International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 3, Issue 11, November 2014

AES-128 KEY EXPANSION IMPLEMENTED ON FPGAUSING TWO APPROACHES

Vengadapathiraj.M[1] Rajendhiran.V[2] Gururaj.M[3] Mohamed nizar.S[4] Gomathi.R[5]

ABSTRACT—In this paper two approaches is used for theimplementation of AES-128 key expansion. In one approach, look-up table is used for the Sub Word transformation while the combinational logic based on Galois field arithmetic is used in another. Both implementations have data path of 32 bits. Implementation with look-up table attains throughput of 8Gbps while occupying 793 slices and implementation With combinational logic for Sub Word attains throughput of 5.8Gbps with only 402 occupied slices. These results are obtained after the implementation of designs on Virtex 4 (xc4vlx40-12ff1148) FPGA using Xilinx ISE v9.1. Timing simulation is performed on Modelsim SE 6.2c.

Keywords-AES, S-Box, Galois Field, Key Expansion

I.INTRODUCTION

Advanced Encryption Standard (AES) is a cryptographic algorithm which is accepted as FIPS standard by National Institute of Standards and Technology (NIST) in November 2001. It is a symmetric key block cipher. It can encrypt and decrypt 128 bit data with 128 or 192 or 256 bit key. AES has broad range of applications including smart cards, cellular phones, Automated Teller Machines (ATMs), Smart Phones, Digital Video Recorders, etc. AES can be implemented on hardware using various approaches depending on the need of application. There is always a trade-off between area occupied by hardware, speed of operation and power consumption. Many hardware implementations of AES have been proposed till date.

The block and key can in fact be chosen independently from 128,160,192,224,256 bits and need not be the same. However, the AES standard states that the algorithm can only accept a block size of 128 bits and a choice of three keys - 128,192,256 bits. Depending on which version is used, the name of the standard is modified to AES-128, AES-192 or AES-256 respectively. As well as these differences AES differs from DES in that it is not a feistel structure. Recall that in a feistel structure, half of the data block is used to modify the other half of the data block and then the halves are swapped. In this case the entire data block is processed in parallel during each round using substitutions and permutations. A number of AES parameters depend on the key length. For example, if the key size used is 128 then the number of rounds is 10 whereas it is 12 and 14 for 192 and 256 bits respectively. At present the most common key size likely to be used is the 128 bit key. This description of the AES algorithm therefore describes this particularimplementation.

ISSN: 2278 – 909X

A fully pipelined FPGA implementation for algorithm acceleration has been proposed [2]. It achieves throughput of 21.2Gbps for encryption and 16.6Gbps for decryption. Another fully subpipelined architecture with 6 sub stages in encryption and key expansion achieves very high throughput of 21.6Gbps and 43.71Gbps on Xilinx XCV1000e-8bg560 and XC2VP20-7fg676 devices, respectively [3].A Balanced hardware implementation is proposed considering several existing methods [2]. It achieves Throughput of 2.33Gbps on Quartus II family of Stratix family for AES-128 encryption.

In this paper, we are presenting two implementations of AES-128 key expansion. One implementation uses look-up table called as S-Box, containing beforehand calculated values of byte substitution transformation (SubByte) in AES. In AES Key Expansion procedure, this transformation is termed as"SubWord" . Other implementation uses combinational logic based on Galois Field arithmetic which calculates S-Box values on-the-fly. First method achieves better throughput but occupies more area than the second method.Rest of the paper is organized as follows. Section II describes AES algorithm. Section III describes our implemented architectures and Section IV shows implementation results.

II. AES ALGORITHM

The AES is a private key block cipher that processes data blocks of 128 bits with key length of 128, 192, or 256 bits. The AES algorithm's operations are performed on a 2-D array of 4 times 4 bytes called the State. The initial State is the plaintext and the final State is the cipher text. The State consists of 4 rows of bytes. As the block length is 128 bits, each row of the State contains 4 bytes.The four bytes in each column form a 32 bit word. After an initial round key addition, a round function consisting of four transformations - SubBytes, ShiftRows, MixColumns, and AddRoundKey is applied to each data block. The round function is applied 10, 12, or 14 times depending on the key length. AES-128 applies the round function 10 times, AES-192 - 12 times, and AES-256 - 14 times. The transformations are reversible linear and non-linear operations to allow decryption using their inverses. Every transformation affects all bytes of the State. The transformation SubBytes is a nonlinear byte substitution that operates on each byte of the State using a table (S-box). The numbers of the table is computed by a finite field inversion followed by an affine transformation. The resulting table is called an S-box. The ShiftRows transformation is a circular shifting operation, which rotates the rows of the State with different numbers of bytes (offsets). The offset equals to the row index: the second row is shifted one byte to the left, the third row - two bytes to the left, the fourth row - three bytes to the left, and first row - four bytes to the left. MixColumns trans-formation mixes the bytes in each column by multiplying the State with the polynomial modulo $x^4 + 1$. The State bytes are the coefficients of the polynomial. The AddRoundKeytransformation is an XOR operation that adds the round key to the State in each round. The round keys are generated during the key expansion process. The initial round key equals to secret key (Zhang and Parhi, 2002; Su *et al*., 2003; Hsiao *et al*., 2005; Feldhofer*et al*., 2005).

