

Third Prize

High Aberrance AES System Using a Reconstructable Function Core Generator

Institution: I-Shou University, Department of Computer Science and Information Engineering

Participants: Chen JianHong, Liu Yu, and Chia-Hau Shiu

Instructor: Ming-Haw Jing

Design Introduction

Cryptography is an essential part of communication or information security. The Advanced Encryption System (AES) was launched as a symmetrical cryptography standard algorithm by the National Institute of Standard and Technology (NIST) in October 2000. Rijndael provides the AES algorithm's architecture, and the algorithm's operation modules are based on finite field mathematics. Coding contains the SubBytes, ShiftRows, MixColumns, AddRoundKey, KeyExpansion modules, and the corresponding transcoding module. We use a look-up table (LUT) for the SubBytes and KeyExpansion modules. This LUT, which is referred to as an S-box, takes up 256[x]8-bit memory.

In addition to containing the original AES specification, a flexible architecture is needed to produce additional inputs that can change to irreducible polynomials, Affine transform matrix, and round number parameters. This algorithm design makes AES decryption impossible even with the golden key, and its variability can be expected to increase by more than 10 million times. This design needs the software and hardware to cooperate, and takes advantage of the FPGA architecture to realize a highly variable AES quickly.

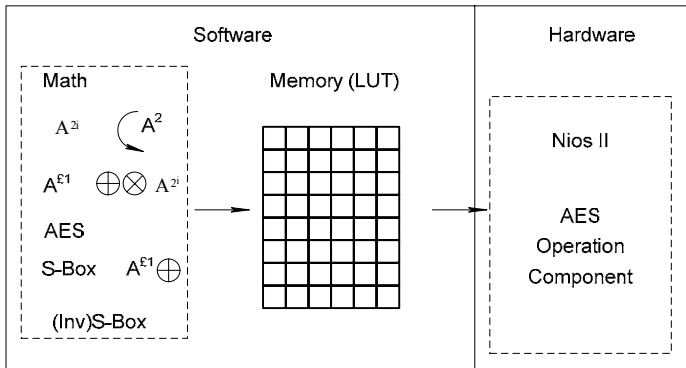
Using the SOPC Builder tool we were quickly able to set up parameters to generate the Nios® II control modules required for development. The Nios II microprocessor uses a RISC core, and can be combined with a variety of peripherals, custom instructions, and custom hardware accelerators, including algorithm logic operation, bit (group) operation, data transfer, flow control, condition instruction, and so on. You can program these hardware accelerators as function calls in the C or C++ languages. Our system adds fundamental components based on finite field mathematics and implements a high-speed

calculator and functional modules in software. We also performed special functions with custom instructions and used the GNU C/C++ compiler and Eclipse IDE.

Design Concept

We used hardware-software co-design to complete the test platform for the AES software and hardware data. Using the test platform, we were able to properly assess AES hardware and the control program module operation as shown in Figure 1.

Figure 1. Co-Design of Software & Hardware



Today, many designers use a fixed irreducible polynomial for higher efficiency and a smaller footprint of the AES intellectual property (IP). For long-term use, however, the fixed irreducible polynomial has been proven to make the system’s golden key obvious, thus increasing the decryption rate of confidential files. The decryption methods include side channel, time channel, and power side channel attacks. Some systems can even be decrypted by an inside job. To overcome these deficiencies, we designed an AES with high variability that can generate a LUT in real time through parameter input to provide a dynamic AES core. See Figure 2. The input variables of this system are different from that of the traditional AES, which cannot decrypt the encrypted document. See Figure 3.

Figure 2. AES with High Variability

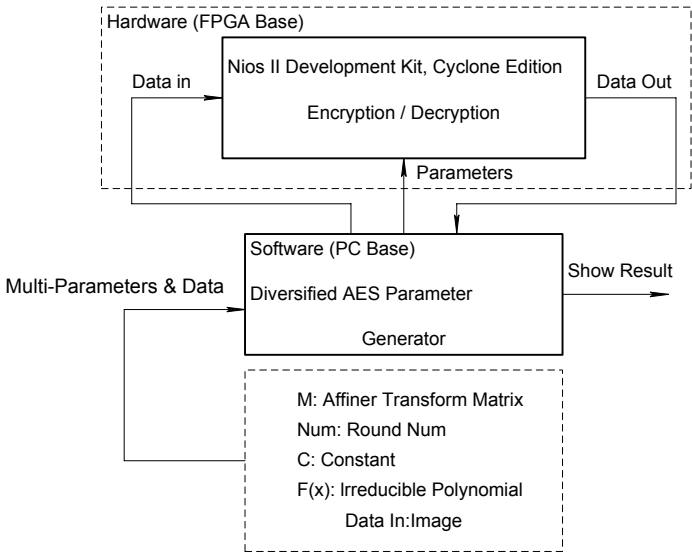
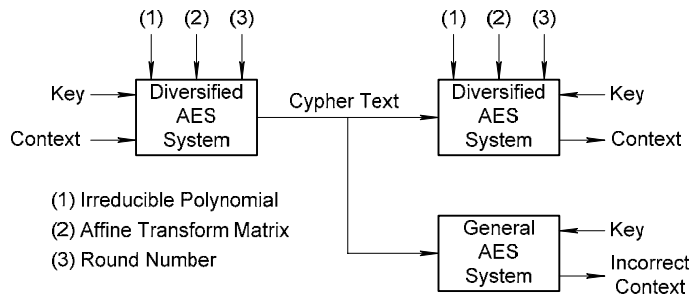


Figure 3. Traditional AES Cannot Decrypt the Encrypted Document



Based on Rijindael’s AES theory, we divided the functions into encoding and transcoding. The operation module for both parts is shown in Table 1 (each module is described in later sections). The generation of the S-Box form (see “Implementation Method” for more information) is the key to using AES theory. However, this generation must largely use finite field mathematical operations, such as multipliers and squarers. These operations can be realized in software, so we can generate the required S-Box and (Inv) S-Box. See Figure 4. This group integrates the operation modules of the AES encoding/transcoding functions and requires the inclusion of four main components: (Inv)SubBytes, (Inv)ShiftRows, (Inv)MixColumns and (Inv)AddRoundKey. You can implement the functions using the Nios II software or by using hardware to accelerate the complete flow of encoding/transcoding. For instance, the (Inv)ShiftRows and (Inv)MixColumns components are created in hardware.

