A LOW BIT ARCHITECTURE FOR A VERY COMPACT HARDWARE IMPLEMENTATION OF THE AES ALGORITHM

Abdullah Haroon Rasheed, M. Essam, Umair Khalid, Sheikh M. Farhan[1], Dr. Shoab A. Khan

Dept. of Computer Engineering, College of Electrical & Mechanical Engineering, National University of Sciences and Technology, Rawalpindi, Pakistan

[1]University of Engineering and Technology, Texila, Pakistan

ahrasheed@gmail.com, +92-333-5255468, EME College, Peshawar Road, Rawalpindi, Pakistan

arain_essam@hotmail.com, +92-300-6689766, EME College, Peshawar Road Rawalpindi, Pakistan

umairsuleri@hotmail.com, +92-345-6691988, EME College, Peshawar Road, Rawalpindi, Pakistan

smfarhan@carepvtltd.com.pk, +92-300-8526163, 19-Attaturk Avenue G-5/1, Islamabad, Pakistan

shoab@carepvtltd.com , +92-300-8568714, EME College, Peshawar Road, Rawalpindi Pakistan

ABSTRACT

This paper presents the implementation of the Advanced Encryption Standard algorithm on an 8-bit compact architecture. Encryption, key scheduling and decryption are implemented by small resources and extensive resource sharing. The architecture is perfectly suited for low cost applications which require moderately high data rates. Among the various cost effective and compact implementations already available, this architecture presents a novel design of the data path which has been modelled on 8-bit systolic architecture. The S-Box required for byte substitution has been implemented in BRAMS further reducing the consumed area. The design has been further embellished by a memory based controller which simplifies the control process and makes it viable for very effective hardware utilization. This produces one of the smallest implementations of the algorithm on FPGA with a reasonably high throughput. Considering the aforementioned, it minimizes area and power consumption, the basic factors of a low cost implementation. Comparisons drawn from FPGA implementation with other architectures have also been presented.

KEY WORDS

Encryption AES Security 8-bit-systolic-architecture

A LOW BIT ARCHITECTURE FOR A VERY COMPACT HARDWARE IMPLEMENTATION OF THE AES ALGORITHM

1 INTRODUCTION

Rijndael algorithm [2] was selected as the Advanced Encryption Standard (AES) [1] by the National Institute of Standards and Technology (NIST) in November 2001. This cryptographic algorithm is and will be used extensively for maintaining data security and reliability in various wireless and wired transmissions. As the application of the new encryption algorithm grows, so does the need for variety in design approach.

For meeting the currently rising trend of wireless communication, architectures must cover the constraints posed by power and cost. Keeping these factors to the minimum and yet developing an architecture with reasonably high throughput is a challenge posed to the designers and the one addressed by this paper. It has been observed that speeds up to 60 Mbps are fairly high for wireless communications. Thus the design trade-offs can be explored by keeping in perspective the throughput requirements and the minimized area constraints of low-end applications such as handheld devices and different low-cost surveillance and monitoring systems.

The main focus in the approach presented by this paper is to generate an amenably controlled architecture with significant reduction in resources consumed by the data path. The following sections present some related designs and their respective differences from the proposed model, an overview of the encryption algorithm, designed architecture and the obtained results along with comparisons with similar and other area-optimized alternatives.

2 COMPARABLE WORK

A considerable amount of work has been done on low cost FPGA implementations of the AES algorithm. Most of the research focused on efficient use of FPGA resources such as [5], [6].

Other architectures used a 32-bit data path and optimized the design to meet area minimization [3], [4]. Such designs were believed to be limited due to the MixColumn transformation which prevented the design of a narrower data path.

An 8-bit application specific processor (ASIP) was proposed eventually [7] but an architecture incorporating a smaller data path was still not available. An ASIC architecture was developed [8] with quite a bit of control overhead. This design incorporates a similar 8-bit systolic data path but reduces the control overhead on area, by the design of a micro-coded controller. It further makes use of the techniques, mentioned in Section 4 and 5, for efficient use of the FPGA resources available.

