Chapter 9
Introduction to Finite State Machines
SKEE2263 Digital Systems

Mun'im/Ismahani/Izam

FEE, Universiti Teknologi Malaysia

April 19, 2016

Table of Contents

1 FSM Modeling

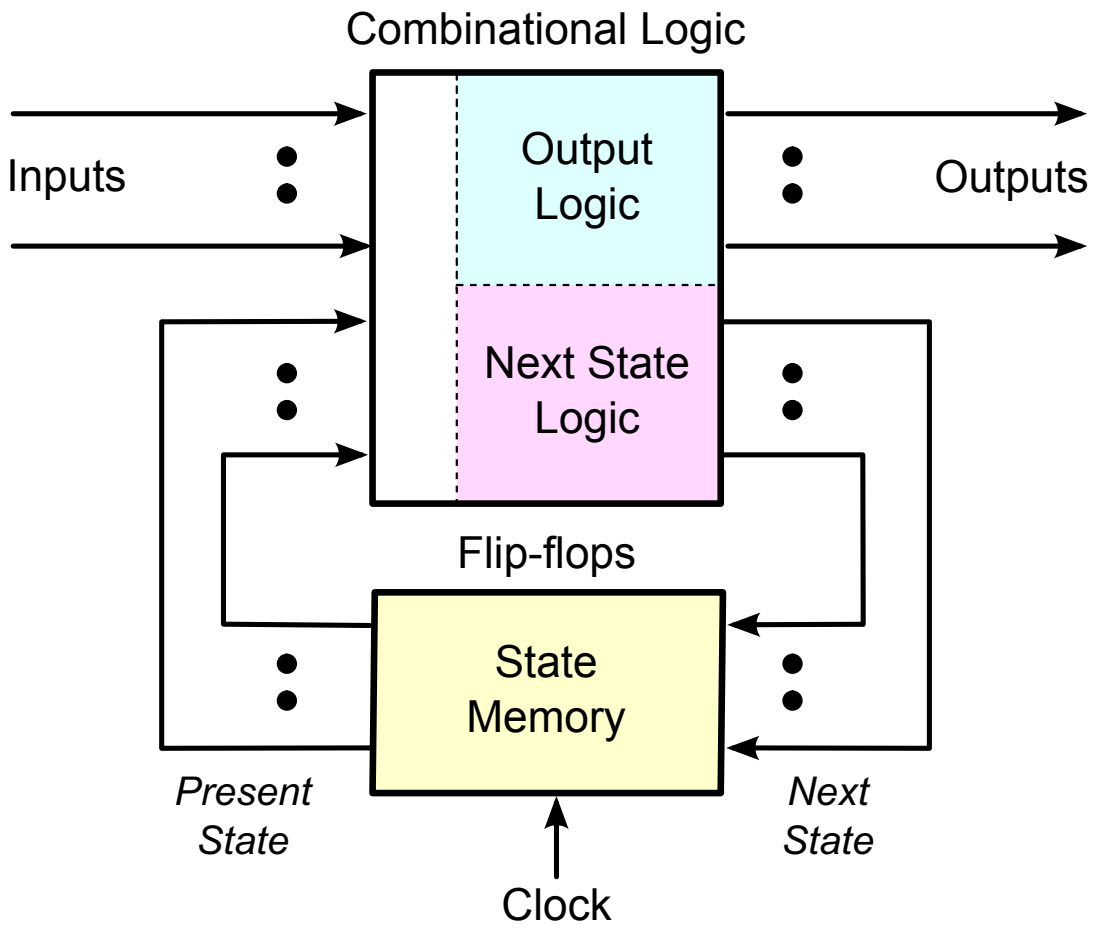
2 FSM Design

3 Moore FSM

4 Mealy FSM

5 Moore vs Mealy

General Structure of Finite State Machines (FSM)



