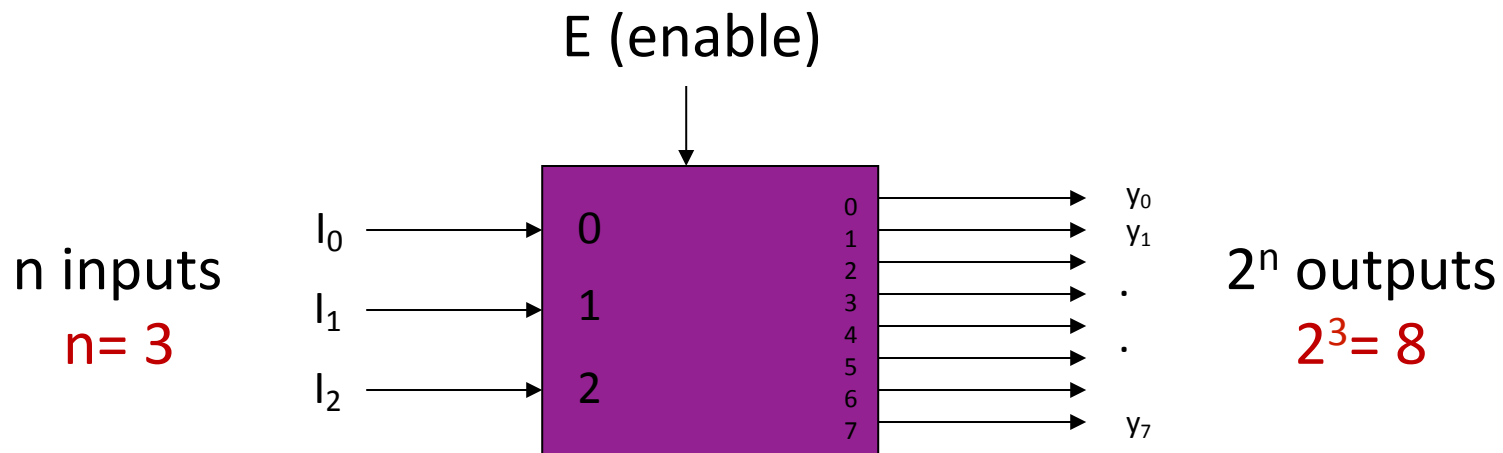


CSE 140 Discussion

Homework 4

Decoder Definition

A digital module that converts a binary address to the assertion of the addressed device.



$$y_i = 1 \text{ if } E = 1 \ \& \ (I_n, I_{n-1}, \dots, I_0) = i$$
$$y_i = 0, \text{ otherwise}$$

Decoder Application

- {Decoder, OR} is a universal set
 - Any Boolean function can be implemented with a set of decoders and OR gates.
- If the function is described in the sum-of-products (minterms) form:
 - Decoder produces the minterms.
 - OR gate produces the sum from the minterms.
 - Ignore the don't-cares.

