جامعة بيرزيت

BIRZEIT UNIVERSITY

ANSWER BOOKLET

Student: Digital Number 13

Course: Department: Number:

Division: Instructor:

Date: Day Month Year

For Instructor's Use

Question	Grade
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
Total	

(implication table)

	a	b	c	d	e	f	g
b	d,e ✓						
c	X	X					
d	X	X	X				
e	X	X	X	✓			
f	c,d X	se a,b X	X	X	X		
g	X	X	X	d,e ✓	d,e ✓	X	

(a,b), (d,e), (d,g), (e,g)

↓

3 equivalent states

⇒ final states

(a,b) (c) (d,e,g) (f)

↓ ↓ ↓ ↓

take a c d f

Present state	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
a	d	a	0	0
c	d	f	0	1
d	a	d	1	0
f	c	a	0	0

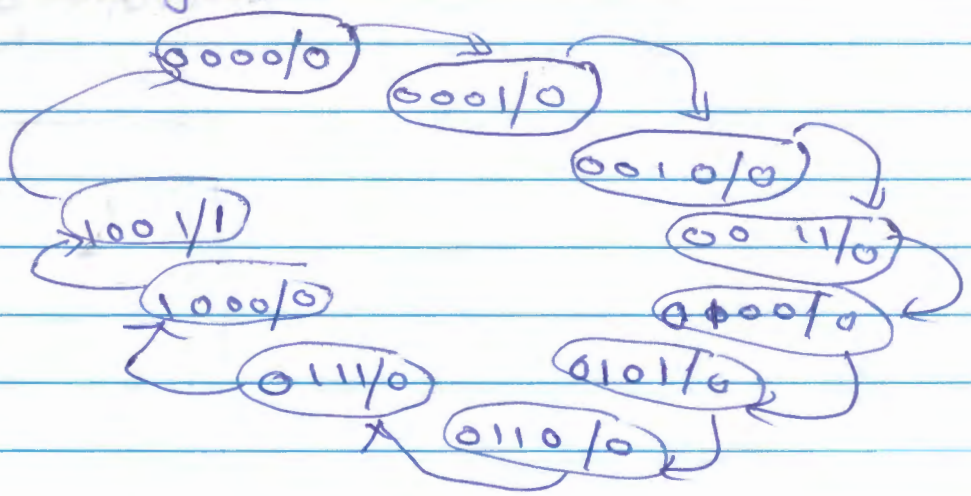
Synchronous Counters

- Binary Counter (given)
- up down binary counter (give)

BCD Counter (Decade counter)

- counting from 0000 to 1001 and back to 0000
- we must go through the procedures of sequential Design because this is not a regular counter.

⇒ State Diagram



- In this way, y can enable the count of the next higher significant decade while the same pulse switches the present decade from 1001 to 0000.

⇒ State table

Present state				Next State				output	FF inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	T_{Q_8}	T_{Q_4}	T_{Q_2}	T_{Q_1}
0	0	0	0	0	0	0	1	0	0	0	0	1
⋮				⋮				⋮				
1	0	0	1	0	0	0	0	1	1	0	0	1

- All unused states for minterm 10 to 15 are taken as don't care term ⇒

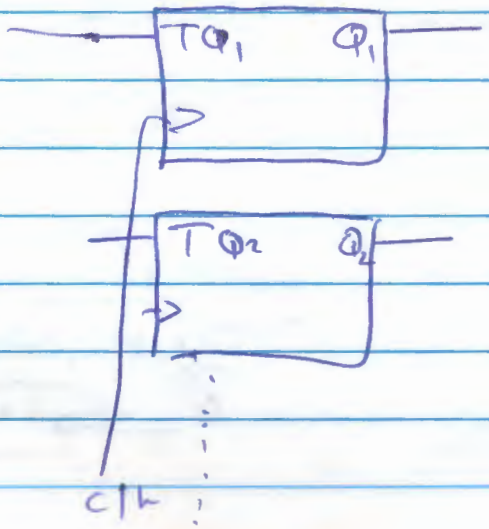
$TQ_1 = 1$

$TQ_2 = Q_8' Q_1$

$TQ_4 = Q_2 Q_1$

$TQ_8 = Q_8 Q_1 + Q_4 Q_2 Q_1$

$y = Q_8 Q_1$

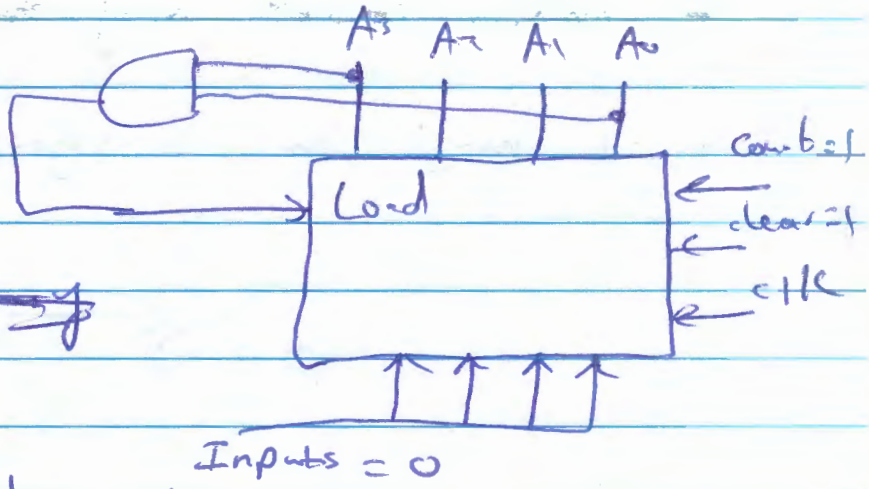


④ Binary counter with parallel load

clear	clk	Load	Count	Function
0	x	x	x	clear to 0
1	↑	1	x	Load inputs
1	↑	0	1	count Next state
1	↑	0	0	No change

- This counter good if we want to start counting from any initial state.

