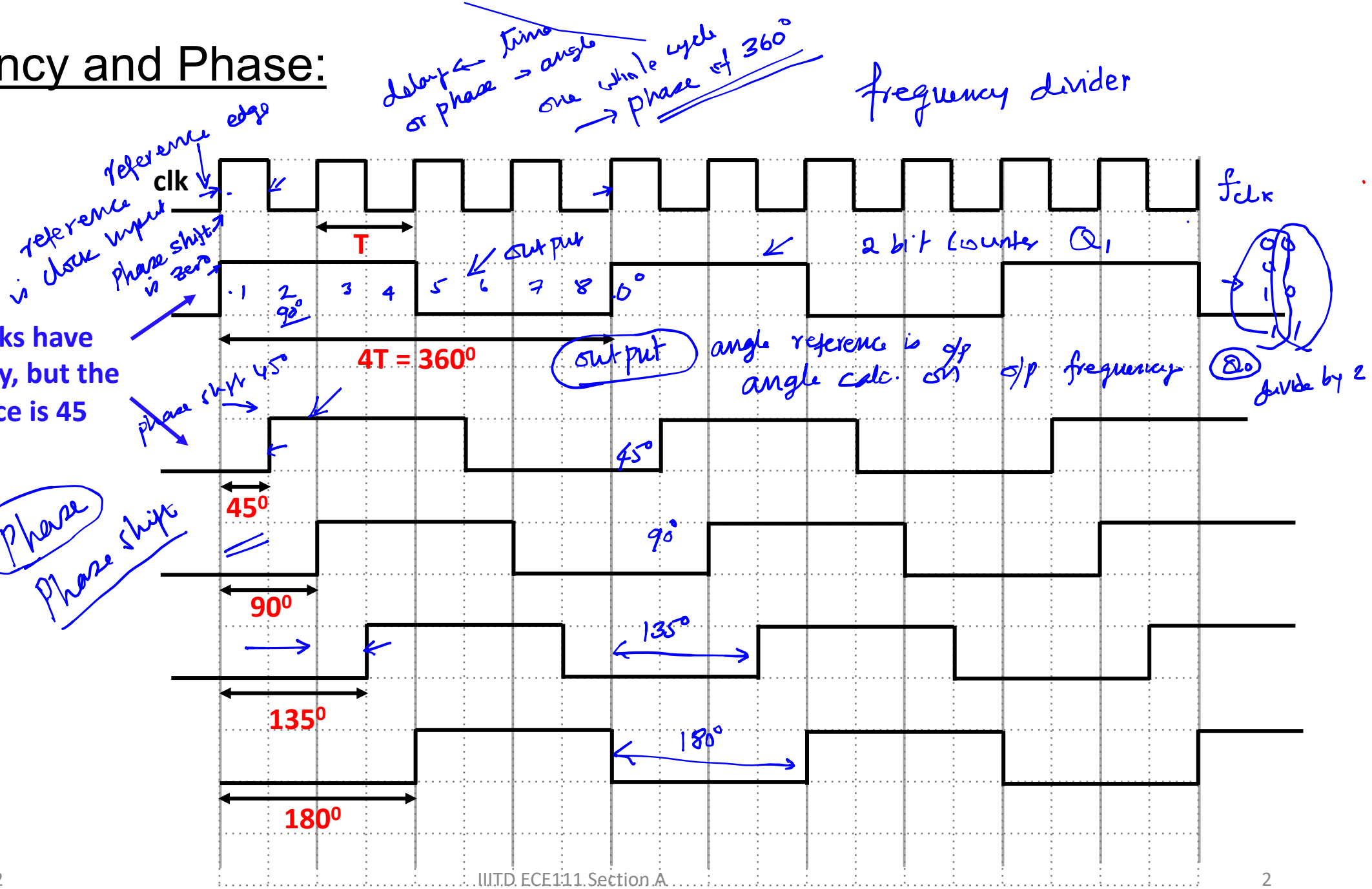
Frequency and Phase:



These two clocks have same frequency, but the phase difference is 45 degree.

