- Σ, Λ, Q → finite sets
  δ, λ → functions from

$\delta: \Sigma \times Q \to Q$

$\lambda: \Sigma \times Q \to \Lambda$

→ Refer Class notes for Minimization of Mealy Machine.

Quiz Questions

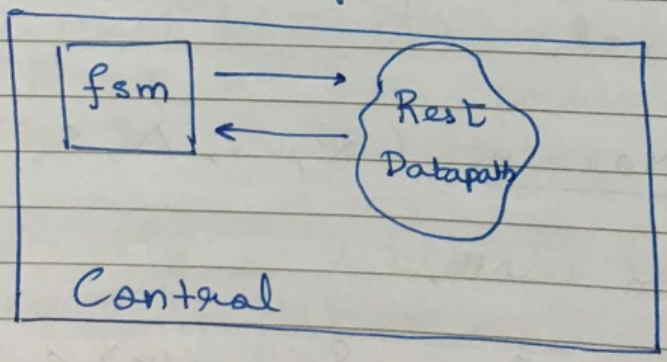
12.3.18

Part 2 Courses

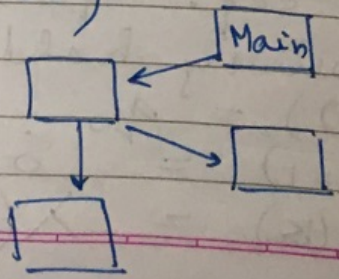
→ Recap ★ (Design)

- FSM · D flip flops
- RTL · memories — RTL = Register transfer level
- Turing Machines
- RAM based machines — Store & Retrieve information
- Test & Verification

→ FSMs are used for control sequences.

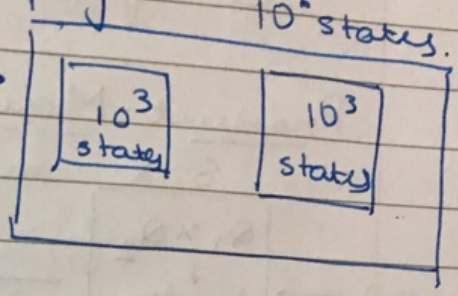


- FSM is like a program (Now onwards define it like so)



Advantages.

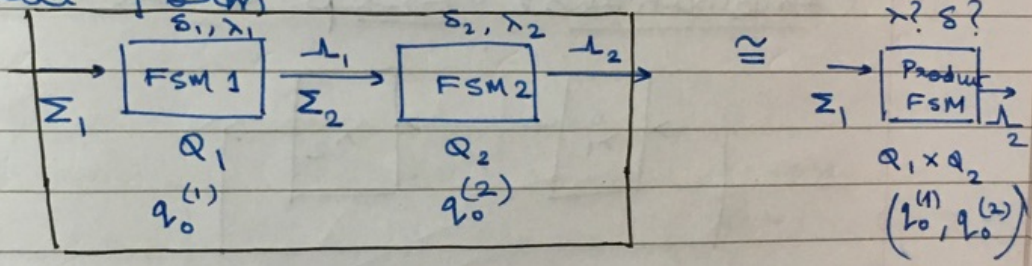
- Modularity
- Easy to debug
- ↓ size of prg.



- Practical,
  Concept,
  ↳ Design
  ↳ Verification

→ Complex FSMs from simpler FSMs

1) Cascade form



→ λ and δ of product FSM

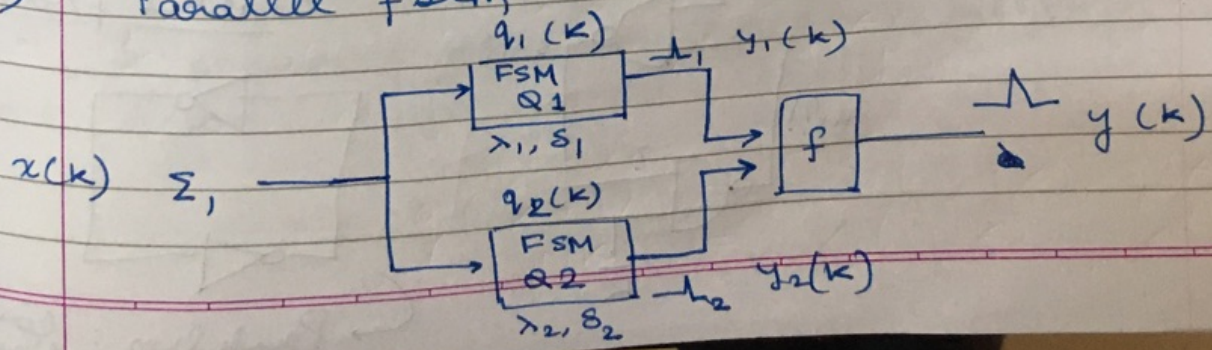
$$\delta((q_1, q_2), x) = (nq_1, nq_2)$$

↳ Next State function

$$= (\delta_1(q_1, x), \delta_2(q_2, \lambda_1(q_1, x)))$$

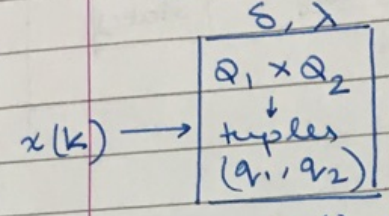
$$\lambda((q_1, q_2), x) = \lambda_2(q_2, \lambda_1(q_1, x))$$

2) Parallel form



- If $f \equiv XOR$, then Q_1 and Q_2 can be equivalent iff $f \neq 1 \; \forall \; \Sigma$

Product Machine



$RST \equiv (q_0^{(1)}, q_0^{(2)})$

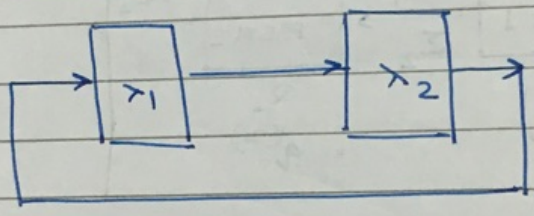
$\delta((q_1, q_2), x) = (\delta_1(q_1, x), \delta_2($

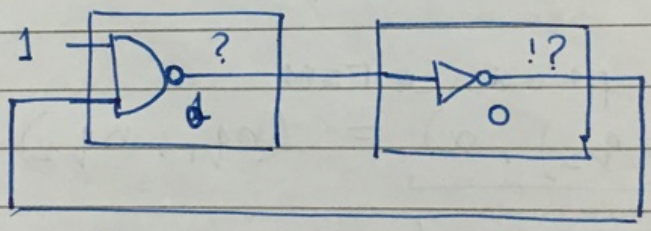
$\lambda((q_1, q_2), x) = f((\lambda_1(q_1), x), (\lambda_2(q_2),$

→ Can be thought as factoring the prob

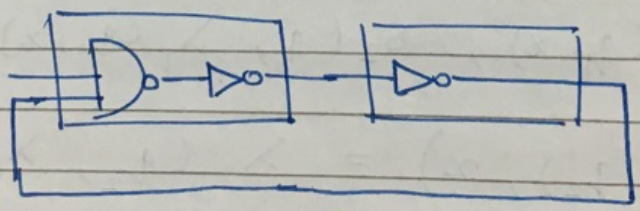
→ Problems

- Combinational loop & Latches / flip-flo

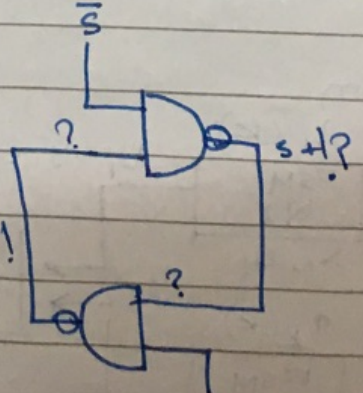




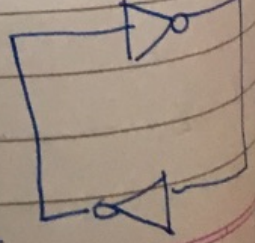
Vs.



