ECE/Comp Sci 352 Digital System Fundamentals

Project Part 2 (Summer 2000) DUE 8/1/00

The project, Part 2 has been designed to advance your familiarity with modern digital design tools and concepts and to help your understanding of material covered in class. It will require the use of Mentor Graphics and skills learned in class. You MAY use a partner for Part 2. The work should start as soon as possible. It will enhance your understanding of the material covered in Chapters 5, 6, 7, and 8.

The first problem of Part 2 implements a 4-bit universal shift register (similar to that one covered in Homework 3, Problem 12 except that it will "clear" instead of shift left) that will be used for other parts of the project. You will use Mentor Graphics tools to enter and simulate the device. Of particular importance is the ability to parallel load the and clear the register.

The second problem of Part 2 uses a quad register and a full carry look-ahead adder to implement an up/down counter with hold and clear. You will again use Mentor Graphics tools to enter the circuits and simulate their behavior. This portion of the project will help your understanding of the operation of the carry look-ahead adder and the sequential control sequences to stimulate the device. It will also be used in problem 4.

The third problem of Part 2 develops a device consisting of several universal shift registers, and a control memory, implementing a very general micro-programmed control sequencer. You will develop and test a micro-program to implement a sequence recognizer for the sequence "1101".

The fourth problem of Part 2 closely follows the simple register/adder/shifter structure of Figure 8-6, and the state sequence control of Table 8-2 to implement an 8x8 multiplier with an 8 bit parallel add/sequential multiplier. It will integrate components developed in problems one and two of the project Part 2.

AN OPTIONAL fifth problem of Part 2 will replace the hardware sequencer portion of the fourth problem with a microprogrammed sequencer, similar to the one developed in the third problem, that will be loaded with a microprogram and tested.

The project Part 2 is due Tuesday, August 1, 1:10 PM in class.
1. Universal Shift Resister (40 Points) [Reference: Problem 12 from HW#3] Use a 1-bit, D-type, positive edge triggered Flip-Flop and 4-to-1 multiplexer to build a single cell of a universal shift register. Design the circuit that will implement the following shift register commands:

<table>
<thead>
<tr>
<th>Control Input</th>
<th>Meaning</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1 S0</td>
<td>0 0 Hold</td>
<td>recirculate Q to D</td>
</tr>
<tr>
<td></td>
<td>0 1 Shift Right</td>
<td>shift data right (MSB toward LSB)</td>
</tr>
<tr>
<td></td>
<td>1 0 Clear</td>
<td>load &quot;0&quot; into the bit</td>
</tr>
<tr>
<td></td>
<td>1 1 Load</td>
<td>pre-load the data from four separate input lines</td>
</tr>
</tbody>
</table>

1a. Design and implement a 1-bit cell from a 4-to-1 MUX and a D-type Flip-Flop. Make a symbol body of the cell with the following pins:
- S1, S0: Control input pins.
- RSI: "Right Shift In" input.
- I: "Data Input" for Load operation.
- D: "Data Output" from the flip-flop.
- C: Clock Input

1b. Connect 4 above cells together into a four-bit universal shift register "USR" block. Make a new symbol body for the device with the following input pins:
- C1, C0: Control inputs (same meaning as S1, S0).
- RI: "Register Right Shift Input" -- data in for the most significant bit.
- D(3:0): Four data outputs. (D(3) is MSB)
- I(3:0): Four data inputs. (I(3) is MSB).
- CLK: Clock Input

1c. Simulate the circuit using QUICKSIM and test the device by providing a force file that assures that each bit can be loaded with a "1", a "0", and that "1"s and "0"s can be shifted right, shifted left, and can be cleared.

