Announcements

• Homework 9 up on course webpage, due on Monday, 12/7
  – Final homework assignment

• Please fill out Course Evaluations online.
  – Class time on Monday, 12/7
  – Make sure to bring in laptop, phone, etc.
Agenda

• Last Time:
  – State Table Reduction (7.4)
  – The State Assignment Problem (7.5)

• This Time:
  – Finish The State Assignment Problem (7.5)
  – Completing the Design of Clocked Synchronous Sequential Networks (7.6)
Modeling clocked synchronous sequential network behavior

• Approach for the synthesis of clocked synchronous sequential networks:
  – State table/state diagram is constructed from word specifications.
  – State reduction technique to obtain a state table with minimum number of states.
  – Transition table is formed by coding the states of the state table.
  – Excitation table is constructed based on the flip-flop types to be used.
  – From the excitation table, the excitation and output expressions for the network are determined.
  – Finally, the logic diagram is drawn.
Last time we left off...

Table 7.17 Illustrations of state assignments. (a) State table. (b) Transition table for state assignment in binary order. (c) Transition table for state assignment based on guidelines.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output (z)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input (x)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>*A</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>D</td>
<td>F</td>
<td>G</td>
</tr>
<tr>
<td>E</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>F</td>
<td>D</td>
<td>H</td>
</tr>
<tr>
<td>G</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>H</td>
<td>F</td>
<td>G</td>
</tr>
</tbody>
</table>

Input (x)
\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 \\
\end{array}
\]
Next Step: Constructing Transition Table from State Table

• A binary code representation for the states of the state table is selected.

• This is referred to as the state-assignment problem.

• Different state assignments result in realizations of different costs.
  – We want to find a state assignment that minimizes the cost of the network realization.
State Assignment

• If there are $s$ states to be coded, the minimum number of binary digits $p$ required is the smallest integer greater than or equal to the base-2 logarithm of $s$.

• This guarantees minimal number of flip-flops but not necessarily minimum cost realization.

• Even using the minimum binary digits the state assignment problem is not necessarily simple.
  
  – There are $2^p!/(2^p - s)!$ Ways of assigning a unique binary code of $p$ digits to the $s$ states.
  
  – For a six-row state table in which 3 binary digits are used to code each state, there are 20,160 different state assignments.
Simplest Approach

- Use the first $s$ binary integers as the binary-code representation of the $s$ states.
Guidelines for Obtaining State Assignments

• Define two states as being adjacent if their binary codes differ in exactly one bit.

• Two input combinations are adjacent if they differ in exactly one bit.
Guidelines for Obtaining State Assignments

• Rule I: Two or more present states that have the same next state for a given input combination should be made adjacent.

• Rule II: For any present state and two adjacent input combinations, the two next states should be made adjacent.

• Rule III: Two or more present states that produce the same output symbol, for a given input combination should be made adjacent (only needs to be done for one of the two output symbols).
Rationale for Guidelines

• \( n \) input variables \( p \) state variables.

• Consider \((n + p)\)-variable K-maps for each bit of next state and output.

• Rule I: provide for large subcubes on K-map by causing identical entries to appear in adjacent cells.

• Rule II: Cells in K-map will be the same for \( p - 1 \) of the maps corresponding to bits of the state.

• Rule III: Does to the output maps what Rule I does to the next-state maps.
Example

- State \( B \) is the next state for both present states \( B, G \) when \( x = 0 \). Rule 1: \( B, G \) should be adjacent.
- States \( C, F \) should be coded as adjacent states since their next states are both state \( D \).
- States \( D, H \) should be coded as adjacent states since their next states are both \( F \).

Rule I: \((B, G)(2 \times), (C, F), (D, H)(2 \times), (A, E)\) 
\((2 \times)\) indicates that the recommended adjacency conditions appear twice and should be given higher priority than those that appear only once.
Next consider Rule II:

- Since $x = 0, x = 1$ are adjacent, the next-state pair for each present state should be made adjacent according to Rule II.

Rule II: $(A, B), (B, C)(3 \times), (D, E), (F, G)(2 \times), (D, H)$
Next consider Rule III:
Look at present states that produce output symbol 1 on the same input.

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output (z)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input (x)</td>
<td>Input (x)</td>
</tr>
<tr>
<td>*A</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Rule III: \((D, F), (E, G)\)
State Assignment Map

- K-map for the state variables in which each cell of the map denotes a combination of the binary digits that can be assigned to a state of the sequential network.

<table>
<thead>
<tr>
<th>A</th>
<th></th>
<th></th>
</tr>
</thead>
</table>

I.S.