- also it is good to design other types of counters.

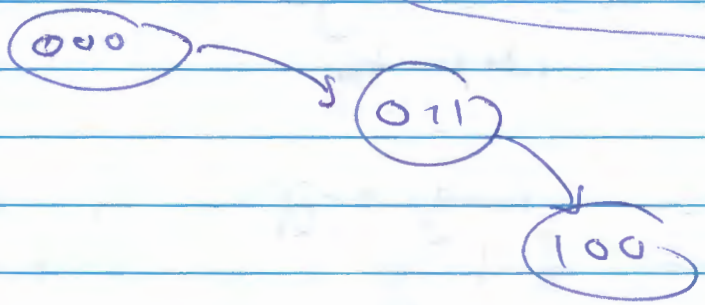


→ we design a ~~binary~~ BCD counter using the parallel load

(design a counter that counts up to 12 using Parallel load binary counter)

Irregular Counters:

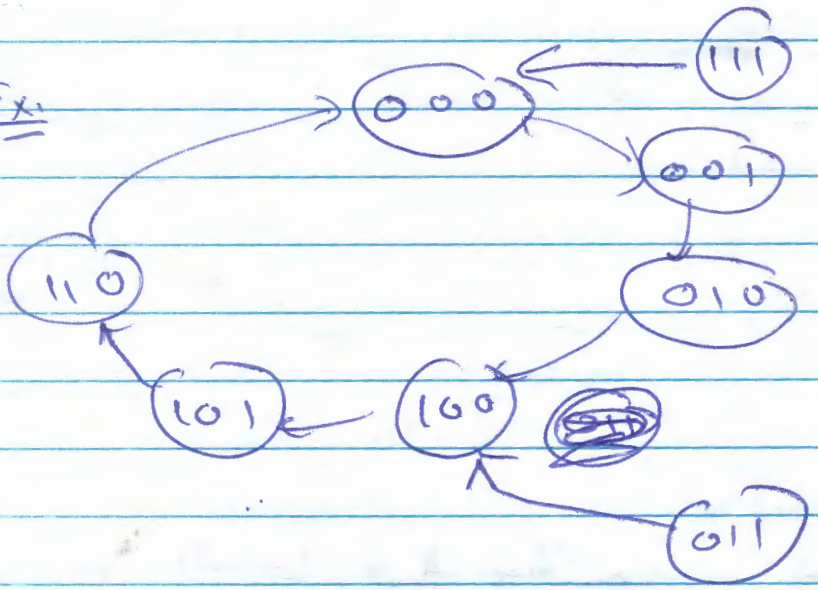
~~Design~~ Design a counter that counts in the following sequence → 0, 3, 4, 6, 5, 7, 2, 1,



⊛ Counter with unused state

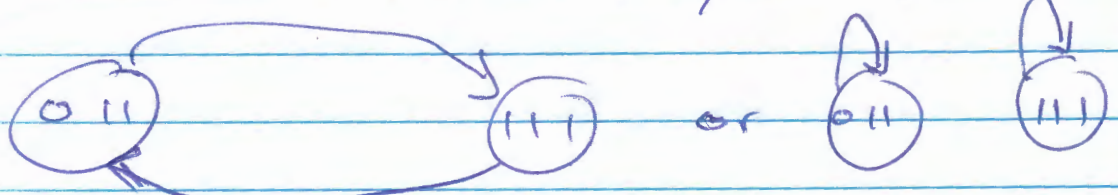
- unused states may be treated as don't care
- or may be assigned specific next state.

Ex:



in this example we ~~assign~~ force the unused states to go to a specific state in order to avoid lock case.

lock case may happen if the system for some failure goes to unused state, and for some design (not all of them) the unused state will go to another unused state, and so on



