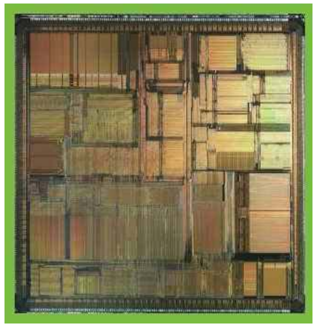
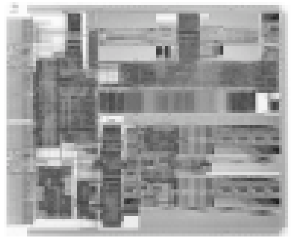
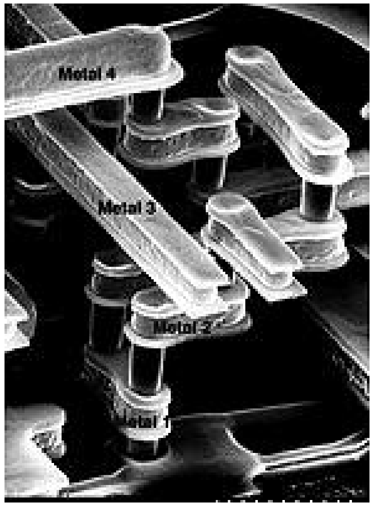
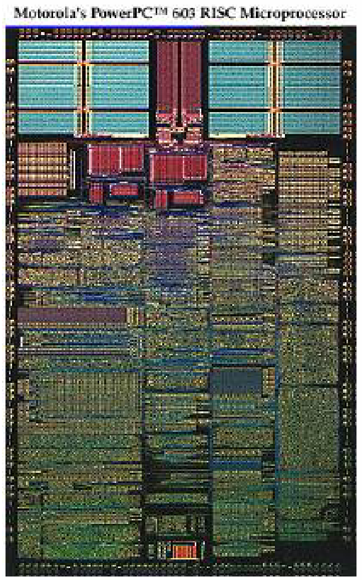




VLSI Arithmetic

Lecture 6





Prof. Vojin G. Oklobdzija
University of California

http://www.ece.ucdavis.edu/acsel

Review

Lecture 5

Prefix Adders
and
Parallel Prefix Adders

ADDITION: TWO-STEP PROCESS

1. Obtain carries (carry at i depends on $j \leq i$)
   – non-trivial to do fast
2. Compute sum bits (local function)

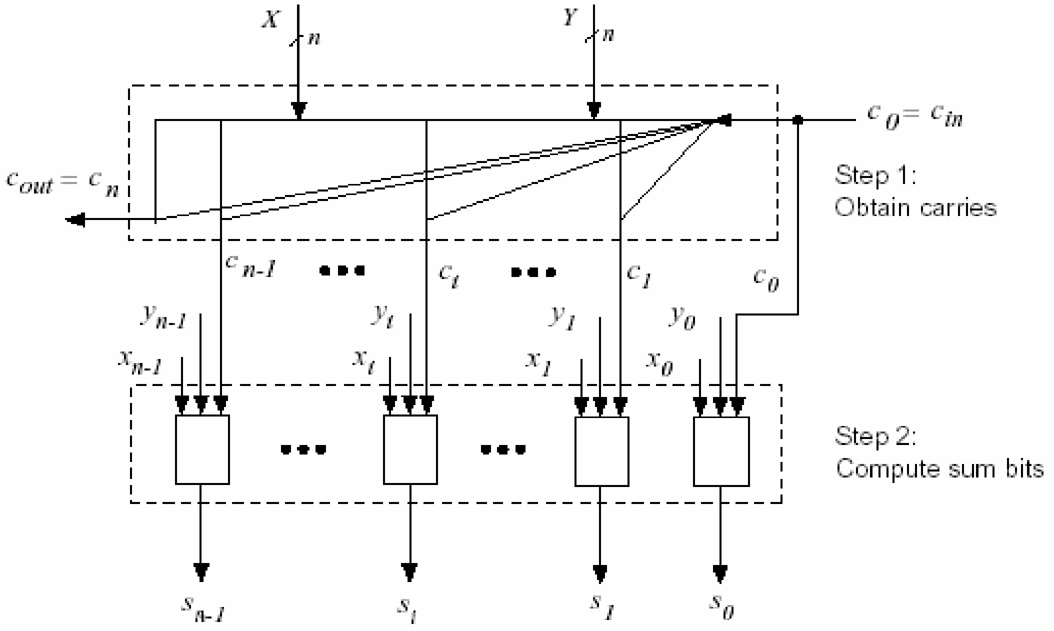


Figure 2.2: Steps in addition.

from: Ercegovac-Lang

Prefix Adders

Following recurrence operation is defined:

$$(g, p)o(g',p')=(g+pg', pp')$$

such that:

$$G_i, P_i = \begin{cases} (g_0, p_0) & i=0 \\ (g_i, p_i)o(G_{i-1}, P_{i-1}) & 1 \leq i \leq n \end{cases}$$

$c_{i+1} = G_i$ for $i=0, 1, \ldots.. n$

$c_1 = g_0 + p_0 c_{in}$ $(g_{-1}, p_{-1})=(c_{in},c_{in})$

This operation is associative, but not commutative

It can also span a range of bits (overlapping and adjacent)

Parallel Prefix Adders: S. Knowles 1999

operation '•':

$$\begin{pmatrix} g \\ \overline{k} \end{pmatrix}_i \bullet \begin{pmatrix} g \\ \overline{k} \end{pmatrix}_j = \begin{pmatrix} g_i + \overline{k}_i . g_j \\ \overline{k_i . k_j} \end{pmatrix}$$

$$\begin{pmatrix} g \\ \overline{k} \end{pmatrix}_{h\ldots j} \bullet \begin{pmatrix} g \\ \overline{k} \end{pmatrix}_{j\ldots k} = \begin{pmatrix} g \\ \overline{k} \end{pmatrix}_{h\ldots i} \bullet \begin{pmatrix} g \\ \overline{k} \end{pmatrix}_{i\ldots k}$$

operation is associative: h>i≥j≥k

$$\begin{pmatrix} g \\ \overline{k} \end{pmatrix}_{h\ldots j} \bullet \begin{pmatrix} g \\ \overline{k} \end{pmatrix}_{i\ldots k} = \begin{pmatrix} g \\ \overline{k} \end{pmatrix}_{h\ldots k}$$

operation is idempotent: h>i≥j≥k

$$\begin{pmatrix} c_{i+1} \\ \overline{k_i . k_{i-1} . k_{i-2} \ldots k_0} \end{pmatrix} = \begin{pmatrix} g \\ \overline{k} \end{pmatrix}_i \bullet \begin{pmatrix} g \\ \overline{k} \end{pmatrix}_{i-1} \bullet \begin{pmatrix} g \\ \overline{k} \end{pmatrix}_{i-2} \cdots \bullet \begin{pmatrix} g \\ \overline{k} \end{pmatrix}_0$$

produces carry: $c_{in}=0$

Prefix adders

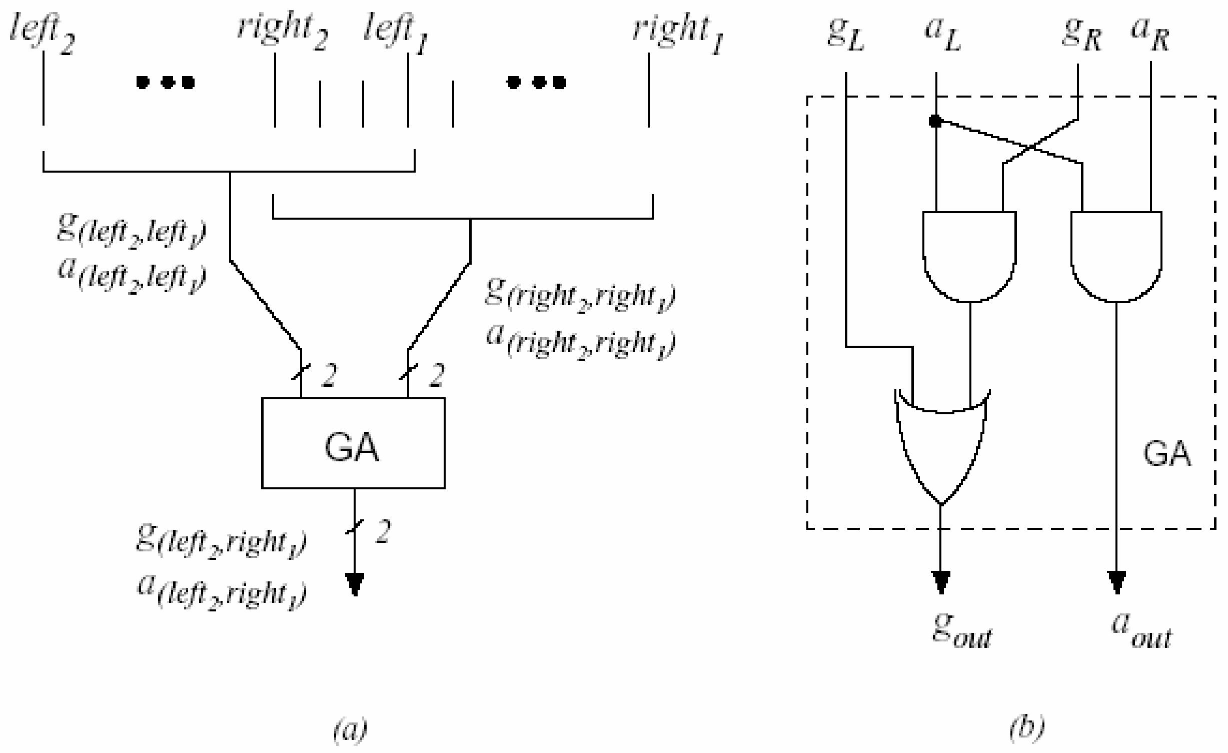
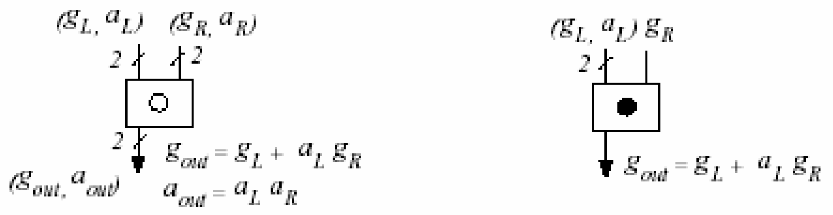


Figure 2.17: Composition of spans in computing (g, a) signals.

