ECE2002 –Digital logic And Design

Embedded Lab-

Fall semester 2020~2021

Slot: L41+L42

E-Record

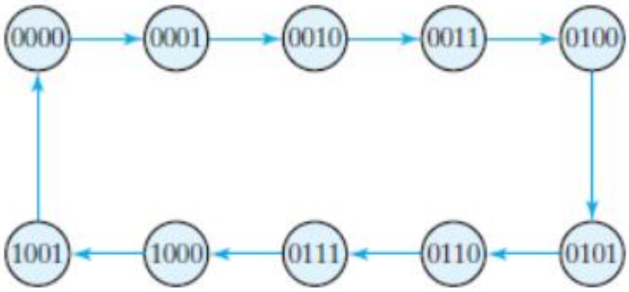
Experiment No. : 6

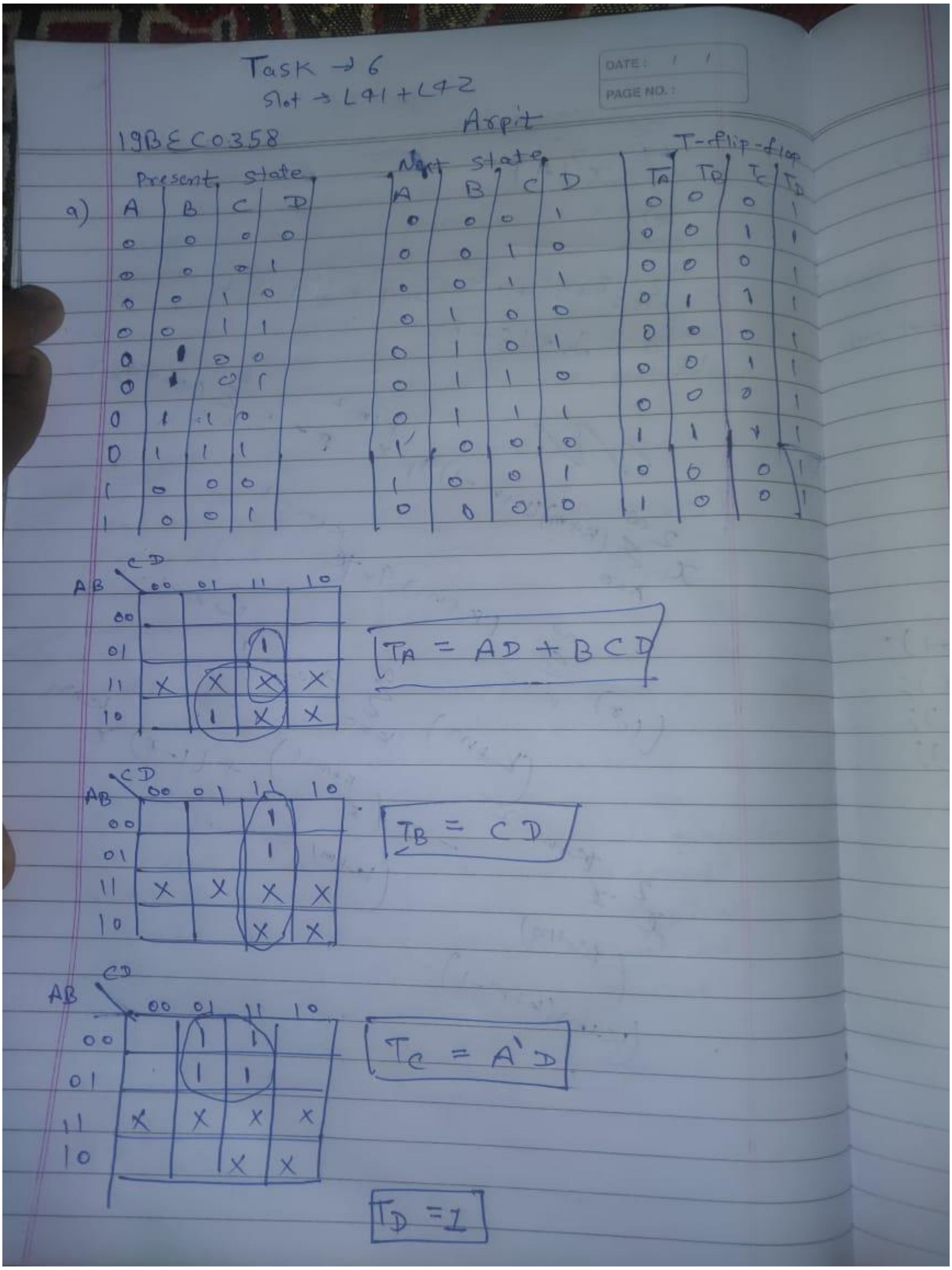
Submitted by

Name of the Student: ARPIT PATAWAT

Reg. no.: 19BEC0358

a) Design a BCD ripple counter using "T flip flop". Block diagram is given below.





Task → 6

Slot → L41 + L42

Arpit

19BEC0358

DATE : / /

PAGE NO. :

a)

Present state				Next state				T-flip-flop			
A	B	C	D	A	B	C	D	T_A	T_B	T_C	T_D
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	1

