Available online at www.ejournals.uofk.edu

Proceedings Vol. 6 pp. 10-19
7th Annual Conference for Postgraduate Studies and Scientific Research
Basic Sciences and Engineering Studies - University of Khartoum
Theme: Scientific Research and Innovation for Sustainable Development in Africa
20-23 February 2016**,** Friendship Hall, Khartoum, Sudan

An Efficient FPGA-based Design and Implementation of AES Algorithm

Mayada E. Mohamed[1], Sharief F. Babiker[2]

[1]Department of Ground station, Institute of Space Research and Aerospace(ISRA), Khartoum, Sudan (E-mail: mayadaelsir@hotmail.com)
[2]Department of Electrical and Electronic Engineering, University of Khartoum, Khartoum, Sudan (E-mail: shariefbabikir@gmail.com)

Abstract: This paper presents an FPGA based hardware design and implementation of a 128 bit AES encryption processor. Synthesis is achieved using Verilog code implemented on the FPGA. Two different architectures are presented, the basic iterative architecture which achieves low FPGA resources requirements, 347 slices and 10 BRAM and a maximum throughput is 1.3988Gbps. And the fully pipelined architecture of AES encryption processor for higher speed applications. The second architecture achieved 31.4574 Gbps as maximum throughput and using 30 Block RAM. These designs utilize the low cost and low power Spartan3E$^{(TM)}$ FPGA. Hardware verification has been performed on the Spartan-3E starter board (xc3s500e-4) and the results were similar to simulation results.

Keywords:*AES; FPGA; S-Box; pipeline; Verilog; FSM; Spartan-3E.*

1. INTRODUCTION

The Advanced Encryption Standard also known as Rijndael is the standard symmetric key block cipher known for its robust security properties and simple implementation in both hardware and software [1]. It is capable of supporting block length of 128 bits and key lengths of 128, 192 and 256 bits [8]. The actual key size depends on the desired security level. The different versions are most often denoted as AES-128, AES-192, and AES-256. AES algorithm encrypts 128-bit blocks of plain text by repeatedly applying the same round transformation, as outlined in Fig. 1 [1] [9] [10]. AES-128 applies the round transformation 10 times, AES-192 uses 12, and AES-256 uses 14 iterations [7].

The AES algorithm can be efficiently implemented in hardware or software. Software implementations are very resourceful, but they offer a limited physical security and slower processing. In addition to the growing requirements for high speed, high volume secure communications combined with physical security, hardware implementation of cryptography takes place. The AES algorithm hardware implementation is faster and more secure than software implementation. There have been various hardware implementations of AES for ASIC e.g. [2], [5], [7] and [30] and FPGA, e.g. [22], [23], [26], [34] and [35]. FPGA implementation is an intermediate solution between general purpose processors (GPPs) implementation and application specific integrated circuits (ASICs) implementation.This has the benefits of being customizable, and the cost of an FPGA can be less than the more powerful CPU.

Using an FPGA should result in a fair gain in performance. The design and implementation of AES encryption processor on the low cost and low power Spartan-3E FPGA is presented in this paper. Two architectures are presented; the first feature is the relatively low speed (1.39Gbps) and low FPGA resources (347 slices and 10 Block RAM) which makes it suitable for most low-end embedded applications.

The second feature is the high throughput (31.4574Gbps) in which a fast and area efficient composite field implementation of the byte substitution phase is designed using an optimum number of pipeline stages. Hardware verification has been done on the Spartan-3E starter board and the results were similar to simulation results.

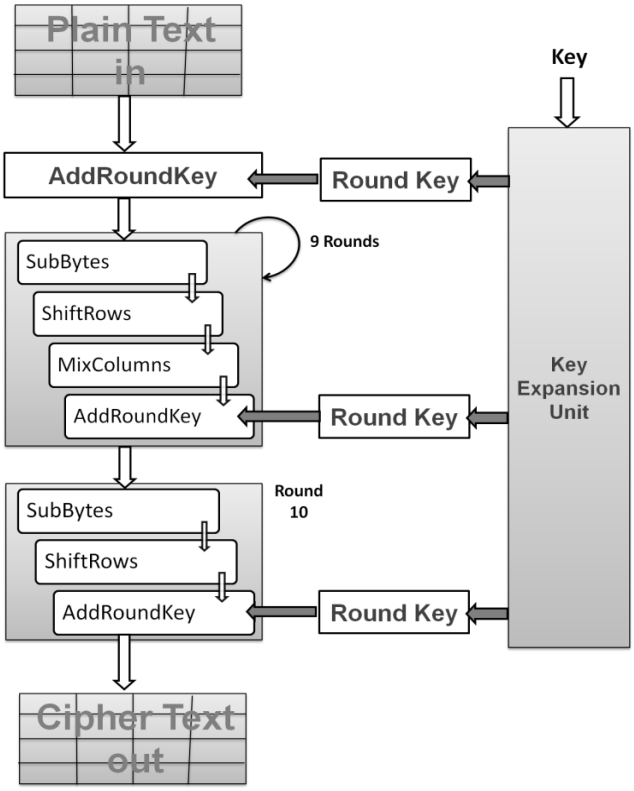

Fig.1:AES Process Block Diagram

1.1 *Description OfAes Algorithm*

The main functions and a block diagram of the algorithm are shown in Fig. 3. The function SubBytes performs a non-linear transformation on each byte of the input stateindependently [1]. It substitutes all bytes of the State using a look-up table called S-Box as can be seen in Fig. 2.

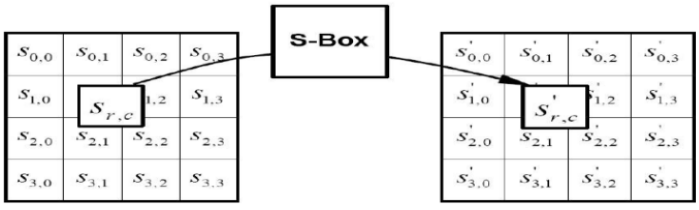

Fig.2: SubBytes function operates on state [1]

The ShiftRows step is a straightforward byte transposition. It rotates the rows of the state to the left by an offset. The offset equals the row index, the first row is not rotated at all; the second, third and fourth rows are rotated to the left by one, two and three bytes respectively as shown in Fig. 3.

