

EEL3701

Menu

- MSI elements
 - > Decoders (pages 2-11)
 - > Encoders (page 12-15)
 - > Multiplexers (pages 16-19, 22-30)
 - Wild cards
 - > DeMux (page 20)
 - > Uses of MUXes

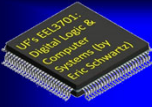
See Lam
Figures from
chapter 4

See hyperlink on web (Classes):
[Lam_Roth_MSI_figs_Lam_ch4_7-22.pdf](#)



University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo
1

1

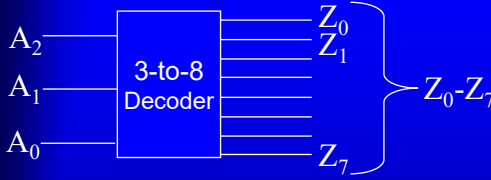


EEL3701

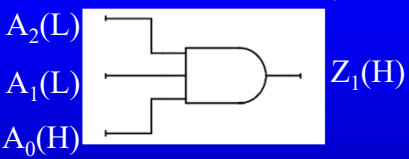
Decoder

3-to-8 Decoder

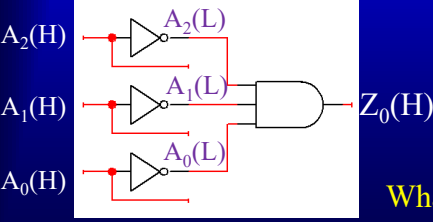
(3 input to 8 output; binary to singulary #)



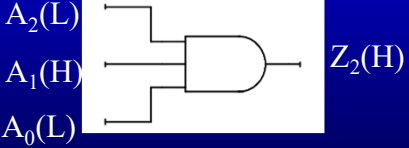
Look at $Z_1 = \neg A_2 / A_1 A_0$



Look at $Z_0 = \neg A_2 / A_1 / A_0$



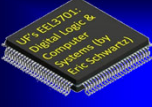
Look at $Z_2 = \neg A_2 A_1 / A_0$



What is the equation of Z_6 ? $Z_6 = A_2 A_1 / A_0$
 Draw the circuit.

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo
2

2



EEL3701

Decoder

- A **decoder** is a circuit that performs the following logic:

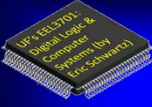
$$\begin{array}{l} X_1(H) \rightarrow A_1 \\ X_0(H) \rightarrow A_0 \end{array} \rightarrow \text{Decoder} \rightarrow \begin{array}{l} Y_0 \rightarrow Z_0(H) \\ Y_1 \rightarrow Z_1(H) \\ Y_2 \rightarrow Z_2(H) \\ Y_3 \rightarrow Z_3(H) \end{array}$$



> If $X_1X_0=00$, **$Z_0=1$** , $Z_1=Z_2=Z_3=0$, i.e., $Z_3Z_2Z_1Z_0=0001 = u_0$
 > If $X_1X_0=01$, **$Z_1=1$** , $Z_0=Z_2=Z_3=0$, i.e., $Z_3Z_2Z_1Z_0=0010 = u_1$
 > If $X_1X_0=10$, **$Z_2=1$** , $Z_0=Z_1=Z_3=0$, i.e., $Z_3Z_2Z_1Z_0=0100 = u_2$
 > If $X_1X_0=11$, **$Z_3=1$** , $Z_0=Z_1=Z_2=0$, i.e., $Z_3Z_2Z_1Z_0=1000 = u_3$

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

3



EEL3701

Decoder with Enable

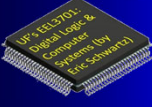
- A **decoder** with an **enable, E**, performs the following logic:

$$\begin{array}{l} X_1(H) \rightarrow A_1 \\ X_0(H) \rightarrow A_0 \\ \mathbf{E(H)} \rightarrow \mathbf{E} \end{array} \rightarrow \text{Decoder} \rightarrow \begin{array}{l} Y_0 \rightarrow Z_0(H) \\ Y_1 \rightarrow Z_1(H) \\ Y_2 \rightarrow Z_2(H) \\ Y_3 \rightarrow Z_3(H) \end{array}$$

> If $E=1$, then $Z_i=1$ and all other Z 's=0
 – One selected
 > If $E=0$, the all Z 's=0
 – None selected
 > Thus, when $E=1$ and $X_1X_0=10_2=2$, $Z_2=1$, $Z_0=Z_1=Z_3=0$.
 > A decoder (when enabled) decodes a binary word
 – In this example, a **two bit** binary word

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

4



EEL3701

Decoder: Why?



