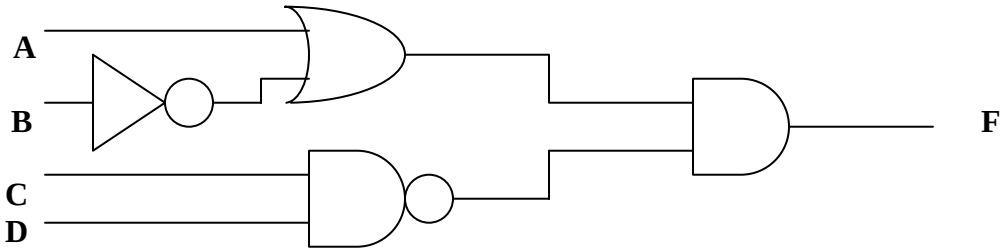
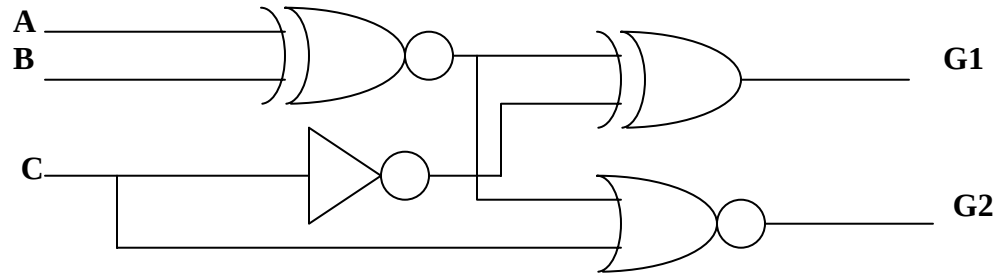
COURSE READER PROBLEMS

Problem 1 (10 points): Write down Boolean expressions for the outputs of the following circuits (you do not need to simplify these expressions). Give the truth table for each output. (You may write a single table with two output columns for the second circuit).

(a)



(b)



Problem 2 (15 points): This problem extends the staircase lamp example in class to three switches. Assume that we have a lamp in the garden that can be controlled by a switch in the bedroom, a switch in the garage, and a switch at the home entrance. If you flip any one of these switches, the garden lamp will flip (that is, from ON to OFF or OFF to ON). Design a controller for the garden lamp (you must define your input and output variables clearly to receive any credit). Write down a Boolean equation and give a circuit schematic using only XOR gates.

Problem 3 (14 points): A flow rate sensing device used on a liquid transport pipeline functions as follows. The device provides a 5-bit output where all five bits are zero if the flow rate is less than 10 gallons per minute. The first bit (**MSB**) is 1 if the flow rate is at least 10 gallons per minute (and the rest of the bits are zero); the first and the second bits (**MSB**) are 1 if the flow rate is at least 20 gallons per minute (and the rest of the bits are zero); the first, second, and third bits (**MSB**) are 1 (and the rest of the bits are zero) if the flow rate is at least 30 gallons per minute; and so on. The five bits, represented by the logical variables A, B, C, D, and E, (**MSB** to **LSB** in the given order) are used as inputs to a device that provides two outputs Y and Z.

(a) Write an equation for the output Y if we want Y to be 1 iff the flow rate is less than 30 gallons per minute. Draw a schematic for Y in terms of AND, OR, NOT gates.
(b) Write an equation for the output Z if we want Z to be 1 iff the flow rate is at least 20 gallons per minute but less than 50 gallons per minute. Draw a schematic for Z in terms of AND, OR, NOT gates.

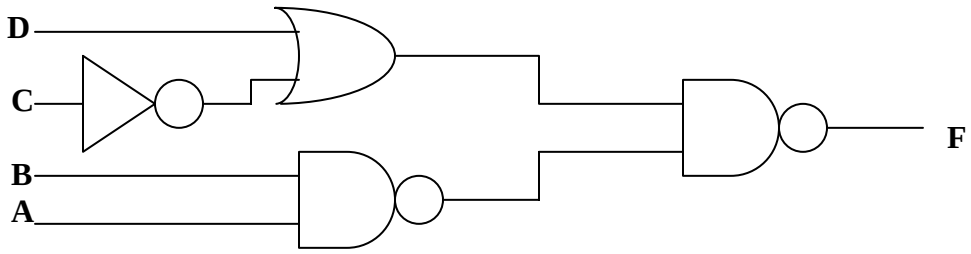
Problem 4 (16 points): For each of the following statements, state whether the statement is true or false. If the statement is true, prove it algebraically. If the statement is false, prove this fact by giving a counter-example. (Note that a counter-example must specify a concrete set of input values that falsifies the statement.)

(a) If A + B = C, then AD' + BD' = CD'
(b) If A'B + A'C = A'D, then B + C = D
(c) If A + B = C, then A + B + D = C + D
(d) If A + B + C = D + C, then A + B = D

When you come to an exam, keep in mind what you have learned in this problem, since some of the above constitute common mistakes in manipulating Boolean equalities.

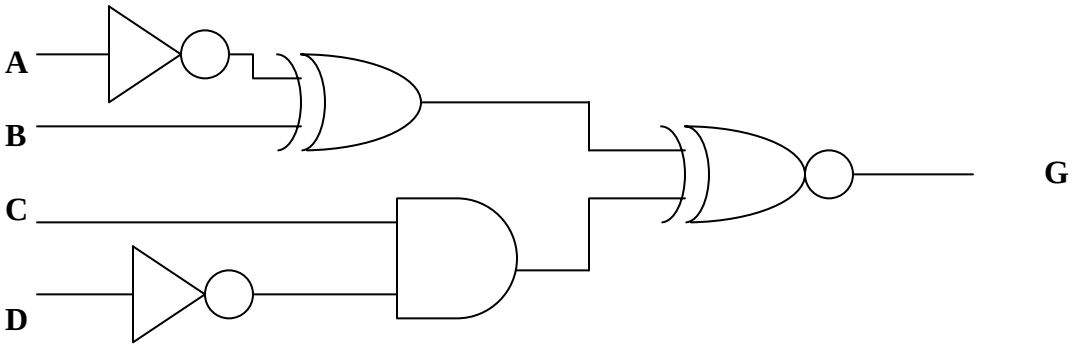
Problem 5 (15 points): De Morgan's laws and their bubble-pushing interpretation are very important in logic design. In this problem, you will get experience in simplifying circuits using bubble-pushing:

(a)

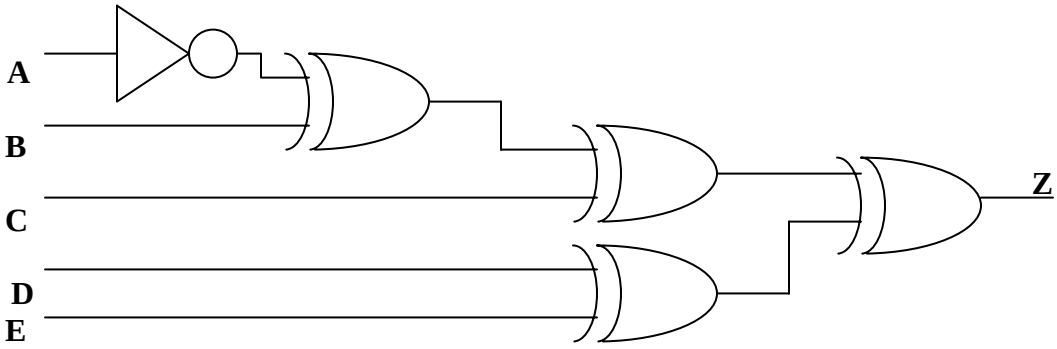


Find a simple expression for F by pushing the bubbles from the output towards the input variables, step by step. Write down a simplified expression for F at the end.

(b) We also showed bubble-pushing for the XOR gate in the class. Simplify the following circuit. By bubble-pushing, obtain the simplified expression for G that uses 1 inverter, 2 XOR gates, and 1 AND gate.



(c) Using bubble-pushing, show that A' ⊕ B ⊕ C ⊕ D ⊕ E = A ⊕ B ⊕ C ⊕ D ⊕ E'.



Problem 6 (20 points): We know that the AND and OR operations are defined to be the duals of each other; that is,

$$(A \bullet B)^D = A + B$$
$$(A + B)^D = A \bullet B$$

(a) Are the NAND and NOR operations duals of each other? For the purposes of this exercise, denote the NAND operation by the symbol $\downarrow$ and denote the NOR operation by the symbol $\uparrow$. Then, we are asking whether the following statements are true:

$$(A \downarrow B)^D = A \uparrow B$$
$$(A \uparrow B)^D = A \downarrow B$$

(b) Does the XOR operation have a dual? If so, find its dual. If not, prove that it does not have a dual.

Problem 7 (30 points):
(a) Using the K-maps below, perform logic minimization to find all minimum sum-of-products (SOP) forms. (Write down the Boolean expressions for each.) Clearly show both on the K-map and on the Boolean expression, which implicants are prime and which are prime essentials.

Output F:

CD \ **AB**	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	1	1	1
10	1	0	0	0

Output G:

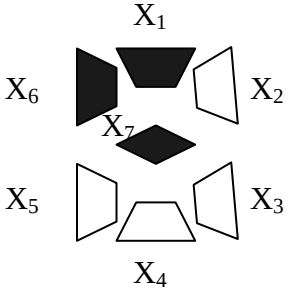
C \ **AB**	00	01	11	10
0	X	X	0	0
1	X	1	X	X

Output H:

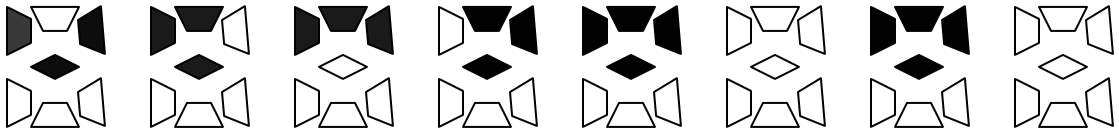
CD \ **AB**	00	01	11	10
00	0	X	1	X
01	1	X	0	1
11	X	0	1	X
10	1	X	X	0

(b) For each K-maps in Part (a), find all minimum product-of-sums (POS) expressions. Clearly show both on the K-map and on the Boolean expression which implicants are prime and which are prime essential.

Problem 8 (20 points): Design a “disk spinning” animation circuit for a CD player. The input to the circuit will be a 3-bit binary number $A_1A_2A_3$ provided by another circuit. It will count from 0 to 7 in binary, and then it will repeat. (Hence, you should not design this external counter.) The animation will appear on the top four lights of the seven-segment display in the following figure:



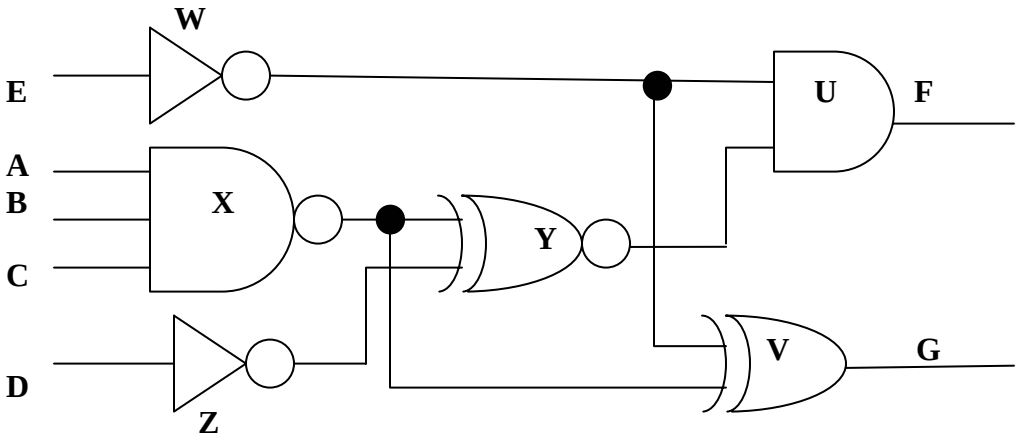
In the figure below, we display the animation output corresponding to each input that comes from the counter. Note that the animation displays on X_1, X_2, X_7 and X_6, going clockwise. Then, upon "101", it displays blank. Upon "110", it displays the full 4 blocks. Upon "111", it displays blank again. Then, it goes back to its revolving pattern beginning with "000".



000 (A1A2A3) **001** **010** **011** **100** **101** **110** **111**

Design a digital circuit that takes the input from the counter and displays the correct output pattern on the 7-segment display. Your circuit should use only 2-, 3-, and 4-input NOR gates and inverters. Try to minimize the number of gates required. Any solution which uses 11 or fewer gates (not counting the four inverters for the inputs) is acceptable.

Problem 9:



In this problem, the letters U,V,W,X,Y,Z label gates; A, B, C, D, E are inputs, and F and G are outputs.

Assumptions: Assume that the gates have the following delays:

Number of Inputs	Gate Type	Delay (ns)
3-inputs	NAND	4
2-inputs	NOR	4.5
2-inputs	AND	5
2-inputs	OR	5.5
2-inputs	XOR, XNOR	3
1-input	NOT	2

Assume that the external wires have negligible (i.e. zero) delay.

Each part of this problem may introduce some additional assumptions. These additional assumptions are FOR THAT PART ONLY. Do not carry these additional assumptions into the other parts of the problem.

(a) **(1 point)** What is the gate count of the above circuit?
(b) **(2 points)** How many levels of logic are there for each of the output functions F and G? (Note that we assume that the complements of inputs are not available.)
(c) **(2 points)** What is the number of levels for this multiple-output circuit?
(d) **(6 points)** In a table, show the fan-in and fan-out of each of the gates (U,V,W,X,Y and Z) shown in the diagram. (These are numbers.)
(e) **(4 points)** What is the maximum fan-in in this circuit? Which gates have the maximum fan-in? What is the maximum fan-out in this circuit? Which gates have the maximum fan-out?
(f) **(8 points)** What is the maximum delay of the above circuit?
(g) **(8 points)** What is the minimum delay of the above circuit?

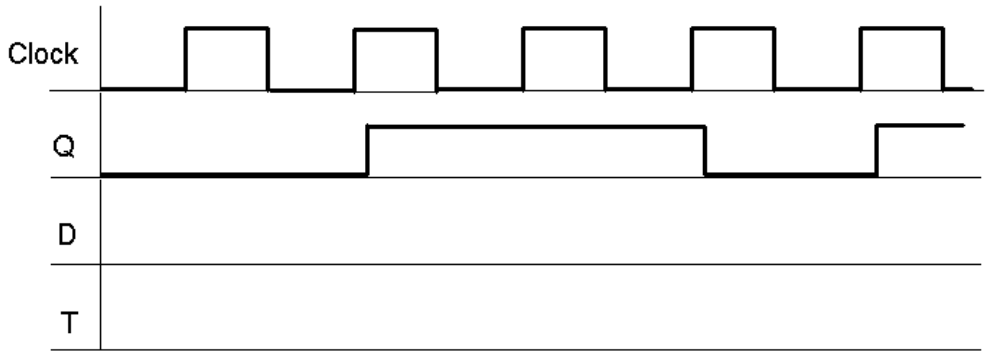
(h) **(15 points)** What is the critical path in the above circuit? (Indicate each path by the input variable sequence of gates and the output variable.) What is the critical delay? List all of the critical transitions.

(i) **(10 points)** Assume that the device technology has improved and the 3-input NAND gate has a delay of 2 ns (instead of 4 ns) and the 2-input AND gate has a delay of 3 ns (instead of 5 ns). Does the critical path of the circuit change? If so, give the new critical path and the maximum propagation delay.

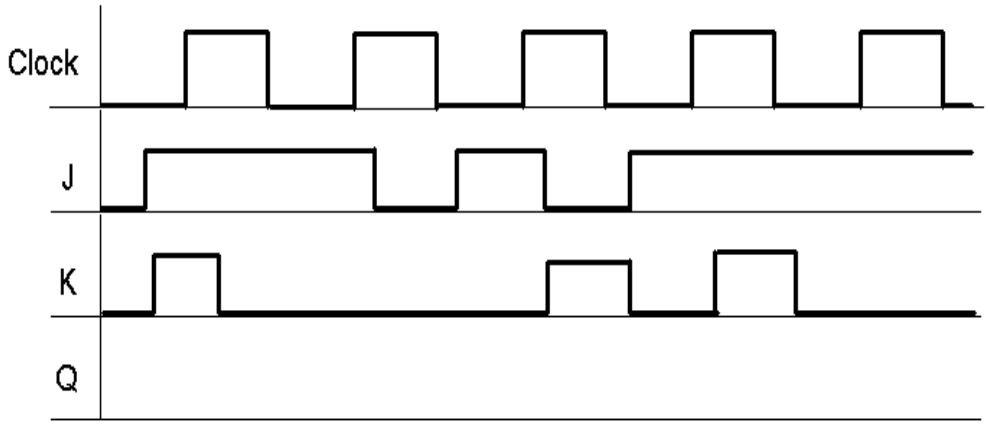
Problems 10-14 of the course reader are in hand-written format and will be provided in a separate file. This will be a link on the web page next to the main course reader link.

Problem 15:

(a) **(10 points)**: Find the input to a **positive-edge triggered** D flip-flop which would produce the output Q as shown. Repeat for a **positive-edge triggered** T flip-flop.
Note: The slight delay in Q is due to the propagation delay of the flip-flop (which we have not covered in class yet, but we will later.)



(b) **(10 points)** Fill in the timing diagram for a **negative-edge triggered** JK flip-flop for when Q = 0 initially. Repeat the process for when Q = 1 initially.

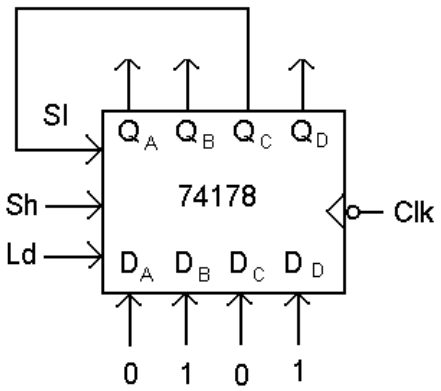


Problem 16 (20 points):

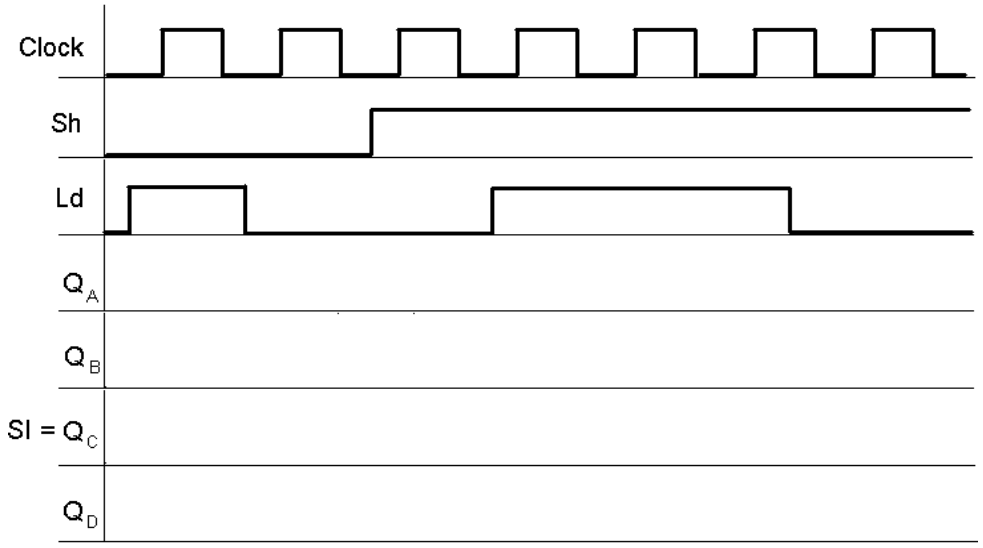
A 74178 shift register works as follows: All state changes occur on the 1-0 transition of the clock (i.e. on the falling edge). The control signal Sh instructs the shifter to shift, and the control signal Ld instructs the shifter to load the 4-bit data vector. Below, to the right, we have displayed a table of the action that is taken upon different valuations of (Sh, Ld). For example, when (Sh, Ld) = (0, 0), the shifter holds the current state. When (Sh, Ld) = (0, 1), the shifter loads the data vector. When Sh = 1, regardless of what Ld's value is, the shifter shifts the bits to the right as shown. The input SI is the "serial input"; namely, the bit that shifts in.

We have connected this shift register as shown below on the left. Note that Q_C is fed back to the Serial Input (SI) terminal of this shifter. Also note the particular data input vector specified in the diagram.

Based on this information, complete the timing diagram below.



Sh	Ld	Q_A^+ Q_B^+ Q_C^+ Q_D^+
0	0	Q_A Q_B Q_C Q_D
0	1	D_A D_B D_C D_D
1	X	SI Q_A Q_B Q_C



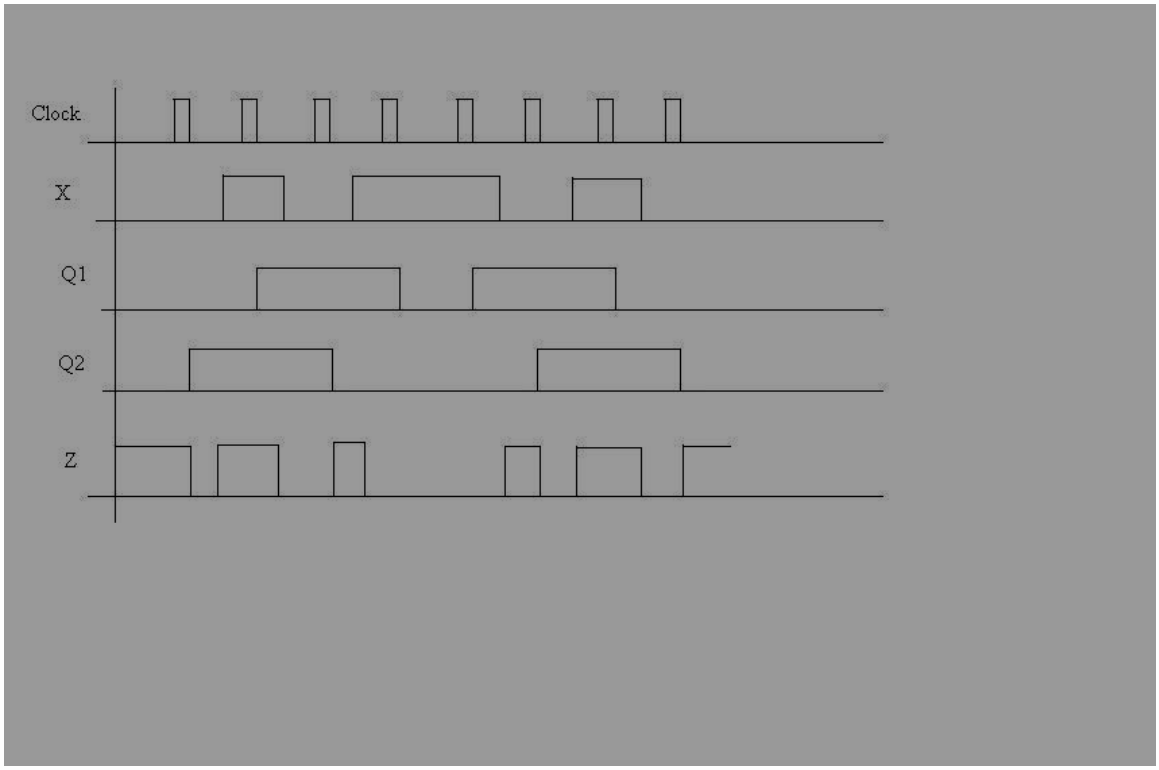
Problem 17 (20 points):

Design a synchronous 2-bit, binary, up-counter using the following specification: The counter has five inputs (not including the clock) and three outputs. The inputs are CLR ("Clear"), LOAD, COUNT, L_B, and L_A. CLR takes precedence over LOAD, which in turn takes precedence over COUNT. The outputs are B, A, and RCO. The output RCO goes high only when the counter has reached the highest value ("11" in this case).