The MixColumns transformation operates on the columns of the state in which the four elements of each column are treated as a four-term polynomial. The four elements of each column are multiplied by a constant polynomial and reducing to x^4+4.

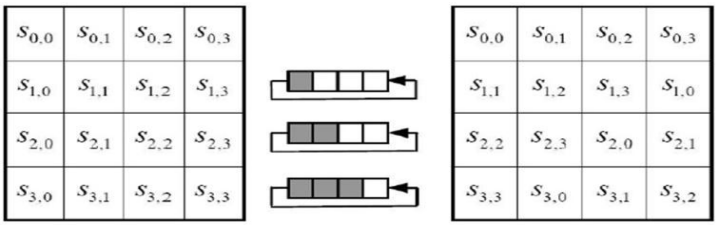

Fig.3: ShiftRows operates on the rows of the state [1]

The mapping between input and output of MixColumns is defined by the matrix multiplication given in Eq. 1.

$$\begin{pmatrix} b_{0,c} \\ b_{1,c} \\ b_{2,c} \\ b_{3,c} \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} a_{0,c} \\ a_{1,c} \\ a_{2,c} \\ a_{3,c} \end{pmatrix} \quad (1)$$

The key addition is the final cipher function denoted AddRoundKey. In which, the current state is modified by combining it with current round key using bit-wise XOR operation.

2. HARDWARE DESIGN

The datapath block diagram of the proposed architecture design is shown in Fig. 4. A 128-bit architecture is used to offer the greatest degree of parallelism to increase concurrency of AES computations that leads to a higher throughput. The whole AES hardware blocks are composed of five operational modules, which are the SubBytes, the ShiftRows, the Mix- Columns, the AddRoundKey and the Key expansion circuits. In the following sections, the design of high performance architectures for the Sub-Bytes, the MixColumns and the Key expansion operational modules are described.

2.1 *Subbytes*

S-Box based on Galois Field GF(2^8) is directly constructed by performing two transformations; first taking a multiplicative inverse in the Galois Field GF(2^8) and then applying the standard affine transformation over Galois Field GF (2^8) [6]. The polynomial representation in GF(2^8) is

$b(x)= b^7 x^7 +b^6 x^6 +b^5 x^5 +b^4 x^4 +b^3 x^3 +b^2 x^2 +x+ 1$.

1) Affine Transformation:

The affine transformation f can be described as a polynomial multiplication, followed by the XOR with a constant as outlined in Eq.2 [4].

$$
\begin{aligned}
b_7 &= a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_3 \\
b_6 &= \overline{a_6 \oplus a_5 \oplus a_4 \oplus a_3 \oplus a_2} \\
b_5 &= \overline{a_5 \oplus a_4 \oplus a_3 \oplus a_2 \oplus a_1} \\
b_4 &= a_4 \oplus a_3 \oplus a_2 \oplus a_1 \oplus a_0 \\
b_3 &= a_7 \oplus a_3 \oplus a_2 \oplus a_1 \oplus a_0 \\
b_2 &= a_7 \oplus a_6 \oplus a_2 \oplus a_1 \oplus a_0 \\
b_1 &= \overline{a_7 \oplus a_6 \oplus a_5 \oplus a_1 \oplus a_0} \\
b_0 &= \overline{a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_0}
\end{aligned}
\qquad (2)
$$

2) Multiplicative Inversion in GF(2^8):

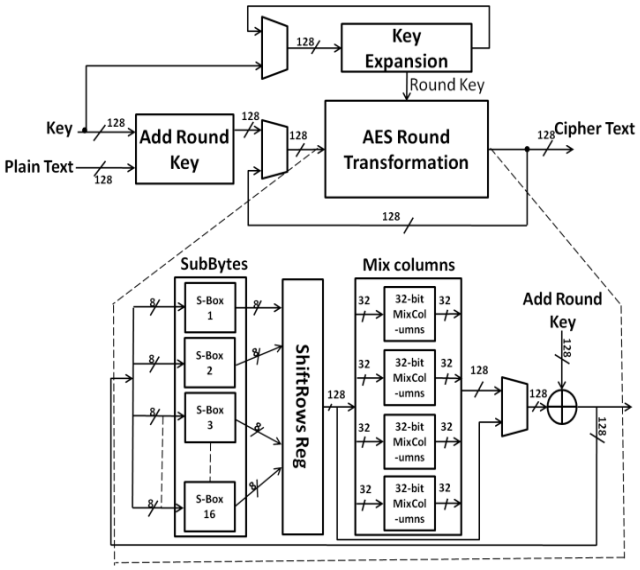

Fig.4: The detailed design of the proposed iterative AES architecture

The composite field procedure used for calculating multiplicative inverses is an efficient method which was proposed by [2] and [13]. From [2], [3], [10] and [11], the multiplicative inverse circuit in GF(2^8) can be produced as shown in Fig. 5.All of these blocks have been converted to polynomial representation by decomposing the representation of the field elements, such as performing calculations in a composite field of GF($(2^4)^2$) and GF($(((2^2)^2)^2$) instead of GF(2^8) because all Galois Field representations of the same order (for example, GF(2^8), GF($(2^4)^2$) and GF($(((2^2)^2)^2$) are isomorphic, this will lower the complexity of the execution of multiplicative inversion [12].

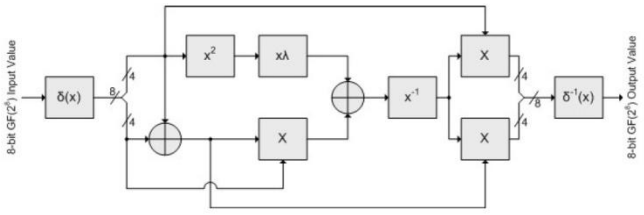

Fig.5: Multiplicative inversion module for the S-Box

Where:

δ	: Isomorphic mapping to Composite Fields
x^2	: Squarer in $GF(2^4)$
$x\lambda$	: Multiplication with constant λ in $GF(2^4)$
x^{-1}	: Multiplicative Inversion in GF(2^4)
×	: Multiplicative operation in GF(2^4)
δ^{-1}	: Inverse Isomorphic mapping to GF(2^8)

2.2 *Mixcolumns*

In MixColumns operation, the columns of the state are considered as polynomials over GF(2^8) and multiplied by modulo x^4+1 with a fixed polynomial c(x), where $c(x) = 03x^3 + 01x^2 + 01x + 02$ [1]. This can be written as a matrix multiplication as stated in Eq.3. Where ($a_{3,c}$, $a_{2,c}$, $a_{1,c}$, $a_{0,c}$) is a four-byte column of the state and the output column of MixColumns is ($b_{3,c}$, $b_{2,c}$, $b_{1,c}$, $b_{0,c}$).

$$
\begin{aligned}
\begin{pmatrix} b_{0,c} \\ b_{1,c} \\ b_{2,c} \\ b_{3,c} \end{pmatrix} &= \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} a_{0,c} \\ a_{1,c} \\ a_{2,c} \\ a_{3,c} \end{pmatrix} = \\
&\begin{pmatrix} 02 & 02 & 00 & 00 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} a_{0,c} \\ a_{1,c} \\ a_{2,c} \\ a_{3,c} \end{pmatrix} \oplus \\
&\begin{pmatrix} 00 & 01 & 01 & 01 \\ 01 & 00 & 01 & 01 \\ 01 & 01 & 00 & 01 \\ 01 & 01 & 01 & 00 \end{pmatrix} \begin{pmatrix} a_{0,c} \\ a_{1,c} \\ a_{2,c} \\ a_{3,c} \end{pmatrix}
\end{aligned}
\qquad (3)
$$

Multiplication with the value 00 or 01 involves no processing at all; multiplication with value 02 is denoted Xtime(x) and can be implemented efficiently with a dedicated routine that consists of a shift operation and a conditional XOR operation as shown in Eq.4 [1]. Multiplication with 03 is implemented as a multiplication with 02 plus an additional XOR operation with the operand as shown in Eq.3. Fig. 6 shows the circuit diagram of Xtime function, which needs 3 XOR gates.

$$
\begin{aligned}
a \times 2 &= (a_7 x^8 + a_6 x^7 + a_5 x^6 + a_4 x^5 + a_3 x^4 + a_2 x^3 + \\
&\quad a_1 x^2 + a_0 x) \bmod m(x) \\
&= a_6 x^7 + a_5 x^6 + a_4 x^5 + (a_3 \oplus a_7) x^4 + \\
&\quad (a_2 \oplus a_7) x^3 + a_1 x^2 + (a_0 \oplus a_7) x
\end{aligned}
\qquad (4)
$$

Fig. 7 illustrates the proposed hardware for 32-bit MixColumns which can be realized in a small series of instructions. The only finite field multiplication used in this algorithm is multiplication with the element 02, denoted by 'Xtime'.

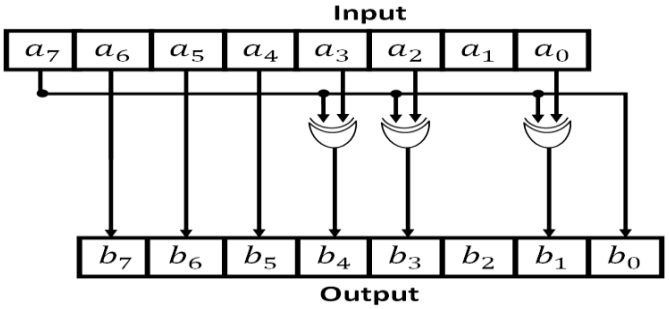

Fig.6: Circuit diagram of Xtime

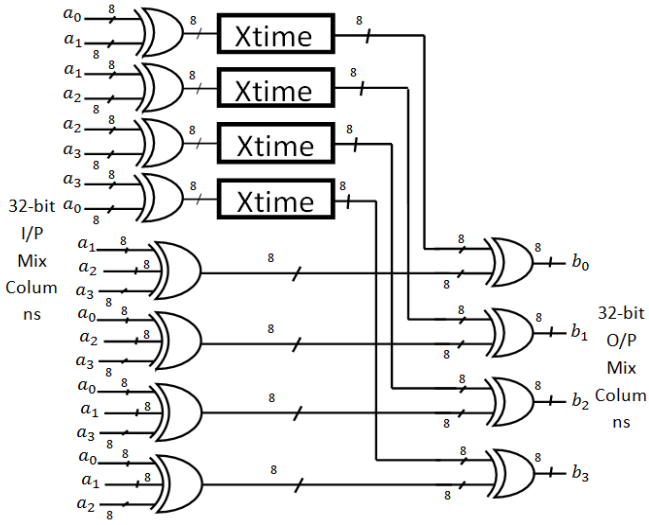

Fig.7: Efficient implementation of MixColumns

2.3 *Key Expansion Module*

The purpose of the key expander is to calculate a round key for each round based on the original input key. The implementation of the key schedule unit is done on-the-fly to lower storage requirements. In such implementations, a new round key is derived every iteration of the round transformation from the previous round key. The initial round key equals the original secret key. Fig. 8 shows the implementation of the key scheduler datapath. The calculation of the round keys is based on the SubBytes function and uses additionally some simple byte-level operations like XOR. In each round of the key schedule the last 32-bit word is rotated then the S-Box has to be applied. As a result four modules of the S-Box are required.

A constant called Roundconstant is also added to the output of the S-Box circuit with an XOR gate. This result is combined with the first 32-bit word with an XOR to generate the first 32-bit word of the new round key. The other three new 32-bit words are computed from the oldvalues and an XOR operation with the other inputs according to the algorithm as illustrated in Fig. 8.

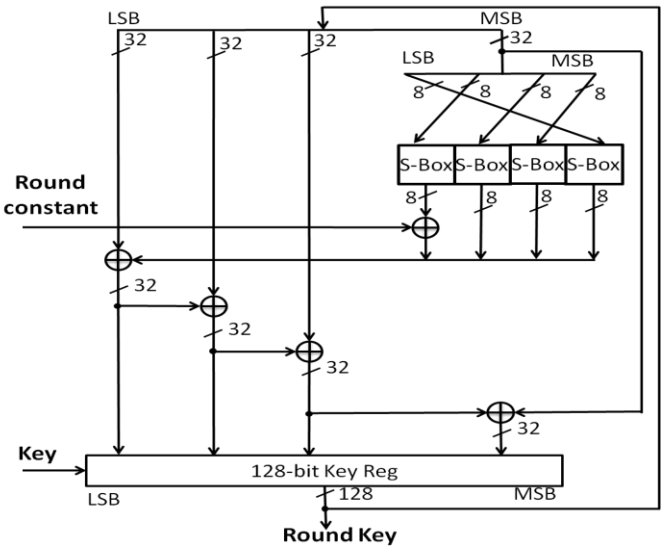

Fig.8: Key Expansion implementation circuit.

