# Carry Lookahead Adder Notes

## *Carry Lookahead Adder (CLA)*

A fast but costly adder.

Speed due to computing carry bit $i$ without waiting for carry bit $i-1$.

## These Notes

Intended to supplement other sources.

For a basic introduction see Brown & Vranesic 3rd Edition Section 3.4.

Describe an ordinary (also called *flat*) and *hierarchical* CLA.

Provide simple delay and cost estimates.

The cost analysis for the hierarchical CLAs has been omitted for now.

## *Carry Lookahead Cell (CLC)*

Operates on single bits.

### Ports for CLC $i$

Operands and Carry In

$a$, $b$, $c$

Sum

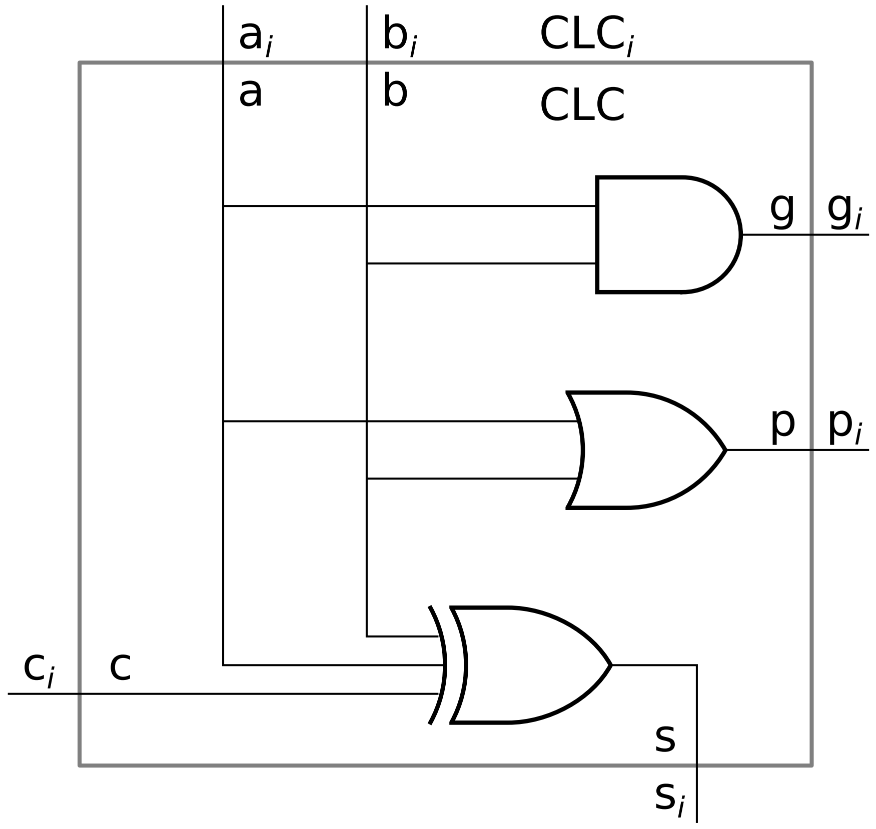
$s = a \oplus b \oplus c.$

Propagate Signal

$p = a + b$

Generate Signal

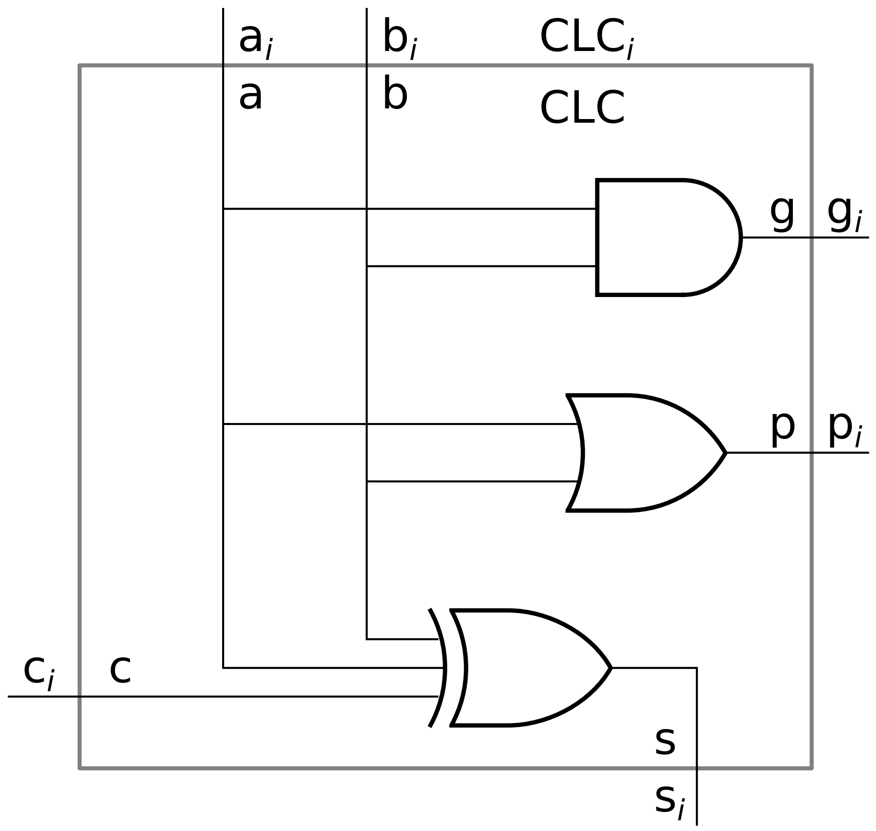
$g = ab$



## CLC Important Features

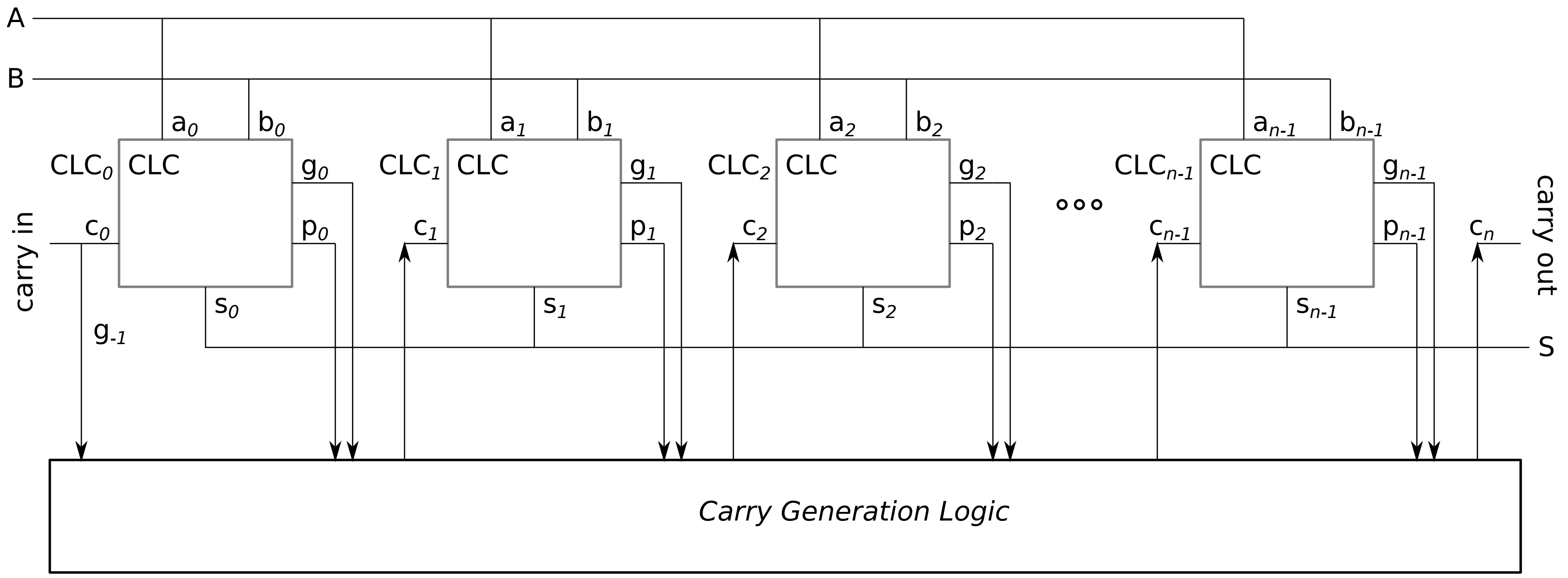
There is no carry out.

Signal $p$ and $g$ **do not** depend on $c$.



## Overview of CLA

### Structure of $n$-bit Carry Lookahead Adder



Types of carry generation logic (CGL): *lookahead* and *ripple*.

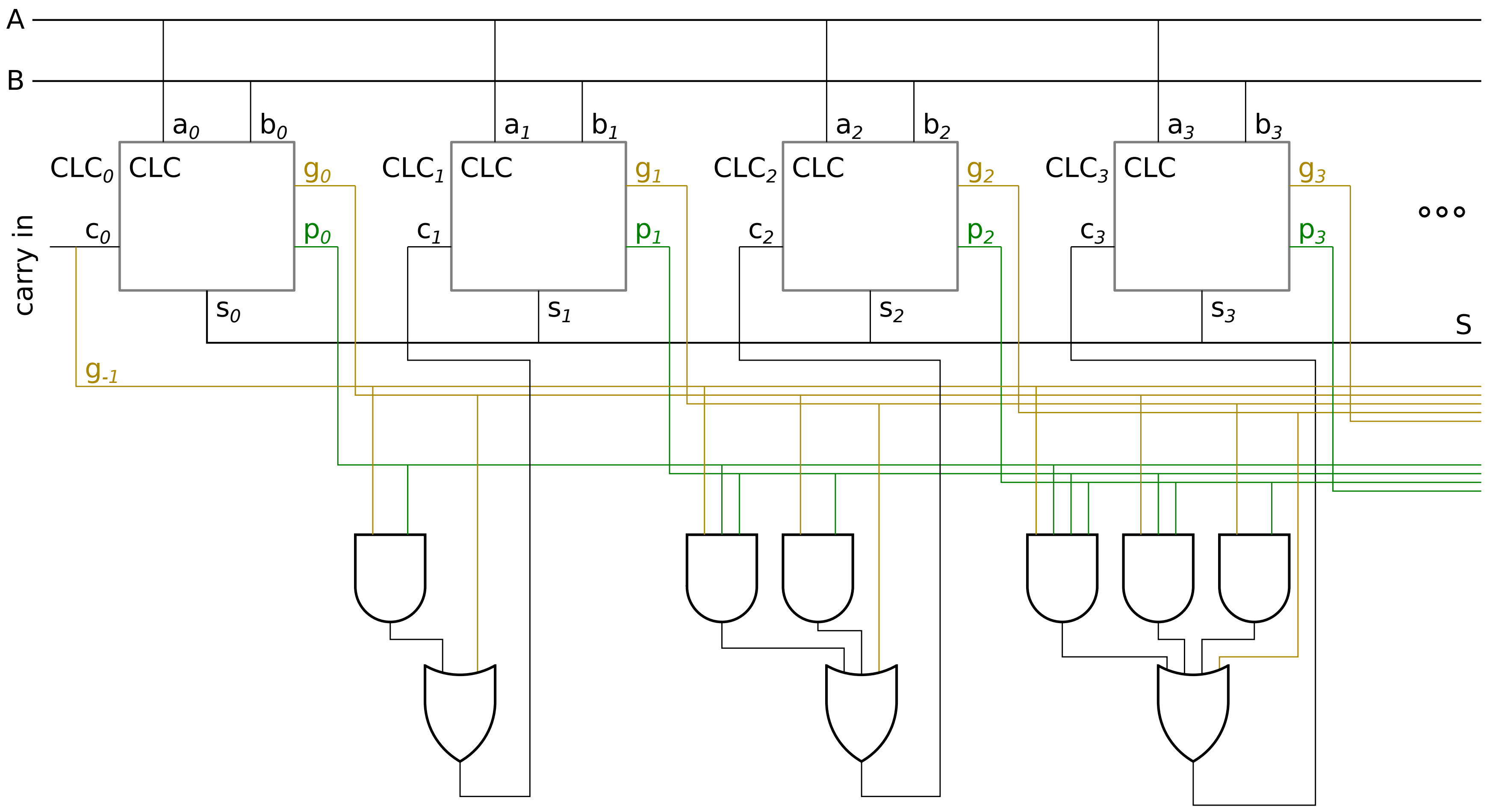
With lookahead CGL adder above is a CLA.

