University of Manouba

National School of Computer Science

ENSI

ÉCOLE NATIONALE DES SCIENCES
DE L'INFORMATIQUE

Implementation of NIOS II processor with AES peripheral in Cyclone III FPGA-ALtera

"Integration Project"

Supervised by:

M.Lobna Kriaa & Mr Mohammed Masmoudi

Realised by :

Mallouli Wassim

Karaa Hela

Academic year 2015-2016

Summary:

I-Introduction

Cryptography plays an important role in the security of data. It enables to store sensitive information or transmit it across insecure networks so that unauthorized persons cannot read it. Advanced Encryption Standard (AES) is the most common encryption algorithm widely used in applications such as wireless communication and also it has been selected by the US government for protecting sensitive social information. Due to its simplicity and elegant algebraic structure.

II- AES Algorithm Description

The AES algorithm is a symmetric-key cipher in whish, the encryption and decryption keys are the same. Thus communicating parties must agree on a secret key before they wish to communicate.

The length of the plain text is fixed to be 128 bits, while the key length can be either 128,192, or 256 bits. The key length selected is of 128 bits.

AES algorithm is an iterative algorithm. Every iteration can be called a round, and the total number of rounds is 10, 12, or 14, when key length is 128, 192, or 256 respectively.

Number of rounds(Nr)	128 Bit Data	192 Bit Data	256 Bit Data
128-Bit key	10	12	14
192-Bit key	12	12	14
256-Bit Key	14	14	14

The 128 bit algorithm is divided into 16 bytes. These bytes are represented into 4x4 array called the state array.

1-Encryptipon Process of the AES algorithm:

The Encryption Process of Advanced Encryption Standard algorithm is presented blow in figure 1 and it shows all the different operations of this algorithm such as:

1. Substitution Bytes

2. Shift Rows

3. Mix Columns

4. Add Round Key

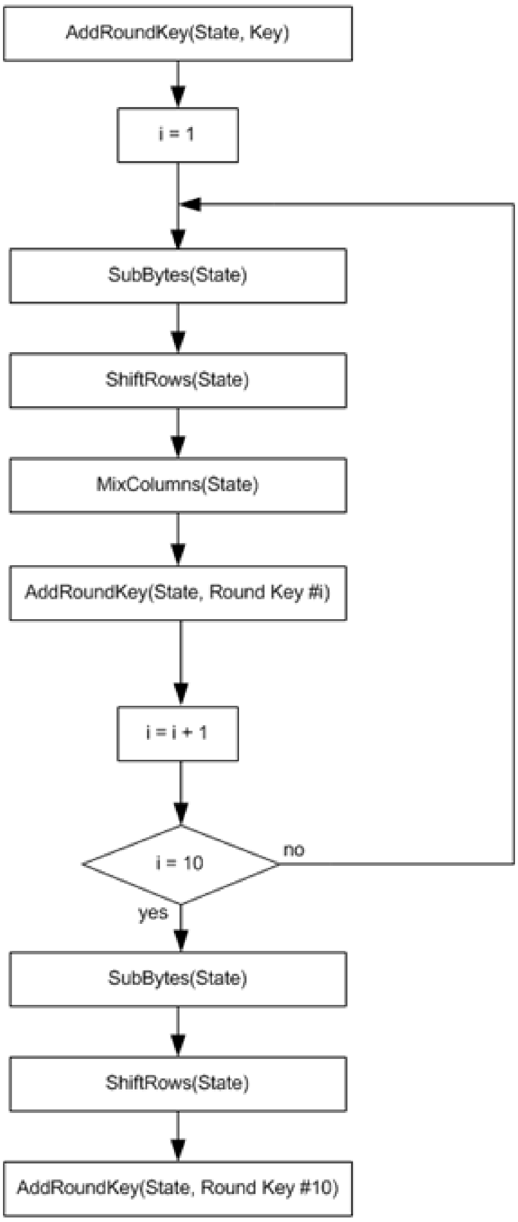
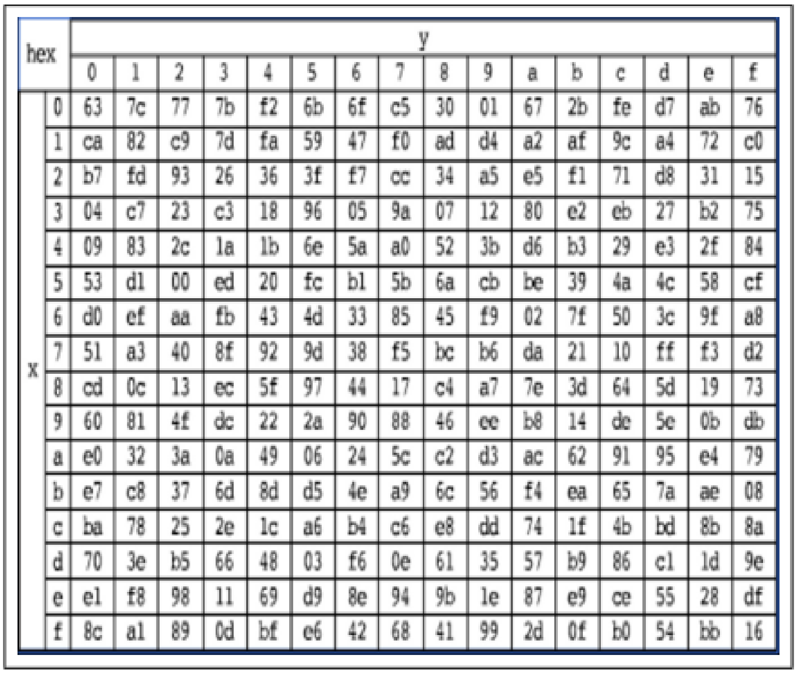


