	sum	car
0+0 =	0	0
0+1/1+0 =	1	0
1+1	0	1

Unit-4

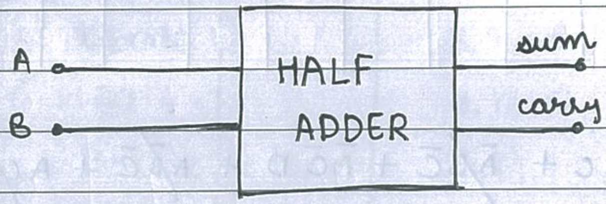
combinational circuits

combinational circuits

combinational circuits are those digital circuits in which output of the circuit depends only on the present input. There is no need of feedback ~~in~~ ~~circuits~~ and memory in combinational circuits.

eg- half adder, encoder, decoder, subtracter, multiplexer, demultiplexer.

1) HALF ADDER



A	B	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

for sum

A \ B	0	1
$\bar{A}$ 0	0 (0)	1 (1)
A 1	1 (2)	0 (3)

$\Rightarrow$ sum = $\bar{A}B + A\bar{B}$

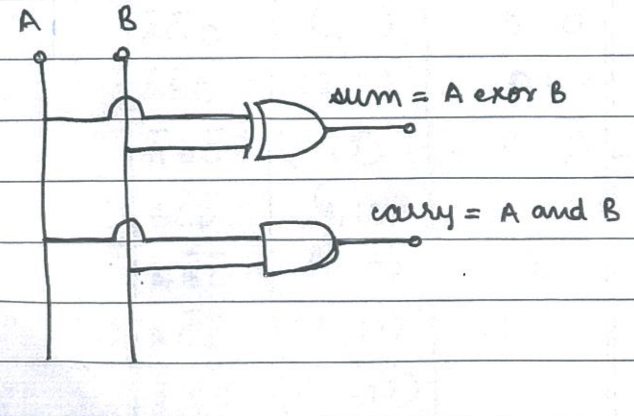
$\Rightarrow$ sum = A exor B

for carry

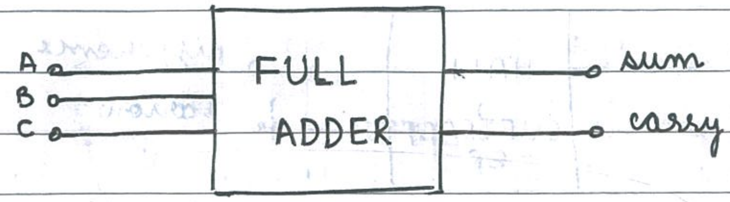
A \ B	0	1
0	0 (0)	0 (1)
1	0 (2)	1 (3)

carry = A B

carry = A and B



2) FULL ADDER



A	B	C	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

for sum

A,B \ C	0	1	
00	0 (0)	1 (1)	$\bar{A}\bar{B}C$
01	1 (2)	0 (3)	$\bar{A}B\bar{C}$
11	0 (6)	1 (7)	ABC
10	1 (4)	0 (5)	$A\bar{B}\bar{C}$

for carry

A,B \ C	0	1	
00	0 (0)	0 (1)	
01	0 (2)	1 (3)	BC
11	1 (6)	1 (7)	AB
10	0 (4)	1 (5)	AC

sum = $\bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C}$

sum = $\bar{A}(\bar{B}C + B\bar{C}) + A(BC + \bar{B}\bar{C})$

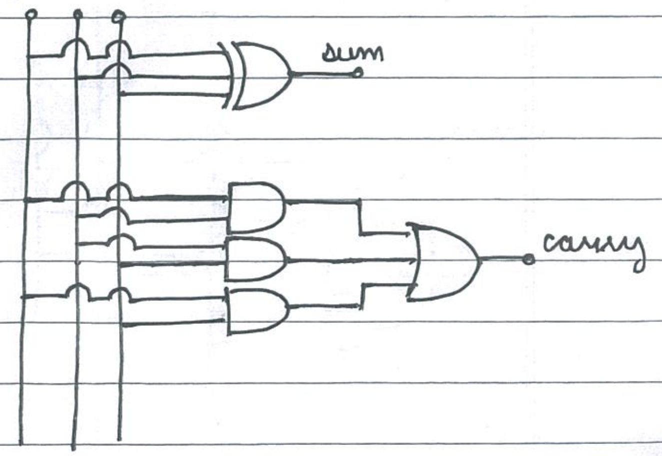
sum = $\bar{A}$ (B exor C) + A (B exnor C)

