7.1 a) These butterfly pairs, which computed concurrently, are determined by the relative indices.
We assume that the butterflies are labeled with numbers 0 to 7 from top to down for each butterfly in Figure 7.4.
First alternative:
Execute butterflies p and $p + N/4$ simultaneously for all stages, the butterfly pairs are $\{0, 4\}$, $\{1, 5\}$, $\{2, 6\}$, and $\{3, 7\}$ for all stages.
Second alternative:
Execute butterflies p and $p + N_s/2$ at the first stage, p and $p + N_s$ for other stages.
Note that $N_s = 2^{4 - stage}$, the butterflies pares are therefore $\{0, 4\}$, $\{1, 5\}$, $\{2, 6\}$, and $\{3, 7\}$ for the first and the second stage, $\{0, 2\}$, $\{1, 3\}$, $\{4, 6\}$ and $\{5, 7\}$ for the third stage, and $\{0, 1\}$,$\{2, 3\}$,$\{4, 5\}$, and $\{6, 7\}$ for the final stage.

b) Obviously, m ranges from 0 to 3. $N_s = 2^{n - stage}$, i.e., $N_s = 8$ for the first stage, $N_s = 4$ for the second stage, $N_s = 2$ for the third stage, and $N_s = 1$ for the final stage. See a) for the range of p-values.

First alternative:
$k_1 = 2N_s\lfloor m/N_s \rfloor + [m \bmod(N_s)]$

$$k_2 = \begin{cases} k_1 + N/4 & Stage = 1 \\ k_1 + N/2 & Stage \geq 2 \end{cases}$$

$Stage = 1$:
$2N_s\lfloor m/N_s \rfloor = 0$ and $m \bmod(N_s) = m$, $k_1 = m$, $k_2 = m + N/4 = m + 4$,
$k_{1N_s} = k_1 + N_s = m + 8$ and $k_{2N_s} = k_2 + N_s = m + 4 + 8 = m + 12$.
In the same manner, we can determine the k_1, k_2, k_{1N_s}, and k_{2N_s} for the other stage. The result is listed in the following table.

Stage	k_1	k_2	k_{1N_s}	k_{2N_s}
1	0,1,2,3	4,5,6,7	8,9,10,11	12,13,14,15
2	0,1,2,3	8,9,10,11	4,5,6,7	12,13,14,15
3	0,1,4,5	8,9,12,13	2,3,6,7	10,11,14,15
4	0,2,4,6	8,10,12,14	1,3,5,7	9,11,13,15

Second alternative:
$k_1 = 4N_s\lfloor m/N_s \rfloor + [m \bmod(N_s)]$

$$k_2 = \begin{cases} k_1 + N_s/2 & Stage = 1 \\ k_1 + 2N_s & Stage \geq 2 \end{cases}$$

$Stage = 1:$

$4N_s\lfloor m/N_s \rfloor = 0$ and $m \bmod(N_s) = m$, $k_1 = m$, $k_2 = k_1 + N_s/2 = m + 4$,

$k_{1N_s} = k_1 + N_s = m + 8$ and $k_{2N_s} = k_2 + N_s = m + 4 + 8 = m + 12$.

In the same way, we can determine the values for k_1, k_2, k_{1N_s}, and k_{2N_s} at each stage. This results the following table:

Stage	k_1	k_2	k_{1N_s}	k_{2N_s}
1	0,1,2,3	4,5,6,7	8,9,10,11	12,13,14,15
2	0,1,2,3	4,5,6,7	8,9,10,11	12,13,14,15
3	0,1,8,9	4,5,12,13	2,3,10,11	6,7,14,15
4	0,4,8,12	2,6,10,14	1,5,8,13	3,7,11,15

c) We consider the simplification of one index in the first alternative, the other simplifications are left to the readers.

All indices is represented with binary numbers, for example, m is $m = m_1 \cdot 2^1 + m_0$, where $m_i = 0, 1$. We give only on example for the simplification here, i.e., k_2.

