University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

EECS150 J. Wawrzynek
Spring 2010 3/31/09

Midterm Exam

Name: _____

ID number: _____

This is a *closed-book, closed-note* exam. No calculators or any other electronic devices, please.

Read all the questions **before** you begin. Each question is marked with its number of points (one point per expected minute of time). Although you might not need it, you have until 9pm.

You can tear off the spare pages at the end of the booklet and/or use the backs of the pages to work out your answers. Neatly copy your answer to the places allocated for them.

Neatness counts. We will deduct points if we need to work hard to understand your answer. **Simplicity also counts.** In the design problems, correct simpler designs with fewer components will be awarded a higher score than more complex designs with more components.

Put your name and SID on each page.

problem	maximum	score
1	8pts	
2	6pts	
3	10pts	
4	15pts	
5	12pts	
6	10pts	
7	24pts	
8	16pts	
9	19pts	
Total	120pts	

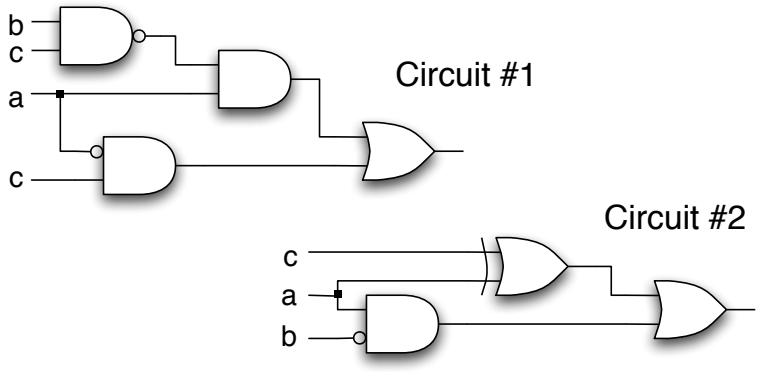
1. Multiplexor Implementation [8pts].

   Consider the design of a 4-to-1 multiplexor circuit with four data inputs, `d0`, `d1`, `d2`, and `d3`, two control inputs, `s0` and `s1`, and a single output, `y`.

   Using only simple logic gates (ANDs, ORs, NANDs, NORs, inverters), but no transmission gates, sketch the circuit diagram for a multiplexor circuit *optimized for minimum delay* from the data inputs, `d0`–`d3`, to the output, `y`. You may use gates with any number of inputs, but remember that the delay through a logic gate grows with the square of the number of inputs.

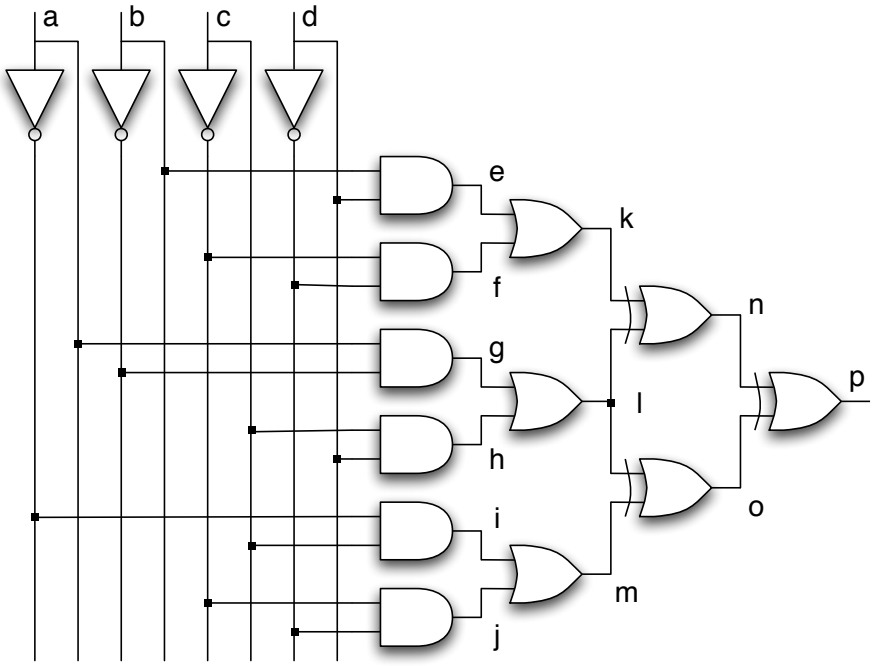
2. Combinational Logic Circuits [6pts].