let (B exor C = x, B exnor = $\bar{x}$)

sum = $\bar{A}x + A\bar{x}$

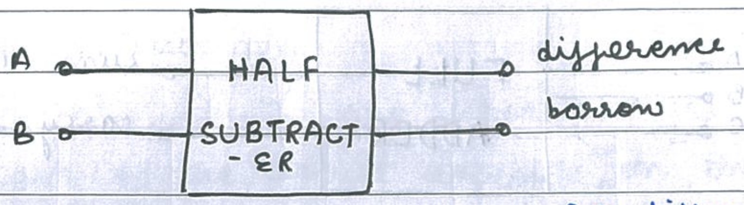
sum = A exor x = A exor B exor C

carry = AC + AB + BC



0	0	0	B
0	1	1	0
1	0	1	0
1	1	0	0

iii) Half Subtracter



A	B	diff.	borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

For difference

A \ B	0	1	
0	0 (0)	1 (1)	— $\bar{A}B$
1	1 (2)	0 (3)	— $A\bar{B}$

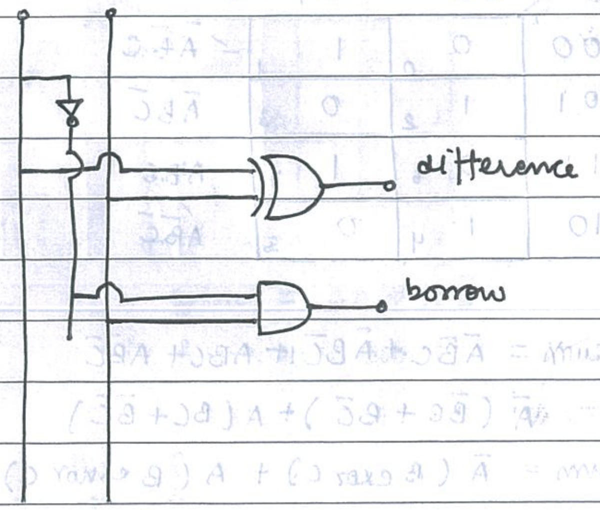
difference = $\bar{A}B + A\bar{B}$

= A exor B

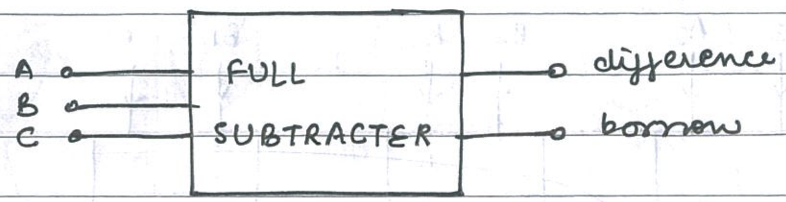
For ~~diff~~ borrow

A \ B	0	1	
0	0 (0)	1 (1)	$\bar{A}B$
1	0 (2)	0 (3)	

Borrow = $\bar{A}B$



iv) Full Subtracter



A	B	C	diff.	borr-
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

For difference

AB \ C	0	1	
00	0 (0)	1 (1)	— $\bar{A}\bar{B}C$
01	1 (2)	0 (3)	— $\bar{A}B\bar{C}$
11	0 (6)	1 (7)	— ABC
10	1 (4)	0 (5)	— $A\bar{B}\bar{C}$

