8.2  State Reduction
Given the state diagram in Figure Ex. 8.2, obtain an equivalent reduced-state diagram containing a minimum number of states. You may use row-matching or implication charts. Put your final answer in the form of a state diagram rather than a state table. Make it clear which states have been combined.

The following are equivalent:
A = B
E = F = G
8.3 State Reduction
Create a Moore state diagram for the 4-bit string recognizer of Figure 8.1. Does it have more states than the Mealy version? Use the implication chart method to reduce the number of states. Do you end with more, less, or the same number of states? Why?

Start with any Moore machine which has the same function on Fig 8.1 and reduce the states.

8.8 State Assignment
Given the state diagram in Figure Ex. 8.8, select a good state assignment, justifying your answer in terms of the state assignment guidelines.

In Decreasing Priority:

Guideline 1: \{A, E, C\} \{B, C, E, D\}
Guideline 2: \{A, B\}, \{C, D\}, \{A, D\}, \{D, E\}
Guideline 3: \{A, B, C, D, E\}, \{A, B, C\}, \{D, E\}

Use these guidelines; answer can vary
8.11 State Assignment

Given the next-state function of the finite state machine shown in Figure Ex. 8.11, use the implication chart method to find the most reduced state diagram.

<table>
<thead>
<tr>
<th>$Q_0$</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>$P_0$</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Cur State</td>
<td>Next State</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_0$</td>
<td>$S_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_1$</td>
<td>$S_0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>$S_9$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td>$S_{10}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_4$</td>
<td>$S_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_5$</td>
<td>$S_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_6$</td>
<td>$S_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_7$</td>
<td>$S_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_8$</td>
<td>$S_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_9$</td>
<td>$S_9$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{10}$</td>
<td>$S_{10}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{11}$</td>
<td>$S_{10}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{12}$</td>
<td>$S_6$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{13}$</td>
<td>$S_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{14}$</td>
<td>$S_{14}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{15}$</td>
<td>$S_{14}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Combine: $S_{14}$, $S_{15} => S_{14}$

$S_4$, $S_{13} => S_4$

$S_3$, $S_{10}$, $S_{11} => S_{10}$

$S_7$, $S_8 => S_7$

$\{S_0, S_1\}, \{S_0, S_9\}, \{S_6, S_9\}$ are not possible because they need to be distinct outputs from other states.

At the end we have:

If multiple numbers are in the same bubble, the circled number is the state assignment.
Your friend from Stanford has implemented a 1us counter using a 27MHz clock by first dividing down the clock in a Verilog module called 1usClockGen(clockIn27MHz, clockOut1MHz), and then using the 1MHz clock from that module as the clock input for a normal counter. He finds that sometimes when he synthesizes the circuit to an FPGA, he gets very weird results. The counter seems to work fine by itself, but when he hooks the counter outputs up to another state machine FSM2 that has a 27MHz clock input as well, the state machine doesn't work. Explain his problem in detail (show an example of FSM2, and waveforms for the various clocks and outputs), and how you would solve it. (Hint: enable.)

Suppose you want to count 4us after reset,

Due to delay after 1us clock, output from 2-bit counter is asynchronous with 27 MHz CLK. If transition happens at CLK edge, the behavior is unpredictable.

For example:

The fix would be: