# DIGITAL DESIGN

CS/ECE/EEE/INSTR F215

Sarang Dhongdi

## Example on MEV

**Ex 2**: Find the minimum Sum of Product form for function $F_1$ and $F_2$ using Variable Entered Mapping (VEM) technique. Assume D as Map Entered Variable (MEV).

| A | B | C | D | $F_1$ | $F_2$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | X |
| 0 | 1 | 0 | 0 | X | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | X |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | X | X |
| 1 | 1 | 0 | 1 | 0 | X |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

## Example on MEV

**Ex 2**: Find the minimum Sum of Product form for function $F_1$ and $F_2$ using Variable Entered Mapping (VEM) technique. Assume D as Map Entered Variable (MEV).

| A | B | C | D | $F_1$ | $F_2$ | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | X | |
| 0 | 1 | 0 | 0 | X | 1 | $\bar{D}\phi$ |
| 0 | 1 | 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 0 | 1 | X | $\bar{D}$ |
| 0 | 1 | 1 | 1 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | |
| 1 | 1 | 0 | 0 | X | X | $\bar{D}\phi$ |
| 1 | 1 | 0 | 1 | 0 | X | |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | |

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | D’ | D’φ |
| 1 | 1 | D | 0 | D’φ |

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | D+D’ | D’ | D’φ |
| 1 | D+D’ | D | 0 | D’φ |

$\bar{A}C\bar{D}$

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | D+D’ | D’ | D’φ |
| 1 | D+D’ | D | 0 | D’φ |

$\bar{B}CD$

## Step 2

- MEV → 0
- MEV' → 0
- 0 → 0
- φ → φ
- MEV. φ → 0
- MEV'. φ → 0

- 1 → 1 (If not completely covered)
- 1 → φ (If completely covered)

- MEV+MEV' φ → 1 (If not covered at all or only φ covered )
- MEV+MEV' φ → φ (If completely covered or necessary terms are covered)
- Same for MEV'+MEV φ

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | φ | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |

$A\bar{B}\bar{C}$

## Example on MEV

**Ex 2**: Find the minimum Sum of Product form for function $F_1$ and $F_2$ using Variable Entered Mapping (VEM) technique. Assume D as Map Entered Variable (MEV).

| A | B | C | D | $F_1$ | $F_2$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | X |
| 0 | 1 | 0 | 0 | X | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | X |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | X | X |
| 1 | 1 | 0 | 1 | 0 | X |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

## Example on MEV

**Ex 2**: Find the minimum Sum of Product form for function $F_1$ and $F_2$ using Variable Entered Mapping (VEM) technique. Assume D as Map Entered Variable (MEV).

| A | B | C | D | $F_1$ | $F_2$ | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | |
| 0 | 0 | 1 | 0 | 1 | 1 | $\bar{D}+D\phi$ |
| 0 | 0 | 1 | 1 | 1 | X | |
| 0 | 1 | 0 | 0 | X | 1 | $\bar{D}$ |
| 0 | 1 | 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 0 | 1 | X | $\bar{D}\phi$ |
| 0 | 1 | 1 | 1 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 1 | 0 | $D$ |
| 1 | 0 | 0 | 1 | 1 | 1 | |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | |
| 1 | 1 | 0 | 0 | X | X | $\phi$ |
| 1 | 1 | 0 | 1 | 0 | X | |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | |

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | D'+Dφ | D' φ | D' |
| 1 | D | 1 | 0 | φ |

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | D+D' | D'+Dφ | D' φ | D' |
| 1 | D | D+D' | 0 | φ |

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | D+D' | D'+Dφ | D' φ | D' |
| 1 | D | D+D' | 0 | φ |

$\bar{A}\bar{D}$

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | D+D' | D'+Dφ | D' φ | D' |
| 1 | D | D+D' | 0 | φ |

$\bar{B}D$

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | φ | φ | 0 | 0 |
| 1 | 0 | 1 | 0 | φ |

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | φ | φ | 0 | 0 |
| 1 | 0 | 1 | 0 | φ |

$\bar{B}C$

# TWO LEVEL IMPLEMENTATION

## Two level implementations

- 4 types of gates – AND, OR, NAND, NOR
- Use 1 type of gate for level 1 and one type for level 2.
- Same type can be used for level 2.
- Total 16 combinations.
- 8 combinations – degenerate form
- Remaining 8 – Non-degenerate form