3 FPGA IMPLEMENTATION OF THE PROPOSED AES DESIGN ARCHITECTURE

The proposed iterative design of the AES processor shown on Fig. 4 was used for the implementation on the FPGA. Two different architectures were optimized for implementations on the FPGA which are iterative architecture and fully-pipelined architecture for increasing the speed at the cost of increased area.

3.1 *The Basic Iterative Architecture*

In this architecture each round manipulates 128 bits together. All 10 rounds are identical with the exception of the final round, which does not include the MixColumns transformation. The iterative method allows the calculation of one AES round per clock cycle. This leads to maximum hardware utilization in a comparison to the unrolled architecture because the same piece of hardware is used for all round transformations while the result is iteratively stored in a register and used as input to the next round. The key expansion unit generates one round key per clock cycle. Fig. 9 shows the general design of basic iterative AES core based on FPGA implementation.

The design consists of three main units:

- Controller unit.
- AES round transformation unit.
- Key expansion unit.

The round transformation unit and key expansion unit are shown in Fig.4 and Fig.8 respectively and described in detail in the previous section.The total design has 388 pins. It requires the Plain_text, Key and Cipher_text which have a 128 bit length. The signals used to control the proper operations of the core are clk, reset, start and done**.**

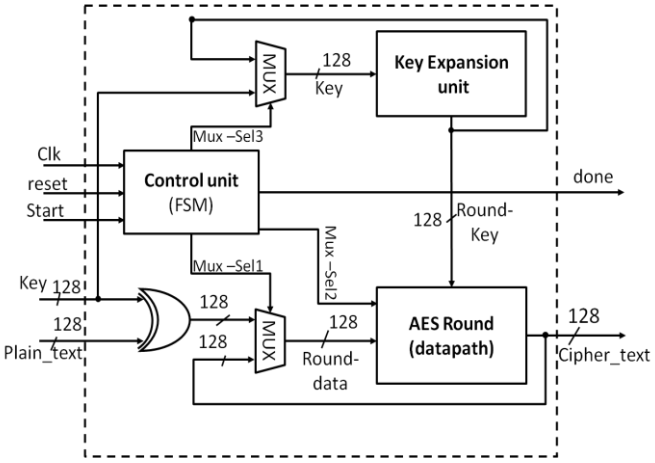

Fig.9: General Architecture of the proposed design

As the IOBs requirements of the basic iterative AES architecture, exceeds the xc3s500e-4fg320 device resources, which has just 232 IOBs, Plain text and key are multiplexed into one wire called Key_plain_text and stored in registers to be processed later on in parallel. So, the final architecture multiplexes the Plain text and Key 128-bit buses. No additional clock cycles are required. In a given clock cycle, a bus is registered, and in the next clock cycle, the other bus. The output Cipher_text is divided into two 64-bit nipples, the lower nipple (from bit 0 to 63) is sent first when the signal done is activated and the higher nipple (from bit 64 to 127) is sent in the following clock cycle so, to obtain the 128-bit Cipher_text output an extra clock cycles is required in the last round (Round10) where the processor stays for two clock cycles. Fig. 10 shows the resulting modification in I/O diagram of the AES processor.

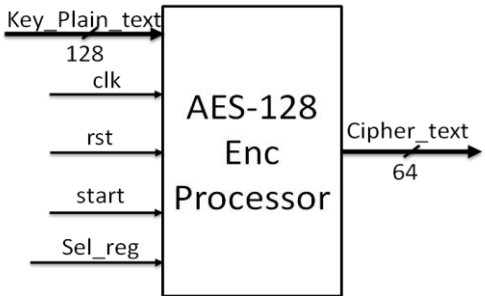

Fig.10: I/O diagram of AES processor

3.1.1 *Controller Unit*

A 12-state Finite State Machine, FSM was used to implement the controller and to keep track of the current round since it is easy to debug and upgrade. The output control signals of the FSM are described in Table 1. The state diagram and FSM initial values are shown in Fig.11. As can be seen, the input signal start modifies the current state from Idle state to Round0 (initial round) state in which the key and input-text (Cipher-text XOR Key) are registered. In this state, the two control signals Mul-Sel1 and Mul-Sel3 have value of logical one to control the data flow of both input-text and round-cipher-text (1 for input-text and 0 for and round-cipher-text), while the Mul-Sel13 selects the input key or round-key (1 for input key and 0 for round-key). Also the next ten states compute the ten roundsleft. Each round key, as well as round transformation, is completed in one clock cycle. In each of the following clock cycles, Round1 to Round10 states are active, in each, different ROUND-Constant value is applied to the Key Expansion module. The process is completed in Round10 when the done signal is pulsed.Only in this state, done=1 indicates a valid output. The done signal is activated only for two clock cycle because the output Cipher text is divided into two nipples. The lower 64-bits nipple is passed then in the next cycle the higher 64-bit nipple is passed. This division is done due to the lack of IOB pin of the target FPGA.

Table1: Control Signals

State	Control Signals				
	Round-	Mul-Sel1	Mul-Sel2	Mul-Sel3	done
Idle	-	-	-	-	-
Round0(R0)		1	0	1	0
Round1(R1)	00000001	0	0	0	0
Round2(R2)	00000010	0	0	0	0
Round3(R3)	00000100	0	0	0	0
Round4(R4)	00001000	0	0	0	0
Round5(R5)	00010000	0	0	0	0
Round6(R6)	00100000	0	0	0	0
Round7(R7)	01000000	0	0	0	0
Round8(R8)	10000000	0	0	0	0
Round9(R9)	00011011	0	0	0	0
Round10(R10	00110110	0	1	0	1

