

Design and Implementation A Compact AES Architecture for FPGA Technology

A. Amaar, I. Ashour and M. Shiple

Abstract—This paper presents a compact implementation of advanced encryption standard AES using different devices of FPGA technology. This implementation can be carried out through several trade-off between area and speed. A 128 bit AES (The Advanced Encryption Standard) and the relative key expansion are designed, and synthesized using Xilinx ISE 6.1 simulated by ModelsimSE 6.5 then downloaded to various Xilinx FPGA devices.

Keywords—FPGA, AES, cryptography, VHDL, cipher

I. INTRODUCTION

DURING the selection process of the AES [1], an important criterion was the efficiency of the cipher in different platforms, including FPGAs. Since 2001, various implementations have consequently been proposed, exploring the different possible design tradeoffs ranging from the highest throughput to the smallest area [2]. Each of those implementations usually focuses on a particular understanding of “efficiency”. The three major design targets with respect to hardware realization are: optimization for area or cost, low latency that minimizes time to encrypt a single block and high throughput to encrypt multiple blocks in parallel. All these design criteria involve a trade-off between area and speed. There is a wide range of equipment where encryption is needed for authentication and security but throughput is not the principal concern. A low cost, small area design could be used in smart card applications as well as in other storage devices and low speed communication channels [3].

II. THE AES ALGORITHM

The AES is a substitution permutation network (SPN) developed by Joan Daemen and Vincent Rijmen, has been approved by the U. S. National Institute of Standards and Technology (NIST) as the new Advanced Encryption Standard (AES). It became official in October 2000, replacing DES (FIPS 197, 2001).

(AES, Rijndael) algorithm is a symmetric block cipher that processes data block of 128, 192 and 256 bits using, respectively, keys of the same length. In this paper, only the 128 bit encryption version (AES-128) is considered. Rijndael operates on a state that is initialized with a plaintext block, and after encryption this contains the cipher text.

A. AddRoundKey

AddRoundKey is an XOR between the state and the round key. This transformation is its own inverse.

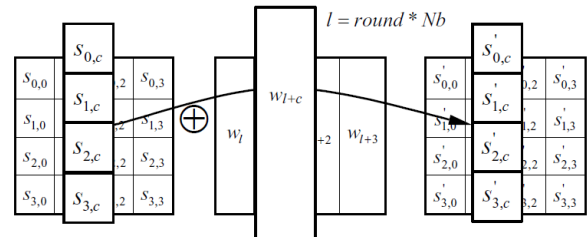


Fig. 1 Add Round Key transformation

B. SubBytes

SubBytes is a substitution of each byte in the block independent of the position in the state.

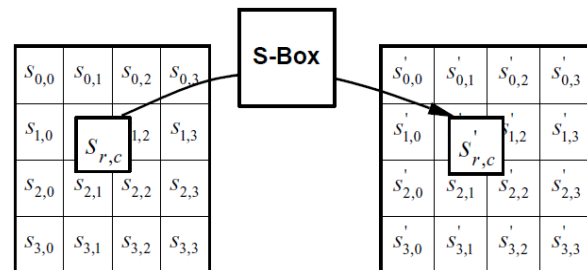


Fig. 2 SubByte transformation

This is an S-box. It is a bijection on all possible byte values and therefore invertible (the inverse S-box can easily be constructed from the S-box). This is the non-linear transformation. The S-box used is proved to be optimal with regards to non-linearity. The S-box is based on arithmetic in $GF(2^8)$.

C. ShiftRows

ShiftRows is a cyclic shift of the bytes in the rows in the state and is clearly invertible (by a shift in the opposite direction by the same amount).

