ECE 370: Digital Systems-Logic Design

Sample Test: Chapter 7-Part 4: Registers
Spring 2005

Understanding Registers

1. Describe the general purpose of a register.
2. What are some uses of registers?
3. What types of Flip-Flops allow for the design of registers simple?
3. What is the difference between parallel in/serial out and serial in/parallel out data registers?
4. What is a shift register? What is one application where shift registers are used?
5. Describe a fast way to multiply and divide a binary number by a power of 2.
6. In general, what kind of "control" inputs can a register have? In other words, what is a universal data register?
7. What is the difference between a ring counter and a Johnson counter?

Registers

8. Provide a schematic of a 4-bit serial in/serial out, right-shift data register. Fill in the following table based on the following input/output values.

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
Serial In	*1*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*1*	*1*	*0*	*0*
Q_0	*0*												
Q_1	*0*												
Q_2	*0*												
Q_4	*0*												
Serial Out													

9. A way to perform serial-to-parallel data conversion is to use a shift register. Add to the schematic of Problem 8 the necessary wires, inputs, outputs, and/or labels to convert the 4-bit serial-in/serial-out data register into a serial-to-parallel data converter.

10. Re-design the register in Problem 8 to be a left shift serial-in/serial-out data register.

11. Compare the designs of Problem 8 and Problem 10. Design a 4-bit serial-in/serial-out data register with an additional input, RIGHT/LEFT, that controls whether or not the data register shifts left or right. *Hint: Think about the two different designs of the 2's complement adder/subtractor unit.*

12. Design a 1-bit data register using a D-Flip Flop. Show the schematic of your design. The inputs are as follows with an initial truth table to get you started. *Hint: you need to understand the D Flip-Flop characteristic equation*..

- P: Data being loaded into the 1-bit data register.
- Load: When asserted (logic HIGH), "loads" data *P* into the 1-bit register.
- Clear: When unasserted (logic LOW), synchronously clears the 1-bit Data register *Note:* Not an asynchronous CLEAR.

P	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
Load	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*	*1*	*1*	*1*	*1*
Clear	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*	*0*	*0*	*1*	*1*
Q(t)	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*0*	*1*
Q(t+1)																

13. Use your design in Problem 12 to design a *3-bit* parallel-load data register.

Shift Register Counters

14. Consider *n*-bit ring counter. Re-design an *n-bit* ring counter using a *modulo-n* UP counter and an *n-to-2*^{n} binary decoder

15. Explain the how the following Johnson counter differs from a "plain" Johnson counter. Analyze the operation for each switch { 1, 2, 3, 4, 5, 6 }