3 ADVANCED ENCRYPTION STANDARD ALGORITHM

The encryption algorithm approved by NIST by an FIPS publication [1], has the provision of using 128,192 or 256 bit key lengths. In this algorithm a 128 bit key has been used.

The algorithm takes a 128-bit data block and is initialized by addition of the cipher key to the data block. Subsequent transformations occur on this block, which is referred to as a state. The cipher is iterative and with 128-bit key length, the state goes through ten rounds of same set of transformations.

The following pseudo code presents a general overview of the transformations in the encryption process.

```
Round(State,RoundKey)
{
ByteSub(State);
ShiftRows(State);
MixColumn(State);
AddRoundKey(State,RoundKey);
}
```

The final round of the cipher is slightly different. It is defined by:

```
FinalRound(State,RoundKey)
{
ByteSub(State) ;
ShiftRows(State) ;
AddRoundKey(State,RoundKey);
}
```

The ByteSub Transformation is a non-linear byte substitution, operating on each of the State bytes independently. The substitution table (or S-box) is invertible and obtained by taking multiplicative inverses in GF(2^8) and applying an affine transformation.

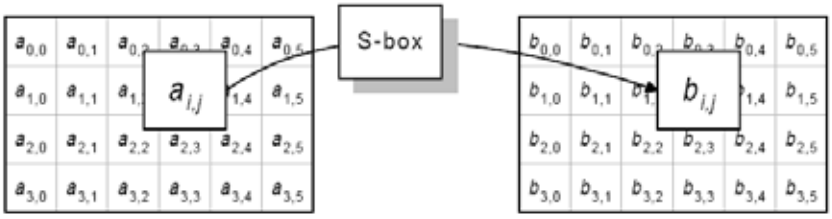

Figure1. Byte Substitution through S-box

In ShiftRows, the rows of the State are cyclically shifted over different offsets. Row 0 is not shifted, row 1 is shifted over 1 byte, row 2 over 2 bytes and row 3 over 3 bytes.

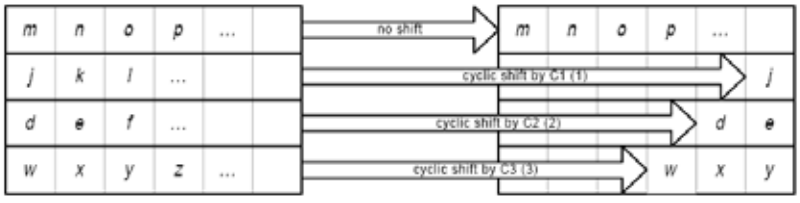

Figure2. The ShiftRows Transformation

In the MixColumn transformation matrix multiplication is performed. Each column, in the 4x4 byte block of data is multiplied by a matrix as shown in Figure 3. The entries are in GF(2^8) field.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Figure3. The MixColumn Transformation

A Round Key is added to the state by a simple bitwise XOR. The Round Key is derived from the Cipher Key by means of the key schedule.

The key scheduling consists of two components: the Key Expansion and the Round Key Selection. The basic principle is that the total number of Round Key bits is equal to the number of rounds plus one multiplied by the block length. The Cipher Key is expanded into an Expanded Key.

4 LOGICAL DESIGN

The proposed design makes use of the fact that the algorithm can be generally mapped on a low bit architecture. The main hindrance in this exercise is the MixColumn transformation. This can be overcome by the use of a systolic architecture which uses the systolic matrix multiplication technique [8]. This systolic architecture although very effective in area reduction, generates quite a bit of overhead on the FPGA space.

The design proposed in this paper minimizes this overhead by implanting a controller which is not architectural, but in the form of a micro-coded statemachine. This micro-coded statemachine has capabilities of iteratively running instructions through the use of a down counter. All the controller information i.e. the control signals are pre-calculated according to the algorithm and the signals only need to be sent to the appropriate data path units for controlling and manipulating data.

The ShiftRows transformation can be implemented before the ByteSub transformation. Using this property of the cipher the ShiftRows transformation can be eliminated from the main encryption round by supplying the data to the data path in accordance with the ShiftRow transformation.

