1

USN

CMRIT

Internal Assessment Test 2 – Jan.2026

Sub:	Digital System Design using Verilog				Sub Code:	BEC302	Branch:	ECE
Date:	06-01-2026	Duration:	90 Minutes	Max Marks: 50	Sem/ Sec:	3/A,B,C,D		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	a) Obtain the characteristic equation & excitation table for following flip-flops 1) JK 2) SR 3)T FF 4) D FF b) Explain ring counter and Johnson(twisted ring) counter with timing diagram.	[10]	CO3	L3
2	Design a MOD-6 synchronous counter using clocked T flip flop	[10]	CO3	L2
3	Explain the working of M/S JK FF with a functional table and timing diagram. Show how race around condition is overcome.	[10]	CO3	L2
4	Explain arithmetic operators,logical operators and relational operators in verilog	[10]	CO4	L2
5	Write the verilog code for a Positive Edge-Triggered JK flip-flop using the CASE statement with the timing waveform.	[10]	CO4	L2
6	Write the Verilog code for 2*1 multiplexer using If-Else statement and Else-If statement using the below logic (i) when enable is low and selection line is high, the output is equal to input 2 (A) (ii) when enable is low and selection line is low, the output is equal to input 1 (B) (iii) If enable signal is high, the output reaches high impedance state (z)	[10]	CO4	L2
7	Write the Verilog code for the Full Adder using the data-flow description, behavioral and gate level description. 3+3+4	[10]	CO4	L2
8	Write the syntax of **IF-ELSE , IF-ELSE IF, CASE, FOR loop, WHILE loop statements, FOREVER and WAIT** with examples.	[10]	CO4	L2

COURSE INSTRUCTORS

5/1/26
CCI

5/1/2026
HOD

1. a) 1) JK flip flop

> The characteristic equation of this JK flip flop is given as,

$\boxed{Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n}$ or $\boxed{Q_{n+1} = J\bar{Q} + \bar{K}Q}$

> where the toggling condition as well as the unchanged, reset, set and all conditions are executed

> The characteristics equation will be obtained by the waveform or the table or the excitation table in the JK flip flop. The components used are.

> inputs - J, K

> output - Q_n, Q_{n+1}

CLK	J	K	Q_n	Q_{n+1}	
↑	0	0	0	0	NC
↑	0	~~1~~	0	1	RS
↑	1	0	1	0	S
↑	1	1	Toggle		

Q_{n+1} - Next state of the JK flip flop

Q_n - Present state of JK FF

J, K - inputs

CLK - single clock pulse

2) SR flip flop

> The characteristic equation of SR flip flop is,

$\boxed{Q_{n+1} = SQ_n + \bar{R}Q_n}$

> The last where the clock for both S and R are high then the output becomes invalid.

> The SR flip flop almost mirrors the JK flip flop.

CLK	S	R	Q_n	Q_{n+1}	
↑	0	0	0	0	No change
↑	0	1	0	1	Reset
↑	1	0	1	0	Set
↑	1	1	Invalid		

3) T flip flop

> The characteristic equation for T flip flop is given as,

$\boxed{Q_{n+1} = T \oplus Q_n}$ or $\boxed{Q_{n+1} = T \text{ XOR } Q_n}$

The T flip flop is XOR of the T and the present clock state

CLK	T	Q_n	Q_{n+1}
↑	0	Q'	Q
↑	1	Q	Q'

4) D flip flop

> D flip flop characteristic equation is given by,

$\boxed{Q_{n+1} = D}$

CLK	D	Q_n	Q_{n+1}
↑	0	0	1
↑	1	1	0

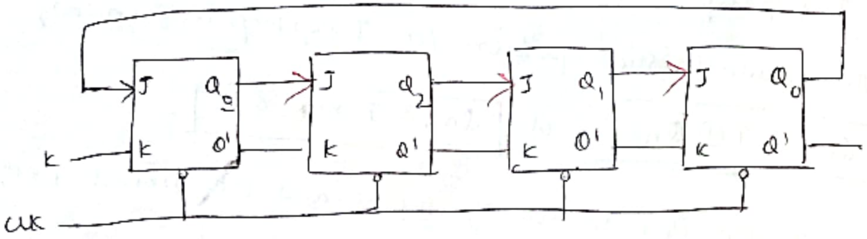
> D flip flop is same as the JK flip flop when the condition, J = K = D applies

