DO NOT TURN THIS PAGE UNTIL YOU ARE TOLD TO DO SO

<table>
<thead>
<tr>
<th>Problem</th>
<th>Weight</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

INSTRUCTIONS

1. You have 2 hours to complete this exam.

2. This is a closed book exam. You may use one 8.5” × 11” note sheet.

3. Calculators are allowed.

4. Solve each part of the problem in the space following the question. If you need more space, continue your solution on the reverse side labeling the page with the question number; for example, Problem 1.2 Continued. NO credit will be given to solutions that do not meet this requirement.

5. **DO NOT REMOVE ANY PAGES FROM THIS EXAM.** Loose papers will not be accepted and a grade of ZERO will be assigned.

6. The quality of your analysis and evaluation is as important as your answers. Your reasoning must be precise and clear; your complete English sentences should convey what you are doing. **To receive credit, you must show your work.**
Problem 1: (34 Points)

1. (24 points) Turnstiles, like the one shown in Figure 1, control access and enforce one-way traffic of people in settings such as amusement parks, stadiums, mass transit stations, and ski resorts. Initially, the arms are locked, preventing customers from passing through. Depositing a coin or token in a slot on the turnstile unlocks the arms so they can rotate by one-third of a complete turn, allowing a single customer to push through. After the customer passes through, the arms lock again until another coin is inserted.

In this problem you will represent the operation of the turnstile using a finite state machine. Two states, Locked (S0) and Unlocked (S1), describe the operation of the turnstile. Two digital inputs affect the state of the Turnstile. Placing a coin in the turnstile sets the coin input signal, C, to the logic-high state (C = 1). Pushing the arm forward sets the push input signal, P, to the logic-high state (P = 1). The system has a single output signal, L. When L is set to a logic-high state, L = 1, the turnstile gate is locked, and when L is set to a logic-low state, L = 0, the gate is unlocked.

In the locked state, pushing on the arm has no effect. Regardless of how many times the user pushes the arm, the turnstile remains locked. Putting a coin in shifts the state from Locked to Unlocked. In the Unlocked state, putting additional coins in has no effect, the system remains in the unlocked state. However, when a customer pushes through the arms, the state shifts back to Locked.

Figure 1: Waist-high turnstile
(a) (12 points) Represent the operation of the turnstile using a Moore finite state machine by sketching the state diagram. To receive full credit, label the states as $S_0$ and $S_1$, show the state of the output signal $L$, and the value of the inputs $C$ and $P$.

(b) (12 points) Construct the state table.
2. (10 points) Figure 2 shows the implementation of two different finite state machines. For each finite state machine, specify the maximum number of possible states and state whether the implementation is that of a Moore or Mealy machine. Justify your answers using one or two short sentences.

(a) (5 points) Machine in Figure 2(A):

- Maximum number of possible states: 
- Machine type (Moore or Mealy): 

(b) (5 points) Machine in Figure 2(B):

- Maximum number of possible states: 
- Machine type (Moore or Mealy): 

Figure 2: Two separate finite state machines
Problem 2: (33 Points)
Figure 3 shows the state table for a Moore finite state machine with input $x$ and output $y$.

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_A$ $Q_B$</td>
<td>$Q_A^<em>$ $Q_B^</em>$</td>
<td>$y$</td>
</tr>
<tr>
<td>$S_0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0</td>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>$S_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1</td>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>$S_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td>0 0</td>
<td>1</td>
</tr>
</tbody>
</table>

1. (5 points) Sketch the state diagram. To receive full credit, label the states, show the state of the output signal $y$, and the value of the input signal $x$. 

Figure 3: State table for a Moore finite state machine
2. (14 points) Suppose we implement the finite state machine with two D-type flip-flops. Using the state table in 3, determine expressions for the flip-flop inputs and the finite state machine output using a three-variable Karnaugh map. For each flip-flop input and the finite state machine output, draw a separate three-variable Karnaugh map using Figure 4 as a guide.
Figure 4: Three variable Karnaugh map
3. (14 points) Realize the finite state machine using two D-type flip-flops and two-input NAND gates. Neatly draw a circuit diagram of your implementation.
Problem 3: (33 Points)

