Unit 14

Derivation of State Graphs and Tables

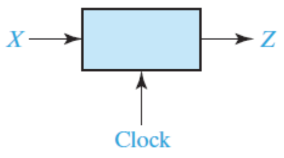
Logic Circuits (Spring 2022)

Design Procedure

1. Determine the required relationship between the input and output sequences and derive a state table. For many problems, it is easiest to first **construct a state graph**.
2. Reduce the table to a minimum number of states.
3. If the reduced table has m states ($2^{n-1} < m \leq 2^n$), n flip-flops are required. **Assign a unique combination of flip-flop states** to correspond to each state in the reduced table.
4. **Form the transition table** to specify the next states of the flip-flops and the output in terms of the present states of the flip-flops and the input.
5. Plot next-state maps for each flip-flop and **derive the flip-flop input equations**. **Derive the output functions**.
6. Realize the flip-flop input equations and the output equations using the available logic gates.
7. Check your design by signal tracing, computer simulation, or laboratory testing.

Sequence Detector by Mealy Machine

- Specification
  - Examine a string of 0's and 1's applied to the X input
  - Generate an output $Z = 1$ only when an input sequence '101' occurs
  - Assume that the input X can only change between clock pulses.
- Block diagram

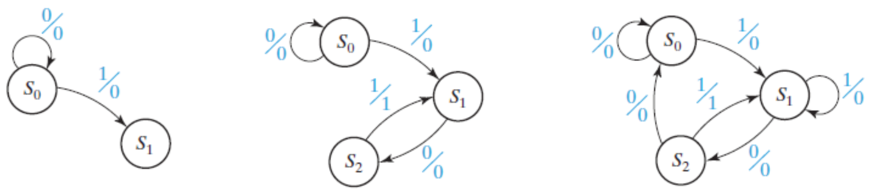


- Example input/output sequence

$X =$	0	0	1	1	0	1	1	0	0	1	0	1	0	1	0	0
$Z =$	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0
(time:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15)

Sequence Detector by Mealy Machine

- We do not know how many flip-flops are required
- State graph



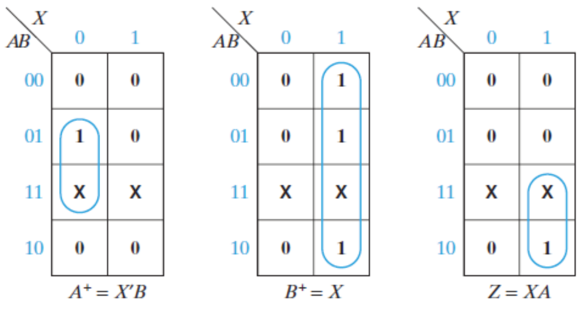
- State table

Present State	Next State $X = 0$	Next State $X = 1$	Present Output $X = 0$	Present Output $X = 1$
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_1	0	1

AB	A^+B^+ $X = 0$	A^+B^+ $X = 1$	Z $X = 0$	Z $X = 1$
00	00	01	0	0
01	10	01	0	0
10	00	01	0	1

Sequence Detector by Mealy Machine

- Next-state and output maps

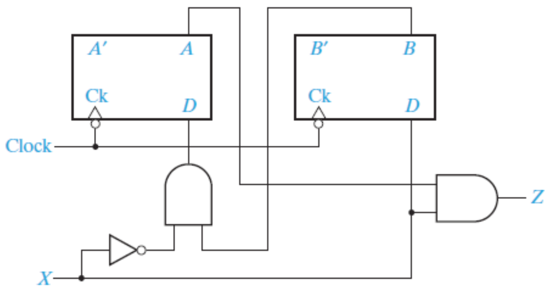


- Flip-flop input equations and output equation

$$D_A = A^+ = X'B$$
$$D_B = B^+ = X,$$
$$Z = XA$$

Sequence Detector by Mealy Machine

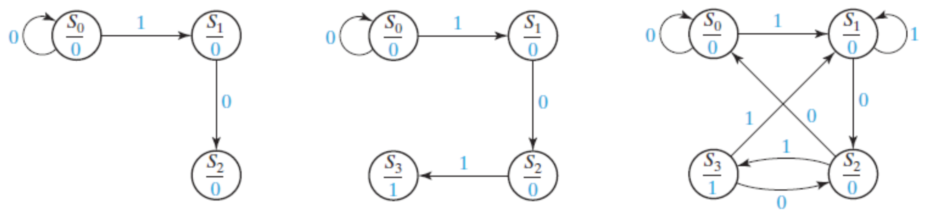
- Logic diagram



- Limitations
  - Read the value of Z after the input X has changed and before the active clock edge to avoid false outputs

Sequence Detector by Moore Machine

- The output changes with the state instead of with the state transition
- State graph



- State table

Present State	Next State $X=0$	Next State $X=1$	Present Output(Z)
S_0	S_0	S_1	0
S_1	S_2	S_1	0
S_2	S_0	S_3	0
S_3	S_2	S_1	1