$Stage = 00: k_2 = (01m_1m_0)_2$

$Stage = 01: k_2 = (10m_1m_0)_2$

$Stage = 10: k_2 = (1m_10m_0)_2$

$Stage = 11: k_2 = (1m_1m_00)_2$

which means that $k_2 = s_1 + s_0, s_1 \cdot m_1 + \overline{s_1} \cdot \overline{s_0}, \overline{s_1} \cdot m_1 + s_1 \cdot m_0 \cdot s_0, (\overline{s_1} + \overline{s_0}) \cdot m_0$.

Hence the addition is not necessary in the computation of k_2.

d) Observe that the separation of $\{k_1, k_{1N_s}\}$ and $\{k_2, k_{2N_s}\}$ does not effected by m. Hence the butterflies operations does not changed except the orders.

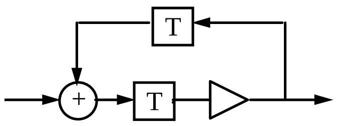
7.2 The modification does not differ too much from the Box 7.5. It is left to the readers. Some variables and procedures should be notified here: Ns initiated with N/2 in stead of N, the range of the loop variable m should be reduced to [0 ((N/8)-1)], two sets of twiddle factor should be generated instead of one, four butterflies are computed concurrently and four results should be written and read in stead of two.

7.3 In this case we have: $T_{min} = \frac{1}{2}(T_{add} + T_{mult})$. The critical path is $T_{add} + T_{mult}$.

In order to reach the maximal sample rate we will have to either use interleaving or pipelining.

a) **Unit time processors**

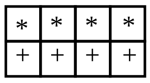
Here we assume that T_{add} = T_{mult} = 1 t.u. The minimal sample period, T_{min} is 1 t.u. and the critical path 2 t.u.



Pipelined filter.

Pipelining

After pipelining the critical path is split in two sections of equal length. We can start a new addition and a new multiplication in each sample interval. The schedule for processors is shown below.

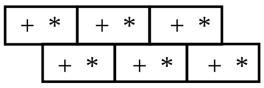


Schedule for pelined filter.

We will need only one processor of each type and their degree of utilization is 100%.

Interleaving

Interleaving of resources must be done if the critical path (adder — multiplier) is indivisible. We will have to start a new set of addition-multiplication every sample period. The schedule for processors is shown in the figure to the right.



Schedule using interleaving of resources.

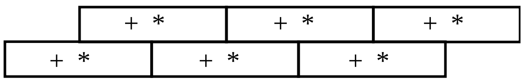
Now, we need two adders and two multipliers. More processors are needed since the processing is sequential. Further, their degree of utilization is only 50%.

b) **Non unit time processors**

Now, assume that T_{mult} = 3 T_{add} and T_{add} = 1 t.u. The minimal sample period will be T_{min} = 4/2 = 2 t.u. and the critical path 4 t.u.

Interleaving

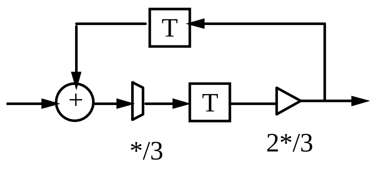
If the adder and the multiplier are indivisible we will have the schedule shown below. We need two processors of each type which will be utilized to 25% and 75%, respectively.



Schedule using interleaving of resources.

Pipelining

If we pipeline, the critical path will have to be split in two parts of length 2. This is assuming that the multiplier can be split into two parts. The pipelined filter is shown to the right while the operation schedule for this is shown below.



Pipelined filter with multiplier split into two parts. The first part has an execution time of 1/3 while the second has an execution time of 2/3 of a complete multiplicationpar

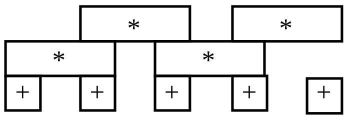
2*/3	2*/3	2*/3	2*/3
+*/3	+*/3	+*/3	+*/3

Schedule for filter with multiplier split into two parts.

Here we have one processor performing +*/3 and one 2*/3. They are both used 100% of the time.

Pipelining and interleaving

In many cases we can not divide a processor in two parts. If we do not divide the multiplier but still pipeline as much as possible, i.e., move one delay element so that the



Schedule for not fully pipelined filterd.

