Fast Adders

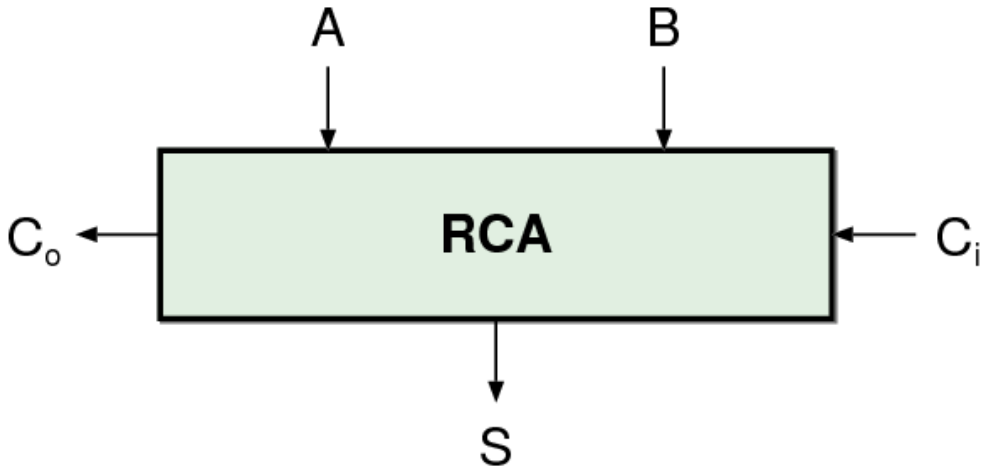
Design of Digital Circuits 2014
Srdjan Capkun
Frank K. Gürkaynak

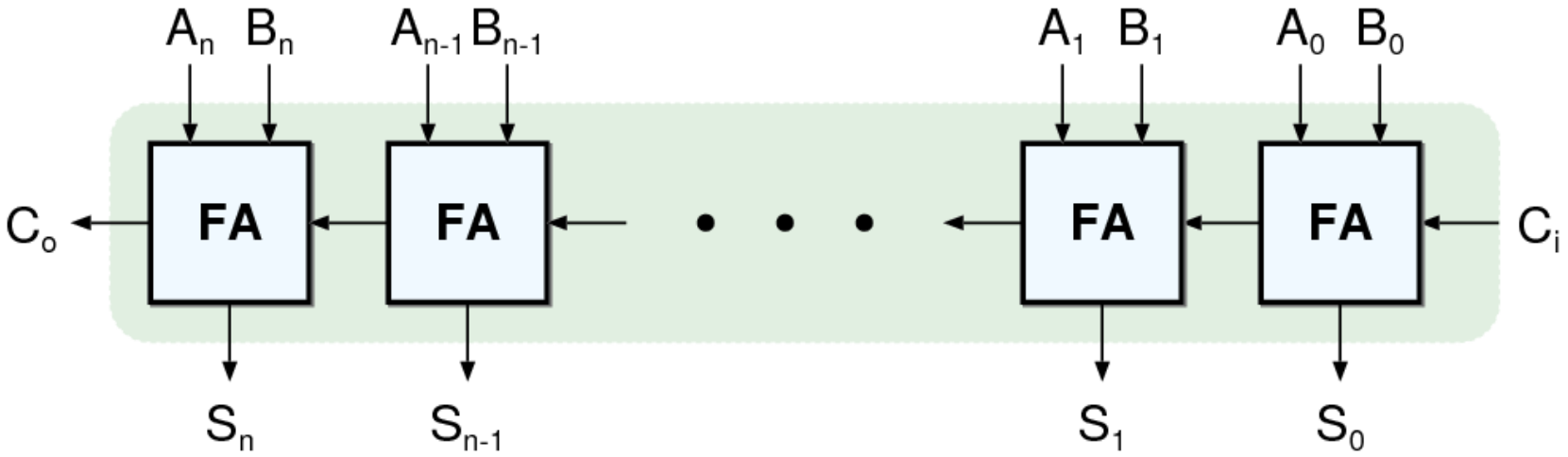
http://www.syssec.ethz.ch/education/Digitaltechnik_14

In This Lecture

- **What slows down adders**
  - The “curse of the carry”
- **Fast adder principles**
- **Parallel Prefix Adders**
  - Graph representation for parallel prefix adders
  - Some example fast adders

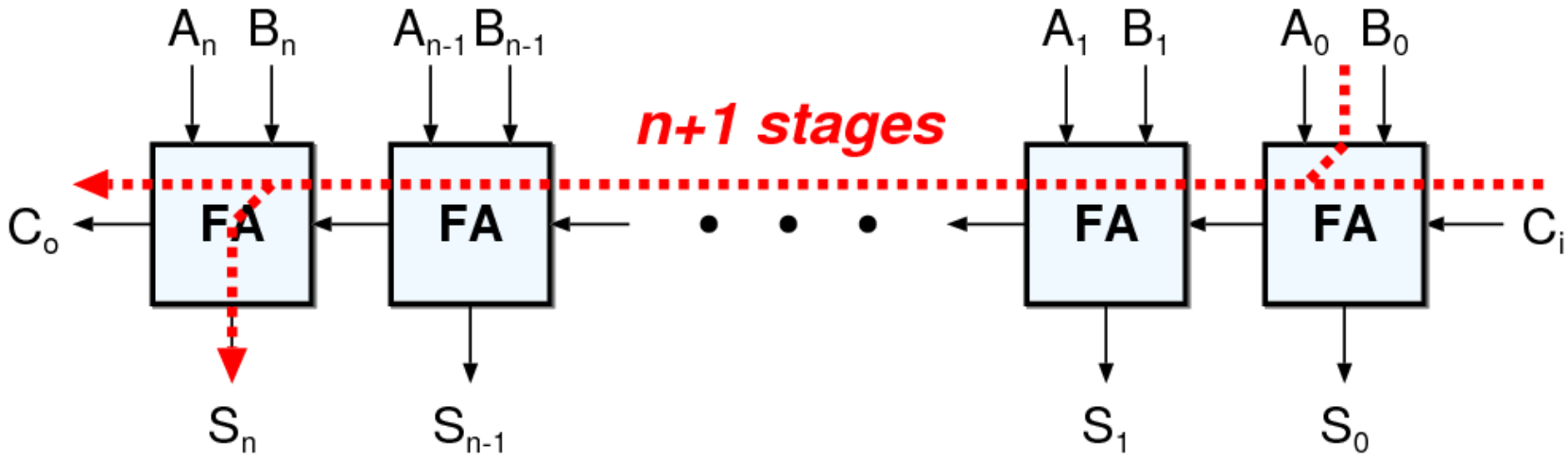
Ripple Carry Adder (RCA)





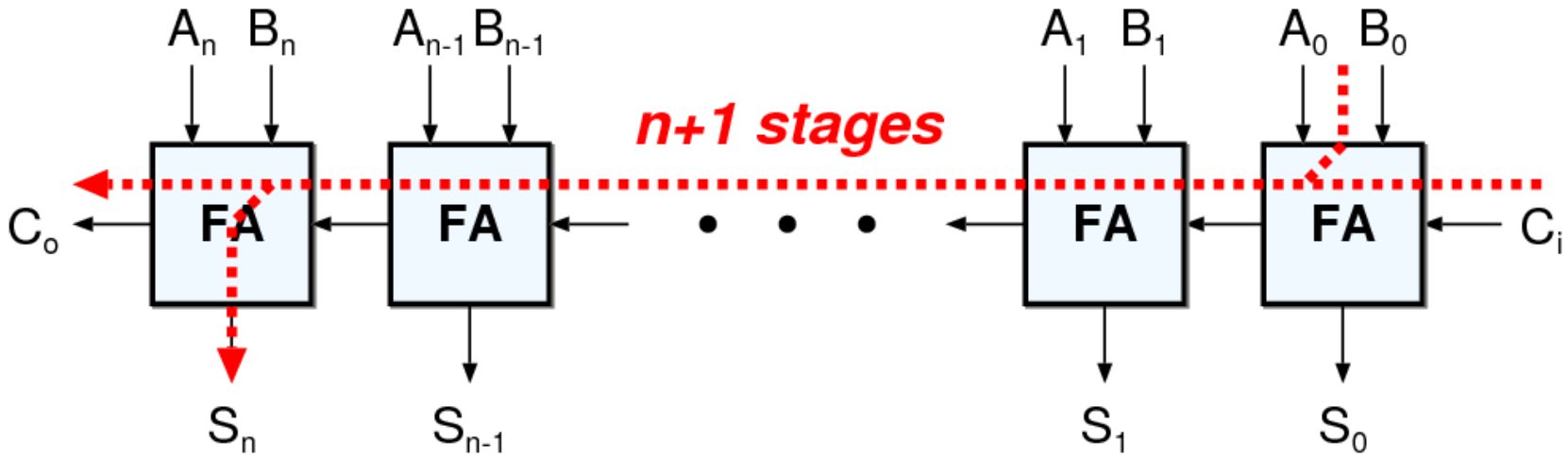
Curse of the Carry

The most significant outputs of the adder depends on the least significant inputs



