MULTIPLICATION

$p = x \times y$

x (multiplicand), y (multiplier), and p (product) signed integers

- SCHEMES

  a) SEQUENTIAL ADD-SHIFT RECURRENCE

  * CPA, CSA, SIGNED-DIGIT ADDER
  * HIGHER RADIX AND RECODING

  b) COMBINATIONAL

  * CPA, CSA, SIGNED-DIGIT ADDER
  * HIGHER RADIX AND RECODING

  c) COLUMN REDUCTION

  d) ARRAYS WITH $k \times l$ MULTIPLIERS

TOPICS (cont.)

- MULTIPLY-ADD AND MULTIPLY-ACCUMULATE
- SATURATING MULTIPLIERS
- TRUNCATING MULTIPLIERS
- RECTANGULAR MULTIPLIERS
- SQUARERS
- CONSTANT AND MULTIPLE CONSTANT MULTIPLIERS

SIGN-AND-MAGNITUDE

- EACH OPERAND:

  sign with value $+1$ and -1 and n-digit magnitude
- RESULT: a sign and a $2n$-digit magnitude
- HIGH-LEVEL ALGORITHM

$$sign(p) = sign(x) \cdot sign(y)$$

$$|p| = |x||y|$$

- REPRESENTATIONS OF MAGNITUDES

$$\begin{array}{lll} X = (x_{n-1}, x_{n-2}, \ldots, x_0) & |x| = \Sigma_{i=0}^{n-1} x_i r^i & \text{(multiplicand)} \\ Y = (y_{n-1}, y_{n-2}, \ldots, y_0) & |y| = \Sigma_{i=0}^{n-1} y_i r^i & \text{(multiplier)} \\ P = (p_{2n-1}, p_{2n-2}, \ldots, p_0) & |p| = \Sigma_{i=0}^{2n-1} p_i r^i & \text{(product)} \end{array}$$

TWO'S COMPLEMENT

- RADIX-2 CASE
- EACH OPERAND: n-BIT VECTOR
- RESULT: $2n$-BIT VECTOR

$$-(2^{n-1})(2^{n-1}-1) \leq p \leq (-2^{n-1})(-2^{n-1}) = 2^{2n-2}$$

- x_R, y_R and p_R – positive integer representations of x, y, and p
- HIGH-LEVEL ALGORITHM

$$p_R = \begin{cases} x_R y_R & \text{if } x \geq 0,\ y \geq 0 \\ 2^{2n} - (2^n - x_R) y_R & \text{if } x < 0,\ y \geq 0 \\ 2^{2n} - x_R(2^n - y_R) & \text{if } x \geq 0,\ y < 0 \\ (2^n - x_R)(2^n - y_R) & \text{if } x < 0,\ y < 0 \end{cases}$$

TYPES OF ALGORITHMS

1. ADD-AND-SHIFT ALGORITHM
   - SEQUENTIAL
   - COMBINATIONAL
2. COMPOSITION OF SMALLER MULTIPLICATIONS

RECURRENCE FOR MAGNITUDES

$$p[0] = 0$$
$$p[j+1] = r^{-1}(p[j] + x \cdot r^n y_j) \text{ for } j = 0, 1, \ldots, n-1$$
$$p = p[n]$$

RELATIVE POSITION OF OPERANDS

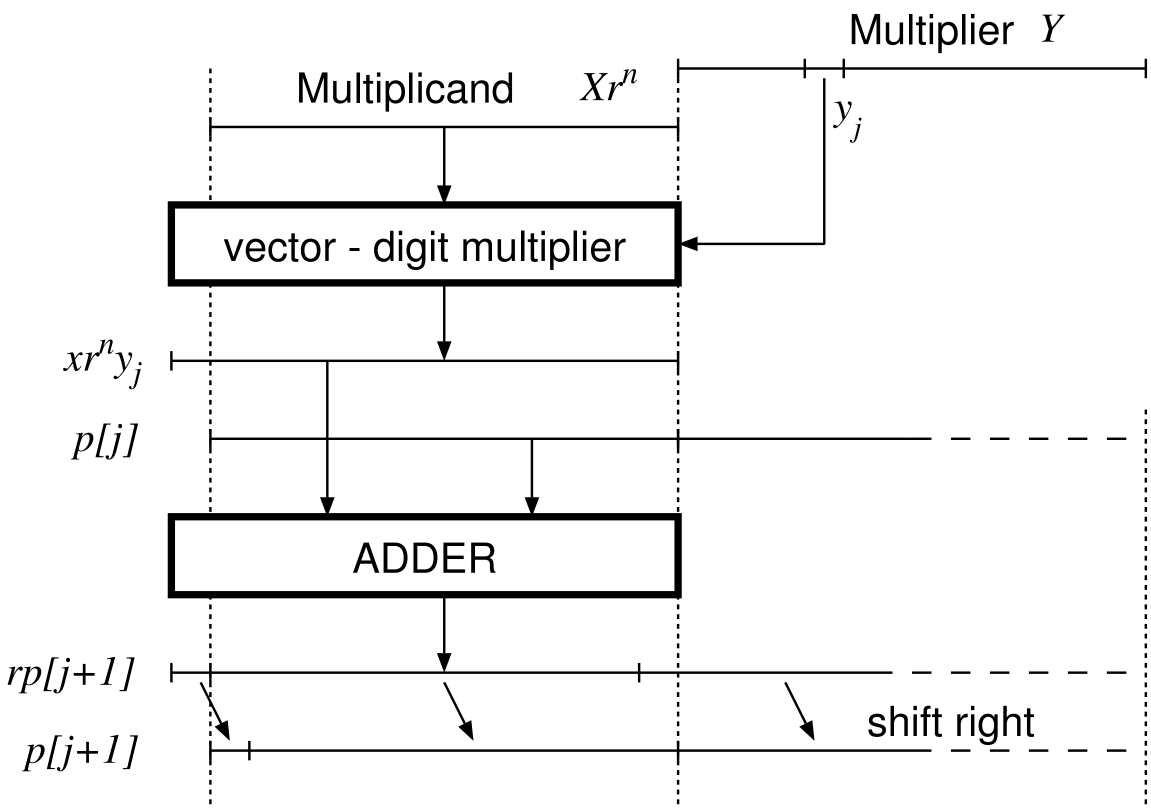


Figure 4.1: RELATIVE POSITION OF OPERANDS IN MULTIPLICATION RECURRENCE

$$T = n(t_{digmult} + t_{add} + t_{reg})$$

SEQUENTIAL MULTIPLIER WITH REDUNDANT ADDER

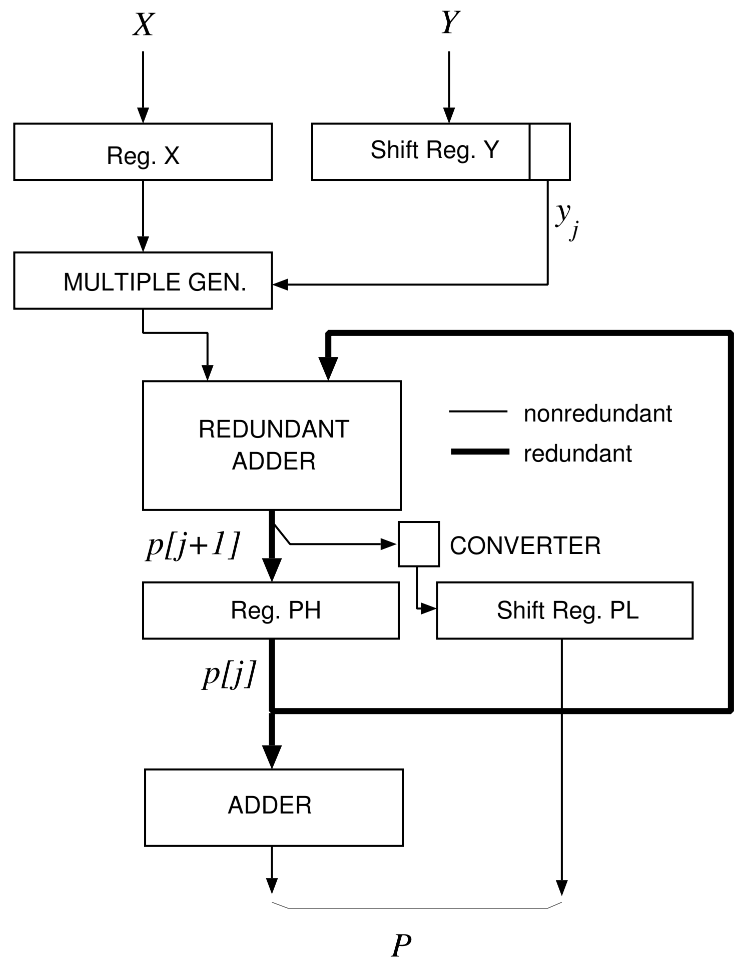


Figure 4.2: SEQUENTIAL MULTIPLIER WITH REDUNDANT ADDER

RADIX-4 SEQUENTIAL MULTIPLIER RECODING

- MULTIPLIER RECODING TO AVOID VALUES $z_i = 3$

$$z_i = y_i + c_i - 4c_{i+1}$$

$y_i + c_i$	z_i	c_{i+1}
0	0	0
1	1	0
2	2	0
3	-1	1
4	0	1

RADIX-4 MULTIPLIER IMPLEMENTATION

THREE PIPELINED STAGES

- Stage 1: MULTIPLIER RECODING
- Stage 2: GENERATING THE MULTIPLE OF THE MULTIPLICAND
- Stage 3: ADDITION AND SHIFT (with conversion of the shifted-out bits).

cycle	0	1	2	3	4	5	...	$m+1$	$m+2$	
	LOAD X LOAD Y									
Stage 1	0	z_0	z_1	z_2	z_3	z_4				
Stage 2	0	0	Xz_0	Xz_1	Xz_2	Xz_3		Xz_{m-1}		
Stage 3	0	0	0	PS[1] SC[1]	PS[2] SC[2]	PS[3] SC[3]		PS[$m-1$] SC[$m-1$]	PS[m] SC[m]	
CPA										Final product