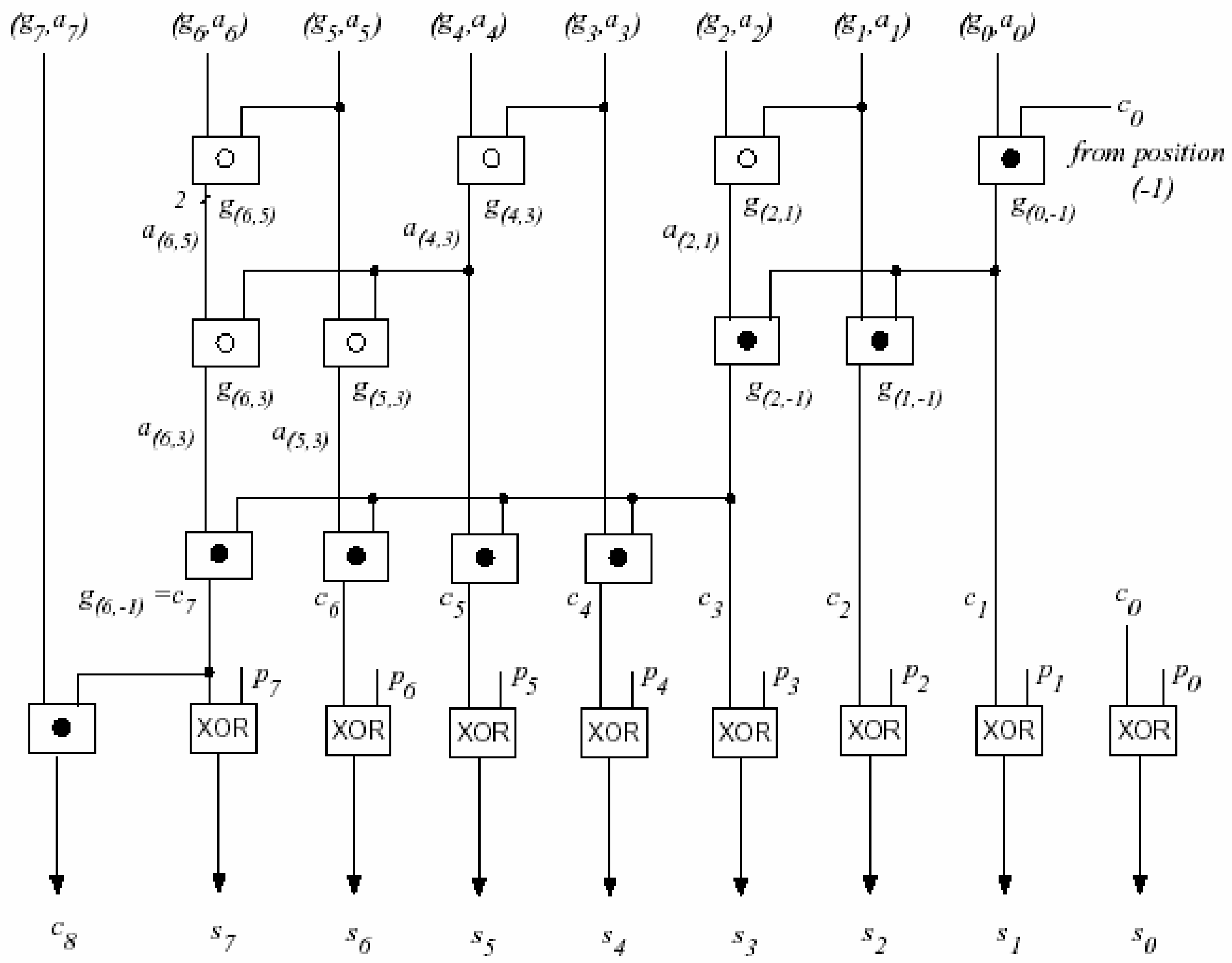
from: Ercegovac-Lang

Parallel Prefix Adders: variety of possibilities



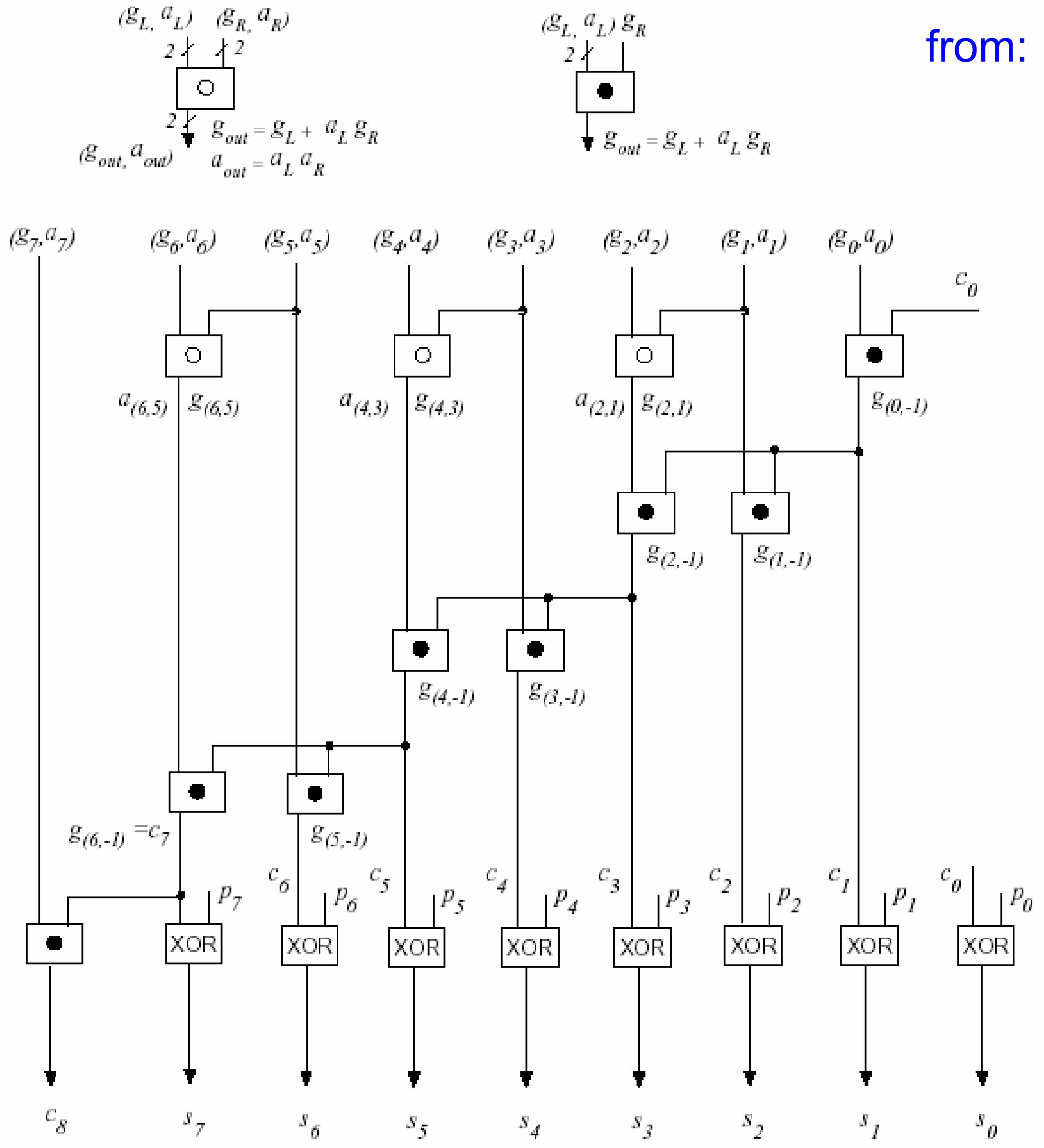
(g_L, a_L) (g_R, a_R)
2
2
○
2
$g_{out} = g_L + a_L g_R$
$a_{out} = a_L a_R$
(g_{out}, a_{out})
(g_L, a_L) g_R
2
●
$g_{out} = g_L + a_L g_R$

from: Ercegovac-Lang



Parallel Prefix Adders: variety of possibilities

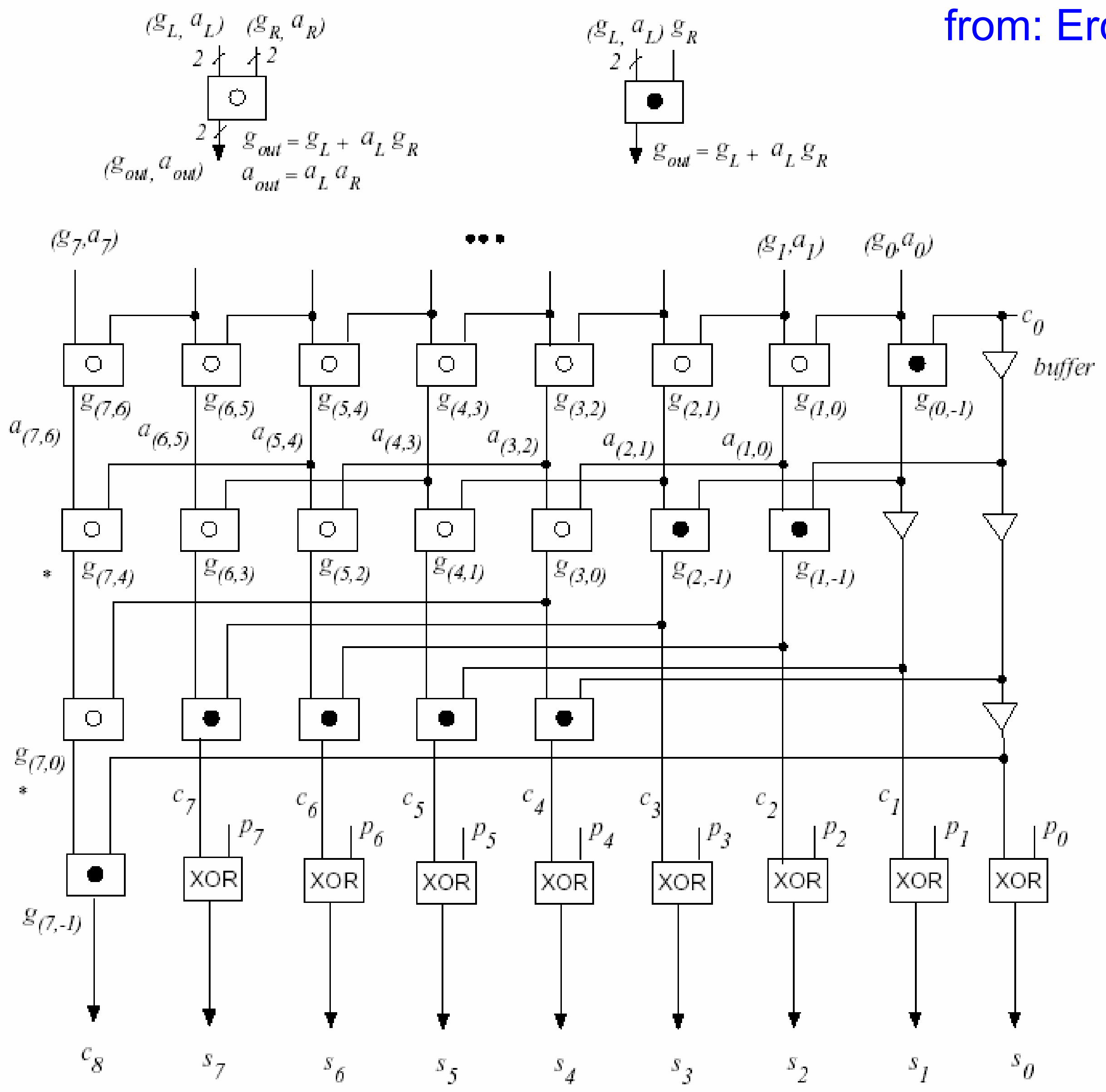
from: Ercegovac-Lang



(g_L, a_L) (g_R, a_R)
2
2
2
(g_{out}, a_{out})
$g_{out} = g_L + a_L g_R$
$a_{out} = a_L a_R$
(g_L, a_L) g_R
2
$g_{out} = g_L + a_L g_R$
(g_7,a_7) (g_6,a_6) (g_5,a_5) (g_4,a_4) (g_3,a_3) (g_2,a_2) (g_1,a_1) (g_0,a_0)
c_0
$a_{(6,5)}$ $g_{(6,5)}$
$a_{(4,3)}$ $g_{(4,3)}$
$a_{(2,1)}$ $g_{(2,1)}$
$g_{(0,-1)}$
$g_{(2,-1)}$ $g_{(1,-1)}$
$g_{(4,-1)}$ $g_{(3,-1)}$
$g_{(6,-1)} = c_7$
$g_{(5,-1)}$
c_6 c_5 c_4 c_3 c_2 c_1 c_0
p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0
XOR
c_8 s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0

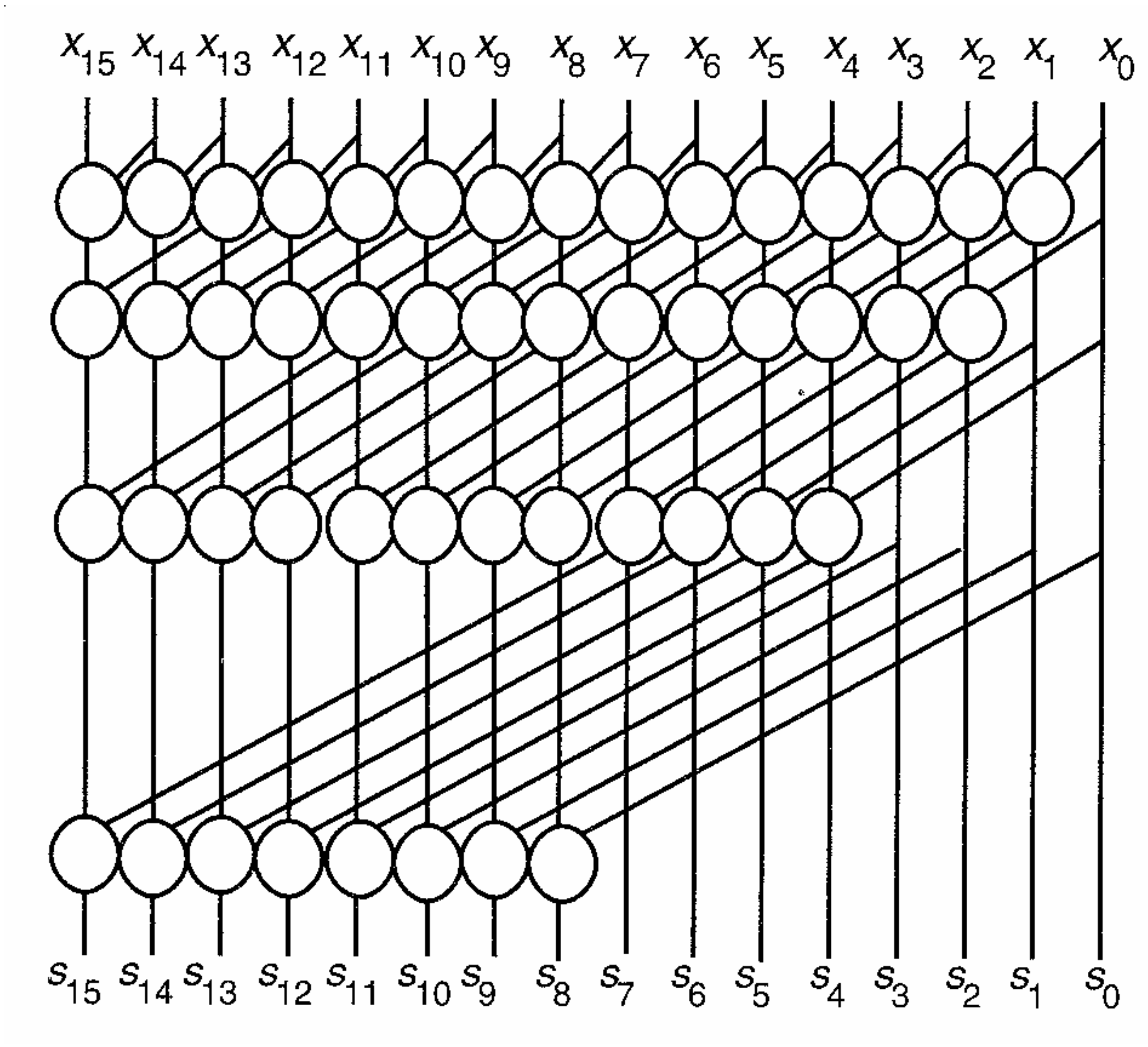
Parallel Prefix Adders: variety of possibilities

