DIGITAL ARITHMETIC
Miloš D. Ercegovac and Tomás Lang
Morgan Kaufmann Publishers, an imprint of Elsevier Science, ©2004
– Updated: February 13, 2004 –

Chapter 4: Solutions to Selected Exercises

– With contributions by Elisardo Antelo –

Exercise 4.1

x= 30 X = 011110
y = -25 Y = 100111 Z= (-2)2(-1)

	CSA	shifted out		
$PS[0]$	00000000			
$SC[0]$	00000000			
xZ_0	11100001			
$4PS[1]$	11100001			
$4SC[1]$	0000000**1**			
$PS[1]$	11111000	10		
$SC[1]$	00000000			
xZ_1	00111100			
$4PS[2]$	11000100			
$4SC[2]$	01110000			
$PS[2]$	11110001	0010		
$SC[2]$	00011100			
xZ_2	11000011			
$4PS[3]$	00101110			
$4SC[3]$	1010001**1**			
$PS[3]$	00001011	010010		(cin=1)
$SC[3]$	11101000			
P	110100	010010	=	-750

From Figure 4.4 we determine that the number of cycles to obtain $PS[3], PC[3]$ is 6 (including one cycle to load X and Y).

In the last pass through the pipeline the register values are :

Register X = 011110 Register Y = ...10 Register C=0
Register XY = 11000100
Register SCH = 11101000 Register PSH=00001011
Register CS[1,0]=(10,11) Register PL = 0010

Exercise 4.3

To reduce the effect on the cycle time, the outputs of the carry-save adder are latched before being used as inputs to the converter. The input/output arithmetic relation is

$$2(PS_1[j-1] + SC_1[j-1]) + (PS_0[j-1] + SC_0[j-1] + w_0[j-1])$$
$$= 4w_0[j] + 2p_{2j+1} + p_{2j}$$

where $w[0]$ is the state. Since $0 \le 2(PS_1[j-1] + SC_1[j-1]) + (PS_0[j-1] + SC_0[j-1] \le 6$ and $0 \le 2p_{2j+1} + p_{2j} \le 3$ we get $0 \le w_0 \le 1$.

This is implemented with a 2-bit adder with $w_0[j-1]$ as the carry-in and $w_0[j]$ as the carry-out. The corresponding delay is $T_{conv} = t_{ab-c} + t_{c-c}$ which is somewhat larger than t_{ab-s} of the CS adder.

To keep the cycle time at t_{ab-s} as determined by the CSA, the scheme requires additional pipelining. The latency of the converter pipeline should not exceed the latency of the CPA used to obtain the MS bits of the product.

Exercise 4.5

A two's complement sequential multiplier with operands X and Y of 16 bits is designed similarly to the sequential multiplier in Figure 4.3. Note that the scheme in Figure 4.3 uses positive n-bit operands. This requires extension by two bits to handle negative multiples in radix 4. In this exercise, the operands are in the two's complement, thus one bit extension is suffucient. To reduce the cycle time, the design is pipelined (Figure E4.5a).

The delay and area of components are obtained with respect to NAND-2 using Tables 2.4 and 5.4 and summarized next

	delay	area
NOT	0.7	1
NAND-3	1.2	2
NOR-3	1.7	2
NOR-2	1.1	1
XOR	1.7	3
buffer	1.8	2.6
MUX-2	1.4	3
FA	4.2	6.7
flip-flop	4	4

The modules are

- Stage 1: Radix-4 recoder

  The sequential recoder for magnitudes described on p.185 and implemented in Fig. 4.5 produces radix-4 digits in the set {-1,0,1,2}. Since the multiplier in this exercise is in the two's complement system, the most significant radix-4 digit

$$z_7 = -2y_{15} + y_{14} + c_7$$

  is in the set {-2,-1,0,1,2}.