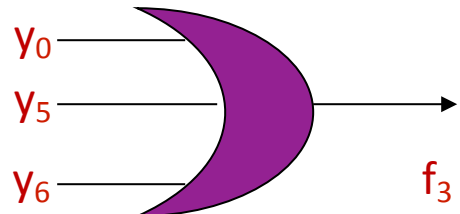
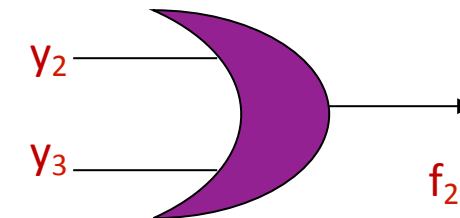
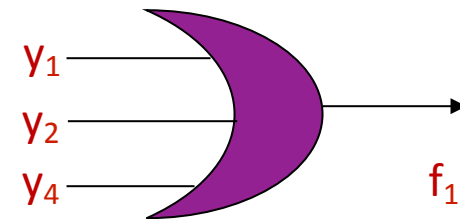
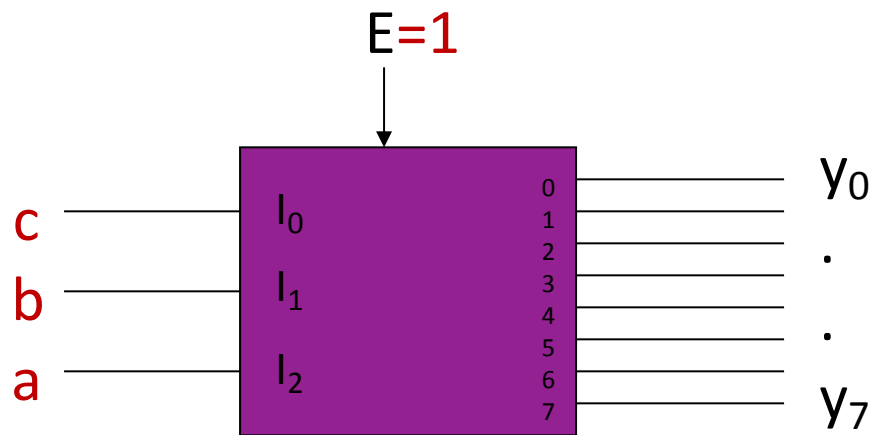
Problem 1: Example

Implement 3-variable functions with a 3:8 decoder and OR gates.

i) $f_1(a,b,c) = \Sigma m(1,2,4) + \Sigma d(0,5),$

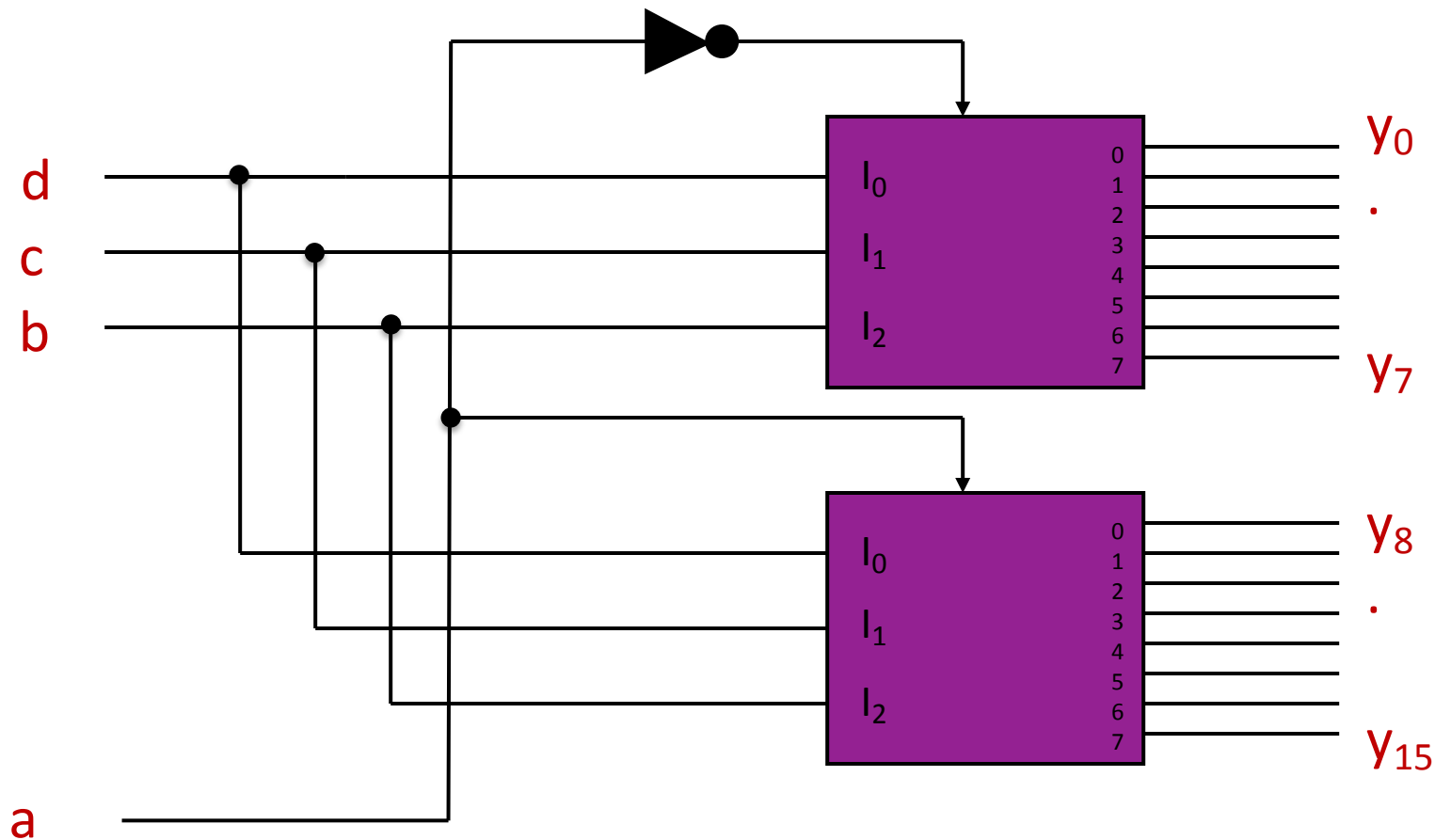
ii) $f_2(a,b,c) = \Sigma m(2,3) + \Sigma d(1,4),$

