EE2000 Logic Circuit Design

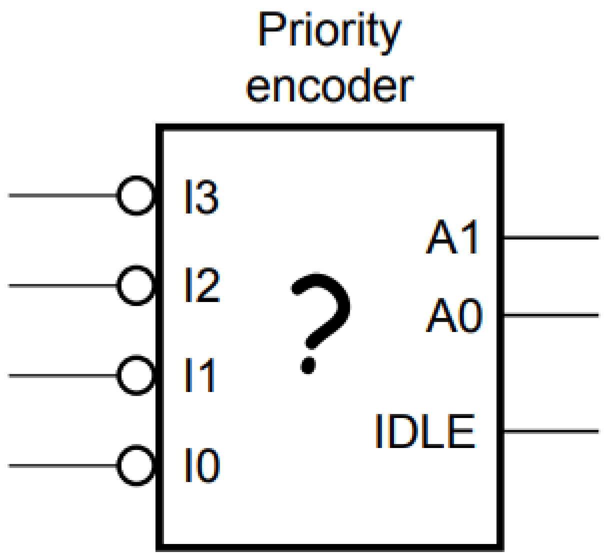
Recap Lecture 4 – Combinational Functional Blocks

electronicscoach.com

Exercise (Active Low)

- Design an Active Low 4-input priority encoder whereby inputs with higher subscript numbers has higher priority.
- Output IDLE is High when all inputs are high.

Input				Output		
I_3	I_2	I_1	I_0	A_1	A_0	IDLE
1	1	1	1	X	X	1
1	1	1	0	0	0	0
1	1	0	X	0	1	0
1	0	X	X	1	0	0
0	X	X	X	1	1	0



Exercise (Active Low)

Input				Output		
I_3	I_2	I_1	I_0	A_1	A_0	IDLE
1	1	1	1	X	X	1
1	1	1	0	0	0	0
1	1	0	X	0	1	0
1	0	X	X	1	0	0
0	X	X	X	1	1	0

$$\mathrm{IDLE} = I_3 I_2 I_1 I_0$$

$$A_0 = I_3' + I_3 I_2 I_1'$$

$$= I_3' + I_2 I_1'$$

$$A_1 = I_3 I_2' + I_3'$$

$$= I_2' + I_3'$$

Simplification
a’+ ab = a’ + b

Exercise (Active Low)