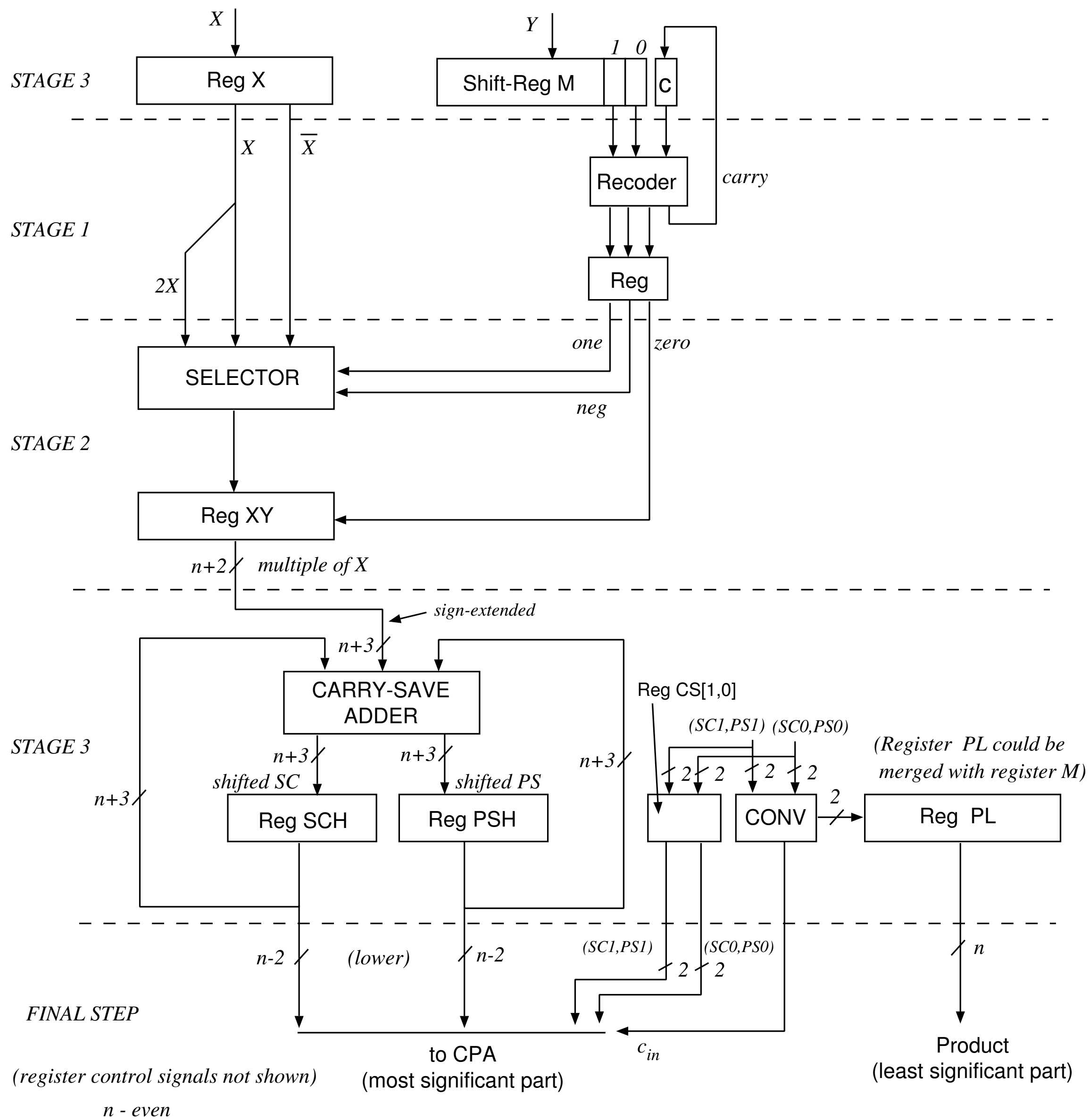


Figure 4.3: RADIX-4 MULTIPLIER.

RECODING IMPLEMENTATION

- BASED ON MULTIPLIER BITS (M_1, M_0) and CARRY FLAG C

$$one = M_0 \oplus C = \begin{cases} 0 & \text{select } 2x \\ 1 & \text{select } x \end{cases}$$

$$neg = M_1 \cdot C + M_1 \cdot M_0 = \begin{cases} 0 & \text{select direct} \\ 1 & \text{select complement} \end{cases}$$

$$zero = M_1 \cdot M_0 \cdot C + M_1' \cdot M_0' \cdot C' = \begin{cases} 0 & \text{load non} - \text{zero multiple} \\ 1 & \text{load zero multiple (clear)} \end{cases}$$

$$C_{next} = M_1 M_0 + M_1 C$$

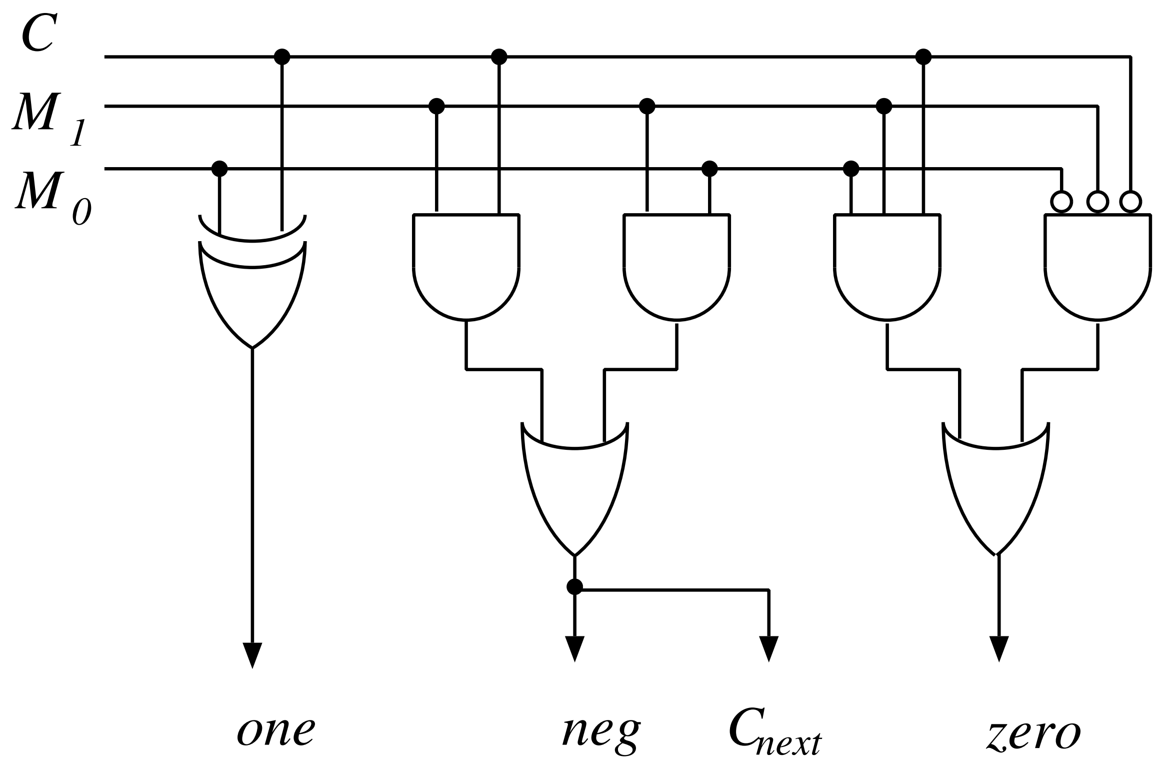


Figure 4.4: RECODER IMPLEMENTATION.

GENERATION OF $(-1)x$

$$
\begin{array}{lllllll}
PS[j] & PS_{n+2} & PS_{n+1} & PS_n & \cdots & PS_1 & PS_0 \\
SC[j] & SC_{n+2} & SC_{n+1} & SC_n & \cdots & SC_1 & SC_0 \\
-x & X'_{n+2} & X'_{n+1} & X'_n & \cdots & X'_1 & X'_0 \\
\hline
CSA & s_{n+2} & s_{n+1} & s_n & \cdots & s_1 & s_0 \\
 & c_{n+2} & c_{n+1} & c_n & \cdots & c_1 & 1^*
\end{array}
$$

$*$ for 2's complement of x

EXAMPLE OF RADIX-4 MULTIPLICATION

$n = 5 \quad m = 3$ radix-4 digits

$x = 29 \qquad X = 11101$

$y = 27 \qquad Y = 11011$

$Z = 2\overline{1}\overline{1} \quad (z = y) \quad (-1 = \overline{1})$

	CSA	shifted out
$PS[0]$	00000000	
$SC[0]$	00000000	
xZ_0	11100010	
$4PS[1]$	11100010	
$4SC[1]$	0000000**1**	
$PS[1]$	11111000	11
$SC[1]$	00000000	
xZ_1	11100010	
$4PS[2]$	00011010	
$4SC[2]$	1100000**1**	
$PS[2]$	00000110	1111
$SC[2]$	11110000	
xZ_2	00111010	
$4PS[3]$	11001100	
$4SC[3]$	0110010**0**	
$PS[3]$	11110011	001111
$SC[3]$	00011001	
P	1100	001111 $=$ 783

EXTENSION TO HIGHER RADICES

- EXTENSION TO HIGHER RADICES REQUIRES PREPROCESSING OF MORE MULTIPLES

- ALTERNATIVE: USE SEVERAL RADIX-4 AND/OR RADIX-2 STAGES IN ONE ITERATION

EXAMPLE: RADIX-16 MULTIPLIER DIGIT {0,...,15} RECODED INTO A RADIX-16 SIGNED-DIGIT v_i IN THE SET {-10,...,0,...,10} AND DECOMPOSED INTO TWO RADIX-4 DIGITS u_i and w_i SUCH THAT

