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

1. (10 points) Consider the following Fortran-like program:

```fortran
C This is an implementation of back-substitution algorithm
DO 10 I=N,1,-1
  S(I)=B(I)
  IF I<N THEN
    DO 20 J = N,I+1,-1
      S(I)=S(I)-U(I,J)*X(J)
    20
    X(I)=S(I)/U(I,I)
10
```

(a) (4 points) Convert this program into single assignment format with localized data communication.

(b) (3 points) Plot the iteration DG for N =5.

(c) (3 points) Project this DG onto a linear systolic array by choosing a projection vector \( \mathbf{d} \) and a linear schedule vector \( \mathbf{s} \). Perform node mapping, arc mapping and I/O mapping.

2. (10 points) Consider the iteration DG given below:

For each pair of scheduling vector \( \mathbf{s} \) and projection (assignment) vector \( \mathbf{d} \), (i) determine if they are permissible, (ii) if they are permissible, compute the processor space \( \mathbf{P} \), and perform the node mapping, arc mapping, and I/O mapping.

(a) (3 points) \( \mathbf{s} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} ; \quad \mathbf{d} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \)

(b) (2 points) \( \mathbf{s} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} ; \quad \mathbf{d} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \)
3. (10 points) Given that the dependence vectors of a shift-invariant DG are:

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

(a) (3 points) Determine which of the following schedule vectors are permissible linear schedules.
(i) \( s = [1 \ 0]^T \), (ii) \( s = [1 \ -1]^T \), (iii) \( s = [1 \ 1]^T \).

(b) (5 points) Suppose that the DG consists of index points \( \{(i, j); 0 \leq i \leq 5, 0 \leq j \leq 5, i + 2j \leq 12\} \). Find a permissible schedule (needs not be the same as any of the schedule vector in part (a)) that minimizes the total computing time.

(c) (2 points) Find a permissible schedule \( s \) and a project vector \( d \) that minimizes the pipelining period \( \alpha = s^T d \).

4. (10 points) Consider the correlation of two 1-dimensional sequences \( \{x(n)\} \) and \( \{y(n)\} \) with \( 0 \leq n \leq N-1 \).

\[
r(n) = \sum_{k=0}^{N-n-1} x(k)y(k+n)
\]

(a) (2 points, CC) Write a C program (or use any high level language) source code that compute \( \{r(n)\} \) for \( N = 5 \), and \( n = 0, 1, 2 \).

(b) (3 points) Convert this program into single assignment format.

(c) (3 points) Convert this program so that it has local data communications.

(d) (2 points, CC) Plot the corresponding iterative DG.

5. (20 points) (Binary number multiplication) Consider two unsigned, \( n \)-bit binary numbers \( A = A_{n-1}A_{n-2} \ldots A_1A_0 \), and \( B = B_{n-1}B_{n-2} \ldots B_1B_0 \), where \( A_i, B_i \in \{0, 1\} \), \( 0 \leq i \leq n-1 \). Denote \( C = A \times B = C_{2n-2}C_{2n-3} \ldots C_1C_0 \) to be an \( 2n-1 \) bit binary number representing the product of \( A \) and \( B \).

(a) (5 points) Derive an mathematical expression of \( C_k \) in terms of \( A_i \)s and \( B_i \)s.

(b) (3 points) Write a C or Matlab® program to compute \( C_k \) for \( 0 \leq k \leq 2n-2 \). Make sure this program has single-assignment format. Use variable indices as \( A(I), B(I), C(K), \) etc.

(c) (3 points) Localize the program in part (b) by converting all variables that need to be broadcast (used by more than one iterations) during the execution into transmittal variables that will be passed along from an iteration to the next.

(d) (3 points, CC) Plot the logic schematic diagram of the architecture to implement the operations of the loop body as specified in part (c). Note that \( A(I), B(I) \) are both binary variables.

(e) (3 points) For \( n = 4 \), plot the DG of the localized program in part (c), and identify the critical path of this DG. A hardware implementation of this DG is called an array multiplier.

(f) (3 points) Use cut-set retiming to convert the DG directly into a 2D systolic array that implements a fully pipelined version of the array multiplier.