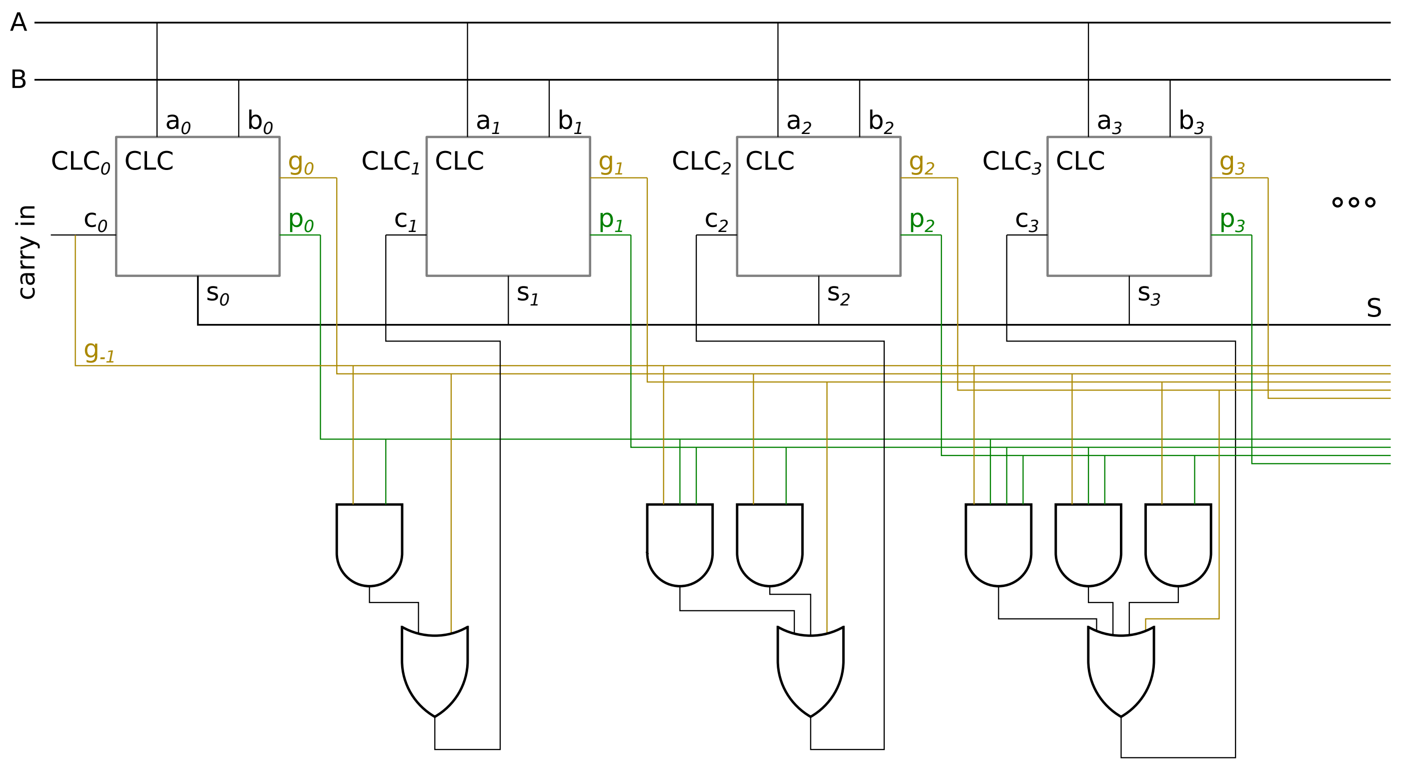
With ripple CGL adder above is equivalent to a ripple adder.