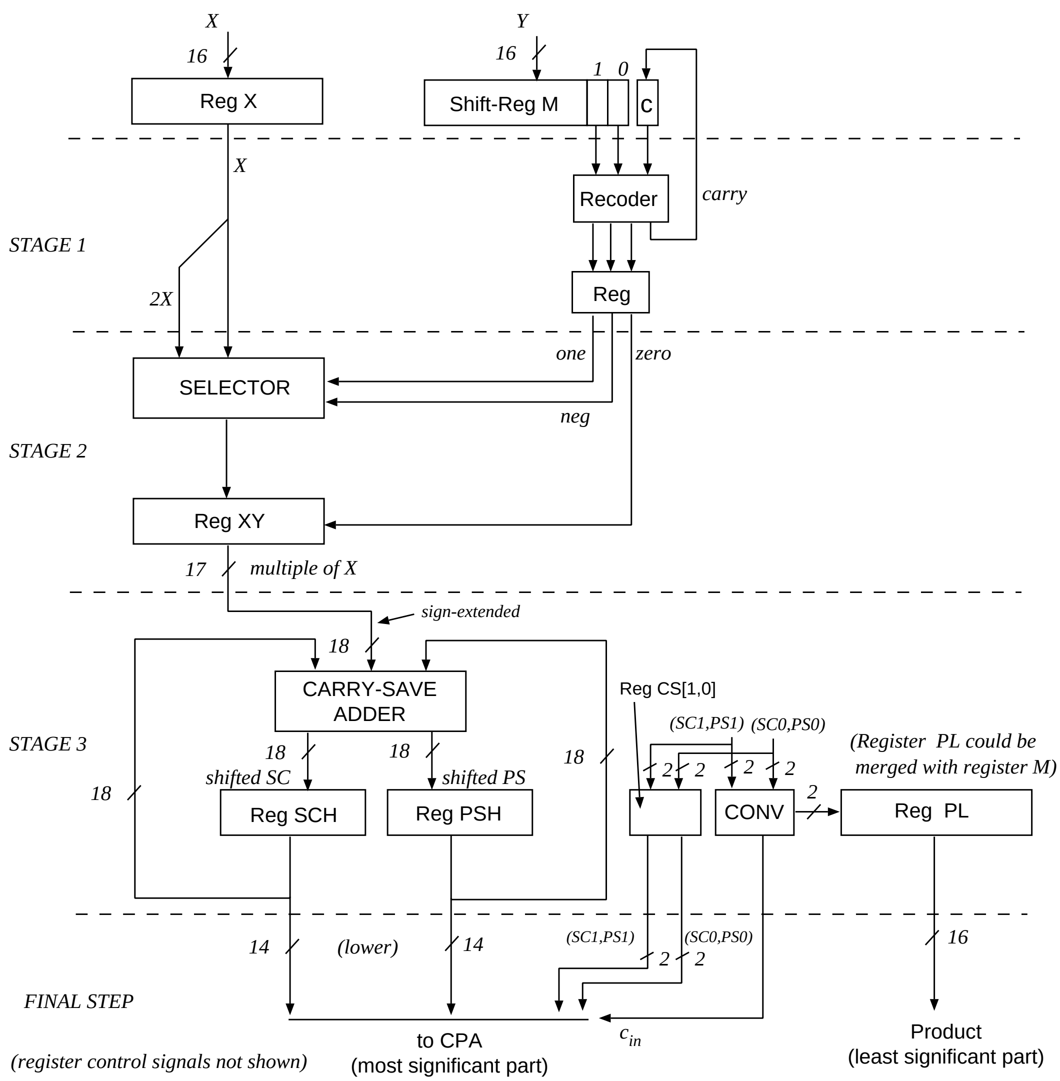

Figure E4.5a: 16-bit two's complement sequential multiplier. (Exercise 4.5)

The recoder of Fig. 4.5 is modified to produce a (-2) when $M1 = 1$, $M0 = 0$ and $C = 0$ in the cycle when z_7 is produced ($last = 1$). This results in a modified expression for neg while one, $zero$, and C_{next} remain unchanged:

$$neg = M1C + M1M0 + last \cdot M1M0'C' = M1(C + M0 + last \cdot M0'C')$$
$$= M1(C + M0 + last)$$

The modified recoder is shown in Figure E4.5b.

