8 MULTIOPERAND ADDITION

Chapter Goals

Learn methods for speeding up the addition of several numbers (needed for multiplication or inner-product)

Chapter Highlights

Wallace/Dadda trees, parallel counters
Modular multioperand addition

MULTIOPERAND ADDITION: TOPICS

Topics in This Chapter
8.1 Introduction to Multioperand addition
8.2 Carry-Save Adders
8.3 Wallace and Dadda Trees
8.4 Parallel Counters and Compressors
8.5 Modular Multioperand Adders

8.1 INTRODUCTION TO MULTIOPERAND ADDITION

Some applications of multioperand addition

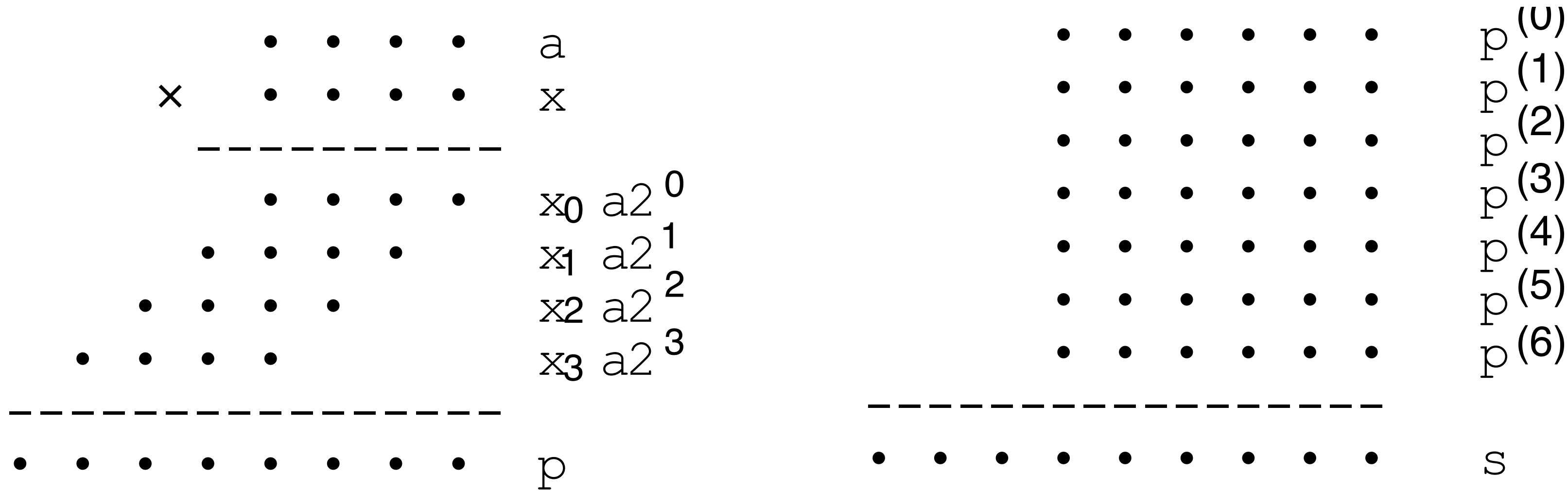


Fig. 8.1 Multioperand addition problems for multiplication or inner-product computation in dot notation.

PARALLEL IMPLEMENTATION AS TREE OF ADDERS

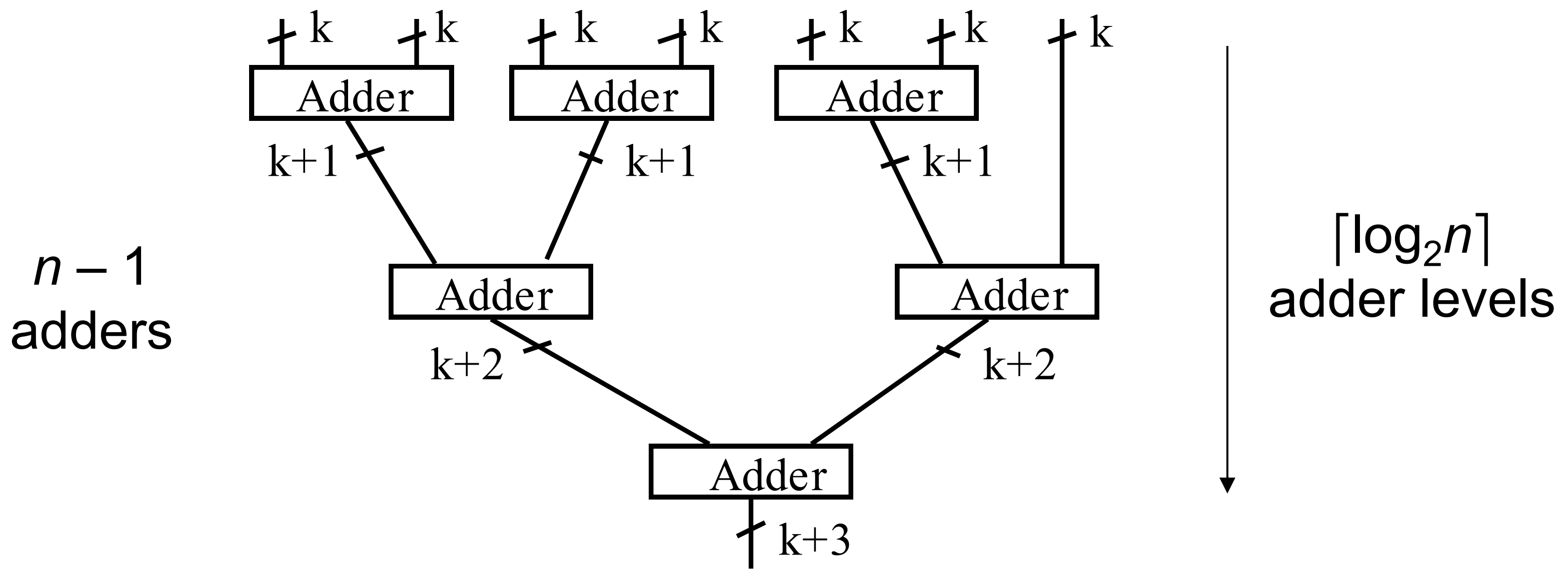
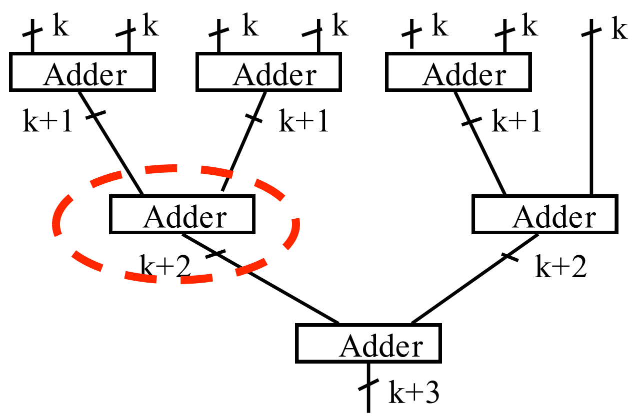


