EE671: VLSI Design
Spring 2024/25

Laxmeesha Somappa

Department of Electrical Engineering

IIT Bombay

laxmeesha@ee.iitb.ac.in

Lecture – 15
Arithmetic IP: Adders

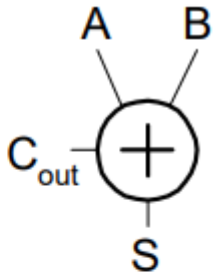
laxmeesha@ee.iitb.ac.in

1-BIT ADDER

Half Adder

$$S = A \oplus B$$

$$C_{out} = A \bullet B$$

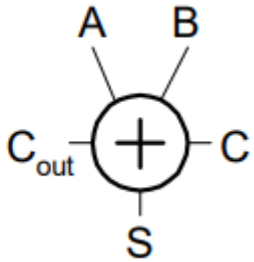


A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Full Adder

$$S = A \oplus B \oplus C$$

$$C_{out} = MAJ(A,B,C)$$



A	B	C	C_{out}	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

$$S = \overline{A} \cdot \overline{B} \cdot C_{in} + \overline{A} \cdot B \cdot \overline{C_{in}} + A \cdot B \cdot C_{in} + A \cdot \overline{B} \cdot \overline{C_{in}}$$

$$C_{out} = A \cdot B + B \cdot C_{in} + C_{in} \cdot A = A \cdot B + C_{in} \cdot (A + B)$$

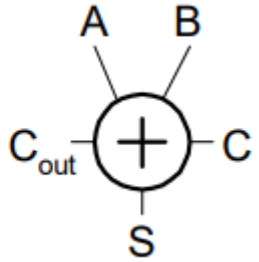
laxmeesha@ee.iitb.ac.in

1-BIT ADDER

Full Adder

$$S = A \oplus B \oplus C$$

$$C_{out} = MAJ(A,B,C)$$



$$\begin{aligned} C_{out} &= A \cdot B + C_{in} \cdot (A + B) \\ \overline{C_{out}} &= \overline{A} \cdot \overline{B} + \overline{C_{in}} \cdot (\overline{A} + \overline{B}) \end{aligned}$$

$$\begin{aligned} \text{sum} &= \overline{A} \cdot \overline{B} \cdot C_{in} + \overline{A} \cdot B \cdot \overline{C_{in}} + A \cdot \overline{B} \cdot \overline{C_{in}} + A \cdot B \cdot C_{in} \\ \overline{\text{sum}} &= A \cdot B \cdot \overline{C_{in}} + A \cdot \overline{B} \cdot C_{in} + \overline{A} \cdot B \cdot C_{in} + \overline{A} \cdot \overline{B} \cdot \overline{C_{in}} \end{aligned}$$

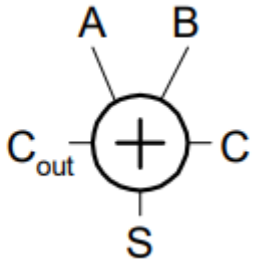
- Sum and Carry functions: are their complements
  - **Property-1:** NMOS and PMOS can have symmetric configuration!
  - **Property-2:** The hardware that generates Sum and Carry with A, B and C_{in}, will generate complement of Sum and Carry with $\overline{A}$, $\overline{B}$ and $\overline{C_{in}}$

