Name: SOLUTIONS
Perm#:

ECE 152A-Winter 2004
Prof. Volkan Rodoplu

FINAL EXAMINATION

Write your name and Perm# above.

DO NOT TURN TO THE OTHER PAGES OF THIS EXAM UNTIL YOU ARE INSTRUCTED TO START.

INSTRUCTIONS:

PARTIAL CREDIT will be given only to true statements that make progress towards the correct answer. Partial credit may be given to correct reasoning in developing structures such as K-maps and truth tables in which the variables are clearly labeled.

NO partial credit will be given for incorrect statements or statements to which no truth value can be assigned (such as a bunch of numbers or algebraic expressions).

NO partial credit will be given for statements that use symbols that the problem statement or you have not defined.

NO credit will be given for any work that is not clearly labeled with the part and problem number to which this work provides an answer. (If you are using the backs of any of the pages, make sure that your answers are clearly labeled.)

PROBLEMS:

Problem 1: Finite State Machines and Verilog

A finite state machine (FSM) takes a serial bit stream as input and outputs a serial bit stream. The FSM sets the output equal to the input only if the current input bit is the same as the previous input bit. Otherwise, it sets the output bit to the previous output bit.

For proper operation, the machine is initialized by feeding in two consecutive 1's in the input sequence. That is, at times t = 0 and t = 1, the input is always 1. This sets the output of the machine to 1 at t = 1. **In Parts (a) and (b) below, your job is to model the behavior of this machine from t = 1 onwards, assuming that the machine has been initialized as described above.**

The following is an example of the input and output bit streams for this FSM:

time	0	1	2	3	4	5	6	7	8	9	10	11
Input	1	1	0	1	0	0	1	0	0	1	1	1
Output	x	1	1	1	1	0	0	0	0	0	1	1

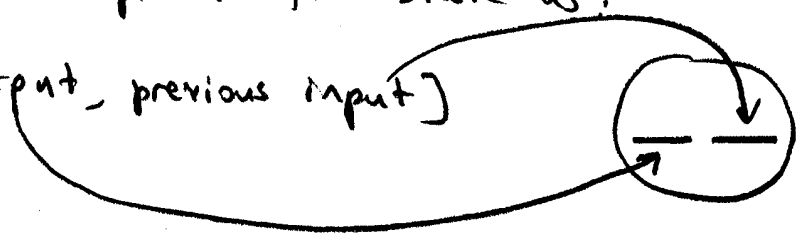
Note how the machine is first initialized by feeding in 1's at t = 0 and t = 1. (In the above diagram, x denotes "unknown".)

After initialization has been completed, we proceed as follows in the above example: At t = 2, the current input is 0 and the previous input is 1. These don't match, so the machine holds the previous value of output (which is 1). Note how the machine holds this value until t = 5 since the current input does not match the previous input bit until t = 5. At t = 5, the current and previous input bits are both 0; therefore, the output is set to the current (t = 5) input bit, which is 0.