$$v_i = 4u_i + w_i \quad u_i, w_i \in \{-2, -1, 0, 1, 2\}$$

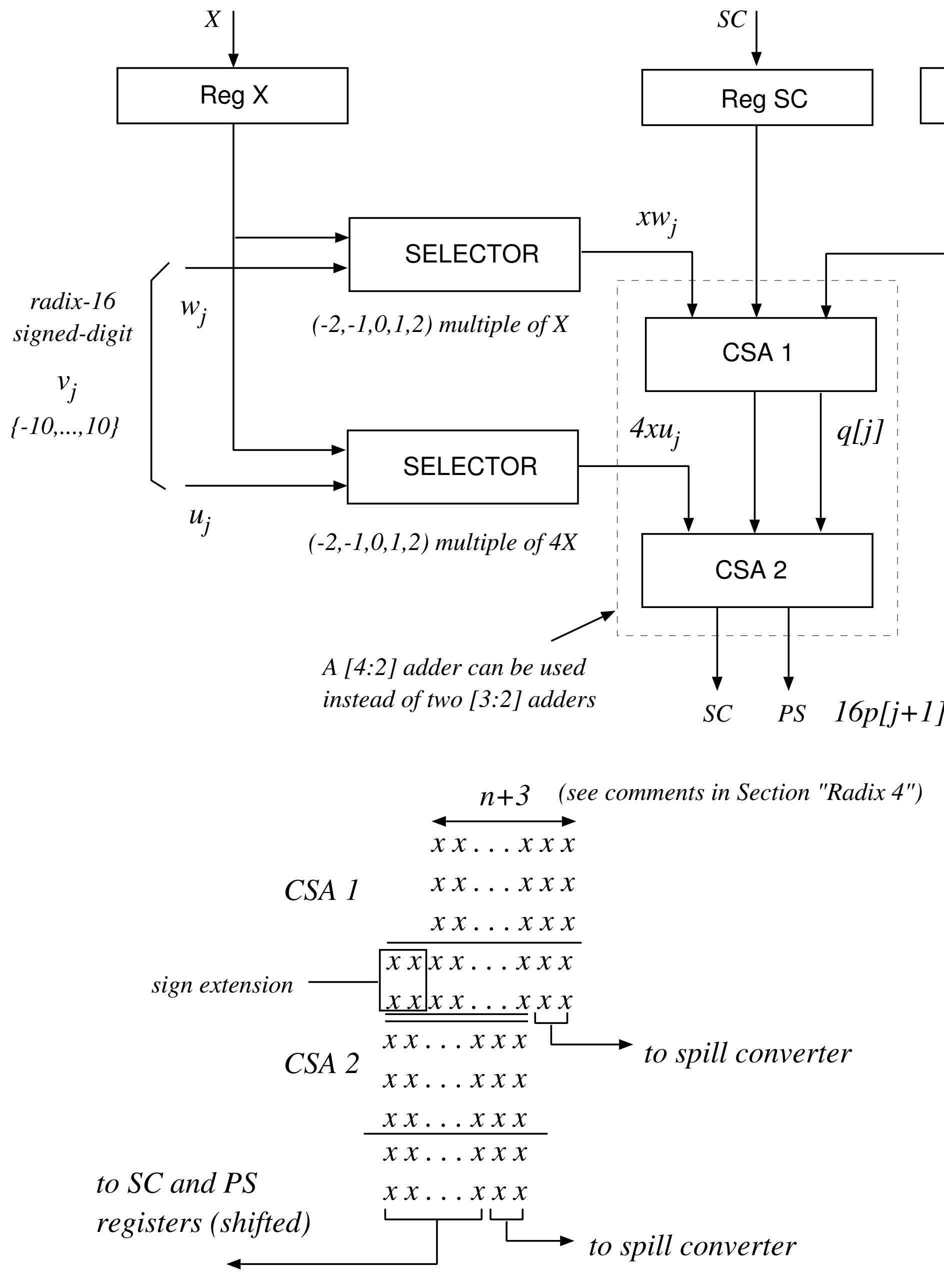


Figure 4.5: RADIX-16 MULTIPLICATION DATAPATH (partial).

TWO'S COMPLEMENT

- MULTIPLICAND IN 2'S COMPLEMENT $\Longrightarrow$ ADDITION AND SHIFT OPERATIONS PERFORMED IN THIS SYSTEM
- THE EFFECT OF 2'S COMPLEMENT MULTIPLIER TAKEN INTO ACCOUNT IN TWO WAYS:

1. BY SUBTRACTING INSTEAD OF ADDING IN THE LAST ITERATION

$$y = -y_{n-1}2^{n-1} + \sum_{i=0}^{n-2} y_i 2^i$$

$\Longrightarrow$ CORRECTION STEP.

2. BY RECODING THE MULTIPLIER INTO A SIGNED-DIGIT SET

COMBINATIONAL MULTIPLICATION

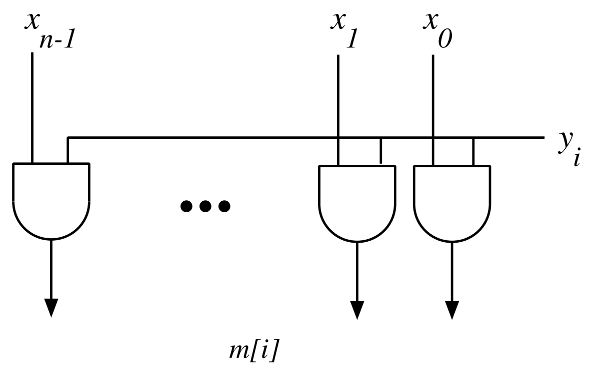
$$p = \sum_{i=0}^{n-1} x y_i r^i$$

DONE IN TWO STEPS:

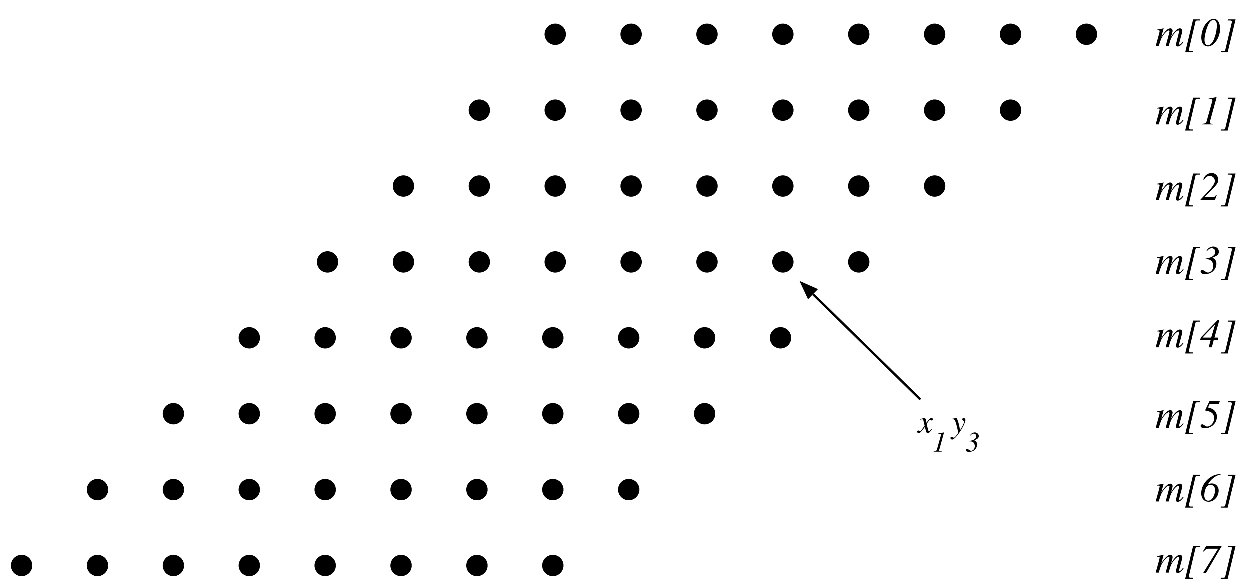
1. GENERATION OF THE MULTIPLES OF THE MULTIPLICAND

$$(x \times y_i) r^i$$

2. MULTIOPERAND ADDITION OF THE MULTIPLES GENERATED IN STEP 1.



(a)



(b)

Figure 4.6: (a) RADIX-2 MULTIPLE GENERATION. (b) BIT-MATRIX FOR MULTIPLICATION Of MAGNITUDES ($n = 8$).

RADIX-2 TWO'S COMPLEMENT MULTIPLICATION

1. EXTEND RANGE BY REPLICATING THE SIGN BIT OF MULTIPLES
   - PRODUCT HAS $2n$ BITS

2. THE MULTIPLE $xy_{n-1}2^{n-1}$ SUBTRACTED INSTEAD OF ADDED

$$y = -y_{n-1}2^{n-1} + \sum_{i=0}^{n-2} y_i 2^i$$

   - COMPLEMENT AND ADD

3. RECODE THE (2'S COMPLEMENT) MULTIPLIER INTO THE DIGIT SET {-1,0,1}
   - NO ADVANTAGE IN FOLLOWING THIS APPROACH

BIT-MATRIX IN RADIX-2 2'S COMPLEMENT MULTIPLIER

- Simplification of sign extension based on

$$(-s) + 1 - 1 = (1 - s) - 1 = s' - 1$$

Consequently,

$$\begin{array}{lccccc} & x_{n-1}y_i & x_{n-2}y_i & \ldots & x_0y_i \\ \text{is replaced by} & & & & \\ & (x_{n-1}y_i)' & x_{n-2}y_i & \ldots & x_0y_i \\ & \text{-1} & & & \end{array}$$

7	6	5	4	3	2	1	0
x_3y_0	x_3y_0	x_3y_0	x_3y_0	x_3y_0	x_2y_0	x_1y_0	x_0y_0
x_3y_1	x_3y_1	x_3y_1	x_3y_1	x_2y_1	x_1y_1	x_0y_1	
x_3y_2	x_3y_2	x_3y_2	x_2y_2	x_1y_2	x_0y_2		
$x_3'y_3$	$x_3'y_3$	$x_2'y_3$	$x_1'y_3$	$x_0'y_3$			
				y_3			

(a)

7	6	5	4	3	2	1	0
				$(x_3y_0)'$	x_2y_0	x_1y_0	x_0y_0
			$(x_3y_1)'$	x_2y_1	x_1y_1	x_0y_1	
		$(x_3y_2)'$	x_2y_2	x_1y_2	x_0y_2		
	$(x_3'y_3)'$	$x_2'y_3$	$x_l'y_3$	$x_0'y_3$			
				y_3			
0	-1	-1	-1	-1			

(b)

7	6	5	4	3	2	1	0
			y_3	$(x_3y_0)'$	x_2y_0	x_1y_0	x_0y_0
			$(x_3y_1)'$	x_2y_1	x_1y_1	x_0y_1	
		$(x_3y_2)'$	x_2y_2	x_1y_2	x_0y_2		
1	$(x_3'y_3)'$	$x_2'y_3$	$x_l'y_3$	$(x_0y_3)'$			

(c)

Figure 4.7: Constructing bit-matrix for two's complement multiplier ($n = 4$).

