International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-2 Issue-6, August 2013

Design and Implementation of Extended Version of AES Algorithm with DSP Units

Sagar Deshpande, Leelavathi G.

Abstract*— Advanced Encryption Standard (AES), also known as Rijndael, is a block cipher algorithm that has been analyzed extensively and is now used widely. The AES algorithm hardware implementation is faster and more secure than software implementation. AES algorithm is used to encrypt and decrypt data as this can make the whole process much faster and secured communication is also established in the system.This is also extended to 176 and 192 bits in this work. 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, for higher throughput over 1 Giga bits per second (Gbps). However, the studies of low power, low area and low cost implementations, which normally have throughput less than 1Gbps and use the data path less than 32-bit, have recently appeared in ASIC as well as in FPGA for wireless communication and embedded hardware application. In the proposed work the encryption of 128,176 and 192 bits are aimed for accurate AES implementation. This proposed work has been divided into two main phases software development and hardware development. In the development of software, it is involved with writing the code, simulation process with Xilinx 13.2 ISE tool. The hardware development covers the Xilinx Spartan 6 FPGA target board development. An AES cipher implementation that is based on the BlockRAM and DSP units embedded within Xilinx's Spartan-6 FPGAs. An iterative "basic" module outputs a 32 bit column of an AES round in each clock cycle, with the throughput of 1.76 Gbit/s when processing a 128 bit inputs, one 176 bits data and 192 bits data. Finally, the "round" module is replicated ten times for a fully unrolled design that yields over 55 Gbit/s of throughput. High throughput implementations are mainly used for high-end devices such as accelerator cards for e-commercial service and security trunk communications. In order to achieve higher performance in today's utilization of hardware accelerators for cryptography algorithms and heavily loaded communication networks is more efficient.

***Index Terms*— DSP, BRAM, AES, FPGA, ASIC, RIJNDAEL**

I. INTRODUCTION

The Advanced Encryption Standard (AES) is a block cipher used in many applications with a rich literature discussing how to optimize implementations of it for both software and hardware. Most available AES implementations for reconfigurable hardware, however, are based on traditional configurable logic and do not exploit the full potential of modern devices. The proposed work focuses on new embedded functions inside of the Xilinx Virtex-5 and Spartan -6 FPGA, such as large dual-ported RAMs and digital signal processing (DSP) blocks with the goal of minimizing the use of registers and look-up tables that could otherwise be used for other functions. Therefore, the design presented in the eork will be especially appealing in applications where user logic is scarce, yet not all embedded memory and DSP blocks are used. A "basic" eight-stage pipeline module based on a combination of two 36 Kbit BlockRAM (BRAM) and four DSP blocks, which outputs one 32 bit column of an AES round each cycle with a feedback loop for iterative operation is designed. This basic module is replicated four times for a full AES round with a 128 bit datapath, which, in turn, is replicated ten times for a fully unrolled operation, the key expansion function in these designs are excluded. Security of data is becoming an important factor for a wide spectrum. Advanced Encryption Standard (AES) has replaced its predecessor, Double Encryption Standard (DES), as the most widely used encryption algorithm in many security applications. It offers a good combination of efficiency, performance, implement ability, flexibility and security''. Although key size determines the strength of security, area and the power consumption. Where lower area and power consumption becomes crucial. The differances between the level of security, throughput and area consumption is left in the hands of the implementers depending on the need in the proposed work, we present hardware implementations of the AES encryption using an approach which includes modules memory and lookup tables for 128-bit, 176-bit and 192-bit key. The higher the key size, the more secure the ciphered data, but also the more rounds needed. The simulation and synthesis of AES encryption algorithm hardware implementation has been performed using Very High Speed Integrated Circuit Hardware Description (VHDL) language and Xilinx ISE 13.2i simulator to see and compare throughput and area of hardware implementations of three variants of AES key sizes: 128, 176 and 192.

Manuscript published on 30 August 2013.
* Correspondence Author (s)
Mr. Sagar Deshpande*, VLSI Design and Embedded System, VTU Extension Center, UTL Technologies Ltd., Bangalore, India.
Mrs. Leelavathi G, VLSI Design and Embedded System, VTU Extension Center, UTL Technologies Ltd., Bangalore, India.