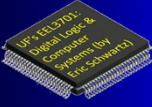
$X_1(H) \rightarrow A_1$
 $X_0(H) \rightarrow A_0$
 $E(H) \rightarrow E$

$Y_0 \rightarrow Z_0(H)$
 $Y_1 \rightarrow Z_1(H)$
 $Y_2 \rightarrow Z_2(H)$
 $Y_3 \rightarrow Z_3(H)$

- Decoders are useful because quite often information is “encoded”
- Why? A: To reduce I/O pins in ICs.
- What is an enable input? Why do we need it?
 - > Added to:
 - (1) Keep the circuit out of the way when not needed
 - (2) Because in a true singular representation all zeros are not allowed. Thus, all z_i 's 0 is not legal (by definition).
 - > The IC works if Enable = 1, else chip is “out” of the circuit.

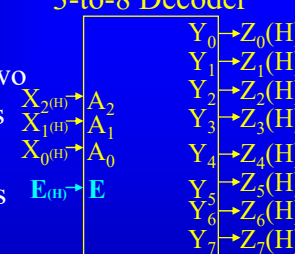
University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo
5

5



EEL3701

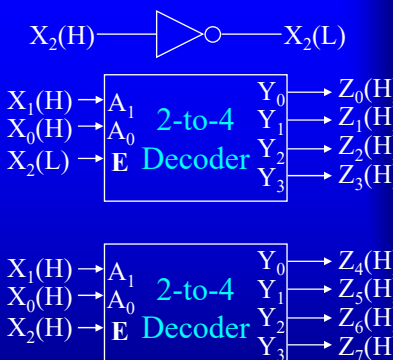
Decoder Expansion



$X_2(H) \rightarrow A_2$
 $X_1(H) \rightarrow A_1$
 $X_0(H) \rightarrow A_0$
 $E(H) \rightarrow E$

$Y_0 \rightarrow Z_0(H)$
 $Y_1 \rightarrow Z_1(H)$
 $Y_2 \rightarrow Z_2(H)$
 $Y_3 \rightarrow Z_3(H)$
 $Y_4 \rightarrow Z_4(H)$
 $Y_5 \rightarrow Z_5(H)$
 $Y_6 \rightarrow Z_6(H)$
 $Y_7 \rightarrow Z_7(H)$

- Make a 3-to-8 decoder with two 2-to-4 decoders
- Can you do this and keep an enable on the new 3-to-8 decoder? Do it!



$X_2(H) \rightarrow \text{Inverter} \rightarrow X_2(L)$
 $X_1(H) \rightarrow A_1$
 $X_0(H) \rightarrow A_0$
 $X_2(L) \rightarrow E$

$Y_0 \rightarrow Z_0(H)$
 $Y_1 \rightarrow Z_1(H)$
 $Y_2 \rightarrow Z_2(H)$
 $Y_3 \rightarrow Z_3(H)$



Functional Block Diagram

See Lam: Fig. 4.9, 4.10, 4.11

NOTE: We can build 4-to-16, 5-to-32 decoders using 74'138's.

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo
6

6

EEL3701
Another Decoder Expansion

• Make a 4-to-16 decoder with two 74'138 (3-to-8) decoders

4-to-16 Decoder

Inputs: $X_3(H) \rightarrow A_3$, $X_2(H) \rightarrow A_2$, $X_1(H) \rightarrow A_1$, $X_0(H) \rightarrow A_0$
 Outputs: $Y_0 \rightarrow Z_0(L)$, $Y_1 \rightarrow Z_1(L)$, $Y_2 \rightarrow Z_2(L)$, $Y_3 \rightarrow Z_3(L)$, ..., $Y_{15} \rightarrow Z_{15}(L)$
 Enable: $E(L) \rightarrow E$

