FPGA-based Hardware Implementation of Compact AES Encryption Hardware Core

ATEF IBRAHIM
Department of Computer Engineering & Department of Microelectronics
Prince Sattam Bin Abdul-Aziz University & Microelectronics Research Institute
Kharj & Cairo
SAUDI ARABIA & EGYPT
attif_ali2002@yahoo.com

Abstract: - Most of current embedded applications need AES algorithm implementations of small size and low power consumption to assure safe information conveyance. In this article, we present the implementation of a compact ASE hardware encryption core that is suitable for resource-limited applications based on FPGA technology. The core has 8-bit data path structure and supports encryption with 128-bit keys. The core has been described using VHDL language. The simulation and synthesis results are obtained using ModelSim and Xilinx ISE software tools, respectively. This implementation is compared to the previously reported compact implementations in terms of speed, area, and consumed energy. The implementation results showed that the adopted design achieves significant reduction in area (up to 32.4%) and consumed energy (up to 66.7%). Also, it has a significant increase in speed by ratios ranging from 28.6%to 44.5%. This makes the adopted design more suitable for resource-limited embedded applications.

Key-Words: - Compact AES hardware implementation, Embedded systems, FPGA, VHDL, Low power hardware design, Hardware security.

1 Introduction

The use of cryptographic algorithms with the purpose of security is mostly linked to applications that require economization power preservation, such as Wireless LAN (WLAN), Wireless Sensor Networks (WSN), Personal Area Networks (PAN), and smartcards. In order for such algorithms to run with optimum performance and power consumption, hardware is preferred to software. Having been standardized and regarded as secure, Advanced Encryption Standard (AES) ended up being the prime choice in a multitude of applications, among which are the wireless standard technologies WPA2 (IEEE 802.11i) [1], IEEE802.15.4 [2], and ZigBee [3]. The implementation of FPGA-based AES encryption core that we adopt in this paper will meet the requirements of cheap and low-power applications.

It is established that hardware architectures with pipelines and unwound loops make it possible to realize very high-speed AES designs. However, they entail high space occupancy and high power consumption. Moreover, such architectures are not fully exploitable in modes of operation that use feedbacks [4, 5], which are most often used with the purpose of encryption and/or generation of MAC (Message Authentication Code); for example, as with the security plans described in [1-3]. Loop-back designs rather enable less-resource-consuming implementations usable at full-speed also in feedback modes (because they comprise a single processor that is fed with its own output over and over [5, 6]). Data path width can also be reduced so that logic area and power are minimized [4-11].

The presented AES core [12] has a byte of data path (8-bit) and it has the potential to perform encryption with keys of 128 bits length. For instance, the operation modes described in [1-3] demand only the AES key of 128 bits. That preserves hardware resources as the inverse and direct AES transformations are in part different. Compared to the previous 8-bit FPGA implementations of [4, 8, 20], the adopted core implementation achieves considerable reduction in area occupancy and energy consumption. Also, it increases speed significantly. This makes the adopted design more suitable for resource constrained embedded applications.

Here is the scope of work of this paper. Section 2 gives a short description of the algorithm of AES. Section 3 displays the compact AES implementations reported previously in the literature. Section 4 depicts the hardware structure of the AES core adopted here. Section 5 compares the implementation results of the adopted encryption core and preceding implementations. With section 6, we conclude this article.

2 Spotlights on AES

AES is considered a private-key algorithm [13]. It has three Rijndael cipher family members, each has 128-bits data block size, but three various lengths of key: 128, 192 and 256 bits, which be composed of 10, 12, or 14 repetition (iterations) cycles, respectively. Each iteration cycle scrambles plain data with a round key, which is derived from the cipher key. Decryption inverts the cycles of repetition bringing about, in part, a different data path.

Fig. 1 depicts the iterative operations of encryption. The cipher internally saves a 4-bytes × 4-bytes matrix, called State, by whose aid encryption processes are carried out. On the initial round, State is loaded with a block of input data and joined with the encryption key using a bitwise XOR operation. The rest of the rounds, except the last, comprise operations called Byte-Sub, Shift-Rows, Mix-Columns, and Add-Round-Keys. The final round is free of Mix-Columns. The encryption key length is what determines the number of iterations (rounds).

