COL215 Digital Logic and System Design

Combinational Logic Minimization,

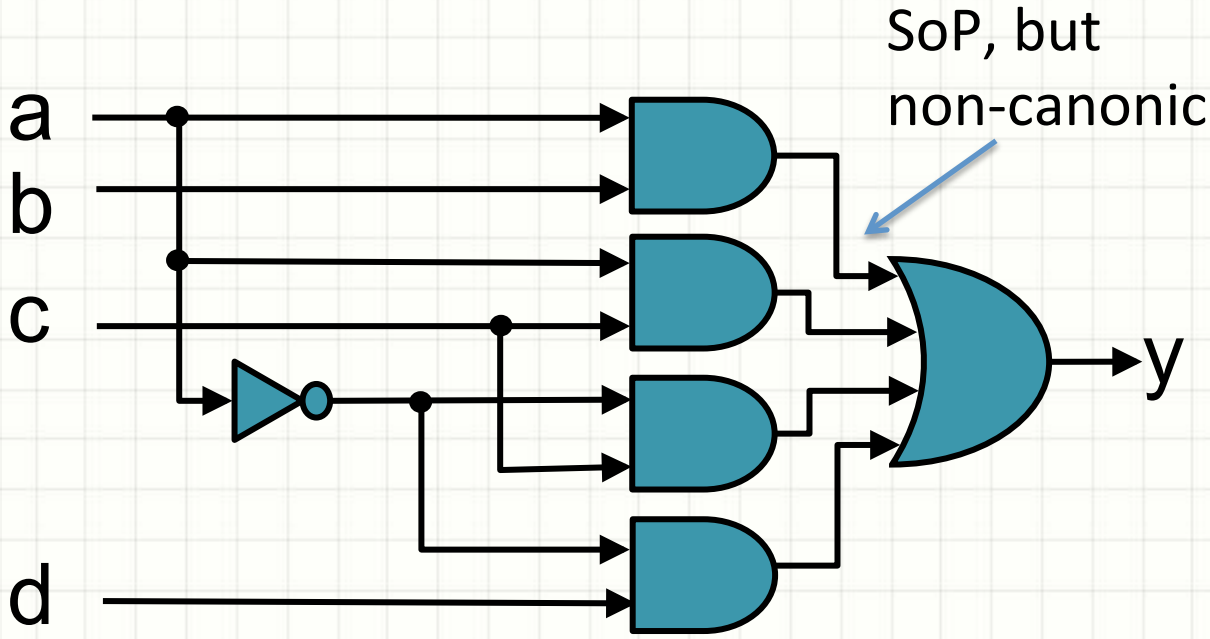
Combinational Modules

01 August 2017

Lecture Outline

- SoP, PoS forms
- Nand, Nor forms
- Minimization using Karnaugh maps
- Some combinational logic modules
- Lab exercise 2

Canonical SoP form



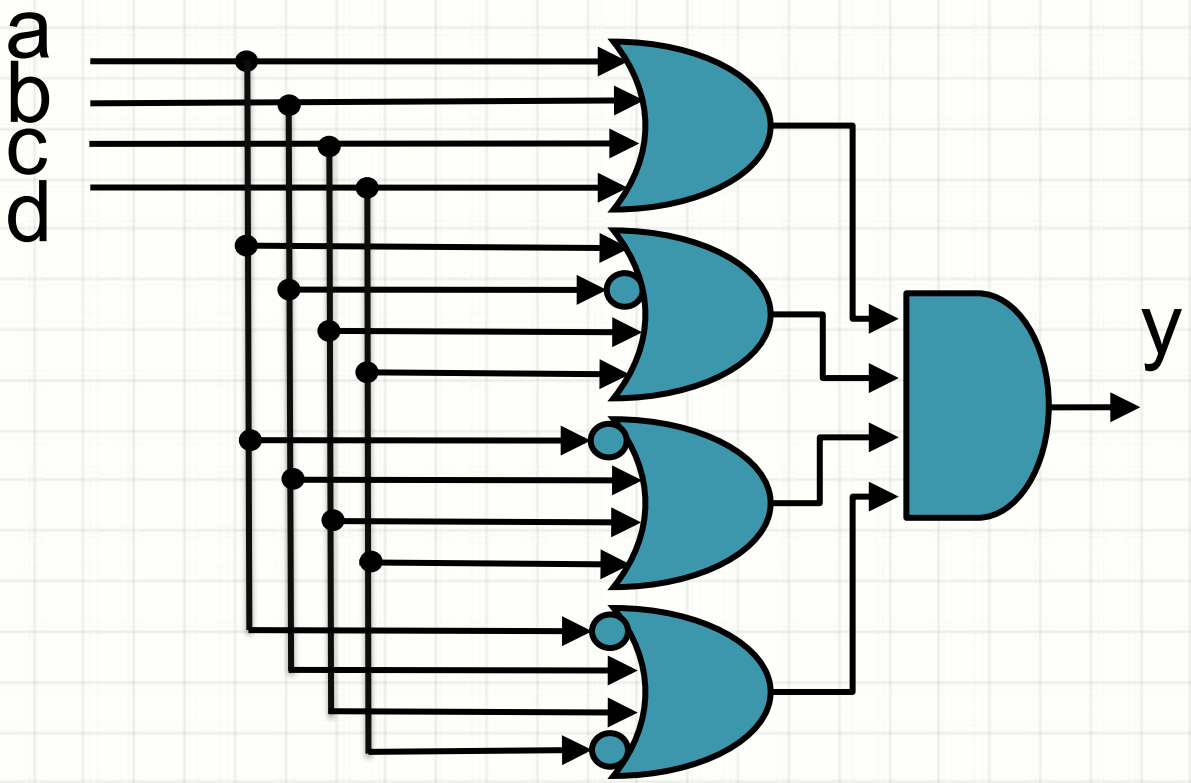
a	b	c	d	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Canonical Sum of Product (SOP) form

