

Comp 120 Computer Organization
Spring 2005

Solutions to Problem Set #3

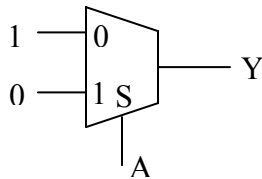
Problem 1. "MUX Madness"

(A) $Y = (I_0 \bar{A} + I_1 A) \bar{B} + (I_2 \bar{A} + I_3 A) B = I_0 \bar{A} \bar{B} + I_1 A \bar{B} + I_2 \bar{A} B + I_3 A B$

Functions	I_0	I_1	I_2	I_3
AND(A,B)	0	0	0	1
OR(A,B)	0	1	1	1
XOR(A,B)	0	1	1	0
NAND(A,B)	1	1	1	0
NOR(A,B)	1	0	0	0

(B) Yes. From the expression of Y in (A), we can see that Y contains all possible combination of A and B.

(C) An inverter can be constructed as follows:



By connecting one of the two inputs to selector input of a MUX, and using the inverter shown above to complement of the other input it is possible to construct any 2-variable Boolean function using only 2 muxes. By inspecting the truth table of any 2-input function, one can determine whether each MUX input should be connected to 0, 1, the second input, or the complement of the second input. For the following truth table, if we connect A to the selector, we find that "0 input" should be connected to B because when A=0, Y=B; and the "1 input" should be connected to \bar{B} because when A=1, Y= \bar{B} .

Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Layout of MUX

	Value
0	B
1	\bar{B}

Problem 2. "Assign Priorities"

(A)

I ₃	I ₂	I ₁	I ₀	A ₁	A ₀	E
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

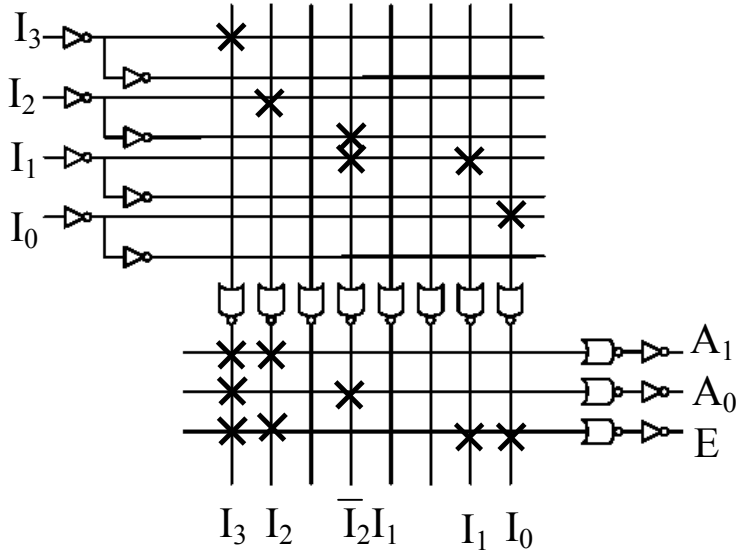
$$A_1 = I_3 + I_2$$

$$A_0 = I_3 + \overline{I_2}I_1$$

$$E = I_3 + I_2 + I_1 + I_0$$

There are 16 words, and 3 bits per word, for a total of $16 \cdot 3 = 48$ bits.

(B)



