L9 – State Assignment and gate implementation
States Assignment

- Rules for State Assignment
- Application of rule
- Gate Implementation

- Ref: text Unit 15.8
Rules for State Assignment

- Situation: You have arrived at the reduced state table and no further state reduction can be made.
- Does it matter how you assign the binary encoding to the states – YES!!!
- But how to do it!!!
Guidelines for State Assignment

- To try all equivalent state assignments, i.e., and exhaustive exploration of all possible state assignments. This is a \( n\)-\( p \) complete problem.
- Do not panic!!! (where does this come from?)
- There are guidelines that help
  - 1. States which have the same next state for a given input should be given adjacent assignments.
  - 2. States which are the next states of the same state should be given adjacent assignments.
  - And third
  - 3. States which have the same output for a given input should be given adjacent assignments.
The starting state

- Assign the starting state to the “0” square on an assignment map. (An assignment map looks much like a K-map for logic minimization.)
Reason for assign "0"

- Reasons for assigning "0" as the starting state:
  - The clear input on Flip Flops can be used for initialization.
  - The clear input can also be used on a reset.
  - The alternative is error prone – using a combination of preset and clears to set a specific value can lead to implementation errors.
  - A good practice even when using FPGAs.
Guidelines

- Adjacency conditions from Guideline 1 and those from Guideline 2 that are required 2 or more times should be satisfied first.

- Example – Guideline 1 for the table $S_0$, $S_2$, $S_4$, and $S_6$ should be made adjacent as they all have $S_1$ as the next state on a 0 input.

- $S_3$ and $S_5$ should have adjacent assignment.

- $S_4$ and $S_6$ should have adjacent assignment.
Using guidelines

- From the state table find the following groupings:
  1. (S0, S1, S3, S5) (S3, S5) (S4, S6) (S0, S2, S4, S6)
  2. (S1, S2) (S2, S3) (S1, S4) (S2, S5)2x (S1, S6)2x
Two possible ways

- Two possible ways of satisfying the guidelines are:
  1. \((S_0,S_1,S_3,S_5), (S_3,S_5), (S_4,S_6), (S_0,S_2,S_4,S_6)\)
  2. \((S_1,S_2), (S_2,S_3), (S_1,S_4), (S_2,S_5)_{2x}, (S_1,S_6)_{2x}\)
Next state maps

- Next state maps may help choose the better assignment.
- Look at the next state given current state and input and how this will simplify K-maps for logic.
Choose an assignment

- Choose an assignment and implement in gates. Using the left assignment map get the next state map below with encoding.
- Map the encoding to K-maps
Implement in gates

- Notes on implementation
  - All F/F outputs are used
  - 6 gates are needed for next state generation only
    1 of which is 3 inputs.
Another example

Example 15-16 in text

- Use guidelines
- Next states (b,d) (c,f) (b,e) (a,c)
- Next state of a state (a,c)2x (d,f) (d,b) (b,f) (c,e)

- But is state table minimum?
Assignment map

- State table is not minimum but will continue
- The two assignment maps are
Transition table

- Resulting in a transition table of and equations of

(Figure 15-17) from the transition table. The D flip-flop input equations can be read directly from these maps:

\[
D_1 = Q_1^+ = XQ_1Q_2 + XQ_1'
\]
\[
D_2 = Q_2^+ = Q_3
\]
\[
D_3 = Q_3^+ = XQ_3'Q_2 + X'Q_3
\]

and the output equation is

\[
Z = XQ_2Q_3 + X'Q_2'Q_3 + XQ_2Q_3'
\]

The cost of realizing these equations is 10 gates and 26 gate inputs.

<table>
<thead>
<tr>
<th>$Q_3Q_2Q_1$</th>
<th>$Q_1^+Q_2^+Q_3^+$</th>
<th>$X = 0$</th>
<th>$X = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0</td>
<td>100 000</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>1 1 1</td>
<td>011 010</td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td>000 100</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td>011 111</td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>1 0 1</td>
<td>111 010</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td>000 101</td>
<td>1 0</td>
<td></td>
</tr>
</tbody>
</table>
Next state generation K-maps

- The K-maps for next state generation are
Another example

- From our previous work.

<table>
<thead>
<tr>
<th>Present State</th>
<th>X=0</th>
<th>X=1</th>
<th>X=0</th>
<th>X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>S1</td>
<td>S4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S1</td>
<td>S1</td>
<td>S2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S2</td>
<td>S3</td>
<td>S4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>S5</td>
<td>S2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>S3</td>
<td>S4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5</td>
<td>S1</td>
<td>S2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Use guidelines

- Same next state
  - (S0, S1, S5) (S2, S4) (S0, S2, S4) (S1, S3, S5)

- Next state pairs
  - (S1, S4) (S1, S2) 2x (S3, S4) 2x (S2, S5)

<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>S0</td>
<td>S1</td>
<td>S4</td>
</tr>
<tr>
<td>S1</td>
<td>S1</td>
<td>S2</td>
</tr>
<tr>
<td>S2</td>
<td>S3</td>
<td>S4</td>
</tr>
<tr>
<td>S3</td>
<td>S5</td>
<td>S2</td>
</tr>
<tr>
<td>S4</td>
<td>S3</td>
<td>S4</td>
</tr>
<tr>
<td>S5</td>
<td>S1</td>
<td>S2</td>
</tr>
</tbody>
</table>
The assignment map

- Choose S0 as the “0” state and then use guidelines
- A possible solution
Next State Table

- Enter the state assignment onto the table
- Then generate K-maps and generate logic

<table>
<thead>
<tr>
<th>Present State</th>
<th>Next State</th>
<th>Output</th>
<th>( X = 0 )</th>
<th>( X = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>( \bar{X} = 0 )</td>
<td>( \bar{X} = 1 )</td>
<td>( X = 0 )</td>
<td>( X = 1 )</td>
</tr>
<tr>
<td>S0 000</td>
<td>S1 001</td>
<td>S4 101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S1 001</td>
<td>S1 001</td>
<td>S2 100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S2 100</td>
<td>S3 111</td>
<td>S4 101</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>S3 111</td>
<td>S5 011</td>
<td>S2 100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S4 101</td>
<td>S3 111</td>
<td>S4 101</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S5 011</td>
<td>S1 001</td>
<td>S2 100</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
The K maps

- Generate the K maps
  - Next State logic A (2 gates) B (1 gate) C (2 gates)

\[ A = X + AB' \]

\[ B = X'A \]

\[ C = X' + C' + AB' \]
K map for the output Z

- 3 gates for the output (2-3 input AND) (1 OR)
- Total logic count
  - 3 D F/Fs
  - 2 – 3 input AND gates
  - 3 – 2 input AND gates
  - 2 – 2 input OR gates
  - 1 – 3 input OR gate

\[
Z = X'AC' + XA'B
\]
Lecture summary

- Have seen several examples of implementation from the statement of the problem (specification) to implementation.