# Sequential logic design

Vikas Dhiman for ECE275

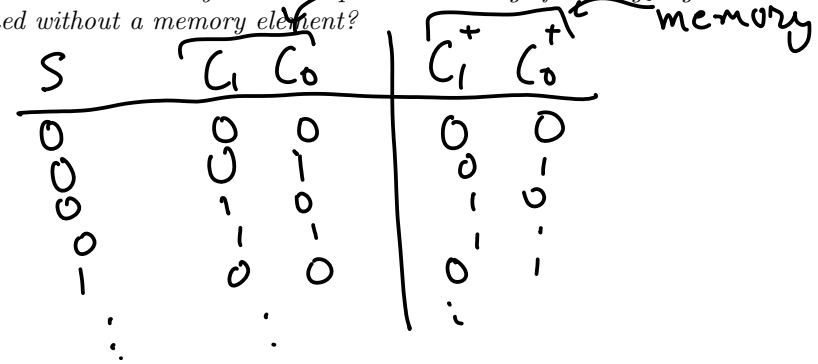
October 11, 2023

## 1 Objectives

1. Understand timing diagrams, gate delays and critical path
2. Design Hazard-free two level circuits
3. Building blocks of sequential circuits
4. Analyze a sequential circuit and derive a state-table and a state-graph
5. Derive a state graph or state table from a word description of the problem
6. Understanding the structure of an FPGA

## 2 Why do we need sequential circuits?

**Example 1.** *Think about this problem: Design an occupancy counter that depends on a sensor $S$ at the class door. The sensor is triggered every time a person passes through the door. The counter can be reset to zero with a reset button. Assume we only need up to two bit counter $C_1C_0$. Draw a truth table for this circuit. Do you have requisite knowledge for designing this circuit? Can this circuit be designed without a memory element?*

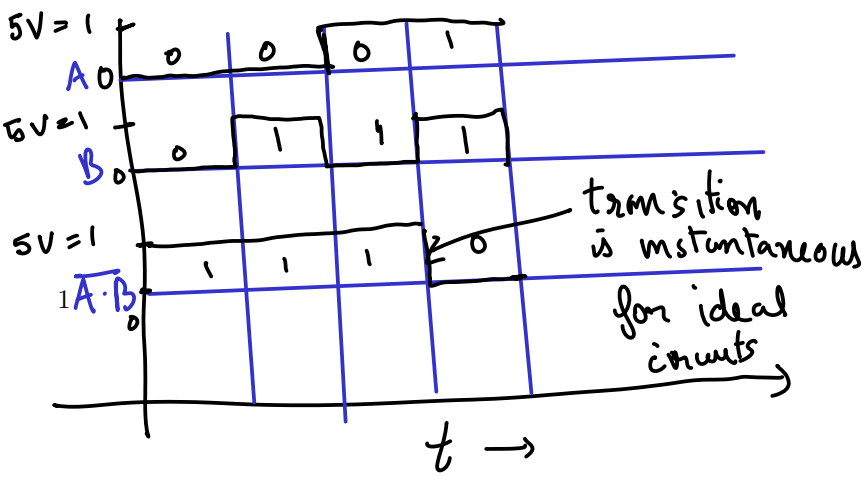



## 3 Timing diagrams and propagation delays

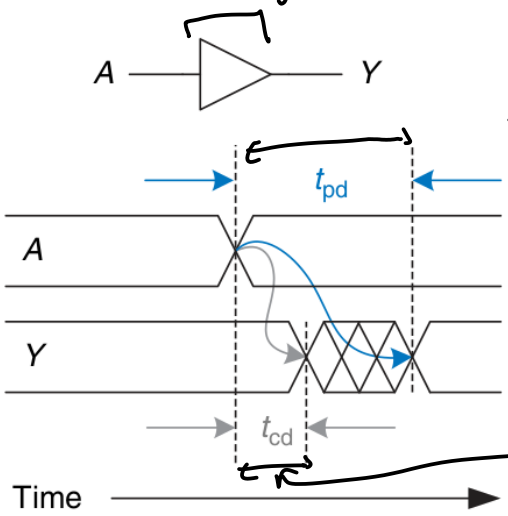
**Example 2** (Timing diagram)**.** *Draw a timing diagram for an ideal NAND gate.*

Truth table for NAND gate

| A | B | $\overline{A \cdot B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



### 3.1 Delays




| A | Y |
|---|---|
| 0 | 0 |
| 1 | 1 |

$t_{pd}$ = Propagation delay

$t_{cd}$ = Contamination delay

**Definition 1** (Propagation delay ($t_{pd}$))**.**

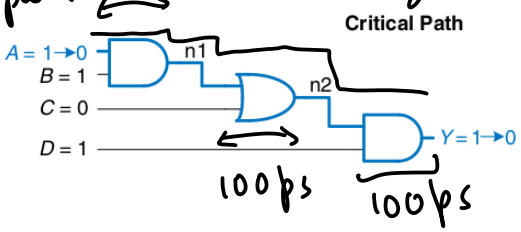
Time delay between change in input to a stablized desired output.

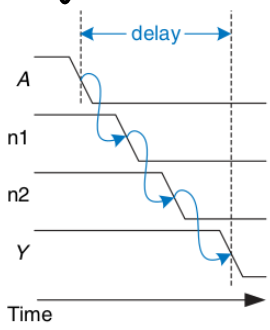
**Definition 2** (Contamination delay ($t_{cd}$))**.**

Time delay between change in input to *any* change in output.

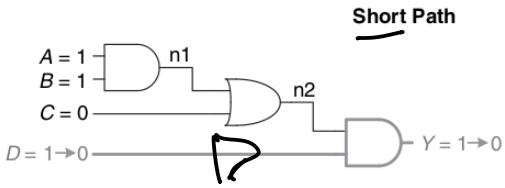
### 3.2 Paths

Use critical path for finding $t_{pd}$ ✓ = Longest path from any input to the output in the circuit

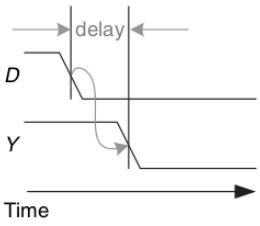



Use short path for finding $t_{cd}$ ✓




$t_{cd} = 60$ ps

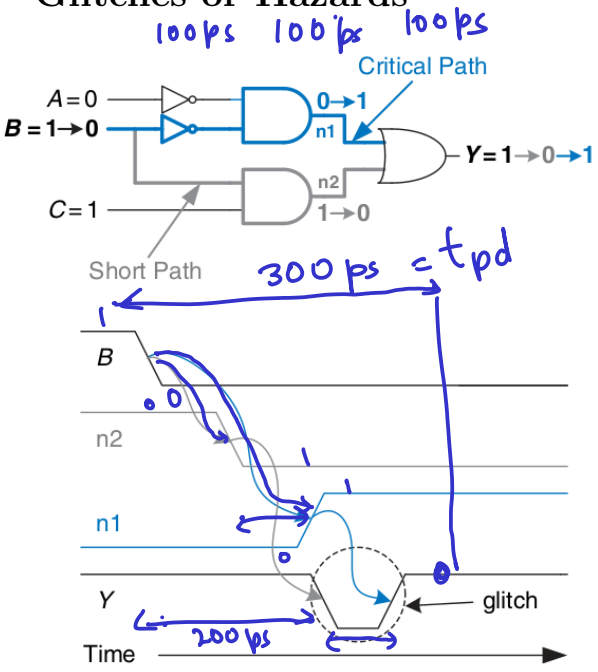


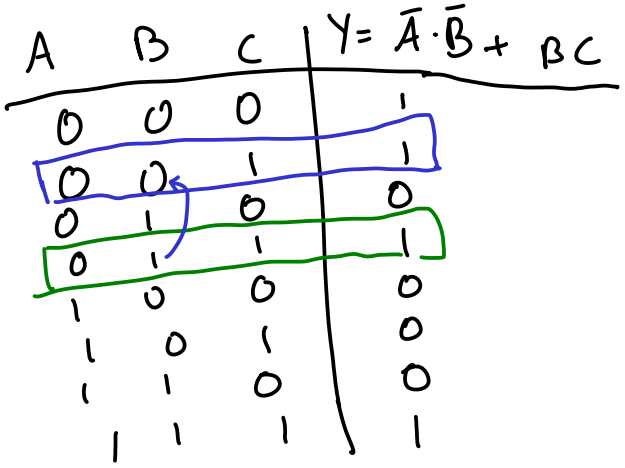

is the shortest path:

**Example 3.** *Find the propagation delay of the circuit above given that propagation delay of each gate is* 100*ps add contamination delay of* 60*ps*

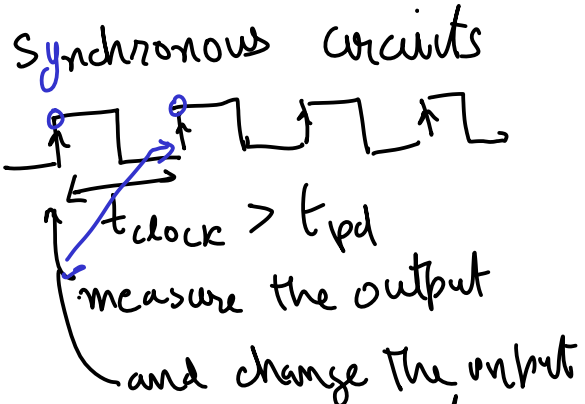
$t_{pd} = 300$ ps $\quad t_{cd} = 60$ ps