As an example, Figure 5 shows the data-flow code for implementing a finite-state machine using the ATMEAL 750CL programmable logic device. Using Figure 5 as a guide, write dataflow code for implementing the finite state machine in Figure 6. Assign the clock to pin 1, the input to pin 2 and the output to pin 23. Set the reset (clear) and set input of all D-type flip-flops to zero. With reference to Figure 6, label the D-type flip-flop outputs, from left to right, as QA through QD. **Do not** include the header information in your code, but you must include comment lines to improve the readability of your code.
Figure 5: Example dataflow code

```c
/* **************** INPUT PINS ******************/
PIN 1 = CLK;  /* external clock input */
PIN 2 = INP;  /* input bit stream */

/* **************** OUTPUT PINS ******************/
PIN 23 = Q0;  /* output = 1 when odd parity */

Q0.SP = 'b'0;
Q0.AR = 'b'0;
Q0.OE = 'b'1;
Q0.CK = CLK;
Q0.D = INP $ Q0;
```

Figure 6: Finite State machine
DO NOT TURN THIS PAGE UNTIL YOU ARE TOLD TO DO SO

<table>
<thead>
<tr>
<th>Problem</th>
<th>Weight</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

INSTRUCTIONS

1. You have 2 hours to complete this exam.

2. This is a closed book exam. You may use one 8.5” × 11” note sheet.

3. Calculators are allowed.

4. Solve each part of the problem in the space following the question. If you need more space, continue your solution on the reverse side labeling the page with the question number; for example, Problem 1.2 Continued. NO credit will be given to solutions that do not meet this requirement.

5. **DO NOT REMOVE ANY PAGES FROM THIS EXAM.** Loose papers will not be accepted and a grade of ZERO will be assigned.

6. The quality of your analysis and evaluation is as important as your answers. Your reasoning must be precise and clear; your complete English sentences should convey what you are doing. **To receive credit, you must show your work.**
Problem 1: (34 Points)

1. (24 points) Turnstiles like the one shown in Figure 1, control access and enforce one-way traffic of people in settings such as amusement parks, stadiums, mass transit stations, and ski resorts. Initially, the arms are locked, preventing customers from passing through. Depositing a coin or token in a slot on the turnstile unlocks the arms so they can rotate by one-third of a complete turn, allowing a single customer to push through. After the customer passes through, the arms lock again until another coin is inserted.

In this problem you will represent the operation of the turnstile using a finite state machine. Two states, Locked (S0) and Unlocked (S1), describe the operation of the turnstile. Two digital inputs affect the state of the Turnstile. Placing a coin in the turnstile sets the coin input signal, C, to the logic-high state (C = 1). Pushing the arm forward sets the push input signal, P, to the logic-high state (P = 1). The system has a single output signal, L. When L is set to a logic high state, L = 1, the turnstile gate is locked, and when L is set to a logic-low state, L = 0, the gate is unlocked.

In the locked state, pushing on the arm has no effect. Regardless of how many times the user pushes the arm, the turnstile remains locked. Putting a coin in shifts the state from Locked to Unlocked. In the Unlocked state, putting additional coins in has no effect, the system remains in the unlocked state. However, when a customer pushes through the arms, the state shifts back to Locked.

Figure 1: Waist-high turnstile
(a) (12 points) Represent the operation of the turnstile using a Moore finite state machine by sketching the state diagram. To receive full credit, label the states as S0 and S1, show the state of the output signal L, and the value of the inputs C and P.

\[ p = x, c = 0 \]

\[ c = 1, p = x \]

\[ \text{S0, } L = 1 \]

\[ \text{S1, } L = 0 \]

\[ \text{Locked} \]

\[ \text{P} = 1, \text{C} = x \]

\[ \text{Unlocked} \]

\[ X = \text{Don't Care} \]

(b) (12 points) Construct the state table.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Input</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>0</td>
<td>S0</td>
<td>1</td>
</tr>
<tr>
<td>S0</td>
<td>1</td>
<td>S1</td>
<td>0</td>
</tr>
<tr>
<td>S1</td>
<td>0</td>
<td>S1</td>
<td>0</td>
</tr>
<tr>
<td>S1</td>
<td>0</td>
<td>S0</td>
<td>1</td>
</tr>
<tr>
<td>S1</td>
<td>1</td>
<td>S0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Present state</th>
<th>Input</th>
<th>Next state</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>c</td>
<td>S0</td>
<td>1</td>
</tr>
<tr>
<td>S1</td>
<td>c</td>
<td>S1</td>
<td>0</td>
</tr>
<tr>
<td>S0</td>
<td>p</td>
<td>S1</td>
<td>1</td>
</tr>
<tr>
<td>S1</td>
<td>p</td>
<td>S0</td>
<td>0</td>
</tr>
</tbody>
</table>
2. (10 points) Figure 2 shows the implementation of two different finite state machines. For each finite state machine, specify the maximum number of possible states and state whether the implementation is that of a Moore or Mealy machine. Justify your answers using one or two short sentences.