Fig. 8.4 Adding 7 numbers in a binary tree of adders.

$$T_{\text{tree-fast-multi-add}} = O(\log k + \log(k+1) + \ldots + \log(k + \lceil \log_2 n \rceil - 1))$$

$$T_{\text{tree-ripple-multi-add}} = O(k + \log n)$$

ELABORATION ON TREE OF RIPPLE-CARRY ADDERS



$T_{\text{tree-ripple-multi-add}} = O(k + \log n)$

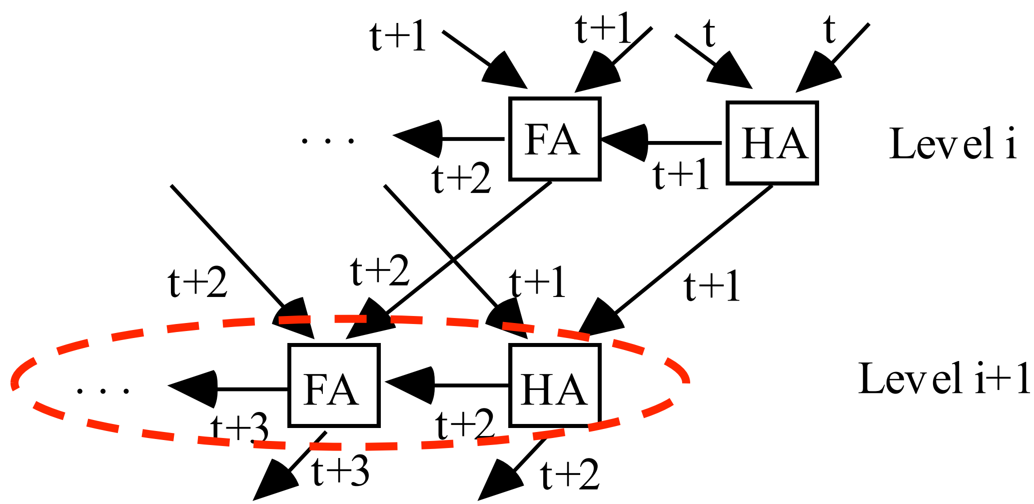


Fig. 8.5 Ripple-carry adders at levels *i* and *i* + 1 in the tree of adders used for multi-operand addition.

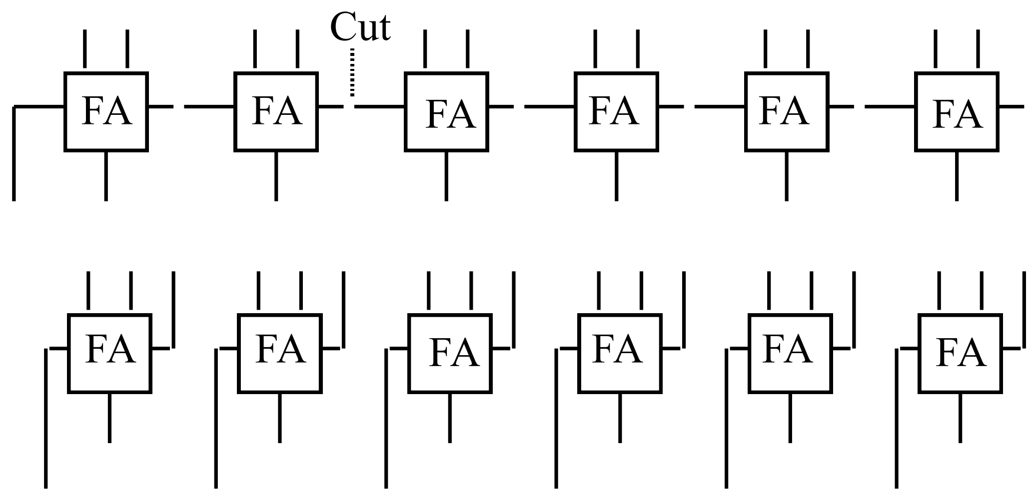
It is possible to accelerate this?