Moreover the key expansion and encryption processes have been implemented on the same data path, as is not the case with generally available algorithms. The control for key expansion has been implemented in main controller.

5 PHYSICAL DESIGN

All registers and interconnections used in the architecture are 8-bits wide. The controller and the data paths are described in detail over the next sections.

5.1 Micro-Coded Controller

The controller has been implemented on the dual port Block SelectRAM™ (BRAM) available in the FPGA. Four BRAMs of size 36x256 have been allocated for this purpose. A down counter is

implemented in the architecture to iteratively move through the same instructions. The pre-calculated instructions are translated in the state register and sent to the appropriate mux units. Three BRAMs have been used for the encryption instructions and one for the key expansion instructions.

The last of the BRAMs holds signals for key expansion. As soon as the cipher key is written in one of the memories as explained in Section 5.3, the controller loads the state register with contents from this BRAM. Once the key expansion process is finished the controller returns to the encryption process. The obstruction in speed posed by this wait for key expansion is traded-off in terms of area saved with use of the same architecture for both processes.

As mentioned earlier, it is very important to note that the implementation of controller on BRAMs not only makes it efficient in its responsiveness, but also makes the control logic flexible. The data path can be potentially used for other purposes or supporting pragmatic changes in the algorithm. The BRAMs provide a better access time then any other controller implementation and better performance. *Figure5* shows the general design of the controller.

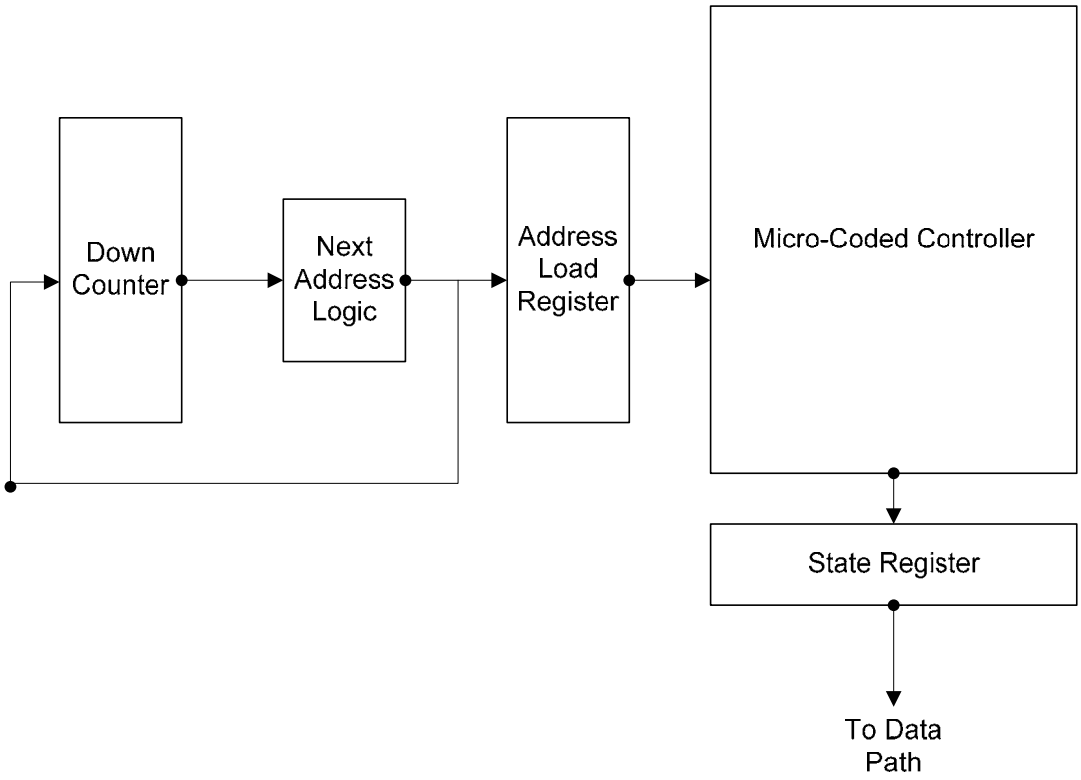

Figure5. The Micro-Coded Controller

5.2 Data Path

The data path consists of a few multiplexers and 8-bit registers as shown in *Figure6*. The design is power efficient and occupies very little space. The AddRoundKey transformation is simple bitwise XOR of the key with data. In this case 8 bits of key are XORed with 8 bits of state data.