RADIX-4 MULTIPLICATION

- REDUCE NUMBER OF STEPS TO $n/2$
- PARALLEL OR SEQUENTIAL RECODING
- TWO CASES
  1. BIT ARRAY ADDED BY A LINEAR ARRAY OF ADDERS
     – SEQUENTIAL RECODING INTO {-1,0,1,2} SUFFICIENT
  2. BIT ARRAY ADDED BY A TREE OF ADDERS
     – PARALLEL RECODING INTO {-2,-1,0,1,2} REQUIRED

PARALLEL RADIX-4 RECODING

- RADIX-2 MULTIPLIER

$$y_{n-1}, y_{n-2} \ldots, y_1, y_0$$

y_i – multiplier bit; $v_j \in \{0, 1, 2, 3\}$ – radix-4 multiplier digit

$$v_j = 2y_{2j+1} + y_{2j} \qquad j = (\frac{n}{2} - 1, ..., 0)$$

- RECODING ALGORITHM

1. Obtain w_j and t_{j+1} such that

$$v_j = w_j + 4t_{j+1}$$

2. Obtain

$$z_j = w_j + t_j$$

- TO AVOID CARRY PROPAGATION:

$$-2 \leq w_j \leq 1 \quad 0 \leq t_{j+1} \leq 1$$

PARALLEL RADIX-4 RECODING ALGORITHM

$$(t_{j+1},\ w_j) = \begin{cases} (0,\ v_j) & \text{if } v_j \leq 1 \\ (1,\ v_j - 4) & \text{if } v_j \geq 2 \end{cases}$$

$$z_j = w_j + t_j$$

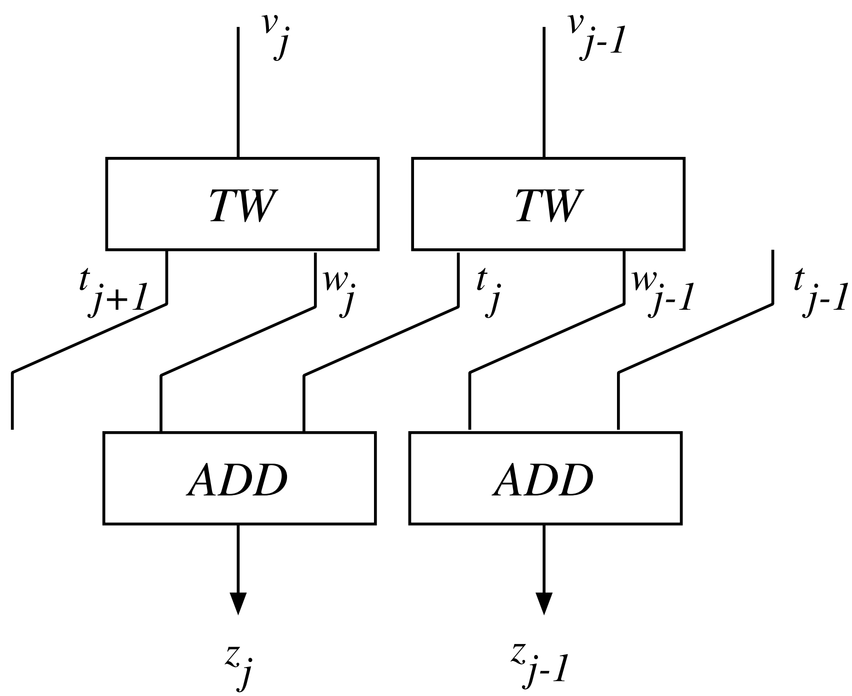


Figure 4.8: RADIX-4 PARALLEL RECODING FROM {0,1,2,3} INTO {-2,-1,0,1,2}.

BIT-LEVEL IMPLEMENTATION

- radix-2 multiplier

$$Y = (y_{n-1}, y_{n-2}, \ldots, y_0) \quad y_i \in \{0, 1\}$$

- recoded radix-4 multiplier

$$Z = (z_{m-1}, z_{m-2}, \ldots, z_0) \quad z_i \in \{-2, -1, 0, 1, 2\}$$

y_{2j+1}	y_{2j}	y_{2j-1}	z_j
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0

EXAMPLES OF RECODING

$$y = 01011110 \quad y = 10001101$$

$$z = 1 \;\; 2 \;\; 0 \;\; \overline{2} \quad z = \overline{2} \;\; 1 \;\; \overline{1} \;\; 1$$

RECODER IMPLEMENTATION

- $sign = 1$ if z_j is negative
- $one = 1$ if z_j is either 1 or -1
- $two = 1$ if z_j is either 2 or -2.

$$\begin{aligned} sign &= y_{2j+1} \\ one &= y_{2j} \oplus y_{2j-1} \\ two &= y_{2j+1}y'_{2j}y'_{2j-1} + y'_{2j+1}y_{2j}y_{2j-1} \end{aligned}$$

- carry-in: $c = sign$

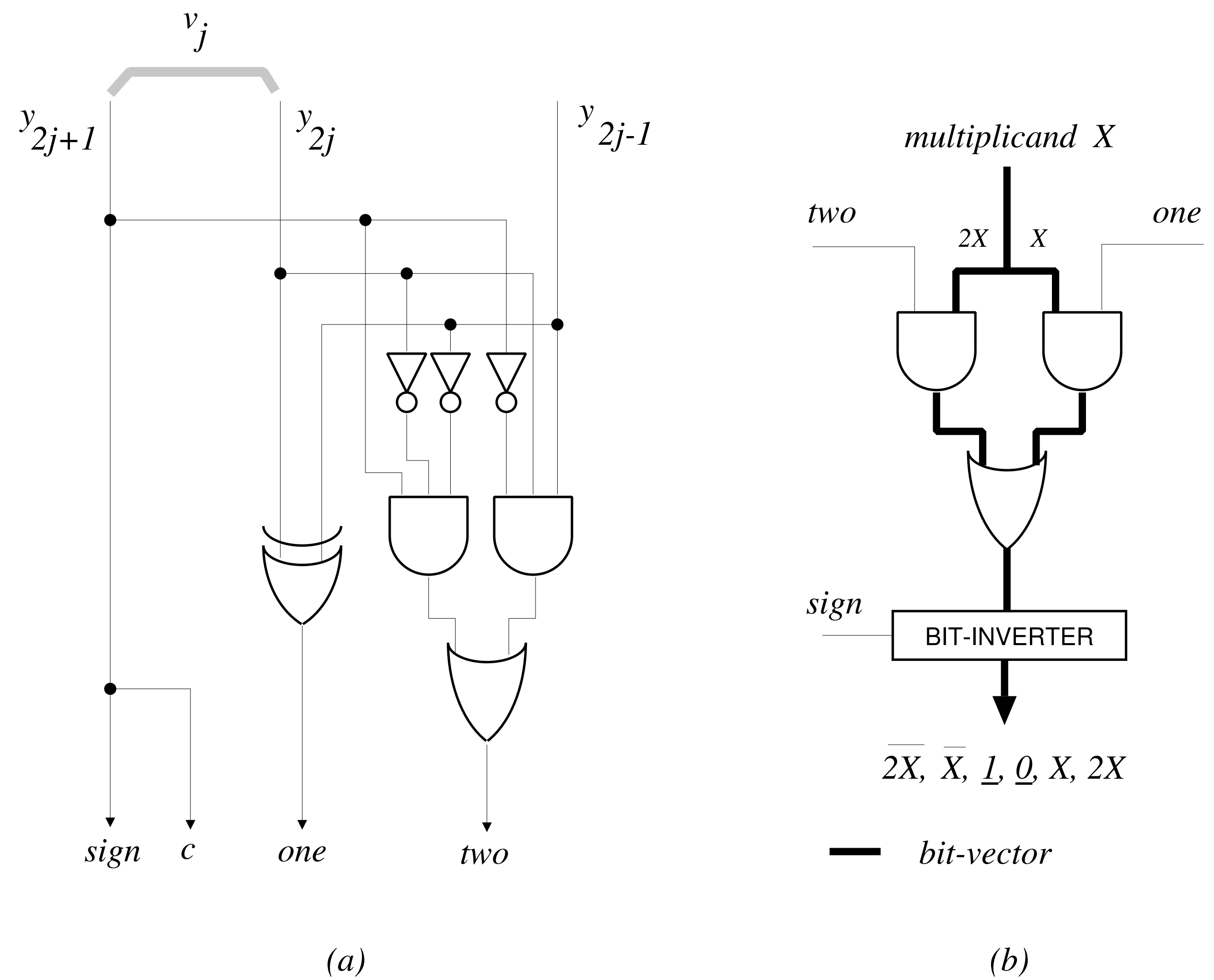


Figure 4.9: (a) IMPLEMENTATION OF RECODER. (b) IMPLEMENTATION OF MULTIPLE GENERATOR.

	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xz_0:	s_e	s_e	s_e	s_e	s_e	s_e	e	e	e	e	e	e	e	e
xz_1:	s_f	s_f	s_f	s_f	f	f	f	f	f	f	f	f		c_e
xz_2:	s_g	s_g	g	g	g	g	g	g	g	g		c_f		
xz_3:	h	h	h	h	h	h	h	h		c_g				

(a)

xz_0:						s'_e	e	e	e	e	e	e	e	e
xz_1:				s'_f	f	f	f	f	f	f	f	f		c_e
xz_2:		s'_g	g	g	g	g	g	g	g	g		c_f		
xz_3:	h	h	h	h	h	h	h	h		c_g				
		-1		-1		-1								

(b)

xz_0:	1		1	s'_e	s_e	s_e	e	e	e	e	e	e	e	e
xz_1:				s'_f	f	f	f	f	f	f	f	f		c_e
xz_2:		s'_g	g	g	g	g	g	g	g	g		c_f		
xz_3:	h	h	h	h	h	h	h	h		c_g				

(c)

Figure 4.10: RADIX-4 BIT-MATRIX FOR MULTIPLICATION OF MAGNITUDES ($n = 7$).

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xz_0:	s_e	s_e	s_e	s_e	s_e	s_e	s_e	s_e	e	e	e	e	e	e	e	e
xz_1:	s_f	s_f	s_f	s_f	s_f	s_f	f	f	f	f	f	f	f	f		c_e
xz_2:	s_g	s_g	s_g	s_g	g	g	g	g	g	g	g	g		c_f		
xz_3:	s_h	s_h	h	h	h	h	h	h	h	h		c_g				
										c_h						

(a)

xz_0:								s'_e	e	e	e	e	e	e	e	e
xz_1:						s'_f	f	f	f	f	f	f	f	f		c_e
xz_2:				s'_g	g	g	g	g	g	g	g	g		c_f		
xz_3:		s'_h	h	h	h	h	h	h	h	h		c_g				
		-1		-1		-1		-1		c_h						

(b)

xz_0:	1		1		1	s'_e	s_e	s_e	e	e	e	e	e	e	e	e
xz_1:						s'_f	f	f	f	f	f	f	f	f		c_e
xz_2:				s'_g	g	g	g	g	g	g	g	g		c_f		
xz_3:		s'_h	h	h	h	h	h	h	h	h		c_g				
										c_h						

(c)

Figure 4.11: RADIX-4 BIT-MATRIX FOR 2'S COMPLEMENT MULTIPLICATION ($n = 8$).