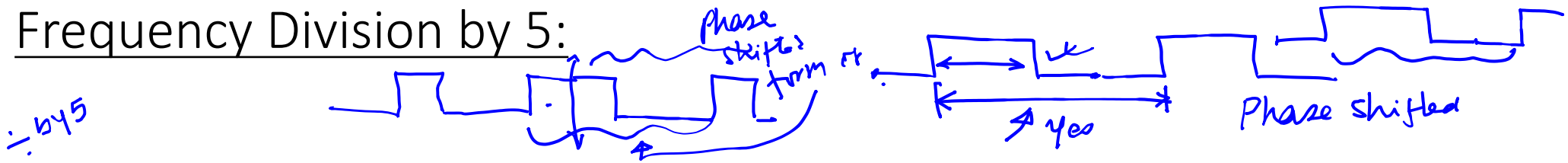
$T_{out} = 4\,T_{clk}$

$f_{out} = \frac{1}{4} f_{clk}$

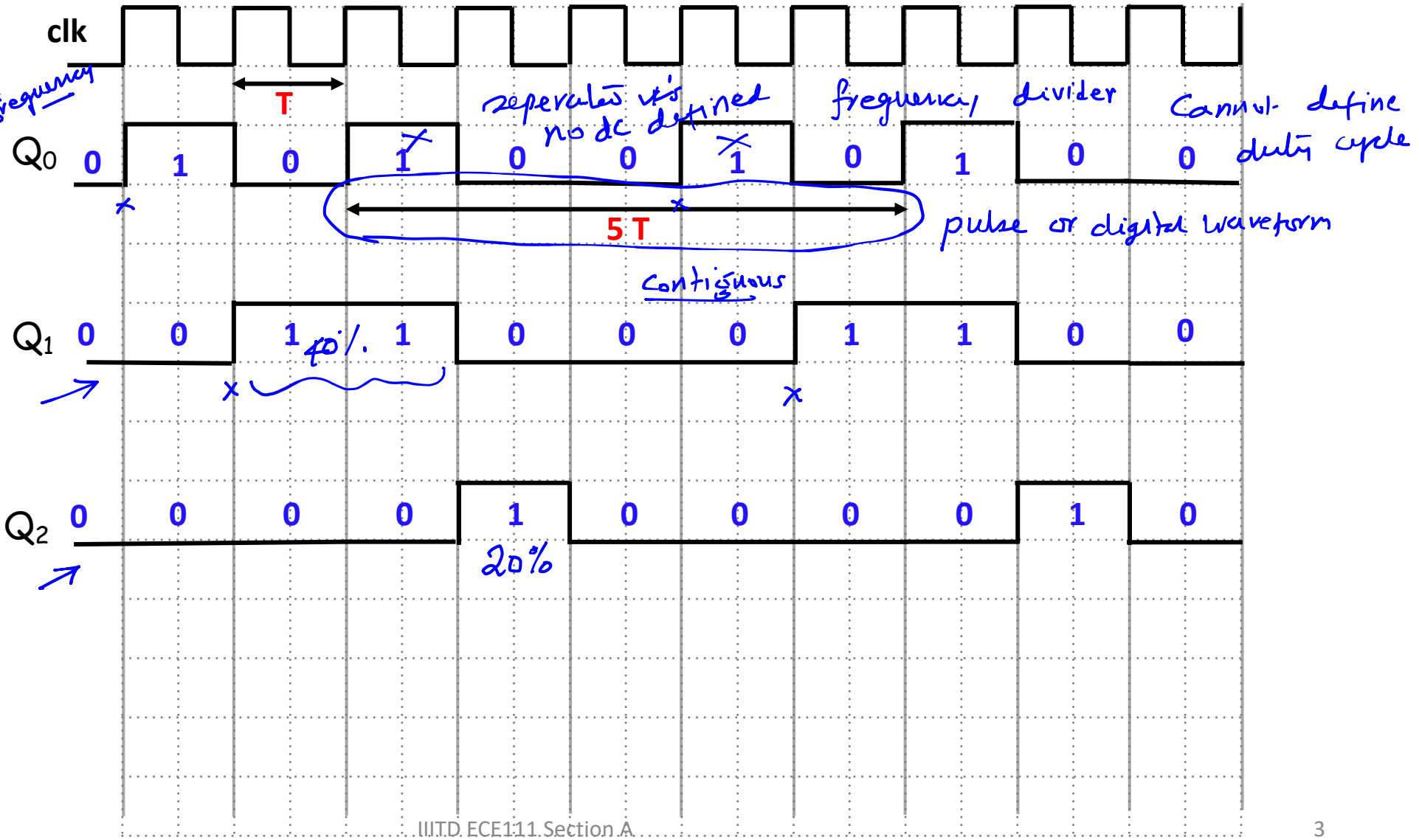
Phase

Phase shift

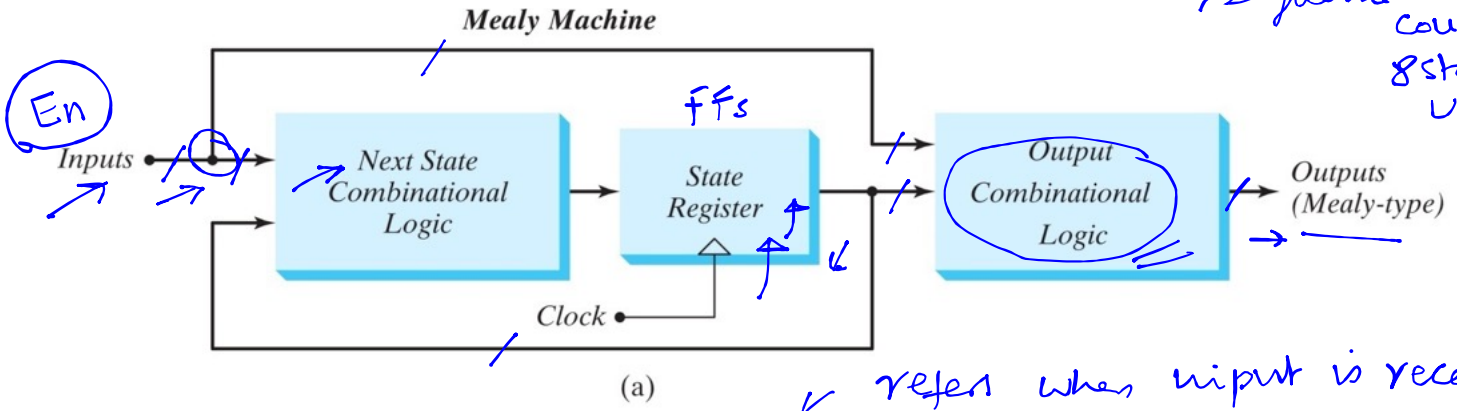
Frequency Division by 5:



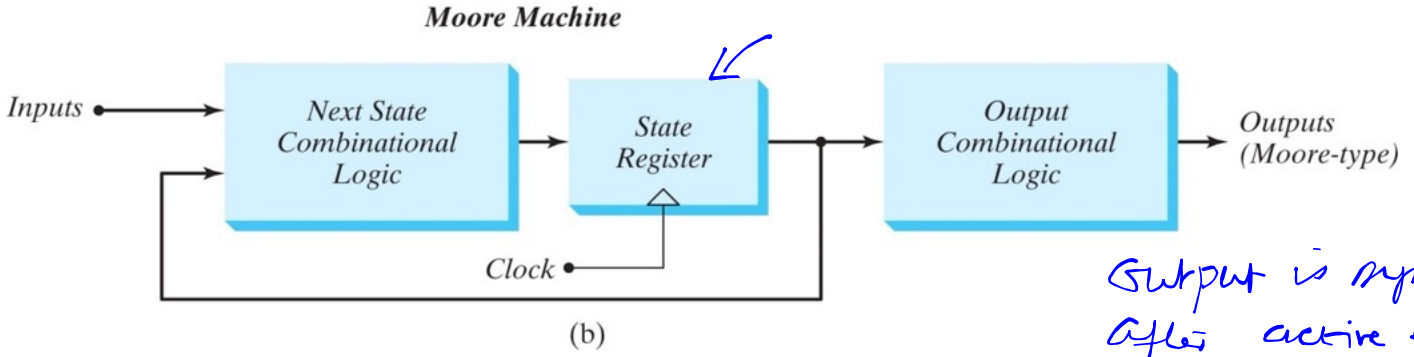
Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0



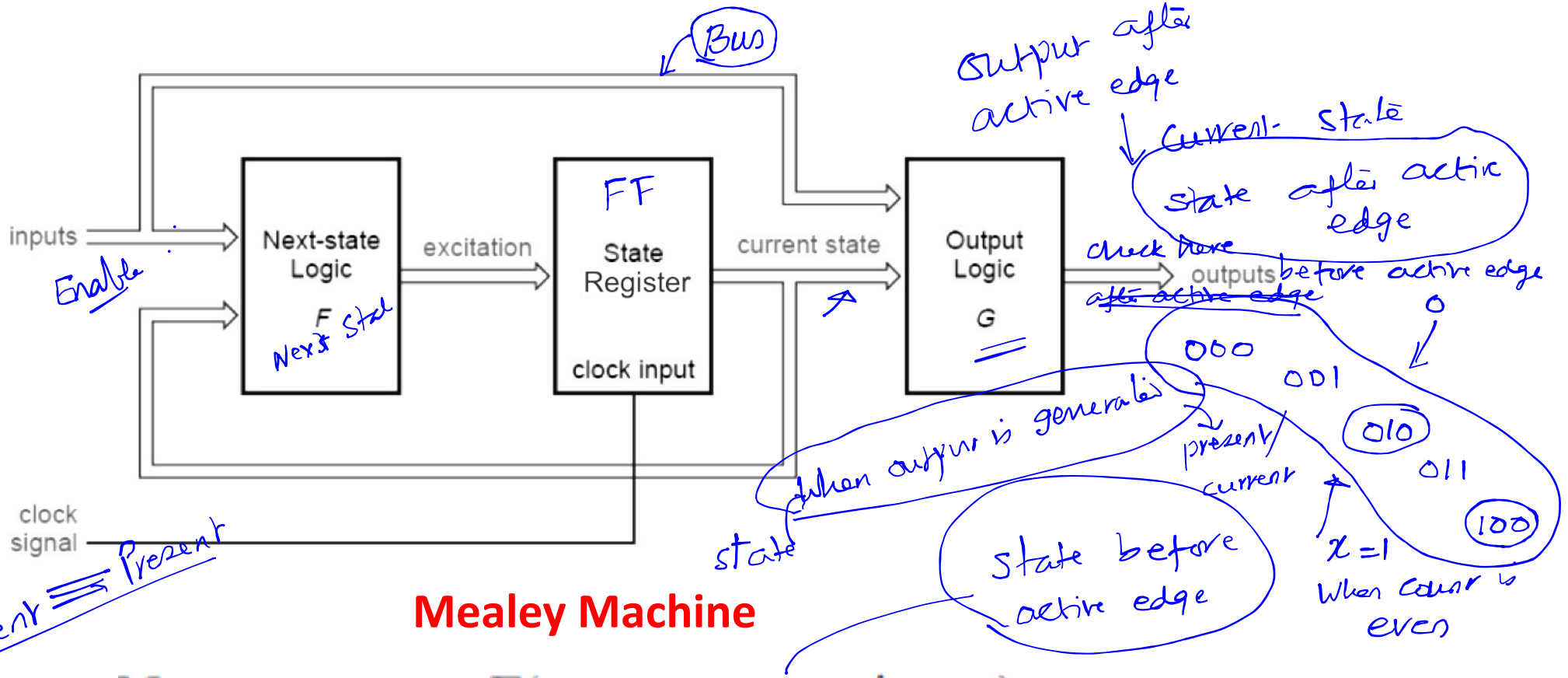
Finite State Machine:



The output depends on the present state and present input, the input that would cause a change in state at the next active clock edge.



The output depends on the present state that has been reached after an active edge, independent of the input that caused the change in state.



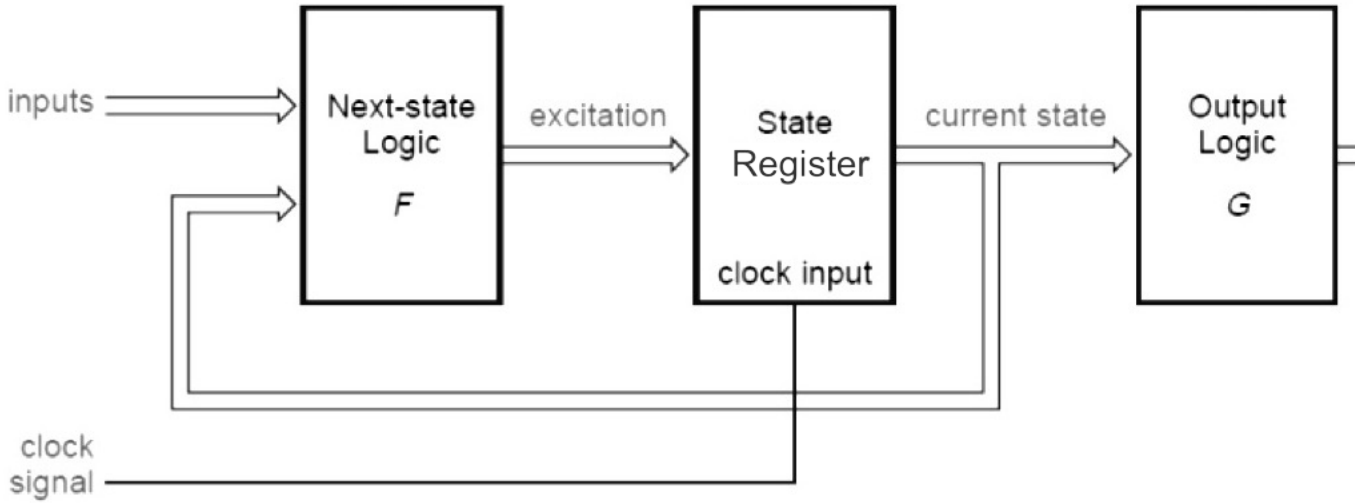
Mealey Machine

$$\text{Next state} = F(\text{current state}, \text{input})$$

$$\text{Output} = G(\text{current state}, \text{input})$$

pre-active edge reference

The output change is not synchronized with the clock.

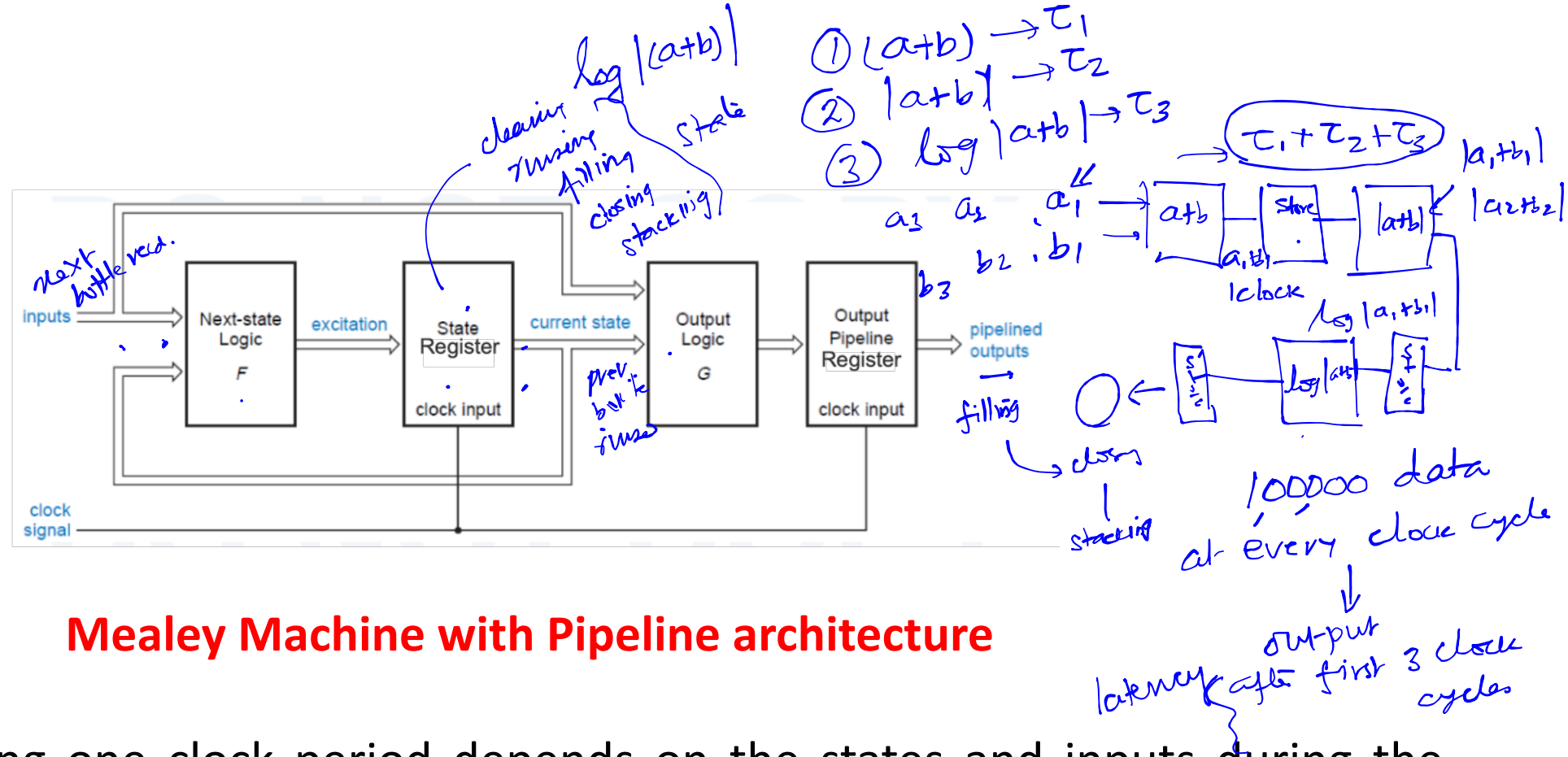


Moore Machine

$$\text{Next state} = F(\text{current state}, \text{input})$$

$$\text{Output} = G(\text{current state})$$

The output change is synchronized with the clock.