AB \ CD	00	01	11	10
00				
01			1	
11	X	X	X	X
10		1	X	X

$$\boxed{T_A = AD + BCD}$$

AB \ CD	00	01	11	10
00			1	
01			1	
11	X	X	X	X
10			X	X

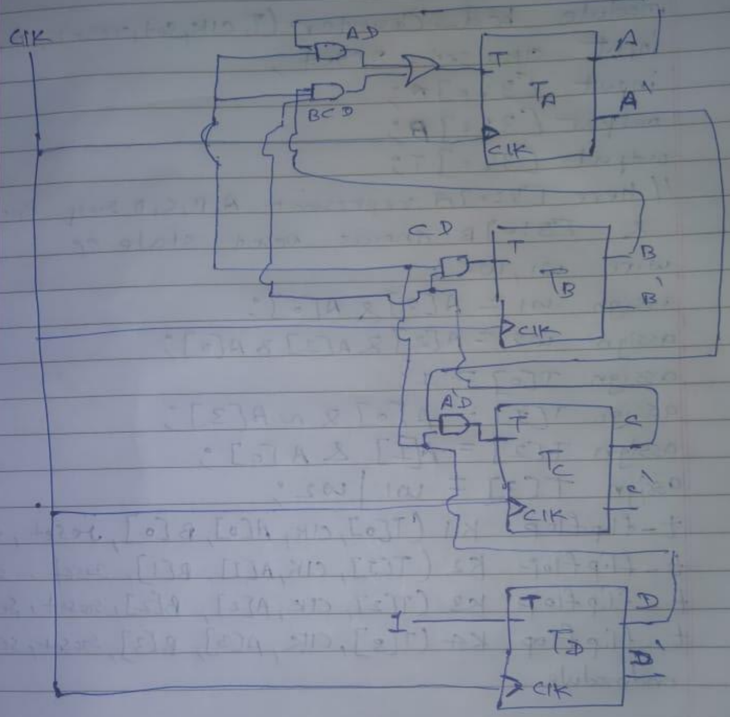
$$\boxed{T_B = CD}$$

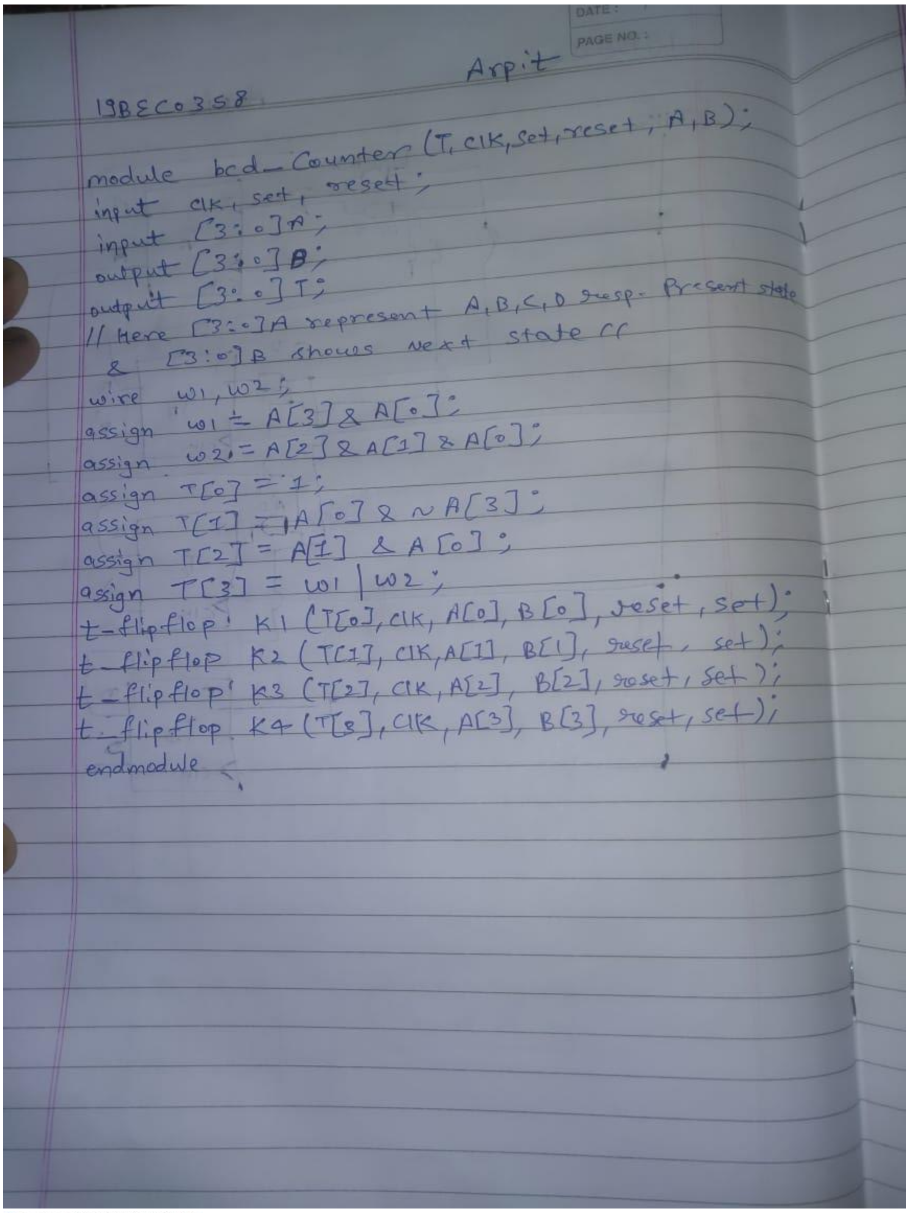
AB \ CD	00	01	11	10
00		1	1	
01		1	1	
11	X	X	X	X
10			X	X

$$\boxed{T_C = A'D}$$

$$\boxed{T_D = 1}$$

19BEC0358 Arpit





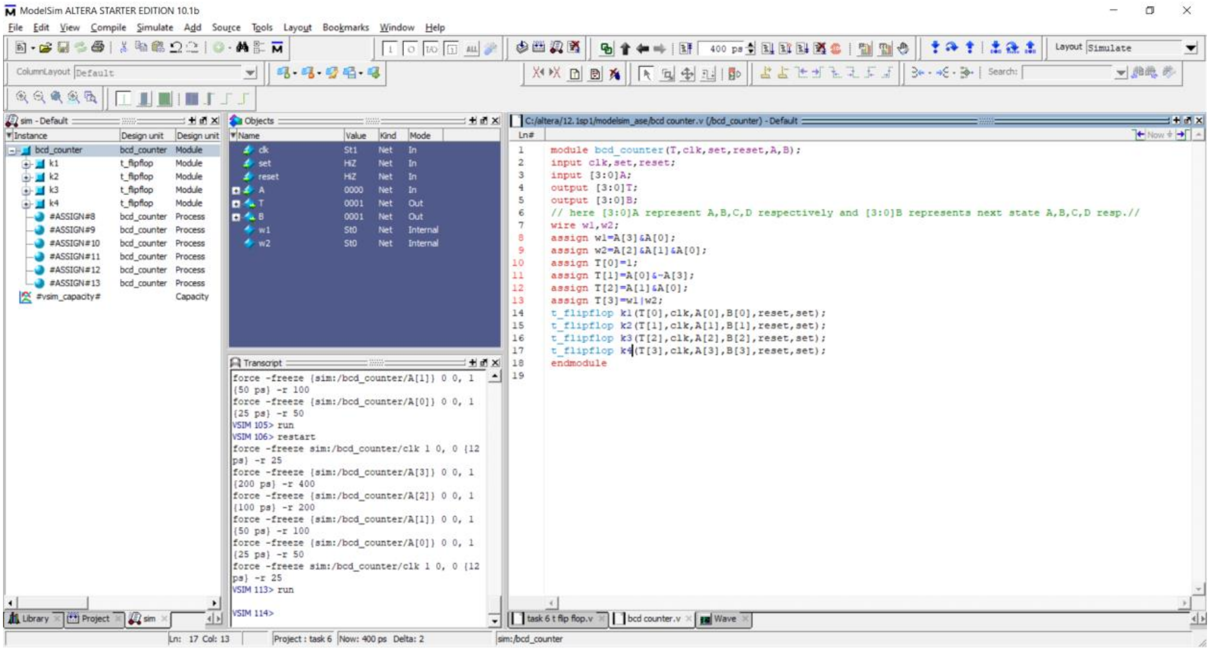
DATE :