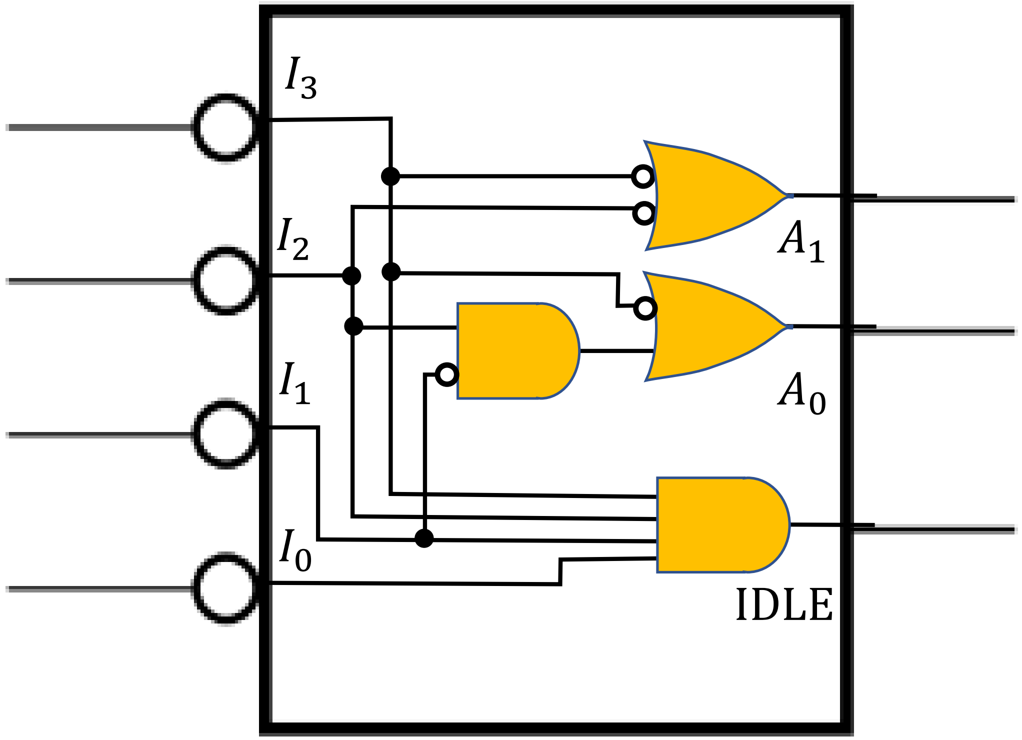
$\mathrm{IDLE} = I_3 I_2 I_1 I_0$

$A_0 = I_3' + I_3 I_2 I_1'$

$= I_3' + I_2 I_1'$

$A_1 = I_3 I_2' + I_3'$

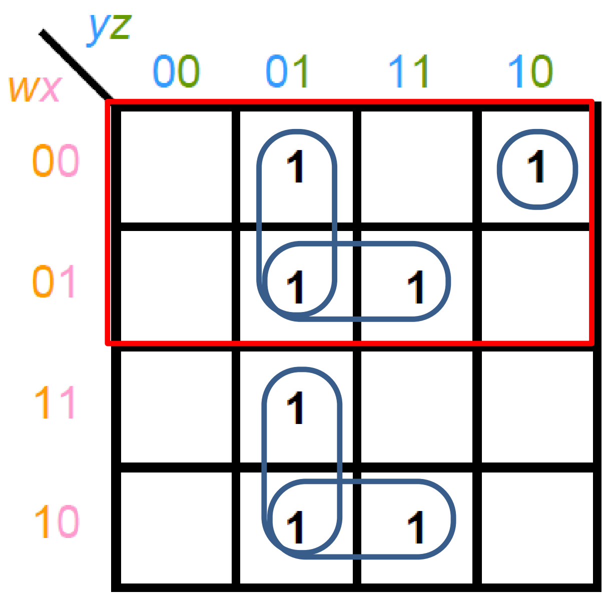
$= I_2' + I_3'$



Exercise

Realize the function $f(w, x, y, z) = \sum m(1, 2, 5, 7, 9, 11, 13)$ using a 2-to-1 MUX

$$w = S_0$$



$$f(w = 0) = y'z + xz + x'yz'$$

$$f(w = 1) = y'z + x'z$$

Exercise

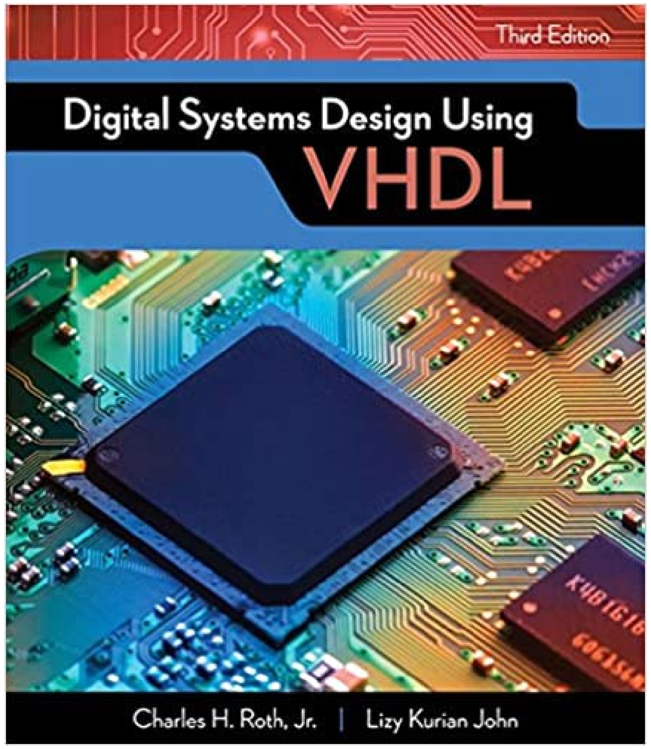
Realize the function $f(w, x, y, z) = \sum m(1, 2, 5, 7, 9, 11, 13)$ using a 8-to-1 MUX

wxyz	*F*	$S_2=w$, $S_1=x$, $S_0=y$
0000	0	$I_0=z$
0001	1	
0010	1	$I_1=z'$
0011	0	
0100	0	$I_2=z$
0101	1	
0110	0	$I_3=z$
0111	1	

wxyz	*F*	$S_2=w$, $S_1=x$, $S_0=y$
1000	0	$I_4=z$
1001	1	
1010	0	$I_5=z$
1011	1	
1100	0	$I_6=z$
1101	1	
1110	0	$I_7=0$
1111	0	

EE2000 Logic Circuit Design

Lecture 5- VHDL


Third Edition
Digital Systems Design Using
VHDL
Charles H. Roth, Jr. | Lizy Kurian John

Question

Which of the following identifier is illegal to be used as an entity name in VHDL?

