CS152A / EE116L

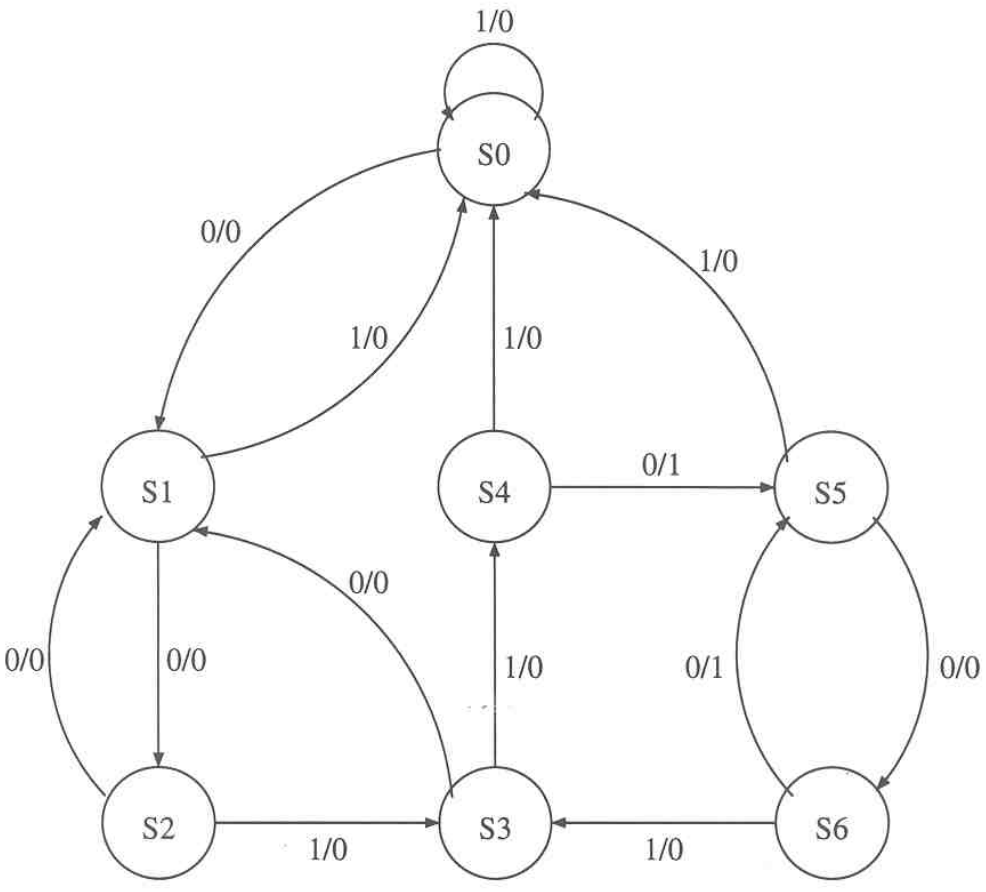
LAB 2 Implementation of a State Machine

In this experiment, you are going to show your ability of synthesizing sequential circuits and implement, test, and debug them using the MAX+PLUSII software.

You are required to implement a pattern recognizer that recognize an even number of 0s, followed by two 1s, and then an odd number of 0s, i.e. the pattern **$(00)^{+}110(00)^{*}$** Pattern recognition is done by a state machine that takes an single-bit input *xbit*; when that pattern is detected, the machine produces a positive signal on output *zbit*. Pattern overlap is allowed. As an example:

```
xbit: 00100110000000011000000...
zbit: 00000001010101000101010...
```

A possible state diagram of the state machine:



Pre-lab

Prior to the first meeting of this lab, your should do the following:

1. Verify that the state diagram that is given is in fact correct, or modify it so it is correct.
2. Does the state diagram show a Mealy machine or a Moore machine? What kind of change shall you make to design an equivalent Mealy (or Moore) machine if the given is a Moore (or Mealy) machine? What will be the difference on their behaviors?
3. Assign state codes to the states. You may use binary number codes to number the states (i.e. State 6 becomes state code "110"), or you may use Gray codes for next states where appropriate, or you may use other numbering schemes.
4. Derive the next-state excitation functions and output function with the knowledge you have learnt from the logic design class.
    a. In order to minimize the logic complexity, use K-maps. Each K-map will minimize the logic circuit gate complexity.
    b. Consider "gate sharing" (i.e. using same product term in more than one function) to reduce the set of gates needed to implement all functions.
    c. Derive your NAND-NAND excitation functions (i.e. using NAND gates only. NOT gates can be viewed as a special case of NAND gates).
5. Draw the circuit by hand (you may use the attached page as a prototype) and write down the excitation functions.
6. Review the "clean clock" part in Lab1.You will need the circuit to generate clean clock signal to demonstrate your design.
7. Read thoroughly the notes and tutorial on Altera MAX+PLUSII and AHDL, and learn to use them.