PAGE NO. :

Arpit

19BEC0358

```
module bcd_Counter (T, clk, set, reset, A, B);
input clk, set, reset;
input [3:0]A;
output [3:0]B;
output [3:0]T;
// Here [3:0]A represent A,B,C,D resp. Present state
  & [3:0]B shows Next state cc
wire w1, w2;
assign w1 = A[3] & A[0];
assign w2 = A[2] & A[1] & A[0];
assign T[0] = 1;
assign T[1] = A[0] & ~A[3];
assign T[2] = A[1] & A[0];
assign T[3] = w1 | w2;
t-flipflop K1 (T[0], clk, A[0], B[0], reset, set);
t_flipflop K2 (T[1], clk, A[1], B[1], reset, set);
t_flipflop K3 (T[2], clk, A[2], B[2], reset, set);
t_flipflop K4 (T[3], clk, A[3], B[3], reset, set);
endmodule
```



Instance	Design unit	Design unit
bcd_counter	bcd_counter	Module
k1	t_flipflop	Module
k2	t_flipflop	Module
k3	t_flipflop	Module
k4	t_flipflop	Module
#ASSIGN#8	bcd_counter	Process
#ASSIGN#9	bcd_counter	Process
#ASSIGN#10	bcd_counter	Process
#ASSIGN#11	bcd_counter	Process
#ASSIGN#12	bcd_counter	Process
#ASSIGN#13	bcd_counter	Process
#vsim_capacity#		Capacity

Name	Value	Kind	Mode
clk	St1	Net	In
set	HiZ	Net	In
reset	HiZ	Net	In
A	0000	Net	In
T	0001	Net	Out
B	0001	Net	Out
w1	St0	Net	Internal
w2	St0	Net	Internal

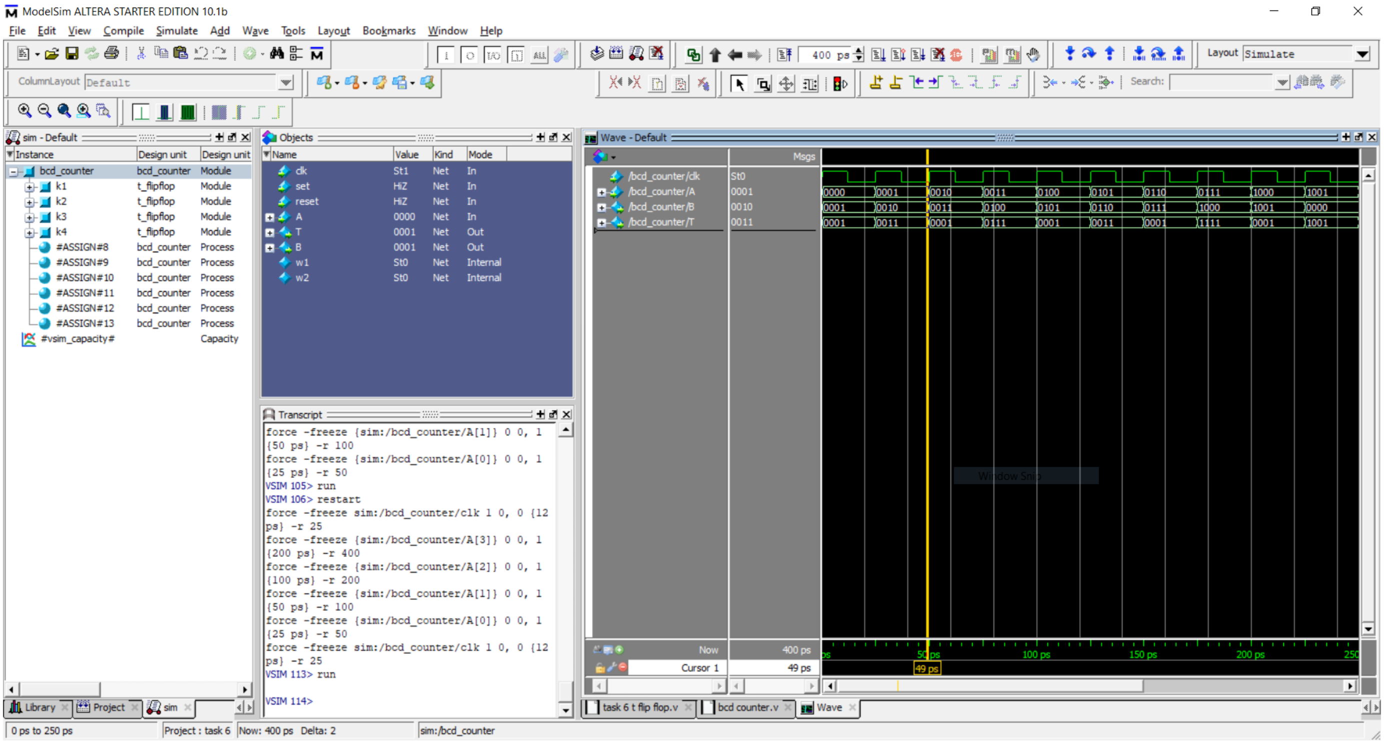
```
force -freeze {sim:/bcd_counter/A[1]} 0 0, 1
{50 ps} -r 100
force -freeze {sim:/bcd_counter/A[0]} 0 0, 1
{25 ps} -r 50
VSIM 105> run
VSIM 106> restart
force -freeze sim:/bcd_counter/clk 1 0, 0 {12
ps} -r 25
force -freeze {sim:/bcd_counter/A[3]} 0 0, 1
{200 ps} -r 400
force -freeze {sim:/bcd_counter/A[2]} 0 0, 1
{100 ps} -r 200
force -freeze {sim:/bcd_counter/A[1]} 0 0, 1
{50 ps} -r 100
force -freeze {sim:/bcd_counter/A[0]} 0 0, 1
{25 ps} -r 50
force -freeze sim:/bcd_counter/clk 1 0, 0 {12
ps} -r 25
VSIM 113> run

VSIM 114>
```