The absolute best latency that we can hope for is O(log *k* + log *n*)

We will see shortly that carry-save adders achieve this optimum time

8.2 CARRY-SAVE ADDERS

Fig. 8.6 A ripple-carry adder turns into a carry-save adder if the carries are saved (stored) rather than propagated.



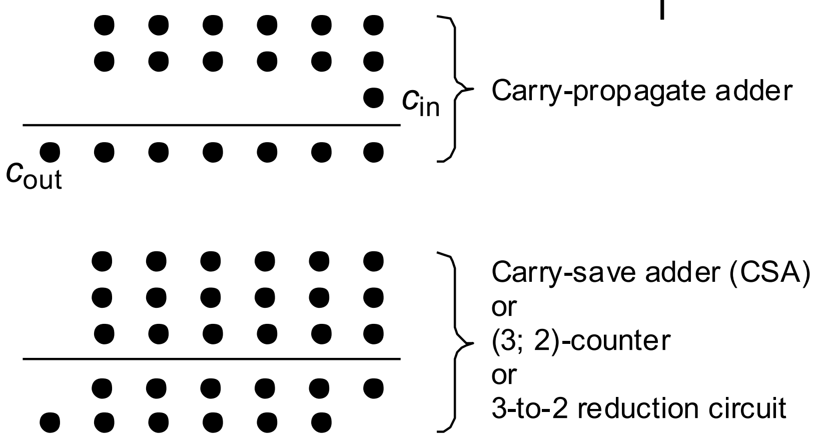


Fig. 8.7 Carry-propagate adder (CPA) and carry-save adder (CSA) functions in dot notation.

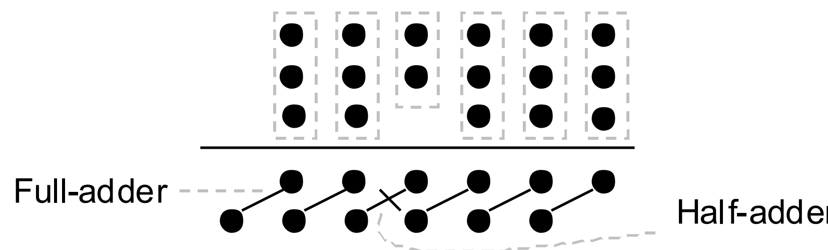


Fig. 8.8 Specifying full- and half-adder blocks, with their inputs and outputs, in dot notation.

MULTIOPERAND ADDITION USING CARRY-SAVE ADDERS

$$T_{\text{carry-save-multi-add}} = O(\text{tree height} + T_{\text{adder}})$$
$$= O(\log n + \log k)$$

