This homework consists of questions taken from the notes and open-ended questions. You must do the homework by yourself. **No collaborations are allowed.** There are total 100 points. This homework is worth 10% of your overall grades.

1. (5 points)
   (a) (3 points) Consider the following Fortran-like program
   
   ```fortran
   DO I=1,N
   DO J=2,N
       A(I,J)=B(I,J)+C(I,J)
       C(I,J)=D(I,J)/2
       E(I,J)=A(I,J-1)**2+E(I,J-1)
   END
   END
   ```
   Assume that each array has already been initialized before executing this program.
   Rewrite the program into a vectorized format.

   **Answer:** Note the anti-dependence of 1\textsuperscript{st} and 2\textsuperscript{nd} statement. There is a true dependence relation with iteration dependence vector (0, 1) in the third statement. That is the only dependence vector in each iteration. Thus, all operations with the same I index can be vectorized. This leads to
   
   ```fortran
   Do J=2,N
   Doall I=1,N
       A(:,J)=B(:,J)+C(:,J)
       C(:,J)=D(:,J)/2
       E(:,J)=A(:,J-1)**2+E(:,J-1)
   End Doall
   ```

   (b) (2 points) Consider the following Fortran-like program
   
   ```fortran
   DO I=1,5
   A(I)=B(I)+I
   D(I)=A(I)+A(I+1)
   END
   ```
   Assume that arrays \{A(I)\} and \{B(I)\} have already been initialized before executing this program. Rewrite the program into a vectorized format.

   **Answer:**
   
   ```fortran
   A1(1:5) = B(1:5) + [1 2 3 4 5]
   D(1:5) = A1(1:5) + A(2:6)
   ```

2. (10 points) Consider the following Fortran-like program
   
   ```fortran
   C listing 1
   DO 10 I=1,3
   DO 10 J=a*I+b,c*I+d
   X(I,J)=X(I+2,J-1)+X(I-1,J)
   10 CONTINUE
   ```
(a) (4 points) let \(a, b, c, \) and \(d\) be four constant integers. Find the condition(s) such that this nested loop is a regular nested loop.

**Answer:** Rewrite the index constraints in the formats of \(P_i \leq p_0, q_0 \leq Q_i:\)

\[
1 \leq i \leq 3; \quad ai + b \leq j \leq ci + d \Rightarrow \\
\begin{bmatrix}
1 & 0 \\
1 & -1
\end{bmatrix} \begin{bmatrix} i \\
 j
\end{bmatrix} \leq \begin{bmatrix} 3 \\
-1
\end{bmatrix} = p_0 \quad \text{and} \quad \begin{bmatrix}
1 & 0 \\
c & -1
\end{bmatrix} \begin{bmatrix} i \\
 j
\end{bmatrix} \geq \begin{bmatrix} 1 \\
-1
\end{bmatrix} = q_0
\]

For a regular nested loop, it is required that \(P = Q\). Hence, one must have \(a = c\).

(b) (1 points) Derive the dependence matrix \(D\) of this algorithm.

**Answer:**

\[
D = \begin{bmatrix}
-2 & 1 \\
1 & 0
\end{bmatrix}
\]

(c) (2 points) If \(a = c = 0, b = 1, \) and \(d = 2\). In order to execute this algorithm, which elements of the \(X\) array must be given as initial conditions?

**Answer:** The states to be executed are:

```c
C listing 1
DO 10 I=1,3
   DO 10 J=1,2
      X(I,J)=X(I+2,J-1)+X(I-1,J)
   10 CONTINUE
X(1,1)=X(3,0)+X(0,1)
X(1,2)=X(3,1)+X(0,2)
X(2,1)=X(4,0)+X(1,1)
X(2,2)=X(4,1)+X(1,2)
X(3,1)=X(5,0)+X(2,1)
X(3,2)=X(5,1)+X(2,2)
```

The initial conditions are array elements that are marked with yellow shades.

(d) (3 points) This algorithm as specified in part (c) may have a problem executing properly. Discuss what may be the problem and provide a reformulated program that can be executed correctly. Assuming all the initial conditions of the array \(X\) are available in the memory.