- (B,G) 2x
- (C,F)
- (D,H) 2x
- (A,E)
- (A,B)
- (B,C) 3x
- (D,E)
- (F,G) 2x
- (D,H)
- (D,F)
- (E,G)
State Assignment Map

- K-map for the state variables in which each cell of the map denotes a combination of the binary digits that can be assigned to a state of the sequential network.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
State Assignment Map

- K-map for the state variables in which each cell of the map denotes a combination of the binary digits that can be assigned to a state of the sequential network.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>G</th>
</tr>
</thead>
</table>

- (B,G) 2x
- (C,F)
- (D,H) 2x
- (A,E)
- (A,B)
- (B,C) 3x
- (D,E)
- (F,G) 2x
- (D,H)
- (D,F)
- (E,G)
State Assignment Map

- K-map for the state variables in which each cell of the map denotes a combination of the binary digits that can be assigned to a state of the sequential network.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>G</td>
<td></td>
</tr>
</tbody>
</table>

- (B,G) 2x
- (C,F)
- (D,H) 2x
- (A,E)
- (A,B)
- (B,C) 3x
- (D,E)
- (F,G) 2x
- (D,H)
- (D,F)
- (E,G)
State Assignment Map

- K-map for the state variables in which each cell of the map denotes a combination of the binary digits that can be assigned to a state of the sequential network.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>G</td>
<td></td>
</tr>
</tbody>
</table>

- (B,G) 2x
- (C,F)
- (D,H) 2x
- (A,E)
- (A,B)
- (B,C) 3x
- (D,E)
- (F,G) 2x
- (D,H)
- (D,F)
- (E,G)
State Assignment Map

- K-map for the state variables in which each cell of the map denotes a combination of the binary digits that can be assigned to a state of the sequential network.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>G</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
State Assignment Map

- K-map for the state variables in which each cell of the map denotes a combination of the binary digits that can be assigned to a state of the sequential network.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>G</td>
<td>F</td>
<td>D</td>
</tr>
</tbody>
</table>

- (A,B) 3x
- (A,E) 2x
- (B,C) 3x
- (B,G) 2x
- (C,F)
- (D,E)
- (D,F) 2x
- (D,H)
- (E,G)
State Assignment Map

- K-map for the state variables in which each cell of the map denotes a combination of the binary digits that can be assigned to a state of the sequential network.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>G</td>
<td>F</td>
<td>D</td>
</tr>
</tbody>
</table>

- (B,G) 2x
- (C,F)
- (D,H) 2x
- (A,E)
- (A,B)
- (B,C) 3x
- (D,E)
- (F,G) 2x
- (D,H)
- (D,F)
- (E,G)
State Assignment Map

• K-map for the state variables in which each cell of the map denotes a combination of the binary digits that can be assigned to a state of the sequential network.

![State Assignment Map](image)
Transition Table

- Using the state assignment map and state table, a transition table is constructed.

![Transition Table Diagram]

**Table 7.17** Illustrations of state assignments. (a) State table. (b) Transition table for state assignment in binary order. (c) Transition table for state assignment based on guidelines.

<table>
<thead>
<tr>
<th>Present state \ (Q_1, Q_2, Q_3)</th>
<th>Next state \ (Q'_1, Q'_2, Q'_3)</th>
<th>Output (z)</th>
</tr>
</thead>
</table>
| \begin{align*}
  \*A & \rightarrow 000 \\
  B & \rightarrow 001 \\
  C & \rightarrow 011 \\
  D & \rightarrow 100 \\
  E & \rightarrow 110 \\
  F & \rightarrow 111 \\
  G & \rightarrow 101 \\
  H & \rightarrow 010
\end{align*} | \begin{align*}
  000 & \rightarrow 001 \\
  001 & \rightarrow 011 \\
  110 & \rightarrow 100 \\
  111 & \rightarrow 101 \\
  011 & \rightarrow 001 \\
  110 & \rightarrow 010 \\
  001 & \rightarrow 011 \\
  111 & \rightarrow 101
\end{align*} | \begin{align*}
  0 & \rightarrow 0 \\
  0 & \rightarrow 0 \\
  0 & \rightarrow 0 \\
  0 & \rightarrow 0 \\
  0 & \rightarrow 1 \\
  0 & \rightarrow 0 \\
  0 & \rightarrow 1 \\
  0 & \rightarrow 0
\end{align*} |
Unused States

• With $p$ bits, the number of states $s$ that can be coded is given by
  \[ 2^{p-1} < s \leq 2^p \]

• In general, when coding $s$ states with $p$ bits some binary combinations are not assigned to any state.
Unused States

• Approach 1:
  – The corresponding entries in the K-maps are don’t cares.
  – This provides greater flexibility when obtaining minimal expressions for next-state and output functions.

