# A 16-Bit Architecture of Advanced Encryption Standard for Embedded Applications

**Conference Paper** · December 2014

DOI: 10.1109/FIT.2014.49



**3 authors**, including:

Imran Ali
Sungkyunkwan University
**33** PUBLICATIONS **54** CITATIONS

SEE PROFILE

Gulistan Raja
University of Engineering and Technology, Taxila
**65** PUBLICATIONS **225** CITATIONS

SEE PROFILE

# A 16-bit Architecture of Advanced Encryption Standard for Embedded Applications

Imran Ali, Gulistan Raja, Ahmad Khalil Khan
Department of Electrical Engineering
University of Engineering and Technology
Taxila, Pakistan
engr.imranalimirza@gmail.com, {gulistan.raja, ahmad.khalil}@uettaxila.edu.pk

***Abstract*—Advanced Encryption Standard (AES) is the most widely used public cipher algorithm for crypto related applications in embedded systems. This paper presents an area efficient 16-bit AES architecture for key expansion, encryption and decryption. In the proposed design, a modular approach is adopted and it is capable of performing all transformations for 128, 192 and 256-bit cipher key lengths. The resources are reduced by minimizing the slice registers and BRAMs without compromising the throughput. The slice count is cut down by sharing the hardware logic resources. Instead of using separate memories for plain text, cipher text and intermediate results, only one BRAM is used. Also for cipher key and round keys, single BRAM is incorporated. The design is synthesized and implemented using Xilinx Virtex-5 FPGA. A comparison is made with existing architectures of different datapaths.**

***Keywords—16-bit; advanced encryption standard (AES); cryptography; embedded system; encryption; decryption; FPGA***

## I. INTRODUCTION

In the last decade, the reliance of mankind on information security has increased a lot. As information become precious, its transfer, storage and handling become a serious issue. The researchers have employed various methods, implementations, techniques and mathematical algorithms to provide reliable and secure solutions. National Institute of Standards and Technology (NIST) concentrated for the development and standardization of new encryption algorithms and selected Rijndael as new Advanced Encryption Standard (AES) in 2001[1]. Since then, AES retains best combination of security against attacks, performance, efficiency, implement-ability and flexibility. The AES, due to its unique design offers the designers to implement it for varying crypto related applications. In previous work, the extensive research efforts have been made to find suitable hardware architectures specific to particular applications [2-14]. The AES has been implemented in 8, 16, 32, 64 and even 128 bit datapaths [2-14]. The smaller datapath architectures [2, 3, 4, 5] provide security for area constrained devices like PDAs, cell phones, RFID, WSN and smart card applications. Some other implementations have also focused for minimum resources [6, 7, 8, 9, 10, 11, 12] with considerable data rate. For high data rate applications, different pipelined/non-pipelined designs are focused to achieve higher throughput at the price of hefty slice registers and BRAMs [13, 14]. A balanced compromise between 8-bit and 32-bit architectures is achieved in 16-bit architecture which provides comparatively high throughput at the cost of low area. The design presented in this paper is a 16-bit architecture which is implementable for all possible key lengths. The modular approach is used to implement encryption/decryption engines and key expansion mechanism. This makes the proposed design a complete module that can be integrated in any FPGA based crypto product, hence delivering maximum flexibility with low area, high throughput and maximum operating frequency. The implementation of proposed design on Virtex-5 FPGA maintains excellent throughput to area ratio with minimum slice registers and Block RAMs (BRAM).

## II. PROPOSED AES ARCHITECTURE

In the proposed AES design AES16, the original 32-bit architecture is mapped to 16-bit for battery operated, resource constrained, area efficient applications. All the datapaths including memories, registers, multiplexers, other logic operations and word size are 16-bit. In this design, the fixed data block size has eight words (Nb = 8) and cipher key contains 8, 12 or 16 words (Nk = 8, 12 or 16) for 128, 192 or 256-bit key lengths respectively. Fig. 1 depicts the block diagram of the presented design. It incorporates two dual port BRAMs, available in FPGAs and it consists of key expansion and encryption/decryption units. This architecture is capable of working for all possible key lengths. The FPGA BRAMs are secure and prove their efficiency when compared with on board external memories and have better read/write time, performance and cost.