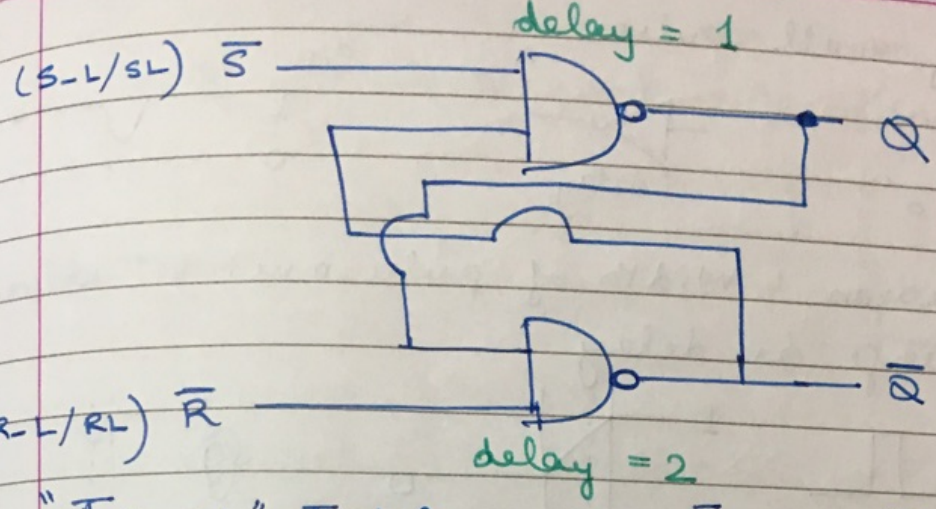
→ RS latch



Started fa



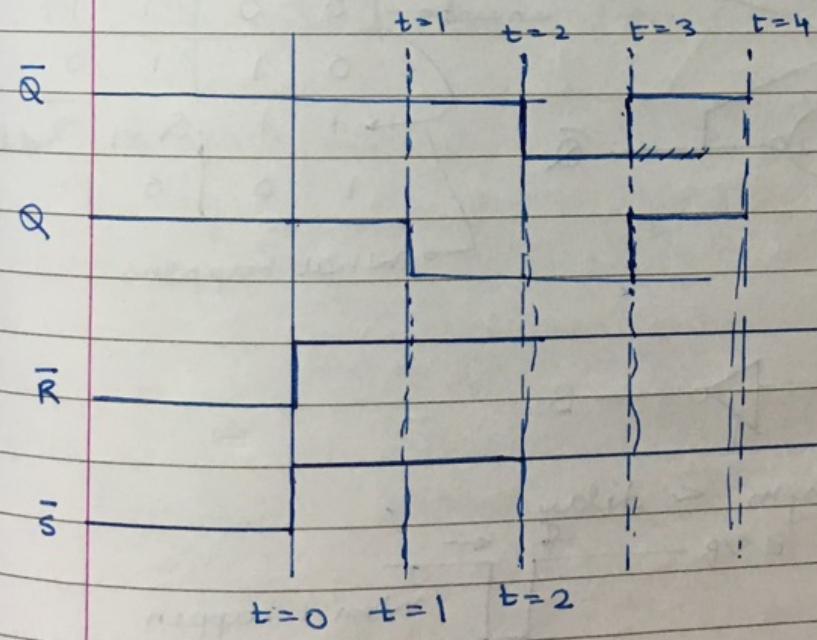
Problem



"Truth" Table

	$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$
Not Allowed! →	0	0	1	1
Set →	0	1	1	0
	1	1	Q_{old}	$\bar{Q}_{old}$
Reset	1	0	0	1

→ What happens when we start at (0,0)



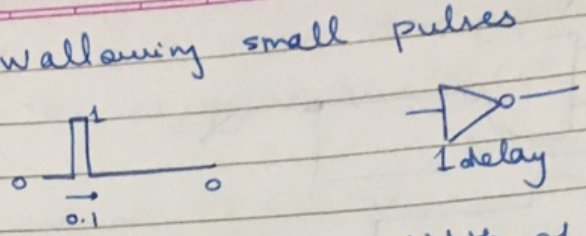
Cases

- $dt = 1$, $D = 2$
- $d = 1$, $D = 1.1$
- $d = 0.1$, $D = 1$

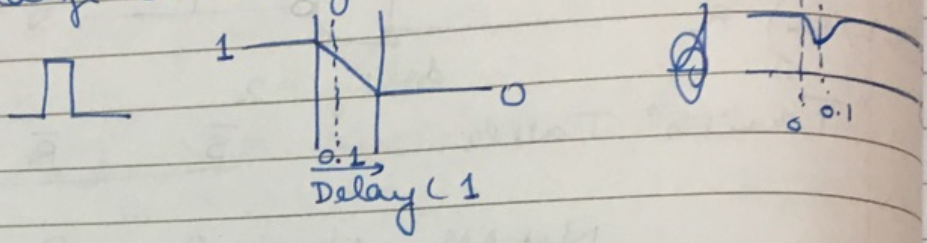
→ All oscillate?

→ In practice the ↑ ↓ ripple in voltages will die down (necessarily)

→ Swallowing small pulses



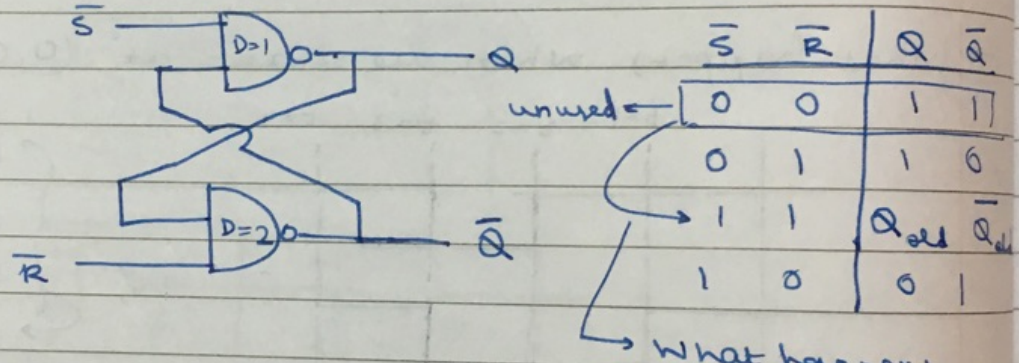
Why?, Reason : Width of pulse must be atleast as large as delay



→ Conclusion : Never use (0,0)

15.3.18

SR - bistable

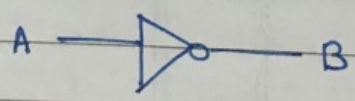


S̄	R̄	Q	Q̄
0	0	1	1
0	1	1	0
1	1	Q_{old}	$\bar{Q}_{old}$
1	0	0	1

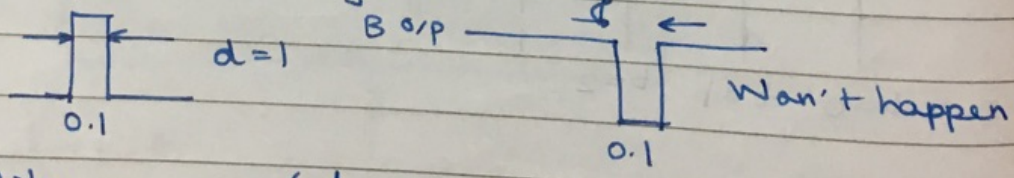
unused ←

→ What happens.

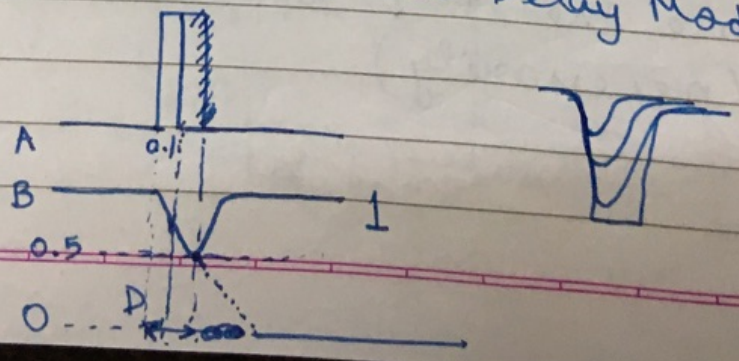
→ Delay Model



Small pulse of length < delay



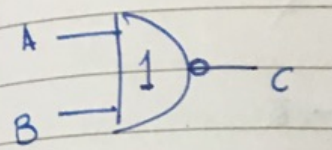
Reality (Inertial Delay Model)