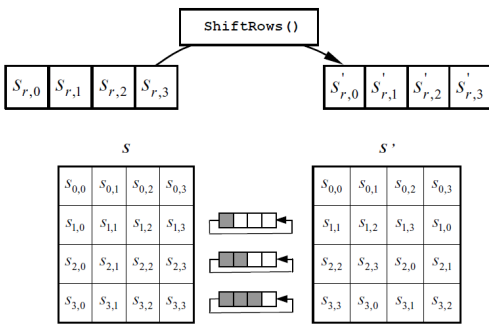


Fig. 3 ShiftRows transformation

D. MixColumns

Each column in the state is considered a polynomial with the byte values as coefficients. The columns are transformed independently by multiplication with a special polynomial $c(x)$. $c(x)$ has an inverse $d(x)$, that is used to reverse the multiplication by $c(x)$.

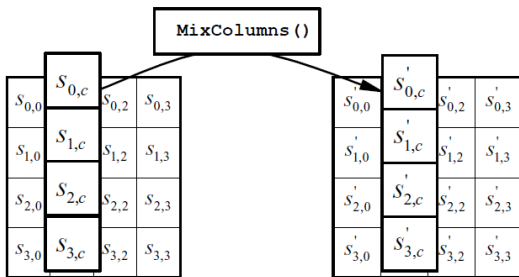


Fig. 4 MixColumns Transformation

III. THE PROPOSED ARCHITECTURE

There are many techniques to design AES architecture to yield optimized implementation.

Basic architecture in which each round manipulates 128 bit together and encrypts them by one clock cycle [4].

Pipelining in which throughput is increased versus area through adding more inner registers to achieve multiple processing simultaneously.

Loop Unrolling in which k rounds are implemented rather than one round to gain high speed versus area.

Chopping architecture in which the round data width is decreased to one quarter of the basic round since the whole plain text needs looping four times to complete one round. This technique will gain compact area over speed [5].

Proposed architecture is implementing 128 bits data-path for both cipher key and plaintext. The developed architecture combines basic architecture with one round and chopping technique to compromise the area with speed.

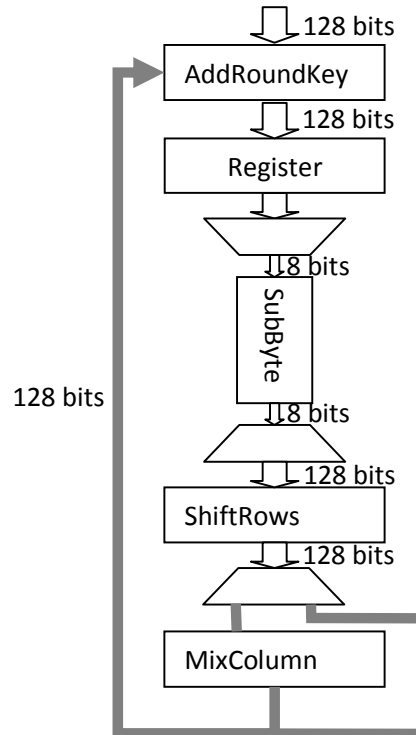


Fig. 5 The proposed architecture

The code of the proposed design is implemented in VHDL code see figure 8. Since the main point of the proposed architecture is to compromise the area and speed. Register-Transfer-Level (RTL) is checked frequently to avoid redundant hardware creation. Moreover, eliminate all in-between registers and latches to save the use of RAM except at the beginning round.

Since the Sbox consumes a large area, the proposed architecture implements one sbox (256 byte). The scenario of processing is established on the concept of chopping for Sbox. From this point of view, the 16 bytes iteratively loop through the implemented Sbox block. This save about 85 % the total area (the one over sixteen of sbox is implemented. for the looping scenario; the data need a control block to manage this iterations). On the other hand, this consumes 16 clock cycles to accomplish the substitution process. This technique saves area over the speed but some embedded applications require smaller throughput (wireless communication, digital cinema, pay TV, smart cards,.....)