## 4 Glitches or Hazards

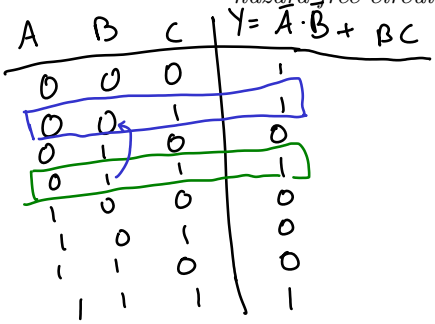


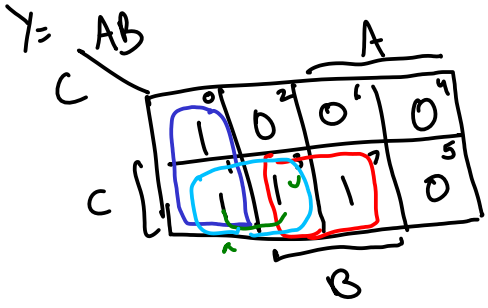




**Definition 3** (Glitch or Hazard)**.**




**Example 4.** *Design a circuit that fixes the glitch in the above circuit (also known as glitch-free or hazard-free circuit).*




$Y = \bar{A}\bar{B} + BC$

Glitch free

$Y = \bar{A}\bar{B} + \bar{A}C + BC$

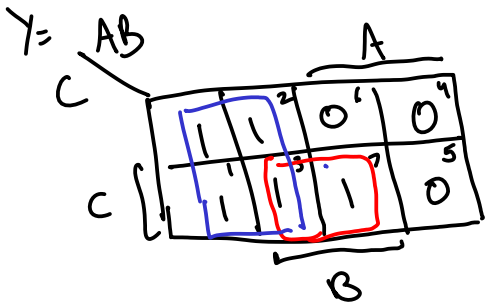
## 5 How to create memory element from circuits

Two types of memory

1. Volatile memory. For example, RAM, CPU registers.
2. Non-volatile memory. For example, SSD, Flash drives. (Not covered in this course)
   (a) Memories that require periodic refreshing. For example, DRAM: Dynamics Random Access memory (Not covered in this course)

Detecting Glitch from K-map

1. If there exists PI that are next to each other but non-overlapping, then there is a glitch when transitioning from neighboring minterms.



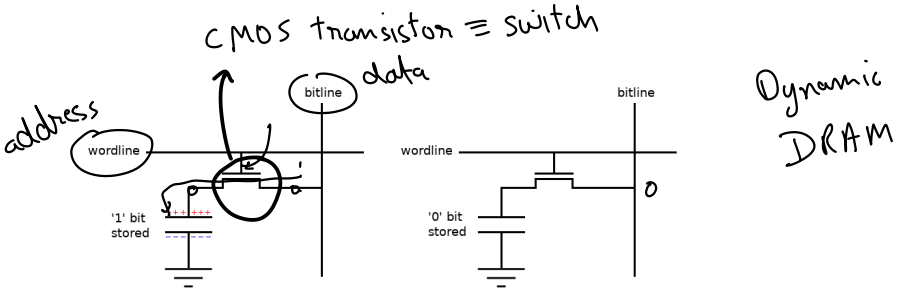

$Y = \bar{A} + BC$

$Y = \bar{A} + BCA$

Glitch would happen on $m_7 \rightleftharpoons m_3$ transition

Removing a Glitch (Hazard) using a K-map

1. Add a redundant PI that makes PIs to be overlapping (or groups the offending minterms).




[1]

(b) Memories that are always refreshing. For example, SRAM: Static Random Access memory [2, Appendix B.64]

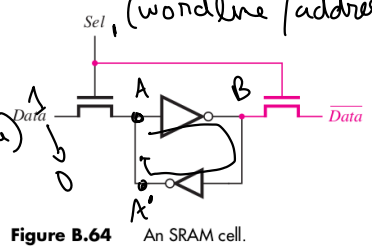
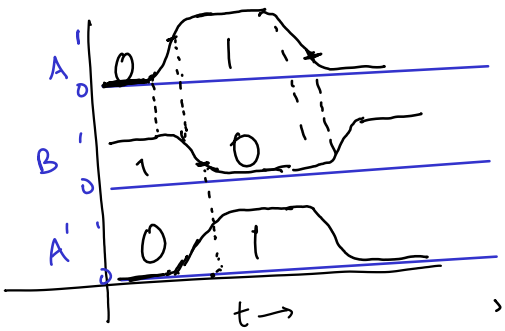



**Figure B.64** An SRAM cell.

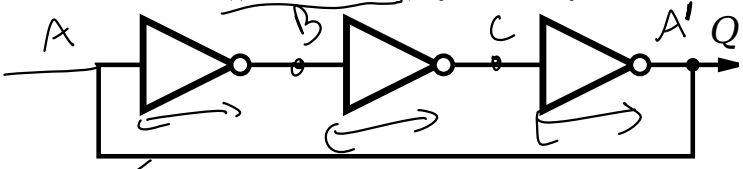
BISTABLE circuit




Memory element

## 6 Latches and Flip-Flops [1, Sec 3.2]

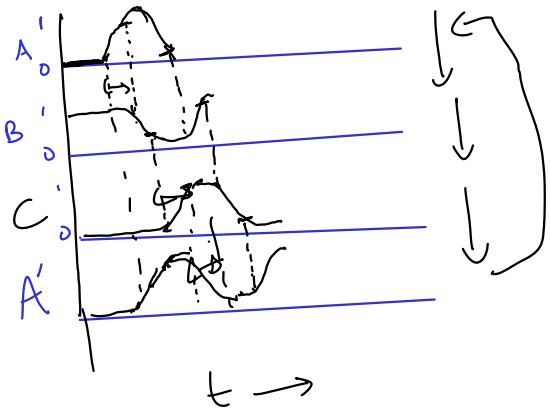
**Example 5** (Ring oscillator). *[1, Sec 3.31] How many stable states does the following circuit have?*




ASTABLE circuit

OSCILLATOR

$t_{clk} = 2(3t_{pd})$

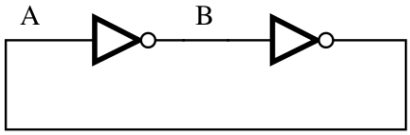


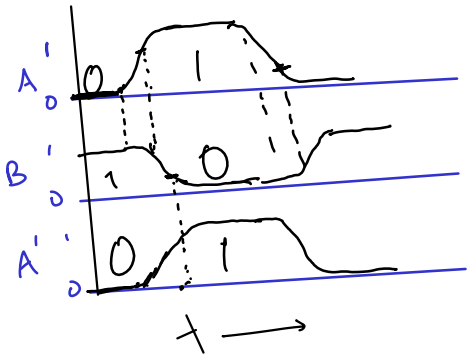

**Definition 4** (Astable circuits).

A circuit without a stable state.

**Example 6.** *Analyze the timing diagram of the following circuit.*

[1] Image source: `allaboutcircuits.com/technical-articles/introduction-to-dram-dynamic-random-access-memory/`




**Definition 5** (Bistable circuits)**.**

The circuits that have two stable states.

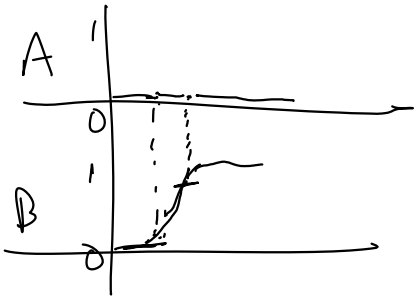
**Definition 6** (Characteristic or state table)**.** *Draw the characteristic or state table of the above circuit.*

| $A_t$ | $B_{t+t_{pd}}$ | $A_{t+\Delta t}$ | $B_{t+\Delta t}$ |
|---|---|---|---|
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 0 | x | x |

tclk tclk

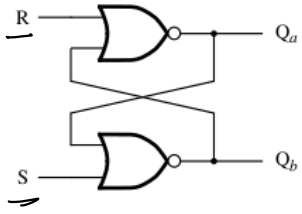
invalid state

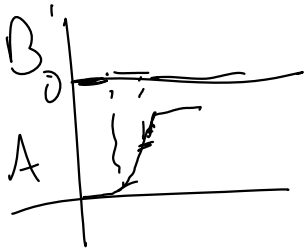
Indeterminate



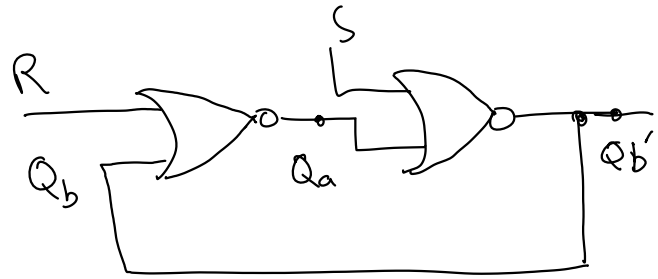

## 6.1 SR (Set-Reset) latch [1, Sec 3.2.1]

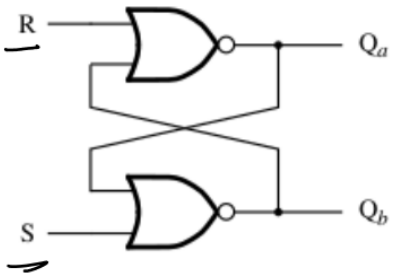
**Definition 7** (SR latch)**.** *The following circuit is called the SR latch.*

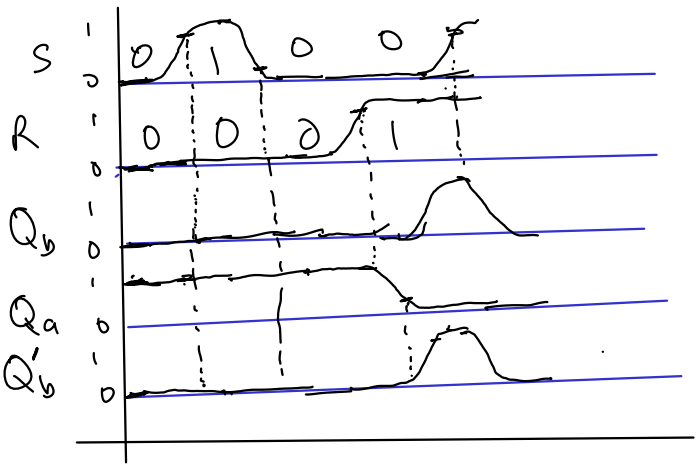



1. *How many stable states does this circuit have?*
2. *Draw its characteristic or state table.*
3. *Draw SR latch symbol*