II. THE HARDWARE SYSTEM

A. *Iterative Design of AES Processor*

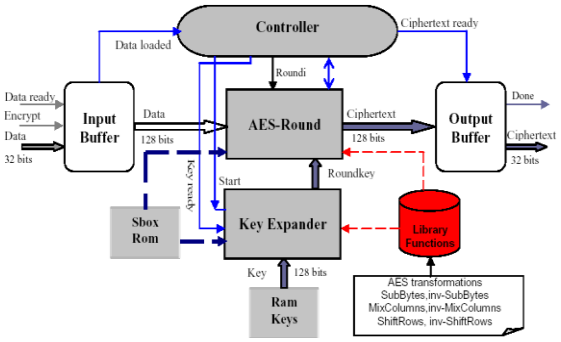

Fig 1: Design of AES Processor

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The encryption of data is performed for keeping it confidential and illegible to people who are not meant to see it in its "plaintext" form. Encryption is used in a wide range of applications, some requiring large amounts of data to be encrypted or decrypted at very high speeds. Symmetric encryption using block ciphers is often used, with the security relying on a pre-established secret key shared between sender(s) and receiver(s) [3]. AES has been designed as a substitution-permutation network (SPN) and uses between 10 to 14 encryption rounds (depending on the length of the key) for a full encryption and decryption of one 128 bit block. In a single round, the AES operates on all of the 128 input bits represented as a 4X4 matrix of bytes. Fundamental operations of the AES are performed based on byte-level field arithmetic over the Galois Field GF(28) so that operands can be represented in 8 bit vectors [7]. The AES cipher has been designed to be efficient in both software and hardware, and is versatile in which it can be made either area-optimized, iterative, and slow, or "unrolled" and fast by parallelizing round operations and pipelining. Its 8 bit vector representation allows implementations on very small processing units, while 128 data paths allow for gigabit throughput. .

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

An AES round consists of mainly four basic operations on A:

1. **SubBytes:** all input bytes of A are substituted with values from a non-linear 8 X 8 bit S-Box.
2. **ShiftRows:** the bytes of rows R_i are cyclically shifted to the left side by 0, 1, 2 or 3 positions.
3. **MixColumns:** columns C_j are matrix-vectormultiplied by a matrix of constants in GF (2^8).
4. **AddRoundKey:** a round key K_i is added to the input using GF(2^8) arithmetic.

The sequence of these four operations defines an AES round, and they are iteratively applied for a full encryption or decryption of a single 128 bit input block. Since some of the operations above rely on GF (2^8) arithmetic we are able to combine them into a single complex operation. The Advanced Encryption Standard defines such an approach for software implementations on 32 bit processors with the use of large lookup tables. This approach requires four 8 to 32 bit lookup tables for the four round transformations, each the size of 8 Kbit.

III. DESIGN AND IMPLEMENTATION

Adaptation of the software-oriented approach into modern reconfigurable hardware devices in order to achieve high throughput for modest amounts of resources. We use the Xilinx Spartan-6 FPGA which has advanced features that are useful for our application beyond traditional LUTs and registers. These are dual ported 36 Kbit BlockRAMs (BRAM) - ones that have independent address and data buses for the same stored content and versatile digital signal processing (DSP) cores. The DSP cores allow the designer to implement timing- or resource-critical functions such as arithmetic operations on integers or Boolean expressions that would otherwise be considerably slower or resource demanding if implemented with "ordinary" logic elements. The DSP blocks were introduced in the Virtex-4 family of FPGAs to perform 18X18 bit integer along with a 48 bit accumulator, though they were limited to 24 bit bit-wise logic operations. 48 bit bit-wise logic operations were added in Spartan-6, and can run at up to 550 MHz, the maximum frequency rating of the device. The internal datapath inside of the DSP block is 48 bit wide, except for integer multiplication. The Spartan-6 DSP blocks come in pairs that span the height of five configurable logic blocks (CLB), and they can be efficiently cascaded between pairs with additional dedicated paths to adjacent DSP tiles. A single dualported 36 Kbit BRAM also spans the height of five CLBs and matches the height of the pair of DSP blocks, with a fast datapath between them. Our initial observation was that the 8 to 32 bit lookup followed by a 32 bit XOR AES operation perfectly matched this architectural alignment for efficient and fast implementation. Based on these primitives, we developed a basic AES module that performs a quarter (one column) of an AES round transformation given by Equations.