ShiftRows block is implemented by shuffling the routing nets without using any shift registers (zero occupied slices) next figure shows the synthesise report of shiftRows block. This also saves the gate propagation delays and accelerates the processing time

* Final Report	
Final Results	
RTL Top Level Output File Name	: testshift.ngx
Top Level Output File Name	: testshift
Output Format	: NGC
Optimization Goal	: Speed
Keep Hierarchy	: NO
Design Statistics	
# IOs	: 256
Cell Usage :	
# IO Buffers	: 256
# IBUF	: 128
# OBUF	: 128
Device utilization summary:	

Selected Device	: 3s400pq208-4
Number of bonded IOBs:	256 out of 141
WARNING:Xst:1336 - (*) More than 100% of Device resources a	

Timing Summary:	

Speed Grade:	-4
Minimum period:	No path found
Minimum input arrival time before clock:	No path found
Maximum output required time after clock:	No path found
Maximum combinational path delay:	7.045ns

Fig. 6 Synthesizer report of ShiftRows

According to the ShiftRows block is synthesized alone, the summation of input/output pins exceeds the available resources. That is the cause of warning message. In our Design, plaintext and ciphertext are accessed in serial format.

AddRoundKey and MixColumns are implemented by combinational logic with full range data width (128 bits).

Initially, the proposed architecture was suffering from bad design strategies which tended to consume very large propagation delay (Maximum frequency=78MHz). Finite state machine (Moore machine) design strategy is considered to increase the speed of the processing. 0 shows the timing summary of XST synthesizer of whole cipher blocks

Timing Summary:

Speed Grade: -4
Minimum period: 3.935ns (Maximum Frequency: 254.130MHz)
Minimum input arrival time before clock: 4.957ns
Maximum output required time after clock: 6.555ns
Maximum combinational path delay: No path found

Completed process "Synthesize".

Fig. 7 Timing summary of proposed algorithm

FSM code style enhance the performance of the design by reducing the redundant loops and transparent latches in the way the maximum frequency increased 450%

IV. RESULTS

The paper proposes AES design combines key schedule and cipher block. Precomputation key is considered in this algorithm. The proposed algorithm try to chopping the main block consuming the area "SBOX", minimize in-between unwanted latches and shift registers to save area. Shift raw block is rejected and implemented by twisting the routing tracks. Mix column is implemented by combination gates.

The proposed minimum area AES architecture which is

described by VHDL is simulated using ModelSim to verify the functionality as a primer verification tool. Moreover, the proposed algorithm is synthesized and implemented (translate, fit, place and route) using Xilinx 6.2. The proposed architecture is downloaded at FPGA kit is designed by the research team. Some blocks are added to communicate with Matlab program to check the algorithm since the plain text sent to FPGA and incoming cipher text is checked with the reference. Matlab file checks the comparison between reference pattern and incoming cipher text if they are matching an OK sign is printed otherwise false sign is printed. Not just a block by block is testing but a complete file of plaintext is used to check the algorithm to check its reliability. Table I illustrates the comparison of proposed architecture and previous works. This comparison is done over various platforms. To be fair, the paper generates the keys and stores them to be used later in encryption so this comparison considers the stored keys algorithms not "on-the-fly" key schedule which affect deeply on the area consumption. The table shows how many saved slices are obtained from the proposed blocks. On the other hands, We have to consider low throughput according to the chopping of SBOX which means more processing clocks to process all 128bits of the input plaintext.

V. CONCLUSION

This paper presents a low area, cost effective AES cipher for encryption /decryption using a 128 bit iterative architecture. In this work, the amount of hardware resources has been optimized with respect to various proposed designs on alternative platforms (Spartan 3, and VirtixE). The cipher has been synthesized using Xilinx 6.2, simulated using modelsimSE 6.2, and tested using matlab. The algorithm is implemented by VHDL.

The architecture needs fewer logic cells than other ciphers and uses a few memory blocks as possible. Future work should concentrate on speed performance.