Functional Block Diagram

See LogicWorks Simulation

74'138

Inputs: $X_3(H) \rightarrow G1$, $X_2(H) \rightarrow C$, $X_1(H) \rightarrow B$, $X_0(H) \rightarrow A$
 Outputs: $Y_7 \rightarrow Z_{15}(L)$, Y_6 , Y_5 , Y_4 , Y_3 , Y_2 , Y_1 , $Y_0 \rightarrow Z_8(L)$
 Enable: $E(L) \rightarrow G2B$, $G2A$, Gnd

74'138

Inputs: $V_{cc} \rightarrow G1$, $X_3(H) \rightarrow G2A$, $X_2(H) \rightarrow C$, $X_1(H) \rightarrow B$, $X_0(H) \rightarrow A$
 Outputs: $Y_7 \rightarrow Z_7(L)$, Y_6 , Y_5 , Y_4 , Y_3 , Y_2 , Y_1 , $Y_0 \rightarrow Z_0(L)$
 Enable: $E(L) \rightarrow G2B$, $G2A$

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

4-16_decoder.cct

7

EEL3701
BCD to 7-Segment Decoder

• BCD = Digits 0~9; $Y_3Y_2Y_1Y_0 = 0000, 0001, \dots, 1001$
 • Hex numbers A~F are not used.
 > Hence, the outputs for these inputs are “don’t cares.”

BCD-to-7 segment decoder

Inputs: Y_3, Y_2, Y_1, Y_0
 Outputs: a, b, c, d, e, f, g

7-Segment LED Display

Segments: a, b, c, d, e, f, g

See Lam: Fig. 4.11 & 4.12

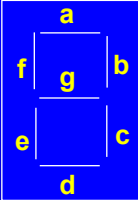
University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

8

EEL3701 **BCD to 7-Segment Decoder: Truth Table**

Note: This 6 is defined as  not .

• When does the *a* segment turn on?



Y3	Y2	Y1	Y0	a	b
0	0	0	0	1	1
0	0	0	1	0	1
0	0	1	0	1	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	1	1	0
...	...	...	...	...	...

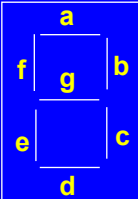
University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

9

EEL3701 **Equations for BCD to 7-Segment Decoder**

Note: This 6 is defined as  not .

• When does the *a* segment turn on?



		Y ₁ Y ₀			
		00	01	11	10
Y ₃ Y ₂	00	1	0	1	1
	01	0	1	1	0
	11	X	X	X	X
	10	1	1	X	X

K-map for output *a*

- $Y = Y_3 Y_2 Y_1 Y_0 = \{0000, 0001, \dots, 1001\}$
- When Y = illegal BCD codes (A~F) then **Don't Cares** for LED outputs

Or we can use the minterms or maxterms to derive the circuit

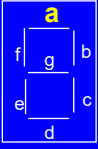
$$a = Y_3 + /Y_2 Y_1 + /Y_2 /Y_0 + Y_2 Y_0$$

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

10

10

EEL3701 Equations for BCD to 7-Segment LED



For 0, $a_0 = \neg Y_3 \neg Y_2 \neg Y_1 \neg Y_0$
 For 2, $a_2 = \neg Y_2 Y_1 \neg Y_0$
 For 3, $a_3 = \neg Y_2 Y_1 Y_0$
 For 5, $a_5 = Y_2 \neg Y_1 Y_0$
 For 7, $a_7 = Y_2 Y_1 Y_0$
 For 8, $a_8 = Y_3 \neg Y_0$
 For 9, $a_9 = Y_3 Y_0$

OR

$a = \sum a_i$
 1 4-bit AND
 4 3-bit ANDs
 2 2-bit ANDs

We can simplify the above SOP, or simplify the sum of minterms, or use a K-Map to derive the minimum circuit (as on previous page):

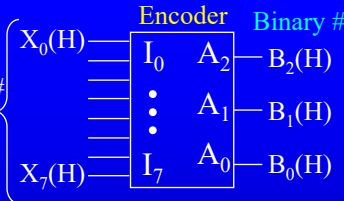
$a = Y_3 + \neg Y_2 Y_1 + \neg Y_2 \neg Y_0 + Y_2 Y_0$

Sum of minterms

University of Florida, EEL 3701 – File 09 © Drs. Schwartz & Arroyo

11

EEL3701 Encoder / Priority Encoder



Singular # X_0 - X_7

Encoder Binary #

If $(X_i=1)$ and $(X_j=0)$ for all $j \neq i$, then $(B_2 B_1 B_0) = i$.