**Answer:** From the answer of part (c), the dependence dictates that the order of execution of this program is as follows:

\(11 \rightarrow 21 \rightarrow 31 \rightarrow 12 \rightarrow 22 \rightarrow 32\)

Clearly, the indices \(I\) and \(J\) must be interchanged. So the modified algorithm looks like this:

```c
C listing 1
DO 10 J=1,2
   DO 10 I=1,3
      X(I,J)=X(I+2,J-1)+X(I-1,J)
   10 CONTINUE
```

3. Iteration bound problems

(a) (5 points, CC) Text book [Parhi], Chapter 2, problem 1.
(b) (5 points) Text book [Parhi], Chapter 2, problem 2.
(c) (5 points) Text book [Parhi], Chapter 2, problem 3.
(d) (5 points) Text book [Parhi], Chapter 2, problem 4.
4. Chapter 3 problems.
   (a) (4 points) Text book [Parhi], Chapter 3, problem 1.
   (b) (4 points) Text book [Parhi], Chapter 3, problem 2.
   (c) (4 points) Text book [Parhi], Chapter 3, problem 3. Hint: y(n) is the sum of two FIR filters.
   (d) (4 points) Text book [Parhi], Chapter 3, problem 5.

5. Chapter 4 problems
   (a) (4 points) Text book [Parhi], Chapter 4, problem 1 (a), (b)
   (b) (6 points) Text book [Parhi], Chapter 4, problem 2.
   (c) (4 points) Text book [Parhi], Chapter 4, problem 3.
   (d) (6 points) Text book [Parhi], Chapter 4, problem 5.
   (e) (4 points) Text book [Parhi], Chapter 4, problem 7.
   (f) (6 points) Text book [Parhi], Chapter 4, problem 8.
   (g) (4 points) Text book [Parhi], Chapter 4, problem 10.
   (h) (8 points) Text book [Parhi], Chapter 4, problem 11.

6. (7 points) Refer to figure 4.20, page 115 of the text book [Parhi]. Assume the addition takes 2 t.u. and multiplication takes 5 t.u.
   (a) (2 points) Find the iteration bound $T_{\infty}$.
   Answer: $= 5 + 2 + 2 = 9$ t.u.
   (b) (2 points) Identify the critical path $P_{cr}$. Give the answer in the form of $n_1 \rightarrow n_2 \rightarrow \ldots$
   where $n_1, n_2$ are node numbers.
   Answer: $4 \rightarrow 2 \rightarrow 1 \rightarrow 7$ or $3 \rightarrow 2 \rightarrow 1 \rightarrow 7$. The length is 11 t.u.
   (c) (3 points) In order to reduce the clock cycle time to the iteration bound, retiming are performed. Among many possible solutions, some of them will NOT add delays to the computation of the output (i.e. $y(n)$ will be computed during the same clock cycle $x(n)$ is sampled), and will NOT increase (decrease is OK) the total number of registers. Find two such solutions by giving their corresponding retimed DFGs.
   Answer:

7. (10 points) Download the H.264 reference software jm10.2.zip from the web. Compile the program on a platform of your choice (Window or Unix). Run the test sequence on encoder first to produce encoded bit stream. Then run the decoder to decode the bit stream. Post the encoded bit stream and decoded video sequence on your class website.

8. (15 points) Motion estimation is the most computational intensive task in video coding. In this problem, we consider motion estimation over fixed $16 \times 16$ macro-block only. Assume that the video frame has a size of $640 \times 480$ (VGA), the frame rate is 15 frames per second. Also assume the video sequence is subsampled at 4:1:1 ratio among the YCbCr components.
   (a) (2 points) How many macro-blocks will need to be processed per second for the given frame size and frame rate?
   Answer: Let us consider Y component only. The number of macro-blocks to be processed per second is:
   \[
   \frac{640 \times 480}{16 \times 16} \times 15 = 18000 \text{ MBs/sec}
   \]
   The Cb, Cr components together contains 50% of the MBs of the Y components. So the total number of MBs to be processed per second is $18000 \times 1.5 = 27000$ MB/s
(b) (2 points) Assume that there is only a single reference frame used in motion estimation and the search range is ±8 pixels. For each pixel in the reference frame, how many times it will be used for the motion estimation computation? Assume the pixel is at the interior of the reference frame, and hence the boundary situation can be ignored. Also assume that full-search algorithm is used.