A. Basic Module

The basic construct we started out with is shown in Figure 1. Since each column requires all four T-table lookups with their last-round T-table counterparts, that means that we needed to fit a total of eight 8 Kbit T-tables in a single 36 Kbit dual-port RAM. For performance and resource efficiency reasons we opted against "reversing" the last operation and searched for a solution that would enable us to fit all tables into a single BRAM. We realized that our design can use the fact all T-tables are byte-wise transpositions of each other, such that by cyclically byte-shifting of the BRAM's output for T-table T_0 we can produce the output of T_1, T_2 and T_3. Using this observation, we only store T_0 and T_2, and also their last round counterparts $T_{0'}$ and $T_{2'}$ in a single BRAM. Using a single byte circular right rotation (a; b; c; d) =>(d; a; b; c), T_0 becomes T_1, and T_2 becomes T_3 and the same for the last round T-tables. In the hardware, it requires a 32 bit 2:1 multiplexer at the output of each BRAM with a select signal from the control logic. For the last round, a control bit is connected to a high order address bit of the BRAM to switch from the regular T-table to the last round's T-table. The memory layout is shown in Figure 2 the first 8 bits of the address is the input byte $a_{i,j}$ to the transformation, bit 9 controls the choice between regular and last round T-table, while bit 10 chooses between T_0 and T_2.

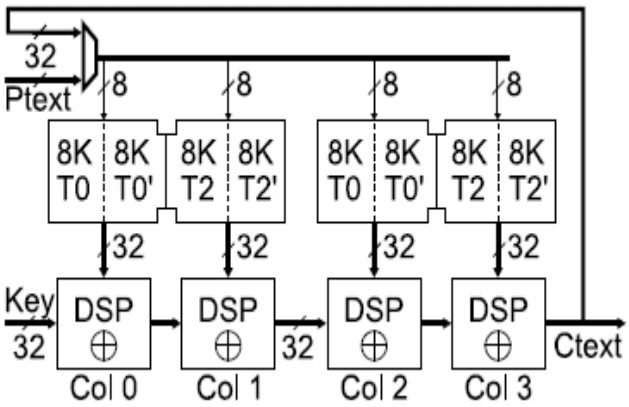

Fig 2: The basic construct structure

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The Figure 2 shows the basic construct structure. Each dual ported BRAM contains four Ttables, two for the first nine rounds, and two for the last one. Each DSP48G block performs a 32 bit bitwise XOR operation. After passing through the four DSP blocks, column results are fed back as the input to the next round.

Using two such BRAMs with identical content, we get the necessary lookups for four columns, each column is capable of performing all four T-table lookups. Both the BRAMs and DSP blocks provide internal input and output registers for pipelining along the datapath; we get these registers "for free" without use of any flip flops in the fabric. At this point, we already had six pipeline stages that could not have been easily removed if our goal was high throughput. So we decided that instead of trying to reduce the pipeline stages, we can add two more so that we are able to process two input blocks at the same time, doubling the throughput. One of these added stages is the 32 bit register after the 2:1 multiplexer that shifts the T-tables at the output of the BRAM; these are the only user logic registers we use for the basic construct (shown inside dotted line in Figure 3).

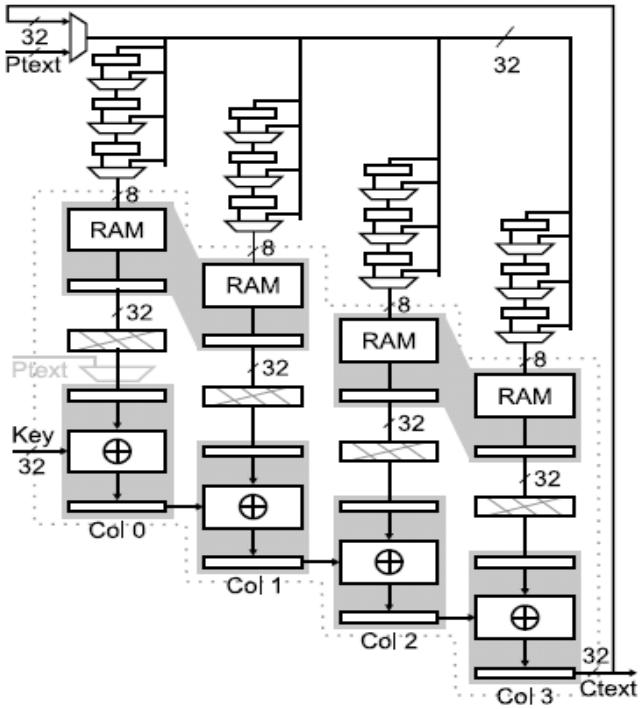

Fig 3: The Basic Iterative Round (Without Control Logic)

