Project 3 Complex Sequential Circuit = Data path + control unit

CSc 137, Fall 2015, Faroughi

Due in Class Dec. 7

Consider a multi-cycle data path similar to the one discussed in class. The data path can perform RTNs R ← A + B + C ^ D or R ← A + B - C ^ D. Assume the control unit for the data path inputs signals *start*, *mode*, and outputs the data path control signals, a *done* signal, and an *error* signal (described below) when it is done computing the RTN. Do the following:

1) Using the ALU you designed in Project 2, create a data path to compute R ← A + B ± C ^ D depending on the *mode* value. When *mode* = 0 the data performs R ← A + B + C ^ D or R ← A + B - C ^ D when *mode* = 1.
2) Use "if-else" statements to design MUXs. Note, "if-else" statements must be within always blocks.
3) Create a behavioral model for an 8-bit parallel-load register (see Chapter 4 for an example of 1-bit FF); an 8-bit register has eight FFs (use vector, not individual FF d and q bits).
4) Construct a behavioral model of a FSD for the control unit (see Chapter 5 for examples of Moore and Mealy Verilog models). The complex sequential circuit shall operate as follows:

   Upon reset the data path and control unit will initialize. Upon *start** = 1, the control unit shall compute the requested RTN according to the similar steps outlined in class in several cycles according to the value of *mode*. After each step, the controller must check for the *ovf*. If the *ovf* is asserted, the controller shall transition to an "error" state and also assert *done* = 1 and *error* = 1 and will remain there until it is reset. It is assumed that the *start* signal will remain 1 for only two clock cycles. In the absence of any arithmetic overflow, the controller will perform the RTN, set *done* = 1 with error = 0, and then return to the initial state. Another computation can start without the need to reset the circuit when no overflow was encountered.

   * The *start* signal will be activated (set to 1) in the tester file and then set to 0 after two clock cycles. For example, if the clock period is 10ns, the following two Verilog statements will set and reset the *start* signal:

   ```
   start = 1;
   #20 start = 0;
   ```

   See tester models in Example 5.9 (Section 5.7) and Example 6.5 for additional information.

   a. For the control unit use a "case" statement and "if-else" statements to model the NSG module and a "case" statement and "if-else" statements to model the OG.

The OG module shall output the exact control signals to control the data path (also see step 7).

5) Combine the data path and control unit modules to create the complex sequential circuit.
6) Create a test-bench to validate your design using the following test cases where the mode signal is set to 0 and again to 1.
    a. A = 1; B = 2; C = 3; D = 8'hF0; mode = 1'b0;
    b. A = 1; B = 2; C = 3; D = 8'hF0; mode = 1'b1;
    c. A = 8'h7f; B = 8'h01; C = 8'h01; D = 8'h00; mode = 1'b0;
    d. A = 8'h7e; B = 8'h01; C = 8'h01; D = 8'h00; mode = 1'b0;
    e. A = 8'h7e; B = 8'h01; C = 8'h01; D = 8'h00; mode = 1'b1;
    f. A = 8'hff; B = 8'h01; C = 8'h7F; D = 8'h01; mode = 1'b0;
    g. A = 8'hff; B = 8'h01; C = 8'h55; D = 8'hFF; mode = 1'b1;

Note, the circuit should be reset only if both *done* and *error* were asserted. If done was asserted but error signal was not (i.e., no overflow detected), the circuit should not be reset to apply the next test vector.

7) Turn in your project following the format outlined in Project 1 and also include:
    a. The block diagram of your data path with all modules and signals labeled.
    b. The FSD with RTNs (e.g., like Fig. 6.6), the FSD with control signals (e.g. Fig. 6.7), and the FSD detailed block diagram showing OG, NSG, and FFs with signal labels (e.g., Fig. 6.8).
    c. Print out of the Verilog models (in few pages).
    d. The print out of the output with the analyses that illustrates the correct operation of the circuit (i.e., shows the outputs are correct when compared to hand calculations).