Answer: A pixel will be referred to by every MB that is within ±8 pixels of its distance. Divide each MB within the reference frame into four 8 × 8 sub-blocks. Then, all pixels within the upper-left sub-block will be referred to by the left, upper-left corner, upper MBs in addition to the current MB in the current frame. Thus, each pixel in the reference frame will be used four times.

(c) (2 points) Repeat this problem for the case when the search range is increased to 64.

Answer: With search range increased to 64, each pixel will be referred to by MBs in the current frame within ±64 pixel square range. Hence, it will be referred to by ±4 MB from each of the left-right and upper-bottom direction. Including the corner ones, there are \(8^2 = 64\) MBs in current frames will access this same pixel. Thus, each pixel will be referred 64 times.

(d) (2 points) Now, let us consider implementing the motion estimation on an embedded platform. Assume fixed 16 × 16 MB, ±8 pixels search range, VGA frame size and 15 fps frame rate. Also assume that the video frame has already been converted into YCbCr format sub-sampled at 4:1:1 ratio, with each pixel value at each component represented as an 8-bit unsigned integer. SAD is used as a criterion to determine motion vector. Assume that addition, subtraction, absolute value, comparison each takes one clock cycle to perform, and no two operations can be performed simultaneously. Ignoring memory access delay, compute the minimum clock frequency in GHz (rounded to 3\(^{rd}\) fractional digit) that can support the motion estimation operation in real time.

Answer: For each MB, there are \((2 \times 8 + 1)^2 = 289\) potential motion vector (MVs) to evaluate. For each of the candidate MV, there are \(16^2 = 256\) subtraction, and absolute value evaluation and 255 additions. This will take \(256 + 256 + 255 = 767\) clock cycles to compute the SAD for each MV. Thus, for each MB, the number of clock cycles takes to compute the MV that yields minimum SAD value will be \(289 \times 767 + 288 = 221,951\) clock cycles. From part (a) of this problem, we know that there are 27,000 MBs need to be processed each second, thus, the motion estimation will require \(221,951 \times 27,000 = 5,992,677,000\) clock cycles. For real time processing, these clock cycles must be within 1 second. In other words, the minimum clock frequency will be \(5.993\) GHz.

(e) (7 points) Next, consider variable block size motion estimation as described in H.264 standard. One important observation is that the SAD value of a macro-block is the sum of all the SAD values of its sub-blocks. Implement a HLL (high level language) program that performs full search motion estimation of a single macro-block. The inputs of this program include a 16 × 16 current MB, and a 32 × 32 search region. The output of this program will be the 41 MVs and the corresponding SAD values. Create your own sample inputs (the 16 × 16 MB and 32 × 32 search region), run the program and compute the desired output. Submit (i) the source code of this program with sufficient comments; (ii) an explanation of the algorithm implemented in this program, especially how it make use of the observation mentioned above; (iii) printed output of the example you use.
addition. Post the program and the data you use on the homework web page AFTER the homework is due.

