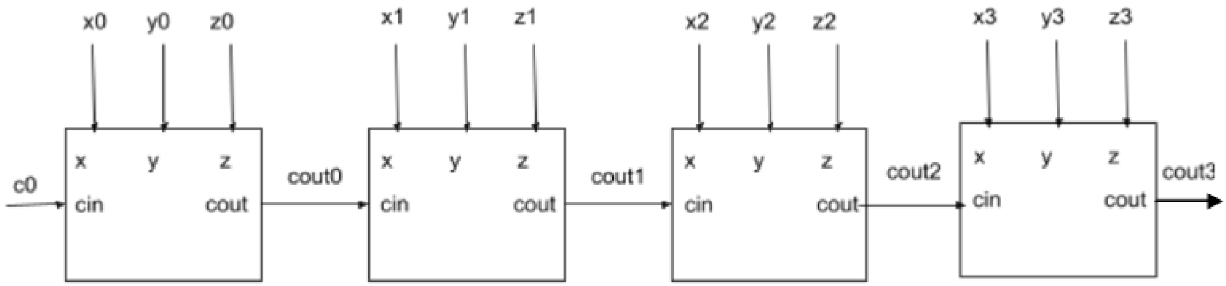
HOMEWORK 13-14 SOLUTIONS

1)



Final output = Cout3

Initialize C0 to 1

Cout = (X'.Y'.Z' + X.Y.Z) .Cin

2)

Truth table for Half-Subtractor:

A	**B**	**Bout**	**Diff**
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

From the above truth table we arrive at the following expressions for borrow and difference:

Borrow = Bout = A'B

Difference = Diff = A XOR B

3)

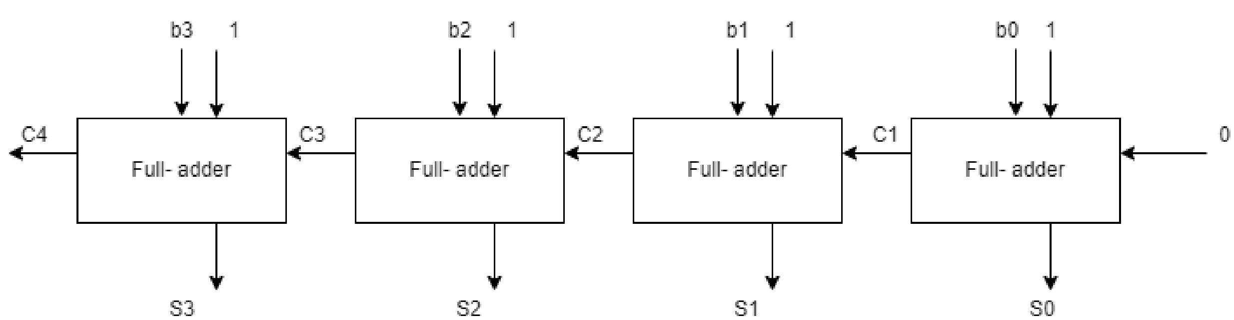
Truth table for 3-input majority function:

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Y = AB + BC + AC
= Cout of a full-adder
We observe that the carry-out of a full-adder circuit equals the output of a majority function.

4)
A four-bit decrementer subtracts 1 from a 4-bit number.
Subtracting 1 from 4-bit number = 4-bit number + (-1)
(-1) in 4-bit 2's complement form = 1111.

Let the 4-bit number be b3 b2 b1 b0.



Carry = C4 C3 C2 C1
Sum = S3 S2 S1 S0

5)

From the truth table, we observe that binary representation of number of ones in input = binary number with Cout of full-adder as MSB and Sum as LSB.