from: Ercegovac-Lang

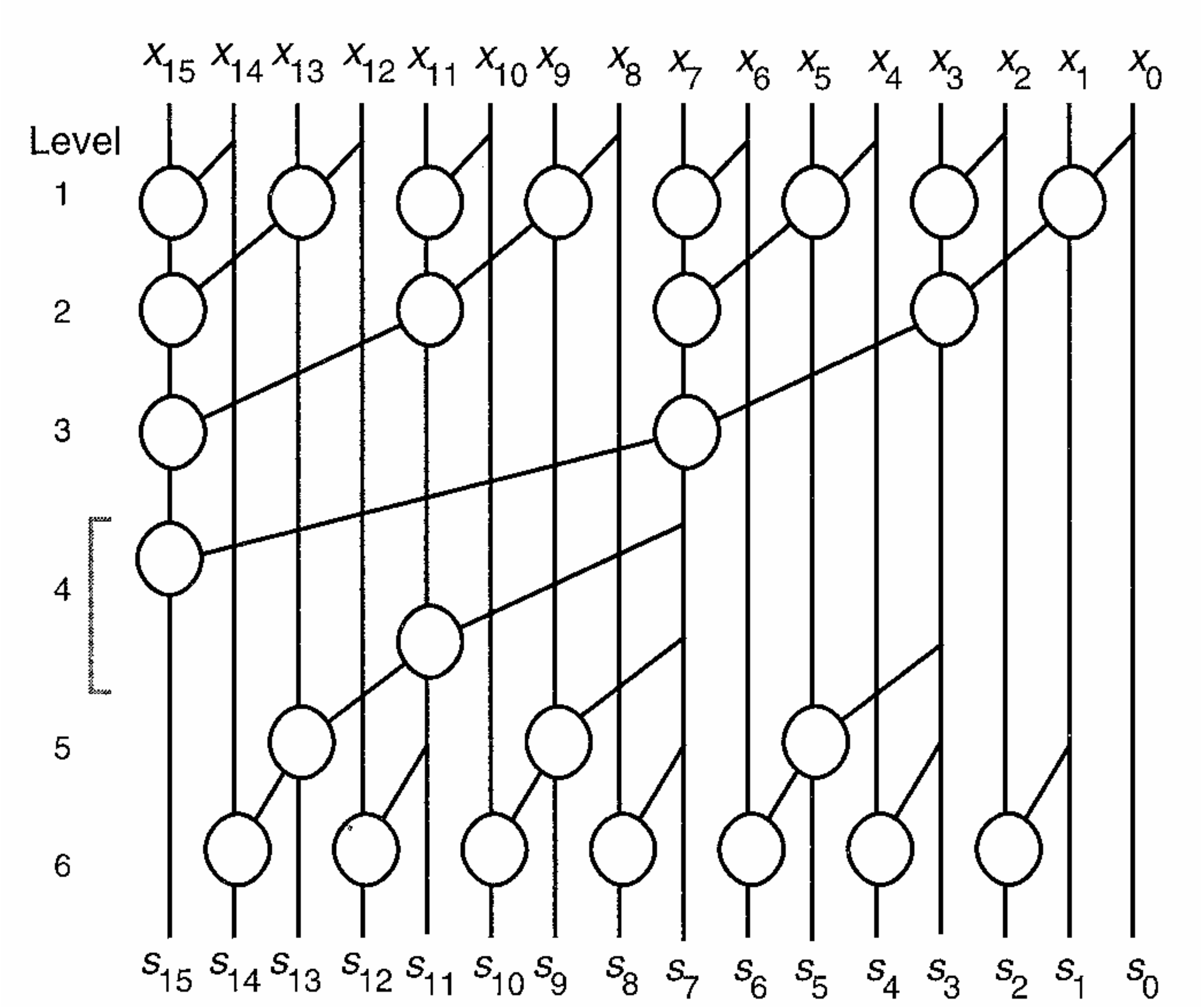


(g_L, a_L) (g_R, a_R)
2
(g_{out}, a_{out})
$g_{out} = g_L + a_L g_R$
$a_{out} = a_L a_R$
(g_L, a_L) g_R
$g_{out} = g_L + a_L g_R$
(g_7, a_7)
(g_1, a_1) (g_0, a_0)
c_0
buffer
$g_{(7,6)}$ $g_{(6,5)}$ $g_{(5,4)}$ $g_{(4,3)}$ $g_{(3,2)}$ $g_{(2,1)}$ $g_{(1,0)}$ $g_{(0,-1)}$
$a_{(7,6)}$ $a_{(6,5)}$ $a_{(5,4)}$ $a_{(4,3)}$ $a_{(3,2)}$ $a_{(2,1)}$ $a_{(1,0)}$
$g_{(7,4)}$ $g_{(6,3)}$ $g_{(5,2)}$ $g_{(4,1)}$ $g_{(3,0)}$ $g_{(2,-1)}$ $g_{(1,-1)}$
$g_{(7,0)}$
c_7 c_6 c_5 c_4 c_3 c_2 c_1
p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0
XOR
$g_{(7,-1)}$
c_8 s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0

Kogge-Stone Adder

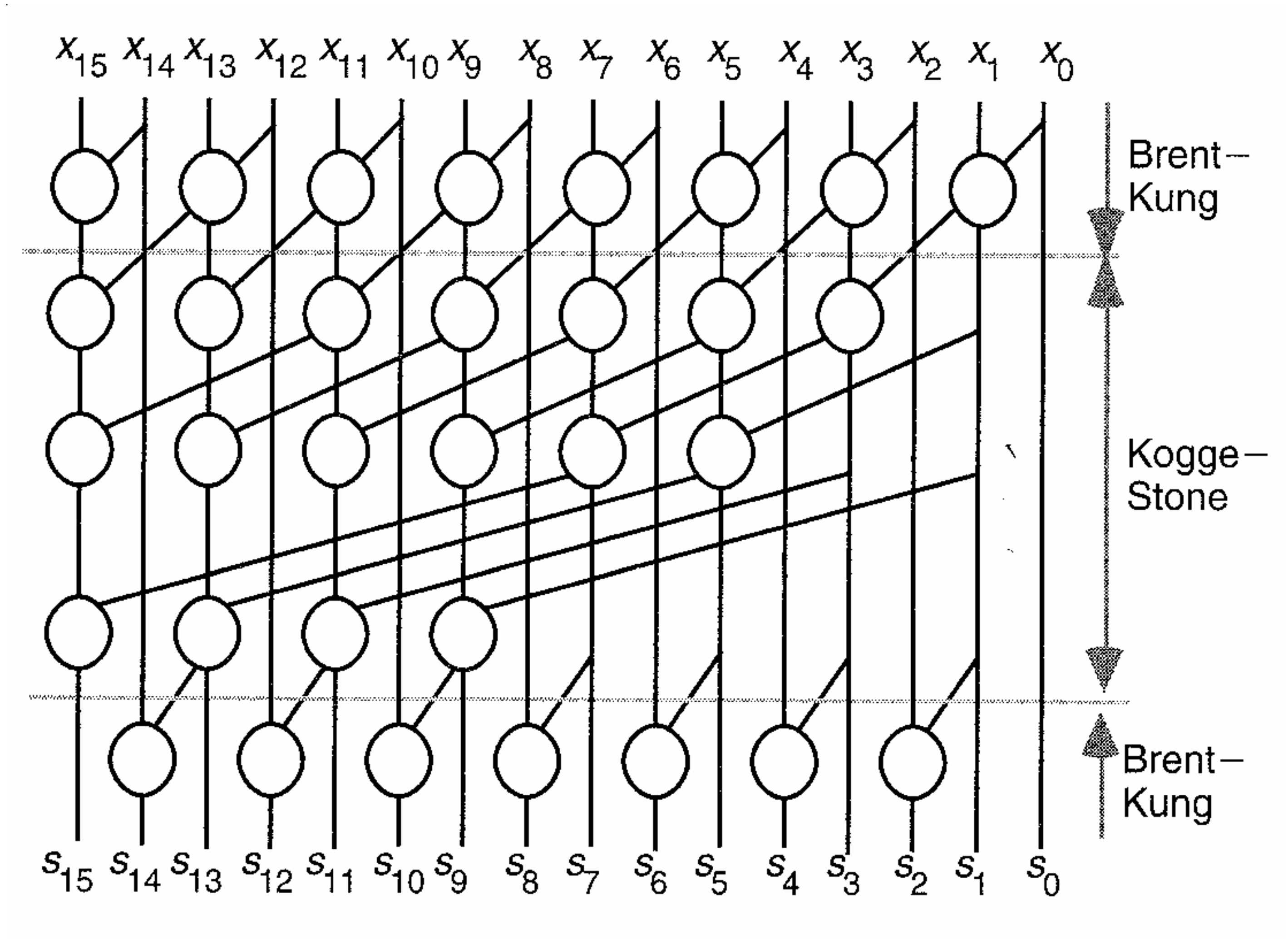

x_{15} x_{14} x_{13} x_{12} x_{11} x_{10} x_9 x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0
s_{15} s_{14} s_{13} s_{12} s_{11} s_{10} s_9 s_8 s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0

Brent-Kung Adder



x_{15} x_{14} x_{13} x_{12} x_{11} x_{10} x_9 x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0
Level
1
2
3
4
5
6
s_{15} s_{14} s_{13} s_{12} s_{11} s_{10} s_9 s_8 s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0

Hybrid BK-KS Adder



x_{15} x_{14} x_{13} x_{12} x_{11} x_{10} x_9 x_8 x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0
Brent–Kung
Kogge–Stone
Brent–Kung
s_{15} s_{14} s_{13} s_{12} s_{11} s_{10} s_9 s_8 s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0

Pyramid Adder:

M. Lehman, “A Comparative Study of Propagation Speed-up Circuits in Binary Arithmetic Units”, IFIP Congress, Munich, Germany, 1962.

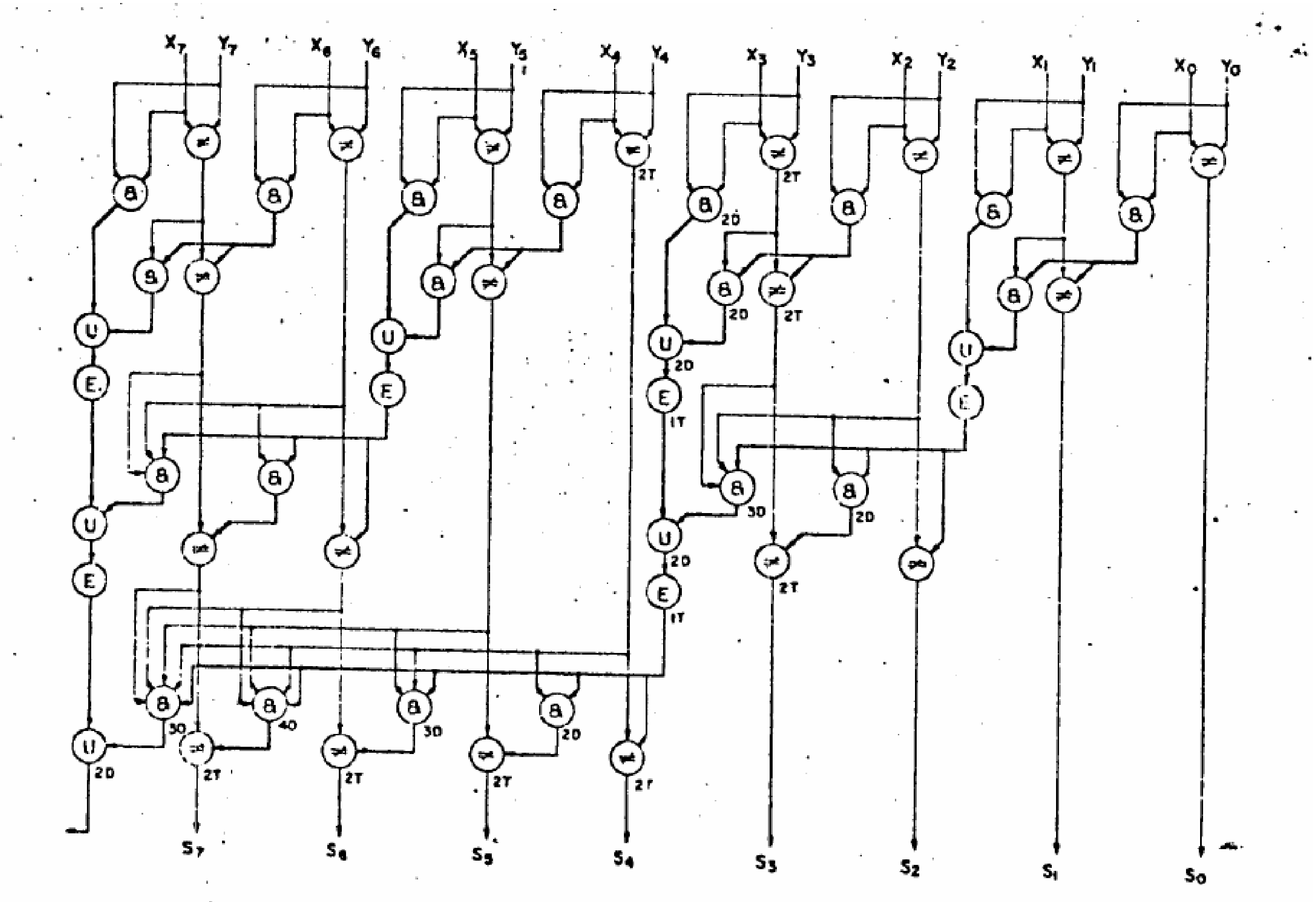


Fig. 2. Pyramid carry (eight bits) (Ex. 9).