The **Byte-Sub** operation is a non-linear substitution where each byte of the State is separately substituted by another byte using a substitution table called S-box. The S-box is derived by computing modular multiplicative inverses over GF (2^8). **Shift-Rows** is an operation applied to each row of the matrix "State", where the first row remains unchanged, and the second, third and fourth rows are cycle-lift-shifted on an *(k-1)* basis, where *k* represents the order of a row. **Mix-Columns** carry out a modular multiplication over GF (2^8) for each column. Within each round, **Add-Round-Keys** carry out XOR operation between State and the round key. The key expansion process (the process of generation of round key) incorporates word rotations, S-box substitutions, and XOR operations carried out on the encryption key. For more details regarding the AES algorithm we refer to [13].

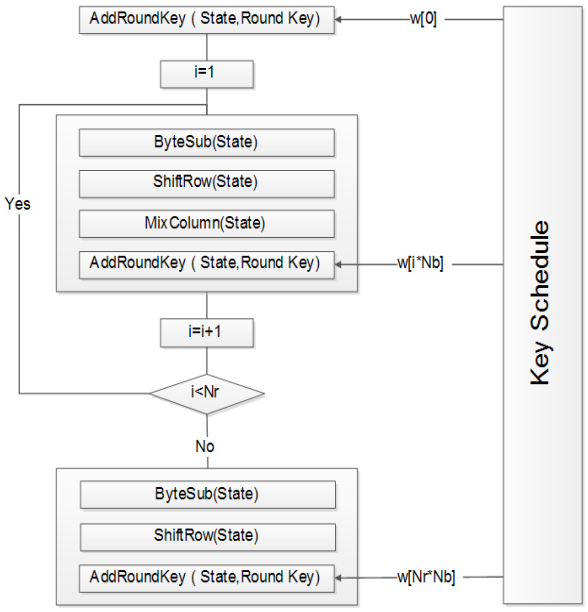

Fig. 1 AES Encryption Process

A 32-bit design of AES, with a pre-calculated key expansion, that is tailored for FPGAs is described in [7]. In this design, S-box is implemented as a LUT, using to its favour the devoted memory pools of FPGAs. The paper suggests an arrangement of the bytes of the matrix "State" so that memory units (shift registers) can hold them efficiently. Such an arrangement enables executing Shift-Rows with addressing logic. A similar technique is suggested in [11] once more. The authors of [14] came up with an idea to decrease the storage space needed and to support many different data path widths. In [5], the decrypting function in [7] was taken off and the hardware core was to run the security procedure described in [2] on FPGA. In [15], there is an enhancement to the FPGA resource consumption of [7] making a good use of the T-box method.

Ref. [8] proposes FPGA implementation of the 8-bit AES algorithm of which the keys are 256-bit, round keys are previously calculated, and "State" is saved in a memory. This allows performing Shift-Rows with logical addressing. Yet, this design uses two pieces of the memory in turns, which consumes resources that could have been saved [14]. In comparison with [9, 10], the method of implementing Mix-Columns is by far more efficient as an input needs to be read one time only from its memory storage, while in [9, 10] the leading 3 bytes in each column have to read twice from memory pending every one of Mix-Columns operations. The sum of all cycles of the AES core described in [8] is less than that of [9, 10]. In our work, we adopt a

suchlike method for Mix-Columns operations as in [8].

Paper [4] proposes an 8-bit AES hardware core based on FPGAs, with a capability of encryption and decryption using 128-bit keys. The data path part of this core consists of an S-box, modular multiplier, and accumulator. The procedure is run by an algorithm saved in the ROM, while RAM acts as a memory for data. The number of cycles in this design far exceeds that of [8, 9]. In our work, FPGA resources are better preserved those of [8].

