DIGITAL ARITHMETIC
Miloš D. Ercegovac and Tomás Lang
with contributions by Elisardo Antelo and Fabrizio Lamberti
Morgan Kaufmann Publishers, an imprint of Elsevier, ©2004

Chapter 3: Solutions to Exercises

Exercise 3.1

As explained in the text, for two's complement representation the most-significant bit of each operand is inverted and $-m$ is added, with its least-significant bit aligned with the most-significant bit of the operands. For $m = 7$ we add -7 = 1001. Moreover, to avoid an extra row, we evaluate $1001 + g_0' = 10g_0'g_0$. The resulting matrix is

$$\begin{array}{rcccc}
a_0'. & a_1 & a_2 & \ldots & a_n \\
b_0'. & b_1 & b_2 & \ldots & b_n \\
c_0'. & c_1 & c_2 & \ldots & c_n \\
d_0'. & d_1 & d_2 & \ldots & d_n \\
e_0'. & e_1 & e_2 & \ldots & e_n \\
f_0'. & f_1 & f_2 & \ldots & f_n \\
10g_0'g_0. & g_1 & g_2 & \ldots & g_n
\end{array}$$

Exercise 3.2

a) We obtain the arithmetic expression of w in terms of a, b and c indicated in Fig. 3.5 where

- a is the modulo-2 sum of the four external inputs in position $i-1$.
- b is an internal carry from position $i-1$ to i when $a = 0$.
- c is the modulo-2 sum of the four external inputs in position i and the carry $h_{i,1}$ from position $i-1$.

Therefore the arithmetic value of w relative to position $i-1$ is

$$w = 2c + 2b + a \leq 4$$

because $ab = 0$.

The largest value $w = 4$ is obtained for $a = 0$, $b = 1$ and $c = 1$. An input pattern resulting in $w = 4$ is

i	$i-1$
1	1
1	0
1	0
0	1
$c=1$	$a=0$
$h=0$	$b=1$

b) Two columns must be considered because a carry can propagate up to two positions. Note that the output c depends on a carry from the previous position.

c) To show that the network in Fig. 3.5 implements a [4:2] carry-save adder module, consider the following arithmetic function (this proof is similar to that performed for Exercise 2.33):

$$vs_i = (x_{i,3} + x_{i,2} + x_{i,1} + x_{i,0} + h_{i,1}) mod 2$$

$$(h_{i+1,1} + h_{i+1,2}) = (x_{i,3} + x_{i,2} + x_{i,1} + x_{i,0} + h_{i,1})/2$$

where vs_i is the module output corresponding to the output of the xor gate.

Clearly, the network of XOR gates produces the correct vs_i (odd parity of $x_{i,3}, x_{i,2}, x_{i,1}, x_{i,0}, h_{i,1}$).

Now for $h_{i+1,1}$ we see that it corresponds to the MAJORITY function of $x_{i,3}$, $x_{i,2}$ and $x_{i,1}$ (make table). That is, if $x_{i,3} + x_{i,2} + x_{i,1} = 2$ or 3 then $h_{i+1,1} = 1$

Moreover,

$h_{i+1,2} = 1$ if

i) $(x_{i,3} + x_{i,2} + x_{i,1} = 1)$ and $(x_{i,0} = 1$ or $h_{i,1} = 1)$

In this case $x_{i,3} + x_{i,2} + x_{i,1} + x_{i,0} + h_{i,1} = 2$ or 3 and we get $(h_{i+1,1} = 0, h_{i+1,2} = 1)$.

ii) $(x_{i,3} + x_{i,2} + x_{i,1} = 2)$ and $x_{i,0} = 1$ and $h_{i,1} = 1$

In this case $x_{i,3}+x_{i,2}+x_{i,1}+x_{i,0}+h_{i,1} = 4$ and we get $(h_{i+1,1} = 1, h_{i+1,2} = 1)$

iii) $(x_{i,3} + x_{i,2} + x_{i,1} = 3)$ and $(x_{i,0} = 1$ or $h_{i,1} = 1)$

In this case $x_{i,3} + x_{i,2} + x_{i,1} + x_{i,0} + h_{i,1} = 4$ or 5 and we get $(h_{i+1,1} = 1, h_{i+1,2} = 1)$

Consequently,

$$x_{i+3} + x_{i,2} + x_{i,1} + x_{i,0} + h_{i,1} = vs_i + 2h_{i+1,1} + h_{i+1,2}$$

and the module is a [4:2] module.

d) Since networks in positions i and $i-1$ are identical, it is sufficient to compare implementations of Fig. 3.5 and 2.41 for one position only.

The internal outputs $h_{i,1}$ of Fig. 3.5 and t_{i+1} of Fig. 2.41 both correspond to the majority function of three inputs. The implementation given in Fig.

3.5 seems good since it shares an xor gate with the implementation of the other two outputs (and it does not affect the critical path). The rest of the network is the same in both implementations. Consequently, the two networks have the same delay and the same number of equivalent gates.

Exercise 3.3

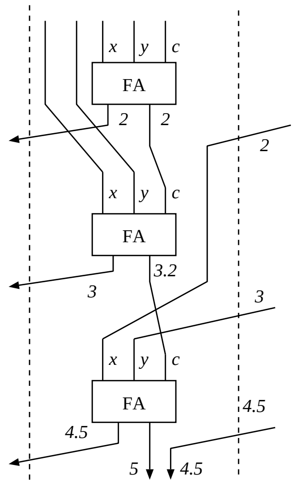


Figure E3.3a: The [5:2] module for Exercise 3.3.

A [5:2] module is shown in Figure E3.3a. and an array of these modules to reduce five 8-bit operands in Figure E3.3b.

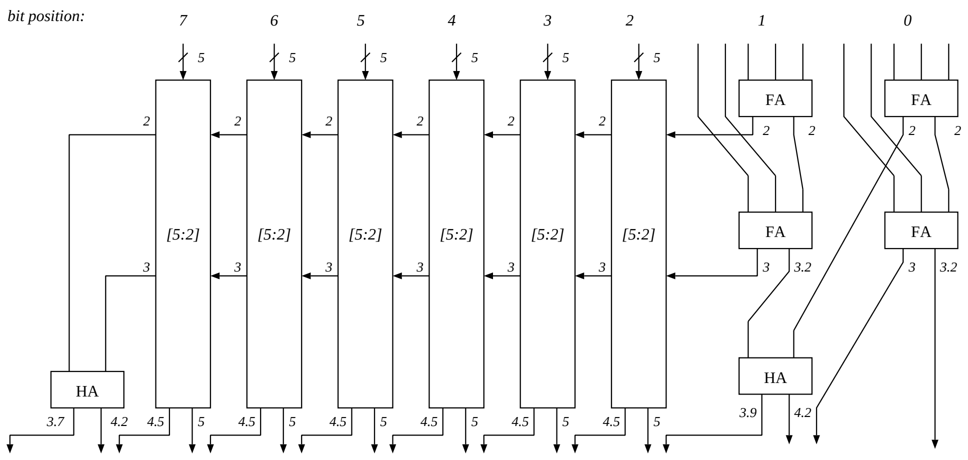
To determine the critical path we use the following delay model, simplified from the model given in Table 2.2:

	FA		HA	
from/to	c_{out}	s	c_{out}	s
(x,y)		2	0.7	1.2
x	2			
y	1.5			
c	1	1.2	-	-

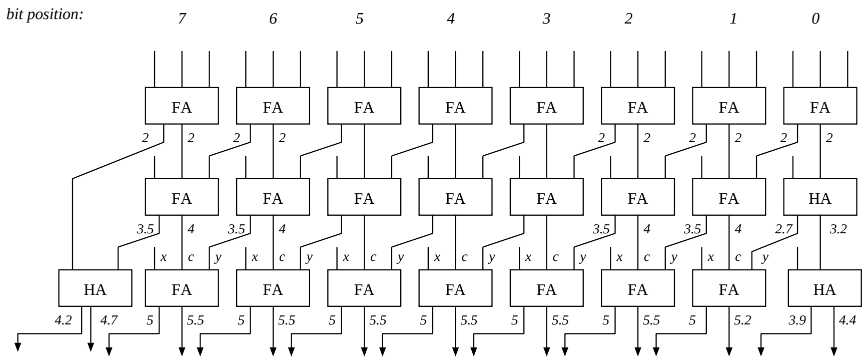
where the delay is normalized to the delay t_{c-c}.

