

## 9 BASIC MULTIPLICATION SCHEMES

### Chapter Goals

Study shift/add or bit-at-a-time multipliers and set the stage for faster methods and variations to be covered in Chapters 10-12

### Chapter Highlights

Multiplication= multioperand addition  
Hardware, firmware, software algorithms  
Multiplying 2's-complement numbers  
The special case of one constant operand

Apr. 2012  
Computer Arithmetic, Addition/Subtraction

SLIDE 1

Veremos neste capítulo esquemas de básicos multiplicação.

## BASIC MULTIPLICATION SCHEMES: TOPICS

### Topics in This Chapter

9.1 Shift/Add Multiplication Algorithms

9.3 Basic Hardware Multipliers

9.4 Multiplication of Signed Numbers

9.5 Multiplication by Constants

Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 2

Os tópicos apresentados serão: descrição básica de hardware associada aos multiplicadores, i.e. a implementação dos multiplicadores simples. Seguiremos com alguns exemplos de multiplicadores de números positivos/negativos e multiplicação por constantes.

## 9.1 SHIFT/ADD MULTIPLICATION ALGORITHMS

Notation for our discussion of multiplication algorithms:

$a$	Multiplicand	$a_{k-1}a_{k-2} \dots a_1a_0$
$x$	Multiplier	$x_{k-1}x_{k-2} \dots x_1x_0$
$p$	Product ( $a \times x$ )	$p_{2k-1}p_{2k-2} \dots p_3p_2p_1p_0$

Initially, we assume unsigned operands

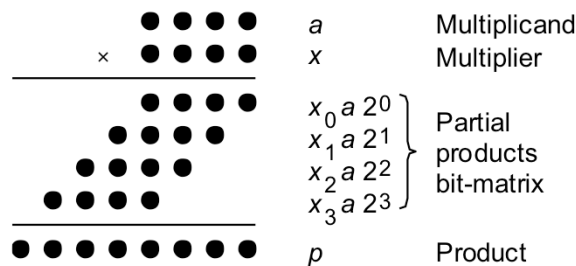


Fig. 9.1 Multiplication of two 4-bit unsigned binary numbers in dot notation.

Apr. 2012  
Computer Arithmetic, Multiplication

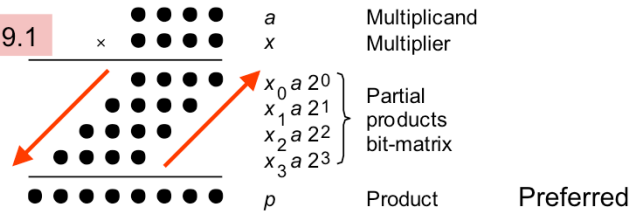
SLIDE 3

A Figura 9.1 mostra a multiplicação de dois números binários sem sinal de 4 bits em notação de ponto. Os dois números “a” e “x” são mostrados na parte superior. Cada uma das quatro linhas de pontos a seguir corresponde ao produto do multiplicando “a” e 1 bit do multiplicador “x”, com cada ponto representando o produto (AND lógico) de dois bits.

O problema da multiplicação binária se reduz basicamente à adição de um conjunto de números, cada um dos quais sendo 0 ou uma versão deslocada do multiplicando a.

## MULTIPLICATION RECURRENCE

Fig. 9.1



Multiplication with right shifts: top-to-bottom accumulation

$$p^{(j+1)} = (p^{(j)} + x_j a 2^j) 2^{-1} \quad \text{with } p^{(0)} = 0 \text{ and}$$

|—add—|  
|—shift right—|

Multiplication with left shifts: bottom-to-top accumulation

$$p^{(j+1)} = 2p^{(j)} + x_{k-j-1}a \quad \text{with } p^{(0)} = 0 \text{ and}$$

|shift|  
|—add—|

Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 4

Vejamos que cada número sucessivo a ser adicionado ao produto parcial cumulativo é deslocado em 1 bit em relação ao anterior

Duas versões de algoritmo para formação dessa matriz podem ser criadas, realizando um processo de cima para baixo ou de baixo para cima.

