Half subtractor:

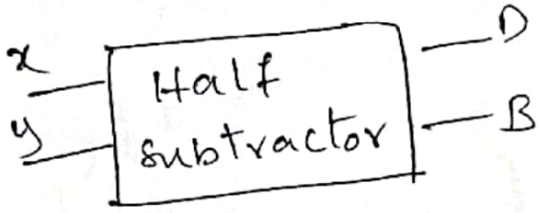
- Half subtractor is a combinational ckt used to get the difference b/w two single bit nos.

Step 1: No. of i/ps = 2 x, y

No. of o/ps = 2 D, B

D → diff, B → borrow

Step 2: Truth Table



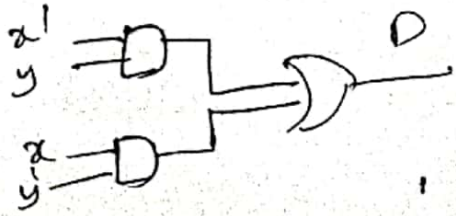
x	y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

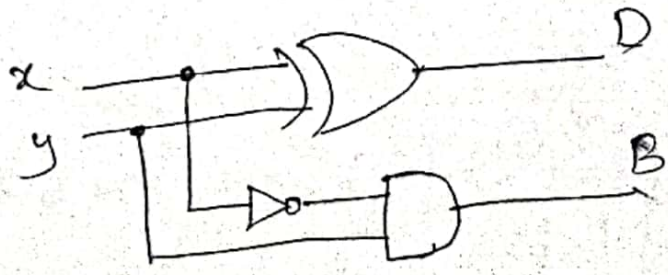
Step 3: Boolean expressions

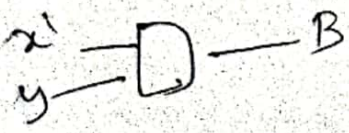
$$D = x'y + xy' = x \oplus y$$

$$B = x'y$$

step 4: logic diagram







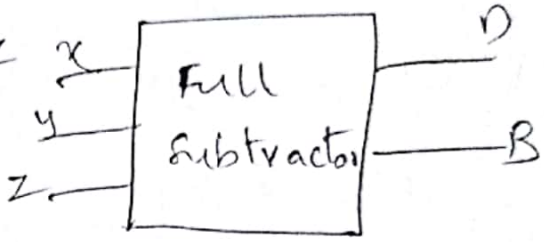
* Full Subtractor:

- Full subtractor is a combinational ckt used to perform subtraction of 3 bits.

Step 1:

No. of i/ps = 3 x, y, z

No. of o/ps = 2 D, B



Step 2: Truth Table

x	y	z	D	B
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

Step 3: K-Map – 3 variable

K Map for D

x \ yz	00	01	11	10
0		1		1
1	1		1	

$$D = xy'z' + x'y'z + xyz + x'yz'$$

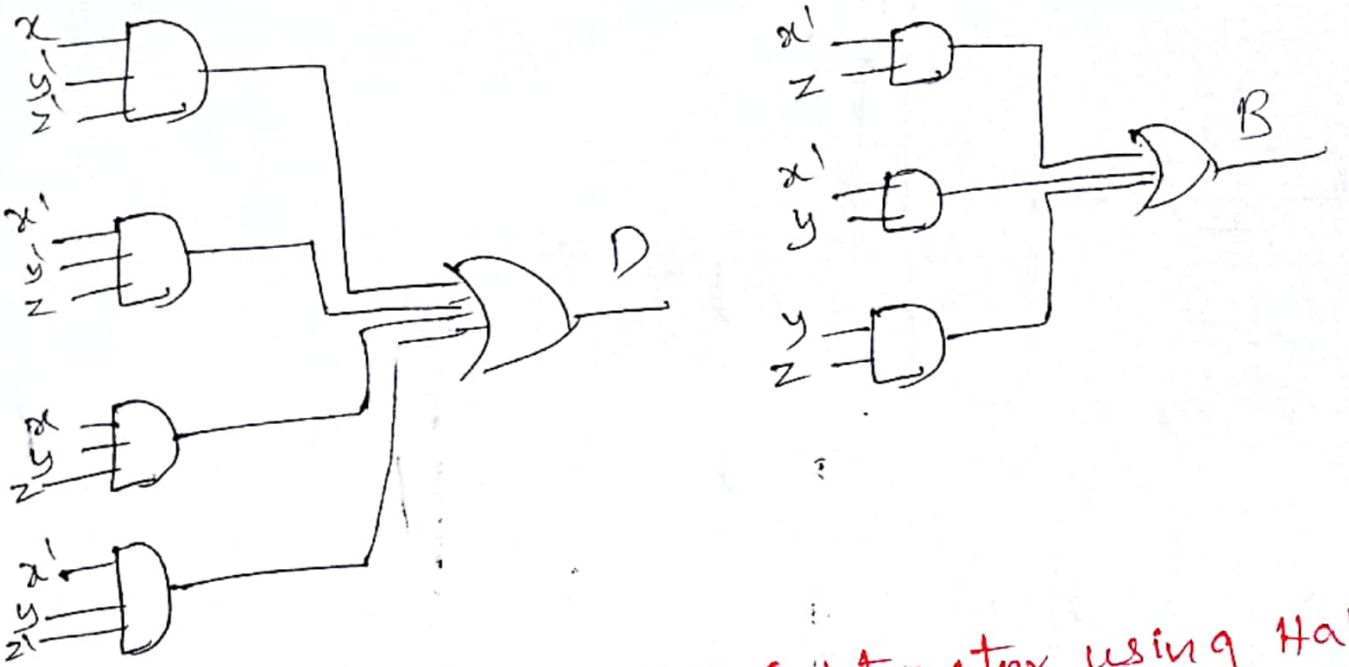
$$= x \oplus y \oplus z$$

K-Map for B

x \ yz	00	01	11	10
0		1	1	1
1			1	

$$B = x'z + x'y + yz$$

Step 4: logic diagram



Implementation of Full Subtractor using Half Subtractor:

WKT For full subtractor,

$$D = xy'z' + x'y'z + xyz + x'yz'$$

$$= x \oplus y \oplus z$$

and $B = x'z + x'y + yz$

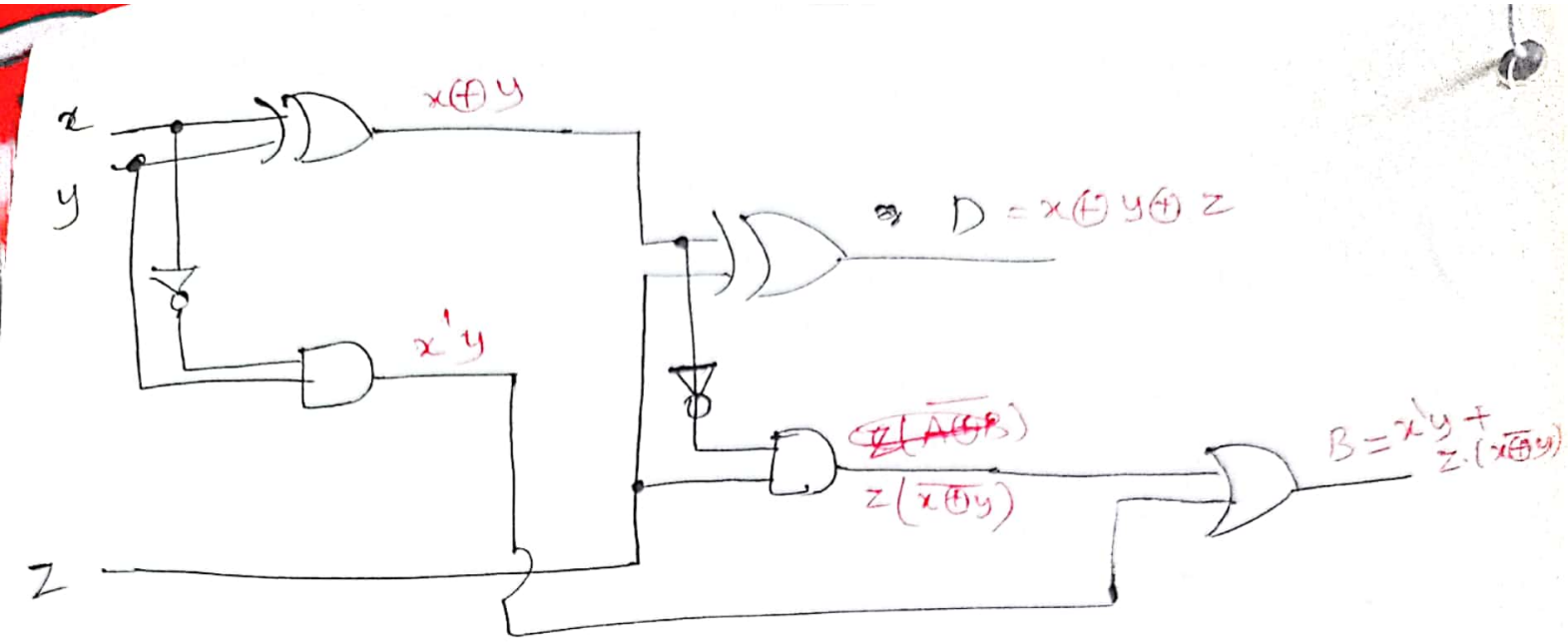
$$= x'(y+y')z + x'y(z+z') + (x+x')yz$$

$$= x'yz + x'y'z + x'yz + x'yz' + xyz + x'yz$$

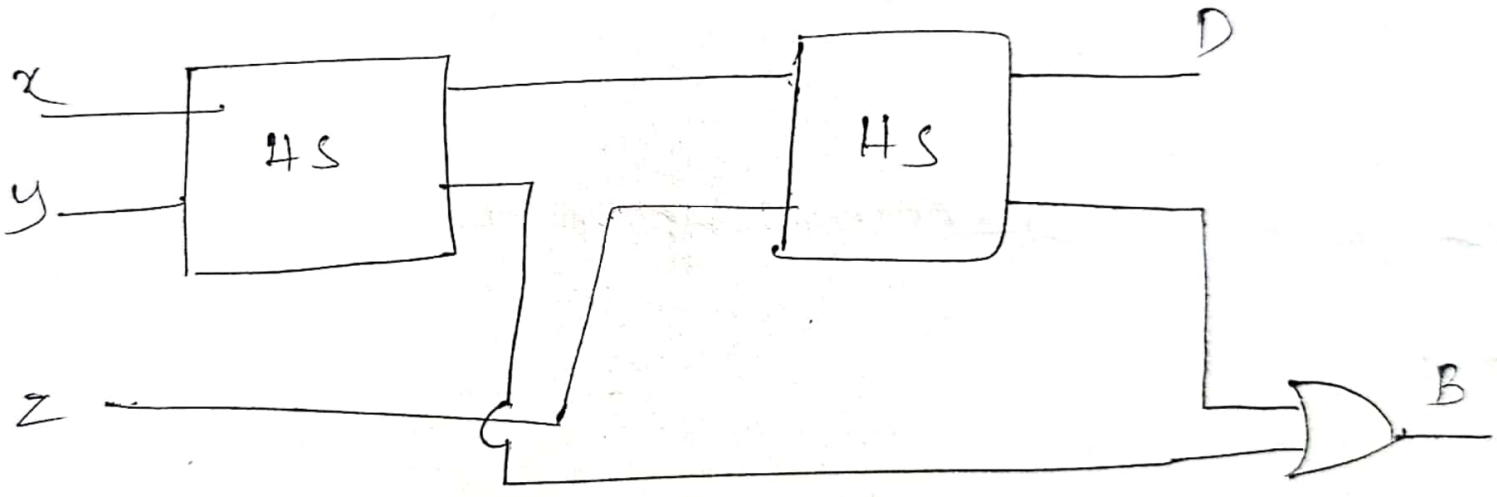
$$= x'yz + x'y'z + x'yz' + xyz$$

$$= x'y(z+z)' + z(xy + x'y')$$

$$= x'y + z(\overline{x \oplus y})$$



(OR)



A half-subtractor can also be realized using universal logic—either using only NAND gates or using only NOR gates—as shown in Figures 4.14 and 4.15 respectively.

NAND logic

$$d = A \oplus B = \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$$

$$b = \overline{A}B = B(\overline{A} + \overline{B}) = B(\overline{AB}) = \overline{B \cdot \overline{AB}}$$

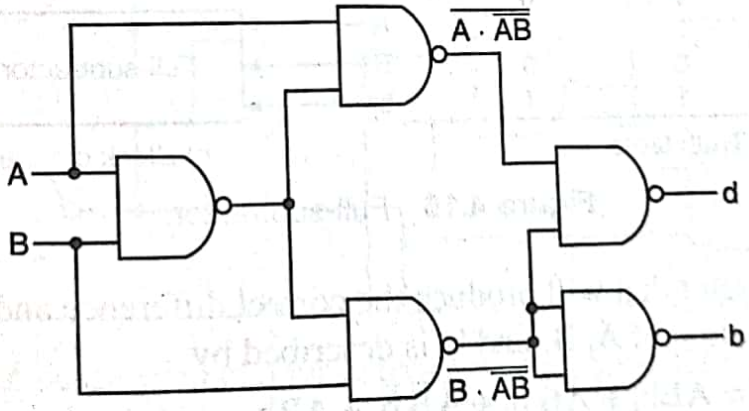


Figure 4.14 Logic diagram of a half-subtractor using only 2-input NAND gates.