iii) $f_3(a,b,c) = \Sigma m(0,5,6)$



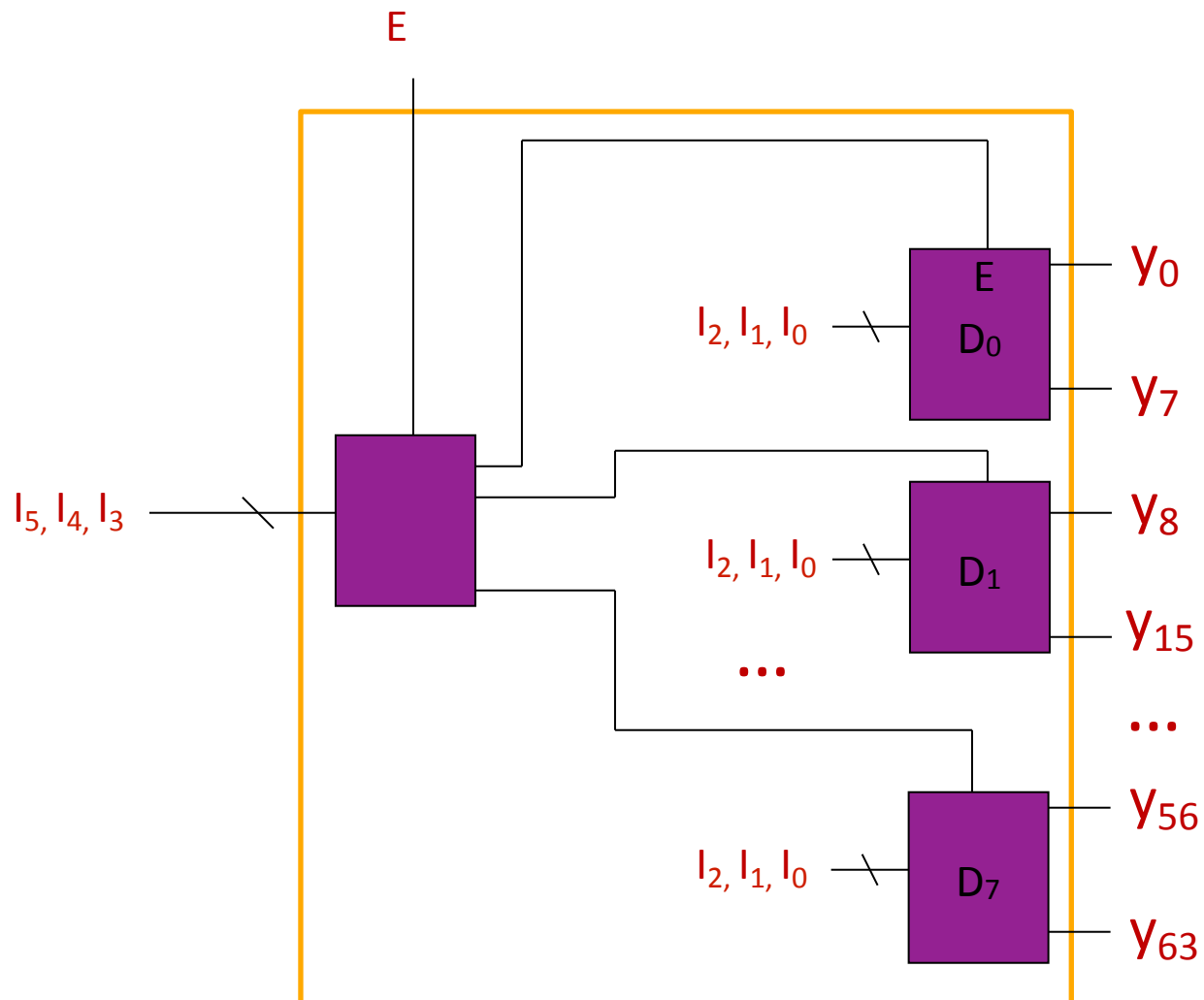
What if the function has more than three variables?