(a) Draw the interface schematic of this counter,
(b) Decide how you can implement this using flip-flops. (You may use D, JK or T flip-flops.)
(c) Draw the implementation schematic.
(d) Draw a timing diagram to illustrate the correct operation of this counter, as it goes through the entire sequence and wraps around. On this timing diagram, you need to display the waveforms for all inputs and outputs. You need to show how you achieve "reset" on this counter in the beginning, how the counter advances and then wraps around. (Through this exercise, you will gain a better understanding of what signals are asserted exactly when.)

Problem 21 (30 points):

A Mealy sequential circuit has one input, one output, and two flip-flops. A timing diagram for the circuit follows. Construct a state table and a state diagram for the circuit. X is the input and Z is the output, and Q1 and Q2 are the states of the two flip-flops.



Problem 22 (80 points):
A sequential circuit has two inputs and two outputs. The inputs X_1 and X_2 represent a 2-bit **unsigned** binary number denoted by N. Here, X_1 is most significant bit of N, and X_2 is the least significant bit of N. The outputs are denoted by Z_1, Z_2. If the present value of N is greater than the previous value, then Z_1 is 1. If the present value of N is less than the previous value, then Z_2 is 1. Otherwise, Z_1 and Z_2 are 0. When the first pair of inputs is received, there is no previous value of N, so we cannot determine whether the present N is greater than or less than the previous value; therefore, the "otherwise" category applies, i.e. both $Z_1$and Z_2 are 0.

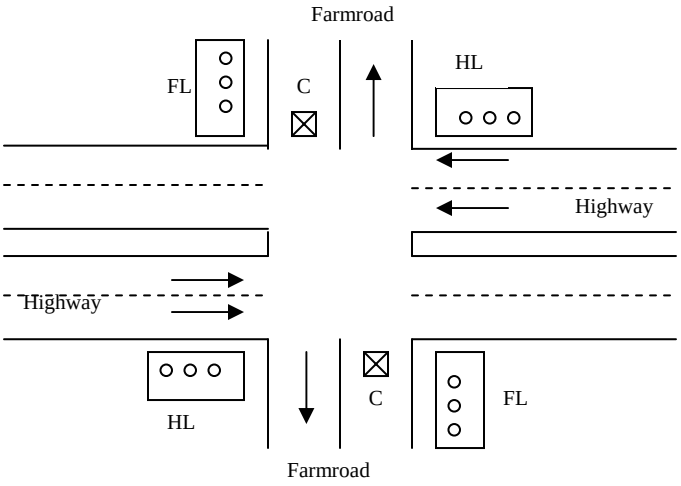
(a) **(20 points)** Write down a Mealy state diagram and state table for the circuit.
(b) **(20 points)** Write down a Moore state diagram and state table for the circuit.
(c) **(20 points)** Write a Verilog module that implements the Mealy machine from Part (a) in **behavioral Verilog**.
(d) **(20 points)** Write a Verilog module that implements the Moore machine from Part (b) in **behavioral Verilog**.

Problem 23 (40 points):
Design a sequential circuit to control a phone answering machine. The circuit should have three inputs, R, A and S, and one output Z. The operation is as follows: R=1 for one clock cycle at the end of each phone ring. A=1 when the phone is answered. S selects whether the machine should answer the phone after two rings (S=0) or four rings (S=1). To cause the tape recorder to answer the phone, the circuit should set the output Z=1 after the end of the second (S=0) or fourth (S=1) ring, and hold Z=1 until the recorder circuit

answers the phone (i.e. when A goes to 1). If a person answers the phone at any point, A will become 1, and the circuit should reset. Assume that S is not changed while the phone is counting rings.

(a) **(20 points)** Give a **Moore** state graph and state table for this circuit. (Note: A "state graph" is the same as a "state diagram".)
(b) **(20 points)** Write a Verilog module to implement the Moore machine from Part (a) in **behavioral Verilog**.

Problem 24 (60 points):
A busy highway is intersected by a little-used farmroad, as shown in the figure below. Detectors are placed along the farmroad to raise the signal C as long as a vehicle is waiting to cross the highway. The traffic light controller should operate as follows. As long as no vehicle is detected on the farmroad, the lights should remain green in the highway direction. If a vehicle is detected on the farmroad, the highway lights should change from yellow to red, allowing the farmroad lights to become green. The farmroad lights stay green only as long as vehicle is detected on the farmroad and never longer than a set interval to allow the traffic to flow along the highway. If these conditions are met, the farmroad lights change from green to yellow to red, allowing the highway lights to return to green. Even if vehicles are waiting to cross the highway, the highway should remain green for a set interval. You may assume there is an external timer that, once set via the control signal ST (set timer), will assert the signal TS after a short time has expired (used for timing yellow lights) and TL after a long interval (for green lights). The timer is automatically reset when ST is asserted.



This problem statement has been written in somewhat fuzzy terms on purpose, which is how real-world problems are often described. Some parts of this specification might be missing. We expect you to fill in those parts by making reasonable assumptions. STATE ALL YOUR ASSUMPTIONS. You may not introduce any assumptions that

contradict the problem statement above. You should also use the variables defined in the problem statement (although you may add new variables if they are needed).

We want to implement this traffic light controller as a **Mealy machine**.

(a) **(5 points)** Understand the problem specification. Clearly define the input and output signals that you use and give a description in English, of what each signal does or achieves.

(b) **(5 points)** Define the states that you will use and give a description of each state. Try to use the minimum number of states that you can (although no formal state minimization is required for this problem.)