Curse of the Carry

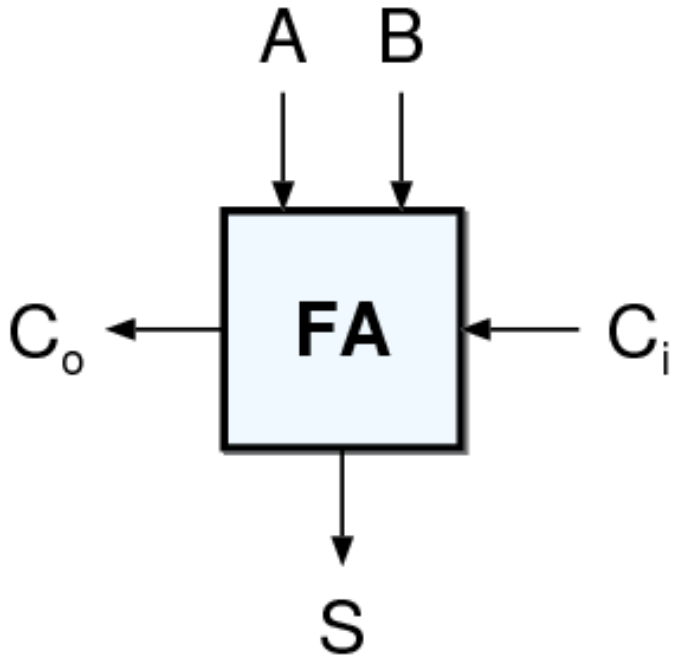
The most significant outputs of the adder depends on the least significant inputs



- **The carry has to propagate from LSB to MSB**
- **Fortunately these cases are rare**

Full-Adder Revisited, Carry Propagation

Consider the path C_i to C_o

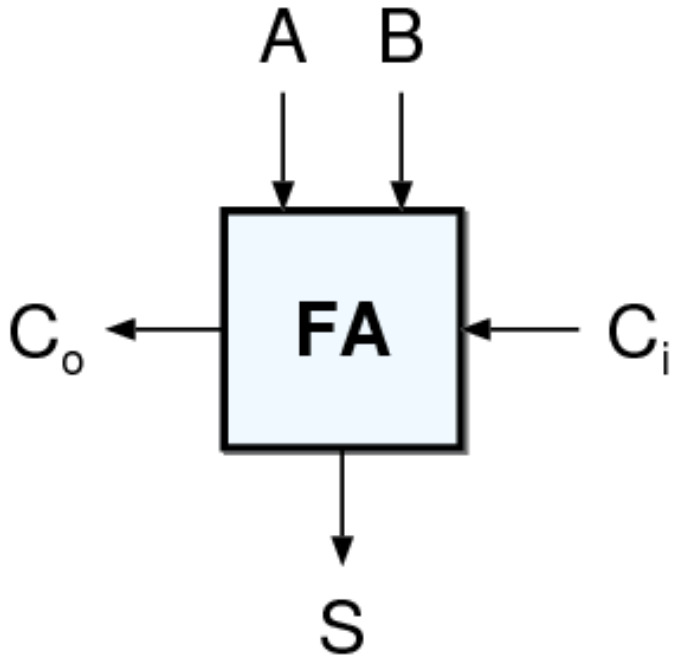


Full-Adder Revisited, Carry Propagation

Consider the path C_i to C_o

- **If A=1 and B=1**
  A carry is generated no matter what the value of carry input (C_i) is

  Generate = G = A & B



Full-Adder Revisited, Carry Propagation

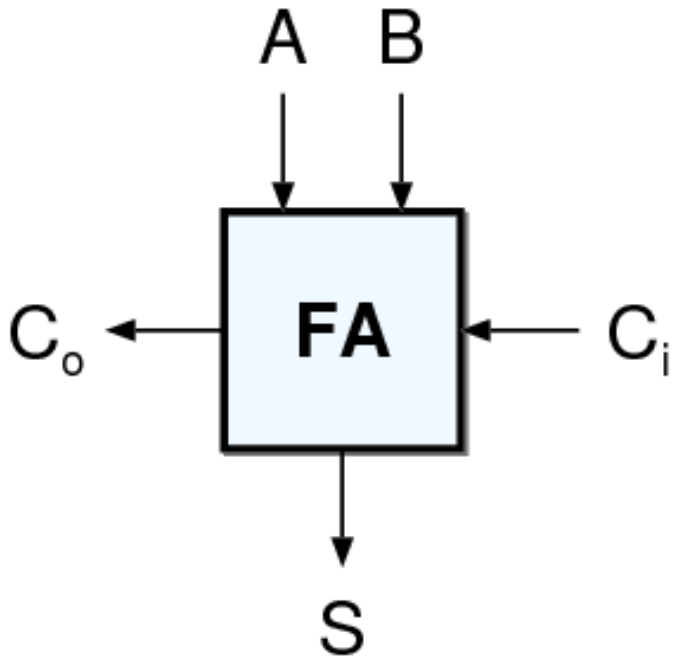
Consider the path C_i to C_o

- **If A=1 and B=1**
  A carry is generated no matter what the value of carry input (C_i) is

  Generate = G = A & B

- **If only A=1 or only B=1**
  The value of carry input is propagated to the carry output

  Propagate = P = A ^ B

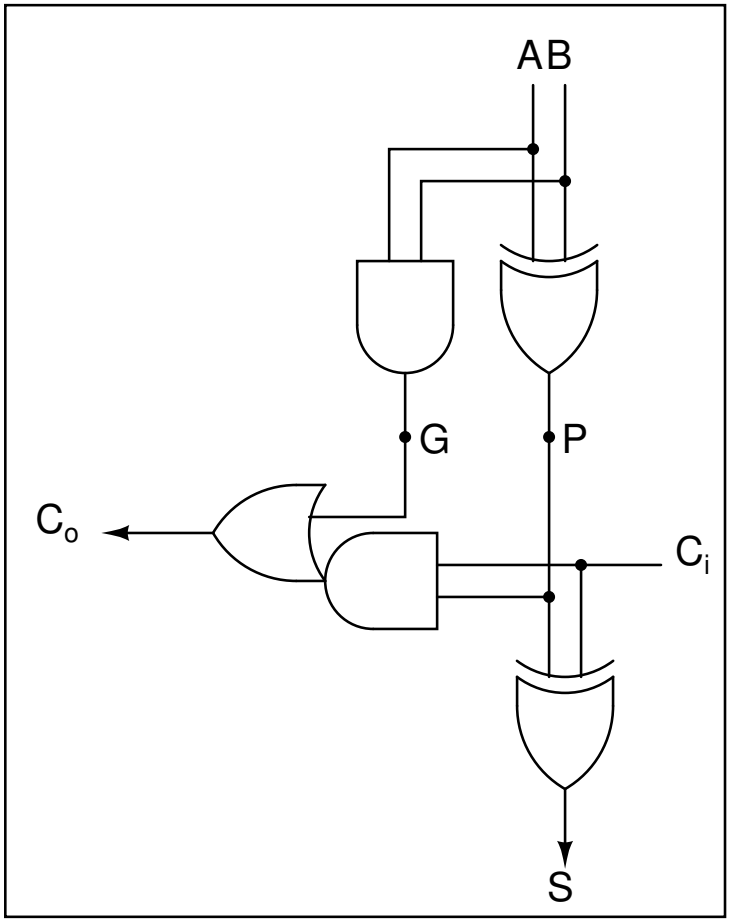


Full-Adder with Propagate and Generate

The Full Adder functionality can be expressed by propagate (P) and generate (G) signals:

$$G = A \,\&\, B$$
$$P = A \,\hat{}\, B$$

$$S = P \,\hat{}\, Ci$$
$$Co = G \mid (Ci \,\&\, P)$$

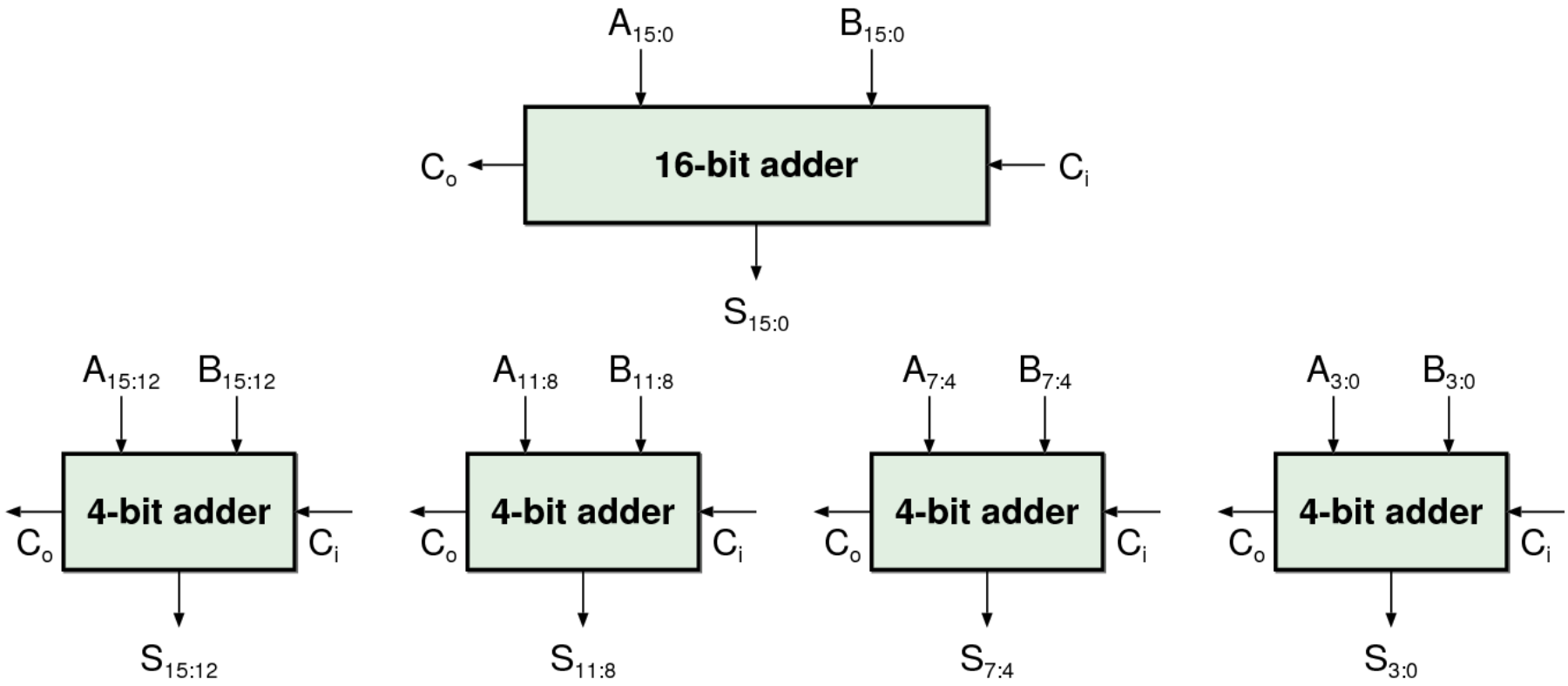


Fast Adder Architectures

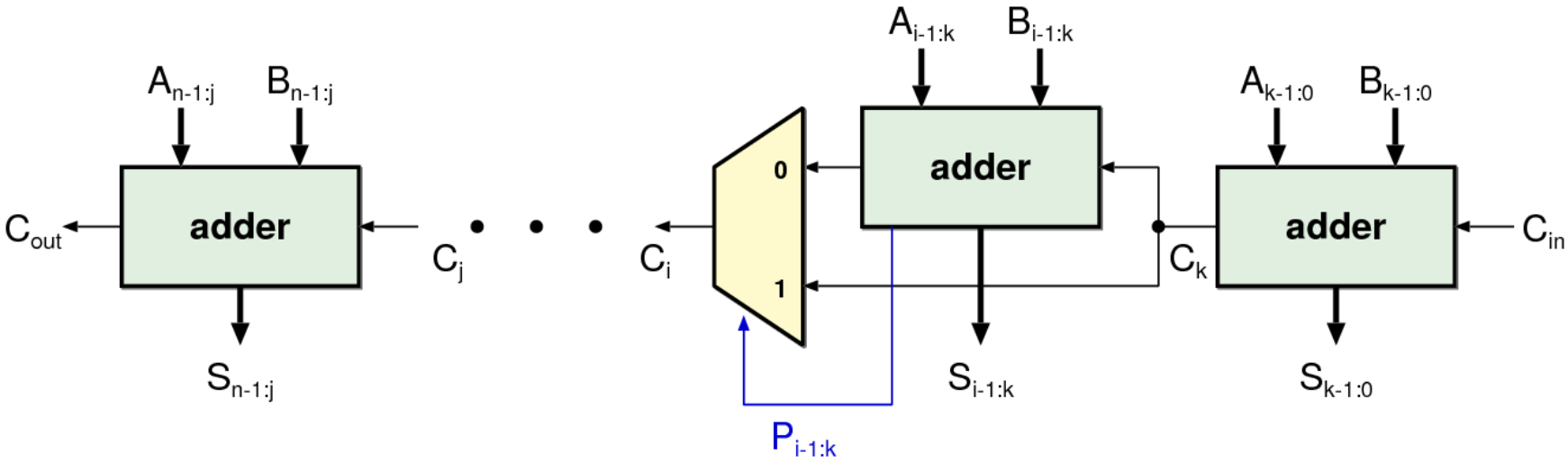
The speed of the adder depends on how fast the carry can be propagated /pre calculated

- **Divide a large adder into a chain of small adders**
  - The critical path is extremely rare (**Carry Skip**)
  - Worst case we need to add one to result (**Carry Increment**)
  - There are only two possible values of carry (**Carry Select**)
  - Calculate the carry in advance (**Carry Lookahead**)

