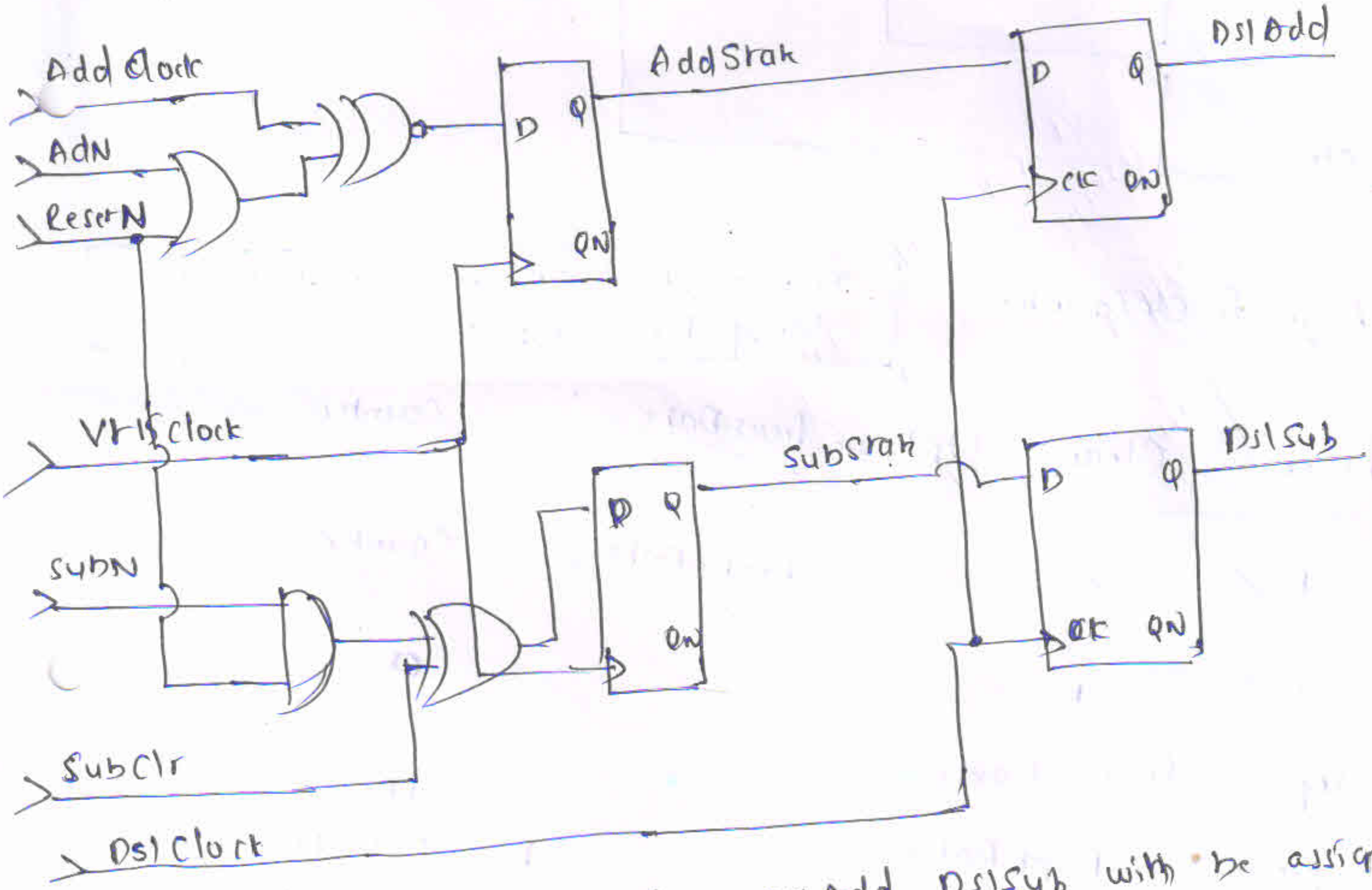
Lecture 23

Multiple clocks

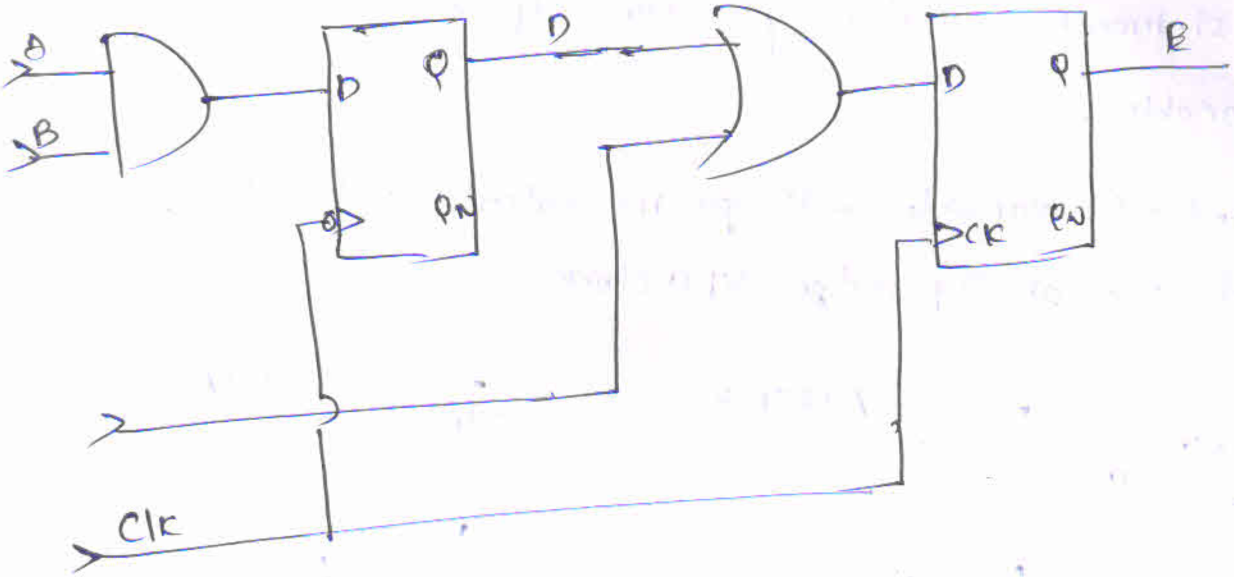
1) AddStart & SubStart are clocked under @ posedge VrisClock. Therefore you can expect FFs for both of always statement. these variables.

2) Left hand side variable will get the calculated value of right hand side at @ posedge Vris clock.



3) Similarly, Dsl clock posedge DslAdd, DslSub will be assigned with values of AddStart & SubStart .. DslAdd & DslSub will also synthesize as FF with @ posedge Dsl Clock.

Multi-phase clocks



Async Pre Clr Counter

~~You will find one more method of doing the same thing.~~

Preset	Clear	~~UpDown~~	Preset Data	Counter
1	0		PresetData	Counter
0	1		2	0

step 1: Preset Logic

preset	Preset Data		PD Preset Data pins at FF	