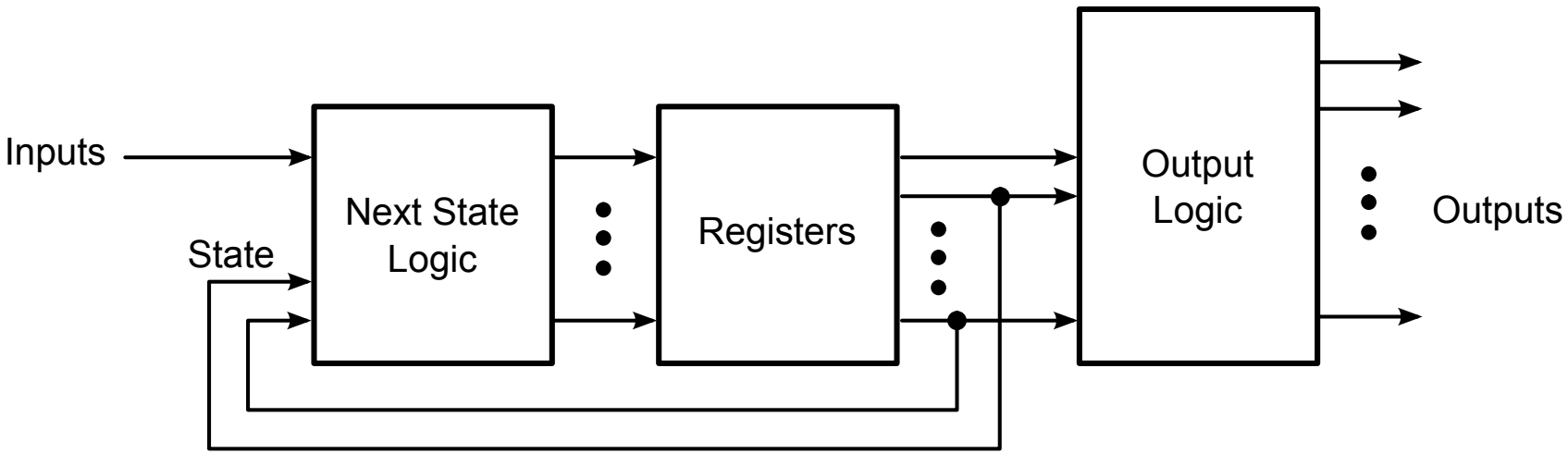
Components & Types

Components:

- **State memory** to store *present state* and has 2^n distinct states which n represents the number of flip-flops used.
- **Next state logic** to determine the *next state* when state transition occurs.
- **Output logic** to determine the output as a function of current state and inputs.

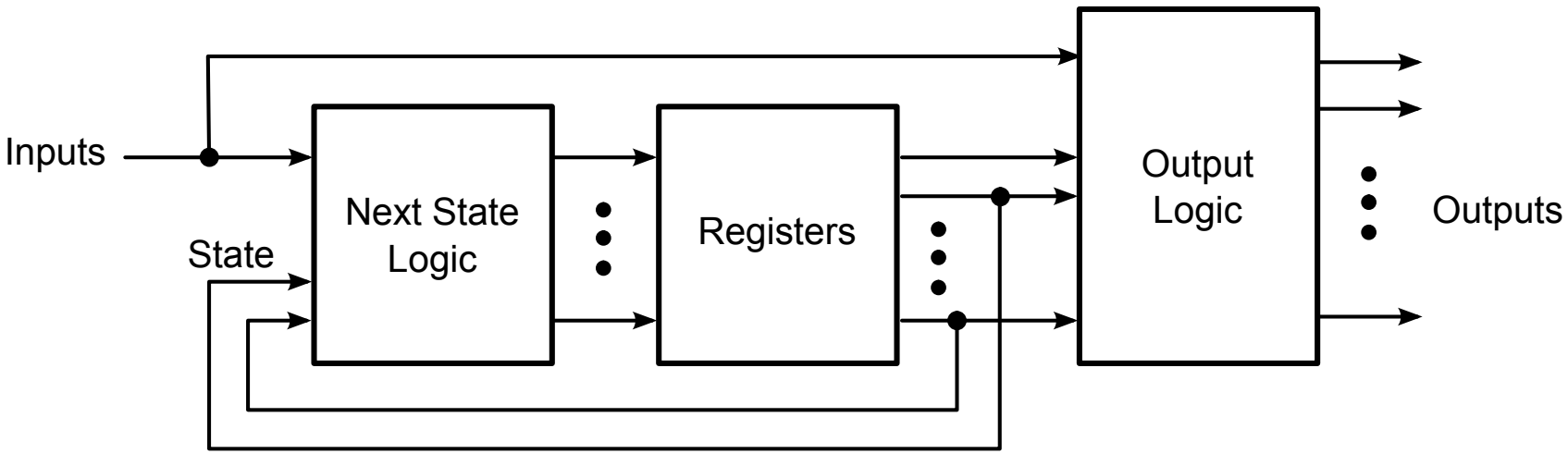
Two widely known types of FSM models are Mealy and Moore.