   Using whatever means possible, prove or disprove that the two combinational logic circuit shown below have equivalent function. Explain your approach and show your work.



3. FPGA Mapping [10pts].

Using only 3-input lookup tables (LUTs), partition the circuit shown below into as few LUTs as possible. *Do not attempt to simplify the gate-level circuit before mapping it to LUTs.* Indicate your answer by filling in the table. Fill in one row for each LUT, assigning node names from the circuit to LUT inputs and outputs. Mark unused LUT inputs with "X" (for unused).

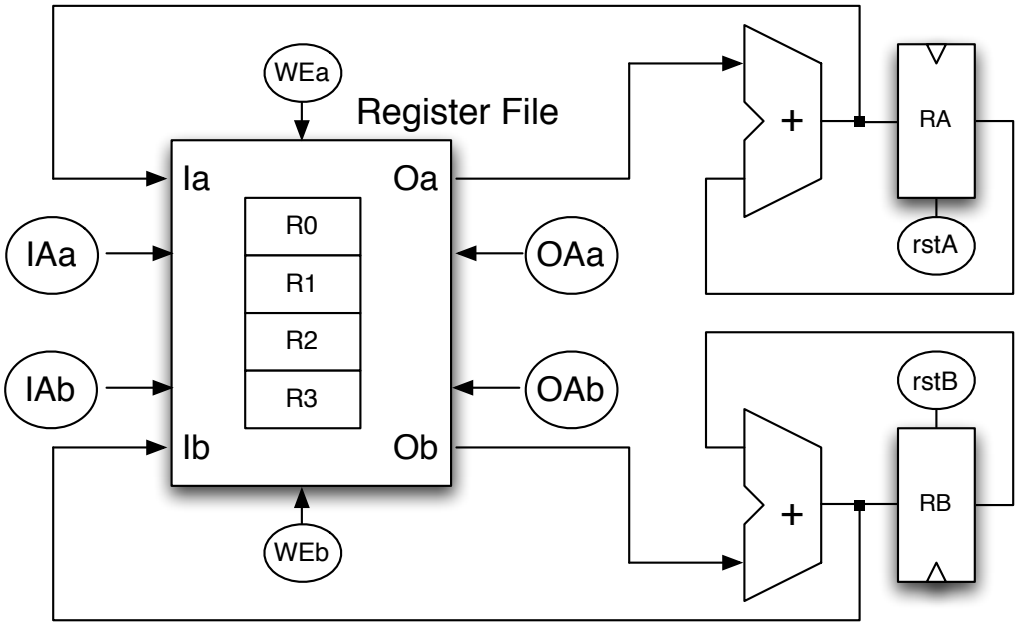


LUT #	input 1	input 2	input 3	output
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				

4. Register Transfers [15pts].

Consider the datapath shown below. A controller (not shown), is used to control operation of the datapath. It sets the value of all the control signals (circled in the datapath diagram).

The block labeled "Register File" stores four data registers, R0–R3, has two asychronous read ports, and two synchronous write ports. The four port addresses are IAa, IAb, OAa, and OAb. Writing to the register file is controlled by the WEa and WEb signals. The data registers at the output of the adders, RA and RB, have synchronous reset inputs, rstA and rstB, respectively.



(a) Assume that the register file is initialized with registers R0–R3 holding the values a, b, c, and d, respectively. Registers RA and RB begin uninitialized.

Your task is to generate the control signal sequence which will result in a in R0, $a+b$ in R1, $a+b+c$ in R2, and $a+b+c+d$ in R3, and do so in the minimum number of clock cycles. Indicate your answer by filling in the table with the control signal values that the controller would generate on each clock cycle (for use on the next positive clock edge).

cycle #	IAa	IAb	OAa	OAb	WEa	WEb	rstA	rstB
1								
2								
3								
4								
5								
6								
7								
8								

(b) Now assume that the Register File has a read access delay of 2ns, a write setup time of 1ns, and a write delay of 1ns, i.e., the written data appears in the proper register 1ns after the clock edge. There is no register file bypassing.

