Compact FPGA Implementation of 32-bits AES Algorithm Using Block RAM

Chi-Wu Huang[1], Chi-Jeng Chang[1], Mao-Yuan Lin[1], Hung-Yun Tai[2]
*1 Institute of Applied Electronics Technology *2 Department of Industrial Education
National Taiwan Normal University

Abstract-**Hardware implementation of Advanced Encryption Standard (AES) algorithm has been in intensive discussion since its first publication by National Institute of Standards and Technology (NIST) in 2000, especially in high throughput over 1 Giga bits per second (Gbps). However, the studies of low area, low power and low cost implementations, which usually have throughput less than 1Gbps and use the datapath less than 32-bit, have been appearing recently in ASIC as well as in FPGA for wireless communication and embedded hardware application. This paper proposes a 32-bit datapath implementation in small Xilinx FPGA Chip (Spartan-3 XC3S200). It uses 148 slice, 11 Block RAMs (BRAMs) and achieves the data stream of 647 Mega bits per second (Mbps) at 287 MHz working frequency. It obtains 3.4 times improvement to the best known similar design in terms of ratio throughput per area (Throughput/Area), and 20% smaller in slice area.**

I. INTERODUCTION

Since the National Institute of Standards and Technology (NIST) selected the Rijndael Algorithm as a new Advanced Encryption Standard[1] in 2000, many hardware implementation of AES for high-throughput design[2][3][4][5] have been proposed. They typically unroll the loops with AES algorithm followed by pipelining of 128-bit data-path to achieve the order of tens Giga bit per second (Gbps). These designs are used mostly for high end devices such as accelerator cards for e-commercial services and secure trunk communication.

However, the 32-bit AES implementation started presenting after 2003 for low area and lower throughput applications. Such as PDAs, wireless network and embedded devices. Chodowiec [2] and Rouvroy [3] both have their 32-bit AES implementations. Table II in Section 4 has more detail.

This paper proposes a 32-bit AES implementation on Xilinx apartan-3(XC3S200) using 148 slice, 11 BRAMs and achieving 647 Mbps at 287MHz working frequency, it has throughput 3 times faster, 20% smaller slice area, and 3.4 times up in throughput/Area than those of [7].

Our implementation shifts the most operations of ShiftRow and KeyExpansion to BRAMs. BRAMs are also used a dual-port ROM lookup tables for Sbox/InvSbox and the multiplication tables (2xbi,4xbi,8xbi), which are needed in MixColumn/InvMixColumn. BRAMs are static RAM distributed around the configuration Logic Block (CLB) in FPGA chip. The size of CLB is expressed by the number of slice. This design keep the slice number as low as 148, only the exclusive-or gates and some counters and control circuits are left in CLB area, the less circuit in CLB results in less critical path delay and gains high clock frequency at 287 MHz. The proper use of BRAMs in this FPGA help save 20% slice area and gain 3 times faster throughput than the design detailed in [7].

The paper is organized and follow; A brief AES algorithm is described in Section 2 ; Section 3 describes our AES implementation; Section 4 describes the overall system and performance comparison; final conclusion is made in Section 5.

II. AES ALGORITHM

2.1. Encryption/Decryption

It's only a brief description of AES in this section, the details is freely available in NIST [1]. Briefly, this block cipher can perform encryption/decryption using repeated operations of a Substitute Permute Network (SPN) on 128-bit data. A different keys (named RoundKey) are supplied at each SPN iteration(or round). RoundKey is generated by an algorithm called KeyExpansion. The 128 data block encryption/decryption algorithm is summarized in Fig. 1, where Nr is the number of round. The first and the final round differ from the middle rounds as can be seen in Fig. 1. RoundKey for the encryption/decryption is the same except in reverse order.