Parallel Prefix Adders: Ladner-Fisher

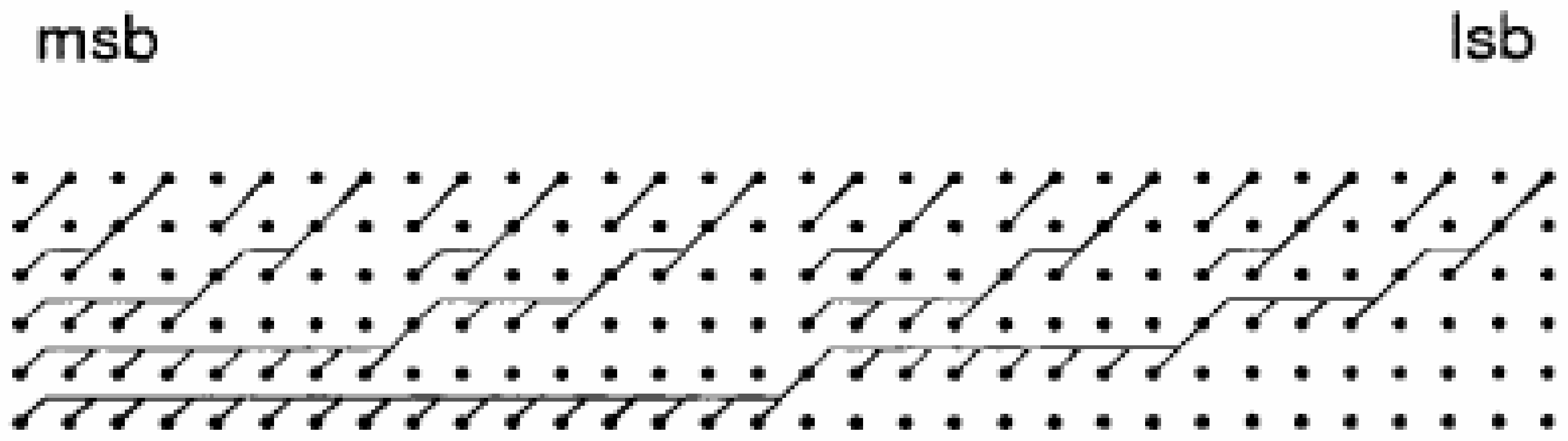
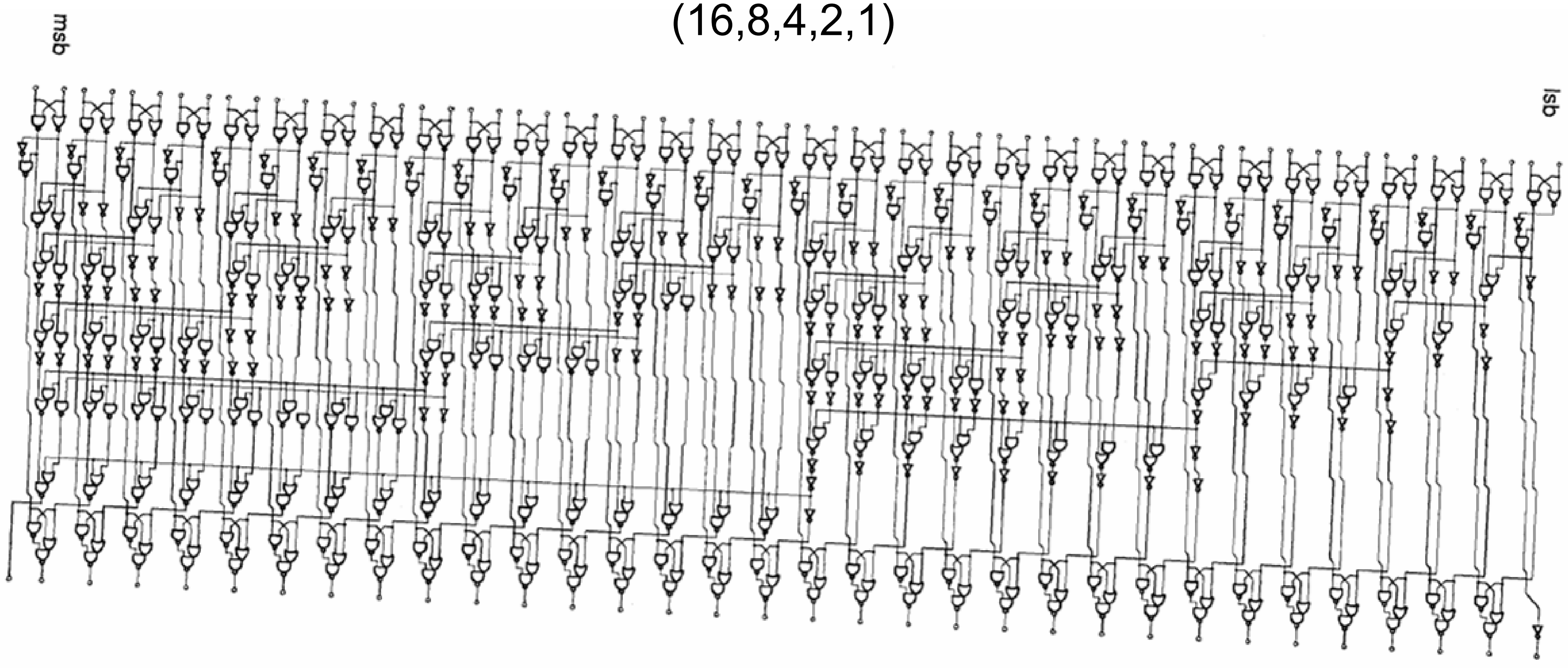


Figure 1: 32b Ladner-Fischer graph [16,8,4,2,1]

Exploits associativity, but not idempotency.
Produces minimal logical depth

Parallel Prefix Adders: Ladner-Fisher

(16,8,4,2,1)



Two wires at each level. Uniform, fan-in of two.

Large fan-out (of 16; n/2); Large capacitive loading combined with the long wires (in the last stages)

Parallel Prefix Adders: Kogge-Stone

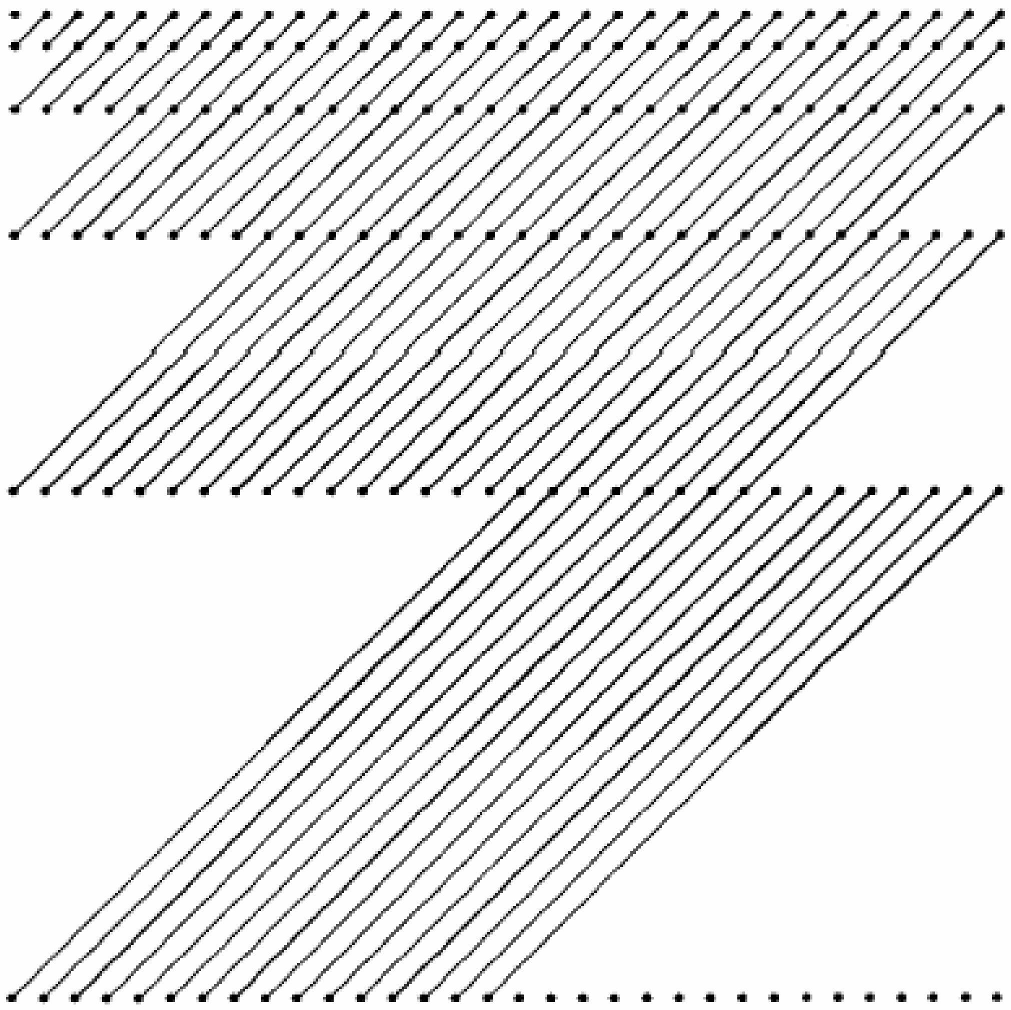


Figure 3: 32b Kogge-Stone graph [1,1,1,1,1]

Exploits idempotency to limit the fan-out to 1. Dramatic increase in wires. The wire span remains the same as in Ladner-Fisher.

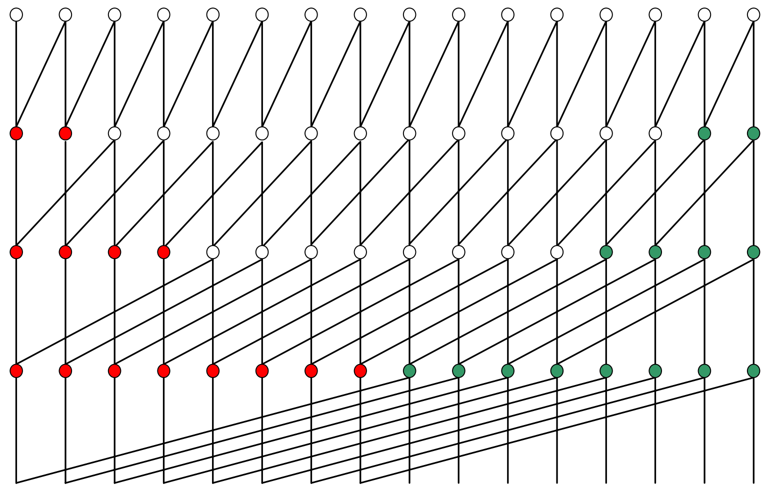
Buffers needed in both cases: K-S, L-F

Parallel Prefix Adders: Brent-Kung

- Set the fan-out to one
- Avoids explosion of wires (as in K-S)
- Makes no sense in CMOS:
  - fan-out = 1 limit is arbitrary and extreme
  - much of the capacitive load is due to wire (anyway)
- It is more efficient to insert buffers in L-F than to use B-K scheme