The Figure 3 shows the "basic" iterative round (without control logic). Plaintext Ptext is chosen as the initial input, then output data is fed back through 8 bit shift registers for a complete AES encryption. The pipeline stage between BRAMs and DSPs block is used as an optional 8 bit right shift when T_0 and T_2 are turned into the T_1 and T_3 respectively.

A full AES operation is implemented by operating the basic construct with an added feedback scheduling in the datapath. Combined with BRAM lookups, we assigned a cascade of DSP blocks to perform the four XOR operations required for computing the AES column output according to Equations. For feeding in the corresponding ai;j for the lookup into the BRAM, we added a sequence of three 8 bit loadable shift registers and an input multiplexer for each column. These 24 bit registers are loaded in sequence, the leftmost (C0) on the first cycle, and the one to its right on the next, and hence on.

This construct has eight pipeline stages with the following operations, in order: lookup L, where the 8 to 32 bit T-table lookup is performed within the BRAM; register R is the BRAM's output register; transform T[0::3], where T0 and T2 are optionally shifted into T1 and T3 content, respectively. DSP D input register; and _, the exclusive-or operation. There are also four columns (C[0::3]) which are staggered as shown in Figure 2. As previously mentioned, the shaded pipeline stages are part of the BRAM or DSP blocks, not "traditional" user logic in the form of CLB flipflops— our goal was to minimize the use of these resources.

The eight pipeline stages in the first thirteen clock cycles along with the plaintext at the top is fed in four 32 bit words, one word per cycle. The first column output E0 0 is produced on the 8th clock cycle and is fed back to the input for processing the second round. Notice that the corresponding outputs are produced as defined by Equation 1. For the second round, after eight clock cycles, the control logic chooses the feedback rather than the plaintext input using a 32 bit 2:1 mux. Our decision to add two pipeline stages to interleave two plaintexts is apparent, as we can see that each pipeline stage is performing an operation only four out of every eight cycles, Which allows us to feed two consecutive 128 bit blocks one after another, in effect doubling our throughout without any additional complexity.

A grayed-out multiplexer is an alternative input, which makes it easy to perform the XOR operation of the key and initial input prior to the first round. For four consecutive clock cycles, C0's DSP performs the XOR operation while the output passes through the other DSPs; this results in the initial round input appearing at the top of the pipeline and the sequence continues as previously described. We have implemented this design, and noticed an expected slight degradation in performance due to the insertion of a 32 bit 2:1 mux in the datapath. There are also additional signals required for the control logic, but those do not affect performance. Finally, we also tried a different approach for computing columns using the same basic structure, but instead of saving the output of each DSP to the one on its right, the data is fed back onto itself for the next XOR operation with the data arriving from the BRAM. We found, however, that this requires the input of a key to each DSP block, extra control logic, different operating modes for the DSP, and a 32 bit 4:1 mux to choose between the output of each DSP for the feedback loop. All those introduced extra delays when routed, and performed worse than the original design. Up to now we focused on the encryption process, though decryption is quite simply achieved with minor modifications to the design. The T-tables are different for encryption and decryption, storing them all would require double the amount of storage, which we want to avoid. Recall, however, that any Ti can be converted into Tj simply by shifting the appropriate amount of bytes. The modification to the design is therefore, replacing the 32 bit 2:1 mux at the output of the BRAM with a 4:1 mux such that all possible byte shifting is possible, and loading the BRAMs with T_i^E, T_i^E, T_i^D and T_i^D, T^E and T^D denoting encryption and decryption

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

T-tables, respectively. Of course, this would degrade the performance because the datapath between BRAM and DSP are now longer. An alternative is to dynamically reconfigure the content of the BRAMs with the decryption T-tables; this can be done from an external source, or even from within the FPGA using the internal configuration access port (ICAP) with a storage BRAM for reloading content through the T-table BRAMs' data input port.