For problem 1 you need to hand in:

1a, 1b: The symbol drawing and schematic sheets for both single and quad devices.
1c: A Copy of the forcefile and QUICKsim output showing proper circuit behavior.
1d: A description of your test sequence and test strategy.
2. 4-bit UP/DOWN Counter with HOLD, CLEAR and LOAD (40 points)

2a. Implement a single bit full adder with Propagate and Generate (see book figure 3-29 "PFA" block) and make a symbol body for it. It should have the following signal pins:
   A -- Augend Input "A"
   B -- Addend Input "B"
   C -- Carry In Input
   G -- Generate Output
   P -- Propagate Output.

2b. Implement a Carry Look-Ahead Generator (CLAG) as shown in Figure 3-29(b). Add an extra Output for the $C_4$ Carry out ($C_4 = G_0 + P_0 \cdot C_0$). Make a symbol body for it. It should have the following signal pins:

   $C_0$ -- Carry Input to the least significant bit.
   $P_3, P_2, P_1, P_0$ -- Propagate Inputs for bits 3, 2, 1, and 0 respectively.
   $G_3, G_2, G_1, G_0$ -- Generate Inputs for bits 3, 2, 1, and 0 respectively.
   $C_3, C_2, C_1$ -- Carry Outputs to adder stage 3, 2, and 1 respectively
   $C_4$ -- Block Carry Output to the next stage.

2c. Use four "PFA" blocks and the "CLAG" block to implement a four-bit Carry Look-Ahead Adder (CLA) as shown in figure 3-29. Make a symbol body for it. It should have the following signal pins:

   $C_0$ -- Carry Input to the least significant bit.
   $A_3, A_2, A_1, A_0$ -- Augend Input for bits 3, 2, 1, and 0 respectively.
   $B_3, B_2, B_1, B_0$ -- Addend Inputs for bits 3, 2, 1, and 0 respectively.
   $S_3, S_2, S_1, S_0$ -- Sum Outputs for bits 3, 2, 1, and 0 respectively.
   $C_4$ -- Block Carry Output to the next stage.

2d. Use one "CLA" block four "D" Flip-flops and four 2-1 MUXes to implement a four-bit UP/DOWN/LOAD/HOLD/CLEAR counter as shown in the diagram below. Design the "Logic" network to transform the $C1, C0$, and U/D input controls to the appropriate $M1, M0, LSB$, and MSB to "LOAD" ($C1C0 = 00$), "CLEAR" ($C1C0 = 01$), "HOLD" ($C1C0 = 10$), and "COUNT" ($C1C0 = 11$). For "COUNT", U/D = 0 implies count up and U/D=1 implies count down. Make a body for it called "UDCNT". It should have the following pins:

   $C1, C0$ -- Input Mode Control pins
   U/D -- Input UP/DOWN Count Control pin
   $D_3, D_2, D_1, D_0$ -- Inputs for Load Data bits 3 to 0 respectively
   $Q_3, Q_2, Q_1, Q_0$ -- Outputs for Counter Data bits 3 to 0 respectively
   CLK -- Clock Input.

2f. Simulate the UP/DOWN/HOLD/LOAD behavior of the counter showing that the circuit is correct.
For problem 2 you need to hand in:

2a, 2b, 2c, 2d The symbol drawing for the full-adder and schematic sheets for each.

2e: A Copy of the force file and QUICKSIM output showing proper circuit behavior. ANNOTATE the Results showing that LOAD, Count-up, Count-down, and HOLD behavior. Make sure that you have simulated at least sixteen count steps in each direction.
3. **Universal Programmable Sequential Machine** (40 Points) Here you will use the memory primitive loaded with a text file, and a universal shift register from problem 1 to implement a universal programmable sequential machine. You will test it by defining a control which recognizes the sequence "1101". The form of the circuit is shown below (Note: C1, C0 and clock are not shown connected):

3a. Implement the circuit detailed above where there is a "Reset" signal (hint: use the Clear signal for the Universal Shift Register) forcing the device into state "0". When reset is de-asserted, the register will load a value equal to the variable "X" and the output of the memory bits as shown.

3b. Program the contents of the memory to implement a sequence recognizer for the sequence "1101" where "Out" is a 1 when the sequence has been recognized (Moore Model only).

