CS2100 Computer Organisation

Tutorial #8: MSI Components

(Week 10: 25 – 29 March 2024)

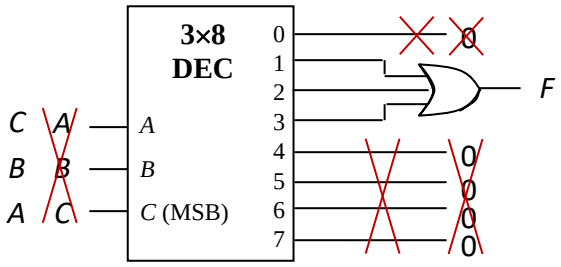
Answers

Discussion Questions:

D1. Given this Boolean function:

$$F(A,B,C) = \Sigma\, m(1, 2, 3)$$

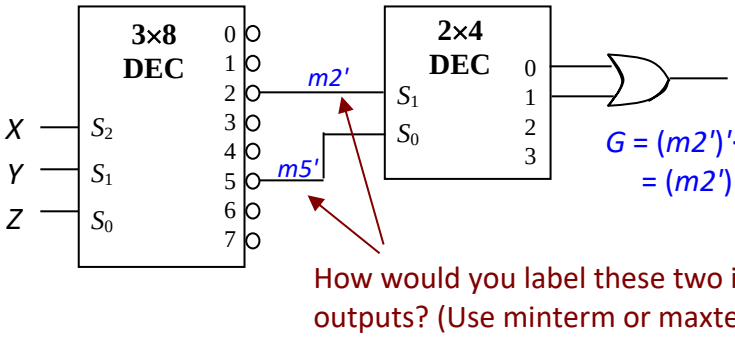
We want to implement this function using a **3×8 decoder with normal outputs** as shown below. Point out the mistakes in the solution below.



Answers:

- Inputs *A*, *B*, *C* should be connected to selector inputs *C*, *B*, *A* respectively.
- Unused outputs of the decoder should not be labelled as 0. Leave them unused.

D2. Given the following circuit comprising a **3×8 decoder with negated outputs** and a **2×4 decoder with normal outputs**, what is the Boolean function *G*(*X*,*Y*,*Z*)?

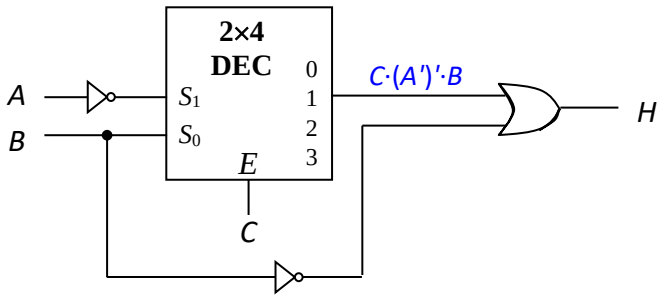


$G = (m2')'\cdot(m5')' + (m2')'\cdot(m5')$
$= (m2')' = m2$

How would you label these two intermediate outputs? (Use minterm or maxterm notation.)

Answer: ***G(X,Y,Z) = X'·Y·Z'***

D3. Given the following circuit comprising a **one-enabled 2×4 decoder with normal outputs**, what is the simplified SOP expression of Boolean function *H(A,B,C)*?



Answer: ***H(A,B,C) = B' + C·A·B = B' + A·C***

Tutorial Questions:

(Make sure you have done the above discussion questions.)

1. Realize the following function with (a) **an 8:1 multiplexer**, and (b) **a 4:1 multiplexer** using the first 2 input variables as the selector inputs.

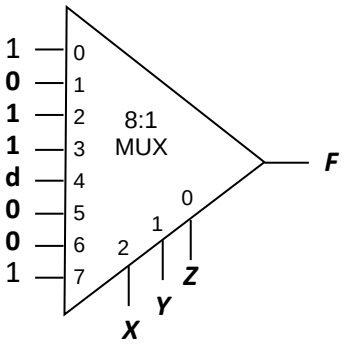
$$F(X, Y, Z) = \Pi M(1, 5, 6) \cdot D(4)$$

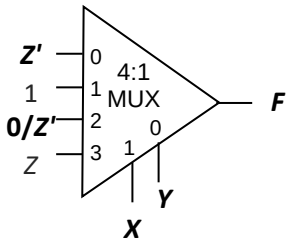
You may write complemented variables instead of drawing an inverter to derive it. If you have several choices for your answer, choose the simplest one (constant logic values 0 and 1 are simpler than literals). You may write "x" or "d" for "don't-care" values.

What if we use the last 2 input variables as the selector inputs instead for the 4:1 multiplexer?