General Structure of Moore FSM



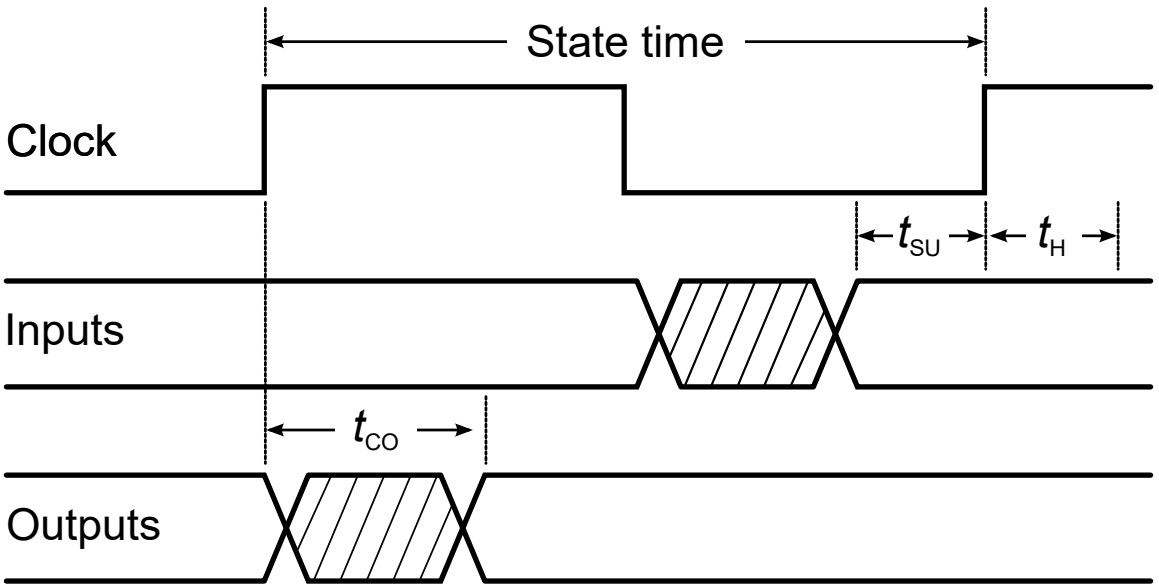
- **Moore model**: The output depends on the present state *only*.

General Structure of Mealy FSM

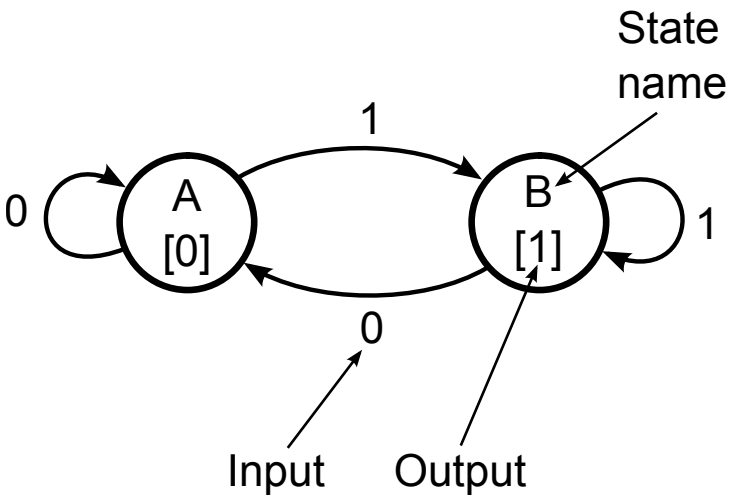


- **Mealy model**: The output depends on the present state *and* the present inputs.

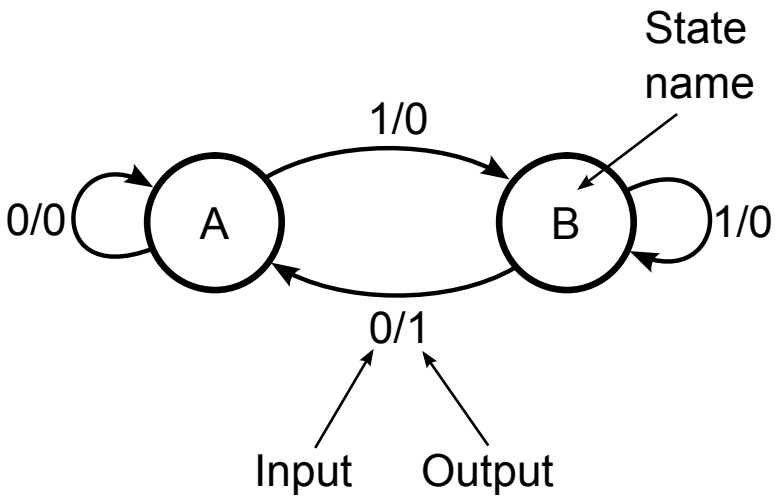
FSM Timing



State Diagrams



Moore.



Mealy.

State Tables

Moore Symbolic State Table

Present State	Input A	Next State	Output B	Comments
A	0	A	0	Remain idle in starting state
	1	B		Go to next state
B	0	A	1	Go back to starting state
	1	B		Remain in current state

Mealy Symbolic State Table

Present State	Input A	Next State	Output B	Comments
A	0	A	0	Remain idle in starting state
	1	B	0	Go to next state
B	0	A	1	Go back to starting state
	1	B	0	Remain in current state

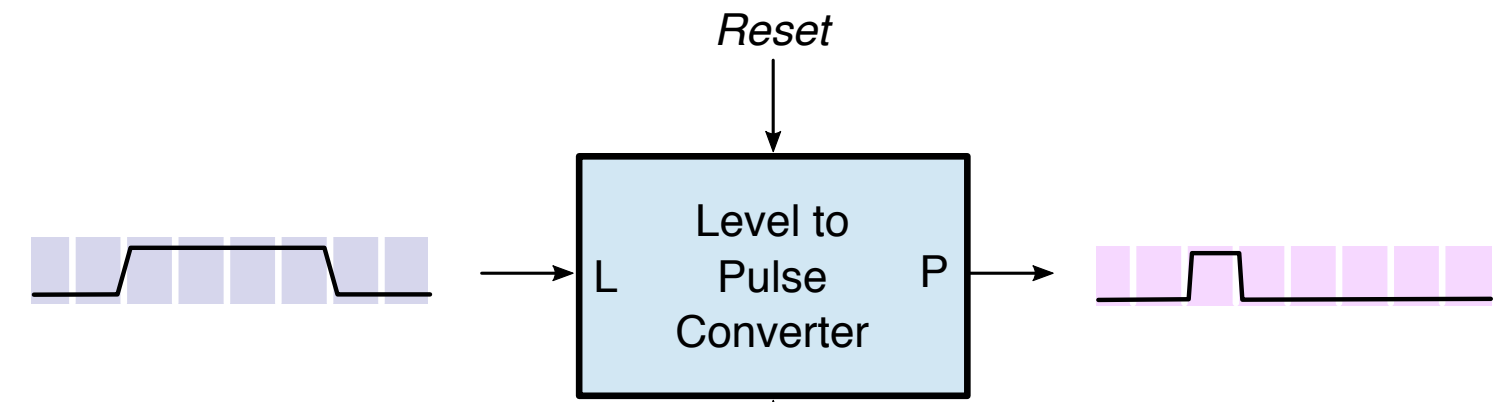
State Encoding

No	Binary	Gray	Johnson	One-Hot	Almost One-Hot
0	000	000	0000	00000001	0000000
1	001	001	0001	00000010	0000001
2	010	011	0011	00000100	0000010
3	011	010	0111	00001000	0000100
4	100	110	1111	00010000	0001000
5	101	111	1110	00100000	0010000
6	110	101	1100	01000000	0100000
7	111	100	1000	10000000	1000000

FSM Design Procedure

1. Conceptualize – Understand the statement of the specification:
   - Define all **inputs** and **outputs**
   - Determine system constrains
2. Translate the concept into a **state diagram**:
   - Determine the **number of states** required by the system
   - Determine the required **transitions**
3. **Assign** a unique binary number to each state.
4. Create the **state table**.
5. Extract the **equations**: Express the logic circuits as Boolean equations.
6. **Implement** the FSM: Enter and verify the design.

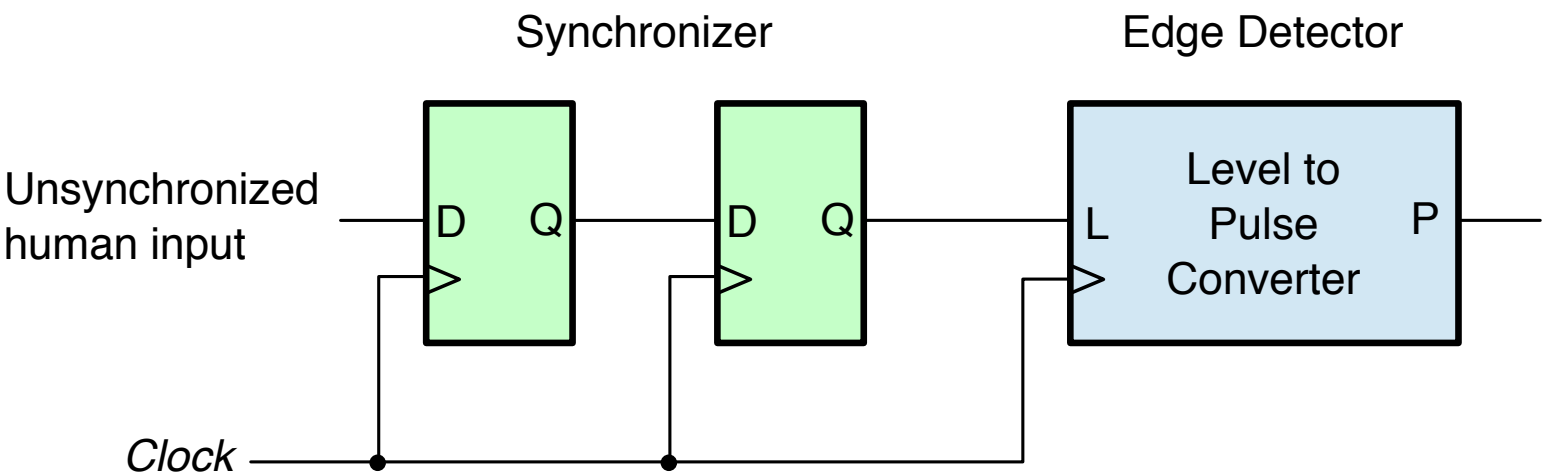
Level to Pulse Converter Top Level



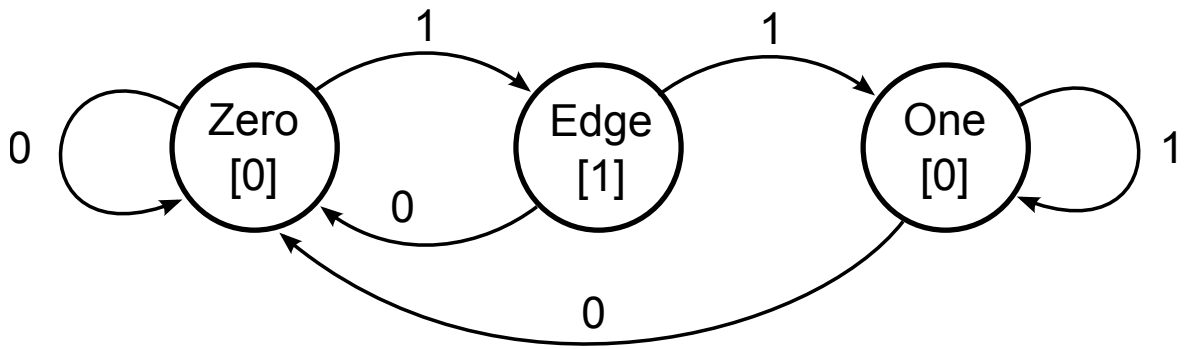
Whenever input L goes from low to high...