Bring a preliminary copy of your pre-lab work to the class.

Lab Work

Implement and demonstrate the state machine using the Altera MAX EPM7128 chip on the UP-1 board.

1. Use the graphic design tool to draw the circuit;
2. Use AHDL to describe the circuit

You need to do both.

Lab Report

Include the following in your lab report:

- copy of your pre-lab work
- printout of your gdf graphic design, and tdf file.
- printout of the waveform of your simulation results.

Notes

- For verifying your circuit, you need to show that the circuit traverses each arc of the state transition diagram as designed, and no other transitions can occur. One way to do this is to make a copy of the state transition diagram, and check off each arc as it is transited. Alternately you may use a state transition table, and check off each transition. You need to verify that all the transitions for which you designed are in fact done, and no other transitions can occur.

- Use input switches and indicator lights (LEDs) on the UP-1 board. You will need a clean clock signal.

- Make the current state (Q of all the D flip-flops) a group of outputs and connect them to LEDs, so you can verify the internal action of the circuit. It will be very helpful when you debug your design.

- You will work your design manually, thus speed of the circuit is of little importance in your design parameters. However, consider the following.

  In almost all digital designs it is essential to run the circuit at (or at least know) the maximum clock frequency that may be used. Read Sec. 8.4 of Ercegovac et al: "Timing Characteristics of Sequential Networks." Note that you have to consider circuit delays, switching propagation times (t_{PLH} and t_{PHL}), set-up and hold times. What is the maximum clock frequency that you can use? Consider using the MAX2PLUSII "Timing Analyzer" to find out. Read the appropriate HELP files.

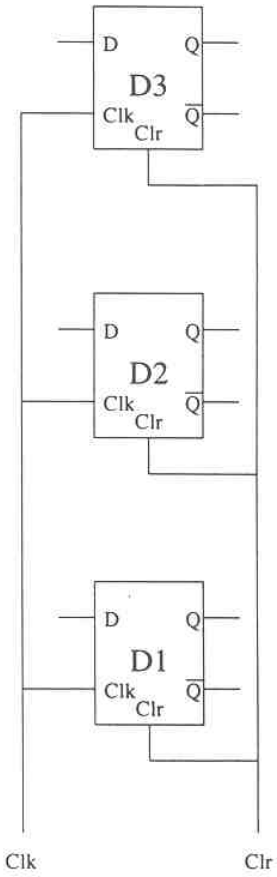
  When the circuit is driven too fast, there may not be enough time to set/reset all of the state variables. Therefore one may wish to use state assignments for successor states that differ only in ONE variable. That assignment is easiest to make with a K-map for the state variables.

- If your have both a Mealy machine and a Moore machine designs, you can verify their actions using Waveform files and implement them on the UP-1 board. It is highly instructive to compare the two waveforms. You can get further insight by implementing both designs on the UP-1 board and carefully compare their input-output behavior.

  Note the differences. Essentially the Moore machine looks at the *x* input (and any other input you may have in the general case) after the clock captures it. The Mealy machine does that as soon as the input appears, regardless of whether that input will remain valid for the rest of the clock cycle. In other words, the output may be only a very short spike, a "glitch" (look up this term in the MAX+PLUSII Help file.)

  If your state machine clocking and the input are operating at vastly different speeds, this may cause unreliable operations of your machine. This is especially true if input is from a producer which can be much faster than the machine which accepts it as an input. The Mealy machine will respond immediately (well, almost immediately - there is a circuit delay after all) and the Moore machine will wait to see if the inputs are valid to the end of current clocking cycle (well, not quite to the end, there is a DFF set-up time whatever circuit delays there may be in getting the action of the inputs to the internal state variables.)

  For this reason we will stress Moore machine designs in the rest of this laboratory course.



D1 =

D2 =

D3 =

Z =