**Solution for the problem in the text book**

Chapter 2

These result are generated from file lpm.m and mcm.m on the course webpage

```
** Answer of Pro. 1.(a) **
L1=
   -1  0  -1
   7 -1  3
   3 -1 -1
L2=
   7 -1  3
   6  7 -1
  -1  3 -1
L3=
   6  7 -1
  14  6 10
 10 -1  6
ans =
3.5000

** Answer of Pro. 1 (b) **
>> ib_cyclemean(inv_Gd)
inv_Gd=
   Inf  0  Inf
  -7  Inf  -3
  -3  Inf  Inf
f0=[0  Inf  Inf]',
f1=[Inf  0  Inf]',
f2=[-7  Inf  -3]',
f3=[-8  -7  Inf]',
ans =
3.5000
2(a)
```
<table>
<thead>
<tr>
<th>L1</th>
<th>0</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>L3</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>L4</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>L5</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
\[ L_6 = \]
\[
\begin{array}{cccccccc}
1 & 1 & 3 & -1 & -1 & -1 & 5 & -1 \\
1 & 1 & -1 & 3 & -1 & -1 & -1 & 6 \\
10 & -1 & -1 & -1 & 13 & -1 & -1 & -1 \\
-1 & 10 & -1 & -1 & -1 & 13 & -1 & -1 \\
1 & 1 & 1 & -1 & -1 & -1 & -1 & 7 \\
1 & -1 & -1 & 4 & -1 & -1 & -1 & 7 \\
11 & -1 & -1 & -1 & 14 & -1 & -1 & -1 \\
-1 & 11 & -1 & -1 & -1 & 14 & -1 & -1 \\
\end{array}
\]

\[ L_7 = \]
\[
\begin{array}{cccccccc}
-1 & 1 & -1 & 3 & -1 & -1 & -1 & 6 \\
1 & -1 & -1 & -1 & 13 & -1 & -1 & -1 \\
10 & -1 & -1 & -1 & 13 & -1 & -1 & -1 \\
-1 & -1 & 10 & -1 & -1 & -1 & 13 & -1 \\
-1 & -1 & -1 & 4 & -1 & -1 & -1 & 7 \\
11 & -1 & -1 & -1 & 14 & -1 & -1 & -1 \\
-1 & 11 & -1 & -1 & -1 & 14 & -1 & -1 \\
\end{array}
\]

\[ L_8 = \]
\[
\begin{array}{cccccccc}
10 & -1 & -1 & -1 & 13 & -1 & -1 & -1 \\
-1 & 10 & -1 & -1 & 13 & -1 & -1 & -1 \\
-1 & -1 & 10 & -1 & -1 & 13 & -1 & -1 \\
-1 & -1 & -1 & 10 & -1 & -1 & 13 & -1 \\
11 & -1 & -1 & -1 & 14 & -1 & -1 & -1 \\
-1 & 11 & -1 & -1 & -1 & 14 & -1 & -1 \\
-1 & -1 & 11 & -1 & -1 & -1 & 14 & -1 \\
\end{array}
\]

\[ \text{ans} = \]
\[ 1.7530 \]

2(b)

\[
\text{inv_Gd} = \\
\begin{bmatrix}
\inf & 0 & \inf & \inf & \inf & \inf & \inf & \inf \\
\inf & \inf & 0 & \inf & \inf & \inf & \inf & \inf \\
\inf & \inf & \inf & 0 & \inf & \inf & \inf & \inf \\
-3 & \inf & \inf & \inf & \inf & \inf & \inf & \inf \\
\inf & \inf & \inf & \inf & \inf & 0 & \inf & \inf \\
\inf & \inf & \inf & \inf & \inf & \inf & \inf & \inf \\
-4 & \inf & \inf & \inf & \inf & \inf & -7 & \inf & \inf \\
\end{bmatrix}
\]

\[ f_0 = \]
\[ [0 \ \inf \ \inf \ \inf \ \inf \ \inf \ \inf \ \inf] \]

\[ f_1 = \]
\[ [\inf \ 0 \ \inf \ \inf \ \inf \ \inf \ \inf \ \inf] \]

\[ f_2 = \]
\[ [\inf \ \inf \ 0 \ \inf \ \inf \ \inf \ \inf \ \inf] \]

\[ f_3 = \]
\[ [\inf \ \inf \ \inf \ 0 \ \inf \ \inf \ \inf \ \inf] \]

\[ f_4 = \]
\[ [-3 \ \inf \ \inf \ \inf \ -6 \ \inf \ \inf \ \inf] \]
\[ f_5 = [\text{Inf} \ -3 \ \text{Inf} \ \text{Inf} \ \text{Inf} \ -6 \ \text{Inf} \ \text{Inf}] \]
\[ f_6 = [\text{Inf} \ \text{Inf} \ -3 \ \text{Inf} \ \text{Inf} \ \text{Inf} \ -6 \ \text{Inf}] \]
\[ f_7 = [\text{Inf} \ \text{Inf} \ \text{Inf} \ -3 \ \text{Inf} \ \text{Inf} \ \text{Inf} \ -6] \]
\[ f_8 = [-10 \ \text{Inf} \ \text{Inf} \ \text{Inf} \ -13 \ \text{Inf} \ \text{Inf} \ \text{Inf}] \]

\[ \text{ans} = 1.7500 \]