Figure E3.3a indicates the module delays using this model. Consequently, the critical path delay is $5t_{c-c}$. The implementation uses 22 FAs and 2 HAs.

For comparison, an array of [3:2] modules to reduce 5 8-bit operands is shown in Figure 3.3c.As shown, the critical path has a delay of $5.5t_{c-c}$. The network cost is cost 22 FAs and 3 HAs. We conclude that both networks have the same cost and that the network using [5:2] modules is somewhat faster than the network using [3:2] modules.



(b)



(c)

Figure E3.3: (b) Network of [5:2] modules to reduce 5 8-bit operands. (c) Network of [3:2] modules to reduce 5 8-bit operands.

Exercise 3.4

To determine the critical path we use the following delay model, simplified from the model given in Table 2.2:

	FA	
from/to	c_{out}	s
(x, y)		2
x	2	
y	1.5	
c	1	1.2

where the delay is normalized to the delay t_{c-c}.

A [6:2] module is shown in Figure E3.4. The delay in the critical path is $T = 6t_{c-c}$.

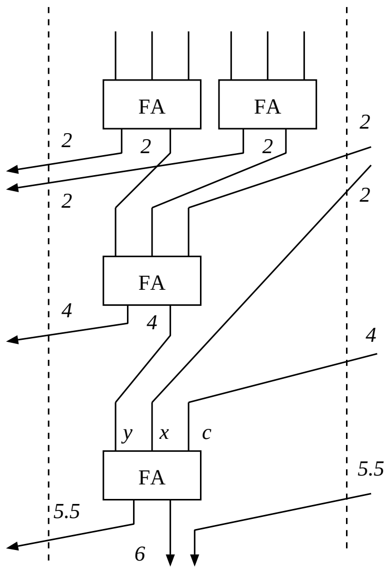


Figure E3.4: The network of FAs for Exercise 3.4.

Exercise 3.5

To determine the critical path we use the following delay model, simplified from the model given in Table 2.2:

	FA	
from/to	c_{out}	s
(x, y)		2
x	2	
y	1.5	
c	1	1.2

where the delay is normalized to the delay t_{c-c}.

A [9:2] module is shown in Figure E3.5. The delay in the critical path is $T = 8t_{c-c}$.

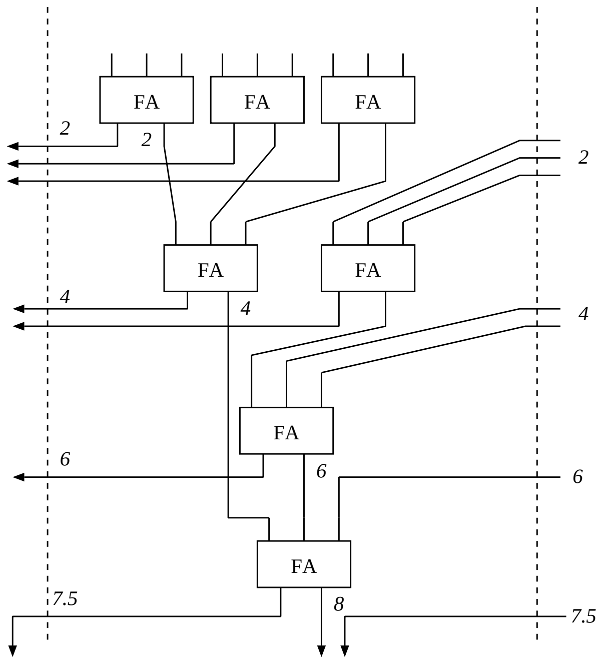


Figure E3.5: The network of FAs for Exercise 3.5.

Exercise 3.6 A design based on a CAD tool - left to the reader.

Exercise 3.7

a) The network output delays, assuming $L = 1$, are

$t_{q_0} = t_{q_{B0}} + t_{XOR} \approx t_{q_{A0}} + t_{XOR} = 0.748 + 0.329 = 1.08$

$t_{q_1} = t_{q_{B0}} + t_{NAND2} + t_{XNOR} = 0.748 + 0.136 + 0.329 = 1.21$

$t_{q_2} = t_{AND-NOR} + t_{NAND2} + t_{NAND2} + t_{XNOR} + t_{NAND3} + t_{NAND3} = 0.192 + 0.136 + 0.106 + 0.36 + 0.119 + 0.119 = 1.03$

The delay of the critical path is $t_{q_1} = 1.21$

b) The cost of the gate network:

$C_{g-net} = 7 \times 1 + 3 \times 2 + 1 \times 2 + 1 \times 2 + 1 \times 2 + 8 \times 3 = 43$ equivalent gates.

c) The delay in the critical path of the (7:3] impelementation with FAs is

$t_{FA-net} = t_{x-s} + t_{y-s} + t_{c-s} = 0.71 + 0.03 \times 1.3 + 0.52 + 0.03 \times 1.3 + 0.46 + 0.03 = 1.8$

d) The cost of the FA network:

$C_{FA-net} = 4 \times 7 = 28$ equivalent gates.

e) The delay of the gate network is about 1.5 times smaller than the delay of the FA network. The size of the gate network is about 1.5 times larger than the size of the FA network.

Exercise 3.8

A network of full-adders implementing a (15:4] counter is shown in Figure E3.8.

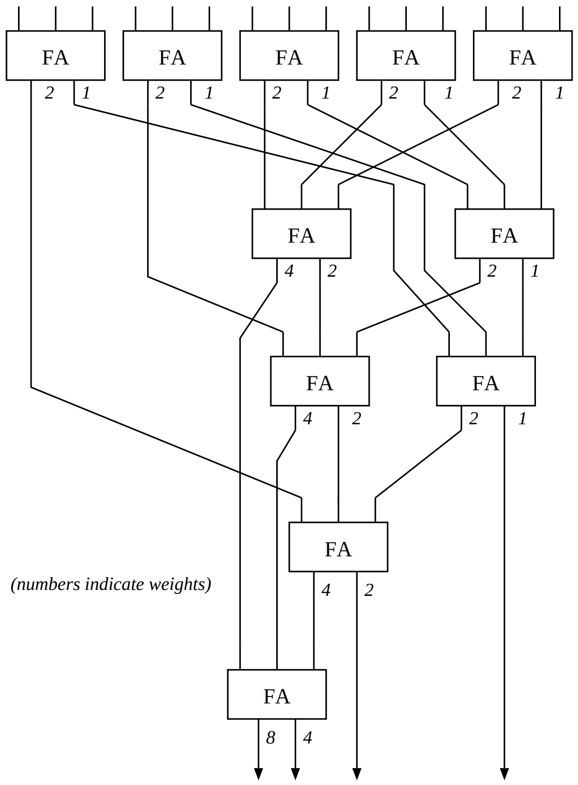


Figure E3.8: A network of FAs implementing (15:4] counter in Exercise 3.8.

Exercise 3.9

The reduction sequence produced by a (15:4] counter is

No. of levels:	1	2	3	...
No. of operands:	15	48	180	...

Therefore, 127 operands can be reduced to four using 3 levels of (15:4] counters.

Exercise 3.10

The maximum value of the sum is $S = 32 \times 127$. Since $2^{11} < S = 2^{12} - 2^5 < 2^{12}$, 12 bits are necessary.

1. The logic diagram of a bit-slice showing only CSA and registers is given in Figure E3.10(a).
2. The block diagram at the word level is shown in Figure E3.10(b).

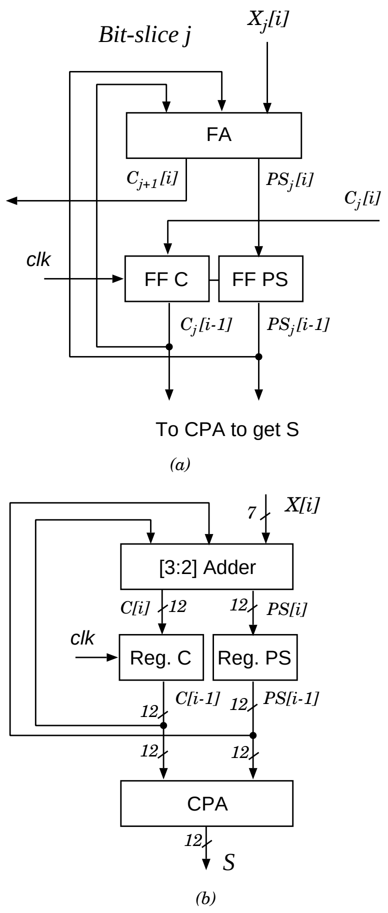


Figure E3.10: (a) Bit-slice of multi-operand adder. (b) Multi-operand adder of Exercise 3.10.

3. The critical path delay: $t_s + t_{reg}$ where t_s is the delay of the sum output of a FA.

4. The latency: $32 \times (t_s + t_{reg}) + t_{CPA} = 32 \times (t_s + t_{reg}) + 11t_c + t_s$ where t_c is the delay of the carry output of a FA.

5. Use a CRA instead of the CSA. In this case the adder has 11 bits plus the carry-out. The critical path is $10t_c + t_s + t_{reg}$. Assume that $t_s = 2t_c$ and $t_{reg} = t_s$. Then the ratio of cycle times in the two alternatives is:

$$(10t_c + t_s + t_{reg})/(t_s + t_{reg}) = 7t_s/2t_s = 3.5$$

The latency of the alternative with CRA is $32 \times (10t_c + t_s + t_{reg})$ and the ratio of latencies is

$$(32 \times (10t_c + t_s + t_{reg})/(32 \times (t_s + t_{reg}) + 12t_c + t_s)$$

$$= (32 \times 7t_s)/(32 \times 2t_s + 6.5t_s) = 224/70.5 = 3.2$$

In terms of hardware, the alterantive with CRA uses only one register and an 11-bit adder. The alternative with CSA uses two registers and two adders. This is roughly twice as much hardware.

Exercise 3.11

1. $X[i] \in [-31, 31]$, two's complement. The number of bits is 11 because the result is in the range

$$[32 \times (-31), 32 \times 31]$$

2. The logic diagram of a bit-slice showing only CSA and registers is given in Figure E3.11(a) .

3. The block diagram at the word level is shown in Figure E3.10(b).

4. The critical path delay: $t_s + t_{reg}$ where t_s is the delay of the sum output of a FA.

5. The latency: $32 \times (t_s + t_{reg}) + t_{CRA-11} = 32 \times (t_s + t_{reg}) + 10t_c + t_s$ where t_c is the delay of the carry output of a FA.

6. Use a CRA instead of the CSA. In this case the adder has 11 bits plus the carry-out. The critical path is $10t_c + t_s + t_{reg}$. Assume that $t_s = 2t_c$ and $t_{reg} = t_s$. Then the ratio of cycle times in the two alternatives is:

$$(10t_c + t_s + t_{reg})/(t_s + t_{reg}) = 7t_s/2t_s = 3.5$$

The latency of the alternative with CRA is $32 \times (10t_c + t_s + t_{reg})$ and the ratio of latencies is

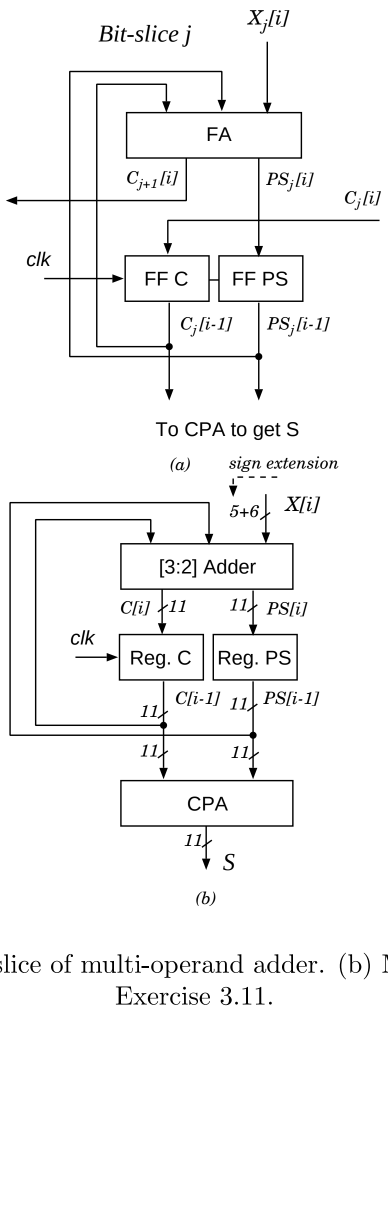


Figure E3.11: (a) Bit-slice of multi-operand adder. (b) Multi-operand adder of Exercise 3.11.

$$(32 \times (10t_c + t_s + t_{reg})/(32 \times (t_s + t_{reg}) + 12t_c + t_s)$$

$$= (32 \times 7t_s)/(32 \times 2t_s + 6t_s) = 224/70 = 3.2$$

In terms of hardware, the alternative with CRA uses only half as much as the alternative with CSA.

Exercise 3.12

The linear array networks are shown in Figure E3.12.

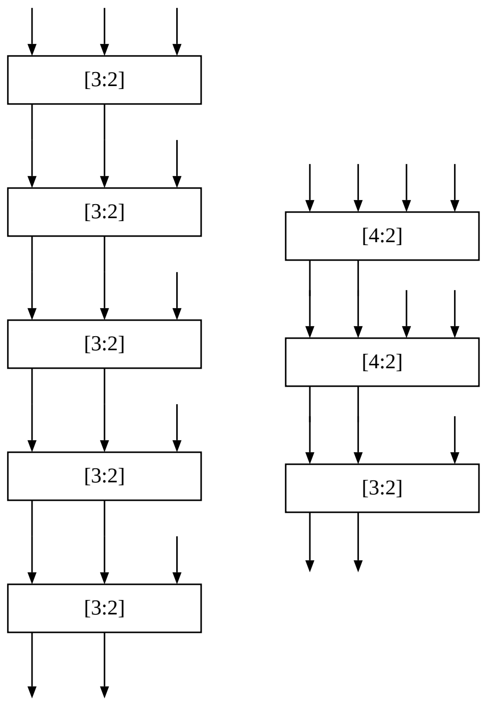


Figure E3.12: The adder networks for Exercise 3.12.

Exercise 3.13

To determine the critical path we use the following delay model, simplified from the model given in Table 2.2:

	FA		HA	
from/to	c_{out}	s	c_{out}	s
(x, y)		2	0.7	1.2
x	2			
y	1.5			
c	1	1.2	-	-

where the delay is normalized to the delay t_{c-c}.

The [5:2] module shown in Fig. E3.13a has a critical path of $5t_{c-c}$.

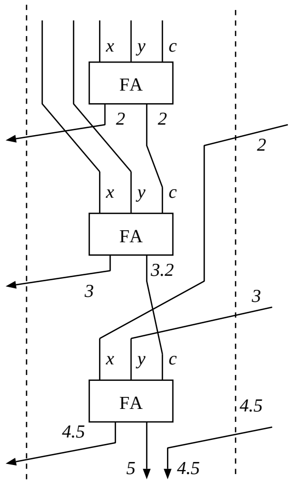


Figure E3.13a: [5:2] module.

To reduce the ten 4-bit operands we use an array of [5:2] modules (forming two adders of 5 inputs each) followed by a [4:2] adder, as shown in Figure E3.13b. The critical path delay is $8t_{c-c}$. The implementation uses 28 FAs and 6 HAs.

For comparison, Figure E3.13c shows an array of [3:2] adders to reduce 10 4-bit operands. At the full-adder level, this array is implemented as shown in Figure E3.13d. The corresponding critical path delay is $9.2t_{c-c}$.