4 variables: Implement a 4:16 decoder using 3:8 decoders.



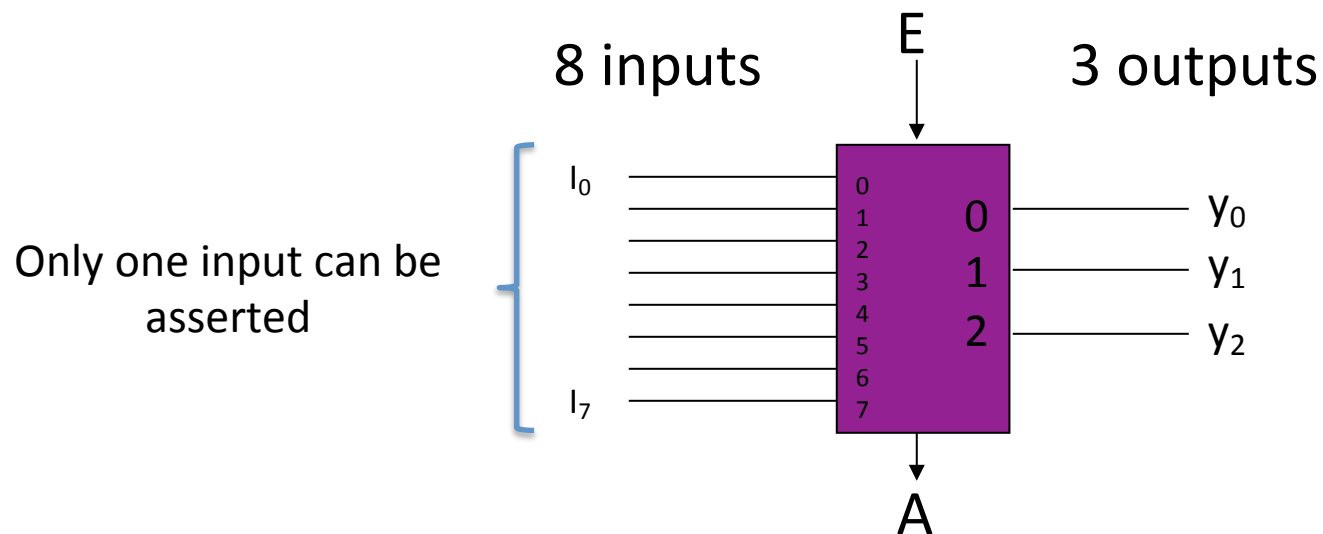
What if the function has more than three variables?

6 variables: Implement a 6:64 decoder using 3:8 decoders.



