Name: _____

CSE 370, Autumn, 2007, Final Exam

Please do not turn the page until instructed to do so.

Rules:

- Please remove everything from your desk area except one sheet of notes and whatever pens/pencils you want to use.
- Please stop working promptly at 10:20.
- If you rip the pages apart, please staple them back together when you are done.

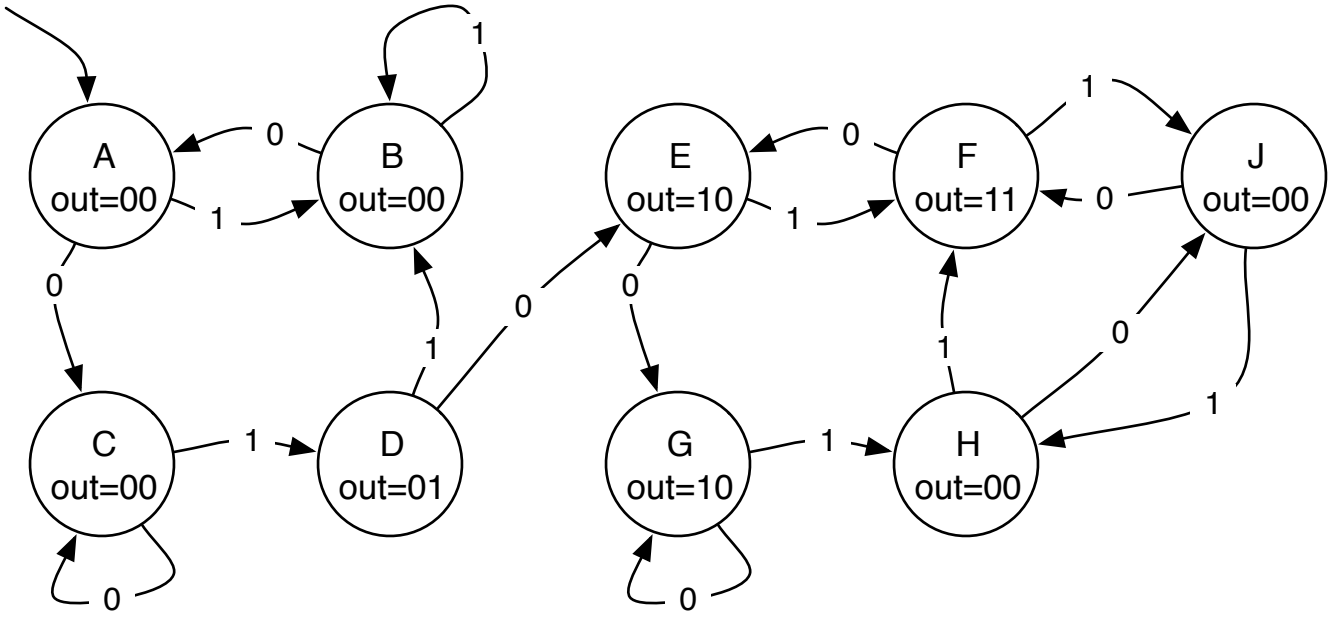
Advice:

- The exam should have 11 pages; check before you start.
- Read questions carefully before you start writing.
- Write down partial solutions for partial credit.
- There are 140 points on the exam distributed unevenly; try to distribute your effort roughly according to point value.
- The questions are not necessarily ordered according to difficulty. Skip around to find parts that are easy for you.
- If you have questions, ask.
- The last 2 exercises are "challenge exercises". They do not count towards your normal class score at all. If you complete them well, it could have a small effect when assigning final grades at the end of the quarter. Do not work on them unless you are 100% sure you are done with the rest of the exam.

Grading Summary	
1:	/ 15
2:	/ 20
3:	/ 20
4:	/ 20
5:	/ 10
6:	/ 15
7:	/ 20
8:	/ 20
Total:	/ 140

1. Finite state machine implementation (15 points)

For the following state machine, write three state encoding tables (<u>not</u> encoded state transition tables): one in the binary style, one in the one-hot style, and one in the output style. For each style, give one advantage a FSM implemented in that style would likely have over each of the other styles (a total of six advantages). You may "reuse" advantages, as long as they are accurate.



2. Important trivia: oxymoron? (20 points)

For each of the following questions, give a brief answer (one or two sentences).

What Boolean logic theorem are both K-map and Quine-McCluskey minimization based on? (Either of the two dual forms is fine.)