For borrow

A,B \ C	0	1	
00	0 (0)	1 (1)	$\bar{A}C$
01	1 (2)	1 (3)	$\bar{A}B$
11	0 (6)	1 (7)	BC
10	0 (4)	0 (5)	

diff = $\bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C}$

= $\bar{A}(\bar{B}C + B\bar{C}) + A(BC + \bar{B}\bar{C})$

= $\bar{A}$ (B exor C) + A (B exnor C)

= let B exor C = x

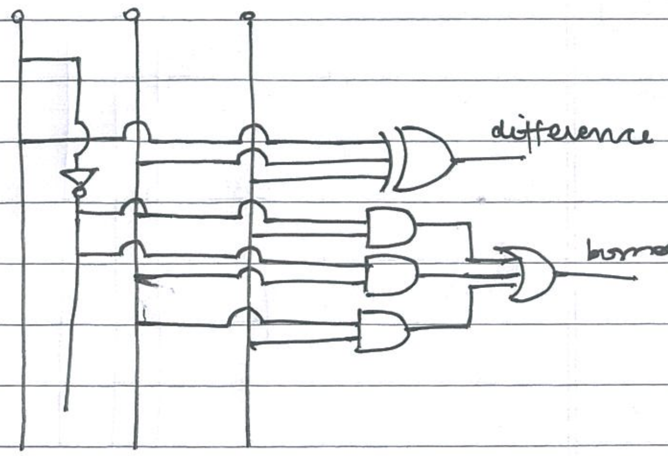
B exnor C = $\bar{x}$

diff = $\bar{A}x + A\bar{x}$

= A exnor x

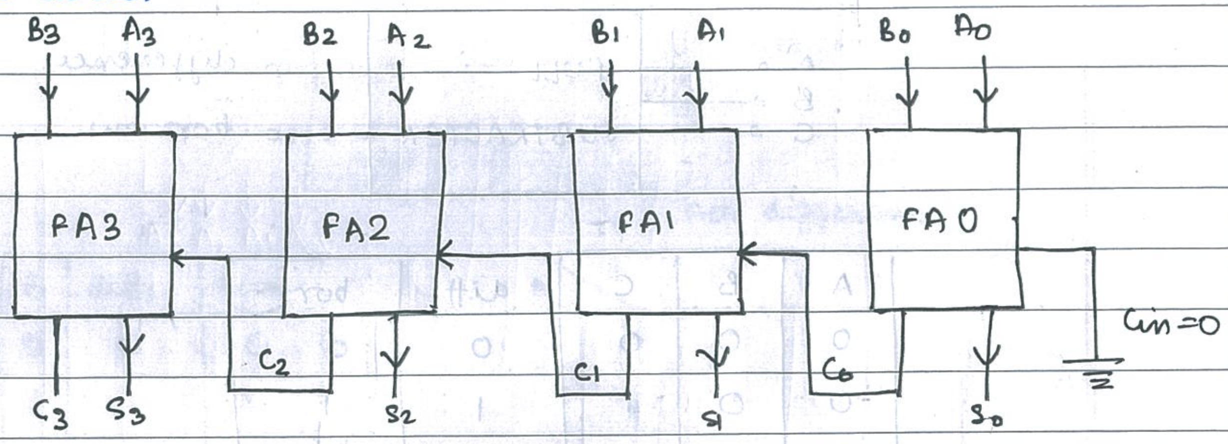
= A exor B exor C

borrow = $\bar{A}C + \bar{A}B + BC$



v) Parallel Adder

4 bit adder



		① ① ①		C_2 C_1 C_0 $C_{in}=0$		
A	=	1 0 1 1	=	A_3	A_2 A_1	A_0
B	=	1 1 0 1		B_3	B_2 B_1	B_0
S	=	① 1 0 0 0		C_3 S_3	S_2 S_1	S_0