Figure1.1: AES Encryption Process

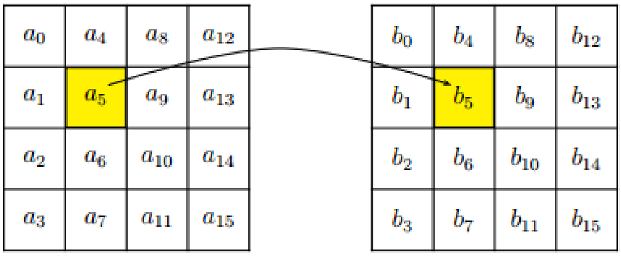
1-1-Substitution Bytes:

Each entry in the state array is of bytes. S-box is a standard substitution table. Every byte in the state array is substituted by the corresponding byte from the S-box. Each byte of the state array is changed.



hex		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

Figure 1.2: Substitution Box (S-Box)


a_0 a_4 a_8 a_{12}
a_1 a_5 a_9 a_{13}
a_2 a_6 a_{10} a_{14}
a_3 a_7 a_{11} a_{15}
b_0 b_4 b_8 b_{12}
b_1 b_5 b_9 b_{13}
b_2 b_6 b_{10} b_{14}
b_3 b_7 b_{11} b_{15}

1-2. Shift Rows:

This is the Second Transformation in the series of 4 Transformations and is extremely simple to implement. It involves rotating the rows of the input matrix circularly upwards.

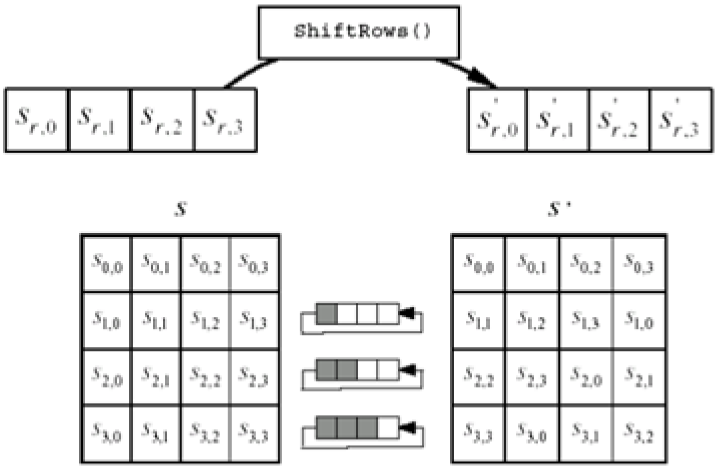
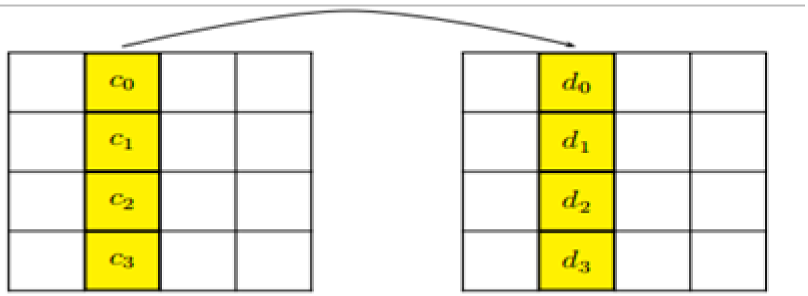