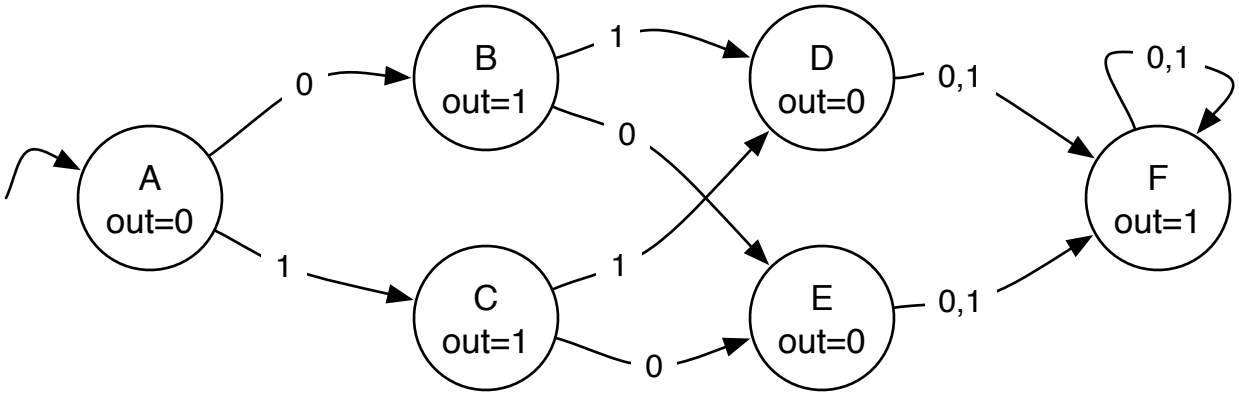
What is the fundamental structural difference between stateless (memoryless) circuits and circuits with memory?

What is the fundamental structural feature of circuits that creates the possibility of kinds of hazards we studied in 370? Remember that the glitches we talked about in 370 are all caused by a single input bit changing from 0 to 1 or 1 to 0. You can draw a little picture, if you want to.

An important trade-off in circuit design is size versus speed. Give one reason that given two equally well-designed circuits, one can be larger and faster than another.

3. State minimization (20 points)

Below is an example of a finite state machine that requires two "rounds" of minimization with the row matching method to be completely minimized. Draw the state transition table for this FSM and indicate which pair of states would be merged first and which pair would be merged second. You do **not** need to draw the minimized FSM.



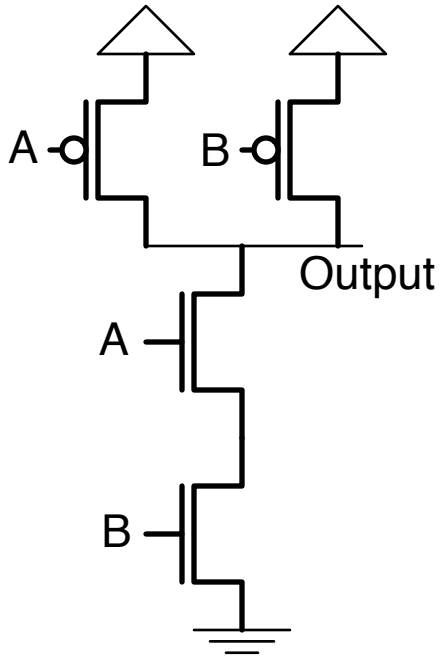
Some FSMs cannot be fully minimized with the row matching method, no matter how many rounds they are given. Draw a <u>simple</u> FSM that can be minimized with the implication chart method, but not the row matching method. Examples with a very small number of states exist, though any correct example will get full credit.

4. Control-datapath design (20 points)

Design a datapath and finite state machine controller to implement the following specification. Your circuit should store two 16-bit numbers X and Y, which are initialized by the magic wand method. Every clock cycle the circuit takes a new 16-bit number as input. In the normal operating mode, the circuit will add the input to X to compute a new value for X. If X gets larger that Y, the circuit should go into a "cool down" mode for three clock cycles, where the inputs are subtracted from X instead of added, and then resume normal operating mode. If X ever overflows, the circuit should go into an error state and remain there. You may use the conventional arithmetic operations (add subtract, mux, compare) as primitive blocks.

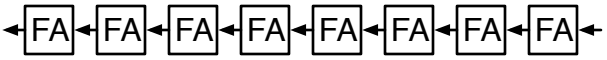
5. Transistors (10 points)