$Q_a = \overline{R + Q_b}$

$Q_b = \overline{S + Q_a}$

if $S = 0$, $Q_a = 1$

then $Q_b = \overline{1} = 0$

S = Set bit = 1

if $S = 1$ $Q_a = 1$

$Q_b = \overline{1+1} = \overline{1} = 0$

S = Set bit = 0

R = Reset bit = 1

if $R = 1$, $Q_b = 0$

$Q_a = \overline{R + Q_b} = \overline{1+0} = \overline{1} = 0$

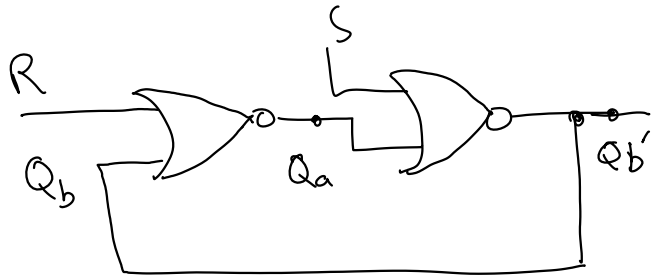
$Q_b = \overline{S + Q_a} = \overline{0+0} = \overline{0} = 1$

if $S = R = 1$

then $Q_a = \overline{R + Q_b} = \overline{1+1} = 0$

$Q_b = \overline{S + Q_a} = \overline{1+1} = 0$

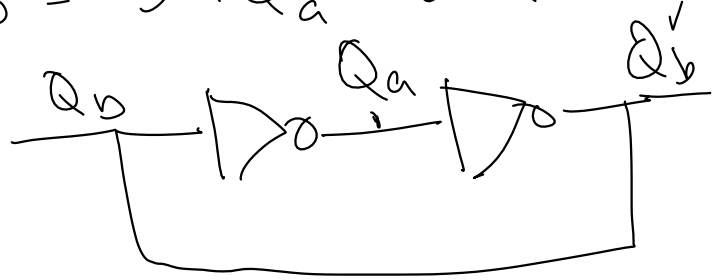
| | S | R | $Q_{at}$ | $Q_{bt}$ | $Q_{at+\Delta t}$ | $Q_{bt+\Delta t}$ |
|---|---|---|---|---|---|---|
| Bistable state | 0 | 0 | 0 | 1 | 0 | 1 |
| | 0 | 0 | 1 | 0 | 1 | 0 |
| Reset → | 0 | 1 | * | * | 0 | 1 |
| Set | 1 | 0 | * | * | 1 | 0 |
| → Invalid state for S-R latch | 1 | 1 | 0 | 0 | 0 | 0 |



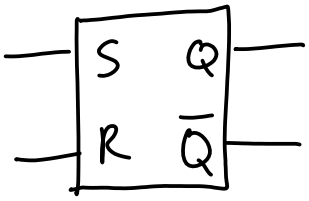

when $R=0$, $S=0$,

$$Q_a = \overline{R+Q_b} = \overline{0+Q_b} = \overline{Q_b}$$

$$Q_b = \overline{S+Q_a} = \overline{0+\overline{Q_a}} = \overline{Q_a}$$




## S-R latch



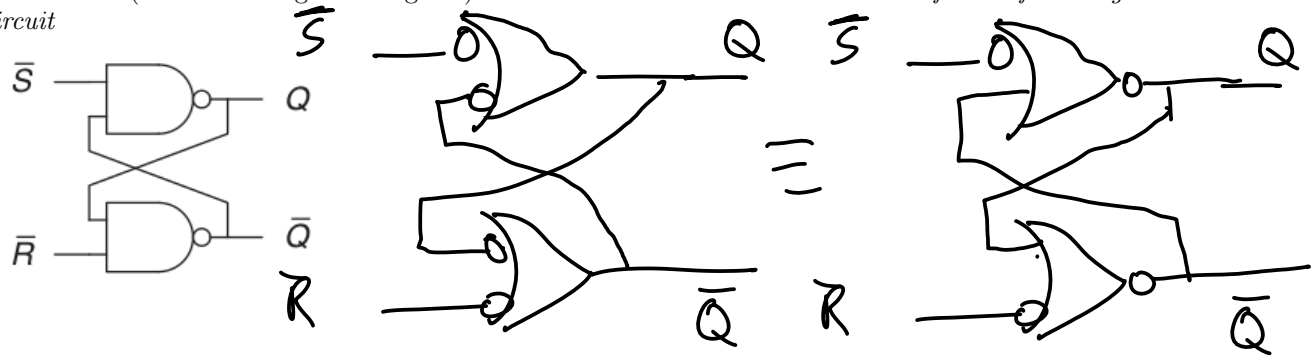

Characteristic table or the State table

| | S | R | $Q_t$ | $Q_{t+\Delta t}$ |
|---|---|---|---|---|
| Hold state | 0 | 0 | 0 | 0 |
| | 0 | 0 | 1 | 1 |
| Reset | 0 | 1 | 0 | 0 |
| | 0 | 1 | 1 | 0 |
| Set | 1 | 0 | 0 | 1 |
| | 1 | 0 | 1 | 1 |
| Invalid | 1 | 1 | 0 | 0 |
| | 1 | 1 | 1 | 0 |

Memory / State

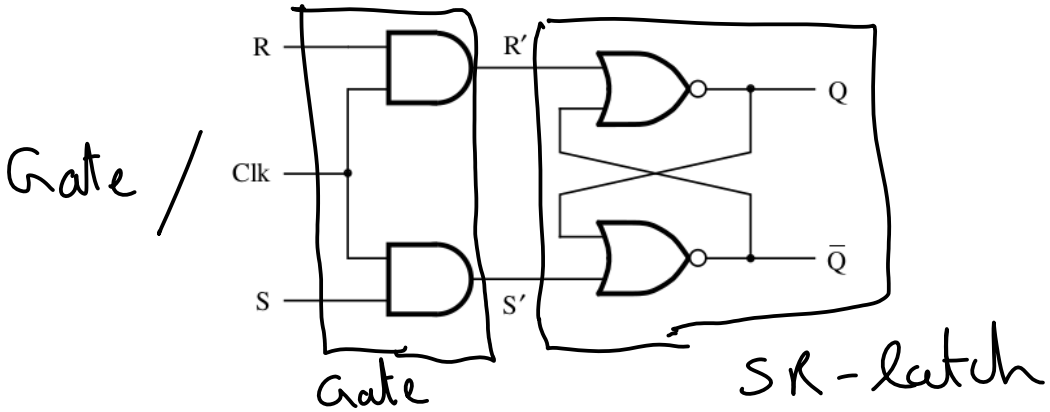
| | S | R | $Q_{t+\Delta t}$ |
|---|---|---|---|
| Hold → | 0 | 0 | $Q_t$ |
| Reset → | 0 | 1 | 0 |
| Set → | 1 | 0 | 1 |
| Invalid → | 1 | 1 | d |

**Problem 1** (SR latch using NAND gates). *Draw the characteristic or state table for the following circuit*




## 6.2 Gated SR latch [2, Sec 5.2]

**Definition 8** (Gated SR latch). *The following circuit is called the Gated SR latch.*




1. *Draw its characteristic table.*
2. *Draw the Gated SR latch symbol*

if Gate = 0 then both $R' = S' = 0$ (Hold state)

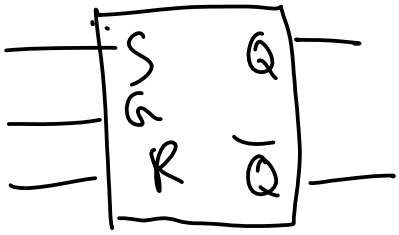
Gate = 1 then $R' = R$

$S' = S$

Characteristic table for Gated SR-latch

| G | S | R | $Q_{t+\Delta t}$ | |
|---|---|---|---|---|
| 0 | * | * | $Q_t$ | ← Hold state |
| 1 | 0 | 0 | $Q_t$ | ← |
| 1 | 0 | 1 | 0 | ← Reset state |
| 1 | 1 | 0 | 1 | ← set state |
| 1 | 1 | 1 | d | ← Invalid |

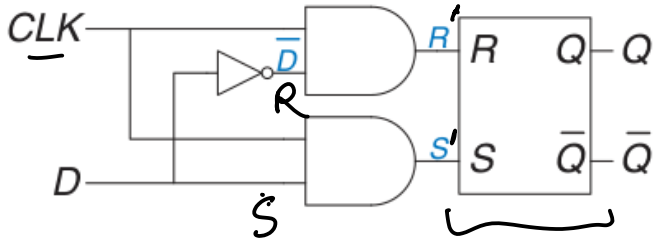
# Symbol for Gated SR-latch

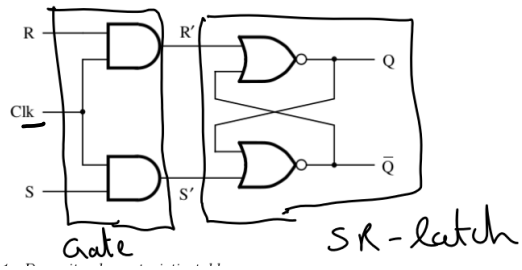


### 6.3 D (Data) latch [1, Sec 3.2.2]

**Definition 9** (D latch). *The following circuit is called the D latch.*

Gate =




1. *Draw its characteristic table.*
2. *Draw the D latch symbol*

S = D

R = D̄

D-latch characteristic table

| Clk | D | S' | R' | $Q_{t+\Delta t}$ | |
|---|---|---|---|---|---|
| 0 | * | 0 | 0 | $Q_t$ | ← Hold state |
| 1 | 0 | 0 | 1 | 0 | ← Reset state |
| 1 | 1 | 1 | 0 | 1 | ← Set state |

