DIGITAL ARITHMETIC
Miloš D. Ercegovac and Tomás Lang
Morgan Kaufmann Publishers, an imprint of Elsevier, ©2004

Chapter 3: Solutions to Selected Exercises

– with contributions by Elisardo Antelo –

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

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

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.

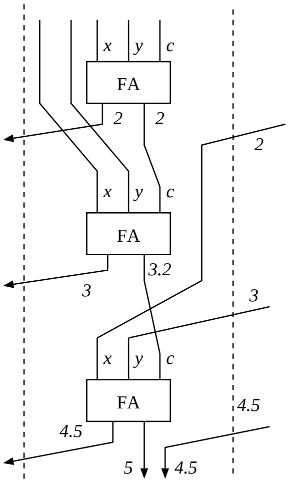


Figure E3.3a: The [5:2] module for Exercise 3.3.

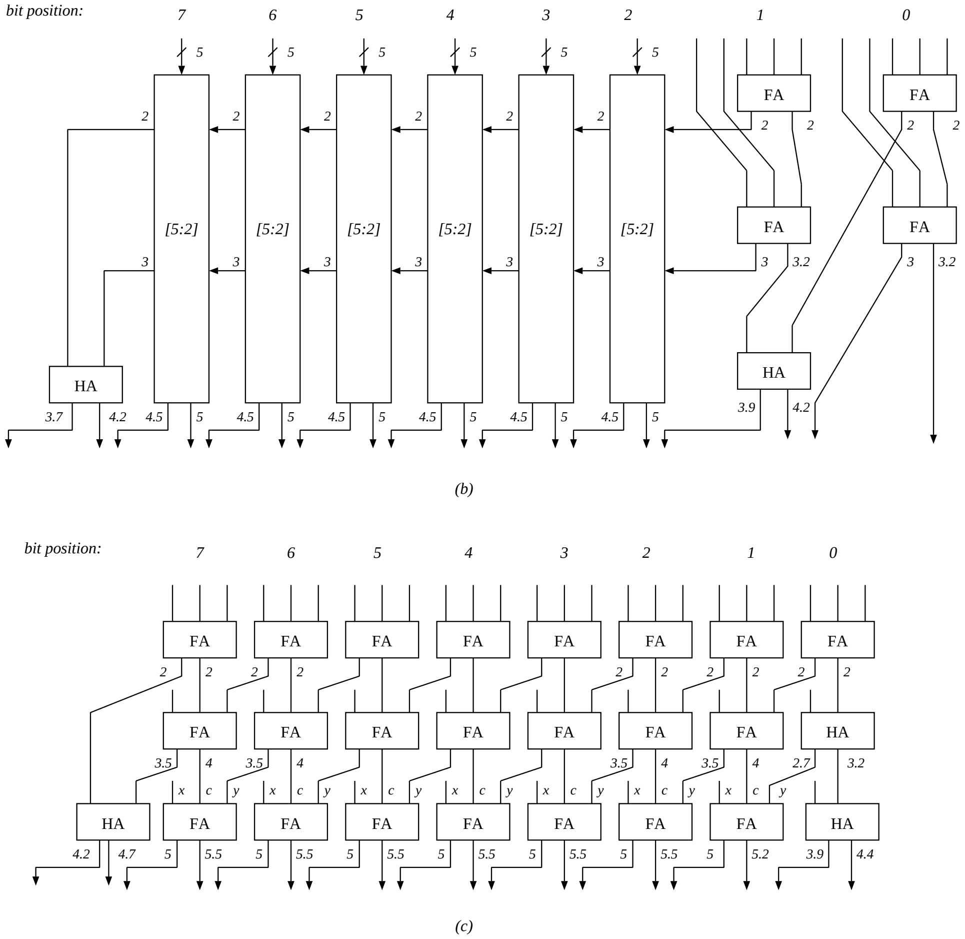
Exercise 3.5

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 $T = 8t_{c-c}$.



(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.