(c) **(10 points)** Draw the state diagram.

(d) **(10 points)** Draw the state table.

(e) **(10 points)** Implement this controller using only D flip-flops. (Draw the schematic.)

(f) **(20 points)** Write a Verilog module to implement this Mealy machine in **behavioral Verilog**.

(Notes: In practice, we write the behavior of the machine like this in Verilog and rely on synthesis tools to synthesize a circuit which will automatically produce a circuit such as in part (e). Since this example is simple, we can do the D flip-flop implementation by hand as in Part (e) and see both approaches. It is illustrative to compare the code in Part (f) with the implementation in Part (e).)

Problem 25:

Reduce the following state table to a minimum number of states.

(a) **(10 points)** First, use row matching to reduce as far as you can.

(b) **(10 points)** Then, use the implication chart method on the output of the row matching procedure and perform any further reductions.

Present State	Next State		Present Output	
	X=0	X=1	X=0	X=1
A	A	E	1	0
B	C	F	0	0
C	B	H	0	0
D	E	F	0	0
E	D	A	0	1
F	B	F	1	1
G	D	H	0	1
H	H	G	1	0

(c) **(15 points)** You are given two identical sequential circuits which realize the preceding state table. One circuit is initially in state B and the other circuit is initially in state G. Specify an input sequence of length three which could be used to distinguish between the two circuits and give the corresponding output sequence from each circuit.

Problem 26 (20 points):

Reduce the following state table to a minimum number of states by first using row matching and then using the implication chart method.

Present State	Next State X = 0 1	Present Output (Z)
A	E E	1
B	C E	1
C	I H	0
D	H A	1
E	I F	0
F	E G	0
G	H B	1
H	C D	0
I	F B	1

Problem 27:

Circuit N and M have the state tables that follow.

State Table of Circuit M

Present State	X = 0 X=1	Output
S0	S3 S1	0
S1	S0 S1	0
S2	S0 S2	1
S3	S0 S3	1

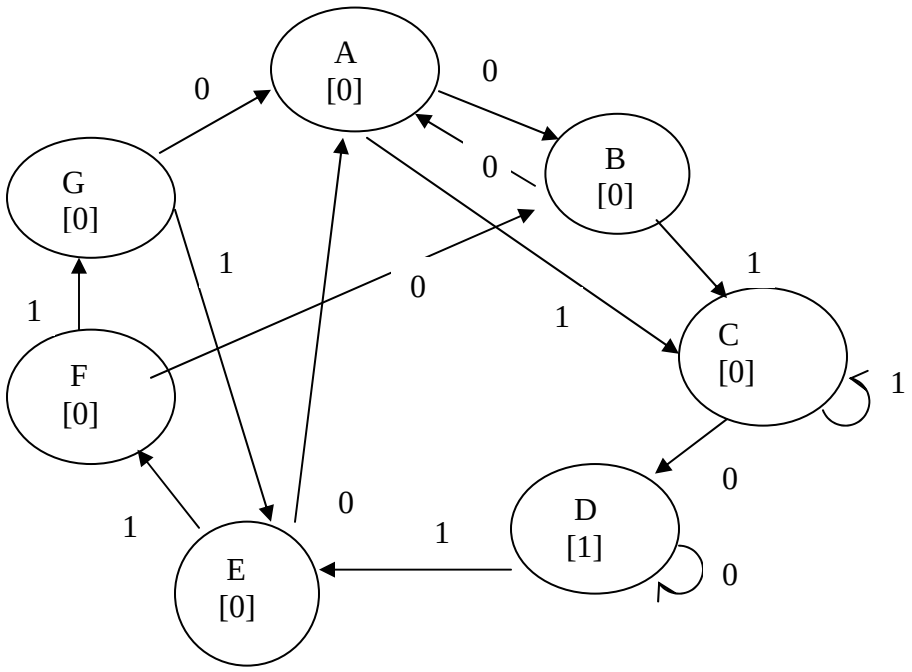
State Table of Circuit N

Present State	X = 0 X=1	Output
A	E A	1
B	F B	1
C	E D	0
D	E C	0
E	B D	0
F	B C	0

(a) First, without first reducing the tables, try to determine whether circuits N and M are equivalent. (This part requires no write-up. It is there to develop your intuition.)

(b) **(20 points)** Then, reduce each table to a minimum number of states, and then decide whether N is equivalent to M by inspecting the reduced tables.

Problem 28 (20 points):
Given the state diagram in the figure below, obtain an equivalent reduced state diagram containing a minimum number of states **by using an implication chart**.
Put your final answer in the form of a **state diagram** rather than a state table.
Make it clear which states have been combined.



Problem 29:
In class, we showed how to implement a 16-bit carry lookahead adder using 4-bit adders and a 4-bit carry lookahead unit. Using the constructed 16-bit carry lookahead adders as building blocks, show how to construct 32- and 64-bit adders with carry lookahead.

(a) **(20 points)** Draw block diagrams for the 32-bit and 64-bit adders, showing all interconnections.
(b) **(30 points)** Analyze the worst-case gate delays encountered in 32- and 64-bit addition. Use simple gate delay model and assume a unit delay (i.e. a delay of 1) for each gate regardless of the fan-in of that gate.

Problem 30 (35 points):
The "Carry Select Adder" is an adder organization that introduces redundant hardware to make the carry calculations go even faster. The figure below illustrates the concept by showing the organization of an 8-bit carry select adder. The 8-bit adder is split in half. The upper half (i.e. bits [7:4]) is implemented by two independent 4-bit adders, one whose carry-in is hardwired to 0 (“adder low”), and another whose carry-in is hardwired to 1 (“adder high”). These two independent adders correspond to two different possibilities: "Adder low" computes what the sum of the upper bits [7:4] would be if a carry-in of 0 were received from the adder for bits [3:0]. In contrast, "adder high" computes what the sum would be if a carry-in of 1 were received from the adder for the bits [3:0]. Thus, in parallel, these compute two alternative sums for the higher-order bits. The carry-out of the lower-order 4-bit adder controls multiplexers that select between the two alternative sums. (Note that for simplicity, the input bits coming from the two 8-bits inputs are not shown in this diagram.)