Encoder Definition

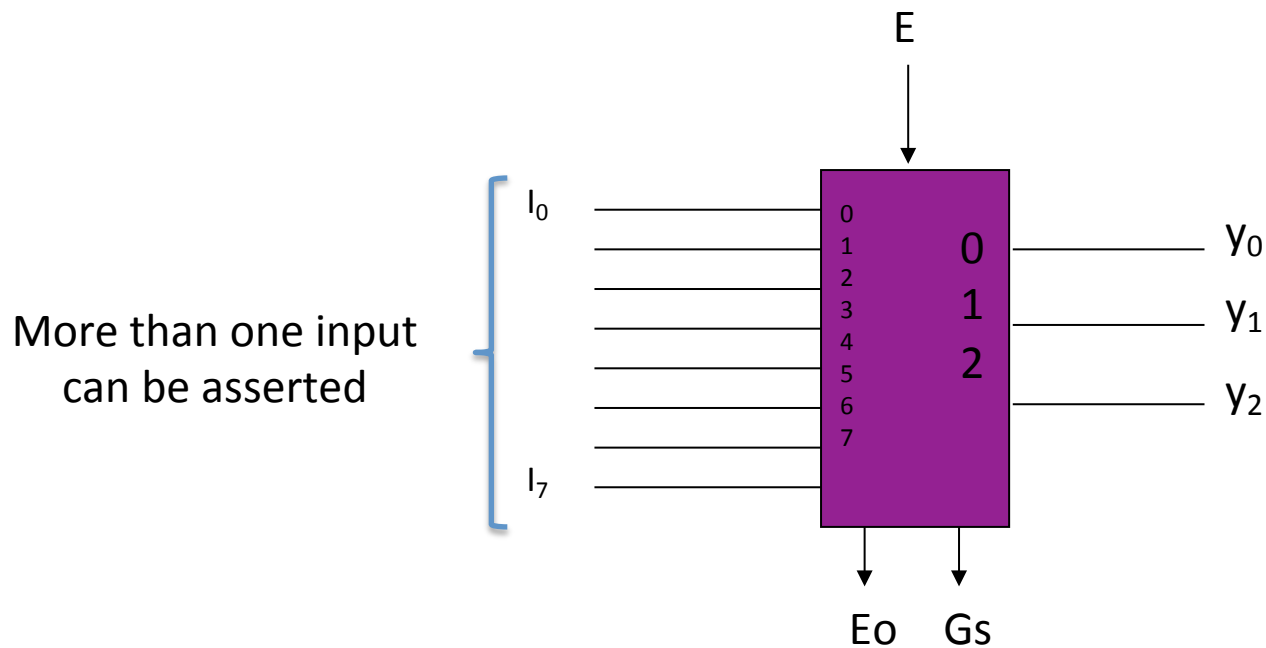
A digital module that converts the assertion of a device to the binary address of the device.



$(y_{n-1}, \dots, y_0) = I$ if $E = 1$ & one i s.t. $I_i = 1$
 $(y_{n-1}, \dots, y_0) = 0$, otherwise.

$A = 1$ if $E = 1$ & one i s.t. $I_i = 1$
 $A = 0$, otherwise.