Answers:

F is a 3-variable function, so there are $2^3 = 8$ rows in its truth table. Using an 8:1 multiplexer, we do not need to collapse any input; we just copy the values of *F* directly to the multiplexer inputs:





X	Y	Z	F	F
0	0	0	1	**Z'**
0	0	1	0	
0	1	0	1	**1**
0	1	1	1	
1	0	0	d	**0 / Z'**
1	0	1	0	
1	1	0	0	**Z**
1	1	1	1	

To use a 4:1 multiplexer, we need to collapse the 8-row truth table to a 4-row table of multiplexer inputs (see table above). For simplicity sake, we choose the most significant variables as our selector lines (unless the instruction says otherwise), and the least significant variable as the input variable into the multiplexer.

Given that the maxterms are *M*1, *M*5 and *M*6 (where *F* is false), we put "0" in the inputs 1, 5 and 6 of the 8:1 multiplexer. How do we deal with the don't-care at *M*4 (or *m*4)? Since it is a don't-care input, we can put a **"0"** or **"1"** or **"d"** (I'm avoiding "X" because "X" happens to be one of the three variables). Since we are going to collapse inputs, we put a "d" to preserve its don't-care state. The collapse works

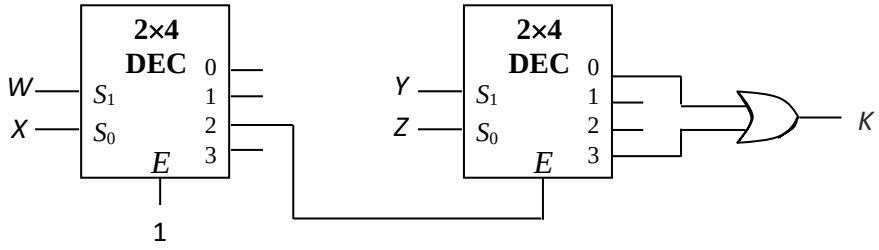
likewise except that we need to deal with the "d". Indeed, there is no fast way to decide whether "d" should be "0" or "1" in order to get the most simplified circuit. However, there is a good heuristic – we let "d" be the same as the other input in that same group. In this case, since input line 5 is a "0", we make the don't-care at input line 4 "0" as well. Doing so will make the design less complicated. (If we make the don't-care "1", then the third multiplexer input would be *Z'* instead of 0.) In my solution above, I list out both answers 0 and *Z'* for the third multiplexer input, but the question asks for the simplest one, so students should give "0". Can we write "d" for the third input? No, because if we write "d", it means that 1 is also a possible answer for the third multiplexer input.

(If time permits, show students how to do the 4:1 multiplexer if the last two variables, i.e. *Y* and *Z*, instead are chosen for the multiplexer selector lines.)

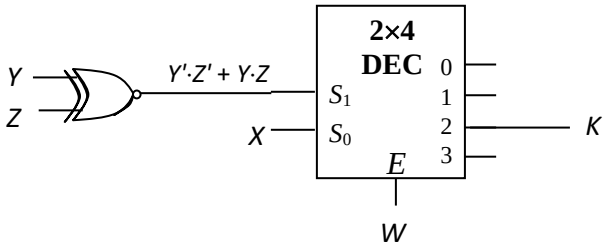
2. You are given the following Boolean function: ***K(W,X,Y,Z)* = Σ*m*(8, 11)**.

   You are to implement this function using the fewest number of one-enabled 2×4 decoder with normal outputs and at most one logic gate? (Logic gates, as you have learned, are NOT, AND, OR, NAND, NOR, XOR, and XNOR.)

   The following is one solution. Is there a simpler circuit using just one decoder and one logic gate?



   Answer: There might be other answers.



3. [AY2011/2 Semester 2 Exam question]
You are to design a converter that takes in 4-bit input *ABCD* and generates a 3-bit output *FGH* as shown in Table 1 below.

Input				Output		
A	***B***	***C***	***D***	***F***	***G***	***H***
0	0	0	0	0	0	0
1	0	0	0	0	0	1
1	1	0	0	0	1	0
1	1	1	0	0	1	1
1	1	1	1	1	0	0
0	1	1	1	1	0	1
0	0	1	1	1	1	0
0	0	0	1	1	1	1