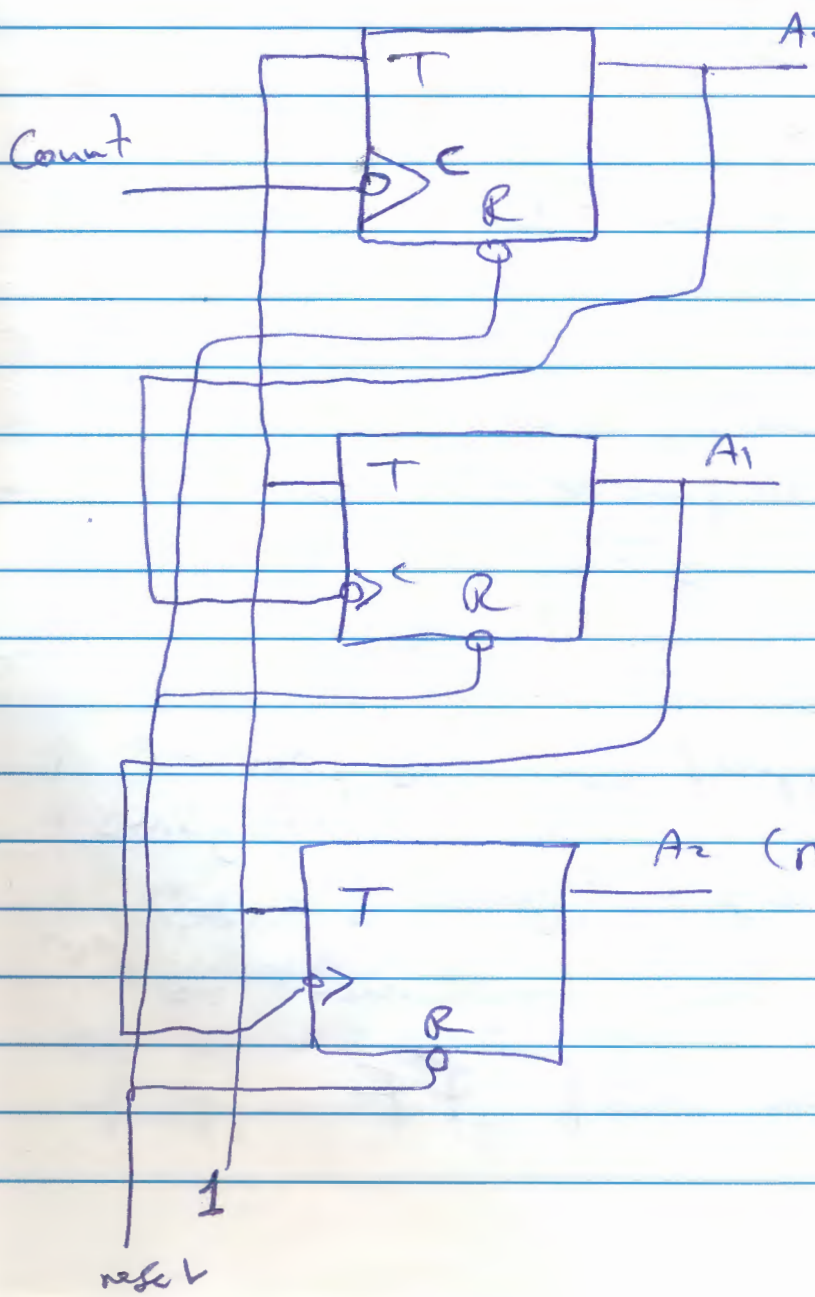
000	001	don't care now
011	xxx	
111	xxx	
000	001	assign force
⋮		
111 →	000	

Ripple Counters

The flip-flop output transition serves as a source for triggering other flip-flops. (i.e. the clk input of some flip-flops are not triggered by the common clock).