7. (10 points) Saturation arithmetic
Saturation arithmetic is a convenient, nonlinear method of handling overflow of fixed-point computation in DSP applications. Based on the types of two operands and one results, there are 8 possible combinations. In practice, the three commonly used options are \( sss \), \( uuu \), and \( uus \) where \( u \) stands for unsigned, and \( s \) stands for signed. In Intel MMX and SSE-2 instruction set, only \( uuu \), and \( sss \) are implemented. The former is called unsigned saturation arithmetic, and the latter is called signed saturation arithmetic. Suppose now there are two packed operands each consists of four 16-bit words as shown below

\[
\begin{array}{cccc}
R_a: & 58 & 14 & 12 \ 77 \\
R_b: & 22 & 192 & 118 \ 36 \\
\end{array}
\]

The content is given in decimal format for convenience. Refer to MMX instruction set,

(a) (5 points) Write a MMX assembly code segment to compute the maximum of each pair of integers at the corresponding position. Assuming the result is stored in register \( R_c \). The final content of \( R_c \) should be

\[
\begin{array}{cccc}
R_c: & 58 & 192 & 118 \ 77 \\
\end{array}
\]

You should prove that your program is correct. (Hint: Use unsigned saturation arithmetic)

(b) (5 points) Still given \( R_a \) and \( R_b \) with their original content shown in part (a) of this problem. Now, our goal is to perform absolute difference between each pair of the words in \( R_a \) and \( R_b \). Using registers \( R_c \), \( R_d \), \ldots to store intermediate results. The final result should be stored in \( R_d \). The absolute difference between \( a \) and \( b \) is \( |a - b| \).

8. (10 points)
Consider the RIA implementation of the FIR filter

\[
\begin{align*}
y_1(n,-1) &= 0, \quad n = 0, 1, 2, \ldots \\
h_1(0,k) &= h(k), \quad k = 0, \ldots, N-1 \\
n &= 0, 1, 2, \ldots, k = 0, \ldots, N-1 \\
y_1(n,k) &= y_1(n,k-1) + h_1(n,k) \times x_1(n,k) \\
h_1(n,k) &= h_1(n-1,k) \\
x_1(n,k) &= x_1(n-1,k-1) \\
y(n) &= y_1(n,N-1), \quad n = 0, 1, 2, \ldots
\end{align*}
\]

And the corresponding DG:
(a) (2 points) Suppose the projection vector \( \mathbf{d} = [1 \ 0]^T \). Choose a scheduling vector such that the resulting DFG implementation is the same as the data broadcasting FIR structure as discussed in chapter 3.

(b) (2 points) Plot the projected DFG

c) (4 points) Find a uni-modular transform matrix \( \mathbf{M} \) to transform the original DG into a new DG such that the same data broadcasting DFG can be obtained if the projection vector and scheduling vector are chosen as \( \mathbf{d} = \mathbf{s} = [1 \ 0]^T \). Give the (i) \( \mathbf{M} \) matrix, and (ii) plot the transformed DG.

d) (2 points) Write down the corresponding RIA formulation of the transformed DG.

9. (10 points)
Assume that four 64-bit wide registers R1, R2, R3, and R4. Each of them stores a packed 4 16-bit words as shown below:

<table>
<thead>
<tr>
<th></th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>B3</td>
<td>B2</td>
<td>B1</td>
<td>B0</td>
</tr>
<tr>
<td>R3</td>
<td>C3</td>
<td>C2</td>
<td>C1</td>
<td>C0</td>
</tr>
<tr>
<td>R4</td>
<td>D3</td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
</tr>
</tbody>
</table>

(a) (5 points) Consider the execution of the following MMX instructions. Fill out the content of the destination registers AFTER the execution of an instruction shown to the left.

1. MOVQ R5, R1
2. PUNPCKHWD R5, R3
3. MOVQ R6, R2
4. PUNPCKHWD R6, R4
5. MOVQ R7, R5
6. PUNPCKHWD R7, R6
7. MOVQ R8, R5
8. PUNPCKLWD R8, R6
9. MOVQ R9, R1
10. PUNPCKLWD R9, R3
11. MOVQ R10, R2
12. PUNPCKLWD R10, R4
13. MOVQ R11, R9
14. PUNPCKHWD, R11, R10
15. MOVQ R12, R9
16. PUNPCKLWD, R12, R10

Page 4 of 5
(b) (2 points) If the data stored in R1, R2, R3, R4 represent a $4 \times 4$ matrix

\[
M = \begin{bmatrix}
A3 & A2 & A1 & A0 \\
B3 & B2 & B1 & B0 \\
C3 & C2 & C1 & C0 \\
D3 & D2 & D1 & D0
\end{bmatrix}
\]

Discuss what above MMX program have achieved?

(c) (3 points) In above program, total 12 registers are used. Optimize the program by minimizing the total number of different registers that are needed. You must discuss your approach to receive credit.