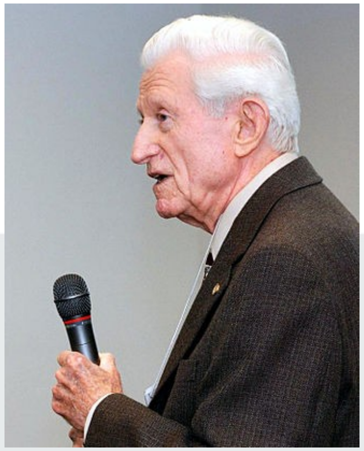
TÉCNICO LISBOA

Computer Electronics

Lecture 8: Adder Circuits – Part 2; Amdhal’s Law

Gene Ahmdhal



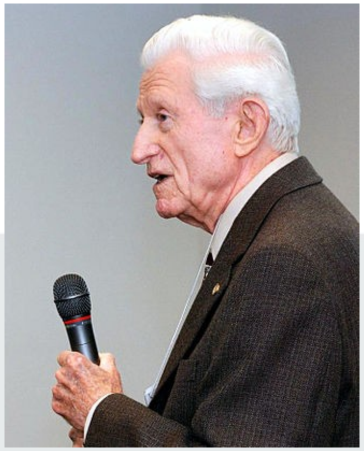
Amdhal’s Law

- One the most important computer architecture laws
- Establish the limits of Acceleration
- A program takes T seconds to execute; a part of the program takes P seconds to execute; if P is accelerated K times than the total Speed-Up SU is given by

$$SU = \frac{T\ before\ acceleration\ of\ P}{T'\ after\ acceleration\ of\ P}$$

$$\boxed{SU = \frac{T}{T - P + \frac{P}{K}}}$$

Gene Ahmdhal



Amdhal’s Law Implications

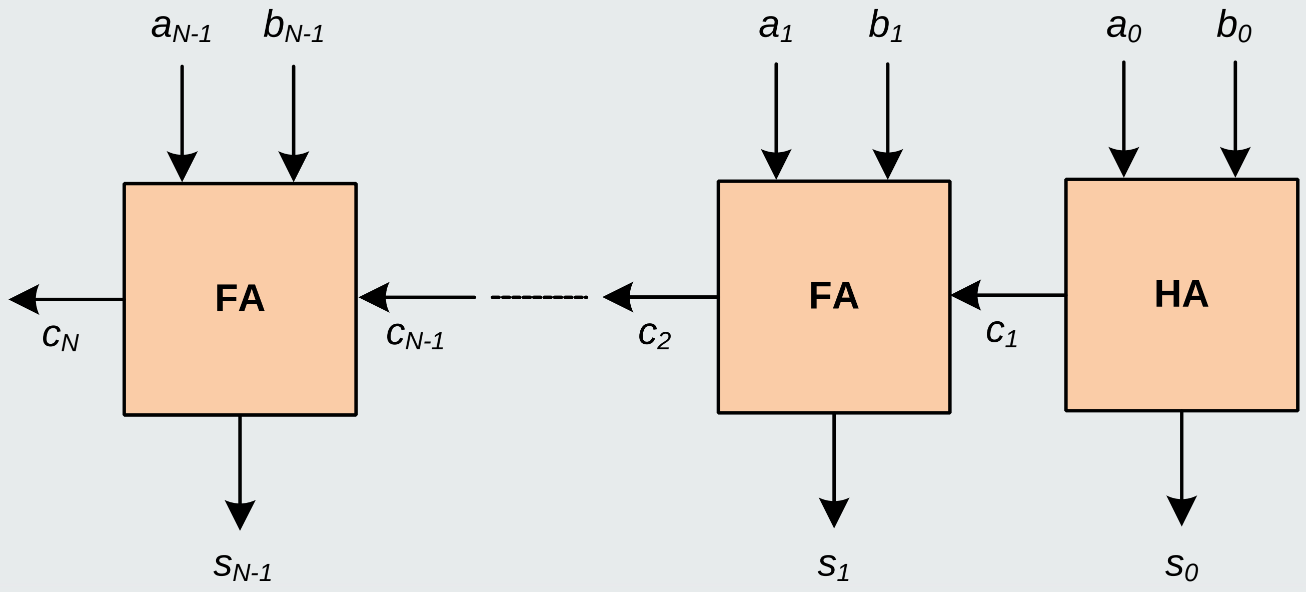
- If P is small do not bother to accelerate it
- Many times, accelerating a task 10 times is as good as accelerating the same task 10000000 times!

$$SU = \frac{T\ before\ acceleration\ of\ P}{T\ after\ acceleration\ of\ P}$$

$$\boxed{SU = \frac{T}{(T-P)+\frac{P}{K}}}$$

Ripple Carry Adder problem: how long will it take for the carry to ripple?