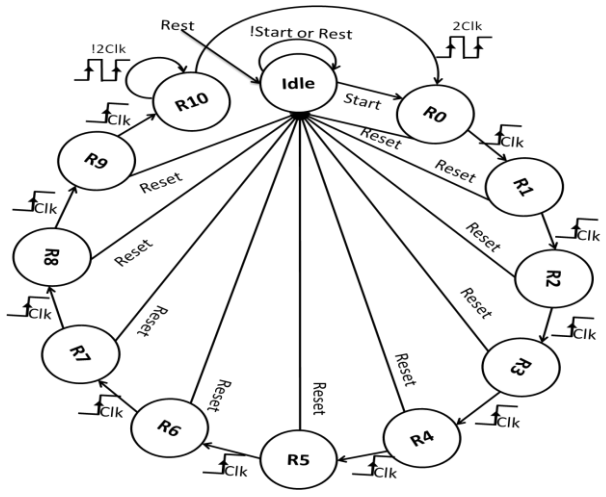

Fig.11: FSM Implementation of the Top controller module of the AES

3.2 *The Pipelined Architecture*

This architecture is used to increase the throughput at the cost of increased area by adding more inner and outer pipelining registers to achieve multiple processing simultaneously. As shown in Fig. 12, the architecture of the proposed fully pipelined AES processor is composed of ten AES functional blocks and key expansion circuits. The inter-pipelined and outer-pipelined stages are utilized for implementations. In the inter-pipelined scheme, the register arrays are allocated among the operational circuits of SubByte, MixColumns and AddRoundKey. The use of the S-Box as one continuous path would be costly in terms of logic delay hence reducing the highest possible achievable clock frequency. The S-Box block is further divided into five pipelined stages to break the logic delay in an attempt to achieve a higher clock frequency. Fig. 13 shows the applied pipeline registers in the S-Box where the dotted line indicates a pipelined register. In addition one Pipelining register is added to the MixColumns. From Round1 to Round7, 5-stages S-Boxes based combinational logic implementation are used, while in Round8, Round9 and Round10, the S-Boxes are mapped into Block RAMs. The number of inner pipeline registers equal to 7×7(form Round1 to Round7) plus 2× 3 due to the use of Block RAMs (from Round8 to Round10).

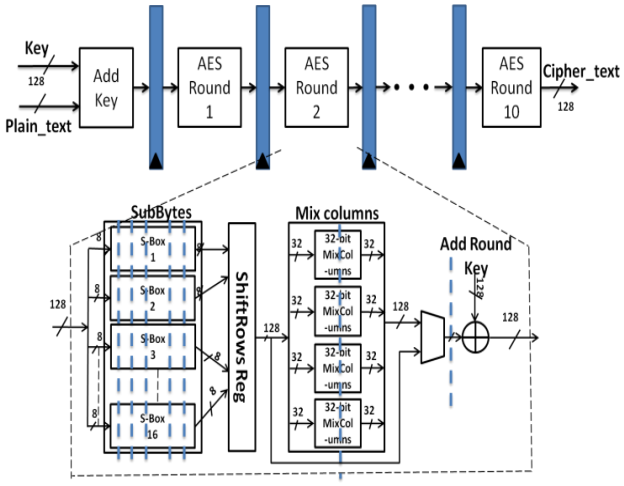

Fig.12:AES fully pipelined architecture

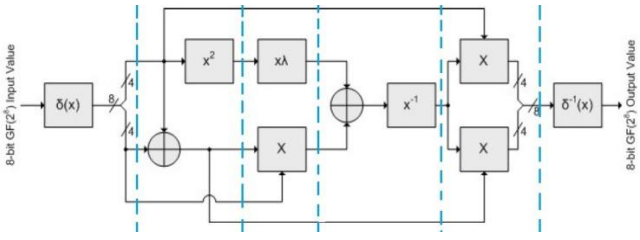

Fig.13: Multiplicative inversion module for the S-Box

In the outer pipelined scheme, ten pipelining registers are added between each AES round computation. Thus, the latency delay of the proposed full-pipelined AES processor is 65 clock cycles (55 inner +10 outer. As result of this architecture, the throughput is 128 bits per clock cycle.

4. PERFORMANCE RESULTS AND COMPARISON WITH RELATED WORKS

Both the basic iterative architecture shown in Fig.4 and the fully pipelined architecture shown in Fig. 12 have been coded by Verilog HDL and implemented on a Xilinx Spartan-3E as a target device. All the results are synthesized, simulated and implemented based on the Xilinx ISE 13.4 design tool, the ISim simulator was used to perform functional and timing simulations for the Verilog design and the Xilinx XST synthesis tool was used for performing logic synthesis. For the basic iterative architecture, two different implementation techniques of S-Box are used, the first method is based on combinational logic implementation of S-Box as described previously and the second method is using 256 × 8 bit Rom lookup table using the special feature of Xilinx Spatan-3E that has block RAMs. These are dedicated embedded memory blocks ideal for implementing S-Box. These block RAMs can either be used as Single or Dual port RAM. In this work, a Dual port RAM is configured as two separate Single port RAMs as shown in Fig.14. Thus two S-Boxes can be implemented in one block RAM only, therefore utilizing each Dual port RAM block as two single ports RAM, 20 S-Boxes are realizable in 10 block RAMs.

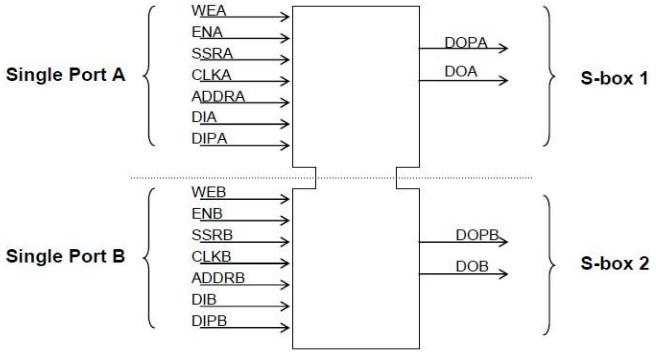

Fig.14: One Block RAM becomes two independent Single-Port RAMs used to implements two S-Boxes

The performance results and FPGA resources required of our proposed architectures are shown in Table 2. These results are taken after Placing and Routing report of the design.As can be seen from Table 2, the resulting utilization hardware by the AES processor with maximum place-and-route efforts for the different architecture is varying according to the device used and techniques used in the design. For instance, the speed grade of the device defines the maximum toggle frequency of the CLB. And using of the Block RAM leads to lower resource in terms of the slices (from 1563 to 347) and lower critical path time, thus higher clock frequency and maximum throughput (from 0.7034Gbps to 1.3988Gbps) regarding the basic iterative architecture. A very high throughput of

31.3341Gbps is achieved using the fully pipelined architecture and a mix of BRAM and combinational logic to implement the S-Boxes.