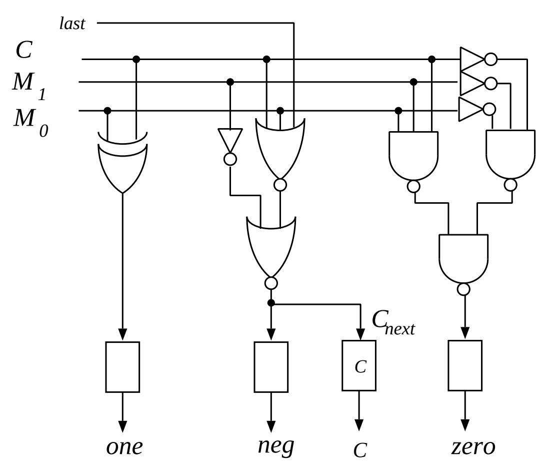

Figure E4.5b: Radix-4 recoder. (Exercise 4.5)

The delay and area of the recoder are:

	delay	area
1 XOR	1.7	3
2 NAND-3	1.2	4
2 NAND-2	1	2
1 NOR-3	1.7	2
1 NOR-2	1.1	1
3 NOT	0.7	3
4 FF	4	16
Total	2.9 +4	31

- Stage 2: Multiple generator

  The multiples $\pm 2 \times X$, $\pm 1 \times X$, and $0 \times X$ are obtained as shown in Figure E4.5c.

  The delay and area of the multiple generator are:

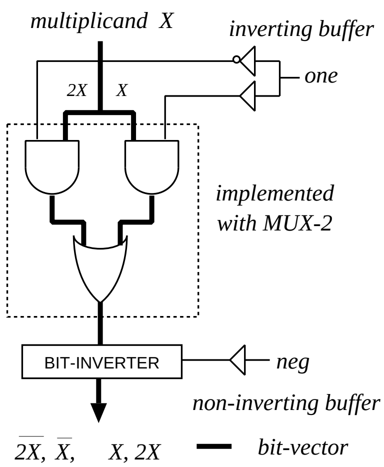

Figure E4.5c: Multiple generator. (Exercise 4.5)

	delay	area
3 BUFF	1.8	7.8
18 MUX-2	1.4	54
18 XOR	1.7	54
18 FF	4	72
Total	4.9+4	$\approx$ 188

- Stage 3: CSA

  The CSA adder consists of 19 FAs. The carry and sum are stored in two 19-bit registers SCH and PSH. The delay and area are:

	delay	area
19 FA	4.2	127.3
2x19 FF	4	152
Total	4.2+4	$\approx$ 280

The converter uses two FAs. To reduce the critical path, the 2-bit adder is pipelined so that only one FA is in the critical path. Four extra FFs are needed for pipelining. There is also a 16-bit register PL which stores the least-significant 16 bits of the product. The cycle time of the converter is 4.2 +4 = 8.2. Its area is $2 \times 6.7 + 8 \times 4 \approx 45$. For PL register the area is $16 \times 4 = 64$.

The cycle time of the multiplier is determined by the delay of Stage 2: 8.9 NAND-2 delays. To reduce this delay, a faster multiple generator could be designed using a 4-to-1 multiplexer to select ± 2 and ± 1 multiples. This would also require a change in the recoder design. The total area uses 544 equivalent gates.

Exercise 4.8

- The cycle time of a radix-2 multiplier is

  $$t_2 = t_{buf} + t_{NAND} + t_{c-s} + t_{reg}$$

  Using the values from Figure 5.4 we get

  $$t_2 = 1.8 + 1 + 2.2 + 4 = 9t_{NAND}$$

- To reduce the cycle time of the radix-16 implementation we pipeline as shown for radix 4 in Figure 4.3. The cycle time is the maximum of the critical paths of the three stages. We assume it is the adder, implemented as a [4:2] adder (Figure 2.41). Consequently, the cycle time is

  $$t_{16} = t_{[4:2]} + t_{reg}$$

  Using the values from Figure 5.4 we get

  $$t_{16} = 6 + 4 = 10t_{NAND}$$

- The total delay corresponds to the iterations (n for radix 2 and $n/4$ for radix 16) plus the two pipeline cycles for radix 16, plus the delay of the final adder). The speedup is

  $$S = \frac{t_2 \times n + t_{CPA}}{t_{16} \times (2 + n/4) + t_{CPA}} = \frac{36n + 4t_{CPA}}{10n + 80 + 4t_{CPA}}$$

