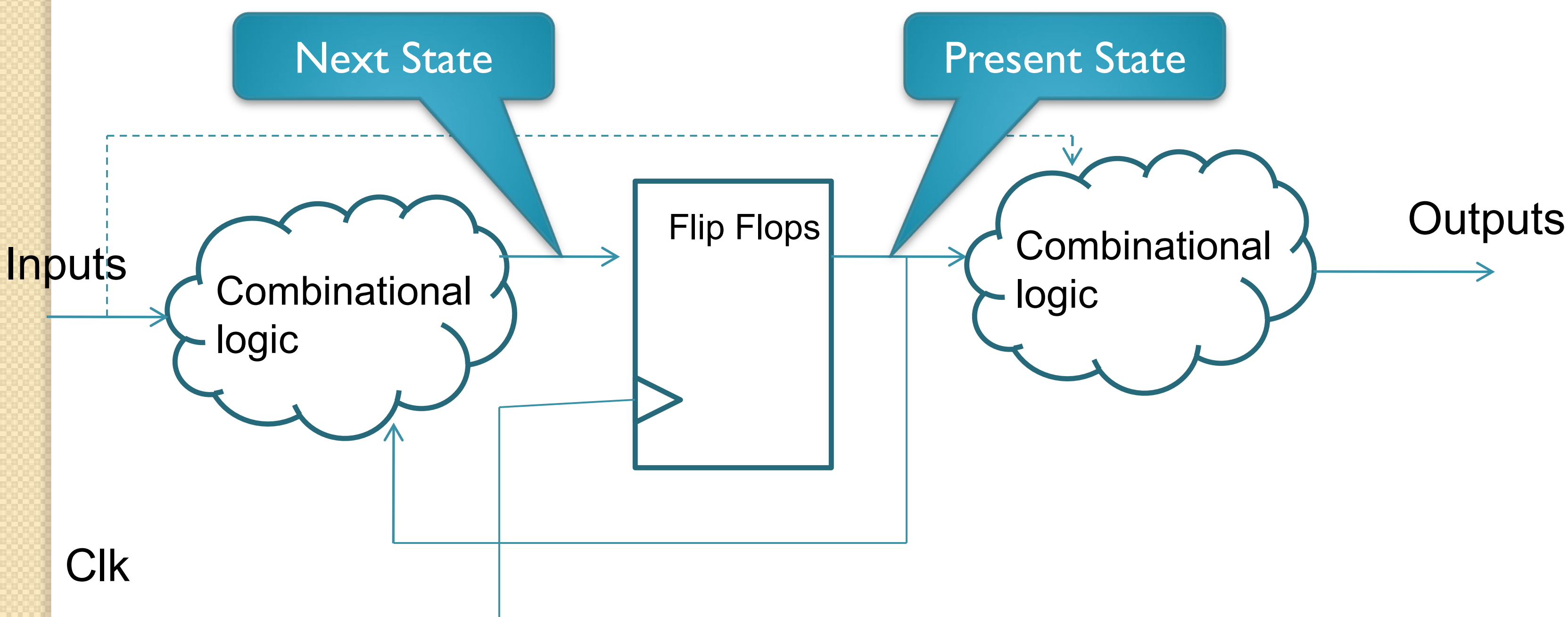
Finite State Machines

What is a Finite State Machine?

- A state machine is a digital device that traverses through a predetermined sequence of states in an orderly fashion.
- The machine is in only one state at a time; the state it is in at any given time is called the *current state*.
- It can change from one state to another when initiated by a triggering event or condition, this is called a *transition*
- A particular FSM is defined by a list of its states, and the triggering condition for each transition.

State Machines

- Synchronous Sequential Circuit
  - Circuit whose outputs depend on both its current inputs and its past sequence of inputs

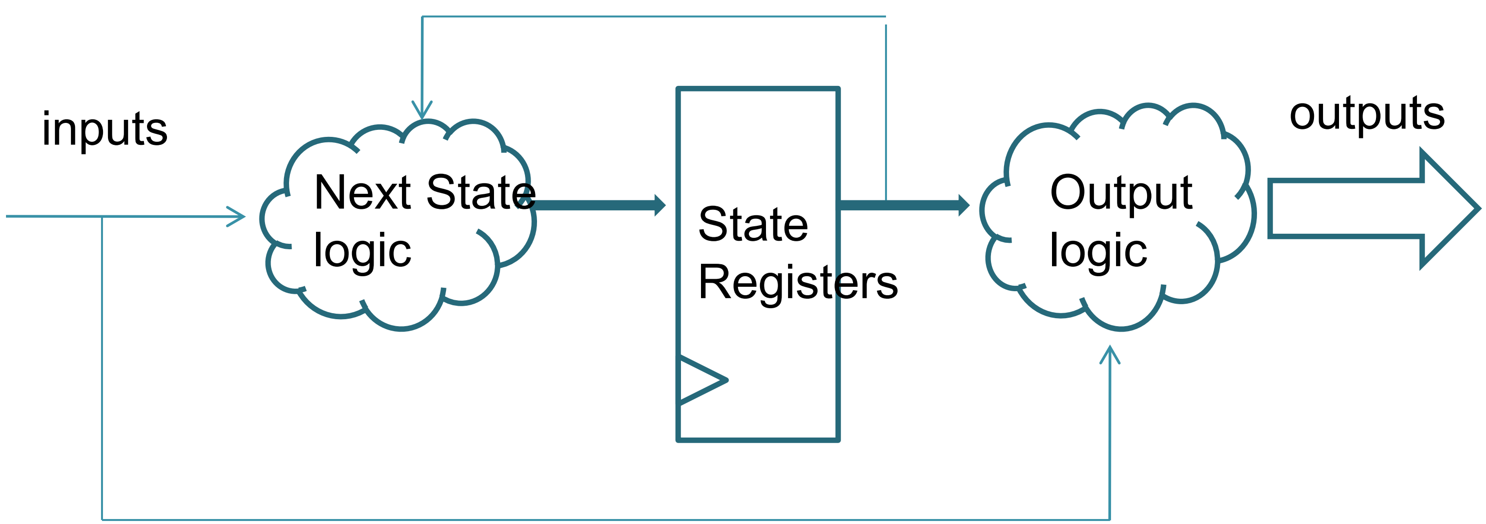


State

- Q's or flip-flop outputs known as the **state**
  - Can be encoded as a binary number
    - For n flip flops (n state variables) there are 2^n possible states
  - Can also be encoded as one-hot
    - For n flip flops (n state variables) there are n possible states
- Inputs and current state determine next state

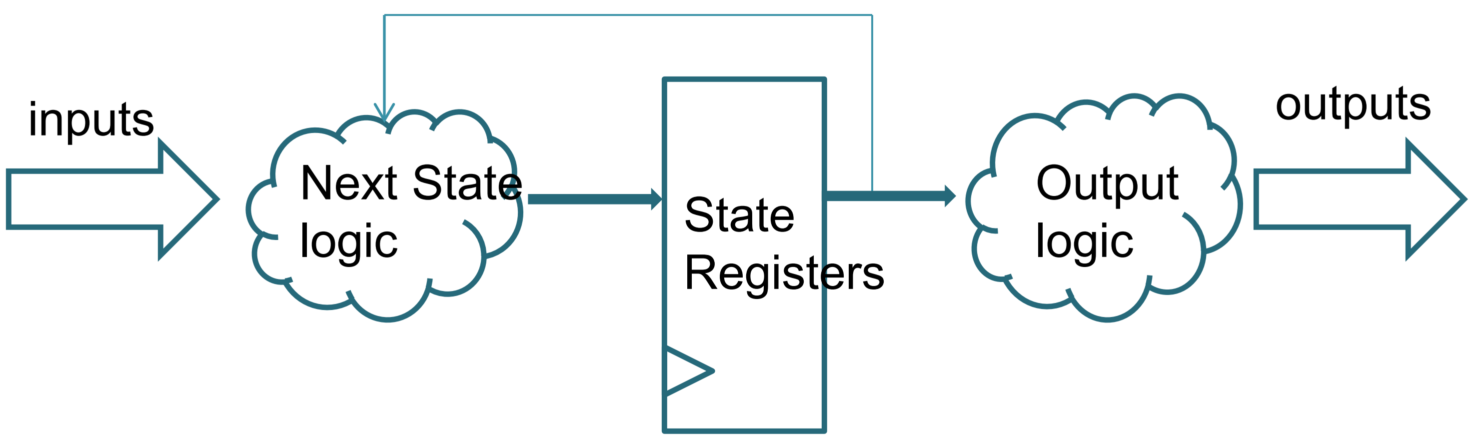
Mealy State Machine

- A Mealy state machine's outputs are based on a logical combination of the inputs and the current state