4. Hardware Architecture of the AES Encryption Core

The AES design solutions with the smallest size with respect to both ASICs [9, 10] and FPGAs [4] have been achieved with loop-back designs utilizing data paths of 8 bits. In this paper, we also use a data path of 8-bit width, but embrace the design approach of [12]. In papers [4, 9, 10], the operations of both AES loop-back and the key expansion are performed in a sequential manner. In the design approach of [12] that is embraced here, the operations are performed in a parallel manner, which to a far extent decreases the total number of cycles and increases the payload. We, still, managed to keep the hardware space occupancy and the power dissipation at a minimum. The top-level structure of the adopted AES encryption hardware core is shown in Fig. 2. The core admits keys of 128-bit length and performs in 16 clock cycles a round at a time. It breaks down into five main building blocks: parallel-in/serial-out converter, byte permutation unit, S-box, Mix-Columns, and key expansion unit. There are two S-box units in this design, and all registers and connections have width of 8-bits.

4.1 The unit of Byte Permutation.

The byte permutation unit (BPU) combines the Shift-Rows processes and storing the matrix "State" partially (Fig. 3). Shift-Rows operation is performed by left-shifting the bytes of "State" in the way described above. The remaining four bytes (the first row) of State matrix are manipulated and kept in the registers of the other AES core data path. The BPU functioning is described in paper [14] in detail. In this unit, data is arranged by cyclically shifting the bytes of the last register inversely with the help of multiplexers. Such inversely allocation is feasible since particular bytes are readout previously, i.e., bypassed with the output multiplexers. There will be no deadlocks because the inversely allocation and bypassing are at equilibrium, and there is an empty slot available every time a byte wants to be inversely allocated.

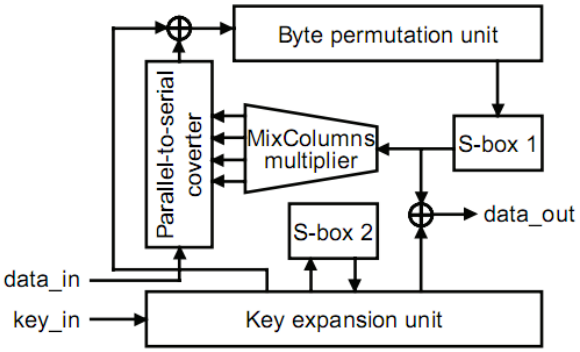

Fig. 2 Top-level structure of AES encryption module.

This BPU unit has the ability to make both right and left shift byte replacements with the help of control signals c_i. When the c_i value is zero, the most significant byte in the input is moved toward the output. The BPU logic circuit latency is 12 clock cycles; this is long enough for a byte to be moved through the permutation. Therefore, a minimum of twelve registers is required and this is the lowest limit for the complexity of registers. The operation of the BPU logic circuit is thus continuous without any break that there is no need for void cycles, and the following 128-bit pattern can be supplied through the time interval following the last byte of the preceding 128-bit pattern. The control signals shown in figure below are given for the permutation of only one 128-bit pattern, where the first clock interval (i.e, t = 0) is the time instant at which the headmost byte is at the input of BPU unit.

