PARALLEL PROCESSING OF AES ENCRYPTION IMPLEMENTATION ON ASAP

M.Meiyarasu,
mmeiyarasuece@gmail.com,
Department of Electronics and Communication Engineering,
K.S.Rangasamy College of Technology

Abstract: In the development of information technology, protecting sensitive information via encryption is becoming more and more important in daily life. The Advanced Encryption Standard (AES) is an encryption standard preferred by the National Institute of Standards and Technology (NIST) in 2001, which has its extract in the Rijndael block cipher. By searching different coarse nessesof data-level and task-level parallelism, AES is represented with four implementations expansion on Asynchronous Array of Simple Processor (ASAP). The smallest design apply only six cores, while the largest requires 107 cores for key expansion and 137 cores for key expansion, respectively. In comparison with AES cipher four implementations on Asynchronous Array of Simple Processor (ASAP), the result of parallelism design is being a higher throughput per unit of chip area and has higher energy efficiency.

1. INTRODUCTION

1.1.Aes

AES (Advanced Encryption Standard) is an algorithm for executing encryption (and the reverse, decryption) which is a series of clear steps that can be followed as a procedure. The pilot data is known as plaintext, and the encrypted form as cipher text. The cipher text message holds all the information of the plaintext message, but is not in a format readable by a human or computer without the thoroughly mechanism to decrypt the original text. The encrypting process is changed depending on the key which alters the detailed procedure of the algorithm. Minus the key, the cipher cannot be used to encrypt or decrypt. In the past, cryptography assisted check secrecy in significant communications, such as those of government cover procedures, military leaders, and diplomats. Cryptography has come to be in widespread use by many civilians who do not have extraordinary necessitates for secrecy, while generally it is transparently built into the base for calculating and telecommunications.

1.2. Encryption and Decryption

In cryptography, encryption is the process of encoding messages in such a way that eavesdroppers or hackers cannot read it, but only authorized parties can. Encryption doesn't prevent hacking but it prevents the hacker from reading the data that is encrypted. In an encryption, the message or information encrypted using an encryption algorithm. The information that is being encrypted is called plaintext and the information obtained after encryption is called cipher text. This is usually done with the use of an encryption key, an authorized party, however, is able to decode the cipher text using a decryption algorithm that usually requires a secret decryption key, which adversaries do not have access to. For technical reasons, an encryption scheme usually needs a key-generation algorithm to randomly produce keys.

1.3. Crpytographic Algorithms

1.3.1. Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a specification for the encryption of electronic data based on the Rijndael cipher. Rijndael is a family of ciphers with different key and block sizes. AES uses three members of the Rijndael family, each with a block size of 128 bits, but three different key lengths: 128, 192 and 256 bits.

The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data. AES is based on a design principle known as a substitution-permutation network, and is fast in both software and hardware. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification per size is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits.

AES operates on a 4×4 column-major order matrix of bytes, termed the state, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field.

The key size used for an AES cipher specifies the number of repetitions of transformation rounds that converts the plaintext into the cipher text. The number of cycles of repetition are as follows:

- ✓ 10 cycles of repetition for 128-bit keys.
- ✓ 12 cycles of repetition for 192-bit keys.
- ✓ 14 cycles of repetition for 256-bit keys.

Each round consists of several processing steps, each containing five similar but different stages, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform cipher text back into the original plaintext using the same encryption key.

1.4. Asynchronous Array of Simple Processors (ASAP)

The targeted Asynchronous Array of Simple Processors architecture is an example of a fine-grained many-core computation platform, supporting globally-asynchronous locally-synchronous (GALS) on-chip network and dynamic voltage and frequency scaling (DVFS). The computational platform is composed of 164 small identical processors, three hardware accelerators and three 16 KB shared memories. All processors and shared memories are clocked by local fully independent oscillators and are connected by a reconfigurable 2D-mesh network that supports both nearby and long-distance communication. Each tile on the platform can be statically configured to take input data from two links, while sending its output to other processors via dynamic configuration. Each simple processor has a 6-stage pipeline, which issues one instruction per clock cycle. Moreover, no application specific instructions are implemented. Each processor has a 128 x 32-bit instruction memory and a 128 x 16-bit data memory.