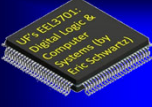
For $X_{7:0}=0$, $(B_2 B_1 B_0) = 0$

Examples: (1) Let $X_5=1$, then $(B_2 B_1 B_0)_2 = 101_2$
 (2) Let $X_4=1$, then $(B_2 B_1 B_0)_2 = 100_2$
 (3) Let X_3 & X_6 be 1, then **ILLEGAL!**

- To solve case 3 we need **priority encoders**.
- Priority encoders** output the index of the highest priority true input.

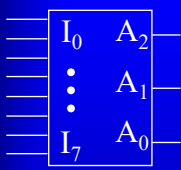
University of Florida, EEL 3701 – File 09 © Drs. Schwartz & Arroyo

12

 **EEL3701**

Encoder

Encoder



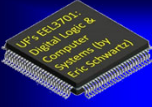
I7	I6	I5	I4	I3	I2	I1	I0	A2	A1	A0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

Lam: Fig 4.13

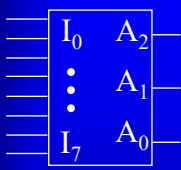
13

13

 **EEL3701**

A Priority Encoder

Priority Encoder



I7	I6	I5	I4	I3	I2	I1	I0	A2	A1	A0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	-	0	0	1
0	0	0	0	0	1	-	-	0	1	0
0	0	0	0	1	-	-	-	0	1	1
0	0	0	1	-	-	-	-	1	0	0
0	0	1	-	-	-	-	-	1	0	1
0	1	-	-	-	-	-	-	1	1	0
1	-	-	-	-	-	-	-	1	1	1

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

14

14

EEL3701
74'148: 8-to-3 Priority Encoder

EI	I7	I6	I5	I4	I3	I2	I1	I0	A2	A1	A0	GS	EO
0	-	-	-	-	-	-	-	-	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	1	-	0	0	1	1	0
1	0	0	0	0	0	1	-	-	0	1	0	1	0
1	0	0	0	0	1	-	-	-	0	1	1	1	0
1	0	0	0	1	-	-	-	-	1	0	0	1	0
1	0	0	1	-	-	-	-	-	1	0	1	1	0
1	0	1	-	-	-	-	-	-	1	1	0	1	0
1	1	-	-	-	-	-	-	-	1	1	1	1	0

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

15

EEL3701
Multiplexer (MUX)

MUX (multiplexers; selectors) / DEMUX (demultiplexers; distributors)

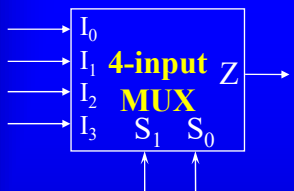
$Z = X_{(S_1, S_0)_2}$
Thus, if $S_1 S_0 = 10_2$, then $Z = X_2$

A MUX switches a choice of any single input to a single output

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

16

EEL3701
Multiplexer (MUX) Equation



- We call this a 4-input MUX (even though there are 6 inputs).
- How many rows in a truth table?
- Use wild cards (- or *) for inputs that can take on both values

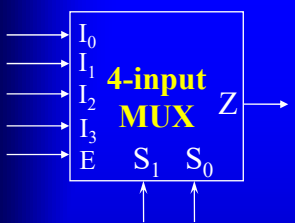
$$Z = \overline{S_1} \overline{S_0} I_0 + \overline{S_1} S_0 I_1 + S_1 \overline{S_0} I_2 + S_1 S_0 I_3$$

