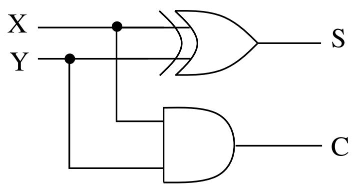
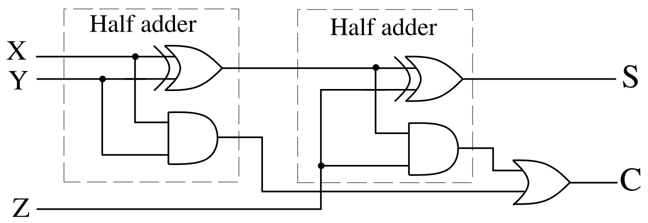
Adder

- Adder delay is dominated by carry chain.
- Carry chain analysis must consider transistor, wiring delay.
- Modern VLSI favors adder designs which have compact carry chains.
- A **half adder** adds two bits, X and Y, to produce a 1-bit **sum**, S, and a 1-bit **carry**, C.
  - $S = (\bar{X}Y + X\bar{Y}) = X \oplus Y$; $C = XY$.
- A **full adder** adds two bits, X and Y, and one **carry input**, Z, to produce a 1-bit **sum**, S, and a 1-bit **carry**, C.
  - $S = (\bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XYZ) = X \oplus Y \oplus Z$;
    $C = XY + XZ + YZ = XY + Z(X \oplus Y)$.



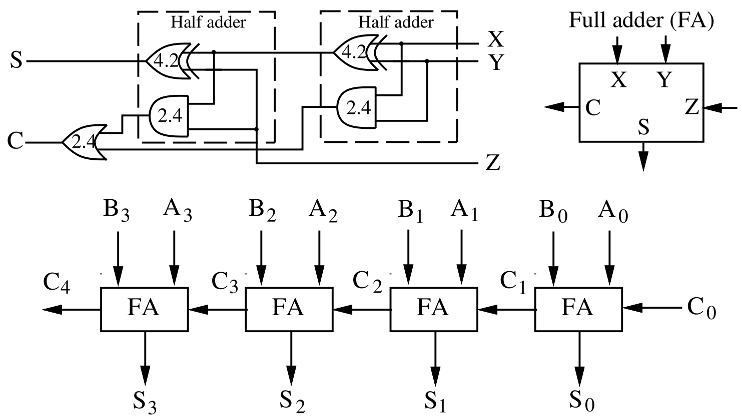
Half adder



Full adder

Ripple Carry Adder

- 4-bit ripple carry adder example: Inputs: C_0, $A = A_{n-1} \dots A_0$, $B = B_{n-1} \dots B_0$; Outputs: $s_{n-1} \dots s_0$ and $c_n \dots c_1$.



- Propagation delays: $C_0(A_0, B_0) \rightarrow C_1$: 4.8(9.0) ns; $C_0(A_0, B_0) \rightarrow C_2$: 9.6(13.8) ns; $C_0(A_0, B_0) \rightarrow C_3$: 14.4(18.6) ns; $C_0(A_0, B_0) \rightarrow C_4$: 19.2(23.4) ns.
- **Question:** Can we improve the delays?
  - The carries are the bottleneck!

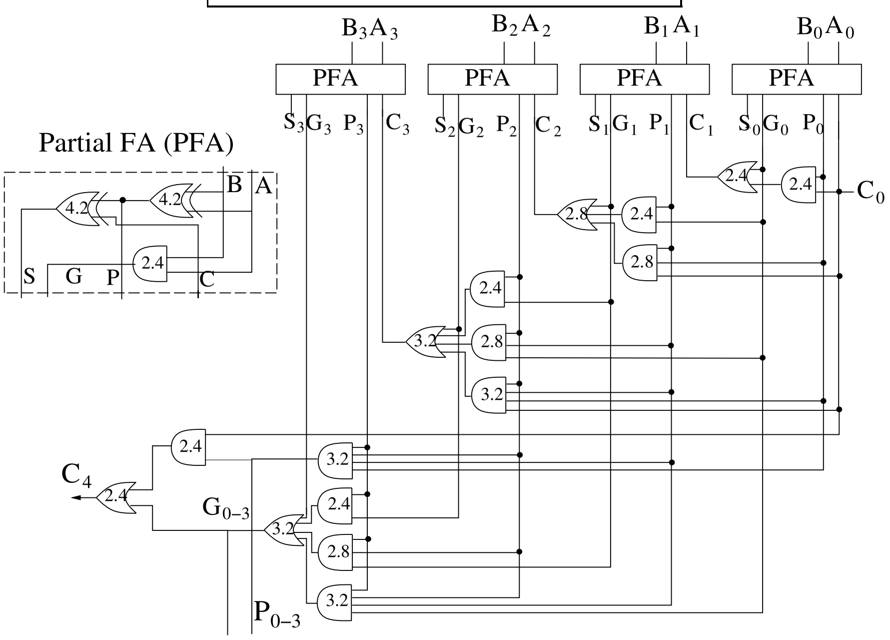
Carry Lookahead Adder

- **Observation:** Major part of the expressions for each carry in the chain can be **precomputed**.
  - Exp: Add $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$.
  - **Carry-generate function:** $G_i = A_iB_i$; **carry-propagate function:** $P_i = A_i \oplus B_i$. Then, the 4 carries C_1, C_2, C_3, C_4 can be expressed as follows:

$$
\begin{aligned}
C_1 &= G_0 + P_0C_0 && (1)\\
C_2 &= G_1 + P_1C_1 && (2)\\
&= G_1 + P_1G_0 + P_1P_0C_0\\
C_3 &= G_2 + P_2C_2 && (3)\\
&= G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0\\
C_4 &= G_3 + P_3C_3 && (4)\\
&= G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0
\end{aligned}
$$

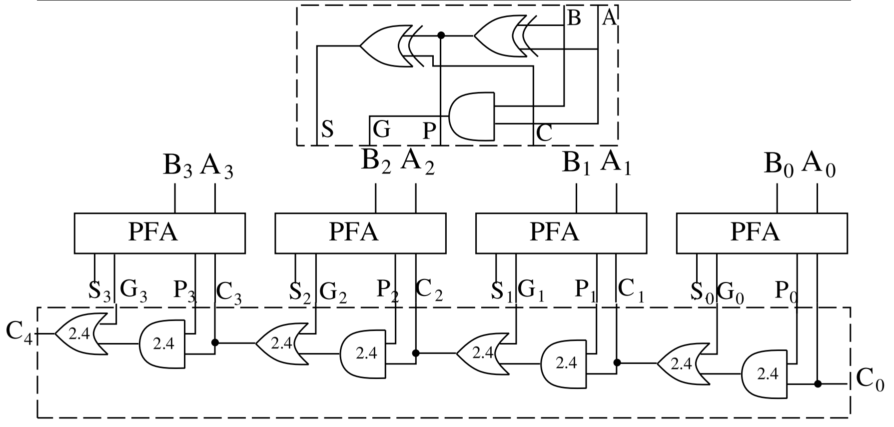
- **Carries can be computed directly from input bits A_i, B_i and input carry C_0.** No ripple effect!
- Extend to n-bit adders?
  - **Problem: Fan-in limitation?**

Carry Lookahead Adder



- **Group propagate function:** $P_{0-3} = P_3P_2P_1P_0$;
  Group generate function: $G_{0-3} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$;
  $C_4 = G_{0-3} + P_{0-3}C_0$.

Carry Lookahead vs. Ripple Carry Adders

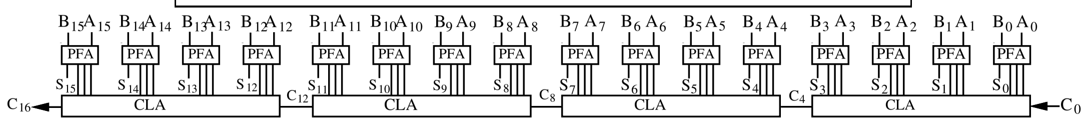


Ripple carry adder

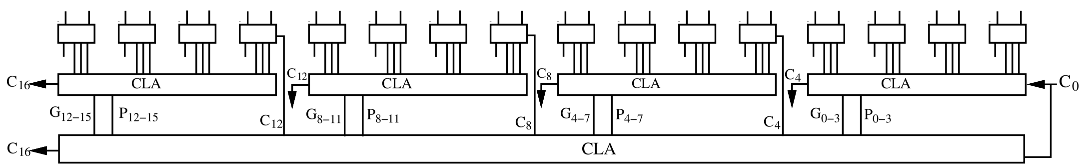
Carry chain	Ripple delay	CLA delay
$C_0(A_0, B_0)$ to C_1	4.8 (9.0)	4.8 (9.0)
$C_0(A_0, B_0)$ to C_2	9.6 (13.8)	5.6 (9.8)
$C_0(A_0, B_0)$ to C_3	14.4 (18.6)	6.4 (10.6)
$C_0(A_0, B_0)$ to C_4	19.2 (23.4)	4.8 (13.0)

- CLA is faster, but much more complex layout.

Carry Lookahead Adder Construction



Single–level CLA generator



Two–level CLA generator

- Carry chain delays for 16-bit adders:

Carry chain	Ripple delay	One-level CLA	Two-level CLA
$C_0(A_0, B_0)$ to C_4	19.2 (23.4)	4.8 (13.0)	4.8 (13.0)
$C_0(A_0, B_0)$ to C_8	38.4 (42.6)	9.6 (17.8)	5.6 (16.2)
$C_0(A_0, B_0)$ to C_{12}	57.6 (61.8)	14.4 (22.6)	6.4 (17.0)
$C_0(A_0, B_0)$ to C_{16}	76.8 (81.0)	19.2 (27.4)	4.8 (19.4)

- n-bit ripple-carry adder has $2n$ gate delays; n-bit adder with $\lceil \log_4 n \rceil$ levels of CLA needs $2\lceil \log_4 n \rceil$ gate delays to generate top-level carries.