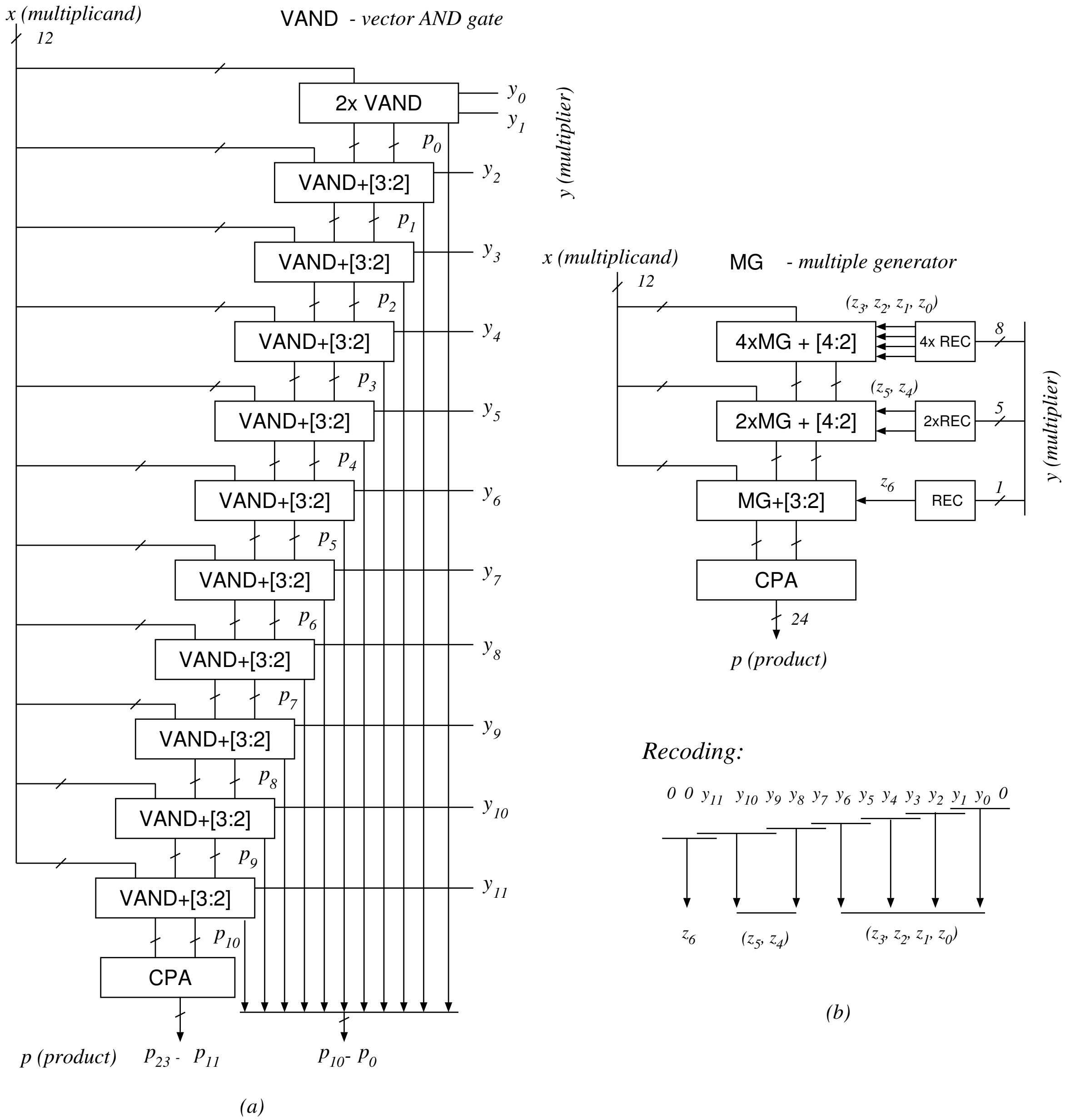


Figure 4.12: LINEAR CSA ARRAY FOR (a) $r = 2$. (b) $r = 4$.

DELAY OF LINEAR ARRAY MULTIPLIERS

- For radix 2,

$$T = t_{AND} + (n-2)t_{fa} + t_{(cpa,(n+1))}$$

- For radix 4

$$T = t_{rec} + t_{AND-OR} + (\frac{n}{2} - 2)t_{fa} + t_{(cpa,n)}$$

REDUCTION BY ROWS: ADDER TREES

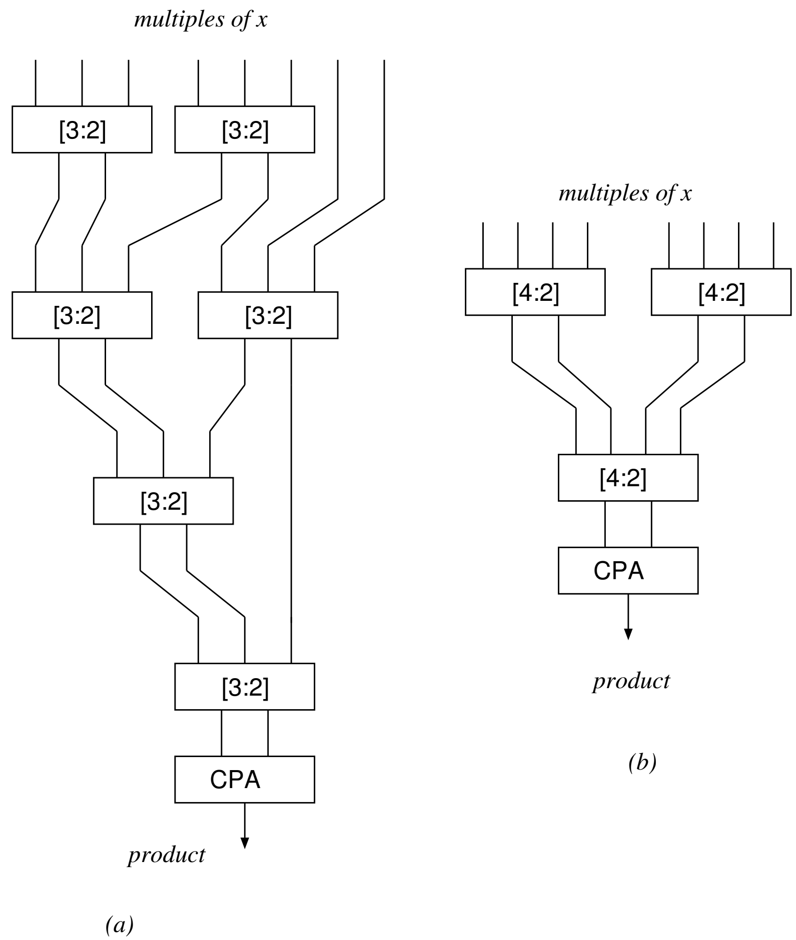


Figure 4.13: TREE ARRAYS OF ADDERS: a) with [3:2] adders. b) with [4:2] adders.

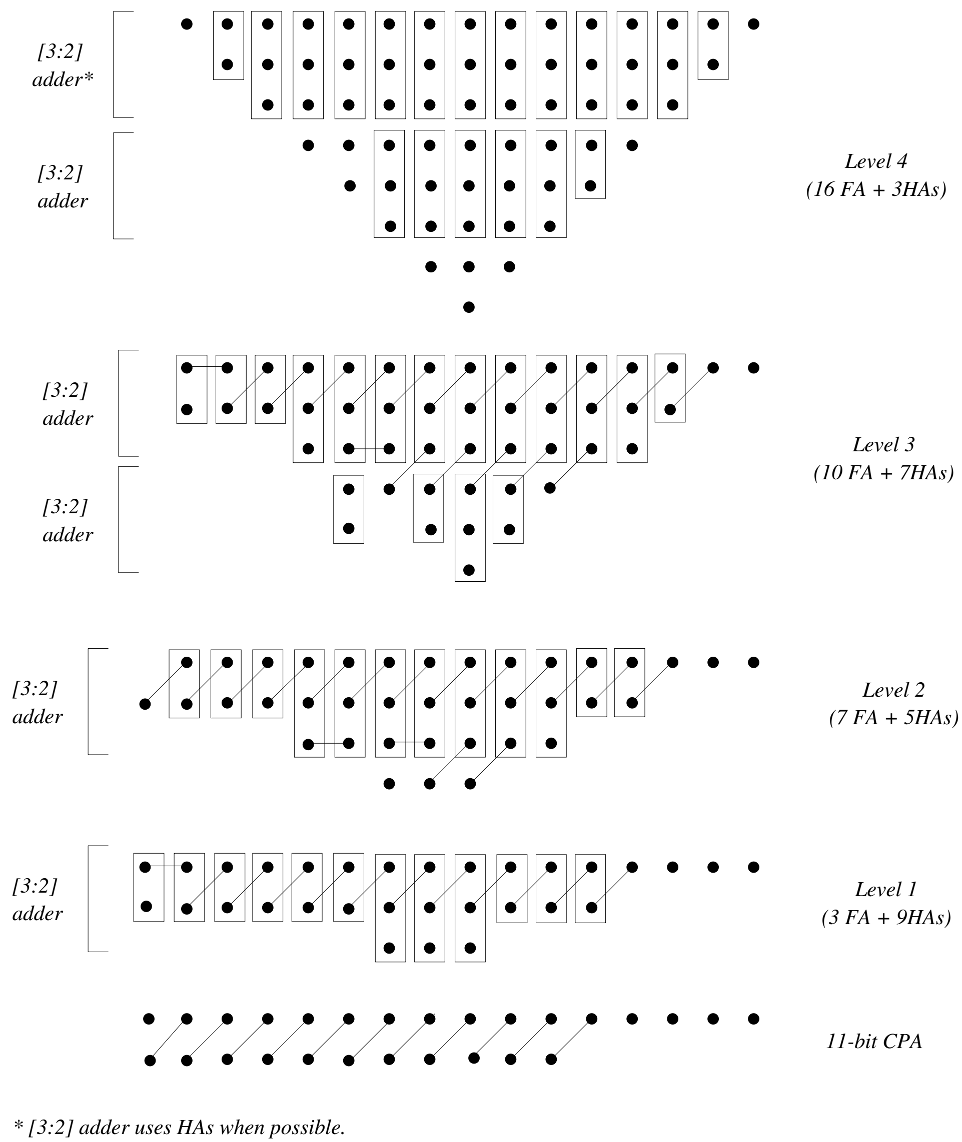


Figure 4.14: REDUCTION BY ROWS USING FAs AND HAs ($n = 8$): Cost 36 FAs, 24 HAs, 11-bit CPA.