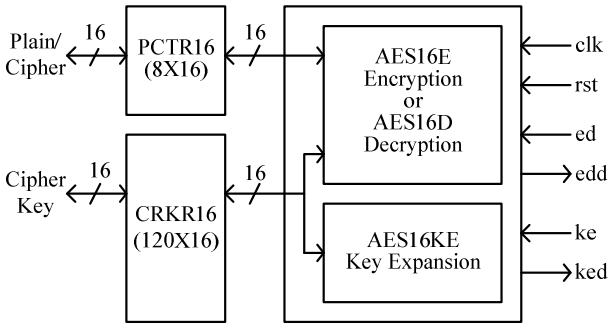


Fig. 1. The 16-bit AES block Diagram

978-1-4799-7505-1/14 $31.00 © 2014 IEEE
DOI 10.1109/FIT.2014.49

The CRKR16 is a dual port 120x16 BRAM and is used for cipher and round key words. In the beginning, the cipher key, K is moved into CRKR16 at its first Nk memory locations. In the key expansion process, Nb(Nr+1) – Nk round key words are generated from cipher key and written back into this BRAM next to the Nk locations. This memory is fully occupied only for 256-bit key length and remains partially empty for 128 and 192 bit key lengths. This effective approach of memory design for keys eliminates the need of separate memories for cipher and round keys opposed to CPR-16 [3]. The PCTR16 is a 16-bit dual port 16x8 BRAM used for plain and cipher text. Initially, 128-bit block of plain/cipher data is loaded into this memory and encryption/decryption is triggered after key expansion. Once, the encryption/decryption starts, the intermediate raw data is ping ponged in this RAM during each round, avoiding extra storage/registers. At the end of final round, it holds cipher/plain text. In this way, this architecture efficiently eliminates the necessity of separate memories for cipher, plain and intermediate data as done in CPR-16 [3].

The AES encryption operates on plain text of fixed 128-bit block size in a loop fashion to generate cipher text of same size. The encryption architecture of proposed AES design is composed of datapath, AES16EDP and control unit, AES16ECU as elaborated in Fig. 2. The figure also explains the memory interfacing with datapath. After each transformation of add round key, byte substitution, shift rows and mix column, carried out by datapath, the intermediate results are written back into PCTR16. A multiplexer, ME serves to direct the results of exactly one transformation to data memory at a time. The AES16ECU is responsible for handling data flow among the datapath components and controls the sequence of individual transformations, schedules the iterations and generates the control signals and addresses for data and key memories. The add round key operation is simply xoring of data words with the corresponding key words. In the initial round, the cipher key K is xored with plain data. Then for the next Nr rounds, the round keys are fetched from CRKR16 and xored with intermediate state.

The byte substitution is a non-linear operation of replacing each byte with another one using S-box lookup table. The individual entries of this lookup table are the multiplicative inverse in Galois Field followed by affine transformation [1].