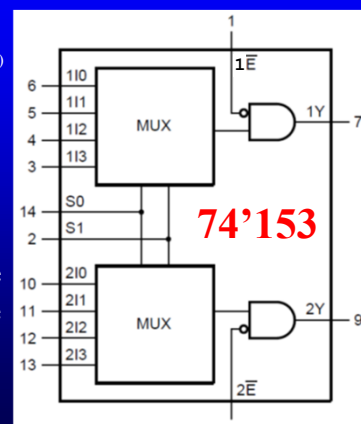
S_1	S_0	I_3	I_2	I_1	I_0	Z
0	0	-	-	-	0	0
0	0	-	-	-	1	1
0	1	-	-	0	-	0
0	1	-	-	1	-	1
1	0	-	0	-	-	0
1	0	-	1	-	-	1
1	1	0	-	-	-	0
1	1	1	-	-	-	1

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

17

EEL3701
MUX with Enable

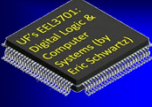


$$Z = E (\overline{S_1} \overline{S_0} I_0 + \overline{S_1} S_0 I_1 + S_1 \overline{S_0} I_2 + S_1 S_0 I_3)$$


- If the enable is true, it works as described previously
- If the enable is false, the output is false
- There is another kind of enable that we will describe later, i.e., a tri-state enable

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

18



EEL3701

Multiplexer (MUX, Selector)

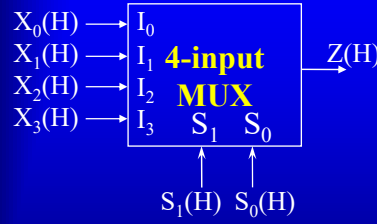


Diagram of a 4-input MUX. Inputs $X_0(H)$, $X_1(H)$, $X_2(H)$, and $X_3(H)$ are connected to I_0 , I_1 , I_2 , and I_3 respectively. Select inputs $S_1(H)$ and $S_0(H)$ are connected to S_1 and S_0 . The output is $Z(H)$.

$$Z = X_{(S_1, S_0)_2}$$

Thus, if $S_1 S_0 = 10_2$, then $Z = X_2$

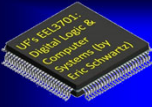
- Simple, but useful circuit. The techniques of K-maps, etc., do not help us implement simple circuit like these.

See Lam: Fig 4.16-4.18

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

19

19



EEL3701

DeMultiplexer (DEMUX, Distributor)

- DEMUX** is like MUX with the data path reversed.

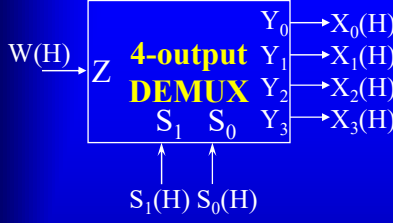


Diagram of a 4-output DEMUX. Input $W(H)$ is connected to Z . Select inputs $S_1(H)$ and $S_0(H)$ are connected to S_1 and S_0 . Outputs Y_0 , Y_1 , Y_2 , and Y_3 are connected to $X_0(H)$, $X_1(H)$, $X_2(H)$, and $X_3(H)$ respectively.

$$\{X\} = \{W \cdot u_{(S_1, S_0)_2}\}$$

Thus, if $S_1 S_0 = 10_2$, then $X_2 = W$ and $X_3 = X_1 = X_0 = 0$.

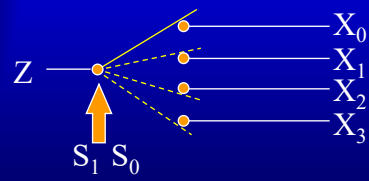


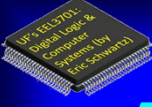
Diagram showing a switch controlled by S_1 and S_0 routing input Z to one of four outputs X_0 , X_1 , X_2 , or X_3 .

A DEMUX takes an input and switches it to a choice any single output

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

20

20

 **EEL3701**
74'138 Decoder used as a DeMUX

74'138 DEMUX

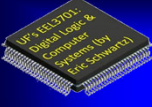
Inputs: Data(L) to E₁, GND to E₂, Vcc to E₃.
 Selection inputs: A₂, A₁, A₀.
 Outputs: Z₀(L), Z₁(L), Z₂(L), Z₃(L), Z₄(L), Z₅(L), Z₆(L), Z₇(L).
 Address inputs: A₂(H), A₁(H), A₀(H).