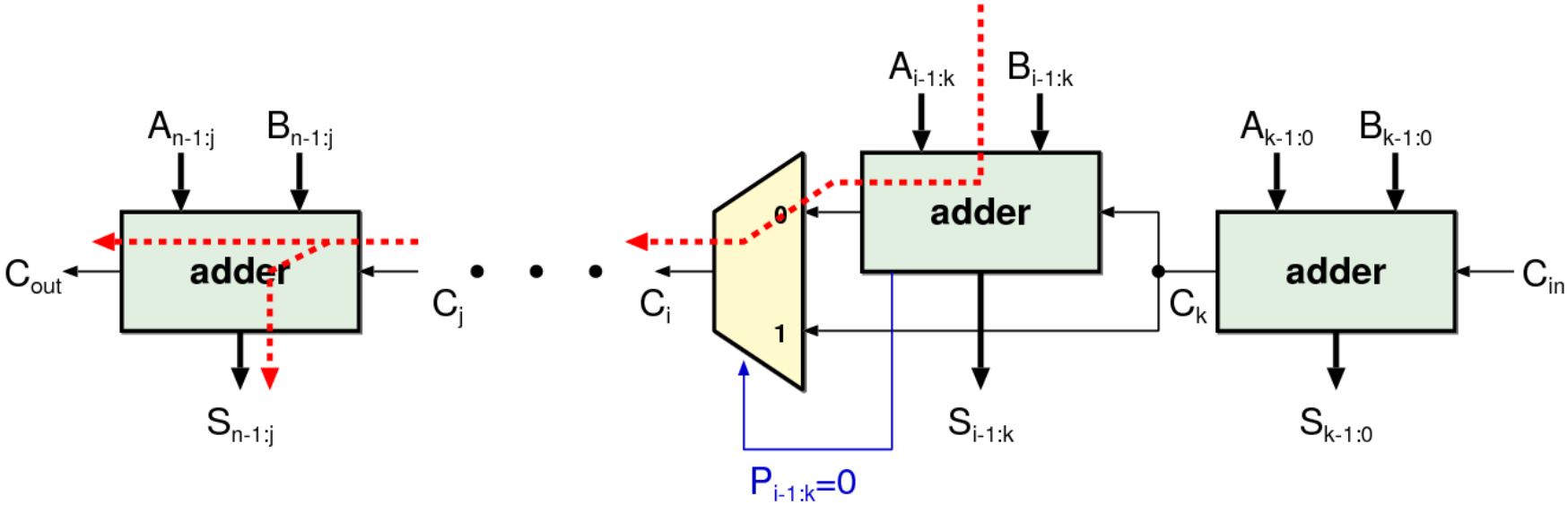
Divide and Conquer



Carry Skip Adder

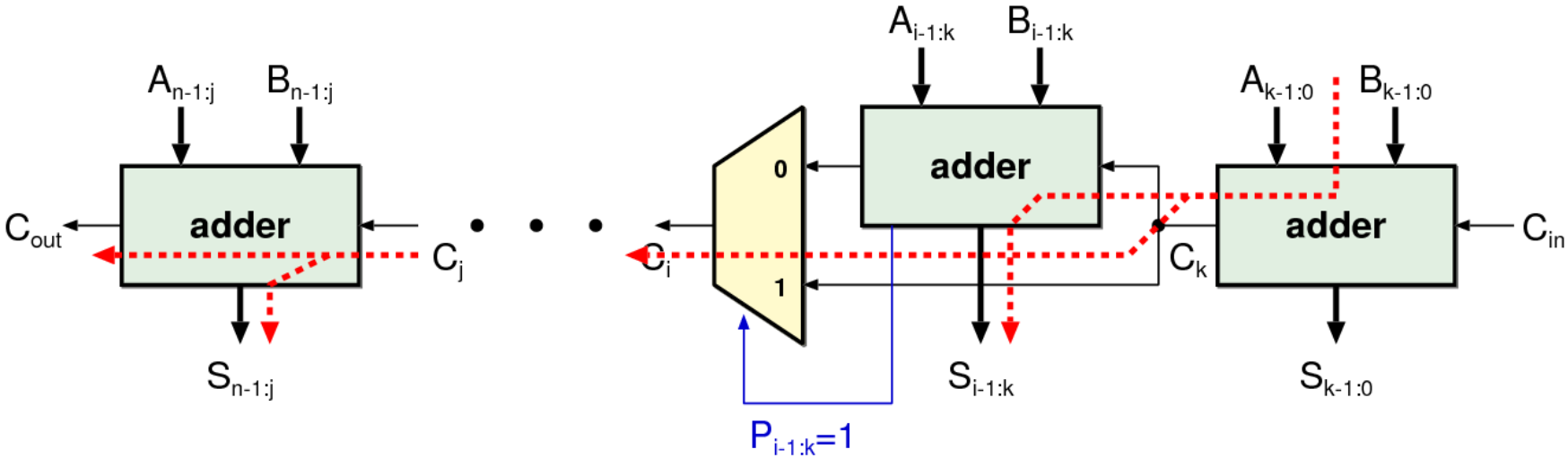


Carry Skip Adder



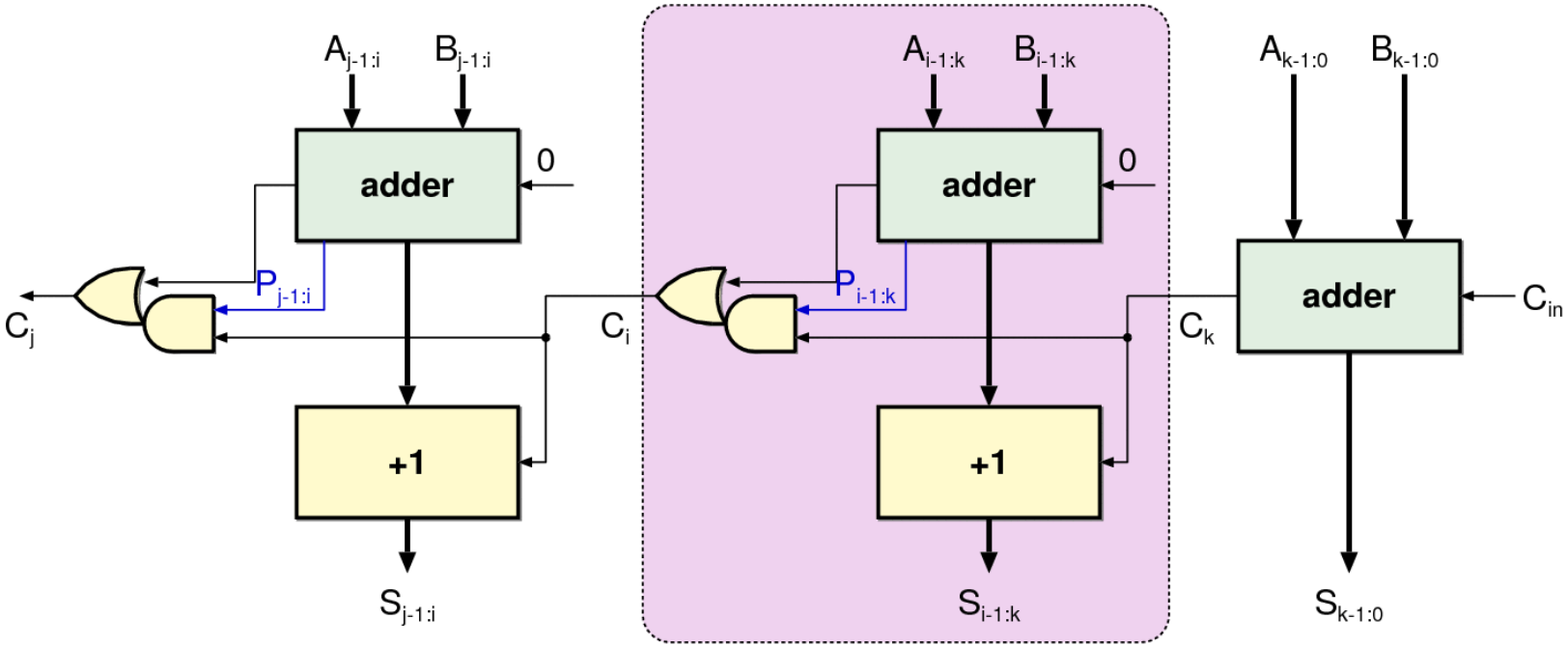
- **$P_{i-1:k}$ = 0 (not all propagate signals from bit i-1 to bit k are 1), the result of the carry is generated within this block.**

Carry Skip Adder



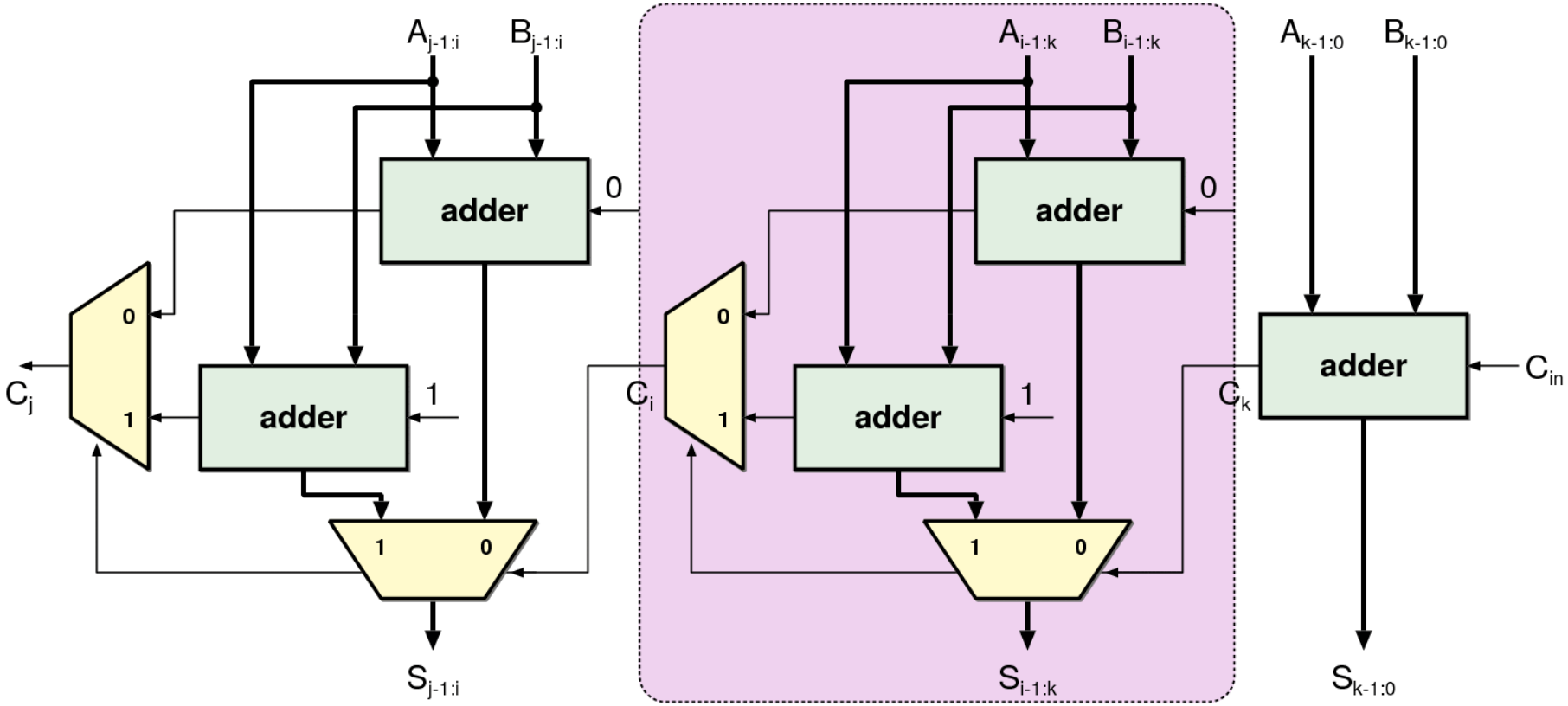
- $P_{i-1:k}$ = 0 (not all propagate signals from bit i-1 to bit k are 1), the result of the carry is generated within this block.
- $P_{i-1:k}$ = 1, the carry of the previous block is propagated