Mealey Machine with Pipeline architecture

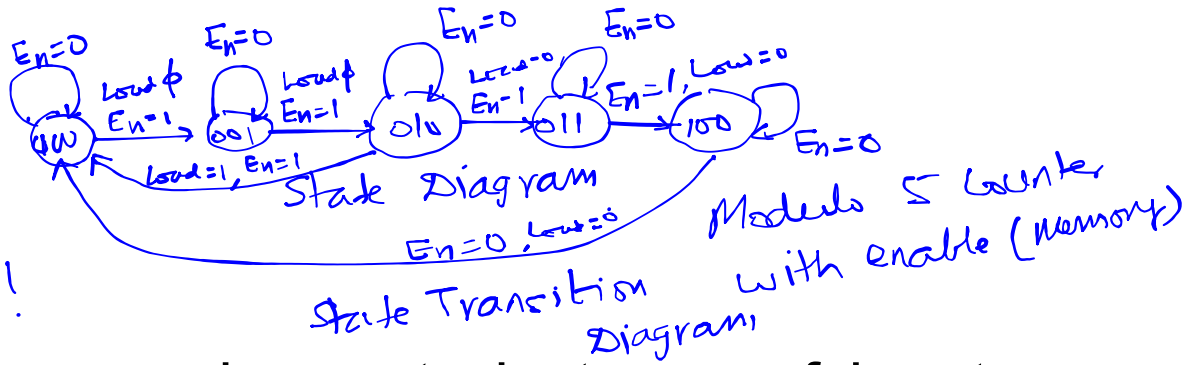
Output during one clock period depends on the states and inputs during the previous clock period

Characteristics Equation of Various Flip Flops:

Useful table when you design sequential circuits

Device Type	Characteristic Equation
S-R Latch	$Q_n = S + \bar{R}Q_{n-1}$
D Latch	$Q_n = D$
Edge-Triggered D flip-flop	$Q_n = D$
D flip-flop with enable	$Q_n = En \cdot D + Q_{n-1}$
Master Slave S-R flip-flop	$Q_n = S + \bar{R}Q_{n-1}$
Master Slave J-K flip-flop	$Q_n = J \cdot \overline{Q_{n-1}} + \bar{K}Q_{n-1}$
Edge Triggered J-K flip-flop	$Q_n = J \cdot \overline{Q_{n-1}} + \bar{K}Q_{n-1}$
T flip-flop	$Q_n = \overline{Q_{n-1}}$
T flip-flop with enable	$Q_n = En \cdot \overline{Q_{n-1}} + \overline{En}Q_{n-1}$

Finite State Machine (FSM):



1. For a given problem, understand the requirements in terms of inputs, outputs and number of states.
2. Draw the FSM or state diagram. ← STD
3. Map FSM to state table for a given type of FF. State Transition Table STT
4. From state table, find out the expressions for FF inputs and FSM outputs in terms of present state (FF outputs) and present external inputs. Use K-maps.
5. Implement the circuit and verify the operation using timing diagram. ← essential

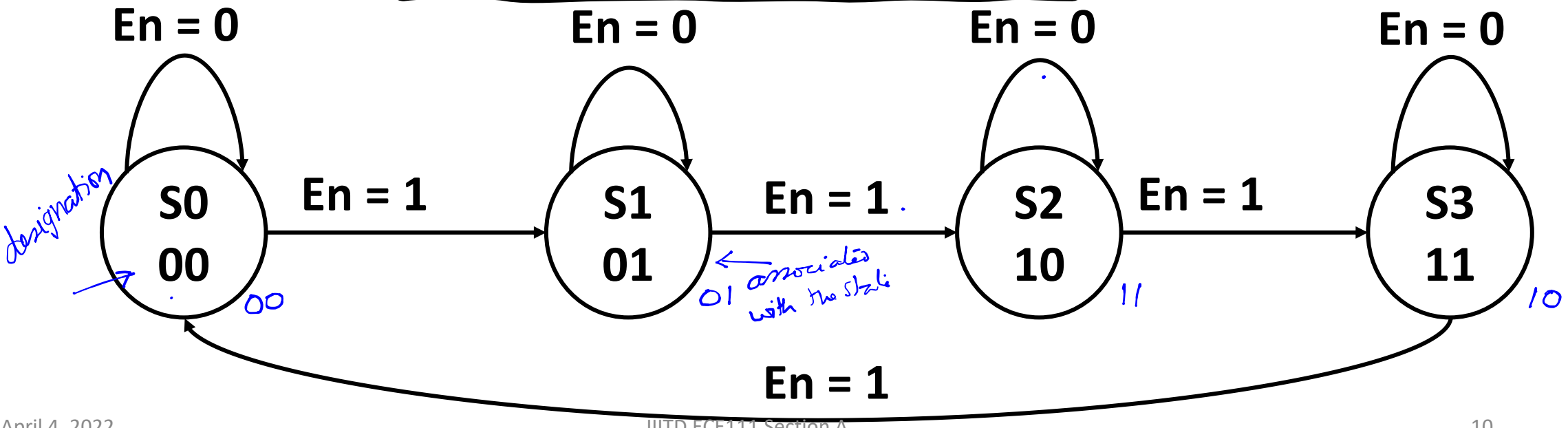
Mod-4 Binary UP Counter giving Gray Code output with Memory (Moore):

- Number of states: 4
- Number of FFs: 2 (lets do it using T-FFs)
- Number of inputs: Clock (by default) and enable (En)
- Number of outputs: Counter value (2 bits)

Output Gray code a function of state value

Binary code	Gray code
00	00
01	01
10	11
11	10

State Transition Graph (STG) or STD



Mod-4 Binary UP Counter giving Gray Code output with Memory (Moore):

State Transition Table (STT)

Present State		Input	Next State		Outputs		FF Inputs	
P_1	P_0	En	N_1	N_0	Z_1	Z_0	T_1	T_0
0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	1
0	1	0	0	1	0	1	0	0
0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	0	0
1	0	1	1	1	1	0	0	1
1	1	0	1	1	1	0	0	0
1	1	1	0	0	0	0	1	1