Priority Encoder

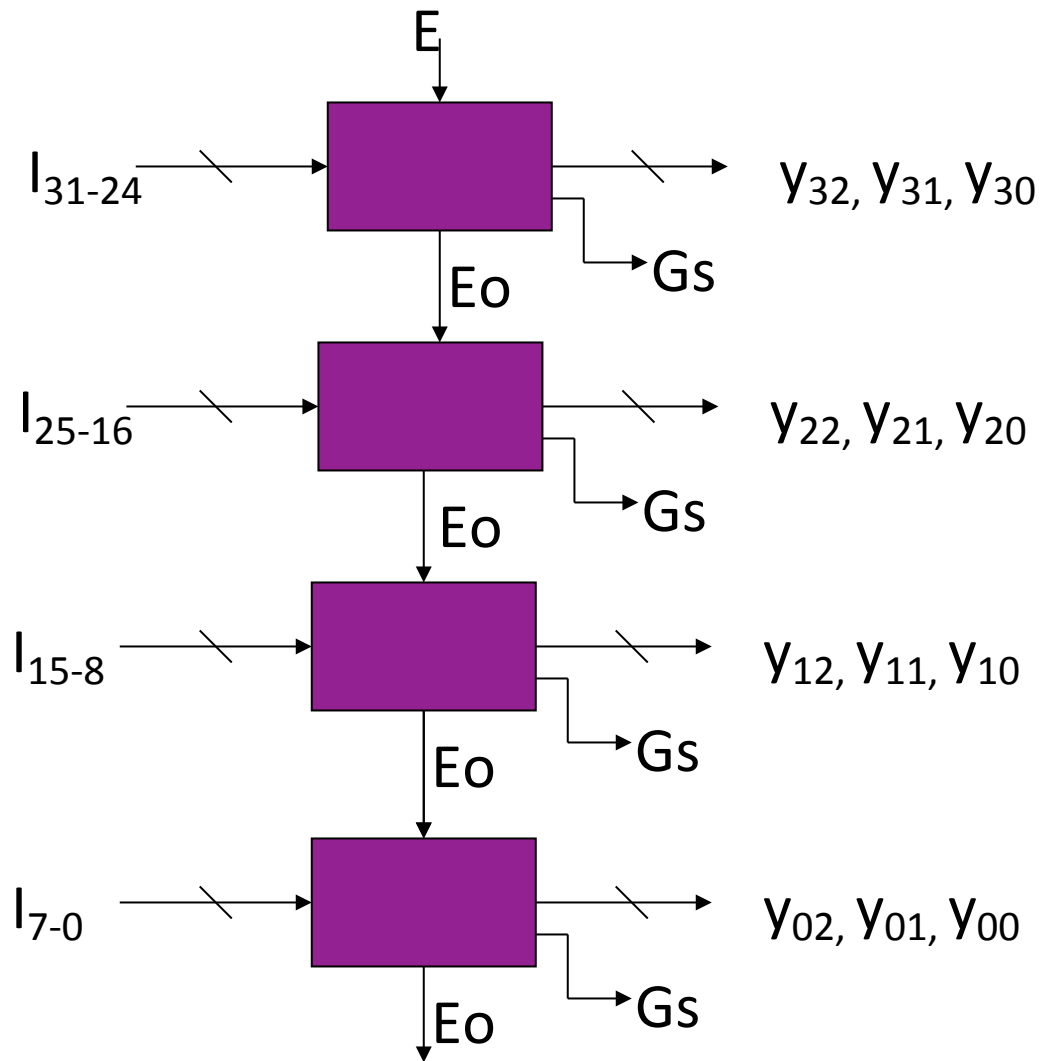


$(y_{n-1}, \dots, y_0) = I_i$ if $E = 1$ & $I_i = 1$ & $I_k = 0$ for all $k > i$
 $(y_{n-1}, \dots, y_0) = 0$, otherwise.

$E_0 = 1$ if $E = 1$ & $I_i = 0$ for all i

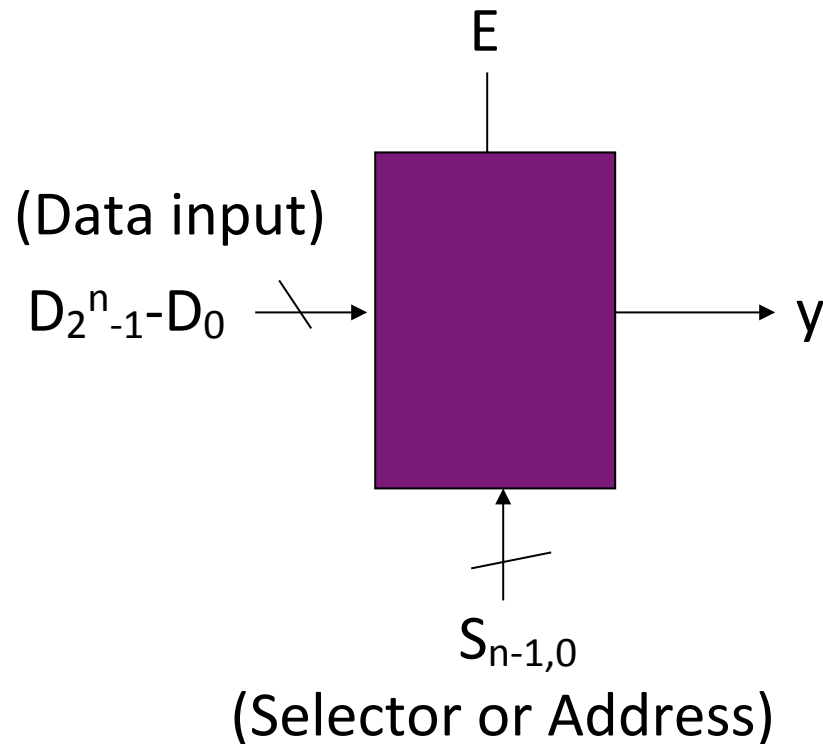
$G_s = 1$ if $E = 1$ & $\exists i$ s.t. $I_i = 1$