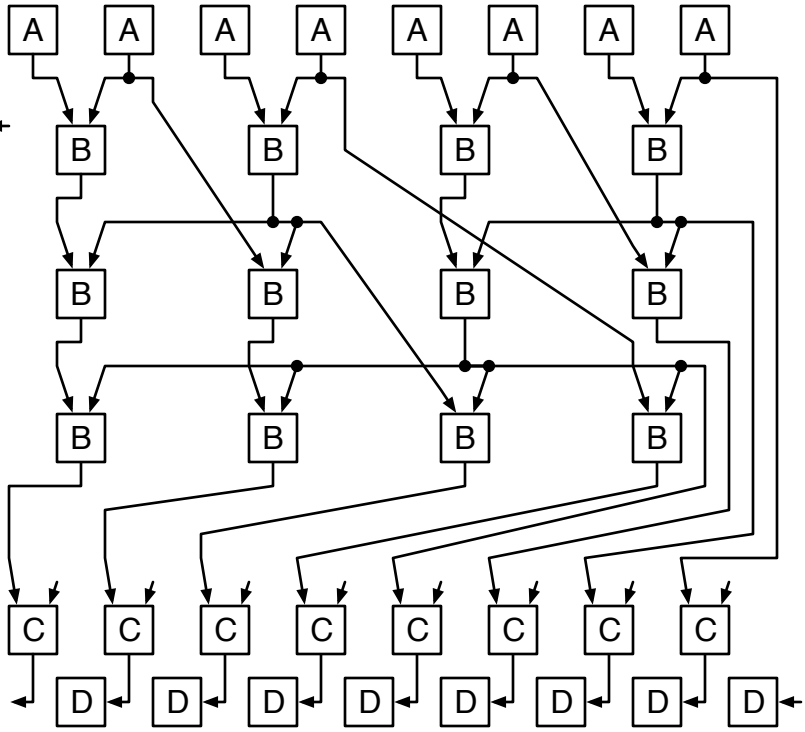
As a reminder, below is a schematic for a 2-input NAND gate in the static CMOS style. Draw a 3-input OR gate in the static CMOS style.



6. Ripple-carry versus carry look-ahead (15 points)

Ripple-carry adders (RCAs) are smaller and slower; carry look-ahead adders (CLAs) are bigger and faster. Below, as a reminder, are a sketch of an 8-bit RCA, and an 8-bit CLA. The letters (A,B,C,D) in the CLA are there just to identify the four different kinds of building blocks it is constructed from. Use the information in the component size and delay table below to complete the size and delay table for adders of different numbers of bits.





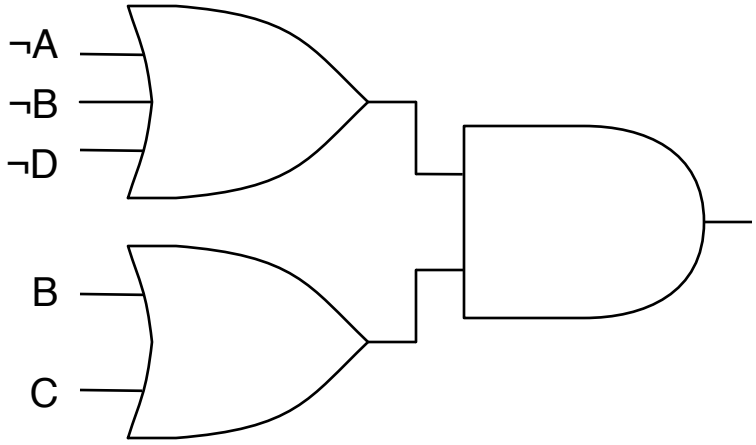
Component	Size	Delay
A	3	2
B	3	2
C	2	2
D	4	4
FA	8	4

Adder bits	RCA Size	RCA Delay	CLA Size	CLA Delay
2	16	8	21	10
4	32		48	
8	64		108	
16	128		240	
32	256		528	

One metric used to compare circuits of different sizes and delays is the size-delay product. As the name suggests, one simply multiplies the size and delay of a circuit together to get its size-delay product. Notice that for 2 bits, the RCA is both smaller and lower delay than the CLA, meaning that it wins easily in size-delay product. What is the smallest number of bits for which the CLA has an advantage in size-delay product?

7. Eliminate the static hazard (20 points)

This circuit has a static hazard. Identify an input pattern transition that could trigger this hazard, and add a single OR gate (and a single AND gate input) to fix the hazard.



8. Finally, some correct Verilog (20 points)

Translate this Verilog code into a finite state machine drawing and a datapath schematic. Assume that all variables are properly declared. i is an input to the datapath.

```
always @ (posedge clk) begin
  if (rst) begin
    state <= 2'b00;
    x <= 8'b00000000;
  end
  else begin
    state <= nextState;
    x <= nextX;
  end
end

always @ (state or x or i)
begin
  nextState = state;
  nextX = x;
  if (i < x) begin
    nextX = x - i;
    nextState = 2'b10;
  end
  else if (state == 2'b00) begin
    nextX = x + i;
    nextState = 2'b11;
  end
  else if (state == 2'b10)
    nextState = 2'b11;
  else
    nextState = 2'b00;
end
```