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

Neste capitulo vamos estudar a operação soma para mais de dois somandos.

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 Primeiramente vamos ver como somar varios somandos com FAs de forma generica.

8.2 Depois explicaremos os CSA que saõ a base dos somadores multioperando (basicamente consiste em usar os Fas para soma em paralelo e não em serie como ate agora com a transmisão do carry)

8.3 Varias soluções de interconexão dos CSAs (ou Fas) serão vistos de seguida

8.4 Os Fas comprimem de 3 entradas a duas, a seguir vamos ver que podemos fazer varios tipos de compressoes.

8.5 Finalmente estudiaremos somadores modulares.

8.1 INTRODUCTION TO MULTIOPERAND ADDITION

Some applications of multioperand addition

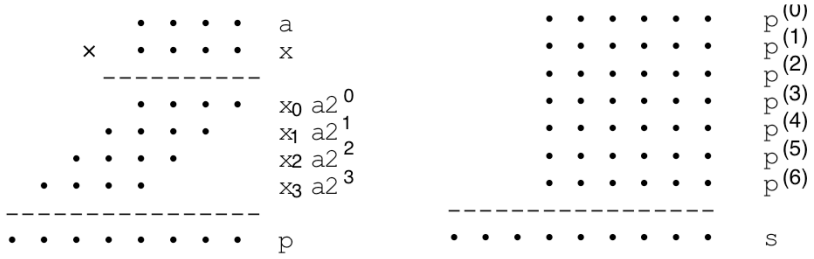


Fig. 8.1 Multioperand addition problems for multiplication or inner-product computation in dot notation.

A adição de multioperandos é necessária tanto em cálculos de multiplicação, quanto no cálculo da operação de produtos internos.
Por exemplo, ao multiplicar um multiplicando “a” por um multiplicador “x” de k dígitos, os k produtos parciais x a devem ser formados e então somados.

O cálculo de médias é uma outra aplicação que requer a adição de vários operandos.

PARALLEL IMPLEMENTATION AS TREE OF ADDERS

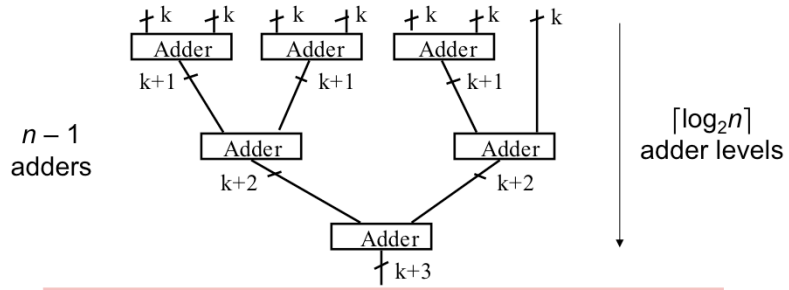


Fig. 8.4 Adding 7 numbers in a binary tree of adders.

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

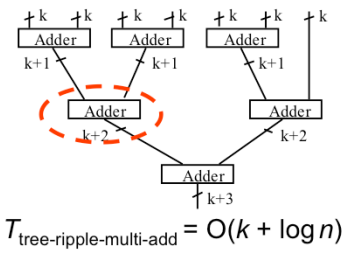
$T_{\text{tree-ripple-multi-add}} = O(k + \log n)$

Apr. 2012
Computer Arithmetic, Addition/Subtraction

SLIDE 4

A árvore binária de somadores de dois operandos precisa de n - 1 somadores e, portanto, é bastante custosa se construída com somadores rápidos. Por mais estranho que possa parecer, o uso de somadores de ripple-carry simples e lentos, pode ser a melhor escolha neste tipo de implementação. Se usarmos somadores de tempo logarítmico rápidos, a latência será como exemplificado no slide.

ELABORATION ON TREE OF RIPPLE-CARRY ADDERS



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

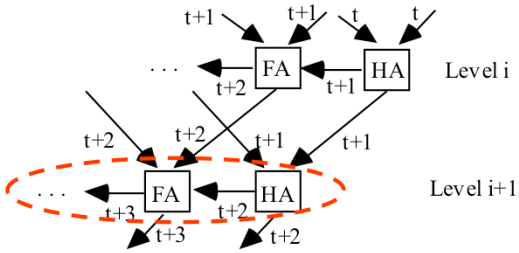


Fig. 8.5 Ripple-carry adders at levels *i* and *i* + 1 in the tree of adders used for multi-operand addition.