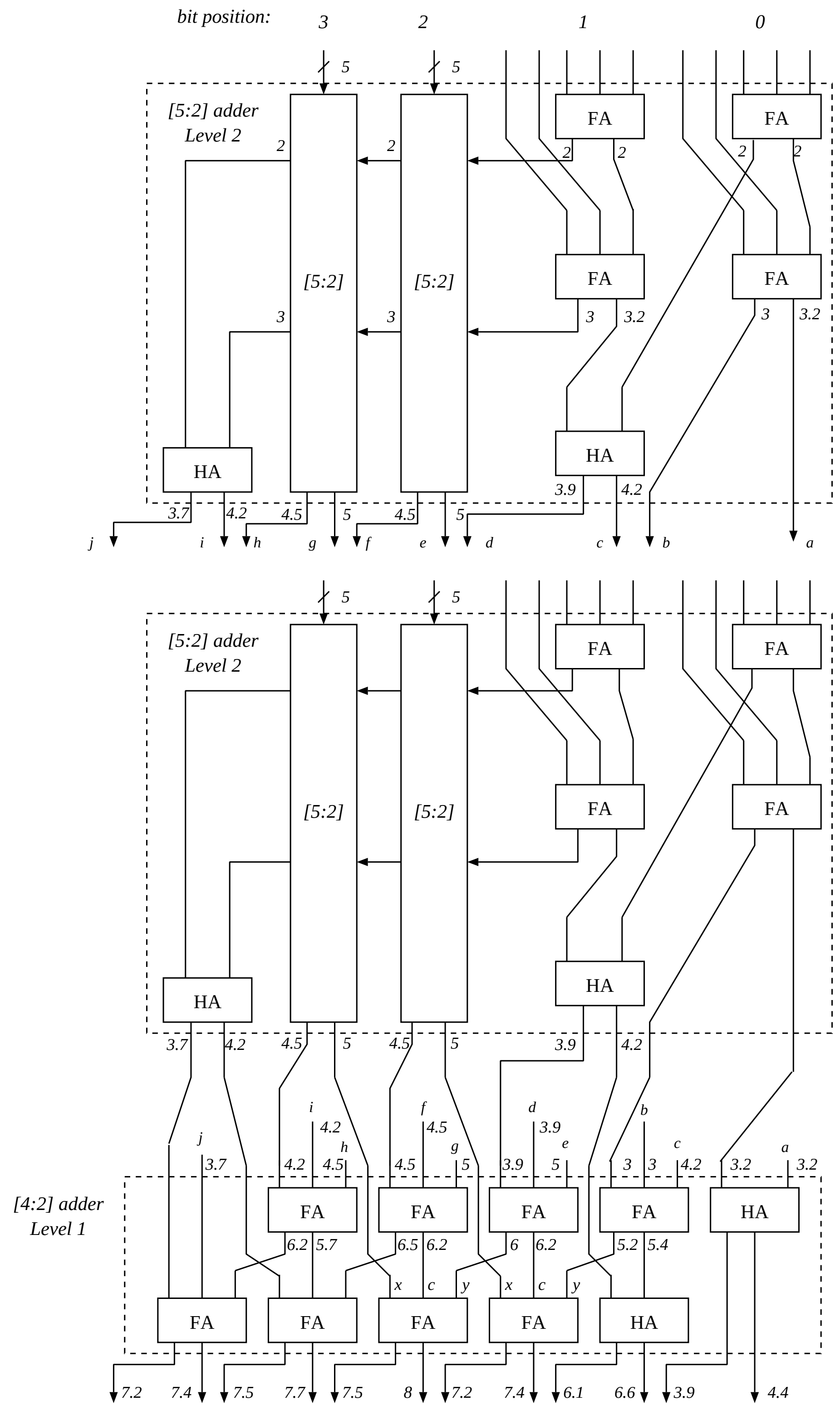


Figure E3.13b: Network of [5:2] and [4:2] modules to reduce 10 4-bit operands.

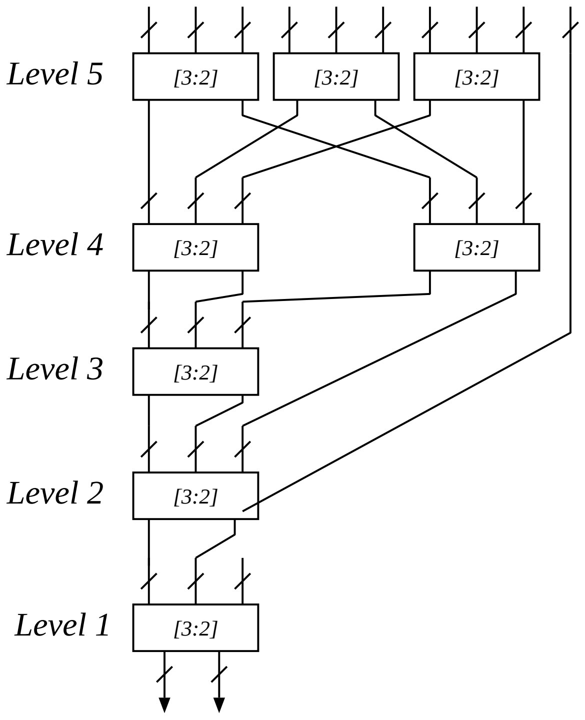


Figure E3.13c: Network of [3:2] adders to reduce 10 4-bit operands.

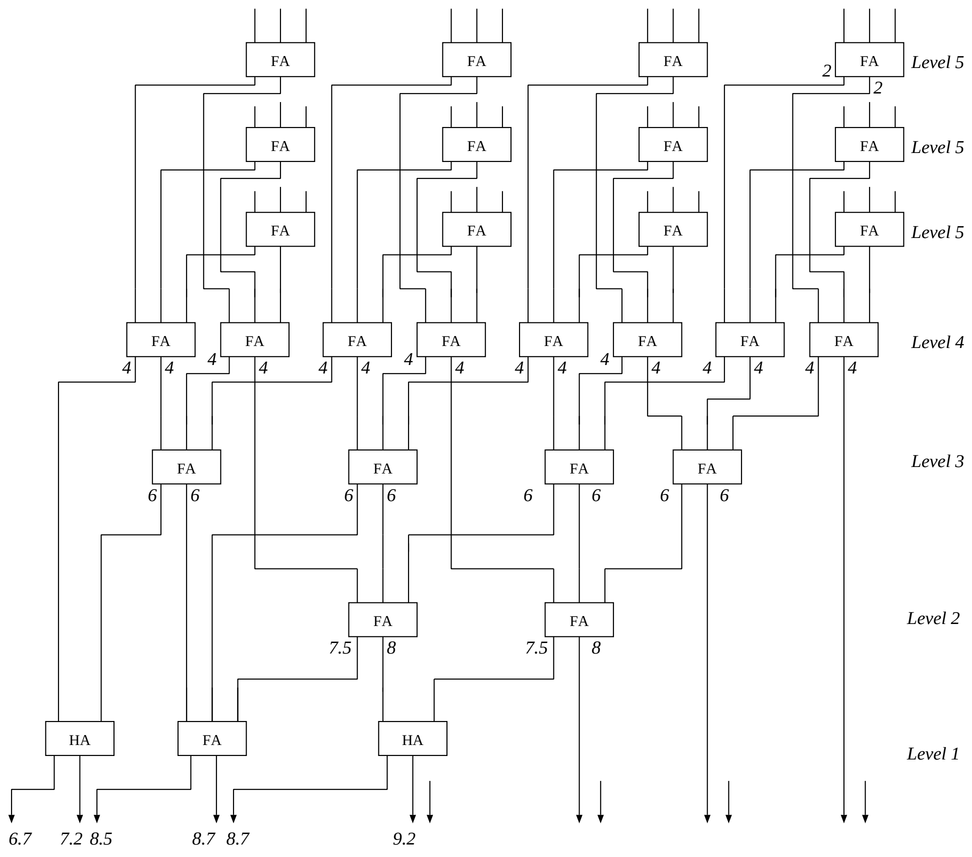


Figure E3.13d: Network of FAs and HAs to reduce 10 4-bit operands.

Exercise 3.14

The final sum is in the range [0,2040] so the result has 11 bits. We use the following delays:

$t_c = t_{XOR} = t_{mux} = d = 2t_{gate}$, $t_s = 2t_{XOR} = 2d = 4t_{gate}$, $t_{buf} = 0.5d = t_{gate}$ (fanout = 4)

(a) With carry-ripple adder

There are seven CRAs, operating in an overlapped mode as indicated below

```
CRA-1:          xxxxxxxx                         dddddddd CRA-1
(8)             xxxxxxxx                     dddddddddd   CRA-2
            ------------                   dddddddddd     CRA-3
CRA-2:         xxxxxxxxx                ddddddddddd       CRA-4
(9)             xxxxxxxx              ddddddddddd         CRA-5
            ------------            ddddddddddd           CRA-6
CRA-3:        xxxxxxxxxx          ddddddddddd             CRA-7
(10)            xxxxxxxx       dddddddddddd               Result
            ------------
CRA-4:        xxxxxxxxxx       Timing diagram of linear array
(10)            xxxxxxxx                   with CRAs
            ------------
CRA-5:       xxxxxxxxxxx
(11)            xxxxxxxx
            ------------
CRA-6:       xxxxxxxxxxx
(11)            xxxxxxxx
            ------------
CRA-7:       xxxxxxxxxxx
(11)            xxxxxxxx
            ------------
Sum          xxxxxxxxxxx
```

The delay of an n-bit CRA is $T_{CRA}(n) = (n-1)t_c + t_s$.

