Assignment # 8

Due Friday March 15

Read Chapter 8 Section 4 (again) and Section 5.
Read Chapter 9 Sections 1, 2, 3 (not 3.4).
Read Chapter 10 Section 1 (not 1.4)
Read Chapter 4 Section 4.2.4 (Tri-state and Open-collector gates).
Read Chapter 11.

1. Chapter 8 Exercise 8.17
2. Chapter 8 Exercise 8.23
3. Chapter 9 Exercise 9.18
4. A tennis-scoring device TS is to be implemented. The game is played by 2 players. In order to win a game, a player must win at least 4 points and must be at least 2 points ahead of the other player. TS has two inputs x_1, x_2 and two outputs z_1, z_2. Input x_i is set to 1 when player i wins a point and is set to 0 otherwise. Output z_i is set to 1 whenever player i wins a game and it is 0 otherwise.
   - Construct a symbolic state table that defines the behavior of TS.
   - Minimize the number of states (it might be that your first attempt would yield a minimal number of states, so justify it in that case).
   - What is the minimal number of flip-flops required to implement TS?
   - How many flip-flops would you use in one-hot encoding?
5. We want to implement a multiplier using only adders, comparators etc. The multiplier is A, the multiplicand is B and the product is P. Assume A and B are positive. One basic algorithm could be:

```
Load:    AR <- A; BR <- B;  PR <- 0;
Forever: While AR > 0
           begin
               PR <- PR + BR; AR <- AR -1
           end {while}
Out:     P <- PR;
```

- If A and B are 32 bits wide, how wide should PR and P be?
- Show a data path, control unit (state diagram), control and feedback lines for such a multiplier at a level of detail similar to that of the GCD processor in the lecture slides.

In some sense this algorithm is asynchronous, i.e., we don't know how many iterations it will take before it terminates. A "synchronous" algorithm would be:

```
Load:    AR <- A; BR <- B;  PR <- 0;
Loop:    for (i=1; i<= 32; i++)
           begin
             if (AR is odd) then PRhigh <- PRhigh + BR;
             Logicalrightshift(AR);  Logicalrightshift(PR)
           end{for}
Out:     P <- PR;
```

Now PRhigh represents the leftmost 32 bits of PR.

- Show a data path, control unit (state diagram), control and feedback lines for such a multiplier at a level of detail similar to that of the GCD processor in the lecture slides (Hint: You certainly don't need to implement an "Is AR odd" function; you could use the rightmost bit of AR as a control and/or feedback line).
- Indicate how one can optimize the design so that we need only registers PR and BR (don't redesign the whole multiplier; explain how how it could be done in a sentence or two).