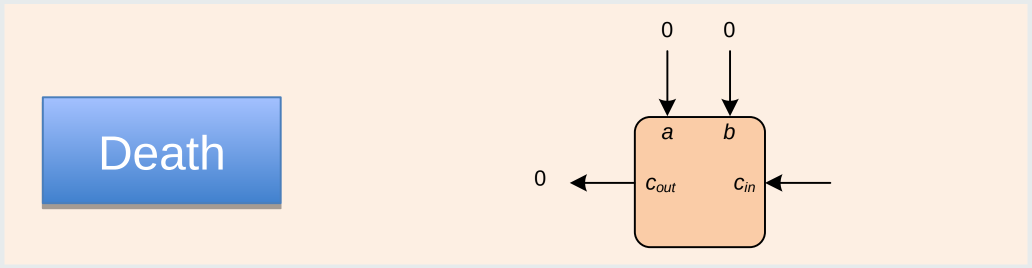
- Answer: it will take the time to traverse all single bit adder blocks…. !

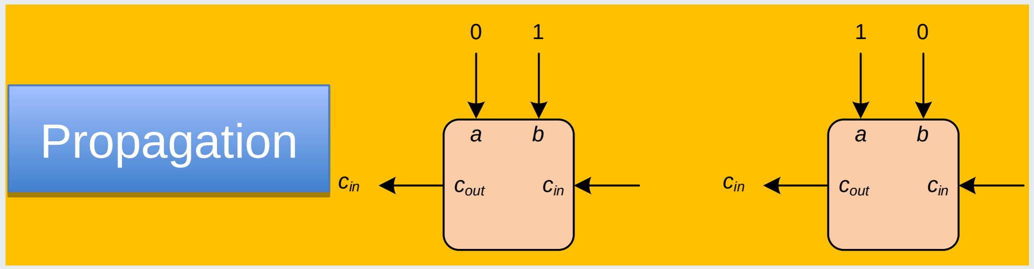


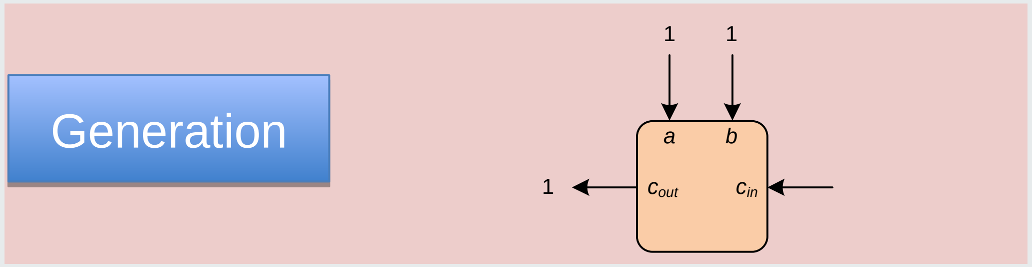
The Carry Look-ahead Adder (CLA)

- Speeds up carry computation by using a tree structure instead of a linear structure
- Linear structure length: N
- Tree structure height: log(N)
- Think of N=1000: a binary tree height is only 10 – dramatic speedup!!