1-BIT ADDER

Full Adder

$$S = A \oplus B \oplus C$$

$$C_{out} = MAJ(A,B,C)$$

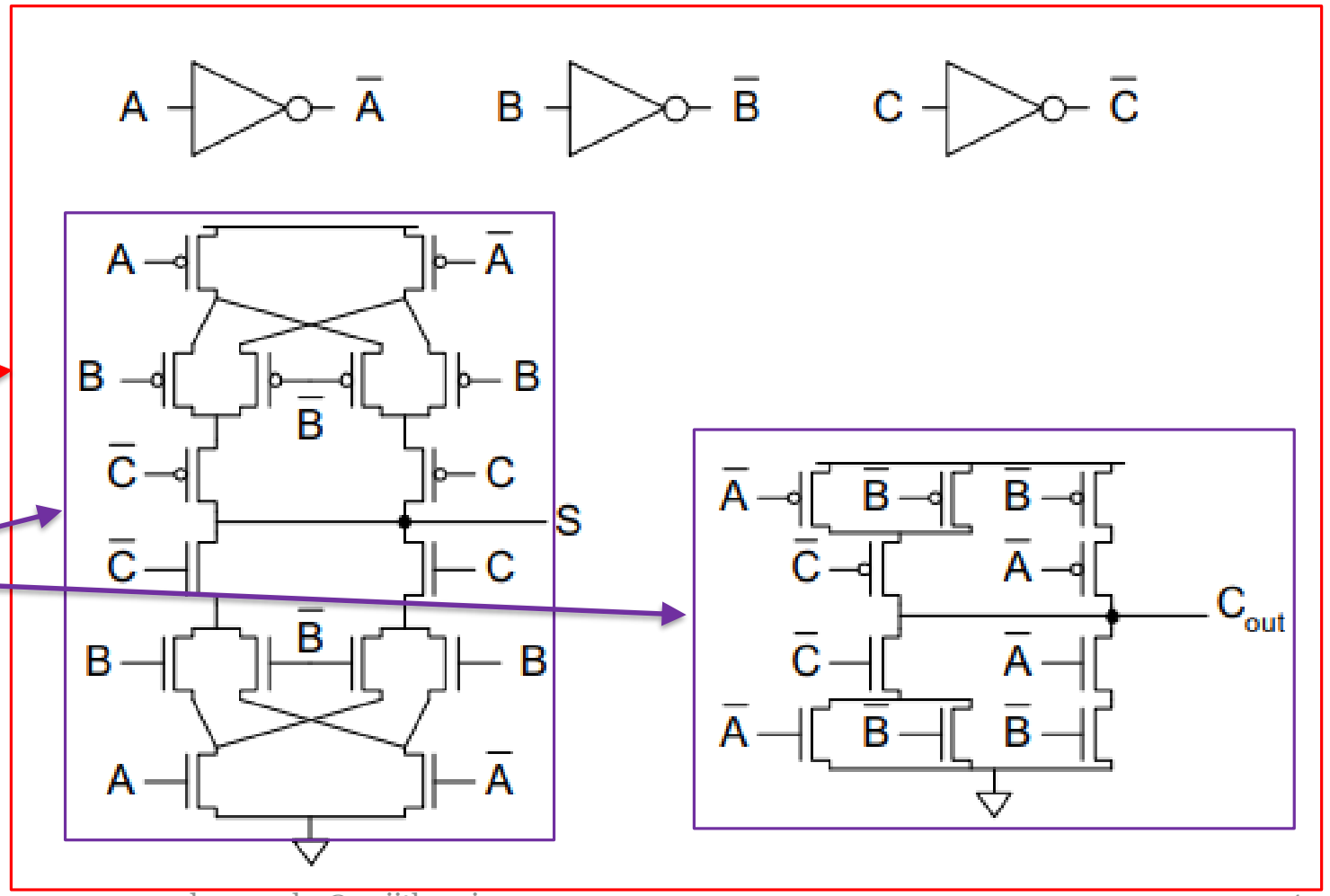


$$S = \overline{A} \cdot \overline{B} \cdot C_{in} + \overline{A} \cdot B \cdot \overline{C_{in}} + A \cdot B \cdot C_{in} + A \cdot \overline{B} \cdot \overline{C_{in}}$$

$$C_{out} = A \cdot B + B \cdot C_{in} + C_{in} \cdot A = A \cdot B + C_{in} \cdot (A + B)$$

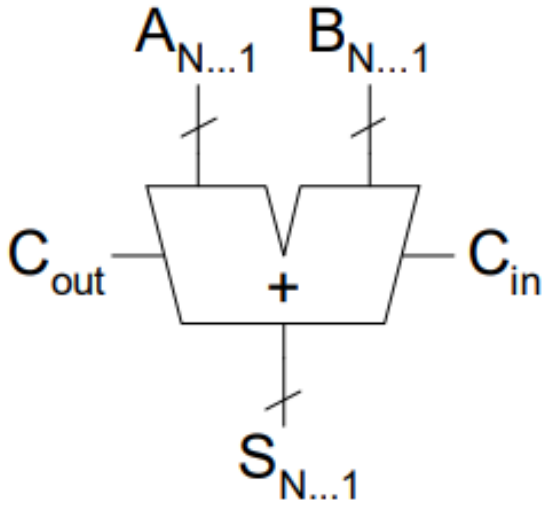
A total of 32 MOSFETs (including inverters) to generate Sum and Carry

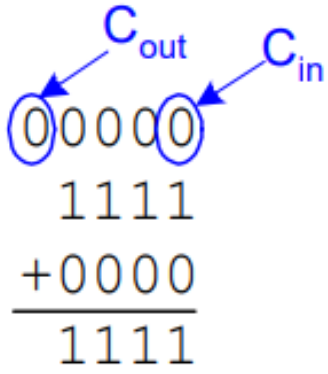
Property-1: Symmetric NMOS and PMOS side