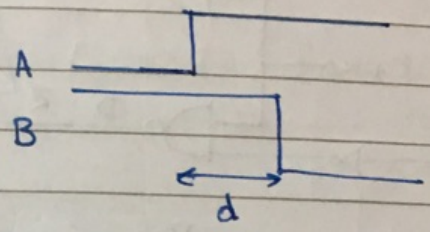


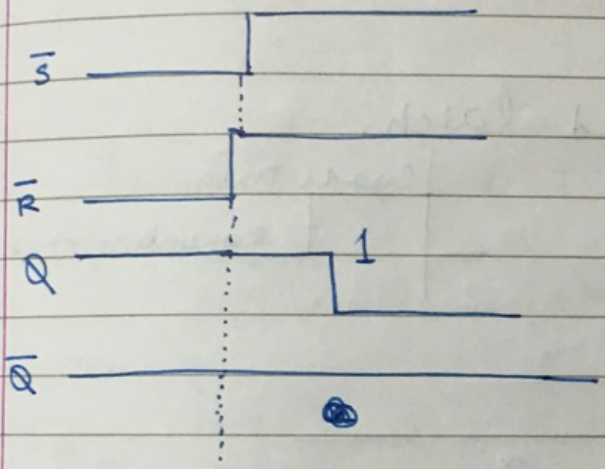
D ≡ Delay = time taken to reach 0.5 (say)

→ If i/p pulse is narrower than Gate Delay it will get swallowed (no effect) (d < D)

(d > D) → Effect





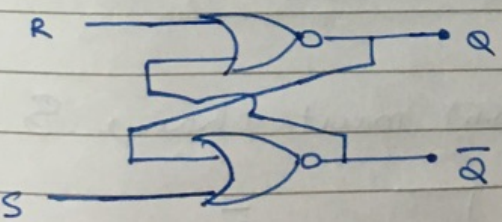


In our inertial delay model.

Doesn't change at t = 2 units, swallow swallows the change

→ If values were Identical, then can oscillate

· Another model for latch

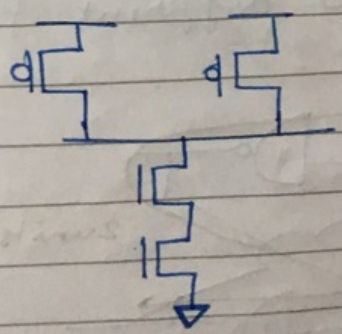


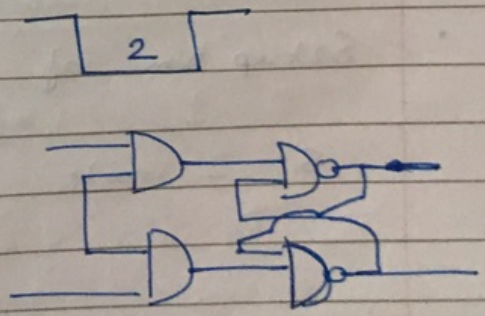
Analyze this latch

· NAND latch more imp. since faster than NOR (In CMOS only)

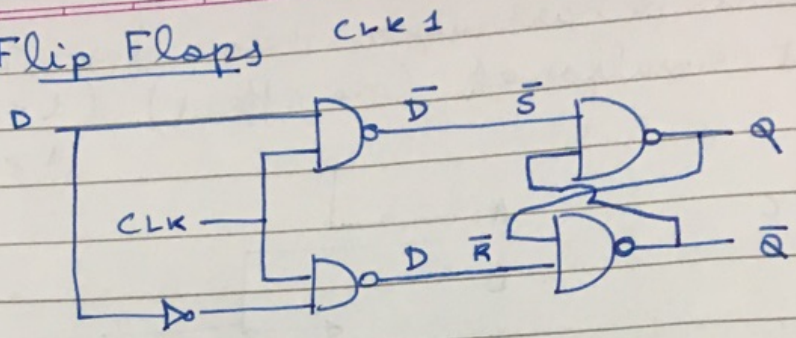
Why?

NAND





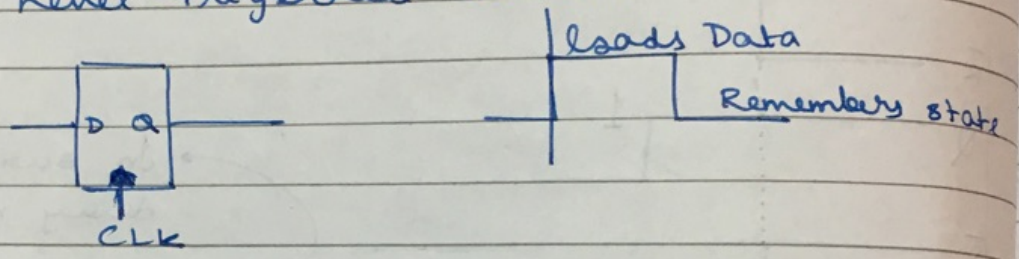
→ Flip Flops



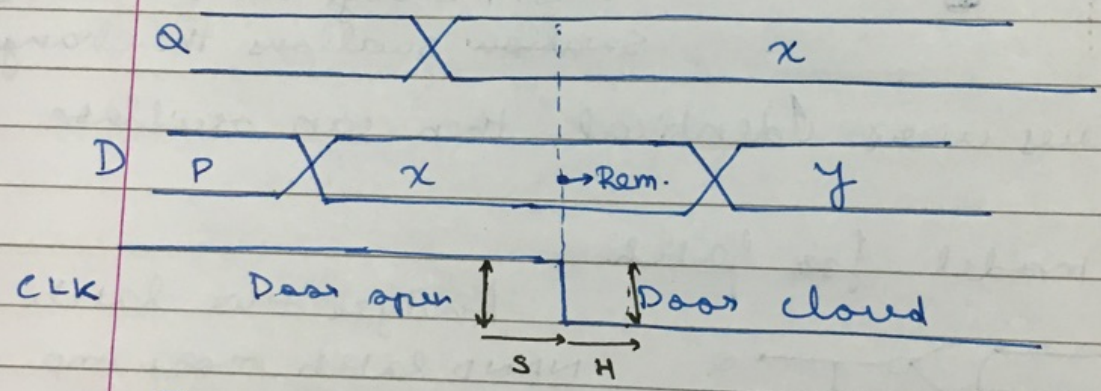
CLK 0 → Previous

CLK 1 → Q = D

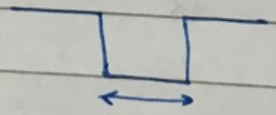
→ +ve level triggered latch.



19.3.1

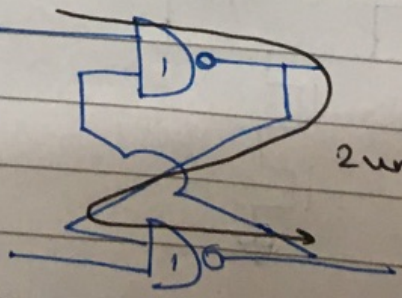


→ Final Data (last person) must change S before -ve edge.



• Setup time of 1 → 0 different from 0 → 1

S = 2 units.



time for 1-0 is d1+d2 but time for 0-1 is d1

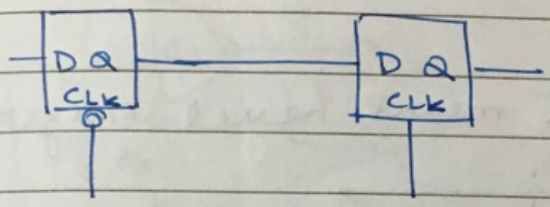
Hold time is defined as the minimum amount of time after the clock's active edge during which input data must be stable

→ When CLK low, $\bar{S}$ → 1 identically, hence, H = 0

. Setup time is defined as the minimum amount of time before the clock's active edge that the data must be stable for it to be latched correctly.