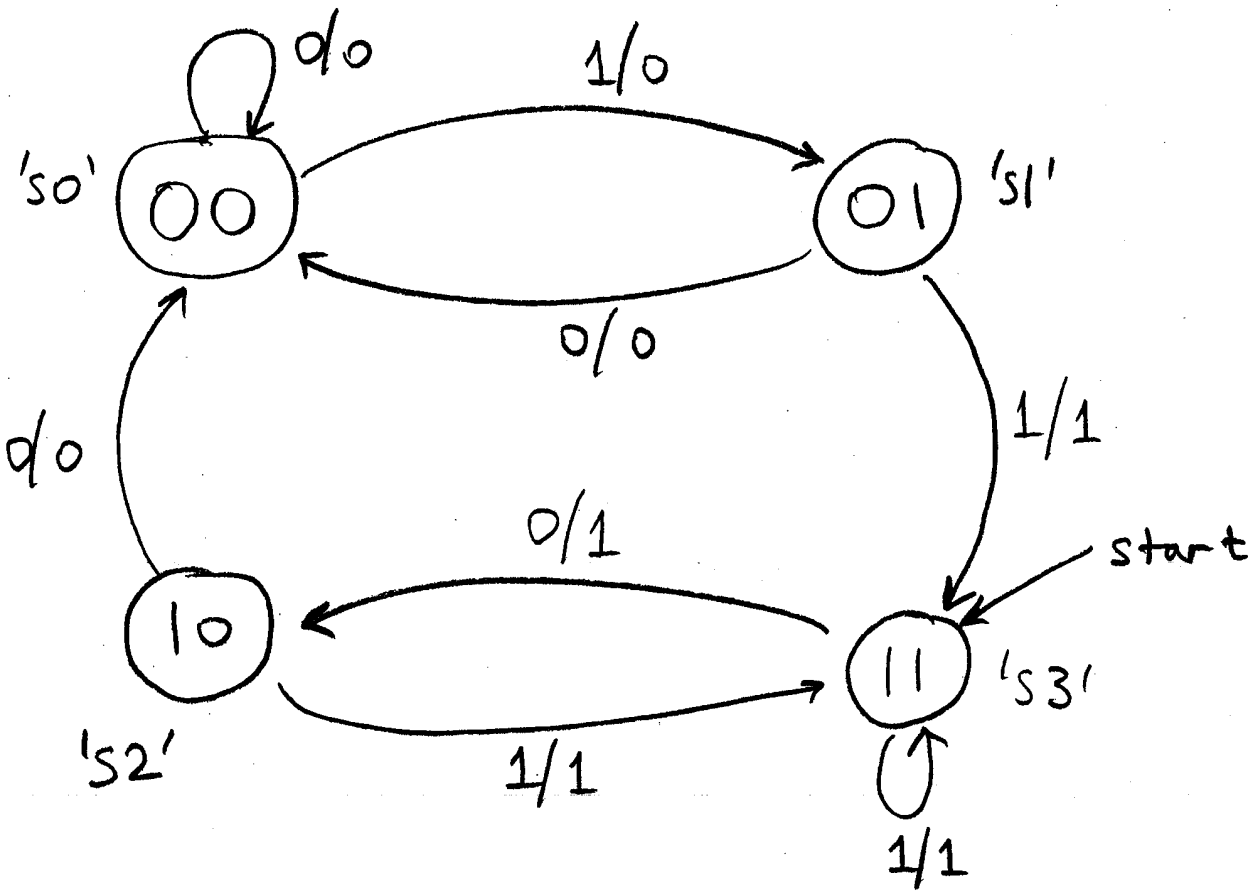
(25 points) a.) Draw a **Mealy** state transition diagram for this FSM. Make sure that you show the "start" state and tell us what the meaning of each state is.

Since we need to remember both the previous output and the previous input, we need 2 state bits. I will represent the state as:

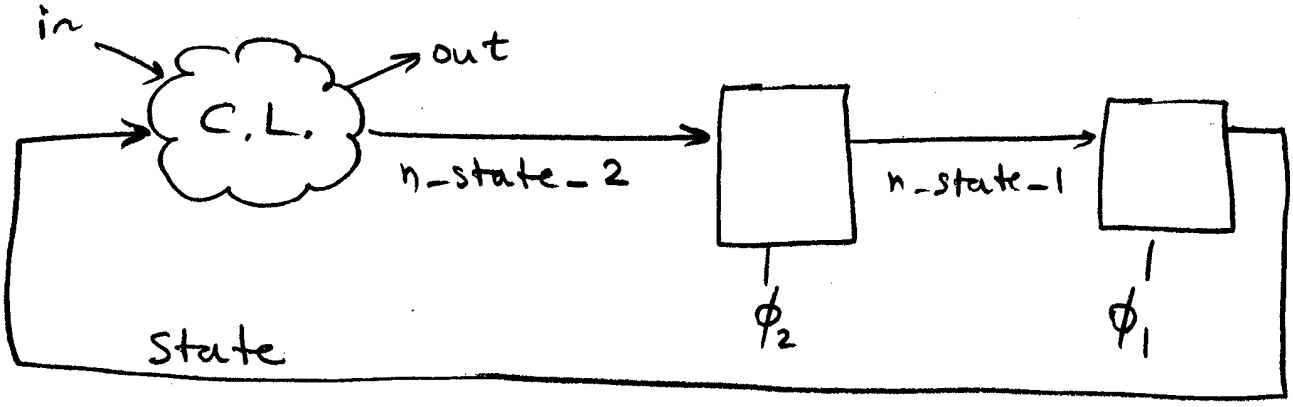
[previous output_ previous input]



(You may continue your scratch work or answer for Problem 1, Part (a) on this page.)