Carry prediction from inputs







Mathematically

$$g_i = a_i \cdot b_i$$

$$p_i = a_i \oplus b_i$$

$$c_{i+1} = g_i + p_i \cdot c_i$$

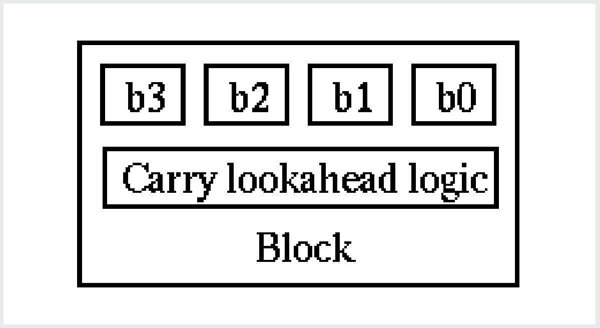
Carry recursive computation

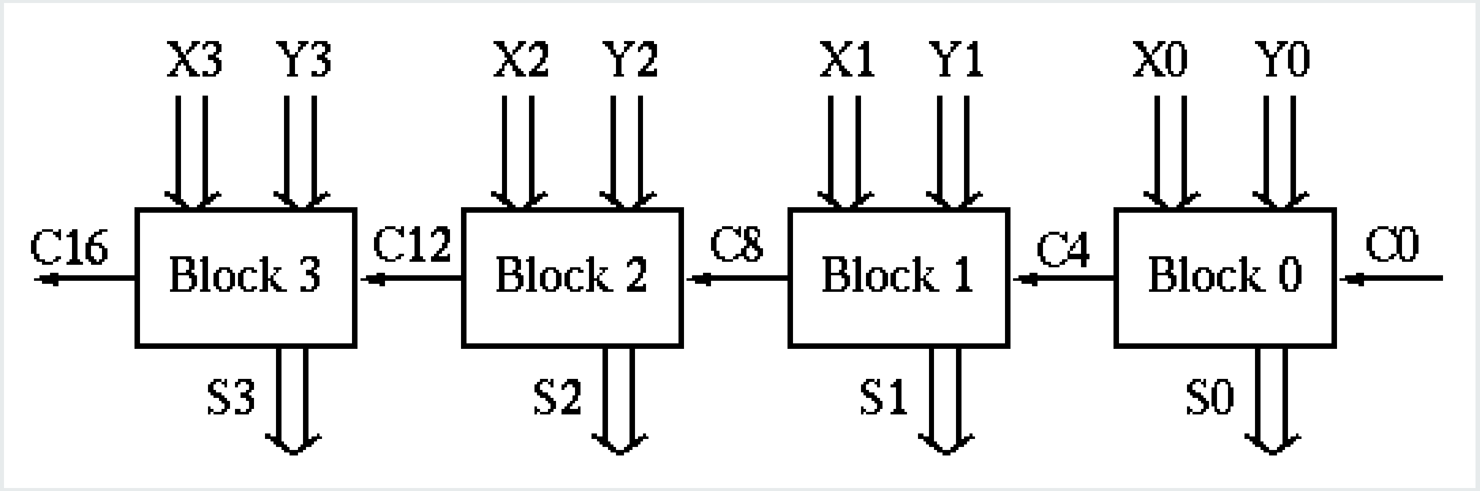
$$
\begin{aligned}
c_{i+1} &= g_i + p_i \cdot c_i \\
&= g_i + p_i \cdot \left(g_{i-1} + p_{i-1} \cdot c_{i-1}\right) \\
&= g_i + p_i \cdot g_{i-1} + p_i \cdot p_{i-1} \cdot \left(g_{i-2} + p_{i-2} \cdot c_{i-2}\right) \\
&\;\ldots.. \\
&= g_i + p_i \cdot g_{i-1} + p_i \cdot p_{i-1} \cdot g_{i-2} + p_i \cdot p_{i-1} \cdot p_{i-2} \cdot g_{i-3} + \ldots + p_i \cdot p_{i-1} \cdots p_1 \cdot p_0 \cdot c_0
\end{aligned}
$$

- Can use a 2-level SUM OF PRODUCTS circuit
- Multi-input OR and AND gates
- Implement as 2-input gate trees!
- CARRY AS TREE FUNCTION OF ALL INPUTS!