Table 1

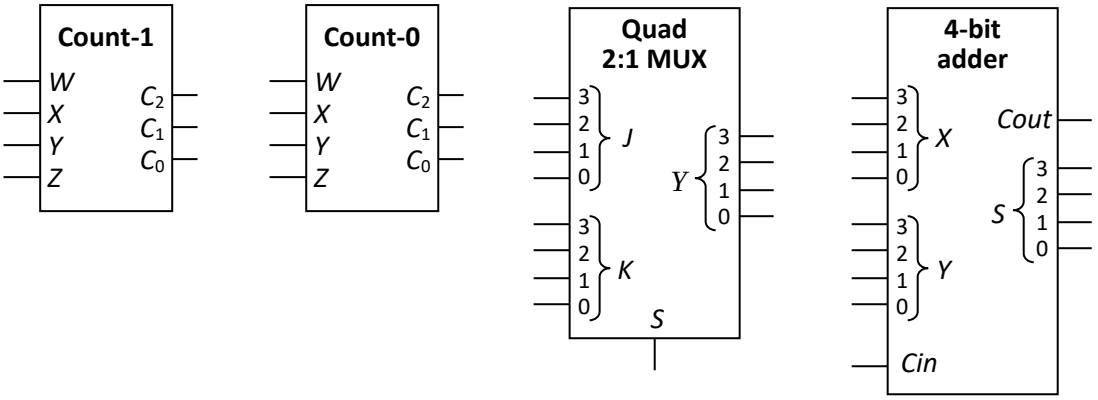
S	$Y_3Y_2Y_1Y_0$
0	$J_3J_2J_1J_0$
1	$K_3K_2K_1K_0$

Table 2

You are given the following components:

a. A **Count-1** device that takes in a 4-bit input *WXYZ* and generates a 3-bit output $C_2C_1C_0$ which is the number of 1s in the input. For example, if *WXYZ* = 0111, then $C_2C_1C_0$ = 011 (or 3).
b. A **Count-0** device that takes in a 4-bit input *WXYZ* and generates a 3-bit output $C_2C_1C_0$ which is the number of 0s in the input. For example, if *WXYZ* = 0111, then $C_2C_1C_0$ = 001 (or 1).
c. A **quad 2:1 multiplexer** that takes in two 4-bit inputs $J_3J_2J_1J_0$ and $K_3K_2K_1K_0$, and directs one of the inputs to its output $Y_3Y_2Y_1Y_0$ depending on its control signal *S*, as shown in Table 2 above.
d. A **4-bit parallel adder** that takes in two 4-bit unsigned binary numbers and outputs the sum.

The block diagrams of these components are shown below:

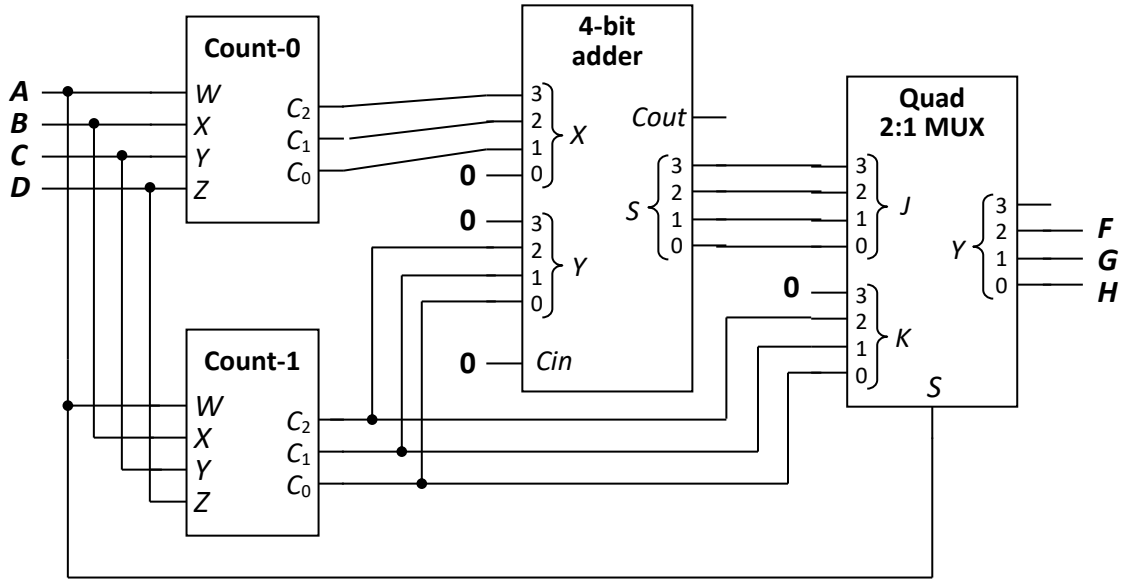


Given the above 4 components, you are to employ block-level design to design the converter, <u>without using any additional logic gate or other devices</u>. You may observe that if *A* = 1, then the output *FGH* is simply the number of 1s in the input *ABCD*. You are to make your own observation for the case when *A* = 0.

[Hint (not given in exam): You need only use one of each of the components. Complete the diagram below.]