	1	0	1	0
0	0	0	don't case	
	0	1		
	1	0		
	1	1		
1	0	0	0	0
	0	1	0	1
	1	0	1	0
	1	1	1	1

dont care reny

②

PD [1]

preset \ presetData 1/0	00	01	11	10
0	X 0	X 1	X 3	X 2
1	4	5	1 7	1 6

PD [0]

0	X 0	X 1	X 3	X 2
1	4	1 5	1 7	6

PD [1] = preset . PresetData [1]

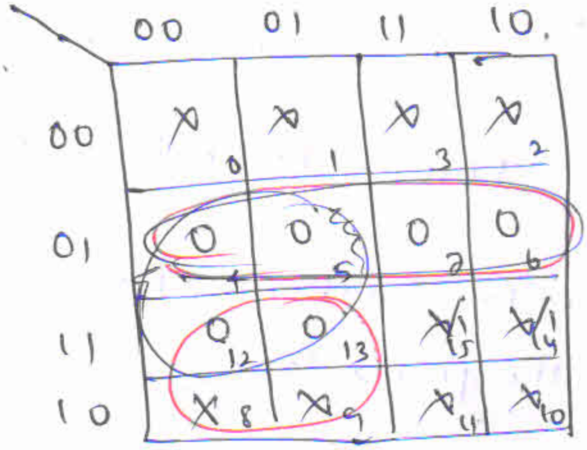
PD [0] = Preset . PresetData [0]

step 2 : finding the CD (clear Data Pin equations). Please note CD has dependency on Preset (since the priority is to PC).

preset	clear	PresetData 1	PresetData 0	clear Data (CD) 1	clear Data (CD) 0
0	0	0	0	X don't care condition (minterms)	
		0	1		
		1	0		
		1	1		
0	1	0	0	0	0
		0	1	0	0
		1	0	0	0
		1	1	0	0
1	0	0	0	X don't care condition (min terms) taken care by Preset	
		0	1		
		1	p		
		1	1		
1	1	0	0	0	0
		0	1	0	1
		1	0	1	0
		1	1	1	1

