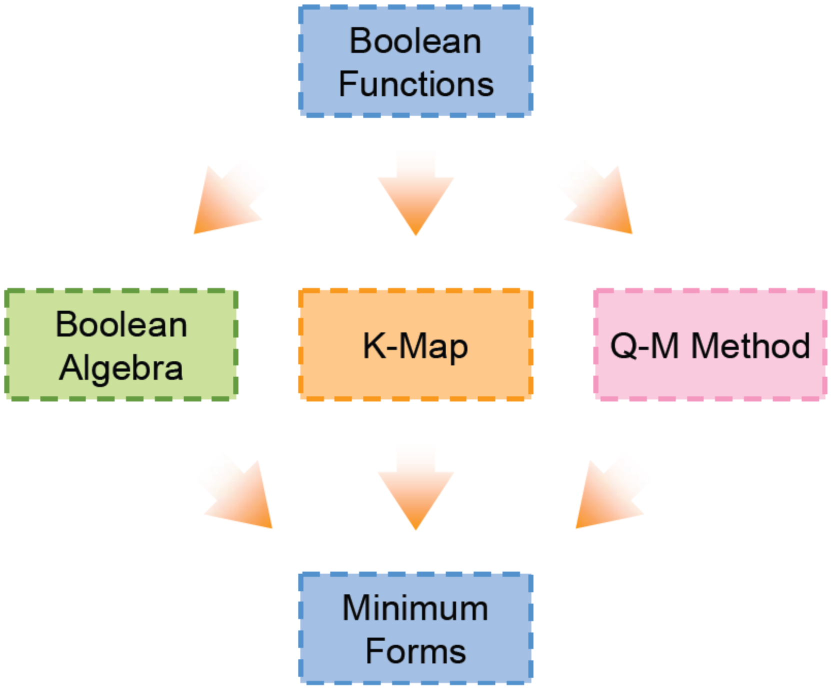
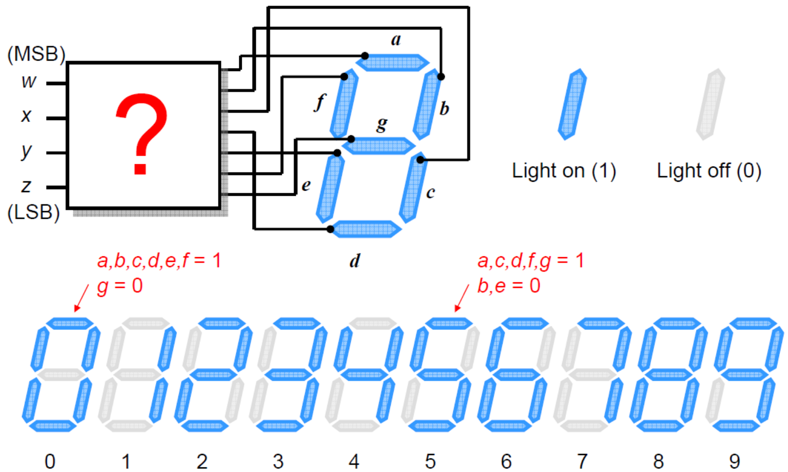
EE2000 Logic Circuit Design

Lecture 4 – Combinational Functional Blocks



Formulation



numbers	Inputs				7-segment display						
	w	x	y	z	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	x	x	x	x	x	x	x
11	1	0	1	1	x	x	x	x	x	x	x
12	1	1	0	0	x	x	x	x	x	x	x
13	1	1	0	1	x	x	x	x	x	x	x
14	1	1	1	0	x	x	x	x	x	x	x
15	1	1	1	1	x	x	x	x	x	x	x

State the case

Design a circuit to convert the BCD 8421 to the 7-segment display

- *w, x, y, z* are the input.
- *a, b, c, d, e, f, g* are the output.