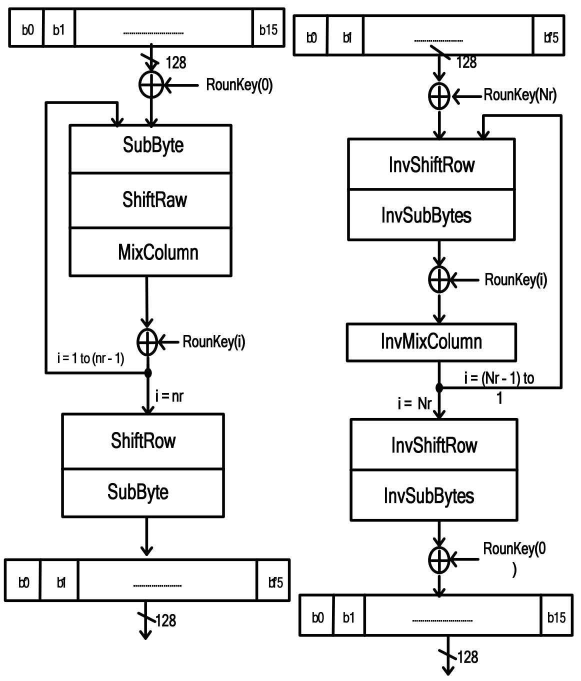

Figure 1. Round structure of AES

2.2. KeyExpansion

Fig. 2 shows four 32-bit registers k0,k1,k2 and k3 for a total of 128-bits KeyExpansion, where key0,key1,key2 and key3 are the key inputs that will be expanded to $k0', k1', k2', k3'$ through 10 rounds, this process is called KeyExpansion, k3 needs to go through "$Sbox$" and "$rcon$" before adding to k0.

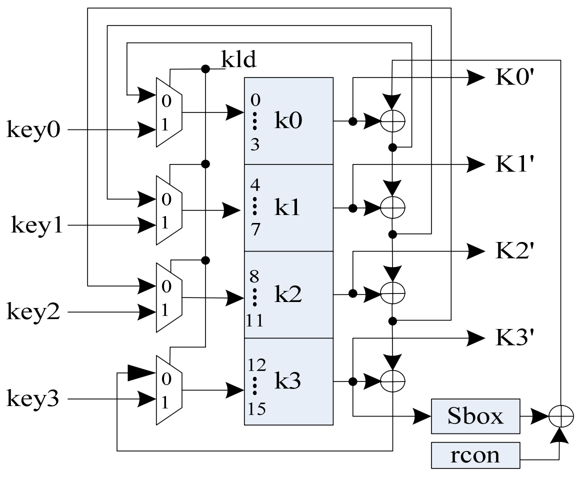

Figure 2. Combined forward/inverse key expansion

2.3. MixColumn

A column vector (b0,b1,b2,b3) is transformed to (b0',b1',b2',b3') by multiplied an constant vector m(x), where m(x)= $\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ is called MixColumn while InvMixColumn is multiplied by m'(x), where m'(x)= $\{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$. Both transformations will be more detailed in section 3.4.

III. PROPOSED AES IMPLEMENTATION

3.1. General architecture of 32-bit AES implementation

Fig.3 shows a 32-bit AES configuration, operating in 4 clocks per round by time multiplexing the 128-bit operation. ShiftRow is accomplished by proper connection between two 16x8 registers S and M. MixColumn is performed after ShiftRow. Adding (exclusive OR) the outputs of MixColumn and KeyExpansion, let the results go through Sbox. It is then ready for starting the next round.

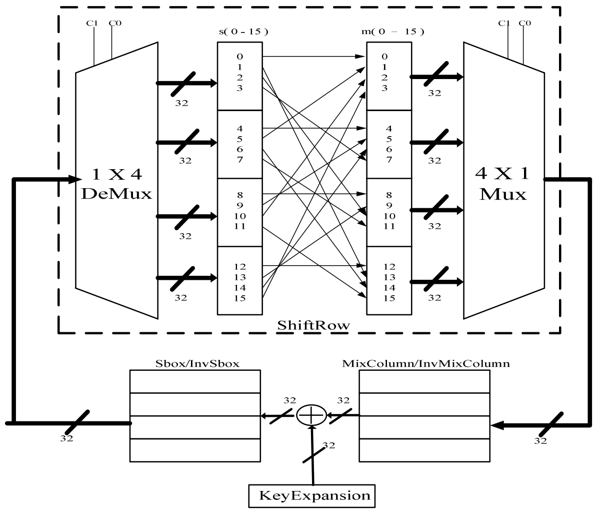

Figure 3. 32-bit datapath, 4 clock per round AES configuration.

