NA, Diaz-Perez A. 4.2 Gbits/sec single chip FPGA implementation of the AES algorithm. *Electronics Letters* 2003; **39**(15): 1115–1116.
17. Sklavos N, Koufopavlou O. Architectures and VLSI implementations of the AES proposal Rijndael. *IEEE Transactions on Computers* 2002; **51**(12): 1454–1459.
18. Hodjat A, Verbauwhede I. Area-throughput tradeoffs for fully pipelined 30 to 70 Gbits/s AES processors. *IEEE Transactions on Computers* 2006; **55**(4): 366–372.
19. Kuo H, Verbauwhede I. Architectural optimization for a 1.82 Gbits/sec VLSI implementation of the AES Rijndael algorithm. *Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems*, 2001; 51–64.
20. McLoone M, McCanny JV. Rijndael FPGA implementation utilizing look-up tables. *Journal of Signal Processing Systems* 2001; **34**(3):349–360.
21. Sirin Kumari M, Mahesh Kumar D, Rama Devi Y. High throughput-less area efficient FPGA Implementation of block cipher AES algorithm. *International Conference On Advanced Computing, Communication and Networks*, 2011; 484–489.
22. Rais MH, Qasim SM. Efficient hardware realization of advanced encryption standard algorithm using Virtex-5 FPGA. *IJCSNS International Journal of Computer Science and Network Security* 2009; **9**(9): 59–63.
23. Bulens P, Standaert F-X, Quisquater J-J, Pellegrin P, Rouvroy G. Implementation of the AES-128 on Virtex-5 FPGAs. In *AFRICACRYPT 2008*, Lecture Notes in Computer Science, Vol. 5023. Springer: Casablanca, Morocco, 2008; 16–26.
24. Lemsitzer S, Wolkerstorfer J, Felbert N, Braendli M. Multi-gigabit GCM-AES architecture optimized for FPGAs. *Workshop on Cryptographic Hardware and Embedded Systems — CHES'07*, 2007; 227–238.
25. Good, Benaissa M. AES on FPGA from the fastest to the smallest. In *Cryptographic Hardware and Embedded Systems, CHES 2005*. Springer: Edinburgh, UK, 2005; 427–440.
26. Kotturi D, Yoo SM, Blizzard J. AES crypto chip utilizing high-speed parallel pipelined architecture. IEEE International Symposium on Circuits and Systems, vol. 5, 2005; 4653–4656.
27. Aziz A, Ikram N. Hardware implementation of AES-CCM for robust secure wireless network. http://icsa.cs.up.ac.za/issa/2005/Proceedings/Research/069_Article.pdf.
28. Zambreno J, Nguyen D, Choudary A. Exploring area/delay tradeoffs in an AES FPGA implementation. In *Proceeding of Field-Programmable Logic and Applications, LNCS*. Springer, 2004; 575–585.
29. Farhan SM, Jamal H, Rahmatullah M. High data rate 8-bit crypto-processor. *ISSA Enabling tomorrow Conference*, Gallagher Estate, Midrand, South Africa, 2004.
30. Hodjat A, Verbauwhede I. A 21.54 Gbits/s fully pipelined AES processor on FPGA. *IEEE Symposium on Field-Programmable Custom Computing Machines*, 2004; 308–309.
31. Sever R, Ismailglu AN, Tekmen YC, Askar M, Okcan B. A high speed FPGA implementation of the Rijndael algorithm. *Euromicro Symposium on Digital System Design*, 2004; 358–362.
32. Wang SS, Ni WS. An efficient FPGA implementation of advanced encryption standard algorithm. *International Symposium on Circuits and Systems*, Vol. 2, 2004; II-597-600.
33. Jarvinen KU, Tommiska MT, Skytta JO. A fully pipelined memory less 17.8 Gbps AES-128 encryptor. *International Symposium on Field Programmable Gate Arrays*, 2003; 207–215.
34. Saggese GP, Mazzeo A, Mazzocca N, Strollo AGM. *An FPGA-based performance analysis of the unrolling, tiling, and pipelining of the AES algorithm.* In *FPL 2003*, LNCS Vol. 2778, Lisbon, Portugal, 2003; 292–302.
35. Vu K, Zier D. FPGA implementation AES for CCM mode encryption using Xilinx Spartan-II. ECE-679. Oregon State University, Spring, 2003.
36. Saqib NA, Henriquez FR, Perez AD. AES algorithm implementation – an efficient approach for sequential and pipeline architectures. *The Fourth Mexican International Conference on Computer Science*, Sep. 2003; 126–130.
37. Standaert F-X, Rouvroy G, Quisquater J-J, Legat J-D. A methodology to implement block ciphers in reconfigurable hardware and its application to fast and compact AES RIJNDAEL. *11th ACM International*

Symposium on Field-Programmable Gate Arrays FPGA'03, 2003; 216–224.

38. Chitu C, Chien D, Chien C, Verbauwhede I, Chang F. A hardware implementation in FPGA of the Rijndael algorithm. *The 45th Midwest Symposium on Circuits and Systems*, vol. 1, 2002; I-507-10.
39. Helion. www.heliontech.com
40. Manavski SA. CUDA compatible GPU as an efficient hardware accelerator for AES Cryptography. *IEEE International Conference on Signal Processing and Communications (ICSPC)*, 2007; 65–68.
41. Harrison O, Waldron J. AES encryption implementation and analysis on commodity graphics processing units. *Cryptographic Hardware and Embedded Systems (CHES)*, 2007; 209–226.
42. Wollinger TJ, Wang M, Guajardo J, Paar C. How well are high-end DSPs suited for the AES algorithms? *3rd Advanced Encryption Standard Candidate Conference*, 2000; 94–105.