Enable		Data	Selection			Outputs							
E ₂	E ₃	E ₁	A ₂	A ₁	A ₀	Z ₀	Z ₁	Z ₂	Z ₃	Z ₄	Z ₅	Z ₆	Z ₇
X	X	0	X	X	X	0	0	0	0	0	0	0	0
0	X	X	X	X	X	0	0	0	0	0	0	0	0
X	0	X	X	X	X	0	0	0	0	0	0	0	0
1	1	1	0	0	0	1	0	0	0	0	0	0	0
1	1	1	0	0	1	0	1	0	0	0	0	0	0
1	1	1	0	1	0	0	0	1	0	0	0	0	0
1	1	1	0	1	1	0	0	0	1	0	0	0	0
1	1	1	1	0	0	0	0	0	0	1	0	0	0
1	1	1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	1	1	0	0	0	0	0	0	0	1

From Lam: Fig 4.22

© Drs. Schwartz & Arroyo

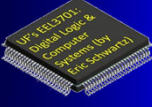
21

 **EEL3701**
Other MSI Components LATER

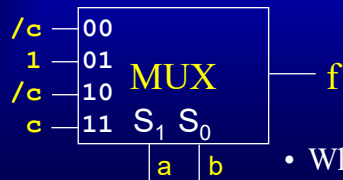
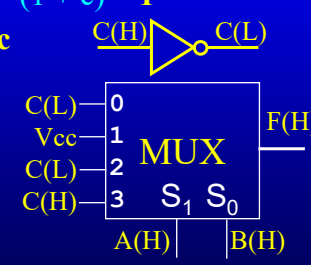
- Later in this lecture:
 - > MUXes used to implement equations
- Future lectures: **More MSI Elements!**
 - > **Open-collector devices**
 - > **Tri-state buffers**
 - > **Adders**/Subtractors (Lam: Fig 4.1 - 4.6)
 - > Magnitude Comparators (Lam: Fig 4.7 & 4.8)
 - > **ALUs (arithmetic logic units)**

University of Florida, EEL 3701 – File 09
 © Drs. Schwartz & Arroyo

22

 **EEL3701 MUX to Implement Logic Design Using Equation**

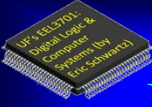
- <Ex 1> $f = /a b + /b /c + bc$
 - > The new way (with a MUX); either from equation
- Chose MUX size
- Chose the inputs for the MUX selector lines. I'll chose "a" & "b"
- When $(ab)_2=00$, then $f = 1*0 + 1*/c + 0*c = /c$
- When $(ab)_2=01$, then $f = 1*1 + 0*/c + 1*c = 1*(1 + c) = 1$
- When $(ab)_2=10$, then $f = 0*0 + 1*/c + 0*c = /c$
- When $(ab)_2=11$, then $f = 0*1 + 0*/c + 1*c = c$

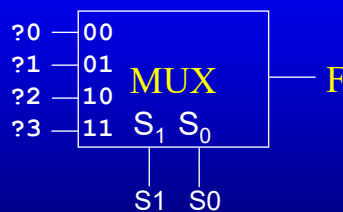
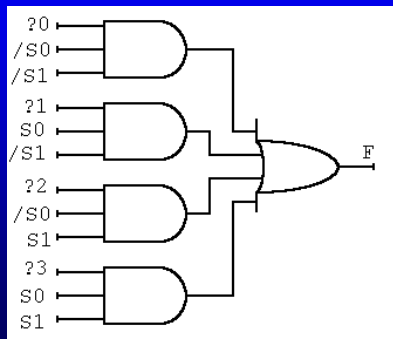
- Which is easier/cheaper/preferable, a MUX implementation or an SOP/POS?

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

23

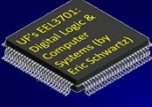
 **EEL3701 MUX to Implement Logic Design**

- Implementation of a general 4-input MUX

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

24

 **EEL3701**

4 Variable Function with 4-input MUX

<Ex 3> How about a 4 variable function with a 4-input multiplexer?