Figure1.3:ShiftRows Method

1-3. Mix columns:

In mix column operation the columns of the State are considered as polynomials over GF (2^8) and are multiplied with a fixed polynomial. The mix column is not used in the last round of the algorithm.



$$S'_{0,c} = (\{02\} \bullet S_{0,c}) \oplus (\{03\} \bullet S_{1,c}) \oplus S_{2,c} \oplus S_{3,c}$$
$$S'_{1,c} = S_{0,c} \oplus (\{02\} \bullet S_{1,c}) \oplus (\{03\} \bullet S_{2,c}) \oplus S_{3,c}$$
$$S'_{2,c} = S_{0,c} \oplus S_{1,c} \oplus (\{02\} \bullet S_{2,c}) \oplus (\{03\} \bullet S_{3,c})$$
$$S'_{3,c} = (\{03\} \bullet S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus (\{02\} \bullet S_{3,c})$$

Figure1.4: Mixcolumns Calculating Method

1-4. Add Round Key

AES addkey: The process is simply the XOR of the current matrix with the corresponding Round Key. Each round key is derived from the cipher key using a key schedule

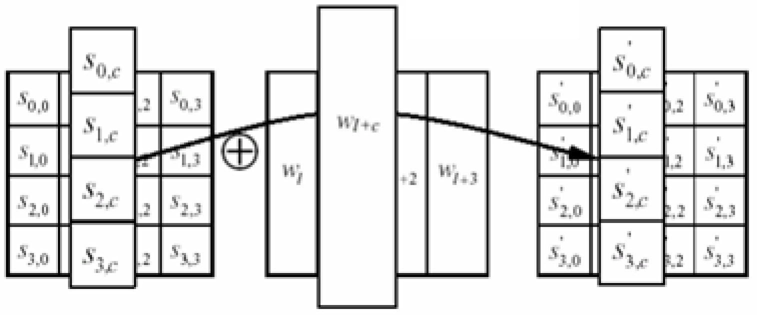


Figure 1.5- Function AddRoundKey

2-Decryptipon Process of the AES algorithm:

The Decryption Process of Advanced Encryption Standard algorithm is presented blow in figure 3 and it shows all the different operations of this algorithm such as:

1. InvSubstitution Bytes
2. InvShift Rows
3. InvMix Columns
4. Add Round Key

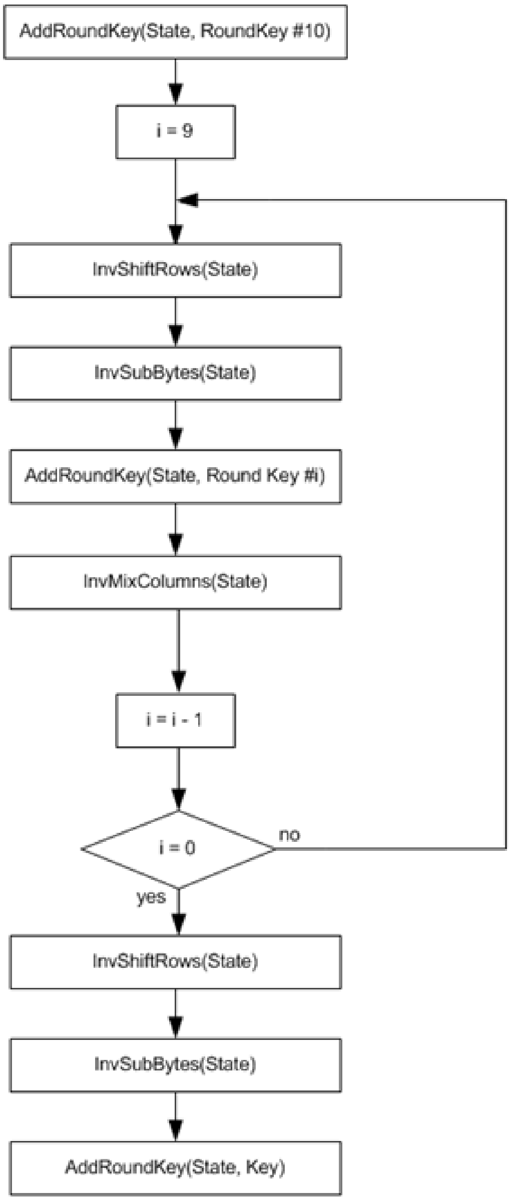
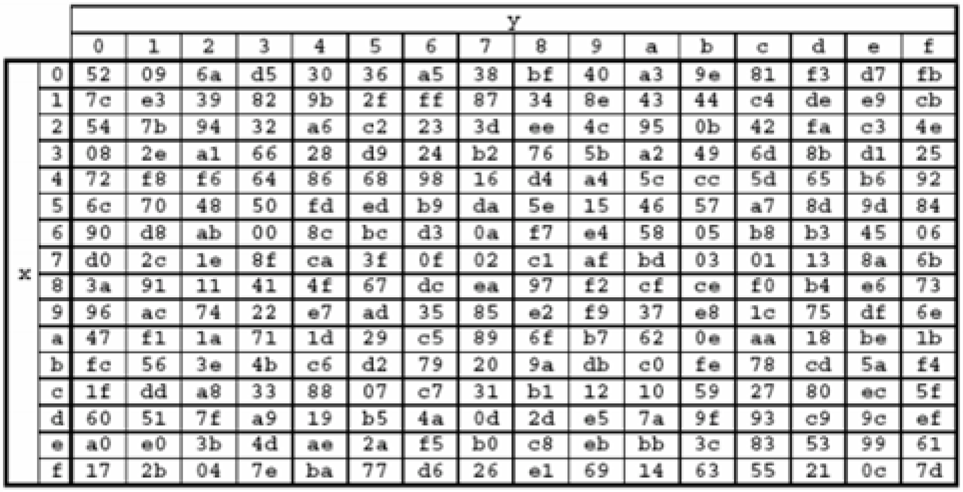


Figure2.1: AES Decryption Process

2-1-InvSubstitution Bytes:

Each entry in the state array is of bytes. Inverse S-box is a standard Inverse substitution table. Every byte in the state array is substituted by the corresponding byte from the inverse S-box. Each byte of the state array is changed.



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

Figure 2.2: Inverse S-Box

2-2. Inverse Shift Rows:

The only difference between InvShiftRow and ShiftRows is the direction of the changes.

InvShiftRows function is shown in this Figure:

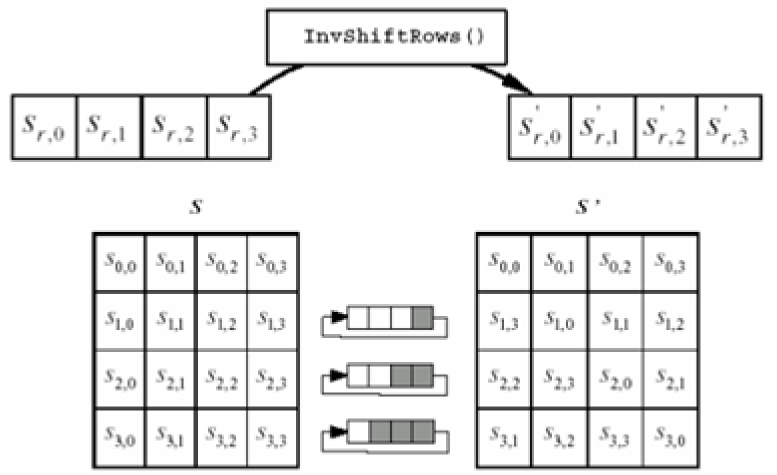

InvShiftRows()
$s_{r,0}$ $s_{r,1}$ $s_{r,2}$ $s_{r,3}$
$s'_{r,0}$ $s'_{r,1}$ $s'_{r,2}$ $s'_{r,3}$
s
s'
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,3}$ $s_{1,0}$ $s_{1,1}$ $s_{1,2}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,1}$ $s_{3,2}$ $s_{3,3}$ $s_{3,0}$

Figure2.3: Inverse ShiftRows

2-2. Inverse Mixcolumns:

The InvMixColumns Method only differs from MixColumns by the way we calculate the column - replacement. The new column is calculated like this (2.4)

$$S'_{0,c} = (\{0e\} \bullet S_{0,c}) \oplus (\{0b\} \bullet S_{1,c}) \oplus (\{0d\} \bullet S_{2,c}) \oplus (\{09\} \bullet S_{3,c})$$

$$S'_{1,c} = (\{09\} \bullet S_{0,c}) \oplus (\{0e\} \bullet S_{1,c}) \oplus (\{0b\} \bullet S_{2,c}) \oplus (\{0d\} \bullet S_{3,c})$$

$$S'_{2,c} = (\{0d\} \bullet S_{0,c}) \oplus (\{09\} \bullet S_{1,c}) \oplus (\{0e\} \bullet S_{2,c}) \oplus (\{0b\} \bullet S_{3,c})$$

$$S'_{3,c} = (\{0b\} \bullet S_{0,c}) \oplus (\{0d\} \bullet S_{1,c}) \oplus (\{09\} \bullet S_{2,c}) \oplus (\{0e\} \bullet S_{3,c})$$

Figure2.4:Mixcolumns: calcul method

III-System Design

In order to implement the AES as a custom hardware, we have to understand first the different characteristics of Altera card so we can use it easily and also to know more about NIOS II processor and the AES peripheral characteristics.

1- Cyclone III FPGA kit from ALTERA

For this project, we had used FPGA Cyclone III of ALTERA's card. Here Below we presented some of the characteristics of this card:

- Cyclone III EP3C25F324
- 25000 Logic elements
- System with USB - Blaster
- Memories :
  - SDRAM : 32 Mbits
  - SRAM : 1 Mbit
  - Flash: 16 Mbits
- Clock Rhythm: oscillator integrated with 50 MHz
- Buttons and Indicators: 7 LEDs and 6 Buttons
- Connectors:
  - HSMC
  - USB Type B
- Pre-programmed with a Nios II design present in the flash
- When the configuration is completed, the Nios II starts executing the boot code presented in the flash.

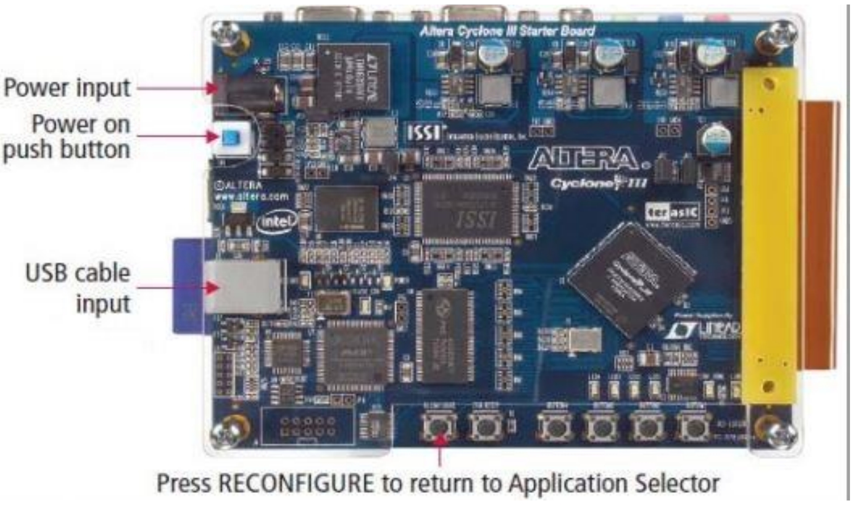

Power input
Power on push button
USB cable input
Press RECONFIGURE to return to Application Selector
Altera Cyclone III Starter Board
ALTERA
Cyclone III
ISSI
terasIC
intel

