9 BASIC MULTIPLICATION SCHEMES

Chapter Goals

Study shift/add or bit-at-a-time multipliersand set the stage for faster methods andvariations 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

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

9.1 SHIFT/ADD MULTIPLICATION ALGORITHMS

Notation for our discussion of multiplication algorithms:

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

Initially, we assume unsigned operands

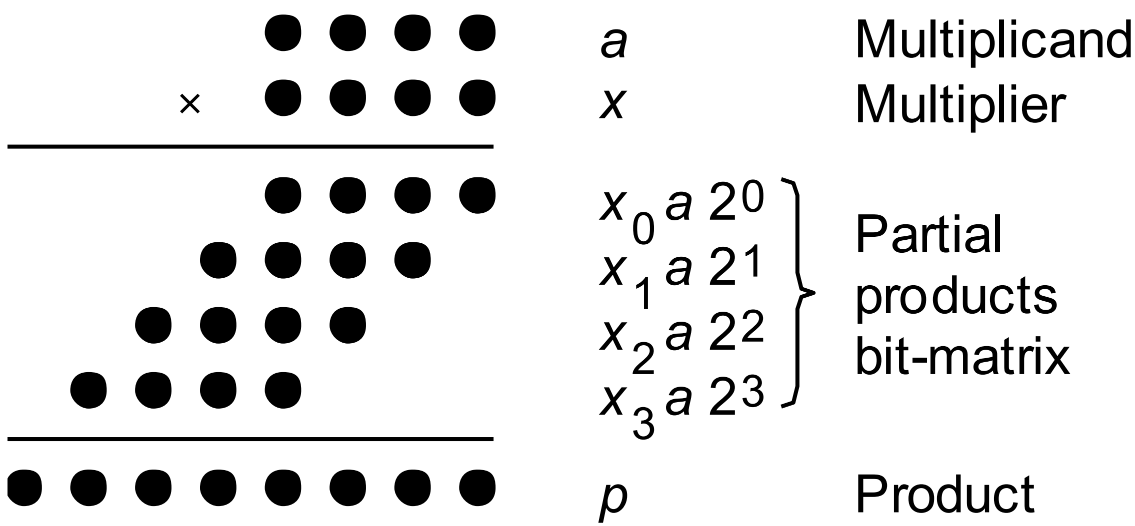
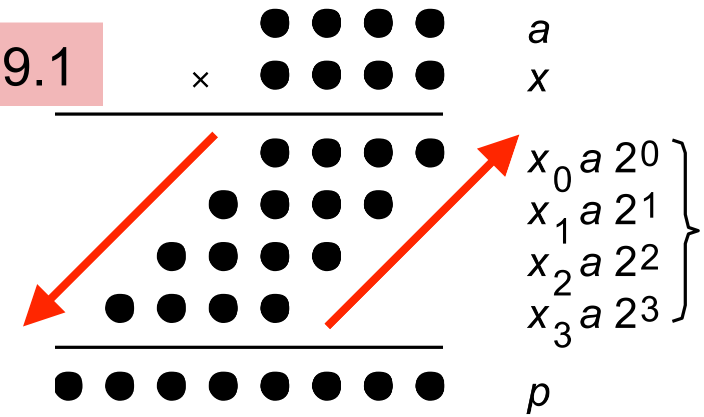


Fig. 9.1 Multiplication of two 4-bit unsigned binary numbers in dot notation.