Como os deslocamentos à direita farão com que o primeiro produto parcial seja multiplicado por  $2^{-k}$  no momento em que terminarmos, pré-multiplicamos “a” por  $2^k$  para compensar o efeito dos deslocamentos à direita.

Esta operação de multiplicação-adição é bastante útil para muitas aplicações e é realizada essencialmente sem custo extra em comparação com a multiplicação simples, por adição.

## EXAMPLES OF BASIC MULTIPLICATION

Right-shift algorithm

a	1 0 1 0	←	1 0 1 0
x			1 0 1 1
$p^{(0)}$	0 0 0 0		
$+x_0a$	1 0 1 0		
$2p^{(1)}$	0 1 0 1 0		
$p^{(1)}$	0 1 0 1 0		
$+x_1a$	1 0 1 0		
$2p^{(2)}$	0 1 1 1 1 0		
$p^{(2)}$	0 1 1 1 1 0		
$+x_2a$	0 0 0 0		
$2p^{(3)}$	0 0 1 1 1 1 0		
$p^{(3)}$	0 0 1 1 1 1 0		
$+x_3a$	1 0 1 0		
$2p^{(4)}$	0 1 1 0 1 1 1 0		
$p^{(4)}$	0 1 1 0 1 1 1 0		

Fig. 9.2  
Examples  
of  
sequential  
multipli-  
cation with  
right and  
left shifts.

$$p^{(j+1)} = (p^{(j)} + x_j a 2^k) 2^{-1}$$

|—add—|  
|—shift right—|

Check:

$$10 \times 11 = 110 = 64 + 32 + 8 + 4 + 2$$

Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 5

No slide vemos a multiplicação do número  $a = 10$  (ou 1010 em binário) e  $x = 11$  (ou 1011 em binário). O resultado do produto é  $p = 110$  (ou 01101110 em binário):

## EXAMPLES OF BASIC MULTIPLICATION (CONTINUED)

Fig. 9.2  
Examples  
of  
sequential  
multipli-  
cation with  
right and  
left shifts.

$$p^{(j+1)} = 2p^{(j)} + x_{k-j-1}a$$

|shift|  
|-----add-----|

Check:  
 $10 \times 11 = 110 = 64 + 32 + 8 + 4 + 2$

Apr. 2012  
Computer Arithmetic, Multiplication

Left-shift algorithm	
$a$	1 0 1 0
$x$	1 0 1 1
$p^{(0)}$	0 0 0 0
$2p^{(0)}$	0 0 0 0
$+x_3a$	1 0 1 0
$p^{(1)}$	0 1 0 1 0
$2p^{(1)}$	0 1 0 1 0 0
$+x_2a$	0 0 0 0
$p^{(2)}$	0 1 0 1 0 0
$2p^{(2)}$	0 1 0 1 0 0 0
$+x_1a$	1 0 1 0
$p^{(3)}$	0 1 1 0 0 1 0
$2p^{(3)}$	0 1 1 0 0 1 0 0
$+x_0a$	1 0 1 0
$p^{(4)}$	0 1 1 0 1 1 1 0

SLIDE 6

Aqui vemos a solução utilizando shift à esquerda.

OBS: As adições no algoritmo de deslocamento para a esquerda têm 2k bits de largura (o carry produzido a partir dos k bits inferiores pode afetar os k bits superiores), enquanto o algoritmo de deslocamento para a direita requer adições de k bits. Por este motivo, a multiplicação com deslocamentos à direita é preferível.