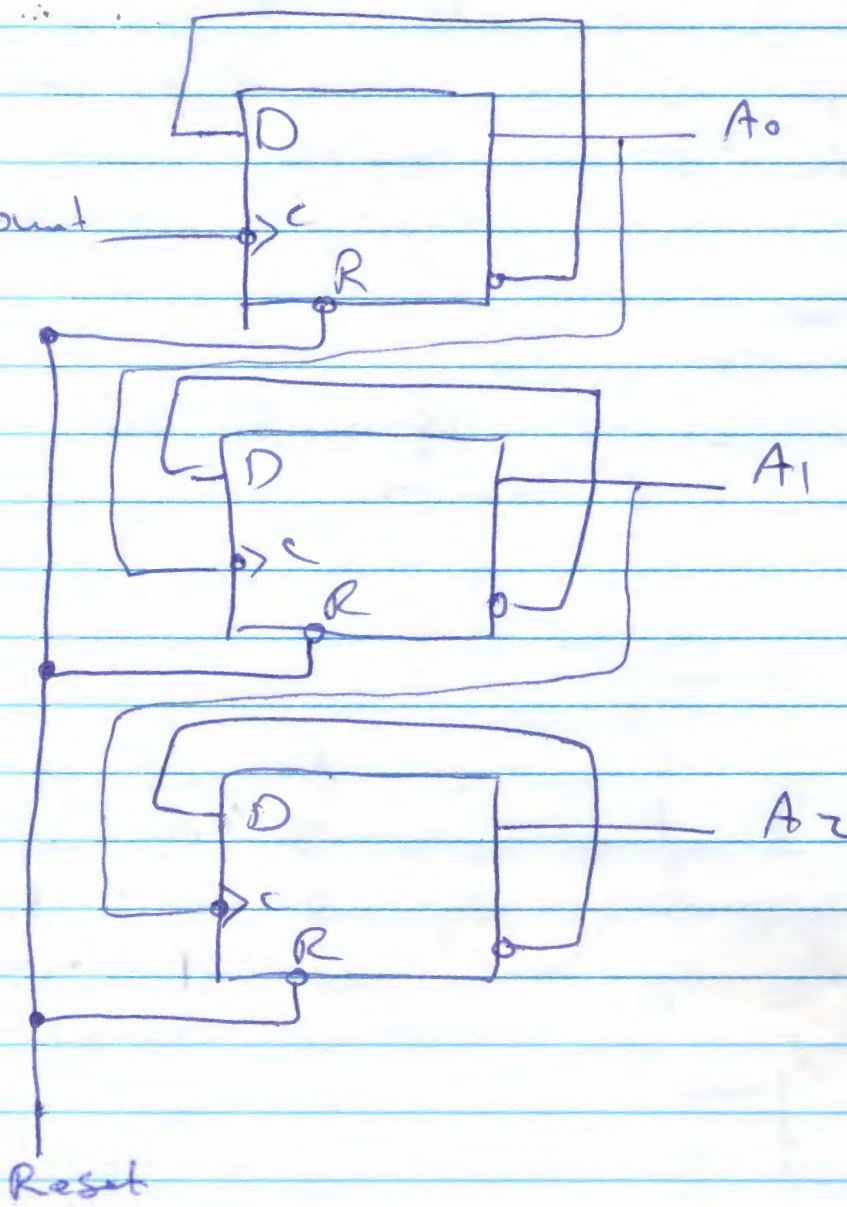
Binary Ripple Counter

Ex. Design a 3-bit binary ripple counter using TFF (up counter



A_2	A_1	A_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
⋮		

(*) Design a 3-bit Counter using D-FF



(*) Count-down-counter

+ve edge triggered (same as previous figures)

OR if -ve edge triggered is used

⇒ connect the complement to trigger the next FF.

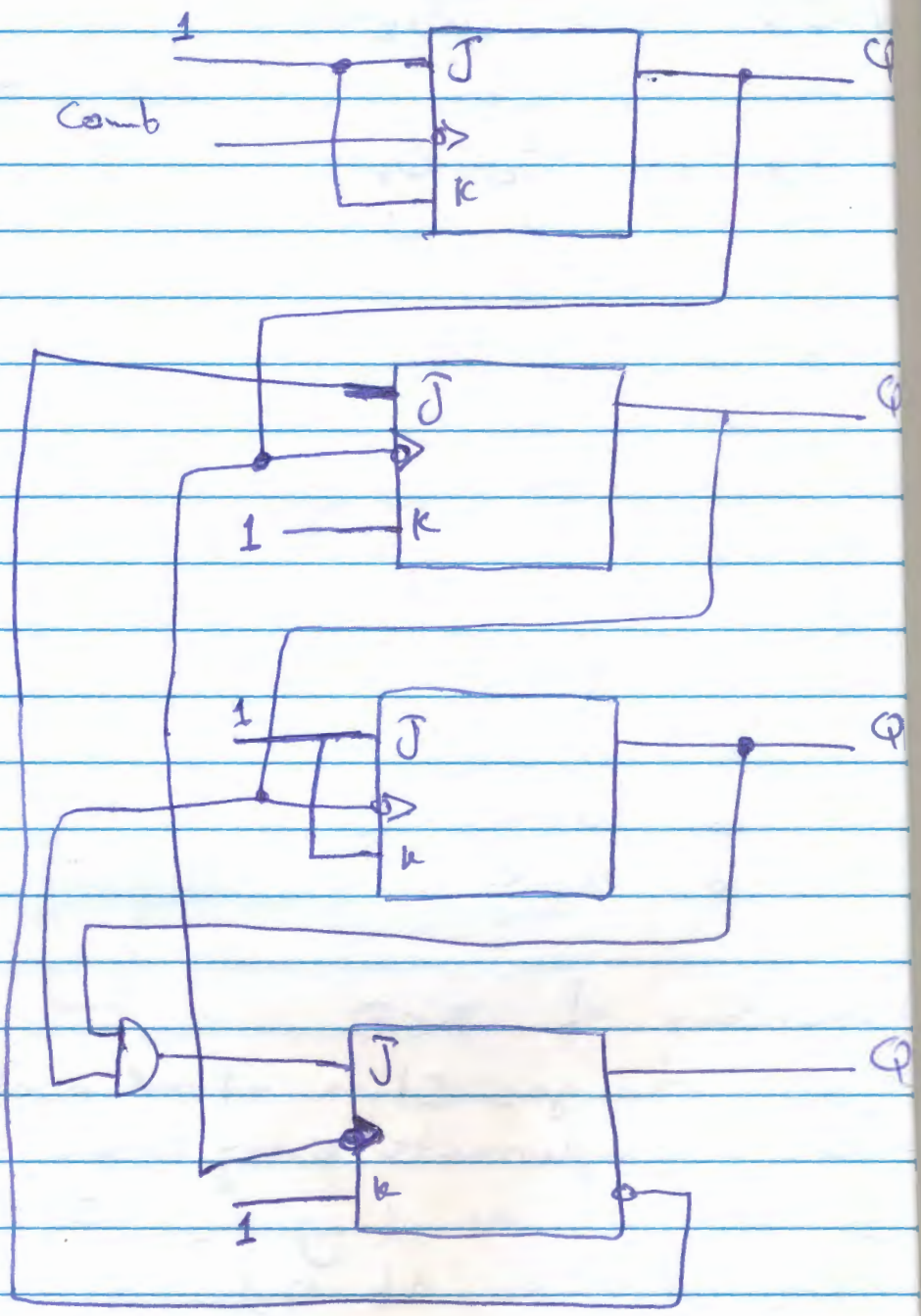
BCD Ripple Counter

- we need 4 bits (4 flip-flops)