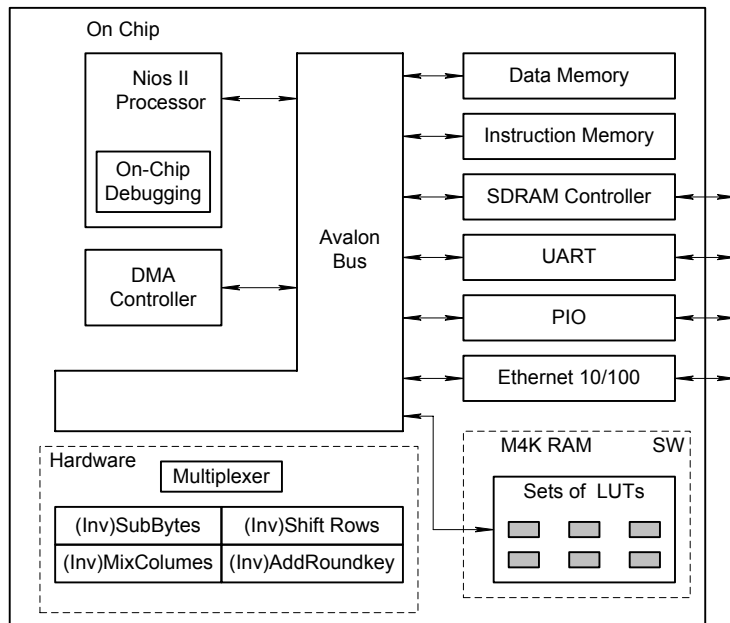
Table 1. Encoding/Transcoding Algorithm in Rijndael’s AES Theory

Encryption	Decryption	Our Implementation
<i>AddRoundKey</i> for Round=1 to N-1 <i>SubBytes</i> ShiftRows MixColumns <i>AddRoundKey</i> end for <i>SubBytes</i> ShiftRows <i>AddRoundKey</i>	<i>InvAddRoundKey</i> for Round=1 to N-1 InvShiftRows <i>InvSubBytes</i> <i>InvAddRoundKey</i> InvMixColumns end for InvShiftRows <i>InvShbBytes</i> <i>InvAddRoundKey</i>	Bold : Software or Hardware Italic : LUT

The architecture of the AES operation core can be divided into three types (see Figure 4):

- *Hardware component*—Operation efficiency for accelerating the AES.
 Operation component of the AES theory: (Inv)ShiftRows, (Inv)MixColumns.
 Selection of the demultiplexer for encryption/decryption.
- *Operation of the dynamic table*—Generation of (Inv)S-Box and (Inv)Key Expansion.
- *Software operation*—Establishment of the dynamic table, system combination, core component control and operation control, data flow control, and interface control.

Figure 4. Architecture of the AES Operation Core



Diversified AES Application Scope

The application scope includes:

- Secure wireless communications.
- Protect network routers.
- Secure electronic financial transactions.
- Secure video surveillance systems.
- Encrypted data storage.
- Secure network storage systems.

Target Users

The target users include:

- Manufacturers of wireless network bridges and wireless network adapters that support the AES security mechanism.
- Manufacturers of encrypted VPN products or firewalls.
- Manufacturers of encrypting chips for mobile phones.
- Manufacturers of private network hardware or high-capacity hardware array.

- Manufacturers of ATM secure-exchange devices.
- Manufacturers of portable communications or storage systems.
- Manufacturers of private sensor network devices.

Nios II Development Kit

We used the Nios II Development Kit, Cyclone™ Edition, which contains the Cyclone EP1C20FC400 FPGA, to implement our design. The board features 36-Kbyte RAM, 1-Mbyte SRAM, 16-Mbyte SDRAM, 8-Mbyte flash, 10/100 Ethernet PHY/MAC, two serial ports (RS-232 DB9 port), and so on. See Figure 5.

Figure 5. Nios II Development Kit, Cyclone Edition



Function Description

This section describes the functionality of the system.

Expected Functionality

To implement this design we:

1. Used the Quartus® II software version 5.0 to implement the various APUs in VHDL for a high-variability AES system.
2. Designed LUT generator and co-processors.
3. Built the entire AES system using Altera's system-on-a-programmable-chip (SOPC) design methodology.
4. Completed real-time transmission of plain text and cryptograph using a 115.2 Kbps UART interface.
5. Completed 128-bit AES encoding/trancoding with SOPC Builder's C++ compiler.

6. Supported a multi-variable input interface to generate different AES encoding/transcoding processes.

Implementation Method

We used the following implementation method:

1. Completed various APUs designed by VHDL for a high variability AES system.
 - a) According to the AES theory, three input methods can generate a high-variability AES system: the irreducible polynomial, the Affine transform matrix, and round numbers.
 - b) The APUs were coordinated according to the input requirements of the multiplier, squarer, S-Box, KeyExpansion, (Inv)SubBytes, (Inv)ShiftRows, (Inv)MixColumns, and (Inv)AddRoundKey.
 - c) Compiled VHDL code in the Quartus II software version 5.0, and completed functional validation.
 - d) According to the specification of Federal Information Processing Standard Publication 197, completed simulation of the software with BCB version 6.0, and validated it.
2. Designed the LUT generator and co-processor.
 - a) Analyzed the operation structure of SubBytes and InvSubBytes according to input parameters, and generated the key required by S-Box and (Inv)S-Box form in the software.
 - b) Downloaded and stored the generated S-Box, (Inv)S-Box, and Key to the development board.
3. Built the system using the Altera® SOPC Builder tool.
 - a) Initiated data sampling using the Cyclone FPGA standard functions.
 - b) Added to the user's customized PIO. See Figure 6. The setting of each PIO is shown in Table 2.

Figure 6. Add User's Customized PIO

Use	Module Name	Description	Clock
<input checked="" type="checkbox"/>	<input type="checkbox"/> seven_seg_pio	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> reconfig_request_pio	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> uart1	seven_seg_pio: 16-bit PIO using 32 serial port	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> sysid	output pins	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> sdram	(avalon) controller	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data0	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data1	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data2	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data3	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data4	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data5	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data6	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data7	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data8	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data9	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data10	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data11	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data12	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data13	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data14	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_data15	PIO (Parallel I/O)	clk
<input checked="" type="checkbox"/>	<input type="checkbox"/> aes_ctl_out	PIO (Parallel I/O)	clk

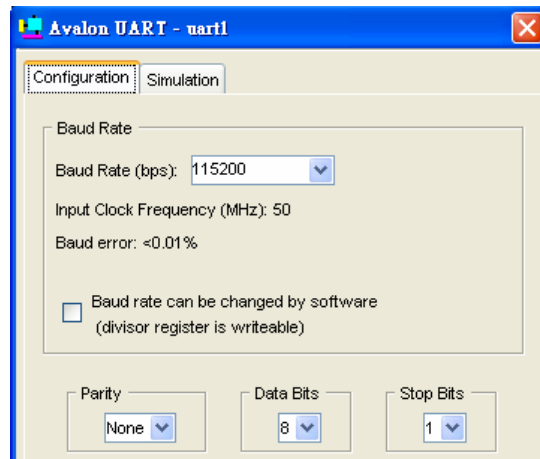
▲ Move Up ▼ Move Down

Table 2. Customized PIO Specification

Name	Size	Direction	Purpose
Aes_data0~15	8 bits	Bidirectional	Transmit encrypted data
aes_ctl_out	32 bits	Export	Control external AES components

- Defined UART baud rate: set as 115.2 Kbps, no parity, data bit=8, stop bit=1; as shown in Figure 7.