y = (a’.b’.c’.d) + (a’.b’.c .d’) + (a’.b’.c .d)
+ (a’.b .c’.d) + (a’.b .c .d’) + (a’.b .c .d)
+ (a .b’.c .d’) + (a .b’.c .d) + (a .b .c’.d’)
+ (a .b .c’.d) + (a .b .c .d’) + (a .b .c .d)

These are “min terms”

Canonical POS form



a	b	c	d	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

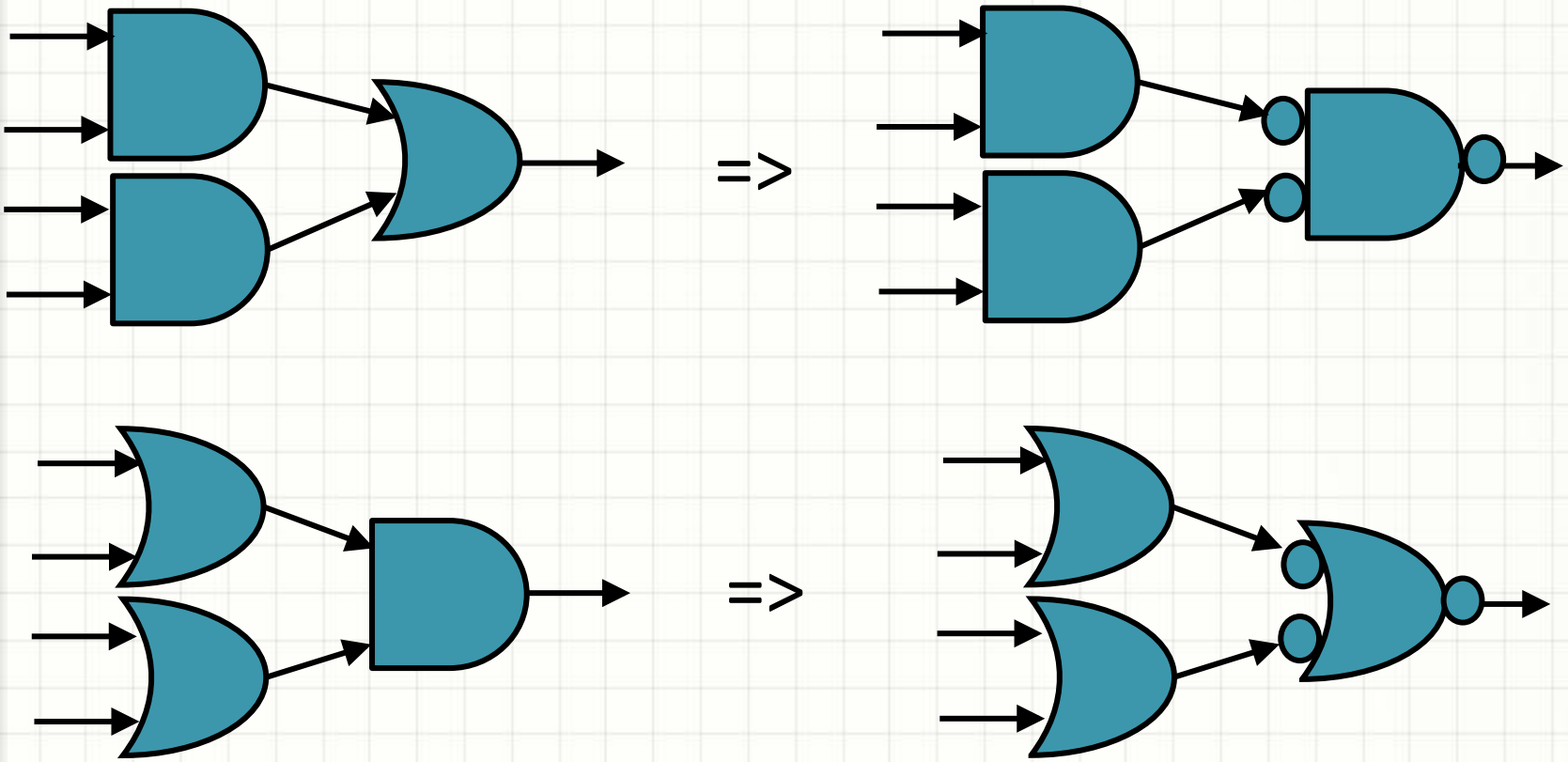
y' = a'b'c'd' + a'b c'd' + a b'c'd' + a b'c'd

y = (a + b + c + d) . (a + b'+ c + d)