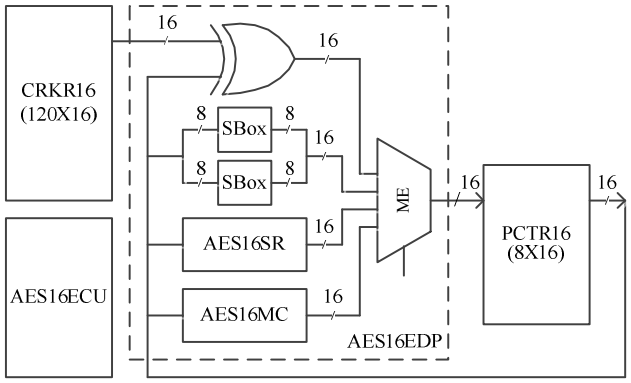


Fig. 2. The 16-bit AES Encryption Architecture- AES16E

The original 32-bit architecture needs four S-box implementations while the proposed design accomplishes this transformation with only two S-box implementations, resulting in significant area reduction. The shift row is a cyclic operation that rotates each row of state matrix towards left using 0, 1, 2 and 3-bytes offset [1]. To accomplish this task, 16-bit shift row architecture, AES16SR is incorporated with the proposed design and is depicted in Fig. 3. The AES16SR can be treated as hybrid of 8 and 16-bit datapaths. Its input and output are 16-bit wide whereas the transformation is performed by splitting 16-bit into 8-bit busses and registers. The intermediate output results are fetched from PCTR16 with the de-multiplexer SRDM, one word per clock cycle, partitioned each 16-bit word into two bytes and patched to the required location into 8-bit registers. The repositioned bytes are joined to form word again and loaded into data RAM. The AES16ECU controls SRM, a 16-bit 8X1 multiplexer for next eight clock cycles.

In encryption, mix column is the most complex linear transformation based on finite filed multiplication [1]. The key feature of proposed design also resides in the mix column architecture, AES16MC and is shown in Fig. 4. The mix column design takes two cycles for single complex multiplication.