Problem 2: Example



Multiplexer Definition

A digital module that selects one of data inputs according to the binary address of the selector.



$y = D_i$ where $i = (S_{n-1}, \dots, S_0)$ if $E = 1$
 $y = 0$, otherwise

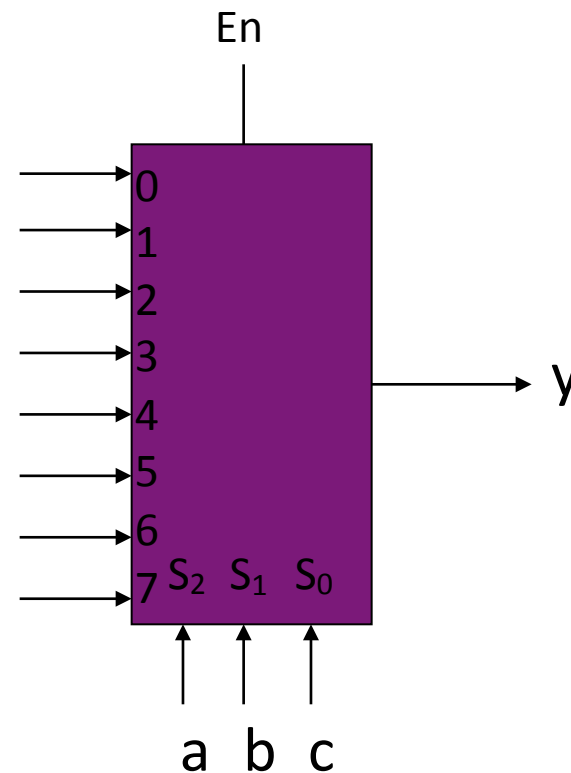
Multiplexer Application

- {Mux} is a universal set.
 - Any Boolean function can be implemented with a set of multiplexers.
 - Use Shannon's expansion to decompose the function into smaller ones
 - simplify the decomposed functions using K-maps
 - Feed the smaller functions to data inputs of the mux and use the the expansion variable as the selector.

Problem 3: Example

Implement $f(a,b,c) = \sum m(0,2,6) + \sum d(1)$ using an 8-input Mux.

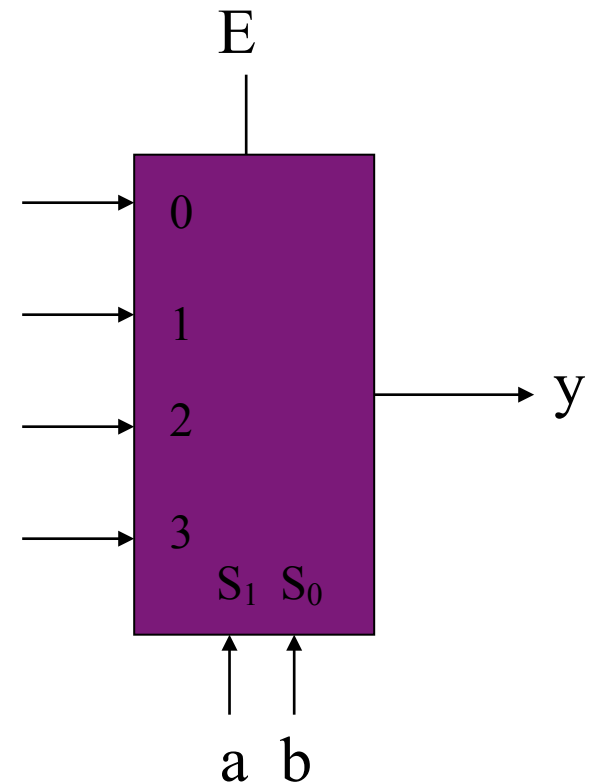
id	abc	f
0	000	1
1	001	-
2	010	1
3	011	0
4	100	0
5	101	0
6	110	1
7	111	0



What if the we have to use smaller muxes?

