Lecture 5:
Conditional Sum, Parallel Prefix Network Adders

ECE 645—Computer Arithmetic

2/26/08

Lecture Roadmap

- Fast Adders cont'd
  - Conditional Sum
  - Parallel Prefix Network Adders

Required Reading

- B. Parhami, *Computer Arithmetic: Algorithms and Hardware Design*
  - Chapter 6, Carry-Lookahead Adders, Sections 6.4-6.5, pp. 98-104.
  - Chapter 7, Variations in Fast Adders, Section 7.4-7.5, pp. 116-120.
- Note errata at:
  - http://www.ece.ucsb.edu/~parhami/text_comp_arit.htm#errors

Fast Adders

Solutions to Carry Propagate Problem

1. Detect the end of propagation rather than wait for the worst-case time

2. Speed-up propagation via
   - **look-ahead**
   - **carry select, etc**.

3. Limit carry propagation to within a small number of bits

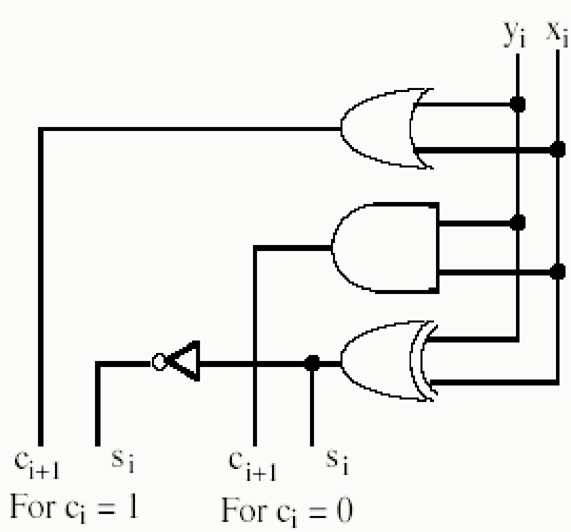
4. Eliminate carry propagation through the redundant number representation

Conditional Sum Adders

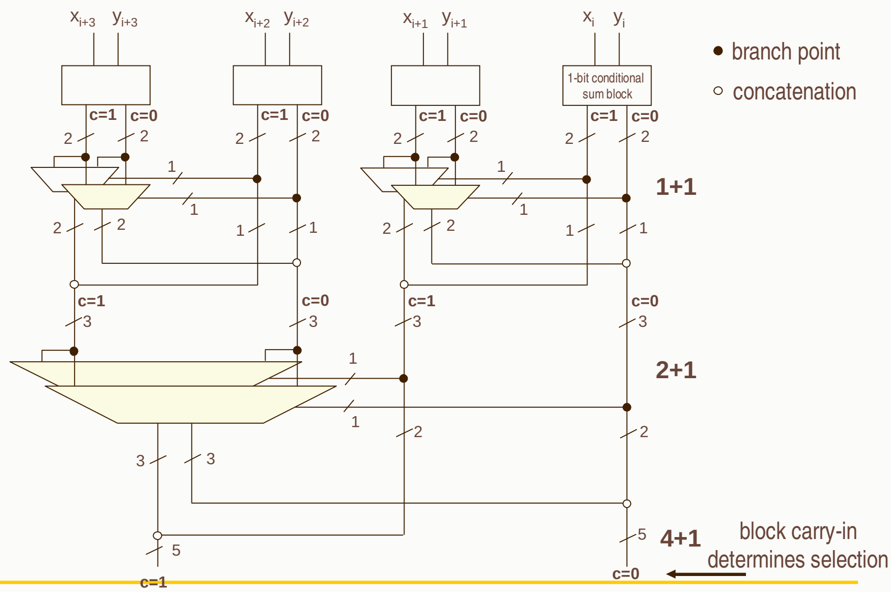
Conditional Sum Adder

- Extension of carry-select adder
- Carry select adder
  - One-level using k/2-bit adders
  - Two-level using k/4-bit adders
  - Three-level using k/8-bit adders
  - Etc.
- Assuming k is a power of two, eventually have an extreme where there are $\log_2 k$-levels using 1-bit adders
  - This is a conditional sum adder