i) 4 bit Subtracter

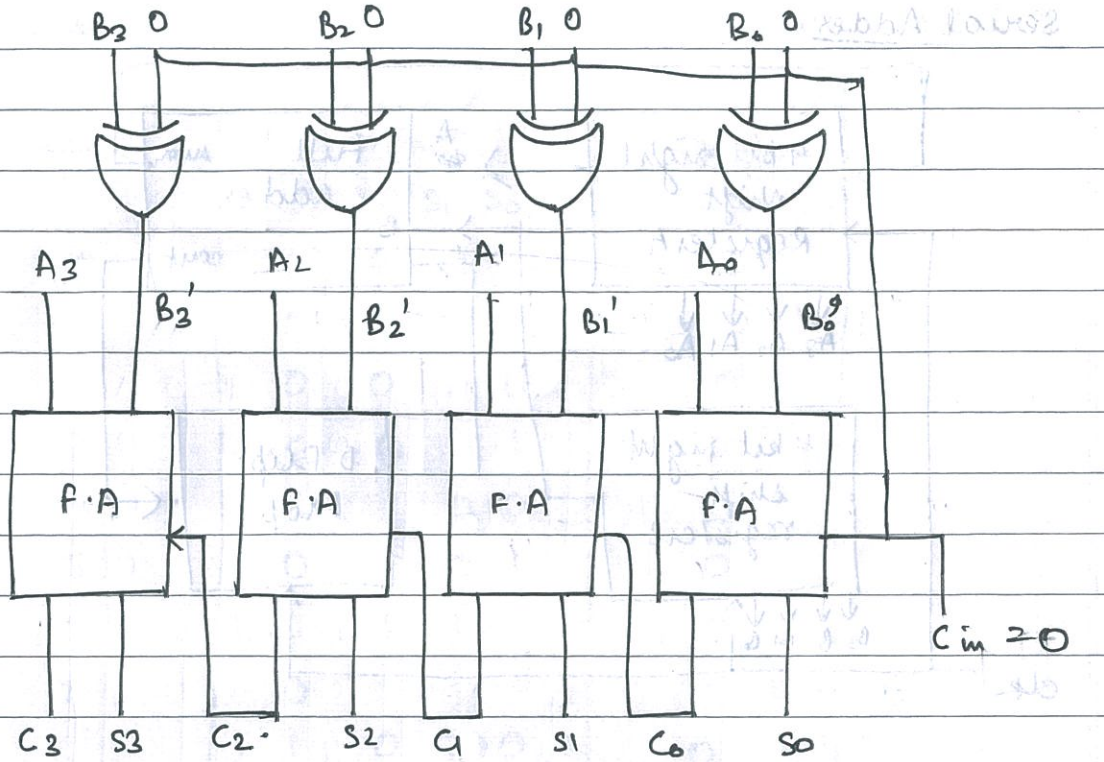
$A = A_3 \; A_2 \; A_1 \; A_0$

$B = B_3 \; B_2 \; B_1 \; B_0$

$A - B = (A)_2 + (-B)_2$

$= A + (B)_{2\text{'s complement}}$

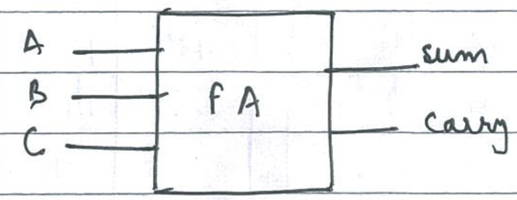
$(B)_{2\text{'s complement}} = B_3' \; B_2' \; B_1' \; B_0'$