Figure 7. Communication Setting of UART Component



5. Completed 128-bit AES encoding/transcoding process with the SOPC Builder C++ compiler.
- a) Compiled GUI interface program with the SOPC Builder C++ compiler.

Figure 8. Provide User's Input Parameter Interface

Matrix(binary)

11110001
11100011
11000111
10001111
00011111
00111110
01111100
11111000

M:

Round(Int)

Num: 11

Constant(binary)

C: 01100011

F(x)(Hex)

F(x): 1B

- b) Based on the Federal Information Processing Standard Publication 197 specification, we completed the software test platform, which validated the whole system, as shown in Figure 9.

Figure 9. Comparison of Specification (Up) & Test Platform (Down)

Round Number	Start of Round	After SuBytes	After ShiftRows	After MixColumns	Round Key Value																																																																																
Input	<table><tr><td>32</td><td>88</td><td>31</td><td>e0</td></tr><tr><td>43</td><td>5a</td><td>31</td><td>37</td></tr><tr><td>f6</td><td>30</td><td>98</td><td>07</td></tr><tr><td>a8</td><td>8d</td><td>a2</td><td>34</td></tr></table>	32	88	31	e0	43	5a	31	37	f6	30	98	07	a8	8d	a2	34	<table><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>																	<table><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>																	<table><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>																	<table><tr><td>2b</td><td>28</td><td>ab</td><td>09</td></tr><tr><td>7e</td><td>ae</td><td>f7</td><td>cf</td></tr><tr><td>15</td><td>d2</td><td>15</td><td>4f</td></tr><tr><td>16</td><td>a6</td><td>88</td><td>3c</td></tr></table>	2b	28	ab	09	7e	ae	f7	cf	15	d2	15	4f	16	a6	88	3c
	32	88	31	e0																																																																																	
	43	5a	31	37																																																																																	
	f6	30	98	07																																																																																	
a8	8d	a2	34																																																																																		
2b	28	ab	09																																																																																		
7e	ae	f7	cf																																																																																		
15	d2	15	4f																																																																																		
16	a6	88	3c																																																																																		
1	<table><tr><td>19</td><td>a0</td><td>9a</td><td>e9</td></tr><tr><td>3d</td><td>f4</td><td>c6</td><td>f8</td></tr><tr><td>e3</td><td>e2</td><td>8d</td><td>48</td></tr><tr><td>be</td><td>2b</td><td>2a</td><td>08</td></tr></table>	19	a0	9a	e9	3d	f4	c6	f8	e3	e2	8d	48	be	2b	2a	08	<table><tr><td>d4</td><td>e0</td><td>b8</td><td>1e</td></tr><tr><td>27</td><td>bf</td><td>b4</td><td>41</td></tr><tr><td>11</td><td>98</td><td>5d</td><td>52</td></tr><tr><td>ae</td><td>f1</td><td>e5</td><td>30</td></tr></table>	d4	e0	b8	1e	27	bf	b4	41	11	98	5d	52	ae	f1	e5	30	<table><tr><td>d4</td><td>e0</td><td>b8</td><td>1e</td></tr><tr><td>bf</td><td>b4</td><td>41</td><td>27</td></tr><tr><td>5d</td><td>52</td><td>11</td><td>98</td></tr><tr><td>30</td><td>ae</td><td>f1</td><td>e5</td></tr></table>	d4	e0	b8	1e	bf	b4	41	27	5d	52	11	98	30	ae	f1	e5	<table><tr><td>04</td><td>e0</td><td>48</td><td>28</td></tr><tr><td>66</td><td>cb</td><td>f8</td><td>06</td></tr><tr><td>81</td><td>19</td><td>d3</td><td>26</td></tr><tr><td>e5</td><td>9a</td><td>7a</td><td>4c</td></tr></table>	04	e0	48	28	66	cb	f8	06	81	19	d3	26	e5	9a	7a	4c	<table><tr><td>a0</td><td>88</td><td>23</td><td>2a</td></tr><tr><td>fa</td><td>54</td><td>a3</td><td>6c</td></tr><tr><td>fe</td><td>2c</td><td>39</td><td>76</td></tr><tr><td>17</td><td>b1</td><td>39</td><td>05</td></tr></table>	a0	88	23	2a	fa	54	a3	6c	fe	2c	39	76	17	b1	39	05
	19	a0	9a	e9																																																																																	
	3d	f4	c6	f8																																																																																	
	e3	e2	8d	48																																																																																	
be	2b	2a	08																																																																																		
d4	e0	b8	1e																																																																																		
27	bf	b4	41																																																																																		
11	98	5d	52																																																																																		
ae	f1	e5	30																																																																																		
d4	e0	b8	1e																																																																																		
bf	b4	41	27																																																																																		
5d	52	11	98																																																																																		
30	ae	f1	e5																																																																																		
04	e0	48	28																																																																																		
66	cb	f8	06																																																																																		
81	19	d3	26																																																																																		
e5	9a	7a	4c																																																																																		
a0	88	23	2a																																																																																		
fa	54	a3	6c																																																																																		
fe	2c	39	76																																																																																		
17	b1	39	05																																																																																		

Round Number	Start of Round	After SubBytes	After ShiftRows	After MixColumns	Round Key Value
input	<div>3243f6a8</div> <div>885a308d</div> <div>313198a2</div> <div>e0370734</div> <div>193de3be</div> <div>a0f4e22b</div> <div>9ac68d2a</div> <div>e9f84808</div>				<div>2b7e1516</div> <div>28aed2a6</div> <div>abf71588</div> <div>09cf4f3c</div> <div>a0fafe17</div> <div>88542cb1</div> <div>23a33939</div> <div>2a6c7605</div>
1		<div>d42711ae</div> <div>e0bf98f1</div> <div>b8b45de5</div> <div>1e415230</div>	<div>d4bf5d30</div> <div>e0b452ae</div> <div>b84111f1</div> <div>1e2798e5</div>	<div>046681e5</div> <div>e0cb199a</div> <div>48f8d37a</div> <div>2806264c</div>	