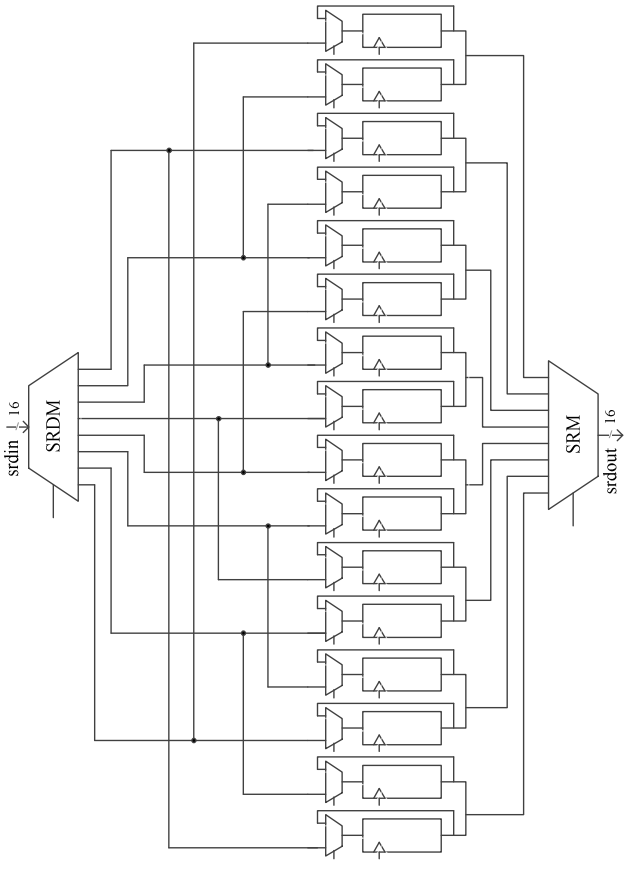


Fig. 3. Shift Rows Datapath - AES16SR

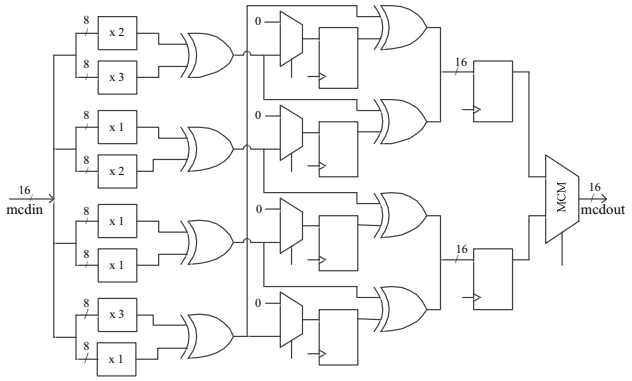


Fig. 4. Mix Column Datapath - AES16MC

In the first clock cycle, the 16-bit word is fetched from PCTR16 and stored after each byte multiplication with coefficients of fixed polynomial. The next word is then loaded for same multiplication and xored with previously calculated results, giving full multiplication on four bytes. The decryption is the reverse process to retrieve plain text from cipher text [1]. This same architecture is applied in decryption. Each round of AES encryption and decryption involves round key operation. In this operation an intermediate result is xored with the corresponding round key. All these round keys are generated according to key expansion routine specified in FIPS-197[1]. In hardware, there are two primary methods for implementing this key expansion routine. The first approach is to compute on-the-fly round keys from the cipher key for every block of data [2]. Secondly, all round keys are computed in advance and stored before starting encryption/decryption. Since this AES architecture accommodates both 16-bit encryption and decryption, so on-the-fly key expansion is no more fruitful and all round keys are pre-computed and stored in memory prior to encryption/decryption. The proposed AES 16-bit key expansion architecture, AES16KE is depicted in Fig. 5 and is capable of generating round keys for all possible 128, 192 and 256-bit cipher key lengths. Initially, the cipher key, K is loaded into CRKR16. The CRKR16 depth is with respect to maximum cipher key size of 256-bit. The AES16KE architecture is composed of datapath and controller. The key expansion datapath performs all the computations such as rotate word, byte substitution, RCON xoring and other operations required for key schedule in all iterations [1].

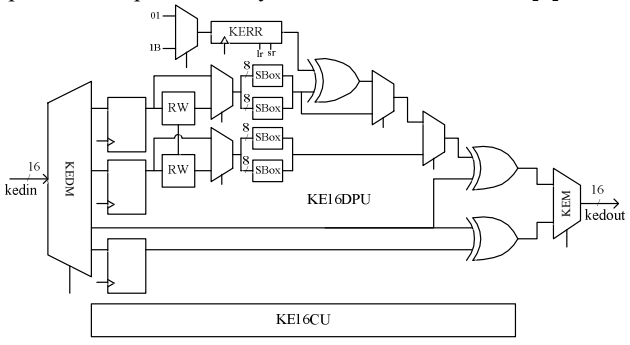


Fig. 5. Key Expansion Architecture-AES16KE

It is composed of three registers, one shift register, six 2x1 multiplexers, one demultiplexer and some other combinational logic as explored in Fig. 5. The key expansion control unit is designed as finite state machine (FSM) and it schedules the iterations and generates the required signals for datapath operation.

## III. Implementation and Performance

The proposed AES design is implemented on Xilinx FPGA using hardware description language. The design is simulated with ModelSim® 6.5 SE before synthesis. For functional verification, each design of encryption, decryption and key expansion is simulated for all possible key lengths (128, 192 and 256 bits) before synthesis. The test vectors for plain text, cipher text and cipher keys are taken from FIPS-197[1]. For the completeness of paper, only simulation results of encryption and decryption for 256-bit key are incorporated. The Fig. 6 depicts the simulation results for encryption. The upper part of this figure is the initial round (Nr = 0) in which the plain text is stored in CPTR16 and cipher key along with all round keys appears in CRKR16. After enabling encryption unit, the design starts encryption process by performing various transformations in each round. At the end of last round (Nr = 14), the cipher text resides in the CPTR16. This situation is shown in the lower half of Fig. 6. Using the same cipher key, this encrypted text is passed through decryption unit and deciphered data, equal to the original one is resulted. The initial and final round (Nr = 14 and Nr = 0) results of decryption for 256-bit key are depicted in upper and lower halves of Fig. 7 respectively.

After successful simulation, the proposed AES design is synthesized and implemented for Xilinx Virtex-5 XC5V5000 FPGA using Xilinx ISE 10.1 synthesis tool. A standard hardware description language Verilog HDL® is used to narrate architecture into hardware. Each implementation is carried out using Algorithmic State Machine and Datapath (ASMD) design.