Carry Increment Adder



- C_{in} is assumed to be 0
- When carry is known, in the worst case the output is incremented

Carry Select Adder



- Two parallel adders, one assumes C_{in} 0, the other 1
- Once the previous stage decides on the Carry, the correct calculation is selected.

Carry Lookahead Adder

- For each S_i you need to know what is $C_{in:i-1}$. This calculation can be made faster (than traditional ripple carry structure)

$$C_0 = C_i$$

$$C_1 = G_0 + P_0C_i$$

$$C_2 = G_1 + P_1G_0 + P_1P_0C_i$$

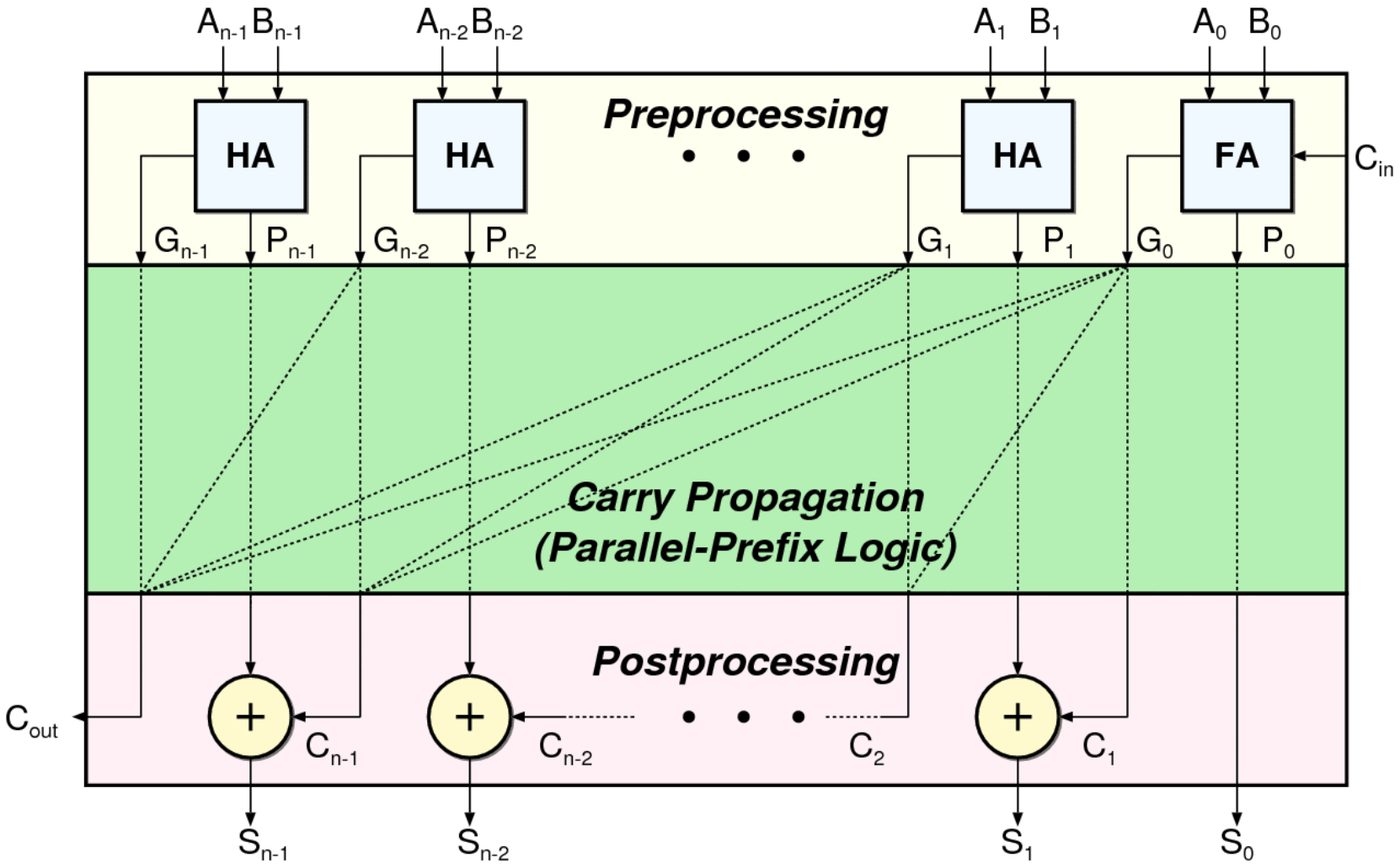
$$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_i$$

Parallel Prefix Adders

Parallel Prefix Adders (PPA) represent a systematic approach to designing optimized adders

- **Parallel Prefix Adders have three stages**
  - Pre-Processing
  - Carry Propagation
  - Post-Processing
- **Adders differ only in the way the carry propagation is calculated. Pre and post processing is identical**

Parallel Prefix Adders



Combined (P)ropagate (G)enerate Signals

$$(G_{i:j}^{k}, P_{i:j}^{k})$$

- is a combined Propagate Generate signal covering the bit range 1 to j at the k^{th} stage of carry propagation.
- The initial Propagate Generate signals are written as:

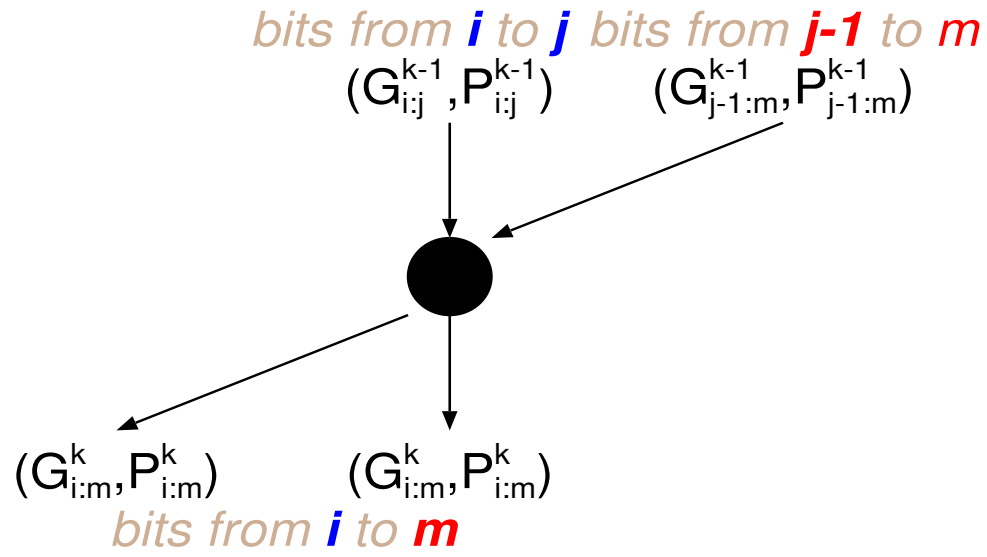
$$(G_{i:i}^{0}, P_{i:i}^{0})$$

- The goal is to calculate

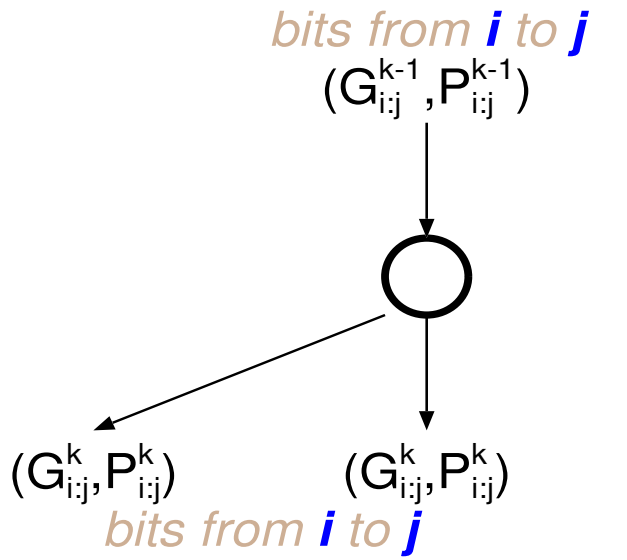
$$(G_{n:0}^{k}, P_{n:0}^{k})$$

for all n in any number of k stages.