Figure 3.1: Altera Cyclone III Starter Board

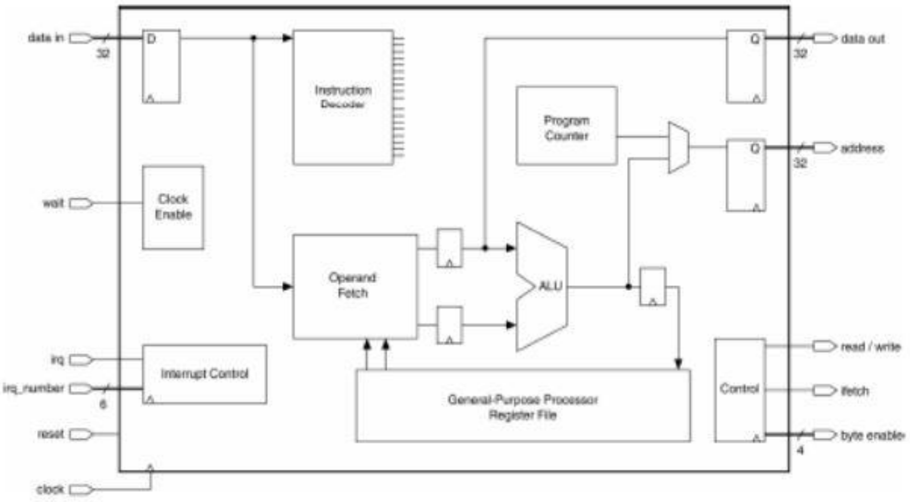
2-NIOS II

Nios II is a 32-bit embedded-processor architecture designed especially for the Altera family of FPGAs.

a- Characteristics of NIOS II

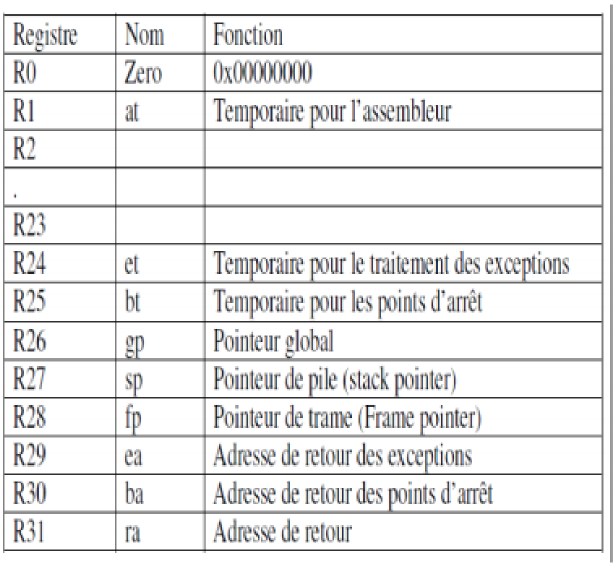
- Harvard Architecture
- Data Master port
- Instruction Master port
- 32 registers of 32 bits

b- NIOS processor Diagram



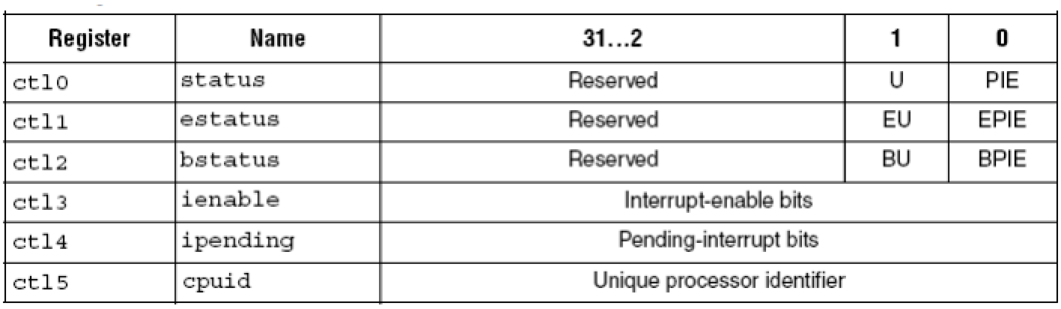
c-General registers of NiosII

- Register r0 contains the 0 constant: we can’t write in this register and it called zero.
- Register r1 is used by the assembler as a temporary register and we can't use it in user programs.
- Registers r24 and r29 are used to deal with exceptions.

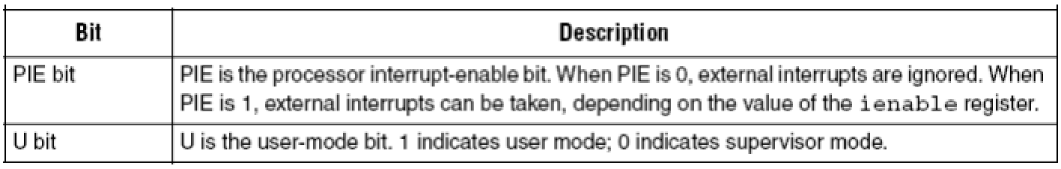


Registre	Nom	Fonction
R0	Zero	0x00000000
R1	at	Temporaire pour l’assembleur
R2		
.		
R23		
R24	et	Temporaire pour le traitement des exceptions
R25	bt	Temporaire pour les points d’arrêt
R26	gp	Pointeur global
R27	sp	Pointeur de pile (stack pointer)
R28	fp	Pointeur de trame (Frame pointer)
R29	ea	Adresse de retour des exceptions
R30	ba	Adresse de retour des points d’arrêt
R31	ra	Adresse de retour