2. METHODOLOGY

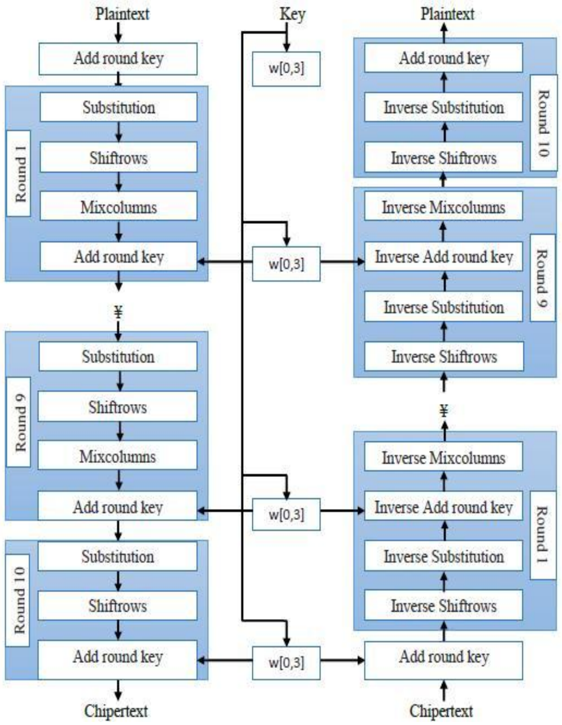


Fig. 2.1 Encryption and Decryption process of AES structure

2.1. Encryption Steps

1. KeyExpansion- round keys are derived from the cipher key. AES requires a separate 128 bit round key block for each round plus one more.

2. InitialRound

Addroundkey- each byte of the state is combined with a block of the round key using bitwise xor.

3. Rounds

Substitution- a non-linear substitution step where each byte is replaced with another according to a lookup table.
Shiftrows- a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
Mixcolumns- a mixing operation which operates on the columns of the state, combining the four bytes in each column.

Addroundkey- each byte of the state is combined with a block of the round key using bitwise xor.

4. Final Round (no Mixcolumns)

Substitution
Shiftrows
Addroundkey.

2.2. Substitution Transformation

The Substitution is a byte substitution operation performed on individual bytes of the State, on each byte of the State using a substitution table (S-box).

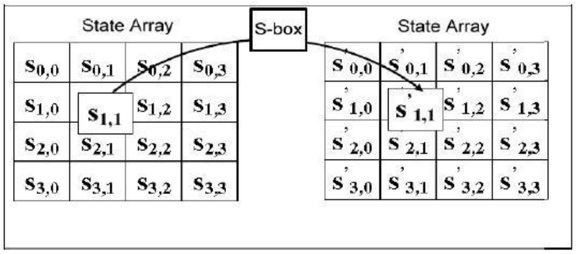
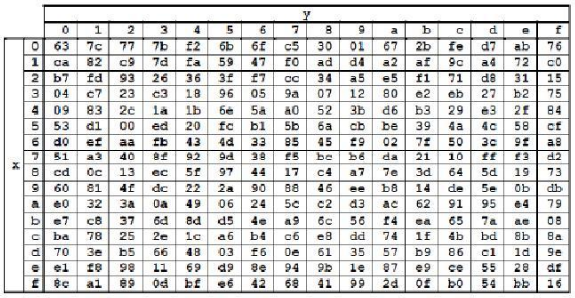

State Array
S-box
State Array
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$S'_{0,0}$ $S'_{0,1}$ $S'_{0,2}$ $S'_{0,3}$
$S'_{1,0}$ $S'_{1,1}$ $S'_{1,2}$ $S'_{1,3}$
$S'_{2,0}$ $S'_{2,1}$ $S'_{2,2}$ $S'_{2,3}$
$S'_{3,0}$ $S'_{3,1}$ $S'_{3,2}$ $S'_{3,3}$

Fig. 2.2 Substitution Transformation

The S-box used in the SubBytes transformation is presented in hexadecimal form in Fig. 3.1. For example, if S1,1 = {53}, then the substitution value would be determined by the intersection of the row with index „5‟ and the column with index „3‟ in Fig. 3.1. This would result having a value of {ed}.