C:/altera/12.1sp1/modelsim_ase/bcd counter.v (/bcd_counter) - Default

```
module bcd_counter(T,clk,set,reset,A,B);
input clk,set,reset;
input [3:0]A;
output [3:0]T;
output [3:0]B;
// here [3:0]A represent A,B,C,D respectively and [3:0]B represents next state A,B,C,D resp.//
wire w1,w2;
assign w1=A[3]&A[0];
assign w2=A[2]&A[1]&A[0];
assign T[0]=1;
assign T[1]=A[0]&~A[3];
assign T[2]=A[1]&A[0];
assign T[3]=w1|w2;
t_flipflop k1(T[0],clk,A[0],B[0],reset,set);
t_flipflop k2(T[1],clk,A[1],B[1],reset,set);
t_flipflop k3(T[2],clk,A[2],B[2],reset,set);
t_flipflop k4(T[3],clk,A[3],B[3],reset,set);
endmodule
```

Ln: 17 Col: 13 | Project : task 6 | Now: 400 ps Delta: 2 | sim:/bcd_counter



Instance	Design unit	Design unit
bcd_counter	bcd_counter	Module
k1	t_flipflop	Module
k2	t_flipflop	Module
k3	t_flipflop	Module
k4	t_flipflop	Module
#ASSIGN#8	bcd_counter	Process
#ASSIGN#9	bcd_counter	Process
#ASSIGN#10	bcd_counter	Process
#ASSIGN#11	bcd_counter	Process
#ASSIGN#12	bcd_counter	Process
#ASSIGN#13	bcd_counter	Process
#vsim_capacity#		Capacity

Name	Value	Kind	Mode
clk	St1	Net	In
set	HiZ	Net	In
reset	HiZ	Net	In
A	0000	Net	In
T	0001	Net	Out
B	0001	Net	Out
w1	St0	Net	Internal
w2	St0	Net	Internal

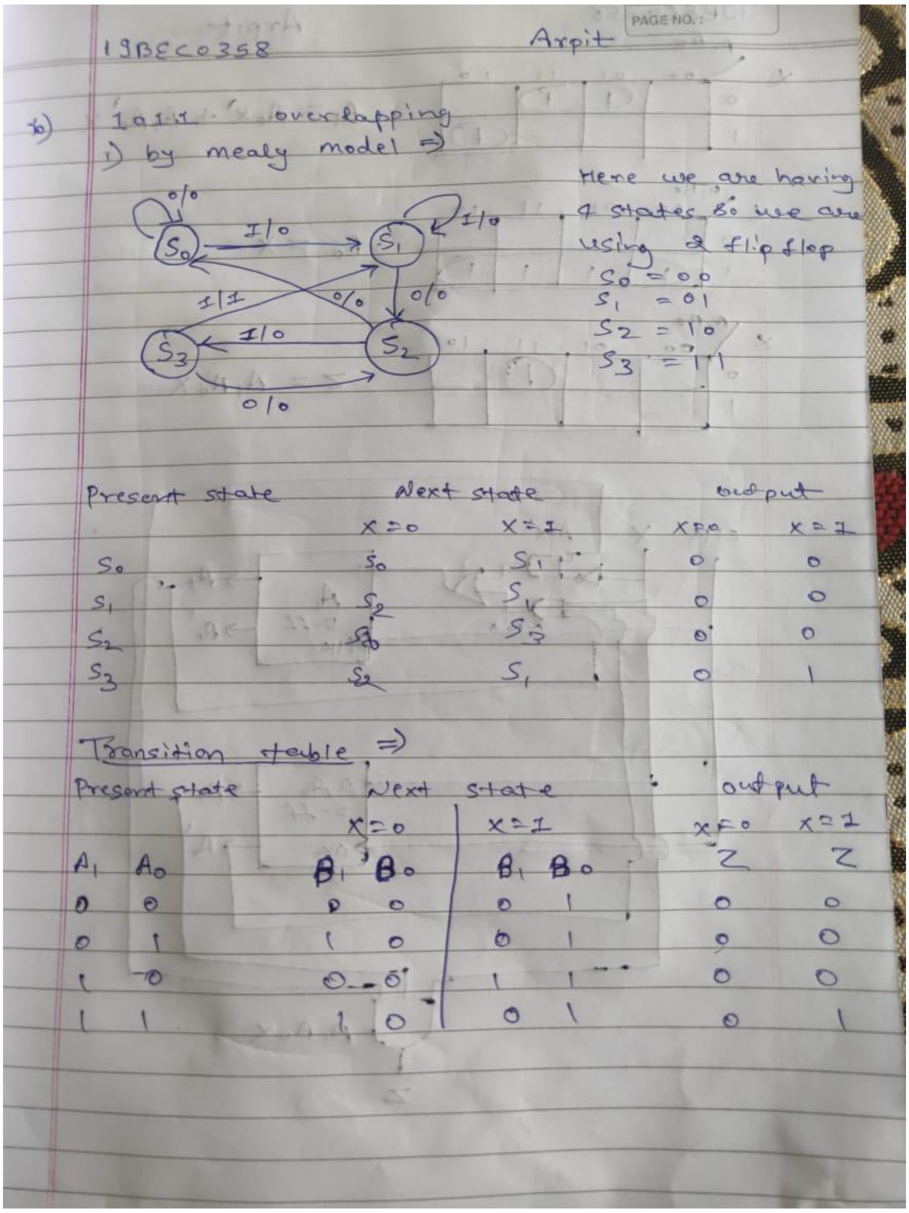
```
/bcd_counter/clk   St0
/bcd_counter/A     0001
/bcd_counter/B     0010
/bcd_counter/T     0011
```

```
force -freeze {sim:/bcd_counter/A[1]} 0 0, 1
{50 ps} -r 100
force -freeze {sim:/bcd_counter/A[0]} 0 0, 1
{25 ps} -r 50
VSIM 105> run
VSIM 106> restart
force -freeze sim:/bcd_counter/clk 1 0, 0 {12
ps} -r 25
force -freeze {sim:/bcd_counter/A[3]} 0 0, 1
{200 ps} -r 400
force -freeze {sim:/bcd_counter/A[2]} 0 0, 1
{100 ps} -r 200
force -freeze {sim:/bcd_counter/A[1]} 0 0, 1
{50 ps} -r 100
force -freeze {sim:/bcd_counter/A[0]} 0 0, 1
{25 ps} -r 50
force -freeze sim:/bcd_counter/clk 1 0, 0 {12
ps} -r 25
VSIM 113> run

VSIM 114>
```

```
0 ps to 250 ps    Project : task 6    Now: 400 ps  Delta: 2    sim:/bcd_counter
```

b) Write a Verilog code for 1011 **overlapping** sequence detector (**Moore or Mealy model**)



