4.2 Gbit/s single-chip FPGA implementation of AES algorithm

F. Rodríguez-Henríquez, N.A. Saqib and A. Díaz-Pérez

A high performance encryptor/decryptor core of the advanced encryption standard (AES) is presented. The proposed architecture is implemented on a single-chip FPGA using a fully pipelined approach. The results obtained show that this design offers up to 25.06% less area and yields up to 27.23% higher throughput than the fastest AES FPGA implementations reported to date.

Introduction: The Rijndael block cipher algorithm was chosen by NIST as the new advanced encryption standard (AES) [1, 2]. The AES encryption and decryption processes' definitions are not identical [2]. This characteristic poses a challenge for efficient AES encryptor/decryptor core hardware implementations, as the implementation of a separate architecture for encryption and decryption processes would imply costly area requirements. However, while efficient hardware implementation was one of the main evaluation criteria for selecting the AES winner, few FPGA encryption designs have been reported to date [3–7], and not all of them implement encryptor/decryptor cores [7]. In this Letter a fully pipelined AES encryptor/decryptor core is presented. First, a description of the AES algorithm and proposed optimisations are discussed, and then performance results are presented and compared with previous reported designs.

AES algorithm: Fig. 1 shows the encryption architecture of the AES cipher algorithm for encrypting one 128-bit block of data using a 128-bit user key. The algorithm treats the input 128-bit block as a group of 16 bytes organised in a 4×4 matrix called *State* matrix. The algorithm consists of an initial transformation, followed by a main loop where nine iterations called *rounds* are executed. Each round is composed of a sequence of four transformations: Byte Substitution (BS), ShiftRows (SR), MixColumns (MC) and AddRoundKey (ARK). For each round of the main loop, a round key is derived from the original key through a process called *Key Scheduling*. Finally, a last round consisting of three transformations, BS, SR and ARK, is executed. The AES decryption algorithm operates similarly by applying the inverse of all the transformations described above in reverse order.

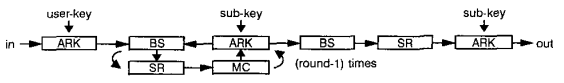

Fig. 1 *Basic algorithm flow*

Byte Substitution (BS): Each input byte of the *State* matrix is independently replaced by another byte from a look-up table called *S*-box. The AES *S*-box is a 256-entry table composed of two transformations: first each input byte is replaced with its multiplicative inverse in $GF(2^8)$ with the element {00} being mapped onto itself; followed by an affine transformation over $GF(2)$ [1, 2]. For decryption, inverse *S*-box is obtained by applying inverse affine transformation followed by multiplicative inversion in $GF(2^8)$ [2].

ShiftRows (SR): This is a cyclic shift operation where each row is rotated cyclically to the left using 0, 1, 2 and 3-byte offset for encryption while for decryption, rotation is applied to the right.

MixColumns (MC): In this transformation each column of the *State* matrix is multiplied by a constant fixed matrix as follows:

$$\begin{bmatrix} c'_{0,i} \\ c'_{1,i} \\ c'_{2,i} \\ c'_{3,i} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,i} \\ c_{1,i} \\ c_{2,i} \\ c_{3,i} \end{bmatrix} \quad (1)$$

where the two column vectors above represent the i-th column of the *State* matrix and the i-th column of the transformed MC *State* matrix, for $i = 0, 1, 2, 3$, respectively. Similarly, for the decryption process, we compute Inverse MixColumns, by multiplying each column of the *State* matrix by a constant fixed matrix as shown below

$$\begin{bmatrix} c'_{0,i} \\ c'_{1,i} \\ c'_{2,i} \\ c'_{3,i} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} c_{0,i} \\ c_{1,i} \\ c_{2,i} \\ c_{3,i} \end{bmatrix} \quad (2)$$

AddRoundKey (ARK): The output of MC is XOR-ed with the corresponding round sub-key derived from the user key. The ARK step is essentially the same for encryption and decryption processes.

Design: A pipelined approach for the AES algorithm implementation is shown in Fig. 2. Keys are generated and fed to each round. After 10 cycles, encrypted 128-bit data blocks appear at each clock cycle. As has been already mentioned, the encryption and decryption processes of AES are not symmetric. Therefore, the inverse of each step must be implemented separately for decryption. This approach is usually not practical as it implies the allocation of large amounts of memory. To alleviate the area requirements for an AES encryptor/decryptor core, we propose the following design optimisations for FPGA implementations.

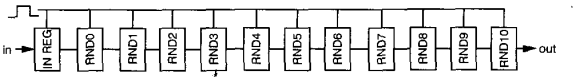

Fig. 2 *Pipeline approach for 128 bit Rijndael encryption*

S-box and Inverse S-box implementation: Both, the *S*-box and the inverse *S*-box, may be computed by implementing the affine (AF) and Inverse affine (IAF) transformations together with a look-up table for Multiplicative Inverse (MI). In this way, the combination MI + AF provides *S*-box for encryption, while IAF + MI computes the Inverse *S*-box needed for decryption. An additional multiplexer block is used to switch the data path for encryption/decryption. This approach is advantageous since the same look-up table can be reused for encryption and decryption.