Table 2.1 S-Box table



		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

2.3. Shiftrow Transformation

The Shiftrows transformation cyclically shifts the last three rows of the state by different offsets. The first row is left unchanged in this transformation. Each byte of the second row is shifted one position to the left. The third and fourth rows are shifted left by two and three positions, respectively.

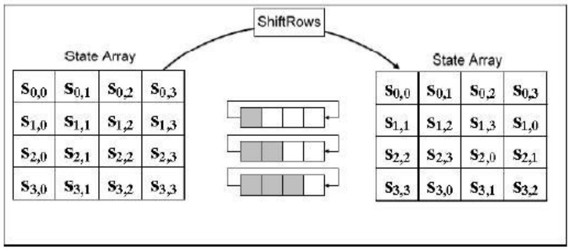

ShiftRows
State Array
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
State Array
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,1}$ $S_{1,2}$ $S_{1,3}$ $S_{1,0}$
$S_{2,2}$ $S_{2,3}$ $S_{2,0}$ $S_{2,1}$
$S_{3,3}$ $S_{3,0}$ $S_{3,1}$ $S_{3,2}$

Fig. 2.3 Shiftrows Transformation

2.4. Mixcolumns Transformation

This transformation operates on the columns of the State, treating each columns as a four term polynomial finite field. Each columns is multiplied modulo with a fixed four-term polynomial. The Mixcolumns transformation can be expressed as a matrix multiplication.

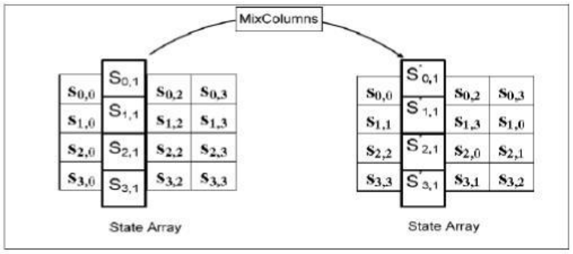

MixColumns
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
State Array
$S_{0,0}$ $S'_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,1}$ $S'_{1,1}$ $S_{1,3}$ $S_{1,0}$
$S_{2,2}$ $S'_{2,1}$ $S_{2,0}$ $S_{2,1}$
$S_{3,3}$ $S'_{3,1}$ $S_{3,1}$ $S_{3,2}$
State Array

Fig. 2.4 Mixcolumns Transformation

2.5. Addroundkey Transformation

During the Addroundkey transformation, the round key values are added to the State by means of a simple Exclusive Or (XOR) operation. Each round key consists of words that are generated from the Key Expansion routine. The round key values are added to the columns of the state.

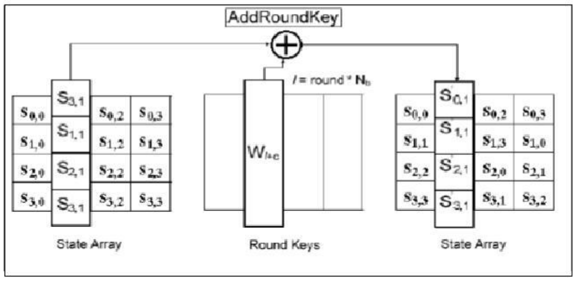

AddRoundKey
l = round * N_b
$S_{0,0}$ $S_{3,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
State Array
W_{l+c}
Round Keys
$S_{0,0}$ $S'_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,1}$ $S'_{1,1}$ $S_{1,3}$ $S_{1,0}$
$S_{2,2}$ $S'_{2,1}$ $S_{2,0}$ $S_{2,1}$
$S_{3,3}$ $S'_{3,1}$ $S_{3,1}$ $S_{3,2}$
State Array

Fig. 2.5 Addroundkey Transformation

2.6. Asynchronous Array of Simple Processors (ASAP)

The second generation ASAP processor contains 167 identical processors with independent clock domains. Each processor is a reduced complexity programmable DSP with small memories, which can dramatically increase the energy efficiency. Each processor can receive data from any two neighbours and send data to any of its four neighbours.