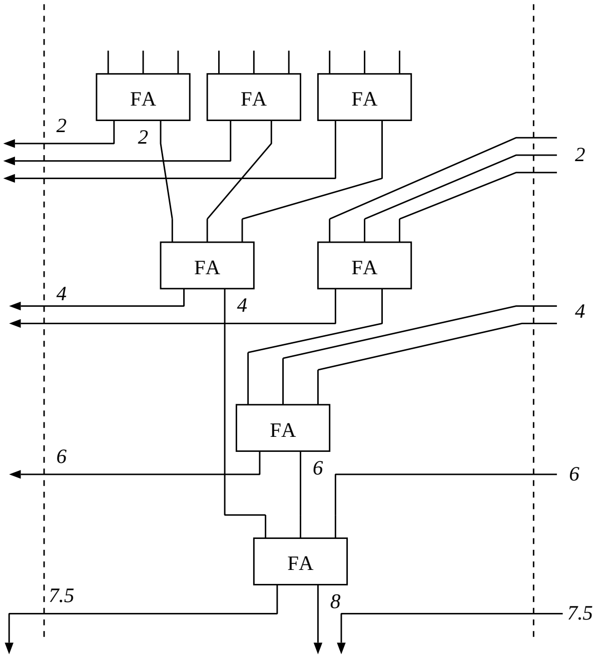


Figure E3.5: The network of FAs for Exercise 3.5.

Exercise 3.8

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

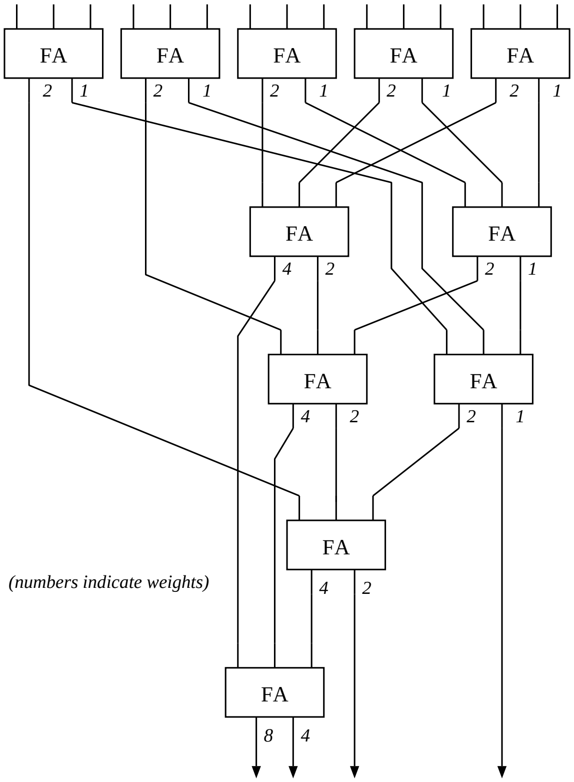


Figure E3.8: A network of FAs implementing (15:4] counter in Exercise 3.8.

Exercise 3.10

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).

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:

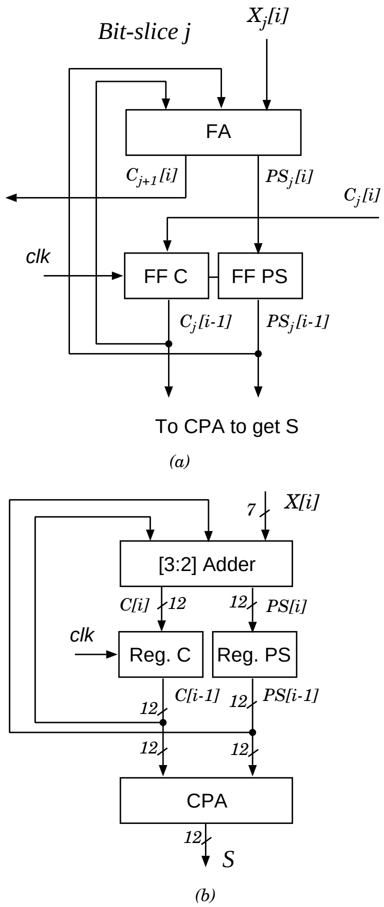


Figure E3.10: (a) Bit-slice of multi-operand adder. (b) Multi-operand adder of Exercise 3.10.

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

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 3.13

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 $5t_{c-c}$.

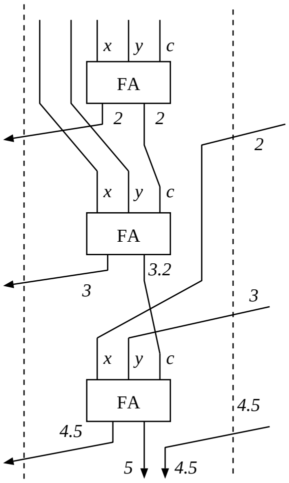


Figure E3.13a: [5:2] module.

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 $9.2t_{c-c}$.