→ Negative level triggered latch

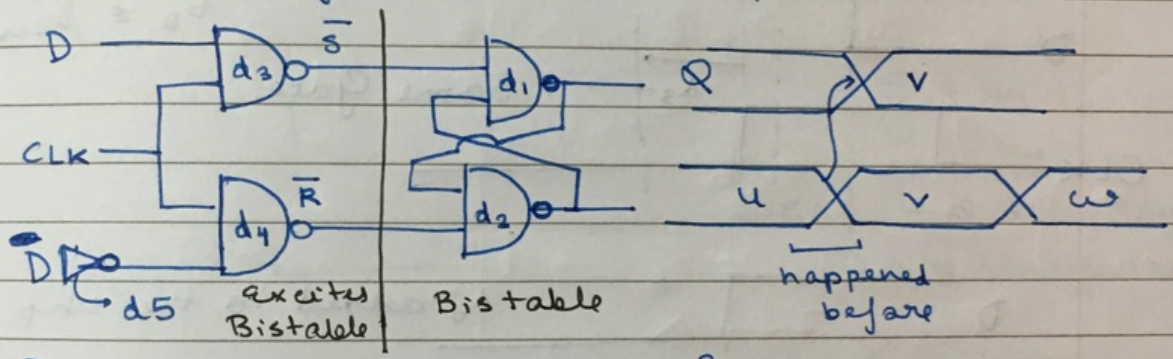
Replace CLK by CLK → ▷ → $\overline{CLK}$



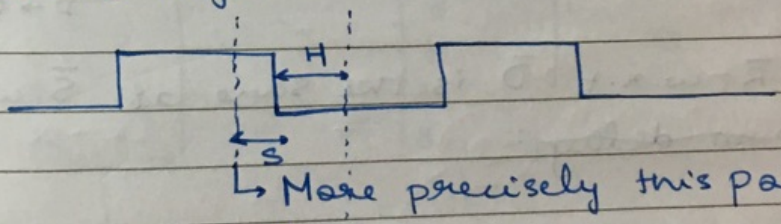
Flip Flop.

19.3.18

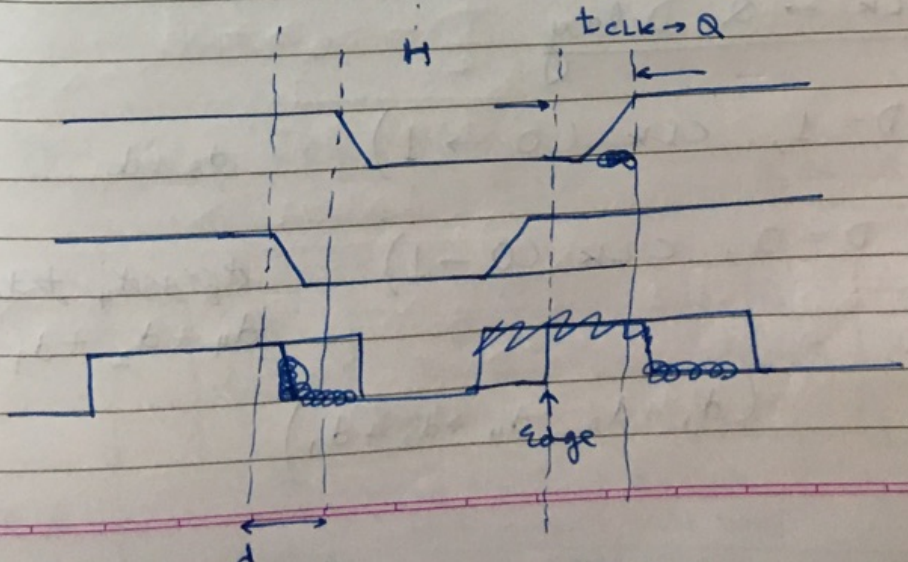
Level Triggered flip-flop



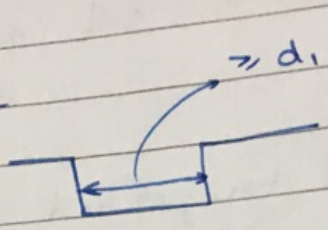
→ Data is sampled at the falling edge of the CLK since Q follows D only when clock is high. when data is sampled when clock is + then it is level triggered

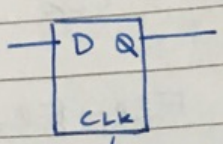


↳ More precisely this point

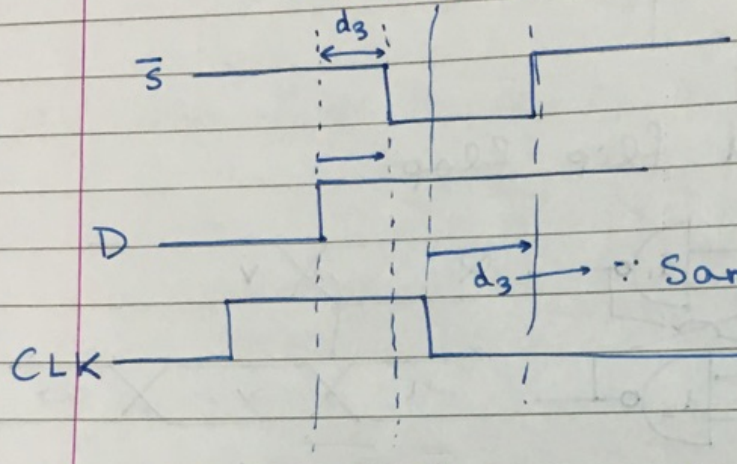


→ Hold time





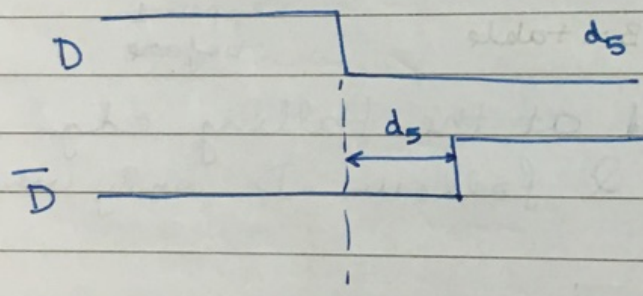
↳ No Δ means level triggered



$t_{CLK} - t_D \geq d_1 + d$

$t_D \leq t_{CLK} - (d$

∵ Same Gate



d_5 added to the setup time.

$S = d_1 + d_2 +$

$H = 0$

$d_{D \to Q} = (d_1 + d_3, d_$

→ Then $\bar{R}$ w.r.t $\bar{D}$ is the same as $\bar{S}$ w.r.t D

→ ~~Minimum delay~~:

CLK → Q Delay

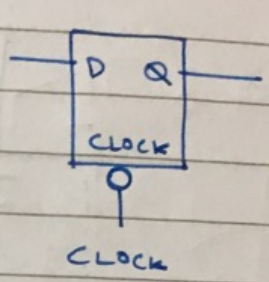
(i) D = 1, CLK (0 → 1) $d_3 + d_1$

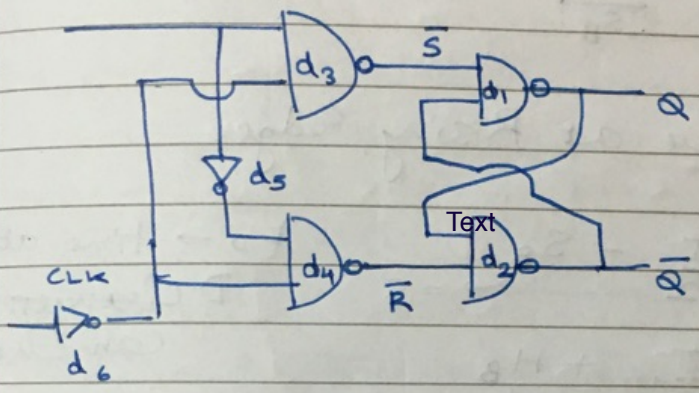
(ii) D = 0, CLK (0 → 1) ~~$d_3 + d_4 + d_2$~~ $d_4 + d_2 + d_1$

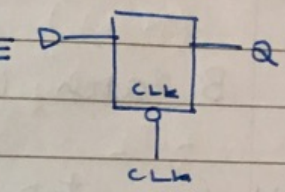
$(d_1 + d_3, d_4 + d_2 + d_1)$

20.3.