AB	A^+B^+ $X=0$	A^+B^+ $X=1$	Z
00	00	01	0
01	11	01	0
11	00	01	0
10	11	01	1

Sequence Detector by Moore Machine

- Next-state and output maps
- Flip-flop input equations and output equation

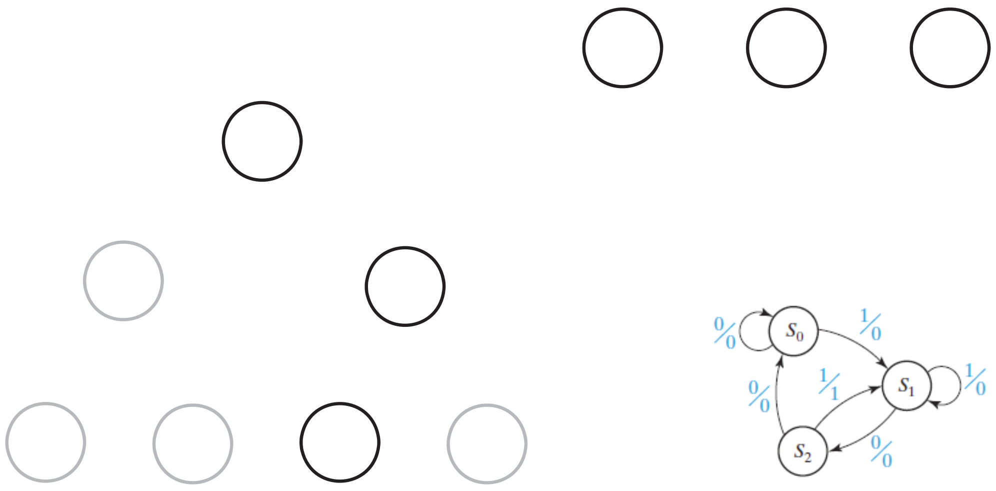
Sequence Detector by Moore Machine

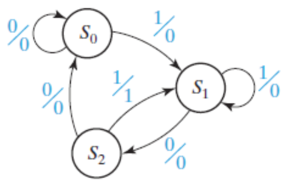
- Logic diagram

- Limitations
  - Could read the value of *Z* any time
  - However, the output *Z* does not count the current input *X* into consideration

State Graph for Mealy Model (Revisited)

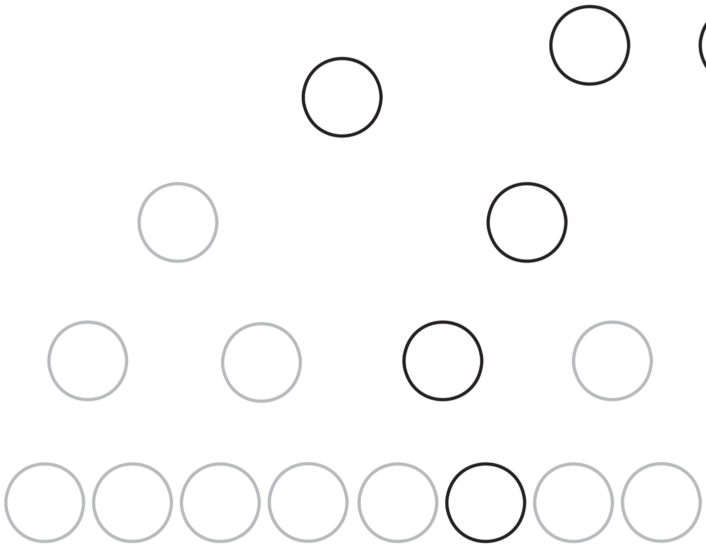
Detecting “101” sequence

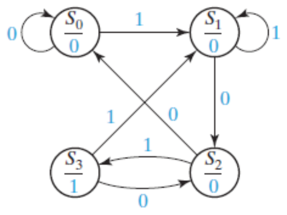




State Graph for Moore Model (Revisited)

Detecting “101” sequence





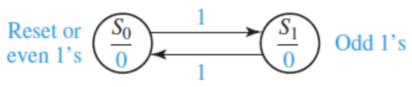
More Complex Design Problems

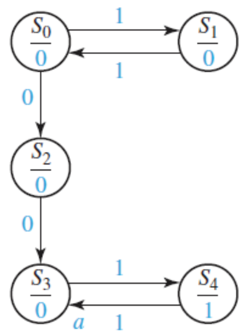
- Specifications
  - Moore sequential circuit
  - One input X
  - One output Z
  - The output Z is 1 if <u>the total number of 1’s received is odd</u> and <u>at least two consecutive 0’s have been received</u>
  - Some sample cases

```
X =   1 0 1 1 0 0 1 1
      ↑     ↑   ↑ ↑ ↑
      a     b   c d e
Z = (0) 0 0 0 0 0 1 0 1
```

More Complex Design Problems

- State diagram

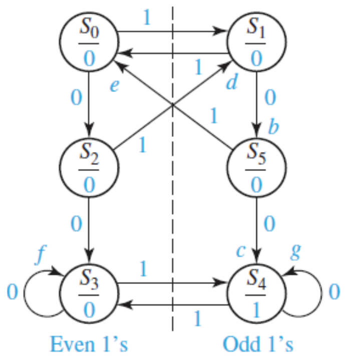




State	Sequence Received
S_0	Reset or even 1's
S_1	Odd 1's
S_2	Even 1's and ends in 0
S_3	Even 1's and 00 has occurred
S_4	00 has occurred and odd 1's

More Complex Design Problems

- State diagram



State	Input Sequences
S_0	Reset or even 1's
S_1	Odd 1's
S_2	Even 1's and ends in 0
S_3	Even 1's and 00 has occurred
S_4	Odd 1's and 00 has occurred
S_5	Odd 1's and ends in 0

Procedure to Derive State Graphs

1. First, construct some sample input and output sequences to make sure that you understand the problem statement.
2. Determine under what conditions, if any, the circuit should reset to its initial state.
3. If only one or two sequences lead to a nonzero output, a good way to start is to construct a partial state graph for those sequences.
4. Another way to get started is to determine what sequences or groups of sequences must be remembered by the circuit and set up states accordingly.
5. Each time you add an arrow to the state graph, determine whether it can go to one of the previously defined states or whether a new state must be added.
6. Check your graph to make sure there is one and only one path leaving each state for each combination of values of the input variables.
7. When your graph is complete, test it by applying the input sequences and making sure the output sequences are correct.