The ByteSub transformation requires S-Box. This S-Box has been implemented on an 8x256 bit space in a BRAM. This implementation makes the design on FPGA core, further reduced and compact.

The MixColumn transformation requires matrix multiplication over GF(2^8). This is achieved by XOR and shift operations. As in each row of the matrix the elements are right shifted by a

position, a systolic architecture can be implemented by implementing the three basic multiplications and calculating the partial sums for every element and then shifting these partial sums to be accumulated with the corresponding partial sums.

The state which is processed out of one round of encryptions needs to be stored in the memory as well. For this purpose 8x32 bit memory is utilized on the same BRAM which incorporates the S Box. After 10 rounds of the above mentioned processes, the encrypted data is available as output in 16 bytes of this space.

5.2.1 Key Expansion

As the key expansion is carried out on the same data path the cipher key needs to be stored in memory along with the generated round keys. As soon as the first 16 bytes of key are written on the memory initially, the key expansion process starts. The cipher key and the generated round keys are stored in 8x160 bit space of a BRAM.

For the key expansion process the last column of four bytes of the key are fed into the MixColumn process, bypassing the ByteSub and AddRoundKey transformations. The MixColumn architecture gives a rotation to the last two bytes. These four bytes are supplied to S-Box for ByteSub transformation. Then they are XORed byte by byte with the first column of the key and again with a column from the RCON table. The RCON table is also stored in 8x10 bits of memory.

After 40 iterations of similar and slightly different processes the 160 bytes of expanded key is available in the memory.

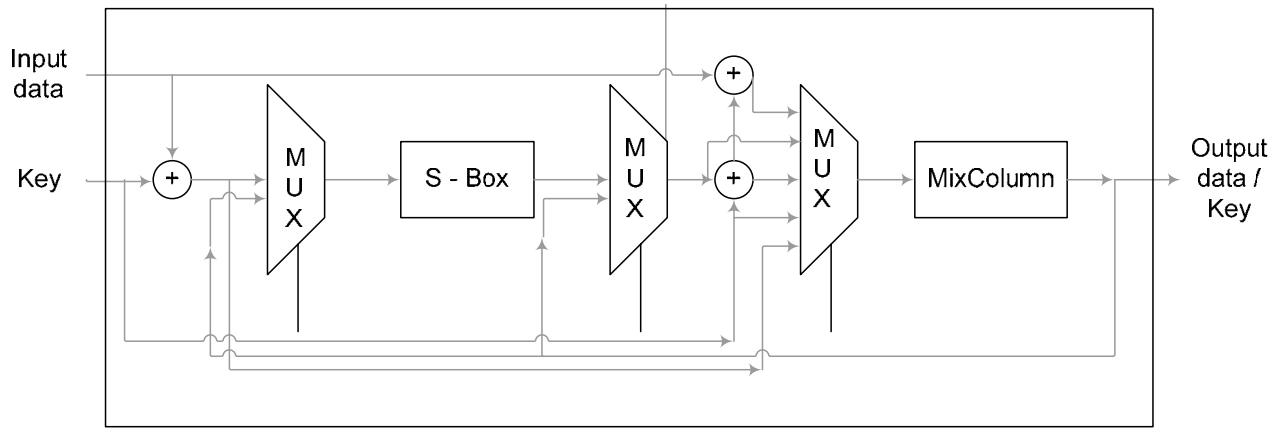

Figure6. The Data Path

6 RESULTS AND OBSERVATIONS

The AES algorithm design was tested and verified using the Advanced Encryption Standard Algorithm Validation Suite (AESAVS) [9] in modelsim simulation environment and then on hardware. It was synthesized on a Xilinx Virtex II XC2V1000 FPGA. The resulting synthesis occupied as low as 235 slices, which is considerably less then area occupied by most 32-bit architectures and quite reduced from the available 8-bit architecture [8]. The architecture can be operated on a clock frequency of 120 Mhz. And has a throughput of 55Mbits/sec.