Note: $g_{-1}$ is used as a synonym for $c_0$.

## Carry Generation Logic

### Lookahead CGL for first three cells.





Boolean expressions for carry signals $c_0$ to $c_3$:

$$c_0 = g_{-1}.$$

$$c_1 = g_{-1}p_0 + g_0.$$

$$c_2 = g_{-1}p_0p_1 + g_0p_1 + g_1.$$

$$c_3 = g_{-1}p_0p_1p_2 + g_0p_1p_2 + g_1p_2 + g_2.$$

## Generalization of Lookahead Carry Generation

$c_0 = g_{-1}.$

$c_1 = g_{-1}p_0 + g_0.$

$c_2 = g_{-1}p_0p_1 + g_0p_1 + g_1.$

$c_3 = g_{-1}p_0p_1p_2 + g_0p_1p_2 + g_1p_2 + g_2.$

Generalizing we get

$$\begin{aligned} c_i &= g_{-1}p_0p_1\cdots p_{i-1} + g_0p_1p_2\cdots p_{i-1} + g_1p_2p_3\cdots p_{i-1} + \cdots + g_{i-2}p_{i-1} + g_{i-1} \\ &= \sum_{j=-1}^{i-1} g_j \prod_{k=j+1}^{i-1} p_k \end{aligned}$$

## CLA Delay Analysis

### Delay Models

### About These Models

Intended for hand analysis.

These models for making rough comparisons of different kinds of adders.

For example, ripple adder v. carry lookahead adder.

These models too crude for choosing between adders with delays within 40% of each other...
... in such cases write HDL descriptions ...
... and use synthesis program to determine delays.

## *Unrealistic* Delay Model

Easiest to use, but exaggerates performance of CLA.

*$n$-input XOR Gate:* $t_g = 2$ delay units.

*$n$-input AND, OR, NAND, NOR Gates:* $t_g = 1$ delay unit.

## *Conservative* Delay Model

More tedious, understates performance of CLA.

*$n$-input XOR Gate:* $t_g = 2 \lg n$ delay units.

*$n$-input AND, OR, NAND, NOR Gates:* $t_g = \lg n$ delay units.

## How To Analyze Delay of Combinational Circuit—A Review

### Steps

(0) Mark all inputs to the circuit with time $t = 0$.

(1) If all gate outputs are marked with a time then go to step (2).

Otherwise, consider gates that do not have a time marked at their outputs ...
... and find one in which all inputs are marked with a time.

Mark the output of the gate with time $\max\{t_1, t_2, \ldots, t_n\} + t_g$ ...
... where $t_1, \ldots, t_n$ are the times marked on the gate's inputs ...
... and $t_g$ is the delay of the gate based on the model in use.

Go to step (1).

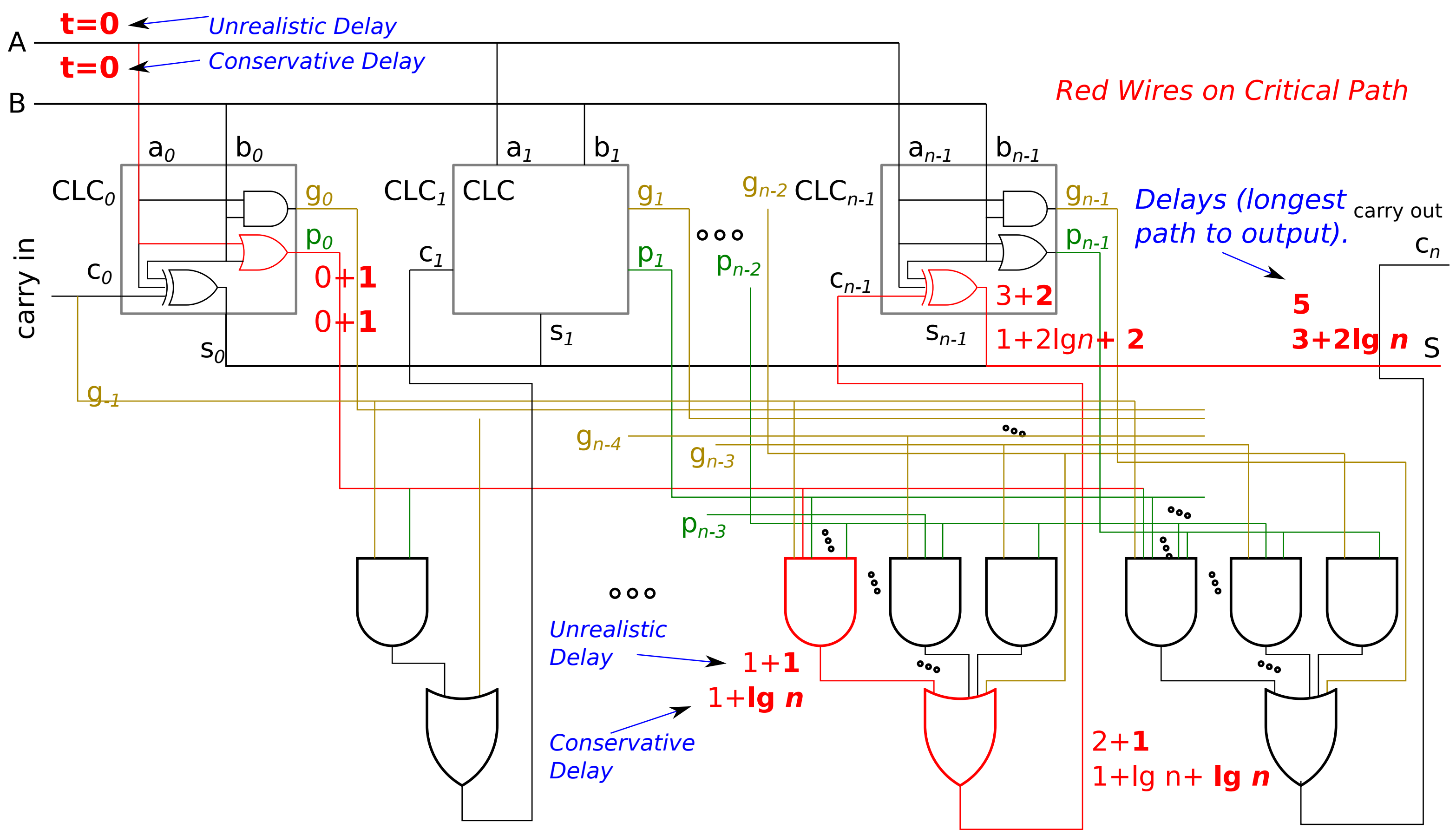
(2) The circuit delay is the largest time assigned to a wire.