AES is an iterative cipher meaning that both encryption and decryption consist of multiple iterations of the same basic round functions, as shown in Fig. 1.

It achieves throughput of 21.2Gbps for encryption and 16.6Gbps for decryption. Another fully sub pipelined architecture with 6 sub stages in encryption and key expansion achieves very high throughput of 21.6Gbps and 43.71Gbps on Xilinx XCV1000e-8bg560 and XC2VP20-7fg676 devices, respectively [3].A Balanced hardware implementation is proposed considering several existing methods [2]. It achieves Throughput of 2.33Gbps on Quartus II family of Stratix family for AES-128 encryption.

ISSN: 2278 – 909X

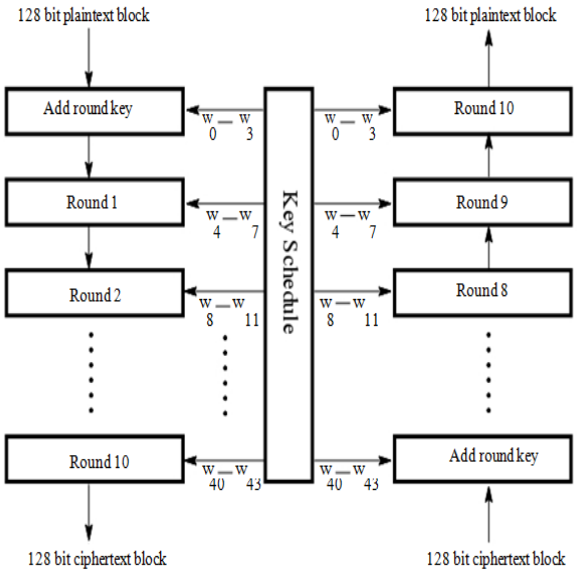

Figure 2.The overall structure of AES for the case of 128-bit encryption key

II.1 State

AES breaks data into states or matrix of bytes of predetermined size and encrypt every state independently of each other. Putting together bytes into a state has no cryptographic significance, yet it is an important process without whom other operations cannot be conducted. In Rijndael with a 128-bit key, an array or a matrix with 4 rows and 4 columns is formed where each entry is a byte or 8 bits so that there are essentially 16 bytes in total. Additionally, we may also think of a 128-bit state space as a vector in $gf(2^8)^{16}$ where each component of the vector is a byte.

II.2SubBytes

In this method each byte, each element in the matrix is replaced using the Ri-jndael's S-Box. This method is broken down into two stages. In the first stage each byte is replaced with its multiplicative inverse. In case of a byte whose value is 0 which does not have a multiplicative inverse, the byte remains 0.The second stage involves performing invertible affine transformation on each byte. In this case, if $x = x_0x_1x_2 \ldots x_7x_8$ is a vector whose elements are the binary representation of the byte in ascending order of power, then the output would be $A \cdot x + b$ where A is an 8×8 matrix and b is a vector which represents the coefficients (again, in ascending order) of some constant.

II.2 Shiftrows

In the *ShiftRows* transformation a cyclic shift with different constant offsets is applied to all rows except the first row of the state matrix.ShiftRows is perhaps the simplest operation to scramble data. Like the name suggest, ShiftRows shifts row n to the left by n−1 unit, so that the first row remains unchanged.

II.3MixColumns

The next method messes with columns instead of rows. MixColumns takes each column in the state and perform linear transformation on it. The *MixColumns* transformation operates on the State column-by-column. The columns are considered as polynomials over GF (2^8) and multiplied with a fixed polynomial modulo x^4+1.