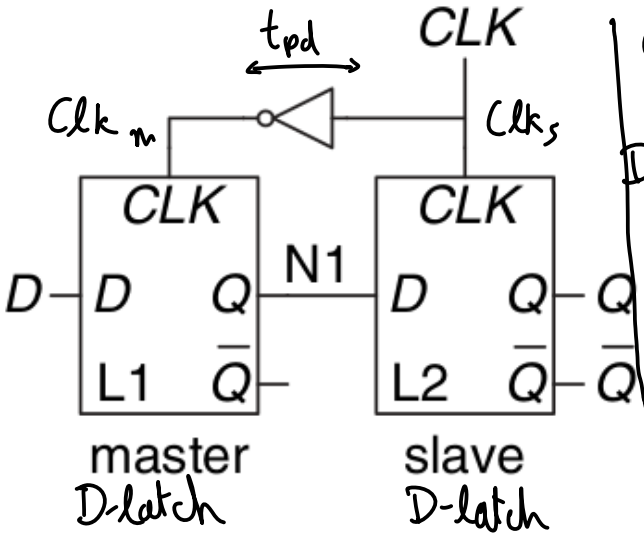
Latches: Level triggered

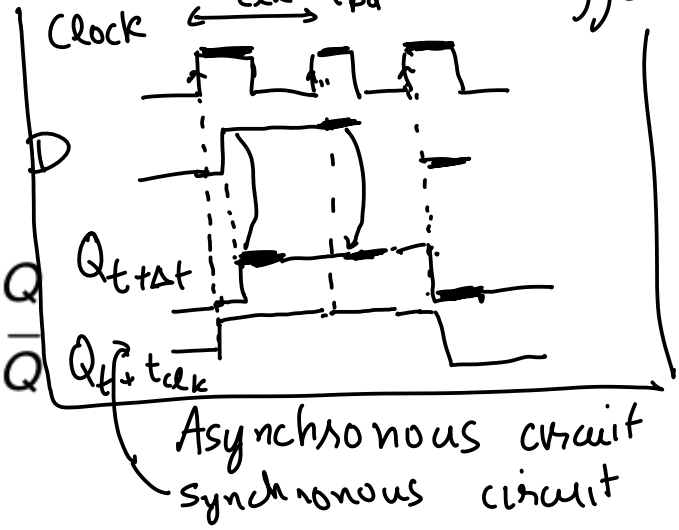
vs

Flip flops: Edge triggered

### 6.4 D flip-flop [1, Sec 3.2.2]

**Definition 10** (D flip-flop). *The following circuit is called the D flip-flop.*



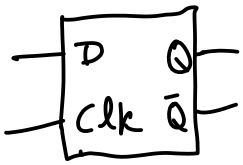
D - latch

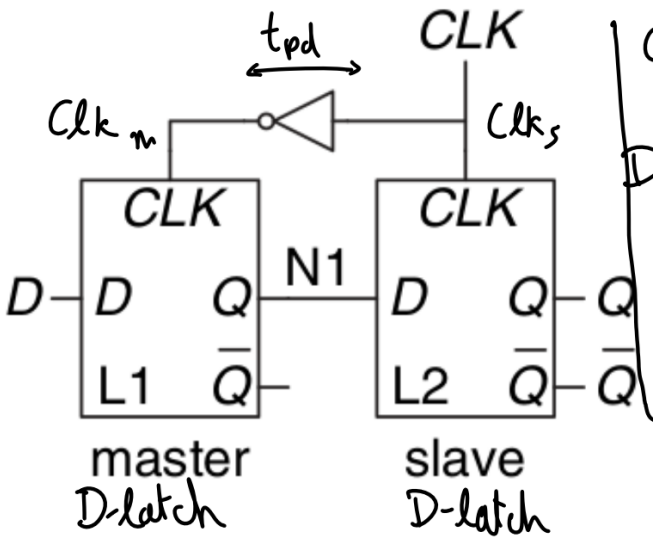
| Clk | D | S' | R' | $Q_{t+\Delta t}$ | |
|---|---|---|---|---|---|
| 0 | * | 0 | 0 | $Q_t$ | ← Hold state |
| 1 | 0 | 0 | 1 | 0 | ← Reset state |
| 1 | 1 | 1 | 0 | 1 | ← Set state |

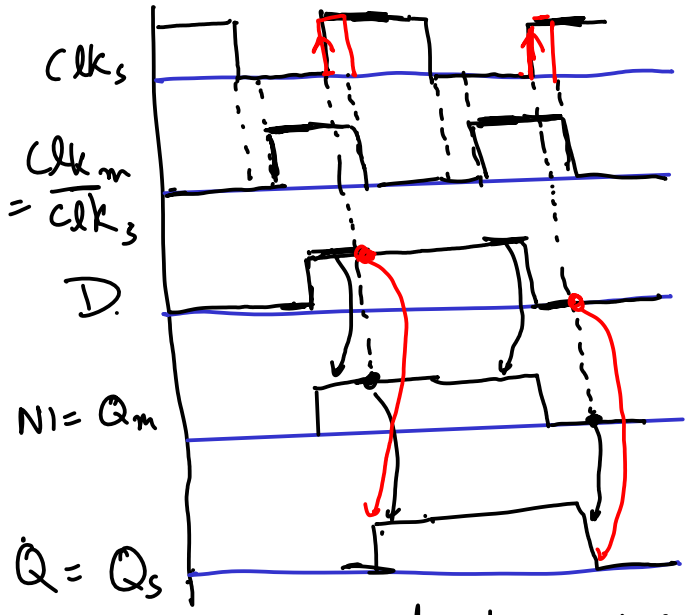
| Clk | $Q_{t+\Delta t}$ | |
|---|---|---|
| 0 | $Q_t$ | ← Hold state |
| 1 | D | ← Transparent state |

D - latch Data latch
Transparent latch

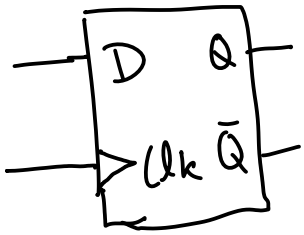
D-latch symbol






## D- flip flop symbol



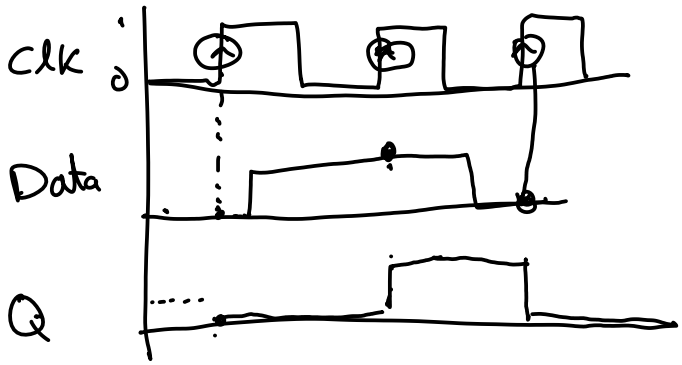

## D- flip flop characteristic table or state table

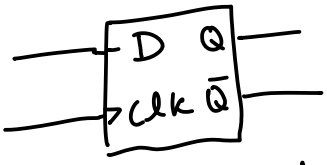
| D | $Q_{t+t_{clk}} = Q_{t+1} = Q^+$ |
|---|---|
| 0 | 0 |
| 1 | 1 |

transparent part

D-flip flop = Data flip flop (edge triggered

1. *Draw its timing diagram*
2. *Draw its characteristic table.*
3. *Draw the D flip-flop symbol*




characteristic table

| D | Q | $Q^+$ |
|---|---|---|
| 0 | d | 0 |
| 1 | d | 1 |

**Remark 1.** *What is the difference between a latch and a flip-flop?*

**Example 7.** *Add a* RESET *signal to the D flip-flop that resets the state of flip-flop to 0.*

**Example 8.** *The toggle (T) flip-flop has one input, CLK, and one output, Q. On each rising edge of CLK, Q toggles to the complement of its previous value. Draw a schematic for a T flip-flop using a D flip-flop and an inverter.*