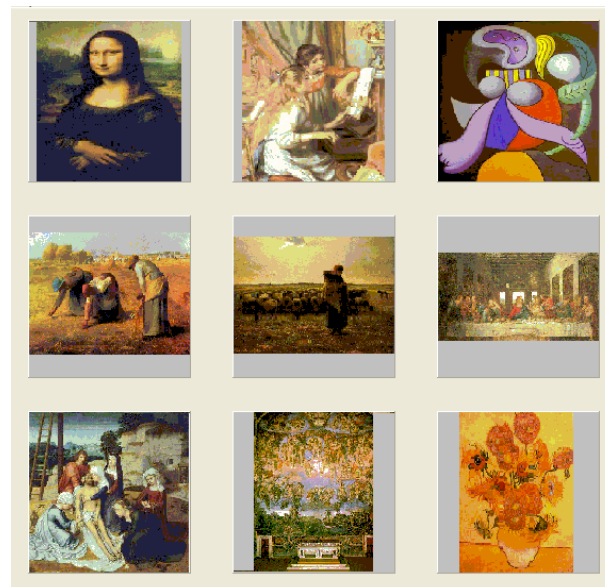
- c) According to the integration of the test data, testing system, and the test pattern provided by the specification of Federal Information Processing Standard Publication 197, in Figure 10, the numbers marked in red are the result of the encoding/transcoding process.

Figure 10. Encoding/Transcoding Results by Test Pattern Input

Input Data(Hex)		
Plain Text:	3243f6a8885a308d313198a2e0370734	
Cipher Key(Hex)		
Round Key:	2b7e151628aed2a6abf7158809cf4f3c	
Matrix(binary)	Round(int)	
M:	num: 11	
	Constant(binary)	
		C: 01100011
	F(x)(Hex)	
		F(x): 18
	Result(Hex)	
	Cipher Text:	3925841d02dc09fbdcl18597196a0b32
	Verify:	3243f6a8885a308d313198a2e0370734

- Supported a multi-variable input interface to generate different AES encoding/transcoding processes, as shown in Figure 11.

Figure 11. Multi-Variable Input Interface



Performance Parameters

The key function of the system while operating on this group is to perform the graphics for encoding/transcoding. Because AES is a symmetric-password system, the method for encoding and transcoding exception sequences is almost the same and the analysis of the following performance parameters lists only the encoding process. The graphics encoding in this group is a 256[×]256-pixel, 8-bit bitmap. Because the system can process 128-bit data each time, it needs $256[×]256[×]8 \div 128 = 4096$ times encoding in all.

Four encoding functions—including SubBytes, ShiftRows, MixColumns, and AddRoundKey—are used during every encoding process, with each function needing a great number of memory read/write actions (in this group, data memory is set as external SDRAM). The performance analysis is shown in Table 3.

Table 3. Performance Analysis During Encoding

Function Description	Memory		Expected Number of Cycle (Times)
	Read (Times)	Write (Times)	
Load this encoding data from memory (128 bit)	16	16	400
SubBytes	48	16	3200
ShiftRows	16	16	5600
MixColumns	16	16	5600
AddRoundKey	32	16	2400
Write Encoded Data to Memory (128 bit)	16	16	400
Flow and Peripheral Control	0	0	2000

Experimenting with minimum and maximum round, the minimum round is 3 and the maximum round is 11. The numbers shown in bold in Table 3 must be operated repeatedly in accordance with different rounds, and the time of the repeated operation is Round-1. We have arranged the expected time and the actual (experienced) time in Table 4, and added the completed PC software simulation time for comparison. It is obvious that the Nios II/s, the standard processor, outperforms the PC software port.

Table 4. Time Analysis

Round Number	Expected Time (s)	Use Time (s)	PC Software Simulation (s)
3	$[400+(3200+5600+5600+2400) \times 2 + 400 + 2000] \times 20\text{ns} \times 4096 \approx 2.98$	3	4
11	$[400+(3200+5600+5600+2400) \times 10 + 400 + 2000] \times 20\text{ns} \times 4096 \approx 13.99$	13	27

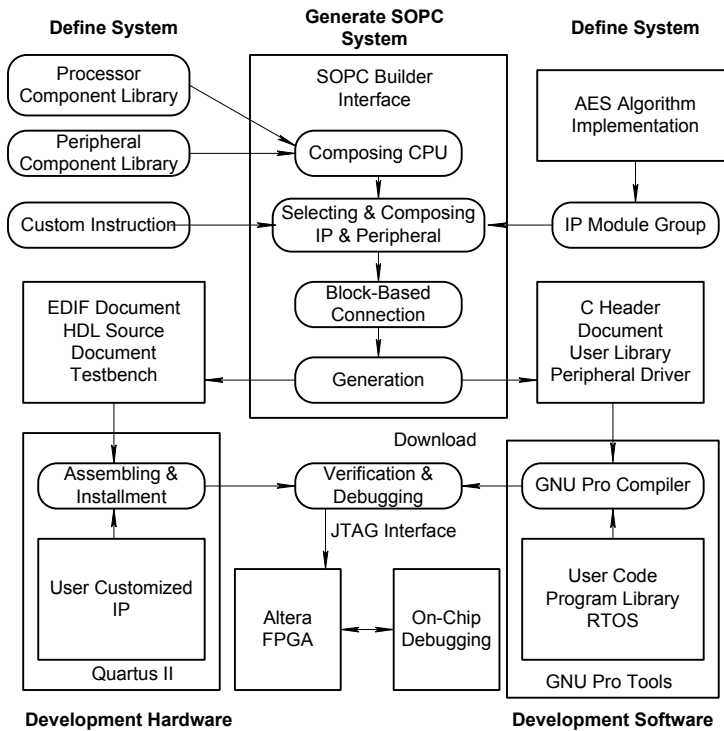
Design Architecture

This section describes the design architecture.

System Design

Figure 12 shows the system design diagram.

Figure 12. Diagram of System Design



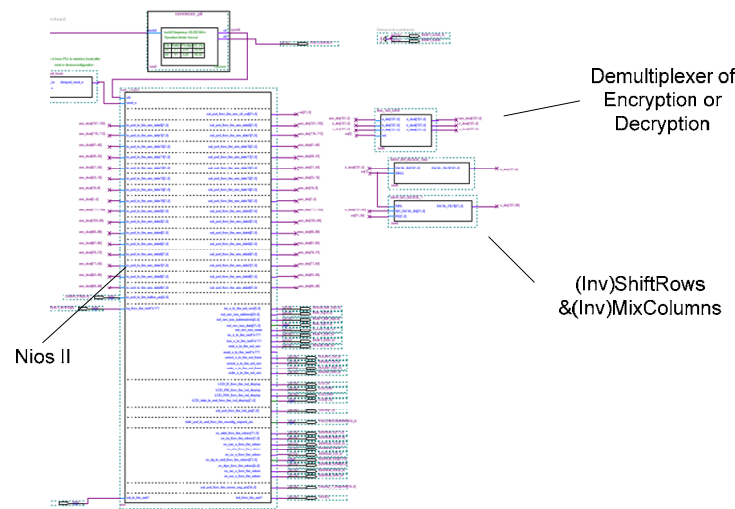
We created the user link library of the system. See Table 5 for each program.