(25 points) b.) Implement this **Mealy** machine in Verilog using LATCHES and 2-PHASE CLOCKING. Make sure to give the entire Verilog code for the module, not just a fragment. Please write down Verilog comments if any of your code needs any explanation. If you want us to understand your code better, please make a rough sketch of the hardware that you would like to synthesize with this code. That is, it would be good to see where you plan to put your phi1 and phi2 latches and where you plan to put your combinational logic blocks.




```
module FSM (phi1, phi2, in, out);
    input phi1, phi2, in;
    output out;
    reg [1:0] state;
    reg [1:0] n_state_1;

    parameter s0 = 2'b00;
    parameter s1 = 2'b01;
    parameter s2 = 2'b10;
    parameter s3 = 2'b11;
```

(You may continue your answer for Problem 1, Part (b) on this page.)

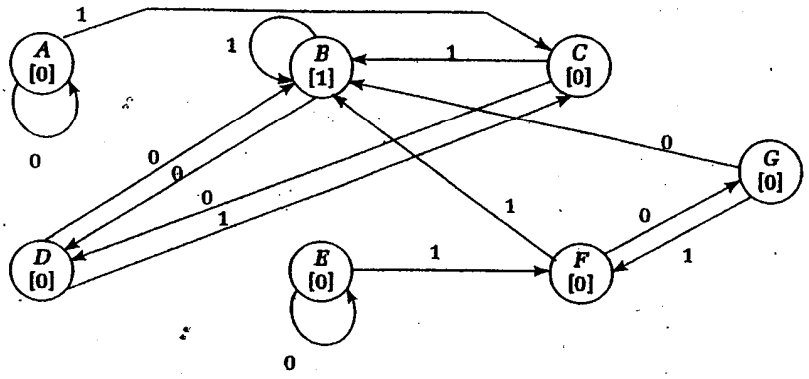
```
assign out = (state[0] ^ in) ? state[1] : in;

always @ (phi1 or n_state_1)
    if (phi1) state <= n_state_1;

always @ (phi2 or in or state)
    if (phi2)
        begin
            case (state)
                s0 :
                    n_state_1 <= in ? s1 : s0;
                s1 :
                    n_state_1 <= in ? s3 : s0;
                s2 :
                    n_state_1 <= in ? s3 : s0;
                s3 :
                    n_state_1 <= in ? s3 : s2;
            endcase
        end

endmodule
```