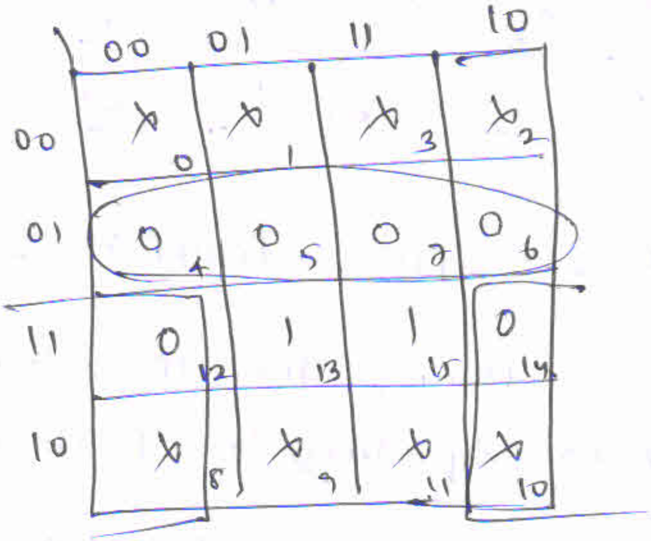
(d)

CD[1]



	00	01	11	10
00	X 0	X 1	X 3	X 2
01	0 4	0 5	0 7	0 6
11	0 12	0 13	X/1 15	X/1 14
10	X 8	X 9	X 11	X 10

CD[0]



	00	01	11	10
00	X 0	X 1	X 3	X 2
01	0 4	0 5	0 7	0 6
11	0 12	1 13	1 15	0 14
10	X 8	X 9	X 11	X 10

we need to simplify for zero's and take inversion / need to simplify for max terms.

$$CD[1] = \overline{\overline{Preset}.\,clear + Preset.\,\overline{PresetData[1]}}$$

$$= (\overline{clear} + Preset)(\overline{Preset} + PresetData[1])$$

$$CD[0] = \overline{\overline{Preset}.\,Clear + Preset.\,\overline{PresetData[0]}}$$

$$= (\overline{clear} + Preset)(Preset + PresetData[0])$$

CD[1], CD[0] expression are for when CD[1], CD[0] become zero. However, we want whenever we have CD[1] = 1 / CD[0] = 1 we want FF output to zero., so, we have to connect $\overline{CD[1]}$ and $\overline{CD[0]}$ to FF inputs.

~~#~~# UpDown should be connected to SD pin of FF.

we need to derive D1, D0 input equations for FF.

UpDown	current state		Next state (counter)			
			1		0	
			D_1	D_0	D_1	D_0
0	0	0	Ⓐ* x/0	1	x/0	1
0	0	1	x/0	0	x/0	0
0	1	0	x/0	0	x/0	1
0	1	1	x/0	1	x/0	0
1	0	0	0	0/x	1	0/x Ⓑ*
1	0	1	1	0/x	0	0/x
1	1	0	1	0/x	1	0/x
1	1	1	0	0/x	0	0/x

Ⓐ* 0, 1, 2, 3 terms are don't care for D_1

Ⓑ* 4, 5, 6, 7 terms are don't care for D_0.

Counter

NS[1]

D_1	00	01	11	10
0	x_0	x_1	x_3	x_2
1	0_4	1_5	0_7	[illegible] $_6$

$$D_1 = \overline{CS[1]}.CS[0] + CS[1].\overline{CS[0]}$$

NS[1]

D_0

	00	01	11	10
0	1	0	1	0
1	x	x	x	x

$$D_0 = \overline{CS[1]}\,\overline{CS[0]} + CS[1].CS[0]$$

NS[0]

D_1	00	01	11	10
0	x_0	x_1	x_3	x_2
1	1_4	0_5	0_7	1_6

$$D_1 = \overline{CS[0]}$$

	00	01	11	10
0	1_0	0_1	0_3	1_2
1	x_4	x_5	x_7	x_6

$$D_0 = \overline{CS[0]}$$

Async flip-flop :→ I method → you will find one more method. ④

```
always @ (negedge Reset or negedge Set or
                                negedge clkA)
  if (! Reset)
        Nextstate <= 12; → 1100 ✓
  else if (! Set)
        Nextstate <= 5; → 0101 ✓
  else
        Nextstate <= Currentstate;
```

1> FFs with Preset/clear will be used.

2> It is to be noted that

Nextstate[0], Nextstate[3] ⇒ FF will have both Preset and clear terminals.

Nextstate[1] ⇒ FF will have only clear terminal since '0' is assigned under both the conditions.