MULTIPLICATION RECURRENCE

Fig. 9.1



Multiplicand
Multiplier
Partial products bit-matrix
Product

Preferred

Multiplication with right shifts: top-to-bottom accumulation

$$p^{(j+1)} = \underbrace{\underbrace{(p^{(j)} + x_j\, a\, 2^k)}_{\text{add}}\, 2^{-1}}_{\text{shift right}} \qquad \text{with} \qquad p^{(0)} = 0 \text{ and}$$

Multiplication with left shifts: bottom-to-top accumulation

$$p^{(j+1)} = \underbrace{\underbrace{2\, p^{(j)}}_{\text{shift}} + x_{k-j-1}\, a}_{\text{add}} \qquad \text{with} \qquad p^{(0)} = 0 \text{ and}$$

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 multiplication with right and left shifts.

$$p^{(j+1)} = \underbrace{\underbrace{(p^{(j)} + x_j\, a\, 2^k)}_{\text{add}}\, 2^{-1}}_{\text{shift right}}$$

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

EXAMPLES OF BASIC MULTIPLICATION (CONTINUED)

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

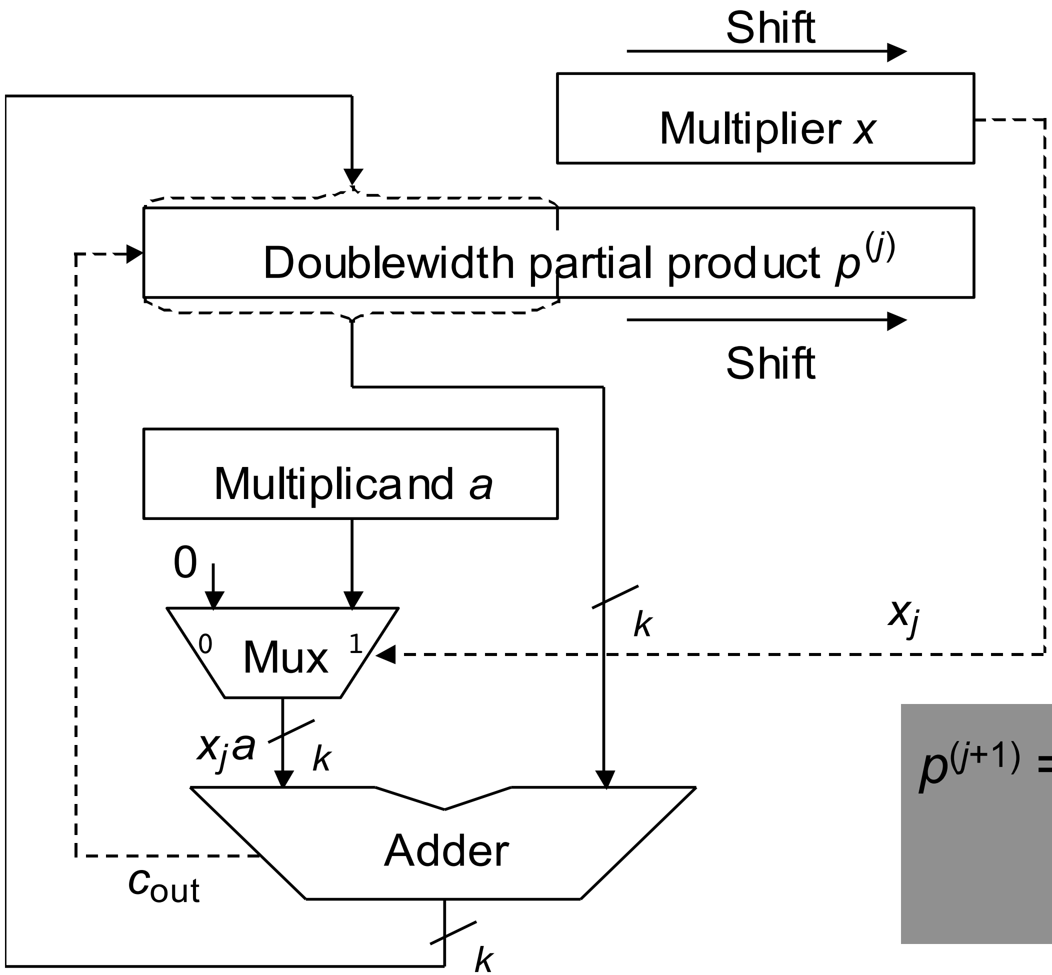
$$p^{(j+1)} = \underbrace{\underbrace{2\,p^{(j)}}_{\text{shift}} + x_{k-j-1}a}_{\text{add}}$$