Conditional Sum Adder: Top-Level Block for One Bit Position



Three Levels of a Conditional Sum Adder



16-Bit Conditional Sum Adder Example

Block width	Block carry-in		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	c_{in}
		x	0	0	1	0	0	1	1	0	1	1	1	0	1	0	1	0	
		y	0	1	0	0	1	0	1	1	0	1	0	1	1	1	0	1	
1	0	s	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1	
		c	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0
	1	s	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0		
		c	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1		
2	0	s	0	1	1	0	1	1	0	1	0	0	1	1	0	1	1	1	
		c	0		0		0		1		1		0		1		0		
	1	s	1	0	1	1	0	0	1	0	0	1	0	0	1	0			
		c	0		0		1		1		1		1		1				
4	0	s	0	1	1	0	0	0	0	1	0	0	1	1	0	1	1	1	
		c	0				1				1				1				
	1	s	0	1	1	1	0	0	1	0	0	1	0	0					
		c	0				1				1								
8	0	s	0	1	1	1	0	0	0	1	0	1	0	0	0	1	1	1	
		c	0								1								
	1	s	0	1	1	1	0	0	1	0									
		c	0																
16	0	s	0	1	1	1	0	0	1	0	0	1	0	0	0	1	1	1	
		c	0																
	1	s																	
		c																	

Block sum and block carry-out

c_{out}

Conditional Sum Adder Metrics

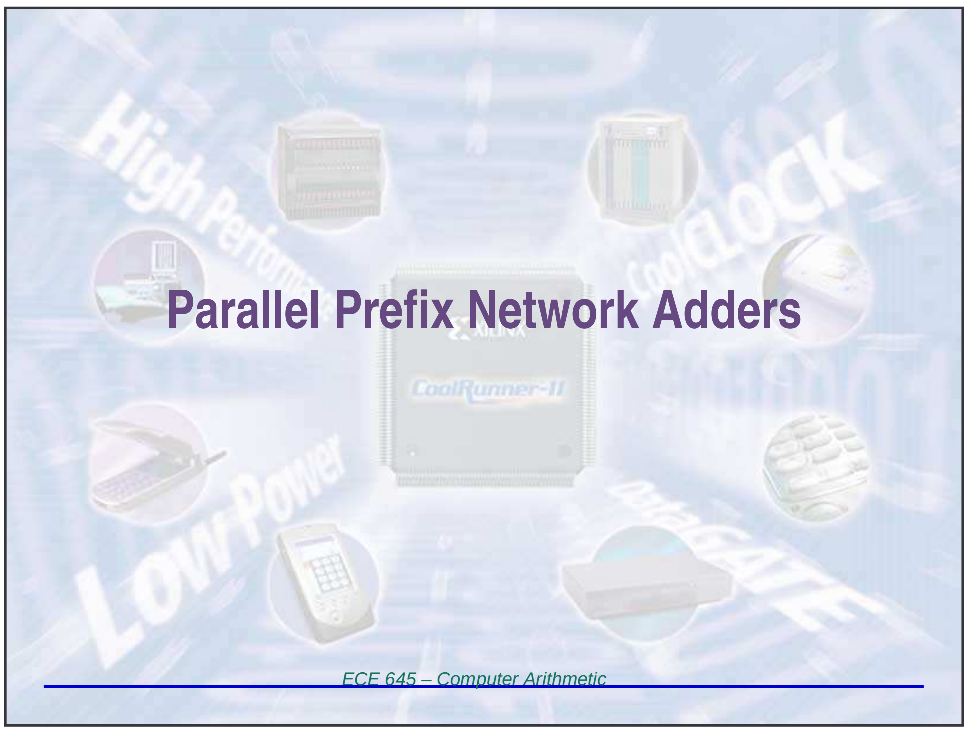
Multilevel carry-select idea carried out to the extreme, until we arrive at single-bit blocks.

$$C(k) \approx 2C(k/2) + k + 2 \approx k(\log_2 k + 2) + k\,C(1)$$

$$T(k) = T(k/2) + 1 = \log_2 k + T(1)$$

where C(1) and T(1) are the cost and delay of the circuit of Fig. 7.11 used at the top to derive the sum and carry bits with a carry-in of 0 and 1

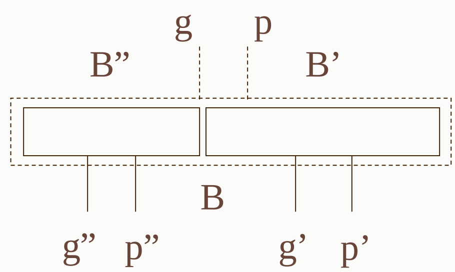
The term k + 2 in the first recurrence represents an upper bound on the number of single-bit 2-to-1 multiplexers needed for combining two k/2-bit adders into a k-bit adder



Parallel Prefix Network Adders

Parallel Prefix Network Adders

Basic component - Carry operator (1)

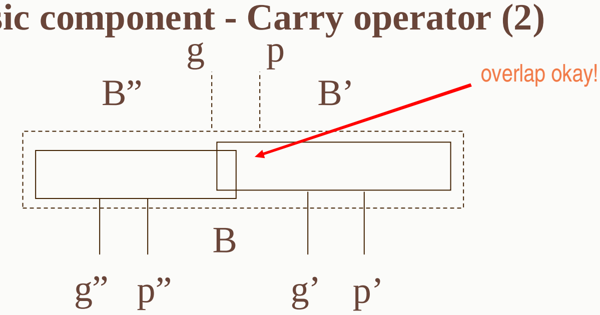


g = g” + g’p”
p = p’p”

(g, p) = (g’, p’) ¢ (g”, p”) = (g” + g’p”, p’p”)

Parallel Prefix Network Adders

Basic component - Carry operator (2)



g = g” + g’p”
p = p’p”

(g, p) = (g’, p’) ¢ (g”, p”) = (g” + g’p”, p’p”)

Properties of the carry operator ¢

Associative

$$[(g_1, p_1) ¢ (g_2, p_2)] ¢ (g_3, p_3) = (g_1, p_1) ¢ [(g_2, p_2) ¢ (g_3, p_3)]$$

Not commutative

$$(g_1, p_1) ¢ (g_2, p_2) \neq (g_2, p_2) ¢ (g_1, p_1)$$

Parallel Prefix Network Adders

Major concept

Given:

(g_0, p_0) (g_1, p_1) (g_2, p_2) …. (g_{k-1}, p_{k-1})

Find:

$(g_{[0,0]}, p_{[0,0]})$ $(g_{[0,1]}, p_{[0,1]})$ $(g_{[0,2]}, p_{[0,2]})$ … $(g_{[0,k-1]}, p_{[0,k-1]})$

block generate from index 0 to k-1

$$c_i = g_{[0,i-1]} + c_0 p_{[0,i-1]}$$

Similar to Parallel Prefix Sum Problem

Parallel Prefix Sum Problem

Given:

x_0 x_1 x_2 … x_{k-1}

Find:

x_0 x_0+x_1 $x_0+x_1+x_2$ … $x_0+x_1+x_2+ \ldots + x_{k-1}$

Parallel Prefix Adder Problem

Given:

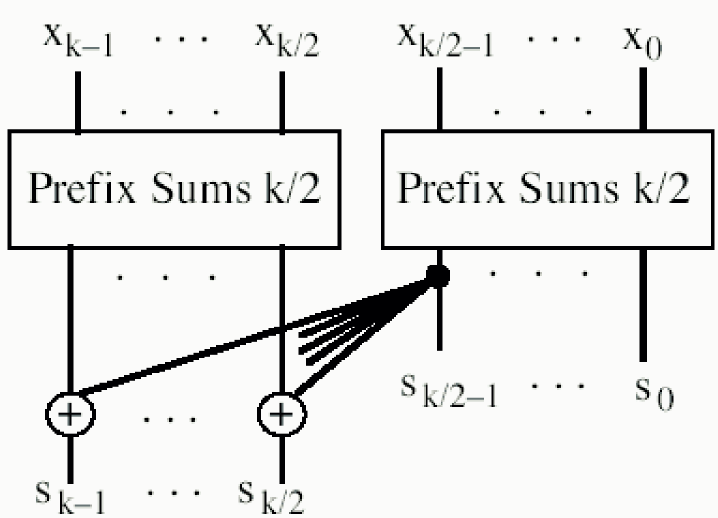
x_0 x_1 x_2 … x_{k-1}

Find:

x_0 x_0 ¢ x_1 x_0 ¢ x_1 ¢ x_2 … x_0 ¢ x_1 ¢ x_2 ¢ … ¢ x_{k-1}

where $x_i = (g_i, p_i)$

Parallel Prefix Sums Network I



Parallel Prefix Sums Network I – Cost (Area) Analysis

Cost = **C(k)** = 2 C(k/2) + k/2 =

= 2 [2C(k/4) + k/4] + k/2 = 4 C(k/4) + k/2 + k/2 =

= …. =

= $2^{\log_2 k-1}$C(2) + k/2 ($\log_2$k-1) =

= **k/2 $\log_2$k**

| C(2) = 1 |

Example:

C(16) = 2 C(8) + 8 = 2[2 C(4) + 4] + 8 =

= 4 C(4) + 16 = 4 [2 C(2) + 2] + 16 =

= 8 C(2) + 24 = 8 + 24 = **32** = (16/2) $\log_2$ 16

Parallel Prefix Sums Network I – Delay Analysis

Delay = **D(k)** = D(k/2) + 1 =

= [D(k/4) + 1] + 1 = D(k/4) + 1 + 1 =

= …. =

= $\log_2$k

| D(2) = 1 |

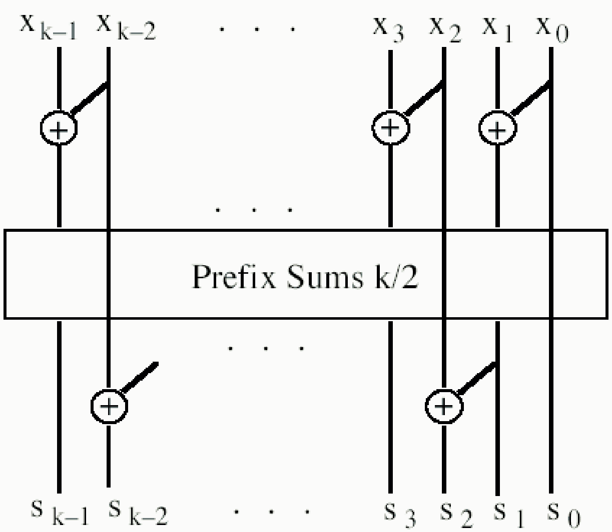
Example:

D(16) = D(8) + 1 = [D(4) + 1] + 1 =

= D(4) + 2 = [D(2) + 1] + 2 =

= **4** = $\log_2$ 16

Parallel Prefix Sums Network II (Brent-Kung)



Parallel Prefix Sums Network II – Cost (Area) Analysis

Cost = **C(k)** = C(k/2) + k-1 =

= [C(k/4) + k/2-1] + k-1 = C(k/4) + 3k/2 - 2 =

= …. =

= $C(2) + (2k - 2k/2^{(\log_2 k-1)}) - (\log_2 k-1)$ =

= $\mathbf{2k - 2 - \log_2 k}$

C(2) = 1

Example:

C(16) = C(8) + 16-1 = [C(4) + 8-1] + 16-1 =

= C(2) + 4-1 + 24-2 = 1 + 28 - 3 = **26**

= $2 \cdot 16 - 2 - \log_2 16$

Parallel Prefix Sums Network II – Delay Analysis

Delay = **D(k)** = D(k/2) + 2 =

= [D(k/4) + 2] + 2 = D(k/4) + 2 + 2 =

= …. =

= 2 log$_2$k - 1

| D(2) = 1 |

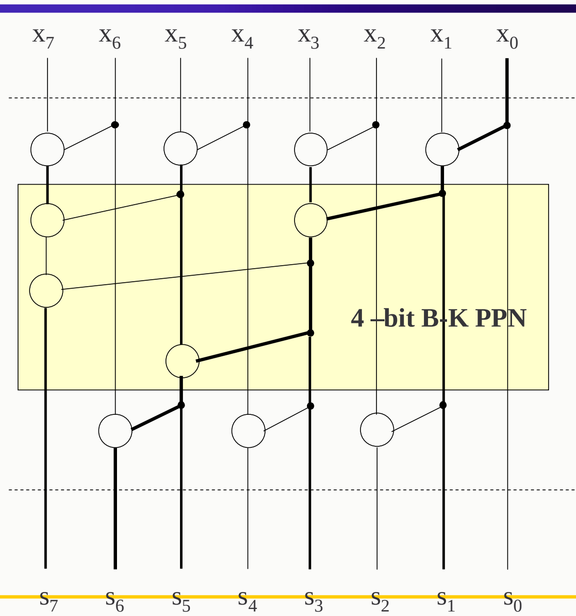
Example:

D(16) = D(8) + 2 = [D(4) + 2] + 2 =

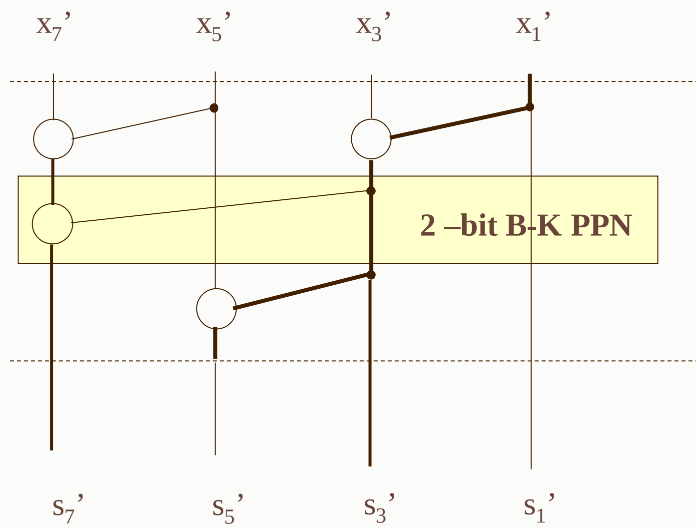
= D(4) + 4 = [D(2) + 2] + 4 =

= **7** = 2 log$_2$ 16 - 1

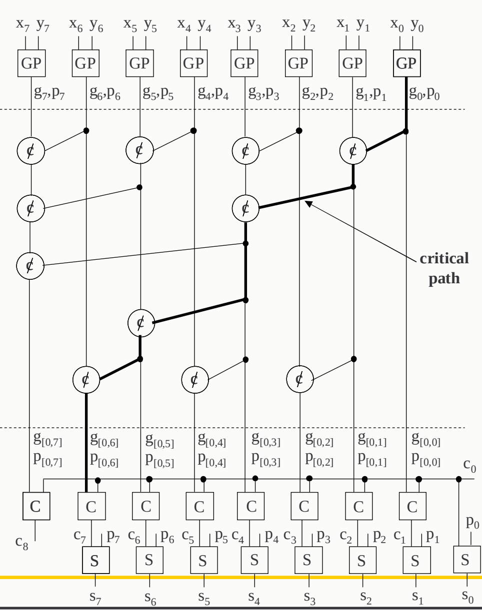
8-bit Brent-Kung Parallel Prefix Network