### 9.3 BASIC HARDWARE MULTIPLIERS

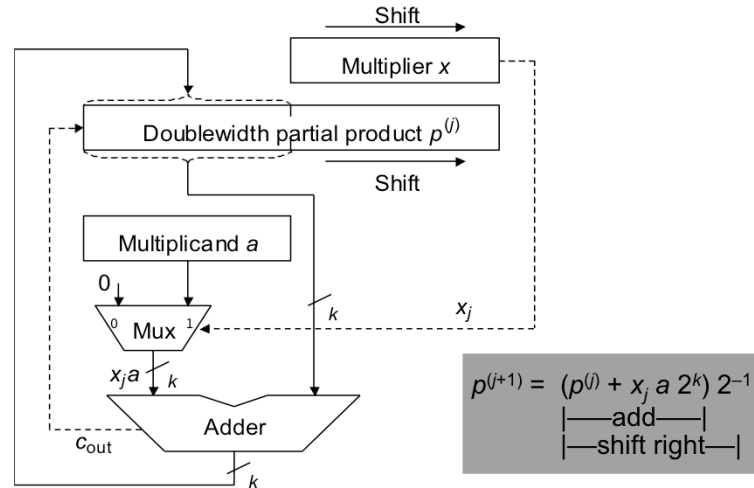


Fig. 9.4 Hardware realization of the sequential multiplication algorithm with additions and right shifts.

Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 7

Vejamos a implementação de hardware do que foi apresentado anteriormente.  
(Right Shift)

O multiplicador “x” (caixa retangular mais superior do esquemático) e o produto parcial cumulativo “p” são armazenados em registradores de deslocamento. O próximo bit do multiplicador a ser considerado está sempre disponível na extremidade direita do registro “x” e é usado para selecionar 0 ou “a” para a adição. A adição e o deslocamento podem ser realizados em 2 ciclos separados ou em 2 subciclos dentro do mesmo ciclo de clock.

## EXAMPLE OF HARDWARE MULTIPLICATION

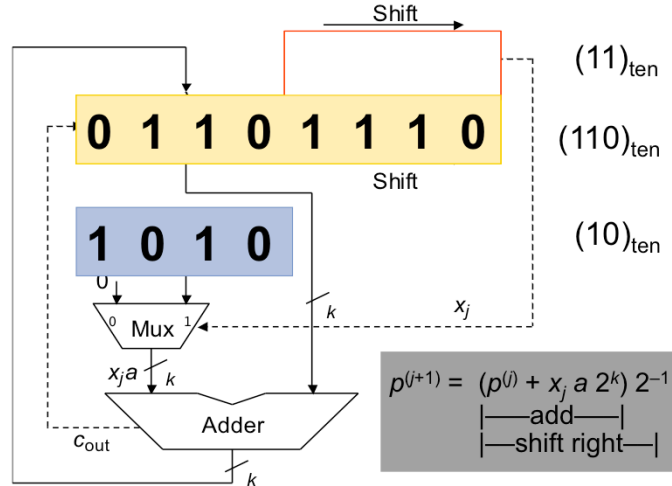


Fig. 9.4a Hardware realization of the sequential multiplication algorithm with additions and right shifts.

Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 8

Aqui um exemplo, de facto o mesmo que vimos anteriormente (multiplicação de 10 por 11).



## SEQUENTIAL MULTIPLICATION WITH LEFT SHIFTS

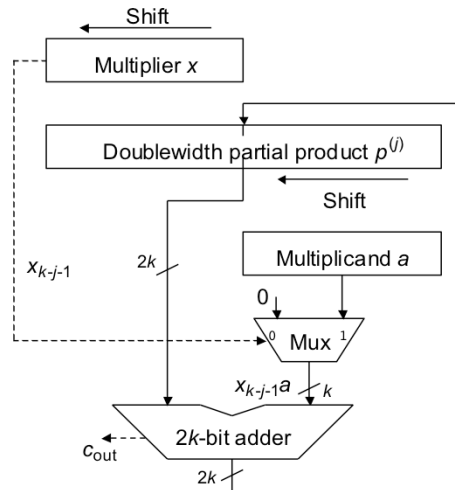


Fig. 9.4b Hardware realization of the sequential multiplication algorithm with left shifts and additions.

Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 9

### (Left Shift)

Aqui também o multiplicador “x” e o produto parcial cumulativo “p” são armazenados em registradores de deslocamento, mas os registradores se deslocam para a esquerda em vez de para a direita. O próximo bit do multiplicador a ser considerado está sempre disponível na extremidade esquerda do registro x e é usado para seleccionar 0 ou a para a adição. Observe que um somador de 2k bits é necessário na realização de hardware de multiplicação com deslocamentos para a esquerda.

## 9.4 MULTIPLICATION OF SIGNED NUMBERS

Fig. 9.6 Sequential multiplication of 2's-complement numbers with right shifts (positive multiplier).

Negative multiplicand, positive multiplier:

No change, other than looking out for proper sign extension

Check:

$$-10 \times 11 = -110 = -512 + 256 + 128 + 16 + 2$$

Apr. 2012

Computer Arithmetic, Multiplication

$a$	1	0	1	1	0				
$x$	0	1	0	1	1				
=====									
$p^{(0)}$	0	0	0	0	0				
$+x_0a$	1	0	1	1	0				
=====									
$2p^{(1)}$	1	1	0	1	1	0			
$p^{(1)}$	1	1	0	1	1		0		
$+x_1a$	1	0	1	1	0				
=====									
$2p^{(2)}$	1	1	0	0	0	1	0		
$p^{(2)}$	1	1	0	0	0		1	0	
$+x_2a$	0	0	0	0	0				
=====									
$2p^{(3)}$	1	1	1	0	0	0	1	0	
$p^{(3)}$	1	1	1	0	0		0	1	0
$+x_3a$	1	0	1	1	0				
=====									
$2p^{(4)}$	1	1	0	0	1	0	0	1	0
$p^{(4)}$	1	1	0	0	1		0	0	1
$+x_4a$	0	0	0	0	0				
=====									
$2p^{(5)}$	1	1	1	0	0	1	0	0	1
$p^{(5)}$	1	1	1	0	0		1	0	0
=====									

SLIDE 10

Uma maneira de multiplicar os valores com sinal por representações de complemento é complementar os operandos negativos, em seguida, multiplicar os valores sem sinal e, em por fim, complementar o resultado se apenas um operando foi complementado no início. Tal esquema de multiplicação indireta é bastante eficiente para números de complemento de 1, mas envolve muita sobrecarga para representação de complemento de 2. É preferível usar um algoritmo de multiplicação direta para esses números, conforme discutido anteriormente.

No exemplo, notamos que os algoritmos apresentados podem funcionar diretamente com um multiplicando de complemento de 2 negativo e um multiplicador positivo. Neste caso, cada termo " $x_j a$ " será um número de complemento de 2 e a soma será corretamente acumulada se usarmos valores estendidos de sinal durante o processo de adição.

## THE CASE OF A NEGATIVE MULTIPLIER

Fig. 9.7 Sequential multiplication of 2's-complement numbers with right shifts (negative multiplier).

Negative multiplicand, negative multiplier:  
In last step (the sign bit), subtract rather than add

Check:  
 $-10 \times -11 = 110 = 64 + 32 + 8 + 4 + 2$

Apr. 2012  
Computer Arithmetic, Multiplication

$a$	1	0	1	1	0	
$x$	1	0	1	0	1	
=====						
$p^{(0)}$	0	0	0	0	0	
$+x_0a$	1	0	1	1	0	
=====						
$2p^{(1)}$	1	1	0	1	1	0
$p^{(1)}$	1	1	0	1	1	0
$+x_1a$	0	0	0	0	0	
=====						
$2p^{(2)}$	1	1	1	0	1	1
$p^{(2)}$	1	1	1	0	1	1
$+x_2a$	1	0	1	1	0	
=====						
$2p^{(3)}$	1	1	0	0	1	1
$p^{(3)}$	1	1	0	0	1	1
$+x_3a$	0	0	0	0	0	
=====						
$2p^{(4)}$	1	1	1	0	0	1
$p^{(4)}$	1	1	1	0	0	1
$+(-x_4a)$	0	1	0	1	0	
=====						
$2p^{(5)}$	0	0	0	1	1	0
$p^{(5)}$	0	0	0	1	1	0
=====						