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	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

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

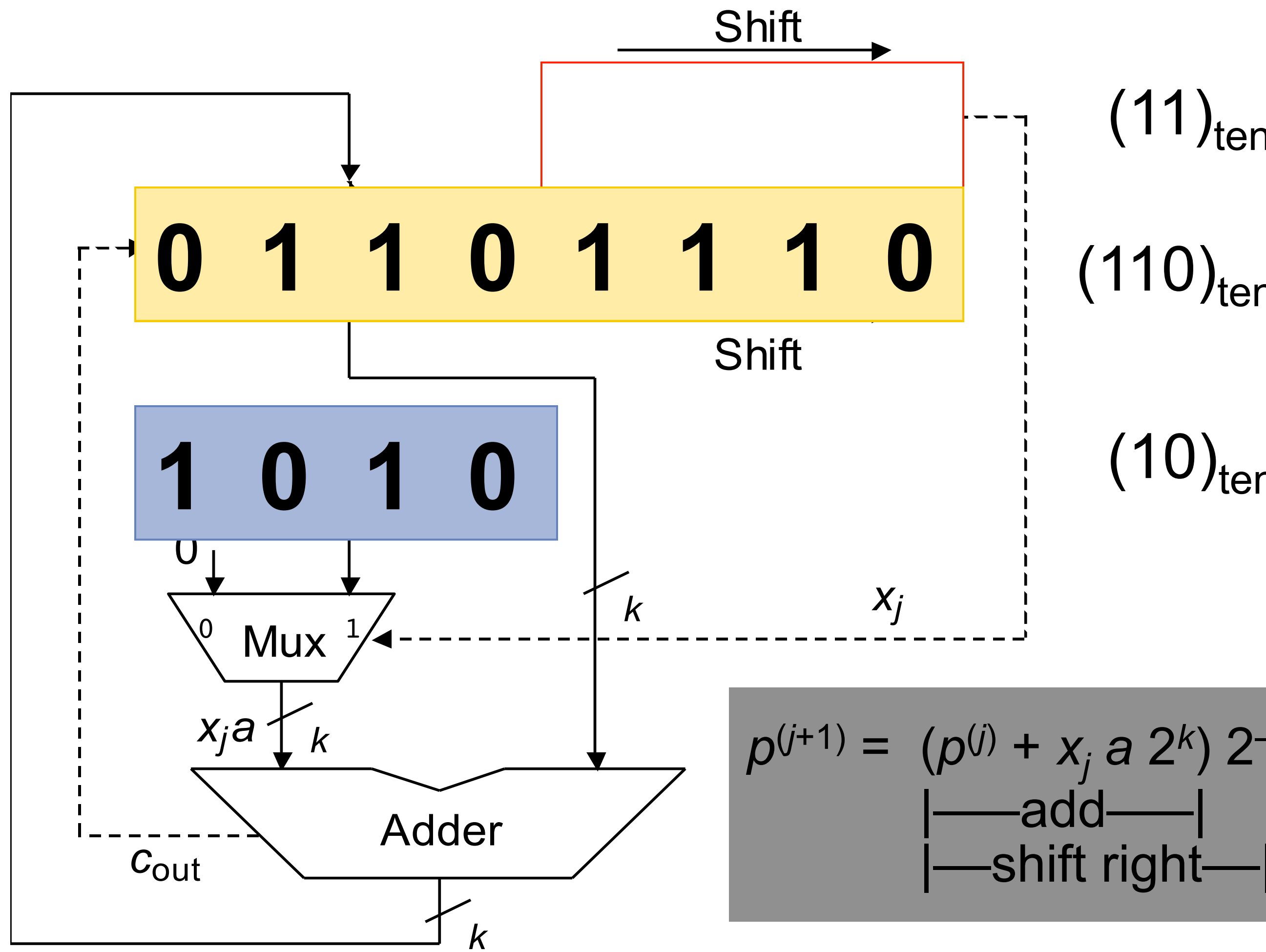
9.3 BASIC HARDWARE MULTIPLIERS



$p^{(j+1)} = (p^{(j)} + x_j\, a\, 2^k)\, 2^{-1}$
|——add——|
|—shift right—|

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