Problem 2: State Reduction

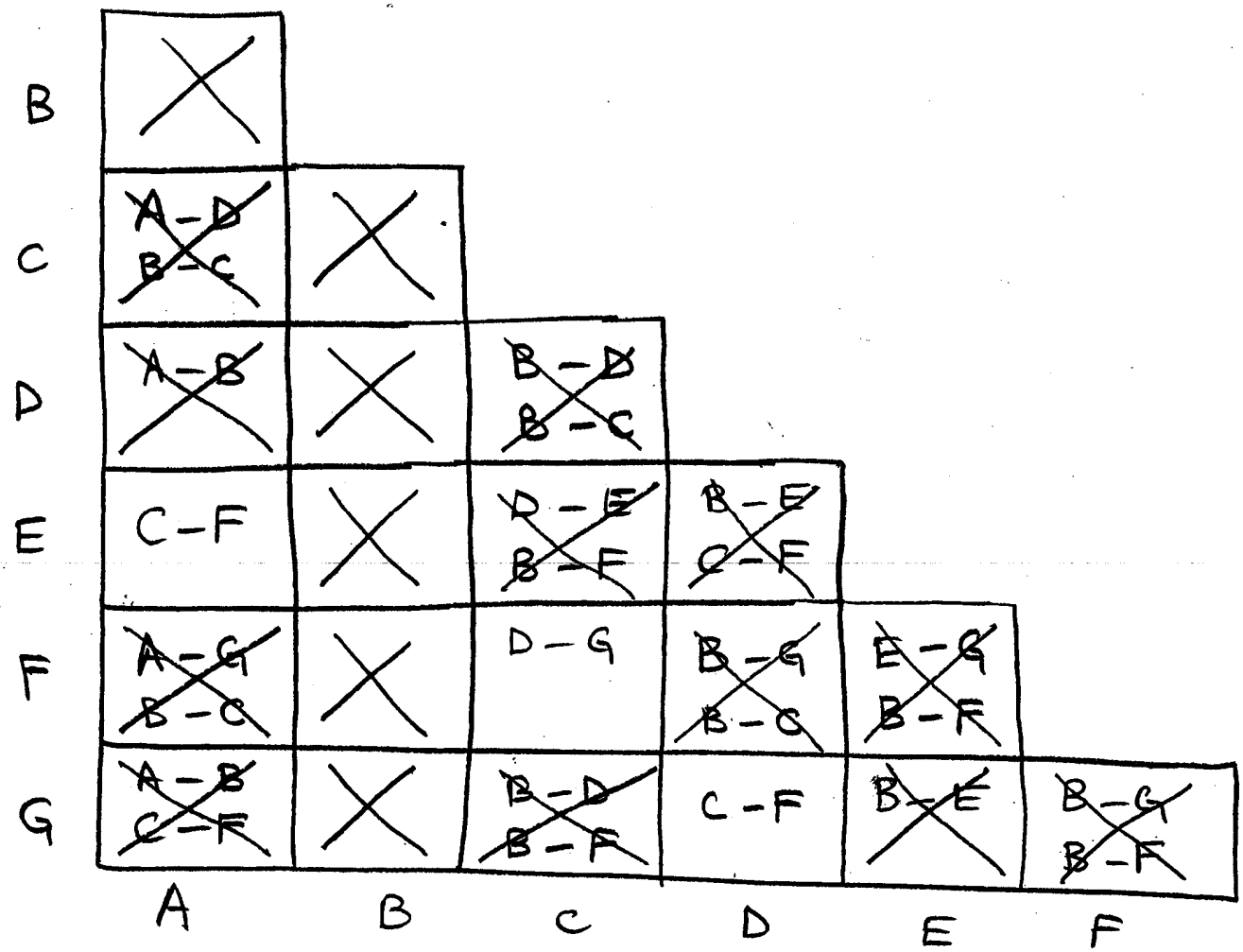


(10 points) a.) Given the state diagram in the figure, use the implication chart method to minimize the number of states. (Do not use row matching.) Which states are equivalent?

State Table:

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

(You may continue your answer for Problem 2, Part (a) on this page.)

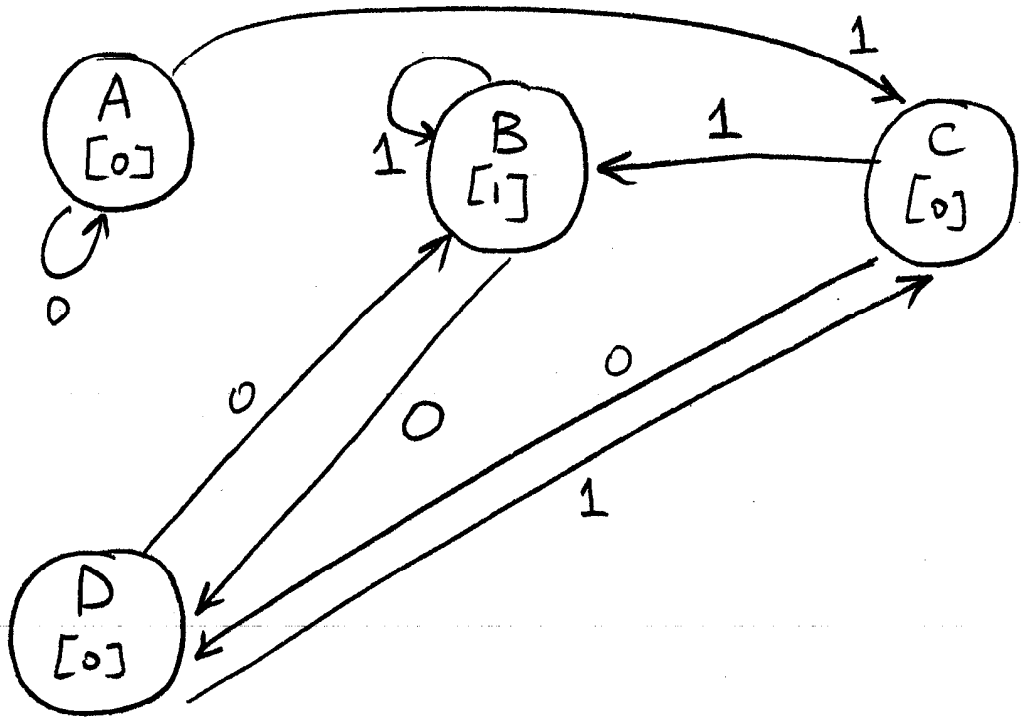


To determine the equivalent states, we read the column-row coordinates of the squares without the X's (not by reading the implication pairs in the squares.)