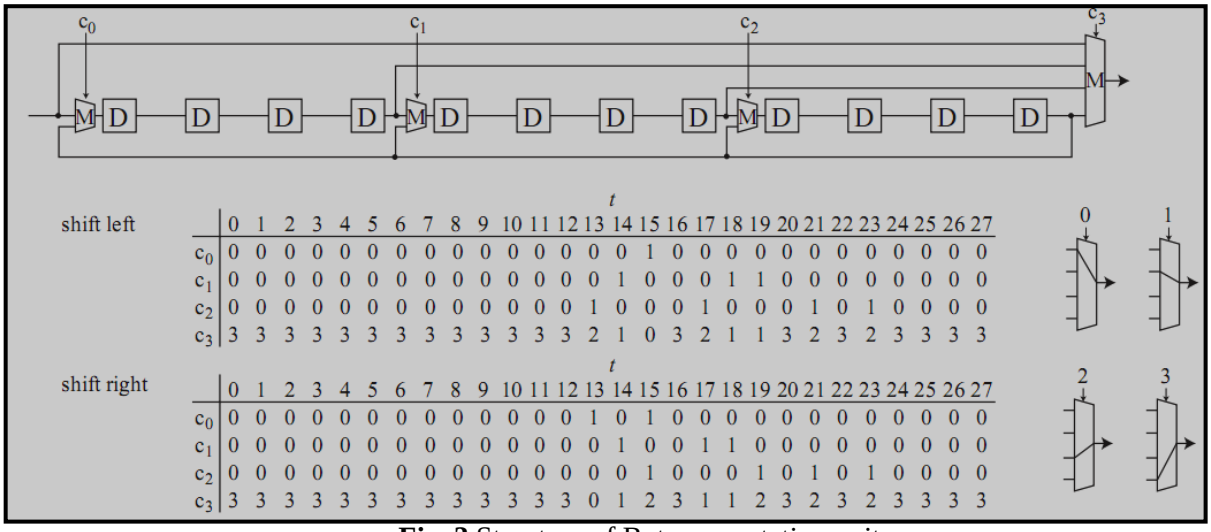

Fig. 3 Structure of Byte permutation unit.

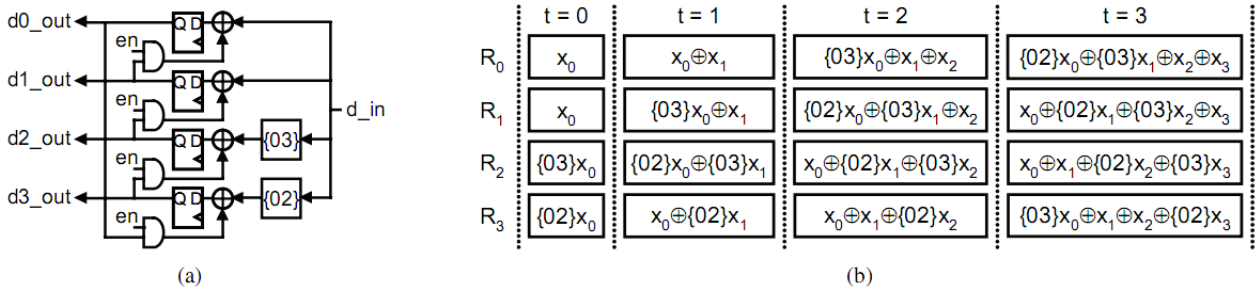

Fig. 4 Mix-Columns Multiplier. (a) Logic diagram and (b) (Ri) registers' contents through the four cycles (t). Multiplying coefficients {03} and {02} point out to the field multiplications of AES with x+1 and x, respectively.

4.2 The Multiplier of MixColumns

Fig. 4(a) shows the Mix-Columns modular multiplier that carries out the modular multiplication process of Mix-Columns. A single "State" column is processed at the same time in four clock cycles as illustrated in Fig. 4(b). Data are fed to the input of unit byte by byte, and the intermediate results are kept in four registers. The coefficients of multiplication remain the same for any element of a column [13], only with a cyclic shift; therefore, a 32-bit parcel of the Mix-Columns function can be carried out by XORing and circular shifting of the intermediate results in the multiplication unit [8]. The registers are zero-masked by means of the signal *"en"* when the first byte of a column is being input. The 32-bit output is input to the PISO (parallel-in/serial-out) register as soon as the byte is completely loaded. A complete Mix-Columns operation is formed in 16 cycles by the Mix-Columns multiplier. This is done in parallel with the other operations of the AES core.

4.3 Implementation of S-box

Recent FPGAs have dictated on-chip memories that can be used efficiently to implement the S-box. There is a multitude of publications that adopt this approach in implementing the S-box [5, 7, 8, 11, 16, 17, 18, 19]. Here, we adopted the S-box implementation of [7]. Fig. 5 exhibits the S-box with a block diagram. The high-level data path consists of a head S-box unit to perform Byte-Sub operations byte by byte. For key expansion, a similar S-box is used.

4.4 Implementation of the Key Expansion Unit

We used the method of [13] to implement the key expansion unit. Fig. 5 (a) describes one round of the Key Expansion transformation [13]. The transformation processes round keys as words of 4 bytes. The words of the new round key are constituted by adding the corresponding word of the old round key to the preceding word. As to the last word, it is cyclically shifted, processed by S-boxes,

and then added to a constant called *"Rcon*". This constant depends on the current round of the key expansion unit.