EXAMPLE OF HARDWARE MULTIPLICATION



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

|——add——|

|—shift right—|

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

SEQUENTIAL MULTIPLICATION WITH LEFT SHIFTS

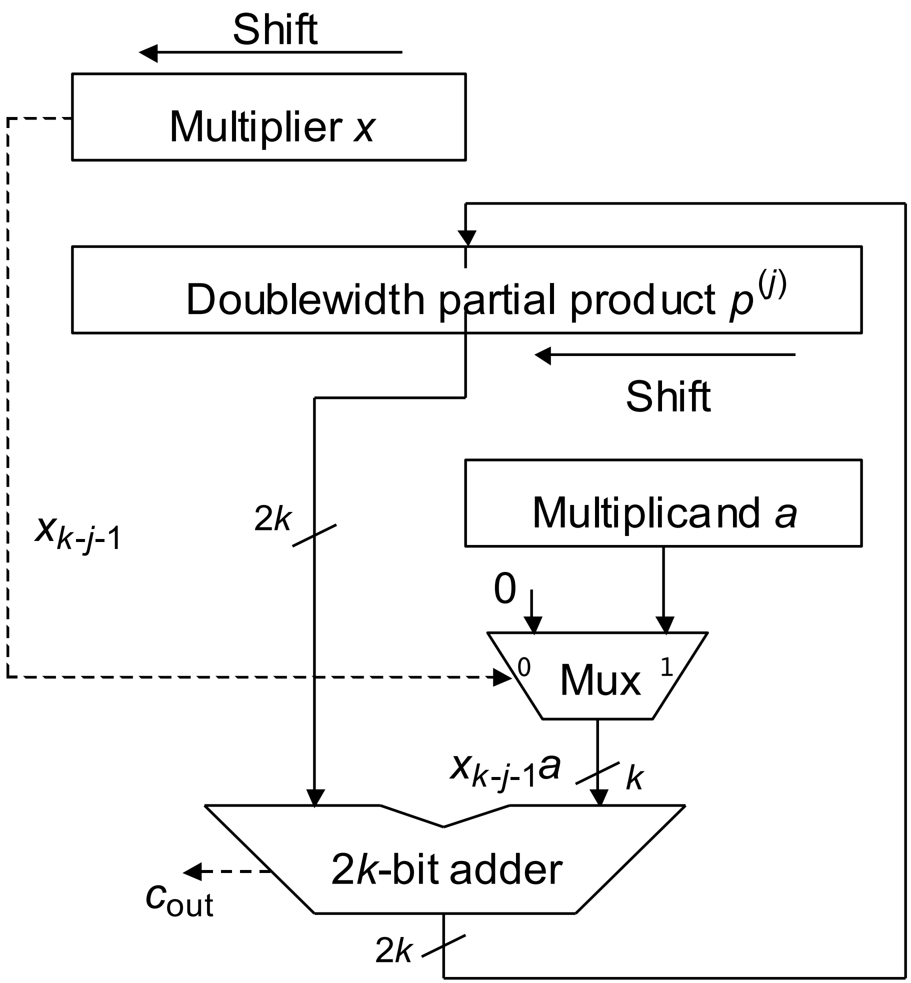


Fig. 9.4b Hardware realization of the sequential multiplication algorithm with left shifts and additions.

