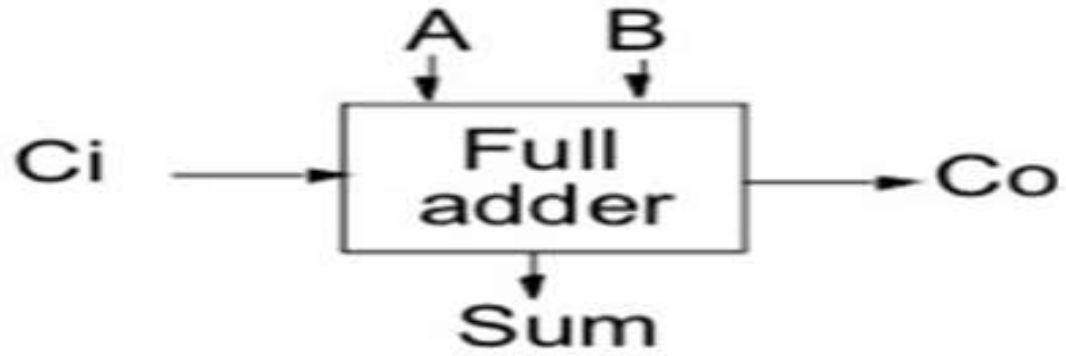
# EC 304 – VLSI
## Module VI

Audio only

# Static Adder




$$S = A \oplus B \oplus C_i$$

$$= A\bar{B}\bar{C}_i + \bar{A}B\bar{C}_i + \bar{A}\bar{B}C_i + ABC_i$$

$$C_o = AB + BC_i + AC_i$$

# Truth Table with carry status

| A | B | $C_{in}$ | S | $C_{out}$ | Carry Status |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | delete |
| 0 | 0 | 1 | 1 | 0 | delete |
| 0 | 1 | 0 | 1 | 0 | propagate |
| 0 | 1 | 1 | 0 | 1 | propagate |
| 1 | 0 | 0 | 1 | 0 | propagate |
| 1 | 0 | 1 | 0 | 1 | propagate |
| 1 | 1 | 0 | 0 | 1 | generate |
| 1 | 1 | 1 | 1 | 1 | generate |

- Three new variables are defined G, P D
  - ✓ Generate (G)
  - ✓ Propagate(P)
  - ✓ Delete(D)
- G=1 ensures that a carry bit will be generated independent of Ci
- D=1 ensures that a carry bit will be deleted independent of $C_i$
- P=1 ensures that an incoming carry will be propagated to $C_o$

**Generate (G) = AB**

**Propagate (P) = $A \oplus B$**

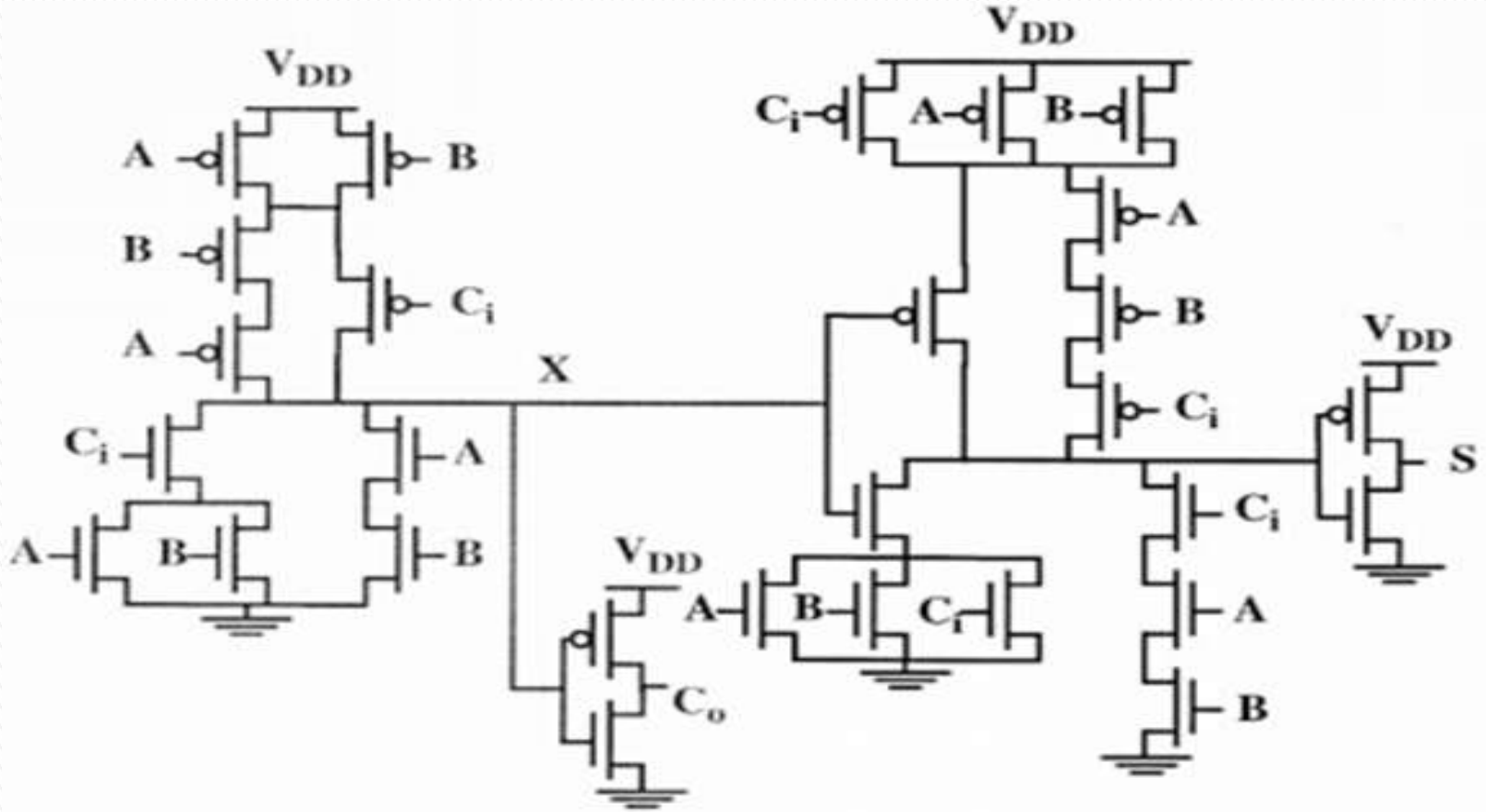
**Delete = $\overline{A}\,\overline{B}$**