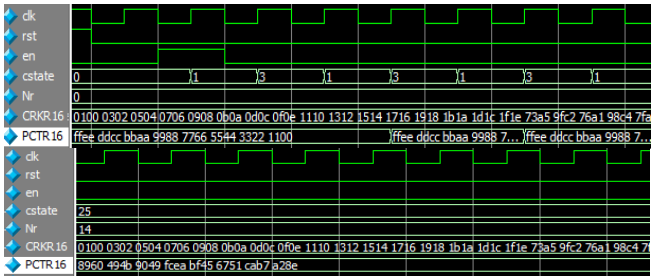


Fig. 6. Simulation Results for 256-bit Encryption

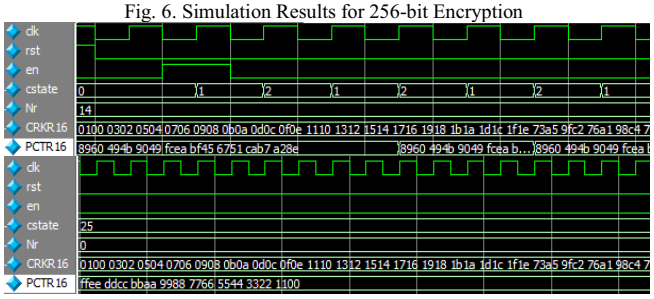


Fig. 7. Simulation Results for 256-bit Decryption

The Moore machine with binary encoding is adapted for Finite State Machine (FSM) implementation. The synthesis results are very satisfactory and demonstrate the effectiveness of the architecture in implementing area efficient design. Each encryption (AES16E), decryption (AES16D) and key expansion (AES16K) modules take one BRAM each and 148, 163 and 122 slices which is less than 1% utilization of target FPGA with maximum operating frequency of 313, 306 and 289 MHz respectively. The decryption takes 15 more slices than encryption due to some additional complex multiplications in inverse mix column transformation. The each of encryption (AES16EK) and decryption (AES16DK) along with key expansion occupies 204 and 236 slices and two BRAMs with frequency of 298 and 269 MHz. When key expansion is integrated into encryption or decryption, then some additional clock cycles are required to generate all round keys before encryption or decryption starts. Table I summarizes the resource utilization (slice registers and BRAMs), clock cycles (for 128-bit key) and maximum operating frequency of key expansion, encryption (with and without key expansion) and decryption (with and without key expansion). The key characteristic of the proposed architecture is that with the same resources, each hardware module performs the required operation for any possible key length and may be used independently. For example, AES16E design can encrypt plain text with any 128, 192 or 256-bit cipher key. Similarly, the key expansion architecture, AES16K may generate round keys for any input cipher key. Table II lists the performance, in terms of throughput (Mbps), Typical Throughout per Slice (TPS) and throughput per maximum operating frequency of encryption and decryption design for all possible key lengths. It is clear from the table, bigger cipher key size; more will be the number of rounds and clock cycles, and lesser will be the throughput. Table III gives the performance measurements for encryption/decryption and key expansion integrated designs. The throughput also incorporates the key expansion clock budgets, though it executes once on system power up or reset. The performance statistics for key expansion design for various key lengths are tabulated in Table IV. Since the cipher key length is directly linked with the number of iterations in key expansion routine [1], therefore the throughput and TPS values are highest for 128-bit key length.

TABLE I. RESOURCE UTILIZATION

| AES Design | Slice Registers | Clock Cycles | Frequency (MHz) | BRAM |
|---|---|---|---|---|
| AES16E | 148 | 184 | 313 | 1 |
| AES16D | 163 | 184 | 306 | 1 |
| AES16K | 122 | 102 | 289 | 1 |
| AES16EK | 204 | 288 | 298 | 2 |
| AES16DK | 236 | 288 | 269 | 2 |

TABLE II. PERFORMANCE OF ENCRYPTION/DECRYPTION WITHOUT KEY EXPANSION

| AES Design | Throughput | TPS (Mbps/Slice) | Mbps/MHz |
|---|---|---|---|
| AES16E-128 | 218 | 1.473 | 0.6965 |
| AES16D-128 | 213 | 1.3067 | 0.6961 |
| AES16E-192 | 162 | 1.0946 | 0.5176 |
| AES16D-192 | 158 | 0.9693 | 0.5163 |
| AES16E-256 | 128 | 0.8649 | 0.4089 |
| AES16D-256 | 126 | 0.773 | 0.4118 |

