The University of Texas at Dallas
Design and Analysis of Reconfigurable Systems
Group Project (<u>Groups of 2</u>)

Objectives

Learn how to build complete HW/SW co-design systems on a configurable SoC FPGA.

Due Date: December 15 11:59pm

Project Overview

Field-Programmable Gate Arrays (FPGAs) are widely used to accelerate computationally intensive applications, especially those with a high degree of parallelism, such as image processing, digital signal processing, and cryptographic algorithms.

In this project, your team will design and implement a complete hardware/software (HW/SW) system that performs AES-128 encryption and decryption. The system will split responsibilities between the HPS (processor) and the FPGA fabric.

Project Objectives

Your HW/SW system must support full AES-128 encryption and decryption as follows:

HPS:

1. **Generate a 128-bit Encryption Key:** The HPS must internally generate a random 128-bit key.
2. **Read User Input (Plaintext):** Through the terminal (using `scanf`), read up to **16 ASCII characters** (16 × 8-bit). This serves as the *plaintext* that will be encrypted.
3. **Communicate with the FPGA AES Module:** Send both the encryption key and the plaintext to the AES hardware module implemented on the FPGA.
4. Receive the resulting encrypted data (*ciphertext*) back from the FPGA.
5. **Perform Decryption on the HPS and Validate:** Decrypt the received ciphertext in software on the HPS.
6. Compare the decrypted output with the original plaintext. Print the following on the terminal:
   Encryption Key: QQQQQQ
   Original plaintext: XXXX
   Ciphertext from FPGA: YYYY
   Plaintext from FPGA: XXXX
   The final plaintext should match the original user input exactly.

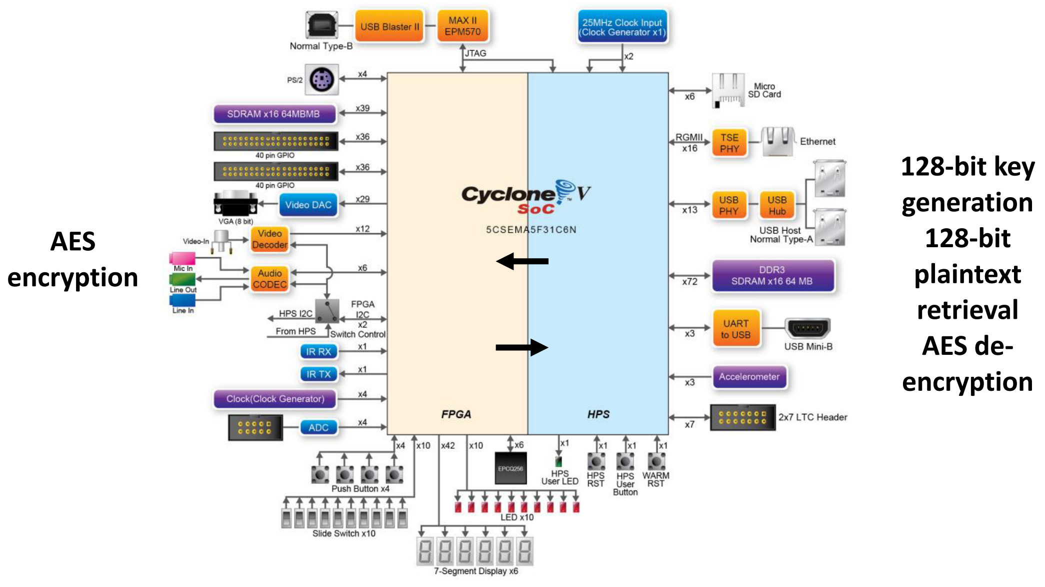

Figure 1. Overview of complete HW/SW system

Detailed Project Description and Deliverables

HPS Part

1. Encryption Key Generation: The HPS must generate a **random 128-bit AES key**.
In the provided AES reference code, the key is currently hardcoded:

```
unsigned char keys[] = {
    0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,
    0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f
};
```

Your task is to modify this code so that the 16 bytes of the key are randomly generated at runtime.

2. Plaintext: The modified `aes.c` program should prompt the user to enter the plaintext message. The HPS must read **up to 16 ASCII characters** (i.e., 16 × 8-bit), which form the 128-bit plaintext block.

The provided reference code uses a placeholder array such as:

```
unsigned char init[] = {
    0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,
    0x88,0x99,0xaa,0xbb,0xcc,0xdd,0xee,0xff
};
```

This must be replaced by the actual user-entered plaintext.

3.Software Encryption/Decryption (Golden Reference): The AES code running on the HPS should be capable of **both encrypting and decrypting** the plaintext. This software version serves as the **golden reference**:

- Use the same randomly generated key.
- Encrypt the plaintext via software.
- Decrypt the ciphertext via software.
- Use these results to verify the correctness of the FPGA-generated ciphertext.

FPGA Part

The FPGA implements the **AES-128 encryption hardware module**. It must:

1. **Receive the 128-bit encryption key** from the HPS.
2. **Receive the 128-bit plaintext** from the HPS.
3. **Encrypt the plaintext** using the provided key (AES-128 encryption only).
4. **Return the resulting 128-bit ciphertext** back to the HPS.

Important: The FPGA is only required to **perform encryption**. Decryption is handled entirely by the HPS software.

Deliverables:

1. Modiy the aes program running on the HPS following the description given above
2. Design an aes hardware accelerator mapped onto the reconfigurable fabric of the FPGA. You can either use RTL (Verilog or VHDL) or HLS.
3. Create a set of ppts slides summarizing the project and showing that it works (not more than 10 slides excluding cover and conclusion). **Create a YouTube video.**
4. **Submit all the project files in a zipped folder.**

Student 1 Name and Email:
Student 2 Name and Email:

Quality of Report Marking Scheme 20%

Feature	**%**		**Comments**
Power point report Clear explanation of system architecture and workflow **(4%)** Block diagrams, timing, memory mapping, and results shown clearly **(3%)** Professional formatting, clarity, and grammar **(3%)**	50		
YouTube video Clear explanation of design decisions, organization, and roles **(4%)** Demonstration of the system running end-to-end **(4%)** Presentation quality (speaking clarity, pacing, visuals) **(2%)**	50		
TOTAL	**100**		

Working Designs Marking Scheme 80%

Feature		**%**		**Comments**
HPS	SW aes generating random 128-bit key and accepting plaintext (show functionality by displaying this information on the terminal)	10		
	SW aes encrypts plaintext and is also able to decrypt it displaying the results on terminal	10		
FPGA	C/Verilog/VHDL aes design Implement AES on FPGA. Include synthesis reports and also structure of aes including AXI interface Correct AES-128 encryption implementation **(10%)** Proper memory-mapped integration with the HPS **(5%)** Correct handling of inputs/outputs (key, plaintext, ciphertext) **(5%)**	20		
SoC	Fully working HW/SW system	60		
	TOTAL	**100**		

Total Marks = Reportx0.2 + Designx0.8 =