## Two level implementations

- Degenerate
  - AND-AND
  - OR-OR
  - AND-NAND
  - OR-NOR
  - NAND-NOR
  - NAND-OR
  - NOR-NAND
  - NOR-AND
- Nondegenerate
  - AND-OR
  - NAND-NAND
  - OR-AND
  - NOR-NOR
  - NOR-OR
  - NAND-AND
  - OR-NAND
  - AND-NOR

## AND-OR-INVERT (AND-NOR)

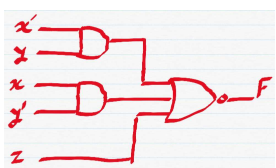
| x \ yz | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |

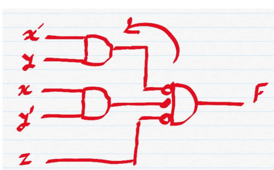
Combine 0's to obtain complement of function in SOP form

F'= z+ xy'+x'y

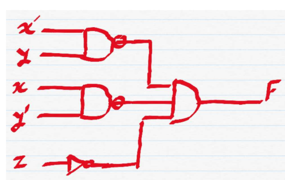
F= (z+ xy'+x'y)'

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




NAND-AND




## OR-AND-INVERT (OR-NAND)

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

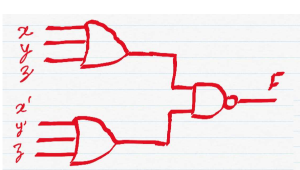
Combine 1's for function F, then take complement of function in POS form

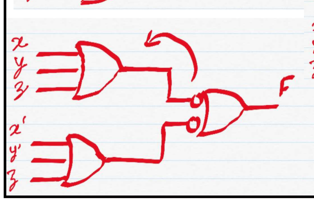
F= x'y'z'+xyz'

F'= (x'y'z'+xyz')' = (x+y+z)(x'+y'+z)

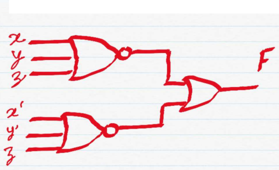
F= [(x+y+z)(x'+y'+z)]'

F= [(x+y+z)(x'+y'+z)]'



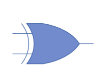




NOR-OR




## Exclusive-OR function

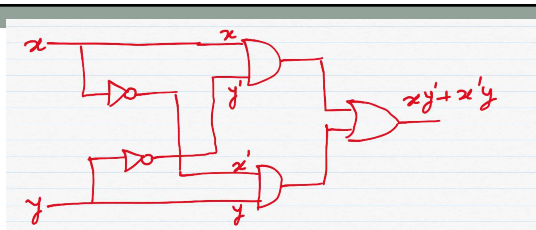
XOR

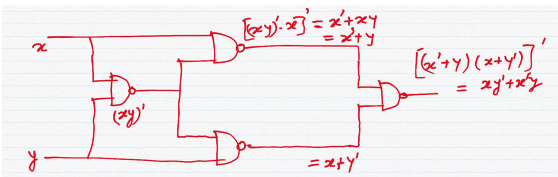


| A | B | F=A.B'+A'B<br>F= A ⊕ B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- A ⊕ 0 =A
- A ⊕ 1 =A'
- A ⊕ A =0
- A ⊕ A' =1
- A ⊕ B' =A' ⊕ B = (A⊕B)'
- A ⊕ B =B ⊕ A
- (A ⊕ B) ⊕ C =
  =A ⊕ (B⊕C)
  =A ⊕ B ⊕ C

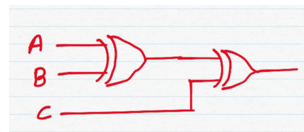
- Ex-OR A ⊕ B = AB'+A'B
- Ex-NOR (A ⊕ B)' = (AB'+A'B)'
- = (A'+B)(A+B')
- = AB+A'B'




## Exclusive-OR function

- For three inputs,
- F = A ⊕ B ⊕ C = (AB'+A'B)C' + (AB+A'B')C
- = AB'C'+A'BC'+ABC+A'B'C
- = ∑ (1,2,4,7)




| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

## Odd and Even function

Exclusive- OR function

ODD function

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | 1 | | 1 |
| 1 | 1 | | 1 | |

Exclusive-NOR function

Even Function

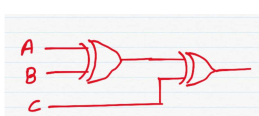
| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | | 1 | |
| 1 | | 1 | | 1 |

## Four variable Ex-OR

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | 1 | | 1 |
| 01 | 1 | | 1 | |
| 11 | | 1 | | 1 |
| 10 | 1 | | 1 | |