TABLE III. PERFORMANCE OF ENCRYPTION/DECRYPTION WITH KEY EXPANSION

| AES Design | Throughput | TPS (Mbps/Slice) | Mbps/MHz |
|---|---|---|---|
| AES16EK-128 | 207 | 1.0147 | 0.6946 |
| AES16DK-128 | 187 | 0.7924 | 0.6952 |
| AES16EK-192 | 154 | 0.7549 | 0.5168 |
| AES16DK-192 | 139 | 0.589 | 0.5167 |
| AES16EK-256 | 122 | 0.598 | 0.4094 |
| AES16DK-256 | 110 | 0.4661 | 0.4089 |

TABLE IV. PERFORMANCE OF KEY EPANSION

| AES Design | Throughput | TPS (Mbps/Slice) | Mbps/MHz |
|---|---|---|---|
| AES16K-128 | 363 | 2.9754 | 1.2561 |
| AES16K-192 | 336 | 2.7541 | 1.1626 |
| AES16K-256 | 280 | 2.2951 | 0.9689 |

## IV. COMPARATIVE ANALYSIS

The implementation results are compared for analysis to show the efficiency of proposed design. The different implementations are reported in this paper for performance comparison. The Table V shows the statistics of various designs, including the present implementation results of encryption (AES16E-128) and decryption (AES16D-128) for one key length. In the existing 16-bit design [3], CPR-16 only supports encryption with key expansion for 256-bit key. The CPR-16 uses separate BRAMs for plain text, cipher text and raw data. Also it incorporates two BRAMs for keys, one for cipher key and one for round keys. The controller is also RAM based which utilizes two additional BRAMS. In this way, it uses six more BRAMS and 6% more logic slices than this proposed design. The encryption and key expansion designs are enclosed under common architecture and cannot be utilized independently. Also in presented encryption design for 256-key length, 36% speed and 8% data rate are improved, thus resulting 3 times better throughput per slice (TPS). The 8-bit design, AES-8 [2] implements both encryption and decryption along with on-the-fly key generation for moderate data rate applications. The performance of this design comes at the cost of increased power consumption because the key expansion unit computes round keys from the cipher key again and again for every block of data. The additional numbers of clock cycles are also consumed for each data block which lowers the throughput. It takes 8% more resources and has very low speed, throughput and TPS as shown in Table V and supports cipher operations for 256-bit key length only. The 32-bit datapath designs [4, 5] occupy more than 40% additional FPGA resources and have low operating frequency and TPS. The design [4] implements both pipelined and non-pipelined architectures for 128-bit key length encryption only using on-the-fly key scheduling. The non-pipelined design takes 341 slices and has maximum operating frequency of 50.1 MHz with of 118.5 Mbps as listed in table. The 32-bit pipelined design has 296 Mbps throughput and 125.1 MHz frequency at the price of 422 logic slices which are 34% more than the proposed design. The design [4] has about half throughput for non-pipelined architecture while the throughput of [5] is little more at the cost of about 178 slices and 2 BRAMS. The implementations [6, 7, 8] fall in the category of 64, 128 and 256-bit architectures. These higher datapath width designs though target low area and have enough throughput suitable for embedded applications but still consume about 44~84% more

TABLE V. PERFORMANCE OF KEY EXPANSION

| Design Reference | AES16E-128 | AES16D-128 | [2]Enc | [2]Dec | [3] | [4] | [5] | [6] | [7] | [8] |
|---|---|---|---|---|---|---|---|---|---|---|
| Datapath bits | 16 | 16 | 8 | 8 | 16 | 32 | 32 | 64 | 128 | 256 |
| Slices (Datapath+Controller) | 148 | 163 | 236 | 280 | 228 | 341 | 326 | 1646 | 375 | 520 |
| Operating Speed(MHz) | 313 | 306 | 117 | 105 | 150 | 121 | 169 | 91 | 303 | 37 |
| BRAMs Used | 1 | 1 | 1 | 1 | 7 | 0 | 3 | 0 | 0 | 0 |
| Throughput (Mbps) | 218 | 213 | 41.6 | 37.3 | 110 | 118 | 270 | 224 | 2588 | 129.7 |
| Typical Throughput/Slice (TPS) | 1.5 | 1.3 | 0.2 | 0.1 | 0.5 | 0.3 | 0.8 | 0.1 | 6.9 | 0.2 |
| Throughput/Mhz | 0.7 | 0.7 | 0.4 | 0.4 | 0.7 | 1 | 1.6 | 2.5 | 8.5 | 3.5 |