PAGE NO.:
0/0
1/0
1/0
1/1
0/0
0/0
1/0
0/0

19BEC0358

Arpit

b) 1011 overlapping

1) by mealy model ⇒

Here we are having 4 states so we are using 2 flip flop

$S_0 = 00$

$S_1 = 01$

$S_2 = 10$

$S_3 = 11$

Present state	Next state		output	
	x = 0	x = 1	x = 0	x = 1
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_3	0	0
S_3	S_2	S_1	0	1

Transition table ⇒

Present state		Next state				output	
		x = 0		x = 1		x = 0	x = 1
A_1	A_0	B_1	B_0	B_1	B_0	Z	Z
0	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0
1	0	0	0	1	1	0	0
1	1	1	0	0	1	0	1

19BEC0358 Arpit

DATE : / /
PAGE NO. :

$X \backslash A_1A_0$	00	01	11	10
0		1	1	
1				1

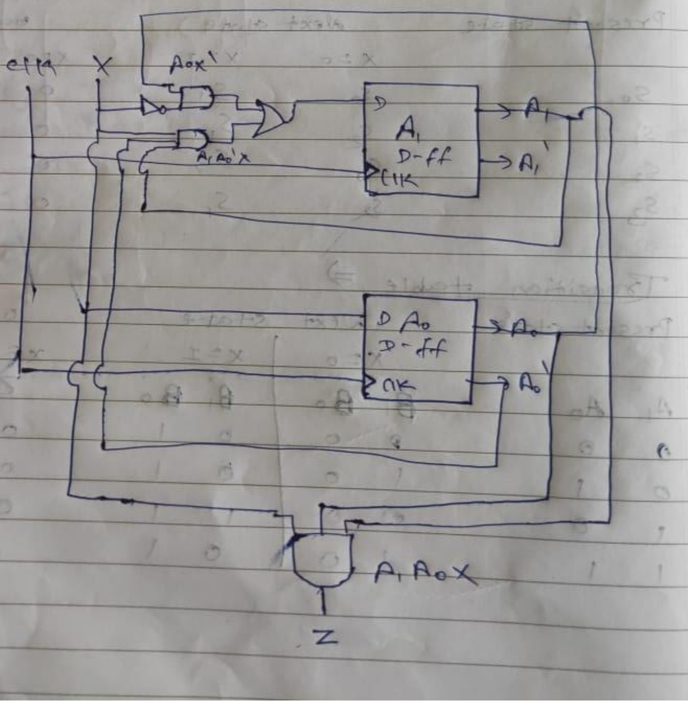
$A_1 = A_0X' + A_1A_0'X$

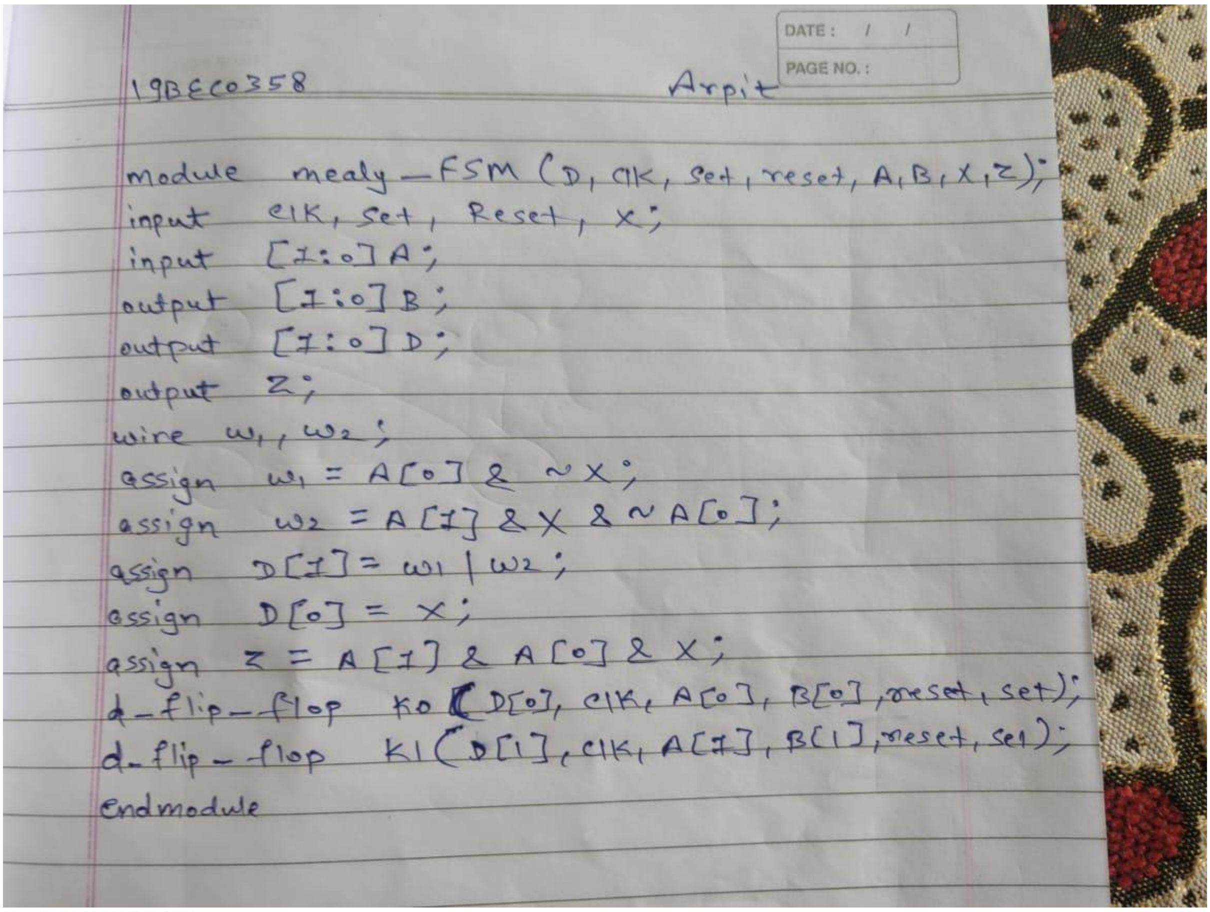
$X \backslash A_1A_0$	00	01	11	10
0				
1	1	1	1	1

