ECE 4170 **Spring 2007**

Assignment # 1

January 24, 2007

Due: February 7th, 2007

The purpose of this assignment is to exercise concurrent signal assignment statements, processes, and introduce you to testbenches. Additional material that you will find useful to understand an that you will need for this assignment will be covered in class on January 29th, 2007.

1 Getting Started

You can obtain/install ModelSim as follows

1) Go to "http://www.model.com/downloads/default.asp" and select Modelsim SE

2) Provide the requested information and follow the directions to download and install ModelSim.

3) Once ModelSim is installed. Go to the license wizard (will start automatically when you run ModelSim for the 1st time), and in the license file location, enter: 1717@mouse.ece.gatech.edu

From within ModelSim, go through the basic tutorial that can be found under the Help→PDF Documentation. Alternatively, you can go through the abbreviated tutorial found on the class website Resources page.

2 Warm-Up

This part of the assignment is intended to help you prepare for the main part of the assignment as well as create a few useful code pieces that you can use.

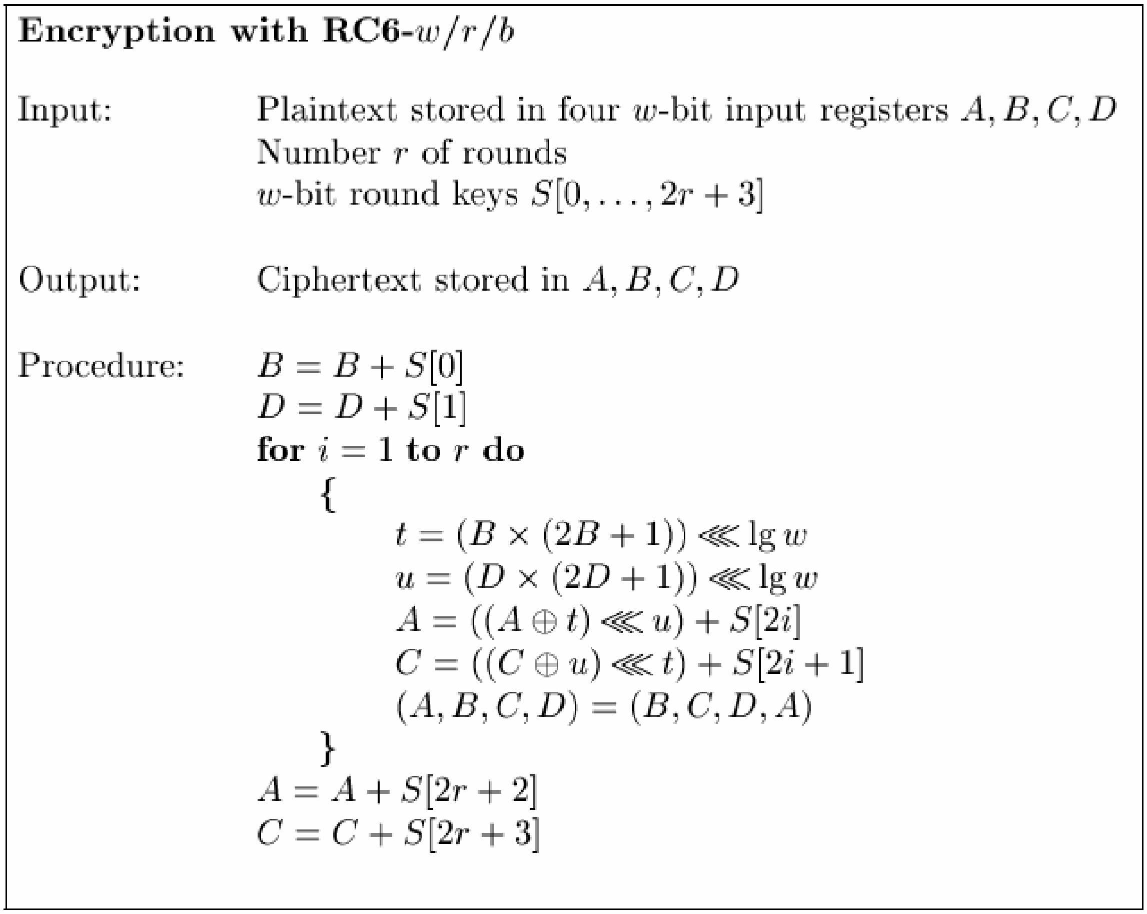
Create a VHDL process that reads 4-bit std_logic data from a file, computes the exclusive-OR of the data with the value x"b" and writes this value to a local (to the architecture) signal. Outside of the process, a concurrent signal assignment statement (CSA) will drive this value of the local signal onto an entity port. Thus the process and the CSA execute concurrently. Use a delta delay model. The entity should have one clock input, and a 4-bit output for the result. The data should appear in the text file as 0-9 and a-f characters, strung together. On the rising edge of each clock, you will read in one character, convert it to a 4-bit value, and xor that value with x"b". Use the entity definition:

```
entity hex_reader is
Port(
        clk : in std_logic;
        rslt : out std_logic_vector(3 downto 0)
    );
end entity hex_reader;
```