$x_0 (\bar{a})$ 00

$x_1 (\bar{b})$ 01

$x_2 (\bar{c})$ 10

$x_3 (\bar{d})$ 11

MUX

$S_1 S_0$

| a | b

→

0 00

c 01

/c d 10

c ⊕ d 11

MUX

$S_1 S_0$

| a | b

$Z_{(H)}$

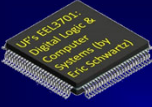
Let $Z = f(a,b,c,d) = \bar{a} b c d + a \bar{c} d + b c \bar{d}$

- When $ab = 00_2$, then $f(a=0, b=0, c, d) = 0 + 0 + 0 = 0$; so $X_0 = 0$
- When $ab = 01_2$, then $f(a=0, b=1, c, d) = cd + 0 + c \bar{d} = c$; so $X_1 = c$
- When $ab = 10_2$, then $f(a=1, b=0, c, d) = 0 + \bar{c} d + 0 = \bar{c} d$; so $X_2 = \bar{c} d$
- When $ab = 11_2$, then $f(a=1, b=1, c, d) = 0 + \bar{c} d + c \bar{d} = c \oplus d$

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

25

25

 **EEL3701**

4 Variable Function with 4-input MUX

<Ex 4> Same equation, but a possibly better choice for the selects.

$x_0 (\bar{a})$ 00

$x_1 (\bar{b})$ 01

$x_2 (\bar{c})$ 10

$x_3 (\bar{d})$ 11

MUX

$S_1 S_0$

| c | d

→

0 00

a 01

b 10

/a b 11

MUX

$S_1 S_0$

| c | d

$Z_{(H)}$

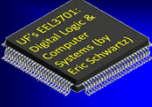
Let $Z = f(a,b,c,d) = \bar{a} b c d + a \bar{c} d + b c \bar{d}$

- When $cd = 00_2$, then $f(a, b, c=0, d=0) = 0 + 0 + 0 = 0$; so $X_0 = 0$
- When $cd = 01_2$, then $f(a, b, c=0, d=1) = 0 + a + 0 = a$; so $X_1 = a$
- When $cd = 10_2$, then $f(a, b, c=1, d=0) = 0 + 0 + b = b$; so $X_2 = b$
- When $cd = 11_2$, then $f(a, b, c=1, d=1) = \bar{a} b + 0 + 0$; so $X_3 = \bar{a} b$

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

26

26

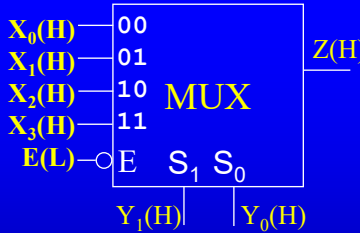
 **EEL3701**

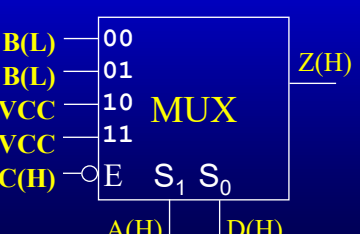
MUX with Enable

- Now let's use a MUX with an enable
- Let $Z = f(A, B, C, D)$

$$Z = A*B*/C + /B*/C*/D + A*/C*D + /A*/B*/C*D$$
- Utilize the enable by finding a common variable in each term
- Pick select lines by finding other most common terms

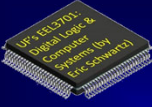
$$Z = /C*(A*B + /B*/D + A*D + /A*/B*D)$$





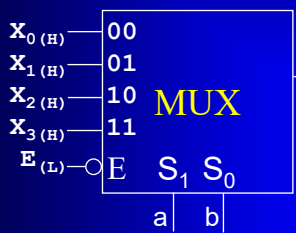
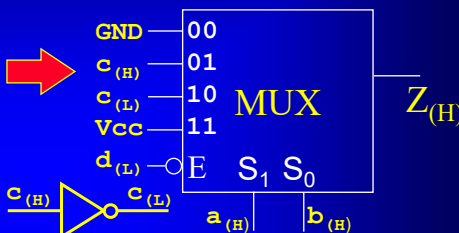
University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

27

 **EEL3701**

4 Variable Function with 4-input MUX

<Ex 5> Similar equation to ex 4 (but /d in last term changed to d)