Table2. Implementation performance results after place and route

Architecture	S-Box technique	FPGA Device	Slices	Block RAM	Max Clock MHz	Latency cycles	Throughput Gbps
Basic iterative	Combinational logic	Spartan3E xc3s500e-5	1563	0	71.444	13	0.7034
Basic iterative	Block RAM	Spartan3E xc3s500e-5	347	10	142.066	13	1.3988
Fully pipelined	Pipeline S-Box	Spartan3E xc3s1600e-5	14710	30	245.761	65	31.4574

Table 3: Comparison with other FPGA implementation of AES basic architecture

Ref #	Device	BRAM or BROM	Slices	Max clock (MHz)	Latency (clk cycles)	Throughput Gbps
[15] Enc/Dec	XC2V8000-5	4	8378	65	10	0.832
[16] Enc	XCV812	36	2744	22.41	11	0.259
[17] Enc/Dec	XCV300 BG432	-	2358	22	11	0.259
[18] Enc/Dec	XCV1000e-8	-	5150	76	21	0.463
[19] Enc/Dec	XC2V3000-6	0	7617	75.3	11	0.876
[20] Enc/Dec	XC2S200E	6	196	28.742	250	0.0164
[20] Enc/Dec	XC2V500	6	192	78.59	250	0.0406
[21]	xc3s200pq208-5	10	481	231.97	11	2.699
[22]	Xilinxxcv1000bg560-4	0	3528	25.3	11	0.2942
[23] Enc	XC2V1000-FG456	20 distrib memory	2335	86.94	10	0.92
[23] Enc	XC2V1000-FG456	10	586	96.42	10	1.45
[24]	Virtex-II Pro	44	2703 LUT	196	-	1.19
[25] Enc/Dec	XC2S30-5	3	222	50	-	0.139
[28]	Spartan-3 XC3S20	11	148	287	-	0.632
[31] 1S-Box	Virtex4	0	2018 LUTs	123	400	0.040
[31] 2S-Box	Virtex4	0	2214 LUTs	130	180	0.0925
[31] 4S-Box	Virtex4	0	2490 LUTs	145	130	0.142

A comparison of our results has been carried out with other similar FPGA implementations of the AES. Table 3 reports the measurements of basic iterative hardware architectures. Noticing that, different device families and speeds will yield different performance results in addition to data-path width of the architecture. Table 4 reports the performance of the pipelined architecture.

Table 4: Comparison with other FPGA implementation of the AES pipelined architecture

Ref#	Device	Block RAMs	Slices Or	Max.clock (MHz)	Latency (clkcycles)	Throughput Gbps
[26] Enc	VirtexII -Pro	84	5177	168.3	31	21.54
[26] Enc	XC2VP20-7	0	9446	169.1	71	21.64
[27]	XCV812e-6	0	9406	71.8	-	8.968
[27]	XCV800-8	0	9406	93.5	-	11.685
[27]	XCV1000-6	0	11014	125.3	-	15.65625
[27] Enc	XCV1000e-8	0	11022	168.4	71	21.56
[28]	Spartan- III	-	20720	240.9	-	30.11
[29]	Virtex- II	-	31674	222.8	-	27.86
[19] ENC/Dec	XC2V3000-6	0	139357	222.2	51	28.4
[10] Enc/Dec	Cyclone II	18	3039 LEs	198.93	40	2.546
[14] Enc	XC2VP70-7	200	5408	232.6	60	29.77

5 SIMULATION RESULTS

The simulation results of AES encryption module are shown in Fig. 15 and Fig. 16. The Plain_text, Key and the expected output Cipher_text vectors were taken from FIPS publication [32] and the AES algorithm validation suite [33]. Here are samples of the data which was applied to the test in the same order:

1) Firstly:
- Plain_text: 3243f6a8885a308d313198a2e0370734.
- Key: 2b7e151628aed2a6abf7158809cf4f3c.
- Cipher_text: 3925841d02dc09fbdc118597196a0b32.

2) Secondly:
- Plain_text: 00112233445566778899aabbccddeeff.
- Key: 000102030405060708090a0b0c0d0e0f.
- Cipher_text: 69c4e0d86a7b0430d8cdb78070b4c55a

3) Thirdly:
- Plain_text: 80000000000000000000000000000000.
- Key: 00000000000000000000000000000000.
- Cipher_text: 3ad78e726c1ec02b7ebfe92b23d9ec34.

4) Fourthly:
- Plain_text: 00000000000000000000000000000000.
- Key: 10a58869d74be5a374cf867cfb473859.
- Cipher_text: 6d251e6944b051e04eaa6fb4dbf78465.

5) Fifthly:
- Plain_text: ffffffffffffffffffffffffffffffc0.
- Key: 00000000000000000000000000000000.
- Cipher_text: 0833ff6f61d98a57b288e8c3586b85a6

Fig. 15 shows the simulation results of the basic iterative AES algorithm. As can be seen, the output Cipher_text result of the encryption was completed in 13 clock cycles and the done signal goes high every 13 clock cycles.

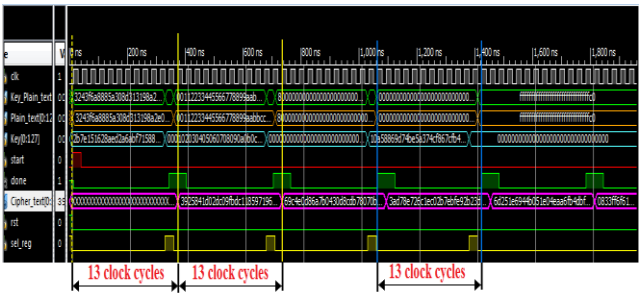

Fig.15: Simulation result of a basic iterative AES architecture

Fig. 16 shows the simulation results of the pipelined AES algorithm. It is clear that the output Cipher text result of the encryption was obtained after 65 clock cycles. The throughput achieved by this architecture is 128bits per clock cycle. All results were successfully verified against the expected sample Cipher text results.