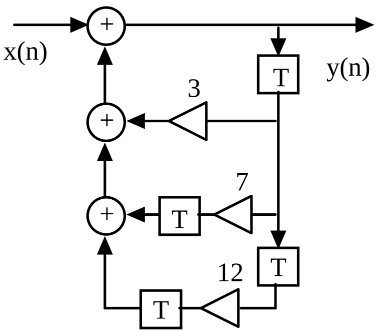
critical path is split into one 1 t.u. section and one 3 t.u. section, we get the schedule shown below. Here we have, one adder which is utilized to 50% and two multipliers utilized 75% each.

Conclusions

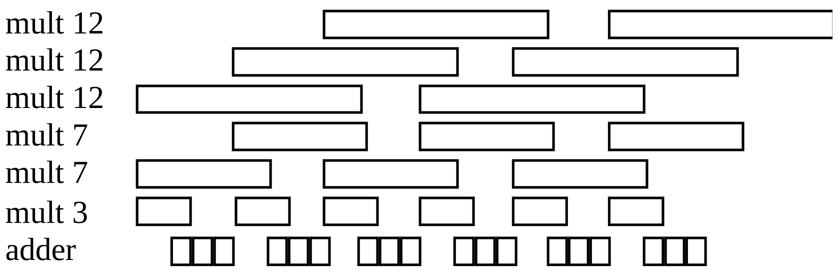
It is desirable to have equal processor execution times. Pipelining will improve the processor utilization.

7.4 We have: $T_{min} = max\left\{\frac{5}{1}, \frac{10}{2}, \frac{15}{3}\right\} = 5$

We need 1 adder, 1 multiplier of the first type, 2 of the second and 3 of the third type. We introduce delays into the critical path so that is broken into smaller pieces. This is often call for retiming. The degree of utilization of the adder is 60% and it is 60%, 70%, and 80% of the multipliers.



Retimed filter section.



Feasible schedule

7.7 a) The iteration bound is determined by

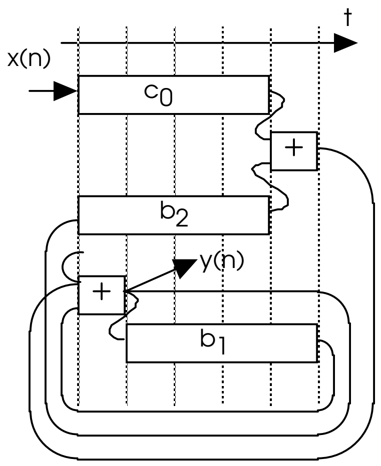
$$T_{min} = max\left\{\frac{T_{op_i}}{N_i}\right\} = max\left\{\frac{4+1+1}{1}, \frac{4+1+1}{2}\right\} = 5 .$$

b) The minimal number of PEs is $N_{PE} = \left|\frac{\sum_i N_{op_i} T_{op_i}}{T_{sample}}\right| = \left\lceil\frac{5 \times 4 + 4 \times 1}{8}\right\rceil = 3 .$

c)

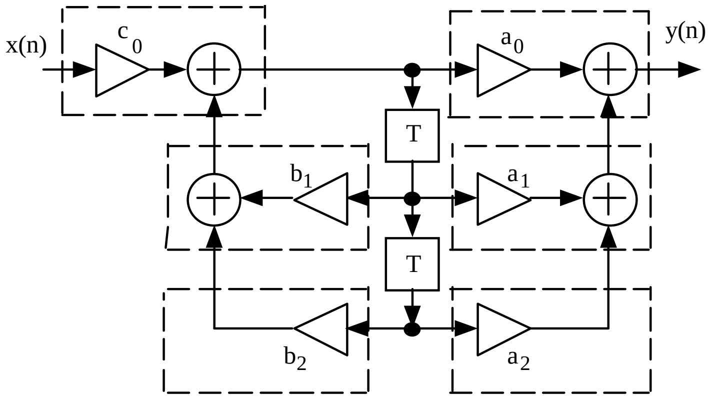
7.8 a) Specify the ordering of the additions. Chose to add c_0 x(n) and b_2 y(n–2) first. We get.