$$C_{\text{carry-save-multi-add}} = (n-2)C_{\text{CSA}} + C_{\text{adder}}$$

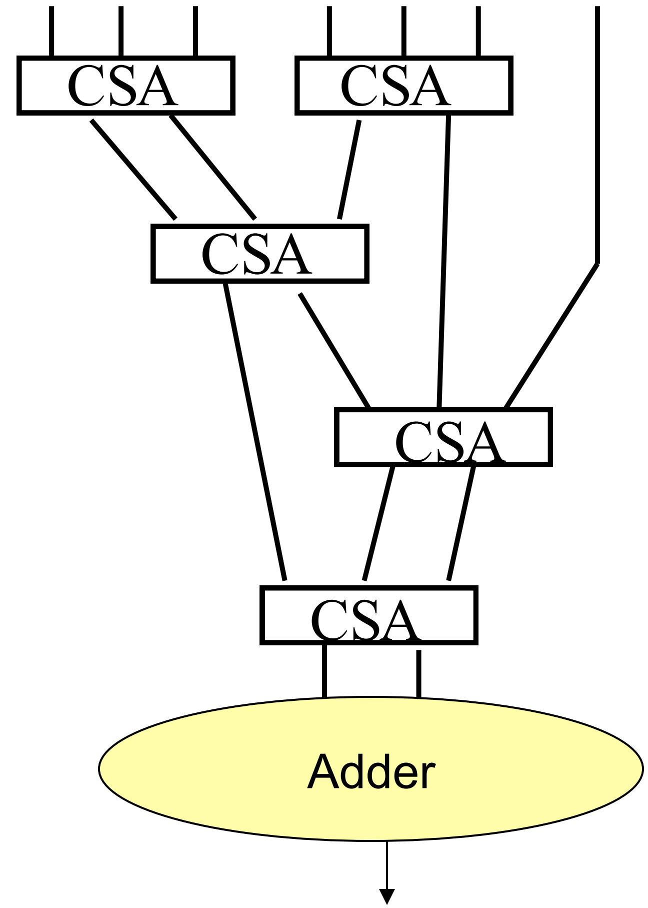
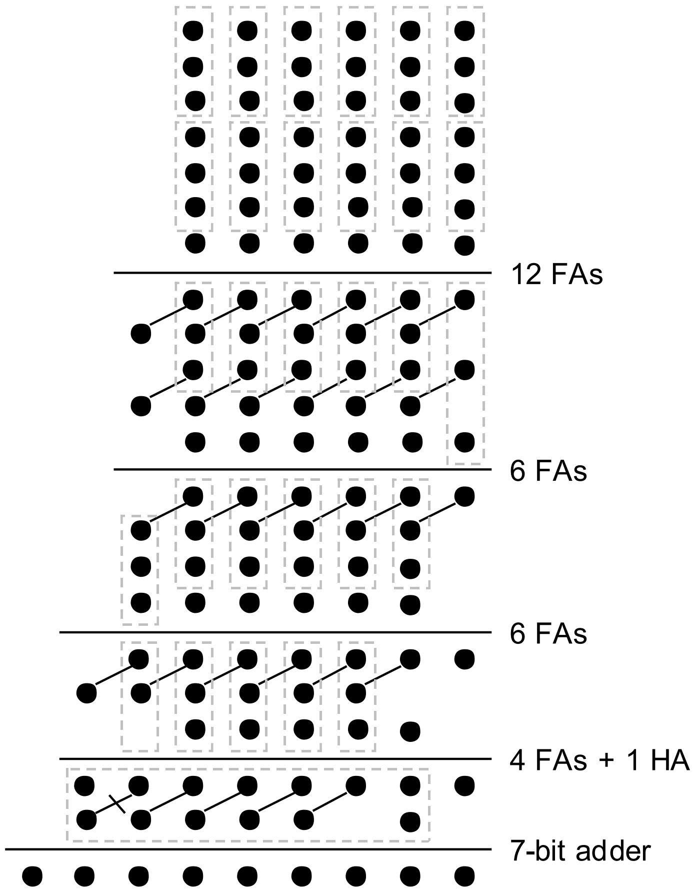


Fig. 8.9 Tree of carry-save adders reducing seven numbers to two.

EXAMPLE REDUCTION BY A CSA TREE



Total cost = 7-bit adder + 28 FAs + 1 HA

Fig. 8.10 Addition of seven 6-bit numbers in dot notation.

8	7	6	5	4	3	2	1	0	Bit position
			7	7	7	7	7	7	6×2 = 12 FAs
		2	5	5	5	5	5	3	6 FAs
		3	4	4	4	4	4	1	6 FAs
	1	2	3	3	3	3	2	1	4 FAs + 1 HA
	2	2	2	2	2	1	2	1	7-bit adder
--Carry-propagate adder--									
1	1	1	1	1	1	1	1	1	

Fig. 8.11 Representing a seven-operand addition in tabular form.

A full-adder compacts 3 dots into 2 (compression ratio of 1.5)

A half-adder rearranges 2 dots (no compression, but still useful)

PROBLEMAS

Problema 8.1. Compacte a informação das seguintes expressões numa matriz de informação, onde A, B, C e D são de 4 bits. Projete um compressor para reduzir a dois vetores a matriz de informação e, finalmente, some eles com um somador completo.

a) 33A+21B+387C+131D.

b) 65A+43B+135C+278D.

Problema 8.2. Implemente um somador de 255 bits paralelo em formato tabular usando contadores de {7;3} e somadores de 4 bits com *carry-in* {2, 2, 2, 3; 5}. .

8.3 WALLACE AND DADDA TREES

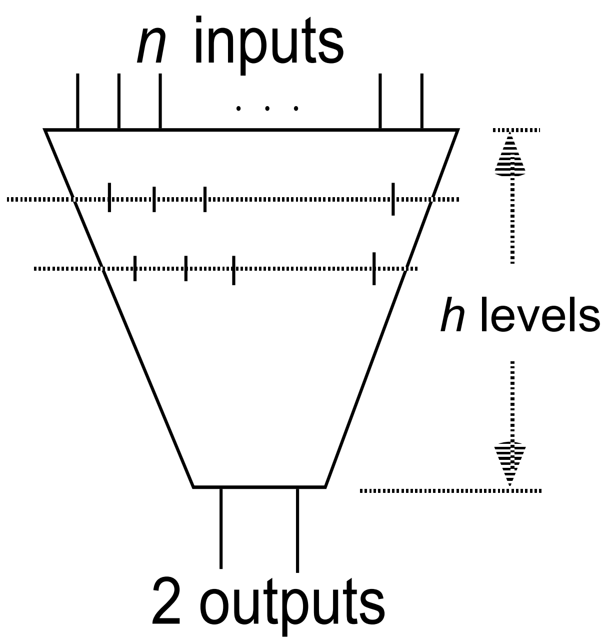
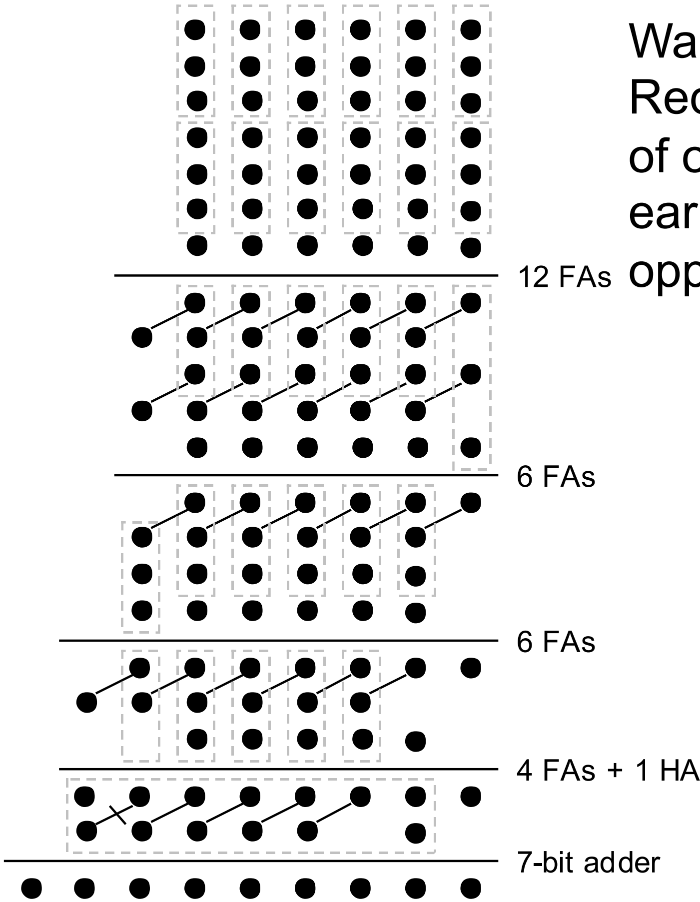