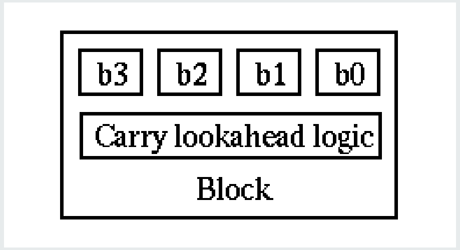
Block Carry Look-ahead Adder

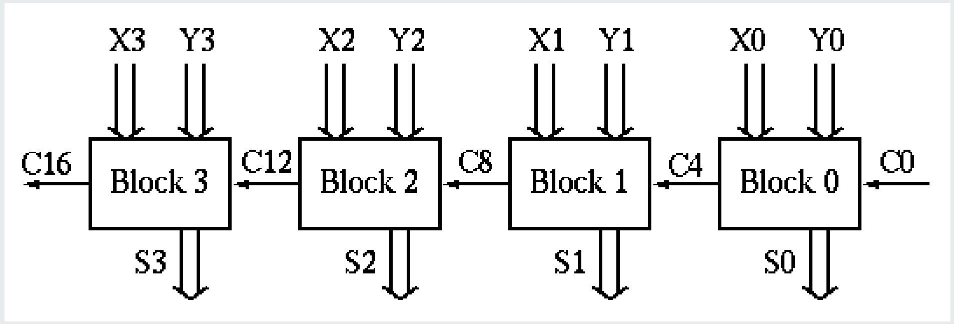
Combine CLA and RCA

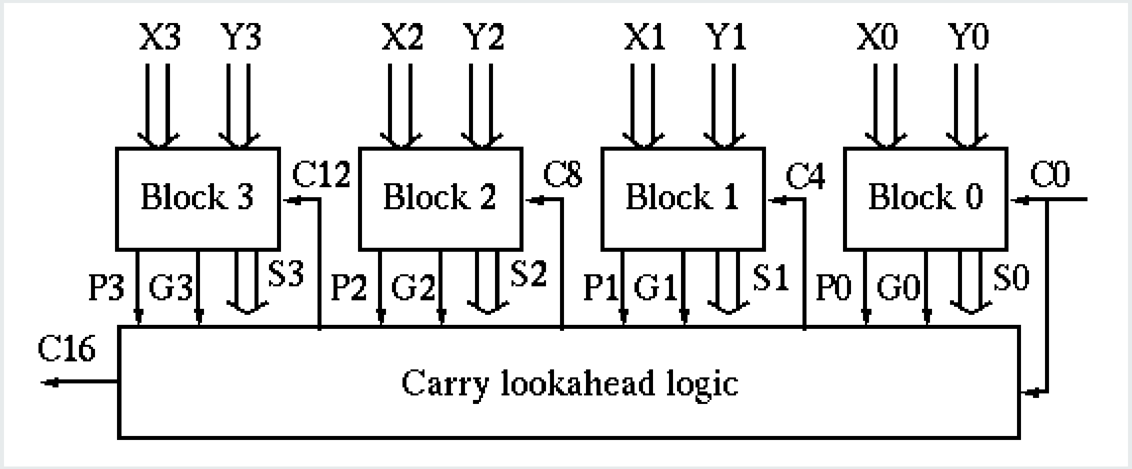




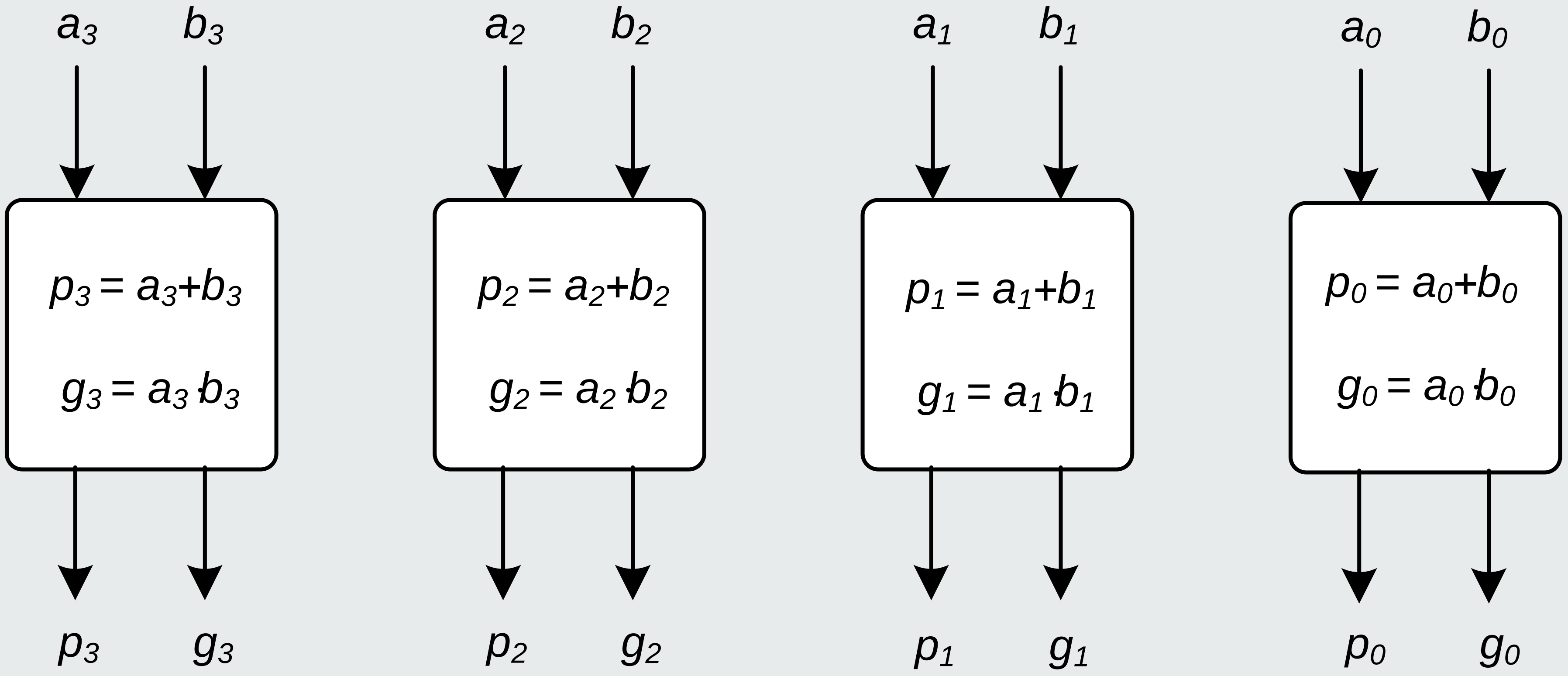