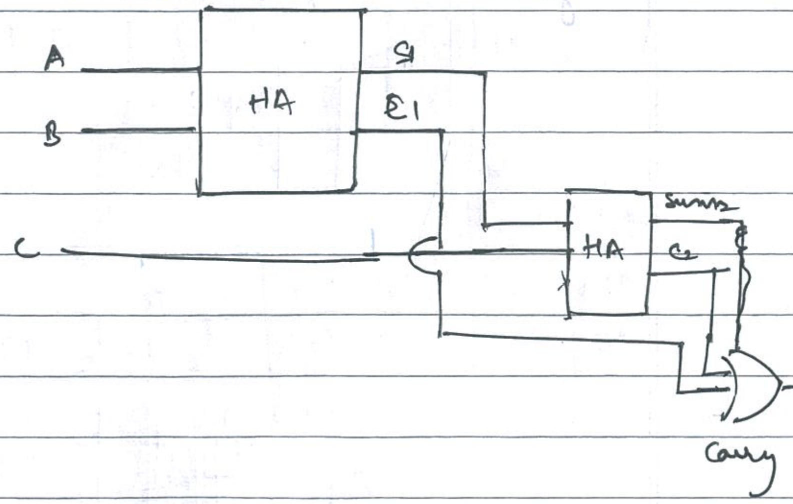


Cin = 1 4 bit Subtracter

0 4 bit Adder

Carry ~~over~~ look ahead adder ~~or~~ full adder using Half Adder

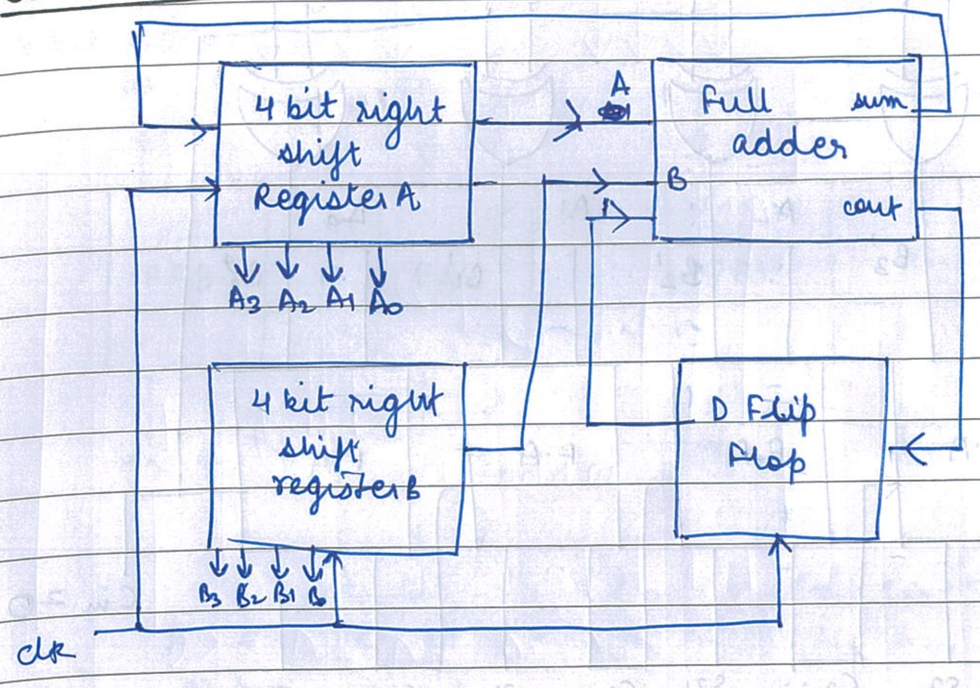




$A + B = S_1$

$C + S_1 = \text{sum}$

Serial Adder



The circuit diagram shows the serial adder. We can add numbers which are stored in a right shift register A and B. The full adder is used to perform bit-by-bit addition and a flip flop is used to store carry output generated after addition.