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

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	0	
$+x_4a$		0	0	0	0	0					
$2p^{(5)}$	1	1	1	0	0	1	0	0	1	0	
$p^{(5)}$		1	1	1	0	0	1	0	0	1	0

Check:
$^{-}10 \times 11 = {}^{-}110 = {}^{-}512 + 256 + 128 + 16 + 2$

THE CASE OF A NEGATIVE MULTIPLIER

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

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	0				
$p^{(2)}$		1	1	1	0	1	1	0			
$+x_2a$		1	0	1	1	0					
$2p^{(3)}$	1	1	0	0	1	1	1	0			
$p^{(3)}$		1	1	0	0	1	1	1	0		
$+x_3a$		0	0	0	0	0					
$2p^{(4)}$	1	1	1	0	0	1	1	1	0		
$p^{(4)}$		1	1	1	0	0	1	1	1	0	
$+(-x_4a)$		0	1	0	1	0					
$2p^{(5)}$	0	0	0	1	1	0	1	1	1	0	
$p^{(5)}$		0	0	0	1	1	0	1	1	1	0

Negative multiplicand, negative multiplier:

In last step (the sign bit), subtract rather than add

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

SIGNED 2's-COMPLEMENT HARDWARE MULTIPLIER

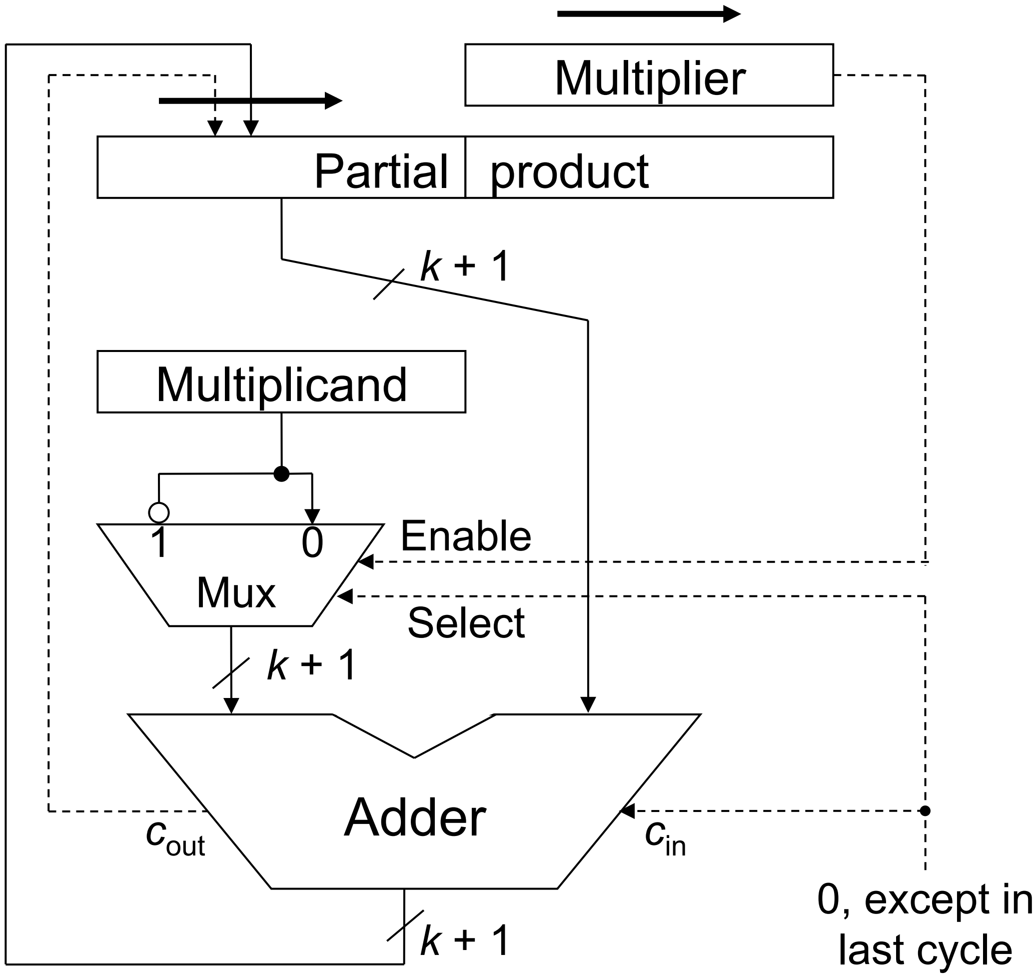


Fig. 9.8 The 2's-complement sequential hardware multiplier.