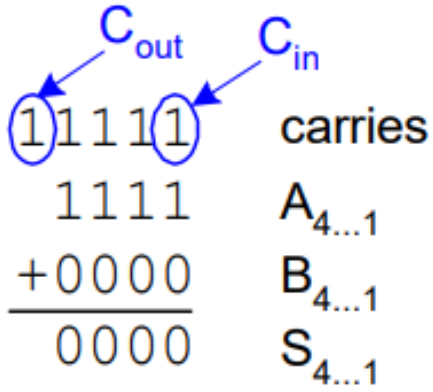


laxmeesha@ee.iitb.ac.in

N-BIT ADDER

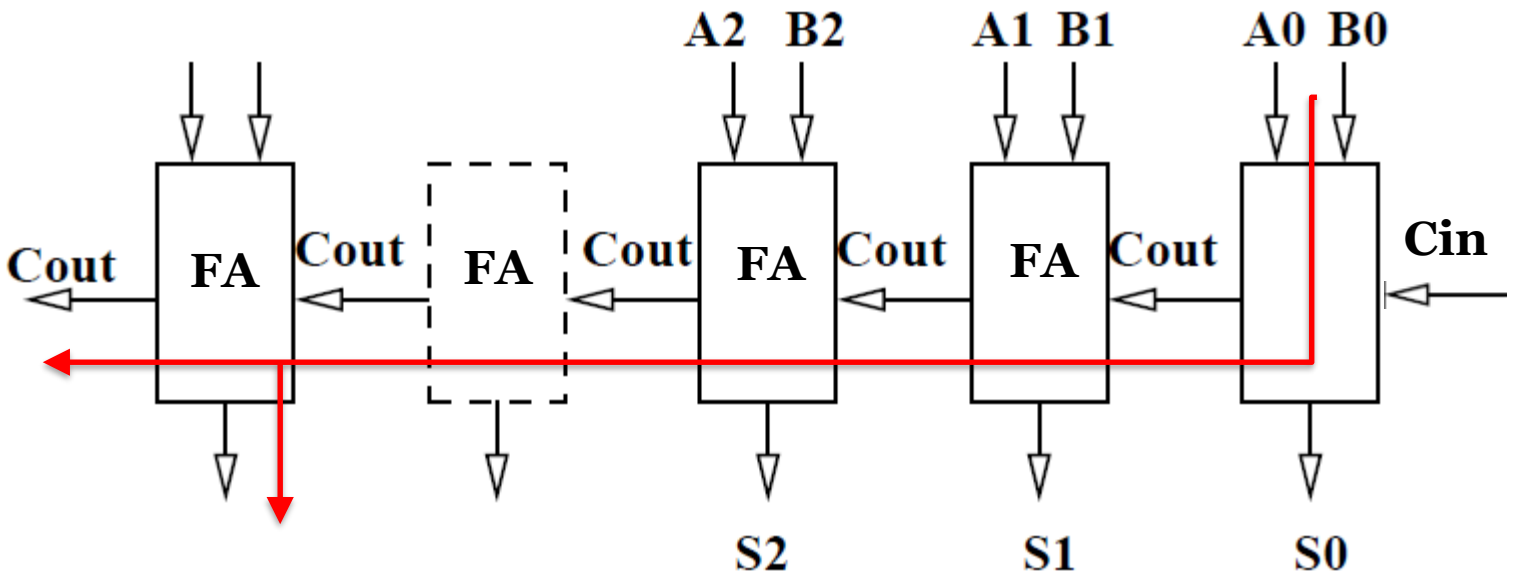






- Simple N-bit adder: carry propagate adder
  - Each sum bit depends on previous stage carry bit
  - Need to compute carry quickly
- Simplest N-bit adder: Ripple Carry Adder (RCA)

N-BIT ADDERS: RIPPLE CARRY ADDER (RCA)

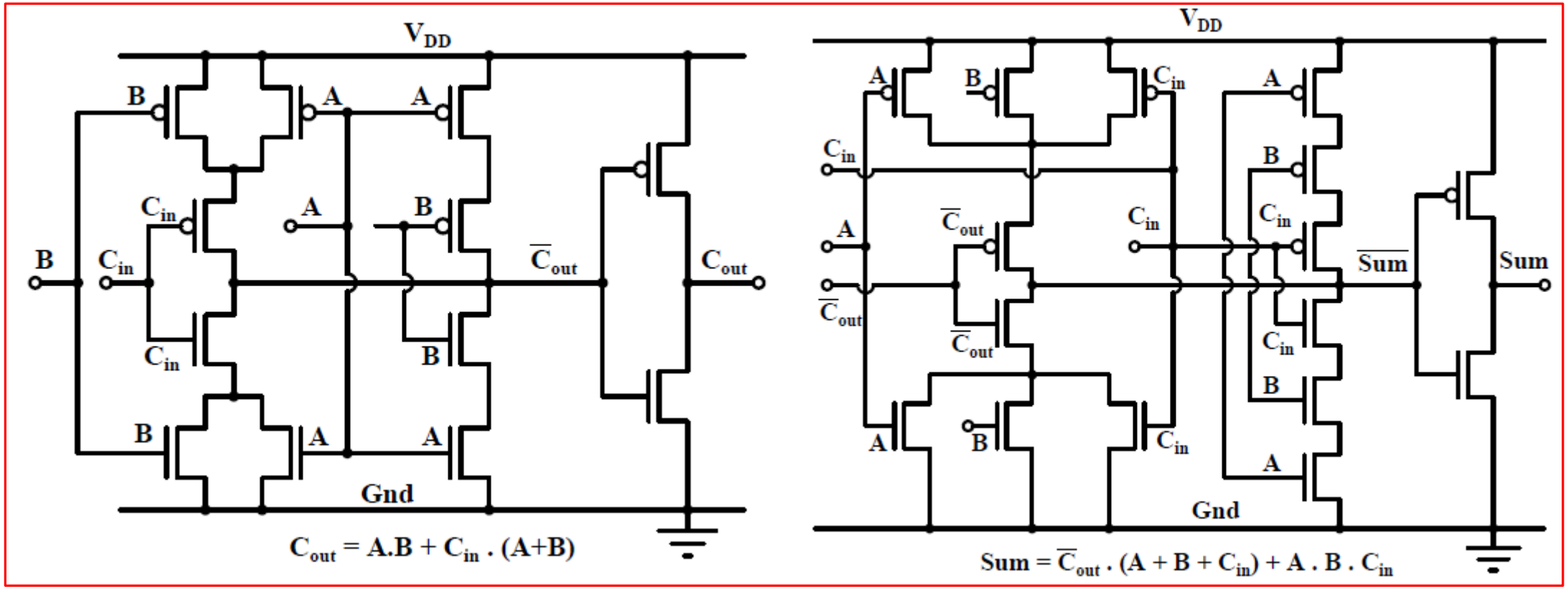


- RCA: Cascade of FAs
- Worst case delay: Carry path (carry propagates from Cin to Cout)
- $t_{adder} = (N-1)\ t_{carry} + t_{sum}$
- Speeding up the adder: reduce carry hardware delay
- Alternately: Sum is not in the critical path !!

RCA: Optimization

- Sum is not in the critical path
  - Can generate sum slowly -> if it can reduce MOSFETs

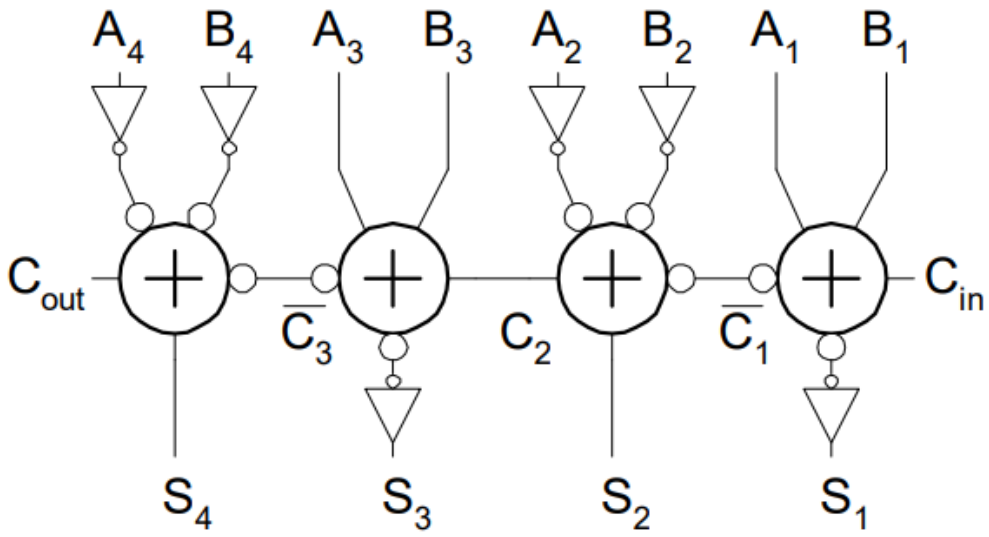
$$sum = \overline{C_{out}} \cdot (A + B + C_{in}) + A \cdot B \cdot C_{in}$$