T_{min} = max((4+1)/1, (4+1+1)/2) = 5 t.u.



b)
c)

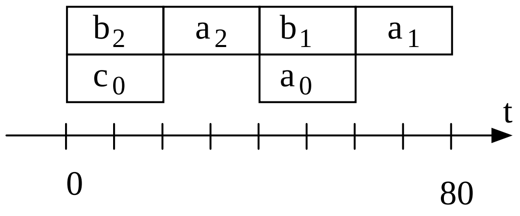
7.12 a) There are several ways to combine additions and multiplications into basic operations, but the critical path will contain at least 4 operations.



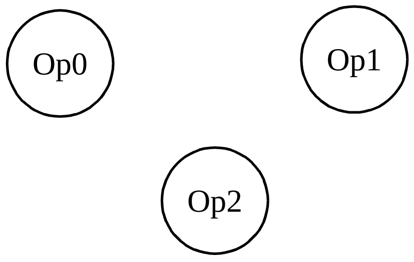
The number of clock cycles per sample is: $\frac{100 \times 10^6}{1.25 \times 10^6} = 80$

6 PE operations requies $\frac{6 \times 20 \times 1.25 \times 10^6}{100 \times 10^6} = 1.5$ => 2 PE

A feasible scheduling is shown below.



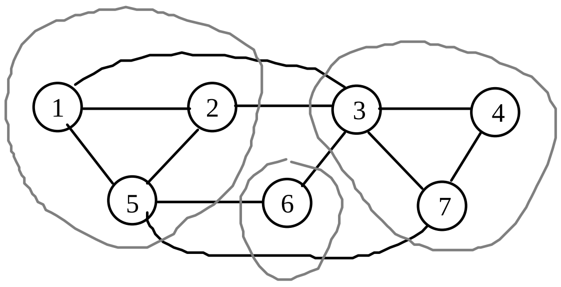
7.13 Assume that we implement for $T = T_\infty$, otherwise we can implement it with 1 PE. Let each operation correspond to a vertex and construct the connectivity graph. Obviously all operations are overlap with each other so that there is no branch between each two vertices.



Connectivity graph

From the connectivity graph we have to allocate three PEs.

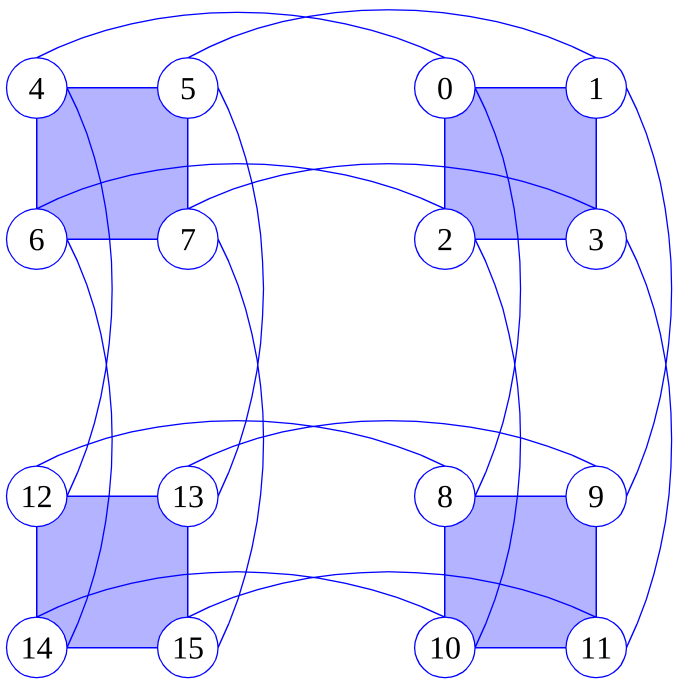
7.14



Thus, 3 resurcers are required.

7.16 We give only one of the assignment alternatives. The other alternatives are left to the readers.

a) The exclusion graph for the 16-point FFT can be constructed as the same way in 7.11.1 for the memory assignment 2.