The delay of the linear array with CRAs is

$T_{LA-CRA} = 6t_s + 10t_c + t_s = (12 + 10 + 2)d = 48t_{gate}$

(b) With single-level carry-skip adder, $m = 4$

A linear array of carry-skip adders is shown in Figure E3.14b. Note that the delay of a skip signal of a group is determined by the latest arriving input to the group. Consequently, the skip paths do not affect the worst-case delay in the array. The reason for this behavior is that the first level does not allow skipping. A different situation would occur, for example, for 12-bit operands.

From the figure we get

$$T_{LA-CSK} = 26d = 52t_{gate}$$

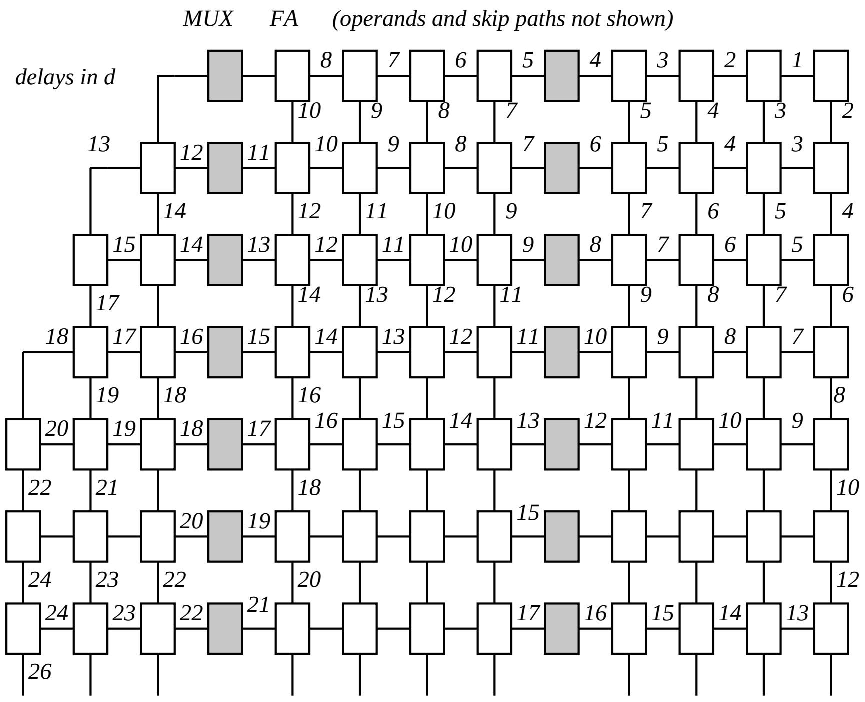


Figure E3.14b: Linear array of carry-skip adders (Exercise 3.14(b))

(c) With parallel prefix adder (min. number of levels, fanout = 2) (Figure 2.20) $T_{PPA}(8) = t_{gap} + 3t_{level} + t_{XOR} = (1+3+1)d = 5d = 10t_{gate}$

$T_{PPA}(10/11) = t_{gap} + 4t_{level} + t_{XOR} = (1+4+1)d = 6d = 12t_{gate}$

The adders in the array cannot operate in an overlapped manner: all inputs to a parallel prefix adder must arrive at the same time. Therefore, the total delay for the linear array of seven parallel prefix adders is

$T_{LA-PPA} = (5 + 6 \times 6)d = 41d = 82t_{gate}$

(d) With carry-select adder (Figure 2.22)

A linear array using carry-select adders is shown in Figure E3.14d.

A carry-select adder consists of two modules: the least-sgnificant part of the result is obtained using a 4-bit CRA; the most-significant part of the result is obtained by selecting one of the two conditional results produced using c_4.

The total delay obtained from the figure is

$T_{LA-CS} = 29d = 58t_{gate}$

Note that the critical path is determined by the conditional adders - not by the carries controlling multiplexers. Therefore, the use of carry-select adders is inappropriate in this case.

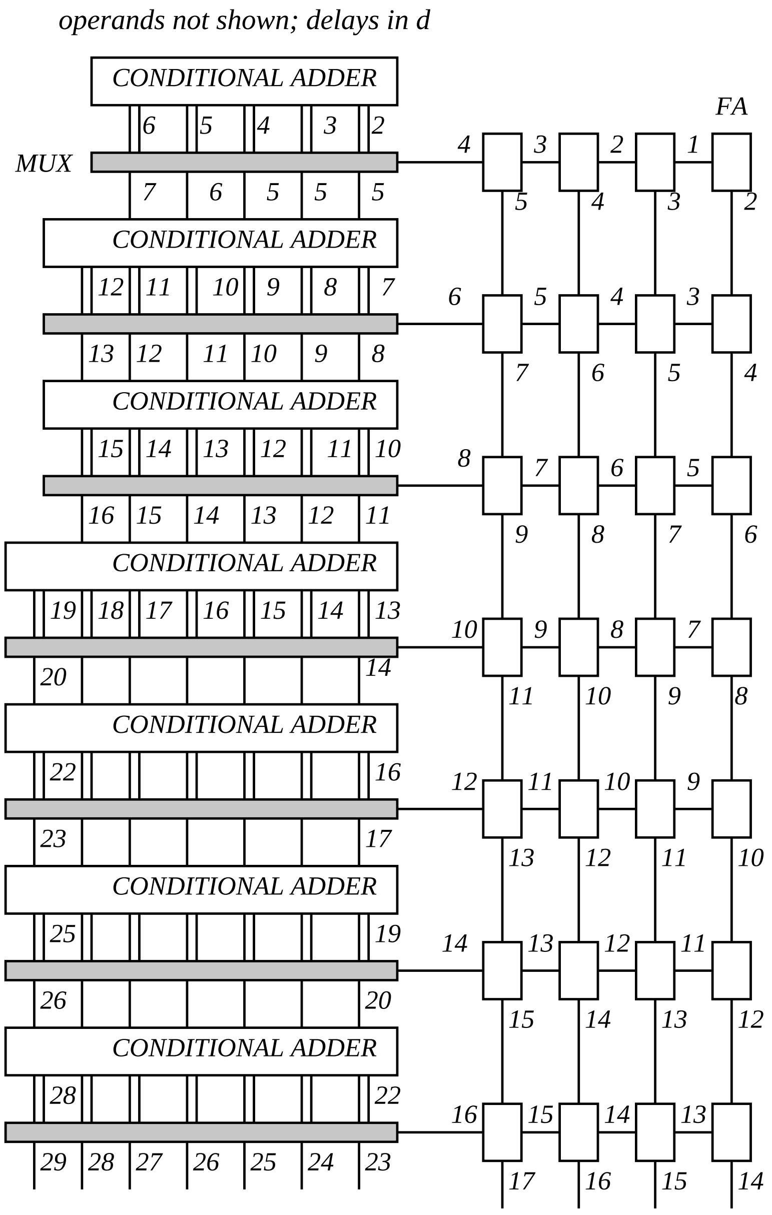


Figure E3.14d: Linear array of carry-select adders (Exercise 3.14(d))

(e) With [4:2] adder followed by parallel prefix adder:

There are three levels of [4:2] adders, each with a delay of three XORs ($3d$), followed by a 11-bit parallel prefix adder with a delay of $6d$ as described in part (c), for a total of

$T_{LA-[4:2]-PPA} = 3 \times 3d + 6d = 15d = 30t_{gate}$

Exercise 3.15

Figure E3.15 shows a linear array of [3:2] adders for producing the sum of 8 6-bit operands (two's complement). To determine the critical path we use the following delay model, simplified from the model given in Table 2.2:

	FA		HA	
from/to	c_{out}	s	c_{out}	s
(x, y)		2	0.7	1.2
x	2			
y	1.5			
c	1	1.2	-	-

where the delay is normalized to the delay t_{c-c}.