II.4 AddRoundKey

AddRoundKey is the most important stage as it is the stage that provides uniqueness to the encryption thus is inevitably a complex operation by it-self. The output of AddRoundKey fully depends on the key or the password specified by the user. In this stage a subkey, which is the same size as the state, is computed from the main key using Rijndael's Key Schedule. Once a subkey is derived, the sum of the subkey and the state is calculated. This process consists of three parts: Rotate,Rcon, and SubByes. The first part is to rotate or to shift the bytes that form the keyword 8 bits to the left, which is similar to what happens to the second row in ShiftRows. The second part is to apply sub-operation called Rcon; And the third part is to perform Ri-jndael's S-Box whose details have been dissected in SubBytes. Another step of this operation is to expand the main key until we have enough subkeys.

ISSN: 2278 – 909X

II.5 AES ENCRYPTION ANDDECRYPTION

AES Encryption begins with addition of initial key with plaintext followed by round function consisting above four transformations. Key expansion procedure generates round keys from the initial key which are added with round data in AddRoundKey transformation. Decryption process of AES basically performs the inverse of each transformation in encryption in reverse order. Last round of encryption does not contain MixColumn transformation and first round of decryption does not contain InverseMixColumn transformation.

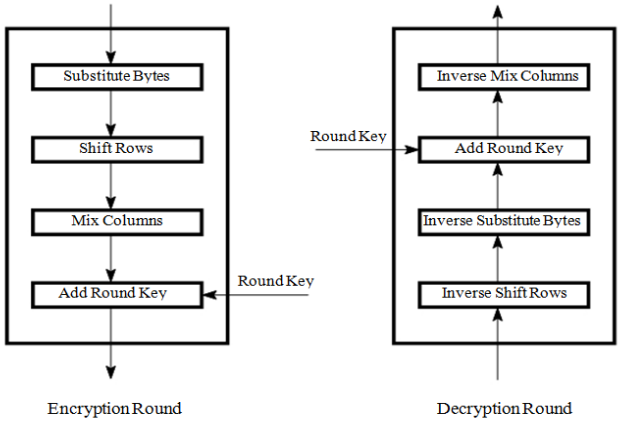

Figure 3: One round of encryption is shown at left and one round of decryption at right.

There are four steps in eachround of processing

STEP 1: (called subbyte for byte-byte substitution during the forward process) (the corresponding substitution step used during decryption is called invsubbytes).

This step consists of using a 16x16 lookup table to find a replacement byte for a given byte in the input state array.

The entries in the lookup table are created by using the notions of multiplicative invrses in GF(2^8) and bit scrambling to destroy the bit level correlations inside each byte.

STEP 2: (called shiftrows for shifting the rows of the state array during thr forward process) (The corresponding transformation during decryption is denoted invshiftrows for inverse shiftrow transformation).

The goal of this transformation is to scramble the byte order inside each 128-bit block.

Step 3:(called mixcolums for mixing up of the bytes in each column separately during the decryption is denoted invmixcolumn and stands for inverse mix column transformation).the goal is here is to further scramble up yhe 128-bit input block.

Step 4: called addroundkey for adding the round key to the output of the previous step during the forward process) (The corresponding step during decryption is denoted invaddroundkey for inverse add round key transformation).

III.IMPLEMENTED ARCHITECTURES FOR KEY EXPANSION

Key expansion can be carried out in two ways. All round keys can be generated and stored in memory or they can be generated on-the-fly while encryption. The design proposed in [2] uses the same encryption/decryption architecture during key expansion thus completely eliminating the need of a separate hardware unit for key scheduling. In [2], key expansion is executed by purely combinational logic. It is done under the assumption that extended key calculation for consecutive round is faster than single round execution time. In [3], key expansion architecture is divided same as the number of existent sub stages in encryption unit. Hence, round key generation can be done simultaneously with encryption operation. In [1], two separate registers are used along with block RAM to store first and last round key.

The SubWord operation involved in the key expansion procedure can be implemented as look up table (LUT), which is a traditional method [9]-[15]. But this approach puts a limitation on speed and also increases area.

ISSN: 2278 – 909X

Hence, alternative way is to use combinational logic for on the fly calculation of S-Box values. This approach is exploited in [2]-[9] with 7 subpipelining stages. Fig. 2 shows the block level diagram for our implementation of Key Expansion. Data path is of 32 bits.

A. Key Expansion using LUT for S-Box

In this approach, S-Box look-up table is used for SubWord operation. The data path is of 32 bits and four instances of S-Box are required at a time, after every four clock cycles.

Initially, cipher key is loaded into four 32-bit registers W0, W1, W2 and W3. For generating W4, W3 is applied with transformations RotWord(), SubWord() and then XORed with RCon and this transformed word is XORed with W0. Thus, in every clock cycle, a 32-bit word is generated by XOR of two 32-bit words. So, at the end of four clock cycles get 128 bit.