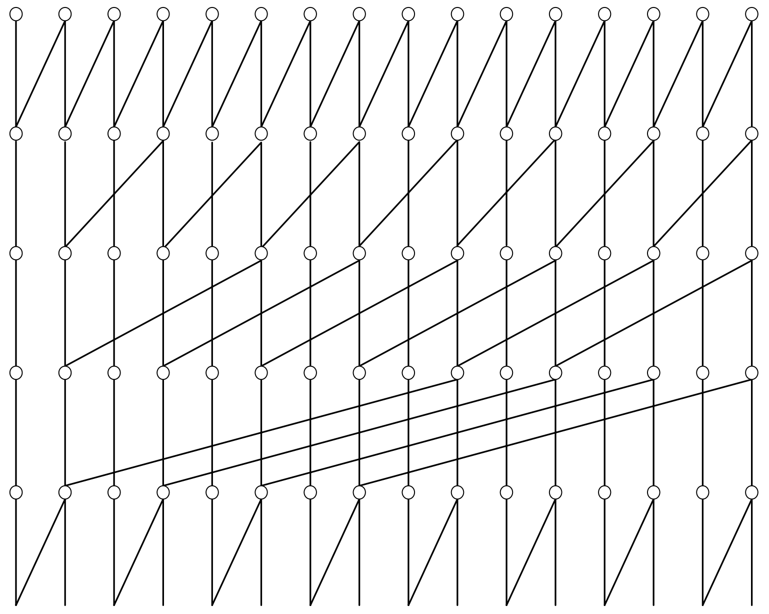
Two Parallel Prefix Adder Structures

Kogge-Stone



- ***log(bits)*** **carry stages**
- **Extra Wiring**

Han-Carlson



- ***log(bits)*** **+ 1 carry stages**
- **Reduced Wiring and Gates**

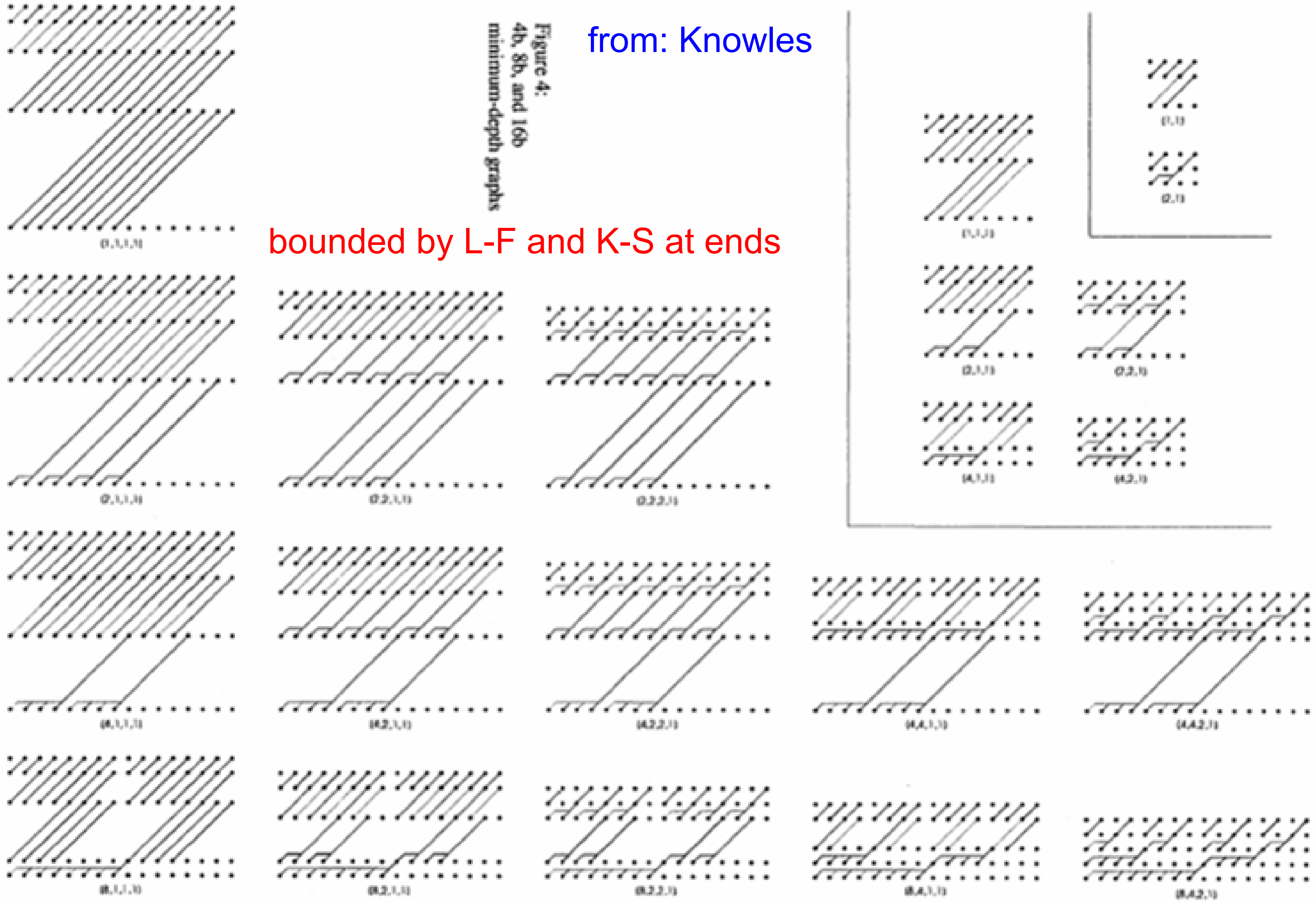
Parallel Prefix Adders: Han-Carlson

- Is a hybrid synthesis of L-F and K-S
- Trades increase in logic depth for a reduction in fan-out:
  - effectively a higher-radix variant of K-S.
  - others do it similarly by serializing the prefix computation at the higher fan-out nodes.
- Others, similarly trade the logical depth for reduction of fan-out and wire.

Parallel Prefix Adders: variety of possibilities

from: Knowles

bounded by L-F and K-S at ends



Parallel Prefix Adders: variety of possibilities

Knowles 1999

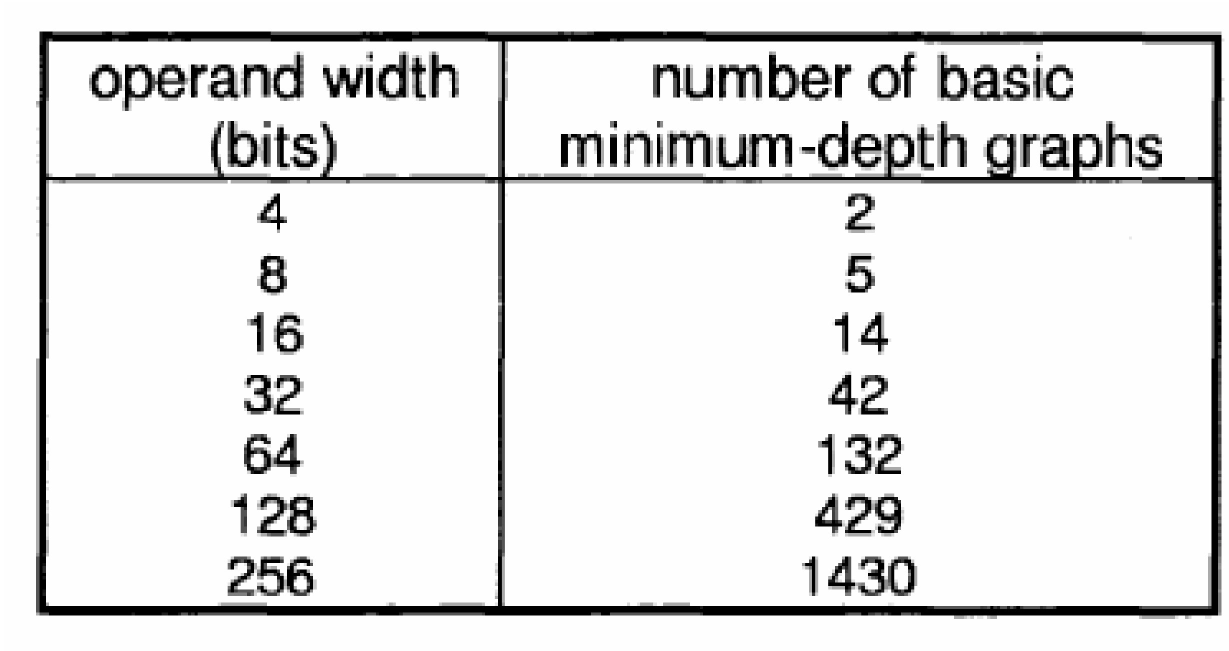
Following rules are used:

- Lateral wires at the j^{th} level span 2^j bits
- Lateral fan-out at j^{th} level is power of 2 up to 2^j
- Lateral fan-out at the j^{th} level cannot exceed that a the $(j+1)^{th}$ level.

Parallel Prefix Adders: variety of possibilities

Knowles 1999

- The number of minimal depth graphs of this type is given in:



operand width (bits)	number of basic minimum-depth graphs
4	2
8	5
16	14
32	42
64	132
128	429
256	1430

- at 4-bits there is only K-S and L-F, afterwards there are several new possibilities.

Parallel Prefix Adders: variety of possibilities

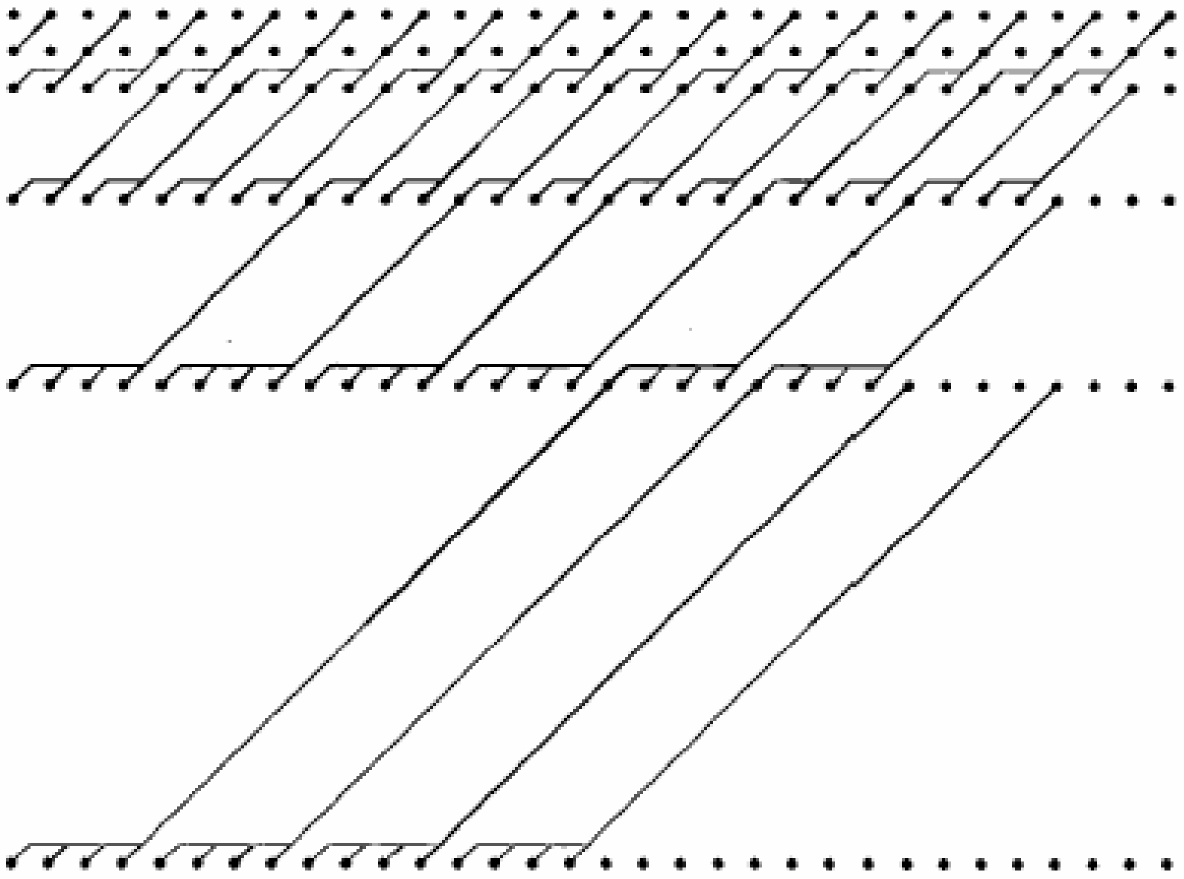
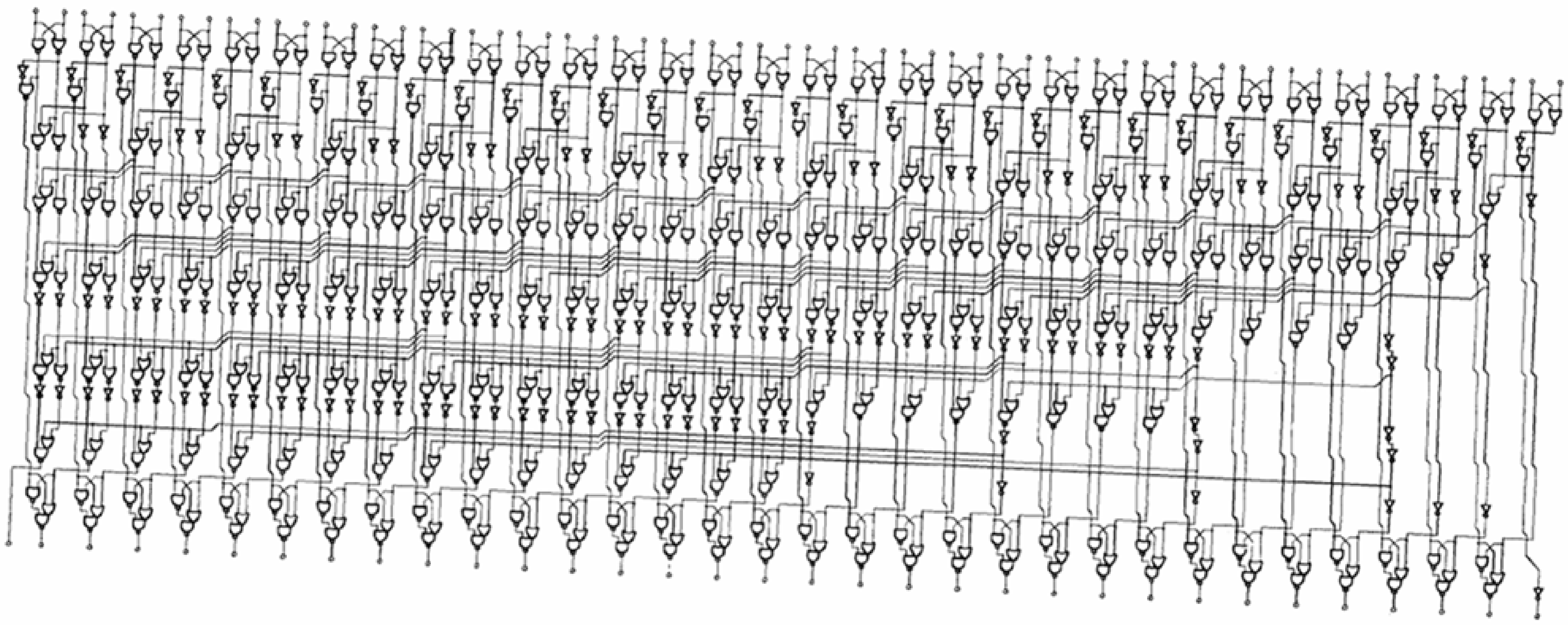


Figure 5: 32b graph [4,4,2,2,1] Knowles 1999

example of a new 32-bit adder [4,4,2,2,1]