3.2. Modified 32-bit AES implementation using BRAMs

Fig. 4 shows a 32-bit AES implementation modified from Fig.3, where registers S and M in Fig.3 are replaced by 4 BRAMs, and Sbox/InvSbox that become a lookup table (including Sbox and InvSbox) in 2 dual-port BRAMs(one dual-port lookups 2 bytes, 2 dual-port lookup 4 bytes). Vector multiplications needed for MixColumn /InvMixColumn are accomplished by three kinds of BRAM lookup tables 2b$_i$, 4b$_i$, and 8b$_i$. A 4×32 key register needed by KeyExpansion is also replaced by a 4×32 BRAM K. The gray-dotted areas represent BRAM in FPGA, while white areas are in CLB, where the size of area is counted by the number of slice.

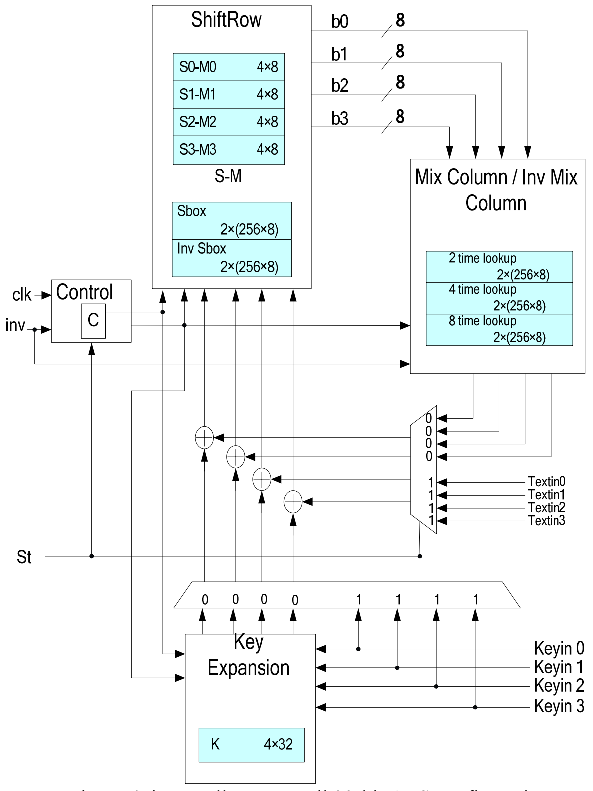

Figure 4. is actually an overall 32-bit AES configuration gray-dotted areas are BRAMs

Signal 'st' is system start signal, 'ck' is system clock, signal "enc" is for encryption or decryption selection, Register C is a 2-bit counter offering necessary BRAM addresses, which will be detailed in the following sections. Control block in Fig. 4 is for controlling sequence differences between encryption and decryption, the first and last round of AES operations, and etc. The details of control are not shown for simplicity.

3.3. ShiftRow operation in BRAM

As shown in Fig.3 ShiftRow is the input and output connections between registers S and M according to AES requirement. Now in Fig.5 , the registers S and M are replaced by 4 BRAMs, namely S0-M0, S1-M1, S2-M2, S3-M3. (S0,S1,S2,S3) is at address from 0 to 3 while (M0,M1,M2,M3) is at address from 4 to 7 of the 4 BRAMs in Fig.2.

ShiftRow is accomplished by sending proper addresses to the BRAMs. It is observed from Table 1-(b), two indices of the ShiftRow matrix can be used for the two addresses of dual-port BRAMs. All the first indices are the same, which can be used for the addrA of 4 input ports, are initialized to '0' (or C) while the second index for the output ports are initialized to '0', '1', '2', '3' (or C,C+1,C+2,C+3) as shown in Fig. 5.