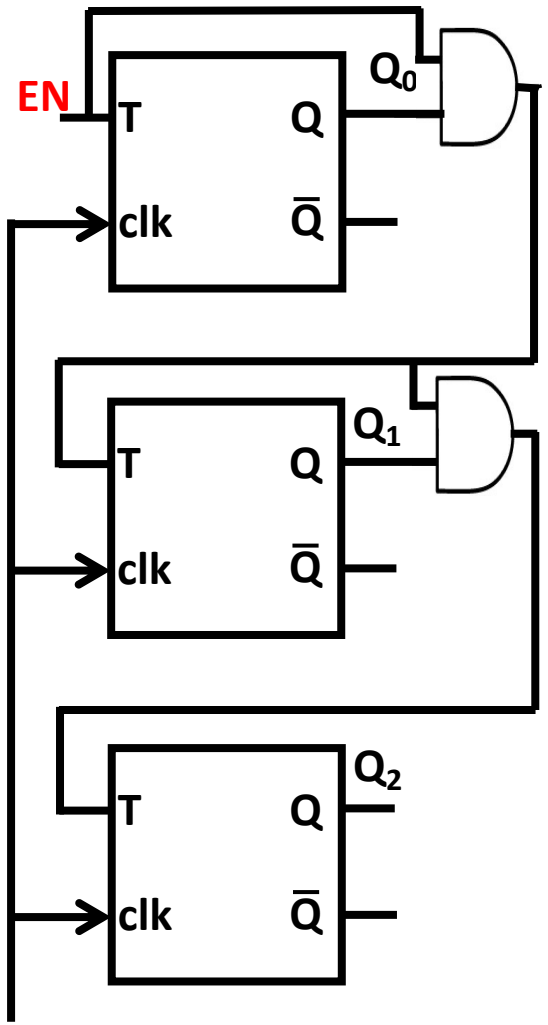
$$Z_1 = N_1 \qquad Z_0 = N_0 \oplus N_1 \qquad T_1 = P_0 \cdot En \qquad T_0 = En$$

The output changes when the state changes on the active clock edge.

The output is available for one clock cycle.

The hold time for En, in this case, is the hold time of the FF

Synchronous Counter:

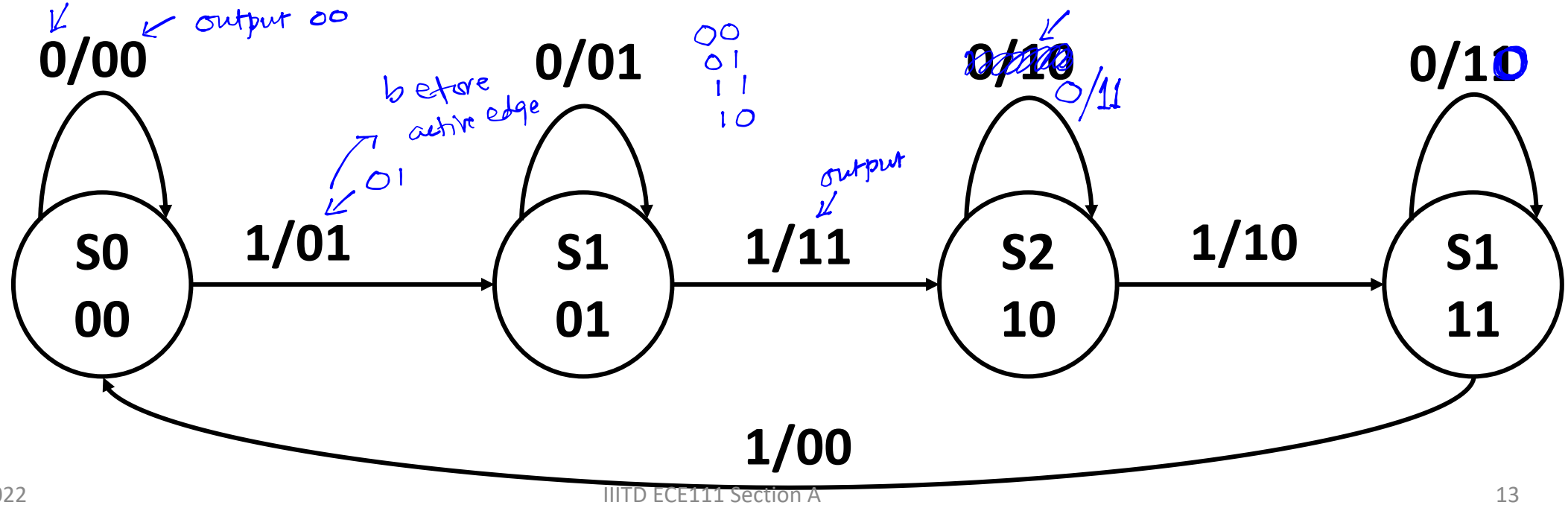


Draw the State Transition Graph for the counter shown. (HW)

and
STT also

Mod-4 Binary UP Counter giving Gray Code output with Memory (Mealy):

- Number of states: 4
- Number of FFs: 2 (lets do it using T-FFs)
- Number of inputs: Clock (by default) and enable (En)
- Number of outputs: Counter value (2 bits)



Mod-4 Binary UP Counter giving Gray Code output with Memory (Mealy):

State Transition Table (STT)

	Present State		Input	Clock	Next State		Output		FF Inputs	
	P_1	P_2	En	clk	N_1	N_0	Z_1	Z_0	T_1	T_0
	0	0	0	ϕ	0	0	0	0	0	0
→ *	0	0	1	↓, –	0	0	0	1	0	1
	0	0	1	↑	0	1	0	1	0	1
	0	1	0	ϕ	0	1	0	1	0	0
→ *	0	1	1	–	0	1	1	1	1	0
	0	1	1	↑	1	0	1	1	1	0
	1	0	0	ϕ	1	0	1	1	0	0
→ *	1	0	1	–	1	0	1	0	0	1
	1	0	1	↑	1	1	1	0	0	1
	1	1	0	ϕ	1	1	1	0	0	0
→ *	1	1	1	–	1	1	0	0	1	1
	1	1	1	↑	0	0	0	0	1	1

$Z_1 = En \cdot (N_1 \oplus N_2)$ $Z_1 = En \cdot \overline{N_1}$ $T_1 = P_0 \cdot En$ $T2 = En$

- The output changes when the enable becomes 1 before the setup time of the FF.
- The output is latched to a register at the immediately following active clock edge.
- The output is available for one clock cycle.
- The value of En should be stable for a time t_s given by $max\{t_{set-up}, t_p(max)\} + t_{hold}$.

Where $t_p(max)$ is the maximum combinational delay of the output function.