- As seen in the expression, the speedup depends on n. This is because of the two additional cycles in radix 16 and of the carry-propagate adder.

  For instance, for $n = 16$ and using a carry-ripple adder we get

  $$S = \frac{36 \times 16 + 4(2.0 \times 16}{10 \times 16 + 80 + 128} = 1.9$$

Exercise 4.11

a) Radix-4 bit-matrix for multiplication of magnitudes with $x = 67$ and $y = 76$ is shown next. The recoded radix-4 multiplier is (11(-1)0).

13	12	11	10	9	8	7	6	5	4	3	2	1	0
1		1	1	0	0	0	0	0	0	0	0	0	0
			0	1	0	1	1	1	1	0	0		0
	1	0	1	0	0	0	0	1	1		1		
0	1	0	0	0	0	1	1		0				
0	1	0	0	1	1	1	1	1	0	0	1	0	0

The result checks: $x \times y = 5092$.

b) Radix-4 bit-matrix for multiplication of 2's complement operands $x = -67$ and $y = -76$. The recoded radix-4 multiplier is ((-1)(-1)10).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1		1		1	1	0	0	0	0	0	0	0	0	0	0
					0	1	0	1	1	1	1	0	1		0
			1	0	1	0	0	0	0	1	0		0		
	1	0	1	0	0	0	0	1	0		1				
									1						
0	0	0	1	0	0	1	1	1	1	1	0	0	1	0	0

The result checks: $x \times y = 5092$.

Exercise 4.13

The reduced bit-matrix for radix-4 multiplication of magnitudes with $n = 12$, corresponding to Figure 4.14(b) is shown in Figure E4.13(a). The linear array has three stages.

- Stage 1 consists of a [4:2] adder and converter K1. The inputs to the converter in Stage 1 are denoted with "k".
- Stage 2 also has a [4:2] adder and converter K2.
- Stage 3 uses a [3:2] adder and a converter.

The partial inputs to Stage 2 and Stage 3 are shown in Figure E4.13(b) and (c), respectively. Each converter produces a conventional radix-4 digit ({0,1,2,3}) and a carry.

- Converter K1 consists of two HAs and its delay is clearly shorter than that of a [4:2] adder.
- Converter K2 uses one FA and one HA, again having a delay not greater than that of a [4:2] adder.
- Converter in Stage 3 could also use one FA and one HA. However, its delay would be longer than $t_{[3:2]} = t_{FA}$. To reduce its delay, bits denoted with "c" are used to produce two conditional 3-bit results (carry + 2 sum bits) in Stage 2. The delay of a 2-bit conditional adder (CA) is not larger than the delay of [4:2] adder. The correct sum is obtained using a MUX in Stage 3 based on the carry produced by converter K2 in Stage 2. This MUX has a shorter delay than a FA. Therefore, conversion of the least-significant radix-4 redundant digits does not increase the delay in the critical path.

Since in each stage two bits of the product are obtained, the final adder has 24 - 6 = 18 bits.

```
                                                    [4:2]  K1
                                                     ---  ----
1   1   1   1   1 s' s  s  x  x  x  x  x  x  x  x  x  x| k  k
                  s' x  x  x  x  x  x  x  x  x  x  x  x|    k
               s' x  x  x  x  x  x  x  x  x  x  x     x|
            s' x  x  x  x  x  x  x  x  x  x  x     x  ___|
         s' x  x  x  x  x  x  x  x  x  x  x     x
      s' x  x  x  x  x  x  x  x  x  x  x     x
      x  x  x  x  x  x  x  x  x  x  x  x  x
```

