SD 192 Digital Systems August 3, 2001

Final Examination

Professor Paul Fieguth

Aids Permitted: Your brain and your pen. Nothing else.

Advice: Read problems carefully before jumping in.

The grade value for each question is indicated in brackets [ ] next to the question number; in most questions I also show the grade breakdown. The exam is graded out of 100. I *will* give part marks for relevant statements or insights. Tell me what you know!

Always apply "sanity-checks" to your answers - do they make sense?

Don't spend too much time one any particular question! Make sure that you leave enough time to answer all problems.

Bonus question – a theory of money.

This is a confusing question – don't spend time here until you're satisfied with the rest of your answers.

How do you explain why people buy lottery tickets? On average you lose money, so it can't be that people are trying to maximize the amount of money they have.

One theory is that people maximize their *satisfaction*. Let $s(m)$ represent the amount of satisfaction a person has with amount of money m. For most people, the relationship between s and m is probably reasonably close to a straight line. However depending on the shape of $s(m)$, it is possible that although on average you lose money playing the lottery, on average you actually increase your satisfaction.

(a) Sketch an example of $s(m)$ corresponding to a person who plays the lottery.

(b) Sketch an example of $s(m)$ corresponding to a very conservative investor, who hates to lose money and is willing to pass up riskier high-return investments.

(c) Take a stab at what $s(m)$ might look like for you.

[27%] Problem 1 – **ASM Charts, Data Processing & Circuit Implementation**

In this problem we will build a fairly simple "pong"-like game involving two competitors. Essentially, we have 8 lights in a row, connected to a shift register. The lights illuminate in sequence, like a ball moving to the right; if player R pushes his switch too early or too late, then he loses. If player R pushes his switch *while* the rightmost light is lit, then the light starts moving to the left, towards player L. Player L needs to push her switch *while* the leftmost light is lit to return the light to R. This process continues until one of the players makes a mistake, at which point the other player wins.

Let's make this somewhat more specific. There are a total of three external inputs:

1. Reset (i.e., start over). This input doesn't need to show up in your ASM chart - you can just make it asynchronous. When the game starts L_8 is lit, and the light is moving to the right.
2. I_L – The switch signal from player L
3. I_R – The switch signal from player R

There are also ten outputs from the circuit:

1. $L_8 \ldots L_1$ – the outputs to the lights (L_8 on the left, L_1 on the right)
2. W_L – Player L wins
3. W_R – Player R wins

There is also a clock signal supplied.

Here's what you need to do:

[7] a) Write down an ASM chart for this problem.

[3] b) Draw the data-processing logic (recall that the circuits to implement ASM charts are divided into two separate pieces — the data-processing logic and the control logic).

[2] c) Specify the interface between control and data processing; that is, identify *clearly* the controls coming from the control logic, all status lines going back to the control logic, and any inputs from or outputs to the outside world.

[3] d) Write down the current-state — next-state table.

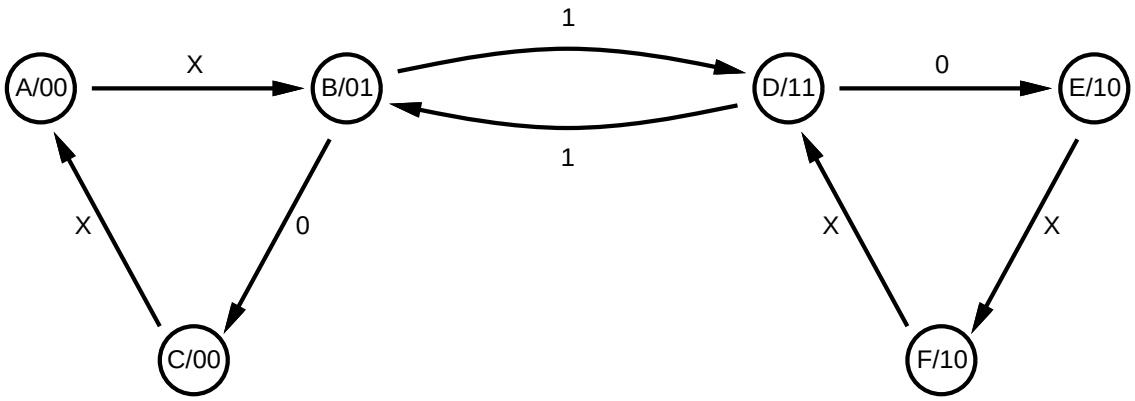
[8] e) Implement the control logic, using whatever method you prefer. I recommend the one flip-flop per state or the MUX/Reg/Decoder approaches, but any of the methods which we discussed in class will work fine.

[4] f) For the game to be playable, the frequency of the clock input should not be arbitrary. Suggest a reasonable range of values for the clock frequency.

Next, suppose the clock was too fast (e.g., 1kHz). Don't show me all of the details, but how would you need to modify your ASM chart or data processing to make the game work with such a fast clock?

[18%] Problem 2 – **State Diagrams & Circuit Implementation**

Suppose I give you the following state diagram (one input i, two outputs y_1, y_2):



[3] a) Starting in state "A", what is the state sequence and output sequence of this circuit given the input sequence 0 1 0 1 1 0 0 0 1 1 0 0

[1] b) Is this a Mealy or a Moore diagram?

[7] c) Design a circuit to implement the state diagram. Use the MUX/Reg/Decoder method to implement the circuit. Clearly label the devices in your final circuit.

[7] d) Implement the circuit using any approach of your choice, other than MUX/Reg/Decoder. If you use one FF per state, you will need to show an asynchronous Reset input.

[18%] Problem 3 – **Designing Counters**

In this problem we will be interested in the design of synchronous counters. To keep things simple, *all* of our counters will have the following properties:

- Unidirectional, synchronous, counting up
- Four bits in size (ie, we need four T-FFs to make the counter)
- All inputs are synchronous (ie, no asynchronous clear etc.)