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

$$2^j + 2^{j-1} + \ldots + 2^{i+1} + 2^i = 2^{j+1} - 2^i$$

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

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

Check:

$^-10 \times {}^-11 = 110 = 64+32+8+4+2$

PROBLEMAS

Problema 9.1. Faça a multiplicação 42x43 usando:

a) Algoritmo *Shift-Left*.

b) Algoritmo *Shift-Right*.

Problema 9.2. Faça a multiplicação -5x-3 usando:

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

9.5 MULTIPLICATION BY CONSTANTS

Multiplication Using Binary Expansion

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

Shift, add Shift

R2	$\leftarrow$	R1 shift-left 1
R3	$\leftarrow$	R2 + R1
R6	$\leftarrow$	R3 shift-left 1
R7	$\leftarrow$	R6 + R1
R112	$\leftarrow$	R7 shift-left 4
R113	$\leftarrow$	R112 + R1

R*i*: 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	$\leftarrow$	R1 shift-left 1 + R1
R7	$\leftarrow$	R3 shift-left 1 + R1
R113	$\leftarrow$	R7 shift-left 4 + R1

MULTIPLICATION VIA RECODING

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

Shift, subtract   Shift   Shift, add

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

MULTIPLICATION VIA FACTORIZATION

Example: Multiply R1 by 119 = 7 × 17 = (8 – 1) × (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

MULTIPLICATION BY MULTIPLE CONSTANTS

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

R9	←	R1 shift-left 3 + R1	Separate solutions: 5 shift-add/subtract operations
R45	←	R9 shift-left 2 + R9	
R7	←	R1 shift-left 3 – R1	
R49	←	R7 shift-left 3 – R7	
R65	←	R1 shift-left 6 + R1	

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

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 978943xA

a) Faça a compressão direta da informação.

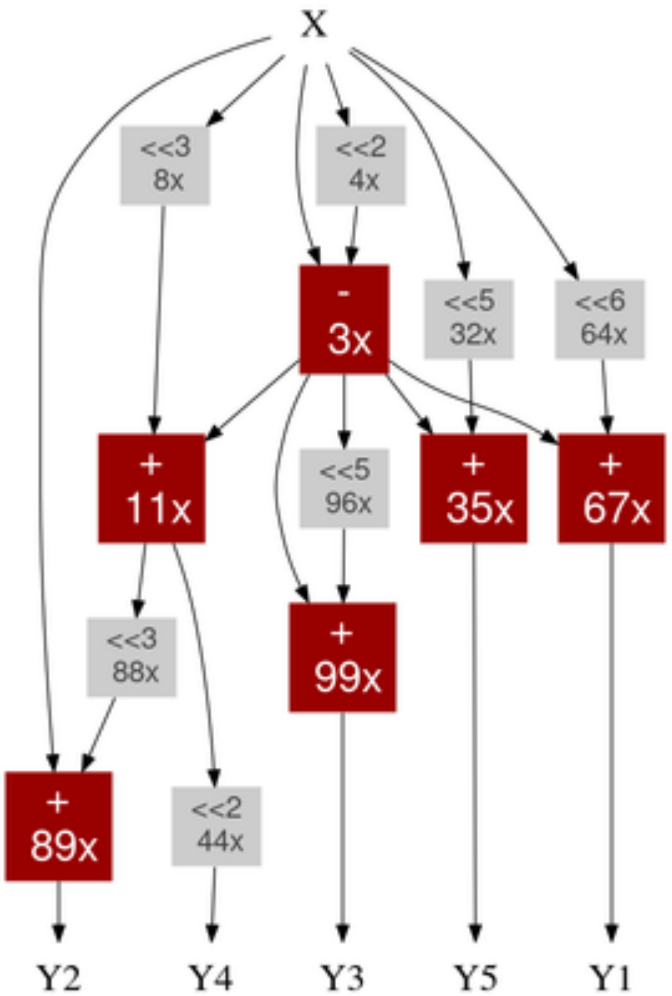
b) Use dois níveis de CSAs com os múltiplos de 7 no reconhecimento de padrão.