Parallel Prefix Operations



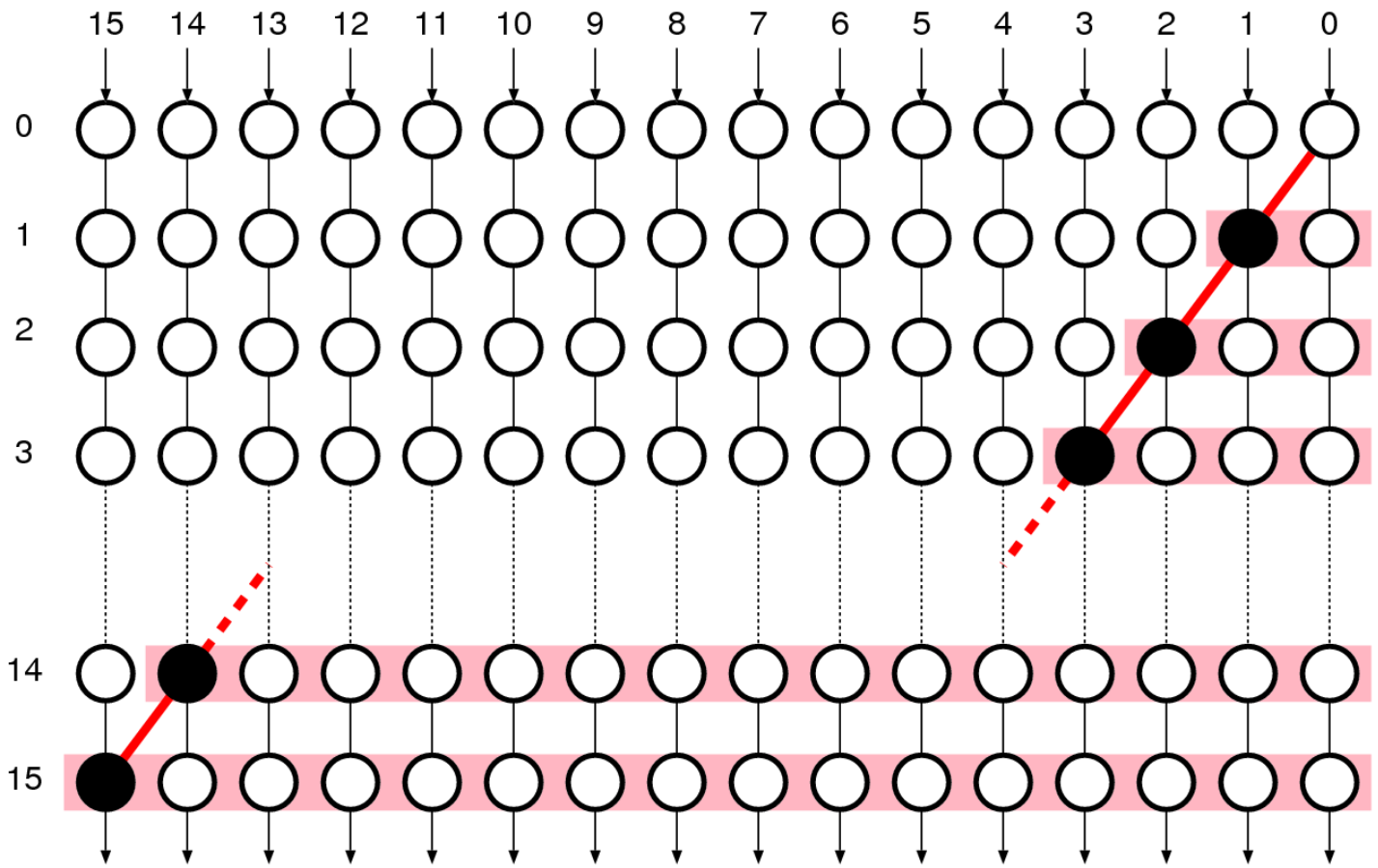
Merge



Feedthrough

- **Merge:** Merges two adjacent (P,G) ranges
- **Feedthrough:** Just copies the signals to next stage

PPA: Ripple Carry Adder



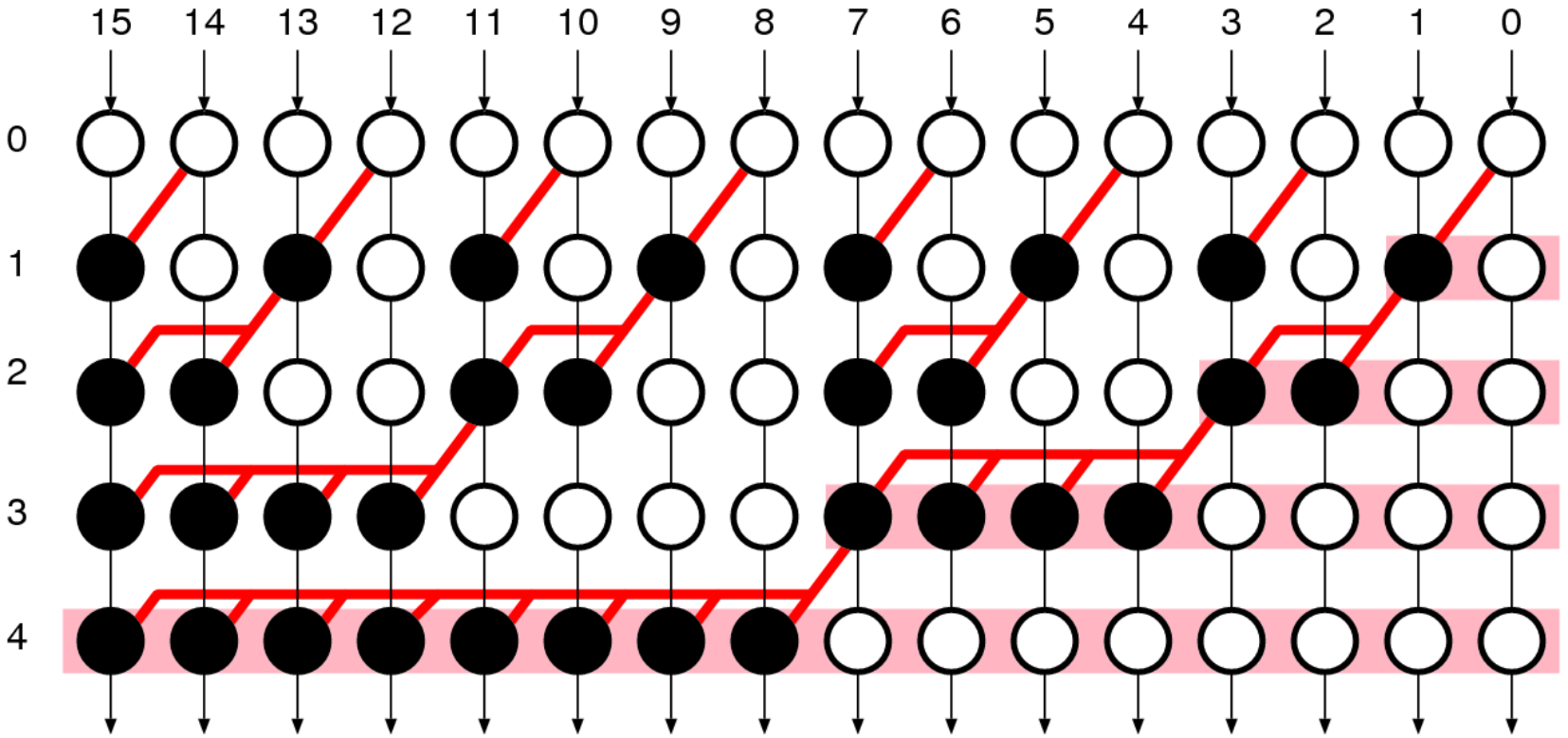
Performance Parameters of PPA

- **Number of black dots: Determines the circuit area, since only the merge operators contains real logic**