A. Round and Unrolled Modules

Since the single AES round requires the computation of four 32 bit columns, we can propagate the basic construct four times and sum up of 8, 16, and 24 bit registers at the inputs of the columns. The first four instances is shown in Figure 3, one byte is fed back to the same instance while 3 bytes are distributed to the other three instances. The latency of this construct is still 80 clock cycles as before, but allows us to interleave eight 128 bit inputs at any given time. This is possible because of the eight pipeline stages, where each of the four instances receives a 32 bit input every clock cycle. As apposed to the previous module, though, the byte arrangements allow that the T-tables be static so the 32 bit 2:1 multiplexers are no longer required. This simplifies the data paths between the BRAMs and DSPs since the shifting can be fixed in routing. The control logic is simple as well, comprising of a 3 bit counter and a 1 bit control signal for choosing the last round's T-tables.

B. Finally, the natural thing to do was to implement a fully unrolled AES design for achieving maximum throughput. For this, we connected ten instances of the "round" design presented above for an 80-stage pipeline using 80 BRAMs[25] and 160 DSP blocks. Since this design does not require any dynamic control logic it produces a 128 bit output every clock cycle. The initial XOR of the input block with the main key can be done by adding one to four DSP blocks (amount will affect latency) as a pre-stage to the round operation, or be performed in "regular" logic. We now move on to performance results.

a. The Encyrption Flow

C. Round_Key_Encrypt
D. Tmp = Add Round Key (Data, Round_Key_Encrypt [0])
E. For round = 1-9 or 1-11 or 1-13:
F. Tmp = ShiftRows (Tmp)
G. Tmp = SubBytes (Tmp)
H. Tmp = MixColumns (Tmp)
I. Tmp = AddRoundKey (Tmp, Round_Key_Encrypt [round])
J. end loop
K. Tmp = ShiftRows (Tmp)
L. Tmp = SubBytes (Tmp)
M. Tmp = Add RoundKey (Tmp, Round_Key_Encrypt [10 or 12 or 14])
N. Result = Tmp

a. Decryption Flow

O. Round_Key_Decrypt
P. Tmp = Add Round Key (Data, Round_Key_Decrypt [0])
Q. For round = **1-9** or **1-11** or **1-13**:
R. Tmp = InvShiftRows (Tmp)
S. Tmp = InvSubBytes (Tmp)
T. Tmp = InvMixColumns (Tmp)
U. Tmp = AddRoundKey (Tmp, Round_Key_Decrypt [round])
V. end loop
W. Tmp = InvShift Rows (Tmp)
X. Tmp = InvSubBytes (Tmp)
Y. Tmp = AddRoundKey (Tmp, Round_Key_Decrypt [**10** or **12** or **14**])
Z. Result = Tmp

IV. SOFTWARE IMPLEMENTION

A. Xilinx

Xilinx ISE (Integrated Software Environment) is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize ("compile") their designs, examine RTL diagrams, perform timing analysis, simulate a design's reaction to different stimulus and configure the target board with the programmer. In the present work Xilinx 13.2 ISE is being used for coding and development along with ADEPT from digilent for hardware linking.

B. Verilog Coding language

Verilog HDL is one of the two most common Hardware Description Languages (HDL) used by integrated circuit (IC) designers. The alternative one is VHDL. HDL allows the design to be simulated earlier in the design cycle in order to correct errors or experiment with different architectures. Designs described in HDL are technology-independent, easy to design and debug, and are usually more readable than schematics, particularly for large circuits.

Verilog can be used to describe designs at four levels of abstraction:

1. Algorithmic level (much like c code with if, case and loop statements).
2. Register transfer level (RTL uses registers connected by Boolean equations).
3. Gate level (interconnected AND, NOR etc.).
4. Switch level (the switches are MOS transistors inside gates).

The language also defines constructs that can be used to control the input and output of simulation.