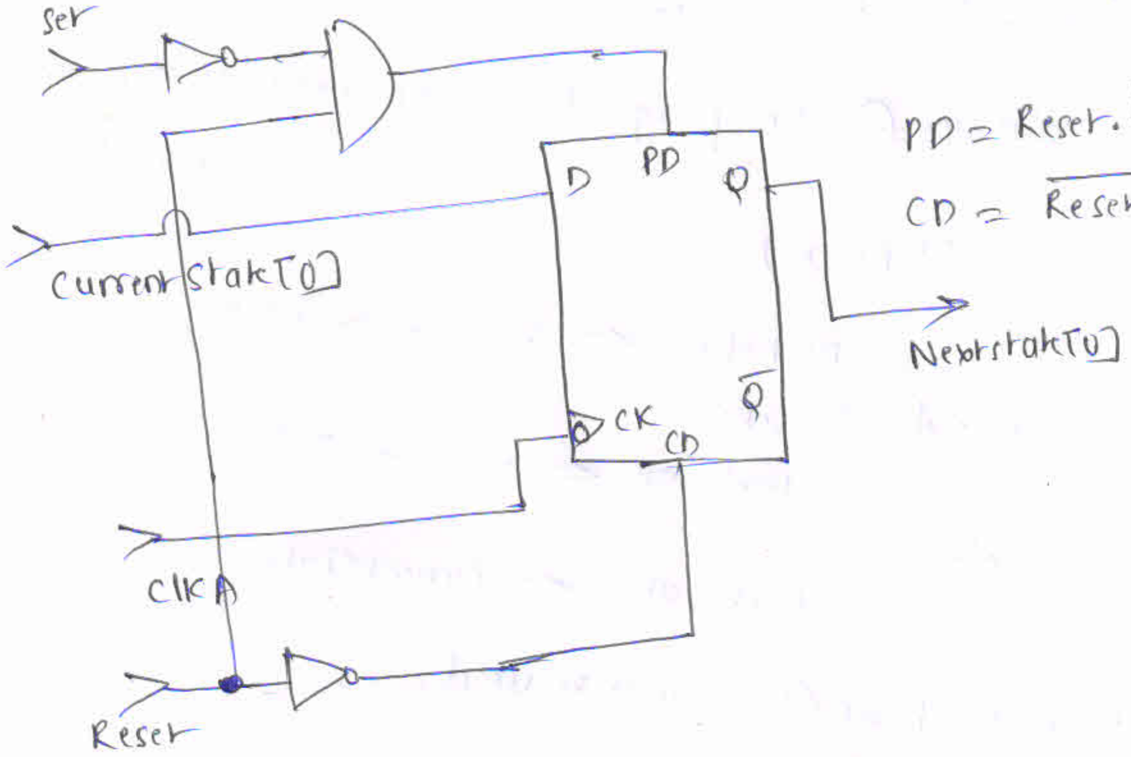
Nextstate[2] ⇒ FF will have only Preset terminal since '1' is assigned under both the conditions.

1100
0101

Reset	Set	Nextstate[0]
0	0	0
0	1	0
1	0	1
1	1	→ this condition take care by clkA.

PD = $\text{Reset}.\overline{\text{Set}}$

CD = $\overline{\text{Reset}}$



$PD = Reset \cdot \overline{Set}$

$CD = \overline{Reset}$

Nextstate[1]

1100
0101

Reset	Set	Nextstate[1]
0	0	0
0	1	0
1	0	0
1	1	→ ClkA takes care of this condition

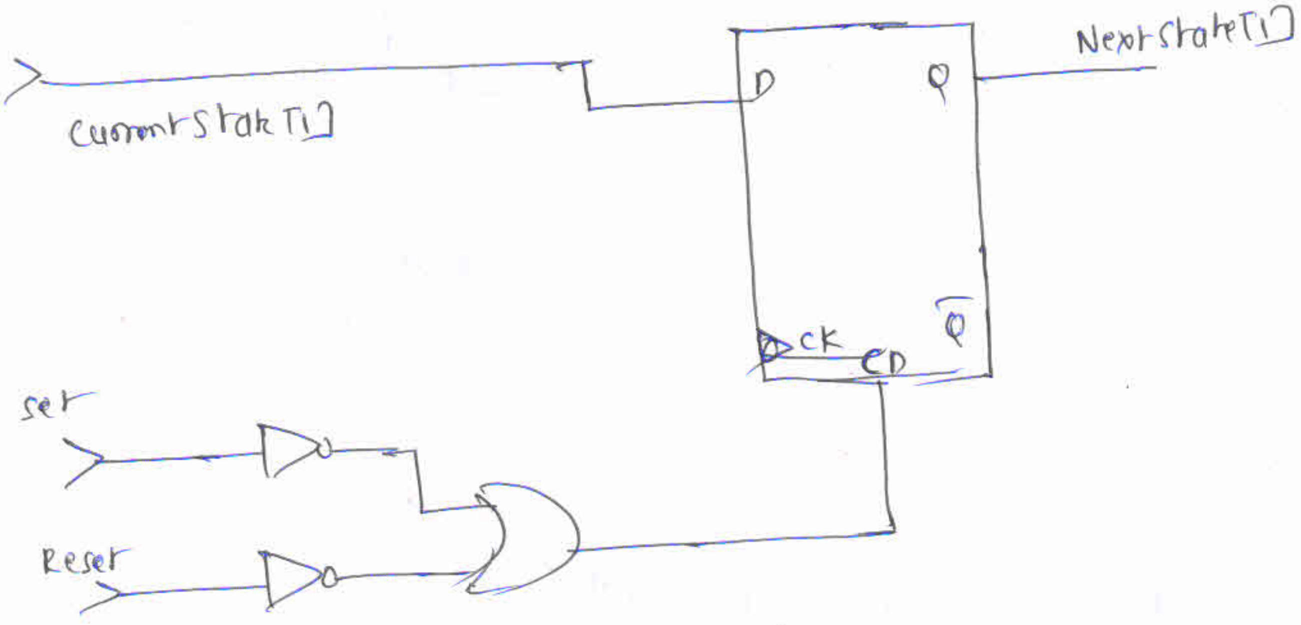
⟹ all on 0's ⟹ only cp pin needed.