Problema 9.5. Na multiplicação 93177183807xA:

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*.

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

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)

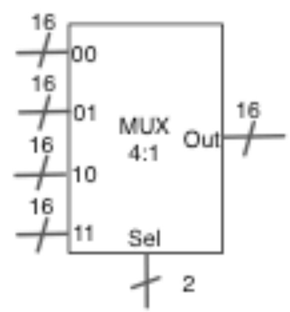
b) O que poderia ser feito para melhorar a eficiência tendo em consideração que o substrator é uma unidade maior e com maior atraso que um somador?

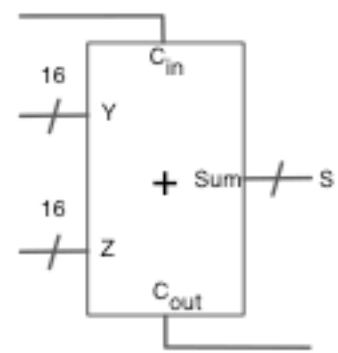
c) Use agora *Algortihm*: BHM na ferramenta e reduza o número de níveis.

MULTIPLICATION BY MULTIPLE CONSTANTS: ANOTHER EXAMPLE

Desenhe um circuito aritmético com uma entrada de seleção de dois bits, $S = s_1 s_0$, que realize as operações aritméticas mostradas na tabela usando unicamente um somador de 8 bits com *carry in* e *carry out* e multiplexadores 4:1. Suponha *A*, *B*, *C* e *D* entradas de 4-bits sendo a entrada A uma entrada sempre par.

S	F
00	$F = 3A + 560B + 256C + 4096D + 16389$
01	$F = 0{,}5A + 25B + 4224C + 4352D + 2049$
10	$F = 8499B + 6144$
11	$F = \dfrac{(9393A + 4096B + 256C + 164)}{2}$

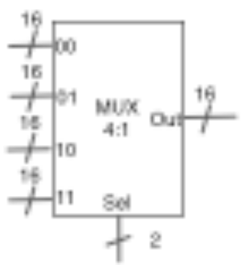


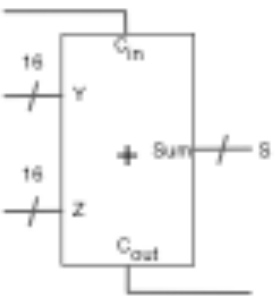


PROBLEMAS

Problema 9.7. Desenhe um circuito aritmético com uma entrada de seleção de dois bits, *S=s1s0*, que realize as operações aritméticas mostradas na tabela usando unicamente um somador de 8 bits com *carry in* e *carry out* e multiplexadores 4:1. Suponha *A, B, C e D* entradas de *4*-bits sendo a entrada A uma entrada sempre par.