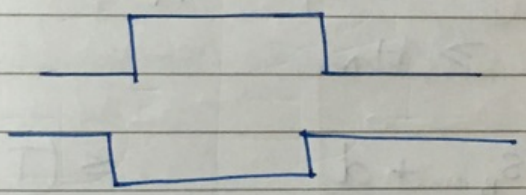
Another latch





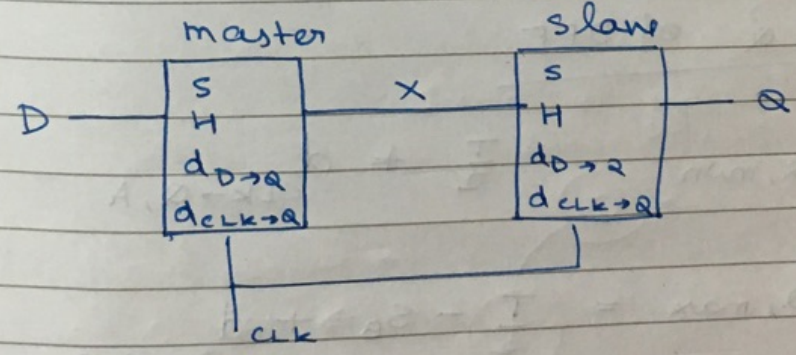


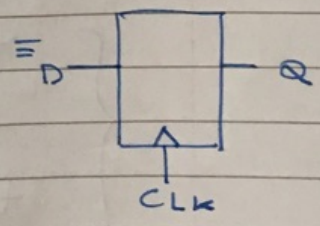
H : 0 → d_6



S ↓ by d_6
H ↑ by d_6
CLK → Q delay ↑ by d_6
D → Q Unaffected?

→ Edge triggered flip-flop (Master-Slave ~~latch~~) Flip-Flop



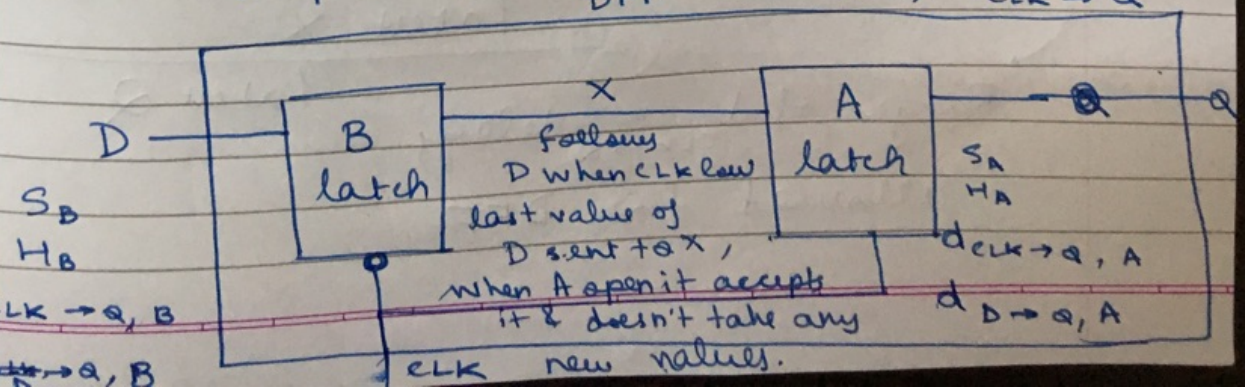


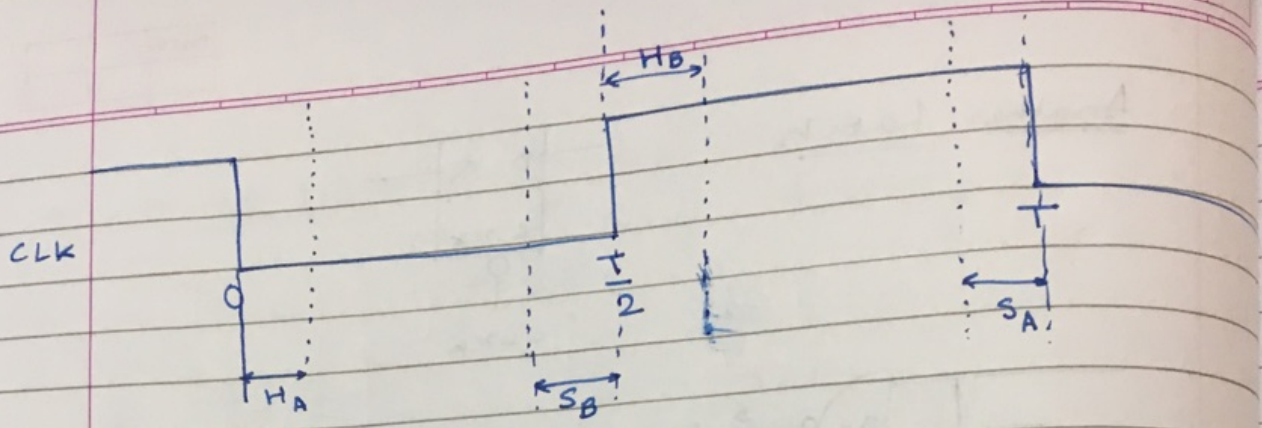
20.3.18

D-flip flop

DFF

S, H, $d_{CLK\to Q}$





B latch influence at Rising Edge.

S & H of DFF

$$t_D \leq \frac{T}{2} - S_B \rightarrow = S$$

$$t_D \geq \frac{T}{2} + H_B \rightarrow = H$$

$t_D \rightarrow$ time at which D ~~changes~~ can change

$$d_{CLK \rightarrow Q, B} \geq H_A$$

$$\frac{T}{2} - S_B + d_{D \rightarrow Q, B} \leq (T - S_A)$$

↳ correctness constraints.

→ $d_{CLK \rightarrow Q}$ of DFF

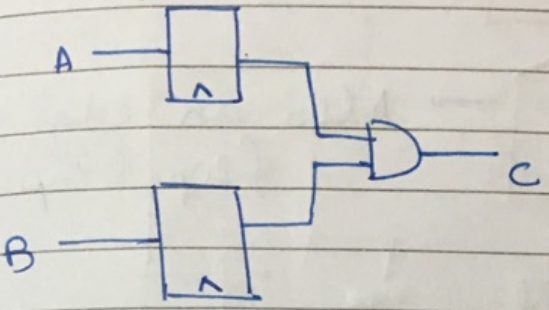
$$t_{Q,min} = \frac{T}{2} + d_{CLK \rightarrow Q, A}$$

$$t_{Q,max} = \underbrace{\frac{T}{2} - S_B}_{\text{latest D}} + \underbrace{d_{D \rightarrow Q, B} + d_{D \rightarrow}}_{\text{latest X}}$$

latest Q

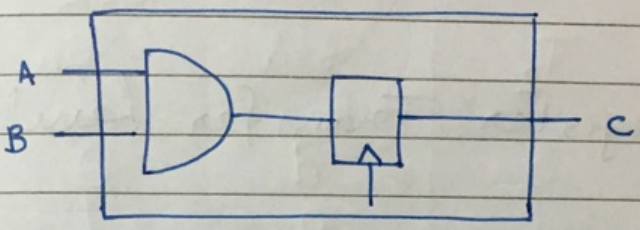
Range of $d_{CLK \rightarrow Q}$ delays allowed for DFF

Retiming



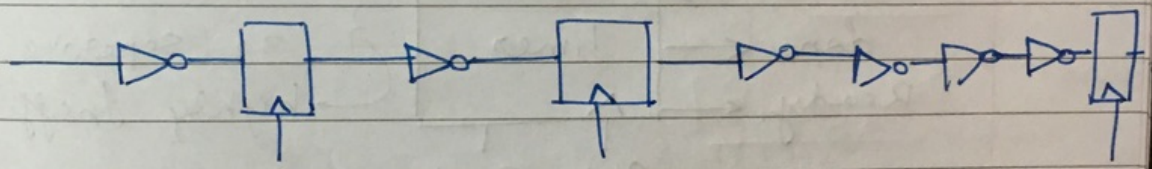
$C(k+1) = A(k) \cdot B(k)$

is the same as

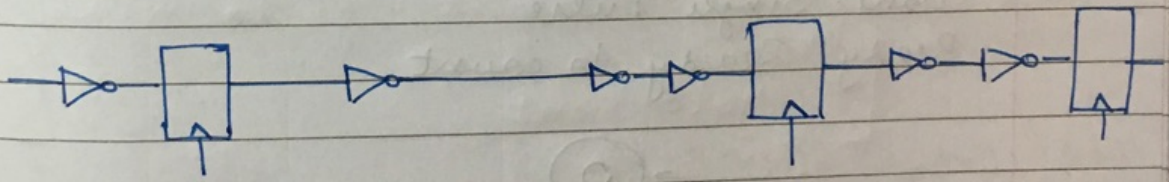


→ Slight Difference in Set-up times & delays

Q→ Minimum CLK period

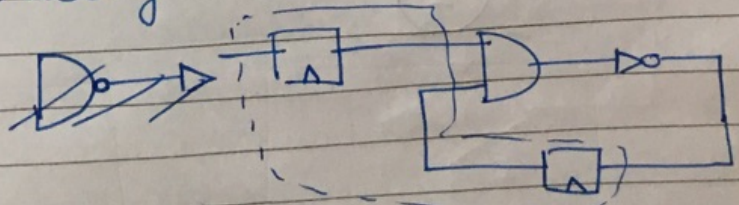


↓ same as

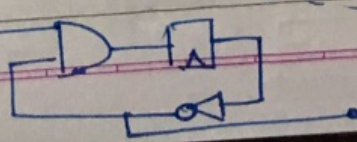


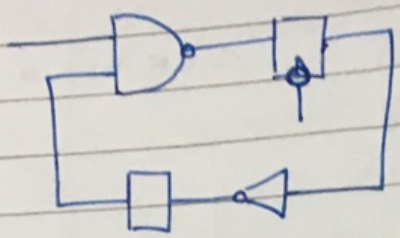
→ Idea: How much ~~time in each~~ delay in each clock cycle
↳ Determines T_{min}

- Ideally should be balanced out.