MC and IMC Implementation: MC for encryption can be computed efficiently by using only three steps [1]: a sum step, a doubling step, and a final sum step. Let the elements of *State* matrix's column one be $a[0]$, $a[1]$, $a[2]$ and $a[3]$ and let $k[0]$, $k[1]$, $k[2]$ and $k[3]$ represent the first, second, third and fourth bytes of a 4-byte key block, respectively. Then the transformed MC column $a'[0]$, $a'[1]$, $a'[2]$ and $a'[3]$ can be efficiently obtained by computing

$$\begin{array}{llll} v = a[1] \oplus a[2] \oplus a[3] & xt_0 = xtime(a[0]) & a'[0] = k[0] \oplus v \oplus xt_0 \oplus xt_1 \\ v = a[0] \oplus a[2] \oplus a[3] & xt_1 = xtime(a[1]) & a'[1] = k[1] \oplus v \oplus xt_1 \oplus xt_2 \\ v = a[0] \oplus a[1] \oplus a[3] & xt_2 = xtime(a[2]) & a'[2] = k[2] \oplus v \oplus xt_2 \oplus xt_3 \\ v = a[0] \oplus a[1] \oplus a[2] & xt_3 = xtime(a[3]) & a'[3] = k[3] \oplus v \oplus xt_3 \oplus xt_0 \end{array} \quad (3)$$

where $xtime(v)$ represents the finite field multiplication of $02 \times v$, and 02 stands for the constant polynomial x in $GF(2^8)$. Notice also that in (3) we have embedded for efficiency reasons the ARK transformation into the MC step. The same strategy utilised above for MC would yield seven steps to compute IMC (four sum steps and three doubling steps). The complexity difference is due to the fact that coefficients in (2) have a higher Hamming weight than those in (1). We overcome this drawback by observing that it should exist as a 4×4 byte matrix $D(x)$ in $GF(2^8)$ such that,

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} D(x) \quad (4)$$

Using the fact that both constant matrixes in (4) are the inverse of each other in the field $F = GF(2^8)$ generated by the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$ [2], it can be easily proved that (4) has a unique solution in the finite field F given by: $d_{0,0} = 5$; $d_{1,0} = 0$; $d_{2,0} = 4$; $d_{3,0} = 0$, where $d_{i,0}$, $i = 0, 1, 2, 3$ represent the four coefficients of the first column of $D(x)$. Hence, (4) can be rewritten as,

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} 05 & 00 & 04 & 00 \\ 00 & 05 & 00 & 04 \\ 04 & 00 & 05 & 00 \\ 00 & 04 & 00 & 05 \end{bmatrix} \quad (5)$$

Equation (5) suggests an efficient way to compute IMC by reusing the MC transformation to obtain the IMC constant matrix. This is useful since the second matrix in the right side of (5) has considerably less Hamming weight than the original constant matrix for IMC. A small modification (ModM) is therefore introduced before MC + ARK steps for decryption. Fig. 3 shows the proposed architecture for AES encryption/decryption processes using the following data path,

Encryption: MI + AF + SR + MC + ARK
Decryption: ISR + IAF + MI + ModM + MC + ARK

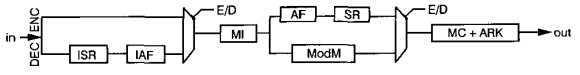

Fig. 3 *Proposed AES implementation for pipeline design*

Performance results: The design shown in Fig. 3 was implemented on an XCV2600E Xilinx Virtex-E FPGA device. The implementation occupies 80 BRAMs (43%), 386 I/O blocks (48%) and 5677 slices (22.3%). Table 1 details the total area required for each block shown in Fig. 3. The design uses a system clock of 34.2 MHz and the data is processed at a rate of 4121 Mbits/s.

Table 1: Area requirements for AES encryption/decryption

Block	CLB slices	BRAMs
KeyBlock	1278	
BS	—	80
SR	—	—
Affine transformation	80 × 10 = 800	
Inverse affine transformation	64 × 10 = 640	
MixColumn + ARK (combined)	152 × 9 = 1368	
Modification (for decryption)	120 × 9 = 1080	
ARK (1st + last round)	128	
Misc. (timing + I/O registers)	383	
Total	5677	80

As shown in Table 2, the results obtained compare quite favourably with existing FPGA implementations. The area requirements have been reduced up to 25.06%, while the expected throughput has been increased more than 27.23% compared to the fastest FPGA design reported to date [3].