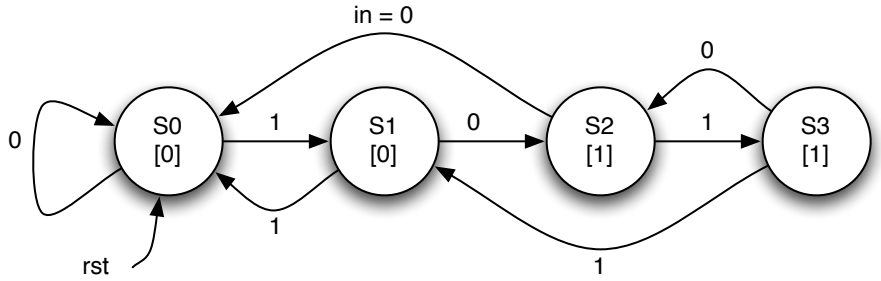
The adders have a combinational logic delay of 4ns, and the output registers have a setup time, 1ns and clock-to-q delay of 1ns, and a hold time of 1ns. Ignore wire delay and clock skew.

What is maximum clock frequency for this circuit?

5. Verilog and Finite State Machines [12pts].

   For the state transision diagram shown below:

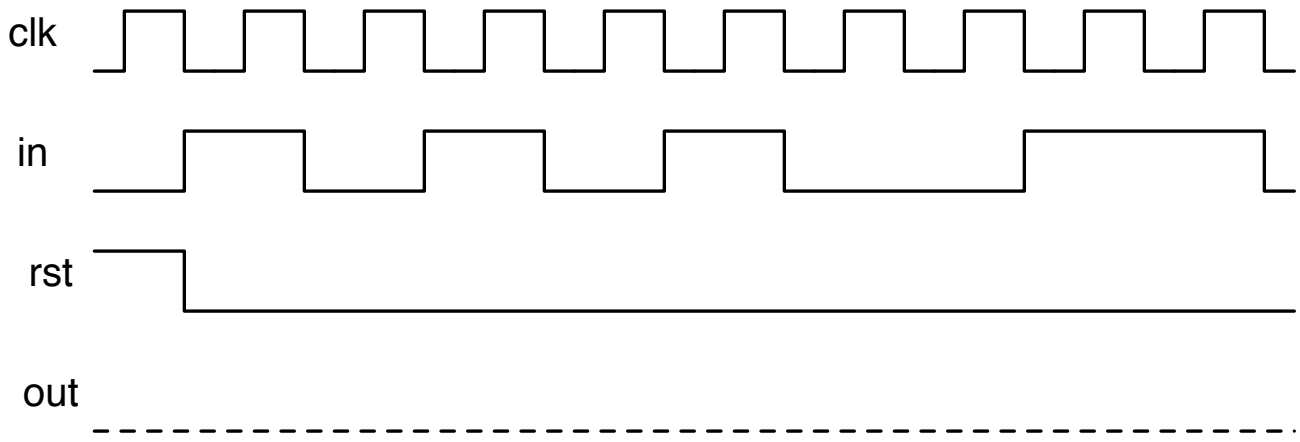
   (a) Complete the Verilog description, following the CS150 style rules.




```
module FSM(clk, rst, in, out);
input clk, rst;
input in;
output out;
```

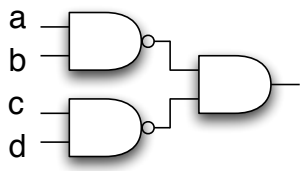
```
endmodule
```

(b) Complete the waveform for "out" corresponding the input signals shown below:



6. Transistor Circuits [10pts].

   Draw a transistor level circuit diagram for the function depicted in the gate level circuit below. Minimize the total number of transistors. No "pass-transistor logic", use only "static cMOS" circuits.



7. Stack Machine Design [24pts].

A stack machine is a type of CPU that uses a hardware stack instead of a register file. All instructions take their operands from the stack and leave their result on the top of the stack. Remember, a stack is a "LIFO" (last in first out data structure) and typically supports "push" and "pop" operations.

In this problem you will design the datapath and specify the control for the following subset of a stack machine instruction set:

Instruction	Description
add	Pops two elements, forms their sum, and pushes the result.
sub	Pops two elements, subtracts the first popped from the second, and pushes the result.
dup	Duplicates the top of the stack.
swap	Exchanges the first two elements on the stack.
load	Loads a word from data memory using the top of the stack as the memory address (address is popped, the data from memory is pushed).
store	Stores the top of the stack in data memory using the next element on the stack as the address (both are popped).
const	Pushes a sign-extended immediate value from the instruction.