≡



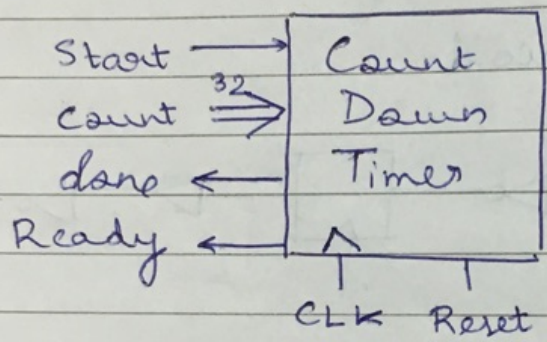


→ Also an edge triggered flip flop

20.3.18 (Night)

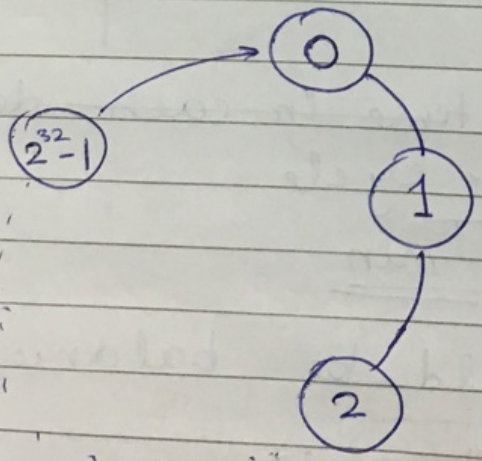
- RTL : Register Transfer Level machine

→ Count Down Timer

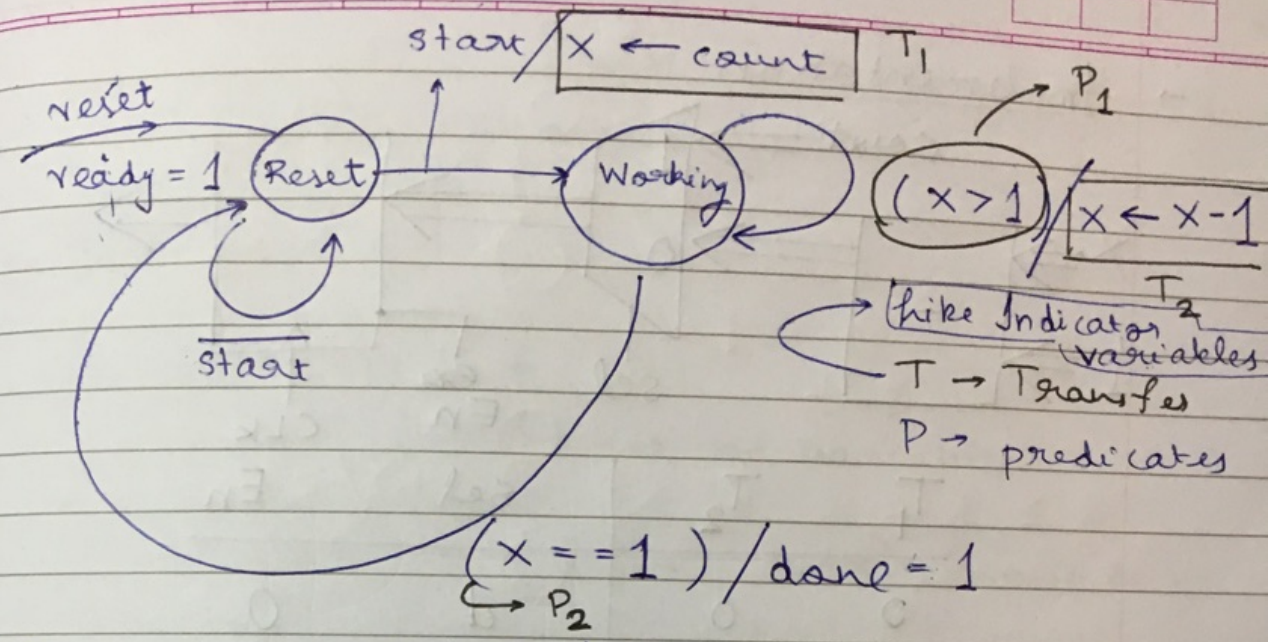


→ 32 bit counter
→ 32 bit number for 32 separate states
→ highly Inefficient

→ Done: Single Pulse
Ready: Ready to count



→ Highly Inefficient
⇒ use a counter variable like a program.

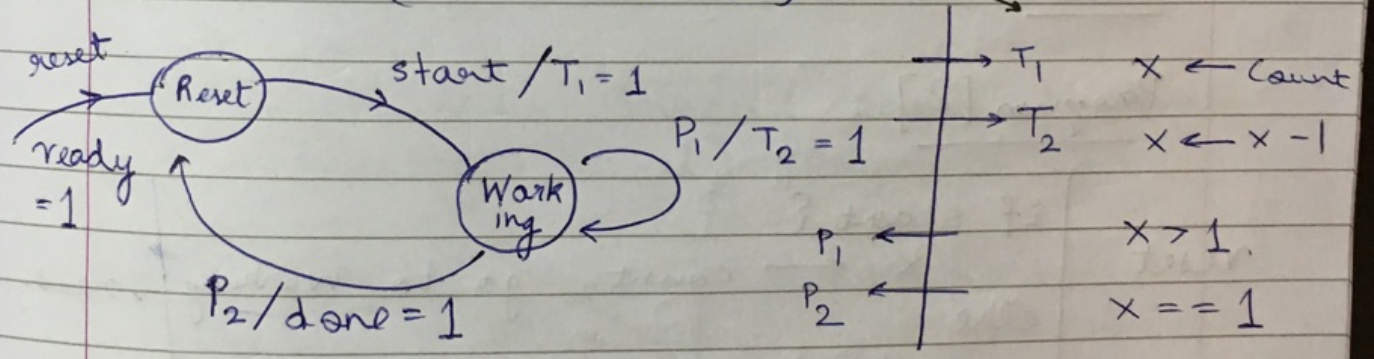


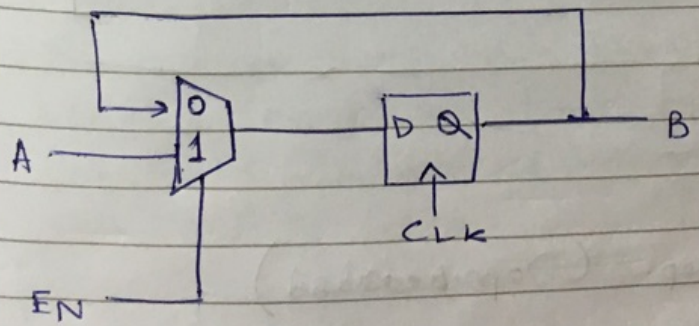
→ Predicates : ~~(X > 1) / X ← X - 1~~ ?