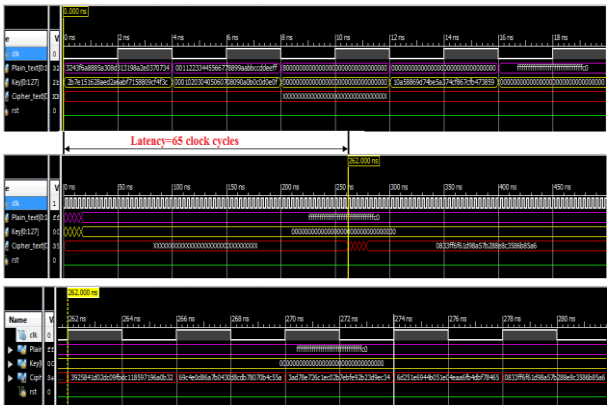

Fig.16: Simulation result of the Pipelined architecture

6 HARDWARE TESTING

For the hardware testing of the AES processor functionality implemented using Spartan-3E Starter Kit board which houses the target FPGA (XC3S500E-4), additional test circuitry and modules were added to the design and integrated to the AES design in order to perform such tests. Most of the IOB pins are interfaced with components on the board and are not available to use. Fig. 17 shows the test circuit, a clock source of a 50MHz clock from oscillator is connected to the AES processor. The same first four vectors of Key and Plain_text were stored in ROM and are read at every 13 clock cycles of AES processor latency. This is calculated by the clock divider module. At each 13-clock cycles different Key and Plain_text are supplied to the AES Enc module. Thus the output result of the AES encryption is stored in BRAM to be read and verified later. Each Cipher_text result was stored in a different location in RAM according to the address controlled by the FSM controller module. When writing to the RAM ends, the reading process takes place. The values written to the RAM are passed to the LCD module to be displayed on the 2 line×16 character LCD. This is enough to display one result at a time.

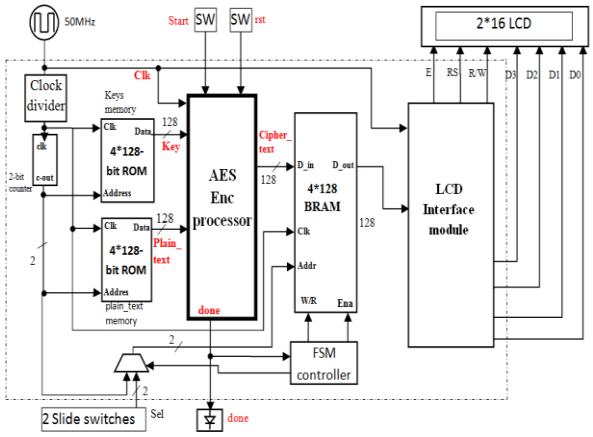

Fig.17: Testing hardware circuits

Fig.18: The first output Cipher text result when sel=2'b00

Fig.19: The second output Cipher text result when sel=2’01

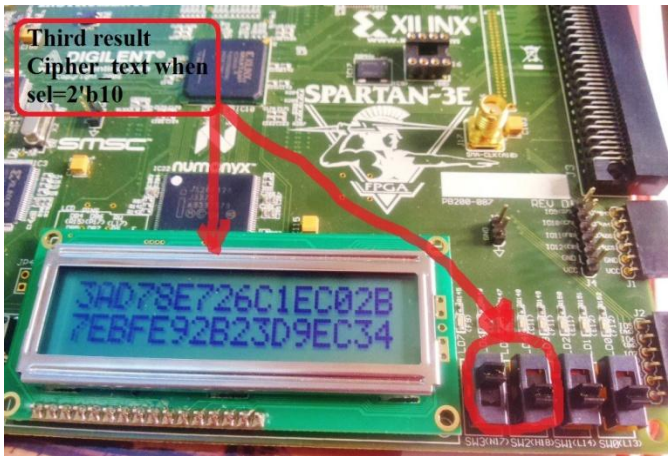

Fig.20: The third output Cipher text result when sel=2’10

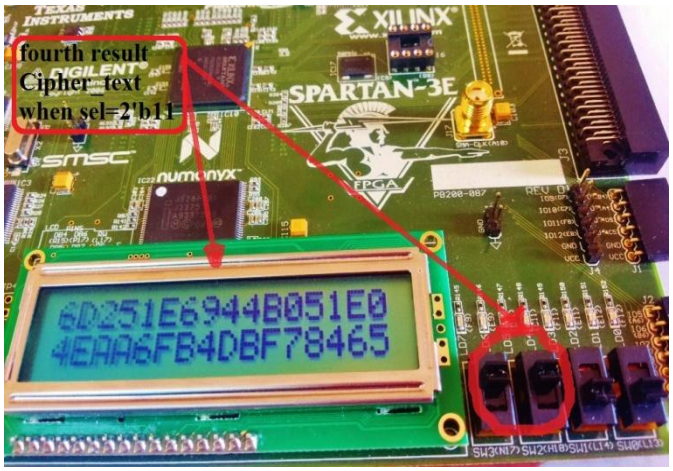

Fig.21: The third output Cipher text result when sel=2’11

As can be seen in Fig. 18, Fig. 19, Fig. 20 and Fig. 21, the results of the hardware testing of the AES encryption processor were verified and found to be identical to simulation results.

7 CONCLUSION

Two different architectures for AES encryption processor are proposed, the basic iterative and the fully pipelined methods. Two implementation techniques for the S-Box, the Block RAM available in the Spartan-3E FPGA andthe fully combinational logic are used.Also mixing between the two techniques in the fully pipelined architecture yields a maximum throughput of 31.4574Gbps using 30 Block RAMs and 14720 Slices of Spartan-3E FPGA with an optimum number of pipeline stages for the S-Box. The encrypted cipher_text results are analysed and proved to be correct using simulation and hardware verification and both results are identical. The encryption efficiency of the proposed AES algorithm was studied and compared with other similar FPGA implementations of AES algorithm. We found that, using of the BRAM leads to lower resource in terms of the slices and lower critical path time, thus higher clock frequency and max throughput.

REFERENCES

[1] J. Daernen and V.Rijrnen, "Specification of Rijndael," in The Design of Rijndael: AES - The Advanced Encryption Standard, Berlin; New York: Springer-Verlag Berlin Heidel- berg, 2002, pp. 31-55.

[2] A. Satoh, S. Morioka, K. Takano and S. Munetoh, "Acompactrijndael hardware architecture with S-box optimization," Springer-Verlag Berlin Heidelberg, 2001.