The architecture was also implemented on a low cost Xilinx Spartan II FPGA. The target device was XC2S30. This implementation can be operated on a clock frequency of 54 Hz. The comparisons drawn with other low cost implementations over reconfigurable hardware are stated in *Table1*.

Table1. Comparison with low-cost architectures

	[3]	[5]	[4]	[8]	**This Implementation**	**This Implementation**
Device	Xilinx XC2S30	Xilinx XCV1000	XC3S50	XC2V1000	**XC2V1000**	**XC2S30**
Process	Enc/Dec	Enc	Enc	Enc	**Enc**	**Enc**
Slices	222	5302	163	337	**235**	**244**
Speed (MHz)	60	14.1/31.8	71	110	**120**	**53**

7 CONCLUSION

The paper presented an 8-bit area efficient design of the AES algorithm implementation on FPGA. The low bit data path, results in efficient use of area and a low power design. The resulting design when implemented on an off the shelf FPGA results in less then 50% resource utilization on the FPGA, thus leaving enough space on a high end FPGA for other glued architectures. The micro-coded controller further expounds this characteristic. And thus relieves the over head caused by low bit mapping of the architecture by efficient use of FPGA RAMs. Shared encryption and key scheduling data paths result in high level of resource utilization and sharing. The 8-bit architecture is fully scaleable, and multiple instances can be used to achieve a throughput higher then the current rate of 55 Mbps

The architecture is perfectly suited for wireless communication and embedded systems and is very practical in its implementation, as it provides speed upto 53 MHz, which is quite enough for wireless communication of the contemporary era .

8 PERMISSIONS

Figure1, Figure2, Figure3 taken from the publicly available FIPS-197 published by NIST [1]

9 REFERENCES

[1] National Institute of Standards and Technology (NIST), Information Technology Laboratory (ITL), *Advanced Encryption Standard (AES)*, Federal Information Processing Standards (FIPS) Publication 197, November 2001.

[2] J. Daemen, V.Rijmen *"The Rijndael Block Cipher" AES proposal, First Candidate conference (AESI),* August 20-22, 1998.

[3] P. Chodowiec, K. Gaj, "*Very Compact FPGA Implementation of the AES Algorithm"*, Proc. Cryptographic Hardware and Embedded Systems (CHES 2003), LNCS Vol. 2779, pp.319 – 333, Springer-Verlag, October 2003

[4] 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"*, Proceedings of the international conference on Information Technology: Coding and Computing 2004 (ITCC 2004), pp. 583 – 587, Vol. 2, April 2004

[5] A.J. Elbirt, W. Yip, B. Chetwynd, and C. Paar, "*An FPGA Based Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists*", Proc. Third Advanced Encryption Standard (AES) Candidate Conf., Apr. 2000.

[6] K. Gaj and P. Chodowiec, "*Comparison of the Hardware Performance of the AES Candidates Using Reconfigurable Hardware*", Proc. Third Advanced Encryption Standard (AES) Candidate Conf., Apr. 2000.

[7] T. Good and M. Benaissa, "*AES on FPGA from the Fastest to the Smallest"*, Proc. Cryptographic Hardware and Embedded Systems (CHES 2005) 7th International Workshop, Edinburgh, UK, August 29 – September 1, 2005

[8] S. M. Farhan, H. Jamal and M. Rahmatullah, *"High Data Rate 8-bit crypto-processor"*, Proc. Peer-reviewed Proceedings (ISBN 1-86854-522-9) of the ISSA 2004 enabling tomorrow Conference, 30 June - 2 July 2004, Gallagher Estate, Midrand, SOUTH AFRICA.

[9] L. E. Bassham, *"The Advanced Encryption Standard Algorithm Validation Suite (AESAVS)"*, National Institute of Standards And Technology, Information Technology Laboratory, November 2002.

[10] Xilinx Virtex™ II Platform FPGAs. URL: www.xilinx.com