$$C_o(G,P) = G + PC_i$$

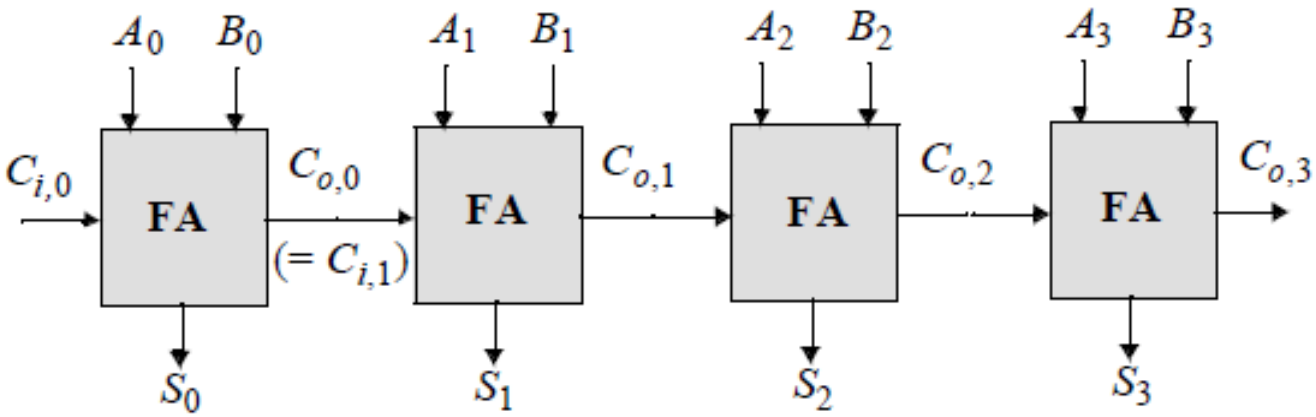
$$S(G,P) = P \oplus C_i$$

# Static CMOS implementation of Full-Adder

- ✓ 28 transistors are required for this implementation



# Ripple Carry Adder

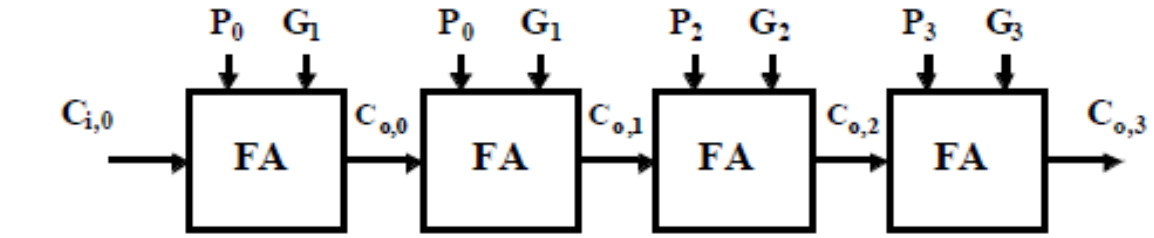


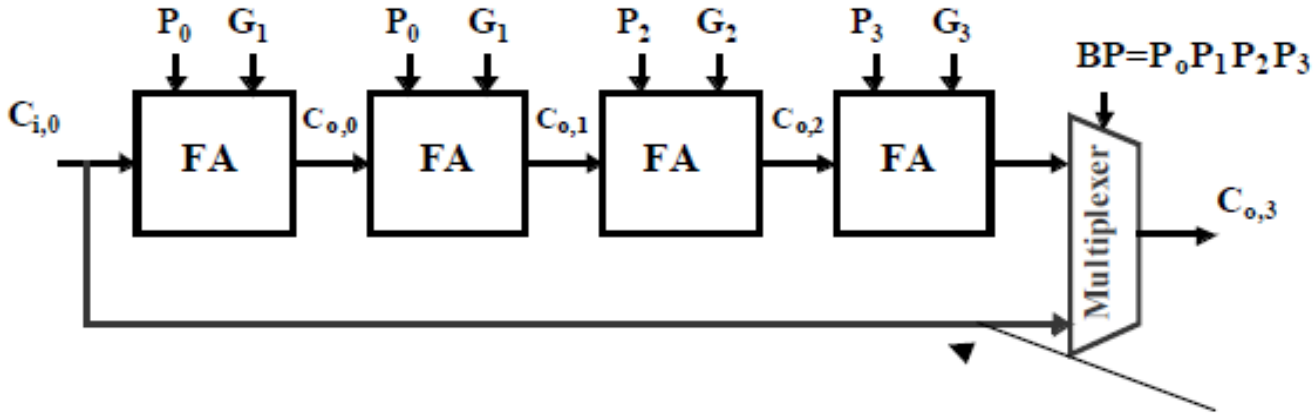

$$t_{adder} \approx (N-1)t_{carry} + t_{sum}$$

Goal: Make the fastest possible carry path circuit