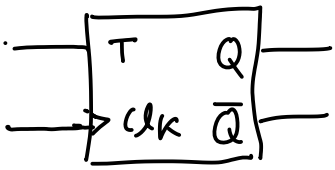
when T = 1

① D-flip flop

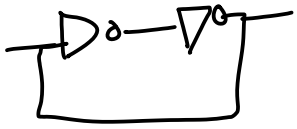
② T-flip flop ≡ Toggle FF

③ JK flip flop

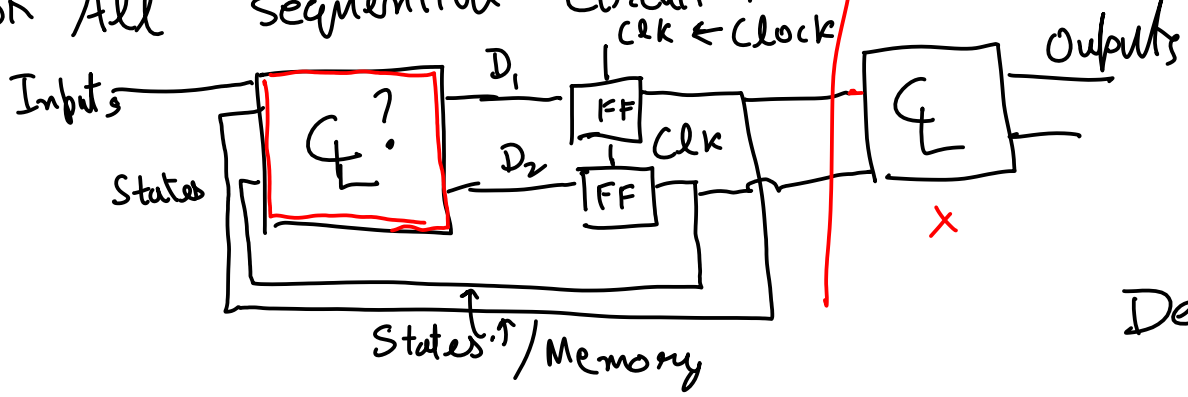
| T | Q | $Q^+$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




we want to design T- FF using D-FF

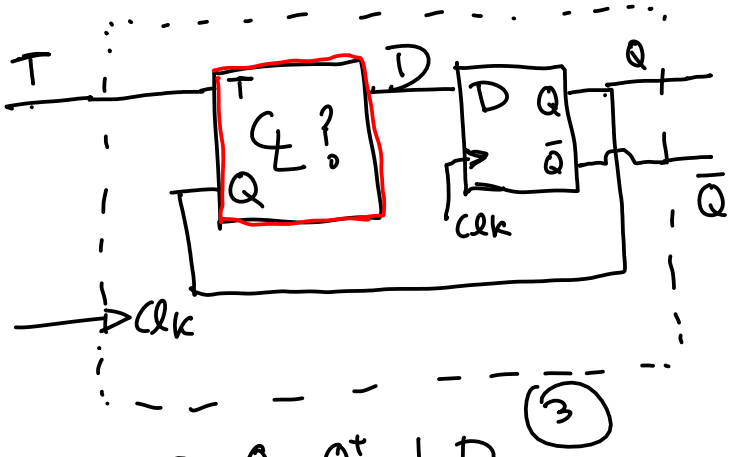


For All sequential circuit :




Toggle ff

1-bit of memory




Desired state table (1)

| T | Q | $Q^+$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Truth table

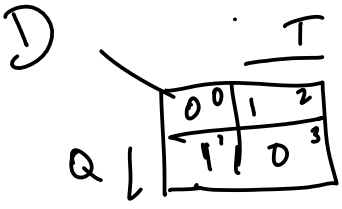
| T | Q | D |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| Q | $Q^+$ | D |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

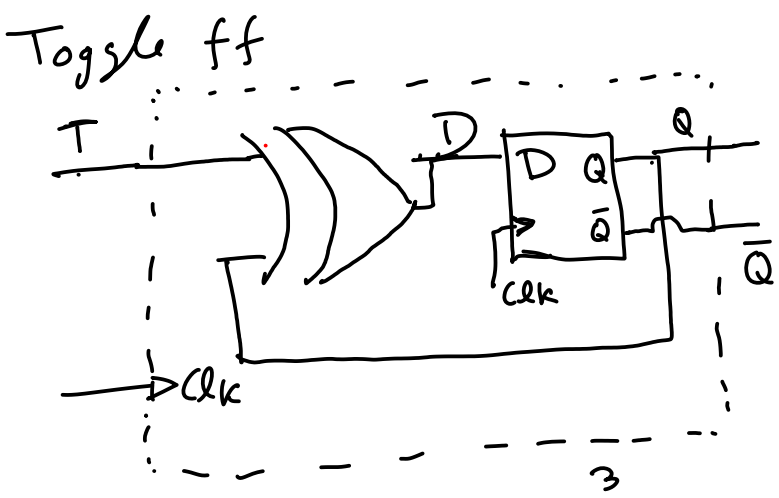
Excitation table

State table (2)

| D | Q | $Q^+$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |




$$D = T\bar{Q} + \bar{T}Q = T \oplus Q$$




Can you design D-ff using Toggle ff?

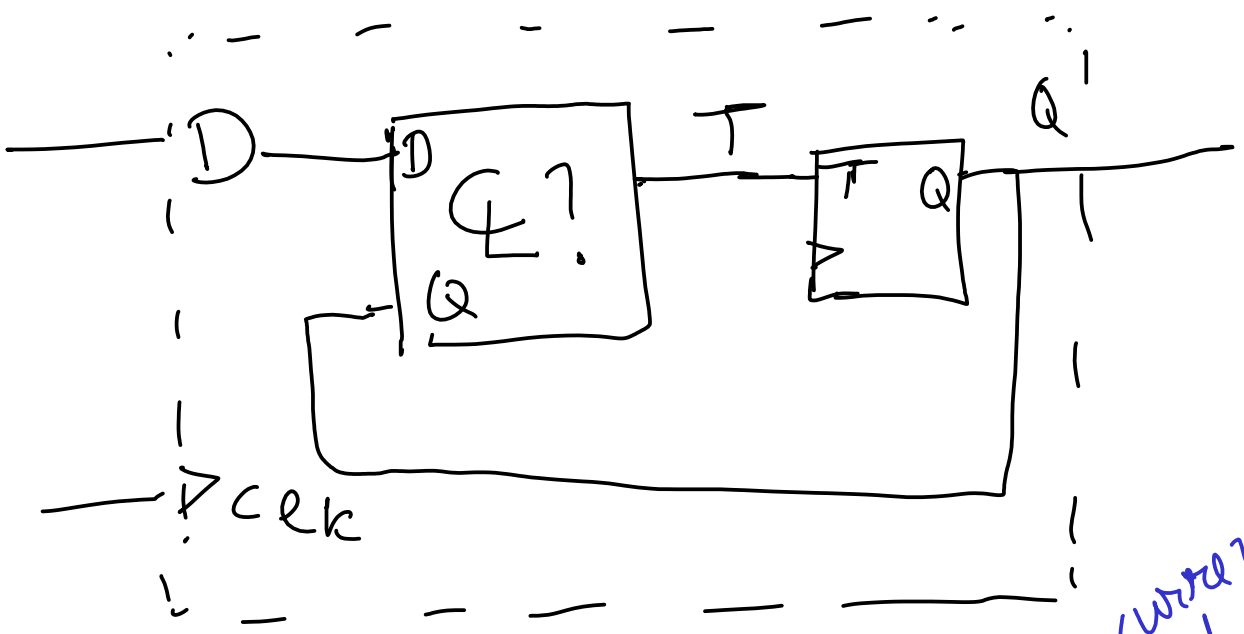
(1) Step 1: Write the desired state table

(2) Step 2: " the state table of given ff

(3) Step 3: " the excitation table of the given ff

(4) Step 4: Use (1) and (3) to write the truth table for G circuit

(5) Draw K-map for 4 and design the circuit




① Desired State table

current state

Next state

| D | Q | $Q^+$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

② State table of the given ff = T-ff

Present state

Next state

| T | Q | $Q^+$ | |
|---|---|---|---|
| 0 | 0 | 0 | Hold |
| 0 | 1 | 1 | |
| 1 | 0 | 1 | Toggle |
| 1 | 1 | 0 | |

(3) Excitation table

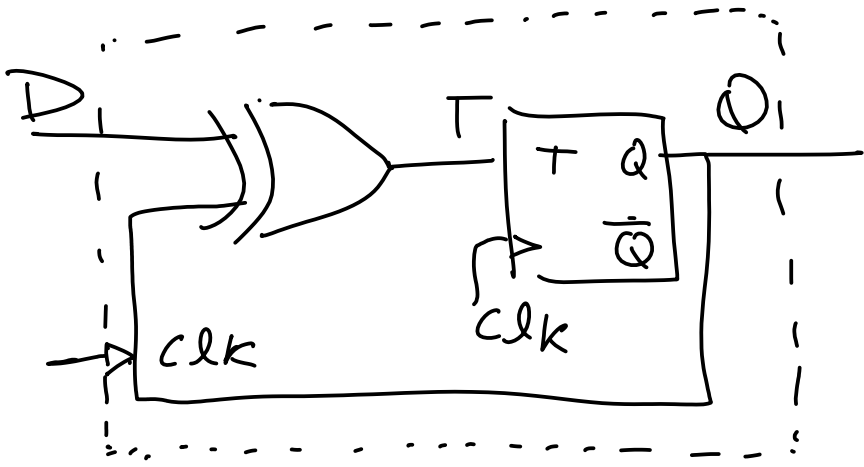
| Q | $Q^+$ | T |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| D | Q | $Q^+$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

(4) Truth table for the Q

| D | Q | T |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$T = D \oplus Q$



**Problem 2.** *A JK flip-flop receives a clock and two inputs, J and K. On the rising edge of the clock, it updates the output, Q. If J and K are both 0, Q retains its old value. If only J is 1, Q becomes 1. If only K is 1, Q becomes 0. If both J and K are 1, Q becomes the opposite of its present state.*

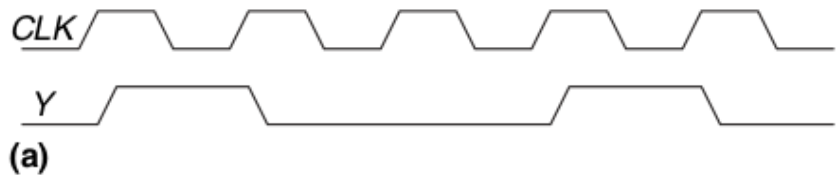
1. *Construct a JK flip-flop using a D flip-flop and some combinational logic.*
2. *Construct a D flip-flop using a JK flip-flop and some combinational logic.*
3. *Construct a T flip-flop (see Exercise 3.9) using a JK flip-flop.*

# 7 Finite State Machines [1, Sec 3.4]

[2]

**Example 9.** *Design an occupancy counter that depends on a sensor S at the class door. The sensor is triggered every time a person passes through the door. Assume that the counter starts at zero. Assume we only need up to two bit counter $C_1C_0$. Draw a state table for this circuit.*