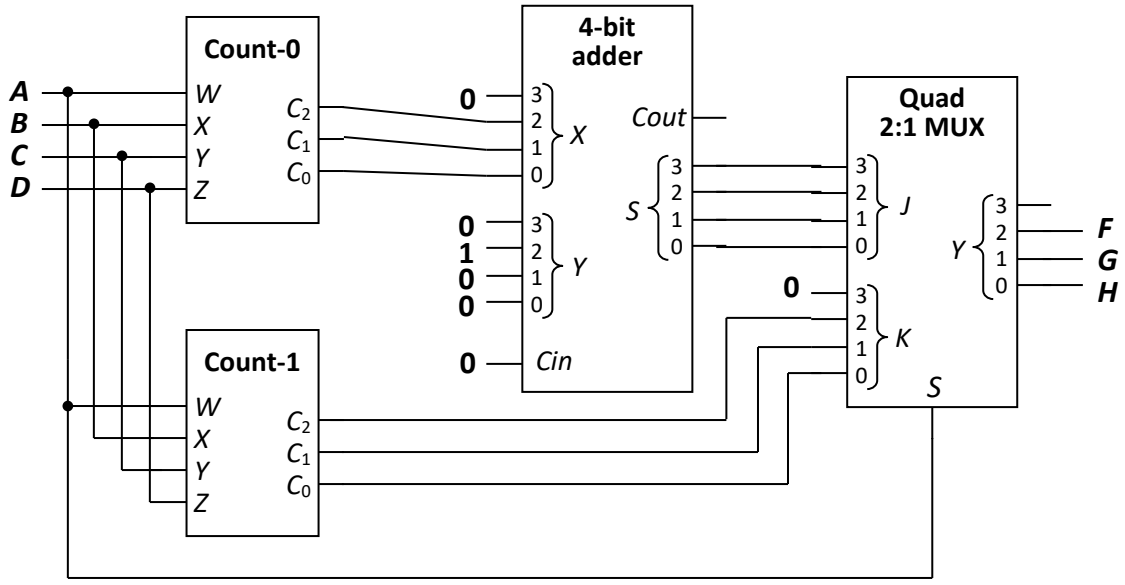
To tutors:
To save time, you may want to project the incomplete diagram on the whiteboard and get student to complete it on the whiteboard.

Key ideas:
1. If *A* = 1 (or *D* = 0), count #1s in *ABCD*.
2. If *A* = 0 (or *D* = 1), either
   a. #1s + 2 × #0s; or
   b. 4 + #0s

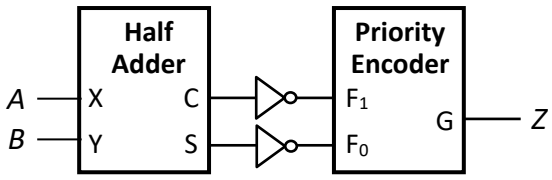


OR

Tutors: Check that all the inputs of the components are connected to some values, not left unconnected. Tell students that in the exam, the incomplete diagram will not be given!



4. [AY2023/24 Semester 1 Exam]
A Boolean function $Z(A,B)$ is implemented using a half adder, two inverters, and a 2-to-1 priority encoder as shown below. The function tables of the half adder and priority encoder are also shown below.



Half adder

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Priority encoder

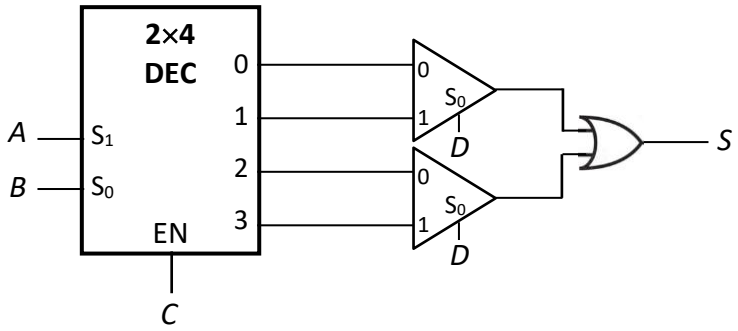
F_1	F_0	G
0	0	X
0	1	0
1	X	1

The circuit above may be replaced by a single 2-input logic gate. What is the logic gate?

Answer: NAND gate

X	Y	C	S	F_1	F_0	G
0	0	0	0	1	1	1
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	0	0	1	0

5. [AY2023/24 Semester 1 Exam]
A Boolean function $S(A,B,C,D)$ is implemented with a 2×4 decoder with one-enable, two 2:1 multiplexers and an OR gate as shown below.



What is $S(A,B,C,D)$ in Σm notation?

Answer:

$S(A,B,C,D) = \Sigma m(\mathbf{2,7,10,15})$.

$S = (D' \cdot (C \cdot A' \cdot B') + D \cdot (C \cdot A' \cdot B)) + (D' \cdot (C \cdot A \cdot B') + D \cdot (C \cdot A \cdot B))$
$= A' \cdot B' \cdot C \cdot D' + A' \cdot B \cdot C \cdot D + A \cdot B' \cdot C \cdot D' + A \cdot B \cdot C \cdot D$
$= m2 + m7 + m10 + m15$