$$a\,(w,x,y,z) = \Sigma\, m\,(0,2,3,5,6,7,8,9) + \Sigma d(10,11,12,13,14,15)$$

K-map for segment ‘e’

cd \ ab	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

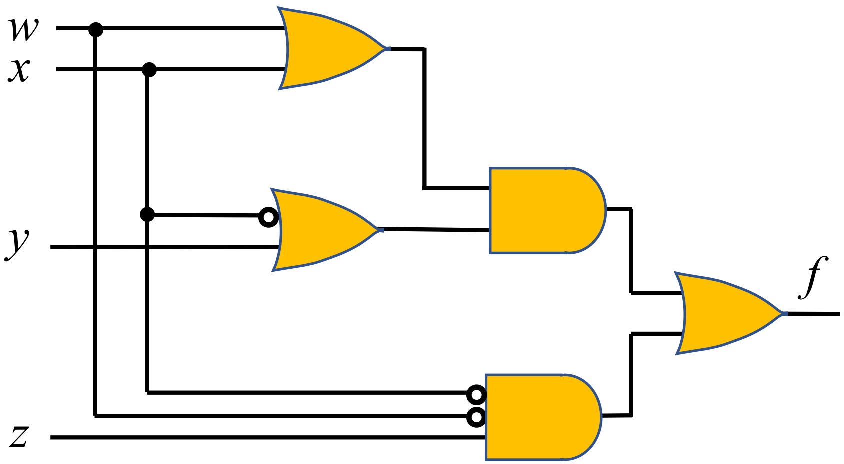
$$e\,(\,w, x, y, z) = \,\Sigma m(0,2,6,8)$$
$$+\Sigma d(10, 11, 12, 13, 14, 15)$$

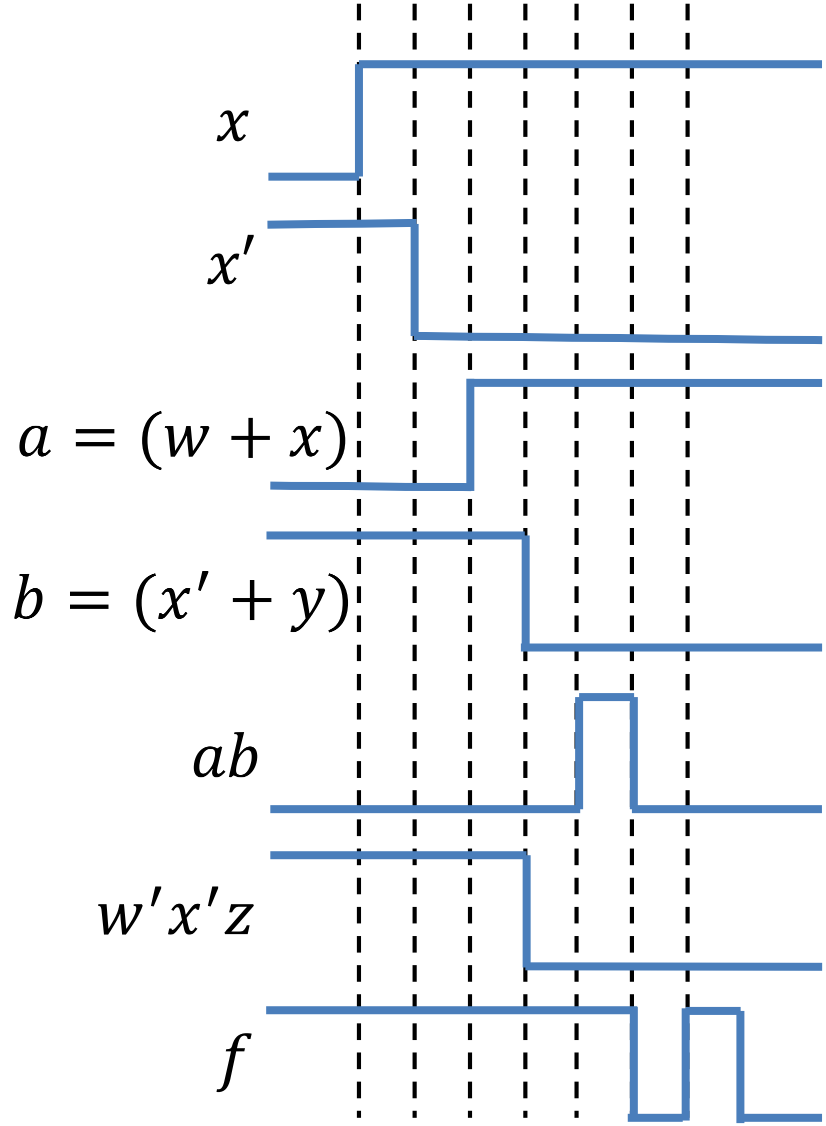
yz \ wx	00	01	11	10
00	1		x	1
01			x	
11			x	x
10	1	1	x	x