BCD Adder

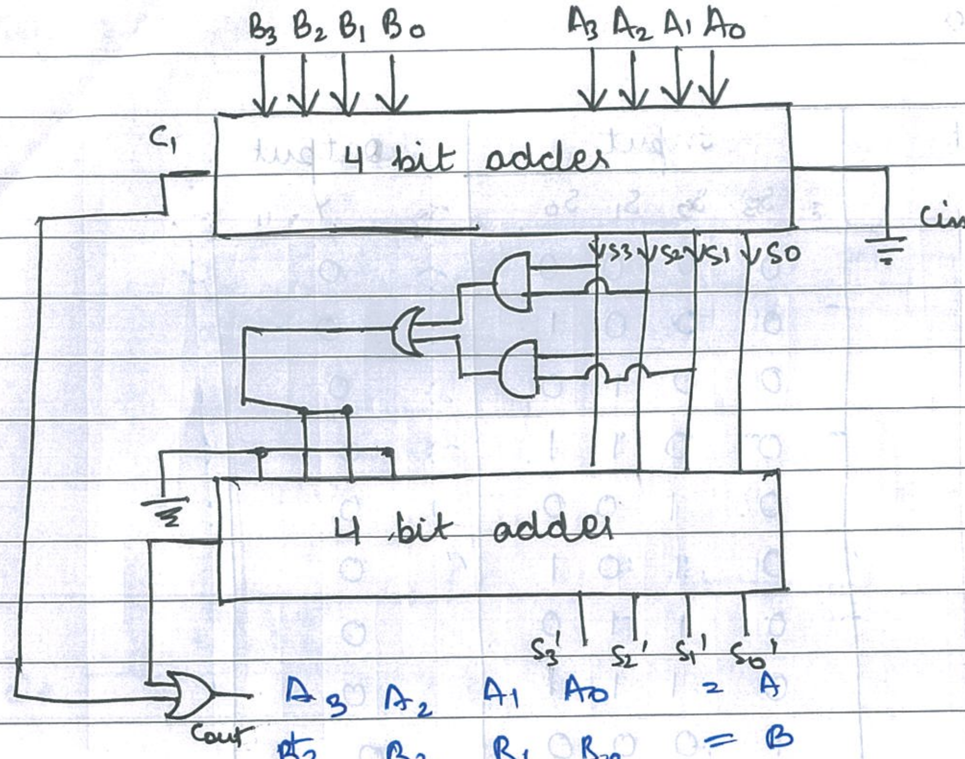
Digit	Input S_3	S_2	S_1	S_0	Output Y
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

S_2,S_3 \ S_1,S_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

S_3S_2

S_1S_3

$$Y = S_3S_2 + S_3S_1$$



$A_3 \; A_2 \; A_1 \; A_0 = A$

$B_3 \; B_2 \; B_1 \; B_0 = B$

$S_3 \; S_2 \; S_1 \; S_0 = S$

(1) $S > 9$ 6 add = 0110 $Y = 1$

$S < 9$ 0 add = 0000 $Y = 0$

G_2

B_3,B_2 \ B_1,B_0	00	01	11	10	
00	0 0	0 1	0 3	0 2	
01	1 4	1 5	1 7	1 6	$\overline{B_3}, B_2$
11	0 12	0 13	0 15	0 14	
10	1 8	1 9	1 11	1 10	$B_3, \overline{B_2}$

$G_2 = \overline{B_3}B_2 + B_2\overline{B_3}$

$= B_3$ exor B_2

G_1

B_3,B_2 \ B_1,B_0	00	01	11	10
00	0 0	0 1	1 3	1 2
01	1 4	1 5	0 7	0 6
11	1 12	1 13	0 15	0 14
10	0 8	0 9	1 11	1 10

$G_1 = B_2\overline{B_1} + \overline{B_2}B_1$

$G_1 = B_2$ exor B_1

Binary to Gray Code Convertor

Digit	Input B_3	B_2	B_1	B_0	Output G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Kmap for G_3

B_3,B_2 \ B_1,B_0	00	01	11	10	
00	0 0	0 1	0 3	0 2	
01	0 4	0 5	0 7	0 6	
11	1 12	1 13	1 15	1 14	B_3
10	1 8	1 9	1 11	1 10	

$G_3 = B_3$

G_0

B_3,B_2 \ B_1,B_0	00	01	11	10
00	0 0	1 1	0 3	1 2
01	0 4	1 5	0 7	1 6
11	0 12	1 13	0 15	1 14
10	0 8	1 9	0 11	1 10

$\overline{B_1}B_0$ $\quad$ $B_1\overline{B_0}$

$G_0 = \overline{B_1}B_0 + B_1\overline{B_0}$

$= B_1$ exor B_0