1.b) Ring counter and Johnson (twisted ring) counter :

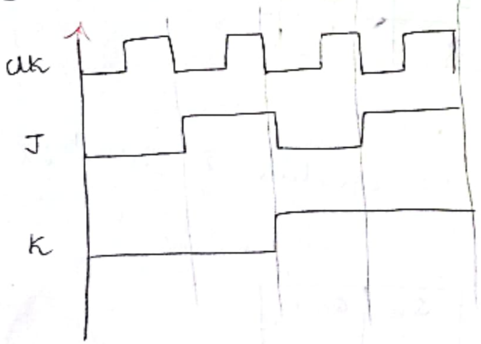
Ring counter :

> The ring counter is also known as a circular shift register.

> The output of the MSB is again applied to the input of the LSB counter or the flip flop used in the Ring counter.



Timing Diagram



Q_0	Q_1	Q_2	Q_3
0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

> Ring counter has no time delay and the high frequency glitch of the higher inputs.

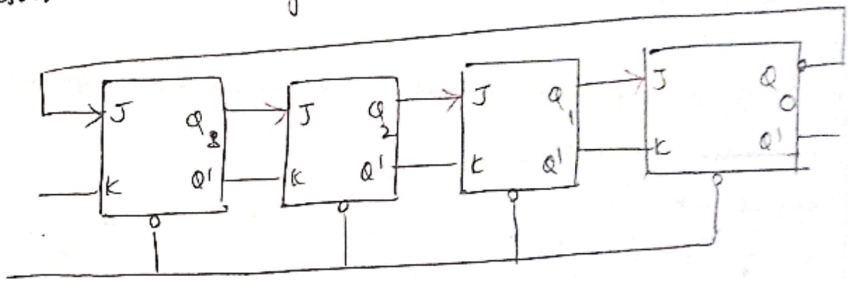
> Ring counter also used Master slave

Johnson (Twisted Ring) counter :

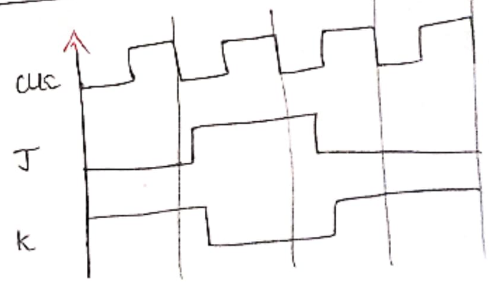
> As the name suggests the output of the ring counter is not directly applied to the input of the counter.

> Rather the output of the counter is inverted or the output which is applied to the counter is inverted Q' goes to the back to the counter.

> Johnson counter always acts as a inverted ring counter



The timing diagram :



Q_0	Q_1	Q_2	Q_3
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1

IAT-2.

2) MOD-6 Synchronous counter using T flip-flop.

→ · Synchronous counter is a counter in which, clock pulse is given to all the flip-flop at once.

· The output changes at a time

· Synchronous counter is a High-speed device it doesn't wait for the previous output like asynchronous counter

MOD 6.

MOD 6 counter.

MOD N · N=6 $1 \to 2 \to 3 \to 4 \to 5 \to 0$

$2^n \geq 6$.

Excitation table of T flip flop

T	Q	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	~~D_2~~ ~~T_2~~	~~T_1~~	~~T_0~~
0	0	0	0	0	1	0		
0	0	1	0	1	0	0		
0	1	0	0	1	1	0		
0	1	1	1	0	0	1		
1	0	0	1	0	1	0		
1	0	1	0	0	0			
1	1	0	X	X	X	X X	X	X
1	1	1	X	X	X.	X X.	X	X

T_2	T_1	T_0
0	0	1
0	1	1
0	0	1
1	1	1
0	0	1
1	0	1
X	X	X
X	X	X.

K map

(i) T_2

$Q_2 \backslash Q_1Q_0$	$\bar{Q}_1\bar{Q}_0$ 00	$\bar{Q}_1Q_0$ 01	Q_1Q_0 11	$Q_1\bar{Q}_0$ 10
$\bar{Q}_2$ 0	0_0	0_1	1_3	0_2
Q_2 1	0_4	1_5	X_7	X_6

$T_2 = Q_0Q_2 + Q_1Q_0$.

