Lab 12

Data Encryption Standard (DES)

MIPSfpga Lab 12: Data Encryption Standard (DES)

1. Introduction

In this lab you will modify your MIPSfpga system so that it can perform DES encryption/decryption upon a DMA transfer. You will design, build, and test an Encryption Engine and integrate it into your system. As always, first read the entire document, especially the **What to Turn In** section, before beginning your work.

2. DES Encryption

DES stands for Data Encryption Standard. It was developed in the 1970's and is used widely in both industry and government. Although it is being superseded by extensions of the base algorithm, the principles of these extended algorithms are the same. You will implement the basic DES algorithm in this lab. The DES Encryption Engine you build will have the interface shown in Table 1.

Table 1. DES Encryption Engine Interface

Name	Polarity	Bit width	Description
Key	input	64	Encryption/Decryption key
encryptDecrypt	input	2	encryptDecrypt = 00 when no encryption/decryption should occur encryptDecrypt = 10 when encryption should occur encryptDecrypt = 01 when decryption should occur
dataIn	input	64	Data to be encrypted/decrypted
dataOut	output	64	The encrypted/decrypted data

The key and encryptDecrypt should be memory-mapped to the addresses shown in Table 2. The low 32-bits of key is called keylo, and the high 32-bits keyhi.

Table 2. DES Encryption Engine Memory-Mapped Registers

Name	Memory-mapped address
keylo (low 32 bits of key)	0xbf300010
keyhi (high 32 bits of key)	0xbf300014
encryptDecrypt	0xbf300018

The Encryption Engine should work with your DMA Engine. That is, before a DMA transfer, all of the DMA registers should be set (see Lab 11) as well as the key and encryptDecrypt registers. Then the DMA transfer can be initiated (DMAstart =1). If encryptDecrypt is 00, a regular DMA transfer should occur (as in Lab 11) – i.e., no encryption or decryption of data should occur. However, if encryptDecrypt is 10, the DMA transfer should occur with encryption – i.e., the source data should be **encrypted** before writing it to the destination address. If encryptDecrypt is 01, the DMA transfer should occur with decryption – i.e., the source data should be **decrypted** before writing it to the destination address.

The width of the AHB-lite bus transfers is 32 bits, but the quantity that can be encrypted is 64-bits, so remember that you will have to deal with this discrepancy in your hardware. You need only deal with quantities of transfers that are multiples of 64 bits.

Here is a description of the DES encryption algorithm and its implementation:

http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm

You will be implementing the basic DES encryption where, given a known key, a 64-bit message block **M** is encrypted into a 64-bit cipher block **C**, or vice versa.

As part of the development of your Encryption (/Decryption) Engine, sketch a block diagram of your design. This sketch should be detailed enough that someone could design the system from your sketch.

3. Program

After designing, implementing, and testing your Encryption Engine, write a program to exercise your design. The program should:

1. **Initialize a 500-word source array** with the values 1-500.
2. **Display the values** of the array in hex using fdc_printf.
3. Write **unencrypted** values of the source array to a 500-element destination array.
4. Then write **encrypted** values of the source array to the destination array using DMA access with encryption and the key 0x12345678AABBCCDD. Display the encrypted destination array elements in hexadecimal using fdc_printf.
5. Then **decrypt** the destination array using DMA access with decryption and write the decrypted array values back to the destination array. Display the decrypted destination array elements in hexadecimal using fdc_printf.

   Pressing any of the pushbutton on the Nexys4 DDR board should **pause** the display of values.

4. What to Turn In

Please turn in each of the following items, clearly labeled and in the following order as a **single** PDF file. Also submit your **rtl_up** folder – that contains your modified MIPSfpga system **and your bitfile** – in a single zipped folder.

1. **Encryption Engine design documents:** Submit any schematics, diagrams, etc. to explain your design. These should clearly, completely, and concisely describe your design. Add short sentences or paragraphs as needed to aid in the explanation. You will be graded on correctness, organization, and effectiveness at portraying the design. You will also be graded on performance of your system (faster is better) and hardware usage (lower hardware usage is better). These documents should also include your modified DMA engine.
2. **Encryption Engine modules and modified DMA Engine:** Submit each of your SystemVerilog modules related to the Encryption Engine and DMA Engine. These should be well-organized and clearly commented.
3. **FPGA usage:** Report the FPGA usage statistics reported by Vivado.
4. **Clock speed:** What was the maximum achievable clock frequency of the FPGA when running your design?
5. **Program:** Your clearly-commented and well organized C program for testing your system.
6. **Image of system:** screenshots of the OpenOCD window showing the results of your Nexys4 DDR board running your program.