## Parity Generation and Checking

| Message Bits | | | Parity Bit |
|---|---|---|---|
| A | B | C | P |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |



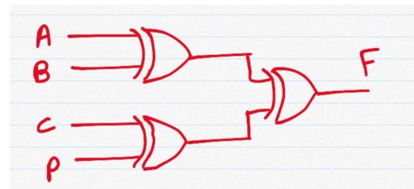

## Parity Generation and Checking

| Received bits | | | | Parity Error Check |
|---|---|---|---|---|
| A | B | C | P | F |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

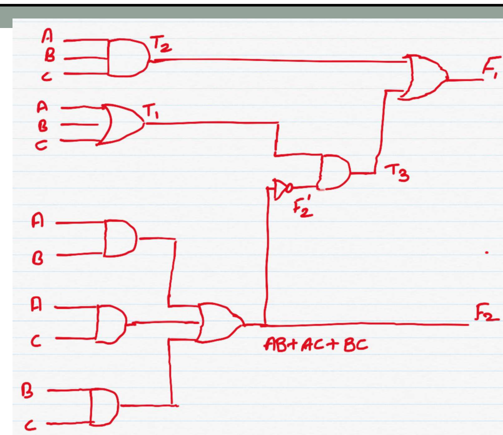
| Received bits | | | | Parity Error Check |
|---|---|---|---|---|
| A | B | C | P | F |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |

## Four variable Ex-OR

| AB \ CP | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | 1 | | 1 |
| 01 | 1 | | 1 | |
| 11 | | 1 | | 1 |
| 10 | 1 | | 1 | |




## COMBINATIONAL CIRCUITS




- $F1 = T_3 + T_2$
- $= F_2'T_1 + ABC$
- $= (AB+AC+BC)'(A+B+C) + ABC$
- $= A'BC' + A'B'C + AB'C' + ABC$

## Truthtable

| A | B | C | F2 | F2' | T1 | T2 | T3 | F1 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |

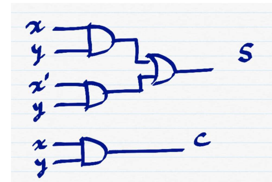
# BINARY ADDER AND SUBTRACTOR

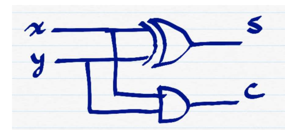
## Half Adder

| X | Y | C | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

S = x'y + xy'

C = xy

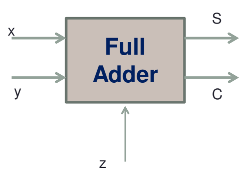




## Full Adder

| X | Y | Z | C | 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 |

S = x'y'z + x'yz' + xy'z' + xyz

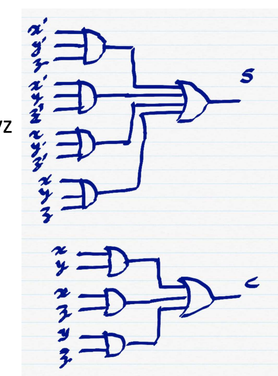
C = xy + xz + yz




## Full Adder

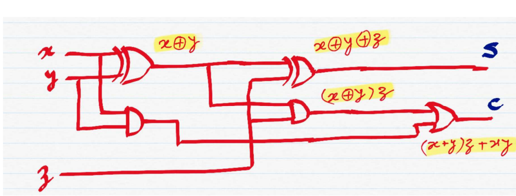
S = x'y'z + x'yz' + xy'z' + xyz

C = xy + xz + yz



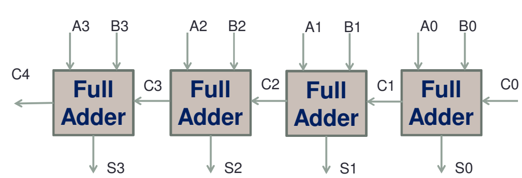
## Circuit implementation

- S = x ⊕ y ⊕ z
- C = xy + xz + yz
- C = xy + z(x⊕y)