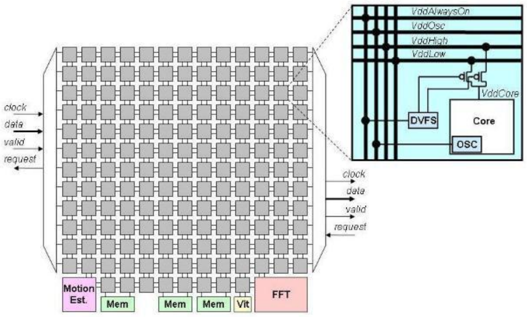


Fig. 2.6 ASAP Processor structure

A second generation 65 nm CMOS design contains 167 processors with dedicated fast Fourier transform (FFT), Viterbi decoder, and video motion estimation processors; 16 KB shared memories; and long-distance inter-processor interconnect. The programmable processors can individually and dynamically change their supply voltage and clock frequency. The chip is fully functional. Processors operate up to 1.2 GHz at 1.3 V which is believed to be the highest clock rate fabricated processor designed in any university. At 1.2 V, they operate at 1.07 GHz and 47 mW when 100% active. At 0.675 V, they operate at 66 MHz and 608 μW when 100% active.

This operating point enables 1 trillion MAC or arithmetic logic unit (ALU) ops/sec with a power dissipation of only 9.2 watts. Due to its MIMD architecture and fine-grain clock oscillator stalling, this energy efficiency per operation is almost perfectly constant across widely varying workloads, which is not the case for many architectures.

2.7. AES Implementations on ASAP

2.7.1. Full-Parallelism

The Full-parallelism AES implementation combines the Parallel-SubBytes-MixColumns model and loop unrolling. The dataflow diagram and the mapping of the Full-parallelism model are shown in Fig. 2.7. As expected, the throughput of this design is the highest among all of the models introduced in this paper since it employs most data and task parallelism.

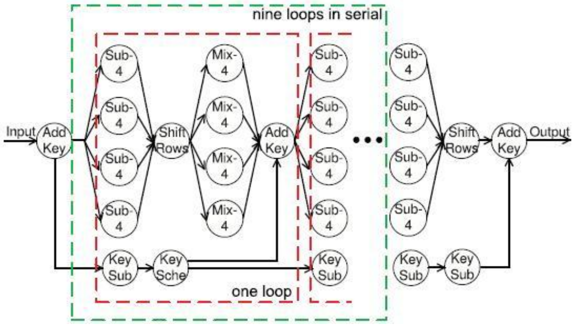


Fig. 2.7 Full-Parallelism Data flow diagram

2.7.2. Loop unrolled three times

To achieve a moderate throughput, I could unroll the main loops in the AES algorithm by S times (S is divisible by Nr-1), instead of Nr-1 times. For this example, the nine loops in the AES algorithm could be split into three blocks, and each block loops three times. The dataflow diagram and mapping of the Loop unrolled Three Times implementation are shown in Fig. 2.8 respectively.

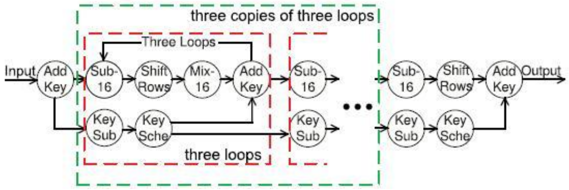


Fig. 2.8 Loop unrolled three times Dataflow diagram

2.7.3. Loop Unrolled Nine Times

To enhance the AES cipher‟s throughput, I apply loop unrolling model and obtain the Loop-unrolled Nine Times dataflow diagram as shown in Fig. 2.9. The loop unrolling breaks the dependency among different loops and allows the nine loops in the AES algorithm to operate on multiple data blocks simultaneously. To improve the throughput as much as possible, I unroll the loops in both the AES algorithm and the key expansion process by Nr-1 and Nr times, which equals 9 and 10, respectively.