The 4-byte data read from (S0, S1, S2, S3) is MixColumned and Stored back to (M0, M1, M2, M3). After 4 cycles of such operation, a round is completed and next round will be switched over to read from (M0,M1,M2,M3) and write back to (S0,S1,S2,S3) triggering by round switch signal (rsw). Signals rsw and ($\overline{rsw}$) are input to the most significant bit of addrA and addrB for such switching operation.

Similar observation form Table 1-(c) for InvShiftRow, adderB are initialized to '0','3','2','1', but not shown in Fig. 3 for simplicity.

Table I
SHIFTROW/INVSHIFTROW OPERATIONS FORM THE ORIGINAL MATRIX

S_{00}	S_{01}	S_{02}	S_{03}	S_{00}	S_{01}	S_{02}	S_{03}	S_{00}	S_{01}	S_{02}	S_{03}
S_{10}	S_{11}	S_{12}	S_{13}	S_{11}	S_{12}	S_{13}	S_{10}	S_{13}	S_{10}	S_{11}	S_{12}
S_{20}	S_{21}	S_{22}	S_{23}	S_{22}	S_{23}	S_{20}	S_{21}	S_{22}	S_{23}	S_{20}	S_{21}
S_{30}	S_{31}	S_{32}	S_{33}	S_{33}	S_{30}	S_{31}	S_{32}	S_{31}	S_{32}	S_{33}	S_{30}

(a) original (b) affer ShiftRow (c)after InvShiftRow

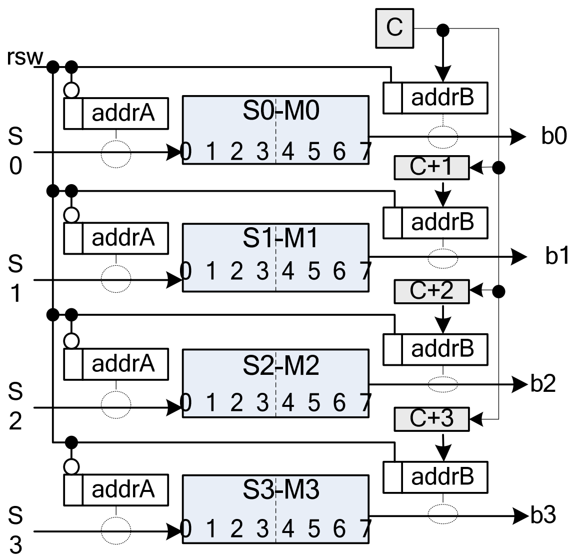

Figure 5. ShiftRow operations controlled by dual-port adderss addrA,addrB gray-dotted areas are 4 BRAMs

3.4. MixColumn/InvMixColumn and Multiplication tables

If $(b0', b1', b2' b3')$ and $(i0, i1, i2, i3)$ represent the forward and inverse MixColumn, respectively. Then $(i0, i1, i2, i3)$ can be expressed in terms of $(b0', b1', b2', b3')$, as mentioned by Satoh [3], shown as follow;

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \otimes \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$= \begin{bmatrix} (2b_0 + 2b_1) + (b_1 + b_2 + b_3) \\ (2b_1 + 2b_2) + (b_2 + b_3 + b_0) \\ (2b_2 + 2b_3) + (b_3 + b_0 + b_1) \\ (2b_3 + 2b_0) + (b_0 + b_1 + b_2) \end{bmatrix} \text{---------------} (1)$$