NOR logic

$$d = A \oplus B = A\overline{B} + \overline{A}B = A\overline{B} + B\overline{B} + \overline{A}B + A\overline{A}$$

$$= \overline{B}(A + B) + \overline{A}(A + B) = \overline{\overline{B + \overline{A + B}} + \overline{A + \overline{A + B}}}$$

$$d = \overline{A}B = \overline{A}(A + B) = \overline{\overline{A}(A + B)} = \overline{A + (\overline{A + B})}$$

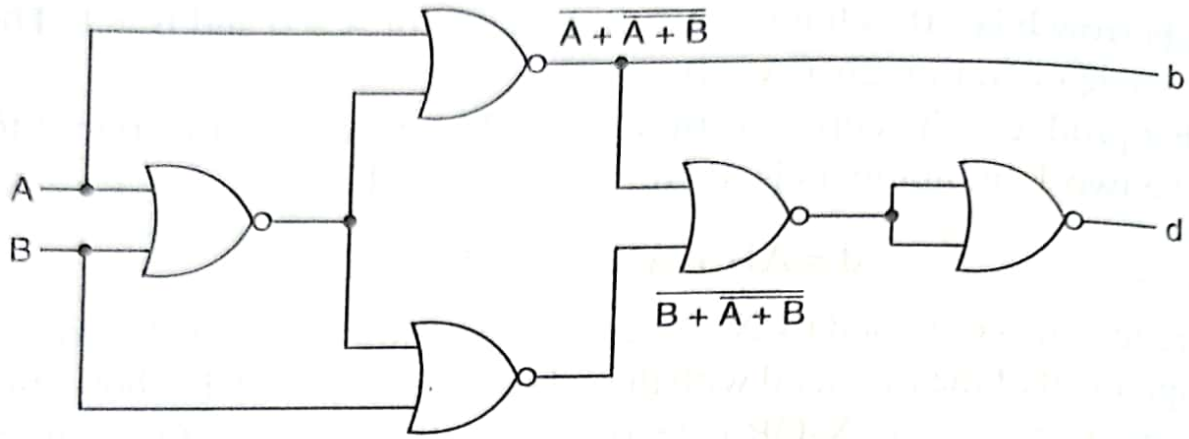


Figure 4.15 Logic diagram of a half-subtractor using only 2-input NOR gates.

The full subtractor can also be realized in universal logic using either only NAND gates or only NOR gates as shown in Figures 4.18 and 4.19 respectively.

NAND logic

$$d = A \oplus B \oplus b_i = \overline{\overline{(A \oplus B) \oplus b_i}} = \overline{\overline{(A \oplus B)\overline{(A \oplus B)b_i}} \cdot \overline{b_i(A \oplus B)b_i}}$$

$$b = \bar{A}B + b_i(\overline{A \oplus B}) = \overline{\bar{A}B + b_i(\overline{A \oplus B})}$$

$$= \overline{\overline{\bar{A}B} \cdot \overline{b_i(A \oplus B)}} = \overline{B(\bar{A} + \bar{B}) \cdot b_i(\bar{b}_i + (\overline{A \oplus B})]}$$

$$= \overline{B \cdot \overline{AB} \cdot \overline{b_i[b_i \cdot (A \oplus B)]}}$$