(a)

```
                                        [4:2]  CA    K2
                                         ---  ----  ----
                  .  .  .  .  x  x  x  x  x  x| c  c  x  x  p  p
                  .  .  .  .  x  x  x  x  x  x| c  c  x  k
            .  .  .  .  .  .  x  x  x  x     x|
         .  .  .  .  .  .  .  x  x     x   ___|
```

(b)

```
                                     [3:2]
                                      ---
                  .  .  .  .  .  x  x  x  x| x        p  p  p  p
                  .  .  .  .  x  x  x  x  x|       k| MUX control
            .  .  .  .  .  x  x  x     ___| c  c  c| MUX data
                                            c  c  c| MUX data
                                            -------
                                              MUX
```

(c)

```
                          CPA
                  ---------------------------
                  .  .  .  .  x  x  x  x  x  x  p  p  p  p  p  p
                  .  .  .  .  x  x  x  x  x  c
```

(d)

Figure E4.13: A linear array of [4:2] and [3:2] adders for 12×12 multiplication of magnitudes: (a) Reduced bit-matrix. (b) Inputs to Stage 2. (c) Inputs to Stage 3. (d) Inputs to CPA.

Exercise 4.15

Tables to determine the number of full and half adders in column reduction for multiplication of 8-bit operands for the following cases are:

(a) Radix-2 operands in two's complement representation, $n = 8$

Bit-matrix:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	$(x'_7y_7)'$	$(x_7y_6)'$	$(x_7y_5)'$	$(x_7y_4)'$	$(x_7y_3)'$	$(x_7y_2)'$	$(x_7y_1)'$	$(x_7y_0)'$	x_6y_0	x_5y_0	x_4y_0	x_3y_0	x_2y_0	x_1y_0	x_0y_0
		x'_6y_7	x_6y_6	x_6y_5	x_6y_4	x_6y_3	x_6y_2	x_6y_1	x_5y_1	x_4y_1	x_3y_1	x_2y_1	x_1y_1	x_0y_1	
			x'_5y_7	x_5y_6	x_5y_5	x_5y_4	x_5y_3	x_5y_2	x_4y_2	x_3y_2	x_2y_2	x_1y_2	x_0y_2		
				x'_4y_7	x_4y_6	x_4y_5	x_4y_4	x_4y_3	x_3y_3	x_2y_3	x_1y_3	x_0y_3			
					x'_3y_7	x_3y_6	x_3y_5	x_3y_4	x_2y_4	x_1y_4	x_0y_4				
						x'_2y_7	x_2y_6	x_2y_5	x_1y_5	x_0y_5					
							x'_1y_7	x_1y_6	x_0y_6						
							y_7	$(x_0y_7)'$							

Reduction table:

	15	14	13	12	11	10	9	8	i 7	6	5	4	3	2	1	0
$l=4$																
e_i	1	1	2	3	4	5	6	8	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	6
h_i		0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
f_i		0	0	0	0	0	1	2	1	0	0	0	0	0	0	0
$l=3$																
e_i	1	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	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	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	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	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	2
h_i		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
f_i		0	1	1	1	1	1	1	1	1	1	1	1	0	0	0
CPA	2	2	2	2	2	2	2	2	2	2	2	2	2	2	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.

(b) Radix 4, magnitudes, multiplier recoding, $n = 7$

Bit-matrix:

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

Reduction table:

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

(c) Radix 4, two's complement, multiplier recoding, $n = 8$

Bit-matrix:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	s'_h	1	s'_g	1	s'_e	s_e	s_e	e	e	e	e	e	e	e	e
		h	h	g	s'_f	f	f	f	f	f	f	f	f		c_e
				h	g	g	g	g	g	g	g		c_f		
					h	h	h	h	h		c_g				
									c_h						