Mod-4 UP Counter with Memory

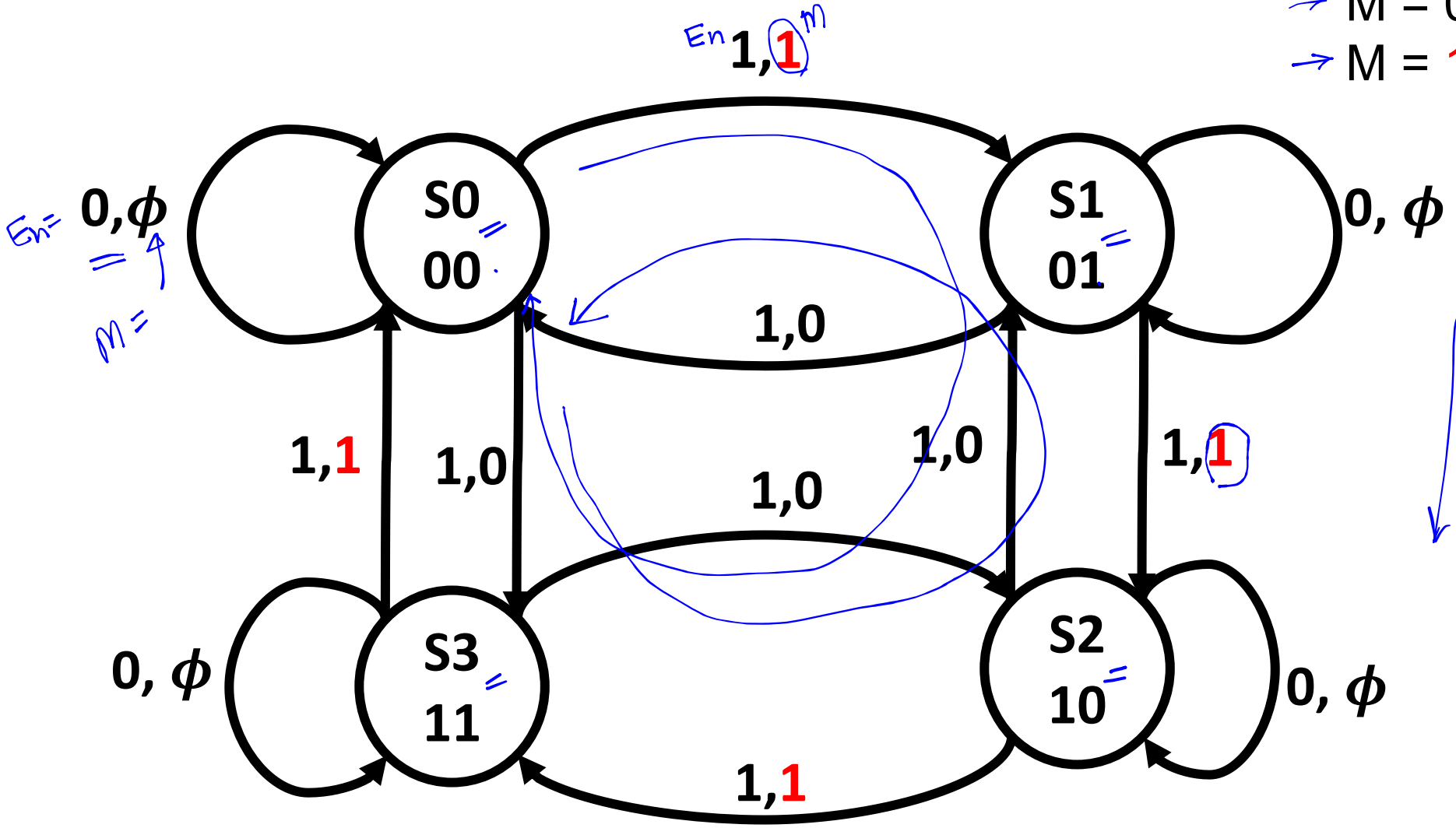
- Repeat the same process for D-FF
- Repeat the same process for JK-FF

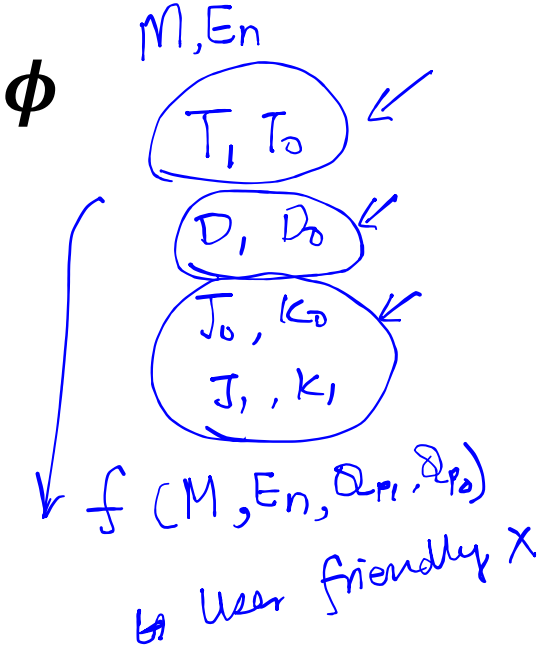
Mod-4 UP/Down Counter with Memory (Moore):