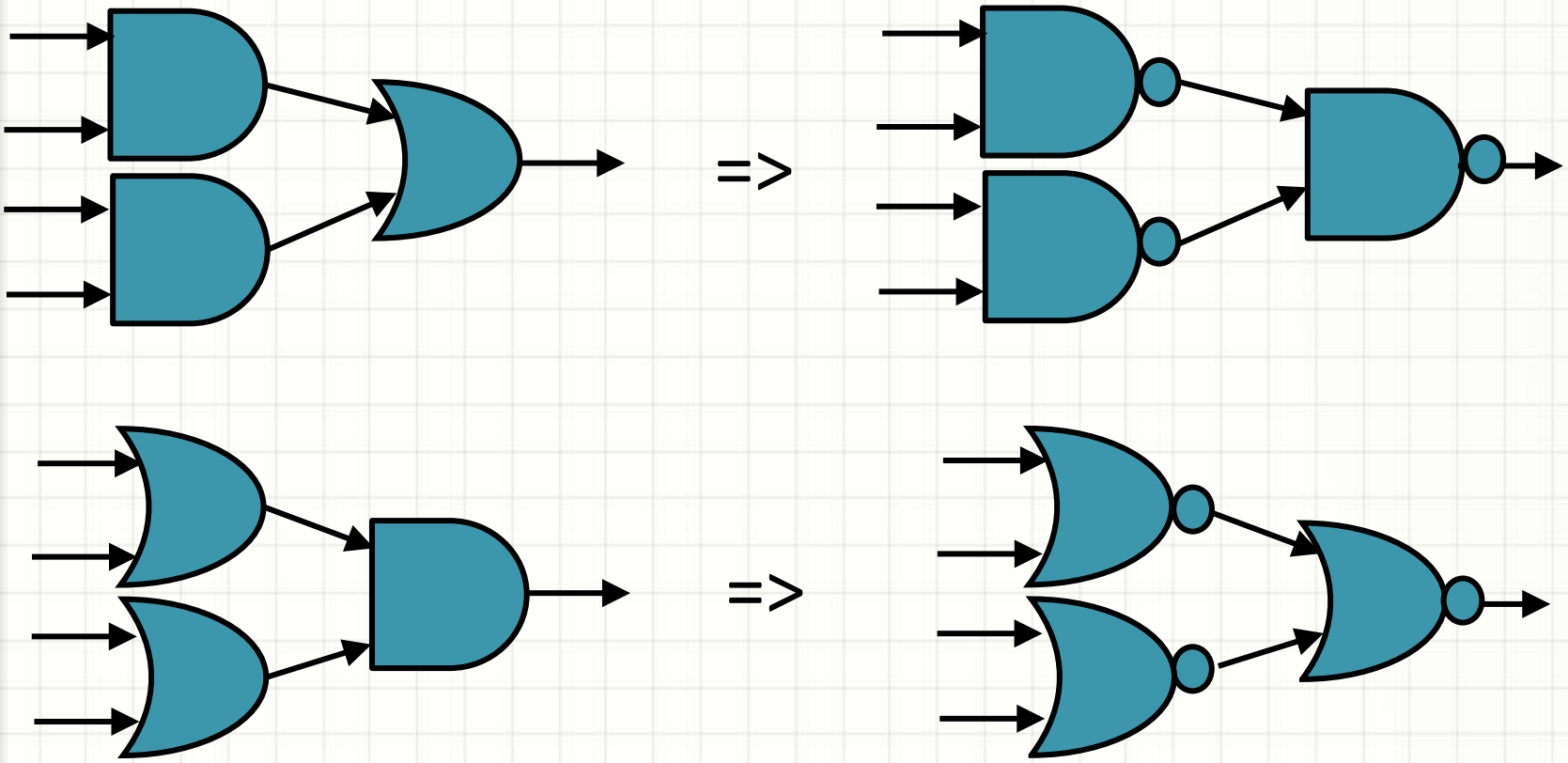
. (a'+ b + c + d) . (a'+ b + c + d')

These are "max terms".