• TwO_gaTE	Legal
• 2_gate	Illegal – cannot start with a number
• T2-gate	Illegal – cannot have other symbol
• _2gate	Illegal – cannot start with underscore
• AND	Illegal – Reserved word

Exercise

```
signal C: bit_vector (0 to 3);
signal D: bit_vector (3 downto 0);
signal A: bit_vector (7 downto 0);
```

```
C <= "1101";
D <= C;
A(6 downto 3) <= D;
```

Determine the stored value in the following bit object

C(0) = 1	C(1) = 1	C(2) = 0	C(3) = 1
D(3) = 1	D(2) = 1	D(1) = 0	D(0) = 1
A(6) = 1	A(5) = 1	A(4) = 0	A(3) = 1

Exercise

Transform the following logic expression to VHDL code

1) f = ab’ + a’b

```
f <= a AND NOT b OR NOT a AND b;

f <= (a AND NOT b) OR (NOT a AND b);
```

2) f = a(b’ + a’)b

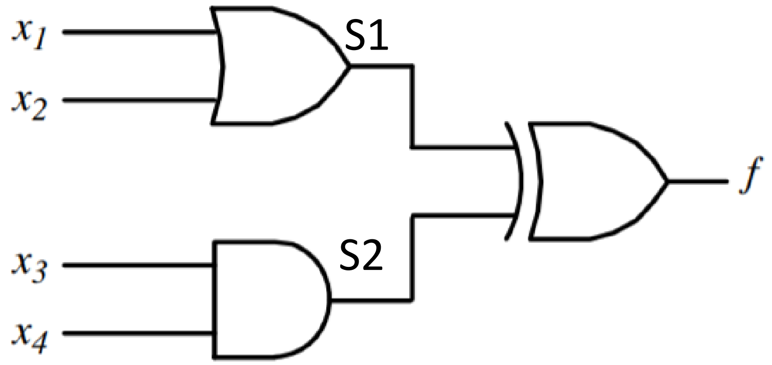
```
f <= a AND (NOT b OR NOT a) AND b;
```

Exercise

Transform the following logic expression to VHDL code

Boolean	VHDL Boolean
$Y = \overline{AB}$	`Y <= NOT(A AND B);` or `Y <= A NAND B;`
$Y = \overline{A + B}$	`Y <= NOT(A OR B);` or `Y <= A NOR B;`
$Y = A + BC$	`Y <= A OR (B AND C);`
$Y = C\overline{X + D}$	`Y <= C AND NOT (X OR D);`
$Y = A\bar{B}C + \bar{A}\,\bar{B}C + \overline{AB}C$	`Y <= (A AND NOT B AND C) OR (NOT A AND NOT B AND C) OR (NOT (A AND B) AND C);`

Exercise



Write your own VHDL statement to describe the logic circuit

```
S1 <= x1 OR x2;
S2 <= x3 AND x4;
F <= S1 XOR S2;
```

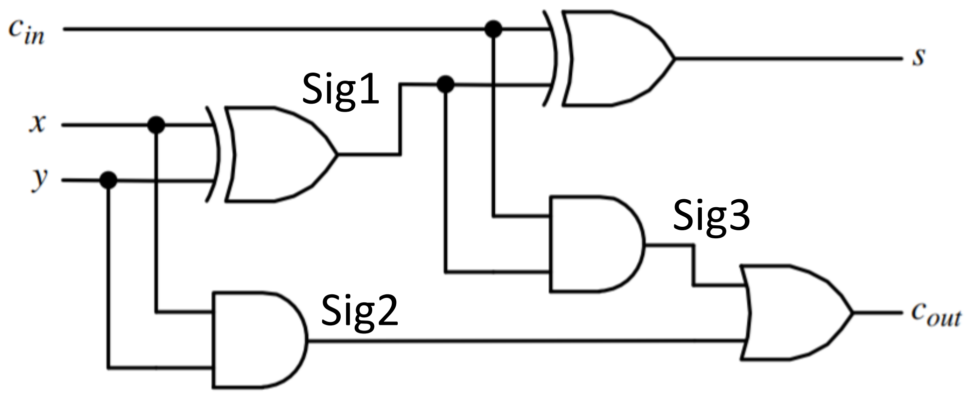
OR

```
F <= (x1 OR x2) XOR (x3
AND x4);
```

- The Left-hand style is more readable and traceable but needs to declare two more signals
- In contrast, the Right-hand style uses fewer lines of code but is a bit difficult for debugging
- They both have the same result after being synthesized

Exercise

Write your own VHDL statement to describe the logic circuit



Sig1 <= x XOR y;
Sig2 <= x AND y;
Sig3 <= Sig1 AND C_{in};
s <= C_{in} XOR Sig1;
C_{out} <= Sig3 OR Sig2;