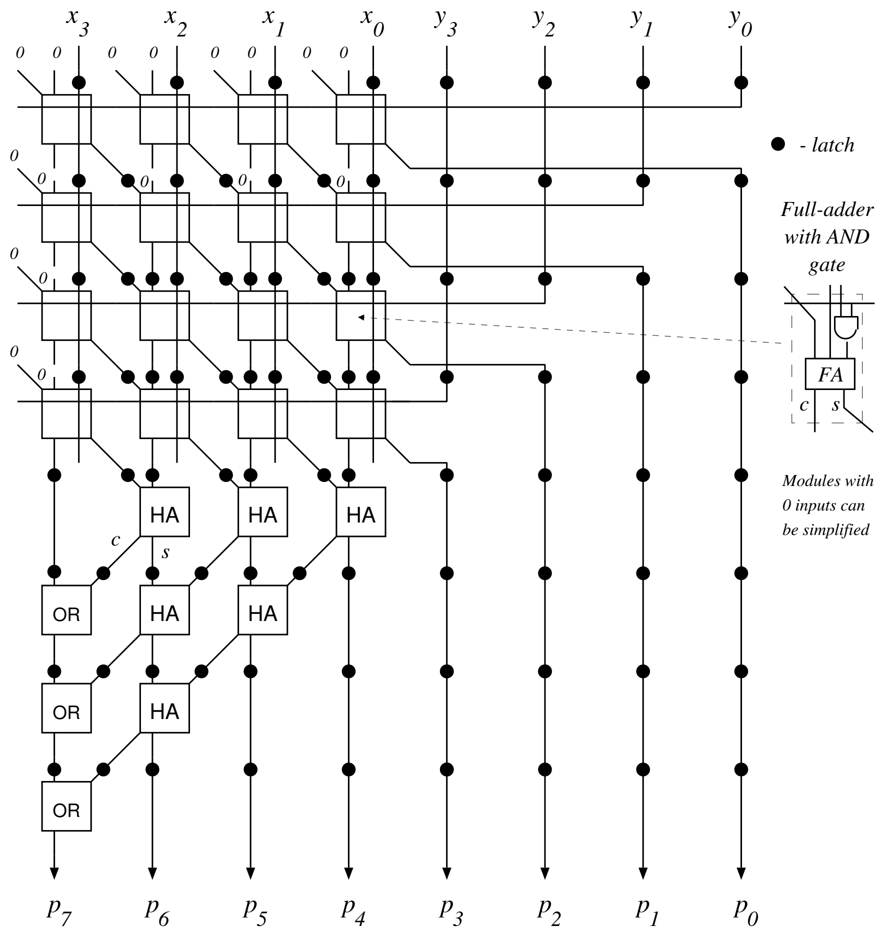


Figure 4.15: PIPELINED LINEAR CSA MULTIPLIER FOR POSITIVE INTEGERS ($n = 4$)

REDUCTION BY COLUMNS USING $(p, q]$ COUNTERS

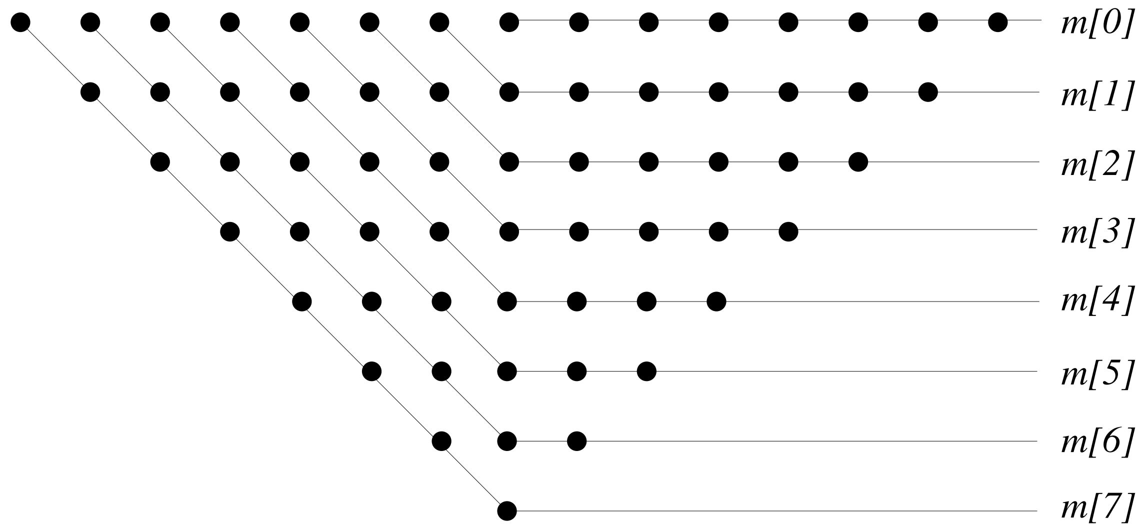


Figure 4.16: BITS OF MULTIPLES ORGANIZED AS BIT-TRIANGLE.

Table 4.3: Reduction by columns using FAs and HAs for 8x8 radix-2 magnitude multiplier.

	i														
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$l = 4$															
e_i	1	2	3	4	5	6	7	8	7	6	5	4	3	2	1
m_3	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
h_i	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
f_i	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
$l = 3$															
e_i	1	2	3	4	6	6	6	6	6	6	5	4	3	2	1
m_2	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
h_i	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
f_i	0	0	0	1	2	2	2	2	2	1	0	0	0	0	0
$l = 2$															
e_i	1	2	4	4	4	4	4	4	4	4	4	4	3	2	1
m_1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
h_i	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
f_i	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0
$l = 1$															
e_i	1	3	3	3	3	3	3	3	3	3	3	3	3	2	1
m_0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
h_i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_i	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0
CPA	2	2	2	2	2	2	2	2	2	2	2	2	1	2	1

e_i is the number of inputs in column i; f_i is the number of FAs; h_i is the number of HAs; m_j is the number of operands in the next level in the reduction sequence.

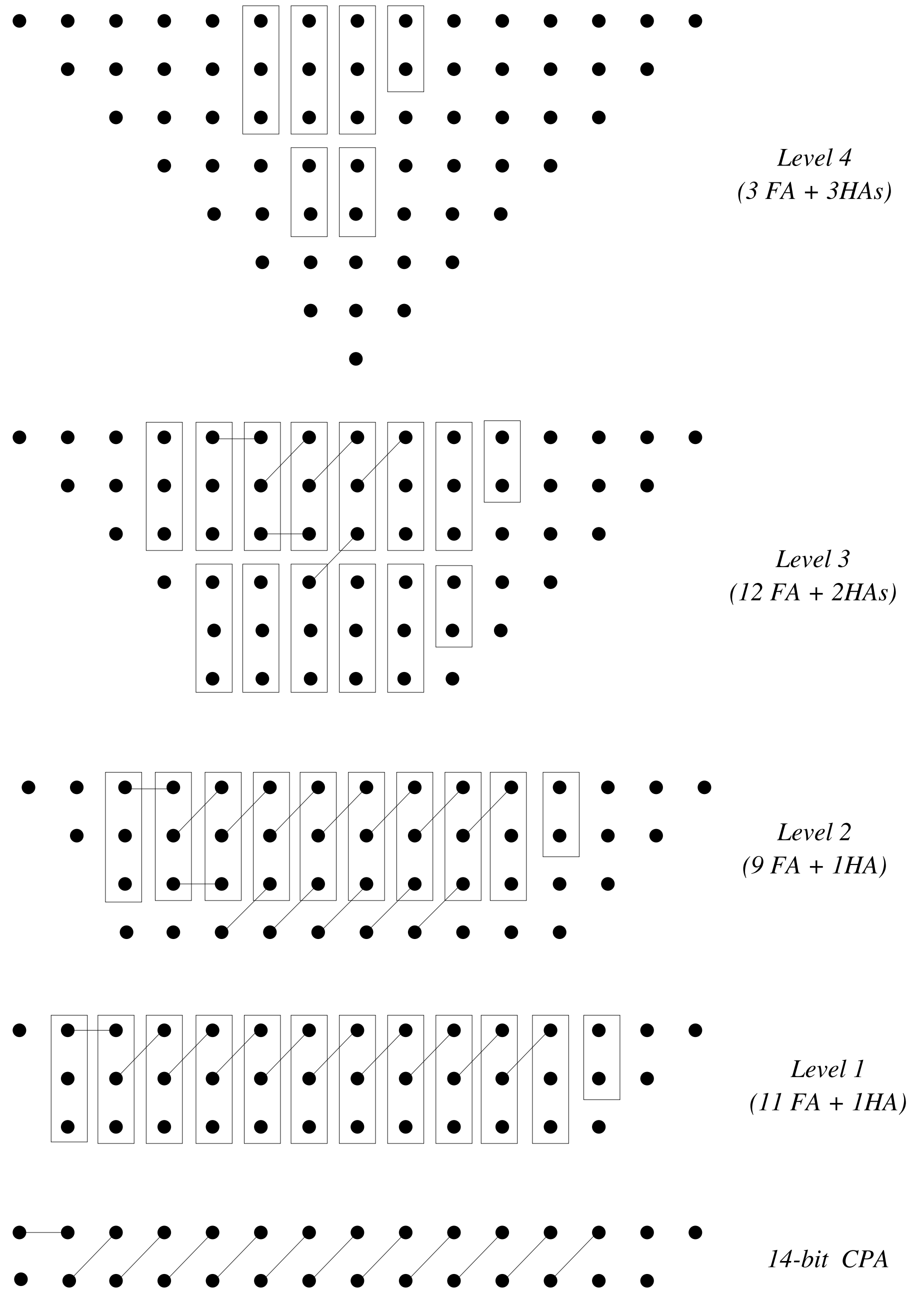


Figure 4.17: REDUCTION BY COLUMNS USING FAs and HAs ($n = 8$): Cost 35 FAs, 7 HAs, 14-bit CPA.

FINAL ADDER

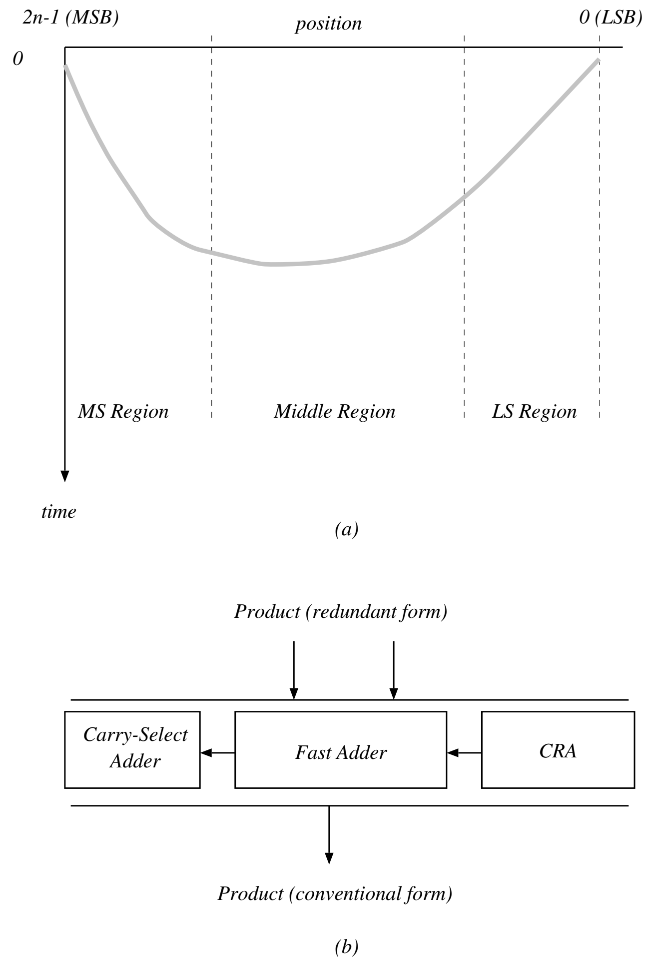


Figure 4.18: Final adder: (a) Arrival time of the inputs to the final adder. (b) Hybrid final adder.