It is possible to accelerate this?

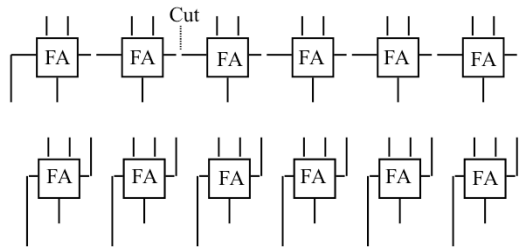
The absolute best latency that we can hope for is O(log *k* + log *n*)

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

Aqui vemos porque o atraso com somadores ripple carry é O (k + log n). Existem log2 n níveis na árvore. Um somador no (i + 1) ésimo nível não precisa esperar que ocorra a propagação do carry completo no nível i, mas pode iniciar sua adição com um atraso de somador completo (FA) após o nível i. Em outras palavras, a propagação do carry em cada nível fica 1 unidade de tempo atrás do nível anterior. Assim, precisamos permitir um tempo constante para todos, exceto o último nível, que precisa de tempo O (k + log n)

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.



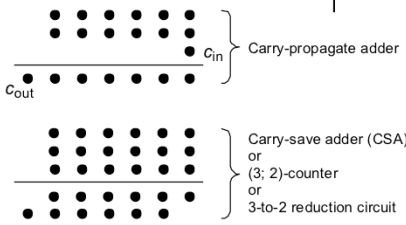


Fig. 8.7 Carry-propagate adder (CPA) and carry-save adder (CSA) functions in dot notation.

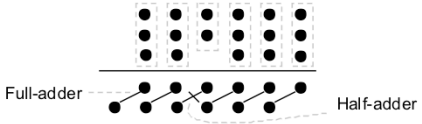


Fig. 8.8 Specifying full- and half-adder blocks, with their inputs and outputs, in dot notation.

Podemos ver uma linha de FAs como um mecanismo para reduzir três números para dois números. (2 Entradas + 1 Entrada de Carry)

Na notação de ponto, é utilizada para especificar mais precisamente como os vários pontos estão relacionados ou são obtidos. Vamos ilustrar quaisquer três pontos que formam as entradas de um FA em uma caixa tracejada e conectar a soma, transportando as saídas de um FA por uma linha diagonal (Fig. 8.8)

Ocasionalmente, apenas dois pontos são combinados para formar um bit de soma e um bit de transporte. Em seguida, os dois pontos são colocados em uma caixa tracejada e o uso de um meio somador (HA) é representado por uma linha cruzada na linha diagonal conectando suas saídas

A notação de ponto sugere outra maneira de ver a função de um CSA, ou seja, como conversor de um número radix-2 com o conjunto de dígitos [0, 3] (3 bits em uma posição) para um com o conjunto de dígitos [0, 2] (2 bits em uma posição).

MULTIOPERAND ADDITION USING CARRY-SAVE ADDERS

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

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

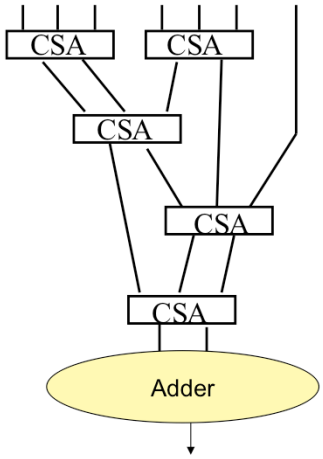


Fig. 8.9 Tree of carry-save adders reducing seven numbers to two.

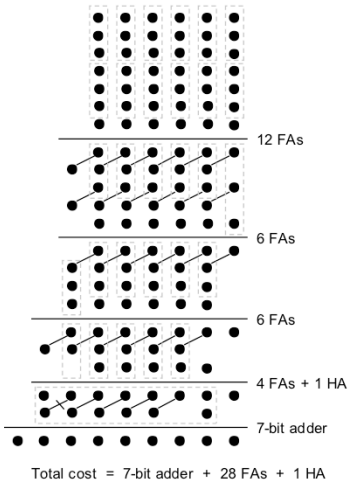
Uma árvore de CSAs (Fig. 8.9) pode reduzir n números binários para dois números com a mesma soma em um número de níveis O (log n)

Mostramos aqui como sería o esquema para reduzir 7 arrays de soma para 2.

Os CSAs necessários são de várias larguras de bits, mas geralmente, as larguras são próximas a k bits; o CPA tem largura no máximo k + log2 n

Ainda é demonstrado o atraso e e custo em area. Como vemos o atraso vai depender mais do CPA do que o CSA.

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)

Apr. 2012
Computer Arithmetic, Addition/Subtraction

SLIDE 8

Aqui temos um exemplo para adicionar sete números de 6 bits, mostrado na Fig. 8.10. Uma representação tabular mais compacta do mesmo processo é ilustrada na Fig. 8.11, onde as entradas representam o número de pontos restantes nas respectivas colunas ou posições de bits. Começamos na primeira linha com sete pontos em cada uma das posições de bit 0–5; esses pontos representam as sete entradas de 6 bits. Dois FAs são usados em cada coluna de 7 pontos, com cada FA convertendo três pontos em sua coluna para um ponto naquela coluna e um ponto na próxima coluna superior.

O que temos em seguida é a distribuição dos pontos mostrados na segunda linha da Fig. 8.11. Em seguida, um FA é usado em cada uma das posições de bit 0–5 contendo três pontos ou mais, e assim por diante, até que nenhuma coluna contenha mais de dois pontos. Quando chegamos apenas em dois vetores restanteso, um CPA é usado para reduzir os dois números resultantes à soma final de 9 bits representada por um único ponto em cada uma das posições de bit 0–8.

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) F_1=33A+21B+387C+131D.

b) F_2=65A+43B+135C+278D+8.

Problema 8.2. Faça a compressão de uma matriz de informação de 8×8 em formato tabular usando contadores de 7 bits, *Full-Adders* (FAs), *Half-Adders* (HAs) e apenas um somador de 8 bits com *carry-in* {2, 2, 2, 2, 2, 2, 2, 3; 9}.

SLIDE 9

Gabarito no Moodle

8.3 WALLACE AND DADDA TREES

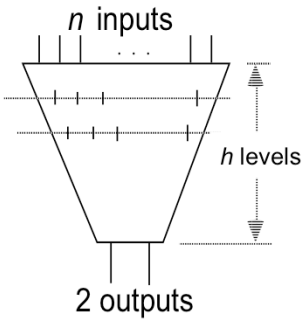


Table 8.1 The maximum number *n* of inputs for an *h*-level CSA tree

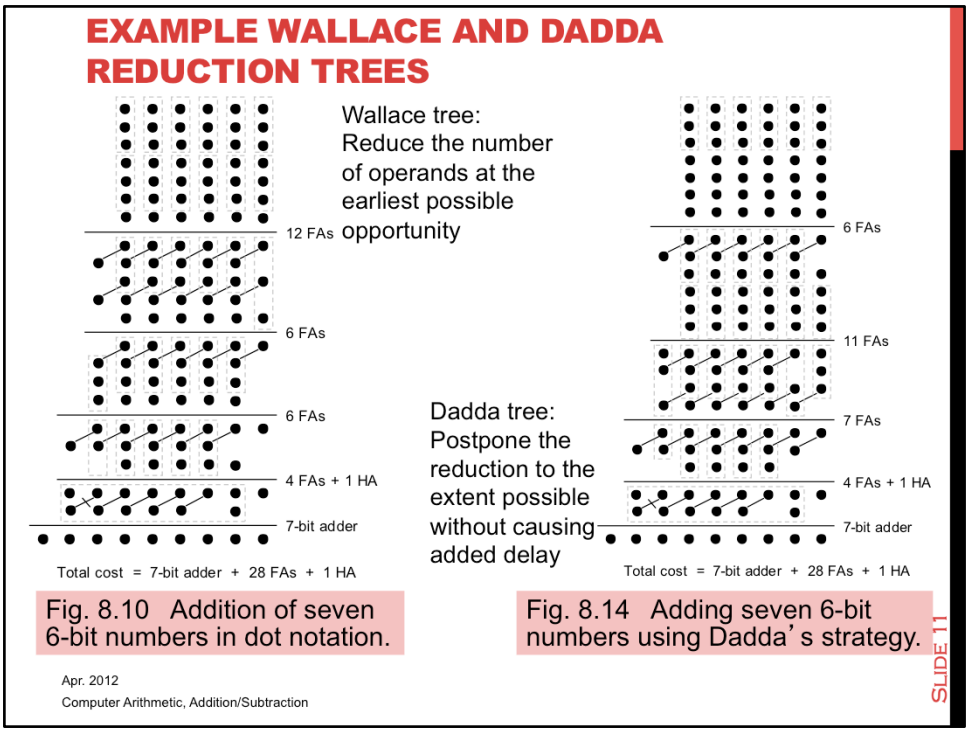
h	*n*	*h*	*n*	*h*	*n*
0	2	7	28	14	474
1	3	8	42	15	711
2	4	9	63	16	1066
3	6	10	94	17	1599
4	9	11	141	18	2398
5	13	12	211	19	3597
6	19	13	316	20	5395