S	F
00	$F = 3A + 560B + 256C + 4096D + 16389$
01	$F = 0{,}5A + 25B + 4224C + 4352D + 2049$
10	$F = 8499B + 6144$
11	$F = \dfrac{(9393A + 4096B + 256C + 164)}{2}$





MULTIPLICATION BY MULTIPLE CONSTANTS: EXAMPLE

Solução:

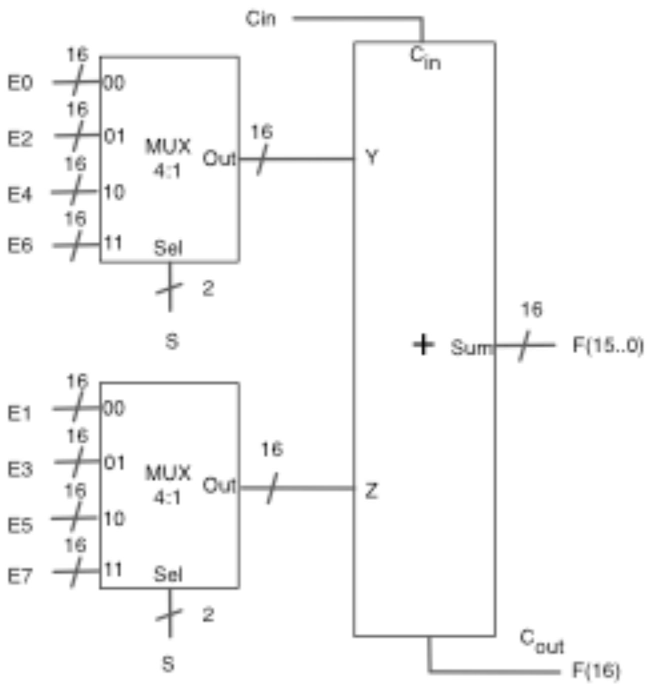
As entradas são:

$A = a_3a_2a_1a_0 = a_3a_2a_10.$

$B = b_3b_2b_1b_0.$

$C = c_3c_2c_1c_0.$

$D = d_3d_2d_1d_0.$



MULTIPLICATION BY MULTIPLE CONSTANTS: EXAMPLE

Temos de inserir nas entradas E_i a informação das operações de multiplicação por constante. A tabela seguinte contem a informação a ser inserida.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cin
E0	d3	d2	d1	d0	c3	c2	c1	c0	b3	b2	b1	b0	a3	a2	a1	1	1
E1	0	1	0	b3	b2	b1	b0	b3	b2	b1	b0	a3	a2	a1	1	1	
E2	c3	c2	c1	c0	1	c3	c2	c1	c0	b3	b2	b1	b0	a3	a2	a1	1
E3	d3	d2	d1	d0	d3	d2	d1	d0	b3	b2	b1	b0	b3	b2	b1	b0	
E4	b3	b2	b1	b0	b3	b2	b1	b0	b3	b2	b1	b0	b3	b2	b1	b0	1
E5	b3	b2	b1	b0	1	1	0	b3	b2	b1	b0	b3	b2	b1	b0	1	
E6	a3	a2	a1	a3	a2	a1	a3	a2	a1	a3	a2	a1	a3	a2	a1	1	1
E7	0	b3	b2	b1	b0	c3	c2	c1	c0	1	0	1	0	a3	a2	a1	

Para F quando S=0 (soma de E0 com E1).
$F=(2^{1}+2^{0})A+(2^{9}+2^{5}+2^{4})B+(2^{8})C+(2^{12})D+(2^{14}+2^{2}+2^{0})$.
Para F quando S=1 (soma de E2 com E3).
$F=(2^{-1})A+(2^{4}+2^{3}+2^{0})B+(2^{12}+2^{7})C+(2^{12}+2^{8})D+(2^{11}+2^{0})$.
Para F quando S=2 (soma de E4 com E5).
$F=(2^{13}+2^{8}+2^{5}+2^{4}+2^{1}+2^{0})B+(2^{11}+2^{10}+2^{1})$.
Para F quando S=3 (soma de E6 com E7) numerador da divisão.
$F=(2^{13}+2^{10}+2^{7}+2^{4}+2^{1}+2^{0})A+(2^{12})B+(2^{8})C+(2^{7}+2^{5}+2^{1})$.