- Propagation delay of ripple carry adder is linearly proportional to N.
- So when designing a full adder for a fast ripple carry adder optimize $t_{carry}$ and $t_{sum}$

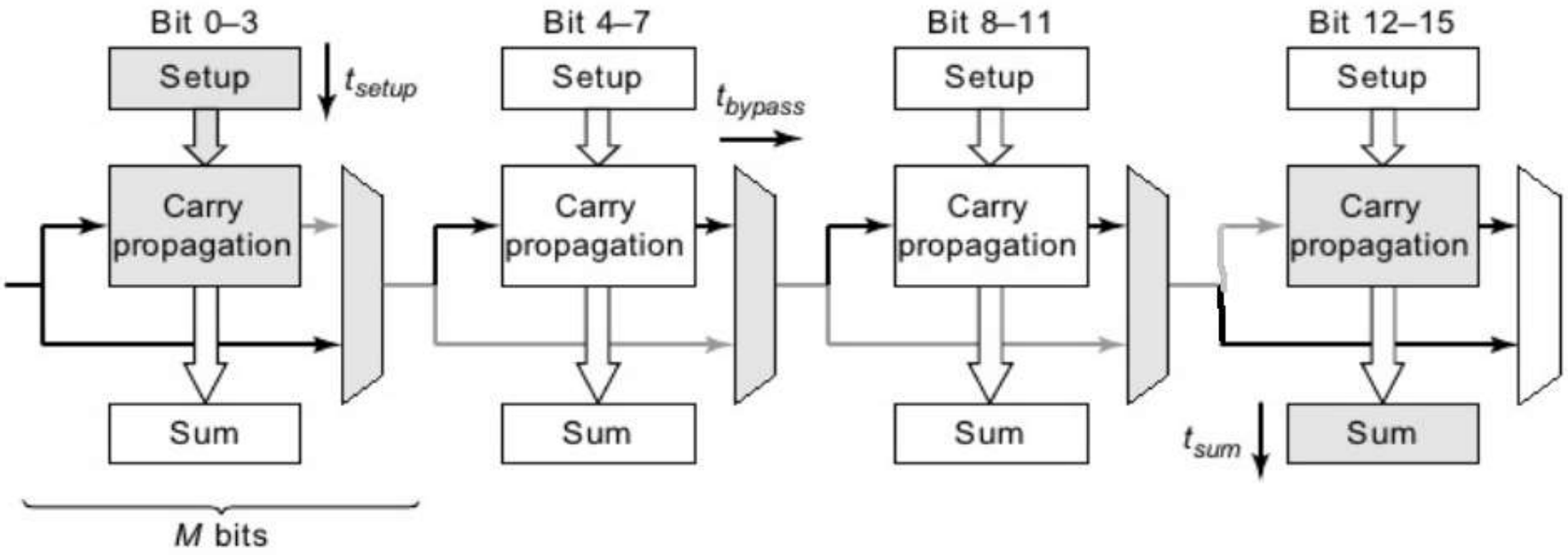
# Carry-Bypass /Carry Skip Adder




Idea: If (P0 and P1 and P2 and P3 = 1) then $C_{o3} = C_0$, else "kill" or "generate".

# Carry Bypass Adder (N=16)




$$t_{adder} = t_{setup} + Mt_{carry} + (N/M\text{-}1)t_{bypass} + (M\text{-}1)t_{carry} + t_{sum}$$