[3] a) First, draw a standard four-bit unidirectional count-up counter. Clearly label the clock input CLK and your four outputs Q3, …, Q0.

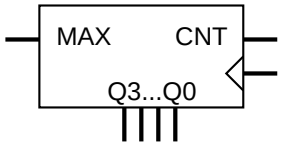
[5] b) The problem with the counter from (a) is that it counts every clock cycle; there are some applications where we might like to have a counter that counts on some clock cycles, but not on others.

Redraw your circuit from (a), adding a new synchronous input CNT. At a clock edge, if CNT is low the counter does not change in value; if CNT is high then the counter counts.

[6] c) The counter in (b) will count from 0 to 15, then back to 0 again. There are many cases where we want to count in decimal: that is, counting from 0 to 9, then back to 0 again.

Redraw your circuit from (b), such that the counter counts 0, 1, …, 9, 0, 1 etc. Also add a new synchronous output MAX. MAX is low for all values of the counter, except when the counter value equals 9, in which case MAX is high.

[4] d) Let the counter from part (c) be represented by a block



Recall how we were able to use multiple T FFs to make a four-bit counter in (a). Similarly we can group counters to count larger numbers. Suppose we have four decimal counters from (c); each counter produces one decimal digit (0, …, 9). Show how to wire up the four counters to count in base 10 like 0000, 0001, 0002, …, 9998, 9999, 0000 etc.

[22%] Problem 4 – **Short Question & Answer**

[3] a) In studying Boolean algebra, we discussed the *Duality Principle*. Define the Duality Principle and give a simple example.

[3] b) In class we spent some time discussing standard forms for Boolean functions using minterms and maxterms. What was the motivation behind studying standard forms? That is, how does having standard forms help us in working with digital logic?

[2] c) How is a JK flipflop different from an RS one?

[6] d) For the following Karnaugh maps, write both $f_1(x, y, z, w)$ and $f_2(x, y, z, w)$ in *both* simplest sum-of-products and product-of-sums forms.

xy \ zw	00	01	11	10
00	X	X	1	1
01	X	X	1	0
11	X	1	0	X
10	X	0	1	1

$f_1(x,y,z,w)$

xy \ zw	00	01	11	10
00	0	1	0	X
01	0	X	1	0
11	0	1	1	X
10	X	0	0	X

$f_2(x,y,z,w)$

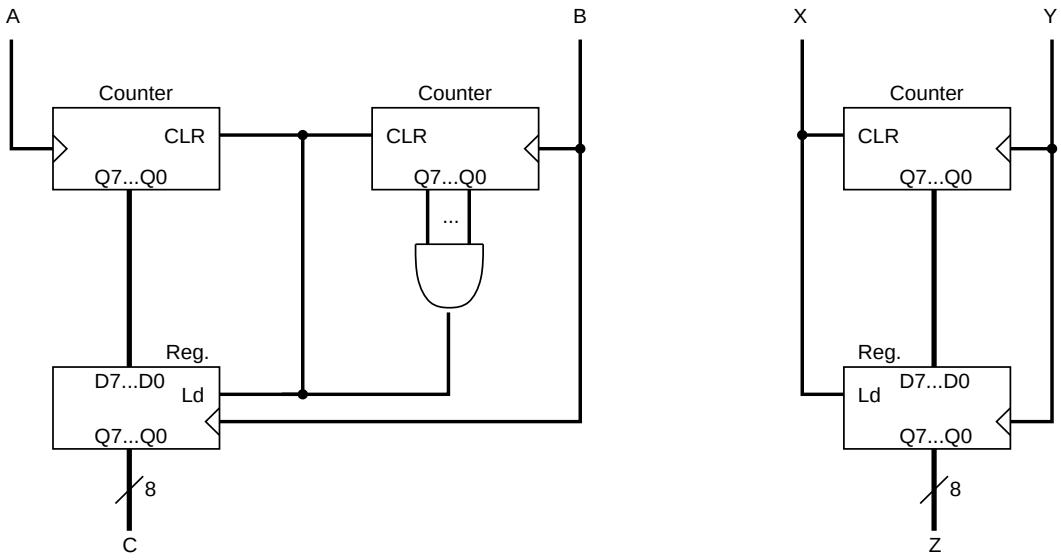
[5] e) We saw that we could group EPROMs together to allow for larger truth tables. Suppose I give you a bunch of 8-input 8-output EPROMs:

- Show how to make an 8-input 32-output EPROM;
- Show how to make a 10-input 8-output EPROM;
- Show how to make a 9-input 16-output EPROM.

[3] f) In class I have very often emphasized that *all* sequential devices in a circuit should have the *same* clock, and that we shouldn't try to modify the clock signal for different devices. Why? What is the danger in having multiple clocks or in modifying a clock signal (e.g., ANDing a clock with some other signal in order to enable / disable a counter).

[15%] Problem 5 – **Reverse Engineering – Counters**

Consider the following two circuits. We have inputs A, B, X, Y and outputs C, Z:



All of the counters and registers are 8-bit devices. All sequential devices are positive-edge triggered, and all "CLR" and "Ld" inputs are synchronous.

[5] a) What does the circuit on the right do? Be reasonably specific. Identify the roles of X, Y, Z as clearly as possible.

[6] b) What does the circuit on the left do? Be reasonably specific. Identify the roles of A, B, C as clearly as possible.

[4] c) Suppose that we set A and X to a 5 Hz clock, and we set B and Y to a 100 Hz clock. What will be the values of the outputs C, Z? Don't try to draw a timing diagram, just find the values of C, Z, showing your work.

Note: You don't need answers for parts (a),(b) to do this part.