RTL

→ Above system is implemented through Registers

(X ← count)

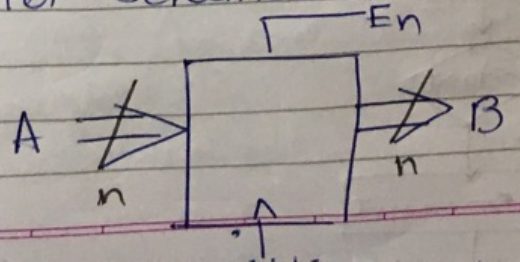




B(k+1) = (En(k) ? A(k) : B(k))

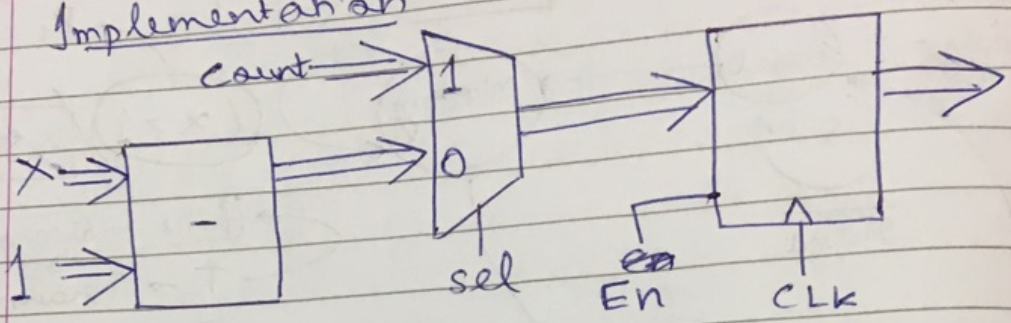
B ← A if En = 1

→ Register Schematic



: Register n bit

→ Implementation



T_1	T_2	sel	En
0	0	d	0
0	1	0	1
1	ϕ	d	d
1	0	1	1

22.3

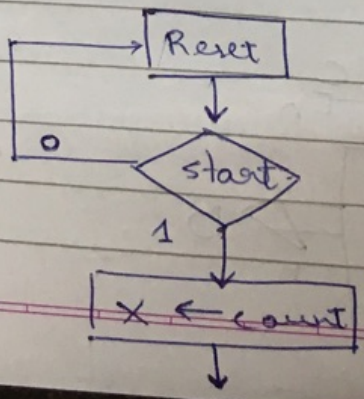
→ Code

Program [?]

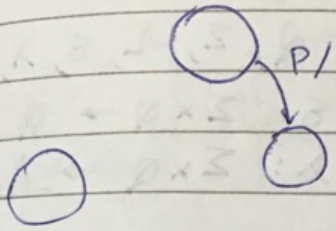
reset
```
if start {
    x ← count, go to working
else
    goto reset
```

Working
if ...
...

→ Flow Chart Rep. (Deprecated)



$$x(k+1) = f(x(k) \dots)$$



$$x(k+1) = f(x(k), \dots)$$

↳ set of variables

$$x(k) = g(q(k), x(k), \text{input}(k))$$

22.3.18

FSM

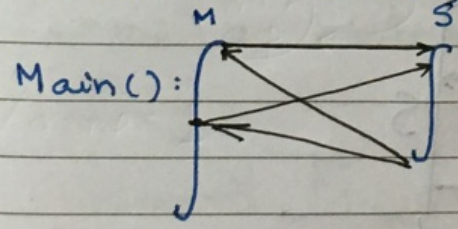
Q states

RTL

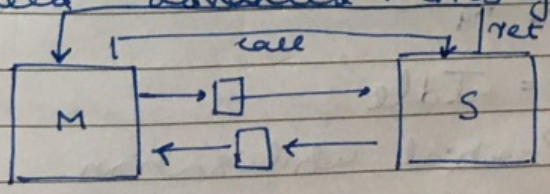
Q × { set of values in registers }

→ Predicate: o/p 1/0, domain can be anything.

→ Now onwards: thread based ~~Multiple~~ (function based) programming.



→ Recursion at this stage is not possible, we would need to emulate a software stack & need "advanced memory"

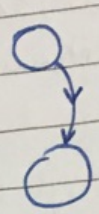


→ M & S are two pieces of hardware running concurrently (Though only 1 is active at a time)

26.3.18

Recap

FSM



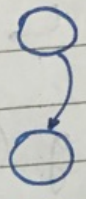
$Q, \Sigma, \Lambda, \delta, \lambda, q_0$

$\delta: \Sigma \times Q \to Q$

$\lambda: \Sigma \times Q \to \Lambda$

↓

RTL



$P(R) / R \leftarrow f(x, R)$

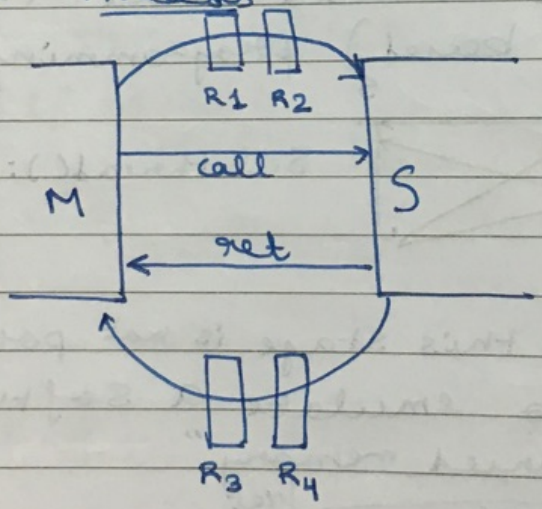
$Q, \Sigma, \Lambda, R, \delta_Q, \delta_R, \lambda$

$\delta_Q: \Sigma \times Q \times R \to Q$

$\delta_R: \Sigma \times Q \times R \to R$

→ Advantage: We don't care about most of the internal states of the register

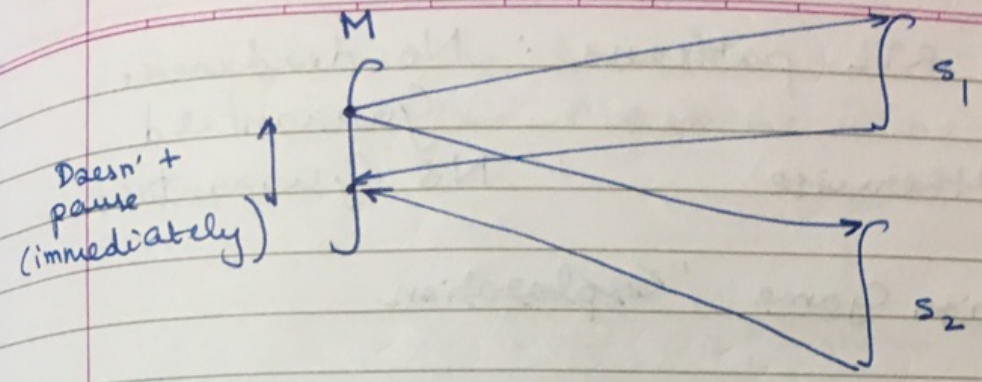
→ RTL Threads → Sequential program.



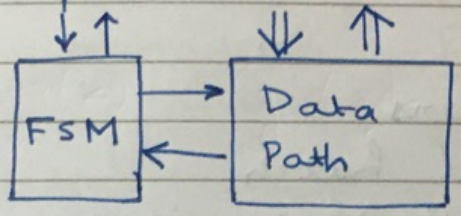
→ $ state := Idle

→ Problem: Partial utilisation

Fork - Join Structures



→ Threads imp.

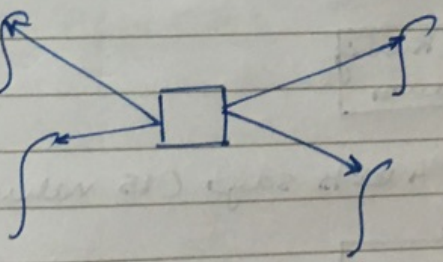


(Logic gates & flip flops / Registers)

→ Metrics of performance:
Internal : % Utilisation
External : Power, time, etc.

→ Utilisation is still low.

→ If there are many slaves, then Master forms a bottle-neck



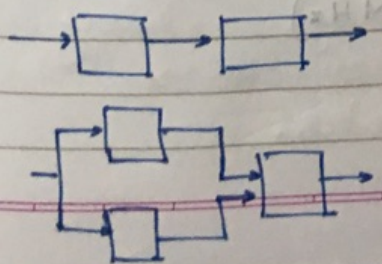
Peer / Synchronised thread

→ Utilisation is maximum

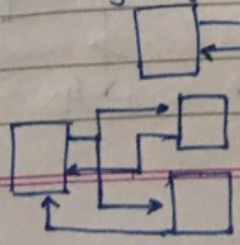
Paradigms

RTL

FSM (No deadlocks)



(Deadlocks possible, since 2 way inf. exchange)



→ Std. RTL patterns : No deadlocks Guaranteed

Otherwise : No Guarantees.