## CLA Delay Analysis

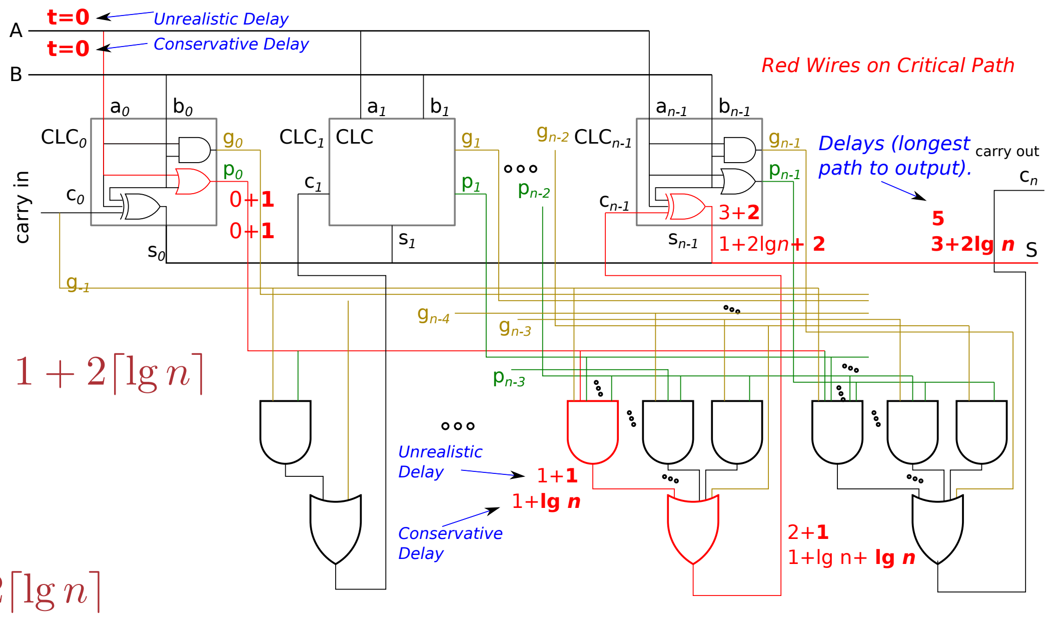
### Critical (longest) Path

Starts at inputs, flows through $p_0$ (or any other $p_i$), $c_{n-1}$, ending at $s_{n-1}$.

*You can follow along by following the red path!*



## Critical (longest) Path:



Generation of $p$ and $g$:

Delay 1, Total 1

Generation of $c$:

*Unrealistic*: Delay 2, Total 3

*Conservative*: Delay $2\lceil \lg n \rceil$, Total $1 + 2\lceil \lg n \rceil$

Computation of $s$:

*Unrealistic*: Delay 2, Total 5

*Conservative*: Delay 2, Total $3 + 2\lceil \lg n \rceil$

## Total Delay:

*Unrealistic Model Delay:* 5 gate delays.

*Conservative Model Delay:* $3 + 2\lceil \lg n \rceil$ gate delays.

## Comparison

$n$-bit ripple: Delay $2n$.

$n$-bit CLA: Delay 5 or $3 + 2\lceil \lg n \rceil$.

8-bit ripple: Delay 16.

8-bit CLA: Delay 5 or 8.

32-bit ripple: Delay 64.

32-bit CLA: Delay 5 or 13.

64-bit ripple: Delay 128.

64-bit CLA: Delay 5 or 15.

CLA looks much better…
… until we compute the cost!

## CLA Cost Analysis

### Cost Model

*Cost of $a$-input AND Gate:* $a - 1$.

