Part 1  Merging two 4-element arrays P[I] and Q[J], presorted in ascending order into one 8-element array R[K] in ascending order (Note: The elements are 8-bits in size):

Given below is a Moore state diagram in completed form to perform the merge operation.

State names: CMP = Compare, SP = Store P[I] in R[K], SQ = Store Q[J] in R[K], RP = Remaining elements of P may be copied to R without any further comparison as all Q[J] are consumed, RQ = Remaining Q[J]. Assume that the three counters, I, J, and K, are 4-bit counters here.

Note: If (P[I]==Q[J], we need to choose to store Q[J], so that all our results are identical.

Mr. _______________ (Trojan/Bruin) says that, in any clock, I and J cannot both be equal to 3 (i.e. (I=3)&(J=3) can never be true). Mr. _______________ (Trojan/Bruin) does not agree and asked him to consider the data P[I] = 4, 5, 6, B and Q[J] = 7, 8, 9, A;

For the above data, complete the table below by recording the sequence of states and the sequence of the counter values {I,J} that the system goes through until it reaches the DONE state.

<table>
<thead>
<tr>
<th>CLOCK#</th>
<th>#5</th>
<th>#6</th>
<th>#7</th>
<th>#8</th>
<th>#9</th>
<th>#10</th>
<th>#11</th>
<th>#12</th>
<th>#13</th>
<th>#14</th>
<th>#15</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE</td>
<td>CMP</td>
<td>SP</td>
<td>CMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{I,J=}</td>
<td>0,0</td>
<td>0,0</td>
<td>1,0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P[I],Q[J]</td>
<td>4,7</td>
<td>4,7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The total number of clocks spent in the five states, CMP, SP, SQ, RP, and RQ ______ (I / II) (I) is a constant (II) is data dependent.

The minimum number of clocks spent in CMP state is ______ and the maximum number of clocks spent in CMP state is ______.

Minimum number of clock spent in RQ or RP is ______. The maximum number of clocks spent in RQ or RP is ________.
Part 2

Now let us convert the above Moore state diagram to a Mealy state diagram by merging the three states CMP, SP, and SQ into one state called CMST (compare and store state) as shown below.

Mr. ______________ (Trojan/Bruin) says that we have to necessarily go through either RQ or RP and cannot go from CMST to DONE state directly, because the last element going into R[K] does not require any comparison.

Complete the state diagram. Since there may not be enough space to write C1 and C2 conditions below in the diagram, please write them below.

C1 = ____________________ ; C2 = ____________________ ;

Note: If (P[I]==Q[J], we need to choose to store Q[J], so that all our results are identical.

Mr. ______________ (Trojan/Bruin) says that, in any clock, I and J cannot both be equal to 3 (i.e. (I=3)&(J=3) can never be true). Mr. ______________ (Trojan/Bruin) does not agree and asked him to consider the data P[I] = 4, 5, 6, B and Q[J] = 7, 8, 9, A;

For the above data, complete the table below by recording the sequence of states and the sequence of the counter values {I,J} the system goes through until it reaches the DONE state.

<table>
<thead>
<tr>
<th>CLOCK#</th>
<th>#5</th>
<th>#6</th>
<th>#7</th>
<th>#8</th>
<th>#9</th>
<th>#10</th>
<th>#11</th>
<th>#12</th>
<th>#13</th>
<th>#14</th>
<th>#15</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE</td>
<td>CMST</td>
<td>CMST</td>
<td>CMST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{I,J}</td>
<td>0, 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P[I],Q[J]</td>
<td>4, 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The total number of clocks spent in the three states, CMST, RP, and RQ _______ (I / II)
(I) is a constant    (II) is data dependent.
The minimum number of clocks spent in CMST state is _____ and the maximum number of clocks spent in CMST state is _____.
Minimum number of clock spent in either RQ or RP is _____. The maximum number of clocks spent in RQ or RP is ______.
Part 3 Let us now create another Mealy design! Now merge the three states (CMST, RP, and RQ) of the previous Mealy design (which is same as merging the 5 states, CMP, SP, SQ, RP, and RQ, of the original Moore design) into one single state called MERGE below. Hint: In the RQ state of the previous Mealy machine I would be 4 and, in the RP state, the J is at 4. So you can say that, in this single state, MERGE, you would check to see if (I==4) first. If it is, then you would do what is done in the RQ state. Else if (J==4), you would do what is done in the RP state. If it is neither, then you would do what is done in the CMST state. The exit condition from the MERGE state is also simple! It can be based on the K index reaching 7! Complete the diagram. Miss ___________ (Trojan / Bruin) says that she can save clocks by looking at (I==3) instead of (I==4), and (J==3) instead of (J==4).

Note: If (P[I]==Q[J], we need to choose to store Q[J], so that all our results are identical.

```
IN
I<=0;
J<=0;
K<=0;
```

```
MERGE
if [I==4]

else if [J == 4]

else // i.e. neither I is at 4 nor J is at 4

```

```
DONE
ACK
ACK
```
Part 4  In a variation of this problem, we need to merge three arrays A[I], B[J], and C[K] into M[L].