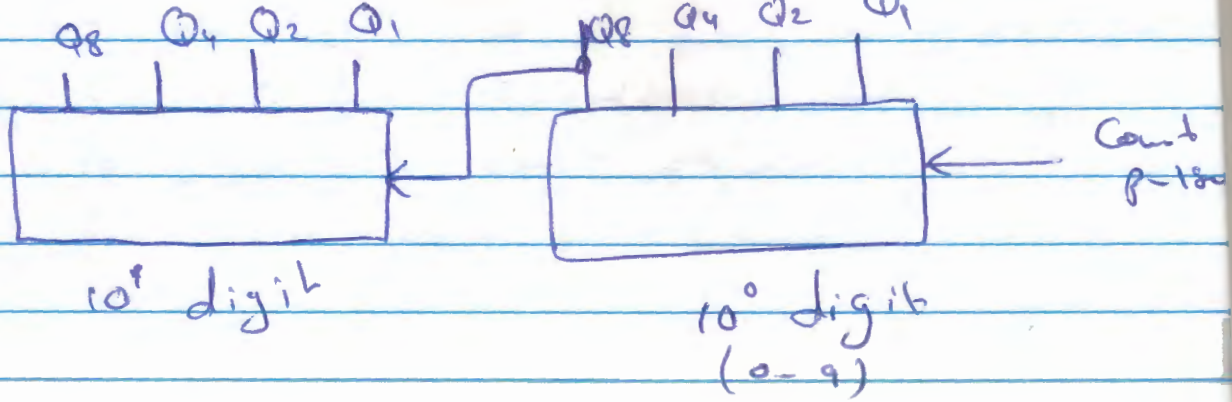
	Q_8	Q_4	Q_2	Q_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
0	0	0	0	0

Behaviour of BCD

- Q_1 changes state after each clock pulse
- Q_2 complements every time Q_1 goes from 1 to 0 as long as $Q_8 = 0$ (when $Q_8 = 1$, Q_2 remains at 0)
- Q_4 complements every time Q_2 goes from 1 to 0.
- Q_8 remains 0 as long as Q_2 or Q_4 is 0. when both Q_2 and Q_4 become 1, Q_8 complements when Q_1 goes from 1 to 0. Q_8 is cleared on the next transition of Q_1



⊛ Cascaded decade Counters

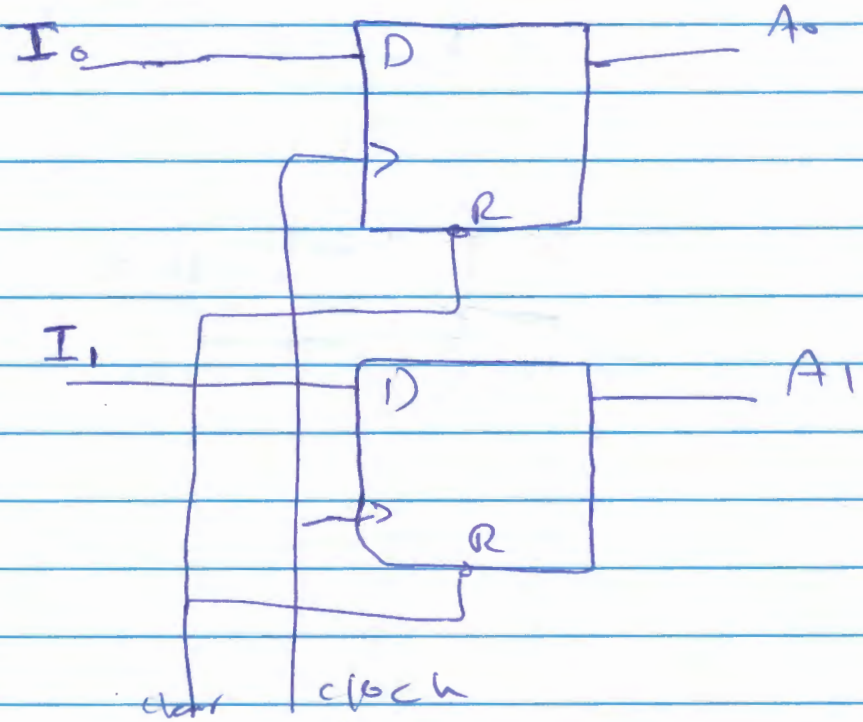


Ripple? why

1 1 1 1
↓
1 1 1 0
1 1 0 0
1 0 0 0
0 0 0 0

Registers

a register is a group of flip-flops. Each flip-flop is capable of storing one bit of information