Table 8.1 The maximum number *n* of inputs for an *h*-level CSA tree

h	*n*	*h*	*n*	*h*	*n*
0	2	7	28	14	474
1	3	8	42	15	711
2	4	9	63	16	1066
3	6	10	94	17	1599
4	9	11	141	18	2398
5	13	12	211	19	3597
6	19	13	316	20	5395

n: Maximum number of inputs for *h* levels

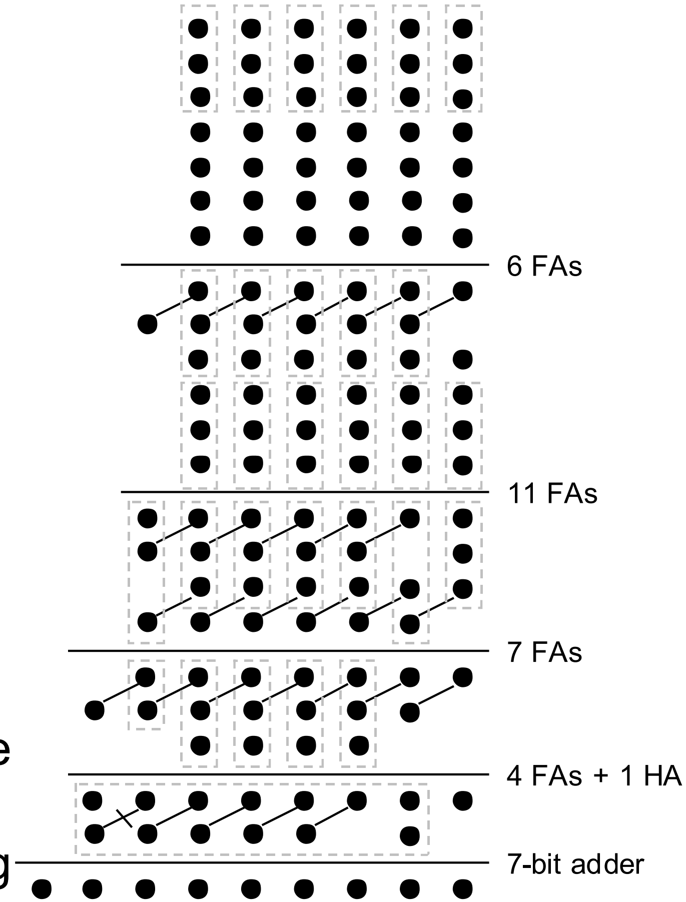
EXAMPLE WALLACE AND DADDA REDUCTION TREES



Total cost = 7-bit adder + 28 FAs + 1 HA

Wallace tree:
Reduce the number of operands at the earliest possible opportunity

Fig. 8.10 Addition of seven 6-bit numbers in dot notation.



Total cost = 7-bit adder + 28 FAs + 1 HA

Dadda tree:
Postpone the reduction to the extent possible without causing added delay

Fig. 8.14 Adding seven 6-bit numbers using Dadda's strategy.

A SMALL OPTIMIZATION IN REDUCTION TREES

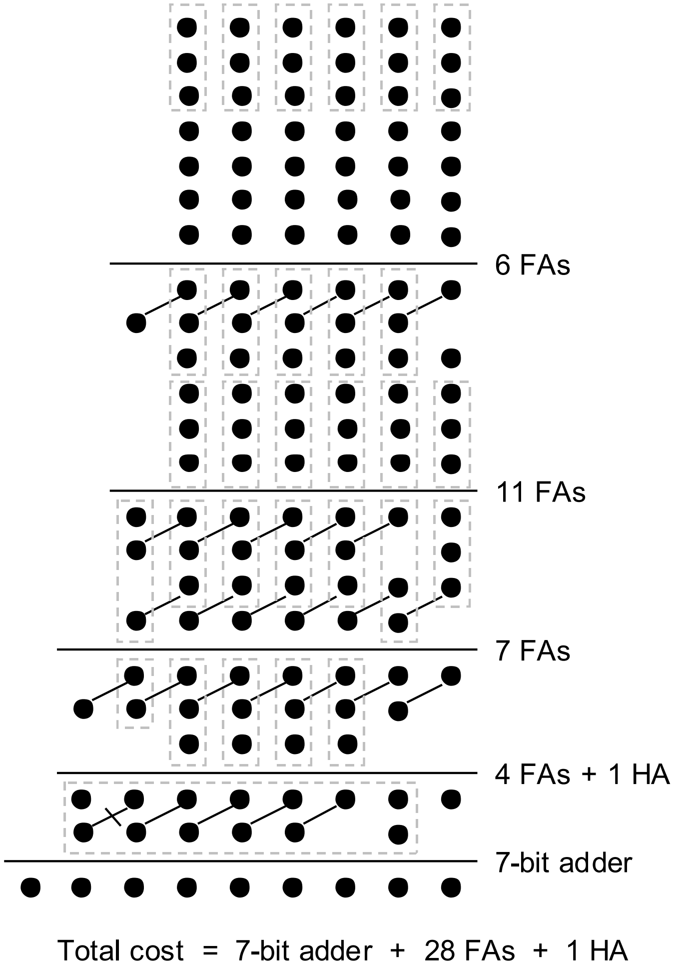
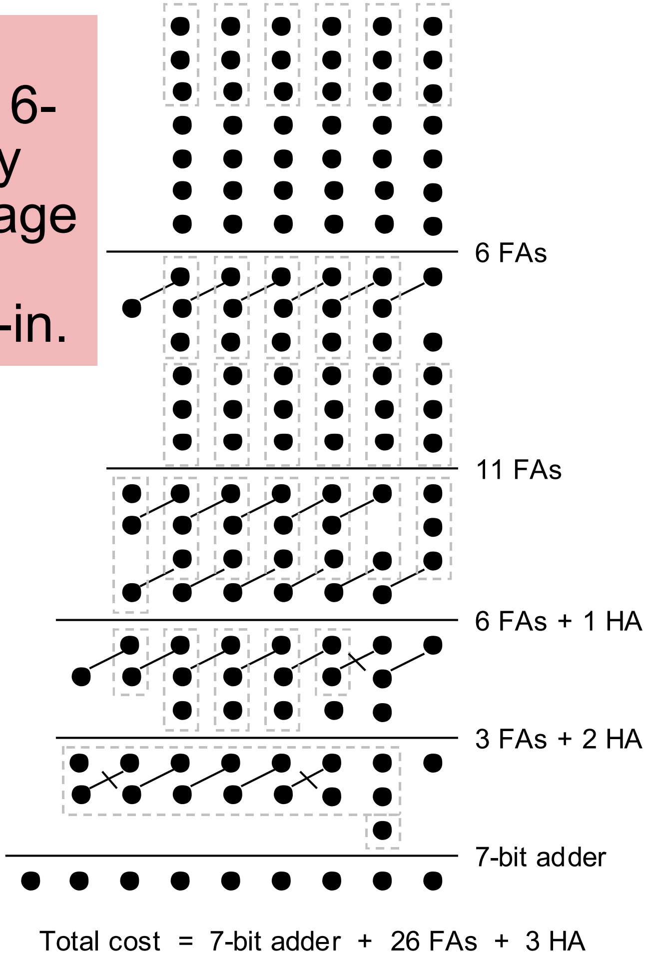


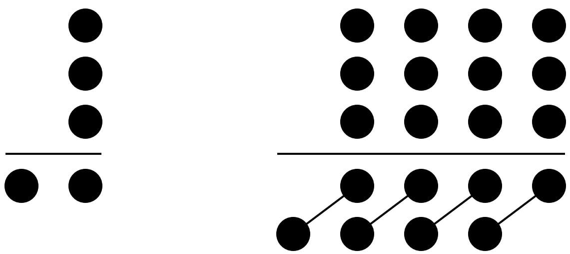
Fig. 8.14 Adding seven 6-bit numbers using Dadda's strategy.

Fig. 8.15
Adding seven 6-bit numbers by taking advantage of the final adder's carry-in.