The A, B, and C arrays have each 4 elements sorted already in ascending order and the result array M at the end shall have all the 12 elements [0:11] in ascending order. All four counters, I, J, K, and L are 4-bit counters. **We have only one comparison unit.** So we compare A with B in CAB state (Compare A with B state) first, and then move to CACS1 state (Compare A with C and Store 1 state) or CBCS1 state (Compare B with C and Store 1 state). If we store C[K] in these states, we remain there and process the next C[K]. Otherwise we return to CAB for comparing A with B.

Eventually one of the three arrays will be over and then, to process the remaining two arrays, we have CABS2, CBCS2, and CACS2 for comparing two items in a clock. Notice that "1" and the "2" in the numbering of the 5 state names, indicating the first stage ("1") of merging in states CACS1 and CBCS1 and the second stage ("2") of merging in CABS2, CBCS2, CACS2 states.

In the third phase, we have RA, RB, and RC states for processing the remaining elements of the last array.

The RTL statements in the state circles are complete. Add the missing state transition arrows and the state transition conditions to complete the state diagram. Since the state diagram is symmetric across the center line (i.e. the lower part is very similar to the upper part), you need to complete state transition conditions for transitions on the center line and above only, and leave the ones below the center line. Feel free to use a convention similar to the one used on the first Mealy machine, wherever there isn’t enough space to write a long boolean condition. For example, you can use C1, C2, C1’& C2’ on the state diagram and write the conditions C1 and C2 separately.

Analysis of the data-dependent behavior: Note: This can take substantial time :(
Consider different data to help arriving at the various maxima and minima of number of clocks spent in the three phases of merging. Examples
(a) none of the A, B, C arrays gets depleted quickly: A[I]=4,5,6,C; B[J]=8,9,A,B; C[K]=0,1,2,D
(b) all of C[K] get consumed first requiring minimal time in phase 1: A[I]=4,5,6,C; B[J]=8,9,A,B; C[K]=0,1,2,3

Minimum and maximum clocks that may be spent in the CAB state:
Min _____, Max ______;

Minimum and maximum clocks that may be spent in the CACS1 and CBCS1 states combined:
Min _____, Max ______;

Minimum and maximum clocks that may be spent in the CBCS2, CABS2, and CACS2 states combined:
Min _____, Max ______;

Minimum and maximum clocks that may be spent in the RA, RB, and RC states combined:
Min _____, Max ______;

Overall in all significant states put together (i.e. in the middle 9 states of the 11-state state diagram) (i.e. leaving the INI and DONE states), the minimum and maximum clocks that may be spent:
Min _____, Max ______;

(25 pts)

(25 pts) (Like/Unlike) in Q#1.3, to combine all the 9 significant states into one single Mealy state, we ________________ (need/do not need) additional "Flag" Flip-Flops to help us to know what we should be doing in that single state [ additional Flag FFs besides (I==4), (J==4), and/or (K==4) conditions]. Explain: ________________________________________________
State sequence for all 12 identical elements

<table>
<thead>
<tr>
<th>Task</th>
<th>Condition</th>
<th>Action 1</th>
<th>Action 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CACS1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CABS2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CBCS1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CBCS2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If \( A[I] = B[J] = C[K] \), we need to choose to store \( C[K] \) first, and if \( A[I] = B[J] < C[K] \), we need to choose to store \( B[J] \) first, so that all our results are identical. If all 12 elements are identical, we will be consuming all C’s followed by all B’s, and finally all A’s!
What you need to do

1. Go through the lab and watch the posted power-point .pptx file animating the merging operation.

2. Complete the previous pages which serves as your paper submission for this lab.

3. Download the two .zip files for coding part 2 (P2) and part 4 (P4):
   
   merge_sort_P2.zip
   merge_sort_P4.zip

4. Extract files into separate directories

   example:
   C:\Modelsim_projects\merge_sort_P2,
   C:\Modelsim_projects\merge_sort_P4

   Files provided for P2 in merge_sort_P2.zip:
   merge_sort_P2.v (incomplete),
   merge_sort_P2_tb.v (complete),
   merge_sort_P2.do (complete),
   merge_sort_P2_wave.do (complete),
   merge_sort_P2.do (complete),
   output_results_P2.txt (empty file to be generate by your simulation)

   Files provided for P4 in merge_sort_P4.zip:
   merge_sort_P4.v (incomplete),
   merge_sort_P4_tb.v (complete),
   merge_sort_P4.do (complete),
   merge_sort_P4_wave.do (complete),
   merge_sort_P4.do (complete),
   output_results_P4.txt (empty file to be generate by your simulation)

5. Online submission commands

   submit -user ee457lab -tag puvvada_merge_sort_P2 merge_sort_P2.v output_results_P2.txt names.txt
   submit -user ee457lab -tag puvvada_merge_sort_P4 merge_sort_P4.v output_results_P4.txt names.txt

   The two files, StateSequence_P2.txt and StateSequence_P4.txt, need not be submitted.
   These are useful for debugging though.

Celebrate your success in completing this challenging lab successfully!