3(a)

\[ L_1 = \begin{bmatrix} 4 & 4 & -1 \\ -1 & 0 & 0 \\ 4 & 4 & -1 \end{bmatrix} \]

\[ L_2 = \begin{bmatrix} 8 & 8 & 4 \\ 4 & 4 & -1 \\ 8 & 8 & 4 \end{bmatrix} \]

\[ L_3 = \begin{bmatrix} 12 & 12 & 8 \\ 8 & 4 & 4 \\ 12 & 12 & 8 \end{bmatrix} \]

\[ \text{ans} = 4 \]

3(b)
\( \text{inv}_G \) =
\[
\begin{array}{ccc}
-4 & -4 & \text{Inf} \\
\text{Inf} & \text{Inf} & 0 \\
-4 & -4 & \text{Inf} \\
\end{array}
\]

\( f0 = \)
\[
[0 \quad \text{Inf} \quad \text{Inf}]^\top
\]

\( f1 = \)
\[
[-4 \quad -4 \quad \text{Inf}]^\top
\]

\( f2 = \)
\[
[-8 \quad -8 \quad -4]^\top
\]

\( f3 = \)
\[
[-12 \quad -12 \quad -8]^\top
\]

\( \text{ans} = \)
\[
4
\]

4(a)

\( L1 = \)
\[
\begin{array}{ccc}
4 & 0 & -1 \\
5 & -1 & 0 \\
5 & -1 & -1 \\
\end{array}
\]

\( L2 = \)
\[
\begin{array}{ccc}
8 & 4 & 0 \\
9 & 5 & -1 \\
9 & 5 & -1 \\
\end{array}
\]

\( L3 = \)
\[
\begin{array}{ccc}
12 & 8 & 4 \\
13 & 9 & 5 \\
13 & 9 & 5 \\
\end{array}
\]

\( \text{ans} = \)
\[
4
\]

4(b)

\( \text{inv}_G \) =
\[
\begin{array}{ccc}
-4 & 0 & \text{Inf} \\
-5 & \text{Inf} & 0 \\
-5 & \text{Inf} & \text{Inf} \\
\end{array}
\]

PDF created with pdfFactory Pro trial version www.pdffactory.com
Chapter 3

1 (a)

\[ T_{\text{sample}} = 4T, \quad f_{\text{sample}} = 1/4T \]

(b) The pipelining levels are shown by the dashed lines in Figure 3.1. 9 registers are required.

2.

(a). The critical path is \( M_1 - M_2 - A_1 - M_2 - A_1 - M_2 - A_1 - M_2 - A_1 \) as shown by the dashed line in Figure 3.2(a).

(b). Pipelining latches are placed on the dotted lines in Figure 3.2(b). As can be seen, the critical path is one multiply and one add, which is 3 time units. \( \blacksquare \)
3.

Let:

\[ y_1(n) = a_1 x_1(n) + a_2 x_1(n-1) + a_3 x_1(n-2) + a_4 x_1(n-3) + a_5 x_1(n-4) \]  
(3.3)

\[ y_2(n) = b_1 x_2(n) + b_2 x_2(n-1) + b_3 x_2(n-2) + b_4 x_2(n-3) + b_5 x_2(n-4) \]  
(3.4)

\[ y(n) = y_1(n) + y_2(n) \]  
(3.5)