**Problem 3.** *A divide-by-N counter has one output and no inputs. The output Y is HIGH for one clock cycle out of every N. In other words, the output divides the frequency of the clock by N. The waveform for a divide-by-3 counter is shown here:*




*Sketch circuit designs for such a counter*

**Problem 4.** *Design a 3-bit counter which counts in the sequence: 001, 011, 010, 110, 111, 100, (repeat) 001, ...*

**Example 10.** *Design an odd-even counter for an single bit input. The output of this circuit should be 1 if the number of 1s to the input have been odd so far and 0 otherwise.*

**Example 11** (Sequence detectors)**.** *A sequential circuit has one input and one output. The output becomes 1 and remain 1 thereafter when at least two 0's and at least two 1's have occurred as inputs regardless of the order of*

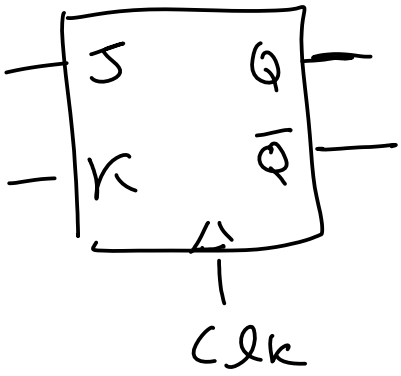
**Example 12.** *Consider the problem of inventing a controller for a traffic light at a busy intersection on campus. There are two traffic sensors, $T_A$ and $T_B$ , on Academic Ave. and Bravado Blvd., respectively. Each sensor indicates TRUE if students are present and FALSE if the street is empty. There are two traffic lights, $L_A$ and $L_B$, to control traffic. Each light receives digital inputs specifying whether it should be green, yellow, or red. When the system is reset, the lights are green on Academic Ave. and red on Bravado Blvd. As long as traffic is present on Academic Ave., the lights do not change. When there is no longer traffic on Academic Ave., the light on Academic Ave. becomes yellow for 5 seconds before it turns red and Bravado Blvd.'s light turns green. Similarly, the Bravado Blvd. light remains green as long as traffic is present on the boulevard, then turns*

[2] These notes will not fit on your note sheet.

① Design a J-K ff using D-ff

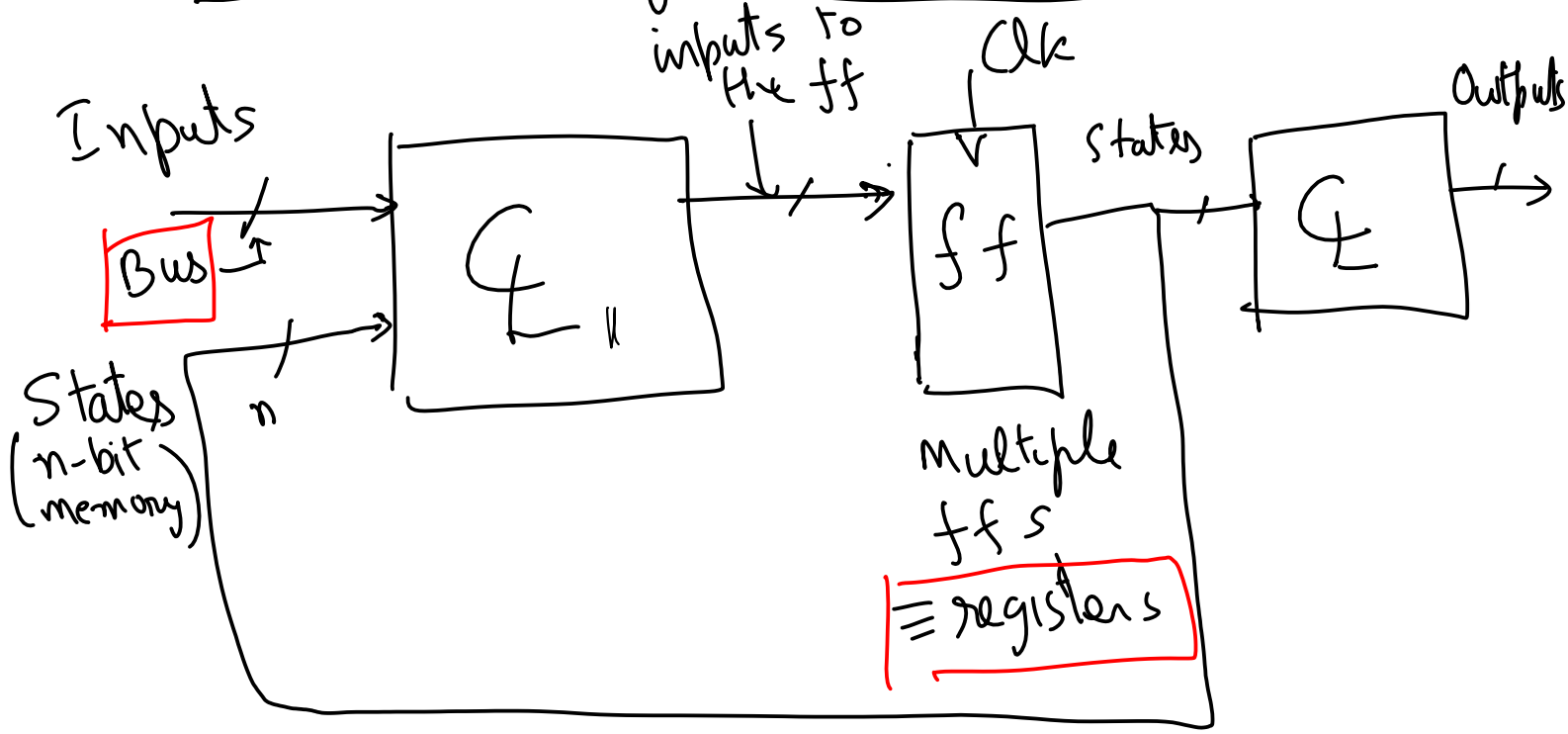
Desired state table

| Set. Reset (PS) J | K | Q | (NS) $Q^+$ | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Hold state |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | Reset state |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | Set state |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 1 | Toggle state |
| 1 | 1 | 1 | 0 | |




| J | K | $Q^+$ |
|---|---|---|
| 0 | 0 | Q |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\bar{Q}$ |

## Template for designing Seq Circuits



Steps

① Desired state table

② State table of ff that we are using

③ Excitation table of ff the we are using

④ Input-output table for the $Q_{+1}$

Inputs

Desired state table

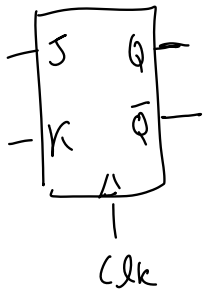
| Set. | Reset | (PS) | (NS) |
|---|---|---|---|
| J | K | Q | $Q^+$ |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Hold state

Reset state

Set state

Toggle state




| J | K | $Q^+$ |
|---|---|---|
| 0 | 0 | Q |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\bar{Q}$ |

State table D-ff

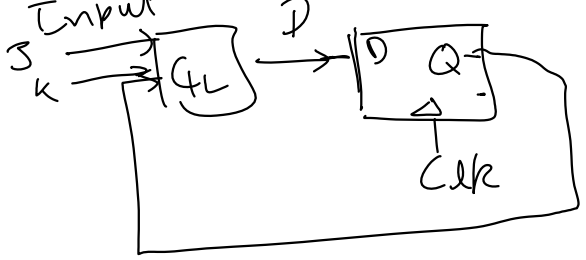
| D | Q | $Q^+$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

$Q^+ = D$

Excitation table

| Q | $Q^+$ | D |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$D = Q^+$




Inputs

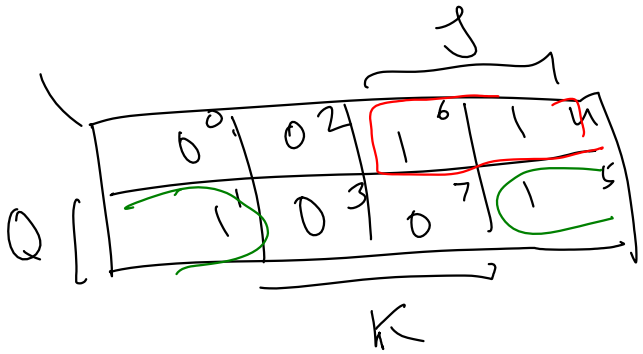
Desired state table

| Set | Reset | (PS) Q | (NS) $Q^+ = D$ | |
|---|---|---|---|---|
| J | K | Q | | |
| 0 | 0 | 0 | 0 | Hold state |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | Reset state |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | Set state |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 1 | Toggle state |
| 1 | 1 | 1 | 0 | |

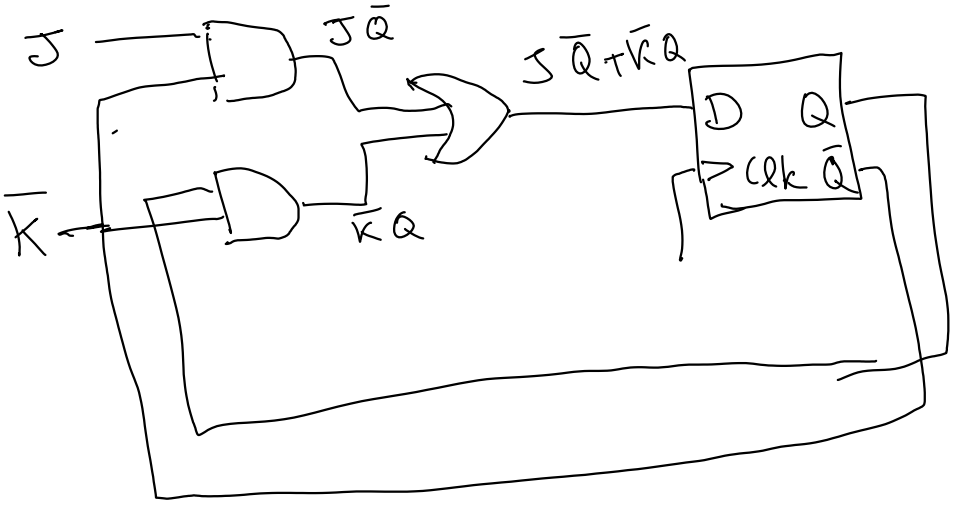
Truth table for FL

| J | K | Q | D |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

D




$D = J\bar{Q} + \bar{K}Q$



**Example 9.** *Design an occupancy counter that depends on a sensor S at the class door. The sensor is triggered every time a person passes through the door. Assume that the counter starts at zero. Assume we only need up to two bit counter $C_1C_0$. Draw a state table for this circuit.*