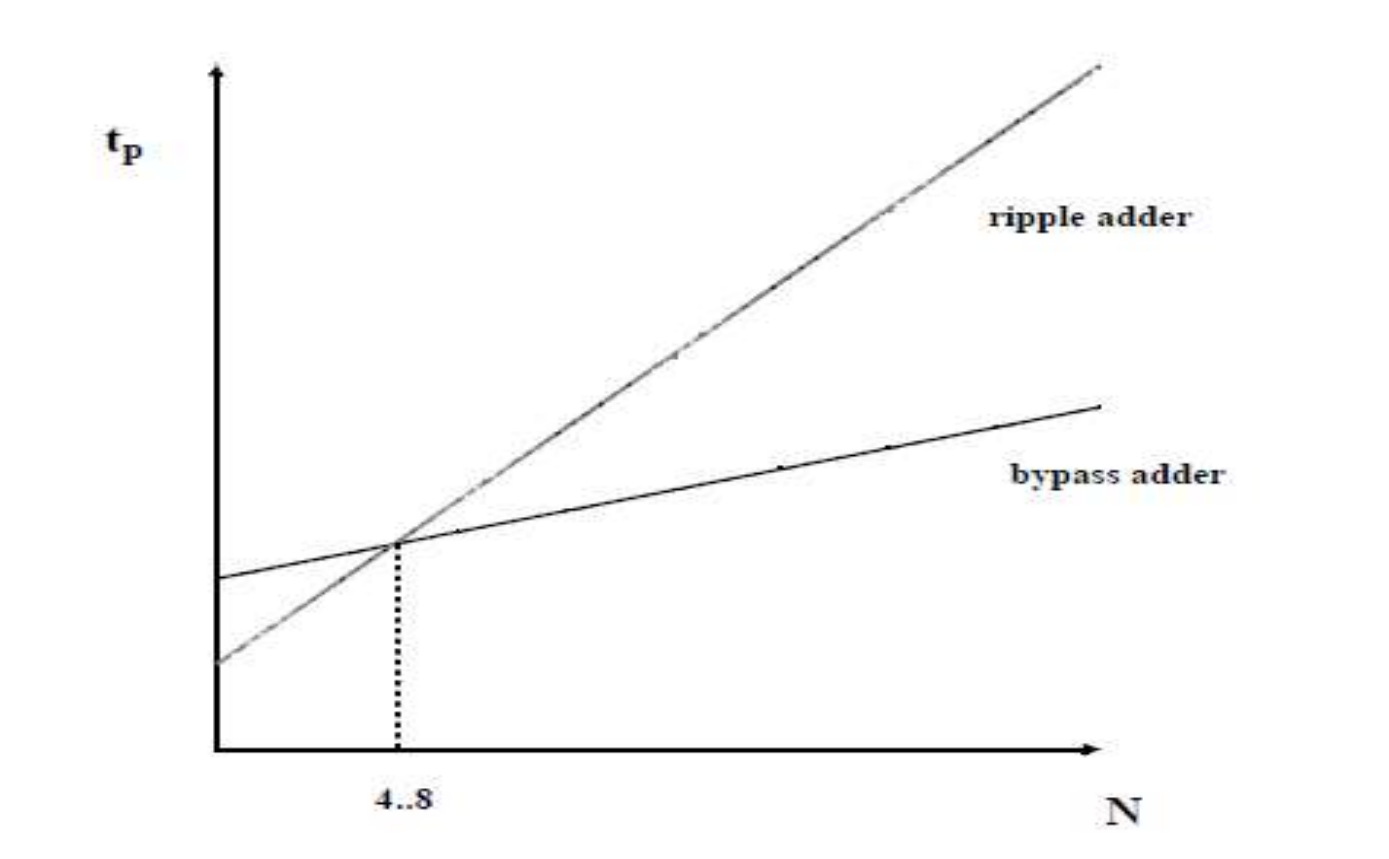
N – No. of bits(16)

M- No. of bits in each bypass stage (4)

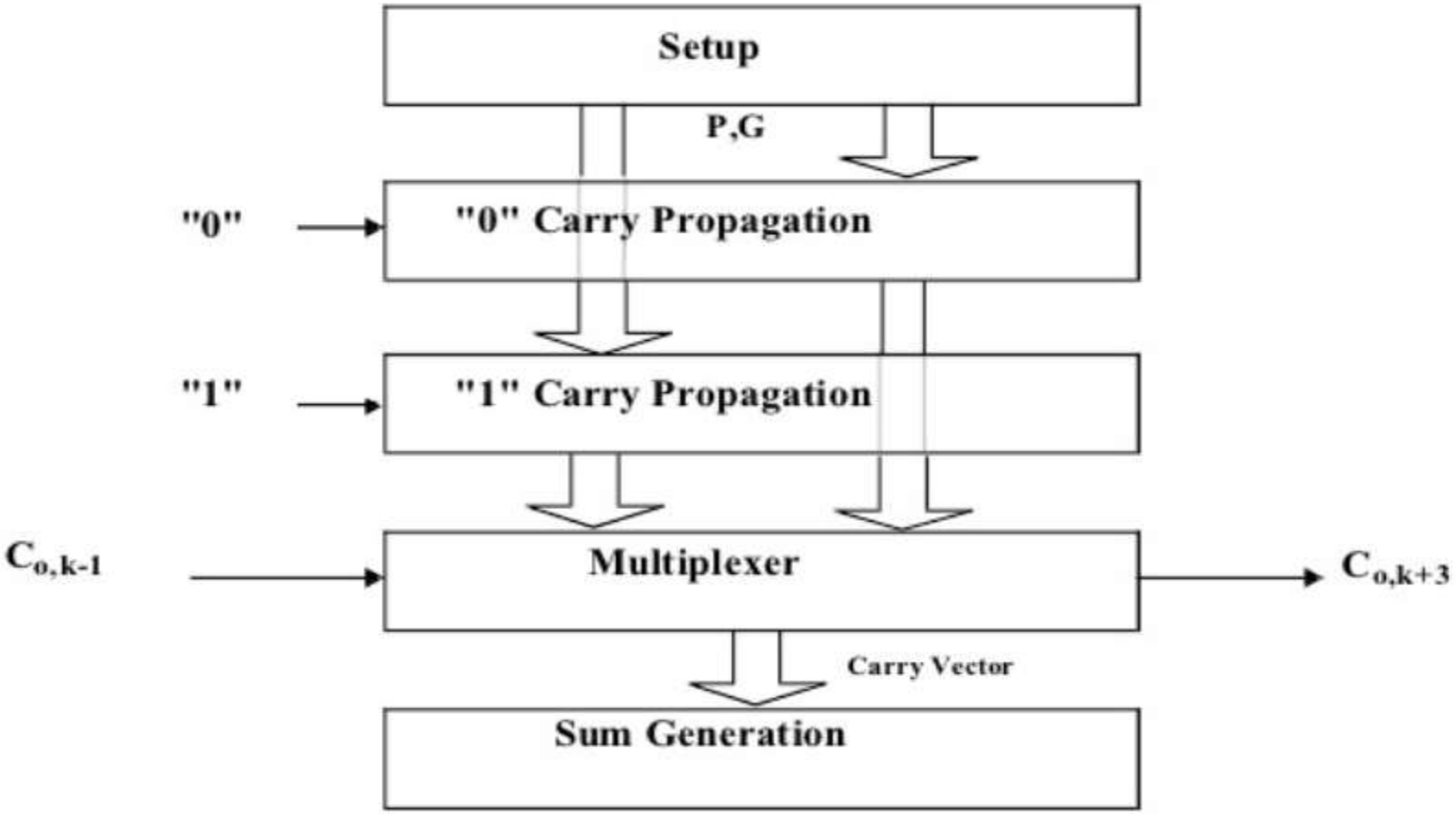
The worst case delay path is shaded in gray