• Approach 2:
  – The network may enter one of the unused states (when first turned on, due to noise, hardware failure, etc.)
  – It may be desirable that the network go to some well-defined state at the end of the clock period.
  – Next state entries for each of the unused states should be specified.
Illustrating Approach 1

<table>
<thead>
<tr>
<th>Present state</th>
<th>Next state</th>
<th>Output (z)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input (x)</td>
<td>Input (x)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **A**
  - A
  - B
  - 0
  - 0

- **B**
  - C
  - D
  - 1
  - 0

- **C**
  - A
  - D
  - 0
  - 1

- **D**
  - E
  - A
  - 1
  - 1

- **E**
  - C
  - B
  - 0
  - 0

(a)

<table>
<thead>
<tr>
<th>Present state (Q₁Q₂Q₃)</th>
<th>Next state (Q₁⁺Q₂⁺Q₃⁺)</th>
<th>Output (z)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input (x)</td>
<td>Input (x)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **A** → 000
  - 000
  - 001
  - 0
  - 0

- **B** → 001
  - 010
  - 011
  - 1
  - 0

- **C** → 010
  - 000
  - 011
  - 0
  - 1

- **D** → 011
  - 100
  - 000
  - 1
  - 1

- **E** → 100
  - 010
  - 001
  - 0
  - 0

(b)
Illustrating Approach 2

<table>
<thead>
<tr>
<th>Present state $(Q_1Q_2Q_3)$</th>
<th>Next state $(Q_1^*Q_2^<em>Q_3^</em>)$</th>
<th>Output $(z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input $(x)$</td>
<td>Input $(x)$</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>$A$ 000</td>
<td>000</td>
<td>00</td>
</tr>
<tr>
<td>$B$ 001</td>
<td>010</td>
<td>10</td>
</tr>
<tr>
<td>$C$ 010</td>
<td>000</td>
<td>01</td>
</tr>
<tr>
<td>$D$ 011</td>
<td>100</td>
<td>11</td>
</tr>
<tr>
<td>$E$ 100</td>
<td>010</td>
<td>00</td>
</tr>
<tr>
<td>101</td>
<td>000</td>
<td>00</td>
</tr>
<tr>
<td>110</td>
<td>000</td>
<td>00</td>
</tr>
<tr>
<td>111</td>
<td>000</td>
<td>00</td>
</tr>
</tbody>
</table>

$\rightarrow d$
Completing the Design

• Choose which type of clocked flip-flops should be used for memory.

• Depending on this choice, appropriate excitation signals must be generated by the combinational logic that precedes the input terminals of the flip-flops.

• Excitation table can be constructed from transition table once flip-flop type is selected.
### Table 7.18

Application tables. (a) D flip-flop. (b) JK flip-flop. (c) T flip-flop. (d) SR flip-flop.

<table>
<thead>
<tr>
<th>$Q$</th>
<th>$Q^+$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>$Q$</th>
<th>$Q^+$</th>
<th>$J$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>–</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>–</td>
<td>0</td>
</tr>
</tbody>
</table>

(b)

<table>
<thead>
<tr>
<th>$Q$</th>
<th>$Q^+$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(c)

<table>
<thead>
<tr>
<th>$Q$</th>
<th>$Q^+$</th>
<th>$S$</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>–</td>
<td>0</td>
</tr>
</tbody>
</table>

(d)
From Transition Table to Excitation Table

<table>
<thead>
<tr>
<th>Present state ((Q_1Q_2Q_3))</th>
<th>Next state ((Q'_1Q'_2Q'_3))</th>
<th>Output ((z))</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (\rightarrow) 000</td>
<td>000 001</td>
<td>0 0</td>
</tr>
<tr>
<td>B (\rightarrow) 001</td>
<td>010 011</td>
<td>0 1</td>
</tr>
<tr>
<td>C (\rightarrow) 010</td>
<td>000 011</td>
<td>0 1</td>
</tr>
<tr>
<td>D (\rightarrow) 011</td>
<td>100 000</td>
<td>0 1</td>
</tr>
<tr>
<td>E (\rightarrow) 100</td>
<td>010 001</td>
<td>0 0</td>
</tr>
<tr>
<td>101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ (b) \]

<table>
<thead>
<tr>
<th>(Q)</th>
<th>(Q^+)</th>
<th>(J)</th>
<th>(K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ (b) \]

<table>
<thead>
<tr>
<th>Present state ((Q_1Q_2Q_3))</th>
<th>Excitation ((J_1K_1, J_2K_2, J_3K_3))</th>
<th>Output ((z))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input ((x))</td>
<td>Input ((x))</td>
</tr>
<tr>
<td>000</td>
<td>0, 1</td>
<td>0, 1</td>
</tr>
<tr>
<td>001</td>
<td>0, 1, -1</td>
<td>0, 1, 0</td>
</tr>
<tr>
<td>010</td>
<td>0, -1, 0</td>
<td>0, -0, 1</td>
</tr>
<tr>
<td>011</td>
<td>1, -1, -1</td>
<td>0, 1, 1</td>
</tr>
<tr>
<td>100</td>
<td>-1, 1, 0</td>
<td>0, 1</td>
</tr>
</tbody>
</table>

\[ (b) \]
K-Maps for Excitation and Output
Completing the Design with D-flip-flops

\[
\begin{align*}
J_1 &= Q_2Q_3\bar{x} \\
K_1 &= 1 \\
J_2 &= Q_3 + Q_1\bar{x} \\
K_2 &= Q_3 + \bar{x} \\
J_3 &= x \\
K_3 &= Q_2 + \bar{x} \\
z &= Q_2x + Q_3\bar{x}
\end{align*}
\]