(C) From truth table

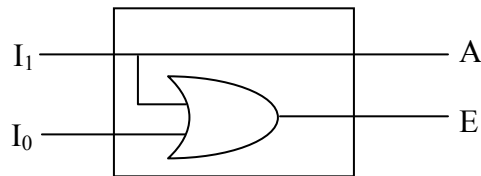
I_1	I_0	A	E
1	X	1	1
0	1	0	1
1	0	0	0

We get

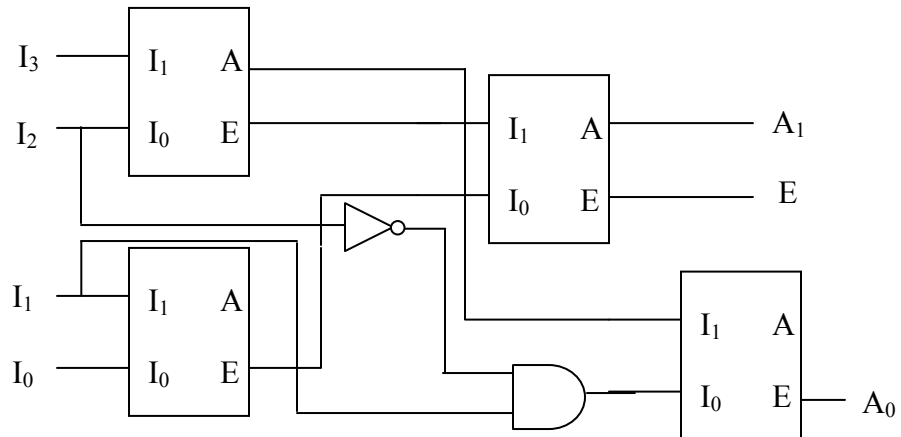
$$A = I_1$$

$$E = I_1 + \bar{I}_1 I_0 = I_1 + I_0$$

Two input priority encoder:



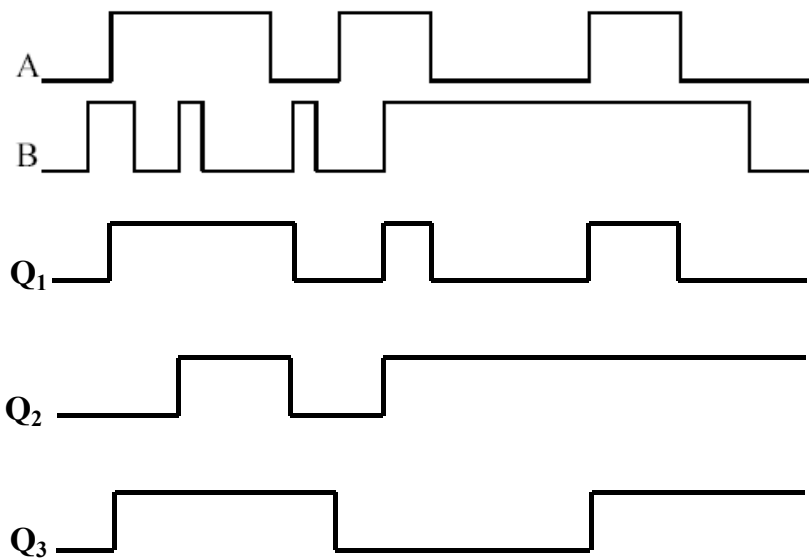
Four-input priority encoder:



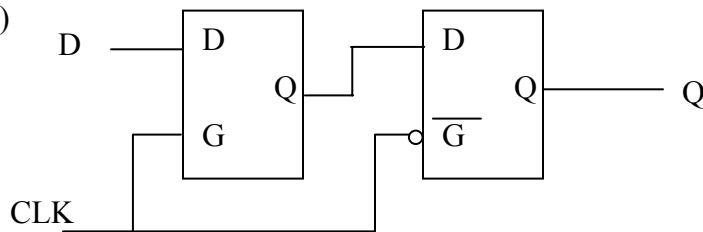
(D) ROM: easy to design, but a waste of gates. The PLA uses fewer gates, but requires logic reduction to realize a design smaller than a ROM. The discrete logic, using gates, approach uses the fewest components and probably have the smallest propagation delay, although it is the most difficult of the 3 approaches.

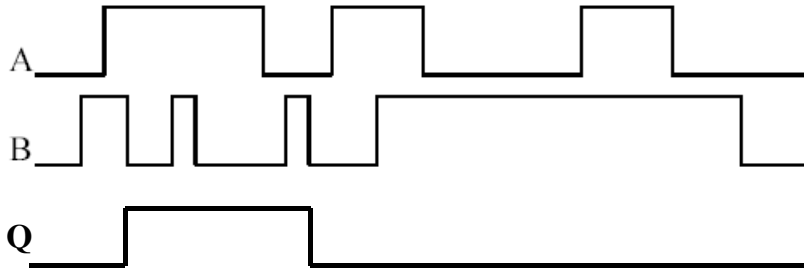
Problem 3. "Getting Faint Memories to Register"