*Cost of 3-input XOR Gate:* $5$

## $n$-bit CLA Cost Analysis

Cost of each CLA Cell: $\overbrace{5}^{\text{Sum}} + \overbrace{1}^{\text{Gen.}} + \overbrace{1}^{\text{Prop.}} = 7$

Cost of logic to compute $c_i$ given $p$ and $g$ signals:

$$\overbrace{i}^{\text{OR Gate}} + \overbrace{\sum_{j=2}^{i+1} j - 1}^{\text{AND Gates}} = i + \frac{i(i+1)}{2} = \frac{i(i+3)}{2}$$

Cost of logic to compute $c_1$ to $c_n$:

$$\sum_{i=1}^{n} \frac{i(i+3)}{2} = \frac{1}{6}n(n+1)(n+5)$$

Cost of entire $n$-bit CLA

$$\overbrace{7n}^{\text{CLA Cells}} + \overbrace{\frac{1}{6}n(n+1)(n+5)}^{\text{Carry Gen Logic}} \approx \frac{n^3}{6}$$

## CLA v. Ripple Cost Comparison

### Cost of Selected Sizes

$n$-bit ripple: $10n$

$n$-bit CLA: $7n + \frac{1}{6}n(n+1)(n+5)$

8-bit ripple: 80

8-bit CLA: 212 or 2.65$\times$ cost of ripple adder.

32-bit ripple: 320

32-bit CLA: 6736 or 21$\times$ cost of ripple adder.

64-bit ripple: 640

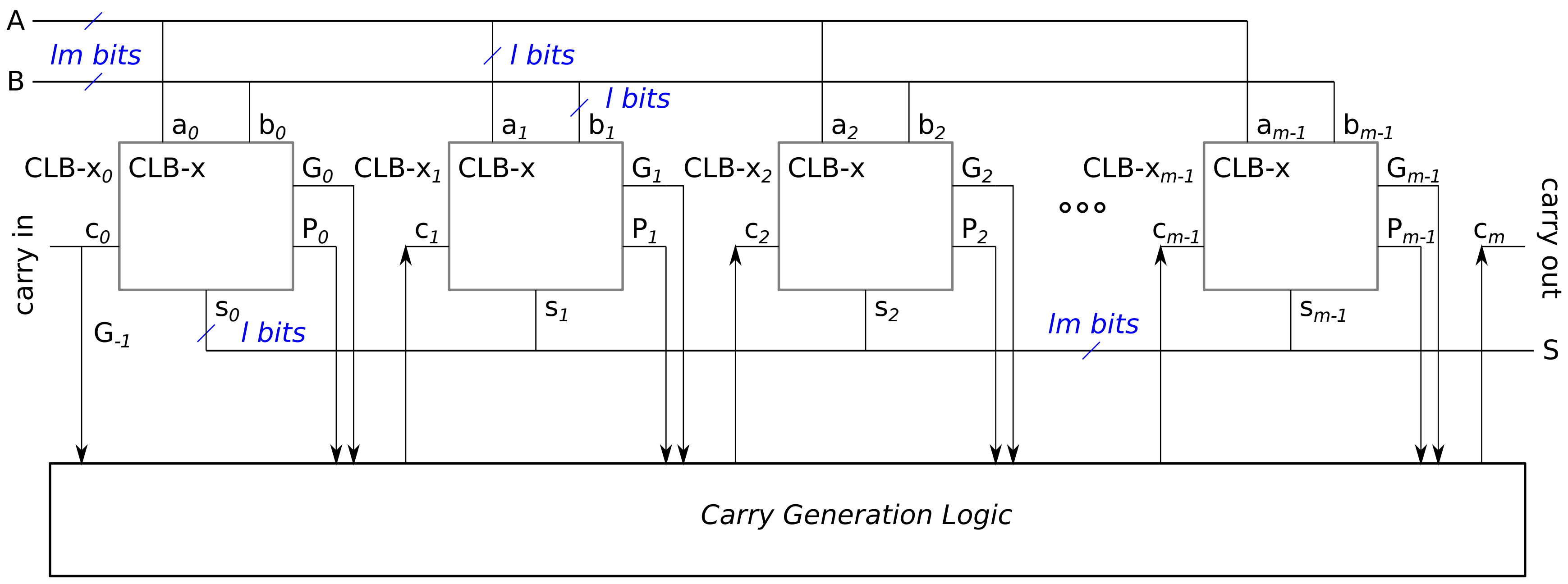
64-bit CLA: 48288 or 75$\times$ cost of ripple adder.

## Hierarchical CLA

Instead of using $n$ 1-bit carry lookahead cells . . .

. . . use $m$ $l$-bit *Carry Lookahead Blocks*. (Note that $n = l \times m$.)

Structure:



The CLB-x blocks can contain any kind of $l$-bit adder, including ripple and CLA.

Carry generation logic (CGL) can be ripple or lookahead.

## Benefit of Hierarchical CLA

Since CGL and the logic to generate $P$ and $G$ operate on fewer bits …

… cost is lower.