Implement $f(a,b,c) = \sum m(0,2,6) + \sum d(1)$ using an 4-input Mux.

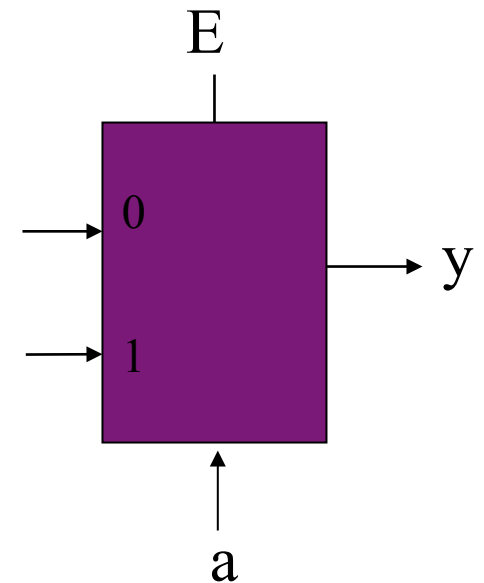
ab	c = 0	c = 1	D(c)
00	1	-	$D_0 = 1$
01	1	0	$D_1 = c'$
10	0	0	$D_2 = 0$
11	1	0	$D_3 = c'$



What if the we have to use smaller muxes?

Implement $f(a,b,c) = \sum m(0,2,6) + \sum d(1)$ using an 2-input Mux.

a\bc	00	01	10	11	D(b,c)
0	1	-	1	0	D_0
1	0	0	1	0	D_1



What if the we have to use smaller muxes?

Implement $f(a,b,c) = \sum m(0,2,6) + \sum d(1)$ using an 2-input Mux.

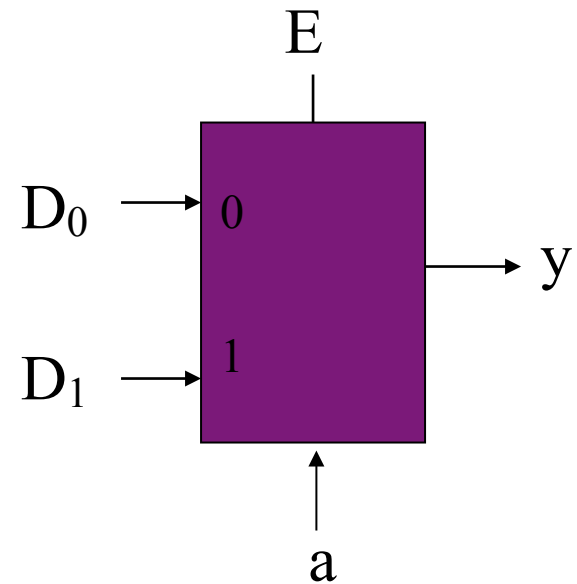
a\bc	00	01	10	11	D(b,c)
0	1	-	1	0	$D_0 = c'$
1	0	0	1	0	$D_1 = bc'$

$$D_0(b,c) = c'$$

c = 0	1	1
c = 1	-	0
	b = 0	b = 1

$$D_1(b,c) = bc'$$

c = 0	0	1
c = 1	0	0
	b = 0	b = 1



What if the we have to use smaller muxes?

Implement $f(a,b,c) = \sum m(0,2,6) + \sum d(1)$ using an 2-input Mux.

$$D_1(b,c) = bc'$$

$b \backslash c$	0	1	$D(c)$
0	0	0	$D_0 = 0$
1	1	0	$D_1 = c'$