SoP/PoS to Nand-Nand / Nor-Nor



SoP/PoS to Nand-Nand / Nor-Nor



LOGIC MINIMIZATION

Karnaugh Map

a b \ c d	00	01	11	10
00	0	1	1	1
01	0	1	1	1
11	1	1	1	1
10	0	0	1	1

a	b	c	d	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Karnaugh Map

a b \ c d	00	01	11	10
00	0	1	1	1
01	0	1	1	1
11	1	1	1	1
10	0	0	1	1

a’. d

a	b	c	d	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Karnaugh Map

a b \ c d	00	01	11	10
00	0	1	1	1
01	0	1	1	1
11	1	1	1	1
10	0	0	1	1

b . d

a	b	c	d	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

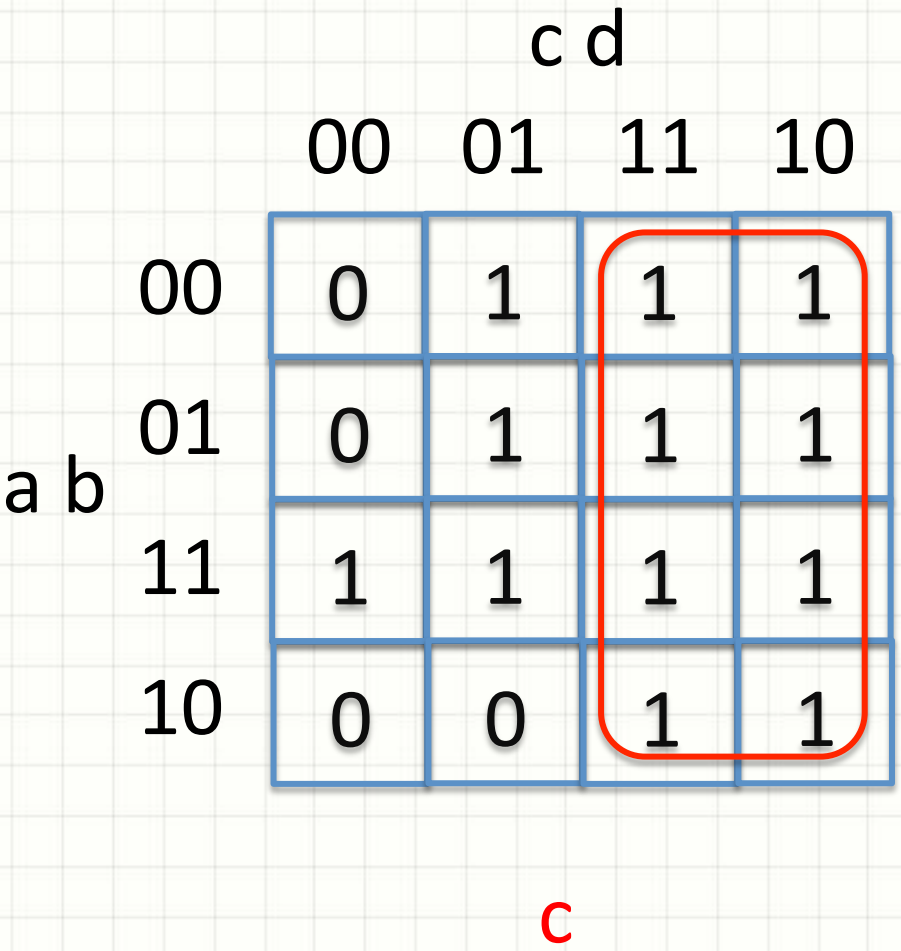
Karnaugh Map

a b \ c d	00	01	11	10
00	0	1	1	1
01	0	1	1	1
11	1	1	1	1
10	0	0	1	1

a . b

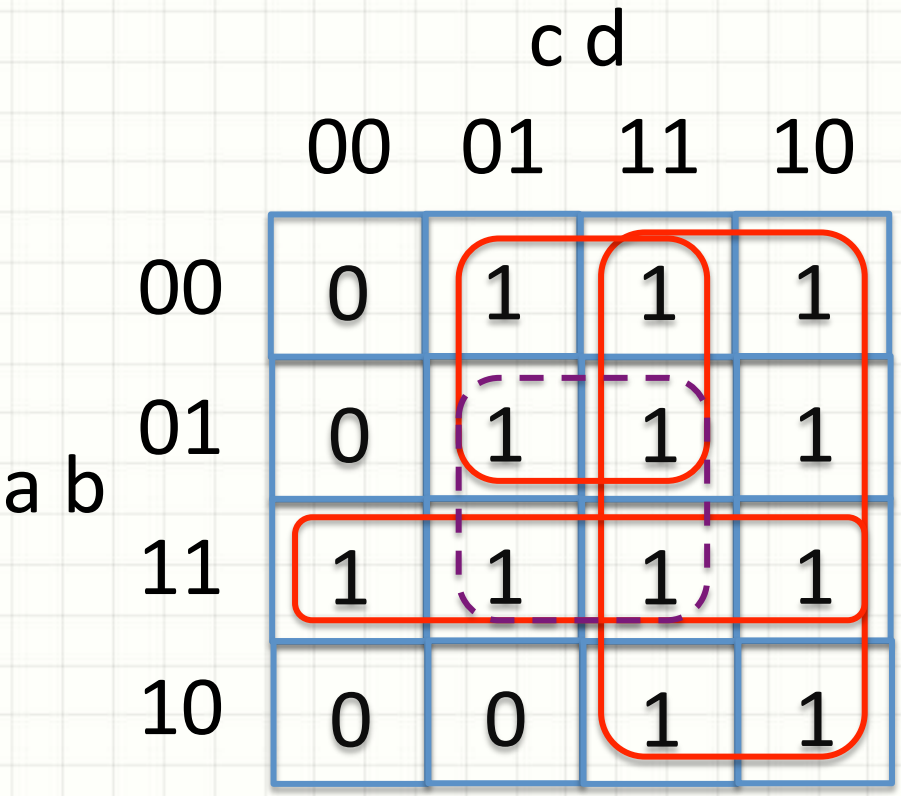
a	b	c	d	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Karnaugh Map