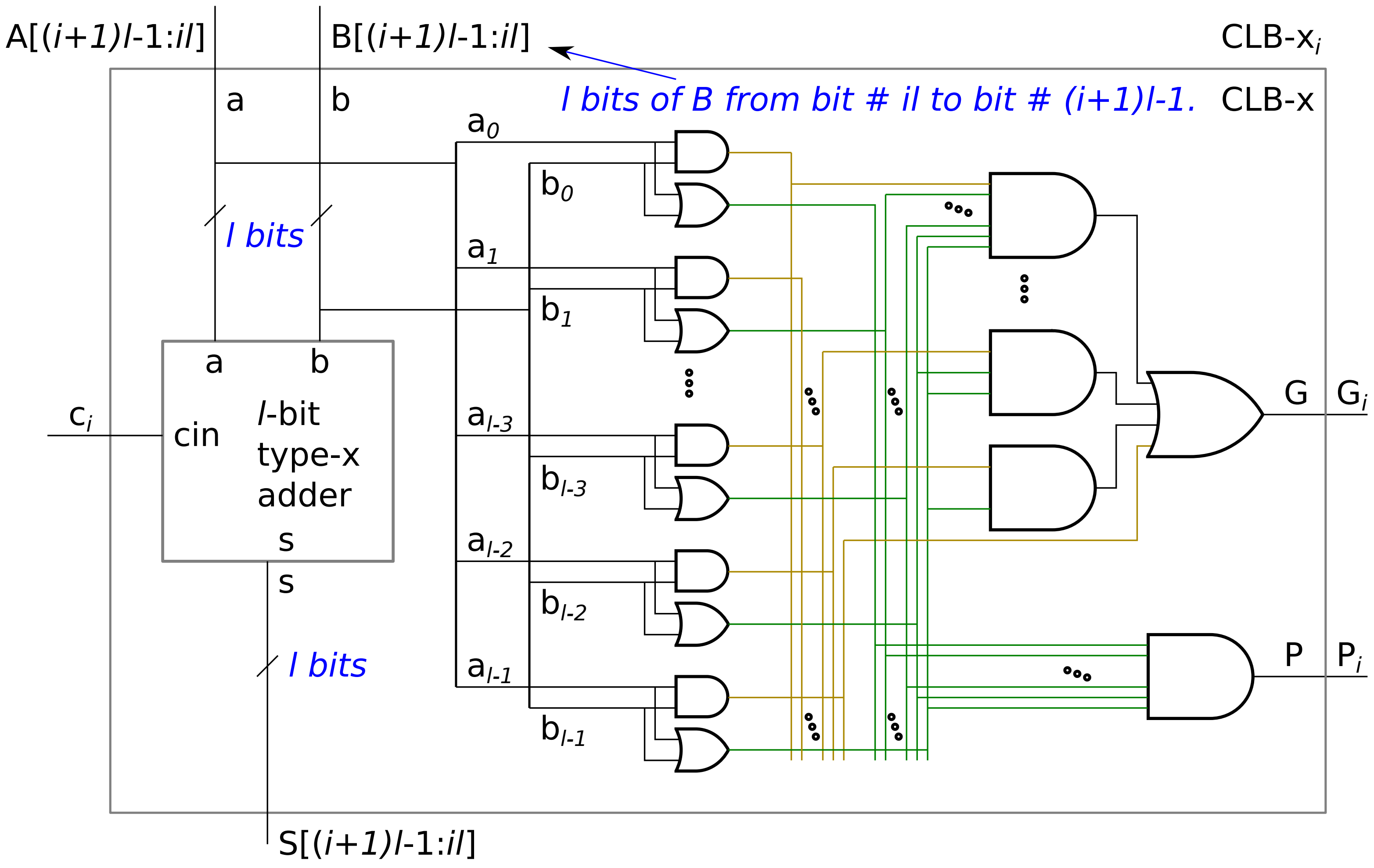
We will consider two kinds of CLB-x modules:

*CLB-c*, in which the $l$-bit adder is a CLA.

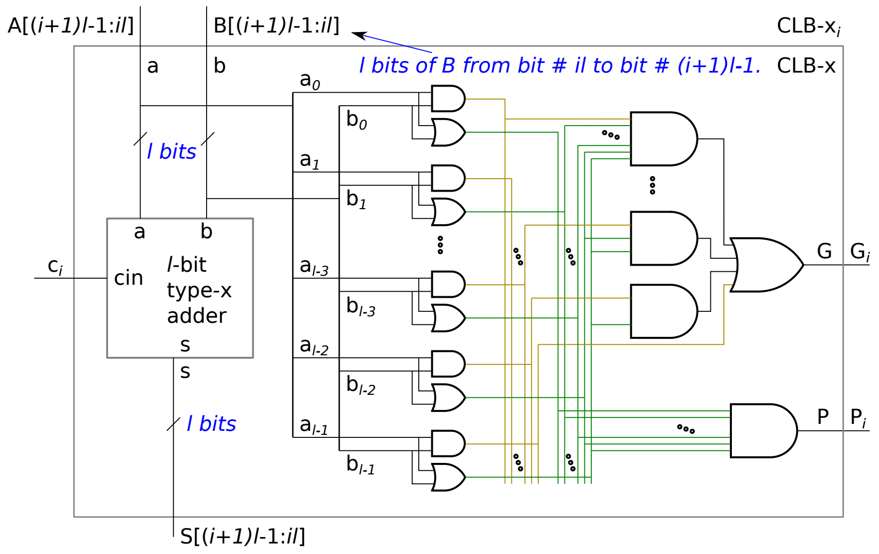
*CLB-r*, in which the $l$-bit adder is a ripple adder.

## Carry Lookhead Block (CLB-x)

CLB-x contains an $l$-bit adder of type $x$ (*e.g.*, ripple or CLA)...
... and logic to compute $P$ and $G$.



## CLB Propagate and Generate Outputs



Let $g_i = a_i b_i$ and $p_i = a_i + b_i$.

$$P = \prod_{j=0}^{l-1} p_j$$

$$= p_0 p_1 \cdots p_{l-1}$$

$$G = \sum_{j=0}^{l-1} g_j \prod_{k=j+1}^{l-1} p_k$$

$$= g_0 p_1 p_2 \cdots p_{l-1} + g_1 p_2 p_3 \cdots p_{l-1} + \cdots + g_{l-2} p_{l-1} + g_{l-1}$$

Notice that logic for $G$ is almost the same as the logic for $c_i$ …

… an important difference is that the carry in is ignored.