yz \ wx	00	01	11	10
00	1		x	1
01			x	
11			x	x
10	1	1	x	x

$$e(w, x, y, z) = x'z' + yz'$$

Exercise





Assume that the propagation delay of NOT gate is $\Delta\tau$ and $2\Delta\tau$ for others .

Work out the timing diagram to identify the presence of any timing hazard when the input condition changes from $(w, x, y, z) = (0,0,0,1)$ to $(0,1,0,1)$.

Dynamic hazard!!!

Exercise

Given $\boldsymbol{f\,(a,\,b,\,c) = \Sigma\, m\,(0,2,4,5)}$

c \ ab	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

a) Minimize the function *f*

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

c \ ab	00	01	11	10
0	1*	1		1
1				1*

$$f(a,b,c) = a'c' + ab'$$

Exercise

c \ ab	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

Given $f\ (a, b, c) = \Sigma\ m\ (0,2,4,5)$

b) Realize *f to a* hazard-free circuit

c \ ab	00	01	11	10
0	1*	1		1
1				1*

$$f(a,b,c) = a'c' + ab'$$

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

$$f(a,b,c) = a'c' + ab' + b'c'$$

Exercise

The following block of code is received based on an even parity, identify the errors and generate the corrected data.

1	0	1	0	0
0	0	1	1	1
1	0	0	0	1
0	0	0	1	1
1	1	0	0	0
1	0	0	0	

1	0	1	0	0
0	1	1	1	0
1	0	1	0	1
0	0	0	1	1
1	1	0	0	0
1	0	0	0	

Exercise

1	0	1	0	0
0	0	1	1	1
1	0	0	0	1
0	0	0	1	1
1	1	0	0	0
1	0	0	0	

1	0	1	0	0
0	1	1	1	1
1	0	0	0	1
0	0	0	1	1
1	1	0	0	0
1	0	0	0	

1	0	1	0	0
0	1	1	1	0
1	0	1	0	1
0	0	0	1	1
1	1	0	0	0
1	0	0	0	

1	0	1	0	0
0	1	1	1	1
1	0	0	0	1
0	0	0	1	1
1	1	0	0	0
1	0	0	0	

or

1	0	1	0	0
0	1	0	1	0
1	0	1	0	0
0	0	0	1	1
1	1	0	0	0
1	0	0	0	

Exercise

Determine the Hamming code using both odd and even parity bit for a data code of 11100

Step 1: Calculate extra bit (***k***) needed for a ***n*** bit of code.

Step 2: Place Parity Bits in the positions of powers of 2.

Hamming Code	H_9	H_8	H_7	H_6	H_5	H_4	H_3	H_2	H_1
Bit									

Exercise

Determine the Hamming code using both odd and even parity bit for a data core of 11100

For a 5-bit data $d_5 d_4 d_3 d_2 d_1$, *n* = 5

$$2^k \geq 6 + k$$

Therefore, minimum value of *k* is 4. We need **4 parity bits**!

Exercise

Step 2: Place Parity Bits in the positions of powers of 2.

Hamming Code	H_9	H_8	H_7	H_6	H_5	H_4	H_3	H_2	H_1
	d_5	p_4	d_4	d_3	d_2	p_3	d_1	p_2	p_1
Bit	9	**8**	7	6	5	**4**	3	**2**	**1**

Exercise

Step 2: Place Parity Bits in the positions of powers of 2.