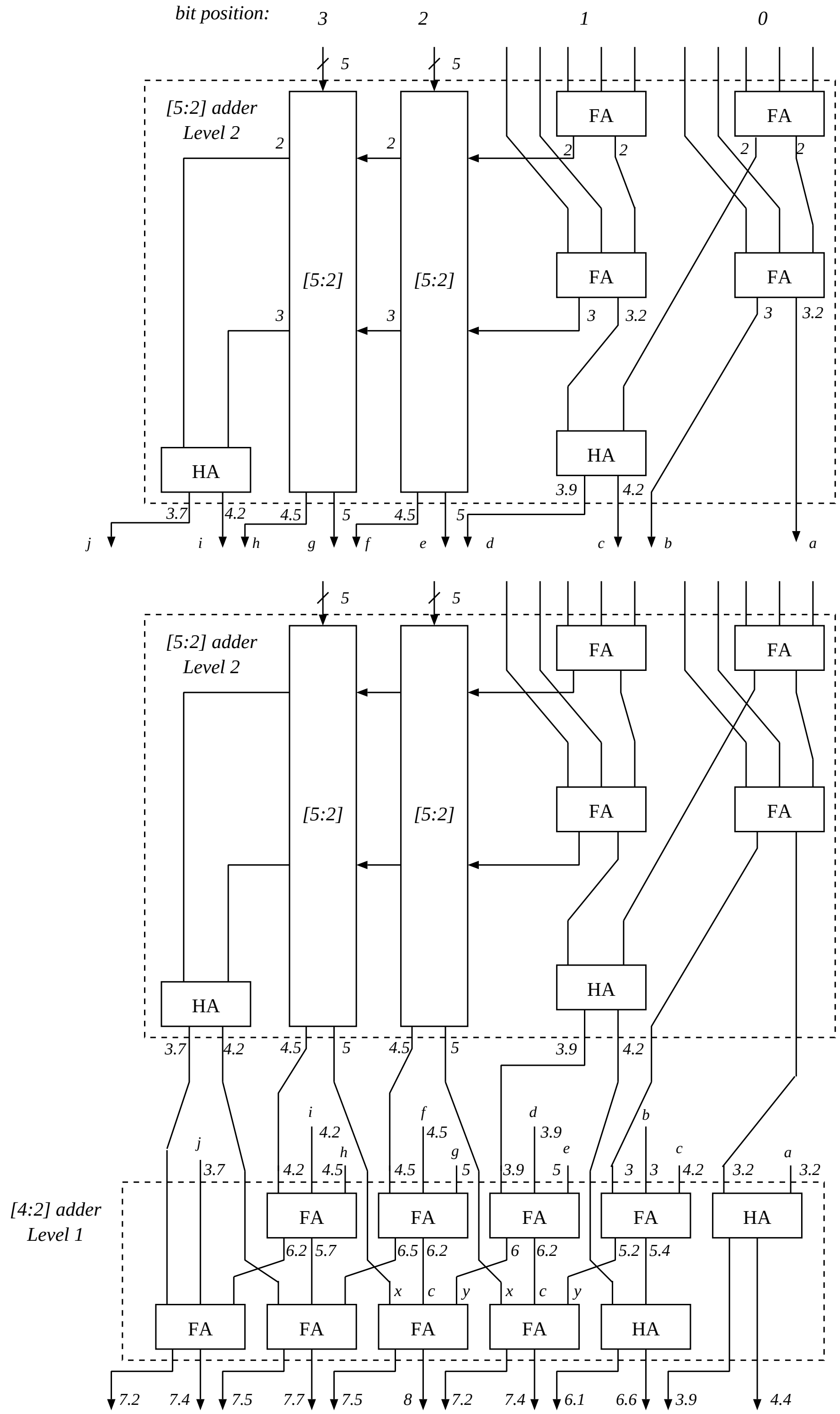


Figure E3.13b: Network of [5:2] and [4:2] modules to reduce 10 4-bit operands.

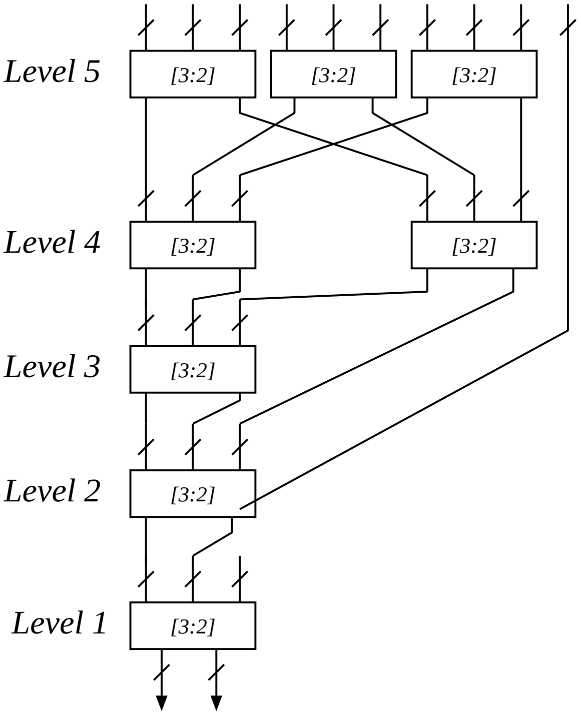


Figure E3.13c: Network of [3:2] adders to reduce 10 4-bit operands.