- $t_{carry}$=the propagation delay through a single bit. The worst case carry-propagation delay through a single stage of M bits is approximately M times larger
- $t_{setup}$= the fixed overhead time to create the generate and propagate signals
- $t_{bypass}$=the propagation delay through the bypass multiplexer of a single stage
- $t_{sum}$=the time to generate the sum of the final stage

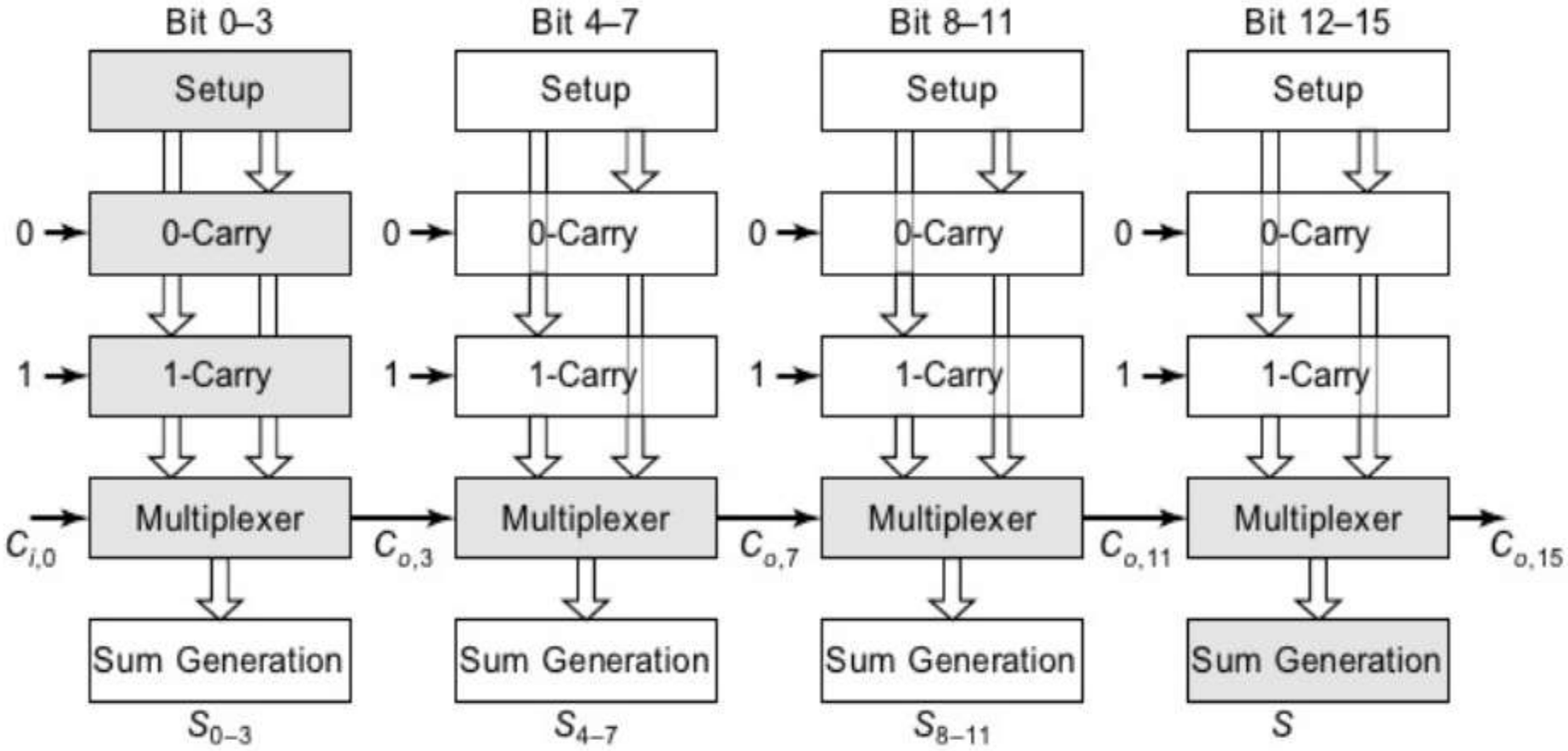
# Delay Comparison



# Linear Carry Select Adder



# Critical Path of Carry Select Adder



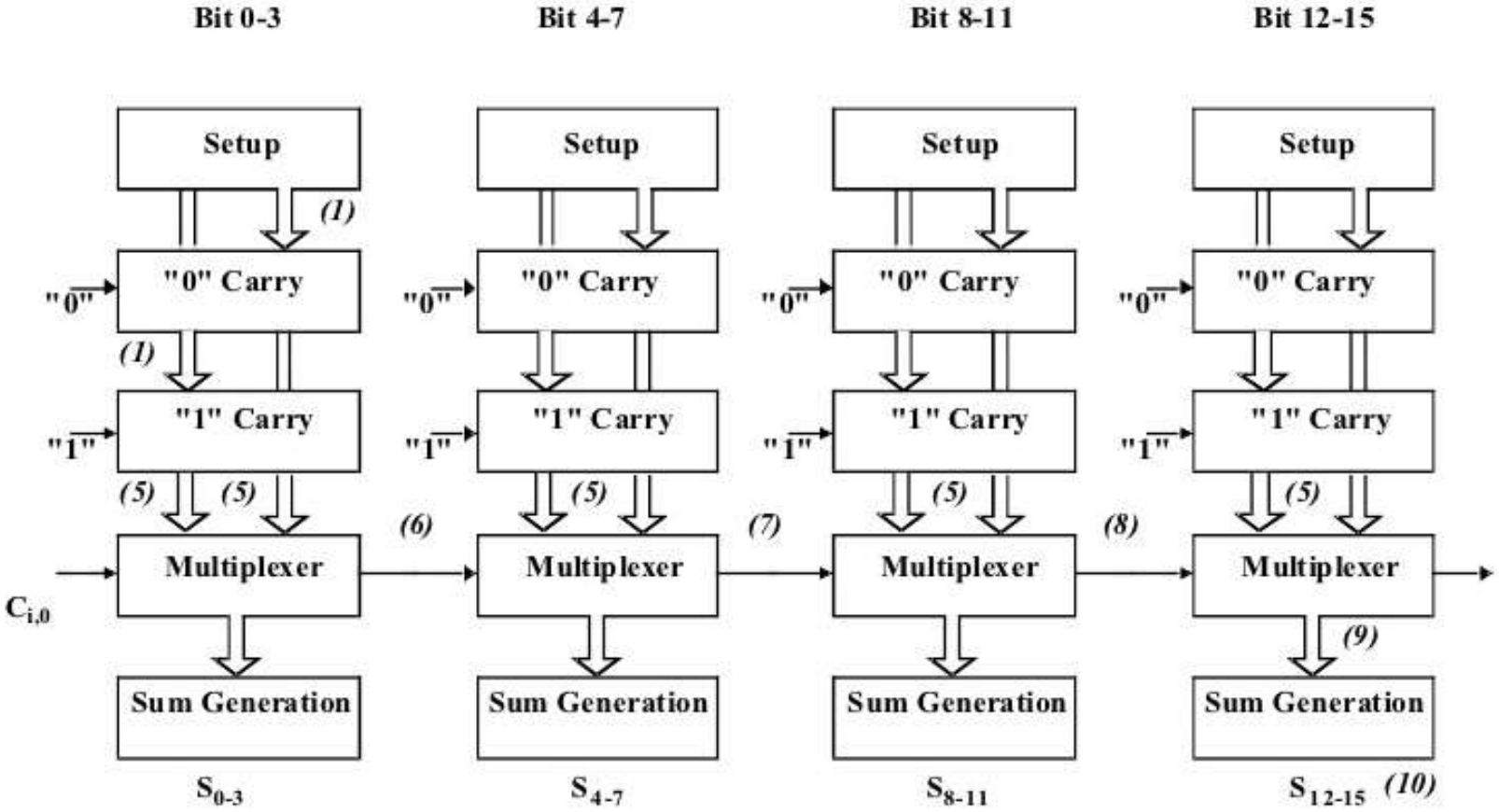
# Propagation delay

$$t_{add} = t_{setup} + Mt_{carry} + (N/M)t_{mux} + t_{sum}$$