PARTIALLY COMBINATIONAL IMPLEMENTATION

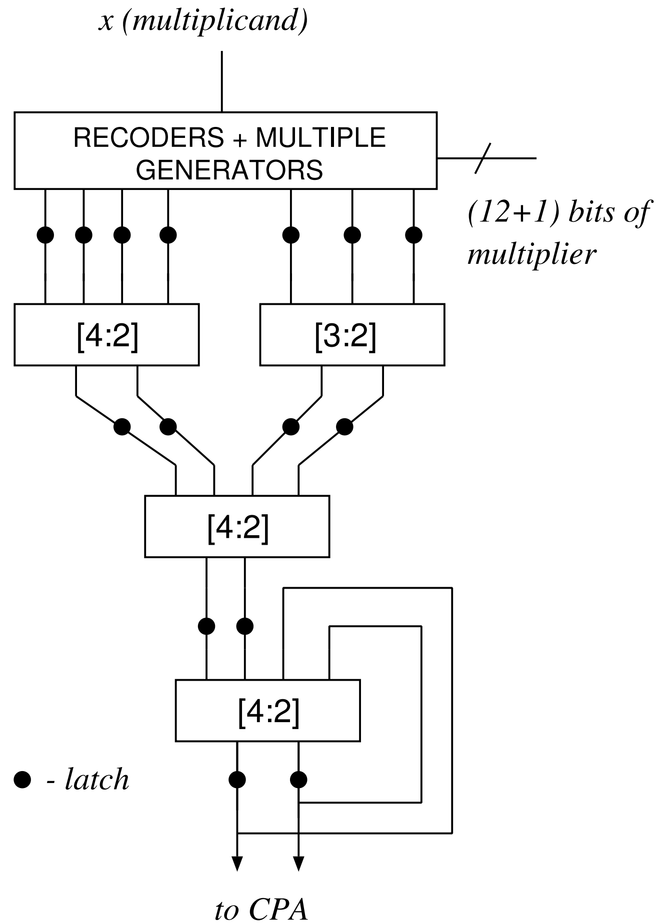


Figure 4.19: RADIX 2^{12} SEQUENTIAL MULTIPLIER USING CSA TREE.

ARRAYS OF SMALLER MULTIPLIERS

$$p = a \times b$$

$$\begin{aligned} A &= (a_{k-1}, a_{k-2}, \ldots, a_0) \\ B &= (b_{l-1}, b_{l-2}, \ldots, b_0) \\ P &= (p_{k+l-1}, p_{k+l-2}, \ldots, p_0) \end{aligned}$$

- USE OF $k \times l$ MODULES
- OPERANDS DECOMPOSED INTO DIGITS OF RADIX 2^k AND 2^l

$$x = \sum_{i=0}^{(n/k)-1} x^{(i)} 2^{ki}$$

$$y = \sum_{j=0}^{(n/l)-1} y^{(j)} 2^{lj}$$

$$\begin{aligned} p = x \cdot y &= \sum_{i=0}^{(n/k)-1} x^{(i)} 2^{ki} \times \sum_{j=0}^{(n/l)-1} y^{(j)} 2^{lj} \\ &= \sum x^{(i)} y^{(j)} 2^{ki+lj} = \sum p^{(i,j)} 2^{ki+lj} \end{aligned}$$

- $(n/k) \times (n/l)$ MODULES NEEDED

EXAMPLE 12×12 USING 4×4 MODULES

$$x = a_x 2^8 + b_x 2^4 + c_x$$
$$y = a_y 2^8 + b_y 2^4 + c_y$$

$$x \times y = a_x a_y 2^{16} + a_x b_y 2^{12} + b_x a_y 2^{12} + a_x c_y 2^8 + b_x b_y 2^8 + c_x a_y 2^8 + b_x c_y 2^4 + c_x b_y 2^4 + c_x c_y$$

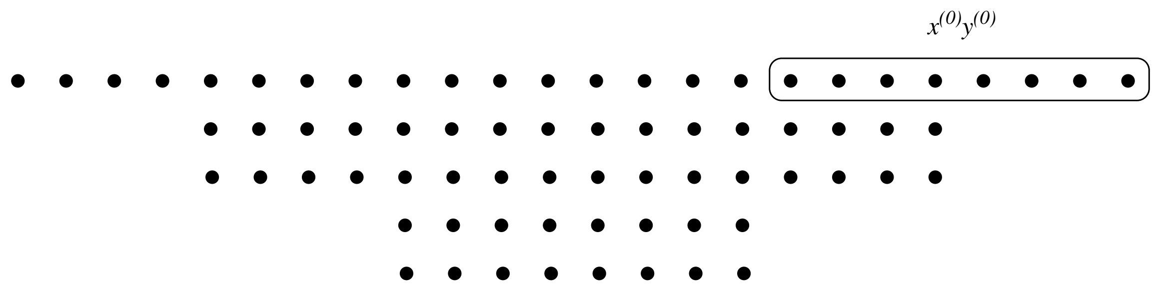


Figure 4.20: 12×12 MULTIPLICATION USING 4×4 MULTIPLIERS: BIT MATRIX.

MULTIPLY-ADD AND MULTIPLY-ACCUMULATE (MAC)

- Multiply-add: $S = X \times Y + W$

	13	12	11	10	9	8	7	6	5	4	3	2	1	0
xz_0:				s'_e	s_e	s_e	e	e	e	e	e	e	e	e
xz_1:			1	s'_f	f	f	f	f	f	f	f	f		c_e
xz_2:	1	s'_g	g	g	g	g	g	g	g	g		c_f		
xz_3:	h	h	h	h	h	h	h	h		c_g				
w:								w	w	w	w	w	w	w

Figure 4.21: Radix-4 bit-matrix for multiply-add of magnitudes ($n = 7$). z_i's are radix-4 digits obtained by multiplier recoding.

- Multiply-accumulate:

$$S = \sum_{i=1}^{m} X[i] \times Y[i]$$

$$S[i+1] = X[i] \times Y[i] + S[i]$$

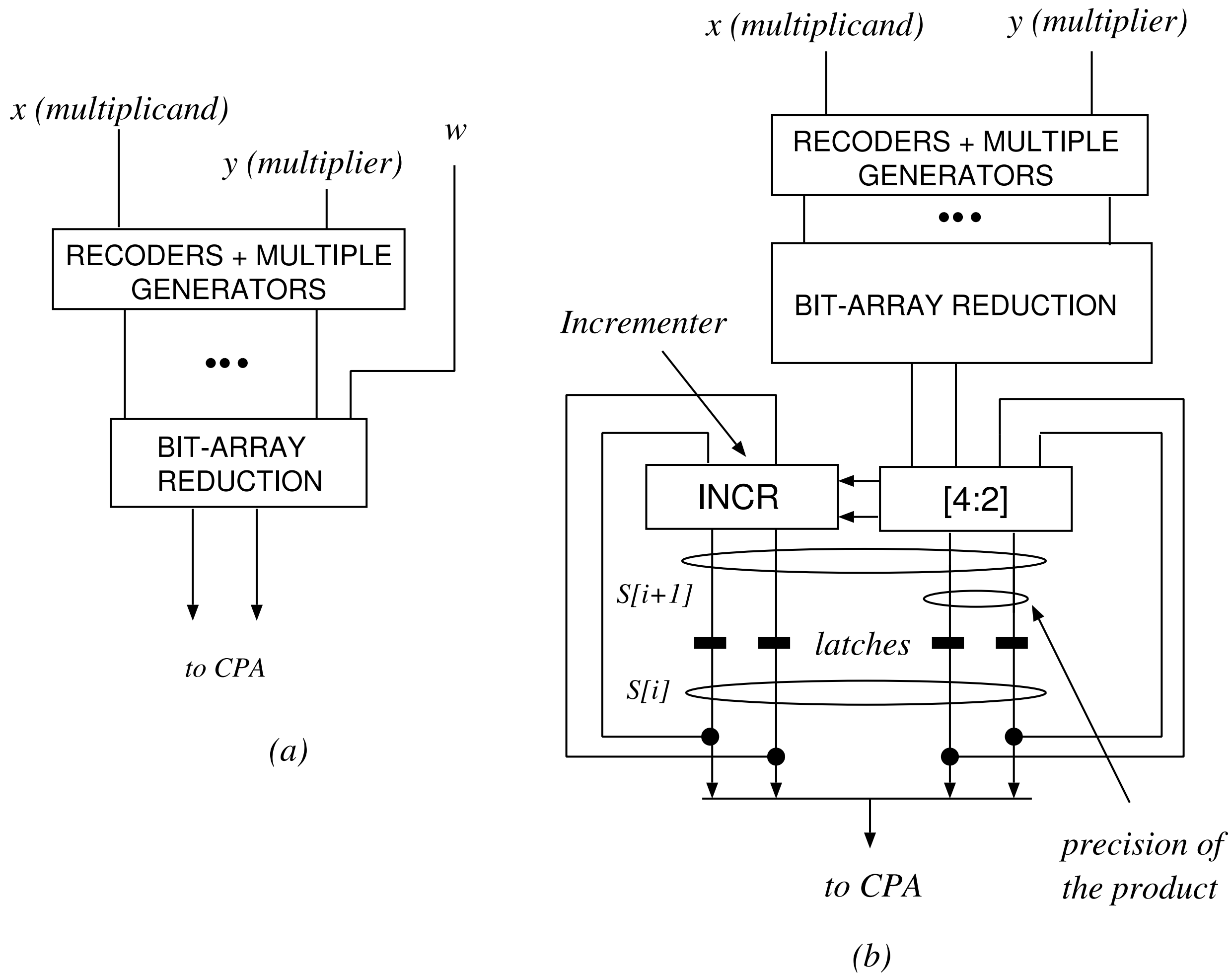


Figure 4.22: Block-diagrams of: (a) Multiply-add unit. (b) Multiply-accumulate unit.