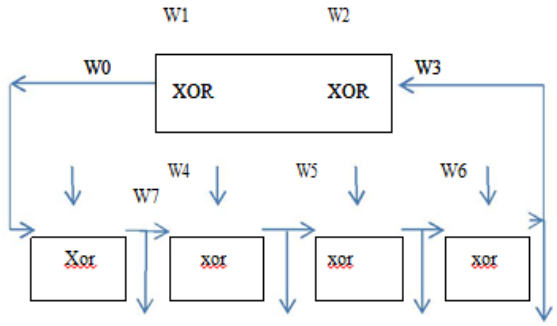

Figure 2: Implemented Key Expansion Module

B. KEY EXPANSION USING ON THE FLY S-BOX CALCULATION

This approach uses same architecture as shown in Fig. 2, only the method used for performing SubWord operation is different in this case. The combinational logic based on composite field arithmetic is used to calculate multiplicative inverse in GF(2^8) which is the first step in calculation of S-Box. And later, the affine transformation is applied to this multiplicative inverse to obtain S-Box value. Derivation of this logic is explained in detail in [6]. Fig. 3 shows the blocks of the combinational logic. In this approach, we don't require separate memory to store S-Box values. The blocks shown in Fig. 3 are implemented using Boolean equations [6] considering value of $\lambda = \{1100\}_2$ and $\varphi = \{10\}_2$

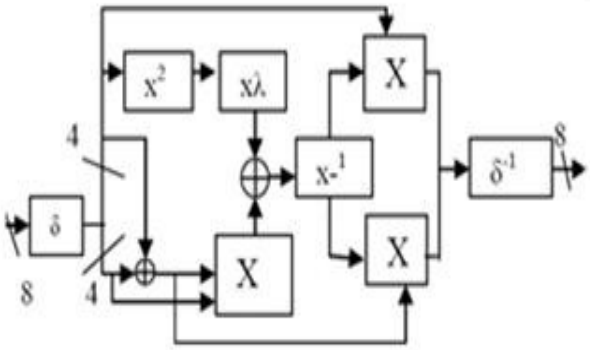

Figure 3: On-the-fly S-Box Calculation Logic

IV. IMPLEMENTATION RESULTS AND COMMENTS

The hardware is described using VHDL codes. It is synthesized and implemented on FPGA Virtex 4 device xcvlx40-12ff1148 using Xilinx ISE 8.1. The first design requires a minimum clock period of 4ns, while the second design requires minimum clock period of 5.5ns. Both the designs generate all 10 round keys in 40 clock cycles. The throughput can be calculated as follows: Where, Tp: Throughput (Mbps)

B: Number of bits generated

F: Clock frequency (MHz)

N: Number of clock cycles required to generate B Bit

The implemented hardware generates 128 bit round key in four clock cycles. Hence B = 128 and N = 4. Thus the throughput of first approach is 8000Mbps and that of second approach is 5817Mbps. Implementation results from Map Report are shown in Table I. As we can observe from Table I, LUT based approach offers more throughput at the cost of increased area. On the fly S-Box calculation approach is highly preferable for the applications which run at slightly less speed but have tight area constraints. Implementation results show that this approach requires almost 50% less number of slices than the LUT based approach. It also offers better Throughput/Slice ratio.

ISSN: 2278 – 909X

Further, we can use sub pipelining in the combinational logic for S-Box calculation to increase the throughput.

Parameter	1st Approach (LUT based S-box)	2nd Approach (On the fly S-Box)
Clock Frequency	250MHz	182MHz
Throughput	8Gbps	5.8Gbps
No. of occupied Slices	793 (4%)	402 (2%)
No. of slice FFs	405 (1%)	135 (1%)
4-Input LUTs	920 (2%)	718 (1%)
Peak Memory Usage	301MB	296MB
Gates	11128	6433
Throughput/Slice (Mbps/Slice)	10.088	14.4278

Table I: Implementation Results

V. CONCLUSION

This paper presented two design approaches for the implementation of AES-128 key expansion procedure. First approach uses LUT based S-Box and 2nd approach Calculates S- Box values on the fly. The implementation results show that on the fly S-Box calculation approach is better in terms of device utilization than the LUT based approach. Its throughput is 5.8Gbps which is less than that of LUT based approach which offers throughput of 8Gbps. But, on the fly S-Box calculation approach uses only 402 slices, 135 slice FFs, 718 LUTs, 6433 gates. While the LUT based approach uses 793 slices, 405 slice FFs, 920 LUTs and 11,128 gates. Second approach also offers Throughput/Slice ratio of 14.4278 which is better than first approach, 10.088.