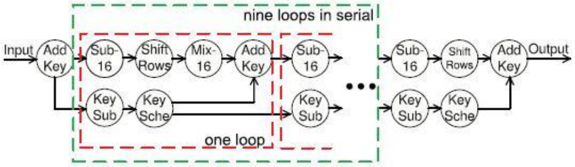


Fig. 2.9 Loop Unrolled Nine Times Dataflow diagram

2.7.4. Parallel Mixcolumn

In the Parallel MixColumns implementation, SubBytes-16 requires clock cycles to encrypt one data block, which contributes the largest execution delay in one loop. In order to increase the throughput further, I parallelize one SubBytes-16 into four SubBytes-4s, which is shown in Fig. 2.10. In this implementation, each SubBytes-4 processes 4 bytes rather than 16 bytes in one data block. The effective execution delay of the SubBytes process is decreased to clock cycles per block, only around one fourth as before. Therefore, the throughput of the Parallel MixColumns model is increased to clock cycles per block.

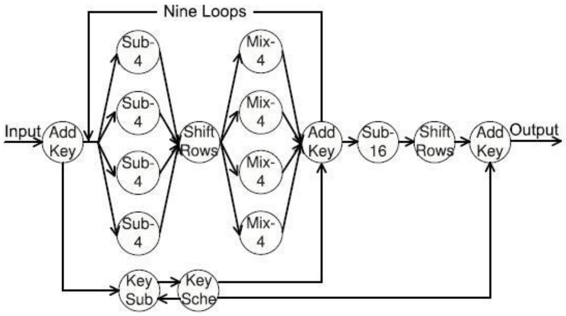


Fig. 2.10 Parallel Mixcolumn Dataflow diagram

3. RESULTS AND DISCUSSION

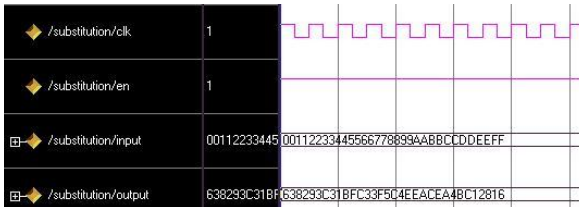
3.1. Tools Used

The Block diagram of the AES and Architecture of the ASAP is coded in Verilog Hardware Description Language (HDL). The whole Architecture is split into several sub modules, each sub modules are then coded separately and are instantiated in the main module. Then the code is compiled and synthesized using Xilinx ISE 13.4 Design Suite and is simulated using the Isim 13.4 software. Then the power, Area and throughput is compiled in design compiler using Synopsys tool.

4. SIMULATION RESULTS

4.1. Substitution Transformation

This Substitution transformation is basic part of the AES unit. It consists of three inputs clk, en and substitution input and one substitution output. Both the inputs and outputs are 128 bit numbers. The Substitution is a byte substitution operation performed on individual bytes of the State.



```
/substitution/clk      1
/substitution/en       1
/substitution/input    00112233445  00112233445566778899AABBCCDDEEFF
/substitution/output   638293C31BF  638293C31BFC33F5C4EEACEA4BC12816
```

Fig. 4.1 Substitution Transformation Result

4.2. Shiftrows Transformation

This Shiftrows transformation is basic part of the AES unit. It consists of three inputs clk, en and shiftrow input and one shiftrow output. Both the inputs and outputs are 128 bit numbers. The Shiftrows transformation cyclically shifts the last three rows of the state by different offsets.

The first row is left unchanged in this transformation. Each byte of the second row is shifted one position to the left. The third and fourth rows are shifted left by two and three positions, respectively.

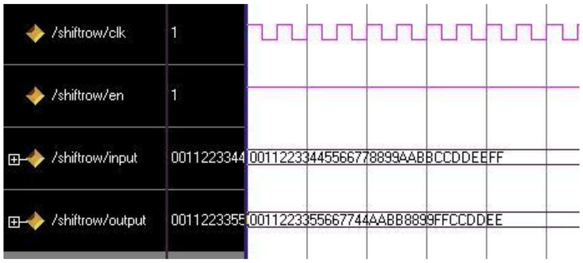


Fig. 4.2 Shiftrows Transformation Result

4.3. Mixcolumns Transformation

This Mixcolumn transformation is basic part of the AES unit. It consists of three inputs clk, en and Mixcolumn input and one Mixcolumn output. Both the inputs and outputs are 128 bit numbers. The Mixcolumns transformation can be expressed as a matrix multiplication.