n: Maximum number of inputs for *h* levels

Há outras abrodagens para conseguir a compresão eficiente.
Nas árvores de Wallace, reduzimos o número de operandos na primeira oportunidade. Em outras palavras, se houver m pontos em uma coluna, aplicamos imediatamente [m / 3] FAs a essa coluna. Isso tende a minimizar o atraso geral, tornando o CPA final o mais curto possível

No entanto, o atraso de um somador rápido geralmente não é uma função que depende apenas da largura da palavra binária. Um somador carry lookahead, por exemplo, pode ter o mesmo atraso para larguras de palavra de 17 a 32 bits.

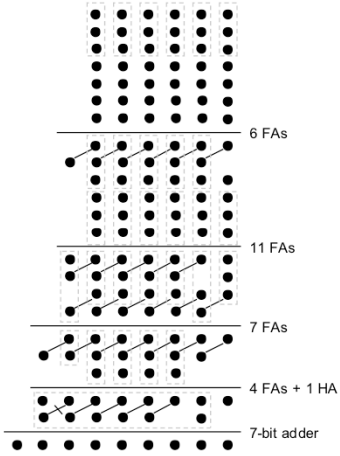
Já nas árvores Dadda, reduzimos o número de operandos para o próximo valor inferior de n (h) na Tabela 8.1 usando o menor número de FAs e HAs possível. A justificativa é que sete, oito ou nove operandos, digamos, requerem quatro níveis de CSA; portanto, não há sentido em reduzir o número de operandos abaixo do próximo valor n (h) inferior na tabela, uma vez que isso não levaria a uma árvore mais rápida.



Aqui vemos as duas estrategias de soma

Na solução de Dadda, começamos com sete linhas de pontos, portanto, nossa primeira tarefa é reduzir o número de linhas para o próximo valor n (h) inferior. Isso pode ser feito usando 6 FAs; em seguida, procuraremos reduzir quatro linhas, levando ao uso de 11 FAs e assim por diante. Neste exemplo particular, as abordagens de Wallace e Dadda resultam no mesmo número de FAs e HAs e a mesma largura para o CPA. Novamente, a largura do CPA poderia ter sido reduzida para 6 bits usando um HA extra na posição 1 do bit.