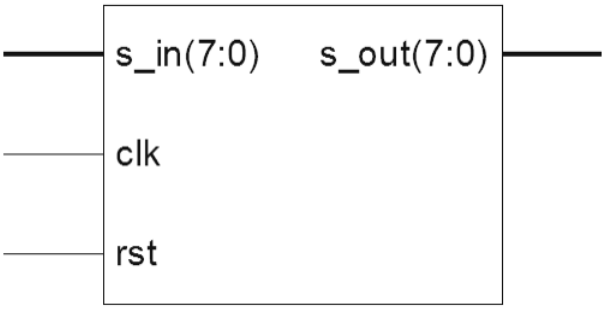

Fig. 4 Block diagram of S-box

The storage for both the round key and the Key Expansion function are enclosed into the data path of the key expansion unit as shown in Fig. 5(b). Through the last iteration, the "rk_last_out" output signal is utilized for Add-Round-Key. Inasmuch as a Mix-Columns operation of 32 bits needs 4 clock cycles, the key expansion unit should have another output called "rk_delayed_out", which is restrained for 4 clock cycles from "rk_last_out". Add-Round-Key needs the output through the systematic rounds and the addition of first (initial) encryption key. That Add-round-Key is carried out by the XOR gate that is available at the entrance of the BPU unit as shown by Fig. 2. The final Add-Round-Key is performed by means of the other XOR gate. The round-dependant constant *(Rcon)* is constructed by means of the round counter by employing time-independent logic circuitry and zero-masked when not in use. Table 1 illustrates how the key expansion unit works.

Shown in this table is a single round of expansion which needs the time (t) of 16 clock cycles. The contents of columns R15 through R0 indicate the contents of the particular registers illustrated in Fig. 5(b). a_{15} through a_0 symbolize the bytes of the previous round key, while b_{15} through b_0 symbolize the bytes of the current round key. $S(a_i)$ means the S-box replacement of the byte a_i. The symbols in bold text represent the bytes that are continuously updated by the processes of the preceding cycle. Amidst the final round, the "b_t" byte of the latest round key is output by *"rk_last_out*" at the "t" cycle. Through regular rounds, the output is held back by 4 clock cycles and produced by control signal *"rk_delayed_out*".

4.5. Structure of the AES Encryption Core Data path.

Fig. 6 illustrates the data path of the implemented AES encryption core. A plain text (data block) along with the encryption key are concurrently get in to the data path byte by byte via input ports "data_in" and "key_in", respectively. Initially, an Add-Round-Key is carried out between the data block and the key throughout the loading. This is done with the XOR gate available at the input of the BPU unit. After 10 iterations, encryption is complete and the cipher text can be serially unloaded from "data_out" output port. The encryption key must be re-entered to the data path accompanying each new data block; this is attributed to that the round keys overwrite the encryption key.

The last iteration, which consists of: Byte-Subs; Shift-Rows; and Add-Round-Keys (no Mix-Columns), is executed while data is being unloaded. As round key is held by the key expansion unit and the State by that of the BPU series of registers, a brand-new data block of plain text and an encryption key are permitted to be fed in to the data path concurrent to the data block of the cipher text. This enhances the performance of the AES core since there are no wasted cycles between successively processed blocks of data. Such a feature is handy; for example, in the CBC-MAC mode, within which a block of cipher text is added to the next block of plain text prior to being processed. The processing one single block of data in our design lasts for 176 clock pulses counting in the loading and the unloading phases. As load and unload are synchronously performable, the effective number of cycles is 160, also with loop-backs.

5. Comparison of Results

The adopted AES core is described in VHDL at the RTL level and synthesized to extract the gate level using Xilinx ISE tools and Xilinx Sparatn-3 FPGA device. Table 2 shows the estimated area in terms of the number of CLBs slices and BRAMs, the power dissipation and the maximum frequency results are obtained at ordinary conditions (1.2 V, 25 ℃) of operation. Power analysis at gate level was carried out for power estimations. We based the assessments and the synthesis of power optimization on switching activities, which have been observed with simulation at gate level using stimuli (random test vectors). Simulations were carried out by ModelSim SE 6.0a simulation tools from Mentor Graphics Corporation.