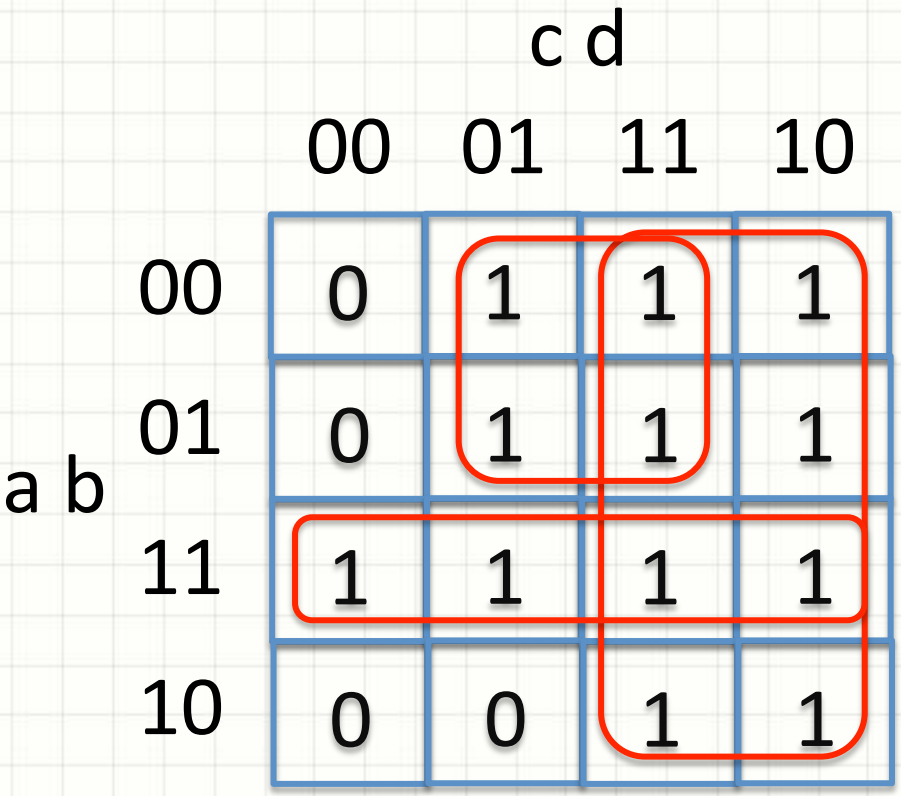
a	b	c	d	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Karnaugh Map



a	b	c	d	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Karnaugh Map



$y = (a \cdot b) + c + (a' \cdot d)$

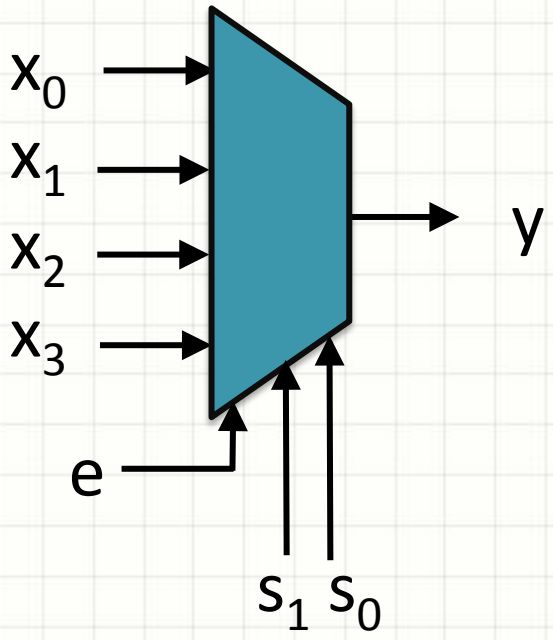
Previously,

$y = (a \cdot b) + (a \cdot c) + (a' \cdot c) + (a' \cdot d)$

a	b	c	d	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Some common combinational logic modules