$$\begin{bmatrix} i_0 \\ i_1 \\ i_2 \\ i_3 \end{bmatrix} = \begin{bmatrix} e & b & d & 9 \\ 9 & e & b & d \\ d & 9 & e & b \\ b & d & 9 & e \end{bmatrix} \otimes \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$= \left(\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} + \begin{bmatrix} 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 \\ 8 & 8 & 8 & 8 \end{bmatrix} + \begin{bmatrix} 4 & 0 & 4 & 0 \\ 0 & 4 & 0 & 4 \\ 4 & 0 & 4 & 0 \\ 0 & 4 & 0 & 4 \end{bmatrix} \right) \otimes \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$= \begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix} + \begin{bmatrix} 8b_0 + 8b_1 + 8b_2 + 8b_3 \\ 8b_0 + 8b_1 + 8b_2 + 8b_3 \\ 8b_0 + 8b_1 + 8b_2 + 8b_3 \\ 8b_0 + 8b_1 + 8b_2 + 8b_3 \end{bmatrix} + \begin{bmatrix} (4b_0 + 4b_2) \\ (4b_1 + 4b_3) \\ (4b_0 + 4b_2) \\ (4b1 + 4b_3) \end{bmatrix} \text{--------} (2)$$

If (m_0', m_1', m_2', m_3') represents the output of MixColumn/InvMixColumn, then it can be multiplexed by (b_0', b_1', b_2', b_3') or (i_0, i_1, i_2, i_3) as follow.

$$(m0', m1' m2' m3') = ((\mathrm{b0'}, \mathrm{b1'}, \mathrm{b2'}, \mathrm{b3}) \text{ and } (\mathrm{enc})) \; or$$
$$= ((i0', i1', i2', i3') \; and \; (\overline{enc})) \text{-------} (3)$$

To realize the expression (1),(2), and (3), list above. Lookup tables $2 \times b_i$, $4 \times b_i$, $8 \times b_i$ are established in BRAM as dual-port ROMs, which contain tables for the multiplication by 2, 4 and 8 respectively. The output of the ROM lookup tables are combined to form (b_0', b_1', b_2', b_3') and (i_0, i_1, i_2, i_3) according to the expression (1), (2), and (3), Fig.4 summarized the MixColumn/InvMixColumn circuit realization. There are 32-bit output form BRAM, because 4 outputs are needed for each b_i. For example, $2 \times b_i$ will generate $2 \times b_0$, $2 \times b_1$, $2 \times b_2$, $2 \times b_3$ which are 4 bytes and occupies 32 bits.

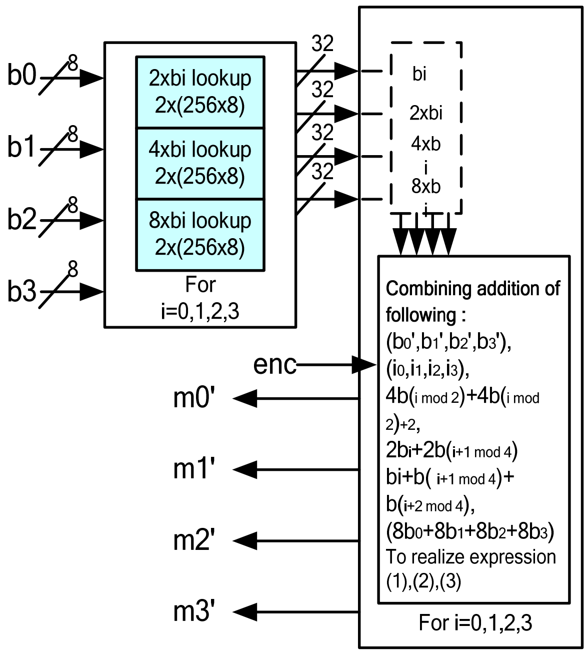

Figure 6. MixColumn/InvMixColumn circuit realization, gray-dotted area is in BRAM.

3-5 KeyExpansion

KeyExpansion shown in Fig. 2, the four registers k0,k1,k2,k3 can be replaced by a 4x3 dual-port BRAM to reduce the slice number. The dual-port BRAM (one write/read port and one read port), as shown in Fig. 7, addrA controls the input and output data (dinA/doutA), where read before write mode is used. Signal addrB controls the read of doubt. Both addrA and addrB are initialized by '0' and '3' (c+3), respectively. There result of (doutA ⊕ doutB) is the keyout, which is also the feedback input (dinA) to BRAM. DoutB goes through Sbox/Rcon only when addrB="3"