The delay of the critical path is $19.2t_{c-c}$. The delay attributed to the CPA is $7.2t_{c-c}$ which corresponds to 38% of the total delay. Using a faster CPA such as a prefix adder of Figure 2.20 with a delay of

$$t_P A = t_{gap} + 3t_{level} + t_{XOR} = (1.2 + 3 + 1.2)t_{c-c} = 5.4t_{c-c}$$

reduces the total delay to $(12{+}5.4)t_{c-c} = 17.4t_{c-c}$ and the percentage attributed to the CPA to 31%.

The speedup due to the use of a faster CPA is about 11%. It would be larger for a larger n.

```
m=8, n=6, [-32,31]x8 = [-256,248] --- 9 bits

Bit-matrix:

         x'xxxxx
         x'xxxxx
         x'xxxxx
         x'xxxxx
         x'xxxxx
         x'xxxxx
         x'xxxxx
      100x'xxxxx

8x (-1) = 1000
```

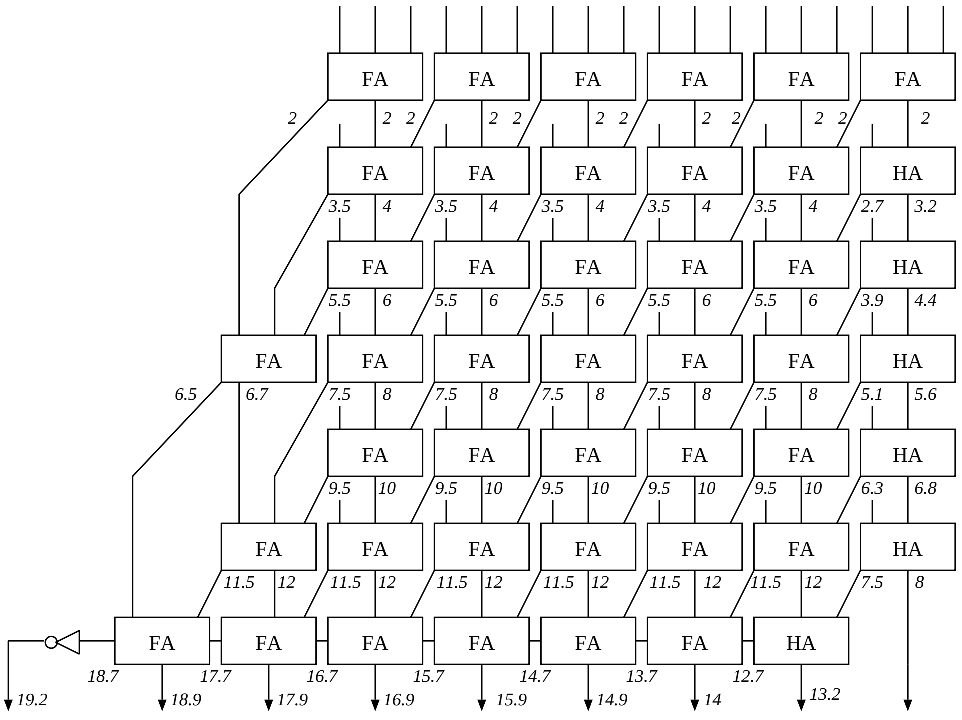


Figure E3.15: Bit-matrix and linear array of [3:2] adders (Exercise 3.15).

Exercise 3.16

a) Since the operands are in the range [-8, 7], the range of the result is [-48, 42] which requires 7 bits. The sign-extension simplification is shown in the following bit-array:

$$\begin{array}{rrrr}
a'_3 & a_2 & a_1 & a_0 \\
b'_3 & b_2 & b_1 & b_0 \\
c'_3 & c_2 & c_1 & c_0 \\
d'_3 & d_2 & d_1 & d_0 \\
e'_3 & e_2 & e_1 & e_0 \\
101f'_3 & f_2 & f_1 & f_0
\end{array}$$

A network of FAs and HAs is shown in Figure E3.16a. The CPA is of a carry-ripple type.

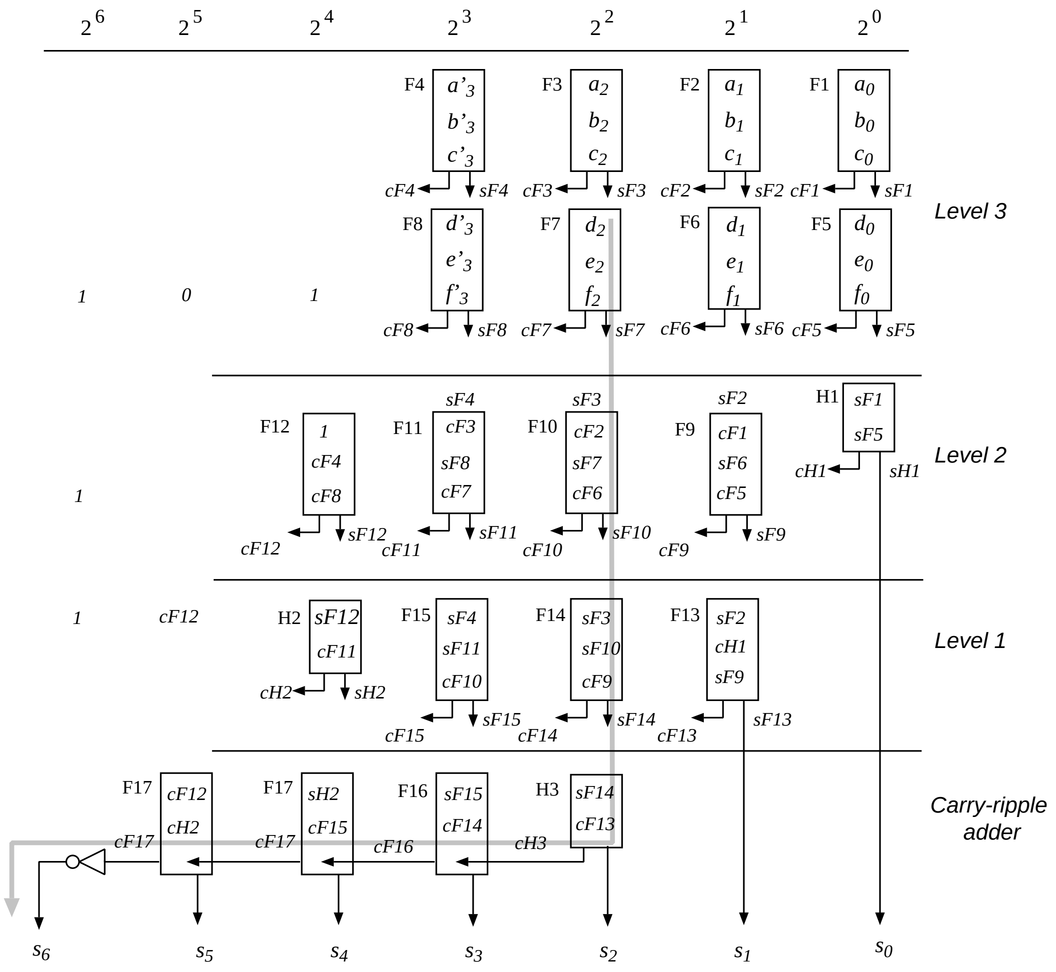


Figure E3.16a: The network of FAs and HAs for Exercise 3.16.

b) The delay of the critical path, indicated on the figure, is

$$T = 3t_{FA(ab-s)} + t_{H(ab-c)} + 3t_{FA(c-c)} + t_{INV}$$

Using the following delay model (simplified from that of Table 2.2)

	FA		HA	
from/to	c_{out}	s	c_{out}	s
(x, y)		2	0.7	1.2
x	2			
y	1.5			
c	1	1.2	-	-

where the delay is normalized to the delay t_{c-c}, we obtain

$$T = 3 \times 2 + 0.7 + 3 \times 1 + 0.5 = 10.2t_{c-c}$$

c) The values of inputs to FAs and HAs for the given operands after the sign-extension simplification are shown in Figure E3.16b.

```
                   Inputs to

      0 0 0 1   -7    Level 3
      1 0 1 0    2
      0 1 1 0   -2
      1 1 0 1    5
      1 0 1 1    3
1 - 1 0 0 1 0   -6  (extended)
------------------
      1 1 0 1         Level 2
    0 0 1 0
    1 0 1 0 0
1 - 1 0 1 1
------------------
      1 1 0 1         Level 1
    0 1 0 0
1 1 0 0 1 1
-----------------
  1 0 0 0 1 1         CPA: Carry-ripple adder
1 0 1 1 0
-----------------
1 1 1 1 0 1 1   -5
```

Figure E3.16b: The bit arrays for Exercise 3.16(c).