Table 1. Operation of the key expansion unit.

t	R_{15}	R_{14}	R_{13}	R_{12}	$R_{11} \dots R_4$	R_3	R_2	R_1	R_0	operations
0	a_{15}	a_{14}	a_{13}	a_{12}	$a_{11} \dots a_4$	a_3	a_2	a_1	a_0	$b_0 = a_0 \oplus S(a_{13}) \oplus Rcon, b_4 = a_4 \oplus b_0$
1	$\boldsymbol{b_0}$	a_{15}	a_{14}	a_{13}	$a_{12} \dots a_5$	$\boldsymbol{b_4}$	a_3	a_2	a_1	$b_1 = a_1 \oplus S(a_{14}), b_5 = a_5 \oplus b_1$
2	$\boldsymbol{b_1}$	b_0	a_{15}	a_{14}	$a_{13} \dots a_6$	$\boldsymbol{b_5}$	b_4	a_3	a_2	$b_2 = a_2 \oplus S(a_{15}), b_6 = a_6 \oplus b_2$
3	$\boldsymbol{b_2}$	b_1	b_0	a_{15}	$a_{14} \dots a_7$	$\boldsymbol{b_6}$	b_5	b_4	a_3	$b_3 = a_3 \oplus S(a_{12}), b_7 = a_7 \oplus b_3$
4	$\boldsymbol{b_3}$	b_2	b_1	b_0	$a_{15} \dots a_8$	$\boldsymbol{b_7}$	b_6	b_5	b_4	$b_8 = a_8 \oplus b_4$
5	b_4	b_3	b_2	b_1	$b_0 \dots a_9$	$\boldsymbol{b_8}$	b_7	b_6	b_5	$b_9 = a_9 \oplus b_5$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
14	b_{13}	b_{12}	b_{11}	b_{10}	$b_9 \dots b_2$	b_1	b_0	b_{15}	b_{14}	(no operations)
15	b_{14}	b_{13}	b_{12}	b_{11}	$b_{10} \dots b_3$	b_2	b_1	b_0	b_{15}	(no operations)

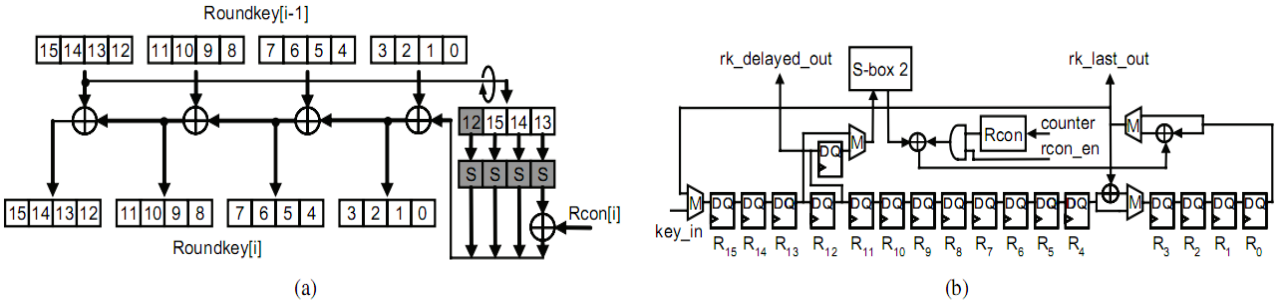

Fig. 5 Structure of key expansion unit: (a) Schematic diagram of the key expansion round. (b) The key expansion unit logic diagram. The organization of bytes shown in (a) synchronized with the registers shown in (b), when the entire round key is produced.