Moore State Machine

- A Moore state machine's outputs are based ONLY on the current state



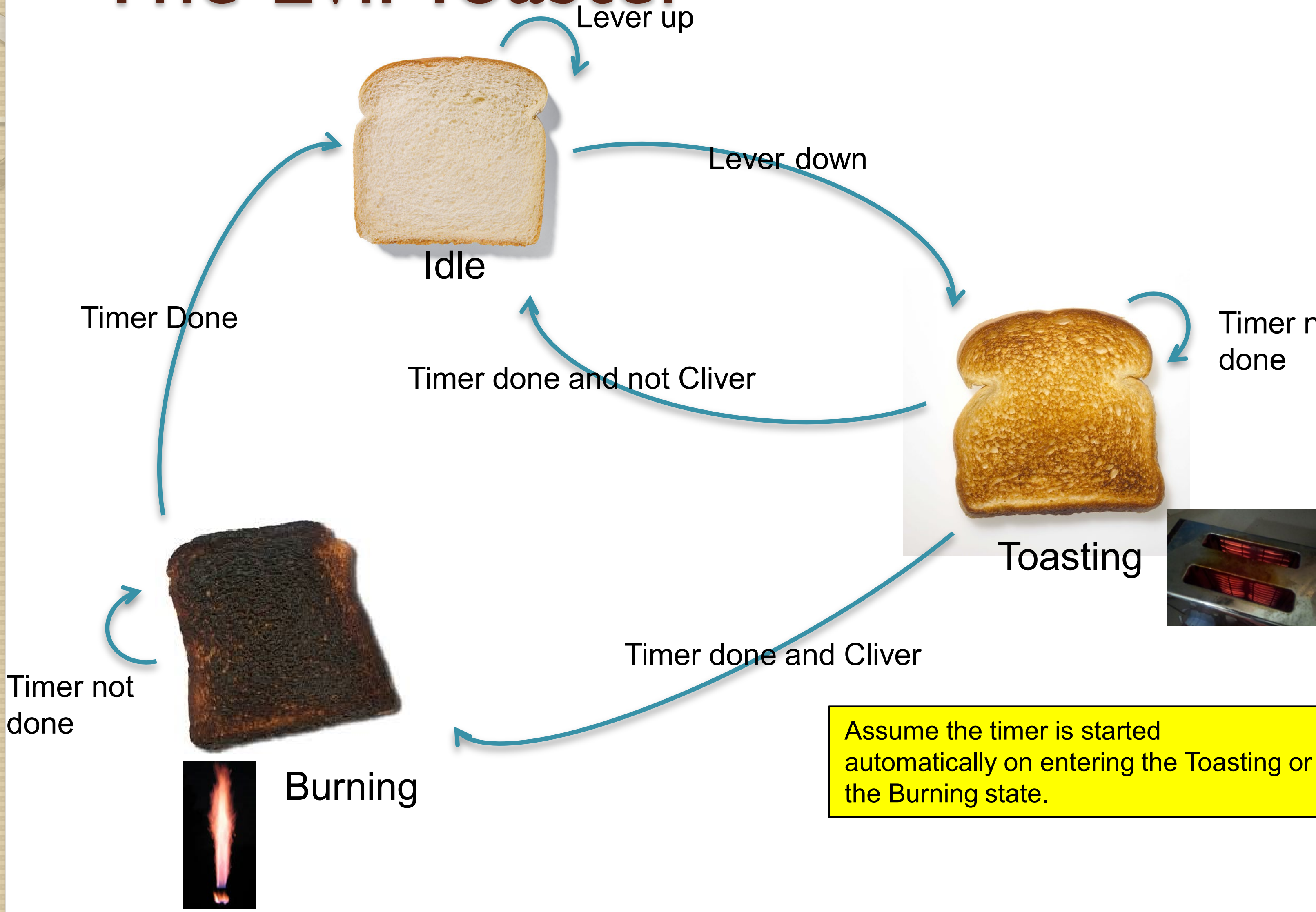
State Machine Example

- The evil toaster
  - Inputs:
    - User (Cliver = 1, not Cliver = 0)
    - Lever (up = 1, down = 0)
    - Timer (done = 1, not done = 0)
  - Outputs
    - Coil (on = 1, off = 0)
    - Flame thrower (on = 1, off = 0)
  - States
    - Idle
    - Toasting
    - Burning

The Evil Toaster

- The toaster sits in the idle state when not in use
- When the user pushes down the lever, the toaster goes to the toasting state
- When in the toasting state:
  - If timer is done and the user is not Prof. Cliver, the toaster returns to idle state
  - If timer is done and the user is Prof. Cliver, the toaster enters the burning state
- When in the burning state if timer is done, the toaster returns to idle

The Evil Toaster



Assume the timer is started automatically on entering the Toasting or the Burning state.

The Evil Toaster

	Outputs	
State	**Coil**	**Flame Thrower**
Idle	Off	Off
Toasting	On	Off
Burning	Off	On

Evil Toaster Logic Design

- Use one-hot encoding
  - In one-hot encoding use 1 flip flop for each state
  - Only one flip flop is set at a time

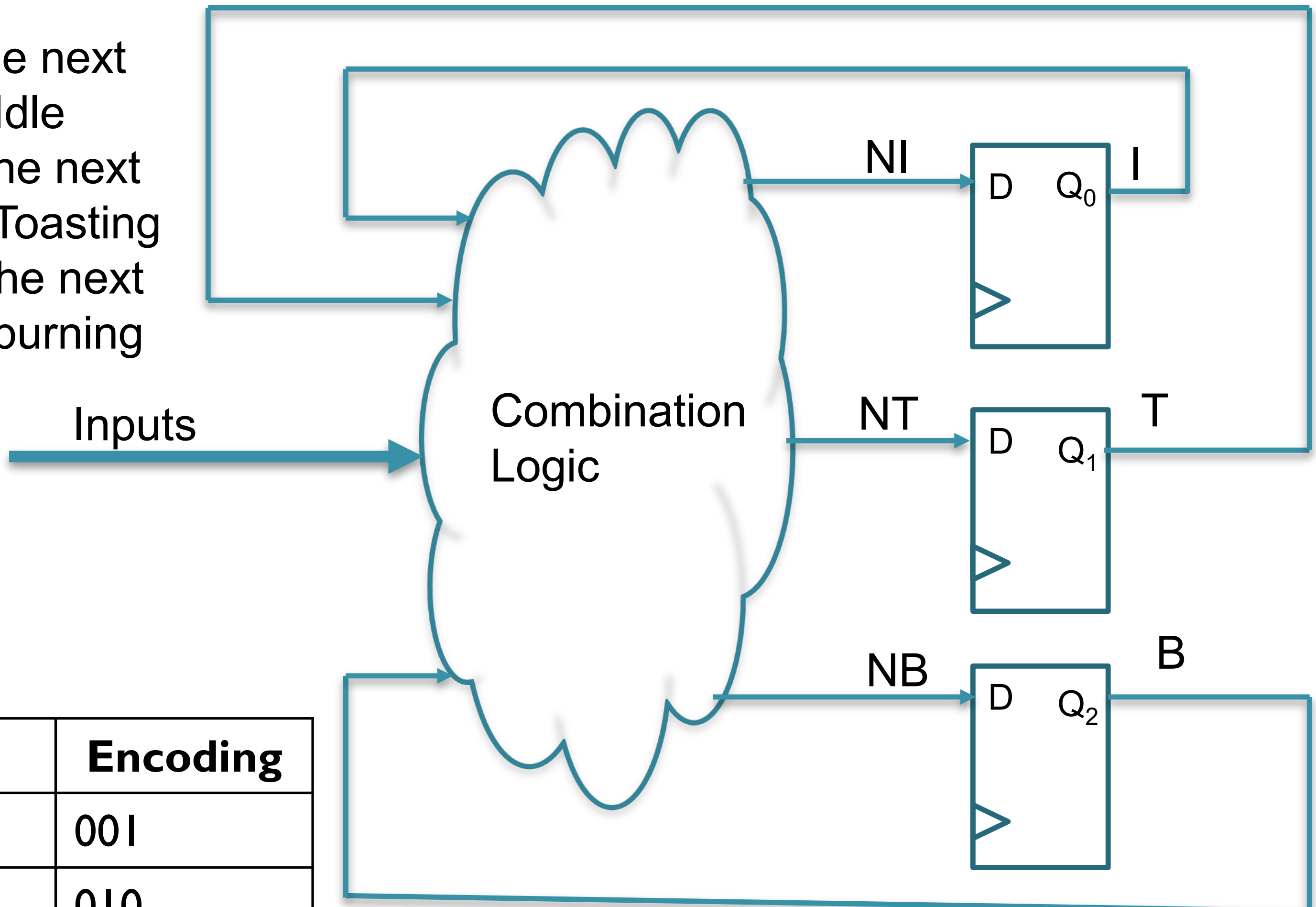
State	Encoding $Q_2Q_1Q_0$
Idle	001
Toasting	010
Burning	100

Note that in each state 1 and only 1 FF is active high

Evil Toaster Logic Design

On the clock,

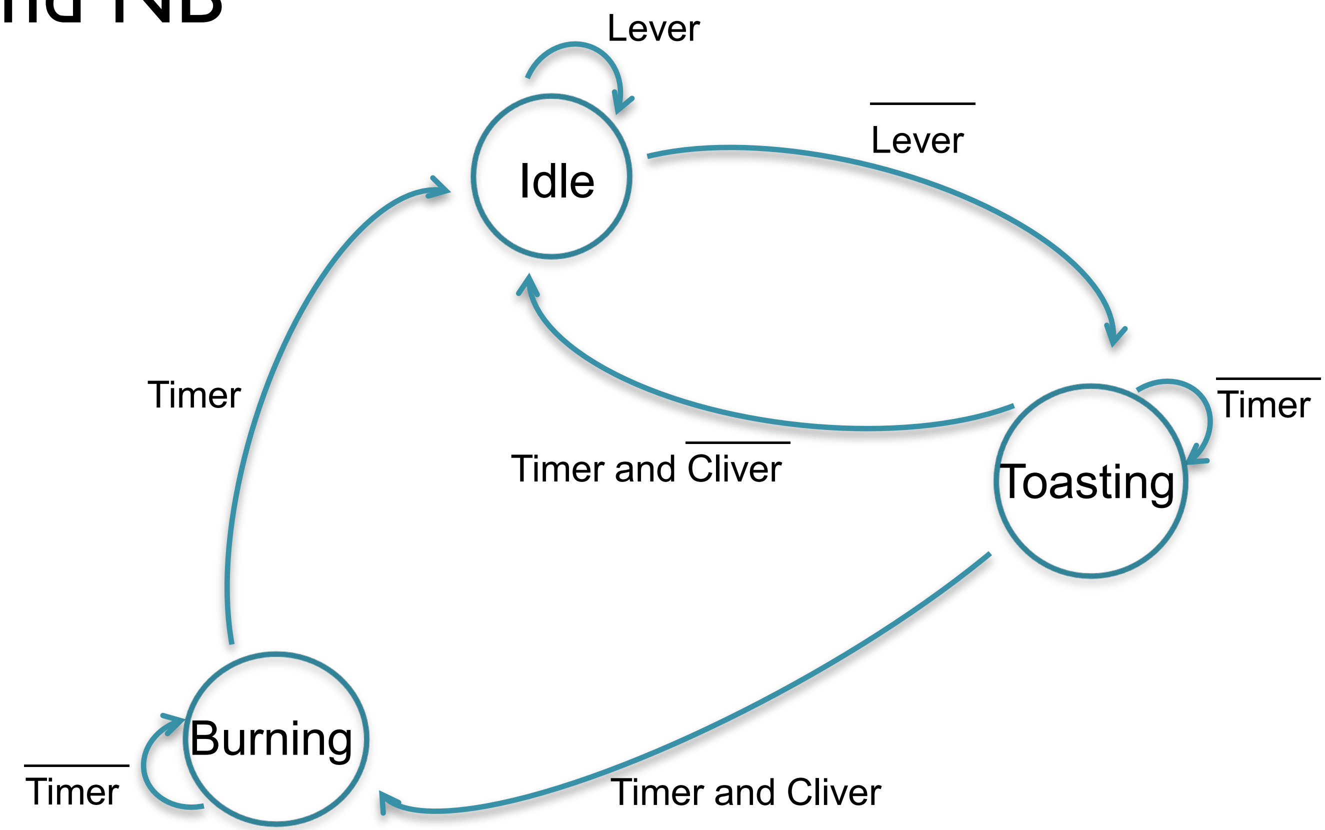
- If NI = 1 the next State will be Idle
- If NT = 1 the next State will be Toasting
- If NB = 1 the next State will be burning



State	Encoding
Idle (I)	001
Toasting (T)	010
Burning (B)	100

Evil Toaster Logic Design

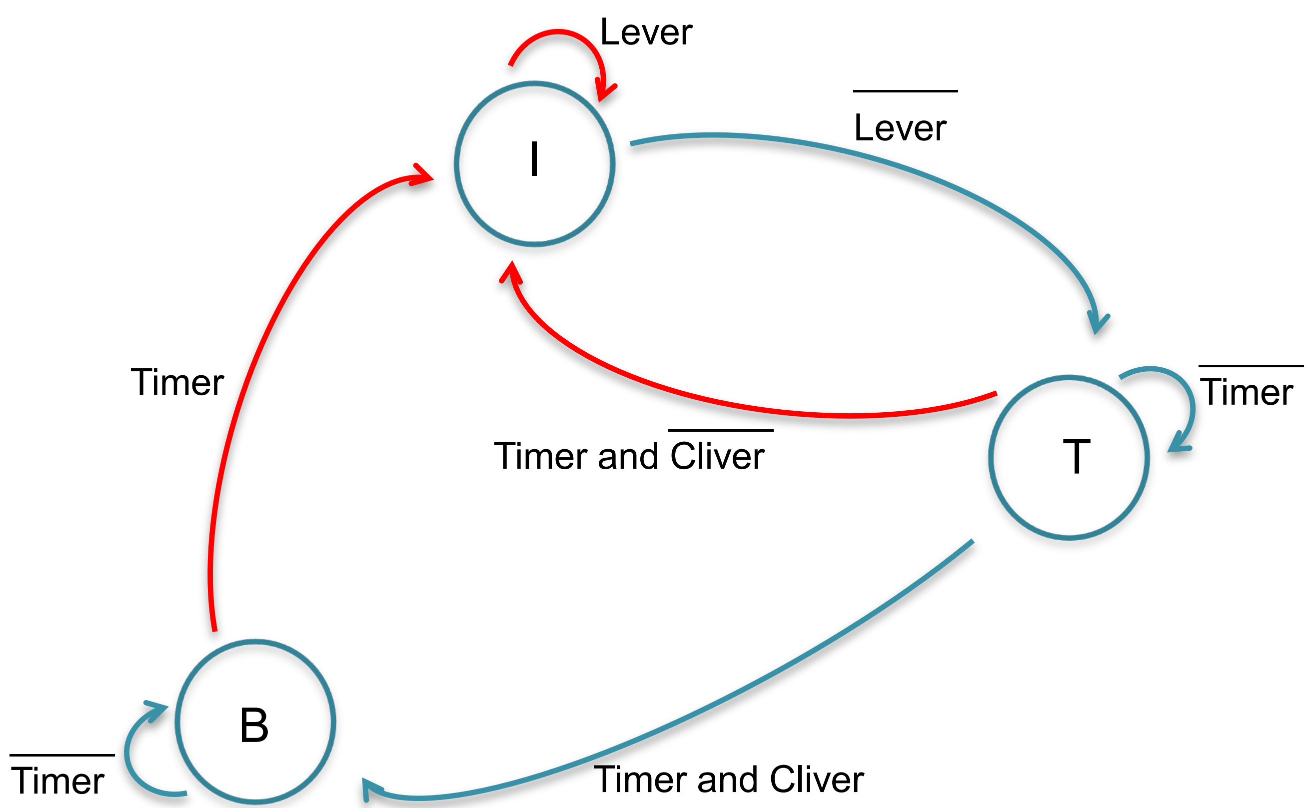
- Use current state and inputs to set up NI, NT and NB



Evil Toaster Logic Design

- Conditions for Idle to be next state

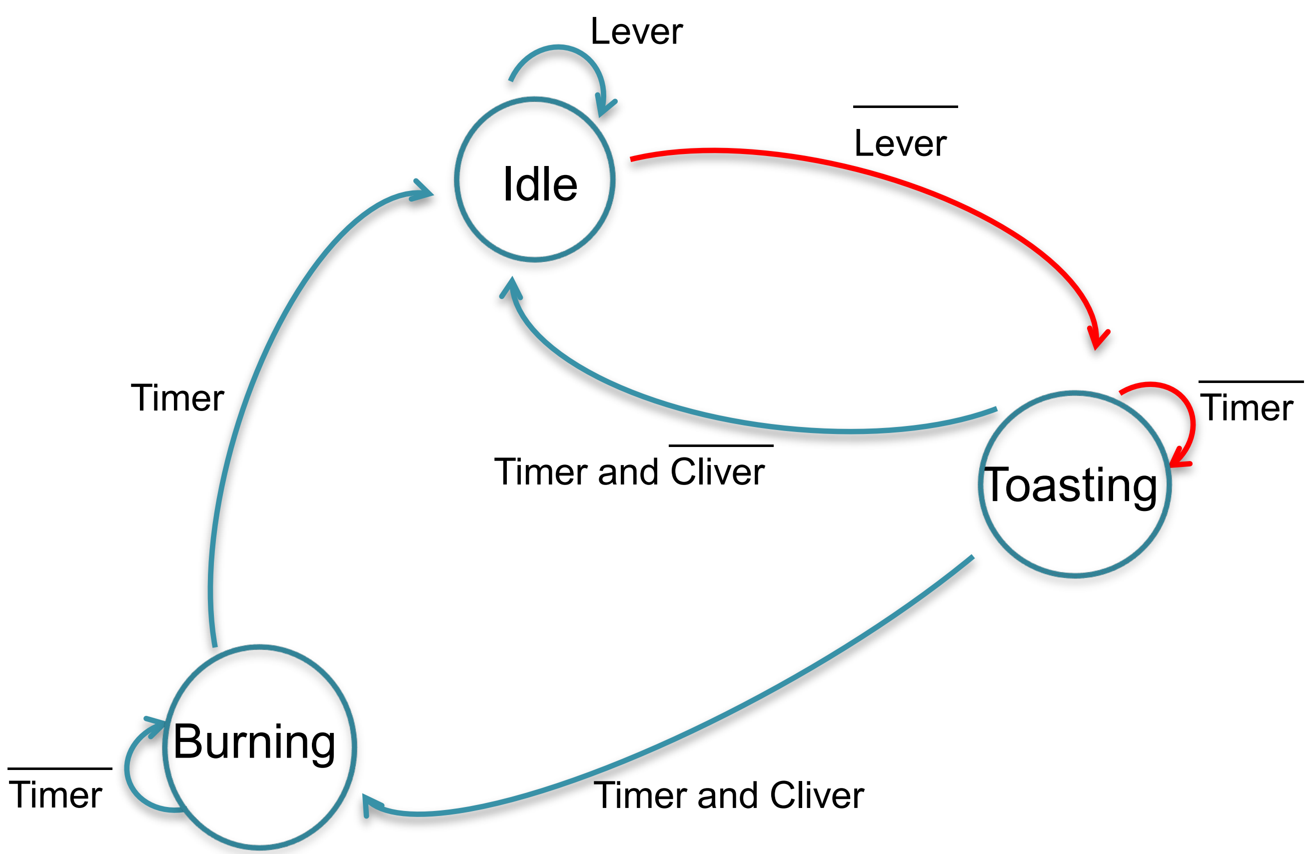
$$NI = (I \cdot Lever) + (T \cdot Timer \cdot \overline{Cliver}) + (B \cdot Timer)$$



Evil Toaster Logic Design

- Conditions for Toasting to be the next state

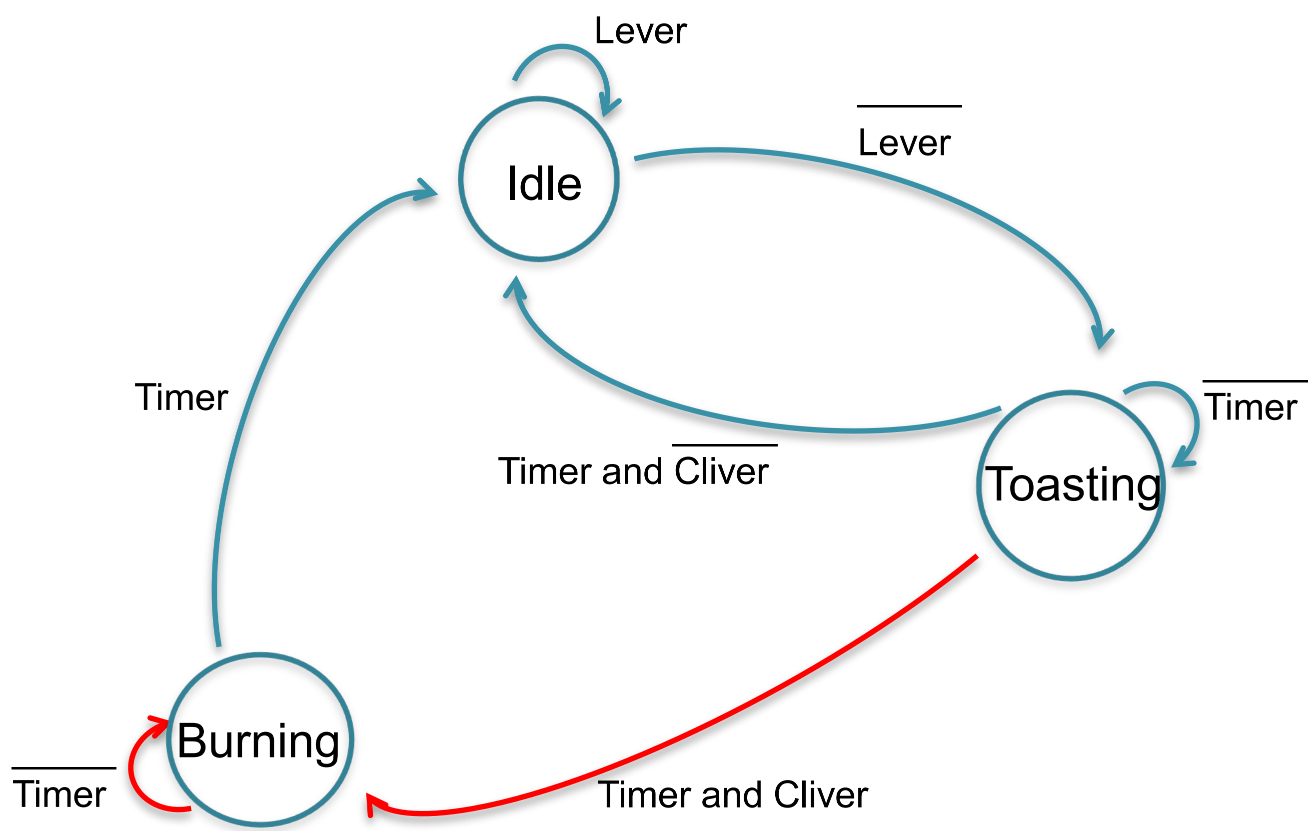
$$\boxed{NT = (I \cdot \overline{Lever}) + (T \cdot \overline{Timer})}$$



Evil Toaster Logic Design

- Conditions for Burning to be the next state

$$NB = (T \cdot Timer \cdot Cliver) + (B \cdot \overline{Timer})$$



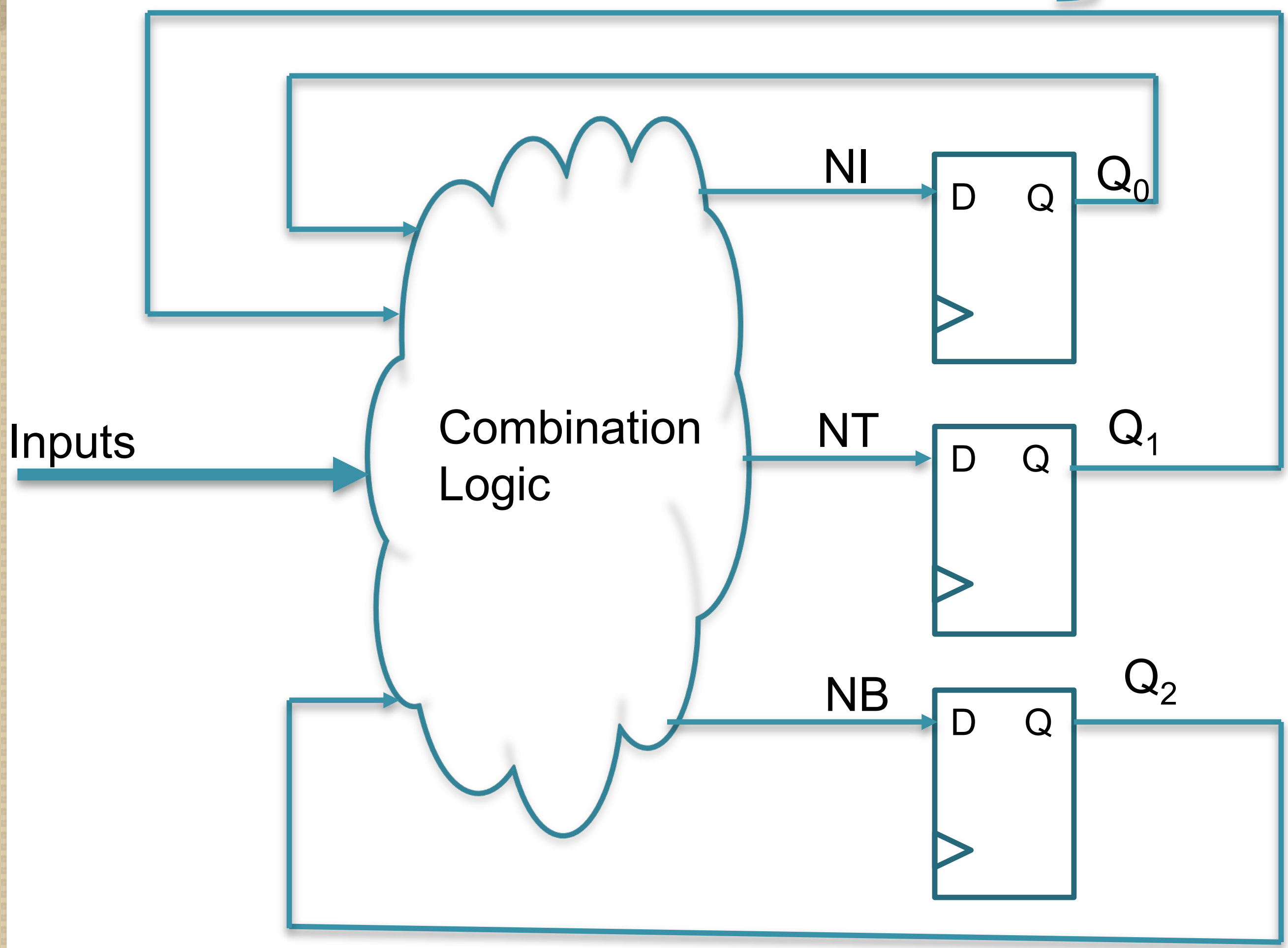
Evil Toaster Logic Design

- $NI = (I \cdot Lever) + (T \cdot Timer \cdot \overline{Cliver}) + (B \cdot Timer)$

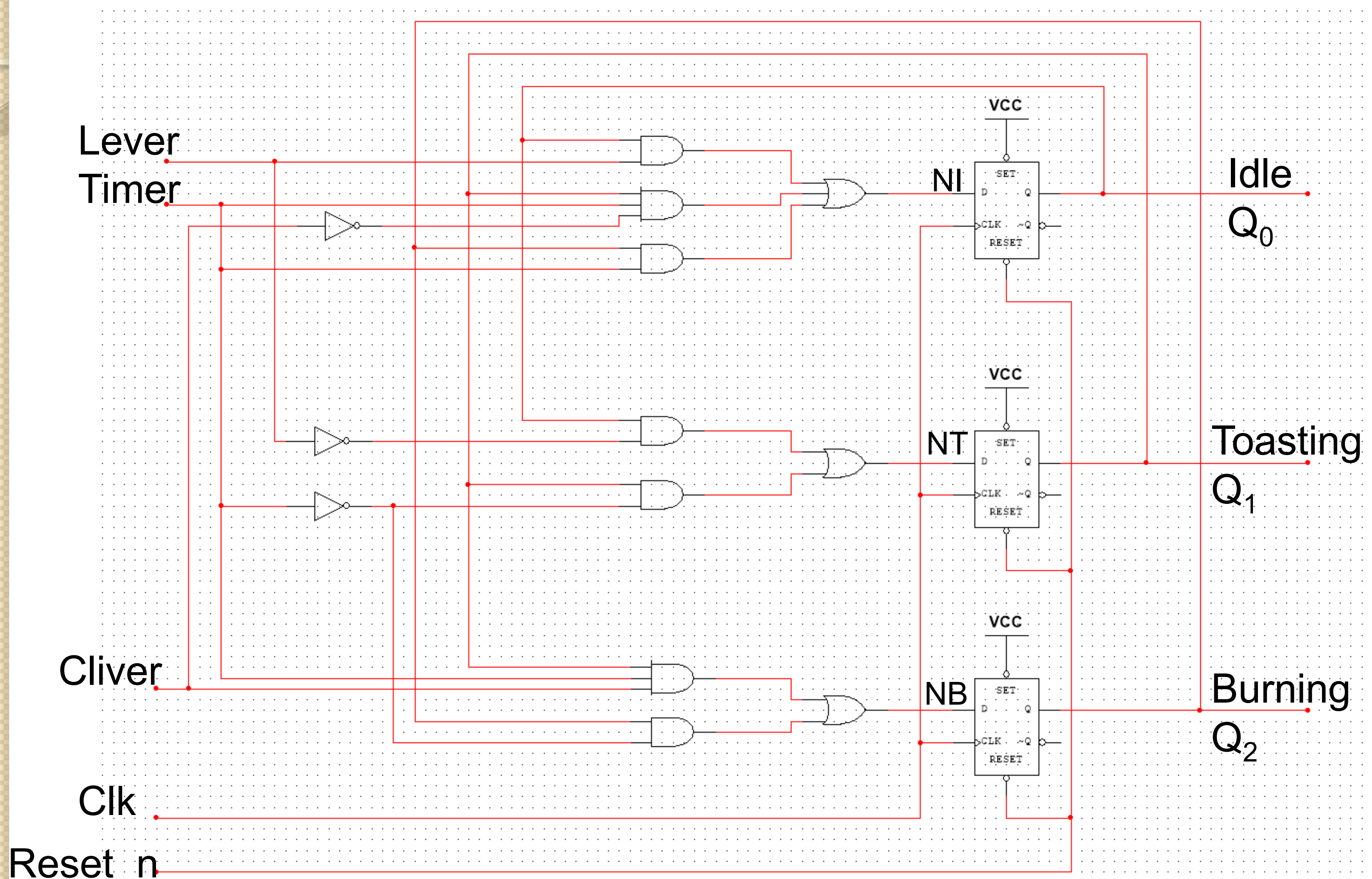
  $NT = (I \cdot \overline{Lever}) + (T \cdot \overline{Timer})$

  $NB = (T \cdot Timer \cdot Cliver) + (B \cdot \overline{Timer})$

The next state equations form The combination logic



Evil Toaster Logic Design



Evil Toaster Logic Design

- Outputs
  - Moore State Machine: outputs dependent on state only

State	Outputs	
	Coil	**Flame Thrower**
Idle (001)	Off	Off
Toasting (010)	On	Off
Burning (100)	Off	On

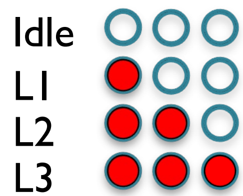
  - Coil = Q_1
  - Flame Thrower = Q_2

Mustang Blinker example

- Consider the turn signal on the Mustang
  - It follows the following pattern:

Mustang Blinker

- Four states



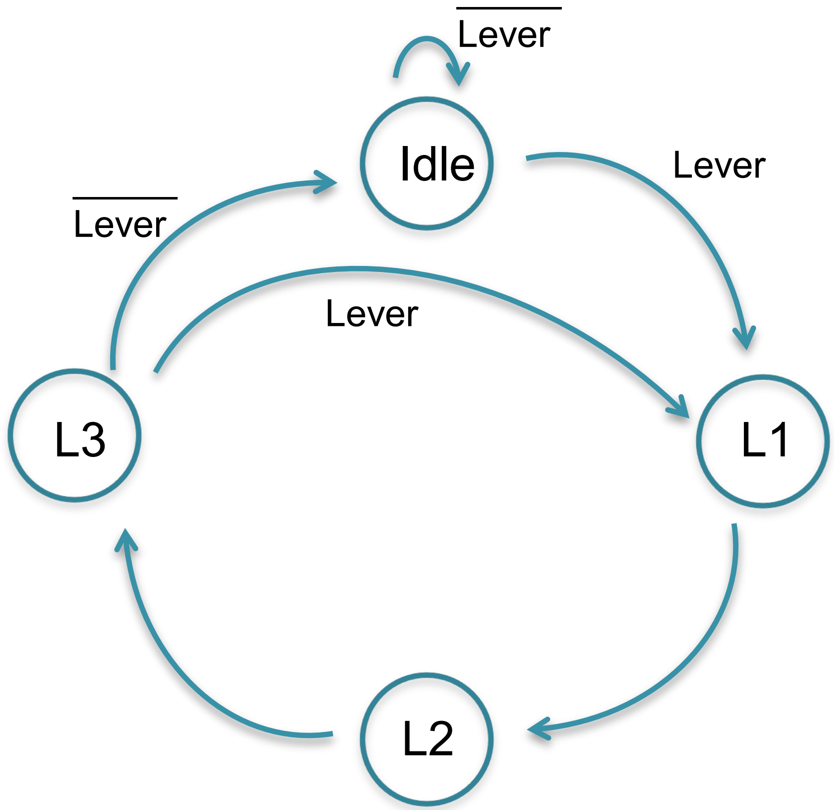
- Inputs
  - Lever (up = 1, neutral = 0)
- Outputs
  - Bulb1 (on = 1, off = 0)
  - Bulb2 (on = 1, off = 0)
  - Bulb3 (on = 1, off = 0)

Mustang Blinker

- All three bulbs are off when not in use
- When the driver raises the directional lever the following happens
  - The first bulb goes on
  - The first bulb stays on and the second bulb goes on
  - The first and second bulb stay on and the third bulb goes on
- After the third bulb goes on
  - If the lever is still up, the sequence above repeats
  - If the lever is in neutral position, all lights go off
- If the lever is returned to the neutral position prior to all 3 bulbs being on, it is ignored

Mustang Blinker

- State Transition Diagram



Mustang Blinker

- Outputs

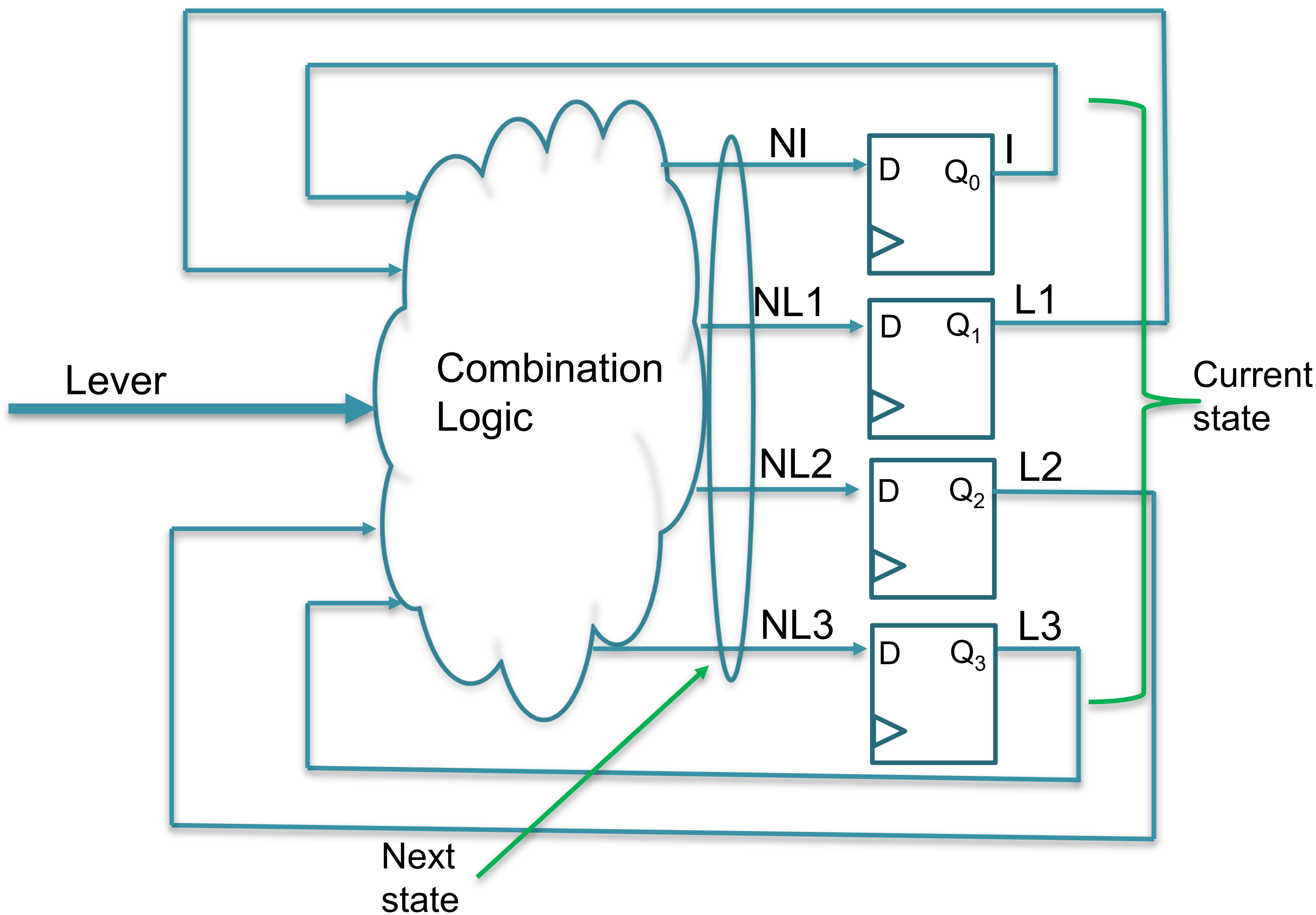
	Outputs		
State	**Bulb1**	**Bulb2**	**Bulb3**
Idle	Off	Off	Off
L1	On	Off	Off
L2	On	On	Off
L3	On	On	On

Mustang Blinker

- Encoding
  - Use one-hot encoding
  - 4 flip flops for 4 states
  - Only 1 flip flop active high at a time

State	Encoding $Q_3Q_2Q_1Q_0$
Idle	0001
L1	0010
L2	0100
L3	1000

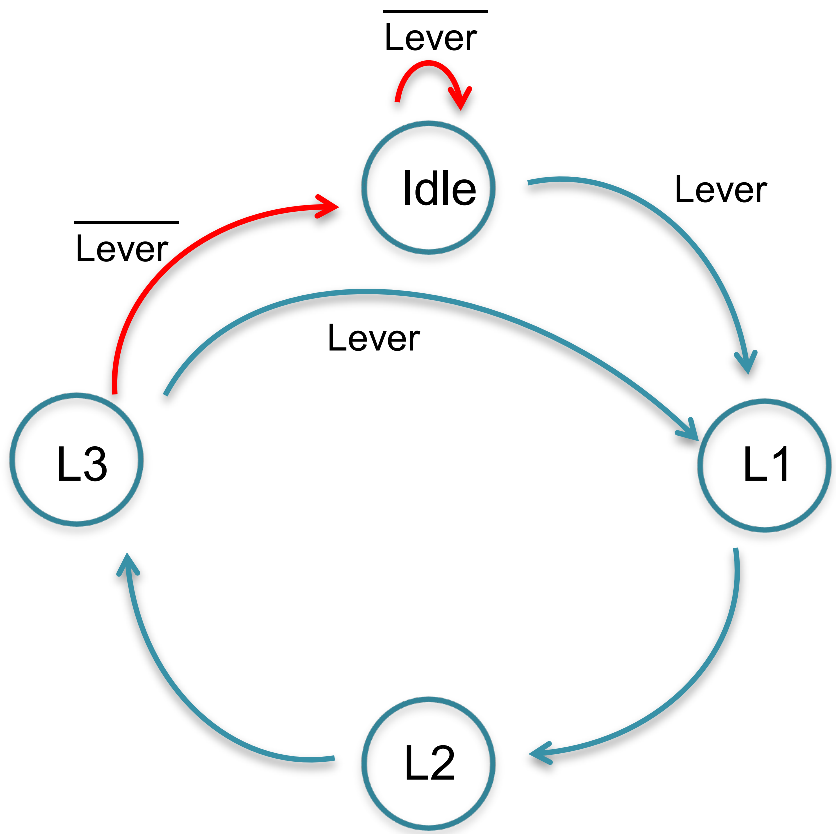
Mustang Blinker Logic Design



Mustang Blinker Logic Design

- Set up NI

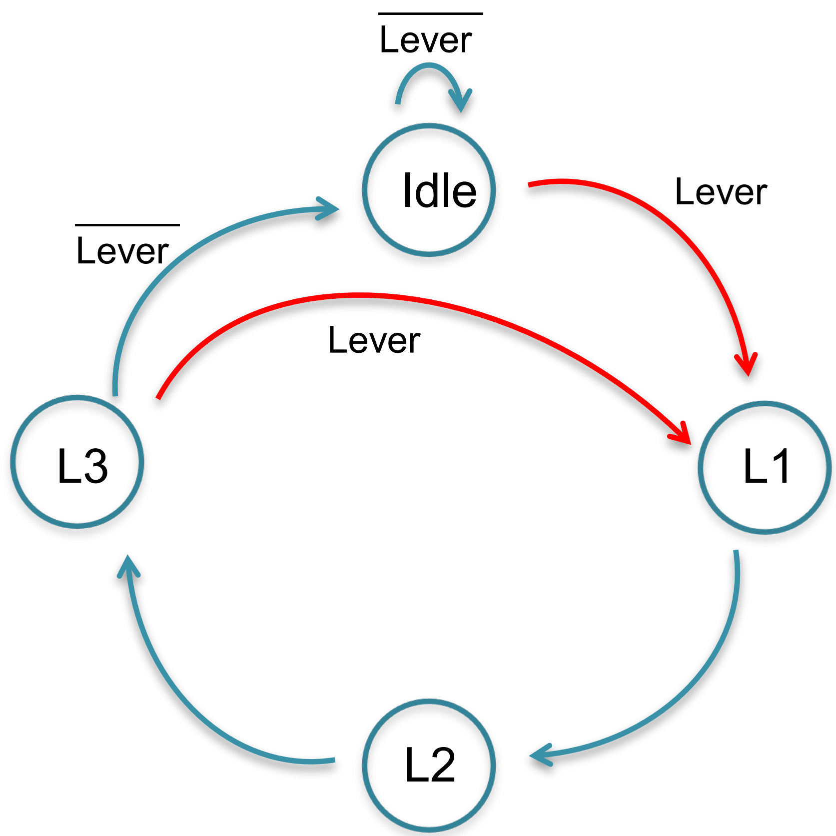
$$NI = (I \cdot \overline{Lever}) + (L3 \cdot \overline{Lever})$$



Mustang Blinker Logic Design

- Set up NL1

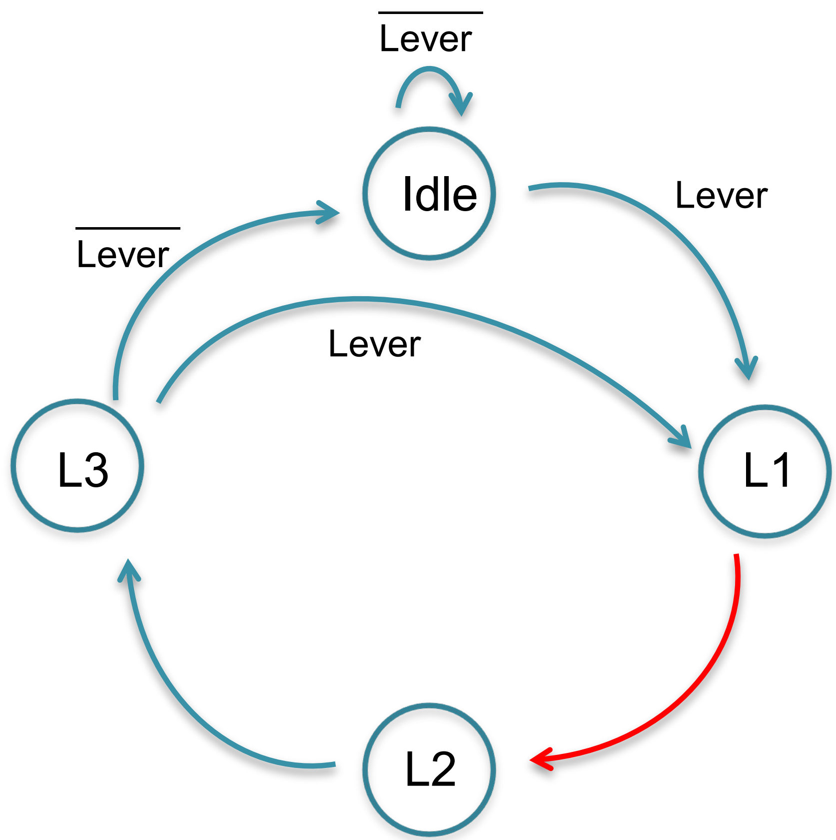
$$\boxed{NL1 = (I \cdot Lever) + (L3 \cdot Lever)}$$



Mustang Blinker Logic Design

- Set up NL2

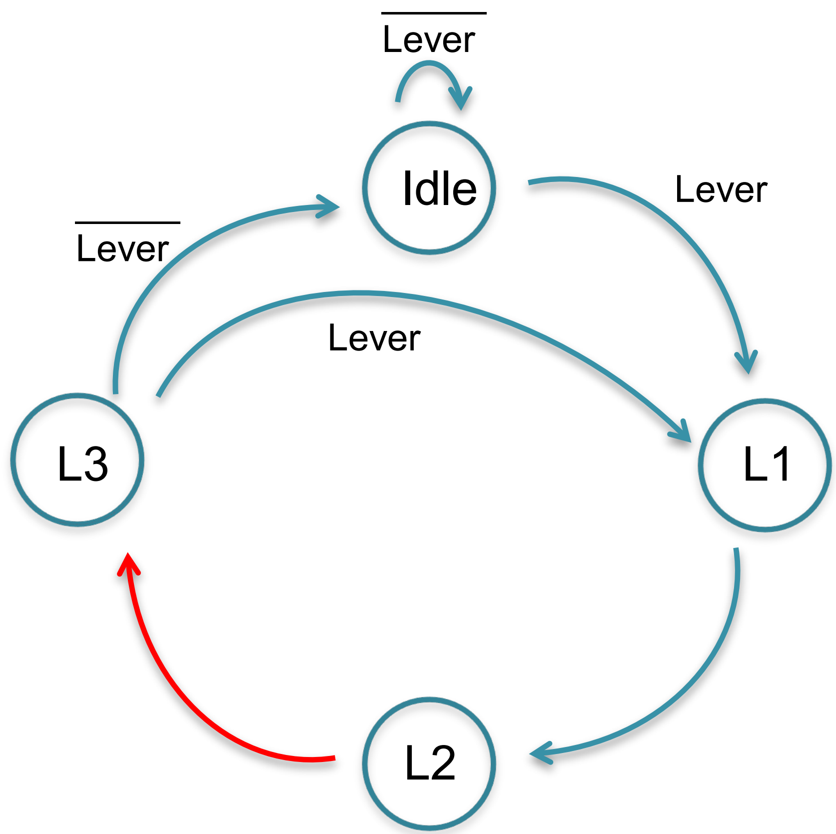
$$\boxed{NL2 = L1}$$



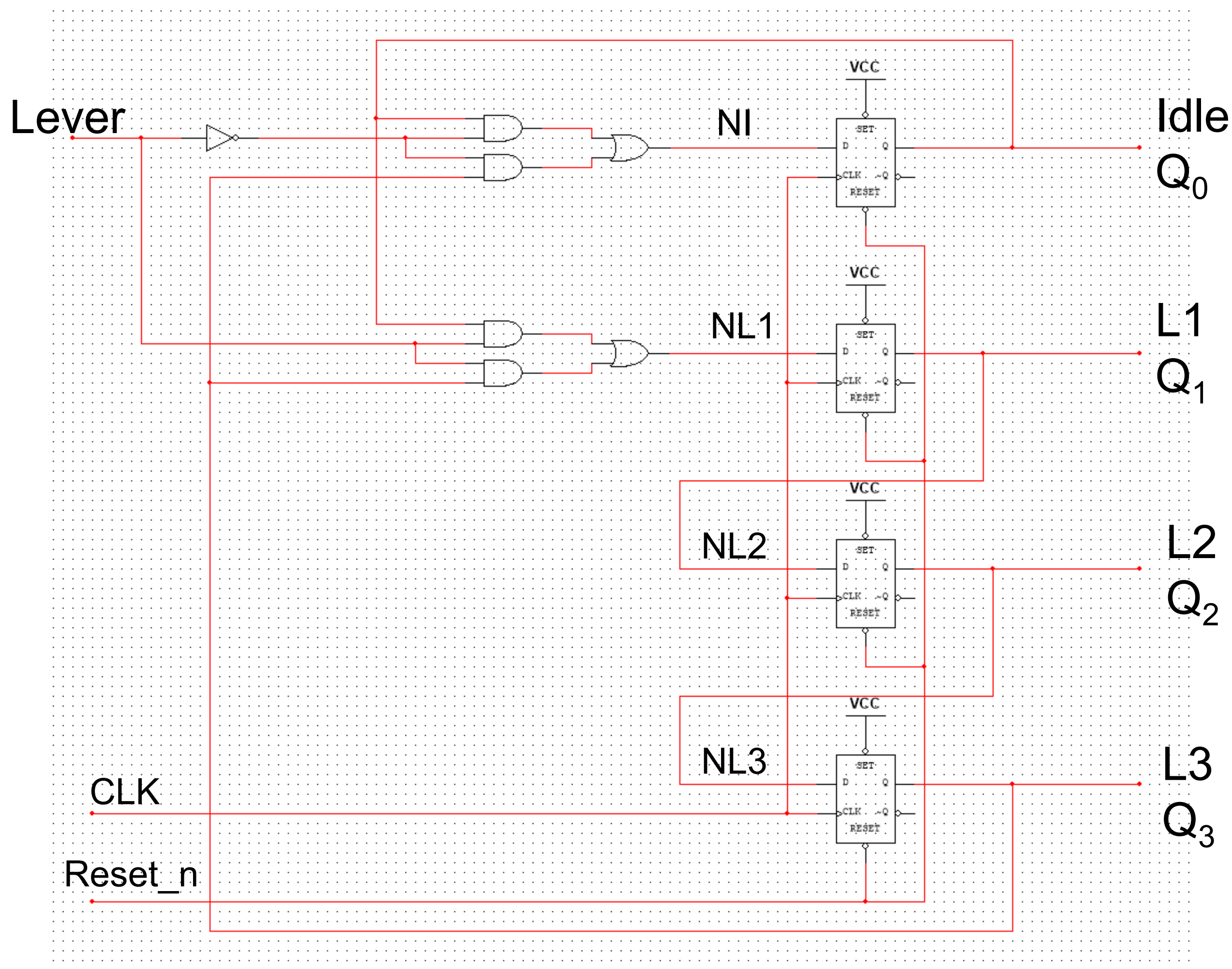
Mustang Blinker Logic Design

- Set up NL3

$$\boxed{NL3 = L2}$$



Mustang Blinker Logic Design



Mustang Blinker Logic Design

- Output Logic

State	Encoding $Q_3Q_2Q_1$ Q_0	Bulb1	Bulb2	Bulb3
Idle	0001	Off	Off	Off
L1	0010	On	Off	Off
L2	0100	On	On	Off
L3	1000	On	On	On

- $Bulb1 = \ Q_1 + Q_2 + Q_3$
- $Bulb2 = \ Q_2 + Q_3$
- $Bulb3 = \ Q_3$

Mustang Blinker Logic Design

- Adding outputs