Using multiple levels



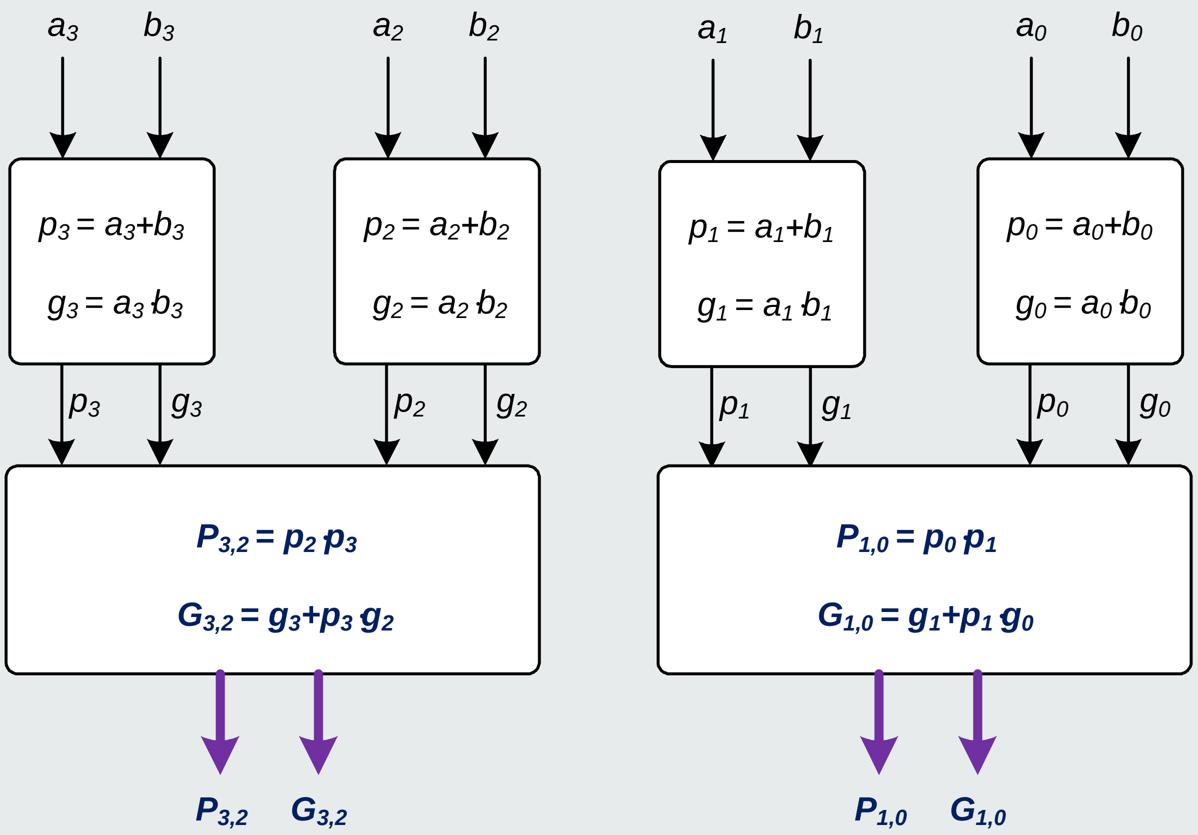




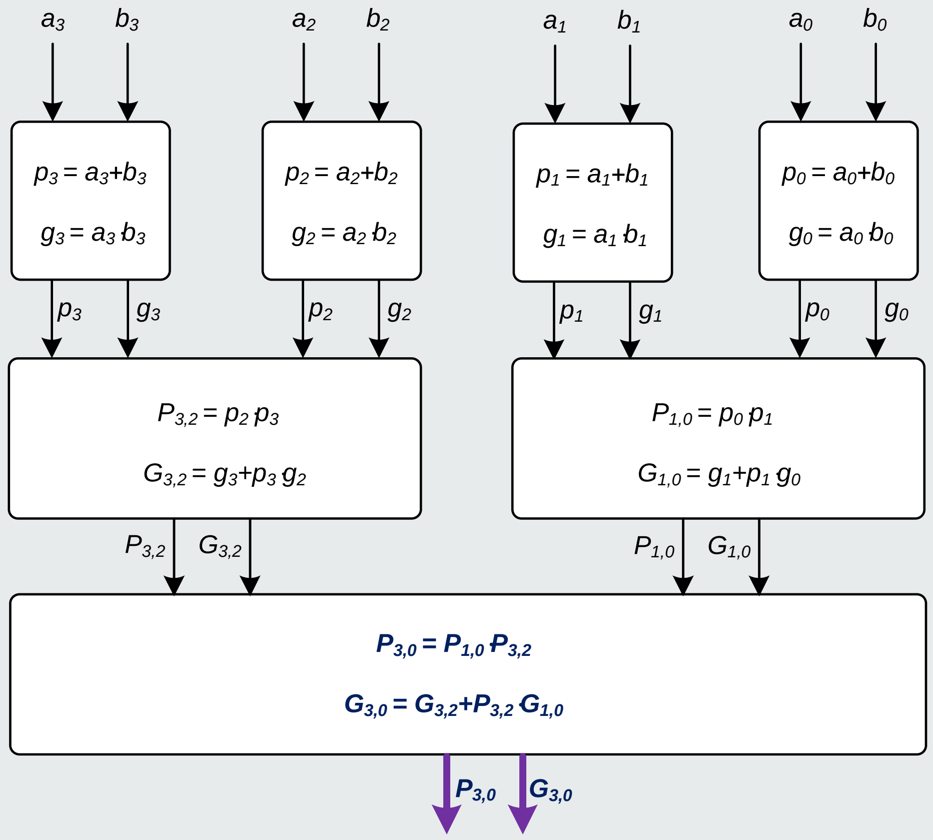
Tree Carry Look-Ahead Adder



Tree Carry Look-Ahead Adder



Carry Look-Ahead Tree

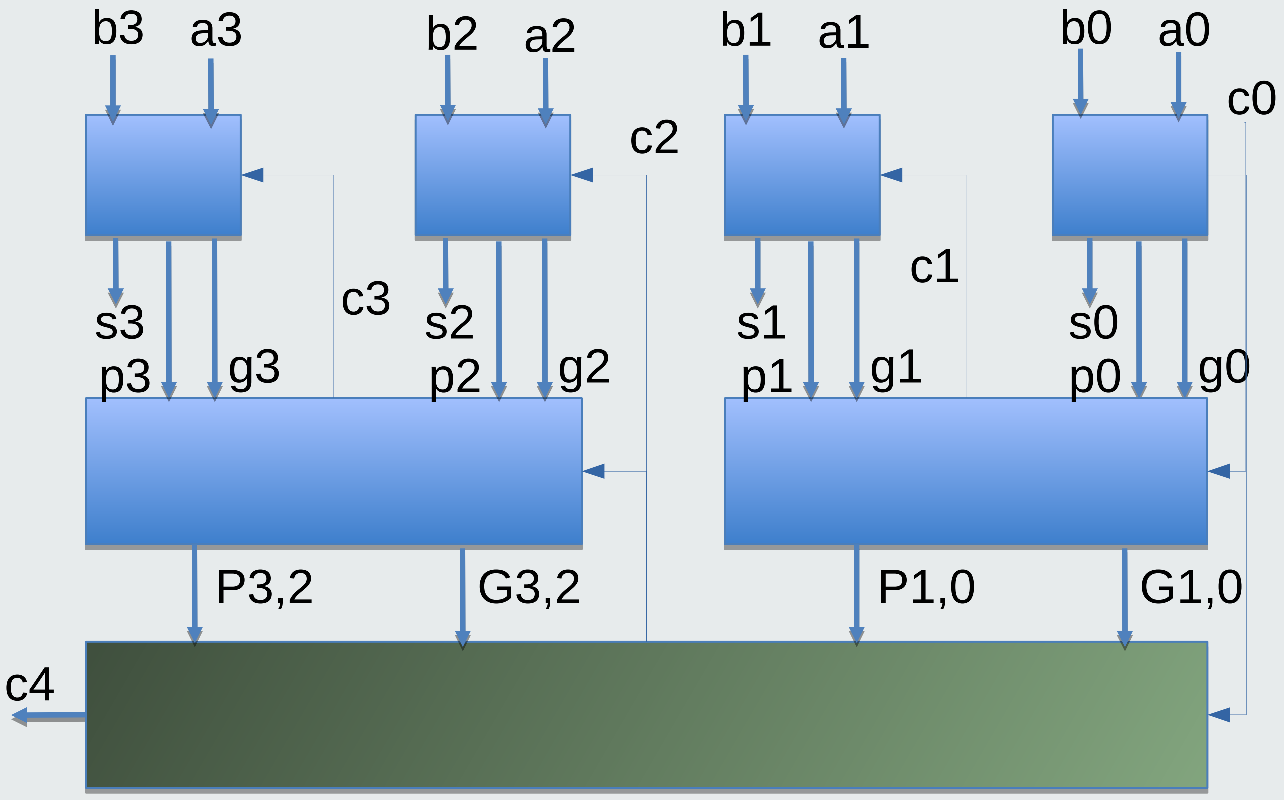


a_3 b_3 a_2 b_2 a_1 b_1 a_0 b_0
$p_3 = a_3+b_3$
$g_3 = a_3 \cdot b_3$
$p_2 = a_2+b_2$
$g_2 = a_2 \cdot b_2$
$p_1 = a_1+b_1$
$g_1 = a_1 \cdot b_1$
$p_0 = a_0+b_0$
$g_0 = a_0 \cdot b_0$
p_3 g_3 p_2 g_2 p_1 g_1 p_0 g_0
$P_{3,2} = p_2 \cdot p_3$
$G_{3,2} = g_3+p_3 \cdot g_2$
$P_{1,0} = p_0 \cdot p_1$
$G_{1,0} = g_1+p_1 \cdot g_0$
$P_{3,2}$ $G_{3,2}$ $P_{1,0}$ $G_{1,0}$
$P_{3,0} = P_{1,0} \cdot P_{3,2}$
$G_{3,0} = G_{3,2}+P_{3,2} \cdot G_{1,0}$
$P_{3,0}$ $G_{3,0}$