SATURATING MULTIPLIERS

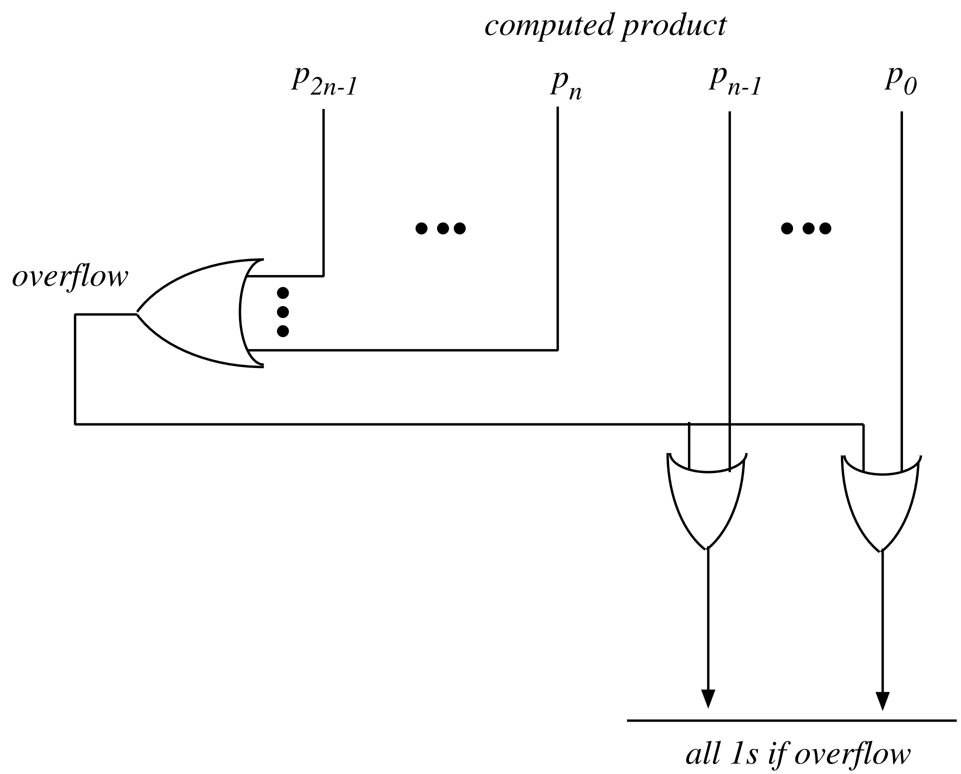


Figure 4.23: Detection and result setting for multiplication of magnitudes.

TRUNCATING MULTIPLIERS

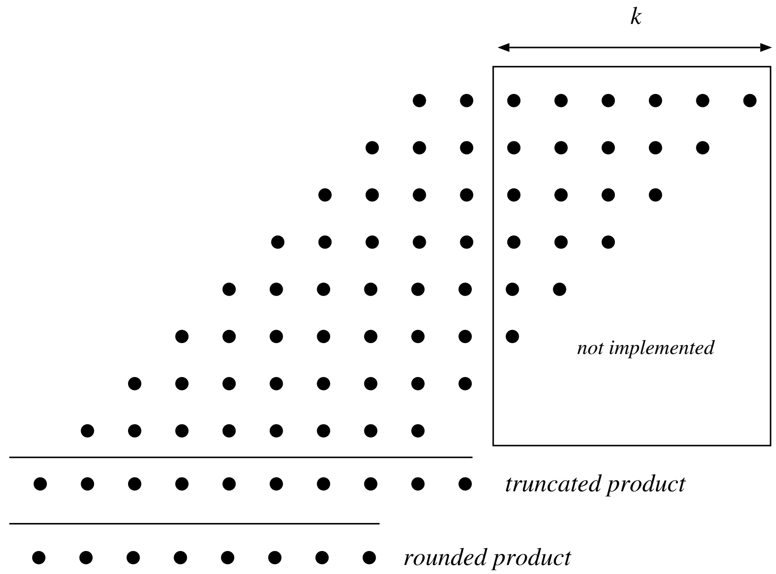


Figure 4.24: Bit-matrix of a truncated magnitude multiplier.

11	10	9	8	7	6	5	4	3	2	1	0
						x_5x_0	x_4x_0	x_3x_0	x_2x_0	x_1x_0	x_0x_0
					x_5x_1	x_4x_1	x_3x_1	x_2x_1	x_1x_1	x_0x_1	
				x_5x_2	x_4x_2	x_3x_2	x_2x_2	x_1x_2	x_0x_2		
			x_5x_3	x_4x_3	x_3x_3	x_2x_3	x_1x_3	x_0x_3			
		x_5x_4	x_4x_4	x_3x_4	x_2x_4	x_1x_4	x_0x_4				
	x_5x_5	x_4x_5	x_3x_5	x_2x_5	x_1x_5	x_0x_5					

11	10	9	8	7	6	5	4	3	2	1	0
	x_5x_4	x_5x_3	x_5x_2	x_5x_1	x_5x_0	x_4x_0	x_3x_0	x_2x_0	x_1x_0		x_0
	x_5		x_4x_3	x_4x_2	x_4x_1	x_3x_1	x_2x_1		x_1		
			x_4		x_3x_2		x_2				
					x_3						

(a)

11	10	9	8	7	6	5	4	3	2	1	0
	x_5x_4	x_5x_3	x_5x_2	x_5x_1	x_5x_0	x_4x_0	x_3x_0	x_2x_0	x_1x_0		x_0
	x_5		x_4x_3	x_4x_2	x_4x_1	x_3x_1	x_2x_1		x_1		
			x_4	x_3x_2	x_3x_2'		x_2				

(b)

Figure 4.25: Bit-array simplification in squaring of magnitudes ($n = 6$).

CONSTANT AND MULTIPLE CONSTANT MULTIPLIERS

$P = X \times C$, C - constant

DONE AS

$P = \Sigma_j X \times C_j 2^j$
$\{j\}$ corresponds to 1's in binary representation of C

- ADDERS ONLY
- HOW TO REDUCE NUMBER OF ADDERS?

1. Recode to radix 4: max $n/2 - 1$ adders
2. Apply canonical recoding: $n/3$ adders avg, $n/2 - 1$ max
3. Decomposition and sharing of subexpressions
4. Multiple constant multiplication - further reductions

CONSTANT MULT. (cont.)

$$45X \ = \ 5X \times 9 = X(2^2+1)(2^3+1)$$

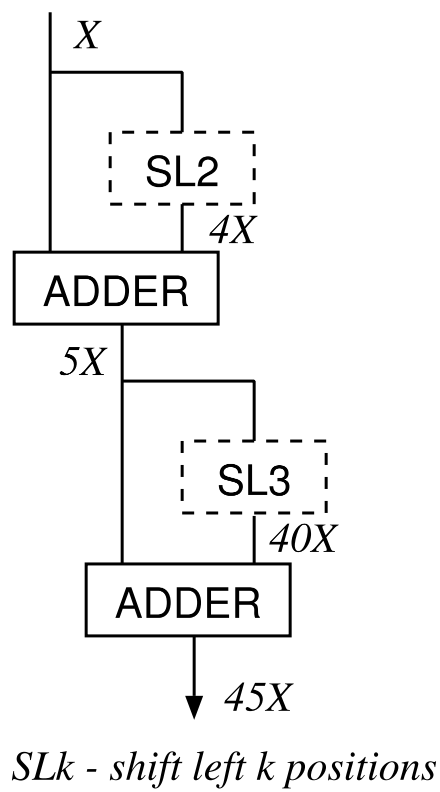


Figure 4.26: Implementation of $P = X \times C$ for $C = 45$ using common subexpressions.

MULTIPLE CONSTANT MULTIPLICATION

COMPUTE

$$P_1 = 9X = 5X + 4X$$
$$P_2 = 13X = 5X + 8X$$
$$P_3 = 18X = 2 \times 9X$$
$$P_4 = 21X = 5X + 16X$$

- WITH SEPARATE CONSTANT MULTIPLIERS: 6 ADDERS
- BY SHARING SUBEXPRESSIONS: 4 ADDERS

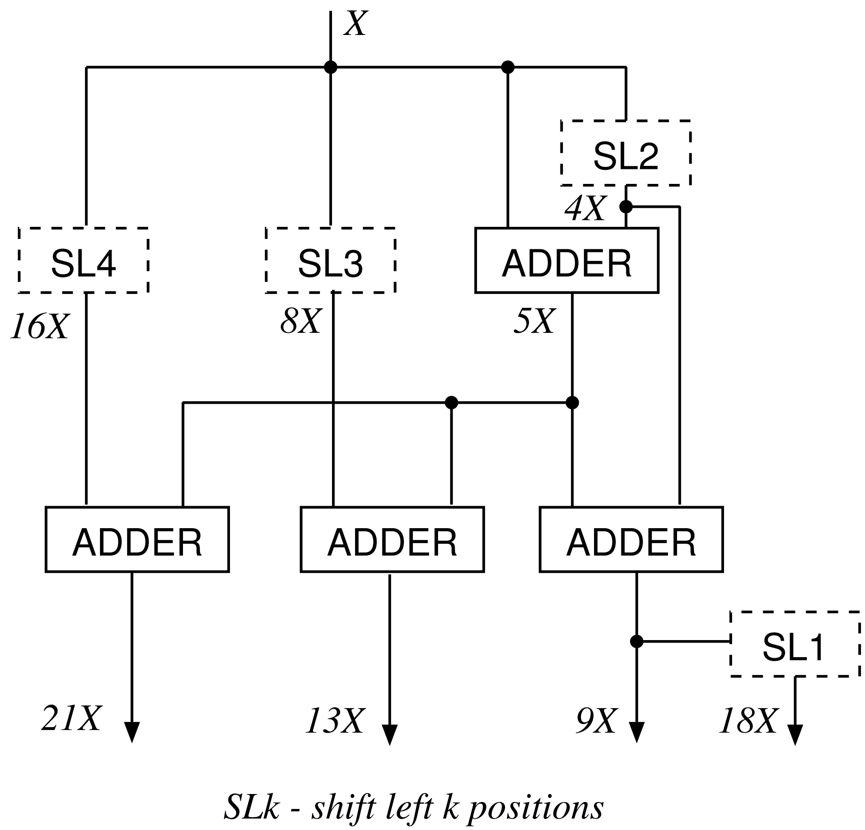


Figure 4.27: An example of multiple constants multipliers.