$B_0 = X$

$X \backslash A_1A_0$	00	01	11	10
0			1	
1				

$Z = A_1A_0X$



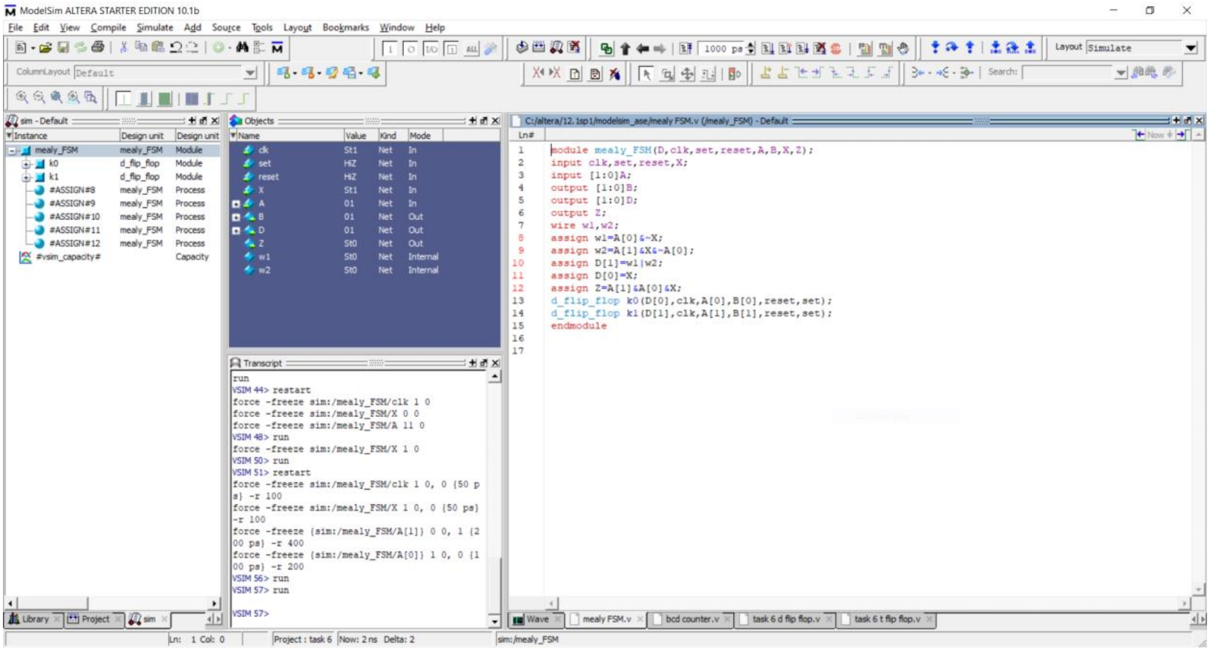


DATE : / /

PAGE NO. :

19BEC0358 Arpit

```
module mealy-FSM (D, clk, Set, reset, A, B, X, Z);
input clk, Set, Reset, X;
input [1:0] A;
output [1:0] B;
output [1:0] D;
output Z;
wire w1, w2;
assign w1 = A[0] & ~X;
assign w2 = A[1] & X & ~A[0];
assign D[1] = w1 | w2;
assign D[0] = X;
assign Z = A[1] & A[0] & X;
d-flip-flop k0 (D[0], clk, A[0], B[0], reset, set);
d-flip-flop k1 (D[1], clk, A[1], B[1], reset, set);
endmodule
```



Instance	Design unit	Design unit
mealy_FSM	mealy_FSM	Module
k0	d_flip_flop	Module
k1	d_flip_flop	Module
#ASSIGN#8	mealy_FSM	Process
#ASSIGN#9	mealy_FSM	Process
#ASSIGN#10	mealy_FSM	Process
#ASSIGN#11	mealy_FSM	Process
#ASSIGN#12	mealy_FSM	Process
#vsim_capacity#		Capacity

Name	Value	Kind	Mode
clk	St1	Net	In
set	HiZ	Net	In
reset	HiZ	Net	In
X	St1	Net	In
A	01	Net	In
B	01	Net	Out
D	01	Net	Out
Z	St0	Net	Out
w1	St0	Net	Internal
w2	St0	Net	Internal

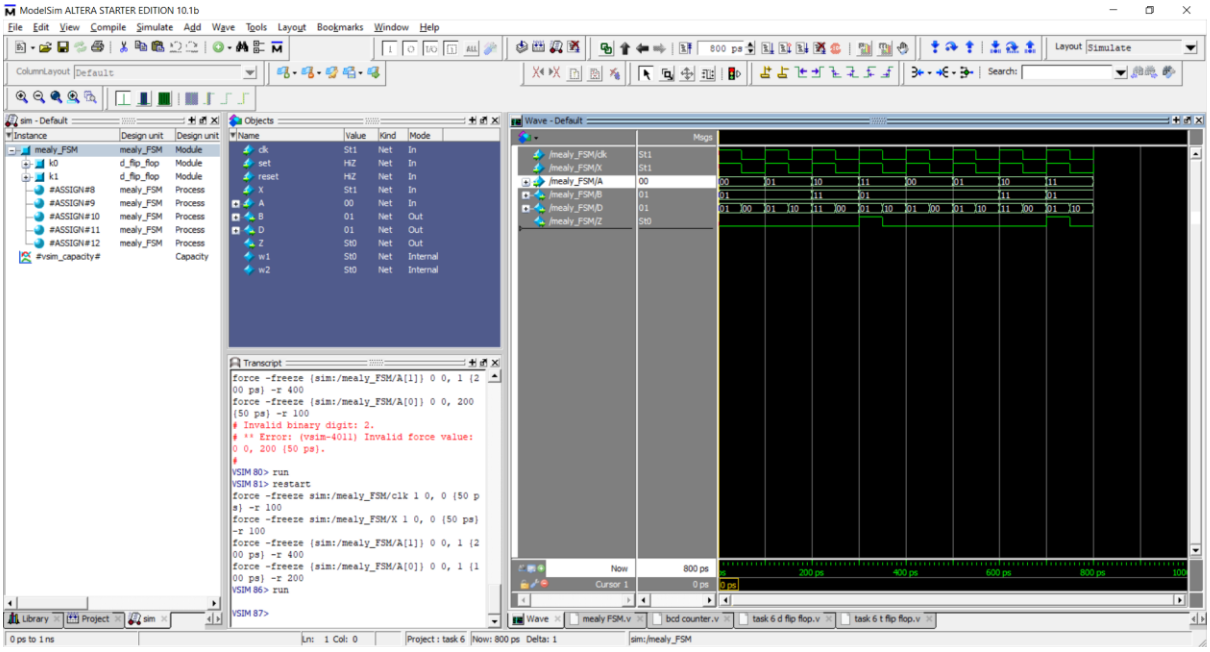
```
run
VSIM 44> restart
force -freeze sim:/mealy_FSM/clk 1 0
force -freeze sim:/mealy_FSM/X 0 0
force -freeze sim:/mealy_FSM/A 11 0
VSIM 48> run
force -freeze sim:/mealy_FSM/X 1 0
VSIM 50> run
VSIM 51> restart
force -freeze sim:/mealy_FSM/clk 1 0, 0 {50 p
s} -r 100
force -freeze sim:/mealy_FSM/X 1 0, 0 {50 ps}
-r 100
force -freeze {sim:/mealy_FSM/A[1]} 0 0, 1 {2
00 ps} -r 400
force -freeze {sim:/mealy_FSM/A[0]} 1 0, 0 {1
00 ps} -r 200
VSIM 56> run
VSIM 57> run

VSIM 57>
```