$A \equiv E$, $C \equiv F$, $D \equiv G$

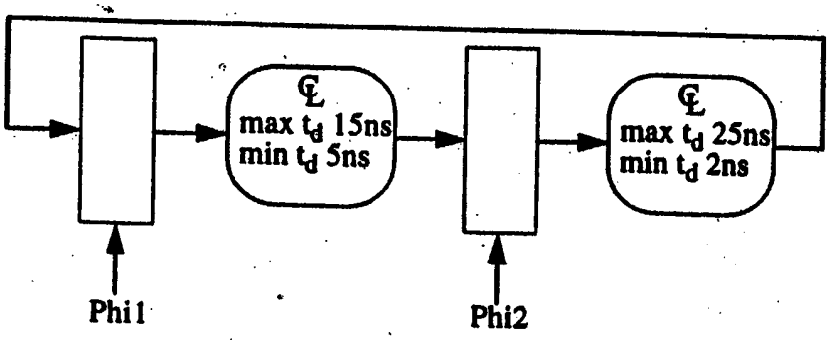
(10 points) b.) Draw the state diagram with the minimum number of states.

We eliminate states E, F, G.
Remaining states are: A, B, C, D.



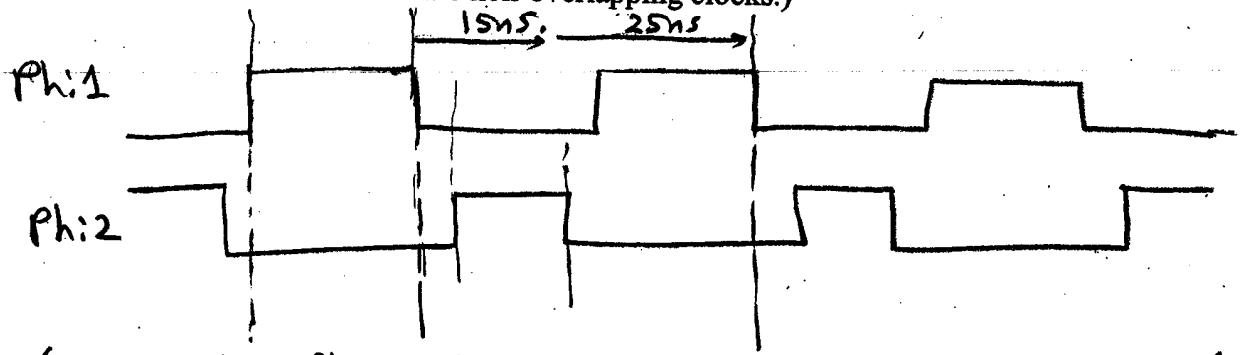
(The start state is not noted in the original diagram. However, after state reduction, the state arrow can be moved to any of the equivalent states. For example, if E is the start state, then we place the start arrow on A in the above reduced state diagram.)

Problem 3: Timing of Sequential Circuits with Latches
(In each part of this problem, you must show your precise reasoning with timing diagrams to get any credit.)



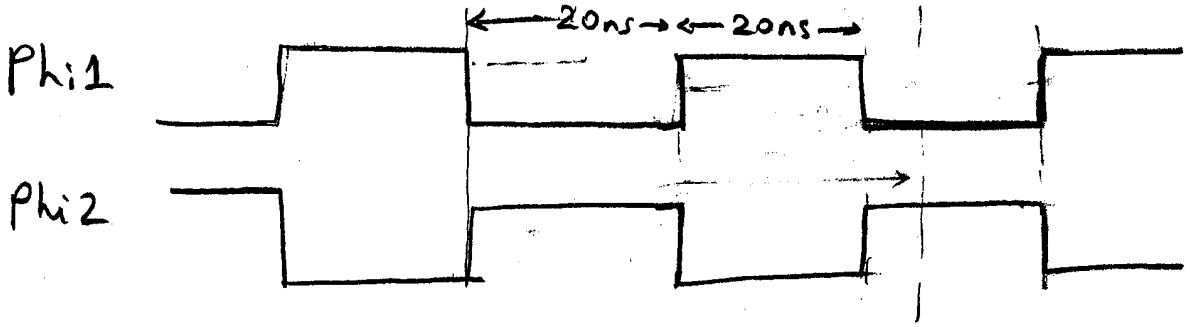
In the 2-phase clocking system shown, assume that there is no clock skew and there is no delay through each latch. Assume ZERO setup and hold times for both latches.

(10 points) a.) What is the minimum clock cycle time such that the circuit will still work? (Note that Phi1 and Phi2 are non-overlapping clocks.)