REFERENCES

- [1] National Institute of Standards and Technology. Advanced Encryption Standard (AES). Federal Information Processing Standards Publications – FIPS 197, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>, November 2001.
- [2] K. Järvinen, M. Tommiska, and J. Skyttä. Comparative survey of high-performance cryptographic algorithm implementations on FPGAs. In IEE Proceedings on Information Security, volume 152, pages 3 – 12, October 2005.
- [3] M. C. LIBERATORI and J. C. BONADERO "Aes-128 Cipher. Minimum Area, Low Cost FPGA Implementation"
- [4] Gaj, K and P. Chodowicz, " Comparison of the hardware performance of the AES candidates using reconfigurable hardware", Proceeding of RSA Security conference – Cryptographer's Track, San Francisco, CA, 2001
- [5] Ga'el Rouvroy, François-Xavier Standaert, Jean-Jacques Quisquater and Jean-Didier Legat, "Compact and Efficient Encryption /Decryption Module for FPGA Implementation of the AES Rijndael VeryWell Suited for Small Embedded Applications", April 2004
- [6] Philippe Bulens, François-Xavier Standaert, Jean-Jacques Quisquater, Pascal Pellegrin, and Gaël Rouvroy. Implementation of the AES-128 on Virtex-5 FPGAs, Progress in Cryptology - AfricaCrypt 2008, Volume 5023 of Lecture Notes in Computer Science, pages 16 - 26, Springer, June 2008

[7] F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, J.-D. Legat, Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements and Design Tradeoffs, in the proceedings of CHES 2003, Lecture Notes in Computer Science, vol 2779, pp 334-350, Cologne, Germany, September 2003, Springer-Verlag

Mustafa Shiple received M. Sc. In electrical and communication from AL-Azher Uni. , Cairo, Egypt 2005 From 2001 to 2005, He was a demonstrator and digital designer at Design center at NTI. In 2008, He promoted to be the hardware team leader at NTI. His primary research interests include digital signal processors, Cryptology, DPA of FPGA and digital processors.

TABLE I
COMPARISON OF PROPOSED ARCHITECTURE AND PREVIOUS WORKS

Device		Slices	Data-path	BRAM	Freq(MHz)	Thr (Gbps)	Ref
Spartan3	Bulens	1800	128	0	150	1.7	6
	Proposed	1274	128	0	254	.08	our
Virtex E	Standaert	2257	128	0	169	2	7
	Proposed	1295	128	0	159	.05	our

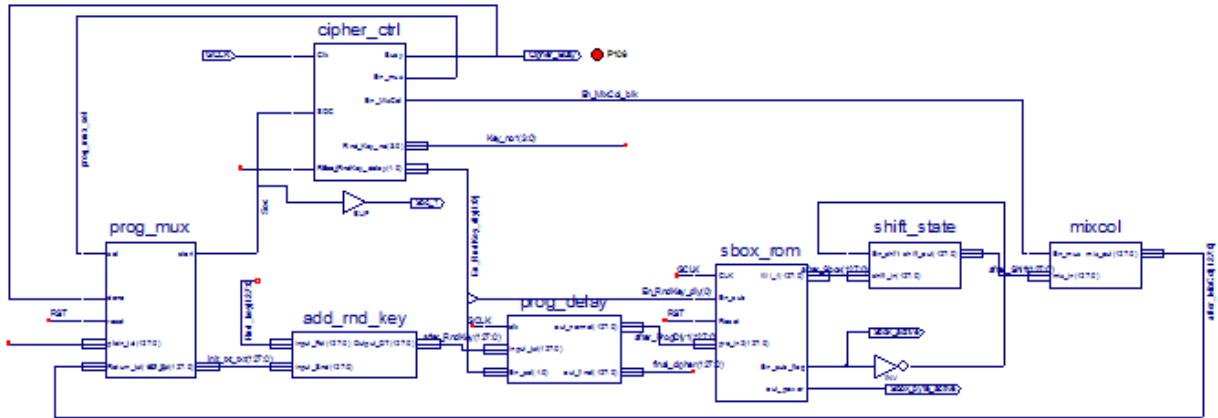


Fig. 8 The Block Diagram of proposed Algorithm