(ii) T_1

$Q_2 \backslash Q_1Q_0$	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	Q_1Q_0	$Q_1\bar{Q}_0$
$\bar{Q}_2$	0_0	1_1	1_3	0_2
Q_2	0_4	0_5	X_7	X_6

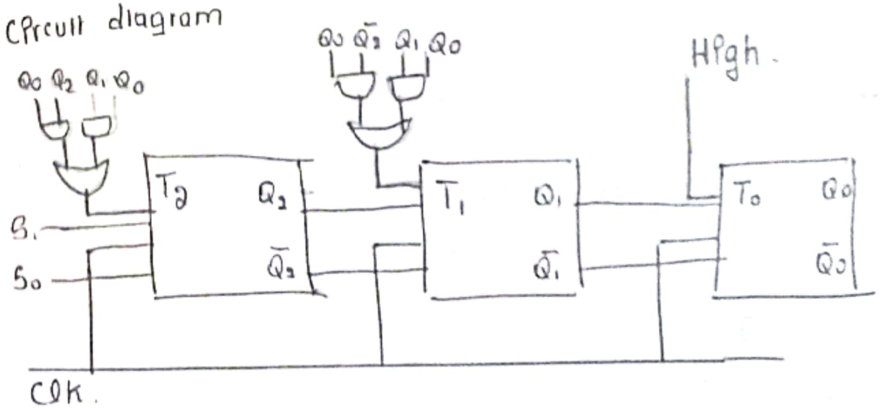
$T_1 = Q_0\bar{Q}_2 + Q_1Q_0$.

T_0

$Q_2 \backslash Q_1Q_0$	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	Q_1Q_0	$Q_1\bar{Q}_0$
$\bar{Q}_2$	1_0	1_1	1_3	1_2
Q_2	1_4	1_5	X_7	X_6

$T_0 = 1$.

Circuit diagram

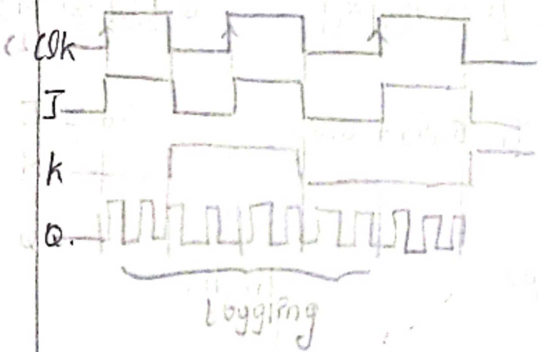


3) Master Slave JK FF

The normal JK flipflop, when both the clock pulse is high i.e J=k=1 then the output keep on changing again & again

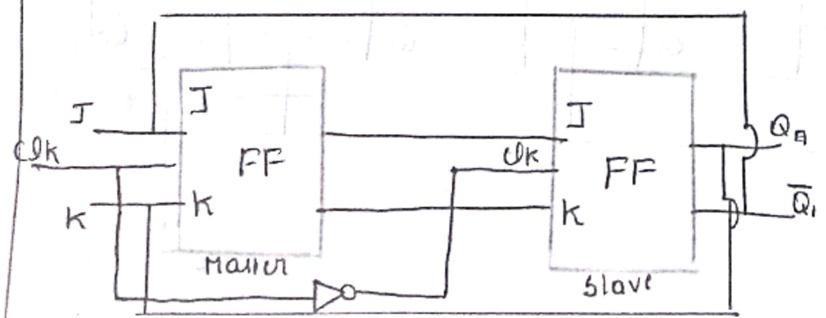
This condition is known as race around condition. The output will also be wrong if we il the problem continues

To solve this problem JK uses two flip-flop
(i) Master flipflop
(ii) Slave flipflop
(After using this)