SLIDE 11

No caso em que ambos são negativos (solução positiva), a extensão de base é feita da mesma forma (Exepto no passo final em que fazemos uma subtração  $-x_4a$  em vez de soma)

Devido ao peso negativo do bit de sinal em números de complemento de 2, um multiplicador de complemento de 2 negativo pode ser manipulado corretamente se " $x_{(k-1)}a$ " for subtraído, em vez de adicionado, no último ciclo. Na prática, a subtração necessária é realizada adicionando o complemento de 2 do multiplicando ou, na verdade, adicionando o complemento de 1 do multiplicando e inserindo um carry-in de 1 no somador.

## Signed 2's-Complement Hardware Multiplier

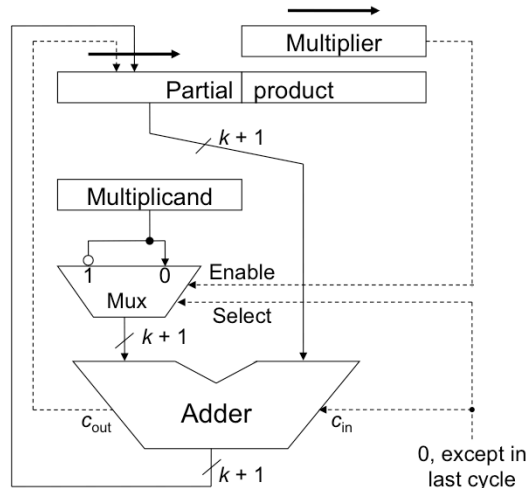


Fig. 9.8 The 2's-complement sequential hardware multiplier.

Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 12

O multiplicador de complemento de 2 possui a estrutura substancialmente igual ao hardware visto anteriormente.

O multiplicando será adicionado ao produto parcial em todos, exceto no ciclo final, quando uma subtração é realizada escolhendo o complemento do multiplicando e inserindo um carry-in de 1.

## Booth's Recoding

Table 9.1 Radix-2 Booth's recoding

$x_i$	$x_{i-1}$	$y_i$	Explanation
0	0	0	No string of 1s in sight
0	1	1	End of string of 1s in x
1	0	-1	Beginning of string of 1s in x
1	1	0	Continuation of string of 1s in x

Example

	1	0	0	1	1	1	0	1	1	0	1	0	1	1	1	0	Operand x
(1)	-1	0	1	0	0	-1	1	0	-1	1	-1	1	0	0	-1	0	Recoded version y

Justification

$$2^i + 2^{i-1} + \dots + 2^{i+1} + 2^i = 2^{i+1} - 2^i$$

Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 13

Uma maneira alternativa de lidar com números de complemento de 2 é usar a recodificação de Booth para representar o multiplicador x no formato de dígitos com sinais.

Booth observou que sempre que houver um grande número de 1s consecutivos em um multiplicador x, a multiplicação pode ser acelerada substituindo a sequência correspondente de adições por uma subtração na porção menos significativa da sequência e uma adição na posição imediatamente à esquerda da porção mais significativa da sequência.

Quanto mais longa for a sequência de 1s, maior será a economia obtida. O efeito dessa transformação é alterar o número binário "x" com conjunto de dígitos [0, 1] para um número binário com sinal "y", usando o conjunto de dígitos [-1, 1]. Portanto, a recodificação de Booth pode ser vista como um tipo de conversão por conjunto de dígitos.

### Example Multiplication with Booth's Recoding

Fig. 9.9 Sequential multiplication of 2's-complement numbers with right shifts by means of Booth's recoding.