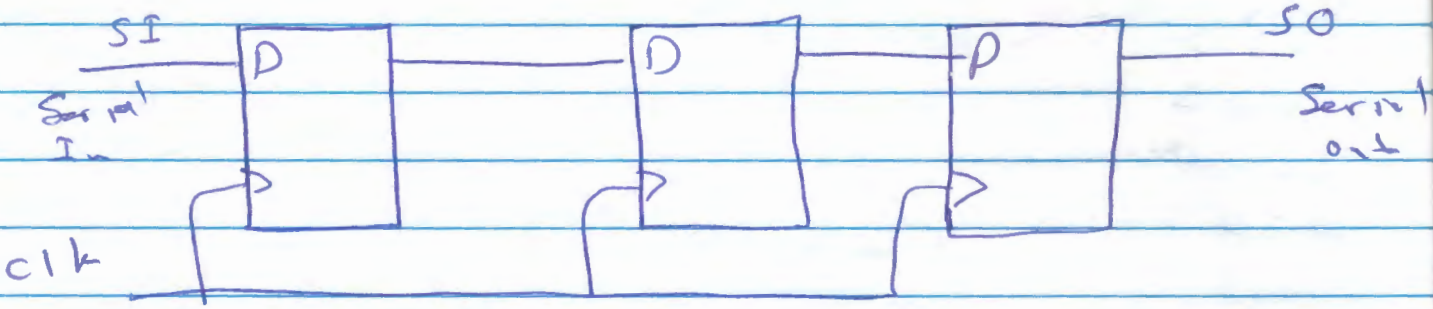
Register with parallel load

if load = 0 ⟶ no load

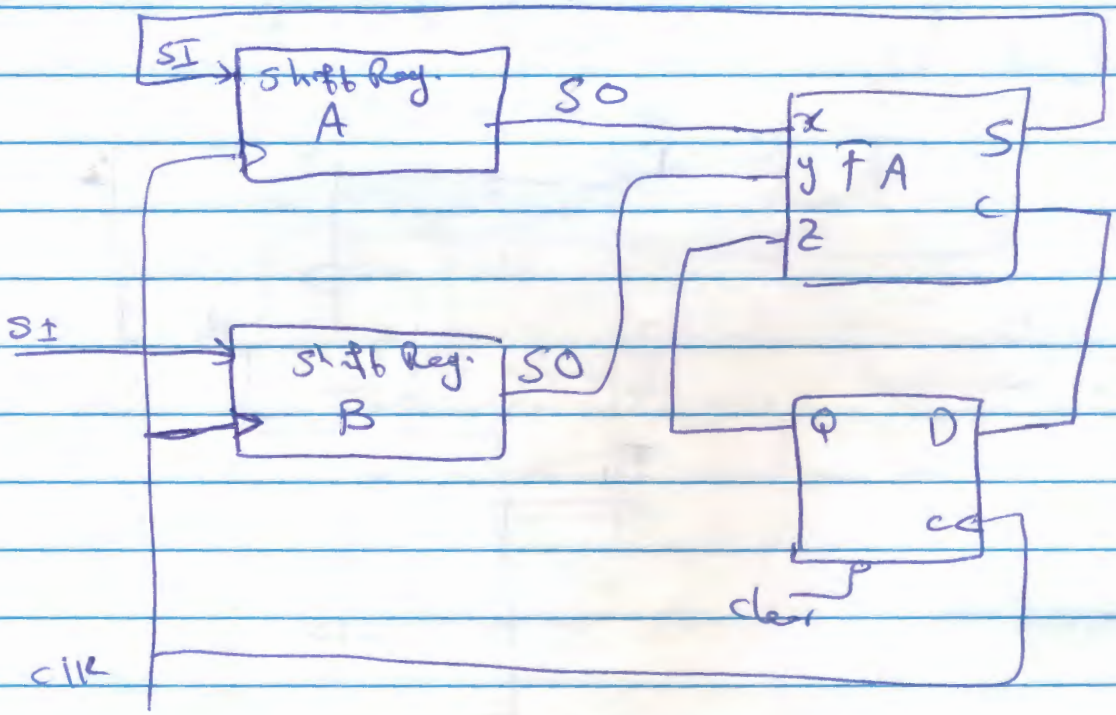
if load = 1 ⟶ load

⊛ Shift Register

Serial In – Serial Out



⊛ Serial Addition



1 0 1

0 1 1

Universal Shift Register

universal : functional complete (the function is controlled via some control inputs)

In General, a universal shift register can shift data right, left, and has a parallel load capabilities.

Example 4-bit universal shift register

$S_1S_0 = 00$ ⇒ The present value of the register is applied to the D inputs of the flip-flops.

⇒ $S_1S_0 = 00$ No change

$S_1S_0 = 01$ ⇒ shift right with serial input

$S_1S_0 = 10$ ⇒ shift left " " "

$S_1S_0 = 11$ ⇒ Parallel load.

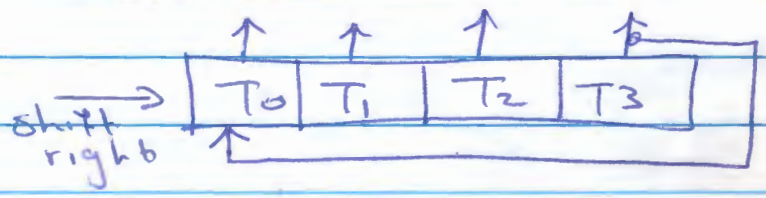
Behavioral Description of Universal shift Reg.

```
module universal(S1, S0, I, A, left, right, clk, clr);
input S1, S0, left, right, clk, clr;
input [3:0] I;
output [3:0] A;
reg [3:0] A;
always @ (posedge CLK or negedge clr)
    if(clr==0)  A = 4'b0000;
else
    case({S1,S0})
    2'b00 : A = A;
    2'b01 : A = {right, A[3:1]};
    2'b10 : A = {A[2:0], left};
    2'b11 : A = I;
endcase
endmodule
```

⊛ Ring Counter

- ring counter is a circular shift register or counter with only one flip flop being set at any particular time, all others are cleared.
- The sigle bit is shifted from one flip flop to the next to produce the sequence timing signals.

