Question 1. Implement this ASM using the mux method.

The mux method uses multiplexers to develop next-state functions and therefore requires control-state encoding. This solution encodes the states as \( W = 00, X = 01, Y = 10, \) and \( Z = 11. \) The multiplexer inputs are re-labeled symbolically to reflect the encodings. Since we need to develop encoded next-state values, it may be helpful to use a truth-table or K-MAPs (but not both) to develop the terms. With practice, the terms can be read from the out-going paths of the ASM.

Output terms can be developed the same way.

Comments: You may have noticed that the conditional output \( OUT2 \) in the upper portion of the chart is redundant because \( OUT2 \) is always asserted in state \( W. \)

Also, looking at the next-state mux inputs it is tempting to try optimizing, based on the optimization that the inputs for states \( Y \) and \( Z \) are always 0. Although such an optimization may be possible it is not likely to be worth the thought, except in extreme circumstances.
Question 2. Implement this ASM using the one-hot method.

The one-hot method allocates one flip-flop for each state in the ASM. In the solution below, a new name is given to each state even though, in this instance, the one-to-one correspondence between states and outputs M, P, S, W, and R makes state-naming unnecessary, but in general one would do it. Next-state values for A’ through E’ are developed by writing out an SOP expression in which each clause describes a path leading to that state as a conjunction of the conditions which determine that path.
Question 3. This ASM has been labelled for an encoded-state implementation with 2-bit encoding $BA$.

(a) Complete the implementation.

(b) Notice that one of the decision signals, $Y^*$, is asynchronous. Routinely, we would synchronize this input to avoid any possibility of a transition race. However, it happens in this case that a transition race will not occur. Explain why this is so.

The intent of the problem is to synthesize an implementation by developing boolean terms for the next-state values in the standard way. The pseudo-K-map represent in the control-state transition function, is broken down in to K-maps for each bit of the encoded state. There is no need to write down a truth-table first. Output equations are synthesized by analyzing the paths in the ASM.

The potential transition race caused by $Y^*$ is benign in this case because there is only one bit of difference between the encodings for states $L$ and $J$. See the textbook’s discussion on unit-distance encodings.