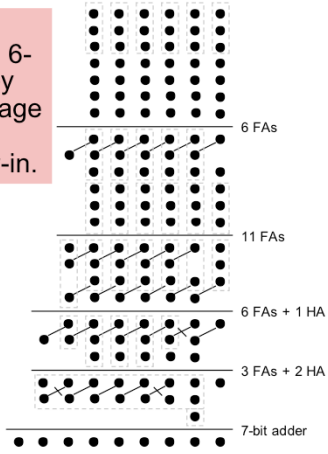
A SMALL OPTIMIZATION IN REDUCTION TREES



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

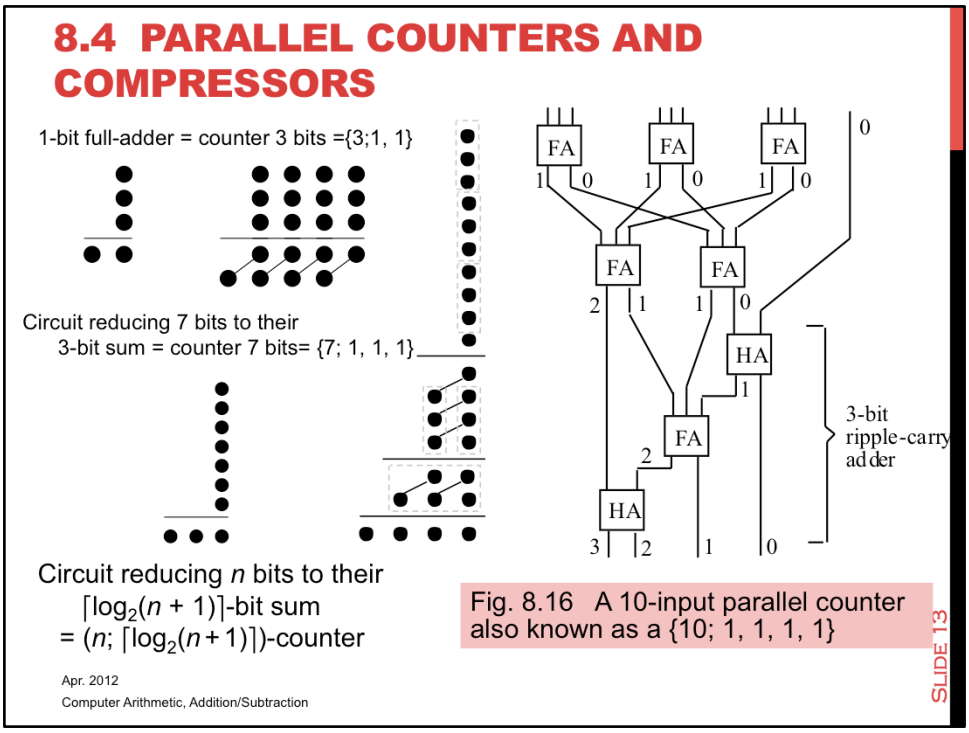
Fig. 8.14 Adding seven 6-bit numbers using Dadda's strategy.

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



Total cost = 7-bit adder + 26 FAs + 3 HA

Uma vez que um CPA tem um sinal de carry que pode ser usado para acomodar um dos pontos, às vezes é possível reduzir a complexidade da árvore CSA deixando três pontos na posição menos significativa do somador



Um FA de 1 bit é às vezes chamado de contador (3; 2), o que significa que conta o número de uns entre seus 3 bits de entrada e representa o resultado como um número de 2 bits. Esse tipo de circuito também é conhecido como um contador paralelo de entrada n

Um contador paralelo de 10 entradas, ou um contador (10; 4) , é representado na Fig. 8.16. Tambem vemos sua representação em termos de notação de ponto e diagrama de circuito com FAs e HAs. Uma linha desses (10; 4) contadores, um por posição de bit, pode reduzir um conjunto de 10 números binários a 4 números binários. A representação de notação de pontos dessa redução é semelhante à dos contadores (3; 2), exceto que cada linha diagonal conectando as saídas de um contador (10; 4) passará por quatro pontos.

Um contador (7; 3) pode ser projetado de forma semelhante.

PROBLEMAS

Problema 8.3. Projete os diagrama de pontos e os circuitos usando *Full-Adders* (FAs) e *Half-Adders* (HAs) que fazem as seguintes compressões, identifique às quais correspondem com blocos conhecidos:

a) {2,2,3; 1, 1, 1, 1};

b) {3; 1, 1};

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

d) {5, 5; 1, 1, 1, 1};

e) {2, 2; 1, 1, 1}

f) {5; 1, 1, 1};

g) {7; 1, 1, 1};

h) {3, 3, 3; 1, 2, 2, 1}.

i) {4, 7; 1, 1, 1, 1}

j) {2, 5; 1, 2, 1}

k) {5; 2, 1}

Obtenha o custo e caminho critico dos blocos considerando *AFA* e *TFA* como a área e atraso por *Full-Adder*, e *0,5×AFA* e *0,5×TFA*, 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.