Parallel Prefix Adders: variety of possibilities

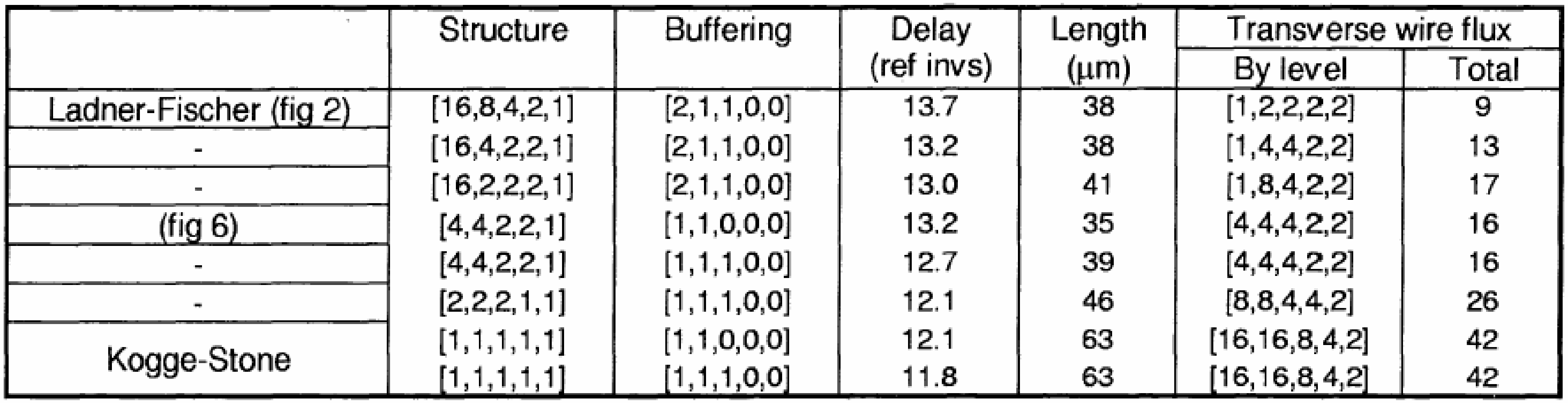
Knowles 1999



Example of a new 32-bit adder [4,4,2,2,1]

Parallel Prefix Adders: variety of possibilities

Knowles 1999



	Structure	Buffering	Delay (ref invs)	Length (µm)	Transverse wire flux	
					By level	Total
Ladner-Fischer (fig 2)	[16,8,4,2,1]	[2,1,1,0,0]	13.7	38	[1,2,2,2,2]	9
-	[16,4,2,2,1]	[2,1,1,0,0]	13.2	38	[1,4,4,2,2]	13
-	[16,2,2,2,1]	[2,1,1,0,0]	13.0	41	[1,8,4,2,2]	17
(fig 6)	[4,4,2,2,1]	[1,1,0,0,0]	13.2	35	[4,4,4,2,2]	16
-	[4,4,2,2,1]	[1,1,1,0,0]	12.7	39	[4,4,4,2,2]	16
-	[2,2,2,1,1]	[1,1,1,0,0]	12.1	46	[8,8,4,4,2]	26
Kogge-Stone	[1,1,1,1,1]	[1,1,0,0,0]	12.1	63	[16,16,8,4,2]	42
	[1,1,1,1,1]	[1,1,1,0,0]	11.8	63	[16,16,8,4,2]	42

- Delay is given in terms of FO4 inverter delay: w.c.
  (nominal case is 40-50% faster)
- K-S is the fastest
- K-S adders are wire limited (requiring 80% more area)
- The difference is less than 15% between examined schemes

Parallel Prefix Adders: variety of possibilities

Knowles 1999

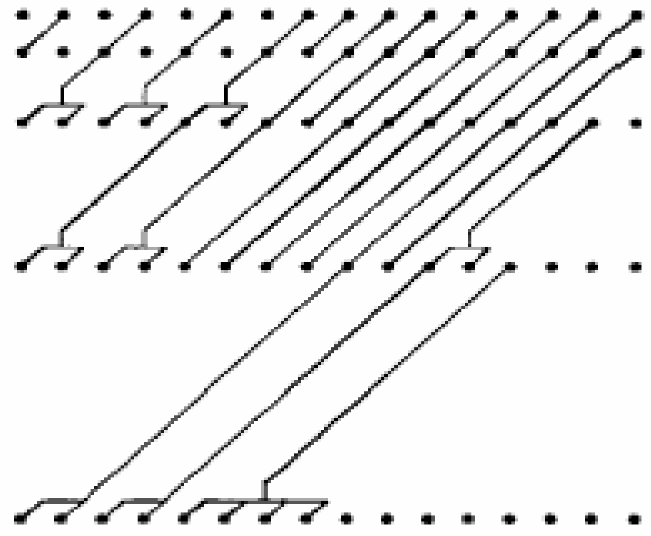


Figure 7: 16b hybrid graph

Conclusion

- Irregular, hybrid schmes are possible
- The speed-up of 15% is achieved at the cost of large wiring, hence area and power
- Circuits close in speed to K-S are available at significantly lower wiring cost

Possibilities for Further Research

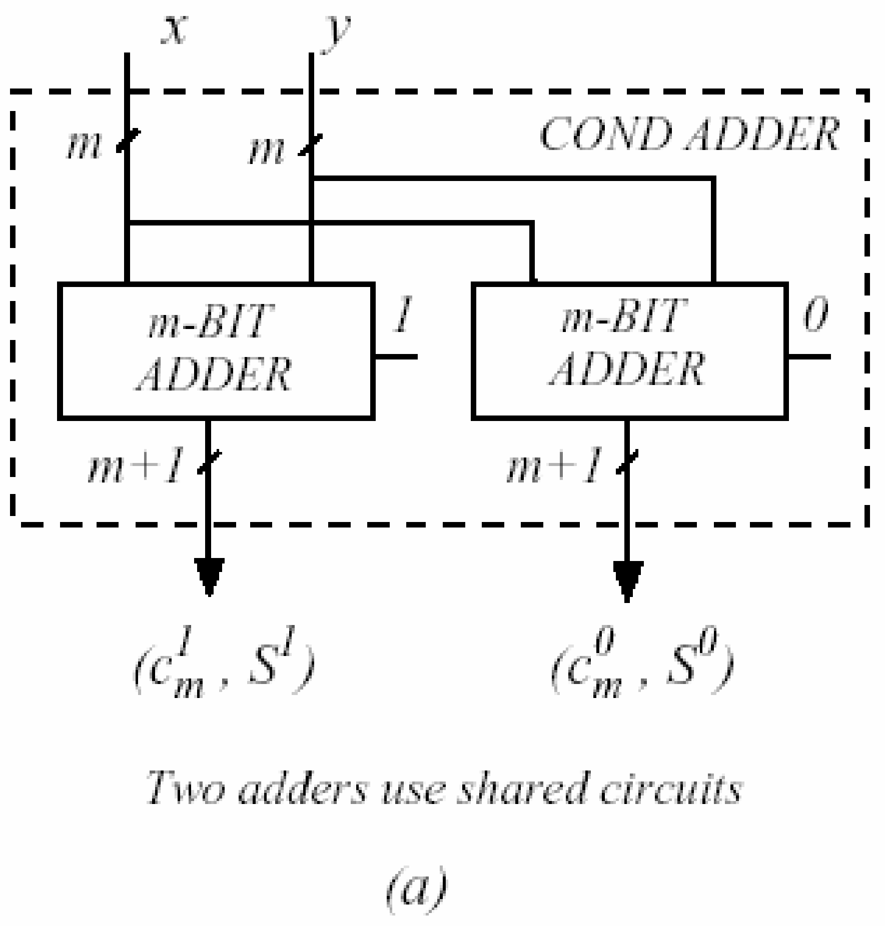
- The logical depth is important (Knowles was right)
- The fan-out is less important than fan-in (Knowles was wrong):
  - It is possible to examine a variety of topologies with restricted and varied fan-in.
- Driving strength and Logical Effort rules were overlooked and at least neglected:
  - It is possible to create number of topologies taking LE rules into account.
  - It is further possible to combine the rules with compound domino implementation taking advantage of two different rules governing “dynamic” and “static”.
- It is still possible to produce a better adder !

Other Types of Adders

Conditional Sum Adder

J. Sklansky, “Conditional-Sum Addition Logic”, IRE Transactions on Electronic Computers, EC-9, p.226-231, 1960.

Conditional Sum Adder



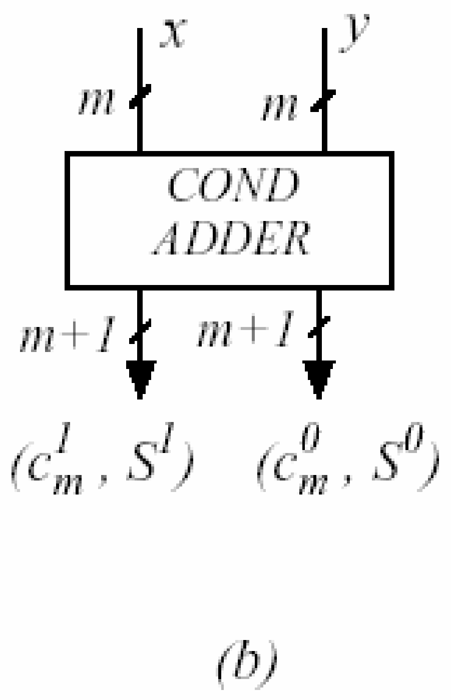


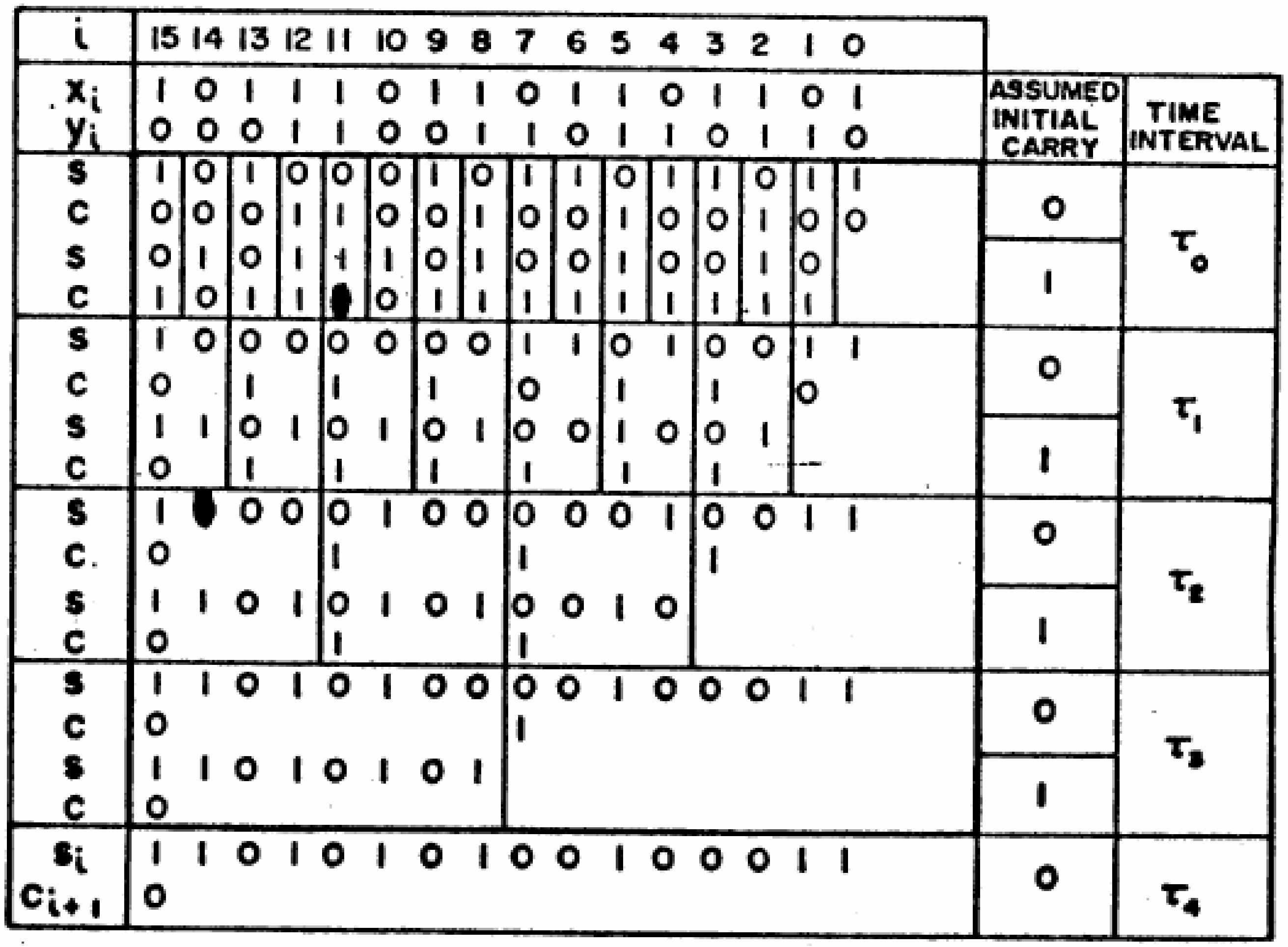
Figure 2.21: (a) Obtaining conditional outputs. (b) Combined conditi onal adder.

from: Ercegovac-Lang

Conditional Sum Adder

$$x = 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1$$

$$y = 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0.$$



i	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	ASSUMED INITIAL CARRY	TIME INTERVAL
x_i	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1		
y_i	0	0	0	1	1	0	0	1	1	0	1	1	0	1	1	0		
s	1	0	1	0	0	0	1	0	1	1	0	1	1	0	1	1	0	τ_0
c	0	0	0	1	1	0	0	1	0	0	1	0	0	1	0	0		
s	0	1	0	1	1	1	0	1	0	0	1	0	0	1	0		1	
c	1	0	1	1	[illegible]	0	1	1	1	1	1	1	1	1	1			
s	1	0	0	0	0	0	0	0	1	1	0	1	0	0	1	1	0	τ_1
c	0		1		1		1		0		1		1		0			
s	1	1	0	1	0	1	0	1	0	0	1	0	0	1			1	
c	0		1		1		1		1		1		1					
s	1	[illegible]	0	0	0	1	0	0	0	0	0	1	0	0	1	1	0	τ_2
c	0				1				1				1					
s	1	1	0	1	0	1	0	1	0	0	1	0					1	
c	0				1				1									
s	1	1	0	1	0	1	0	0	0	0	1	0	0	0	1	1	0	τ_3
c	0								1									
s	1	1	0	1	0	1	0	1									1	
c	0																	
s_i	1	1	0	1	0	1	0	1	0	0	1	0	0	0	1	1	0	τ_4
c_{i+1}	0																	

Fig. 1—Example of a conditional-sum addition.

Conditional Sum Adder

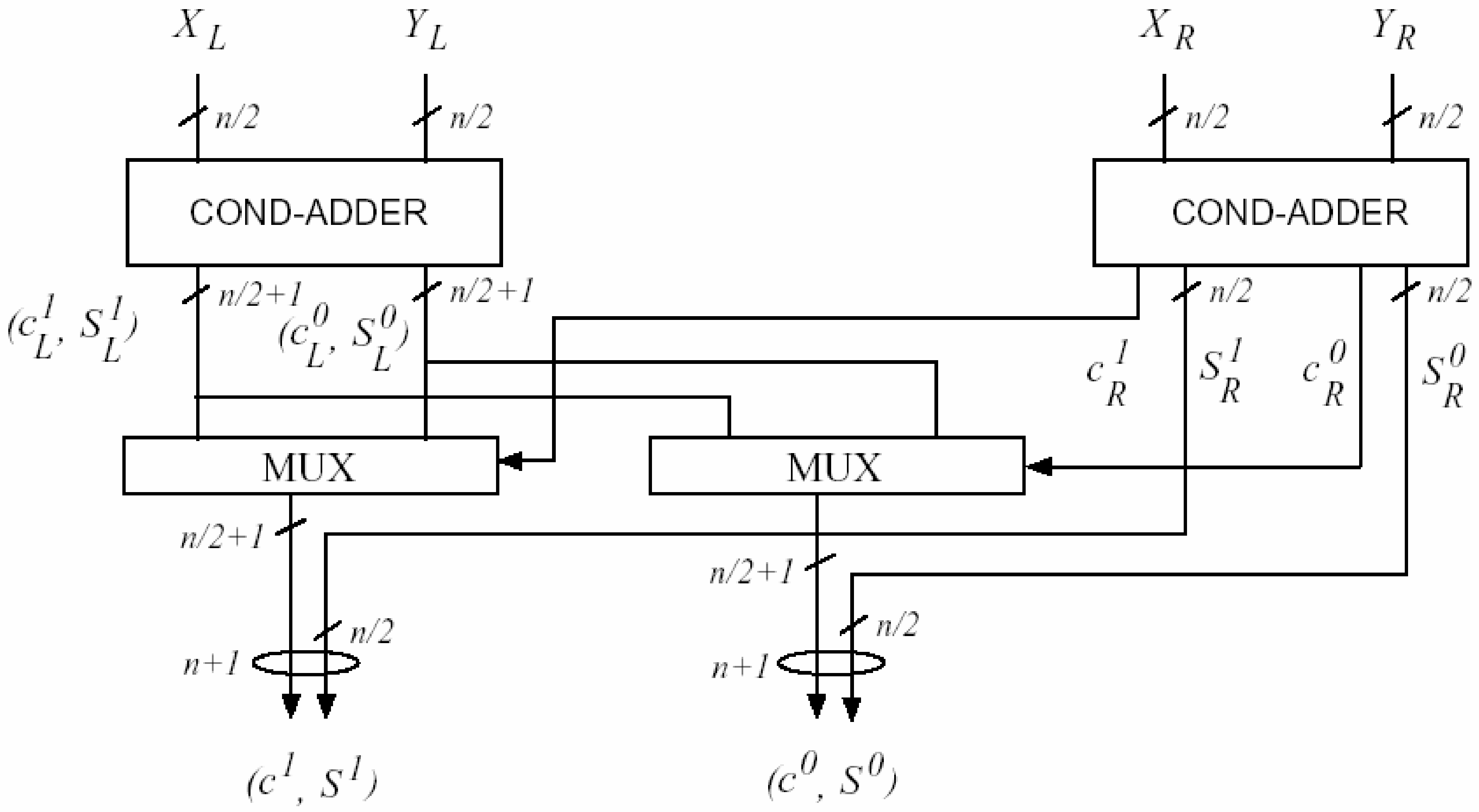


Figure 2.23: Doubling the number of bits of the conditional sum.

from: Ercegovac-Lang

Conditional Sum Adder

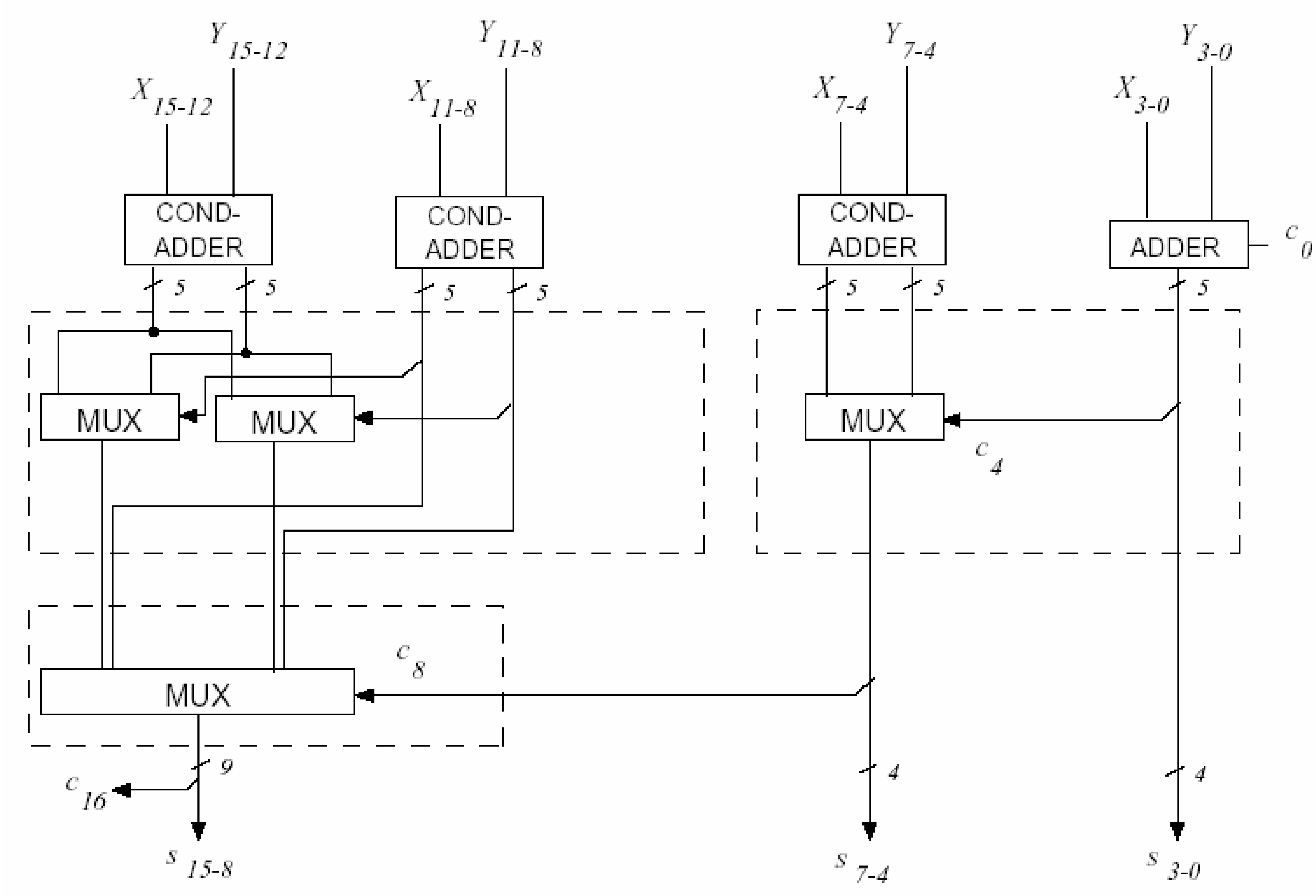
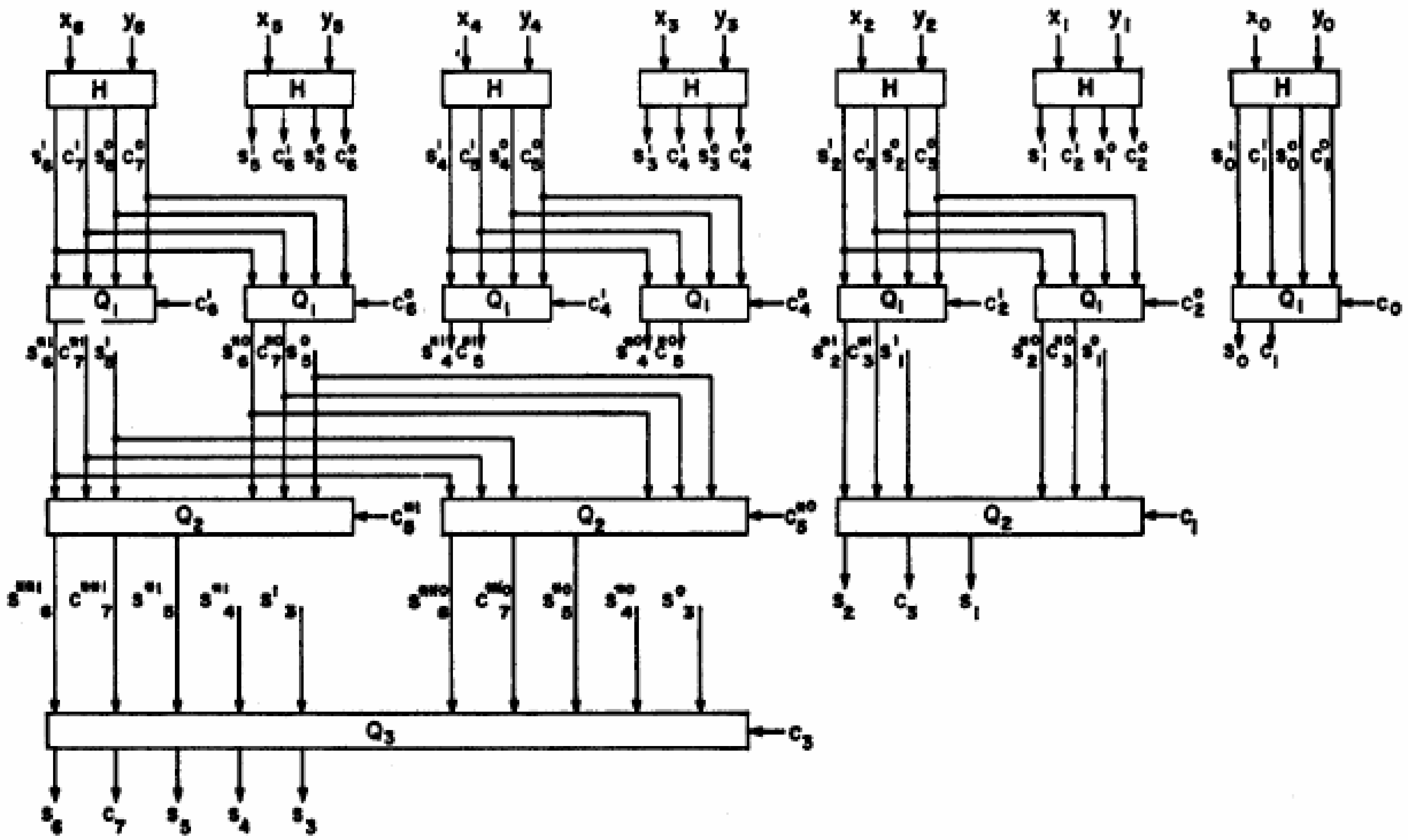


Figure 2.24: 16-bit conditional-sum adder ($m = 4$).