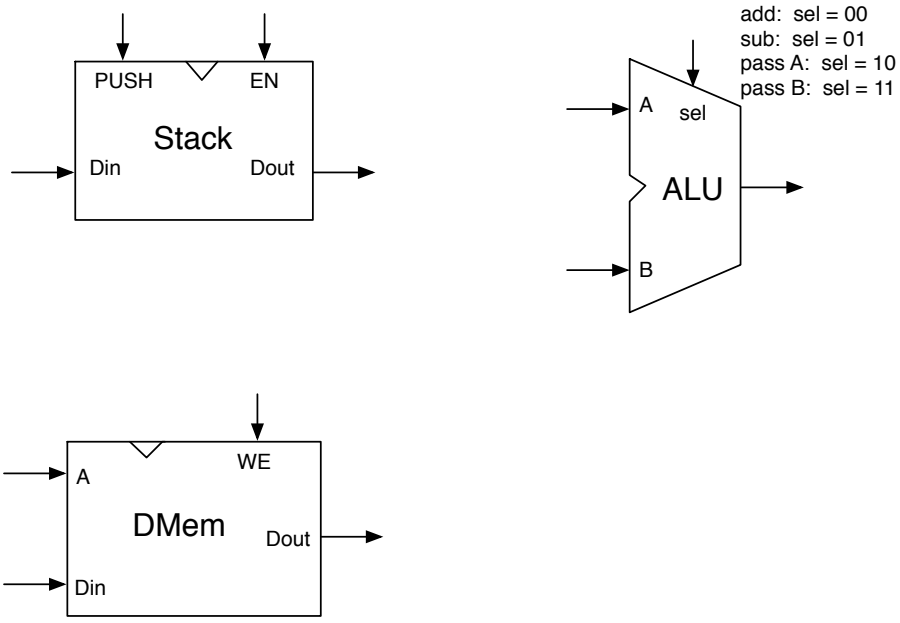
Shown below are a few blocks that you will need for your datapath. The data memory block has asynchronous read and synchronous write. The stack block has a data input and data output, along with two control signals, push (PUSH), and enable (EN). The top of the stack is always available on the data output signal Dout, i.e., it is always possible to peak at the top of the stack. On the positive edge of the clock, if PUSH=1 and EN=1 then the value on the data input is pushed onto the stack; if PUSH=0 and EN=1 then the top of the stack is popped; if EN=0, then the stack is not changed.

(a) In the space provided below add wires and any other blocks that you need for the above instructions. Be neat, and avoid crossing wires when possible. Circle your control signals. You will assign control signal values in part (b), coming up.

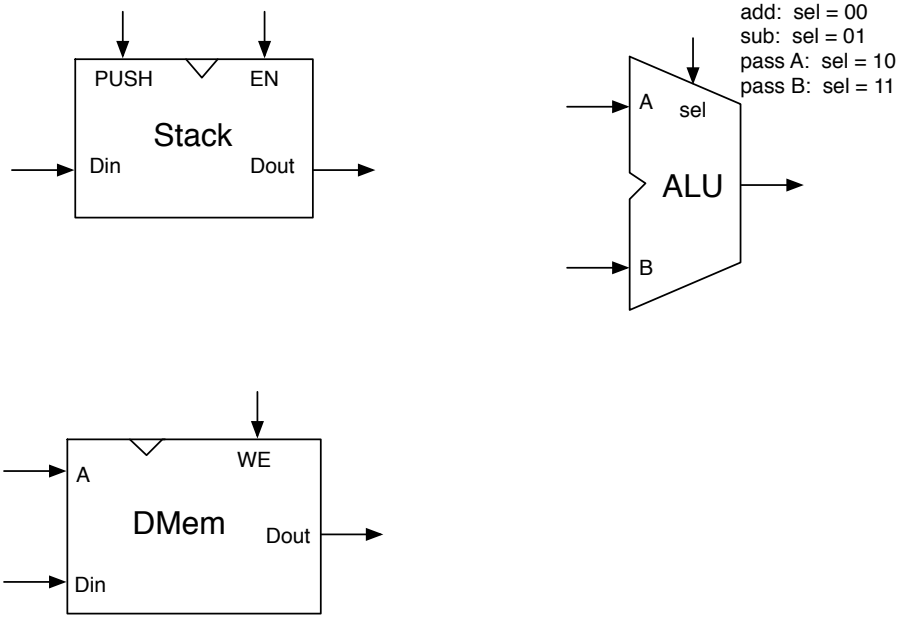
Your top priority is to minimize the number of cycles per instruction, followed by making the datapath as simple as possible. You may ignore instruction fetch for this problem.

Note, two versions of the partial datapath diagram are included. The first one is a working/practice version. Once you are confident in your answer, copy it to the second diagram below. We will only grade the second version.