27.3.18 Reaction Game "explanation

→ Difficulties :

1) Random Numbers

option 1) DFF / latch

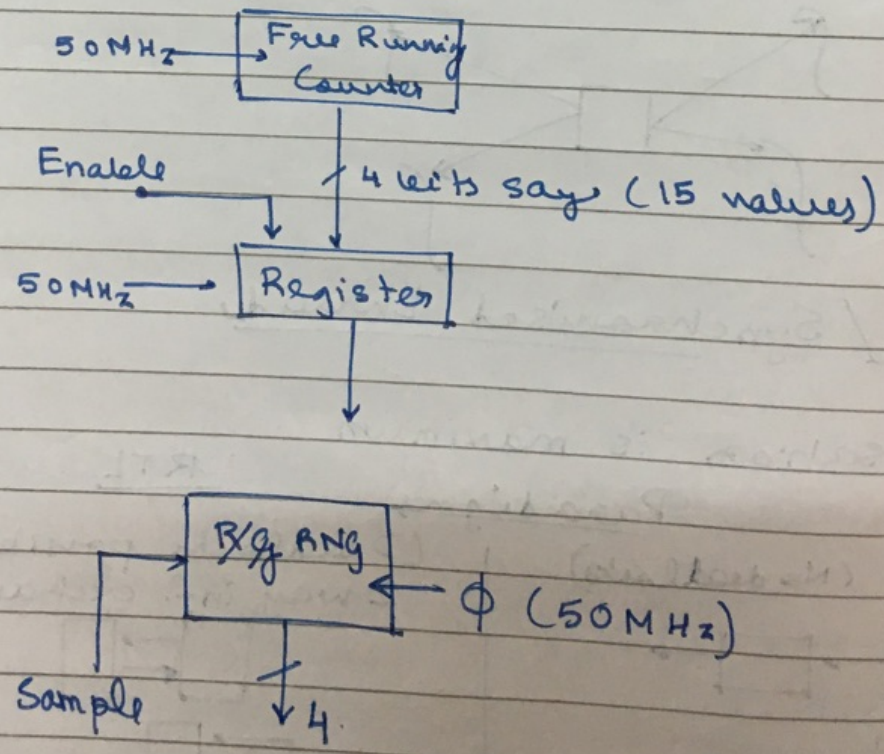
→ Restarts in same state when it switches on → problem

2) Noise in Resis. → Involves I-V measurements

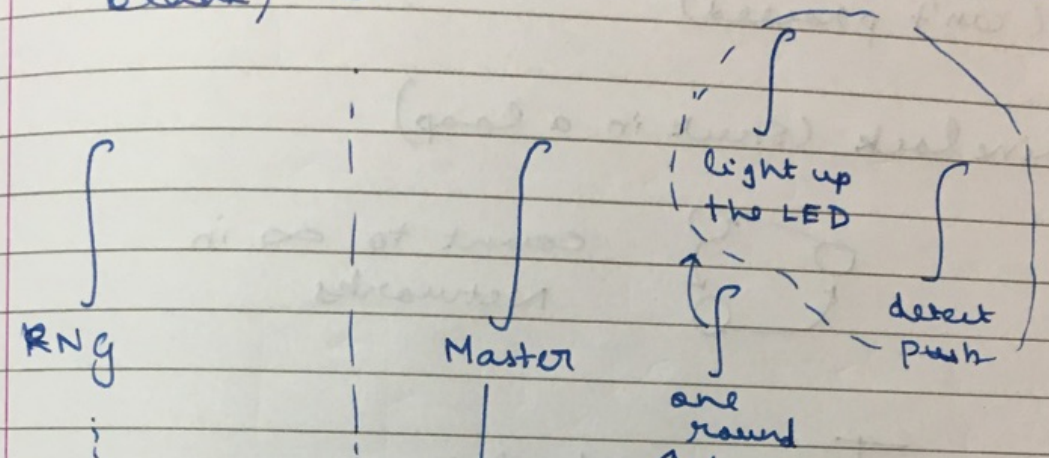
3) PRN : very large time periods, locally random.

→ Too complicated ↳ ??

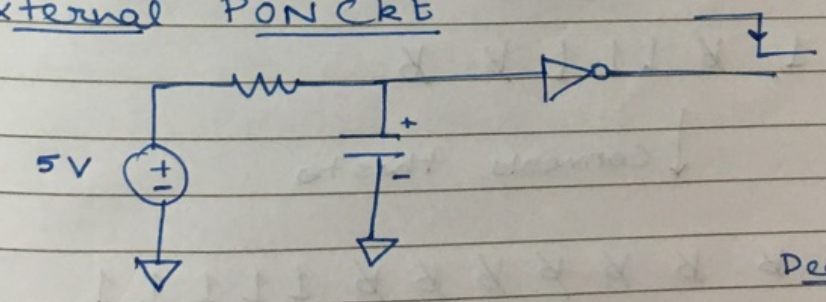
4) Free Running Counter



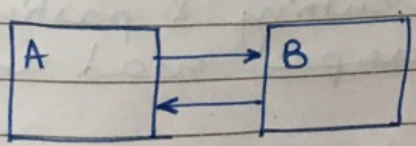
→ Ckt Reset → Another reset to turn on/off the Machine (So that screen is blank)



External PON Ckt

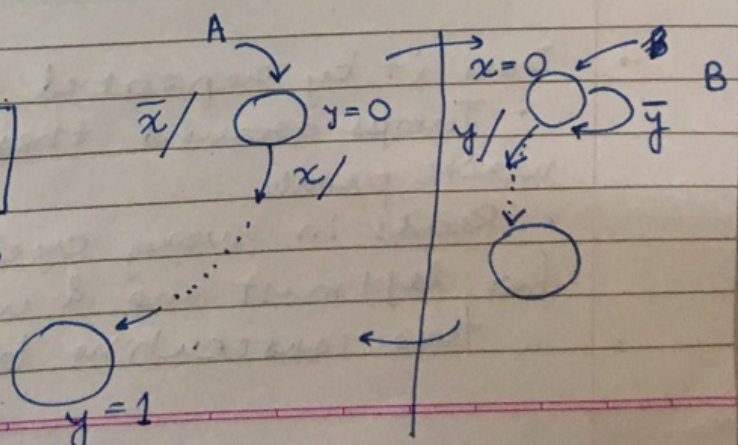


Deadlocks



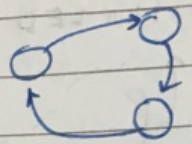
→ Individual, work fine
Join : Problem

Dead lock ex.



→ Deadlock: State with no exit (can't proceed)

→ Livelock (stuck in a loop)

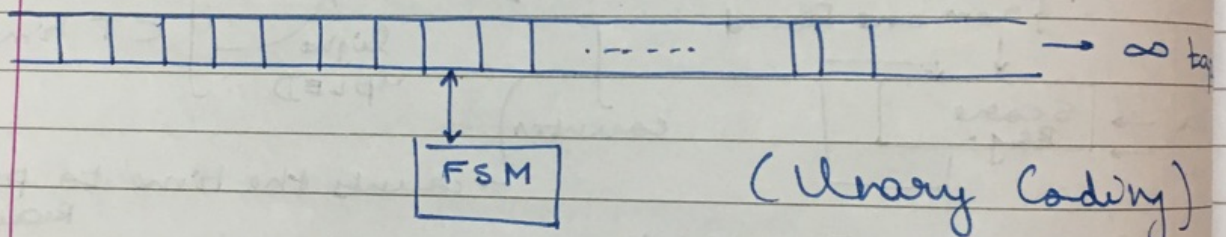


count to ∞ in Networks

Turing Machines

→ Characterize problems that can be solved by a computer.

tape Model



(Unary Coding)

b 1 1 b 1 1 1 b b

↓ converts this to

b b b b b b b b b 1 1 1 1 1

→ Does it by repeated cutting & pasting.
- Jumps around ~~these pair~~ read and write point.
- Reads in every cycle by looking for leftmost one & writes after it sees two consecutive blanks.