FPGA slices. The resource and performance comparisons are also presented graphically in Fig. 8 and Fig. 9 respectively. In the resource comparison chart, the left y-axis represents the number of BRAM utilized while the slice count used is labeled on right y-axis. It shows that the existing 16-bit design [3] consumes the maximum BRAMs while the proposed design occupies minimum logic resources. The TPS and throughput are labeled on left and right axis respectively in Fig. 9. It shows that the proposed architectures have the best resource utilization and performance relationship. An AES processing system [9] for embedded application utilizes a 16-bit CPU core along with hardware modules for transformation. The data width between the hardware modules and CPU core is 32-bit. The byte substitution, shift rows and mix column transformations are implemented in hardware while the key expansion and add round key run on CPU. An alternate architecture also uses the same hardware modules along with MicroBlaze, a soft embedded processor instead of CPU core.

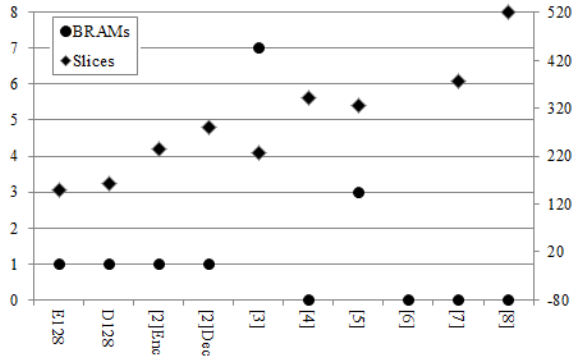


Fig. 8. Resource Comparison

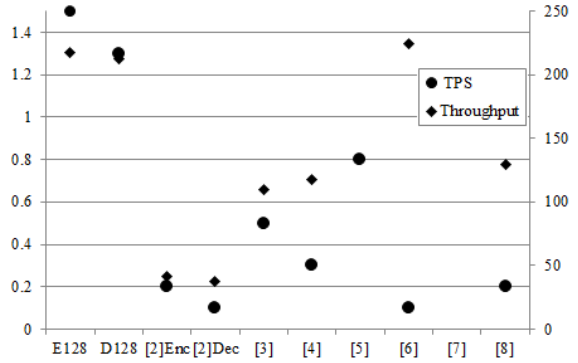


Fig. 9. Performance Comparison

When compared with the proposed design, both of these architectures occupy hefty FPGA resources and clock cycles. For 128-bit key length, the encryption process takes 1546 and 547 clock cycles, 3 and 24 BRAMs and 969 and 1457 slices with CPU Core and MicroBlaze based architectures respectively. The compact AES implementation in [10] supports only 128-bit key length encryption/decryption. It consumes 222 slice registers and 3 BRAMs to give 166 Mbps throughput at maximum operating frequency of 60 MHz. The 16-bit proposed design for 128-bit key length comparatively has up to 14% more throughput and much higher operating speed. One of the two 8-bit architectures in [11] uses Xilinx PicoBlaze, an embedded 8-bit microcontroller soft core, while the other implements Application Specific Instruction Processor (ASIP) for AES-128. These implementations occupy lesser slices and BRAMS but give very low throughputs (2.18, 0.71 Mpbs) and operating frequencies (72.3, 90 MHz). Different block ciphers are implemented in [12] along with AES. The AES design has almost same resource utilization and performance in terms of throughput but have very low maximum operating frequency. This design is 62% slower than the proposed one and cannot be run at a frequency higher than 72 MHz. The 32-bit high throughput and low area designs are focused in [13]. The high speed AES design implements unrolled architecture for encryption/decryption. The low area structure for 128-bit key length gives comparatively very low throughput of 1.98 Mbps at the cost higher FPGA resources of 201 slices and 4 BRAMs giving 56 MHz of maximum operating frequency.