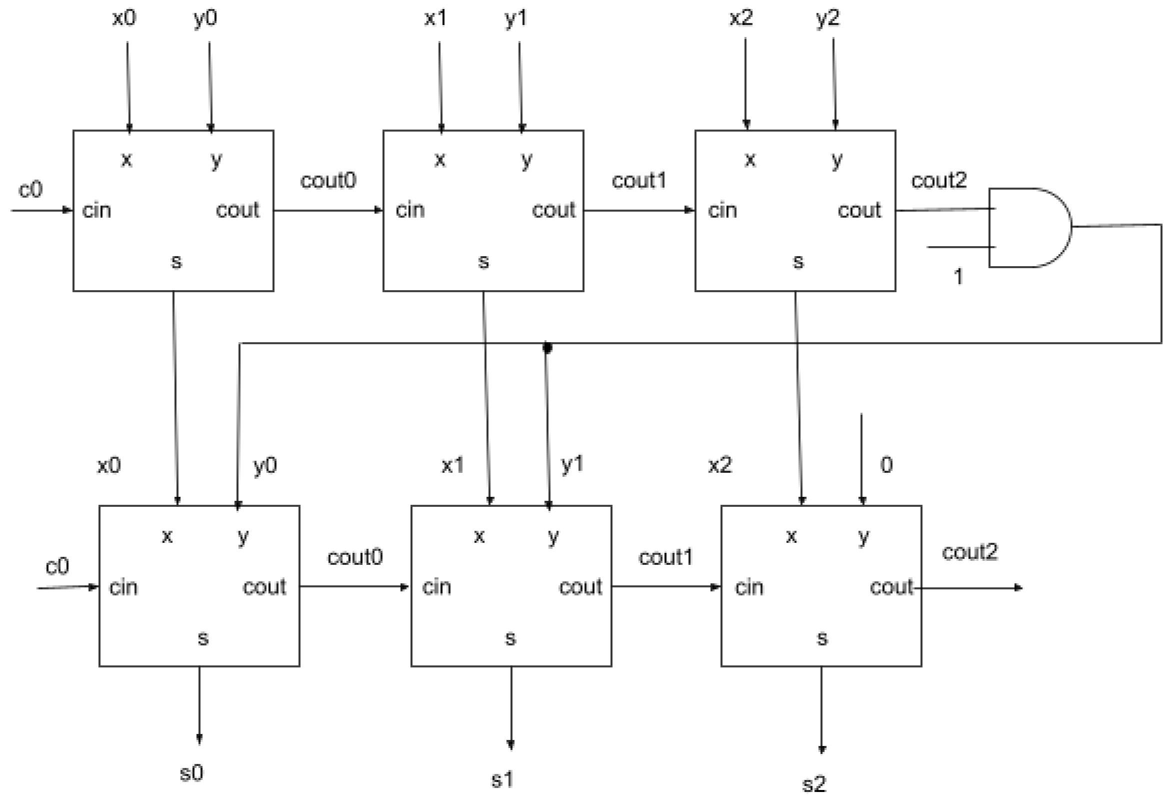
A	**B**	**C**	**Cout**	**Sum**	**Cout-Sum**	**No. of 1s in input**
0	0	0	0	0	00(0)	0
0	0	1	0	1	01(1)	1
0	1	0	0	1	01(1)	1
0	1	1	1	0	10(2)	2
1	0	0	0	1	01(1)	1
1	0	1	1	0	10(2)	2
1	1	0	1	0	10(2)	2
1	1	1	1	1	11(3)	3

6) Worst case delay = delay for the carries to propagate from least significant adder to most-significant adder + delay for the sum to be generated in the most significant adder

= (16-1)*12 + 15 ns

= 195 ns

7)



Each block represents a full adder.