Hamming Code	H_9	H_8	H_7	H_6	H_5	H_4	H_3	H_2	H_1
	d_5	p_4	d_4	d_3	d_2	p_3	d_1	p_2	p_1
Bit	9	8	7	6	5	4	3	2	1
Binary Code	1001	1000	0111	0110	0101	0100	0011	0010	0001
p_1									
p_2									
p_3									
p_4									
Even Parity									
Odd Parity									

Step 3: Calculate the number of ‘1’ in each parity bits

Step 4: Place ‘1’ if odd number of ‘1’ for even parity; else ‘0’; Place ‘0’ if odd number of ‘1’ for odd parity; else ‘1’

Exercise

Step 2: Place Parity Bits in the positions of powers of 2.

Hamming Code	H_9	H_8	H_7	H_6	H_5	H_4	H_3	H_2	H_1
	d_5	p_4	d_4	d_3	d_2	p_3	d_1	p_2	p_1
Bit	9	8	7	6	5	4	3	2	1
Binary Code	1001	1000	0111	0110	0101	0100	0011	0010	0001
p_1	1		1		0		0		
p_2			1	1			0		
p_3			1	1	0				
p_4	1								
Even Parity	1	1	1	1	0	0	0	0	0
Odd Parity	1	0	1	1	0	1	0	1	1

Step 3: Calculate the number of ‘1’ in each parity bits

Step 4: Place ‘1’ if odd number of ‘1’ for even parity; else ‘0’; Place ‘0’ if odd number of ‘1’ for odd parity; else ‘1’

Exercise

Example: data $d_4d_3d_2d_1$ = 1000

Hamming Code	H_7	H_6	H_5	H_4	H_3	H_2	H_1
	d_4	d_3	d_2	p_3	d_1	p_2	p_1
Bit	7	6	5	4	3	2	1
Binary Code	0111	0110	0101	0100	0011	0010	0001
p_1	1		0		0		
p_2	1	0			0		
p_3	1	0	0				
Even Parity	1	0	0	1	0	1	1
Odd Parity	1	0	0	0	0	0	0

Consider odd parity and if we receive a code of 1001000, check the parity bits

$$c_1 = (H_7 \oplus H_5 \oplus H_3 \oplus H_1)' = (1 \oplus 0 \oplus 0 \oplus 0)' = 0$$
$$c_2 = (H_7 \oplus H_6 \oplus H_3 \oplus H_2)' = (1 \oplus 0 \oplus 0 \oplus 0)' = 0$$
$$c_3 = (H_7 \oplus H_6 \oplus H_5 \oplus H_4)' = (1 \oplus 0 \oplus 0 \oplus 1)' = 1$$

$$c_3c_2c_1 = (100)_2 = 4$$

Exercise

$$\begin{array}{lrrl} & & 0 & x \\ & & 0 & y \\ (+) & & 0 & c_{in} \\ \hline & 0 & 0 & \\ & c_o & s & \end{array} \quad \begin{array}{lrrl} & & 0 & x \\ & & 1 & y \\ (+) & & 0 & c_{in} \\ \hline & 0 & 1 & \\ & c_o & s & \end{array} \quad \begin{array}{lrrl} & & 1 & x \\ & & 0 & y \\ (+) & & 0 & c_{in} \\ \hline & 0 & 1 & \\ & c_o & s & \end{array} \quad \begin{array}{lrrl} & & 1 & x \\ & & 1 & y \\ (+) & & 0 & c_{in} \\ \hline & 1 & 0 & \\ & c_o & s & \end{array}$$

$$\begin{array}{lrrl} & & 0 & x \\ & & 0 & y \\ (+) & & 1 & c_{in} \\ \hline & 0 & 1 & \\ & c_o & s & \end{array} \quad \begin{array}{lrrl} & & 0 & x \\ & & 1 & y \\ (+) & & 1 & c_{in} \\ \hline & 1 & 0 & \\ & c_o & s & \end{array} \quad \begin{array}{lrrl} & & 1 & x \\ & & 0 & y \\ (+) & & 1 & c_{in} \\ \hline & 1 & 0 & \\ & c_o & s & \end{array} \quad \begin{array}{lrrl} & & 1 & x \\ & & 1 & y \\ (+) & & 1 & c_{in} \\ \hline & 1 & 1 & \\ & c_o & s & \end{array}$$