Then transpose operation can be applied to \( y_1(n) \) and \( y_2(n) \) separately. The equivalent data-broadcast implementation is shown in Figure 3.3.
(a). The pipelining level is shown by dashed line in Figure 3.5(a).
(b). The block filter architecture is illustrated in Figure 3.5(b). $f_{\text{sample}} = 3/T$

\[
\begin{align*}
    y(3k) &= ax(3k) + bx(3k - 2) + cx(3k - 3) \\
    y(3k + 1) &= ax(3k + 1) - bx(3k - 1) + cx(3k - 2) \\
    y(3k + 2) &= ax(3k + 2) - bx(3k) + cx(3k - 1)
\end{align*}
\]
Chapter 4

1. 
   (a) 
   \[ T_{\text{bounds}} = \frac{T_m + 2T_a}{2} = 18\text{ns} \]
   
   (b) 
   \[ T_{\text{critical}} = 2(T_m + 3T_a) = 88\text{ns} \]

2. 
   (a) The maximum sample rate is limited by the critical path:
   \[ \text{Sample Rate}_{\text{max}} = \frac{1}{T_{\text{critical}}} = \frac{1}{30} \] \hspace{1cm} (4.3)
   
   (b) The fundamental limit on the sample period is determined by the iteration bound:
   \[ \text{Sample Period}_{\text{limited}} = T_{\text{bounds}} = \max \left\{ \frac{30}{2} , \frac{25}{1} \right\} = 25 \] \hspace{1cm} (4.4)

   (c) The answer is shown in Figure 4.2

   ![Figure 4.2: Answer of Exercise 2(c)]

3. 
   (a) By inspection, the iteration bound is 7/4 u.t.
   (b) The critical path time of the circuit is 7 u.t. \( M_2 \rightarrow A_1 \rightarrow A_3 \rightarrow M_3 \rightarrow A_4 \).
   (c) The retimed circuit is shown in Figure 4.3.

   ![Figure 4.3: Retimed data flow graph for Exercise 3]
5.

(a) 

\[ T_\infty = 4 \]
\[ T_{critical} = 7 \] (4.7) (4.8)

(b) The minimum achievable clock period obtained with pipelining and retiming is the iteration bound of the DFG, which equals to 4 u.t. in this problem.

![Fig. 4.5 Retimed data flow graph for Exercise 5.](image)

7.

According to \( w_r(u-v) = w(u-v) + r(v) - r(u) \), we get the retimed DFG as in Figure 4.8.

![Fig. 4.8 Retimed DFG for Exercise 7.](image)

8. Refer to spbf.m and spfw.m on the course webpage.

(a) The constrained graph is shown in Figure 4.9. Using Bellman-Ford algorithm, one solution is given as \( r_1 = r_2 = 0, r_3 = -1, r_4 = -2, r_5 = 0 \).

(b) Using Floyd-Warshall algorithm, the matrix \( R(6) \) is as

\[
\text{ans} = \\
\begin{array}{cccccc}
\text{Inf} & \text{Inf} & 1 & 0 & 2 & \text{Inf} \\
1 & \text{Inf} & 1 & 0 & 3 & \text{Inf} \\
\text{Inf} & \text{Inf} & \text{Inf} & -1 & \text{Inf} & \text{Inf} \\
\text{Inf} & \text{Inf} & \text{Inf} & \text{Inf} & \text{Inf} & \text{Inf} \\
\text{Inf} & \text{Inf} & -1 & -2 & \text{Inf} & \text{Inf} \\
0 & 0 & -1 & -2 & 0 & 0 \\
\end{array}
\]

We get the same answer as using Bellman algorithm: \( r_1 = r_2 = 0, r_3 = -1, r_4 = -2, r_5 = 0 \).
10. The method to reduce the critical path by pipeline and retiming is shown in Figure 4.11.

![Diagram](image)

Fig. 4.11 Retiming/pipeline of Exercise 10

11.

(a) Use cutset retiming, we have retimed DFG as in figure 4.12.

![Diagram](image)

Fig. 4.12 Retiming of Exercise 11(a)
(b) First use 2-D slow down, and then apply cutset retiming. The hardware utilization efficiency of this system is 50%.

Fig. 4.13  Slow down and retiming of Exercise 11(b)