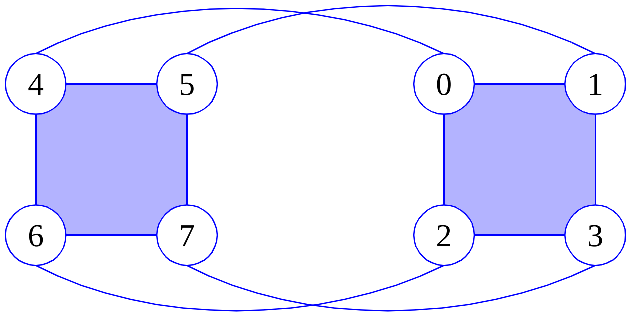
The four RAMs assignment can be expressed in terms of the binary representation of index $i = i_3 i_2 i_1 i_0$. A variable of index i is assigned to RAM$_{P(i)}$ where

$$P(i) = p_1 p_0$$
$$p_0(i) = i_0$$
$$p_1(i) = i_2 \oplus i_1$$

Data index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Data $x(i)$	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
	RAM 0	RAM 1	RAM 2	RAM 3	RAM 2	RAM 3	RAM 0	RAM 1	RAM 2	RAM 3	RAM 0	RAM 1	RAM 0	RAM 1	RAM 2	RAM 3

A memory assignment for a 16-point FFT.

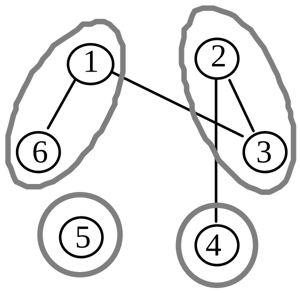
b) Using the PE assignment 2 in 7.11.2, we have the following exclusion graph.



A possible assignment, where butterflies in rows p, $p + N/8$, $p + N/4$ and $p + 3N/8$ are executed parallel, is given here. A butterfly operation in row r is assigned to the PE$_{P(r)}$ where

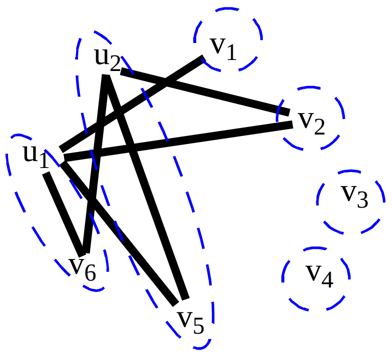
$$P(i) = p_1 p_0$$
$$p_0(i) = r_1$$
$$p_1(i) = r_2$$

7.17 a) Number the variables from the top and downwards. Connect nodes that must lie in the same memeory cell. 4 memory cells is required.



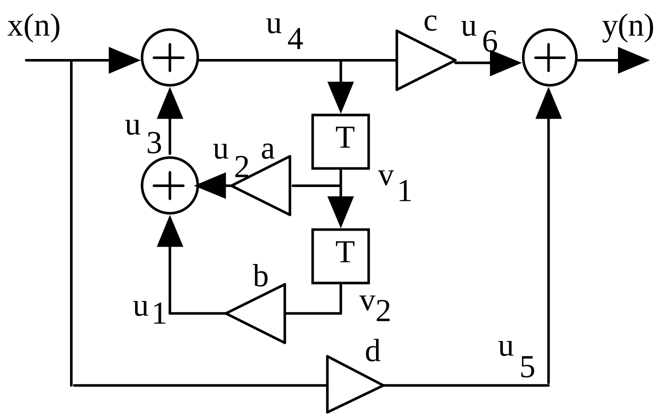
b) 6 variables yield 12 memory accesses per sample period. A memory must have the acess time 12×40 = 480 ns to be able to read and write all variables. This corresponds to a sample frequency of 1/480 ns = 2.08 MHz.