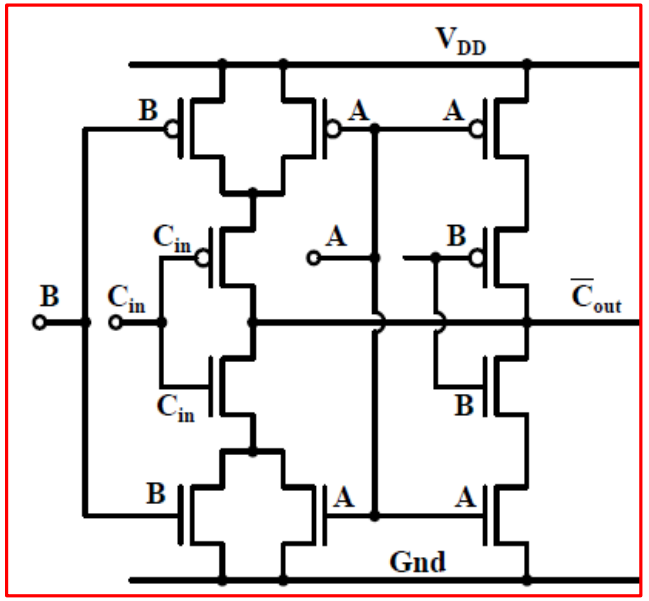


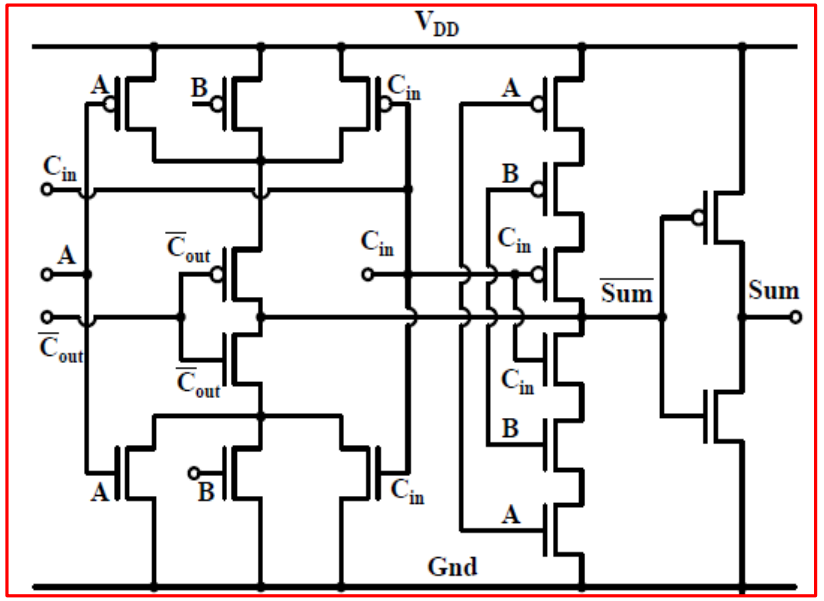
A total of 28 MOSFETs to generate Sum and Carry (Property-1)

RCA: Further Optimization?

- Use property-2 !! -> Odd FAs: complement input







- Using only property-1:
  - [28*4] = 112 MOSFETs, delay: ($N.t_{carry}$)
- Now:
  - [10+16]*2 + [10+14+4]*2 = 108 MOSFETs, delay: ($N.t_{carry\_new}$)
  - t_{carry_new}: one-inverter delay lesser than t_{carry} → i.e., ($N.\ t_{inv}$) faster

FEW OBSERVATIONS BEFORE WE PROCEED TO OTHER ADDERS

A	B	C	C_{out}	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

When both inputs are 0, irrespective of Cin, Cout is always 0.
Kill the Cout. i.e., when,

$$K \equiv \overline{A} \cdot \overline{B} \quad , \text{ Cout} = 0$$

Other input cases (only one input is 0/1), Cout = Cin.
Propagate Cout. i.e., when,

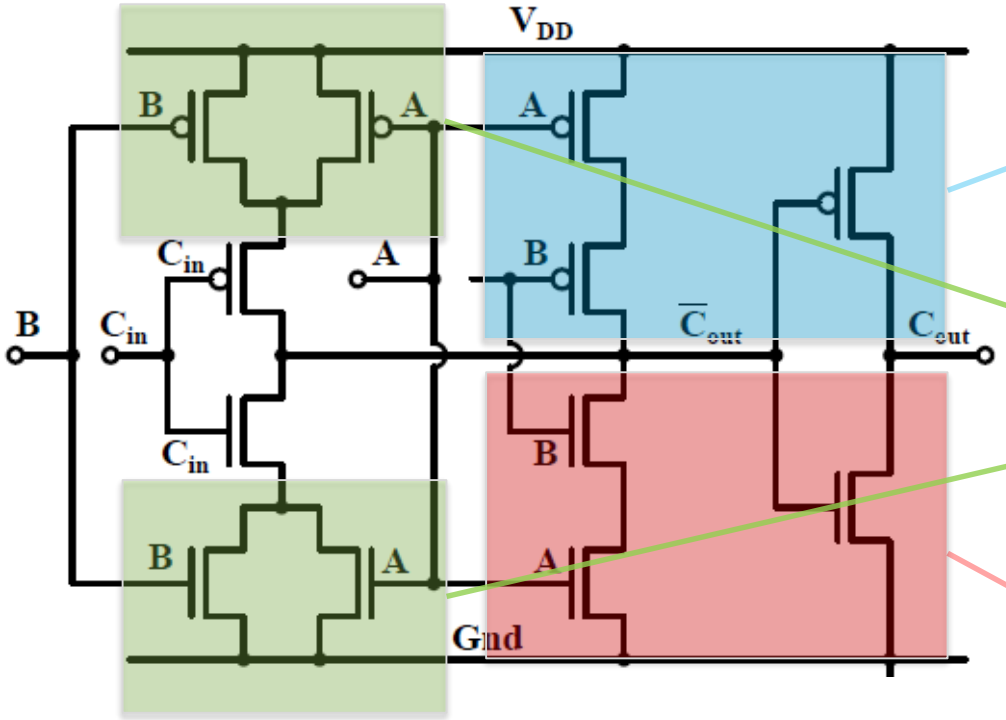
$$P \equiv A \oplus B \quad , \text{ Cout} = \text{Cin}$$