[3] E. NC Mui, "Practical implementation of Rijndael S-Box using combinational Logic," unpublished.

[4] B. Rashidi and B. Rashidi, "Implementation of an optimized and pipelined combinational Logic Rijndael S-Box on FPGA," Published in MECS, Computer Network and Information Security, 2013.

[5] J. Wolkerstorfer, E Oswald, and M Lamberger, "An ASIC implementation of the AES SBoxes," In Bart Preneel, editor, Topics in Cryptology - CT-RSA 2002,The Cryptographers Track at the RSA Conf. 2002.

[6] G. KUMAR and P. MAHESH, "Implementation of AES algorithm using Verilog," International Journal of VLSI and Embedded Systems-IJVES, Vol 04, Article 05090; June 2013.

[7] G. Leelavathi, S. Prakasha, K Shaila, K.R. Venugopal and L.M. Patnaik, "Design and implementation of Advanced Encryption Algorithm with FPGA and ASIC," International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 3, ISSN: 2320 8791, pp.1-8, June-July, 2013.

[8] *Announcing the Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publication 197, November 2001.

[9] H. Trang and N. V. Loi, "An efficient FPGA implementation of the Advanced Encryption Standard algorithm,"978-1-4673-0309-5, IEEE. 2012.

[10] D. Kenney, "Energy efficiency analysis and implementation of AES on an FPGA," M.S. thesis, Dept. Elect. Eng., Univ. of Waterloo, Canada, 2008.

[11] N. Ahmad, R. Hasan and W.M Jubadi, "Design of AES S-Box using combinational logic optimization," in IEEE Symposium on Industrial Electronics & Applications (ISIEA), 2010.

[12] X. Zhang and K. Parhi, "High-Speed VLSI architectures for the AES algorithm," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.2, No. 9, September 2004.

[13] A. Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao and P Rohatgi, "Efficient rijndael encryption implementation with composite field arithmetic," in CHES '01: Proceedings of the Third International Work-shop on Cryptographic Hardware and Embedded Systems, 2001.

[14] D. Kotturi, S. M. Yoo, and J. Blizzard, "AES crypto chip utilizing high- speed parallel pipelined architecture," in IEEE International Symposium on Circuits and Systems, vol. 5, pp.4653-4656, May 2005.

[15] R. Sever, A. N. Ismailglu, Y. C. Tekmen, M. Askar, and B. Okcan, "A high speed FPGA implementation of the Rijndael algorithm," in Euromicro Symposium on Digital System Design, pp.358-362, Sep. 2004.

[16] N. A. Saqib, F. R. Henriquez, and A. D. Perez, "AES algorithm implementation - an efficient approach for sequential and pipeline architectures," The Fourth

[23] I. A. Badillo, C. F. Uribe, and R. Cumplido, "Design and implementation of an FPGA-based 1.452-Gbps non-pipelined AES architecture," M. Gavrilova et al. (Eds.): ICCSA 2006, LNCS 3982, pp. 446 455, 2006. Springer-Verlag Berlin Heidelberg 2006.

[24] J. Lu and J. Lockwood, "IPSec implementation on Xilinx Virtex-II Pro FPGA and its application," Reconfigurables Architectures Workshop (RAW), April 2005.

[25] P. Chodowiec and K. Gaj, "Very compact FPGA implementation of the AES algorithm," in Proc. of CHES , LNCS 2779, pp. 319-333,2003.

[26] A. Hodjat and I. Verbauwhede, "A 21.54 Gbits/s fully pipelined AES processor on FPGA," in IEEE Symposium. on Field-Programmable Custom Computing Machines, pp. 308-309, Apr. 2004.

[27] X. Zhang and K. K. Parhi, "High-Speed VLSI architectures for the AES algorithm," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 9, Sep. 2004.

[28] T. Good and M. Benaissa, "Pipelined AES on FPGA with support for feedback modes (in a multi-channel environment)," IET Information Security, vol. 1, pp. 1-10, 2007.

[29] G. Singh and R. Mehra, "FPGA based high speed and area efficient AES encryption for data security," Mexican International Conference on Computer Science, pp.126-130, Sep. 2003.

[17] N. Sklavos and O. Koufopavlou, "Architectures and VLSI implementations of the AES-Proposal Rijndael," IEEE Transactions on Computers, vol. 51, issue 12, pp.1454-1459, Dec. 2002.

[18] S. S. Wang and W. S. Ni, "An efficient FPGA implementation of advanced encryption standard algorithm," in International Symposium on Circuits and Systems, vol. 2, pp.II-597-600, May 2004.

[19] C. P. Fan and J. K. Hwang, "FPGA implementation of high throughput sequential and fully pipelined AES algorithm," International journal of electrical engineering, Vol.15, NO.6 (2008).

[20] A. Moussa and Z. Ismaili, "Self-partial and dynamic reconfiguration implementation for AES using FPGA," IJCSI international journal of computer science issues, Vol. 2, 2009.

[21] A. Aziz and N. Ikram, "Hardware implementation of AES- CCM for robust secure wireless network," unpublished.

[22] A.J. Elbirt, W. Yip, B. Chetwynd and C. Par, "An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists," Third AES candidate conference, April 2000.
International Journal of Research and Innovation in Computer Engineering, vol. 1, no. 2, pp. 53-56, Feb. 2011.

[30] T. Ichikawa et al, "Hardware Evaluation of the AES Finalists," in Proc.3th AES Candidate Conference, New York, April 2000.

[31] J. Rejeb, T. Lee and S. Kaginele, "Compact and power conscious private- key cryptosystem for wireless devices," in Second International Conference on Wireless and Mobile Communications, ICWMC 2006.

[32] Announcing the Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, November 2001.

[33] L. E. Bassham III, The Advanced Encryption Standard Algorithm Validation Suite (AESAVS), November 15, 2002.

[34] M. McLoone and J. V. McCanny, "Single-chip FPGA implementation of the Advanced Encryption Standard algorithm," FPL 2001, LNCS 2147, pp. 152-161, 2001.

[35] G. P. Saggese, A. Mazzeo, N. Mazzocca and A. G. M Strollo, "An FPGA-Based Performance Analysis of the Unrolling, Tiling, and Pipelining of the AES Algorithm," FPL 2003, LNCS 2778, pp. 292-302, 2003.