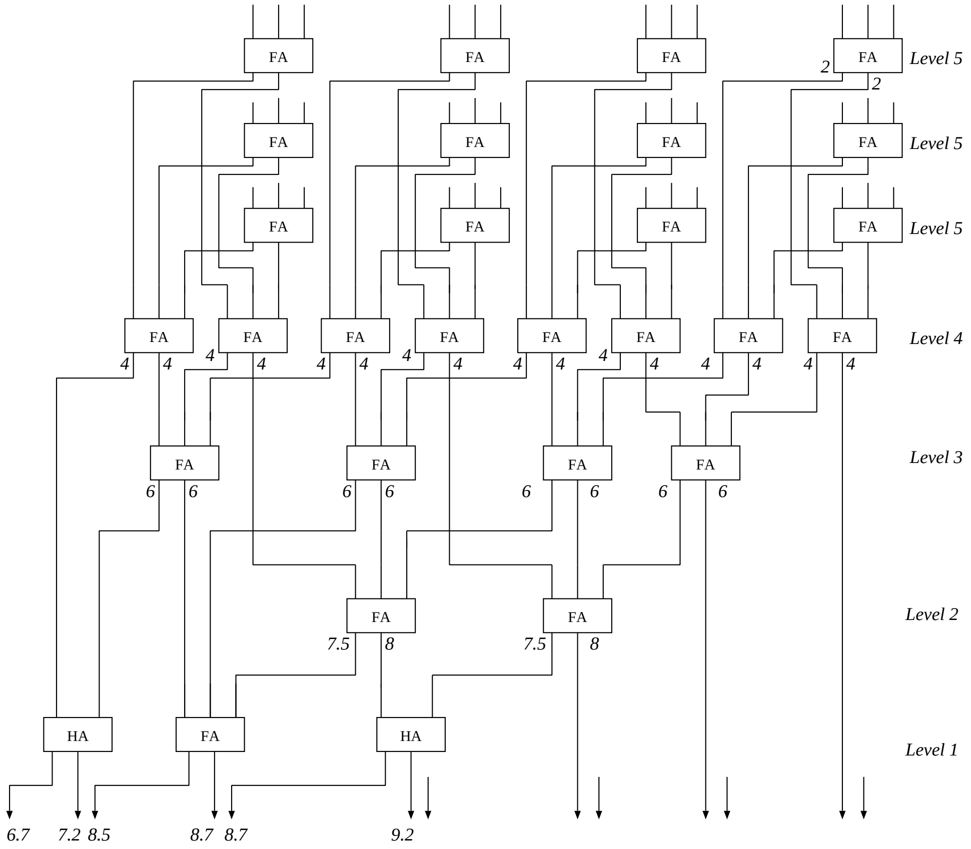


Figure E3.13d: Network of FAs and HAs to reduce 10 4-bit operands.

Exercise 3.18

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 3.22

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 3.26

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
```

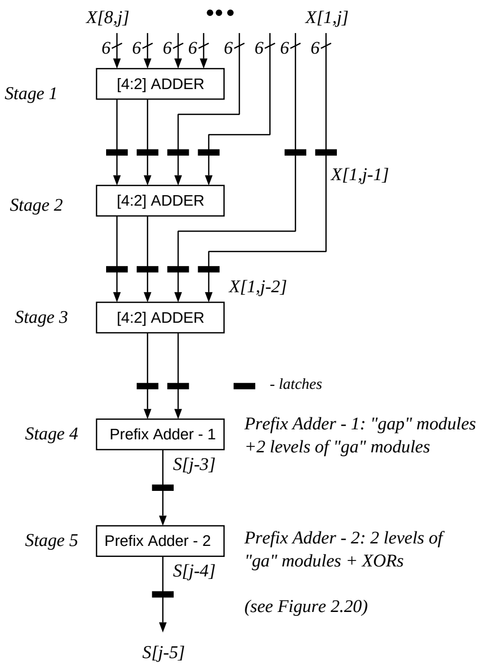


Figure E3.26: Pipelined linear array of [4:2] adders.