The clock stimulus should be provided within ModelSim, i.e, use the stimulus generators in ModelSim.

3 Main Assignment

Your task is to design and implement an RC6 encryption encoder in VHDL, using processes and file I/O. The general algorithm for this encryption process is given as follows:



Encryption with RC6-$w/r/b$

```
Input:       Plaintext stored in four $w$-bit input registers $A, B, C, D$
             Number $r$ of rounds
             $w$-bit round keys $S[0, \ldots, 2r+3]$

Output:      Ciphertext stored in $A, B, C, D$

Procedure:   $B = B + S[0]$
             $D = D + S[1]$
             for $i = 1$ to $r$ do
                 {
                     $t = (B \times (2B + 1)) \lll \lg w$
                     $u = (D \times (2D + 1)) \lll \lg w$
                     $A = ((A \oplus t) \lll u) + S[2i]$
                     $C = ((C \oplus u) \lll t) + S[2i + 1]$
                     $(A, B, C, D) = (B, C, D, A)$
                 }
             $A = A + S[2r + 2]$
             $C = C + S[2r + 3]$
```

3.1 Description of operators

a + b	integer addition modulo 2^w
a × b	integer multiplication modulo 2^w
$a \oplus b$	bitwise exclusive-or of w-bit words
a<<<b	rotate the w-bit word a to the left by the amount given by the least significant lg(w) bits of b
a>>>b	rotate the w-bit word a to the right by the amount given by the least significant lg(w) bits of b
lg(w)	base-2 log of w

Your implementation will perform two rounds of encoding for each input (i.e., r = 2), so you will need to use a loop. We will set w = 4, so the inputs to the encoder will be A, B, C, and D (4 bits each), as well as eight 4-bit keys (s0, s1, s2, s3, s4, s5, s6, s7). The outputs will be the resulting A_out, B_out, C_out, and D_out.

Define your entity as:

```
entity rc6 is
        port(
        clk : in std_logic;
        A_out : out std_logic_vector(3 downto 0);
        B_out : out std_logic_vector(3 downto 0);
        C_out : out std_logic_vector(3 downto 0);
        D_out : out std_logic_vector(3 downto 0)
    );
end entity rc6;
```

The general design of your solution will contain three processes: one to read 4-bit values from a text file (basically the warm up exercise), one to do the actual RC6 encoding, and a third process to write the encoded output to a text file (the output values should be directly readable as text). On the each rising edge of the clock, you will read in four 4-bit numbers at a time and perform the encoding. Continue until you reach the end of the file. The first eight 4-bit values in the text file will be the key values S(0) to S(7), so fetch those first, then start the encoding process. The input text file might look something like:

84ba724f2b0752b47d5f83956392058536d3c854bc930fa9

where the underlined numbers are the key values, and all of the values will be converted to 4-bit equivalents.

4 Submission Instructions:

Your submission should include four files: your warm-up VHDL file, the text file you used to test it with, the VHDL file for your RC6 solution, the text file you used to test it with, and its testbench. Name your files as follows:

lastname_firstinitial_sol0.vhd
lastname_firstinitial_sol0.txt
lastname_firstinitial_sol1.vhd
lastname_firstinitial_sol1.txt.

Zip all files into a compressed folder, and email to **mbales3@gatech.edu** by 9 pm on the due date. Place 4170 Assignment 1 in the subject line.