V. TESTING AND RESULTS

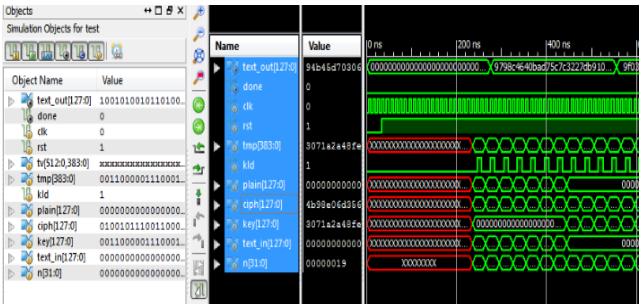

Fig 4: Simulation results for 128 bit with design utilities

The Figure 4 depicts the encryption rate for 128 bit data being fed. The various column of the waveform includes the declared variables in the code along with the predefined clock and data_out signal as shown above. This result has been verified on the Spartan 6 kit. The encryption rate for 128 bit data has been found to be 1.92 Gbits/sec.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

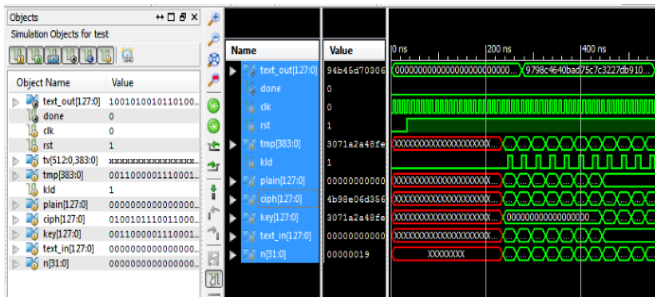

Fig 5: simulation results for 128 bit data input with varied time slice

The above Figure 5 shows the varied time slice output for the 128 bit encryption which also feds the null data on the text_out line as seen above the rate of encryption being approximately same with a slight variation and is found to be 1.89 Gbits/sec.

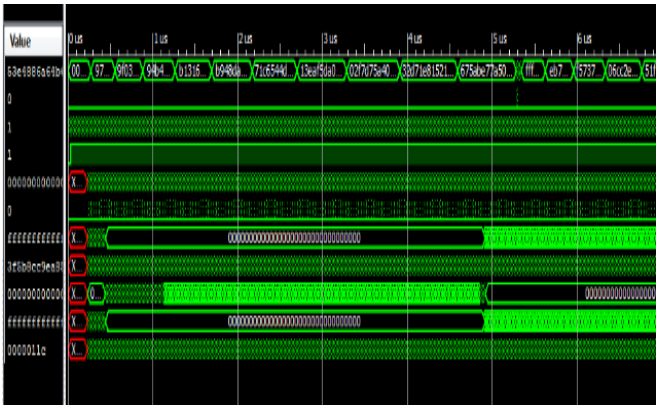

Fig 6: simulation results for 176 bit data input

The 176 bit data encryption has been successfully achieved on the Spartan 6kit and has been verified with the rate of encryption for the same 128 bit cipher key along with the 176 input data. The rate of encryption observed here is 1.82Gbits/sec.

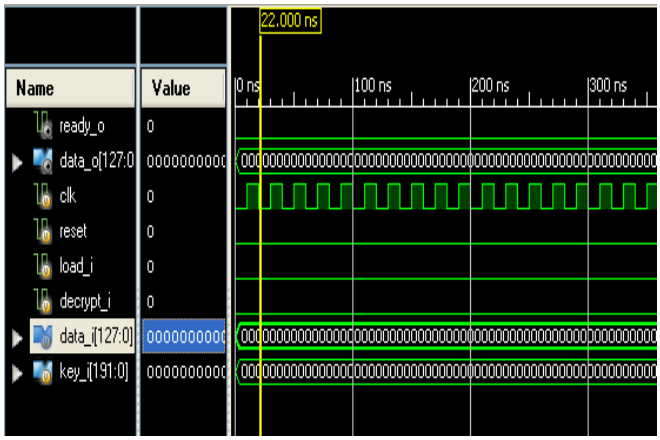

Fig 7: simulation results for 192 bit data input

The Figure 7 depicts the encryption rate for 192 bit data being fed. The cipher key being 128 bit for encryption. The waveform also shows the plain text as well as the cipher key with text output. The encryption rate for 192 bit data has been found to be 1.73 Gbits/sec.