REFERENCES

1. S. Yang, H. Li, X. Zhang, G. Zhao, "Celerity Hardware Implementation of the AES With Data Parallel and Pipelining Architecture Inside the Round Function", IEEE ICTSPCC 2013.

2. X. Zhang, H. Li, S. Yang, S. Han, "On a High-performance and Balanced Method of Hardware Implementation for AES", ICSSRC 2013.

3.H. Samiee, R. E. Atani, H. Amindavar, "A Novel Area-Throughput Optimized Architecture for the AES Algorithm", ICEDSA 2011.

4.S. M. Farhan, S. A. Khan, H. Jamal, "An 8-bit Systolic AES Architecture for Moderate Data Rate Applications", Microprocessors and Microsystems 33 (2009), 221-231.

5.A.Gielata, P.Russek, K.Wiatr "AES Hardware Implementation in FPGA for Algorithm Acceleration Purpose",ICSES 2008

6.N. Nedjah, L. M. Mourelle, M. P. Cardoso, "A Compact Pipelined Hardware Implementation of the AES-128 Cipher" IEEE ITNG 2006.

7.D. Canright, "A very compact S-Box for AES", CHES Volume 3659, Springer 2005.

8.Zhang, K. K. Parhi, "High-speed VLSI architectures for the AES algorithm", IEEE Trans. VLSI Systems, Vol. 12,Iss. 9, Sept. 2004.

9. A. Hodjat, I. Verbauwhede, "A 21.54 Gbits/s Fully Pipelined AES Processor on FPGA", IEEE Sypmosium on FCCM‟ 04), April 2004.

10. M. McLoone and J. V. McCanny, "Rijndael FPGA implementation utilizing look-up tables", in IEEE Workshop on Signal Processing Systems, Sept. 2001.

11. Specification for the Advanced Encryption Standard (AES) Federal Information Processing Standards(FIPS)Publication197 http://csrc.nist.gov/encryption/aes/frn fips197.pdf (Nov -2001).

12.H. Kuo and I. Verbauwhede, "Architectural optimization for a 1.82 Gbits/sec VLSI implementation of the AES Rijndael Algorithm", in Proc. CHES 2001, Paris, France, May 2001.

ISSN: 2278 – 909X

13.V. Fischer and M. Drutarovsky, "Two methods of Rijndael implementation in reconfigurable hardware", in Proc. CHES 2001, Paris, France, May 2001.

14.Edwin NC Mui, "Practical implementation of S-Box using combinational logic", Texco Enterprise Pvt. Ltd.

15.M. McLoone and J. V. McCanny, "Rijndael FPGA implementation utilizing look-up tables", in IEEE Workshop on Signal Processing Systems, Sept. 2001.

BIOGRAPHY

Vengadapathiraj.MReceived the B.E degree inElectronics and communication engineering from governmentCollege of engineering Salem,tamilnadu and currently pursuing M.E degree in Applied electronics from IFETcollege of engineering Villupuramtamilnadu . His current researchinterest includes digital electronics and microprocessor and microcontroller.

Rajendhiran.VReceived the B.E degree inElectronics and communication engineering from E.S college of engineering Villupuram, tamilnadu and currently pursuing M.E degree in Applied electronics from IFETcollege of engineering Villupuramtamilnadu. His current researchinterest includes digital electronics and digital image processing.

Gururaj.MReceived the B.E degree inElectronics and communication engineering from GovernmentCollege of engineering Salem,tamilnaduHis current researchinterest includes digital electronics and microprocessor and vlsi.

Mohamed nizar.S working as an AssistantProfessor in the Department of Electronics andCommunication Engineering at IFET College of Engineering Tamilnadu, India.Obtained B.E., fromVRS College of engineering and technology, Tamil nadu, M.E., in SathyabhamaUniversity, Tamil Nadu.Having 7 years of teaching and 2 years of Industrial experience. Hisareas of interest are High Performance Networking andDigital image processing.

Gomathi.R working as an AssistantProfessor in the Department of Electronics andCommunication Engineering at IFET College of Engineering Tamilnadu, India. Obtained B.E from Idhaya Engineering College for women, china Salem, Tamil nadu, M.E., in S.K.P Engineering College, Tamil Nadu. Having 5years teaching experience. Herareas of interest are Computer Networks Microprocessor and microcontroller and Communication Theory.

ISSN: 2278 – 909X