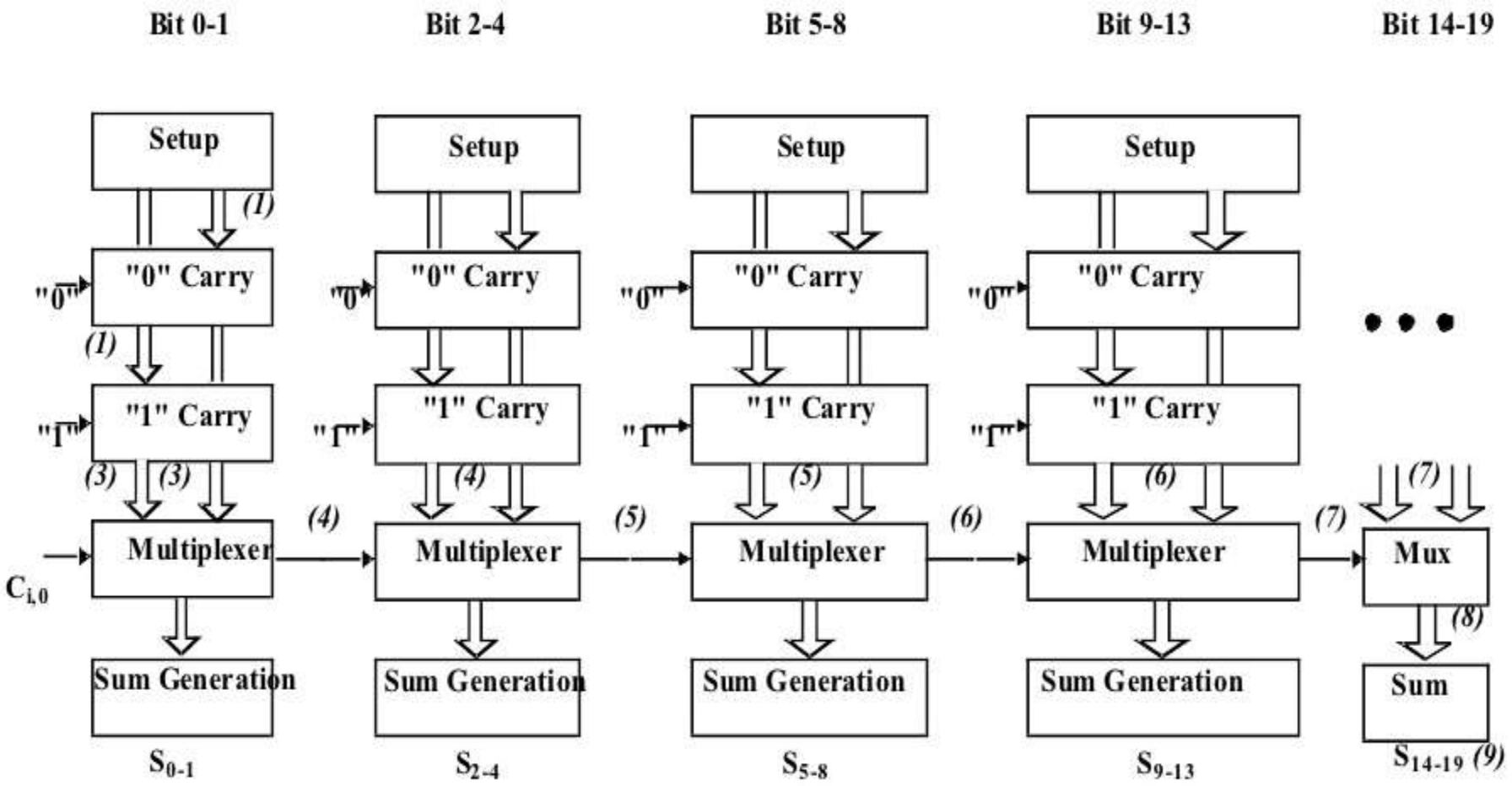
**Problem**:

Determine the delay of a 16-bit linear carry select adder by using unit delays for all cells

# Drawback of carry select adder



# Square root carry select adder



# Propagation delay

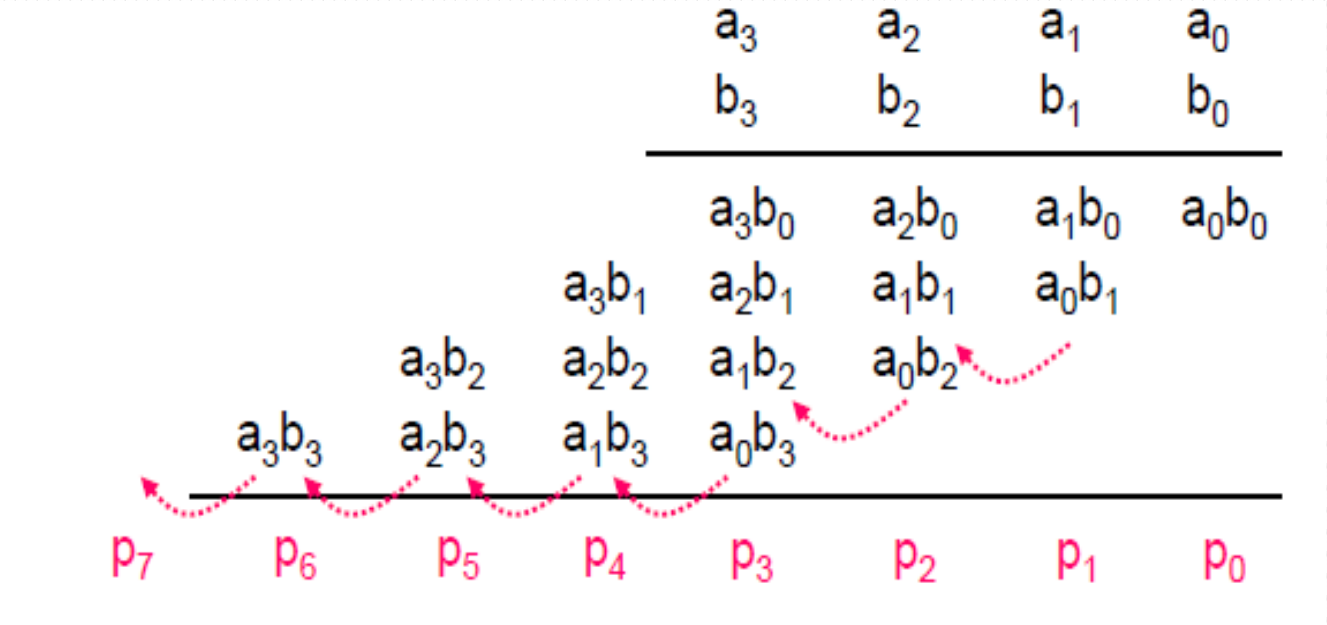
$$t_{add} = t_{setup} + Mt_{carry} + (\sqrt{2N})t_{mux} + t_{sum}$$