Practice Version (will not be graded)



Final Version for Grading



(b) Label the table columns with your control signal names. Fill in the table with the control signal values for each instruction. Use "X" to indicate "don't care". Note that some instructions may take more than one cycle. In those cases, use one row for each cycle of the instruction.

add										
sub										
dup										
swap										
load										
store										
const										

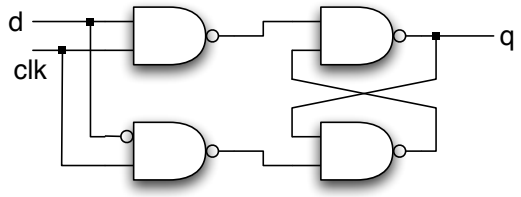
(c) Most stack machines also include a rotate instruction:

Instruction	Description
rot	"Rotate" the top 3 elements; the third element moves to the top and the top two elements move down one.

Is it possible to execute a rotate instruction on your machine without modifying the datapath? If so, how many cycles will it take?

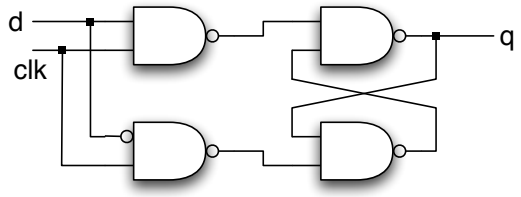
8. Flip-flop Implementation [16pts].

Consider the circuit shown below. Its function is a level-sensitive "high transparent" latch—transparent when the clock level is high.

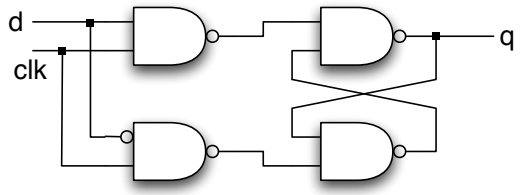


If possible, modify the circuit to achieve the following functions by adding bubbles (inversions) and extra gate inputs (no extra gates or transistors are allowed). If the desired function is not possible, indicate it by writing "not possible".

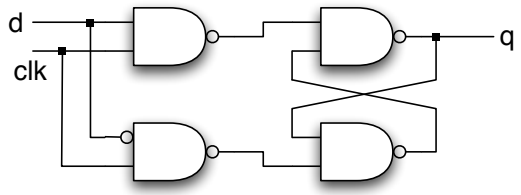
(a) Clock enable (CE) input.



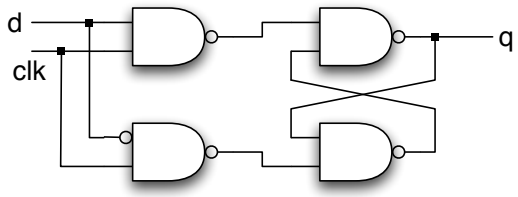
(b) Asynchronous Reset (clear)—sets the latch value to 0 independently of the clock.



(c) Synchronous Reset (reset)—set the latch value to 0 if the clock is high.



(d) Negative Edge-triggered Flip-Flop. For this part you may add another copy of the latch, if needed.



9. Short Answers [19pts].

(a) [1pt] List the primary reason why FPGAs have lower performance than ASICs.

(b) [4pt] Over the years, the size of LUTs have been increased by the FPGA manufacturers (from 3-LUTs to, now, 6-LUTs).
List two reasons why larger LUTs are a good idea:

List two reasons why smaller LUTs are a good idea:

(c) [3pt] Assume a product development based on an FPGA has a NRE cost of $1M and a per unit FPGA cost of $100. An ASIC version of the product has an NRE cost $10M and a per unit cost of $10. Your company plans to ship 10,000 units of the final product. Would you choose FPGA or ASIC, and why?

(d) [2pt] List two reasons why you might choose to use Block RAM (BRAM) over distributed RAM (LUT-RAM) in an FPGA based design.

(e) [1pt] List one reason why you might choose to use LUT-RAM over BRAM in an FPGA based design.

(f) [4pt] Suppose you are given a 256 X 8 simple-dual-ported memory block. How many such memory blocks would it take to implement a 512 X 16 memory with two read ports and two write ports?

(g) [3pt] Consider a video system with a 1K by 1K image, 100 frames/second, and 3 Bytes per pixel. We would like to send a video stream over Ethernet and have available 10Mbps, 100Mbps, and 1000Mbps connections. Which of these connections would provide sufficient bandwidth?

(h) [1pt] Put your name and SID on each page.

Spare page. *Will not be graded.*

Spare page. *Will not be graded.*

Spare page. *Will not be graded.*

Spare page. *Will not be graded.*

Spare page. *Will not be graded.*