Ex: 4-bit shift register connected as ring counter



if initial value = 1 0 0 0

time = 0 $T_0 = 1$ $T_1 = 0$ $T_2 = 0$ $T_3 = 0$

time = 1 clock $T_0 = 0$ $T_1 = 1$ $T_2 = 0$ $T_3 = 0$

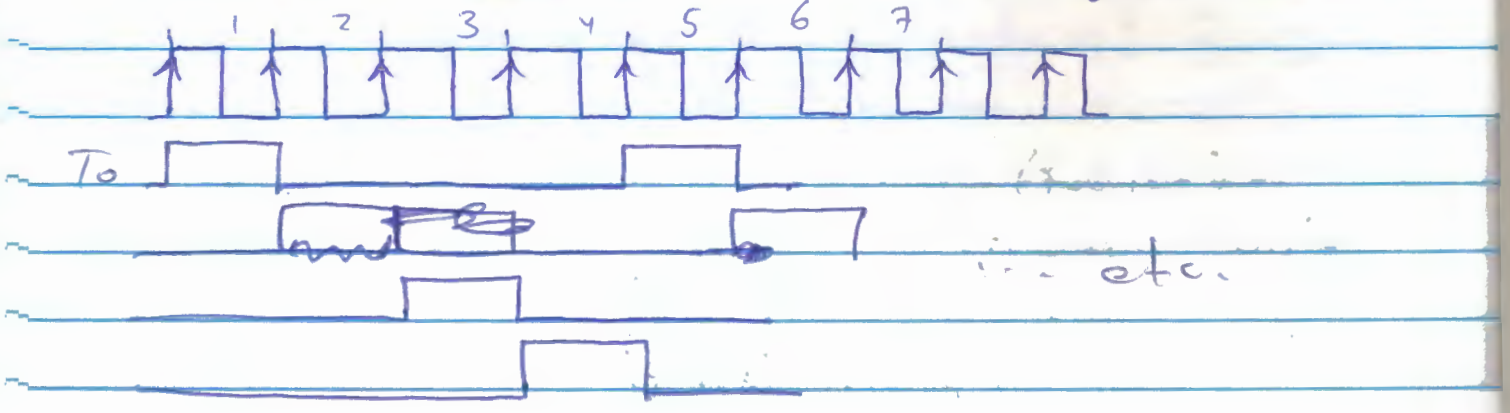
" = 2 " $T_0 = 0$ $T_1 = 0$ $T_2 = 1$ $T_3 = 0$

= 3 " $T_0 = 0$ $T_1 = 0$ $T_2 = 0$ $T_3 = 1$

" = 4 clock $T_0 = 1$ $T_1 = 0$ $T_2 = 0$ $T_3 = 0$

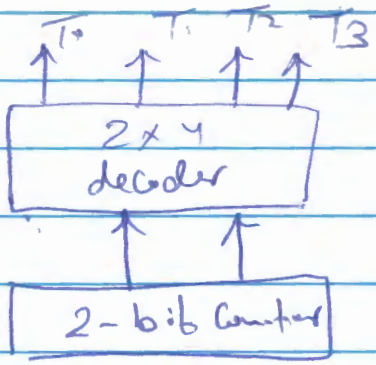
same as initial state

⇒ we generate 4 timing signals



④ Another way to generate the timing signal is to use a ~~binary~~ counter & decoder

Ex:

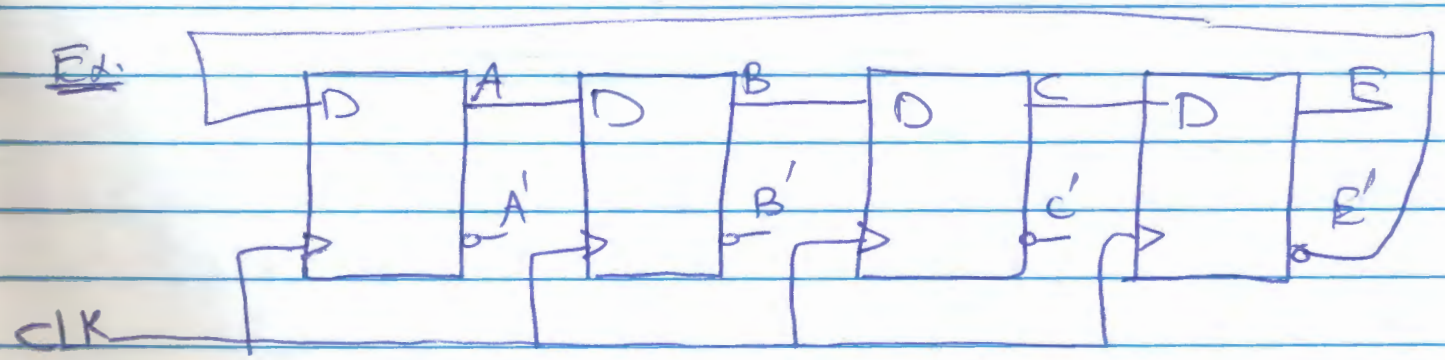


⑤ In general, To generate 2^n timing signal, we need either a shift register with 2^n flip-flops or an n-bit binary counter together with an n-to-2^n line decoder.