	0	1
0	0	0
1	0	

$CD = \overline{Set} + \overline{Reset}$

~~However, [illegible]~~
~~[illegible]~~

⟹



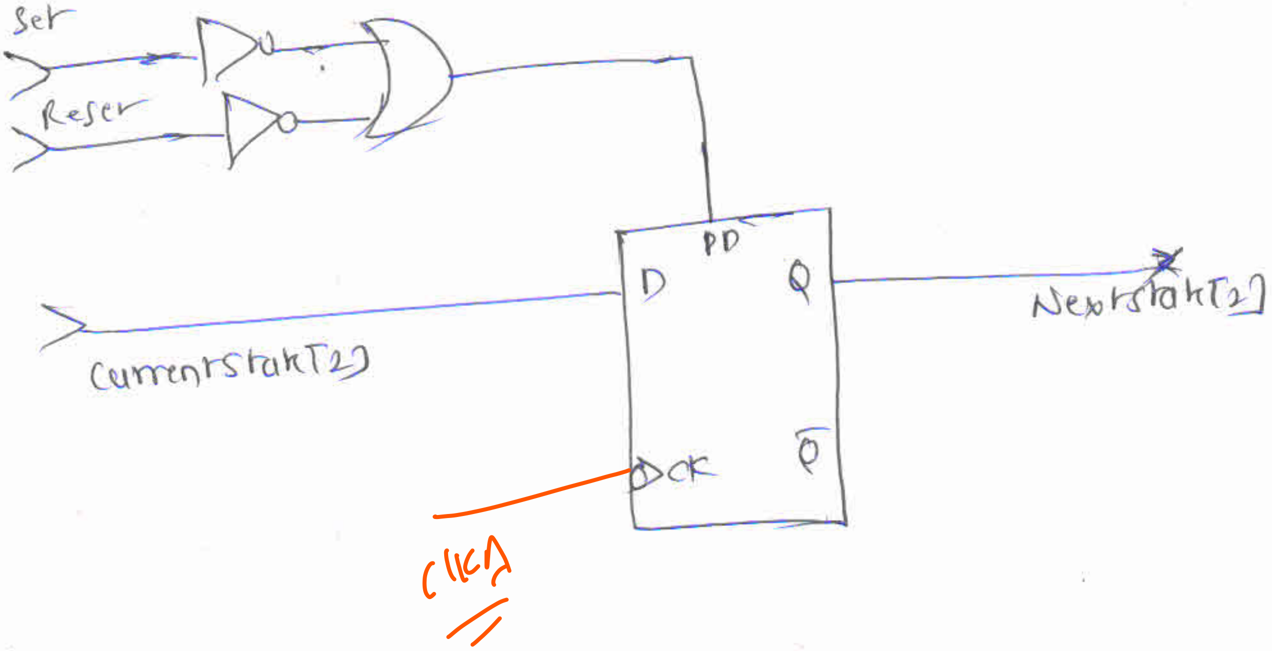
Nextstate[2]

Reset	Set	Nextstate[2]
0	0	1
0	1	1
1	0	1
1	1	→ clkA takes care of this condition

⇒ all are '1's ⇒ only PD pin needed.

PD	0	1
0	1	1
1	1	

$PD = \overline{Set} + \overline{Reset}$



Nextstate[3]:

Reset	Set	Nextstate[3]	
0	0	1	PD logic
0	1	0	
1	0	0	CD Logic
1	1		

1100
0101

	0	1
0		
1	0	

CD = Reset . $\overline{Set}$

CD = Reset . $\overline{Set}$

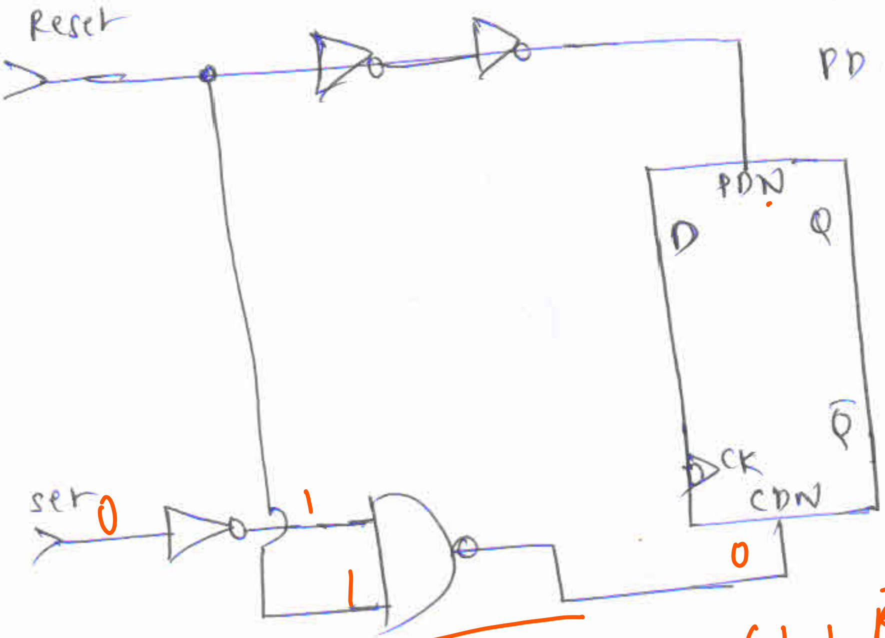
PD =

	1	0
0	1	1
1		

PD = ! Reset

PDN = $\overline{!Reset}$ = Reset

please note $\frac{PDN}{CDN}$ (*)



$\overline{\overline{Set} \cdot Reset} = Set + \overline{Reset}$

Async Flip_Flop:.

I-Method-

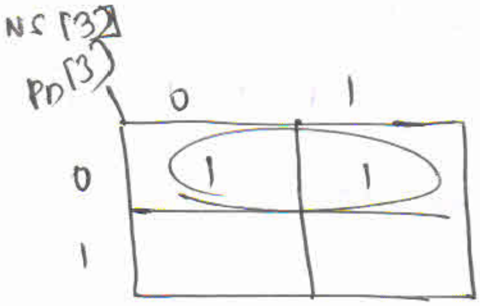
working out the equations for PD & CD inputs. In this ~~Reset~~ particular example ~~set~~ Rreset/clear both depend on the Reset & Set.

NS

		PD				CD			
Reset	Set	3	2	1	0	3	2	1	0
0	0	1	1					0	0
0	1	1	1	Ⓐ			Ⓑ	0	0
1	0		1		1	0		0	
1	1								

Ⓐ Represent Nextstate [1] FF do not need Preset pin.

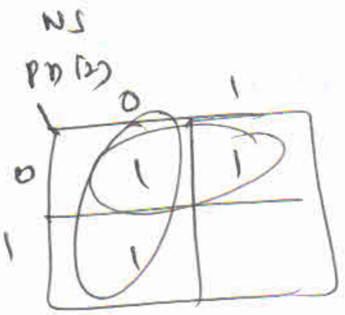
C Represent/indicate that Nextstate [2] FF do not need CD pin.



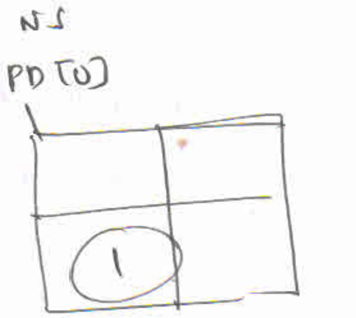
$PD[3] = \overline{Reset}$

~~[illegible]~~

~~RR[3] = Reset~~



$PD[2] = \overline{Set} + \overline{Reset}$



$PD[0] = Reset.\overline{Set}$

NS⁻ PD[1] not needed

NS

CD[3]

	0	1
0		
1	0	

CD[3] = $\overline{Reset}$ + Set

we need to take inversion of n-D & connect it to CD[3] pin

CD[3] = $\overline{Set + \overline{Reset}}$

CD[2]

No-need

CD[1]

	0	1
0	0	0
1	0	

CD[1] = Set + Reset

we need to take complement of it & connect it to CD[1] pin.