$x_i$	$x_{i-1}$	$y_i$
0	0	0
0	1	1
1	0	-1
1	1	0

Check:  
 $-10 \times -11 = 110 = 64 + 32 + 8 + 4 + 2$

Apr. 2012  
 Computer Arithmetic, Multiplication

$a$	1	0	1	1	0
$x$	1	0	1	0	1
$y$	-1	1	-1	1	-1
Multiplier Booth-recoded					
$p^{(0)}$	0	0	0	0	0
$+y_0a$	0	1	0	1	0
$2p^{(1)}$	0	0	1	0	1
$p^{(1)}$	0	0	1	0	1
$+y_1a$	1	0	1	1	0
$2p^{(2)}$	1	1	1	0	1
$p^{(2)}$	1	1	1	0	1
$+y_2a$	0	1	0	1	0
$2p^{(3)}$	0	0	0	1	1
$p^{(3)}$	0	0	0	1	1
$+y_3a$	1	0	1	1	0
$2p^{(4)}$	1	1	1	0	0
$p^{(4)}$	1	1	1	0	0
$y_4a$	0	1	0	1	0
$2p^{(5)}$	0	0	0	1	1
$p^{(5)}$	0	0	0	1	1

SLIDE 14

Aqui está um exemplo.

## PROBLEMAS

**Problema 9.1. Faça a multiplicação  $42 \times 43$  usando:**

- a) Algoritmo *Shift-Left*.
- b) Algoritmo *Shift-Right*.

**Problema 9.2. Faça a multiplicação  $-5 \times -3$  usando:**

- a) Algoritmo *Shift-Right* com recodificação de *Booth*.


SLIDE 15

Gabarito no Moodle

## 9.5 Multiplication by Constants

### Multiplication Using Binary Expansion

Example: Multiply R1 by the constant 113 =  $(1\ 1\ 1\ 0\ 0\ 1)_{\text{two}}$

R2	←	R1 shift-left 1	
R3	←	R2 + R1	
R6	←	R3 shift-left 1	
R7	←	R6 + R1	
R112	←	R7 shift-left 4	
R113	←	R112 + R1	

R<sub>i</sub>: Register that contains *i* times (R1)  
This notation is for clarity; only one register other than R1 is needed

Shorter sequence using shift-and-add instructions

R3	←	R1 shift-left 1 + R1
R7	←	R3 shift-left 1 + R1
R113	←	R7 shift-left 4 + R1

Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 16

Vejamos agora multiplicações por constante.

Uma maneira simples de multiplicar o conteúdo de um registrador (aqui chamado de R1) por uma constante é escrever o multiplicador no formato binário e usar deslocamentos e somas de acordo com os 1s na representação binária. Por exemplo, para multiplicar R1 por 113 =  $(1110001)_{\text{two}}$ , pode-se usar as seguintes sequências de deslocamentos para esquerda, como demonstrado no slide.

Apenas dois registradores são necessários. Um para armazenar o multiplicando a e um para armazenar o resultado parcial mais recente.

Entretanto perceba que há um deslocamento de 4 posições na primeira solução dada. Se apenas deslocamentos de 1 bit forem permitidos, a última instrução na sequência anterior deve ser substituída por três deslocamentos, seguidos por um deslocamento e adição. Observe que o padrão de “shift-and add” e shifts (shift & add, shift & add, shift, shift, shift, shift & add) nesta última versão corresponde ao padrão de bits do multiplicador se seu bit mais significativo for ignorado (110001).



## MULTIPLICATION VIA RECODING

Example: Multiply R1 by 113 =  $(1\ 1\ 1\ 0\ 0\ 0\ 1)_{\text{two}} = (1\ 0\ 0\ 1\ 0\ 0\ 0\ 1)_{\text{two}}$

R8	←	R1 shift-left 3	
R7	←	R8 - R1	
R112	←	R7 shift-left 4	
R113	←	R112 + R1	

Shorter sequence using shift-and-add/subtract instructions

R7	←	R1 shift-left 3 - R1
R113	←	R7 shift-left 4 + R1

6 shift or add (3 shift-and-add) instructions needed without recoding

Se usamos a recodificação de Booth, podemos comprovar que o numero de shifts e somas se reduz a duas.