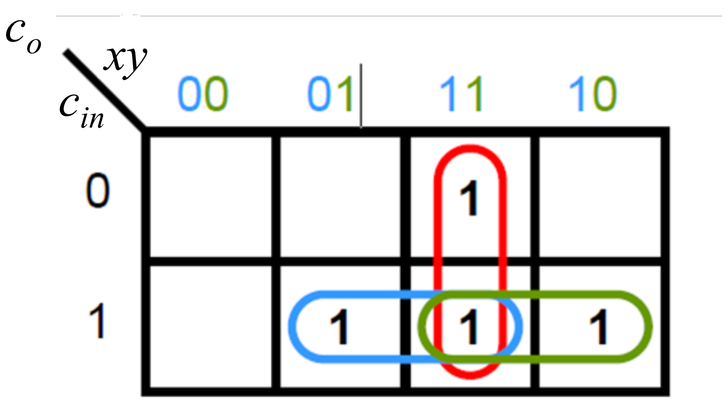
Inputs			Outputs	
x	y	c_{in}	c_o	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

a) Work out the algebraic functions of s and c_o using K-map

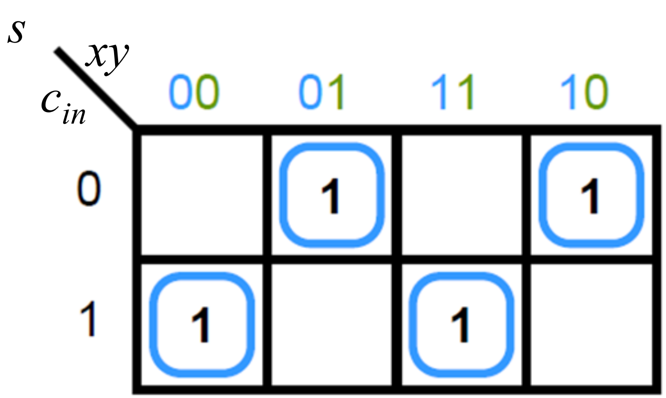
b) Draw the logic circuit diagram of full adder

$c \backslash ab$	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

Exercise



$$c_o = xy + xc_{in} + yc_{in} = xy + c_{in}(x + y)$$
$$= xy + c_{in}(xy' + x'y + xy)$$
$$= xy(1 + c_{in}) + c_{in}(xy' + x'y)$$
$$= xy + c_{in}(x \oplus y)$$