CD[1] = $\overline{Set . Reset}$

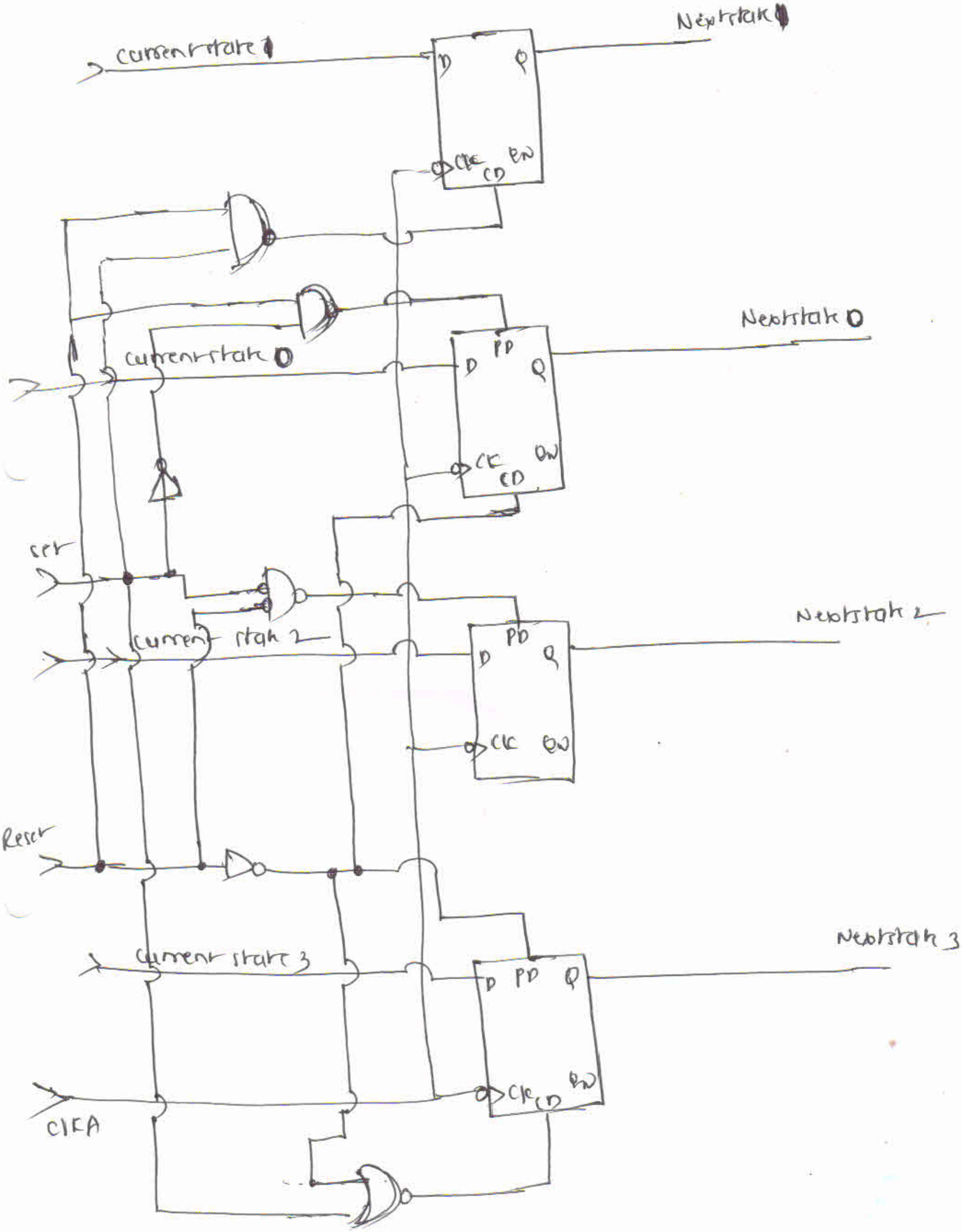
CD[0]

	0	1
0	0	0
1		

CD[0] = Reset

we need to take complement & connect it to the CD[0] pin.

CD[0] = $\overline{Reset}$


currentstate 1
Nextstate 1
D
Q
Clk
CD
EN
Nextstate 0
currentstate 0
D
PD
Q
CK
CD
EN
set
current state 2
Nextstate 2
PD
D
Q
CK
EN
Reset
Nextstate 3
current state 3
D
PD
Q
Clk
CD
EN
ClkA

Sync Preset Counter

we will synthesize with option 2, where Preset & clear logic will be directed towards D inputs. furthermore, we will use FF with two D_1, D_0 inputs & Dataselector pin.

First observation is the following, truth table will be large since we have more no. of i/p variables

Ex:	Preset Data	up down	CS	NS
preset	0		1 0	1 0

6 i/p variables :

~~updown~~

So, let's deal with only Preset input first & Let us assume that Preset=1, we are going to connect it to (CD) pin of FF. then D1 should be supplied with Preset Data.

i.e/

Preset	Preset Data 1	0	NS D_1	D_0
1	0	0	0	0
	0	1	0	1
	1	0	1	0
	1	1	1	1

⇒ NS[1]

D_1 = PresetData1

NS[0] D_0 = PresetData0.

else condition in the code works only when Preset = 0.

i.e mean we need to get Do pin logic.