Reduction table:

i	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$l = 3$																
e_i	1	1	2	2	3	4	4	4	4	5	3	4	2	3	1	2
m_2	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
h_i		0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
f_i		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$l = 2$																
e_i	1	1	2	2	4	4	4	4	4	4	3	4	2	3	1	2
m_1	3	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	1	1	0	0	0	0
f_i		0	0	0	1	1	1	1	1	1	0	0	0	0	0	0
$l = 1$																
e_i	1	1	2	3	3	3	3	3	3	3	3	3	2	3	1	2
m_0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
h_i		0	1	0	0	0	0	0	0	0	0	0	1	1	0	0
f_i		0	0	1	1	1	1	1	1	1	1	1	0	0	0	0
CPA	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Exercise 4.20

(a) The precision of S is 18 because $2^{17} < 127^2 * 16 < 2^{18}$.

(b) Since one pair of elements is available per cycle, a suitable algorithm is

$$S[i] = S[i-1] + A[i]B[i]$$

with $S = S[16]$ and $S[0] = 0$.

The recoding of $B[i]$ produces radix-4 digits. The resulting pipelined linear array with [3:2] adders is shown in Figure E4.20b.

(c) The cycle time is $t_{cycle-b} = max(t_{REC} + t_{buf} + t_{mux}, 2t_{FA})$

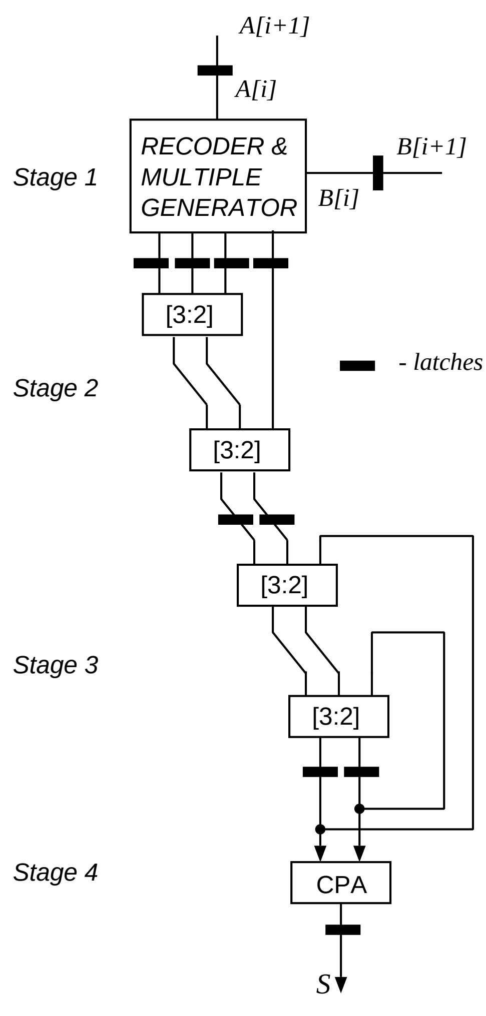

Figure E4.20b: A linear array of [3:2] adders for Exercise 4.20(b).

(d)

```
   1      2      3      4      5      6              19     20
|-----|-----|-----|-----|-----|-----| . . . . |-----|-----|
compute                S[1]   S[2]   S[3]           S[16]
output                        S[1]   S[2]     . . .        S[16]
```

The latency is $T = 3 + 16 + 1 = 20$ clock cycles.

(e) A pipelined linear array with[4:2] adders is shown in Figure E4.20e.

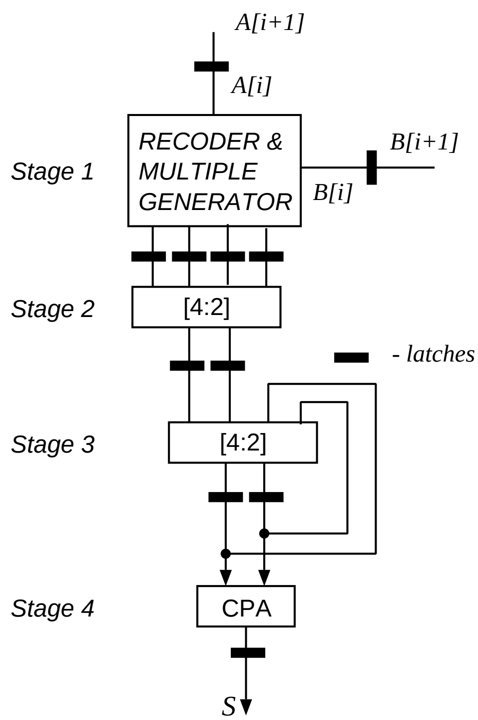

Figure E4.20e: A linear array of [4:2] adders for Exercise 4.20(e).

$$t_{cycle-e} = max(t_{REC} + t_{buf} + t_{mux}, t_{4-2})$$

Comparing with the linear array of part (b): The cycle time is the same if $t_{cycle-e} = t_{REC} + t_{buf} + t_{mux}$. Otherwise it depends on implementation of the [4:2] adder. If implemented with two [3:2] adders, there is no difference. If a gate network is used in implementing [4:2] module with a delay smaller than $2t_{FA}$, this implementation would have a shorter cycle time.

Exercise 4.26

The constant $C = 2925 = 0101101101101$ requires 8 additions.

Using canonical recoding we get C as $2925 = 10\bar{1}00\bar{1}00\bar{1}0\bar{1}01$ which requires 6 additions/subtractions.

Using factoring we get C as $2925 = (4+1)(8+1)(64+1) = (2^2+1)(2^3+1)(2^6+1)$ which requires 3 additions.

We use the factoring approach. The two designs are shown in Figure E4.26.

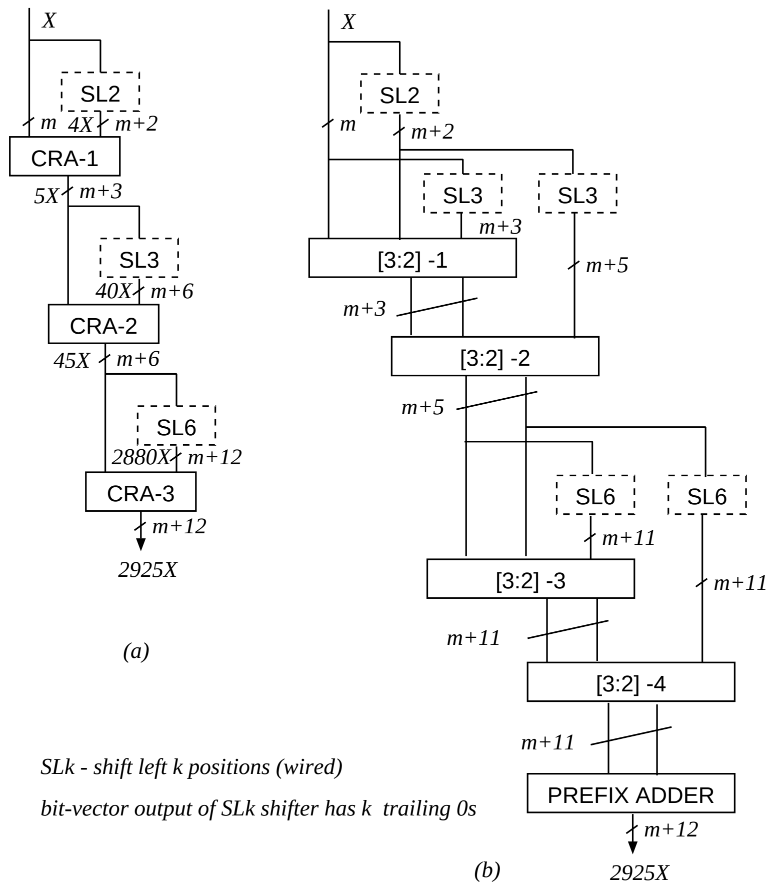

Figure E4.26: Constant multiplier networks: (a) With CRAs. (b) With [3:2] and prefix adder. (Exercise 4.26).