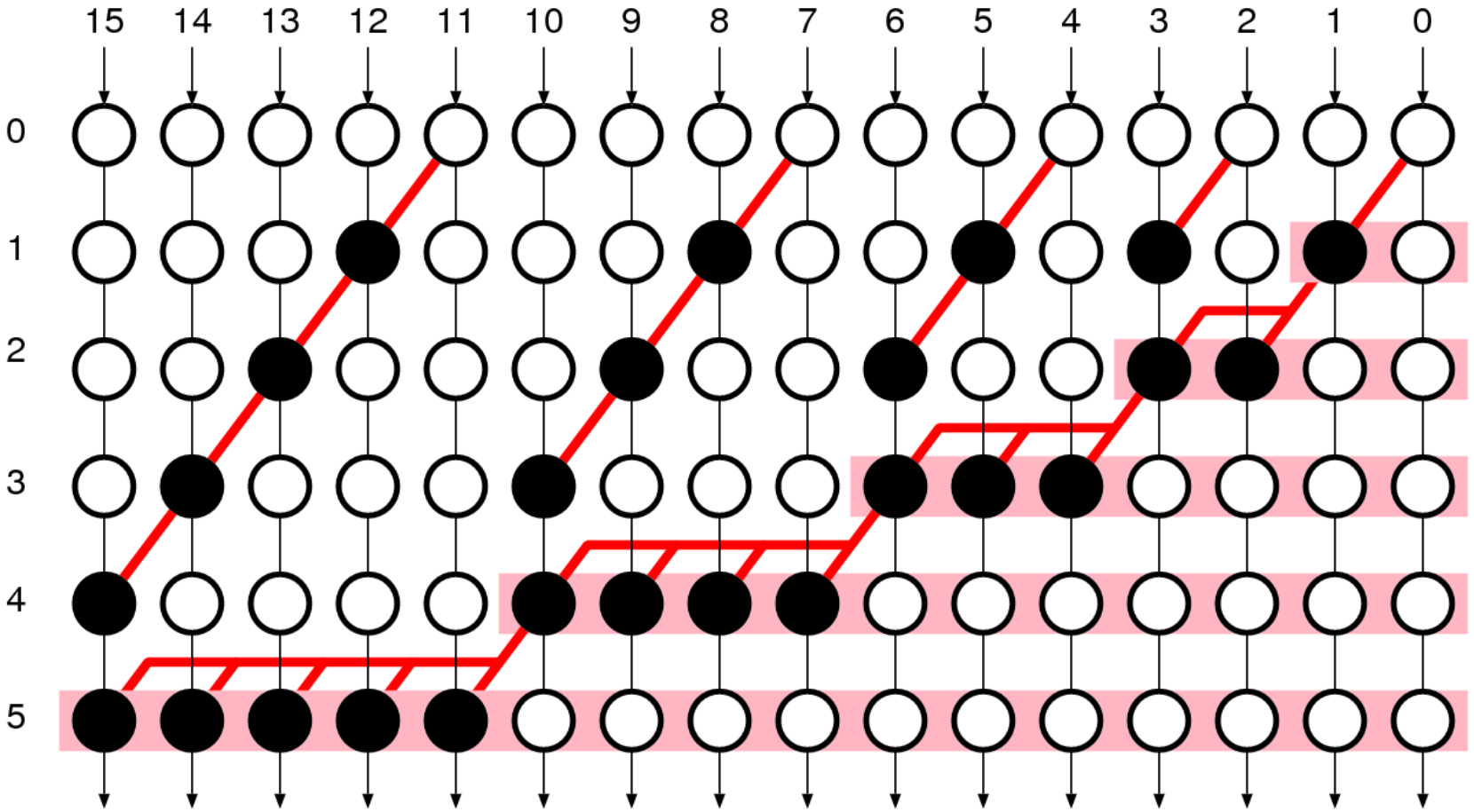
- **Number of Stages: Determines the critical path, the more stages, the longer the critical path**

- **Maximum Fan Out: How many inputs are driven by a single cell, determines stage delay, the more connections, the higher the delay**