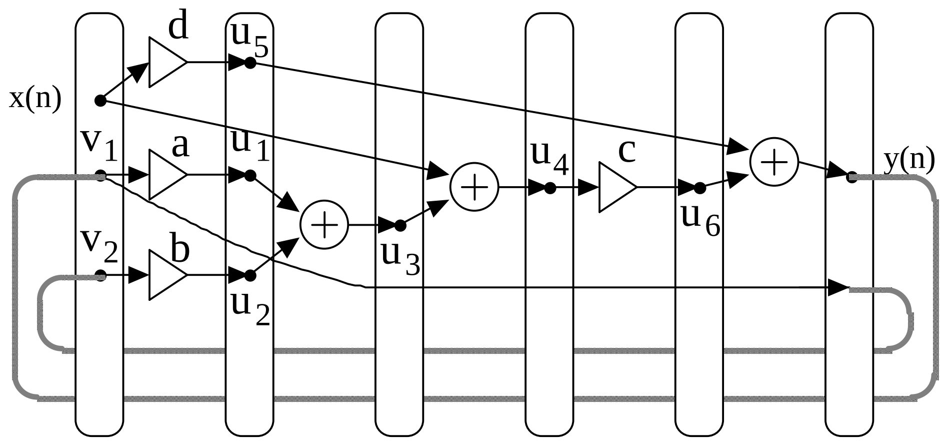
7.18 We can use the clique partitioning and choose the maximum cliques.



Memory cells	Variables
0	v1
1	v2
2	v3
3	v4
4	v5, u2
5	v6, u3

7.19 a) Assign names to all nods:





b) $u_1 := a\ v_1(n)$ Which can be simplified to:

$u_2 := b\ v_2(n)$

$u_5 := d\ x(n)$

$u_3 := u_1 + u_2$

$u_4 := x(n) + u_3$

$u_6 := c\ u_4$

$y(n) := u_5 + u_6$

$v_2(n+1) := v_1(n)$

$v_1(n+1) := x(n)$

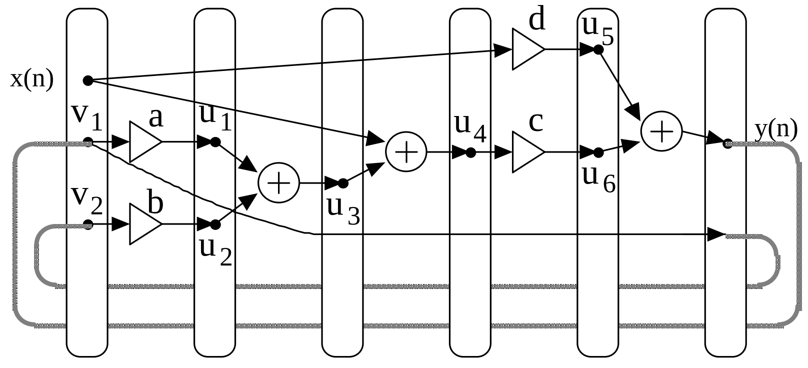
$u_4 := x(n) + a\ v_1(n) + b\ v_2(n)$

$y(n) := d\ x(n) + c\ u_4$

$v_2(n+1) := v_1(n)$

$v_1(n+1) := u_4$

c)



7.20 Figure 7.20a shows computation graph N with delay of operations inserted and T exchanged for —T. The maximal spanning tree is shown in Fig. 7.20b where edges belonging to the tree are drawn with thick lines and the link branches with thin lines.

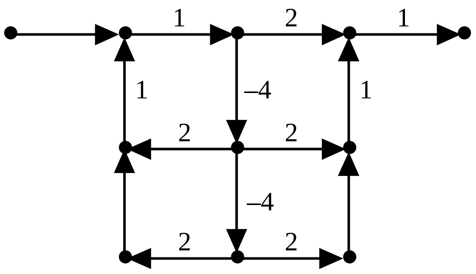


Fig. 7.20a. The network N.

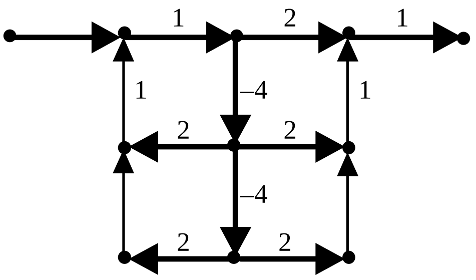


Fig. 7.20b. The maximum spanning tree of N.