... output P produces a single pulse, one clock period wide.

Level to Pulse Converter Block Diagram



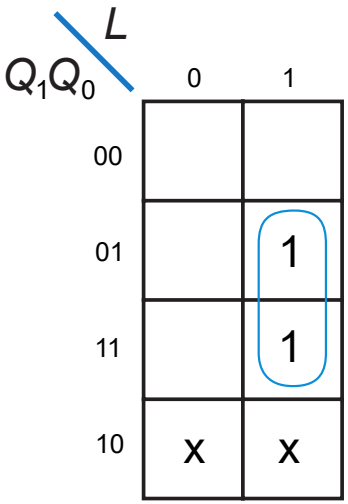
Level to Pulse Converter Moore State Diagram



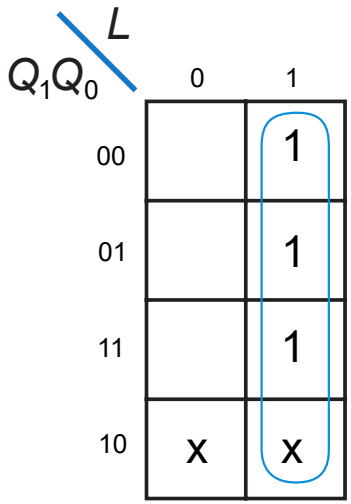
Level to Pulse Converter State Table

Present State		**In**	**Next State**		**Out**	**Comments**
Q_1	Q_0	L	Q_1^+	Q_0^+	P	
0	0	0	0	0	0	Waiting for rising edge
0	0	1	0	1	0	
0	1	0	0	0	1	Edge detected
0	1	1	1	1	1	
1	1	0	0	0	0	Waiting for falling edge
1	1	1	1	1	0	

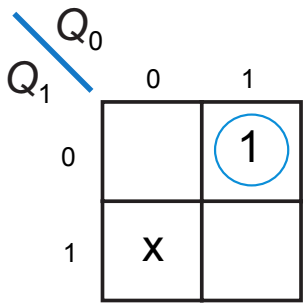
Level to Pulse Converter Kmaps



$$Q_1^+ = LQ_0$$

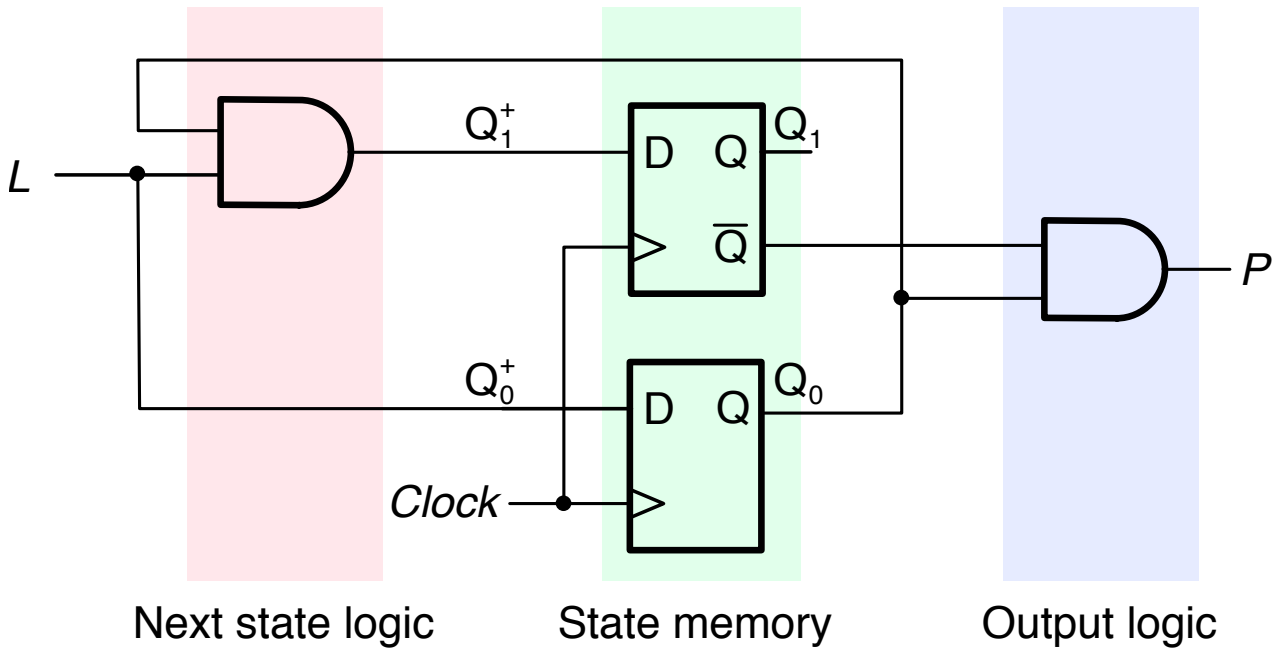


$$Q_0^+ = L$$

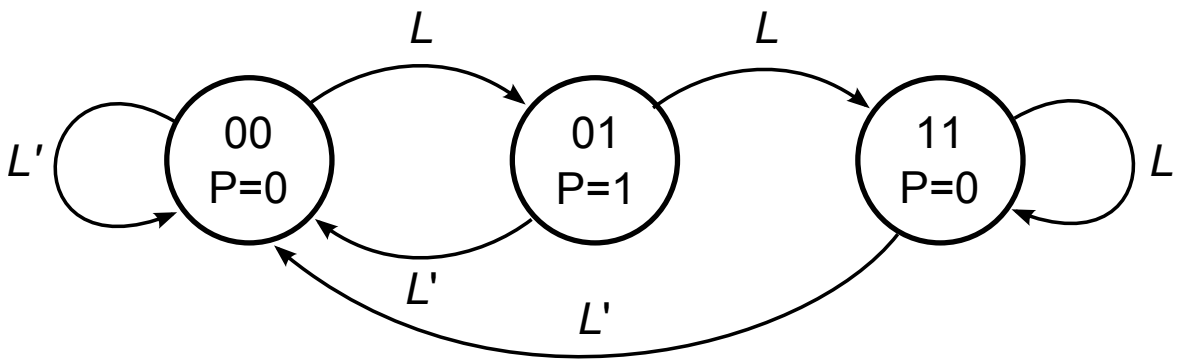


$$P = Q_1'Q_0$$

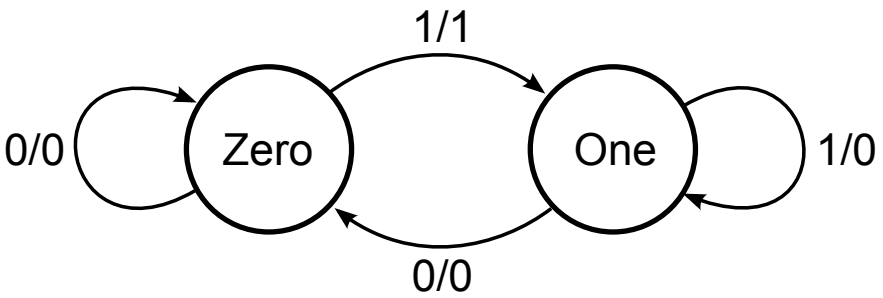
Level to Pulse Converter Schematic: Moore Type