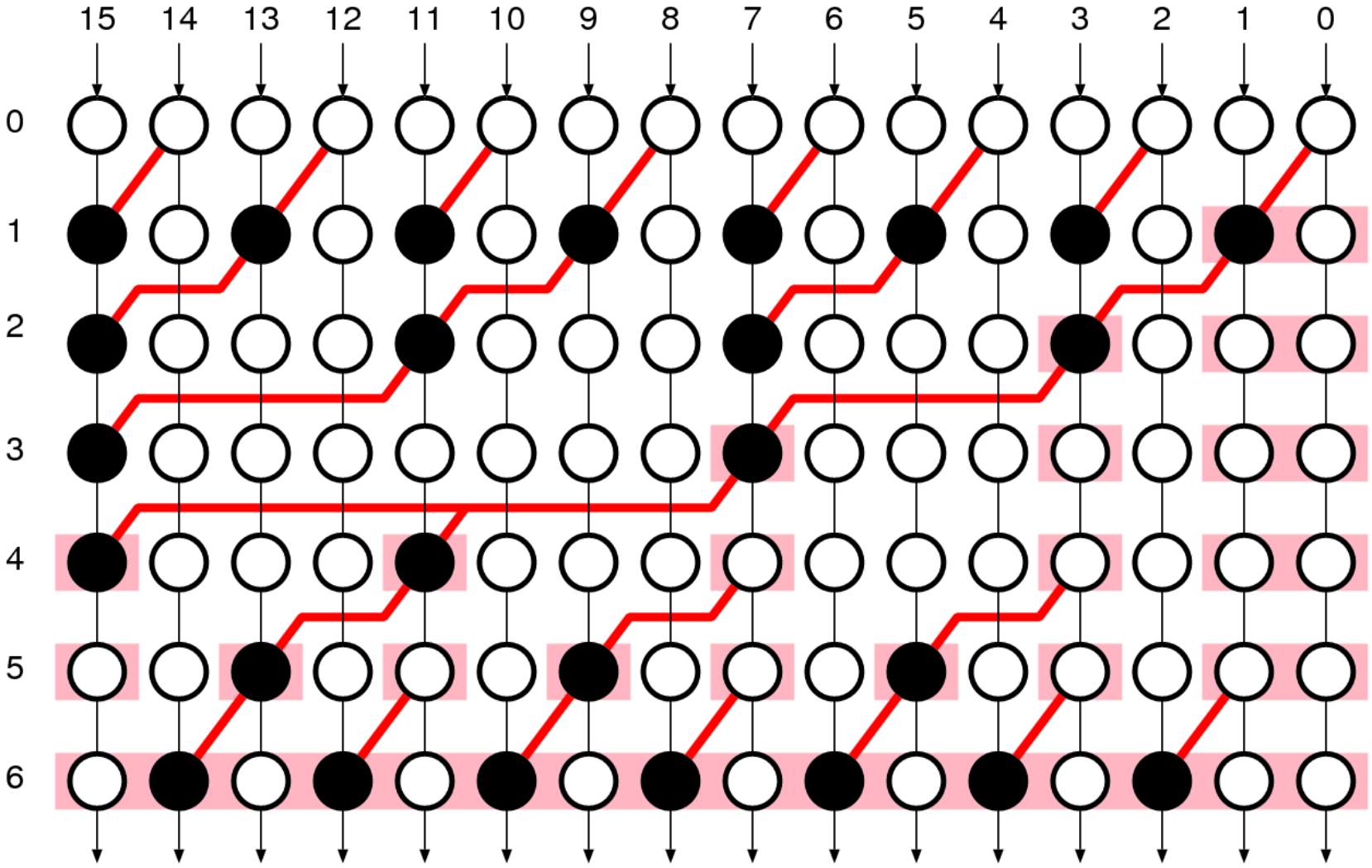
PPA: Sklansky



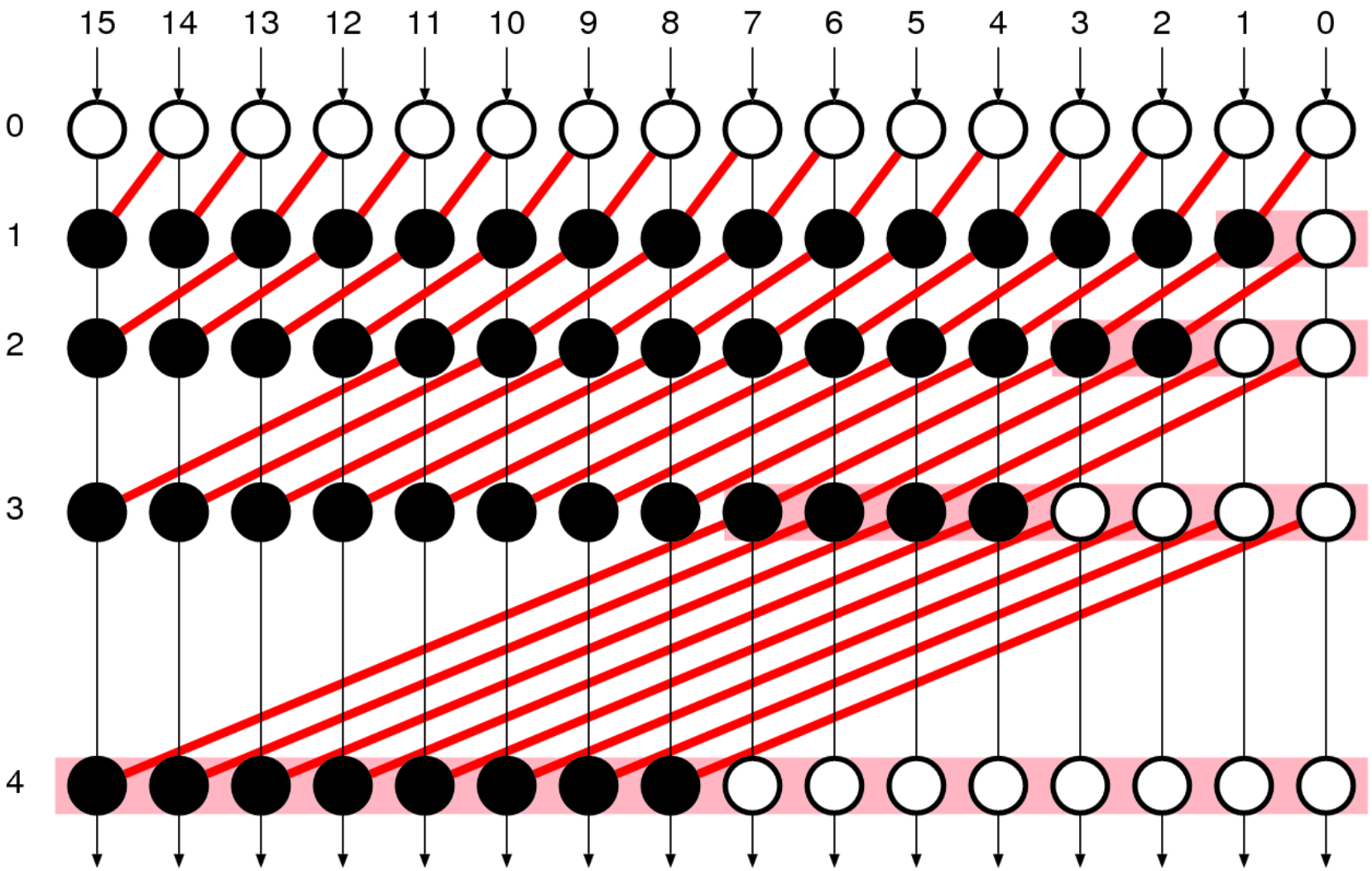
PPA: Carry Increment Adder



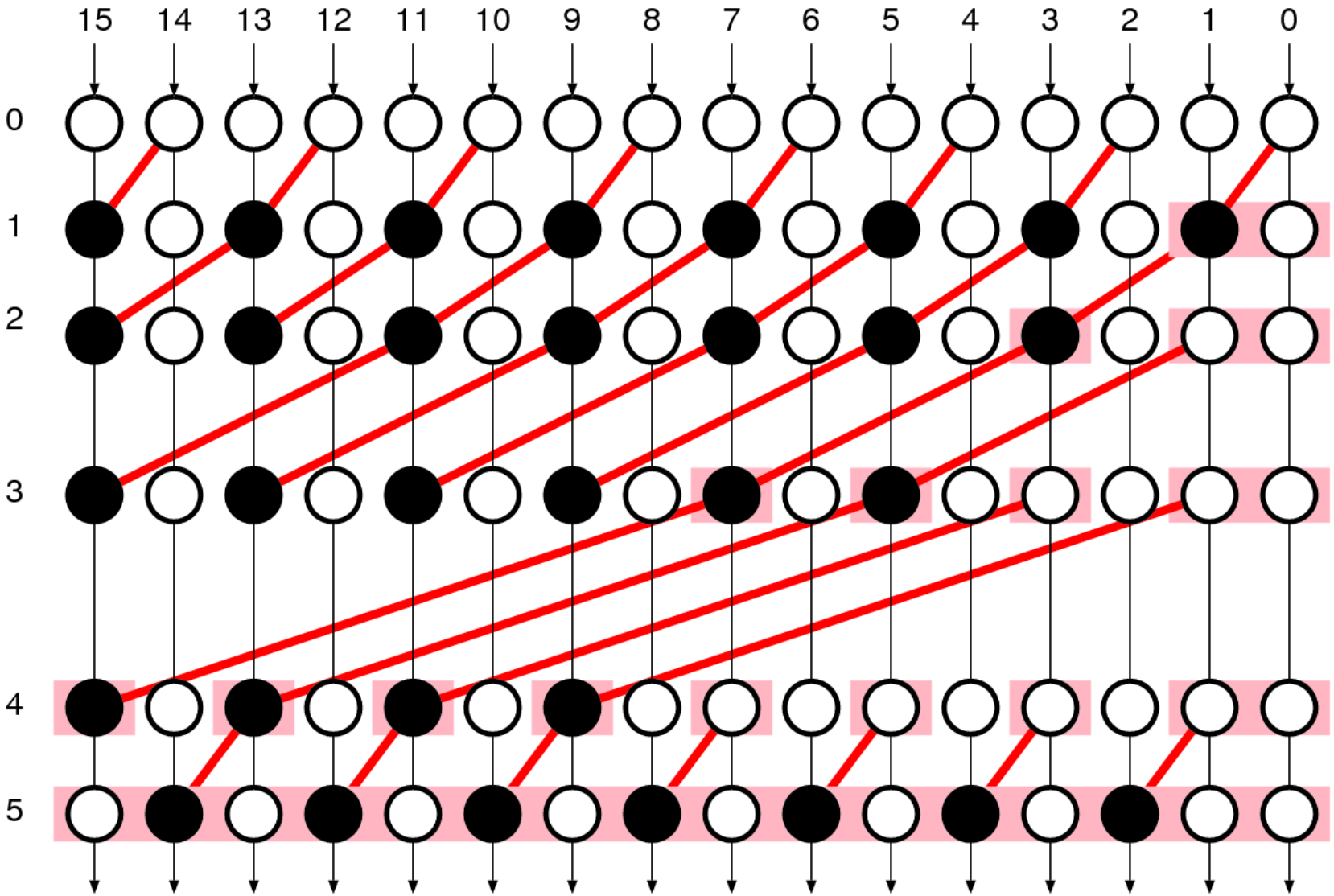
PPA: Brent Kung Adder



PPA: Kogge Stone



PPA: Huan Carlsson



Performance Comparison of Adders

Adder	Area	Speed	FO	$\approx A_\bullet$	$\approx T_\bullet$	$\approx FO_{max}$
RCA	small	slow	min	$n-1$	$n-1$	2
SK	large	fast	high	$\frac{n}{2}\log n$	$\log n$	$\frac{n}{2}$
BK	med	med	med	$2n-\log n$	$2\log n-2$	$\log n$
KS	huge	fast	min	$n\log n$	$\log n$	2
HC	large	fast	min	$\frac{n}{2}\log n$	$\log n+1$	2
CIA	med	med	med	$2n-2\sqrt{n}$	$2\sqrt{n}$	$2\sqrt{n}$

Summary Adders

- **RCA is a very efficient adder**
  - It is the smallest and simplest adder, for small bit widths it is not too slow either
- **BK offers good compromise**
  - Traditional Carry Lookahead Adder is BK with 4-bit groups
- **SK suffers from high fanout**
  - In theory it is very fast, high fanout slows it down
- **KS and HC are very fast, but suffer from routing**
  - The fastest adders, resource and routing intensive

How are Adders Implemented in Verilog?

- **You simply write**

```
assign a = b + c;
```

- **The synthesizer will figure out what to do**
  - Most synthesizers have a large library of adder architectures
  - Depending on the design constraints, an architecture is chosen

- **FPGA design is slightly special**
  - Most modern FPGAs have embedded fast adders, these are faster than adders mapped to logic.