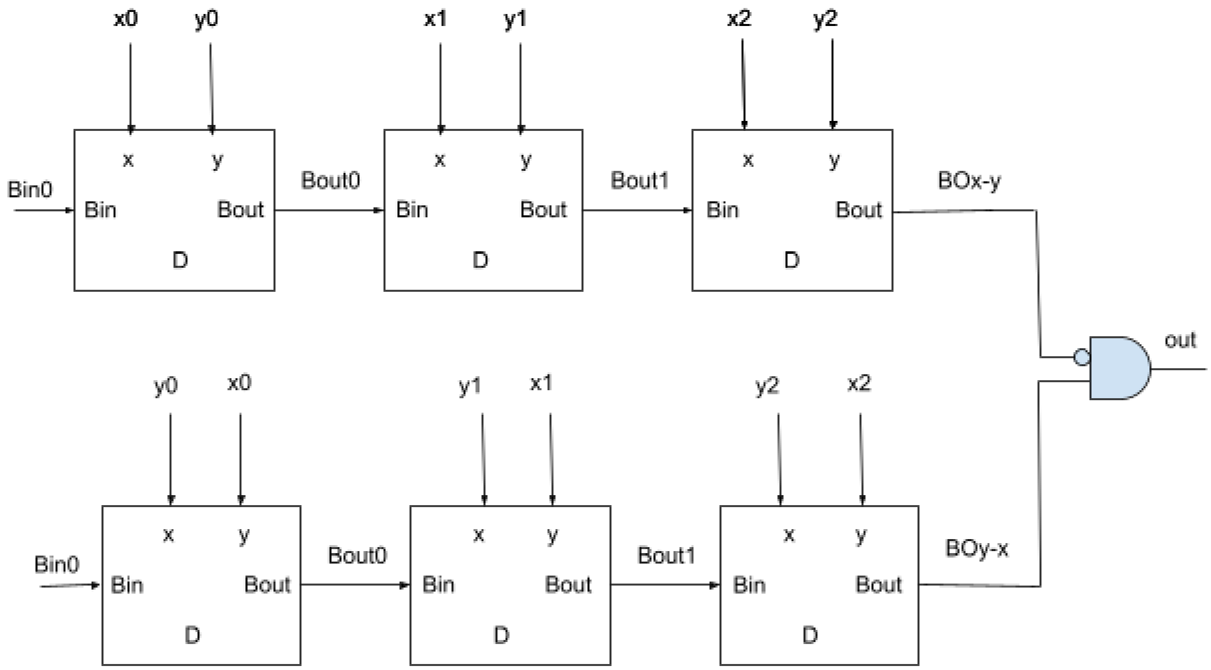
If the sum is greater than 7, then Cout2 in the upper ripple carry adder is 1. To subtract 5 from the sum, we use two's complement representation.

sum - 5 = sum + (-5)

-5 in two's complement = 1011

Adding the less significant 3 bits of -5 to the sum bits, we get the new sum and cout2 bits, which gives us the required result.

8)



Each block is a full subtractor with D = **X** XOR **Y** XOR **Bin**

Bout = X′ · Y + X′ · Bin + Y · Bin

If BOx-y = 0 and BOy-x = 1, then the output is 1 signifying X>Y

9) Carry lookahead adder does not wait for the carry to be propagated from one adder to another. Instead, it takes advantage of the parallelly fed input to obtain the generate carry and propagate carry functions that can produce the result instantaneously.

Cell i of an adder is said to generate a carry if Ci+1 = 1 based only on Xi and Yi, and independent of Xi−1. . .X0, Yi−1. . .Y0, C0. The carry-generate function is denoted by
Gi = Xi · Yi
Cell i of an adder is said to propagate a carry if Ci = 1 results in Ci+1 = 1 based only on Xi and Yi, and independent of Xi−1. . .X0,Yi−1. . .Y0 and C0. The carry-propagate function is denoted by
Pi = Xi + Yi

For a 4-bit carry lookahead adder:

C1 = G0 + P0 · C0

C2 = G1 + P1.C1 = G1 + P1. (G0 + P0.C0) = G1 + P1.G0 + P1.P0.C0

C3 = G2 + P2.C2 = G2 + P2. (G1 + P1.G0 + P1.P0.C0) = G2 + P2.G1 + P2.P1.G0 + P2.P1.P0.C0

C4 = G3 + P3.C3 = G3 + P3. (G2 + P2.G1 + P2.P1.G0 + P2.P1.P0.C0)
= G3 + P3.G2 + P3.P2.G1 + P3.P2.P1.G0 + P3.P2.P1.P0.C0

S0 = X0 xor Y0 xor C0
S1 = X1 xor Y1 xor C1
S2 = X2 xor Y2 xor C2
S3 = X3 xor Y3 xor C3

10)

Let the input value = A = A3 A2 A1 A0.

Output = Y = 3A = 2A + A.

2A can be obtained by left shifting A by 1 bit.

A bit-binary adder can be used to add A and the left shifted value of A.