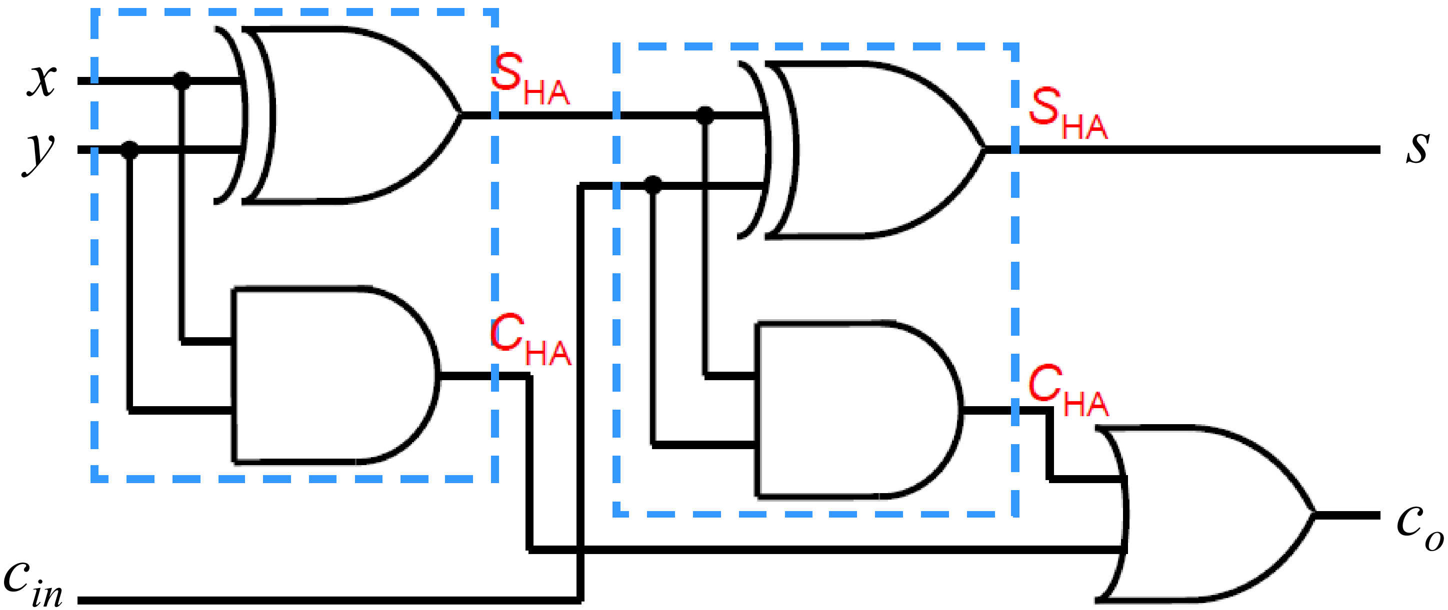


$$s = x'y'c_{in} + xyc_{in} + x'yc_{in}' + xy'c_{in}'$$
$$= c_{in}(x'y' + xy) + c_{in}'(x'y + xy')$$
$$= c_{in}(x \oplus y)' + c_{in}'(x \oplus y)$$
$$= c_{in} \oplus (x \oplus y)$$

Exercise

$c \backslash ab$	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

$c_o = xy + c_{in}(x \oplus y)$ $\quad$ $s = c_{in} \oplus (x \oplus y)$



Exercise

$$\begin{array}{cccc} b_o & 0 & 0 & x \\ (-) & & 0 & y \\ (-) & & 0 & b_{in} \\ \hline & & 0 & \\ & & d & \end{array} \quad \begin{array}{cccc} b_o & 1 & 0 & x \\ (-) & & 1 & y \\ (-) & & 0 & b_{in} \\ \hline & & 1 & \\ & & d & \end{array} \quad \begin{array}{cccc} b_o & 0 & 1 & x \\ (-) & & 0 & y \\ (-) & & 0 & b_{in} \\ \hline & & 1 & \\ & & d & \end{array} \quad \begin{array}{cccc} b_o & 0 & 1 & x \\ (-) & & 1 & y \\ (-) & & 0 & b_{in} \\ \hline & & 0 & \\ & & d & \end{array}$$

$$\begin{array}{cccc} b_o & 1 & 0 & x \\ (-) & & 0 & y \\ (-) & & 1 & b_{in} \\ \hline & & 1 & \\ & & d & \end{array} \quad \begin{array}{cccc} b_o & 1 & 0 & x \\ (-) & & 1 & y \\ (-) & & 1 & b_{in} \\ \hline & & 0 & \\ & & d & \end{array} \quad \begin{array}{cccc} b_o & 0 & 1 & x \\ (-) & & 0 & y \\ (-) & & 1 & b_{in} \\ \hline & & 0 & \\ & & d & \end{array} \quad \begin{array}{cccc} b_o & 1 & 1 & x \\ (-) & & 1 & y \\ (-) & & 1 & b_{in} \\ \hline & & 1 & \\ & & d & \end{array}$$

Inputs			Outputs	
x	y	b_{in}	b_o	d
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

a) Work out the algebraic functions of *d* and b_o using K-map

b) Draw the logic circuit diagram of full subtractor

c \ ab	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

Exercise

c \ ab	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

b_o

b_{in} \ xy	00	01	11	10
0		1		
1	1	1	1	

$$b_o = x'y + x'b_{in} + yb_{in} = x'y + b_{in}(x' + y)$$
$$= x'y + b_{in}(x'y' + x'y + xy)$$
$$= x'y(1 + b_{in}) + b_{in}(x'y' + xy)$$
$$= x'y + b_{in}(x \oplus y)'$$