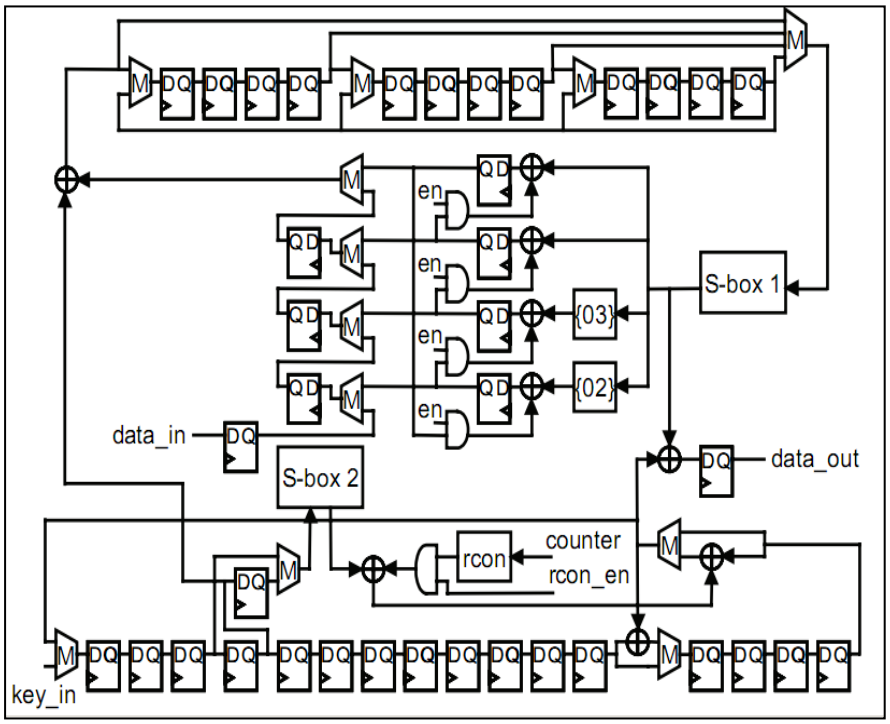

Fig. 6 Structure of the AES encryption core data path.

Table 2 Results of this Implementation in comparison with others.

Implementation	Area (CLB slices + BRAMs)	clock cycles per Block	Maximum Frequency (MHZ)	Total computation time (μs)	Power (μW)	Energy = PDP(nJ)
Ref [8]	222+3	230	50	2.56	45	0.12
Ref [4]	210+3	209	60	2.13	42	0.09
Ref [20]	201+2	198	70	1.99	39	0.08
The proposed	150+2	160	90	1.42	31	0.04

This table shows a comparison among the FPGA implementation results of the adopted 8-bit design to the earlier 8-bit AES implementations [4, 8, 20]. The results exhibit that our design achieves considerable reduction in area (up to 32.4%) and consumed energy (up to 66.7%). Also, it has a significant increase in speed ranging from 28.6% to 44.5%. The reduction in energy consumption is attributed to the lower area consumed by the proposed design besides the lower number of clock cycles required by it to process one cipher block. The enhancement in performance is due to a new plain text data block and encryption key can be fed into the data path concurrent to the data block of the cipher text. This concurrency in feeding data gets rid of the wasted cycles between successively processed blocks of data.

6. Conclusions

Hereby, we presented the hardware implementation of a compact (8-bit) AES encryption core based on FPGA technology. This implementation perfectly fits the applications that require low cost and low power. Also, it enhances the performance of the AES core since there are no wasted cycles between the successively processed blocks of data. This is attributed to that the brand-new data block of plain text and an encryption key are permitted to be fed in to the data path concurrent to the data block of the cipher text. Moreover, the reduction in energy consumption is attributed to the lower area consumed by the proposed design besides the lower number of clock cycles required by it to process one cipher block. Juxtaposed with earlier 8-bit designs, the adopted design achieves considerable reduction in size (up to 32.4%) and energy consumption (up to 66.7%). It significantly increases speed by ratios ranging from 28.6% to 44.5% as well. This makes the adopted design more suitable for resource-limited embedded applications.

References:

[1] IEEE, *IEEE Standard for Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 6: Medium Access Control (MAC) Security Enhancements*, IEEE Std 802.11i, 2004.

[2] IEEE, *IEEE Standard for Local and Metropolitan Area Networks - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPAN)*, IEEE Std 802.15.4, 2003.

[3] ZigBee Alliance. *ZigBee Specification Version 1.0*, Dec. 2004.