(Note that Phi1 and Phi2 must have the same period to remain in phase and non-overlapping.) Consider the signal that arrives just before Latch 1 (the latch for phi1) turns off. This signal goes through the maximum delays 15ns + 25ns and must arrive at Latch 1 again before Latch 1 turns off in the next clock cycle. A similar analysis applies to Latch 2. Hence, the minimum clock period is **40ns**.

(10 points) b.) In this part of the problem, we assume that Phi1 and Phi2 are inverted clocks (i.e. Phi2 is obtained by inverting Phi1) and each clock is high for half of the clock period. (that is, both Phi1 and Phi2 have a 50% duty cycle). Assume that the inverter used to produce Phi2 from Phi1 has zero propagation delay and that there is zero clock skew. Can you achieve the minimum clock cycle time that you computed in Part (a) of this problem by using these inverted clocks with 50% duty cycle?



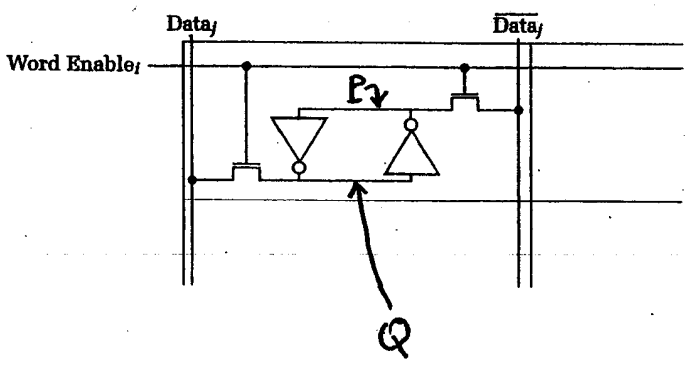
YES. This will work. After $\phi_1 \uparrow$, the signals waiting at the gate of latch 1 travel through the first C_L. Since the maximum delay of first C_L is 15ns, the longest-delay signal waits 5ns at the gate of Latch 2. When Latch 2 turns on, worst signal travels for 20ns in second C_L, then $\phi_2 \downarrow$ and $\phi_1 \uparrow$; that worst signal arrives at Latch 1 5ns after $\phi_1 \uparrow$. In another 15ns, this signal arrives at the entrance to Latch 2 just as $\phi_2 \uparrow$, catching up with the early signals. Hence, the design works, and the minimum clock cycle time of 40ns can be achieved by inverted, 50% duty-cycle clocks.

Problem 4: Memory

(3 points) a.) Is the cache in a computer system implemented in SRAM or DRAM? Why? (You must give a correct justification to get any credit.)

SRAM. SRAM is much faster than DRAM. Because cache is for fast memory access, SRAM is used.

b.) The following figure shows a 6T (6 transistor) SRAM cell. We use active high logic, so a low voltage corresponds to "0" and a high voltage corresponds to "1". Initially, both the bitline and its complement are low; the wordline is low; and this SRAM cell is storing a "1".



(2 points) b.1.) Initially, what are the values of the wires P and Q in the figure?

P = 0, Q = 1 (SRAM stores a "1": this must correspond to the Data line).

(8 points) b.2.) Now, assume that we would like to READ from this SRAM cell. We bring the wordline high. Will we be able to read out the value correctly from this SRAM cell? What is the problem here?

NO, The $\overline{\text{Data}}$ will be fine, but the level of Data will be degraded because N-MOS does not pass "1" (high voltage) well.