d

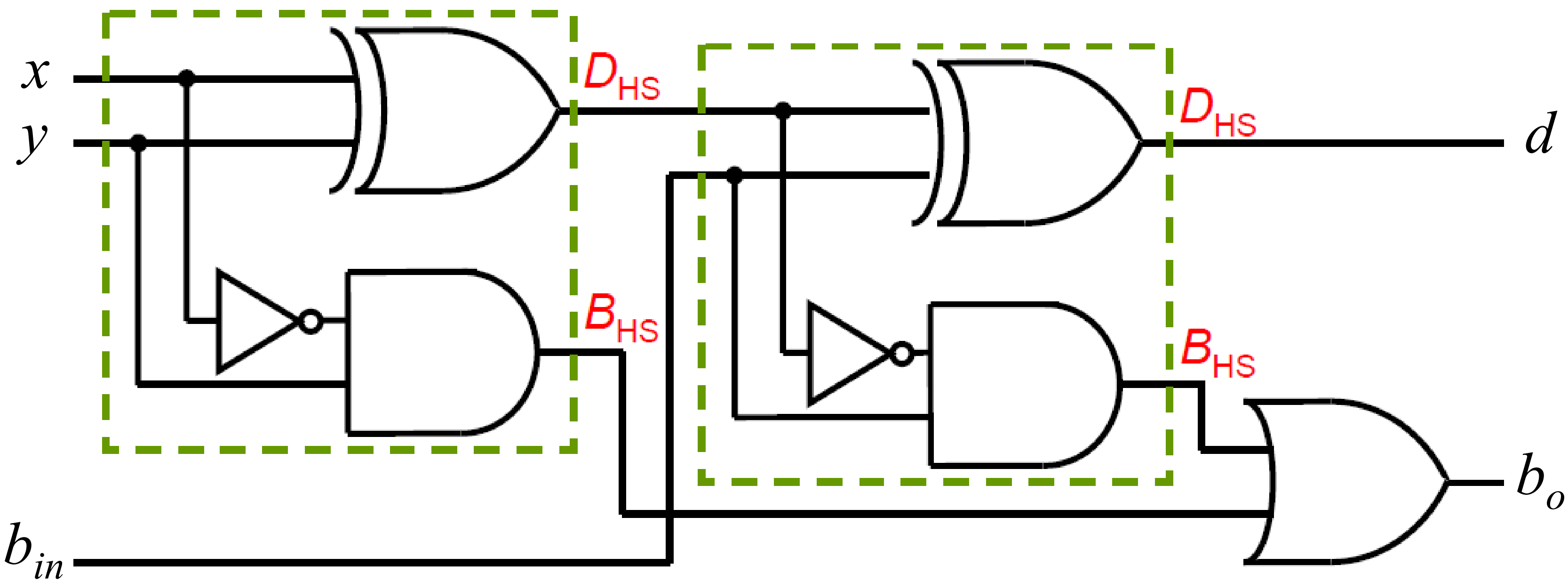
b_{in} \ xy	00	01	11	10
0		1		1
1	1		1	

$$d = x'y'b_{in} + xyb_{in} + x'yb'_{in} + xy'b'_{in}$$
$$= b_{in}(x'y' + xy) + b'_{in}(x'y + xy')$$
$$= b_{in}(x \oplus y)' + b'_{in}(x \oplus y)$$
$$= b_{in} \oplus (x \oplus y)$$

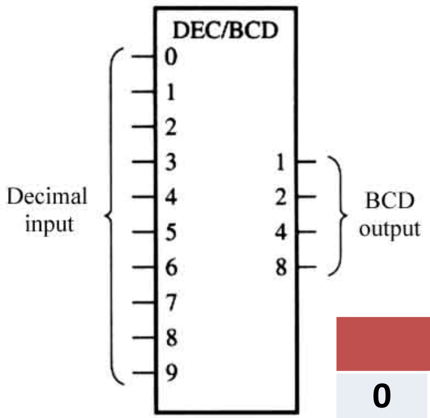
Exercise

$c \backslash ab$	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

$b_o = x'y + b_{in}(x \oplus y)' \quad d = b_{in} \oplus (x \oplus y)$



Exercise (Decimal-to-Binary Encoder)



Inputs										Outputs			
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	A_3	A_2	A_1	A_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

Exercise (Decimal-to-Binary Encoder)

Inputs										Outputs			
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	A_3	A_2	A_1	A_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1