d-Control registers



Register	Name	31…2	1	0
ctl0	status	Reserved	U	PIE
ctl1	estatus	Reserved	EU	EPIE
ctl2	bstatus	Reserved	BU	BPIE
ctl3	ienable	Interrupt-enable bits		
ctl4	ipending	Pending-interrupt bits		
ctl5	cpuid	Unique processor identifier		



Bit	Description
PIE bit	PIE is the processor interrupt-enable bit. When PIE is 0, external interrupts are ignored. When PIE is 1, external interrupts can be taken, depending on the value of the ienable register.
U bit	U is the user-mode bit. 1 indicates user mode; 0 indicates supervisor mode.

3-AES Peripheral

Our project consist on implementing an AES hardware peripheral for NIOS II processor using FPGA.

The features of our AES peripheral are:

Data size = 128 bits

Key size = 128 bits

Our NIOS II processor will communicate with the AES peripheral using a driver.

V-Achievement

For this part of the report, we are going to introduce all the tools that we have used to create our project.

1-ModelSIM

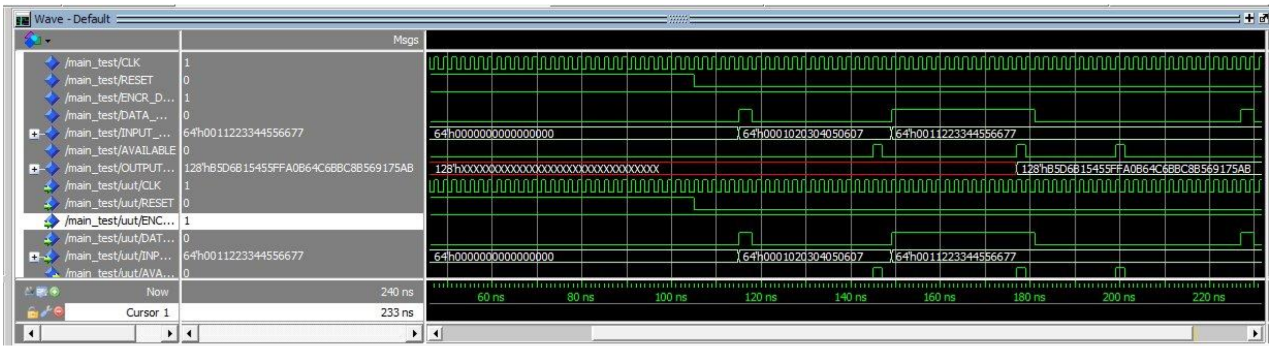
ModelSim is a multi-language HDL simulation environment by Mentor Graphics, For simulation of hardware description languages such as VHDL, Verilog, and System C .

ModelSim can be also used independently, or in conjunction with Altera Quartus or Xilinx ISE. Simulation is performed using the graphical user inetrface(GUI), or automatically using scripts.

Test Bench

Once we have finished writing code for our design using ModelSIM, we need to test whether it is working or not and to verify if the encryption and description methods work correctly. So One method of testing our design is by writing a testbench code which use one Encryption Key and one input.