Table 2: AES algorithm FPGA implementation

	Device	CLB Slices	Throughput (Mbits/s)
Ichikawa *et al.* [4]	VLSI	—	1950
Weeks *et al.* [5]	VLSI	—	5163
Lutz *et al.* [7]	VLSI	—	2260
Elbirt *et al.* [6]	XCV1000	9004	1940
McLoone and McCanny [3]	XCV3200E	7576	3239
This design	XCV2000E	5677	4121

Conclusions: A highly optimised fully pipelined AES encryptor/decryptor core architecture has been presented. The total number of slices required by the design are 5677, while achieved throughput is 4121 Mbits/s. Future work includes possible improvements of the design's performance figures by using different optimisation techniques.

Electronics Letters Online No: 20030746
DOI: 10.1049/el:20030746
2 April 2003

F. Rodríguez-Henríquez, N.A. Saqib and A. Díaz-Pérez (*Computer Science Section, CINVESTAV-IPN, Av. IPN No. 2508, 07360 México D.F.*)

E-mail: francisco@cs.cinvestav.mx

References

1 DAEMEN, J., and RIJMEN, V.: 'The design of Rijndael'. AES-The Advanced Encryption Standard (Springer-Verlag, Berlin, Heidelberg, New York, 2002)
2 TRAPPE, W., and WASHINGTON, L.C.: 'Introduction to cryptography with coding theory' (Prentice-Hall, Upper Saddle River, 2002)
3 MCLOONE, M., and MCCANNY, J.V.: 'High performance FPGA Rijndael algorithm implementations' (Springer-Verlag, 2001), CHES2001, LNCS 2162, pp. 65–76
4 ICHIKAWA, T., KASUYA, and MATSUI, M.: 'Hardware evaluation of the AES Finalists'. The Third Advanced Encryption Standard (AES3) Candidate Conference, New York, USA, April 2000
5 WEEKS, B., BEAN, M., ROZYLOWICZ, T., and FICKE, C.: 'Hardware performance simulations of Round 2 Advanced Encryption Standard Algorithms'. The Third Advanced Encryption Standard (AES3) Candidate Conference, New York, USA, April 2000
6 ELBIRT, J., YIP, W., CHETWYND, B., and PAAR, C.: 'A FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists'. The Third Advanced Encryption Standard (AES3) Candidate Conference, New York, USA, April 2000
7 LUTZ, K., *et al.*: '2 Gbit/s Hardware Realizations of RIJNDAEL and SERPENT'. A comparative analysis (Springer-Verlag, 2002), CHES2002, LNCS 2523, pp. 144–158

Low chromatic-dispersion flat-top arrayed waveguide grating filter

T. Kitoh, Y. Inoue, M. Itoh, M. Kotoku and Y. Hibino

A low chromatic dispersion (CD) flat-top arrayed waveguide grating (AWG) filter with a novel parabolic waveguide horn is proposed, and a 100 GHz-spacing 16-channel flat-top AWG that reduces the CD from −20.9 to −3.2 ps/nm has been successfully fabricated.

Introduction: Routing and add-dropping in the WDM layer have been developed to provide highly flexible advanced optical networks at reduced cost. In such networks, optical signals pass through multi/demultiplexers without being regenerated more frequently than in point-to-point WDM networks. The cumulative passband width of a channel becomes less than that of a single-stage multi/demultiplexer and chromatic dispersion (CD) is accumulated. This has led to a strong need for a multi/demultiplexer with a flat spectral response and low CD. Arrayed waveguide grating (AWG) multi/demultiplexers composed of planar lightwave circuits have been widely used as multi/demultiplexers in WDM systems [1–4]. An effective way of constructing an AWG with a wide passband is to use a configuration with a parabolic waveguide horn as the input waveguide to a slab waveguide [4]. This configuration provides good passband controllability and fabrication stability. While AWG multi/demultiplexers have no CD in principle [3, 5], the phase fluctuation in the arrayed waveguide region caused by fabrication errors and the parabolic waveguide horn has been reported to induce CD in AWGs [3, 5, 6]. This phase fluctuation can be reduced with precise fabrication techniques. However, the CD caused by the parabolic waveguide horn results from the theoretical mode conversion in the horn.

In this Letter, we propose a new structure for a parabolic horn with a multimode waveguide for zero dispersion and describe our successful fabrication of a flat-top AWG filter that made it possible to reduce the CD from −20.9 to −3.2 ps/nm by adjusting the multimode waveguide length.

Configuration: Fig. 1 shows the schematic configuration of a flat-top AWG filter with a parabolic waveguide horn in the input waveguide. The 0th and 2nd mode are excited in the parabolic waveguide horn, and a double-peaked rectangular optical field is formed at the end of the horn [4]. Fig. 2 shows the phase and amplitude distributions calculated by the beam propagation method for the parabolic horn (250 μm long and 28 μm wide).