Inputs En and M

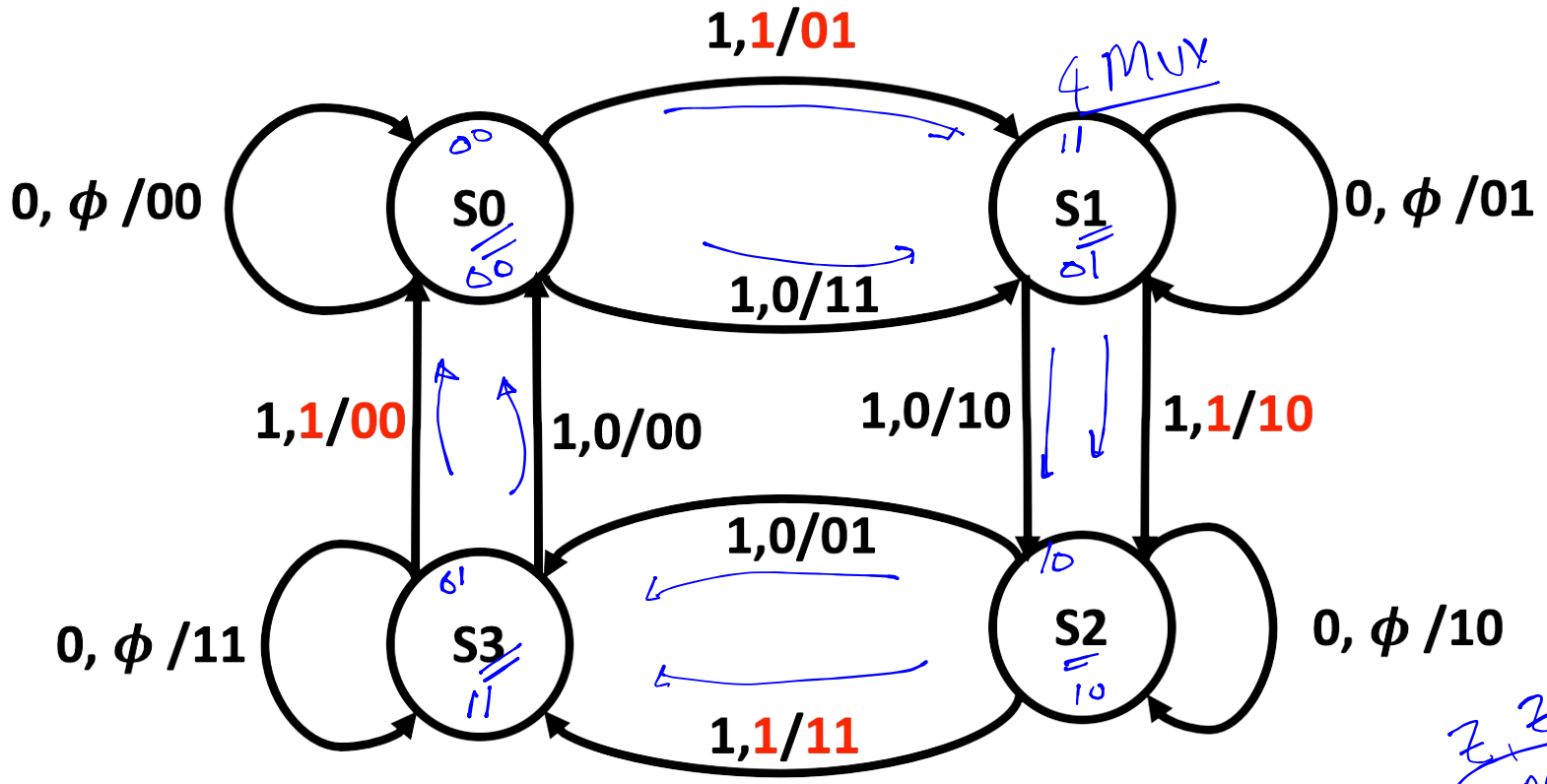
→ M = 0 ---- Down Count

→ M = 1 ---- Up Count





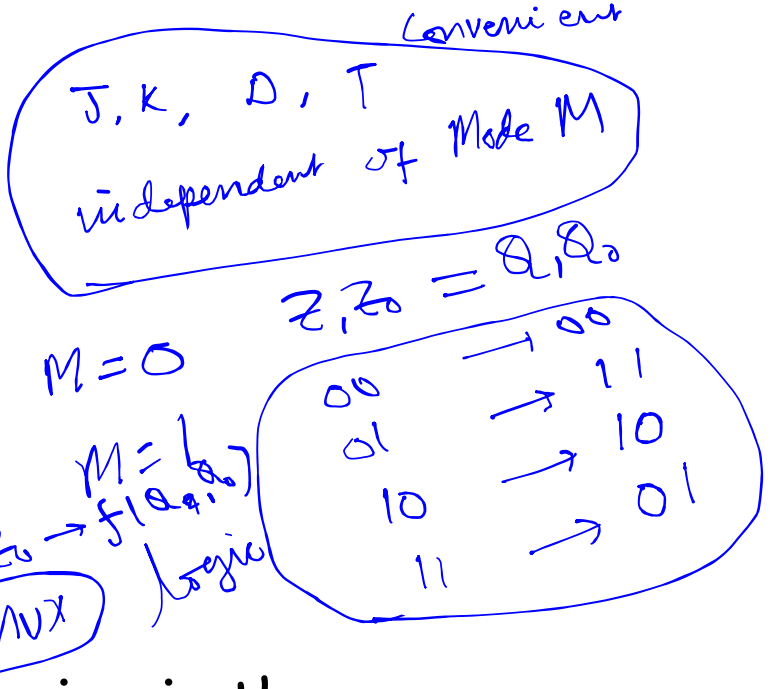
Mod-4 UP/Down Counter with Memory (Mealy):



Inputs En and M

M = 0 ---- Down Count

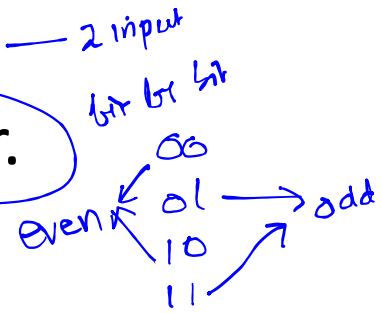
M = 1 ---- Up Count



The advantage of this approach is that the counter design is the same for UP or Down counter but the output changes for the UP or Down Counter.

Homework

- Design an odd-even detector.
- Input: 2 bits
- Output: 2 bits
- When input contains odd number of 1, output should be 01. Else, output should be 10.



(include no 1's also)

Invalid States:

Design counter using T-FF with states $000 \rightarrow 001 \rightarrow 011 \rightarrow 010 \rightarrow 110 \rightarrow 000$

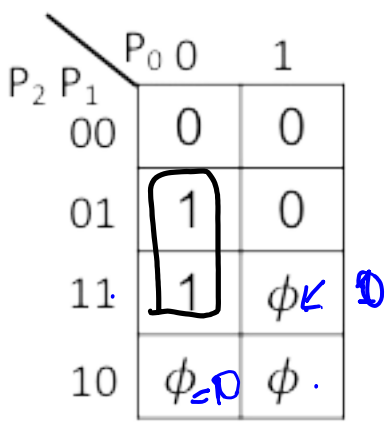
The states 100, 101 and 111 are called the Invalid/Illegal States.

Assume that the next state from any invalid state is a don't care state

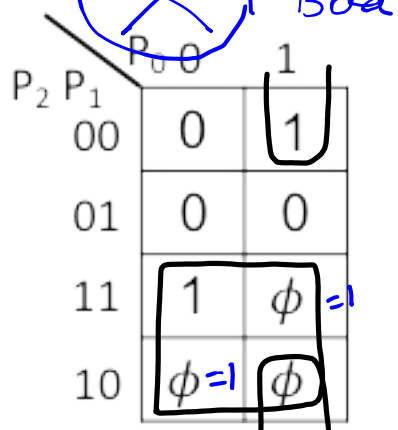
STT for TFF

P_2	P_1	P_0	N_2	N_1	N_0	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	0	1
0	1	0	1	1	0	1	0	0
1	1	0	0	0	0	1	1	0
1	0	0	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
1	0	1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
1	1	1	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