Isso porque se operações de subtração forem permitidas na sequência podemos aproveitar a vantagem da igualdade:

$$113 = 128 - 16 + 1 = 16(8 - 1) + 1.$$

Então pode-se derivar a sequência de instruções como demonstro no slide acima, para multiplicação por 113:

## MULTIPLICATION VIA FACTORIZATION

Example: Multiply R1 by 119 =  $7 \times 17 = (8 - 1) \times (16 + 1)$

```
R8    ←    R1 shift-left 3
R7    ←    R8 - R1
R112  ←    R7 shift-left 4
R119  ←    R112 + R7
```

Shorter sequence using shift-and-add/subtract instructions

```
R7    ←    R1 shift-left 3 - R1
R119  ←    R7 shift-left 4 + R7
```

Requires a scratch register  
for holding the 7 multiple

Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 18

Fatorar um número pode ajudar também a obter um código eficiente. Por exemplo, para multiplicar R1 por 119, pode-se usar o fato de que:

$119 = 7 \times 17 = (8 - 1) \times (16 + 1)$ . Vejamos a primeira sequência no slide.

Com as instruções shift and (add / subtract), a primeira sequência se reduz a apenas duas instruções.

## MULTIPLICATION BY MULTIPLE CONSTANTS

Example: Multiplying a number by 45, 49, and 65

R9 ← R1 shift-left 3 + R1  
R45 ← R9 shift-left 2 + R9

R7 ← R1 shift-left 3 – R1  
R49 ← R7 shift-left 3 – R7

R65 ← R1 shift-left 6 + R1

} Separate solutions:  
5 shift-add/subtract  
operations

A combined solution for all three constants

R65 ← R1 shift-left 6 + R1  
R49 ← R65 – R1 left-shift 4  
R45 ← R49 – R1 left-shift 2

A programmable  
block can perform  
any of the three  
multiplications

Aqui vemos um último exemplo de como realizar multiplicações pelos números 45, 49, 65, utilizando shifts and adds e uma outra solução combinada para chegar ao resultado todos os três cálculos em conjunto.

## PROBLEMAS

**Problema 9.3.** Faça as seguintes multiplicações por constante a nível de transferência de registradores (RTL design):

- a)  $43 \times A$
- b)  $129 \times A$
- c)  $63 \times A$
- d)  $945 \times A$
- e)  $4,5 \times A$

**Problema 9.4.** Na multiplicação  $978943 \times A$

- a) Faça a compressão direta da informação.
- b) Use dois níveis de CSAs com os múltiplos de 7 no reconhecimento de padrão.

**Problema 9.5.** Na multiplicação  $93177183807 \times A$ :

- a) Faça a compressão direta da informação.
- b) Use dois níveis de CSAs com os múltiplos de 21 no reconhecimento de padrão (pode usar os múltiplos ímpares até 21). Obtenha o custo e caminho crítico considerando  $A_{FA}$  e  $T_{FA}$  como a área e atraso por *Full-Adder*.

SLIDE 20

Gabarito no Moodle

## MULTIPLICATION BY MULTIPLE CONSTANTS: HCUB SOFTWARE TOOL

Fazer a multiplicação pelo conjunto  $\{67,89,99,44,35\}$



<http://spiral.ece.cmu.edu/mcm/gen.html>

Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 21

Como já foi visto podemos fazer uso da fatoração para chegar em maneiras mais eficientes de realizar as multiplicações por constante.

O slide acima demonstra que podemos encontrar fatores comuns que podem ser compartilhados entre as diferentes constantes as quais deseja-se multiplicar.

Por exemplo o número  $67 = 64 + 3$ .

Observa-se então um bloco de deslocamento para esquerda duas vezes e em seguida este bloco é subtraído da própria entrada  $x$ . Assim conseguimos a multiplicação igual a  $“3x”$ . A partir daí somamos  $“3x”$  com o bloco que realiza o deslocamento para esquerda em seis vezes. O resultado então é igual a  $“67x”$ . Então observe na imagem como cada bloco pode ser compartilhado entre as constantes.