Ex. 16 timing signal ⇒ 16-bit shift register
or 4-bit Counter & 4×16 decoder.

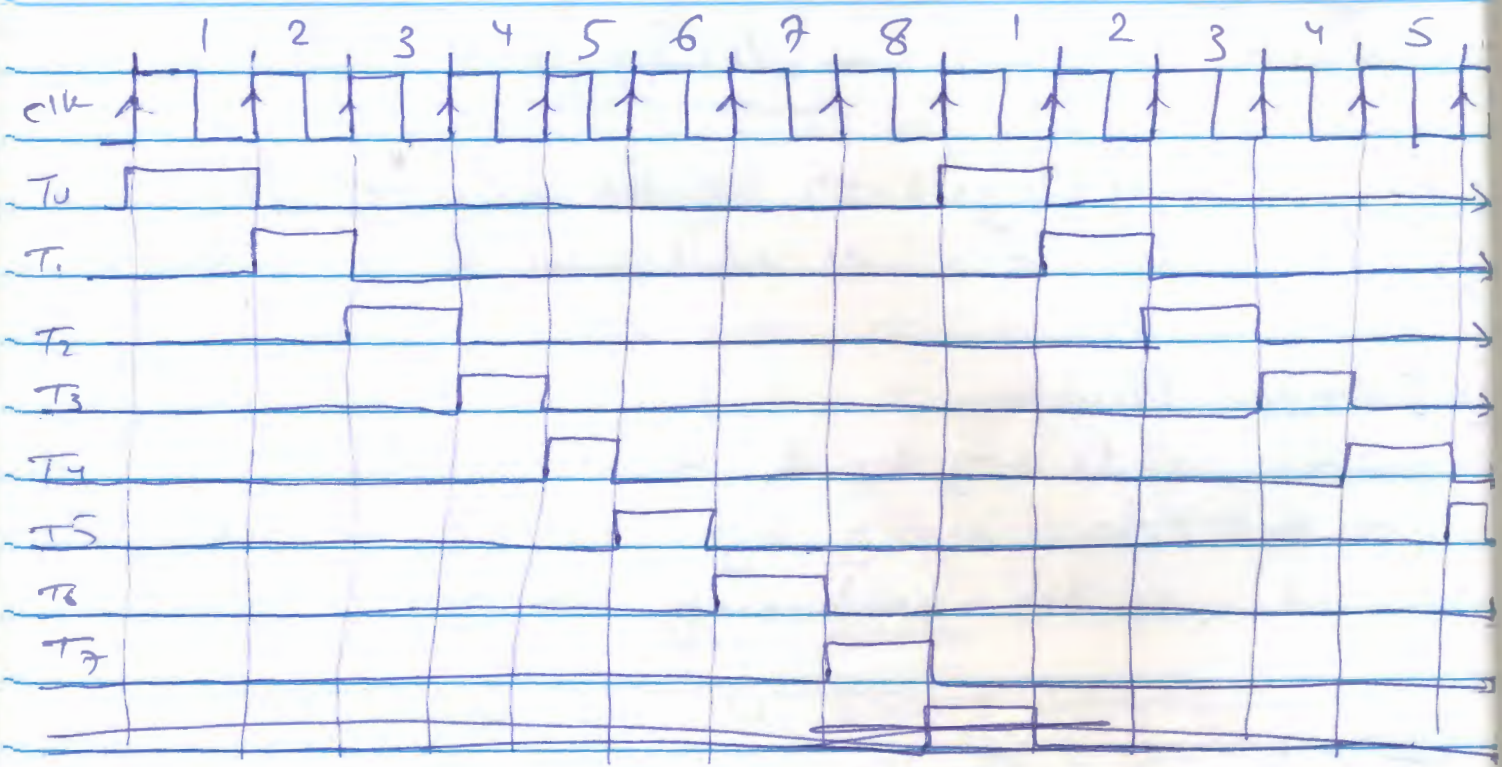
⑥ Johnson Counter
uses shift register & combinational logic to generate timing signals. It is also called switch-tail ring counter

Ex:



(four-stage switch tail ring counter)

Sequence No.	FF outputs				AND gate required for output	
	A	B	C	E		
1	0	0	0	0	$A'E'$	(T_0)
2	1	0	0	0	AB'	(T_1)
3	1	1	0	0	BC'	(T_2)
4	1	1	1	0	CE'	(T_3)
5	1	1	1	1	AE	(T_4)
6	0	1	1	1	$A'B$	(T_5)
7	0	0	1	1	$B'C$	(T_6)
8	0	0	0	1	$C'E$	T_7
(1)	0	0	0	0		



Johnson counter is a k-bit switch tail ring counter with 2k decoding gates to provide output for 2k timing signals.

One disadvantage is the lock if the circuit goes to unused state