C:/altera/12.1sp1/modelsim_ase/mealy FSM.v (/mealy_FSM) - Default

```
module mealy_FSM(D,clk,set,reset,A,B,X,Z);
input clk,set,reset,X;
input [1:0]A;
output [1:0]B;
output [1:0]D;
output Z;
wire w1,w2;
assign w1=A[0]&~X;
assign w2=A[1]&X&~A[0];
assign D[1]=w1|w2;
assign D[0]=X;
assign Z=A[1]&A[0]&X;
d_flip_flop k0(D[0],clk,A[0],B[0],reset,set);
d_flip_flop k1(D[1],clk,A[1],B[1],reset,set);
endmodule
```

Ln: 1 Col: 0 | Project : task 6 | Now: 2 ns Delta: 2 | sim:/mealy_FSM



Instance	Design unit	Design unit
mealy_FSM	mealy_FSM	Module
k0	d_flip_flop	Module
k1	d_flip_flop	Module
#ASSIGN#8	mealy_FSM	Process
#ASSIGN#9	mealy_FSM	Process
#ASSIGN#10	mealy_FSM	Process
#ASSIGN#11	mealy_FSM	Process
#ASSIGN#12	mealy_FSM	Process
#vsim_capacity#		Capacity

Name	Value	Kind	Mode
clk	St1	Net	In
set	HiZ	Net	In
reset	HiZ	Net	In
X	St1	Net	In
A	00	Net	In
B	01	Net	Out
D	01	Net	Out
Z	St0	Net	Out
w1	St0	Net	Internal
w2	St0	Net	Internal

```
force -freeze {sim:/mealy_FSM/A[1]} 0 0, 1 {2
00 ps} -r 400
force -freeze {sim:/mealy_FSM/A[0]} 0 0, 200
{50 ps} -r 100
# Invalid binary digit: 2.
# ** Error: (vsim-4011) Invalid force value:
0 0, 200 {50 ps}.
#
VSIM 80> run
VSIM 81> restart
force -freeze sim:/mealy_FSM/clk 1 0, 0 {50 p
s} -r 100
force -freeze sim:/mealy_FSM/X 1 0, 0 {50 ps}
-r 100
force -freeze {sim:/mealy_FSM/A[1]} 0 0, 1 {2
00 ps} -r 400
force -freeze {sim:/mealy_FSM/A[0]} 0 0, 1 {1
00 ps} -r 200
VSIM 86> run

VSIM 87>
```

Here A is initial state, B is final state, X is input and Z is output.

DATE : / /

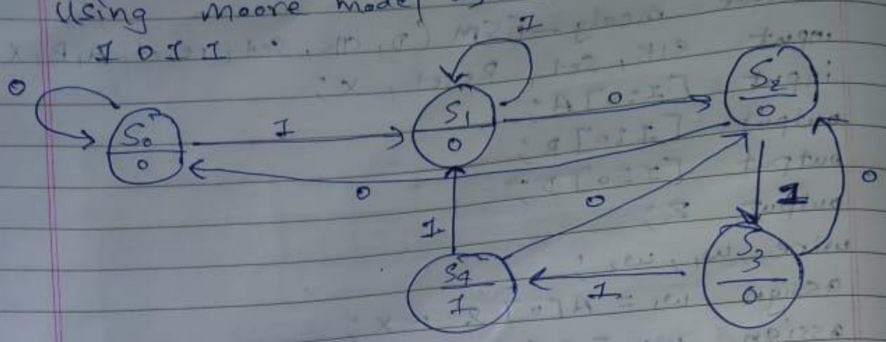
PAGE NO. :

Arpit

19BEC0358

Using moore model →

1011



Present state	Next state x=0	x=1	Present output Z
S0	S0	S1	0
S1	S2	S1	0
S2	S0	S3	0
S3	S2	S4	0
S4	S2	S1	1

	Present state			Next state x=0			x=1			output
	A_2	A_1	A_0	B_2	B_1	B_0	B_2	B_1	B_0	Z
	0	0	0	0	0	0	0	0	1	0
	0	0	1	0	1	0	0	0	1	0
	0	1	0	0	0	0	0	1	1	0
	0	1	1	0	1	0	1	0	0	0
	1	0	0	0	1	0	0	0	1	1
don't Care Contion	1	0	1	X			X			X
	1	1	0	X			X			X
	1	1	1	X			X			X

DATE : / /
PAGE NO. :

19BEC0358 Arpit

in Moore Model, we Have n+1 states. Since Here we are having 5 states, we need 3 flip-flops.
$S_0 = 000$, $S_1 = 001$, $S_2 = 010$, $S_3 = 011$, $S_4 = 100$

