FSM (Finite State Machine) Optimization

State tables

State minimization
- identify and remove equivalent states

State assignment
- assign unique binary code to each state

Combinational logic optimization
- use unassigned state-codes as don’t care

net-list
**State Minimization**

**Goal**: identify and remove redundant states (states which cannot be observed from the FSM I/O behavior)

**Why**

1. **Reduce number of latches**
   - assign minimum-length encoding
   - only as the logarithm of the number of states

2. **Increase the number of unassigned state codes**
   - heuristic to improve state-assignment and logic-optimization
State Minimization Definition

- **Completely-specified state machine**
  - two states are equivalent if outputs are identical for all input combinations
  - Next states are equivalent for all input combinations
  - equivalence of states is an equivalence relation which partitions the states into disjoint equivalence classes

- **Incompletely specified state machines**
  - this problem is more difficult
Basic Principle of State Minimization for Completely Specified Machines

Procedure (fast) for lazy students

- Any two states of Moore Machine that have the same output and transit to the same states under the same input symbols are equivalent and can be combined.
- This step is repeated until no more equivalent states exist.

States SZ and SC are equivalent and are combined to one state by pointing all arrows that go to SY to state SZ and removing SY with its all arrows.
Procedure (fast) for lazy students (for Mealy machines)

- Any two states of Mealy Machine that have the same output for the same input symbol and transit to the same states under the same input symbols are equivalent and can be combined.
- This step is repeated until no more equivalent states exist.

States $SZ$ and $SZ$ are equivalent and are combined to one state by pointing all arrows that go to $SY$ to state $SZ$ and removing $SY$ with its all arrows.
1. Partition the set of internal states based on input output values asserted in the state
2. Define the partitions so that all states in a partition transition into the same next-state partition (under corresponding inputs)

*Only for Completely specified Machines*
Example (FSM in Kiss format)

Ex:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>B</td>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>E</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>C</td>
<td>F</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>D</td>
<td>H</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>D</td>
<td>G</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>E</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>E</td>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>F</td>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>F</td>
<td>E</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>G</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>G</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>H</td>
<td>H</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>H</td>
<td>A</td>
<td>0</td>
</tr>
</tbody>
</table>

- States A, C and E can be combined to one state
- States B and F can be combined to one state
- G has other input-output response than other states
- D has other input-output response than other states because it goes to G which is known to be non-equivalent state-goes to red and blue groups
- B and F go to D
- Please check this using triangular table
- You can also mark each new group with a new symbol and check transitions to thus marked groups