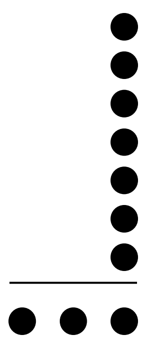


8.4 PARALLEL COUNTERS AND COMPRESSORS

1-bit full-adder = (3; 2)-counter



Circuit reducing 7 bits to their 3-bit sum = (7; 3)-counter



Circuit reducing n bits to their $\lceil \log_2(n + 1) \rceil$-bit sum = $(n; \lceil \log_2(n+1) \rceil)$-counter

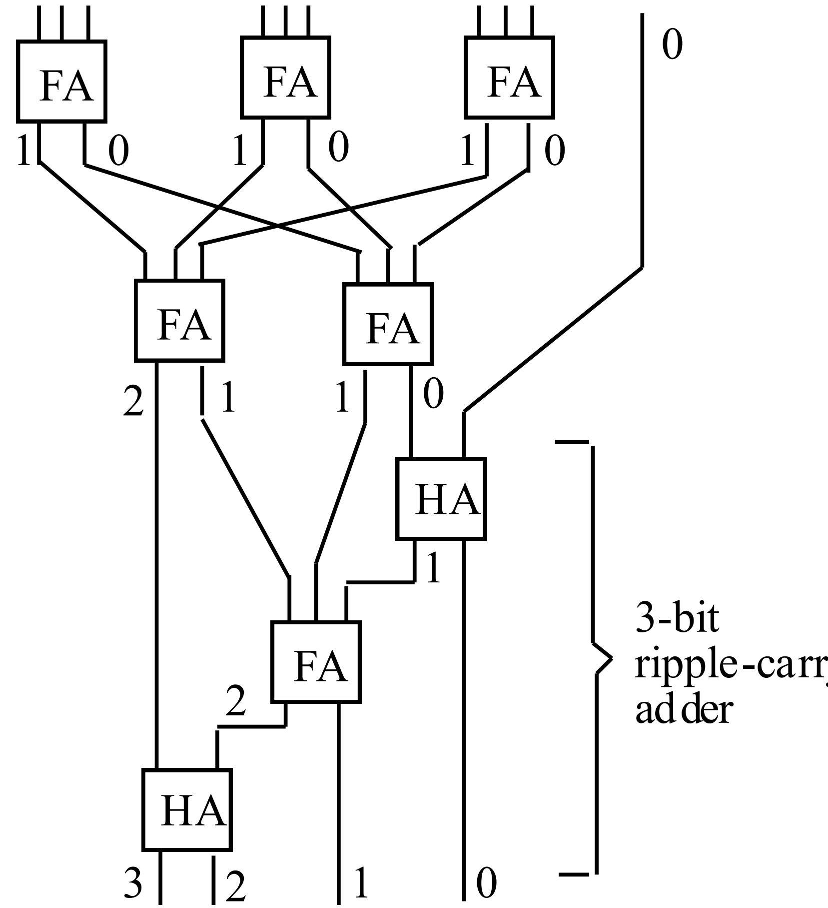


Fig. 8.16 A 10-input parallel counter also known as a (10; 4)-counter.

PROBLEMAS

Problema 8.3. Projete os diagrama de pontos e os circuitos usando CSAs e CPAs que fazem as seguintes compressões, identifique às quais correspondem com blocos conhecidos:

a) {2,2,3; 4};

b) {3; 2};

c) {1, 4, 3; 4};

d) {5, 5; 4};

e) {2, 2; 3}

f) {5; 3};

g) {7; 3};

h) {3; 1, 1};

i) {3, 3, 3; 3, 3}.

j) {4, 7; 4}

Obtenha o custo e caminho critico dos blocos considerando A_{FA} e T_{FA} como a área e atraso por *Full-Adder*, e $0{,}5\times A_{FA}$ e $0{,}5\times T_{FA}$, para o *Half-Adder*.

Problema 8.4. Usando os blocos obtidos no exercício anterior faça a redução das matrizes de informação do exercício 8.1 a dos vectores. Finalmente, some eles com um somador completo.