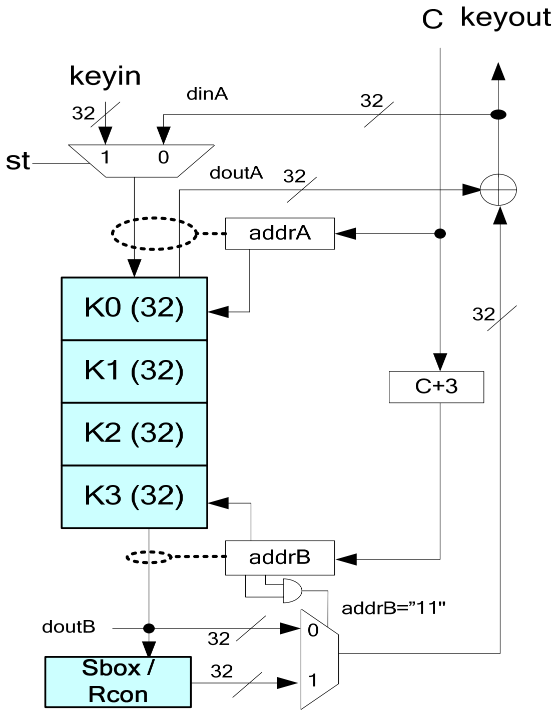

Figure 7. KeyExpansion using one 4×32bit BRAM, 2 BRAMs for Sbox and 1 BRAM for RCON

IV. OVERALL SYSTEM AND COMPARISION

Fig.2 shown the overall system configuration and summarized the use of BRAM. The AES hardware operations are shifted as many as we can to BRAMs that result in very low slice area and high throughput ever found for 32-bit AES implementation as shown in Table II.

TABLE II
PERFORMANCE COMPARISON BETWEEN FOUR AES IMPLEMENTATION

Design & FPGA / Items compared	Chodowiec Spartan-2 XC2S30 [6]	Rouvroy Spartan-3 XC3S50 [7]	Ours Spartan-3 XC3S200
Clock Frequency (MHz)	60	71	287
Datapath Bits	32	32	32
Slices	222	163	148
Block RAMs used	3	3	11
Throughput (Mbps)	166	208	647
Throughput per area (Mbps/slice)	0.75	1.27	4.37

V. CONCLUSION

AES implementation less than 32-bits is suitable for wireless, low area and embedded applications. By proper using BRAM feature in FPGA chip, this proposed implementation successfully reduces the slice to very small number at the expense of increasing BRAM numbers. Therefore, some techniques concerning efficient utilization of BRAMs might become issues for further investigation.

REFERENCES

[1] NIST. Announcing the advanced encryption standard(AES), FIPS 197. Technical report, National Institute of Standards and Technology, November 2001.

[2] X. Zhang and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 9, pp. 957–967, Sep. 2004.

[3] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware ArchitectureWith S-Box Optimization," in Proc. LNCS ASIACRYPT'01, vol. 2248, pp. 239–254, Dec. 2001.

[4] Alireza Hodjat, Ingrid Verbauwhede,"Minimum Area Cost for a 30 to 70 Gbits/s AES Processor", IEEE Computer society Annual Symposium on VLSI, 2004. Proceedings., Page(s):83 - 88, Feb. 2004.

[5] Ricardo Chaves, Georgi Kuzmanov, Stamatis Vassiliadis, Leonel Sousa,"Reconfigurable Memory Based AES Co-Processor", IPDPS 2006. 20th International Parallel and Distributed Processing Symposium, Page(s):8 pp, April 2006.

[6] Pawel Chodowiec, Kris Gaj,"Very Compact FPGA Implementation of the AES Algorithm", CHES 2003, LNCS 2779, pp. 319–333, 2003.

[7] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, J.-D. Legat,"Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications",Information Technology Coding and Computing, 2004. Proceedings. ITCC 2004, Volume 2, Page(s):583 - 587 Vol.2, 2004.

[8] Tim Good, Mohammed Benaissa, "Very small FPGA application-specific instruction processor for AES", IEEE Trans. Circuit and System,vol. 53, no. 7, 2006.