ComS / CprE 583 – Reconfigurable Computing

Homework #4

Assigned: October 19

Due: November 2 (12:00pm)

[Note from Joe: This is a group assignment, so please collaborate with your project group and submit a single write-up to WebCT. Those who are working on individual projects can share ideas with other groups but will be responsible for their own submission. Especially if you are not previously familiar with VHDL it would be a very good idea to start on problems 1, 2, and 3 relatively early, so that you can gain some level of comfort before starting problem 4.]

1) A "Caesar" cipher is an example of a simple cryptographic substitution operation in which a plaintext message p is converted to a ciphertext message c by applying the following equation to each of the characters in the message: $c = E(p) = p + k$. For simplicity sake we assume a restricted 32-entry alphabet that includes [A-Z], a blank space character, and five punctuation characters [.,?!:]. We can also assume that the '+' operation is modular and that $0 <= k <= 31$. *[For example, for k=7, plaintext message "HELLO" would be converted to ciphertext "OLSSV".]*

- **(a)** Provide a VHDL implementation of the Caesar cipher that operates on a single character at a time. In your writeup provide a description of the implementation. Which VHDL design style did you decide to use and why?
- **(b)** Use Modelsim to simulate the encryption of a message using your Caesar cipher implementation. Provide a screenshot and a description detailing the proper execution of the cipher.

2) In a more general permutation cipher, each entry in the plaintext alphabet is mapped to a unique letter in the ciphertext alphabet, according to a permutation function π. For example, for π = "WHATSUP…", input "ABADCADFAD" would be encrypted as "WHWTAWTUWT". Design, describe, and simulate a VHDL implementation of the general permutation cipher, in the same manner as for Problem 1. *[There are numerous ways to implement the general permutation, but encoding the permutation function as a constant array of characters is a good way to start. Use the same 32-character alphabet]*

3) A one-time pad is another type of substitution cipher, in which the plaintext is encrypted using a large non-repeating key. If the input key is as long as the entire plaintext string, then the cipher can be as simple as a single XOR operation. Design a one-time pad in VHDL that can encrypt 16 bits of plaintext every cycle. For this architecture, have four different input keys, where for cycle t key $k0$ is used, for cycle t+1 key $k1$ is used, for cycle t+2 key $k2$ is used, and for cycle t+3 key $k3$ is used. Describe and simulate this implementation of the custom one-time pad in the same manner as for Problems 1 and 2. *[The actual encryption here is just a simple XOR operation. For* *example, if at some time range* t *to* t*+4 plaintext* p *was {0xaaaa, 0xbbbb, 0xcccc, 0xdddd, 0xeeee, etc.} and first key* k0 *was {0x0123, 0x1234, 0x2345, 0x3456, 0x4567, etc.}, then ciphertext* c *would be equal to {0xaaaa XOR 0x0123, 0xbbbb XOR* k1*(*t*+1), 0xcccc XOR* k2*(*t*+2), 0xdddd XOR* k3*(*t*+3), 0xeeee XOR 0x4567}.]*

4) Take a look at the Advanced Encryption Standard (AES) as specified in FIPS publication number 197 that can be found at the following web address: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. The goal of this problem is to design and analyze a VHDL implementation of a single round of AES encryption with a 128-bit key (AES-128E). Sections 3 and 5 of FIPS pub 197 are the most relevant for this purpose. Also see attached file *hw4.zip* which contains some VHDL code that you may find useful for this problem. Specifically, you should find the following files:

- *AES_config.vhd* – a VHDL package containing useful datatypes and constants
- *AESround.vhd* – a partial VHDL implementation of a single AES-128E round

(a) Design and test modules for the SubBytes, MixColumns, ShiftRows, and AddRoundKey transformations found in AES-128E. Provide your VHDL code, a description, and sample Modelsim traces.

(b) Integrate these modules into the top-level AES-128E VHDL component (AESround). The output partial ciphertext should be connected to a synchronous register. Provide your VHDL code, and show a sample Modelsim trace on one of the testvector inputs provided in Appendix C of the FIPS pub 197 document. *[The KeyExpansion part of AES-128E can be tricky to implement, but since you only have to implement a single round you can safely ignore it.]*

(c) Synthesize and implement the completed AESround design onto a Xilinx XC2V250 FPGA using ISE with default settings. Provide a chart detailing the area and performance characteristics. What is the critical path of the design? *[The longest delay path can be found in the synthesis report, and can then be viewed in the RTL/Technology schematic viewers.]*

(d) Modify the design in a systolic fashion, i.e. pipeline the individual transformations. How do the area and performance characteristics change? Based on the new critical path in the design, how would you next optimize one of the transformations in order to increase the maximum clock rate?