Conditional Sum Adder



H
Q1
Q2
Q3
x_6 y_6 x_5 y_5 x_4 y_4 x_3 y_3 x_2 y_2 x_1 y_1 x_0 y_0
c_0
c_1
c_3
s_6 c_7 s_5 s_4 s_3
s_2 c_3 s_1
s_0 c_1

(a)

Carry-Select Adder

O. J. Bedrij, “Carry-Select Adder”, IRE Transactions on Electronic Computers, June 1962, p.340-34

Carry-Select Sum Adder

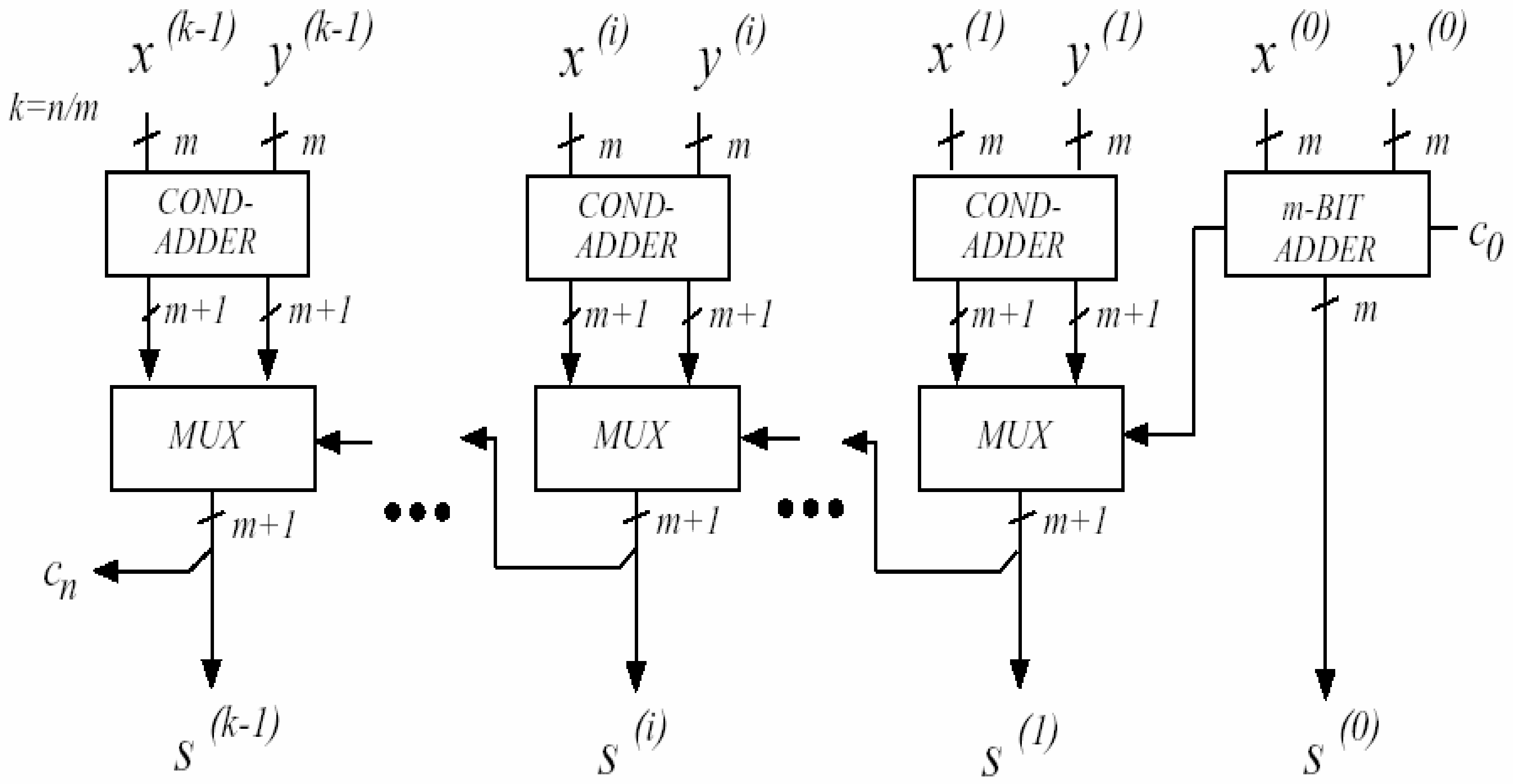


Figure 2.22: Carry-select adder.

Carry-Select Adder

Addition under assumption of C_{in}=0 and C_{in} =1.

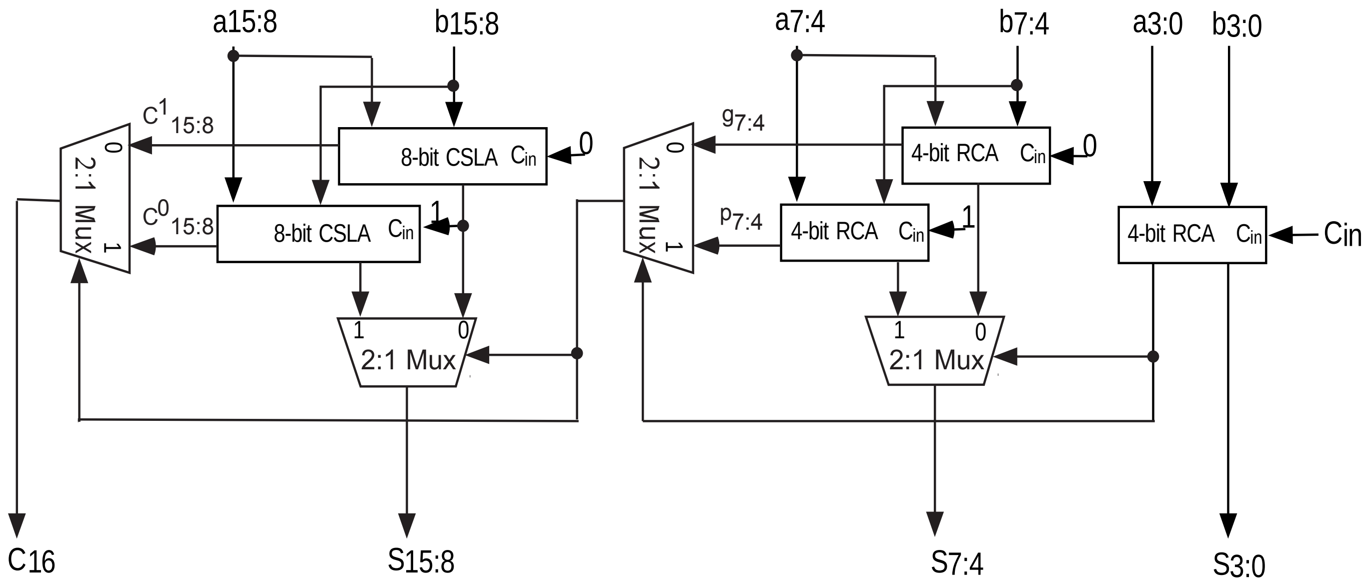
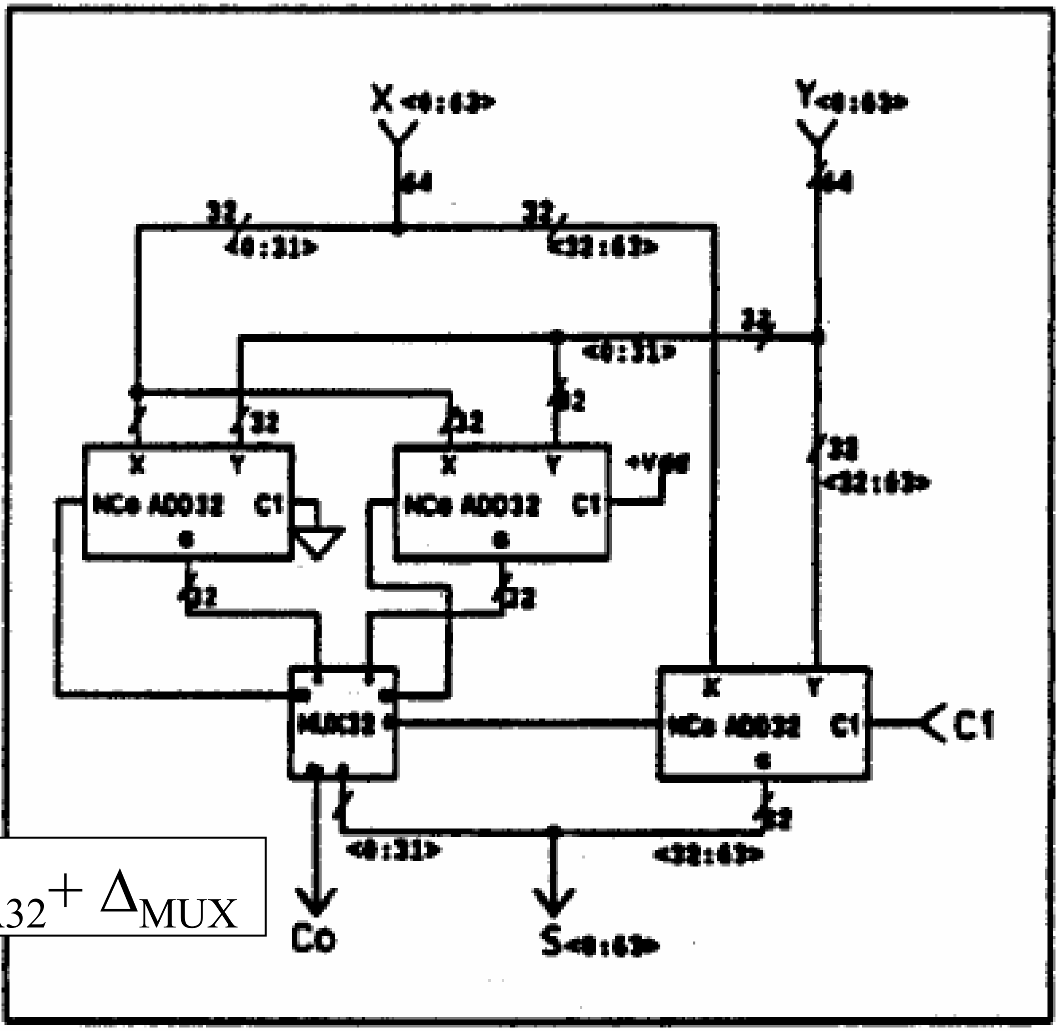


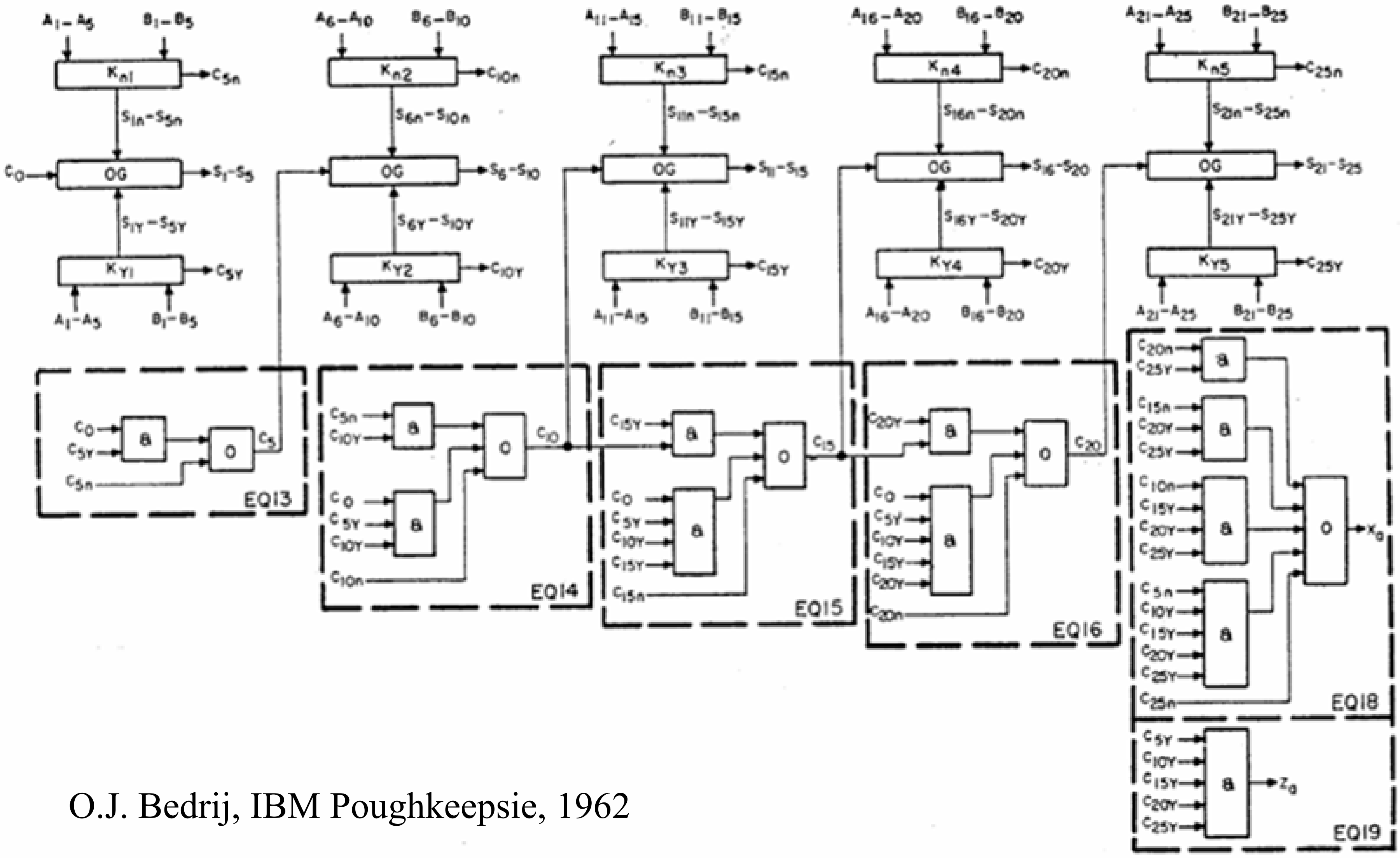
Fig. 11: 16-bit Carry-Select Adder

Carry Select Adder:
combining two 32-b VBAs in select mode



Delay $=\Delta_{VBA32}+\Delta_{MUX}$

Carry-Select Adder



O.J. Bedrij, IBM Poughkeepsie, 1962

Fig. 1—25-bit adder group.