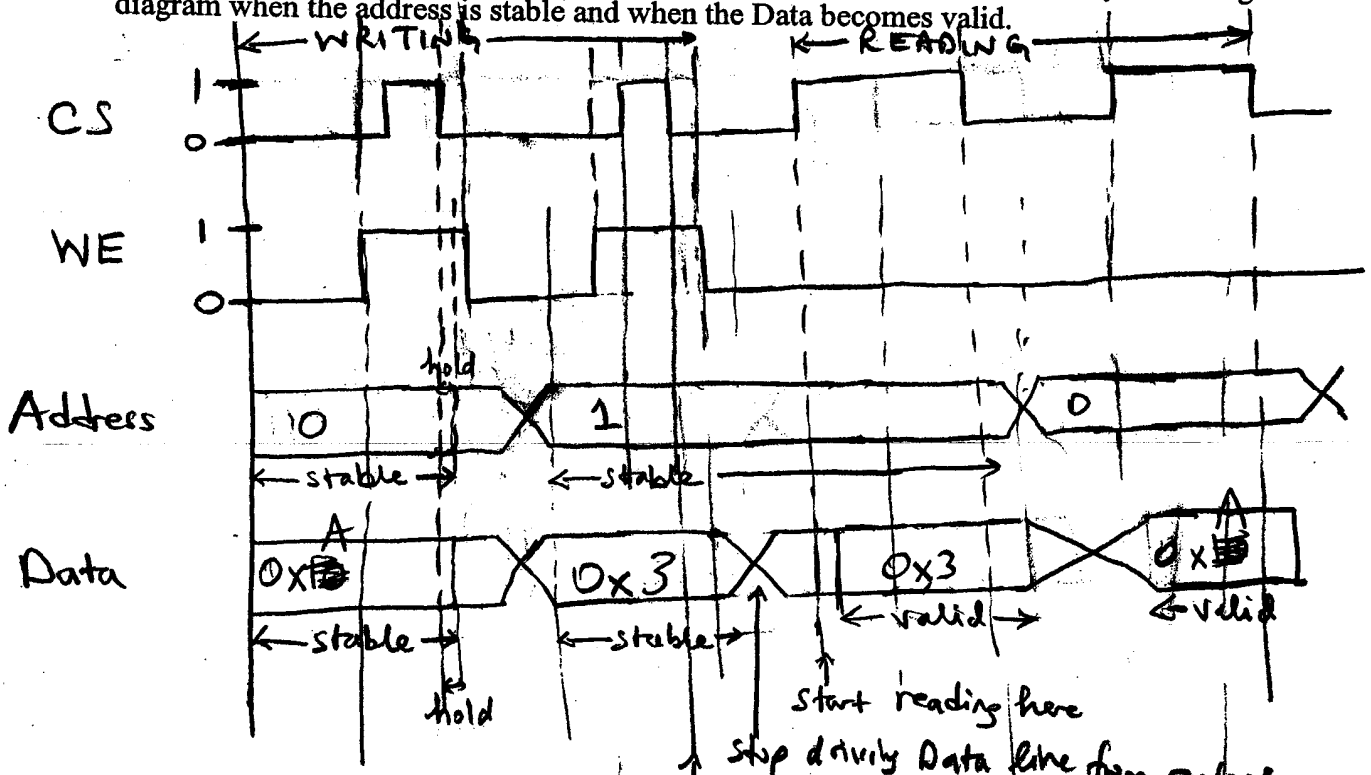
(2 points) b.3.) How can the problem in Part (b.2) be corrected?

Both Data and $\overline{\text{Data}}$ should be precharged to "1". (high). Then, when wordline is asserted, P will be able to pull $\overline{\text{Data}}$ low (enough to be detectable by sense amplifier.)

(25 points) c.) You are given a 1024 × 4 SRAM whose storage matrix has been organized as a square. We will store two 4-bit numbers into this SRAM and subsequently read them out **in reverse order**. (Do NOT alter the order of the bits in each number). The two numbers we want to write into the RAM are 1010, 0011 in this order. Then, we want them out as 0011, 1010.

10 3

Give a timing diagram that shows the values of chip-select (CS), write-enable (WE), Data, and Address signals of the SRAM. For the Data signal, you may use a single line and use hexadecimal numbers. Assume that the first number is written into Address 0 in the RAM. On the address line, indicate the address as a number. (You need to determine how many bits this number should be.) It is very important that you show in your timing diagram when the address is stable and when the Data becomes valid.

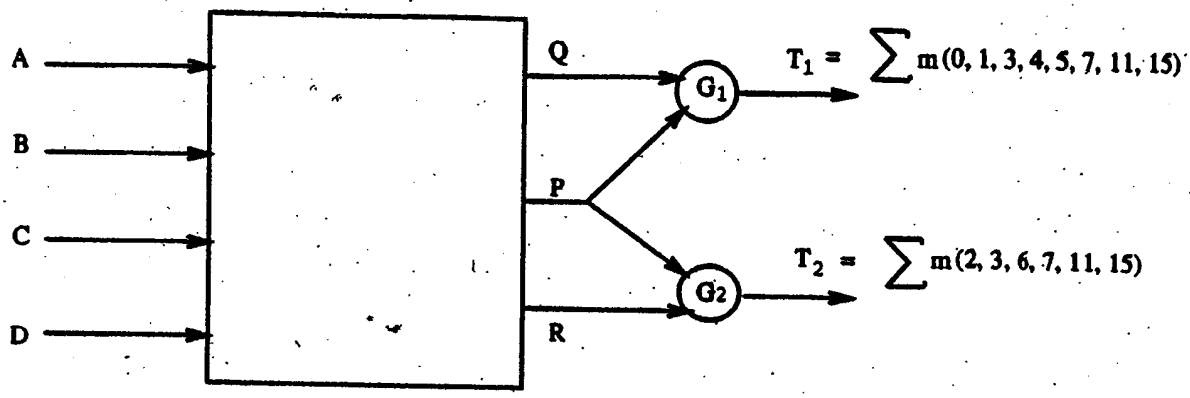


Notes:

1- The "stable" periods (for address & data) are just examples. The absolute requirement for stability is that the address & data be stable a setup time before the last of CS and WE ↑ (CS in this case..)