Exercise 3.17

The two schemes are shown in Figure E3.17. If additions in level 1 and level 2 of Scheme A do not overlap, Scheme B is not faster than Scheme A if $T_{4-2} > T_{CPA}$. If additions in Scheme A overlap, then, for example, using carry-ripple adders in both schemes, we get $T_A = t_s + (n+1)t_c$ and $T_B = t_{4-2} + (n+1)t_c$. In this case Scheme A is faster because $t_s < t_{4-2}$.

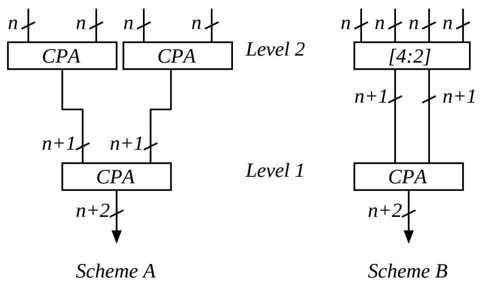


Figure E3.17: Two schemes for reducing four n-bit operands.

Exercise 3.18

We use two [4:2] adders in the first level. Assuming that the range of each operand is -128,127 we get a range of the output of each [4:2] adder of -512,508 requiring a width of 10 bits. Note that the sign extension could be simplified, as done Section 3.1, reducing the width of the adders.

Performing the [4:2] addition using the modules of Figure 2.41, described by

$$t_{i+1} = MAJORITY(x_i, y_i, w_i)$$

$$c_{i+1} = \begin{cases} t_i & \textbf{if} \quad (x_i + y_i + w_i + z_i) mod\ 2 = 1 \\ z_i & \textbf{otherwise} \end{cases}$$

$$s_i = (x_i + y_i + w_i + z_i + t_i) mod\ 2$$

we get

```
  73 0001001001            - 31     1111100001
- 52 1111001100              17     0000010001
  22 0000010110              47     0000101111
-127 1110000001             -80     1110110000
      ---------                      ---------
t     0010011000               t    0001000010
      -----------                   ----------
s     0010001010               s    0000101101
c     1100100010               c    1110100100
```

Now one second-level[4:2] adder. The range of the result is -1024,1016, requiring a width of 11 bits.

```
        00010001010
        11100100010
        00000101101
        11110100100
        -----------
    t   00001010100
        -----------
    s   00001110101
    c   11100001000
        -----------
        11101111101 = -131
```

Exercise 3.19

a) The delay in the critical path of the scheme for reducing $m = 6$ n-bit magnitudes, shown in Figure 3.26a, is

   $t_a = 4t_{FA-s} + (n-1)t_{FA-c} + 2t_{HA-c} + t_{OR}$

   The delay in the critical path of the scheme shown in Figure 3.26b is

   $t_b = 4t_{FA-s} + t_{CPA(n+2)}$

b) The scheme in Figure 3.26b is faster than the scheme in Figure 3.26a if

   $t_{CPA(n+2)} < (n-1)t_{FA-c} + 2t_{HA-c} + t_{OR}$

Exercise 3.20

A modification of Figure 3.21 for two's complement representation is shown in Figure E3.20.

```
m=8, n=5, [-16,15]x8 = [-128,120] --- 8 bits

Bit-matrix:

         x'xxxx
         x'xxxx
         x'xxxx
         x'xxxx
         x'xxxx
         x'xxxx
         x'xxxx
      100x'xxxx

8x (-1) = 1000
```

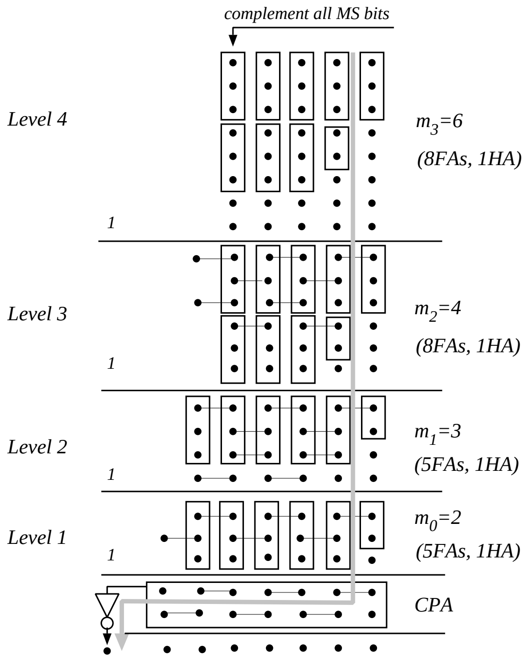


Figure E3.20: Reduction by columns of eight 5-bit two's complement numbers (Exercise 3.20).

Exercise 3.21

a) Magnitudes

l=4	
e_i	9 9 9 9 9 9
m_3	6 6 6 6 6 6
h_i	0 0 0 0 1 1
f_i	3 3 3 3 2 1
l=3	
e_i	3 6 6 6 6 6 6
m_2	4 4 4 4 4 4 4
h_i	1 0 0 0 0 1 0
f_i	0 2 2 2 2 1 1
l=2	
e_i	1 4 4 4 4 4 4 4
m_1	3 3 3 3 3 3 3 3
h_i	0 0 0 0 0 0 0 1
f_i	0 1 1 1 1 1 1 0
l=1	
e_i	2 3 3 3 3 3 3 3
m_0	2 2 2 2 2 2 2 2
h_i	1 0 0 0 0 0 0 1
f_i	0 1 1 1 1 1 1 0

Total: 37FA + 7HA

b) 2's complement

l=4	
e_i	1 1 1 1 9 9 9 9 9 9
m_3	6 6 6 6 6 6 6 6 6 6
h_i	0 0 0 0 0 0 0 0 1 1
f_i	0 0 0 0 3 3 3 3 2 1
l=3	
e_i	1 1 1 4 6 6 6 6 6 6
m_2	4 4 4 4 4 4 4 4 4 4
h_i	0 0 0 0 0 0 0 0 1 0
f_i	0 0 0 1 2 2 2 2 1 1
l=2	
e_i	1 1 2 4 4 4 4 4 4 4
m_1	3 3 3 3 3 3 3 3 3 3
h_i	0 0 0 0 0 0 0 0 0 1
f_i	0 0 0 1 1 1 1 1 1 0
l=1	
e_i	1 1 3 3 3 3 3 3 3 3
m_0	2 2 2 2 2 2 2 2 2 2
h_i	0 0 0 0 0 0 0 0 0 1
f_i	0 0 1 1 1 1 1 1 1 0

TOTAL: 39FA + 5HA

Exercise 3.22

a) From the Figures we see that the reduction by columns (Figure 3.21) has a CPA of 7 bits whereas the reduction by rows (Figure 3.27) has only 5 bits.

b) From the Figures, the critical path for reduction by columns is $4t_s + 5t_c + t_s = 5t_c + 5t_s$ and that for reduction by rows is $5t_s + 4t_c$.

c) Including the CPA, reduction by columns has 32 FA and 4 HA and reduction by rows has 32 FA and 3 HA.

Exercise 3.23

Critical paths for

1. Fig.3.21: $T_{3.21} = 4t_{FA(ab-s)} + t_{FA(ab-c)} + 5t_{FA(c-c)} + t_{FA(c-s)} = 4 \times 4 + 3 + 5 \times 2 + 2 = 31\tau$
2. Fig. 3.27: $T_{3.27} = 4t_{FA(ab-s)} + t_{HA(ab-c)} + 3t_{FA(c-c)} + t_{FA(c-s)} = 4 \times 4 + 1 + 3 \times 2 + 2 = 25\tau$

Exercise 3.24

(a) The range of the result is $-33 \leq z \leq 30$. Therefore, the minimum precision is 7 bits.