8.5 MODULAR MULTIOPERAND ADDERS

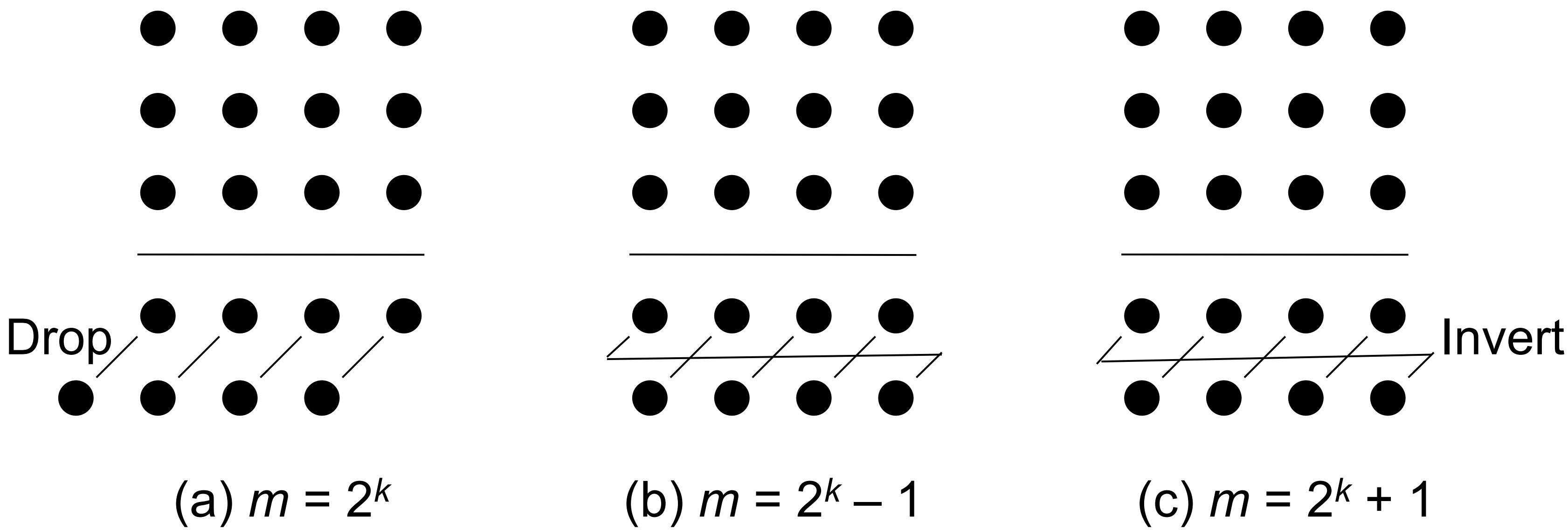


Fig. 8.20 Modular carry-save addition with special moduli.

PROBLEMAS

Problema 8.5. Refaça o exercício 8.1 usando RNS para modulo 63.

MODULAR REDUCTION WITH PSEUDORESIDUES

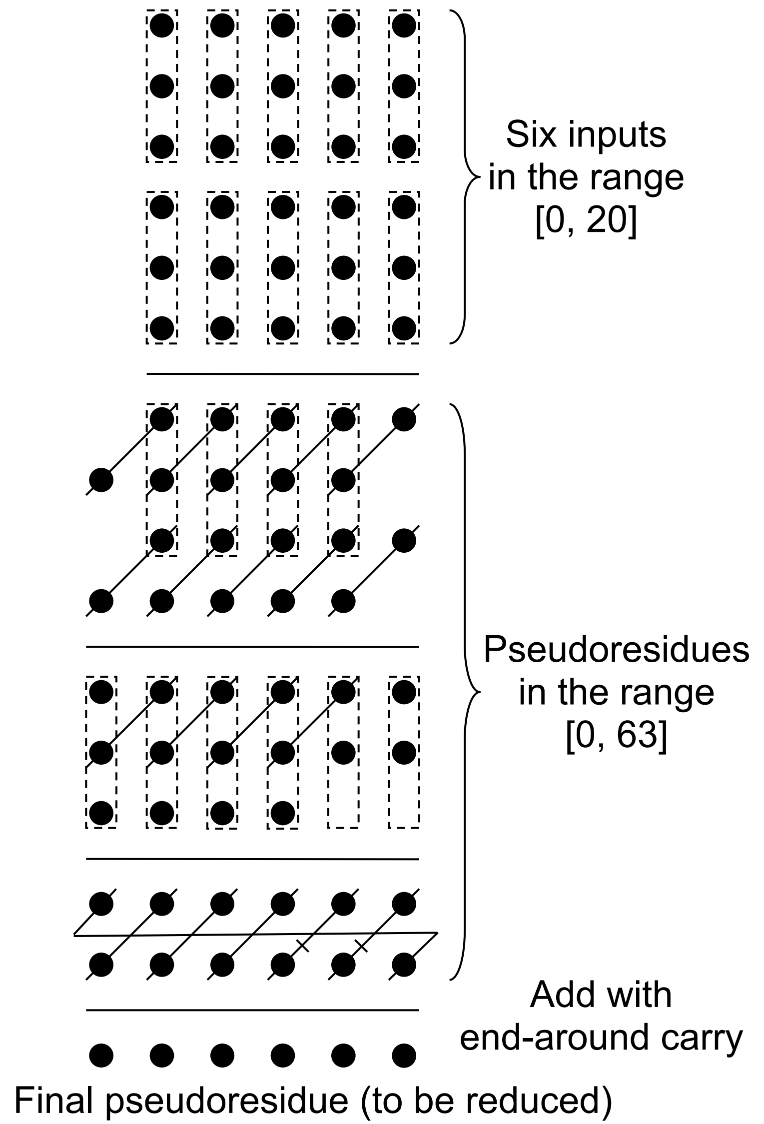


Fig. 8.21 Modulo-21 reduction of 6 numbers taking advantage of the fact that 64 = 1 mod 21 and using 6-bit pseudoresidues.

PROBLEMAS

Problema 8.6. Obtenha o caminho critico por operação de multiplicação ao conjunto modular M1={256,43,85}:

a) Usando os valores modulares dados.
b) Aplicando a ideia de redução modular usando pseudo-modulos.

Delay (ps) Modular Multipliers

# bits	2^n	2^n – 1	2^n+1	2n-k	2^n+k
5	960	1120	1480	2200	2600
7	1130	1360	1670	2840	3020
9	1320	1460	1750	3040	3320
11	1440	1670	1830	3120	3620
13	1590	1820	2010	3360	3580
15	1680	1840	2170	3460	3700
17	1770	2010	2320	3510	3770
19	1870	2200	2350	3760	3740
21	1940	2150	2420	3660	3830
23	1980	2240	2500	3850	3980
25	2090	2380	2590	4010	3980
27	2180	2530	2740	4140	4040
29	2280	2590	2750	4180	4200
31	2320	2530	2800	4340	4340
33	2340	2660	2810	4390	4260
35	2450	2690	2850	4390	4450
37	2470	2770	2960	4435	4393
39	2520	2780	3060	4491	4436
41	2520	2840	3040	4544	4477
43	2600	2900	3100	4600	4500