[4] T. Good, M. Benaissa, AES on FPGA from the fastest to the smallest, *In Proc. 7th Int. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2005)*, Edinburgh, UK, 2005, pp. 427–440.

[5] P. Hämäläinen, M. Hännikäinen, T. Hämäläinen, Efficient hardware implementation of security processing for IEEE 802.15.4 wireless networks, *In Proc. 48th IEEE Int. Midwest Symp. on Circuits and Systems (MWSCAS 2005)*, Cincinnati, OH, USA, 2005, pp. 484–487.

[6] A. Satoh, S. Morioka, K. Takano, S. Munetoh, A compact Rijndael hardware architecture with S-box optimization, *In Proc. 7th Int. Conf. on Theory and Application of Cryptology and Inf. Secur., Advances in Cryptology (ASI-ACRYPT 2001)*, Gold Coast, Australia, 2001, pp. 239–254 .

[7] P. Chodowiec, K. Gaj, Very compact FPGA implementation of the AES algorithm, *In Proc.*

5th Int. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2003), Cologne, Germany, 2003, pp. 319–333.

[8] S. Farhan, S. Khan , H. Jamal , Mapping of high-bit algorithm to low-bit for optimized hardware implementation, *In Proc. 16th IEEE Int. Conf. on Microelectronics (ICM 2004)*, Tunis, Tunisia, 2004, pp. 148–151.

[9] Feldhofer M., Dominikus S., and Wolker-storfer J., Strong authentication for RFID systems using the AES algorithm, *In Proc. 6th Int. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004)*, Boston, MA, USA, 2004, pp. 357–370.

[10] M. Feldhofer, J. Wolker-storfer, V. Rijmen, AES implementation on a grain of sand, *IEE Proc. Inf. Secur.*, Vol. 152, No. 1, 2005, pp. 13–20.

[11] N. Pramstaller, S. Mangard, S. Dominikus, J. Wolker-storfer, Efficient AES implementations on ASICs and FPGAs, *In Proc. 4th Conf. on the Advanced Encryption Standard (AES 2004)*, Bonn, Germany, 2005, pp. 98–112.

[12] P. Hamalainen, T. Alho, M. Hannikainen, T. Hamalainen, Design and Implementation of Low-area and Low-power AES Encryption Hardware Core, *In Proc. 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools*, Dubrovnik, Aug. 30 – Sep. 1, 2006, pp. 577-583.

[13] National Institute of Standards and Technology (NIST). *Advanced Encryption Standard (AES)*, FIPS-197, 2001.

[14] T. Järvinen, P. Salmela, P. Hämäläinen, J. Takala, Efficient byte permutation realizations for compact AES implementations, *In Proc. 13th European Signal Processing Conf. (EUSIPCO 2005)*, Antalya, Turkey, 2005.

[15] G. Rouvroy, F. Standaert, J. Quisquater, J. Legat, Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications, *In Proc. IEEE Int. Conf. on Inf. Tech.: Coding and Computing (ITCC 2004*), Vol. 2, Las Vegas, NV, USA, 2004, pp. 583–587.

[16] D. Canright, A very compact S-box for AES, *In Proc.7th Int. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2005)*, Edinburgh, UK, 2005, pp. 441–455.

[17] S. Kumar, V. Sharma, K. Mahapatra, Low Latency VLSI Architecture of S-box for AES Encryption, *In Proc. Int. Conf. on Circuits, Power and Computing Technologies*, 2013, pp.694-698.

[18] H. Trang, N. Loi, An efficient FPGA implementation of the Advanced Encryption standard algorithm, *In Proc. of Int. Conf. on Computing and Communication Technologies RIVF*, 2012, pp. 1-4.

[19] A. Elazm, M. El-Moursi, H. Elsimary, M. Dessouky, High speed low power composite field SBOX, *In Proc. of IEEE 5th international Design and Test workshop, Abu Dhabi*, 2010, pp. 24-27.

[20] P. Khose, V. Raut, Implementation of AES Algorithm on FPGA for Low Area Consumption, In Proc. of IEEE int. Conf. on Pervasive Computing (ICPC 2015), Pune, India, 2015, pp. 1-4.