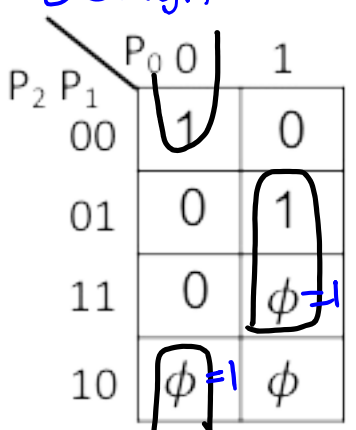
don't care



$T_2 = P_1\overline{P_0}$



$T_1 = P_2 + \overline{P_1}P_0$



$T_0 = P_1P_0 + \overline{P_1}\,\overline{P_0}$

$111 \rightarrow 100; 100 \rightarrow 111; 101 \rightarrow 111$

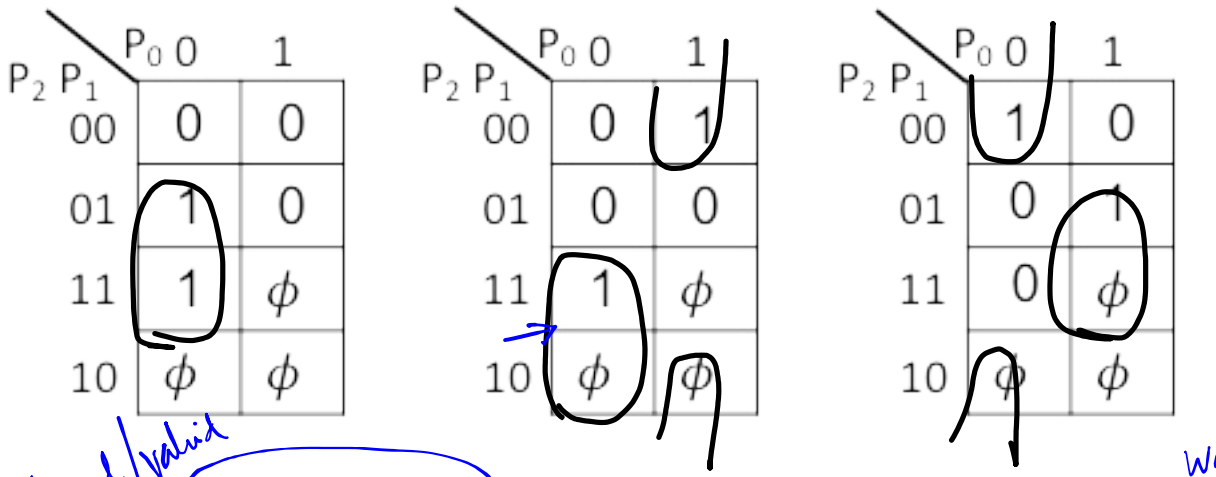
minimization

is this acceptable?

111 → 100 → 101 → 100

Stuck in invalid state

$$T_2 = P_1\overline{P_0}$$
$$T_1 = P_2\overline{P_0} + \overline{P_1}P_0$$
$$T_0 = \overline{P_1}\,\overline{P_0} + P_1P_0$$



P_2P_1 \ P_0	0	1
00	0	0
01	1	0
11	1	ϕ
10	ϕ	ϕ

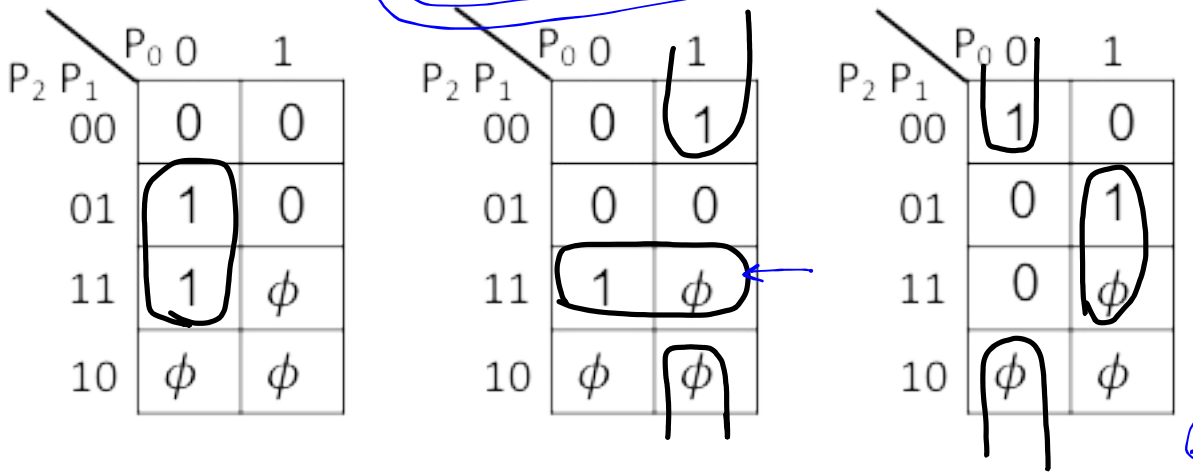
P_2P_1 \ P_0	0	1
00	0	1
01	0	0
11	1	ϕ
10	ϕ	ϕ

P_2P_1 \ P_0	0	1
00	1	0
01	0	1
11	0	ϕ
10	ϕ	ϕ

legal/valid

$111 \rightarrow 110; 100 \rightarrow 111; 101 \rightarrow 111$

waste



P_2P_1 \ P_0	0	1
00	0	0
01	1	0
11	1	ϕ
10	ϕ	ϕ

P_2P_1 \ P_0	0	1
00	0	1
01	0	0
11	1	ϕ
10	ϕ	ϕ

P_2P_1 \ P_0	0	1
00	1	0
01	0	1
11	0	ϕ
10	ϕ	ϕ

$111 \rightarrow 100; 100 \rightarrow 101; 101 \rightarrow 111$

Equally Bad

$$T_1 = P_2P_1 + \overline{P_2}\,\overline{P_1}P_0$$
$$T_2 = P_2 + P_1\overline{P_0}$$
$$T_0 = P_2P_0 + P_1P_0 + \overline{P_2}\,\overline{P_1}\,\overline{P_0}$$

less gates than setting next state 2 cycles. 000 for 111, 100 & 101

Invalid States:

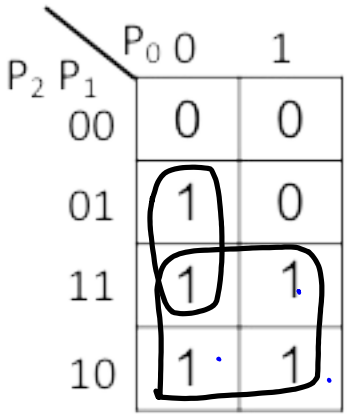
Design counter using T-FF with states $000 \rightarrow 001 \rightarrow 011 \rightarrow 010 \rightarrow 110 \rightarrow 000$

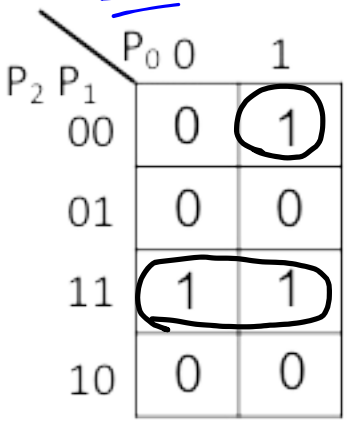
The states 100, 101 and 111 are called the Invalid States.

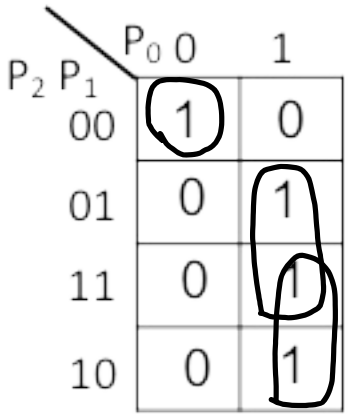
Assume that the next state from any invalid state is 000 state

wastes 1 cycle

P_2	P_1	P_0	N_2	N_1	N_0	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	0	1
0	1	0	1	1	0	1	0	0
1	1	0	0	0	0	1	1	0
1	0	0	0	0	0	1	0	0
1	0	1	0	0	0	1	0	1
1	1	1	0	0	0	1	1	1







$$T_1 = P_2P_1 + \overline{P_2}\,\overline{P_1}P_0$$

$$T_2 = P_2 + P_1\overline{P_0}$$

expensive

$$T_0 = P_2P_0 + P_1P_0 + \overline{P_2}\,\overline{P_1}\,\overline{P_0}$$

larger nr. of gates reqd.