Table 5. System Program Description

Function Name	Function Description
Load Parameters	Load Parameters from PC Port
Receive Image From PC	Load Pictures from PC Port
Encryption	Encrypting
Decryption	Decrypting
Send Cipher to PC	Return Completely Encoding Buffer to PC
Send PlainText to PC	Return Completely Decoding Buffer to PC
Text Device	Test Device
Change Mode	Switch Automatic Mode and Debug mode
Print Source Buffer	Return Gradually Source Buffer to PC
Print Cipher Buffer	Return Gradually Encoding Buffer to PC
Print Plaintext Buffer	Return Gradually Decoding Buffer to PC

Figure 13 shows the system diagram.

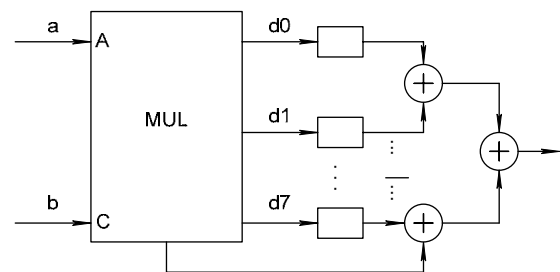
Figure 13. System Diagram



Hardware Design

This section describes the hardware design. We first created the multiplier, including the product and modulation. See Figure 14.

Figure 14. Multiplier Design Diagram



For the product: Computing the result (c with 15 bits) of two 8-bit operations that are multiplied by b. See Figure 15 for the theory, and Figure 16 for the waveform diagram.

Figure 15. Product Design Module

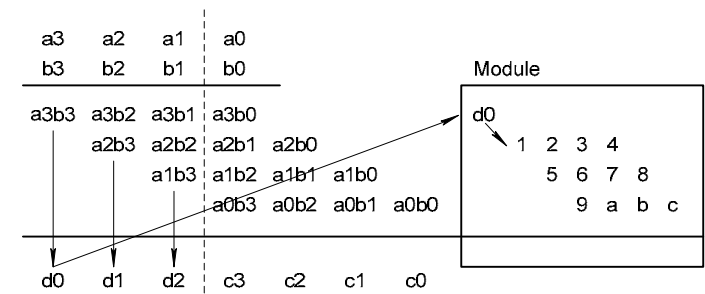
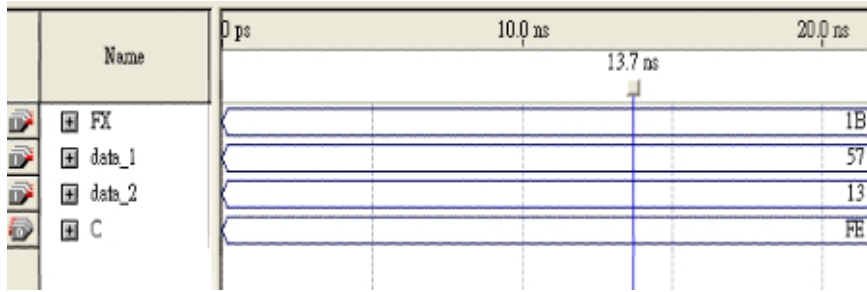
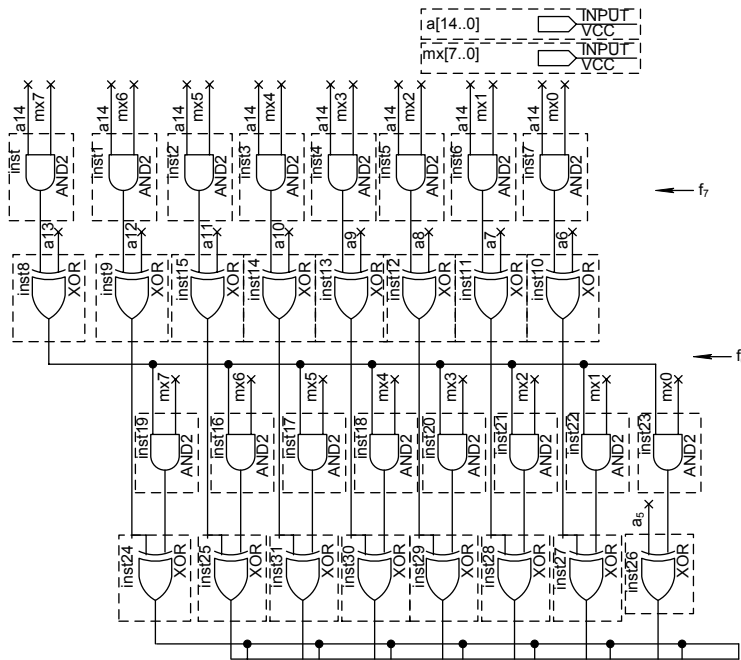


Figure 16. Product Simulation Waveform Diagram



For modulation: Input into C (15 bits), rank-reduced according to different $f(x)$ (where $f(x)$ must be an irreducible polynomial), the result is M (8 bits). The hardware circuit is shown in Figure 17, where f_7 and f_6 denote the top bit and secondary top bit of $f(x)$. These values are input at one port of this layer's AND gate.

Figure 17. Modulation Hardware Circuit Diagram



For (Inv)ShiftRow: Combine ShiftRow and (Inv)ShiftRow components and send required parts by multiplexer. Sel=0 is required one for encoding and sel=1 is required for decoding. See Figures 18 and 19.