SLIDE 14

Gabarito no Moodle

8.5 MODULAR MULTIOPERAND ADDERS

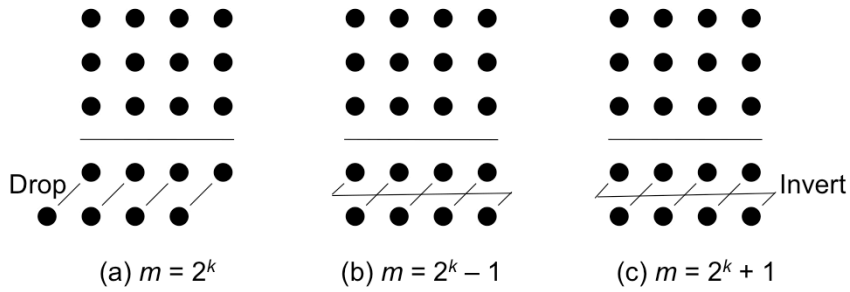


Fig. 8.20 Modular carry-save addition with special moduli.

Aqui são mostrados como seríam as abordagens para fazer a soma de 3 operandos usando modulo 2^{k}, 2^{k}-1 e 2^{k}-1.

Como no caso da adição de dois operandos, os três módulos especiais (2^k), (2 ^ k - 1) e (2 ^ k + 1) são mais fáceis de lidar. Para m = 2 ^ k, simplesmente eliminamos qualquer bit produzido na coluna k. Essa simplificação é ilustrada na Fig. 8.20a.

Para m = 2^k - 1, um bit gerado na posição k é reinserido na posição 0, conforme mostrado na Fig. 8.20b. Dado o slot vazio disponível na posição 0, este “end-around carry” não leva a um aumento na latência.

Para m = 2^k + 1, podemos reinserir o bit, de forma similar ao m = 2^k - 1, entretanto o invertemos.

PROBLEMAS

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

Gabarito no Moodle

MODULAR REDUCTION WITH PSEUDORESIDUES

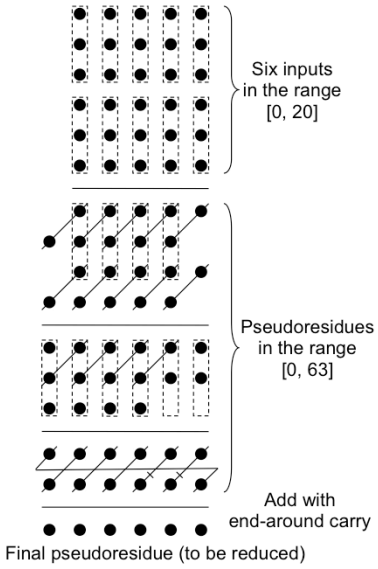


Fig. 8.21 Modulo-21 reduction of 6 numbers taking advantage of the fact that 64 = 1 mod 21 and using 6-bit pseudoresidues.

Apr. 2012
Computer Arithmetic, Addition/Subtraction

SLIDE 17

Para outros módulos, em geral, precisamos de esquemas de adição de multioperandos que são mais elaborados e complexos do que o transporte final (invertido) que realizamos no exemplo anterior. Muitas técnicas foram desenvolvidas para valores específicos de m. Por exemplo, se m é tal que 2^h = 1 mod m para um valor bastante pequeno de h, pode-se realizar a redução da árvore com pseudo-resíduos de h-bit.

Para aplicar este método à adição mod-21 de um conjunto de n inteiros de entrada no intervalo [0, 20], podemos usar qualquer esquema de redução de árvore, mantendo todos os valores intermediários no intervalo [0, 63]. Os bits gerados na coluna 6 são então realimentados para a coluna 0 da mesma maneira que o “end around carry" usado para a redução do módulo 63, dado que 64 = 1 mod 21. Uma vez que todos os operandos foram combinados em dois valores de 6 bits, os últimos são adicionados com "end-around carry" e a soma final de 6 bits é reduzida no módulo 21.

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

SLIDE 18

Gabarito no Moodle