3c. Simulate the circuit using QUICKSIM and test the device by providing a force file for that demonstrates the input sequence is properly recognized.

For problem 3 you need to hand in:

3a: The schematic drawing for the device.
3b: A Text file of the contents of the memory, a state transition table (Note: Memory Address are related to the State Attained), and a mapping of states and Inputs to addresses.
3c: Input Forcefile and QUICKSIM output showing proper circuit behavior. ANNOTATE the Results showing that states attained with your simulated input.
4. 8-bit by 8-bit Micro-programmed Multiplier (80 Points)
[Reference: Figure 8-6, Table 8-1, Figure 8-9]

Here you will start with existing component schematics and symbol bodies from Problems 1, and 2 and additional circuits to construct the data path circuit shown in Figure 8-6. You will need two CLA blocks for the parallel adder (with ripple carry between the blocks). You will need 6 USR blocks -- Two for each of Register B, Register Q, and Register A. Counter P will use one of the UDCNT blocks from Problem 2. The control signals required will be as shown in Table 8-1. The control sequence will be as shown in Table 8-2. A hardwired control is shown in Figure 8-9.

NOTE: You will have to convert the control signals shown in Table 8-1 to the appropriate signals for the USR and UDCNT blocks. You will also have to add logic for the "zero-detect" function (a 4-input OR gate). You will have to add the flip-flop "C" and control circuitry to load and clear it. Control signals to load, and data for, the initial Multiplier and Multiplicand values will be generated by a forcefile sequence.

You will:

4a. Implement the data path circuit of figure 8-6, converting the control signals from Table 8-1 to the USR and UDCNT signals needed. Test the result to verify proper operation using forcefiles to generate control signals.

4b. Add the sequential control block of Figure 8-9 and test the resulting operation.

4c. Generate several test cases (loading the Multiplier and Multiplicand via force files) that test the device completely. Document you test results.

For problem 4 you need to hand in:

4a, 4b: All schematic and symbol drawings for the device.
4c: A description of your test strategy.
4a, 4b, 4c Input Forcefile and QUICKSIM output showing proper circuit behavior. ANNOTATE the Results showing the states and data values attained and final results.
5. (EXTRA CREDIT) Microprogrammed Sequencer for the 8X8 Bit Multiplier
(10 Points -- to be used to boost quiz scores)

This optional Problem 5 will replace the hardware control sequencer from Problem 4 with a microprogrammed sequencer similar to that developed in Problem 3. You must increase the control width by several bits in order to provide the needed control signals to the multiplier datapath. You must also modify it to sample the G, Z, and Q0 bits. There are several ways to do this. See Chapter 8 for one way. Alternatively, increase the address width to take into account the additional signals. You must also provide a way to start the circuit (eg -- clear the control input address to zero.)

The same test file needed to test the first multiplier circuit should be able to be used here as well. Only a few modifications (to initialize the microprogrammed sequencer and load the microinstructions) of the force files are needed.

You will:

5a: Develop and test a microprogrammed sequencer to replace the fixed, hardware sequencer of problem 4, while utilizing the rest of the circuit without modification.

5b: Develop an annotated memory image file for loading into the microprogram memory.

For problem 5 you need to hand in:

<table>
<thead>
<tr>
<th></th>
<th>For problem 5 you need to hand in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>5a</td>
<td>All schematic and symbol drawings for the microprogrammed controller.</td>
</tr>
<tr>
<td>5b</td>
<td>An annotated listing of (with description) the memory microinstruction contents for the microcontroller.</td>
</tr>
<tr>
<td>5a, 5b</td>
<td>Input Forcefile and QUICKSIM output showing proper circuit behavior. ANNOTATE the Results showing the states and data values attained and final results.</td>
</tr>
</tbody>
</table>

**Note:** the force files should be the same as for Problem 4, except for any additional microcontroller initialization.