1. (ASM charts) Problem 8-1, text book, p. 393
   Answer:

   ![ASM chart for Problem 8-1]

   Inputs: \(X_1, X_2\)
   Outputs: \(Z_1, Z_2\)

2. (ASM charts) Problem 8-3, text book, p. 393
   Answer:

   ![ASM chart for Problem 8-3]

<table>
<thead>
<tr>
<th>PS</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>NS</th>
<th>Z</th>
<th>PS</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>NS</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>ST1</td>
<td>0</td>
<td>ST1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>ST3</td>
<td>0</td>
</tr>
<tr>
<td>ST1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>ST1</td>
<td>0</td>
<td>ST3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>ST1</td>
<td>0</td>
</tr>
<tr>
<td>ST1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>ST1</td>
<td>0</td>
<td>ST1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>ST1</td>
<td>1</td>
</tr>
<tr>
<td>ST1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>ST1</td>
<td>0</td>
<td>ST3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>ST1</td>
<td>1</td>
</tr>
<tr>
<td>ST1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>ST1</td>
<td>0</td>
<td>ST1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>ST1</td>
<td>1</td>
</tr>
<tr>
<td>ST1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>ST1</td>
<td>0</td>
<td>ST1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>ST1</td>
<td>1</td>
</tr>
<tr>
<td>ST1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>ST1</td>
<td>0</td>
<td>ST1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>ST1</td>
<td>1</td>
</tr>
<tr>
<td>ST1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>ST1</td>
<td>0</td>
<td>ST1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>ST1</td>
<td>1</td>
</tr>
</tbody>
</table>
3. (ASM design) Problem 8-5, text book, p. 394

Answer:

![Flowchart for Problem 8-5](image)

4. (ASM design) Problem 8-7, text book, p. 395

Answer:

![Flowchart for Problem 8-7](image)

5. (ASM implementation) Problem 8-8, text book, p. 396

Answer:

\[
ST1(t + 1) = ST1 \overline{A} + ST2B \overline{C} + ST3, \quad ST2(t + 1) = ST1A, \quad ST3(t + 1) = ST2(\overline{B} + C), \quad Z = ST2B + ST3
\]

For the D flip-flops, D_{ST1} = STi(t + 1) and STi = Q_{STi}. Reset initializes the flip-flops: ST1 = 1, ST2 = ST3 = 0.

6. (ASM implementation) Problem 8-9, text book, p. 396

Answer:

![Truth Table and Circuit Diagram for Problem 8-9](image)
7. (Binary Multiplier) Problem 8-12, text book, p. 396
   Answer:

<table>
<thead>
<tr>
<th>State</th>
<th>P</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLE</td>
<td>X</td>
<td>1010</td>
<td>X</td>
<td>X</td>
<td>1011</td>
</tr>
<tr>
<td>MCL0</td>
<td>011</td>
<td>1010</td>
<td>0</td>
<td>1010</td>
<td>1011</td>
</tr>
<tr>
<td>MCL1</td>
<td>011</td>
<td>1010</td>
<td>0</td>
<td>0101</td>
<td>0101</td>
</tr>
<tr>
<td>MCL0</td>
<td>010</td>
<td>1010</td>
<td>0</td>
<td>1111</td>
<td>0101</td>
</tr>
<tr>
<td>MCL1</td>
<td>010</td>
<td>1010</td>
<td>0</td>
<td>0111</td>
<td>1010</td>
</tr>
<tr>
<td>MCL0</td>
<td>001</td>
<td>1010</td>
<td>0</td>
<td>0111</td>
<td>1010</td>
</tr>
<tr>
<td>MCL1</td>
<td>001</td>
<td>1010</td>
<td>0</td>
<td>0011</td>
<td>1101</td>
</tr>
<tr>
<td>MCL0</td>
<td>000</td>
<td>1010</td>
<td>0</td>
<td>1101</td>
<td>1101</td>
</tr>
<tr>
<td>MCL1</td>
<td>000</td>
<td>1010</td>
<td>0</td>
<td>0110</td>
<td>1110</td>
</tr>
<tr>
<td>IDLE</td>
<td>111</td>
<td>1010</td>
<td>0</td>
<td>0110</td>
<td>1110</td>
</tr>
</tbody>
</table>

8. (Design) Problem 8-16, text book, p. 396
   Answer:

   R is a synchronous reset that overrides any simultaneous synchronous transfer.
9. (Verilog ASM) Problem 8-21, text book, p. 397
Answer:

module sum_321 (X, CLK, RESET, Z);
input X;
input CLK;
input RESET;
output Z;

reg [1:0] state, next_state;
parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010,
    S3 = 3'b011, S4 = 3'b100;

reg Z;

// State register
always @(posedge CLK or posedge RESET)
begin
    if (RESET == 1)
        state <= S0;
    else
        state <= next_state;
end

// Next state function
always @(X or state)
begin
    case (state)
        S0: if (X) next_state <= S3;
        else next_state <= S1;
        S1: if (X) next_state <= S2;
        else next_state <= S1;
        S2: if (X) next_state <= S3;
        else next_state <= S4;
        S3: if (X) next_state <= S3;
        else next_state <= S4;
        S4: if (X) next_state <= S2;
        else next_state <= S1;
        default: next_state <= S0;
    endcase
end

// Output function
always @(state)
begin
    case (state)
        S2: Z <= 1'b1;
    endcase
endmodule

---

10. (Static Hazards)

A) Given a function F, find all static hazard locations and mark with a “diamond.”

Answer:
B) Draw the logic gate circuit including the extra gates needed to eliminate the logic hazard in part a.

Answer:  
The logic circuit needs to implement:  
\[ F = BC + \overline{A}CD + ABCD + A\overline{BC} + (ABD + BC + \overline{A}BD) \]

C) You plan to implement the function represented in the K-MAP in Part A using sum of products. How many of the static hazards are static 0-hazards? How many are static 1-hazards?

Answer:  
Sum of products implementations do not have static 0-hazards, so all 3 hazards are static 1.

D) For your circuit designed in Part B, are there any dynamic hazards?

Answer:  
No, by adding the logic to deal with the static hazards, the dynamic hazards are also fixed.

11. (Function Hazards) Indicate by double arrows the location of all function hazards in function F.

Answer:  
![Diagram showing function hazards]

Two-input Change Function Hazards Shown