When both inputs are 1, irrespective of Cin, Cout is always 1.
Generate Cout. i.e., when,

$$G \equiv A.B \quad , \text{ Cout} = 1$$

Sum = $P \oplus C$ and Cout = $G + P.C$

GENERATING P AND G



When both inputs are 0, irrespective of Cin, Cout is always 0.
Kill the Cout. i.e., when,

$K \equiv \overline{A} \cdot \overline{B}$, Cout = 0

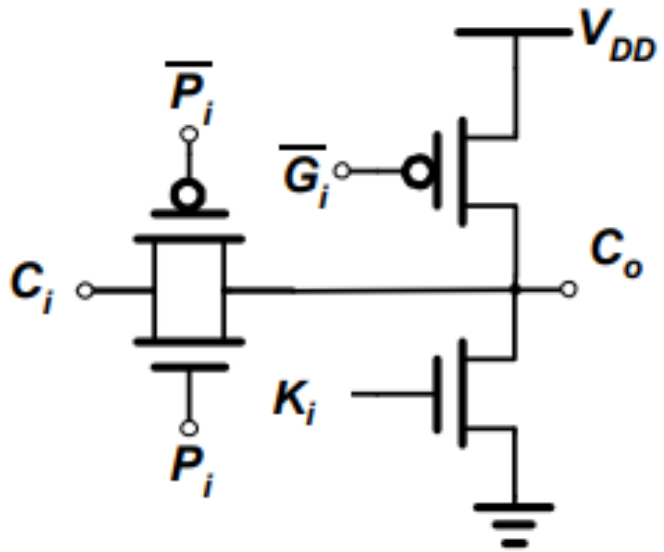
Other input cases (only one input is 0/1), Cout = Cin.
Propagate Cout. i.e., when,

$P \equiv A \oplus B$, Cout = Cin

When both inputs are 1, irrespective of Cin, Cout is always 1.
Generate Cout. i.e., when,

$G \equiv A.B$, Cout = 1

GENERATING P AND G: MANCHESTER CARRY CHAIN



G,P,K can be precomputed in parallel

When both inputs are 0, irrespective of Cin, Cout is always 0.
Kill the Cout. i.e., when,