Figure 18. (Inv)ShiftRow Design Diagram

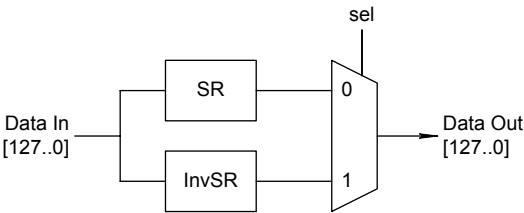
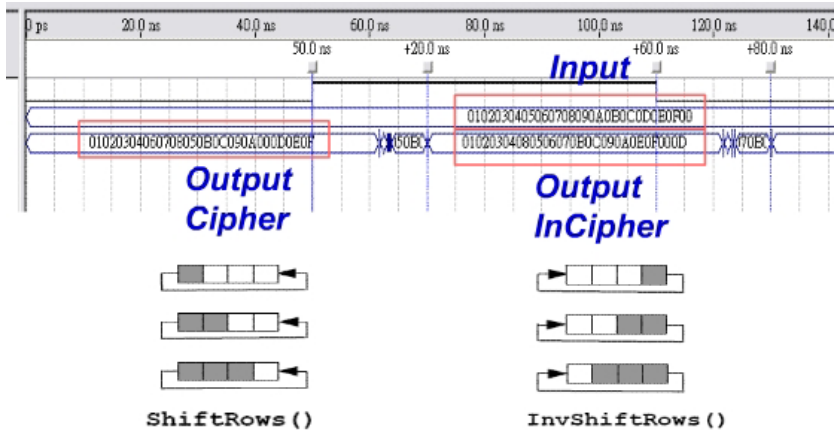


Figure 19. (Inv)ShiftRow Simulation



For (Inv)MixColumns: combine and design MixColumns and InvMixColumns components, generate different Word_MixCs in accordance with input different MixColumns Polynomials; the Word_MixC exports encoding/transcoding data at the same time it is encoding for sel = 0, transcoding for sel=1. See Figures 20 and 21.

Figure 20. (Inv)MixColumns Design Diagram

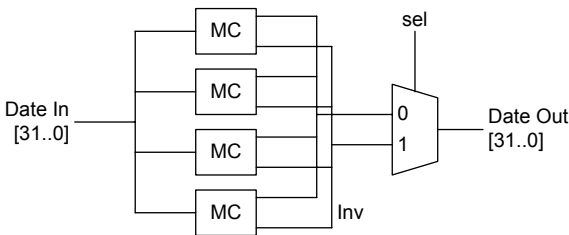
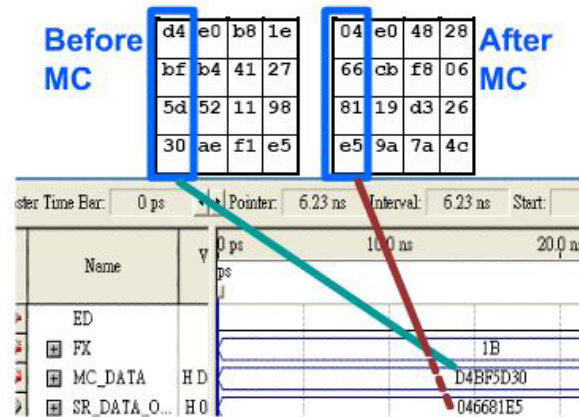


Figure 21. (Inv)MixColumn Simulation

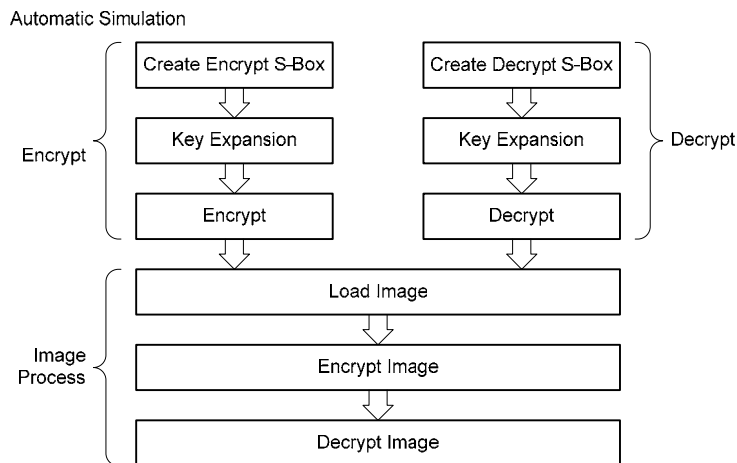


Software Design Flow

The function flow is as follows:

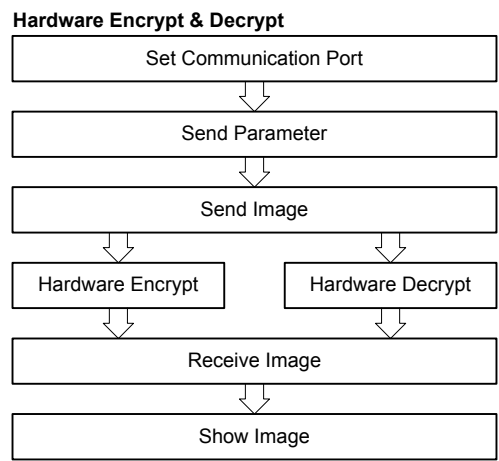
1. *Automatic software simulation*—Create an S-Box and complete encryption/decryption simulation and image encryption/decryption process. See Figure 22.

Figure 22. Automatic Software Simulation Flow



2. *Hardware encryption/decryption*—The RS-232 interface is used to pass parameters and data and to receive the data after verification. See Figure 23.

Figure 23. Hardware Encryption/Decryption Flow



See Table 6 for the description of the software pseudo-code and function.

Table 6. Software Pseudo-Code

DAES Encryption	DAES Decryption
<div>Load_Parameter Generate_Encrypt_SBox Key_Schedule for Image_rowcount=1 to row for Image_colcount=1 to column AddRoundKey for round=1 to N-1 SubBytes ShiftRows MixColumns AddRoundKey end for SubBytes ShiftRows AddRoundKey end for end for</div>	<div>Load_Parameter Generate_Decrypt_SBox Key_Schedule for Image_rowcount=1 to row for Image_colcount=1 to column InvAddRoundKey for round=1 to N-1 InvShiftRows InvSubBytes InvAddRoundKey InvMixColumns end for InvShiftRows InvShbBytes InvAddRoundKey end for end for</div>

See Table 7 for the software function description.

Table 7. Software Functions Description

Coordinate All Required Functions	
Basic function:	
Multiplication	Multiplication that operates on finite field
Inverse	Inverse element that operates on finite field
Matrix Inverse	Inverse Matrix that computes Affine Transform Matrix
Image Process	Sub-program of image process
Debug RS232 communication	Sub-program sent by UART interface
AES function:	
Load Parameter	Load parameter, user can change source parameter
Generate_Encrypt_SBox	Generate_Encrypt_SBox: Required S-Box for generation of encryption
Key Schedule	Key is expanded for the use of AddRoundkey
Create Encrypt S-Box	Table that creates S-Box
Create Decrypt S-Box	Table that creates (Inv)S-Box
Key Expansion	Key that creates each Round
Encrypt	Encrypted sub-program
Decrypt	Decrypted sub-program

The key functions are described as follows:

1. *Parameter input*—The user can input parameters and increase variability function. See Figure 24.