## V. CONCLUSION

In this paper, a 16-bit AES architecture is proposed for resource constrained, battery operated, embedded applications that have throughput requirements less than 200 MHz. The unique design presented in this paper employs efficient usage of hardware resources that results in fewer FPGA slices and BRAMs without compromising the throughput. The number of slices is reduced by sharing the hardware resources. The unique mix column architecture aids to reduce area. Instead of using separate BRAMs for plain text, cipher text and intermediate results, only one memory is incorporated. Also single BRAM is used for cipher and round keys. The same hardware is capable of performing operations of encryption, decryption and key expansion for all three possible key lengths. Lower area and higher throughput make this proposed 16-bit design more efficient than other existing designs.

## REFERENCES


[1] National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES), Federal Information Processing Standards Publications (FIPS PUB) 197, November 26 2001.

[2] Sheikh Muhammad Farhan, Shoab A. Khan and Habibullah Jamal, "An 8-bit Systolic AES Architecture for Moderate Data Rate Applications", Microprocessors and Microsystems, no. 3, pp. 221-231, May 2009.

[3] Habibullah Jamal, Sheikh Muhammad Farhan and Shoab A. Khan, "Low Power Area Efficient High Data Rate 16-bit AES Crypto Processor", The 18th International Conference on Microelectronics, pp. 186-189, December 2006.

[4] Zhou Bin, Yingning Peng, Kris Gaj and Zhonghai Zhou "Implementation and Comparative Analysis of AES as a Stream Cipher", The 2nd IEEE International Conference on Computer Science and Information Technology, pp. 396-400, August 2009.

[5] El Adib, Samir and Naoufal Raissouni, "AES Encryption Algorithm Hardware Implementation Architecture: Resource and Execution Time Optimization", International Journal of Information and Network Security (IJINS), vol. 1, no. 2, pp. 110-118, June 2012.

[6] Liberatori M., Otero F., Bonadero J. C. and Castineira J., "AES-128 Cipher High Speed Low Cost FPGA Implementation", The 3rd Southern Conference on Programmable Logic, pp.195-198, Feburary 2007.

[7] Noura Ben Hadj Youssef, Wajih El Hadj Youssef, Mohsen Machhout, and Rached Tourki, "A Compact 32-bit AES design for Embedded System", The 7th International Conference on Design & Technology of Integrated Systems in Nanoscale Era, pp. 1-4, May 2012.

[8] Monjur Alam, Sonai Ray, Debdeep Mukhopadhayay, Santosh Ghos, Dipanwita Roy Chowdhury and Indranil Sengupta, "An Area Optimized Reconfigurable Encryptor for AES-Rijndael", Design, Automation & Test in Europe Conference & Exhibition, pp.1-6, April 2007.

[9] Tsutsumi D., Ohmura I., Abe T., Yoshimura H. and Inagawa K., "An AES Processing System with a Compact CPU Core for Secure Communication in Embedded Systems", 2012 IEEE Region 10 Conference, pp. 1-5, November 2012.

[10] Pawel Chodowiec and Kris Gaj, "Very Compact FPGA Implementation of the AES Algorithm", Cryptographic Hardware and Embedded Systems, LNCS vol. 2779, pp. 319-333, October 2003.

[11] Tim Good and Mohammed Benaissa, "Very Small FPGA Application-Specific Instruction Processor for AES", IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 53, no. 7, pp. 1477-1486, July 2006.

[12] Rouvroy G., Standaert F.-X., Quisquater J.-J. and Legat J., "Compact and Efficient Encryption/Decryption Module for FPGA Implementation of the AES Rijndael Very Well Suited for Small Embedded Applications", Proceedings of International Conference on Information Technology: Coding and Computing, vol. 2, pp. 583-587, April 2004.

[13] S. S. Naqvi, S. R. Naqvi, S. A. Khan and S. A. Malik, "Application Specific Scalable Architectures for Advanced Encryption Standard (AES) Algorithm", WSEAS Transactions on Electronics, vol. 5, no. 10, pp. 427-436, October 2008.

[14] Ulfat Hussain and Habibullah Jamal, "An Efficient High Throughput FPGA Implementation of AES for Multi-gigabit Protocols", The 10th International Conference on Frontiers of Information Technology, pp. 215-218, December 2012.