(a) (5 points) Machine in Figure 2(A):
• Maximum number of possible states: 4
• Machine type (Moore or Mealy): Mealy

Maximum number of possible states = \(2^4\) (number of Flip Flops)

The FSM is a Mealy machine because the output can be changed during the current state by altering input 2.

(b) (5 points) Machine in Figure 2(B):
• Maximum number of possible states: 16
• Machine type (Moore or Mealy): Moore

Maximum number of possible states = \(2^4\) = 16

The FSM is a Moore machine because the output is determined only by the state of the Flip-Flop outputs, and not the present value of the input.
Figure 2: Two separate finite state machines
Problem 2: (33 Points)
Figure 3 shows the state table for a Moore finite state machine with input $x$ and output $y$.

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x = 0$</td>
<td>$x = 1$</td>
</tr>
<tr>
<td>$S_0$</td>
<td>$Q_A$ $Q_B$</td>
<td>$D_A$ $D_B$</td>
</tr>
<tr>
<td>$S_1$</td>
<td>$Q_A$ $Q_B$</td>
<td>$D_A$ $D_B$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>$Q_A$ $Q_B$</td>
<td>$D_A$ $D_B$</td>
</tr>
</tbody>
</table>

Figure 3: State table for a Moore finite state machine

1. (5 points) Sketch the state diagram. To receive full credit, label the states, show the state of the output signal $y$, and the value of the input signal $x$. 

![State Diagram](image_url)
2. (14 points) Suppose we implement the finite state machine with two D-type flip-flops. Using the state table in 3, determine expressions for the flip-flop inputs and the finite state machine output using a three-variable Karnaugh map. For each flip-flop input and the finite state machine output, draw a separate three-variable Karnaugh map using Figure 4 as a guide.

\[ D_A = x \land Q_B \]

\[ D_B = x \]

\[ Y = Q_A \land Q_B \]
Figure 4: Three variable Karnaugh map
3. (14 points) Realize the finite state machine using two D-type flip-flops and two-input NAND gates. Neatly draw a circuit diagram of your implementation.

\[ D_A = \chi \overline{Q_B} = \bar{\chi} \bar{Q_B} \]

\[ D_B = \chi \]

\[ y = Q_A \overline{Q_B} = \bar{Q_A} \bar{Q_B} \]
Problem 3: (33 Points)

As an example, Figure 5 shows the data-flow code for implementing a finite-state machine using the ATMEL 750CL programmable logic device. Using Figure 5 as a guide, write dataflow code for implementing the finite state machine in Figure 6. Assign the clock to pin 1, the input to pin 2 and the output to pin 23. Set the reset (clear) and set input of all D-type flip-flops to zero. With reference to Figure 6, label the D-type flip-flop outputs, from left to right, as QA through QD. Do not include the header information in your code, but you must include comment lines to improve the readability of your code.

```c
/* *** INPUT PINS *****/
PIN 1 = CLK;  // external clock input
PIN 2 = INP;  // input

/* *** OUTPUT PINS *****/
PIN 23 = OUTPUT;  // output of FSM

/* *** SETUP SP, AR, OE, and CK PINS *****/
[QA, QB, QC, QD]. SP = 'b'0;
[QA, QB, QC, QD]. AR = 'b'0;
[QA, QB, QC, QD]. OE = 'b'1;
[QA, QB, QC, QD]. CK = CLK;

/* *** Flip-Flop inputs *****/
QA.D = !(QA & QB) & QC & QD & INP;
QB.D = QA;
QC.D = QB;
QD.D = QC;

/* *** Generate output *****/
OUTPUT = !(QA & QB & QC & QD);
```
/* *************** INPUT PINS **************************/
PIN 1 = CLK; /* external clock input */
PIN 2 = INP; /* input bit stream */

/* *************** OUTPUT PINS **************************/
PIN 23 = Q0; /* output = 1 when odd parity */
Q0.SP = 'b'0;
Q0.BR = 'b'0;
Q0.CE = 'b'1;
Q0.CK = CLK;
Q0.D = INP $ Q0;

Figure 5: Example dataflow code

Figure 6: Finite State machine