- Implementation with CRAs. To determine delay consider the following input/output diagram. FA and HA are denoted with "f" and "h". All delays are in terms of t_{FA}, and $t_{HA} = 0.5t_{FA}$ (same for sum and carry outputs). We show $m = 8$ in the diagram and generalize the result to arbitrary m.

```
                            xxxxxxxx
                          xxxxxxxx
CRA-1                     hhfffffh
                         xxxxxxxxxxx
                      xxxxxxxxxxx
CRA-2                 hhhfffffffh
                      xxxxxxxxxxxxxx
                xxxxxxxxxxxxxx
CRA-3           hhhhhhfffffffh
                xxxxxxxxxxxxxxxxxxxx
```

The critical path is: h+f+h+f+f+f+h+(fx(m-1))+h+h+h+h+h+h resulting in

$$T_{CRA} = 9t_{HA} + (m+3)t_{FA} = (m+7.5)t_{FA}$$

The equivalent number of full adders is:

$$C_{CRA} = (m-3)FA + 3HA + (m-1)FA + 4HA + (m-1)FA + 7HA$$
$$= 14HA + (3m-2)FA \approx (3m+5)FA$$

- Implementation with [3:2] adders and prefix adder.

  We determine the delay in the critical path and the cost as in the case with CRAs. To reduce the precision of the final adder, we apply [2:1] reduction where applicable.

```
                            xxxxxxxx
                          xxxxxxxx
                         xxxxxxxx
[3:2]-1                   hhfffffh
                         xxxxxxxxxxx
                         xxxxxxxx
                      xxxxxxxx
[3:2]-2                  fffffhhh
                      xxxxxxxxxxxxx
                        xxxxxxxx
                  xxxxxxxx
[3:2]-3               hhffhhhhhh
                  xxxxxxxxxxxxxxxxx
                     xxxxxxxxxx
                xxxxxxxxxxxxx
[3:2]-4           hhhffffffffhh
                xxxxxxxxxxxxxxxxxxx
                 xxxxxxxxxxxxx
PA              ==============
               xxxxxxxxxxxxxxxxxxxx
```

The precision of the PA adder is $m+7$ - reduced from $m+12$ by 5 positions. Using expression (2.61) the delay of the prefix adder is estimated as

$$T_{PA}(m) = t_{ga} + log_2(m)t_{cell} + t_{XOR} \approx 0.5t_{FA} + log_2(m) \times 0.6t_{FA} + 0.5t_{FA}$$

$$= [1 + 0.6 \times log_2(m)]t_{FA}$$

Using expression (2.62), we get the equivalent number of full adders

$$C_{PA}(m) \approx m \times FA + (m/2)log_2(m) \times 0.5FA$$

The critical path is: f+f+f+f+PA(m+7) resulting in

$$T_{[3:2]+PA} = 4t_{FA} + T_{PA}(m + 7) < T_{CRA}$$

The equivalent number of full adders is:

$$C_{[3:2]+PA} = (m{-}3)FA{+}3HA{+}(m{-}3)FA{+}3HA{+}(m{-}6)FA{+}8HA{+}mFA{+}5HA{+}C(PA)$$

$$= (4m - 12)FA + 19HA + (m + 7)FA + 0.25(m + 7)log_2(m + 7)FA$$

$$\approx [5m + 0.25(m + 7)log_2(m + 7)]FA > C_{CRA}$$

Without reducing the precision of the final adder, the input/output diagram is

```
                            xxxxxxxx
                          xxxxxxxx
                         xxxxxxxx
[3:2]-1                   hhfffff
                         xxxxxxxxxxx
                         xxxxxxx x
                      xxxxxxxx
[3:2]-2                  fffff
                      xxxxxxxxxxxxx
                        xxxxx xx x
                  xxxxxxx x
[3:2]-3               hhf
                  xxxxxxxxxxxxxxxxx
                     xxx xxxx xx x
                xxxxxxxxxxxxx
[3:2]-4           hhhffffffff
                xxxxxxxxxxxxxxxxxxx
                 xxxxxxxxxxx  xx x
PA              ===================
               xxxxxxxxxxxxxxxxxxxx
```

Calculation of the delay and cost is left to the reader.