## Binary adder

- Add two 4 – bit nos.
- Augend - A3A2A1A0
- Addend – B3B2B1B0



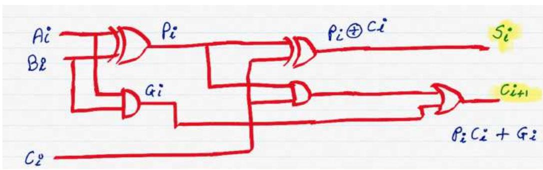
## Carry Propagation

- For a "Carry Lookahead logic",
- we define two new variables,

$G_i$ = carry generator
$P_i$ = Carry Propagate

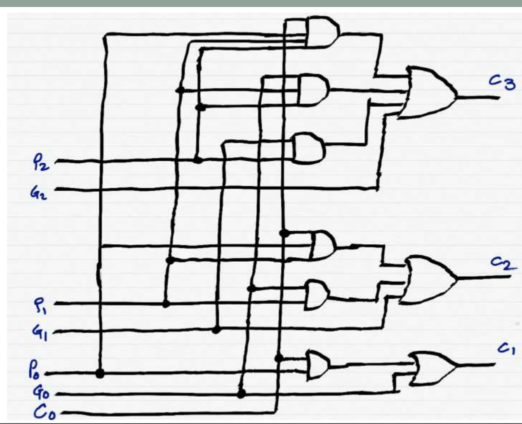
- $P_i = A_i \oplus B_i$
- $G_i = A_iB_i$

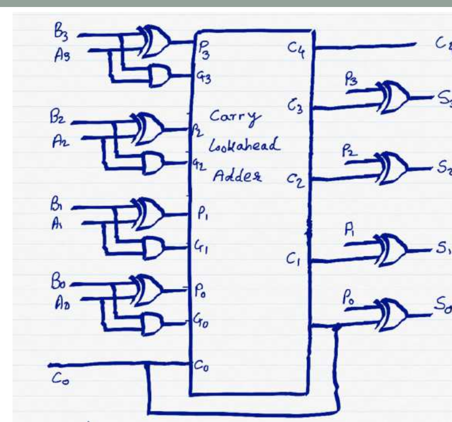
$S_i = P_i \oplus C_i$
$C_{i+1} = G_i + P_iC_i$




## Carry lookahead logic

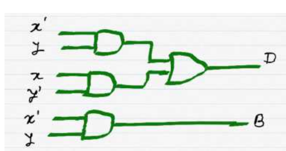
- $C_0$ = Input carry
- $C_1 = G_0 + P_0C_0$
- $C_2 = G_1 + P_1C_1$
- $= G_1 + P_1(G_0 + P_0C_0)$
- $= G_1 + P_1G_0 + P_1P_0C_0$
- $C_3 = G_2 + P_2C_2$
- $= G_2 + P_2G_1 + P_2P_1G_0 + P_3P_2P_1G_0$

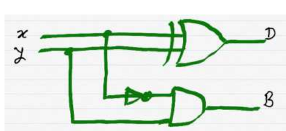



## Half Subtractor

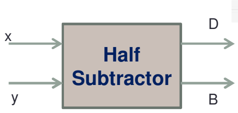
| X | Y | D | B |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |




$D = x'y + xy'$

$B = x'y$

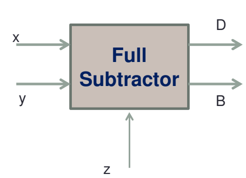


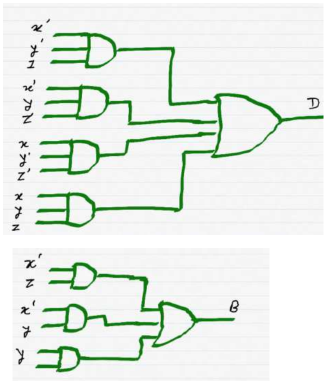

## Full Subtractor

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

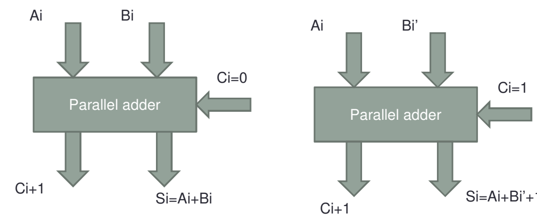
$B = x'z + x'y + yz$

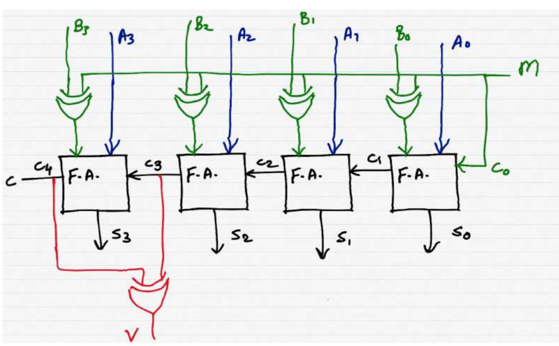
| X | Y | Z | D | B |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |



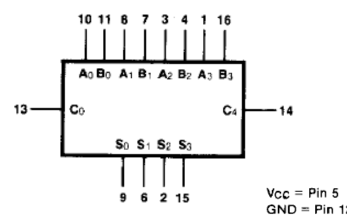



## Binary Subtractor




## MSI IC 7483

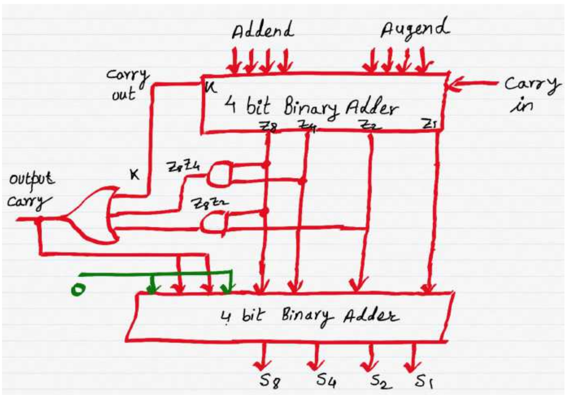



## BCD Adder

| No. | Binary sum | | | | | BCD sum | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | K | Z8 | Z4 | Z2 | Z1 | C | S8 | S4 | S2 | S1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 6 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 7 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 8 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 9 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |

## BCD Adder

| No. | Binary sum | | | | | BCD sum | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | K | Z8 | Z4 | Z2 | Z1 | C | S8 | S4 | S2 | S1 |
| 10 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 12 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 13 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 14 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 15 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 16 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 17 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 18 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 19 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |




## Binary Multiplier

- Multiplication of two 2-bit numbers
- Multiplicand B1B0
- Multiplier A1A0
- - Solve using k-map (and minimum gate implementation)
- - Implement using adders

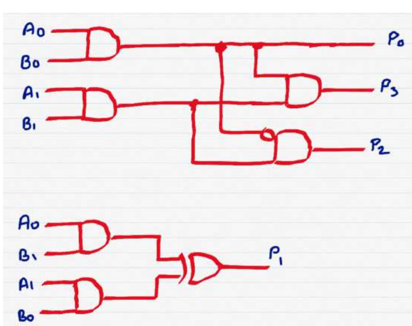
| Inputs | | | | Outputs | | | |
|---|---|---|---|---|---|---|---|
| A1 | A0 | B1 | B0 | P3 | P2 | P1 | P0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |

## Equations

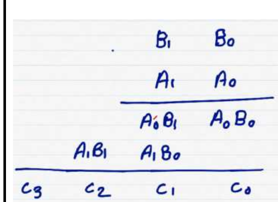
- P3 = F(A1,A0,B1,B0) = ∑ (15)
- P2 = F(A1,A0,B1,B0) = ∑ (10,11,14)
- P1 = F(A1,A0,B1,B0) = ∑ (6,7,9,11,13,14)
- P0 = F(A1,A0,B1,B0) = ∑ (5,7,13,15)

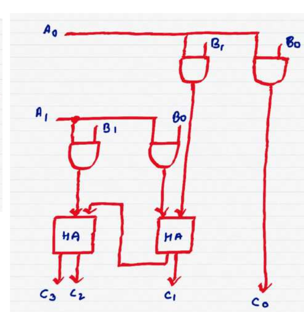
- P3 = A1A0B1B0
- P2 = A1A0'B1+A1B1B0' = A1B1(A0B0)'
- P1 = A1'A0B1+A0B1B0'+A1B1'B0+A1A0'B0 =
- = A0B1(A1B0)'+A1B0(B1A0)'
- P0 =A0B0

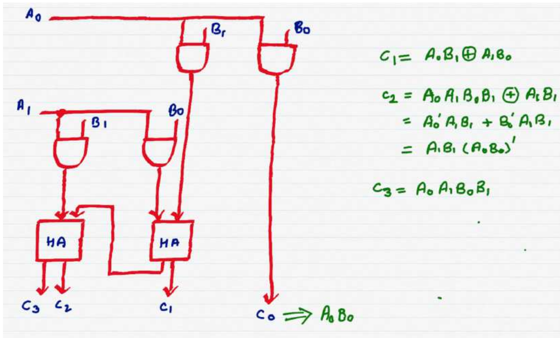
## Implementation




## Implementation using Adders






## Multiplication of 4 bit by 3 bit