Carry out of a block

$$c_{k+1} = G_{k,i} + P_{k,i} \cdot c_i$$

4-bit CLA



s0 = a0 XOR b0 XOR c0
p0 = a0 XOR b0
g0 = a0*b0

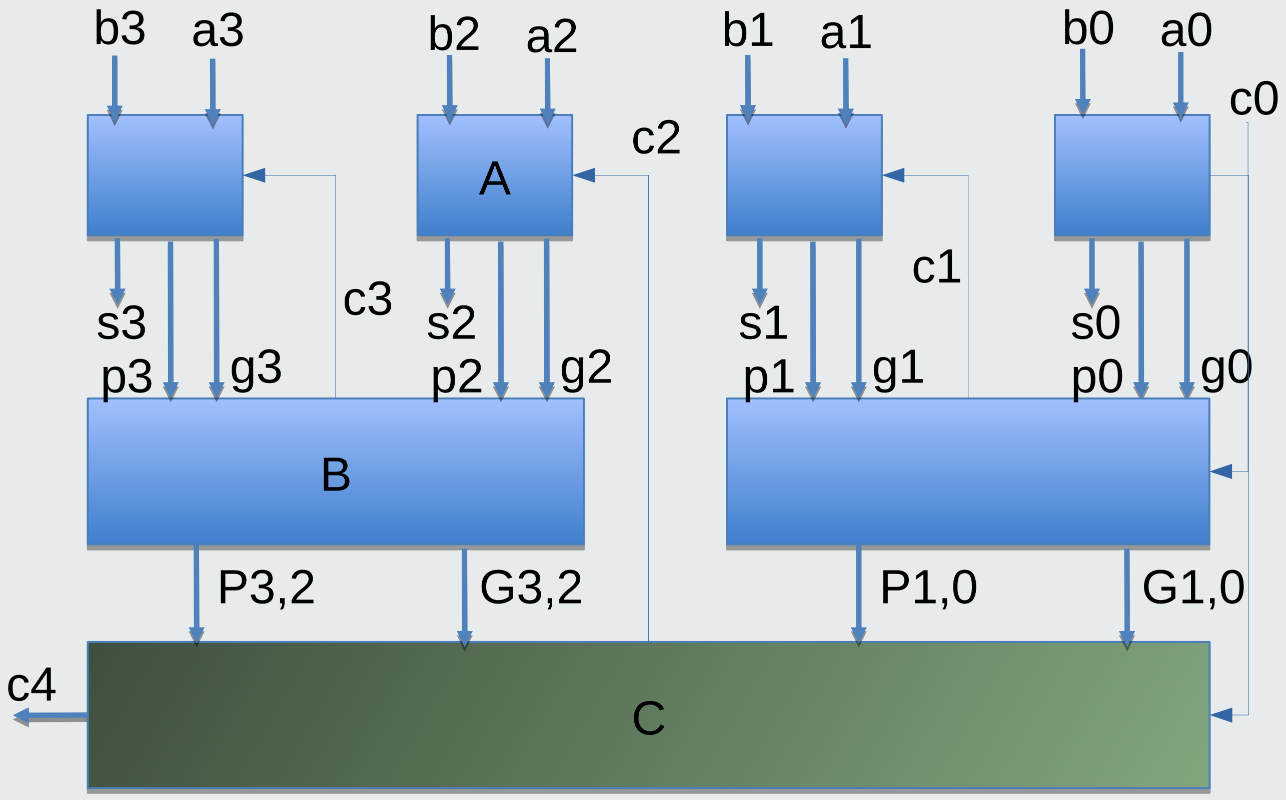
P1,0 = p1*p0
G1,0 = g1+p1*g0
c1 = g0+p0*c0

P3,0=P3,2*P1,0
G3,0=G3,2+P3,2*G1,0
c2 = G1,0+P1,0*c0

Critical Path Length = 2 * log(N) * (Node delay)

c4 = G3,0+P3,0*c0

Worksheet 8: design this Verilog



s0 = a0 XOR b0 XOR c0
p0 = a0 XOR b0
g0 = a0*b0

P1,0 = p1*p0
G1,0 = g1+p1*g0
c1 = g0+p0*c0

P3,0=P3,2*P1,0
G3,0=G3,2+P3,2*G1,0
c2 = G1,0+P1,0*c0

Critical Path Length = 2 * log(N) * (Node delay)

c4 = G3,0+P3,0*c0