Insert the link branches one by one and add shimming delays so that the total delay in the fundamental loops that are formed become zero. Finally, we remove the negative delays elements and arrive at the scheduled graph in Fig. 7.20c. Note that there are no simming delays in the critical loops and that the schedule include

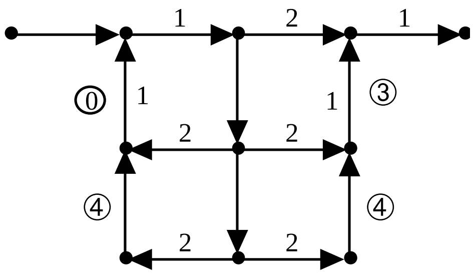
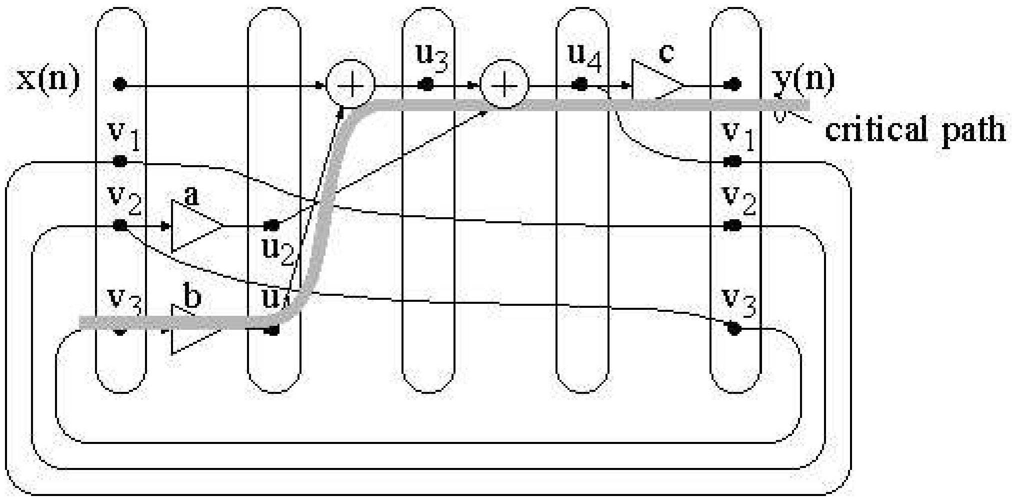


Fig. 7.20c.Final scheduled computation graph.

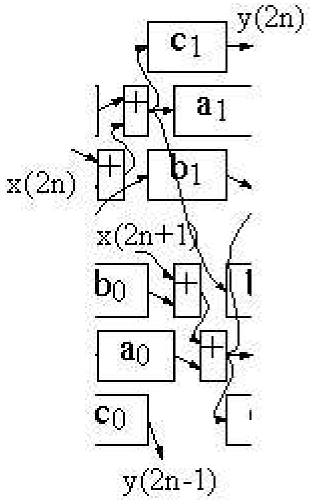
7.21 $$T_{min} = max\left\{\frac{T_{opi}}{N_i}\right\} = max\left\{\frac{3+1}{2}, \frac{3+1+1}{3}\right\} = 2 \text{ clock cycles}$$

$$f_{max} = \frac{1}{T_{min}} = \frac{f_{clk}}{2}$$

The critical path is shown below.



Since the critical loop is large than 1 sampling period, we have to schedule the operations in two sample periods. The scheduling with the maximal sample frequency is shown below.