Multiplexer / Selector

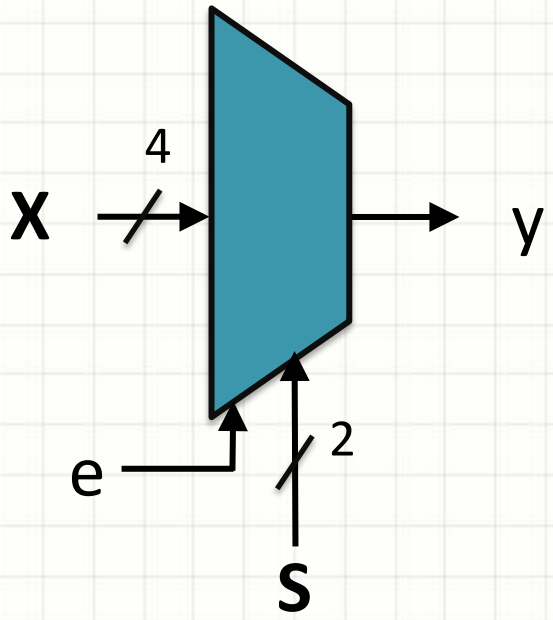


4:1 mux

e	s_1	s_0	y
0	-	-	0
1	0	0	x_0
1	0	1	x_1
1	1	0	x_2
1	1	1	x_3

$$y = e .((s_1' . s_0' . x_0) + (s_1' . s_0 . x_1) + (s_1 . s_0' . x_2) + (s_1 . s_0 . x_3))$$

Multiplexer / Selector

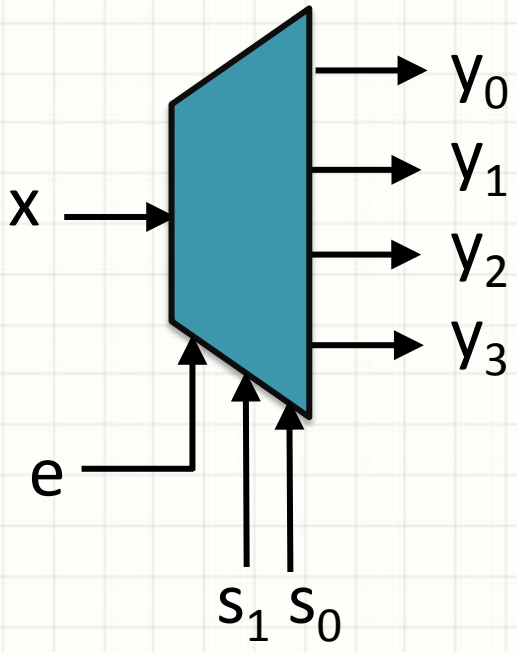


4:1 mux

e	s_1	s_0	y
0	-	-	0
1	0	0	x_0
1	0	1	x_1
1	1	0	x_2
1	1	1	x_3

$$y = e .((s_1' . s_0' . x_0) + (s_1' . s_0 . x_1) + (s_1 . s_0' . x_2) + (s_1 . s_0 . x_3))$$

De-multiplexer



1:4 de-mux

e	s_1	s_0	y_3	y_2	y_1	y_0
0	-	-	0	0	0	0
1	0	0	0	0	0	x
1	0	1	0	0	x	0
1	1	0	0	x	0	0
1	1	1	x	0	0	0