## PROBLEMAS

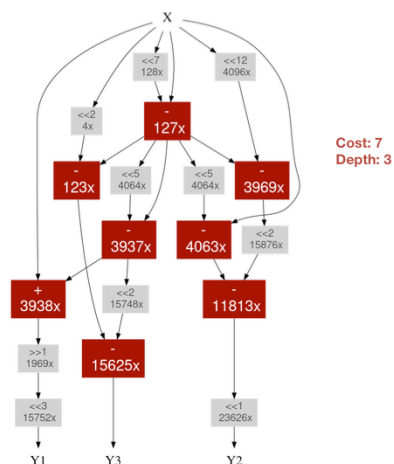
**Problema 9.6.** A partir dos ferramenta que obtém o grafo associado à multiplicação de múltiplas constantes obtenha:

- a) O grafo para a obtenção dos números primos 3, 5, 11, 13, 37, 41 e 43 (para a geração do grafo use *Fractional bits*: 0, *Algortihm*: Hcub e *Depth Limit*: Minimum possible)
- b) O que poderia ser feito para melhorar a eficiência tendo em consideração que o substrator é uma unidade maior e com maior atraso que um somador?
- c) Use agora *Algortihm*: BHM na ferramenta e reduza o número de níveis.

Gabarito no Moodle

## MULTIPLICATION BY MULTIPLE CONSTANTS: BINARY VS RNS

Obtenção de  $Y_1=15752$ ,  $Y_2=23626$  e  $Y_3=15625$  usando Algorithm: Hcub e depth: minimum possible.



Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 23

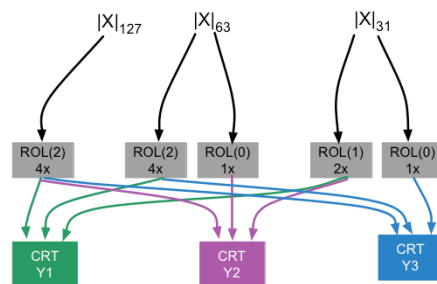
Aqui é apresentado uma árvore de somadores para a obtenção da multiplicação pelas constantes  $Y_1=15752$ ,  $Y_2=23626$  e  $Y_3=15625$  com um custo de 7 somadores e uma profundidade de 3. Compararemos no seguinte slide fazendo uso do conjunto modular  $\{127, 63, 31\}$ .

## MULTIPLICATION BY MULTIPLE CONSTANTS: BINARY VS RNS

Obtenção de  $Y1=15752$ ,  $Y2=23626$  e  $Y3=15625$  usando o conjunto RNS  $\{127, 63, 31\}$

$ 15752 _{127}=4$	$ 23626 _{127}=4$	$ 15625 _{127}=4$
$ 15752 _{63}=2$	$ 23626 _{63}=1$	$ 15625 _{63}=4$
$ 15752 _{31}=4$	$ 23626 _{31}=4$	$ 15625 _{31}=1$

ROL(i)  
 $2^i$   
Operação de Rotação  
para a esquerda (ROL)



Olhando apenas a aritmética:

Cost: 0  
Depth: 0

Apr. 2012  
Computer Arithmetic, Multiplication

SLIDE 24

Nesse slide vemos como ficaram as constantes quando são reduzidas modularmente, ficando todas potências de 2. Sabemos que fazer um deslocamento para modulos do tipo  $2^n-1$  é apenas fazer uma operação de rotação obtendo um custo de zero na aritmética. Não foi considerada nem a conversão de binário a RNS nem de RNS a binário, mas da uma ideia do benefício que podemos obter para algumas multiplicações por constante.



## PROBLEMAS

**Problema 9.7.** A partir dos ferramenta que obtém o grafo associado à multiplicação de múltiplas constantes obtenha:

- a) O grafo para a obtenção da multiplicação de 519184, 778772 e 908566 (para a geração do grafo use *Fractional bits*: 0, *Algortihm*: Hcub e *Depth Limit*: Minimum possible)
- b) Agora faça a multiplicações pelas constantes apresentadas no apartado anterior mas fazendo uso do conjunto modular {511, 255, 127} em RNS. Indique o ganho da aritmética em binário versus RNS.

Gabarito no Moodle