$$K \equiv \overline{A} \cdot \overline{B} \quad , \text{ Cout} = 0$$

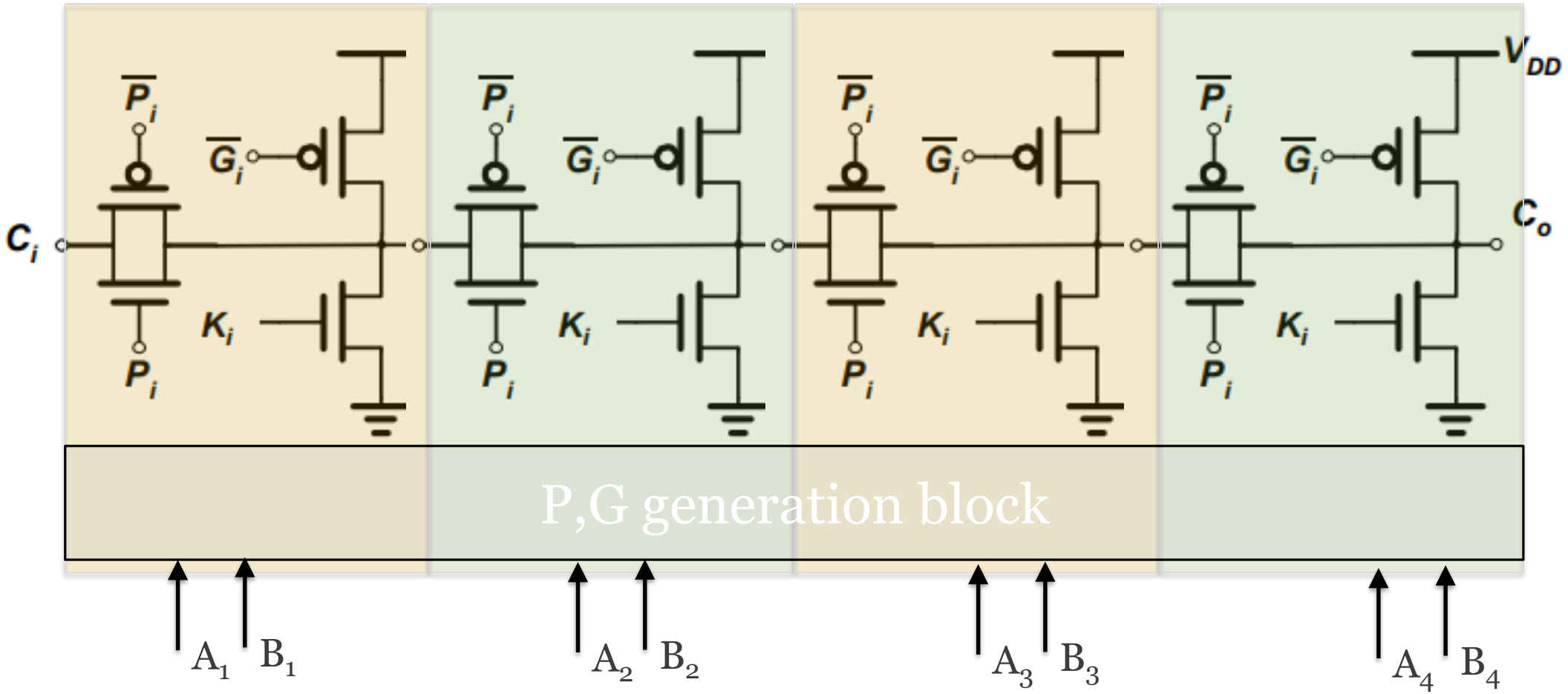
Other input cases (only one input is 0/1), Cout = Cin.
Propagate Cout. i.e., when,

$$P \equiv A \oplus B \quad , \text{ Cout} = \text{Cin}$$

When both inputs are 1, irrespective of Cin, Cout is always 1.
Generate Cout. i.e., when,

$$G \equiv A.B \quad , \text{ Cout} = 1$$

4-BIT MANCHESTER CARRY CHAIN

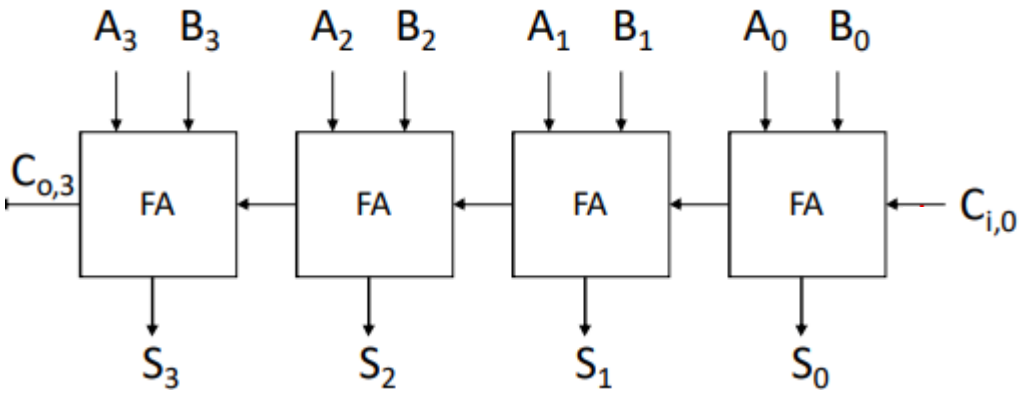


- Worst case delay: All "P" are 1
- Series of RC network → delay high in the absence of static inverter

The Carry recurrence

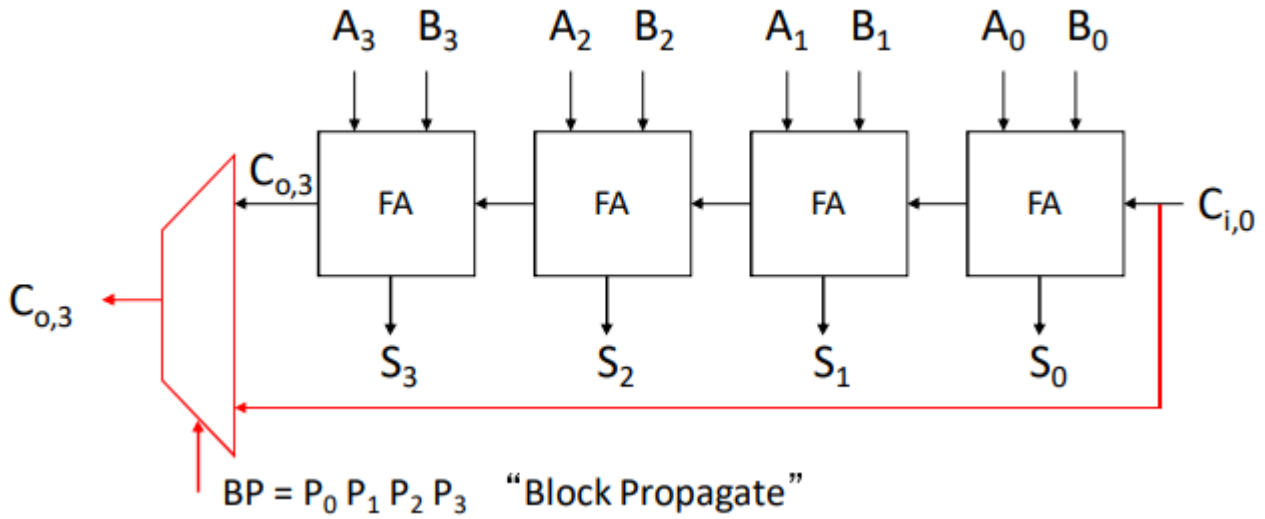
Recurrence: $C_{i+1}=G_i+P_iC_i$

$$C_1 = G_0 + P_0C_0$$
$$C_2 = G_1 + P_1G_0 + P_1P_0\,C_0$$
$$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0\,C_0$$
$$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0\,C_0$$



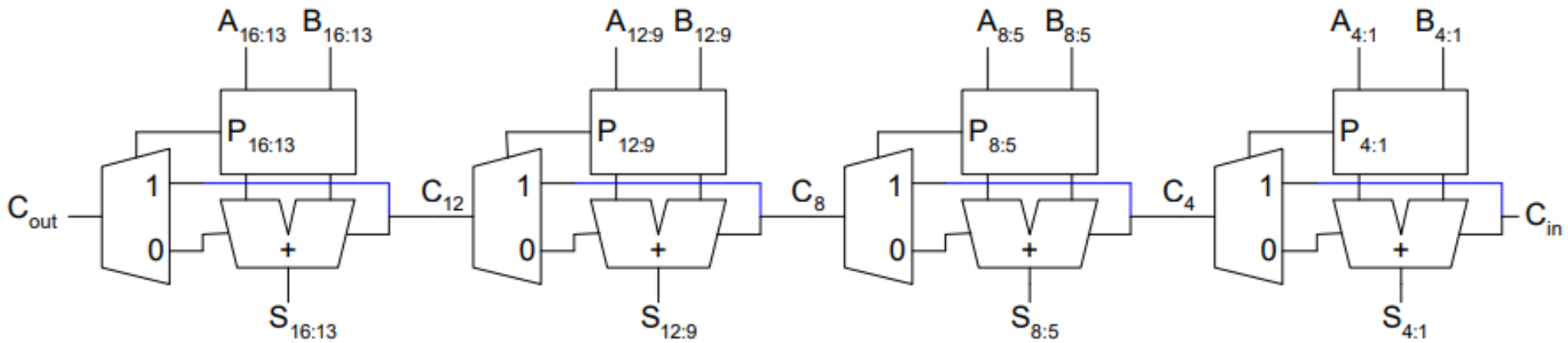
- Consider a 4 bit adder.
- The final Cout (C_4 in this case) can be directly generated from C_0 (neglecting the hardware complexity)
- Practically: to generate C_4 we need high fan-in CMOS gates (max 5 input AND)
- Generating C4 will have the same order of delay as RCA.

Carry Skip (Bypass) Adder



- Compute P and G for each bit parallelly
- If $P_0.P_1.P_2.P_3 = 1$, then carry is propagated (critical path delay)
- If a G is generated in any FA: no propagation → not critical path
- Hence, bypass the critical path through mux !
- No use in its standalone mode. But consider the next case

CARRY SKIP (BYPASS) ADDER



$$S = P \oplus C \text{ and } C = G + P.Cin$$

- Compute $P_i.P_{i+1}.P_{i+2}.P_{i+3}$. These will be computed at the same time across 4 stages.
- Propagate signal are pre-computed for stage-2 onwards
- Critical path can be bypassed from stage-2 onwards