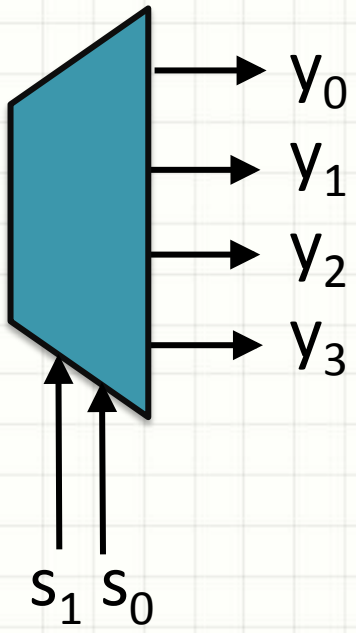
$y_0 = e \,.\, s_1' \,.\, s_0' \,.\, x$

$y_1 = e \,.\, s_1' \,.\, s_0 \,.\, x$

$y_2 = e \,.\, s_1 \,.\, s_0' \,.\, x$

$y_3 = e \,.\, s_1 \,.\, s_0 \,.\, x$

Decoder



2:4 decoder

s_1	s_0	y_3	y_2	y_1	y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

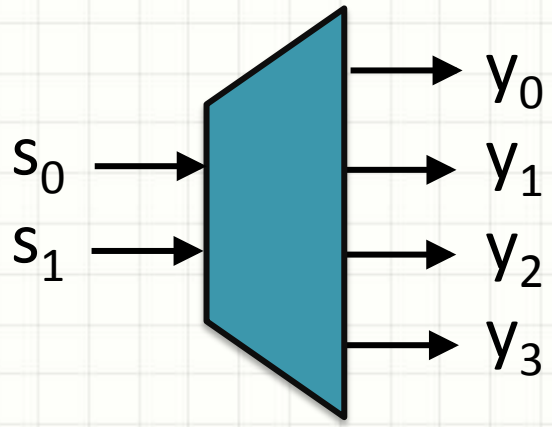
$y_0 = s_1' . s_0'$

$y_1 = s_1' . s_0$

$y_2 = s_1 . s_0'$

$y_3 = s_1 . s_0$

Decoder



2:4 decoder

s_1	s_0	y_3	y_2	y_1	y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

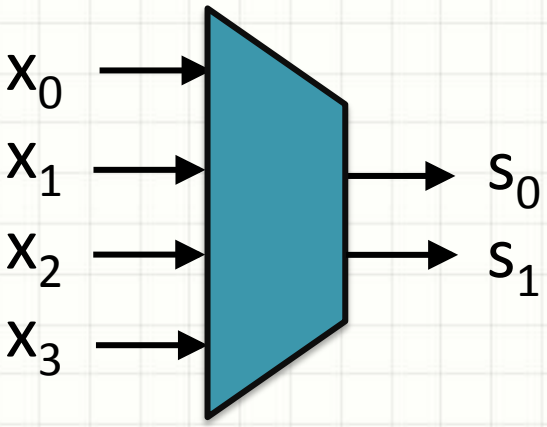
$y_0 = s_1'.s_0'$

$y_1 = s_1'.s_0$

$y_2 = s_1.s_0'$

$y_3 = s_1.s_0$

Encoder

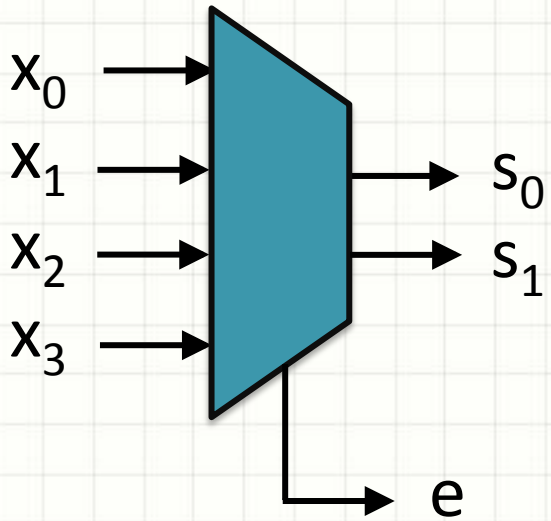


4 input encoder

x_3	x_2	x_1	x_0	s_1	s_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

$$s_0 = (x_3' \cdot x_2' \cdot x_1 \cdot x_0') + (x_3 \cdot x_2' \cdot x_1' \cdot x_0')$$
$$s_1 = (x_3' \cdot x_2 \cdot x_1' \cdot x_0') + (x_3 \cdot x_2' \cdot x_1' \cdot x_0')$$

Priority encoder



4 input priority encoder

x_3	x_2	x_1	x_0	e	s_1	s_0
0	0	0	0	0	-	-
0	0	0	1	1	0	0
0	0	1	-	1	0	1
0	1	-	-	1	1	0
1	-	-	-	1	1	1