## Design Options for Hierarchical CLA

Value of $l$. (Since $n$ is given, if we choose $l$ that fixes $m = n/l$.)

Type of CLB: *CLB-c* (block uses $l$-bit CLA) or *CLB-r* (block uses $l$-bit ripple adder).

Type of CGL: *lookahead* or *ripple*.

## Design Options Example

*Given:* We want a 32-bit adder, money is no object.

Choose $l = 8$, this fixes $m = 4$.

For CLB use a CLA.

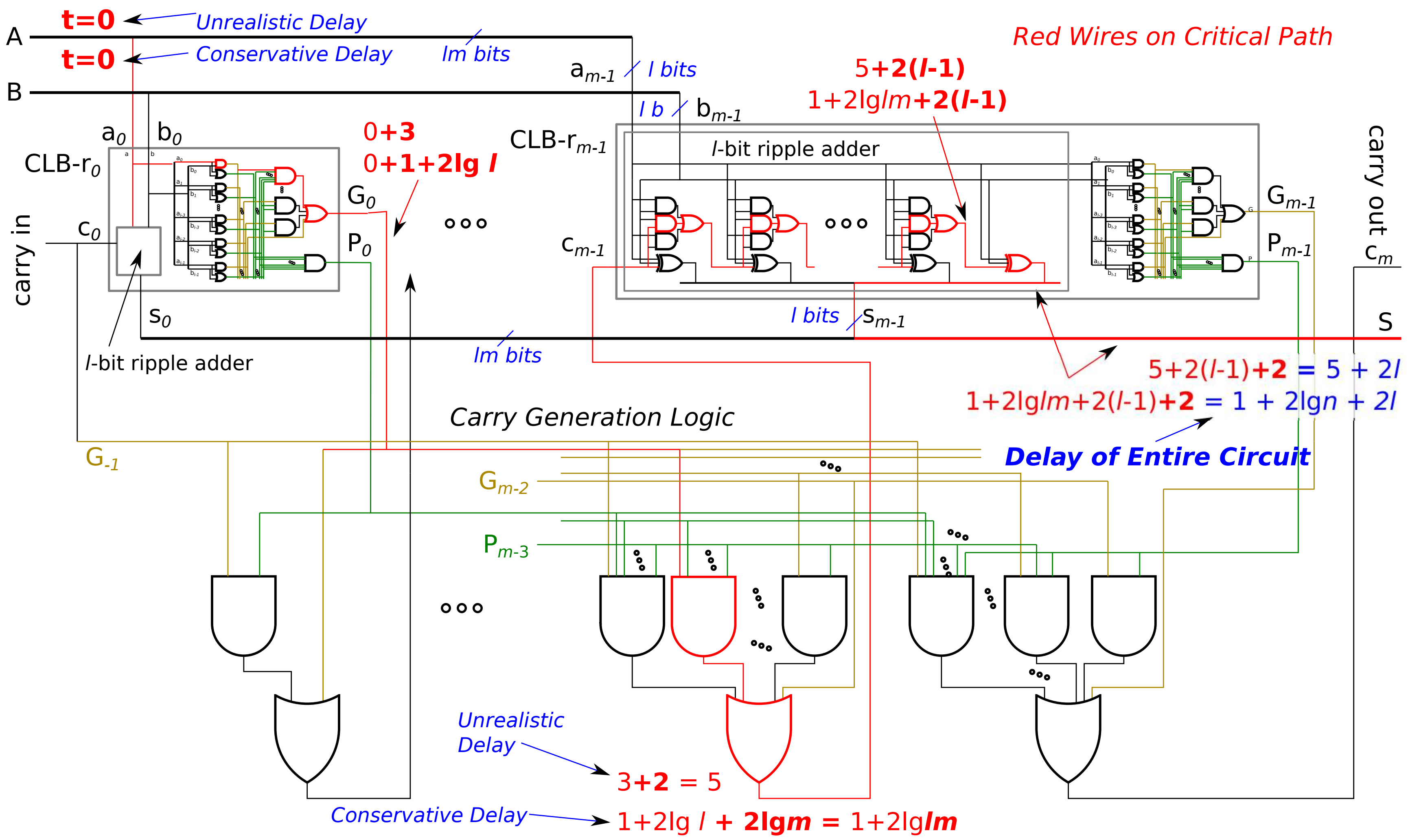
For CGL use lookahead logic.

Are these good choices? To find answer need to analyze cost and performance.

## Hierarchical CLA Delay Analysis

### Delay Analysis for Hierarchical CLA with CLB-r and lookahead CGL

Critical Path and delay under unrealistic and conservative models:



## Hierarchical CLA Delay Analysis

### Delays for $m \times l$-bit Hierarchical CLA. $(n = lm)$

Hierarchical CLA with CLB-r and lookahead CGL

*Unrealistic*: $7 + 2(l - 1)$ gate delays

*Conservative*: $1 + 2\lg n + 2l$ gate delays

Hierarchical CLA with CLB-c and lookahead CGL

*Unrealistic*: $9$ gate delays

*Conservative*: $3 + 2\lg l + 2\lg n$ gate delays

Hierarchical CLA with CLB-r and ripple CGL

*Unrealistic*: $1 + 2m + 2l$ gate delays

*Conservative*: $2\lg l + 2(m + l)$ gate delays

For details see Fall 2013 Homework 2 Solution.

## Hierarchical CLA Cost Analysis

Omitted for now.