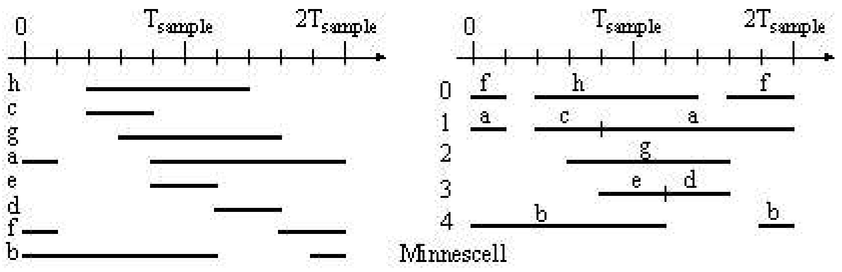


7.22 a) The upper bound of required number of memory cells is equal to the number of variables, i.e., 8.

The lower bound is equal to the total required lifetime divided by the available time, i.e., $\lceil 33/10 \rceil = 4$.

.

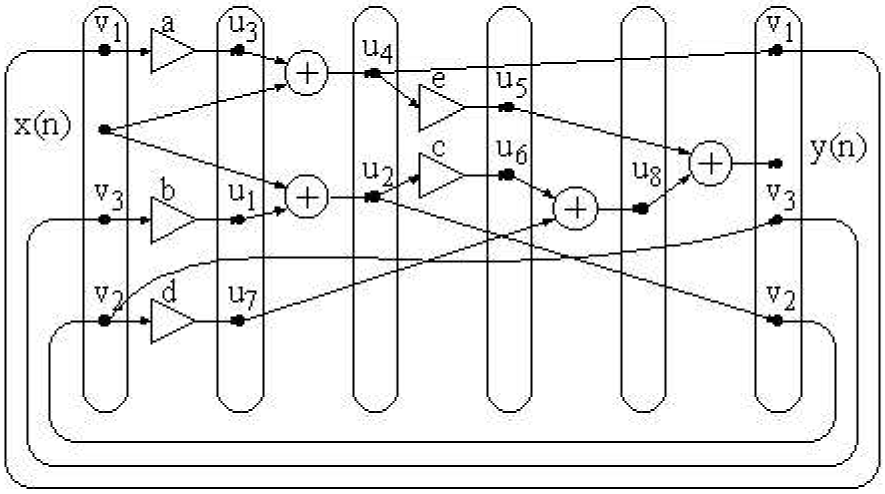
b) Sort the lifetime diagram according to the start time and lifetime and allocate the memories.



c) Each variable requires read and write within 2 sampling periods, i.e., total 16 memory accesses are needed. $T_{access} = \dfrac{2}{15 \times 10^{6} \times 16} = 8.3\ \text{ns}$.

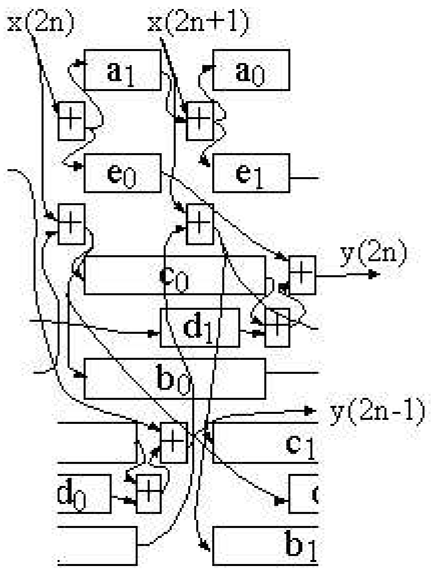
7.23 a) $T_{min} = max\left\{\dfrac{T_{op_i}}{N_i}\right\} = max\{\dfrac{3+1}{1}, \dfrac{7+1}{2}\} = 4$ clock cycles

b) The precedence graph is shown below:



c) Since the critical loop requires 8 clock cycles which is larger than the sampling period, we have to schedule the computation in two sampling periods. The minimal number of PEs is $N_{PE} = \left\lceil \dfrac{\sum_i N_i T_{op_i}}{2T_{sample}} \right\rceil = \left\lceil \dfrac{(3 \times 3 + 2 \times 7 + 4 \times 1) \times 2}{2 \times 5} \right\rceil = \lceil 5{,}4 \rceil = 6$.

d)


x(2n)
x(2n+1)
a1
a0
e0
e1
y(2n)
c0
d1
b0
y(2n-1)
c1
d0
b1