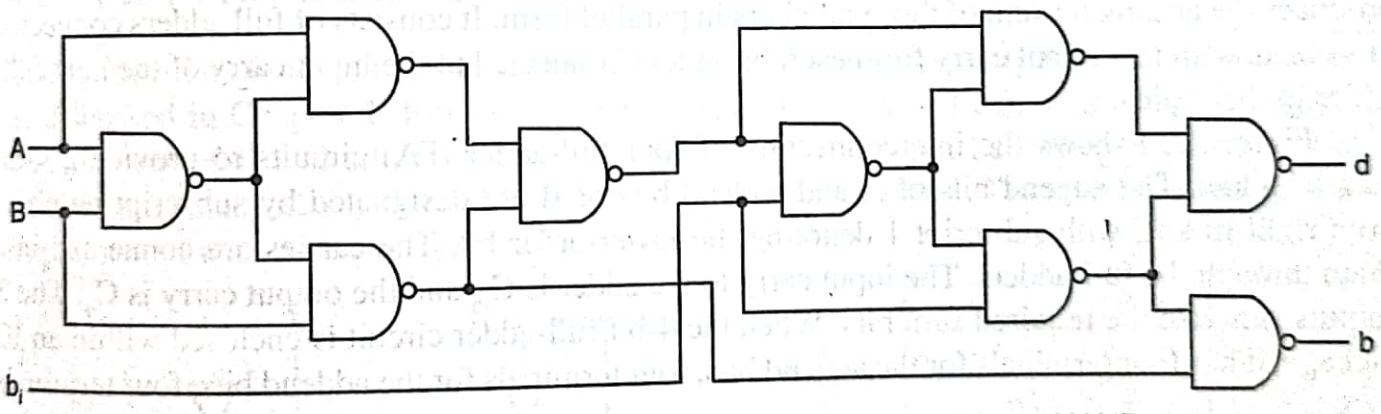


Figure 4.18 Logic diagram of a full-subtractor using only 2-input NAND gates.

NOR logic

$$d = A \oplus B \oplus b_i = \overline{\overline{(A \oplus B) \oplus b_i}}$$

$$= \overline{(A \oplus B)b_i + (\overline{A \oplus B})\bar{b}_i}$$

$$= \overline{[(A \oplus B) + (\overline{A \oplus B})\bar{b}_i][b_i + (\overline{A \oplus B})\bar{b}_i]}$$

$$= \overline{\overline{(A \oplus B) + \overline{(A \oplus B) + b_i}} + \overline{b_i + \overline{(A \oplus B) + b_i}}}$$

$$= \overline{\overline{(A \oplus B) + \overline{(A \oplus B) + b_i}} + \overline{b_i + \overline{(A \oplus B) + b_i}}}$$

$$b = \bar{A}B + b_i(\overline{A \oplus B})$$

$$= \bar{A}(A + B) + (\overline{A \oplus B})[(A \oplus B) + b_i]$$

$$= \overline{\overline{A + \overline{A + B}} + \overline{(A \oplus B) + \overline{(A \oplus B) + b_i}}}$$

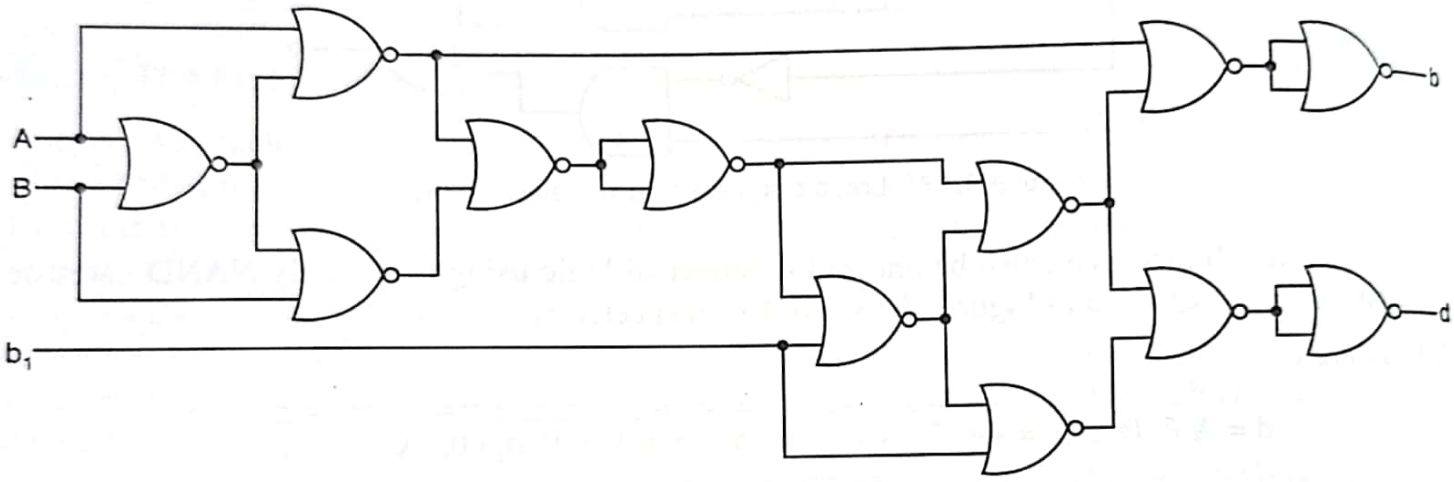


Figure 4.19 Logic diagram of a full subtractor using only 2-input NOR gates.