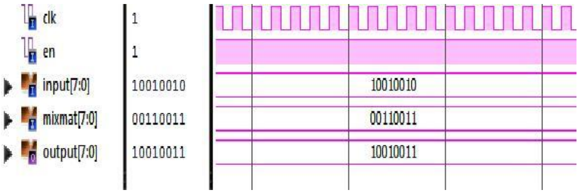


Fig. 4.3 Mixcolumns Transformation Result

4.4. Addroundkey Transformation

This Addroundkey transformation is basic part of the AES unit. It consists of four inputs clk, en input1 and input2 and one Addroundkey output. Input1, Input2 inputs and outputs are 128 bit numbers. Input1 and Input2 are preformed XOR operation.

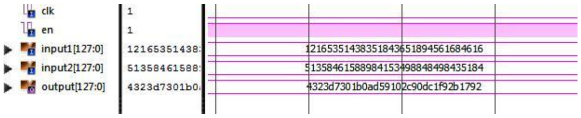


Fig. 4.4 Addroundkey Transformation Result

4.5. Encryption and Decryption

It consists of five inputs clk, ctrl, input encryption, input decryption and input key and two outputs output encryption and decryption. Input encryption, input decryption, input key, output decryption and output encryption are 128 bits.

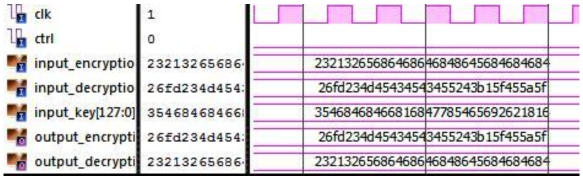


Fig. 4.5 Encryption and Decryption Result

4.6. Full-Parallelism

Encryption and decryption process is performed parallel.

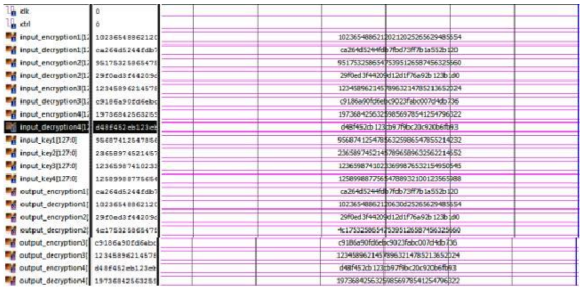


Fig. 4.6 Full-Parallelism Result

4.7. Loop unrolled three times

Encryption and decryption of AES algorithm could be split into three blocks, and each block loops are performed three times.

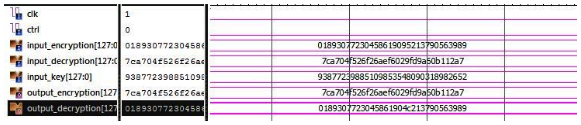


Fig. 4.7 Loop unrolled three times Result

4.8. Loop Unrolled Nine Times

Encryption and decryption of AES algorithm could be split into three blocks, and each block loops are performed one time.

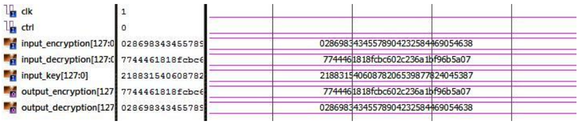


Fig. 4.8 Loop unrolled nine times Result

4.9. Parallel Mixcolumn

In parallel Mixcolumn using nine loop performed by mixcolumnopertaions.

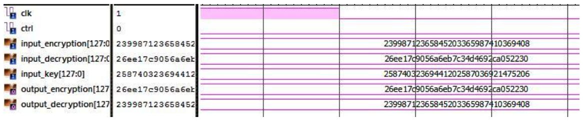
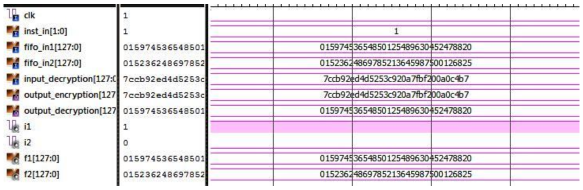