Figure 24. Parameter Input Interface

Altera 2005 NIOS Design Competition : 高變異性AES系統-使用可重組性功能核心產生器 - ISU CSIE 3116 CA Lab

Project Logo | Automatic Simulation | **AES Introduction** | SW / Algorithm simulation | HW / structure block implement | About

Input Information | Create S-Box | Key Expansion | Encrypt | Decrypt | Image | Pixels Value | RS-232

Input Data(Hex)

Plain Text: 3243f6a8885a308d313198a2e0370734

Cipher Key(Hex)

Round Key: 2b7e151628aed2a6abf7158809cf4f3c

Matrix(binary)

M:

11110001
11100011
11000111
10001111
00011111
00111110
01111100
11111000

Round(int)

num: 11

Constant(binary)

C: 01100011

F(x)(Hex)

F(x): 1B

Result(Hex)

Cipher Text: 3925841d02dc09fbdcc110597196a0b32

Verify: 3243f6a8885a308d313198a2e0370734

Encoder

Encrypt S-Box | Key Expansion | Encrypt

Decoder

Decrypt S-Box | Key Expansion | Decrypt

Parameter Input

AES 是對稱式密碼系統(Symmetric Key)

加密與解密都用同一把金匙

AES-128 需要 128bit 的金匙

Plaintext : 3243F6A8885A308D313198A2E0370734

Key : 2B7E151628AED2A6ABF7158809CF4F3C

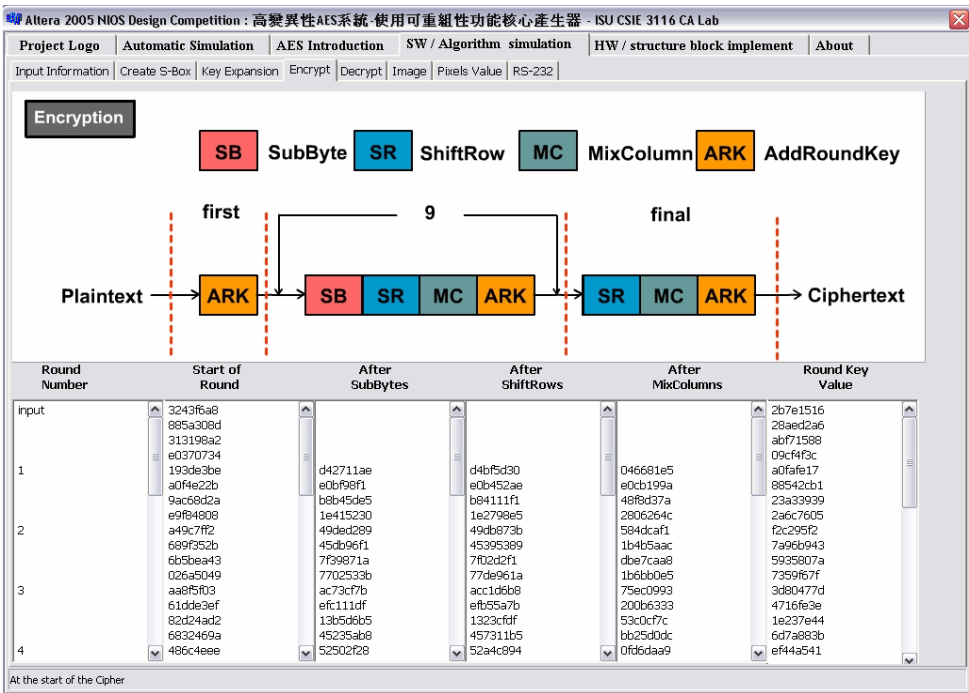
Plaintext (明文) ↔ Ciphertext (密文)

同一把金匙

Altera 2005 NIOS Core Competition

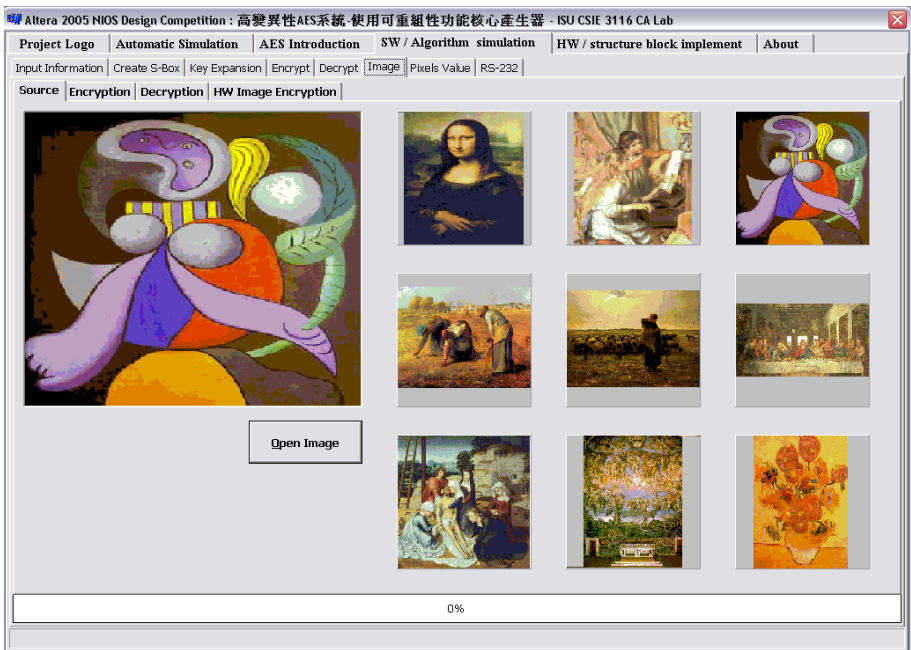
2. *Data verification*—Provide complete encryption/decryption process. See Figure 25.

Figure 25. Data Verification Interface



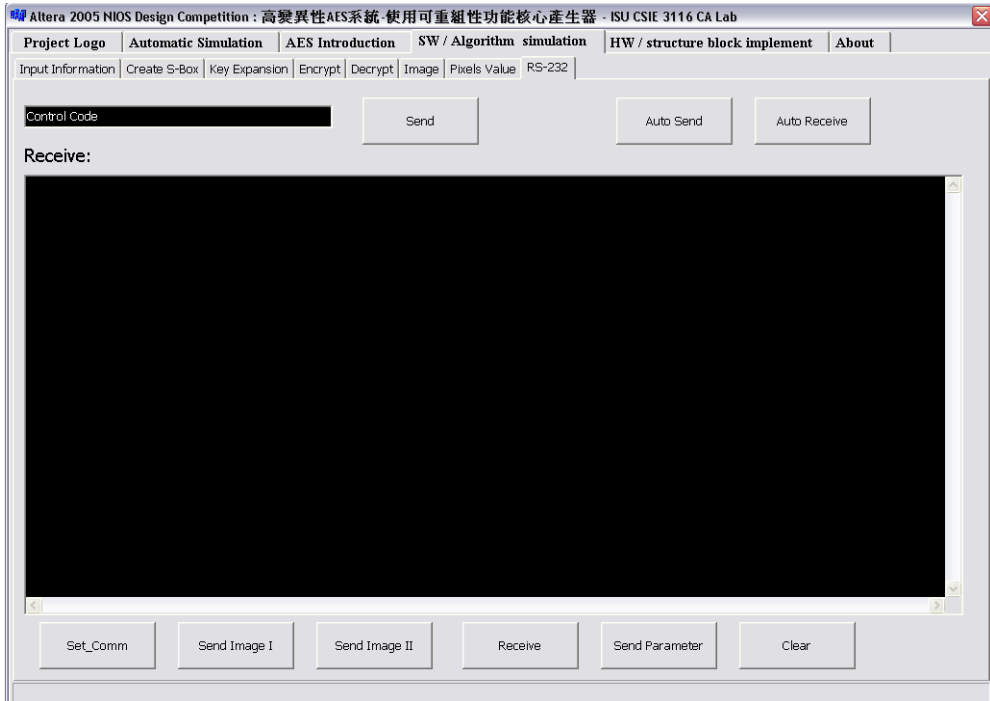
3. Image process—Load a 256[x]256, 8-bit image and perform encryption/decryption process. See Figure 26.

Figure 26. Image Process Interface



4. *Debug communication*—Based on RS-232 communication handle, the hardware returns the data to make debugging easy. See Figure 27.

Figure 27. Communication Debug Interface



Design Methodology

This section describes the design methodology.

Realization Method

The realization method includes the following steps:

1. *Definition of the AES system*—Including the processor, memory, and peripheral components.
2. *Generation of the system*—Generate the document using the SOPC Builder tool.
3. *Design the hardware*—Build the required components with VHDL code, and incorporate, compile, and simulate the circuits.
4. *Design the software*—Use the Nios II integrated development environment (IDE) to generate the related headers and drivers, write the application program, and compile it as an .elf executable file.
5. *Simulation*—Simulate with the ModelSim software tool. If there is a problem, return to step 2 to modify the system design software/hardware.
6. *Verification*—Perform verification by downloading the software/hardware with the JTAG port onto the RAM of the Cyclone development board.

7. *Test*—Produce test results by combining the user interface software of the PC port development board and by delivering huge volumes of data for measurement.

Design Process

The SOPC design approach provides an integrated software/hardware with an IDE including logic part (IP design), storage part (RAM), and computation core (CPU or DSP). The process of our system design is as follows:

- *Selection of algorithm and core*—We were able to handle the entire AES operation process with the Nios II processor by adopting Rijndael's AES algorithm. We added input variables to generate the dynamic form that is stored on the on-chip RAM.
- *Selection of the IP and design of custom IP components*—Our design uses general-purpose IP components, whose detailed explanation files can be downloaded from Altera's website. Additionally, we developed custom IP components according to the requirements of the system design and connected it onto Avalon® bus.
- *Design of the software/hardware system*—Software and hardware modules are used together in this design. This method creates challenges because the development of the software involves the plan and assignment of hardware resources, and is dependant upon system performance. However, the SOPC Builder and the Nios II IDE tools provide us with an integrated software/hardware design development system, making it easy to accelerate the design process.

Design Features

- *Dynamic Form Generation*—Based on the three input variables, the dynamic form of the S-Box and MixColumns Matriqs are generated and stored in RAM. Thanks to the FPGA architecture, we can use the Nios II processor to control the operation component of each AES, and perform data move, access, and operation with the Avalon bus. In this way, we have successfully implemented the high variability AES password system on the FPGA.
- *100% Realization of Software/Hardware*—The software/hardware platform of the Diversity AES project was successfully designed in this group, greatly improving the security of the AES.
- *Personalized Demo Program*—In this group, the whole Diversity AES process is shown at the software port by AutoRun, which enables the user to understand quickly the design operation.
- *Connecting Three Customized IP Functions to the Nios II Core*—Because the Nios II processor is flexible, we were able to design the PIO of external communication according to the design requirements. This means we were able to combine (Inv)ShiftRow, (Inv)MixColumn, and demultiplexer to accelerate the efficiency of the AES encoding/decoding operations in this group.
- *Solution to UART-Related Envelope-Packet Transmissions*—Because the UART IC on the development board connects with a 25-MHz quartz oscillator, frequency errors may occur. As a result, a few envelope packets may not be delivered during a large volume data transmission. Therefore, we reduced the envelope packets in this group, making them suitable for transmission and successfully solving the problem.

Conclusion

During Altera's 2005 Nios II processor competition, our design group divided the design tasks into system integration, hardware development, and software design as follows.

- *System*—The convenience of the Nios II IDE and the SOPC Builder tools gave us the flexibility to realize the design quickly on a prototype machine, which accelerated the development process. Through this competition, we have learned the process of consumer-electronics product development. The SOPC design approach reduces the cost of manpower and material resources during development. Because of this, we believe that the design approach will become popular in the future. While we did not add many components, this competition made us appreciate the potential of more system integration capabilities. Additionally, we hope that Altera can provide a variety of demo board demonstration programs that will enable interested students to quickly grasp the development process of FPGA designs.
- *Hardware*—In this competition, we used a top-down design approach and planned the complete design of the hardware in the beginning. This meant that a set of data stream rules needed to be established at the start of the planning stages. These rules eliminated problems during the design stage, allowing the project to be completed on time. Teamwork became an integral part of this contest. Although the Quartus II tool was easy and flexible to use, there were design issues that required experience; for example, using different frequencies while accessing the RAM. In conclusion, this competition provided us with an important opportunity to learn about teamwork and problem-solving, understand system development, and resolve challenging design questions.
- *Software*—We developed the necessary software interface, stressing communication and message exchange with the Nios II processor. We completed this task with the RS-232 interface, and learned a lot about message transmission. We adopted the SOPC Builder C++ tool to create the software design for the Window's interface. We used it as a verification tool and managed to perform Nios II communication debugging for the phase test. We hope to learn more about SOPC design in the future!

Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, mask work rights, and copyrights.