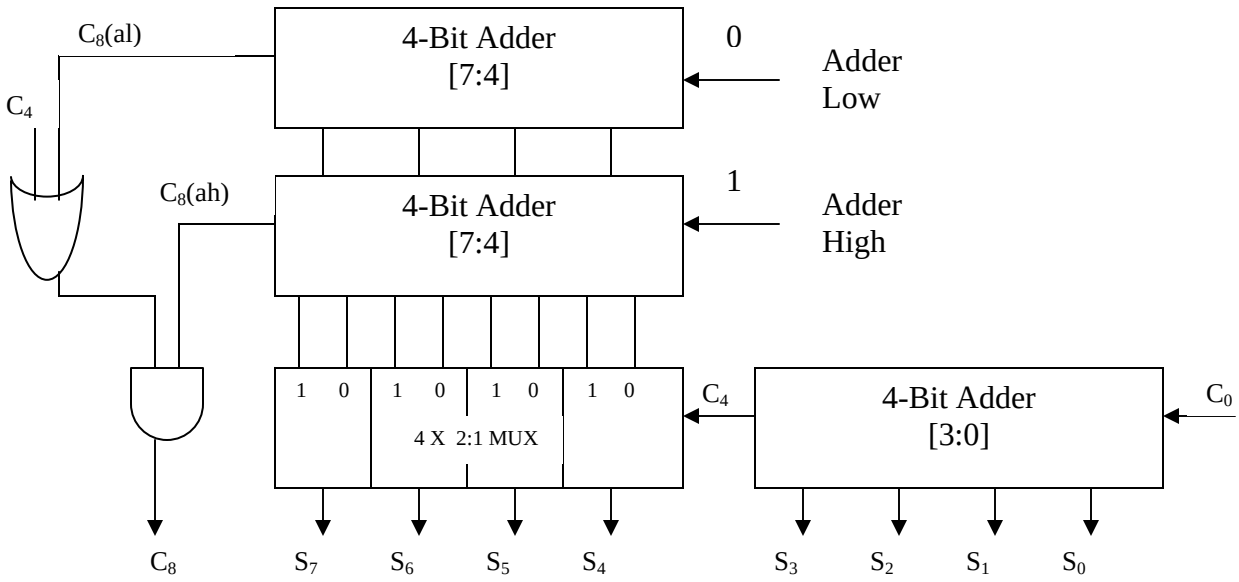


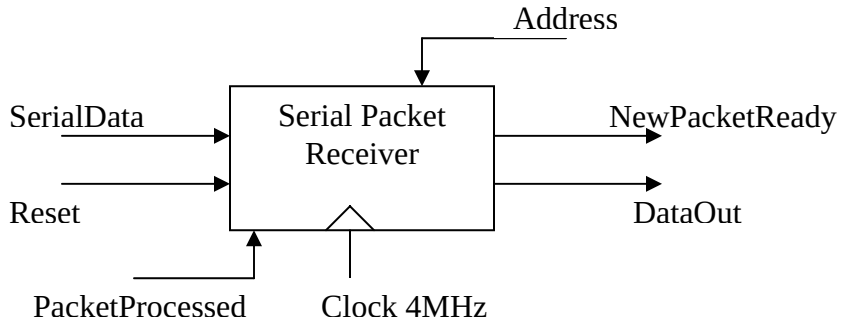
Figure: 8-bit carry select adder

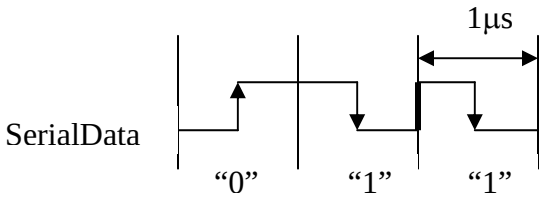
The circuit for C_8 could be a 2:1 multiplexer, but a simpler circuit that reduces the gate count also does the job. In the figure, C_8 is selected from adder high when C_4 is 1 and is the AND of the carries of adder low and adder high when C_4 is 0. Adder low can never generate a carry when adder high does not. So either their carry-outs are the same or adder high's carry-out is 1 while adder low's carry-out is 0. By ANDing the two carries, we obtain the correct carry when C_4 is 0.

Assuming internal carry lookahead logic is used, the 4-bit adders in the above figure take four gate delays to compute their sums and three gate delays to compute the stage carry-out. The 2:1 multiplexers add two further gate delays to the path of the high-order sum bits. Thus the 8-bit sum is valid after only six gate delays. This saves one gate delay over the standard two-level carry lookahead implementation for an 8-bit adder.

Finally, we come to the problem: Using what you learned above, consider a 16-bit adder implemented with the carry select technique described above. The adder is implemented with three 8-bit carry lookahead adders and eight 2:1 multiplexers. Estimate the gate delay and compare it against a conventional 16-bit ripple adder and a 16-bit carry lookahead adder.

Problem 31: In this problem, you will design a serial packet receiver (SPR) that will be used as a part of a router in communication networks. The serial packet receiver receives packets of data bits through a serial port and stores each packet in a 64 × 4 RAM. (Each packet is 256 bits.)





After a packet has been received (and thus the RAM is full), the system asserts a signal called NewPacketReady. Then, the rest of the router will need to read from the RAM (to carry out functions such as IP header look-up for that particular packet.) We will not model the rest of the router, but you need to design the data path in the serial packet receiver so that the serial data can write the packet into the RAM, and the rest of the router can read the bits from the RAM. Once the rest of the router is done with its

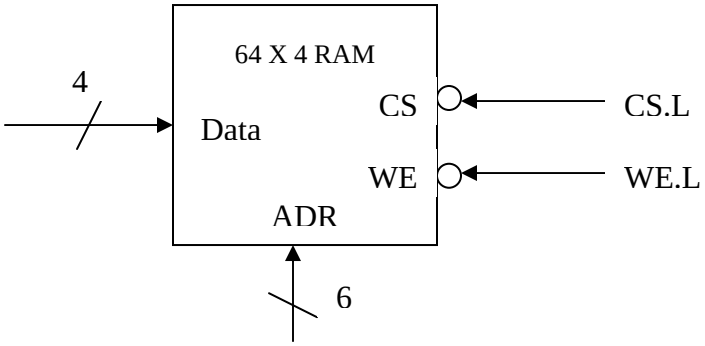
operations on that particular packet, it will assert a signal called PacketProcessed. Upon receiving this signal, the serial packet receiver reads in the next packet from the SerialDataPort.