Fig. 4.9 Parallel Mixcolumn Result

4.10. AES implementation on ASAP

Finally four methodology (Full-Parallelism, Loop unrolled 3 times, Loop Unrolled Nine Times and Parallel Mixcolumn) implemented on ASAP.



```
clk                        1
inst_in[1:0]               1                  1
fifo_in1[127:0]            015974536548501    01597453654850125489630452478820
fifo_in2[127:0]            015236248697852    01523624869785213645987500126825
input_decryption[127:0     7ccb92ed4d5253c    7ccb92ed4d5253c920a7fbf200a0c4b7
output_encryption[127      7ccb92ed4d5253c    7ccb92ed4d5253c920a7fbf200a0c4b7
output_decryption[127      015974536548501    01597453654850125489630452478820
i1                         1
i2                         0
f1[127:0]                  015974536548501    01597453654850125489630452478820
f2[127:0]                  015236248697852    01523624869785213645987500126825
```

Fig. 4.10 AES implementation on ASAP Result

4.11. Synopsys tool

Since the AES ciphers presented in this work are implementedon a programmable platform without any application- specific hardware. Here used the synopsys tool to analyze the Area, power and throughput for ASAP processor. By using 90 nm technology compile the program on tool.

Table 4.1. Synopsys result

Area	throughput	power
8.991 nm	0.74 cycles/byte	1.81 mw

5. CONCLUSION

Advanced Encryption Standard (AES) cipher is presented with 16 different implementations, each of which exploits different levels of data and task parallelism. The smallest design requires only six processors, equaling in an Asynchronous Simple of Array Processor (ASAP) system. The fastest design achieves a high throughput of per byte. It also optimizes the area of each implementation by examining the workload of each processor, which reduces the number of cores used as much as 18 percent. The design on the ASAP system achieves energy efficiencies approximately 2.9-18.1 higher than other software platforms and performance per area on the order of 3.3-15.6 higher. Overall, the ASAP system has been demonstrated to be a very promising platform for software AES implementations.

6. REFERENCE

[1] Bin Liu, Student Member, IEEE, and Bevan M. Baas, Senior Member, IEEE "Parallel AES Encryption Engines for Many-Core Processor Arrays," IEEE Transactions on Computers, vol. 62, no. 3, pp. 536-547, March 2013.

[2] NIST, "Advanced Encryption Standard (AES)," http://csrc.nist.-gov/publications/fips/fips197/fips-197.pdf, Nov. 2001.

[3] NIST, "Data Encryption Standard (DES)," http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf, Oct. 1999.

[4] J. Granado-Criado, M. Vega-Rodriguez, J. Sanchez-Perez, and J. Gomez-Pulido, "A New Methodology to Implement the AES Algorithm Using Partial and Dynamic Reconfiguration," Integration, the VLSI J., vol. 43, no. 1, pp. 72-80, 2010.

[5] S. Morioka and A. Satoh, "A 10-gbps full-AES Crypto Design with a Twisted BDD s-Box Architecture," IEEE Trans. Very Large Scale Integration Systems, vol. 12, no. 7, pp. 686-691, July 2004.

[6] A. Hodjat and I. Verbauwhede, "Area-Throughput Trade-Offs for Fully Pipelined 30 to 70 Gbits/s AES Processors," IEEE Trans. Computers, vol. 55, no. 4, pp. 366-372, Apr. 2006.

[7] S.K. Mathew, F. Sheikh, M. Kounavis, S. Gueron, A. Agarwal, S.K. Hsu, H. Kaul, M.A. Anders, and R.K. Krishnamurthy, "53 gbps Native GF(ð24Þ2) Composite-Field AES-Encrypt/Decrypt Accelerator for Content-Protection in 45 nm High-Performance Microprocessors," IEEE J. Solid-State Circuits, vol. 46, no. 4, pp. 767-776, Apr. 2011.

[8] C.-J. Chang, C.-W. Huang, K.-H. Chang, Y.-C. Chen, and C.-C. Hsieh, "High Throughput 32-Bit AES Implementation in FPGA," Proc. IEEE Asia Pacific Conf. Circuits and Systems, pp. 1806-1809, Nov. 2008.