(A)



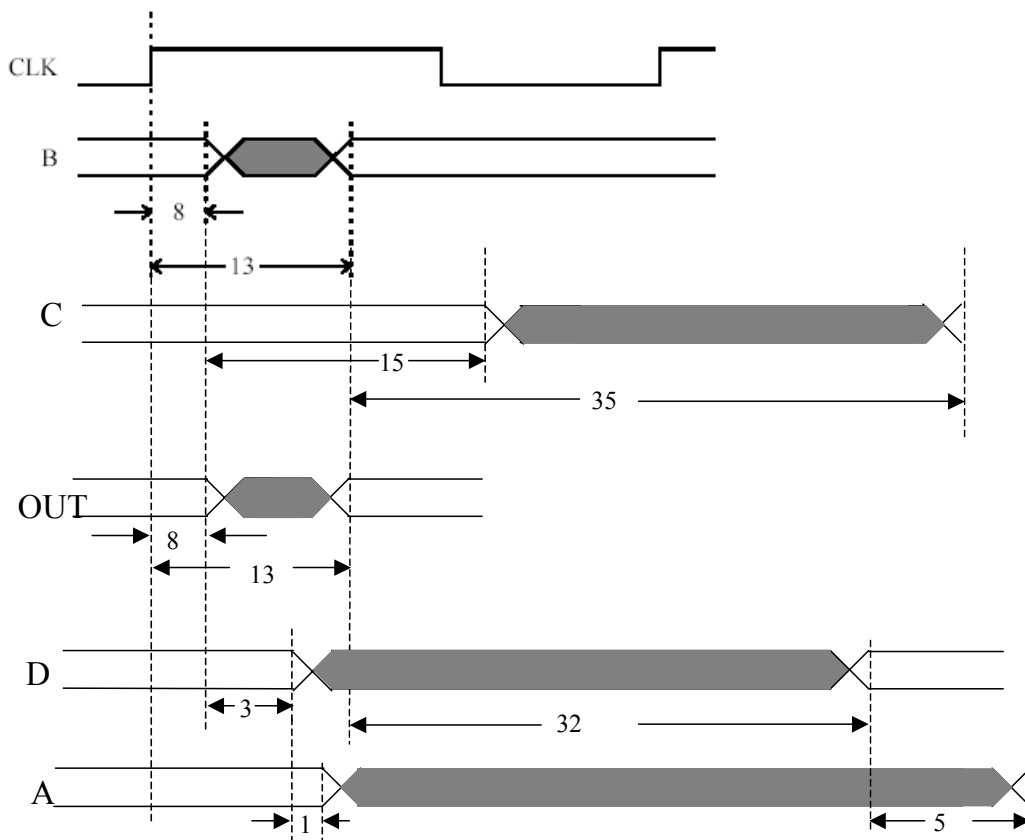
(B)





Problem 4 “Synchronous Circuit Circus”

(A)



(B) From above graph we find that A The CLK must guarantee that A need longest time to reach stable output. After A reach stable status, it still needs to satisfy R_1 's setup time that is 7 ns. So the smallest cycle for A is $13+32+5+7 = 57ns$. And the maximum CLK frequency is

$$(1/57)*10^9=1.75*10^7\text{Hz}$$

(C) Now, A's shortest cycle becomes $13+25+5+7 = 50ns$, while C's shortest cycle is $13+35+7 = 55ns$. Hence, the maximum CLK frequency is decided by path $R_1 \rightarrow CL_2$, which is $(1/55)*10^9=1.81*10^7\text{Hz}$.

(D) In order to meet the setup and hold times of the register, the IN signal must arrive early enough to make sure that A is stable by the setup time before the clock edge, and must be held long enough to make sure that A is stable until the hold time after the clock edge. Thus, IN must be stable and valid:

$$t_{pd(AND)} + t_{s(R1)} = 5ns + 7ns = 12ns \text{ before the clock edge, and}$$

$$t_{hold(R1)} - t_{cd(AND)} = 2ns - 1ns = 1ns \text{ after the clock edge.}$$