State table

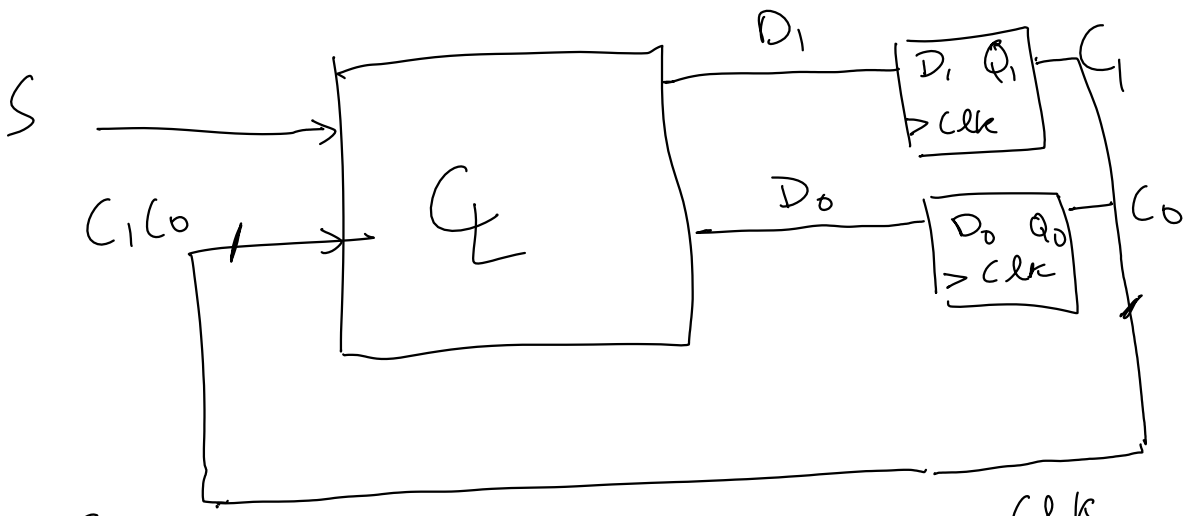
Inputs = ? = S

States = ? = 2 bits of memory = 2 state variable = 2 ffs (2-bit register)

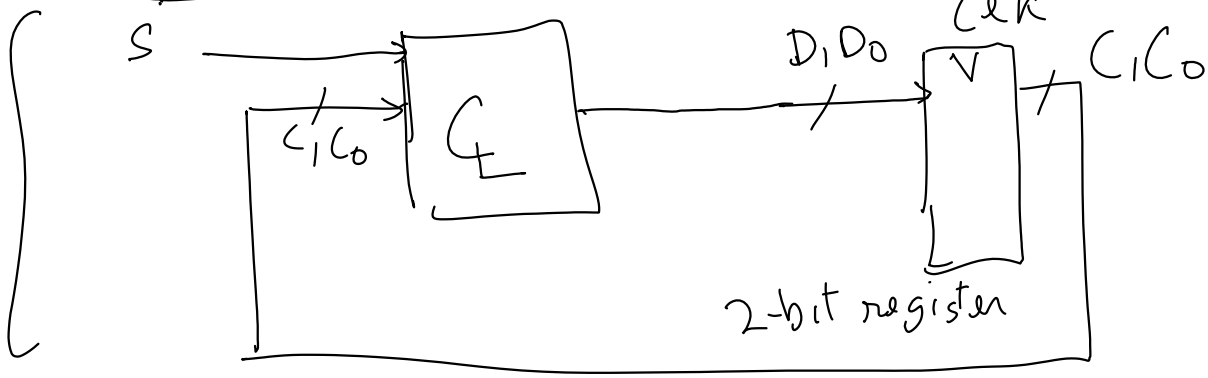
Outputs = $C_1C_0$ = $C_1C_0$

State table

| Inputs | Present State | | Next State | |
|---|---|---|---|---|
| | | | $C_1^+$ | $C_0^+ = D_1D_0$ |
| S | $C_1$ | $C_0$ | | |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |

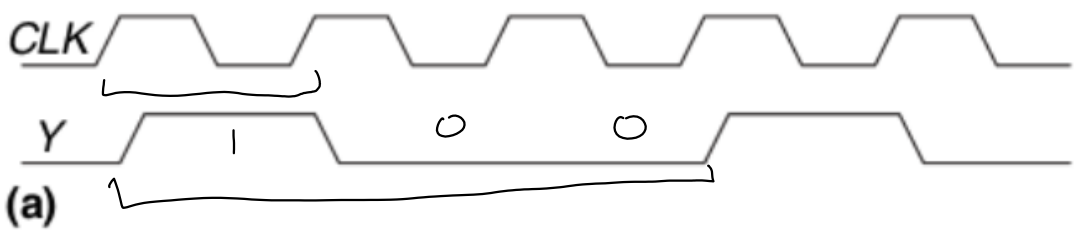



Simpler notation



Use K-maps to design circuit of $D_1$ and $D_0$ with inputs $S, C_1, C_0$

**Problem 3.** *A divide-by-N counter has one output and no inputs. The output Y is HIGH for one clock cycle out of every N. In other words, the output divides the frequency of the clock by N. The waveform for a divide-by-3 counter is shown here:*




*Sketch circuit designs for such a counter*

Divide by 3 - counter : Need to count until 3

2-bits $2^1 < 3 \leq 2^2 = 2 = \text{ceil}(\log_2 3)$

" " 5 - counter $2^2 < 5 \leq 2^3$

3-bits

7 - counter $2^2 < 7 \leq 2^3$

3-bits

9 - counter 4-bits

---

Ping pong game Project.

5 MHz clock $\xrightarrow{\text{Divide by } 10^6 = 10^3 \cdot 1000}$ 5 Hz clock

$\log_2(17)$

$20 \times \log_2(10^6) \approx 10^3 (1024) = 2^{10} (2^{10}) = 2^{20}$

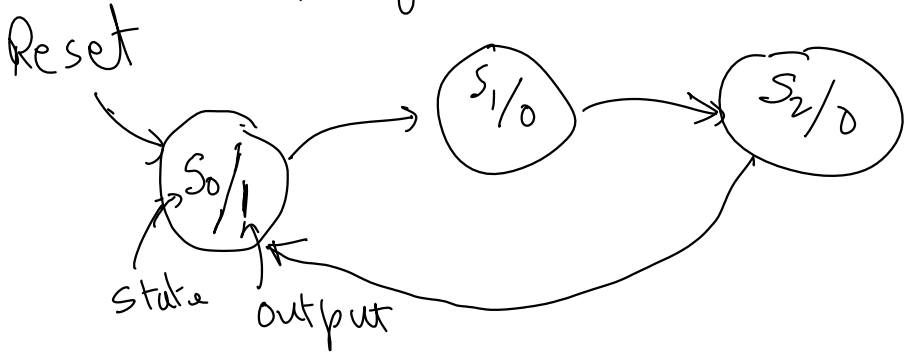
# State transition diagram

Is like drawing the state table as a graph.

But you don't need numbers for states yet.

Also, you add one state at a time

Divide by 3




State table

| Input | PS | NS |
|---|---|---|
| | $S_0$ | $S_1$ |
| | $S_1$ | $S_2$ |
| | $S_2$ | $S_0$ |

State assignment problem

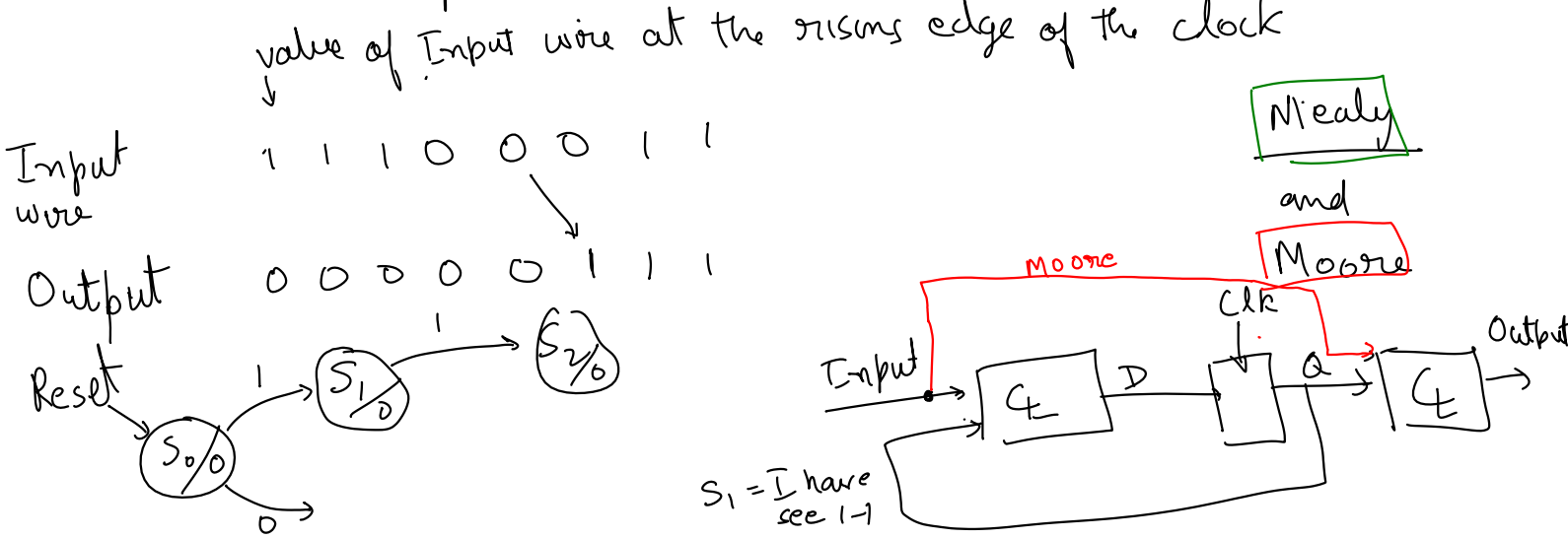
$S_0 = 00$
$S_1 = 01$
$S_2 = 11$

| PS $Q_1 Q_0$ | NS $Q_1^+ Q_0^+$ |
|---|---|
| 0 0 | 0 1 |
| 0 1 | 1 1 |
| 1 0 | d d |
| 1 1 | 0 0 |

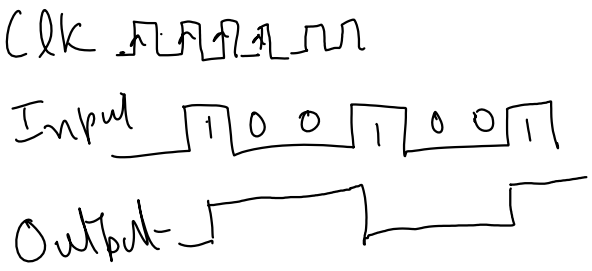
State assignment problem:

How to optimally assign binary encodings (numbers) to each state so that we get the optimal circuit

**Example 11** (Sequence detectors). *A sequential circuit has one input and one output. The output becomes 1 and remain 1 thereafter when at least two 0's and at least two 1's have occurred as inputs regardless of the order of inputs.*



**Example 10.** *Design an odd-even counter for an single bit input. The output of this circuit should be 1 if the number of 1s to the input have been odd so far and 0 otherwise.*




# of Inputs = 1

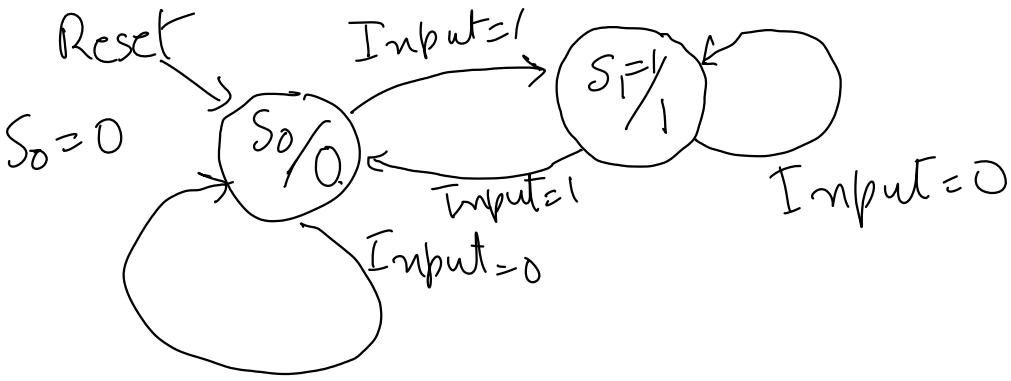
# of States = 2

How many bits?

1-bit memory

1-ff

State transition diagram




State encoding

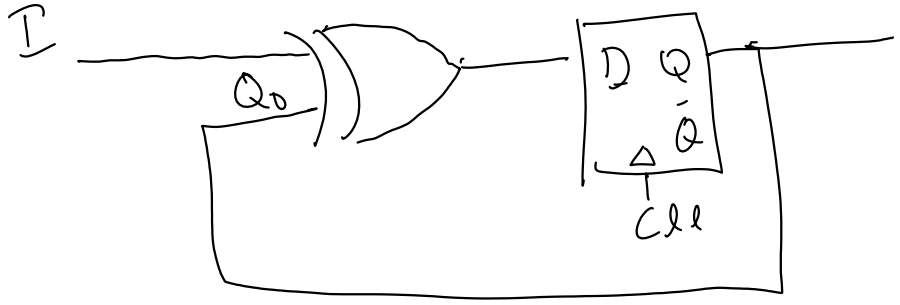
$S_0 = 0$

$S_1 = 1$

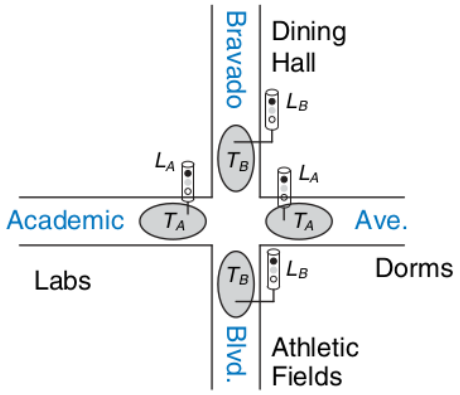
State table

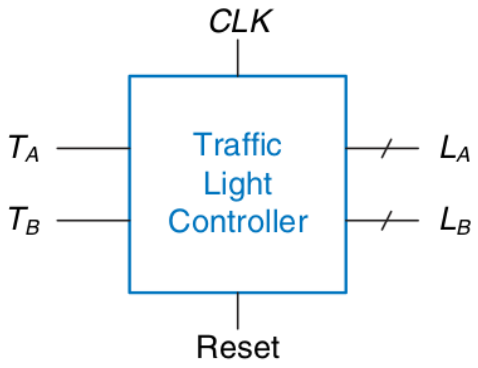
| Input $I$ | Present state $Q_0$ | Next state $Q_0^+ = D_0$ |
|---|---|---|
| 0 | 0 = Even | 0 |
| 0 | 1 = Odd | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$D_0$



*yellow and eventually red.*




1. *Draw a state transition diagram*
2. *Draw a state table*
3. *Assign binary encodings to each of the states*
4. *Redraw the state table with binary encodings. Design a minimal SOP boolean expression.*
5. *Assign binary encodings to each of the output and redraw the output table. Design a minimal SOP boolean expression for the outputs.*

**Problem 5.** *Design a circuit for a 2x2 pixel resolution pong game, where the ball can only occupy 4 possible pixels and a single paddle occupies another 2 pixels. The ball bounces of the paddle when the paddle is in the correct row. To keep it interesting, the ball takes a different path from the source path. Track the score with a single bit counter.*

# References


[1] Sarah L Harris and David Harris. *Digital design and computer architecture*. Morgan Kaufmann, 2022.

[2] Brown Stephen and Vranesic Zvonko. *Fundamentals of digital Logic with Verilog design*. McGraw Hill, 2022.

**Problem 4.** *Design a 3-bit counter which counts in the sequence: 001, 011, 010, 110, 111, 100, (repeat) 001, ...*

1 3 2 6 7 4

# of Inputs: ? = 0

# of States: ? = 6 states $6 \le 8 = 2^3$

Need 3-bit to represent 6 states

3-ff

Desired State table

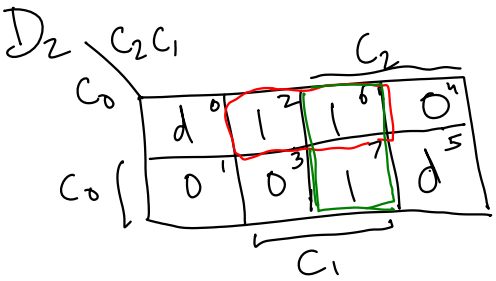
| | Input | Present state $C_2\ C_1\ C_0$ | Next State $C_2^+ = D_2\ \ C_1^+ = D_1\ \ C_0^+ = D_0$ | | |
|---|---|---|---|---|---|
| 0 | | 0 0 0 | d | d | d |
| 1 | | 0 0 1 | 0 | 1 | 1 |
| 2 | | 0 1 0 | 1 | 1 | 0 |
| 3 | | 0 1 1 | 0 | 1 | 0 |
| 4 | | 1 0 0 | 0 | 0 | 1 |
| 5 | | 1 0 1 | d | d | d |
| 6 | | 1 1 0 | 1 | 1 | 1 |
| 7 | | 1 1 1 | 1 | 0 | 0 |

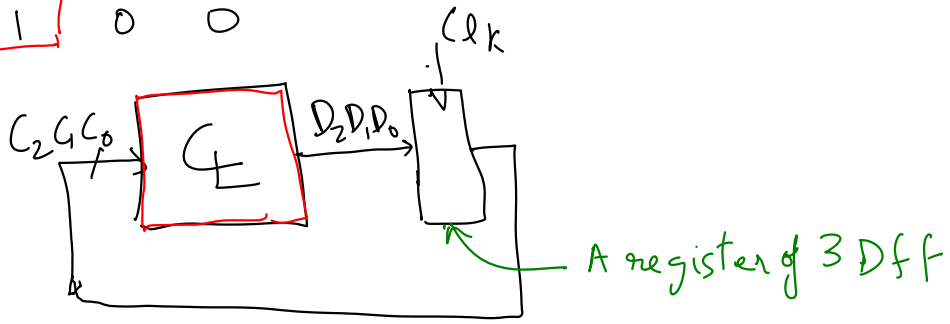
1 → 3 → 2 → 6 → 7 → 4

Using D-ff

$Q^+ = D$

$D_2\ D_1\ D_0$




$D_2 = C_1\bar{C_0} + C_2C_1$

$D_1$ 2 _____

$D_0$ = _____