VI. CONCLUSION

The In the proposed work various ways for performing AES operations at a high throughput using on-chip RAM and DSP blocks with minimal use of traditional user logic such as flip-flops and look-up tables has been successfully verified. The encryption rate of the 128,176 and 192 bits is achieved for a better throughput and also the performance valuations when made resulted in the better throughput for 176 and 192 bits with the better resource utilization with the encryption rate of 1.92, 1.89 and 1.73 Gbits/sec for 128, 176 and 192 bits respectively and this revealed that the higher bit configurations lead to higher security level of data which can be accounted for encryption as well as decryption. The encryption rate for the 128,176 and 192 bits has been achieved with the Block RAMs and four DSP48F processor which had a high throughput characteristic. The Spartan 6 FPGA kit usage in the work also revealed that along with the earlier AES implementation done with Vertex 5, Spartan 3 boards the throughput results were achieved.

ACKNOWLEDGMENT

I would like to take this opportunity to express my deep sense of gratitude to **Dr.V.Venkateswarlu**, HOD and Principal, VTU Extension Centre, UTL Technologies Limited, Bangalore, for his invaluable inspiration and guidance without which the project work would not have progressed to its present state.I cannot forget the constant encouragement and help provided by my internal guide, **Mrs. Leelavathi G**, Visiting Professor, VTU Extension Centre, UTL Technologies Limited, Bangalore, who considered me like a friend and made me feel at ease in times of difficulty during my project work. I express my sincere gratitude to him. I do not know how to express my thanks to my external guide **Mr. Ganesh Kumar P. V.**, Project manager, Ultraa Chipp Tecchnologies, Bangalore, who provided constant encouragement whenever I faced difficulties in my project work. I express my sincere thanks to him.I would finally like to extend my thanks to all the teaching and non teaching staff both at VTU Extension Centre, UTL Technologies Limited, Bangalore.

REFERENCES

[1]. A. J. Elbirt, W. Yip, B. Chetwynd, and C. Paar. “An FPGA based performance evaluation of the AES block cipher candidate algorithm finalists”. IEEE Transactions on Very Large Scale Integration Systems (VLSI), vol. 9, no. 4, pp. 545–557, 2001.

[2]. V. Fischer and M. Drutarovsk´y. “Two methods of Rijndael implementation in reconfigurable hardware”. In Cryptographic Hardware and Embedded Systems (CHES), vol. 2162, pp. 77–92, 2001.

[3]. K. Gaj and P. Chodowiec. “Very compact FPGA implementation of the AES algorithm”. In CHES, vol. 2779, pp. 319–333, 2003.

[4]. F.-X. Standaert, S. B. O¨ rs, and B. Preneel. “Power analysis of an FPGA implementation of Rijndael: Is pipelining a DPA countermeasure?” In CHES, vol. 3156 of LNCS, pp. 30–44, London, UK, 2004. Springer.

[5]. V. Fischer and M. Drutarovsk´y. “Two methods of Rijndael implementation in reconfigurable hardware.” In Cryptographic Hardware and Embedded Systems (CHES), vol. 2162, pap. 77–92, 2001.

[6]. K. Gaj and P. Chodowiec. “Very compact FPGA implementation of the AES algorithm.” In CHES, vol. 2779, pap. 319–333, 2003.

[7]. Helion Technology Ltd. High performance AES (Rijndael) cores for Xilinx FPGAs, 2007 (Rev. 2.3.3). aes_xilinx_helioncore.pdf.

[8]. A. Hodjat and I. Verbauwhede. “A 21.54 Gbits/s fully pipelined AES processor on FPGA.” In Field-Programmable Custom Computing Machines, pp. 308–309. IEEE Computer Society, 2004.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

[9]. T. Ichikawa, T. Kasuya, and M. Matsui. “Hardware evaluation of the AES finalists.” AES Candidate Conference, pp. 13–14, 2000.
[10]. D. Kotturi, S.-M. Yoo, and J. Blizzard. “AES crypto chip utilizing high-speed parallel pipelined architecture.” In IEEE International Symposium on Circuits and Systems, pp. 4653–4656, 2005.

Mr. Sagar Deshpande is pursuing his final year M.Tech degree in VLSI Design and Embedded Systems at VTU Extension Center, UTL Technologies Ltd., Bangalore. His research interest include embedded systems and MEMS.

Mrs. Leelavathi G. working as a visiting professor in Dept. of VLSI Design and Embedded Systems at VTU Extension Centre, UTL Technologies Ltd., Bangalore. Her research interest include embedded systems.