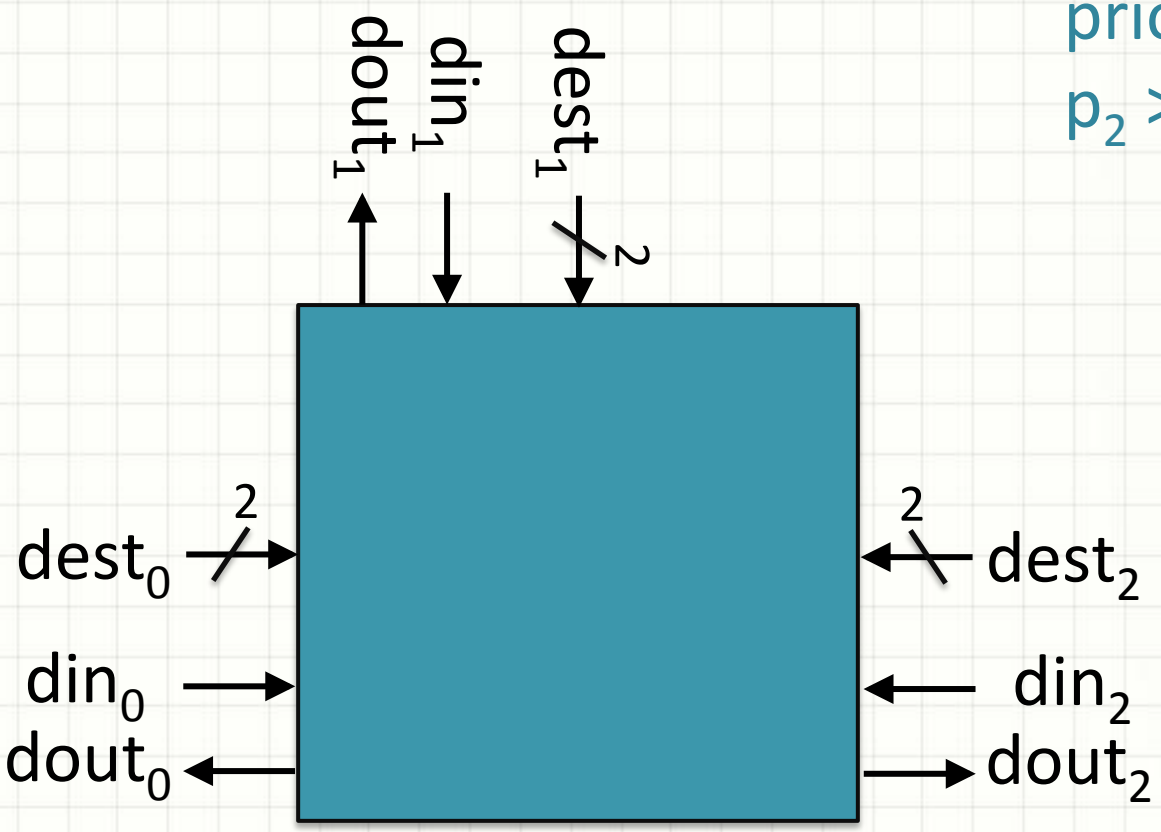
$e = x_3 + x_2 + x_1 + x_0$

$s_0 = (x_3' \cdot x_2' \cdot x_1) + (x_3)$

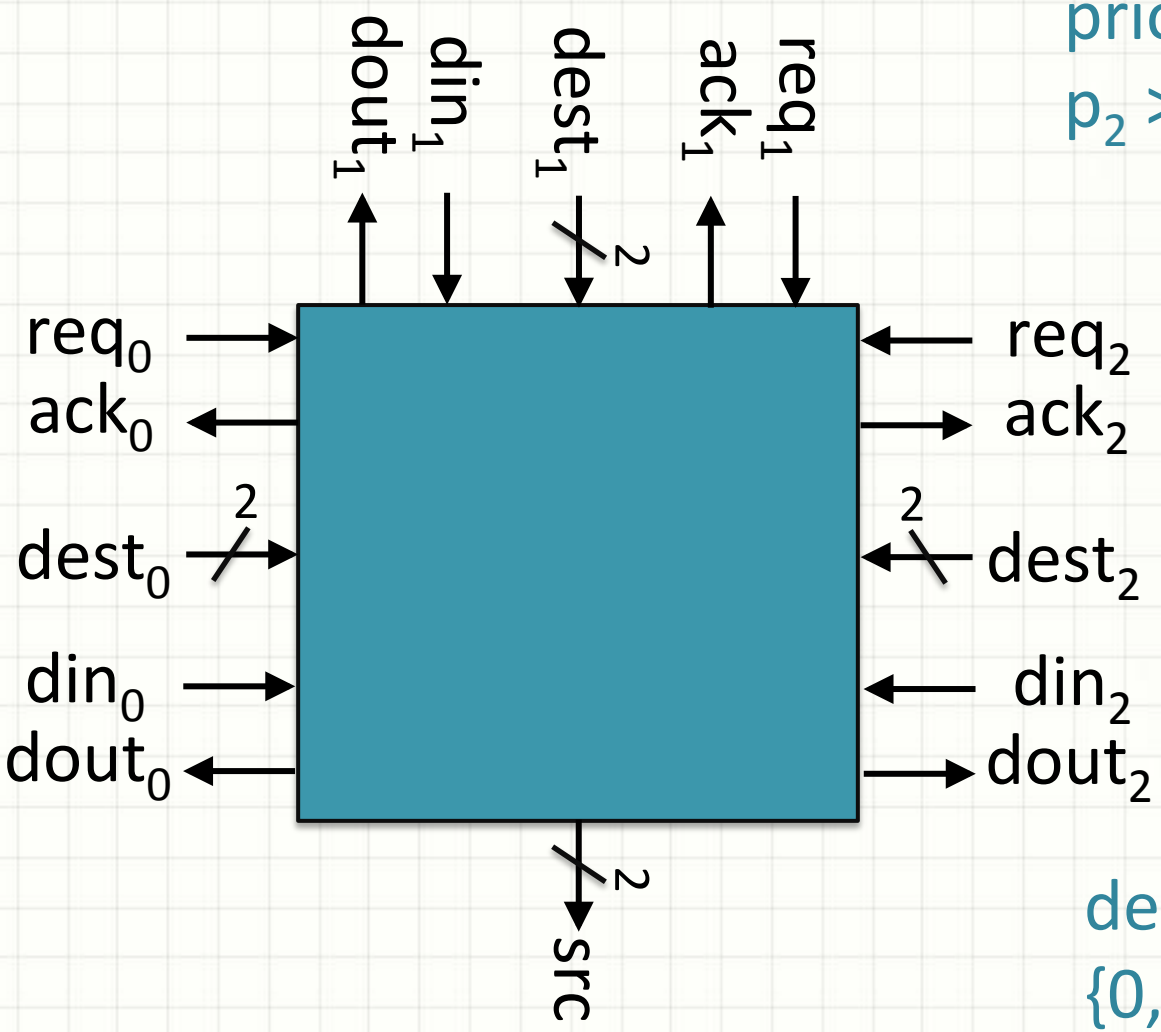
$s_1 = (x_3' \cdot x_2) + (x_3)$

Lab exercise 2 : 3-Port Switch

priority:
$p_2 > p_1 > p_0$



Lab exercise 2 : 3-Port Switch



priority:
$p_2 > p_1 > p_0$

$dest_i$, src:
{0, 1, 2}

QUESTIONS?