4-bit Brent-Kung Parallel Prefix Network



8-bit Brent-Kung Parallel Prefix Network Critical Path



Critical Path

GP

$g_i = x_i \;\; y_i$
$p_i = x_i \oplus y_i$

1 gate delay

¢

$g = g'' + g' \; p''$
$p = p' \; p''$

2 gate delays

C

$c_{i+1} = g_{[0,i]} + c_0 \, p_{[0,i]}$

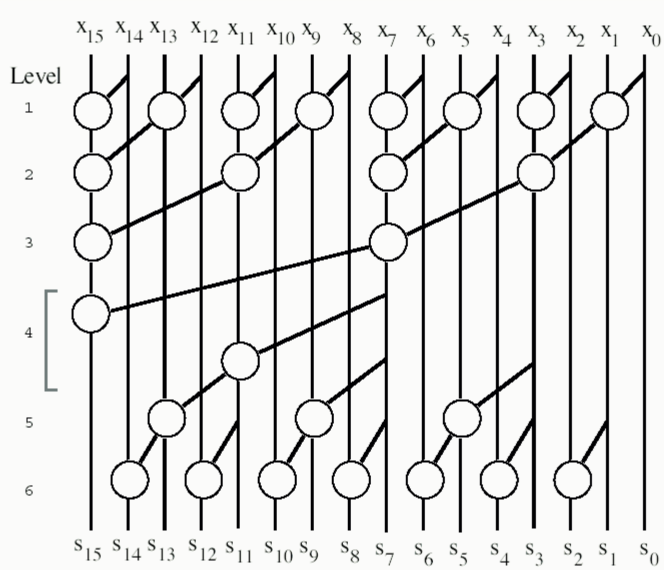
2 gate delays

S

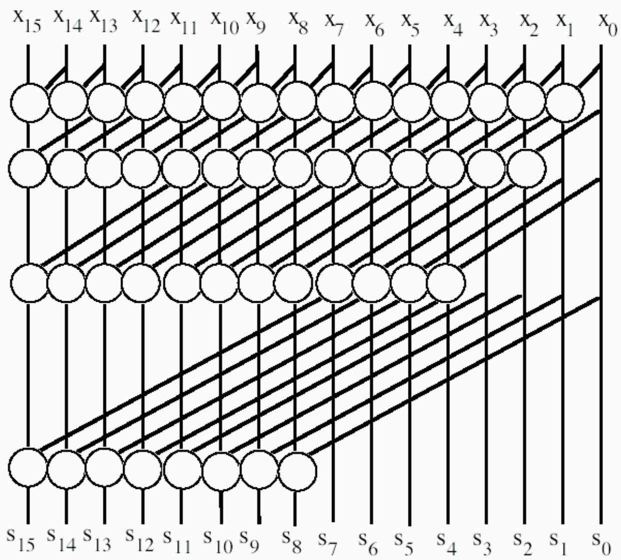
$s_i = p_i \oplus c_i$

1 gate delay

Brent-Kung Parallel Prefix Graph for 16 Inputs



Kogge-Stone Parallel Prefix Graph for 16 Inputs

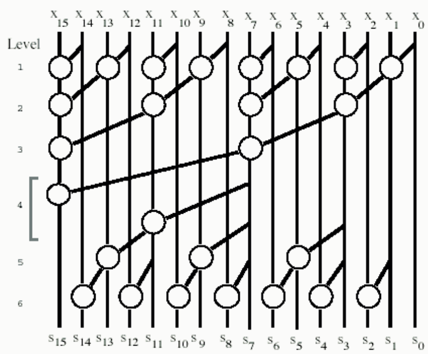


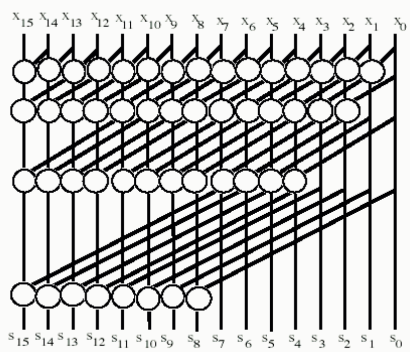
Parallel Prefix Network Adders

Comparison of architectures

	Network 2 Brent-Kung	Hybrid	Kogge-Stone
Delay(k)	$2 \log_2 k - 2$	$\log_2 k+1$	$\log_2 k$
Cost(k)	$2k - 2 - \log_2 k$	$k/2 \log_2 k$	$k \log_2 k - k + 1$
Delay(16)	6	5	4
Cost(16)	26	32	49
Delay(32)	8	6	5
Cost(32)	57	80	129

Latency vs. Area Tradeoff

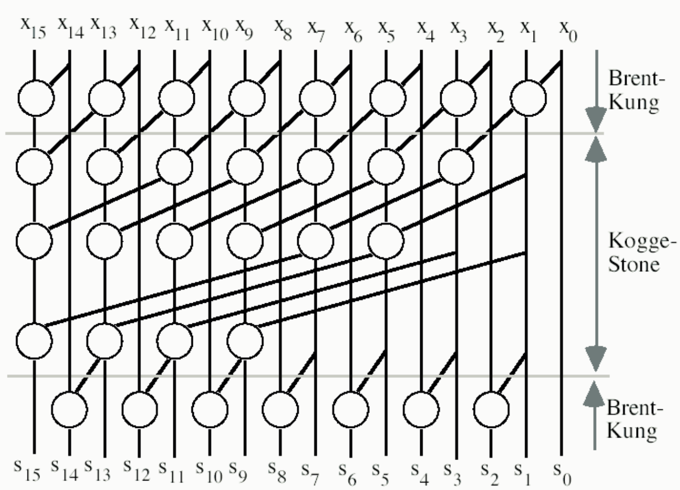




B-K: Six levels, 26 cells   K-S: Four levels, 49 cells

Hybrid Brent-Kung/Kogge-Stone Parallel Prefix Graph for 16 Inputs

Hybrid: Five levels, 32 cells



Fast Adders Summary

- Default adder
  - Carry-ripple
- Fast adders
  - Carry lookahead: breaks carry-in to carry-out chain by having each output sum dependent on carry-in directly (and not carry-out of preceding stage)
  - Carry select: uses duplicated hardware to pre-compute multiple outcomes, correct outcome selected
  - Hybrid adders: a combination of two fast adder structures
  - Conditional sum: extension of carry select using one-bit blocks
  - Parallel prefix network: parallel network architecture to compute group propagate and group generate for each index of the adder