(b) $z = a - 3b + 5c = a + b - 4b + 4c + c$. The initial bit-matrix is shown in Fig. E3.24a. The final matrix, after simplification of sign extension, is shown in Fig. E3.24b.

$$
\begin{array}{ll|ccccccc}
 & a & & & & & a'_2 & a_1 & a_0 \\
 & & & & & & -1 & & \\
 & & & & & & b'_2 & b_1 & b_0 \\
 & & & & & & -1 & & \\
(a) & -4b & & & b_2 & b'_1 & b'_0 & 0 & 0 \\
 & & & & -1 & & 1 & & \\
 & 4c & & & c'_2 & c_1 & c_0 & 0 & 0 \\
 & & & & -1 & & & & \\
 & c & & & & & c'_2 & c_1 & c_0 \\
 & & & & & & -1 & & \\
\hline
 & & 1 & 0 & b_2 & b'_1 & a'_2 & a_1 & a_0 \\
 & & & & c'_2 & c_1 & b'_2 & b_1 & b_0 \\
(b) & & & & 1 & 1 & c'_2 & c_1 & c_0 \\
 & & & & & & b'_0 & & \\
 & & & & & & c_0 & &
\end{array}
$$

Figure E3.24:(a) The initial bit-matrix. (b) The simplified bit-matrix.

(c) To reduce the number of full adders and half adders, we use reduction by columns. The reduction stages are shown in Fig. E3.24c.

The delay for the reduction to two operands is $T = t_{HA} + 2t_{FA}$ The cost is 5FAs and 4HAs

d) To reduce the precision of the carry-propagate adder, reduction by rows is performed, so that the least-significant bits are reduced early. This reduction is shown in Figure E3.24d. The precision of the CPA is 5 bits.

1	-	b_2	b'_1	a'_2	a_1	a_0
		c'_2	c'_1	b'_2	b_1	b_0
		1	1	c'_2	c_1	c_0
				b'_0		
				c_0		
				HA1		
x		x	x	x	x	x
		x	x	x	x	x
		x	x	x	x	x
			x	x		
		HA3	FA1	HA2		
x		x	x	x	x	x
		x	x	x	x	x
		x	x	x	x	x
		FA5	FA4	FA3	FA2	HA4
x	x	x	x	x	x	x
		x	x	x	x	x

Figure E3.24c: Reduction by columns.

1	-	b_2	b'_1	a'_2	a_1	a_0
		c'_2	c'_1	b'_2	b_1	b_0
		1	1	c'_2	c_1	c_0
				b'_0		
				c_0		
		FA5	FA4	FA3,HA1	FA2	FA1
x	x	x	x	x	x	x
		x	x	x	x	
			x			
		HA4	FA6	HA3	HA2	
x	x	x	x	x	x	x
		x	x	x		

Figure E3.24c: Reduction by rows and final adder.

Exercise 3.25

Because of the absolute values, we perform two separate sums, obtain the absolute values and then add. To obtain these absolute values for the results in carry-save format, we determine the sign and, if negative, complement both the sum and the carry vectors. If this complementation is done in 2's complement representation, it is necessary to add two 1s in the least-significant position. Since there is a problem in finding a place for these ones, we use 1s' complement representation instead. We comment later on the effect of this on the component operations.

To reduce the width of the final addition, it would be possibe to saturate each of the two operands before this addition; however, we choose the alternative of saturating only the final sum.

The range of each of the two sums is from -1020 to 1-20, requiring 11 bits.

The bit array for each of the two multioperand sums (with complete sign extension) is as follows:

```
       x x x x x x x x
     x x x x x x x x x 0
       x x x x x x x x
 1 1 1 x'x'x'x'x'x'x'x'
 1 1 x'x'x'x'x'x'x'x'1
 1 1 1 x'x'x'x'x'x'x'x'
------------------------
```

A possible implementation of the 6-to-2 reduction consists of a first level of two [3:2] adders and a second level of one [4:2] adder. Because of the 1s' complement representation, the output carries of the adders have to be connected to the input carries (one carry for the [3:2] adder and two carries for the [4:2] adder).

The addition of the two arrays produce two pairs of sum and carry vectors of 11 bits. For each pair, a sign detector detects the sign and the pair is complemented if the sign is negative. The sign detection is implemented in the same way as for 2's complement representation: obtain the carry into the most-significant bit and xor with the xor of the most-significant bit of both operands).

Since we use 1s' complement representation, the complementation is done by bit inversion of both vectors.

Finally, a [4:2] adder produces two vectors that are added by a CPA. The (positive) result has 11 bits. Calling the bits s_i $(0 \leq i \leq 10)$ the saturated value z is obtained as follows:

$$z_j = s_j + (s_8 + s_9 + s_{10}) \quad (0 \leq j \leq 7)$$

Exercise 3.26

A pipelined linear array of adders is shown in Figure E3.26. For the final adder we use a CRA with four pipelined stages, each stage having a delay similar to a [4:2] adder.

```
m=8, n=6, [0,63]x8 = [0,504] --- 9 bits

Bit-matrix:

   xxxxxx
   xxxxxx  Stage 1
   xxxxxx
   xxxxxx
  ----------
  ooooooo
  oooooo
   xxxxxx  Stage 2
   xxxxxx
  ----------
  ooooooo
  oooooo
  oxxxxxx  Stage 3
  oxxxxxx
 ----------
 oooooooo
 ooooooo   (CPA with 4 pipelined stages)
 ----------
sssssssss
```

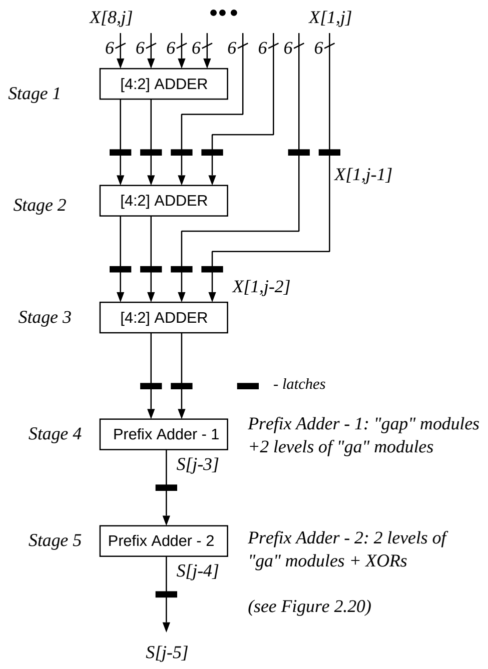


Figure E3.26: Pipelined linear array of [4:2] adders.

Exercise 3.27

A pipelined scheme for reducing six operands per iteration is shown in Figure E3.27. The cycle time is

$$t_{cycle} = 2t_{FA} + t_{reg}$$

The time to add 24 operands is

$$T(24) = (2+4)t_{cycle} + t_{CPA} = 12t_{FA} + 6t_{reg} + t_{CPA}$$

A non-pipelined scheme for reducing six operands to two has a cycle time

$$t_{NPcycle} = 4t_{FA} + t_{reg}$$

The time to add 24 operands using a non-pipelined scheme is

$$T_{NP}(24) = 4t_{NPcycle} + t_{CPA} = 16t_{FA} + 4t_{reg} + t_{CPA}$$

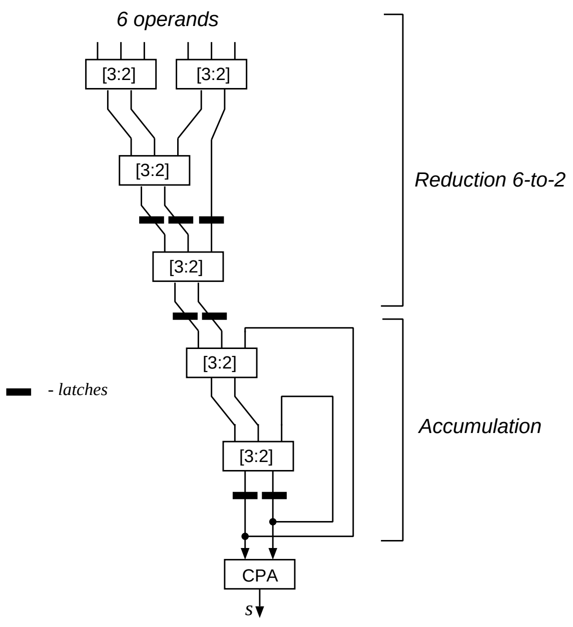


Figure E3.27: A pipelined scheme for reducing six operands per iteration.