2- I have kept Address 1 stable since I know that that's where I'll be reading from at beginning of READING cycle..

Problem 5: Karnaugh Maps



You are given a combinational circuit with 4 inputs (A, B, C, D), 3 intermediate outputs (Q, P, R) and two outputs (T_1, T_2) as shown in the figure.

(7 points) a.) Assuming that both G_1 and G_2 are AND gates, show the Karnaugh map for the function P, such that it has the **fewest** ON-set minterms needed to produce both T_1 and T_2. (Recall that ON-set minterms are represented by 1's on the Karnaugh map.)

T_1

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

T_2

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

Both G_1 and G_2 are AND gates.

ON-SET(P) $\supseteq$ ONSET(T_1)

ON-SET(P) $\supseteq$ ONSET(T_2)

$\therefore$ Minimum ON-SET(P) = ONSET(T_1) $\cup$ ONSET(T_2)

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

(6 points) b.) For the choice of P in part (a), show the Karnaugh maps for Q and R. Indicate explicitly the don't care positions. (Recall that a position on the Karnaugh map is a don't care if placing a 1 or a 0 in that position does not change the output. The output is T_1 in the case of Q and it is T_2 in the case of R.)

Q

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

Q must have 1's where T_1 has 1's
Q must have 0's where T_1 has 0's and P has 1's.

R

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

R must have 1's where T_2 has 1's
R must have 0's where T_2 has 0's and P has 1's.

(7 points) c.) If G_1 and G_2 are both OR gates, find the function P, such that it has the **largest** number of ON-set minterms and can be used to implement both T_1 and T_2.

Maximum ON-SET(P) = ON-SET(T_1) $\cap$ ONSET(T_2).

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

(6 points) d.) For the choice of P in part (c), show the maps for Q and R, again indicating the don't care positions.

Q

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

Q must have 0's where T_1 has 0's,
Q must have 1's where T_1 has 1's and P has 0's.

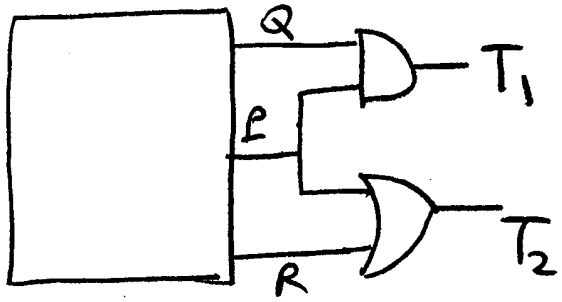
R

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

R must have 0's where T_2 has 0's
R must have 1's where T_2 has 1's and P has 0's.

(7 points) e.) Can both T_1 and T_2 be produced if G_1 is an AND gate and G_2 is an OR gate? (You must give an explanation or a proof to get any credit.)

No



For such an implementation

ONSET(P) $\supseteq$ ONSET(T_1)

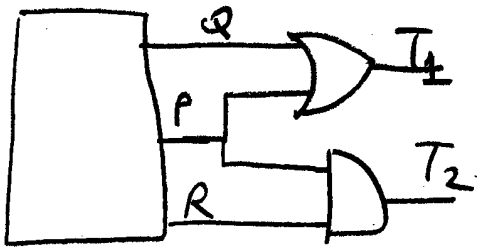
ONSET(P) $\subseteq$ ONSET(T_2)

$\Rightarrow$ ONSET(T_2) $\supseteq$ ONSET(T_1), which is FALSE.

(simply look at the minterms of T_1 and T_2 given in the original diagram.)

(7 points) f.) Can both T_1 and T_2 be produced if G_1 is an OR gate and G_2 is an AND gate? (You must give an explanation or a proof to get any credit.)

No



For such an implementation,

$ONSET(P) \subseteq ONSET(T_1)$

$ONSET(P) \supseteq ONSET(T_2)$

$\Rightarrow ONSET(T_2) \subseteq ONSET(T_1)$, which is FALSE.