→


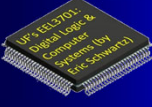
Let $Z = f(a, b, c, d) = /a b c d + a /c d + b c d = d (/a b c + a /c + b c)$

$Z = d * f_1(a, b, c)$; when $d=1$, then $Z = f_1$; when $d=0$, $Z = 0$

- When $ab=00_2$, then $f_1(a=0, b=0, c) = 1*(0 + 0 + 0) = 0$; so $X_0 = 0$
- When $ab=01_2$, then $f_1(a=0, b=1, c) = 1*(c + 0 + c) = c$; so $X_1 = c$
- When $ab=10_2$, then $f_1(a=1, b=0, c) = 1*(0 + /c + 0) = /c$; so $X_2 = /c$
- When $ab=11_2$, then $f_1(a=1, b=1, c) = 1*(0 + /c + c) = 1$; so $X_3 = 1$

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

28

 **EEL3701** MUX to Implement Logic Design Using K-Map

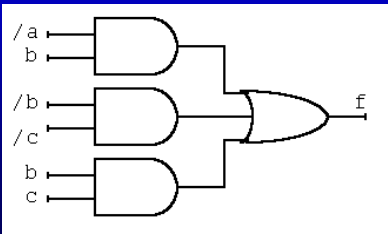
- You can do some useful things with MUXes, e.g., you can **directly implement function**
- <Ex 1> $f = \neg a b + \neg b \neg c + bc$

> The old way

a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

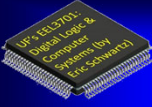
ab	c=0	c=1
00	1	0
01	1	1
11	0	1
10	1	0

K-Map (“game of children”); coming soon!



University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

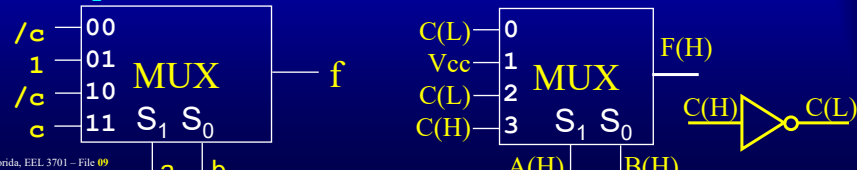
29

 **EEL3701** MUX to Implement Logic Design Using K-Map

- <Ex 1> $f = \neg a b + \neg b \neg c + bc$
- > The new way (with a MUX) and a K-map

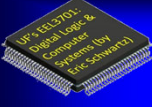
ab	c=0	c=1
00	1	0
01	1	1
11	0	1
10	1	0

- Chose MUX size
- Chose the inputs for the MUX selector lines. I'll chose “a” & “b”
- When $(ab)_2=00$, then $f = (\text{Map values from K-map row: } 1\ 0) = \neg c$
- When $(ab)_2=01$, then $f = (\text{Map values from K-map row: } 1\ 1) = 1$
- When $(ab)_2=11$, then $f = (\text{Map values from K-map row: } 0\ 1) = c$
- When $(ab)_2=10$, then $f = (\text{Map values from K-map row: } 1\ 0) = \neg c$



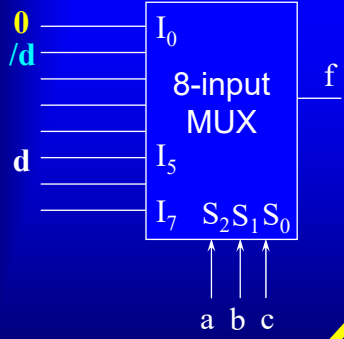
University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

30

 **EEL3701**

MUX and 4 Variable Function

<Ex 2> How about a 4 variable function?



	cd			
ab	00	01	11	10
00	0	0	0	1
01	1	1	1	0
11	0	0	1	0
10	0	1	1	0

when (abc)=000, $f \sim 00 \Rightarrow 0$

when (abc)=001, $f \sim 01 \Rightarrow /d$

...

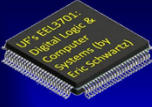
when (abc)=101, $f \sim 10 \Rightarrow d$

etc.

K-Map ("game of children"); coming soon!

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

31

 **EEL3701**

The End!

University of Florida, EEL 3701 – File 09
© Drs. Schwartz & Arroyo

32

32