The SerialData port provides input bits to the serial data receiver after the rest of the router has finished its read operation, and the validity of the serial data input is handled externally. (That is, you don't need to worry about the fact that new packets may be coming in as the RAM is being accessed for a read operation by the rest of the router. The external circuitry assures that when the SPR is ready to take in the next packet, the SerialData port will start providing the input bits of the next packet.)

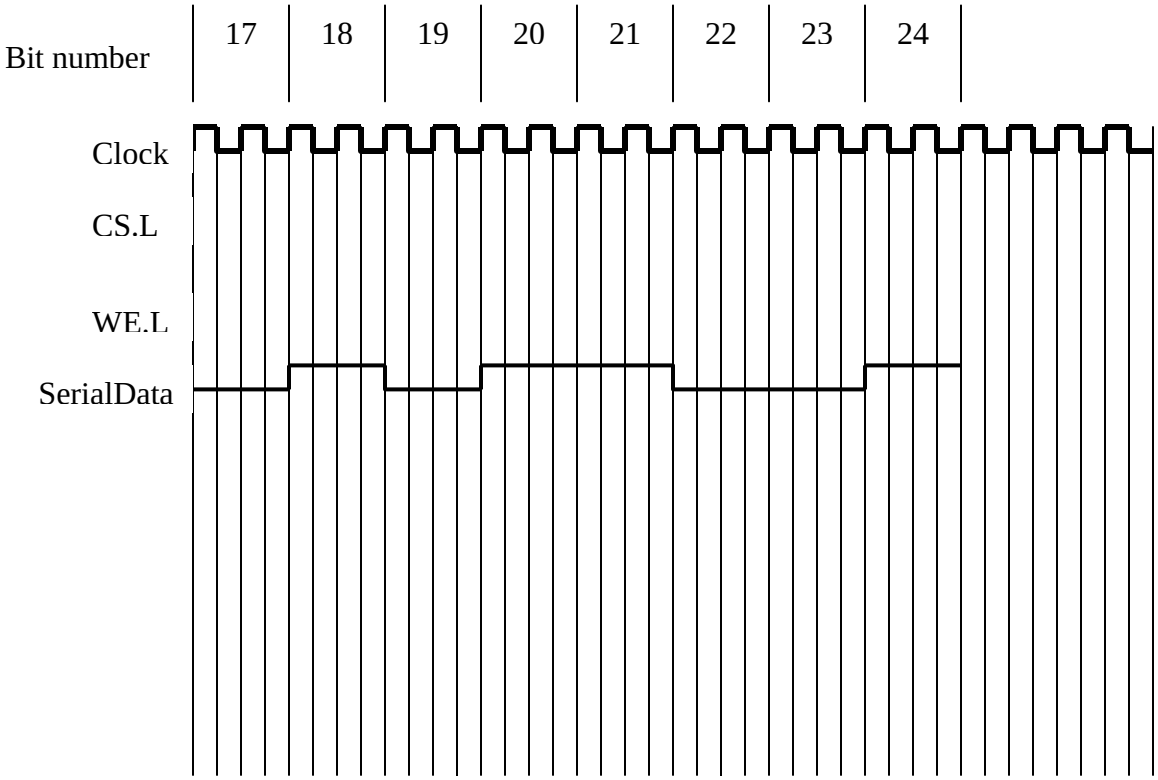
In a packet of 256 bits, there are 128 "information bits": Each information bit has been encoded into two bits: An information bit "1" has been encoded into 10 and "0" has been encoded into 01. Each of 10 and 01 are called codewords. The codewords are presented by the SerialData port beginning with the least significant bit at a rate of 1 MHz. The internal clock of the SPR is 4 MHz.

(30 points) (a) Draw an ASM chart for the SPR. Carefully define each operation that you use. At this point, the ASM chart does not need to define the control signals and may specify operations instead. However, after you do part (b) below, come back to this ASM chart and carefully define and specify the control signals asserted instead of the operations. (See Chapter 10.1-10.2 in your textbook for an example.)

(30 points) (b) Draw a detailed datapath for the SPR. Your block diagram should be at the level of building blocks such as MUXes, registers, counters, shift registers, decoders, etc. You must clearly define what each of your signals corresponds to, both on the interface schematics you use and the signals that go into these terminals.
Use the RAM provided below in drawing your block diagram. On this RAM interface schematic, ADR corresponds to the address, and note that CS.L and WE.L are both active low signals. The terminal CS stands for "chip select", and WE stands for "write enable".



(30 points) (c) Assume that the SPR has already received the first 16 bits of a 256-bit packet. Draw a timing diagram which shows ALL the relevant control signals necessary for writing in the next 8 bits of the 256-bit packet, and updating the RAM address. Assume that the control signals are generated by a Moore-type finite state machine running at 4 MHz. Show as many signals as necessary.



Problem 34: Using **behavioral Verilog**, write a separate module for each of the following. (Your modules may NOT instantiate each other. In addition, you MUST use behavioral Verilog which describes the behavior of these modules: Gate-level specifications are not allowed.)

(10 points) (a) D latch with **asynchronous** reset.
(10 points) (b) D flip-flop with **asynchronous** reset.
(10 points) (c) D flip-flop with **synchronous** reset.
(15 points) (d) T flip-flop with **asynchronous** reset.

Problem 35: (25 points)
Write a Verilog module to implement a 4-bit shift register. Your shift register MUST instantiate 4 copies of the D flip-flop with synchronous reset that you implemented in Problem 34(c). The shift register can only shift to the right, and has only a Serial Input capability (i.e. no parallel load).