Level to Pulse Converter Moore Alternate State Diagram



Level to Pulse Converter Mealy State Diagram

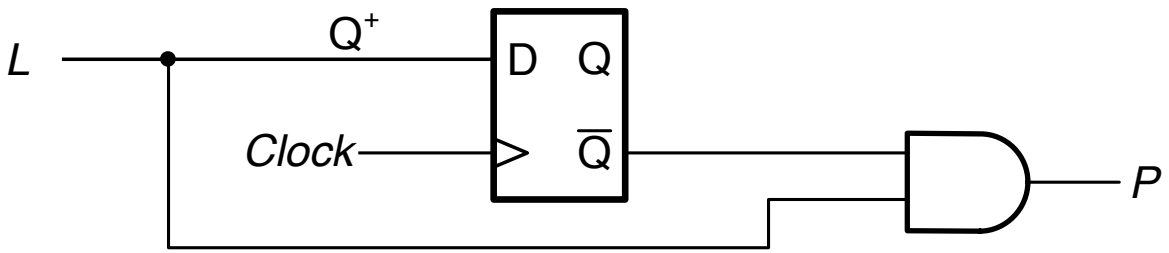


Level to Pulse Converter Mealy State Table

Current State	In	Next State	Out	Comments
Q	L	Q^+	P	
0	0	0	0	Waiting for rising edge
0	1	1	1	
1	0	0	0	Waiting for falling edge
1	1	1	0	

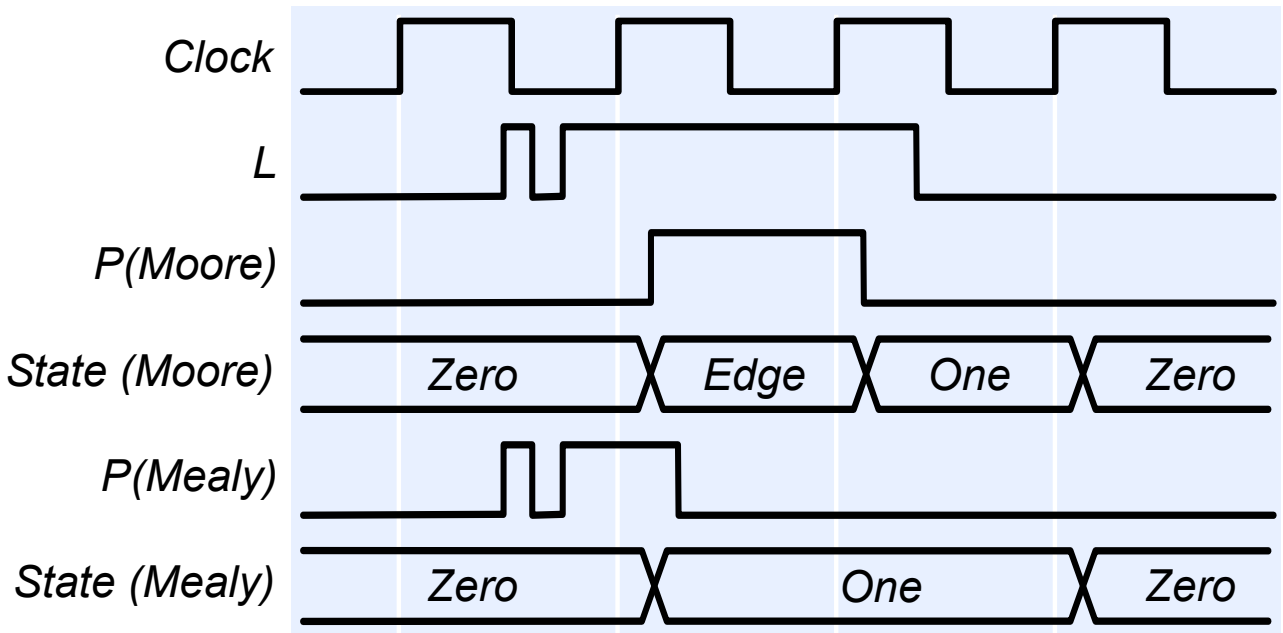
$$Q^+ = L$$
$$P = Q'L$$

Level to Pulse Converter Schematic: Mealy Type



Moore vs Mealy

- Mealy: output = f(state, inputs)
- Moore: output = f(state only)



Moore vs Mealy level to pulse converter.

Synchronous Mealy

- Mealy responds to inputs faster than Moore, but...
- Mealy has glitchy outputs
  - outputs can change in the middle of a clock period
  - i.e. asynchronous outputs
- Solution: pass the output through a register