Updown	CS		NS	
	1	0	1 Do	0 Do
0	0	0	1	1
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

NS[1]
Do

	00	01	1p	10
0	1		1	
1		1		1

NS[0]
Do

	00	01	11	10
0	1			1
1	1			1

NS[0]

$D_0 = \overline{CS[0]}$

NS[1]

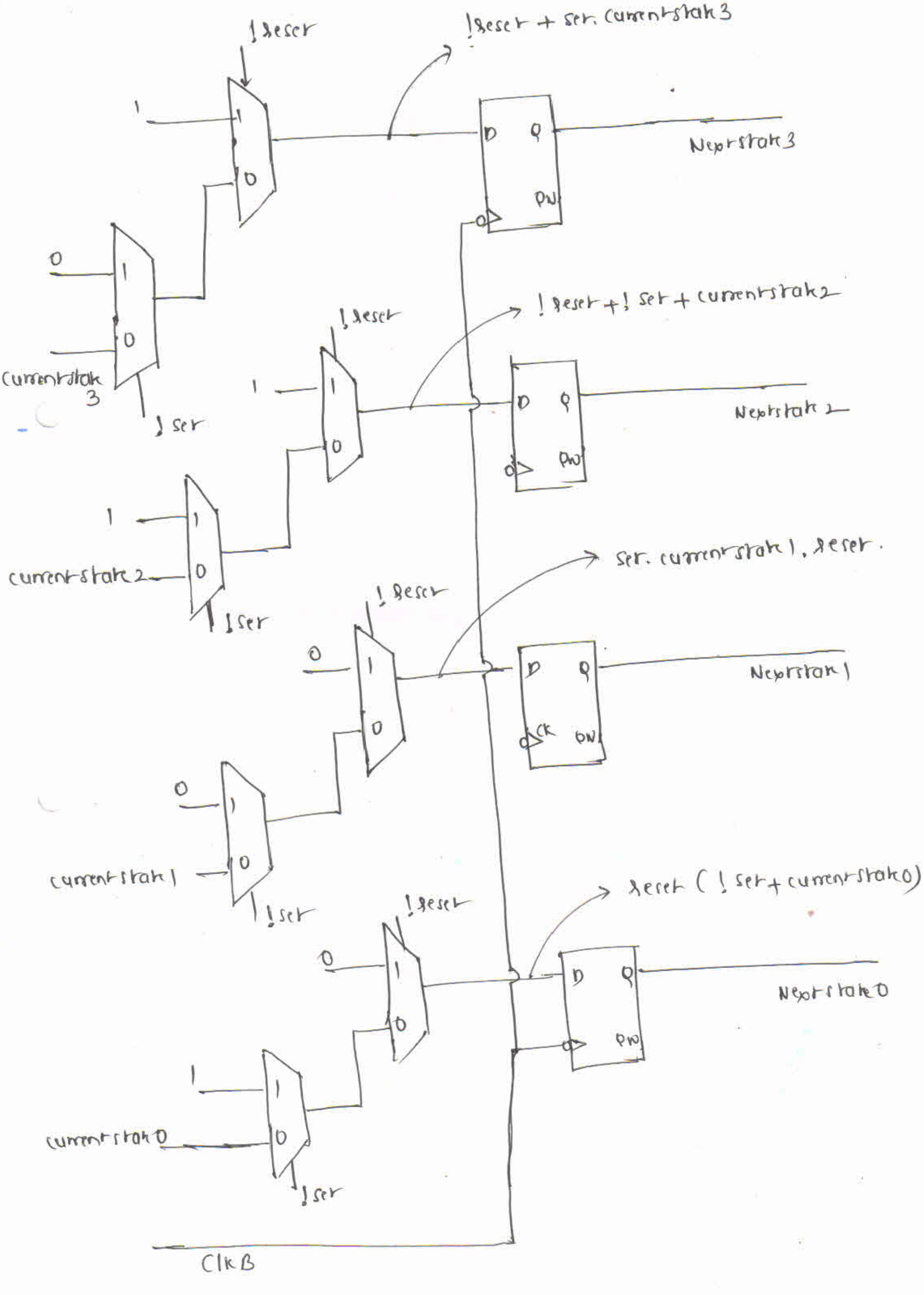
$$D_0 = \overline{updown}.\overline{CS[1]}\,\overline{CS[0]} + \overline{updown}.CS[1].CS[0] + updown\,\overline{CS[1]}\,CS[0] + updown\,CS[1]\,\overline{CS[0]}$$

$$= \overline{updown}\left(\overline{CS[1]}\,\overline{CS[0]} + CS[1]\,CS[0]\right) + updown\left(\overline{CS[1]}CS[0] + CS[1]\overline{CS[0]}\right)$$

$$= \left(\overline{CS[1]}\,\overline{CS[0]} + CS[1]\,CS[0]\right) \oplus updown$$

(*) circuit is not drawn *

Sync Flip Flop


!reset
!reset + set. currentstate3
1
1
0
0
1
0
currentstate 3
!set
D
Q
QN
Nextstate3
!reset + !set + currentstate2
!reset
1
1
0
1
0
1
currentstate2
!set
D
Q
QN
Nextstate2
set. currentstate1 . reset.
!Reset
0
1
0
0
1
0
currentstate1
!set
D
Q
CK
QN
Nextstate1
reset (! set + currentstate0)
!reset
0
1
0
1
1
0
currentstate0
!set
D
Q
QN
Nextstate0
ClkB