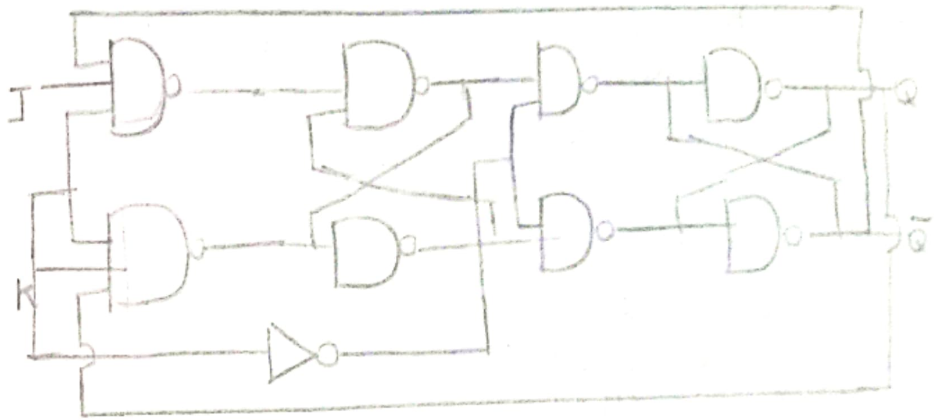


This is during race around condition

Master - Slave J-K flip flop



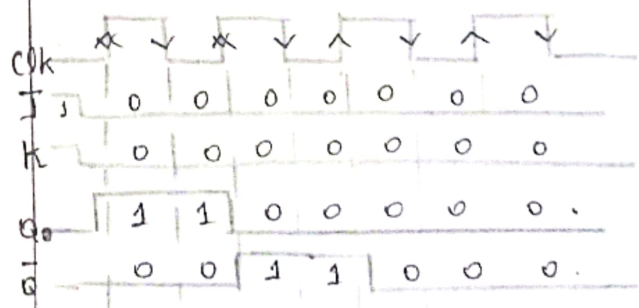
Using NAND Gate.



Truth table

J	k	Q	Qd	
0	0	0	0	no change
0	0	1	1	
0	1	0	0	Q0
0	1	1	0	
1	0	0	1	Q=1
1	0	1	1	
1	1	0	x	Toggle
1	1	1	x	

Waveform



CMR

4) **Arithmetic operator**

OP	Meaning
'+'	Addition
'−'	Subtraction
'*'	Multiplication
'/'	devision
'%'	Modular

logic operator

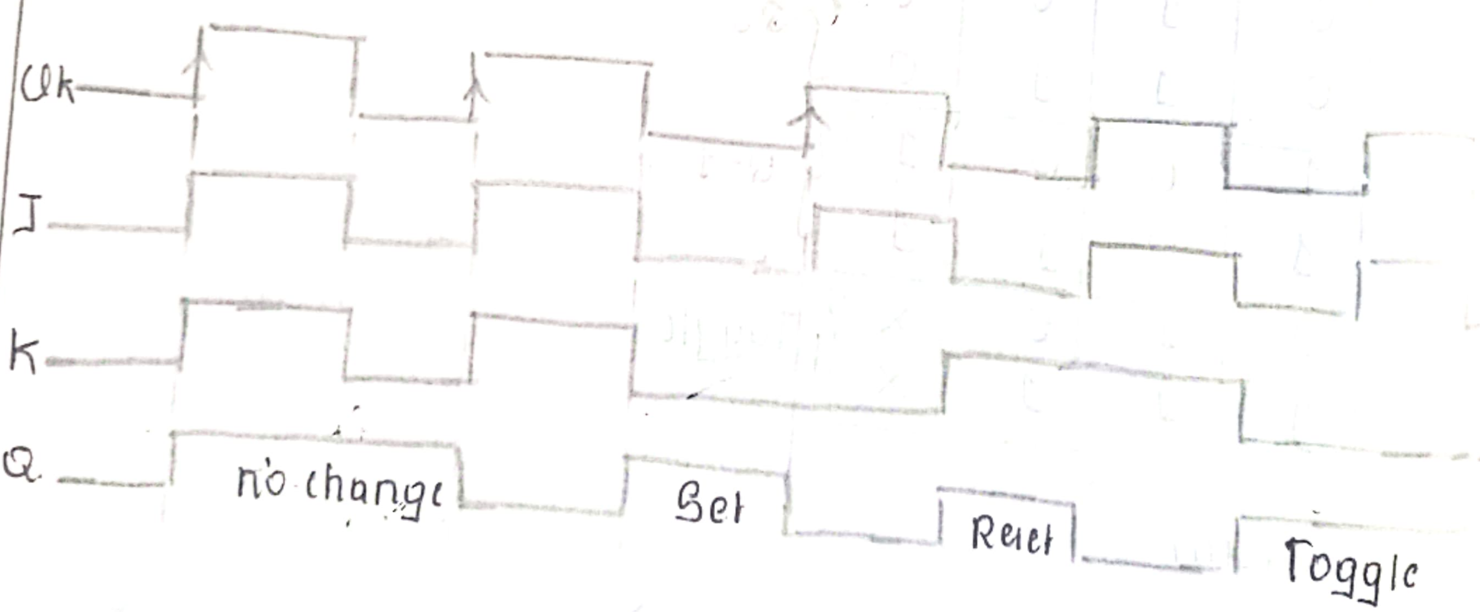
OP	Meaning
'!'	NOT oper
'&&'	AND oper
'||'	OR oper

Relational operator

OP	Meaning
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

5)
```
module Test1 (J, k, clk, Q);
input J,k,clk;
output reg Q;
always @ (posedge clk)
begin
  case ({J,K})
    2'b00 : Q<=Q;
    2'b01 : Q<= 1'b0;
    2'b10 : Q<= 1'b1;
    2'b11 : Q<= ~Q;
  endcase
end
endmodule.
```

Waveform

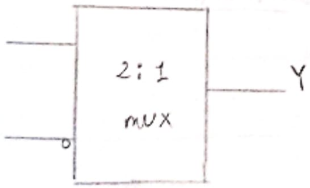


6. 2:1 MUX.

- If- Else: used when we have to check one condition and apply operation, takes to else block if condition not satisfied.

```
module mux (
    input A, B, sel, en
    output reg Y );
    always @ (*)
    begin
      If ( en == 1'b1 )
          Y <= 1'bz ;
      else
        if (sel == 1'b1)
          Y <= A ;
        else
          Y <= B ;
    end
endmodule.
```

en	sel	Y
1	x	z
0	1	A
0	0	B



- Else if: used when we have to check for a condition more than once.

```
module mux (
        input A, B, sel, en
        output reg Y );
        always @ (*)
        begin
          If ( en == 1'b1 )
              Y <= 1'bz ;
          else if ( en == 1'b0 && sel == 1'b1 )
              Y <= A ;
          else
              Y <= B ;
        end
endmodule.
```

7) Full-Adder

Data flow model

```
module FA ( A, B, Cin, Sum, cout);
input A, B, Cin ;
output Sum, cout ;
assign Sum = a^b^cin;
assign cout = (a&b) | (b&c) | (c&a);
end module
```

behavioural model

```
module FA ( A , B, Cin, Sum, cout);
input A, B, Cin;
output reg Sum, cout;
always @ (a or b or c)
begin
  Sum = a^b^cin;
  cout = (a&b) | (b&c) | (c&a);
end
end module
```

gate level description

```
module FA ( A, B, Cin, Sum, cout) ;
input A, B, Cin ;
output Sum, cout ;
wire S1, C1, C2;
XOR (A,B,S1)
AND (A,B, C1)
XOR (S1,Cin Sum)
AND (S1 Sin C2)
OR (Cout C1, C2)
end Module.
```

XOR (S1, cin, Sum)
AND (S1, Sin, C2)
OR (Cout, C1, C2)
end module.

⑧ ## IF-ELSE

Syntax

~~IF_ELSE(~~

Syntax:

```
If (condition);
    Statement;
else;
    Statement;
```

Ex:

```
If (a>b);
    print(a);
else;
    print(b);
```

IF-ELSE-IF

Syntax:

```
If (condition);
    Statement1;
else if;
    Statement2;
else:
    Statement 3;
```

Ex:

```
If (a>b)
    print(a);
else if (b>c)
    print(b);
else:
    print(c);
```

CASE

Syntax

```
Begin
value 1: Statement 1;
value 2: Statement 2;
   :
valueN: StatementN;
```

Ex:

```
2'b00: y=a[0];
2'b01: y=a[1];
2'b10: y=a[2];
2'b11: y=a[3];
```


FOR loop

Syntax:

```
for (initialization; condition; declaration);
```

decrement or increment

Ex:

```
for (i=0; i<n; i++);
```

WHILE LOOP

Syntax:

```
while (condition)
begin
    Statement;
end.
```

```
while (a==0)
begin
    to print True;
end.
```

FOREVER

Syntax.

~~FOREVER (condition) Statement;~~

```
FOREVER
begin
    Statement1;
end.
```

Example:

```
FOREVER
begin
    a>>b;
end
```

WAIT

Syntax

```
WAIT (condition)
Statement1;
```

Example:

```
WAIT (t==0)
print Start;
```