- The delay is proportional to $\sqrt{N}$ for larger adders (N>>M), or $t_{add} = O(\sqrt{N})$

M – No. of bits in first stage
N – Total number of bits

# Multiplier

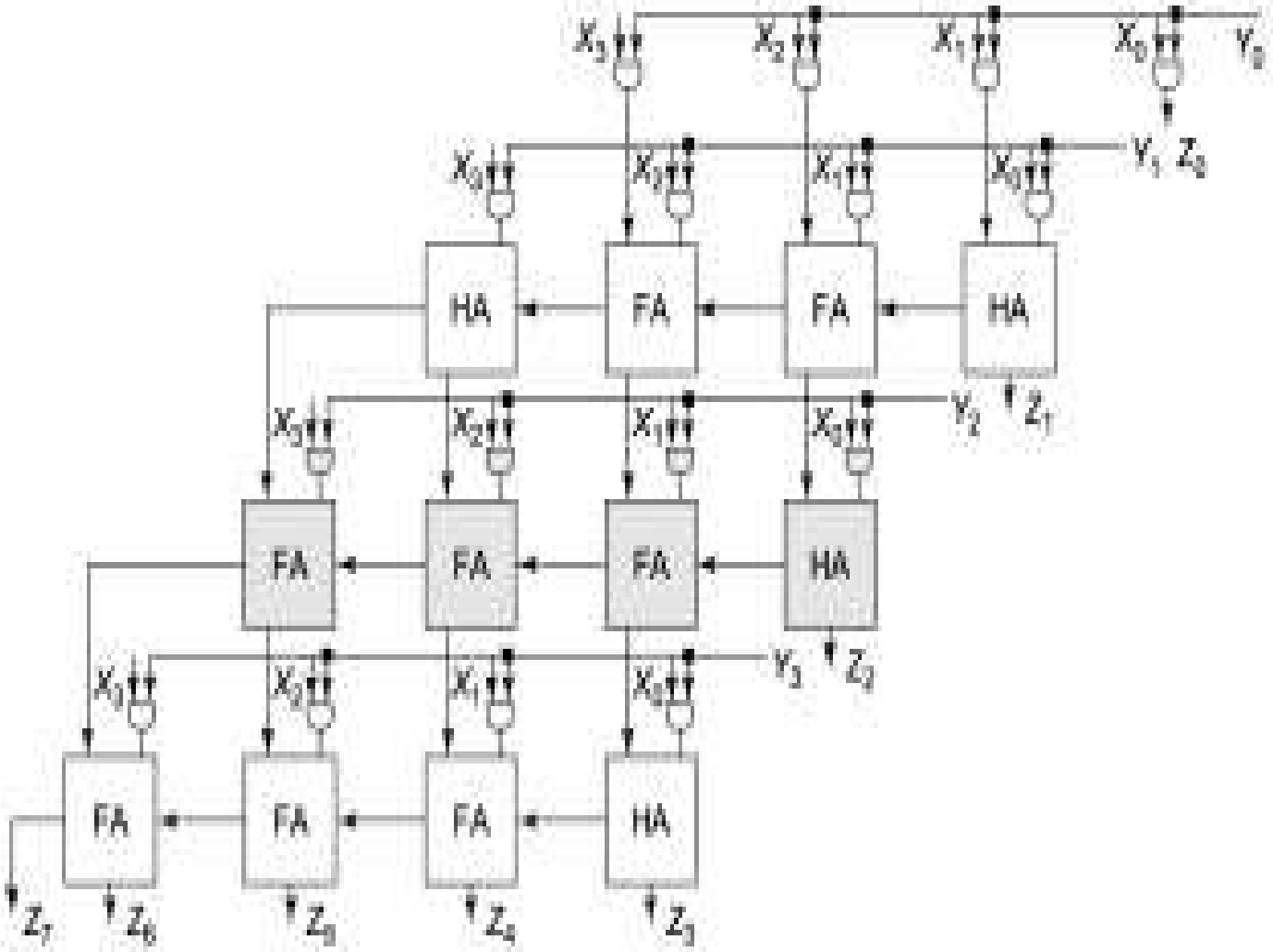
Multiplication of two 4-bit words



# Array Multiplier

- It accepts the multiplier and multiplicand and uses an array of cells to calculate the bit product $a_j * b_k$ individually in a parallel manner
- Each block requires that the bit product $a_j * b_k$ andthen add it to other conntributions in column i=(j+k)

- This produces the sum
- $P_i = \sum_{i=j+k} a_j b_k + c_{i\_1}$


$X_3$
$X_2$
$X_1$
$X_0$
$Y_0$
$Y_1$
$Z_0$
HA
FA
FA
HA
$Y_2$
$Z_1$
FA
FA
FA
HA
$Y_3$
$Z_2$
FA
FA
FA
HA
$Z_7$
$Z_6$
$Z_5$
$Z_4$
$Z_3$