Encryption Simulation



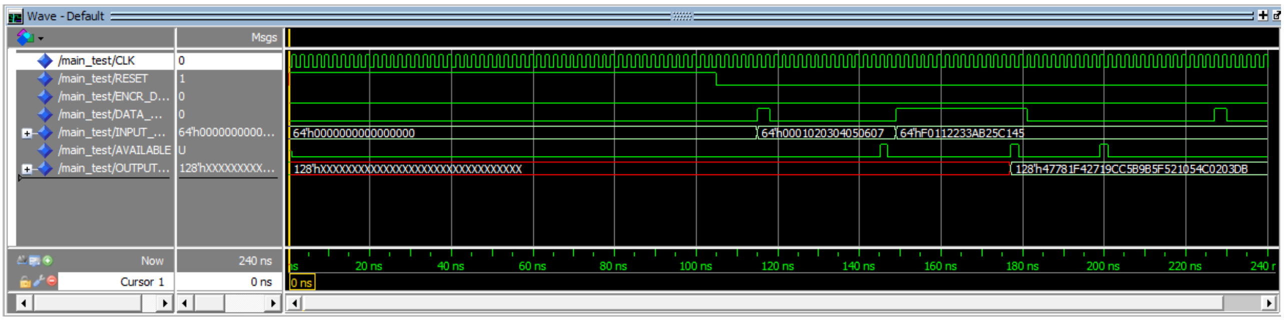
```
Wave - Default
                        Msgs
/main_test/CLK          1
/main_test/RESET        0
/main_test/ENCR_D...    1
/main_test/DATA_...     0
/main_test/INPUT_...    64'h0011223344556677
/main_test/AVAILABLE    0
/main_test/OUTPUT...    128'hB5D6B15455FFA0B64C6BBC8B569175AB
/main_test/uut/CLK      1
/main_test/uut/RESET    0
/main_test/uut/ENC...   1
/main_test/uut/DAT...   0
/main_test/uut/INP...   64'h0011223344556677
/main_test/uut/AVA...   0
Now                     240 ns
Cursor 1                233 ns
```

Key word: 128'h00010203040506070000000000000000

Input: 128'h00112233445566770000000000000000

Output: 128'hB5D6B15455FFA0B64C6BBC8B569175AB

Decryption Simulation



Key word: 128'h00010203040506070000000000000000

Input: 128'hf0112233ab25c1450000000000000000

Output: 128'h47781F42719CC5B9B5F521054C0203DB

2- IDE Quartus

With this IDE, we can do the design of electronic circuits and using as entry

- A standard netlist having EDIF (Electronic Design Interchange Format) as type.
- A text written with a hardware description language as VHDL, Verilog…

For our project, we have used a VHDL codes to make a design of our AES circuit using Quartus and this figure 5.1 shows the global block of AES circuit:

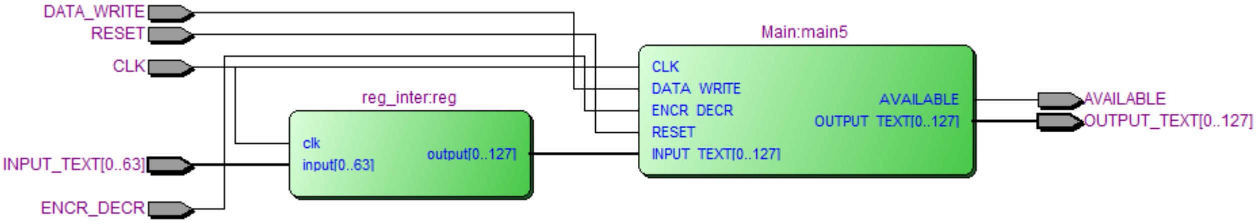


Figure 5.1: AES Total Block

3-Qsys

The Qsys system integration tool saves significant time and effort in the FPGA design process by automatically generating interconnect logic to connect intellectual property (IP) functions and subsystems. Qsys is the next-generation SOPC Builder tool powered by a new FPGA-optimized network-on-a-chip (NoC) technology delivering higher performance, improved design reuse, and faster verification compared to SOPC Builder.

4-Constraints and difficulties

- We wasted a lot of time on reading documentations
- The implementation of encrycption and decryptions methods was the hard part of this project because we were supposed to write an optimized and sythensizable codes so this part took a long time to be done correctly.
- One of the problem in this project was to reduce the number of pins by adding a new register to our first AES architecture so the code can be simulated on Altera having as characteristic Cyclone III EP3C25F324.

V-Conclusion

Optimized and Synthesizable VHDL code was developed for the implementation of both encryption and decryption process in this project. The whole program was tested using a test bench vhdl and also the code was compiled and implemented on an FPGA using altera card.

Perspective:

- This AES algorithm can be parameterized by selection of cipher key bits (128,192 or 256).
- For higher throughput, 16 S-Box can be used completing whole processes around 44-50 cycles(at the same time, compromising the silicon area)
- Graphical Use Interface (GUI) can also be made which may be interactive with the user.

Bibliography

1-https://en.wikipedia.org/wiki/ModelSim

2-https://en.wikipedia.org/wiki/Altera_Quartus

3-https://www.altera.com/products/design-software/fpga-design/quartus-prime/quartus-ii-subscription-edition/qts-qsys.html