$B_2 \Rightarrow$

A_2A_1 \ A_0X	00	01	11	10
00				
01			1	
11	X	X	X	X
10			X	X

$$\boxed{B_2 = A_1 A_0 X}$$

$B_1 \Rightarrow$

A_2A_1 \ A_0X	00	01	11	10
00				1
01		1		1
11	X	X	X	X
10	1		X	X

$$\boxed{B_1 = A_0 X' + A_1 A_0' X + A_2 A_0' X'}$$

$B_0 \Rightarrow$

A_2A_1 \ A_0X	00	01	11	10
00		1	1	
01		1		
11	X	X	X	X
10		1	X	X

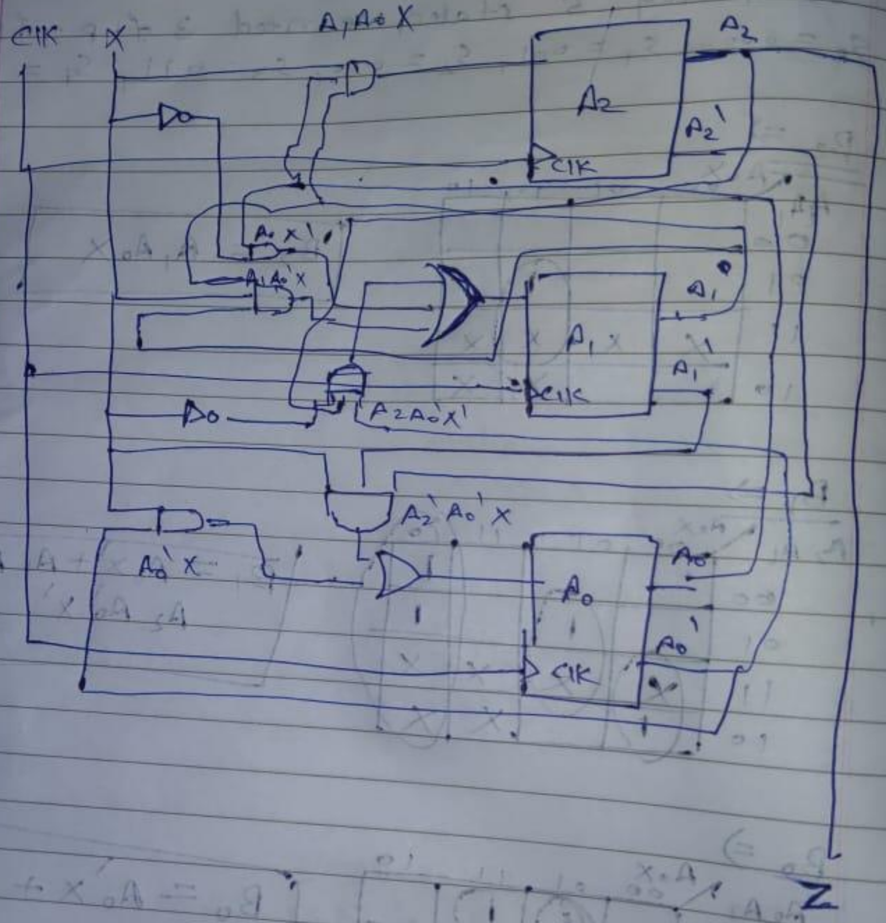
$$\boxed{B_0 = A_0' X + A_2' A_1' X}$$

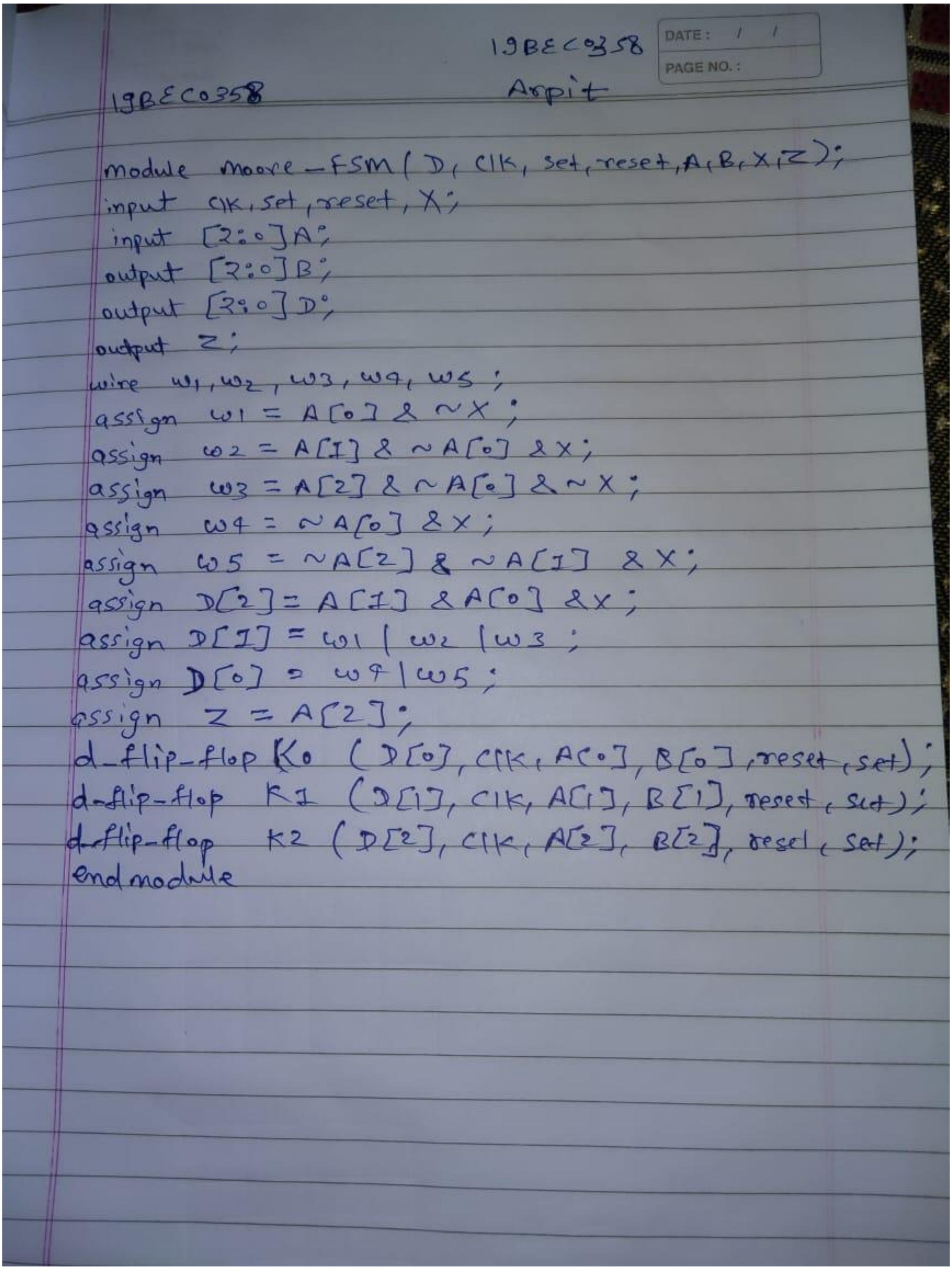
$Z \Rightarrow$

A_2A_1 \ A_0X	00	01	11	10
00				
01				
11	X	X	X	X
10	1	1	λ	λ

$$\boxed{Z = A_2}$$

19BEC0358 Arpit





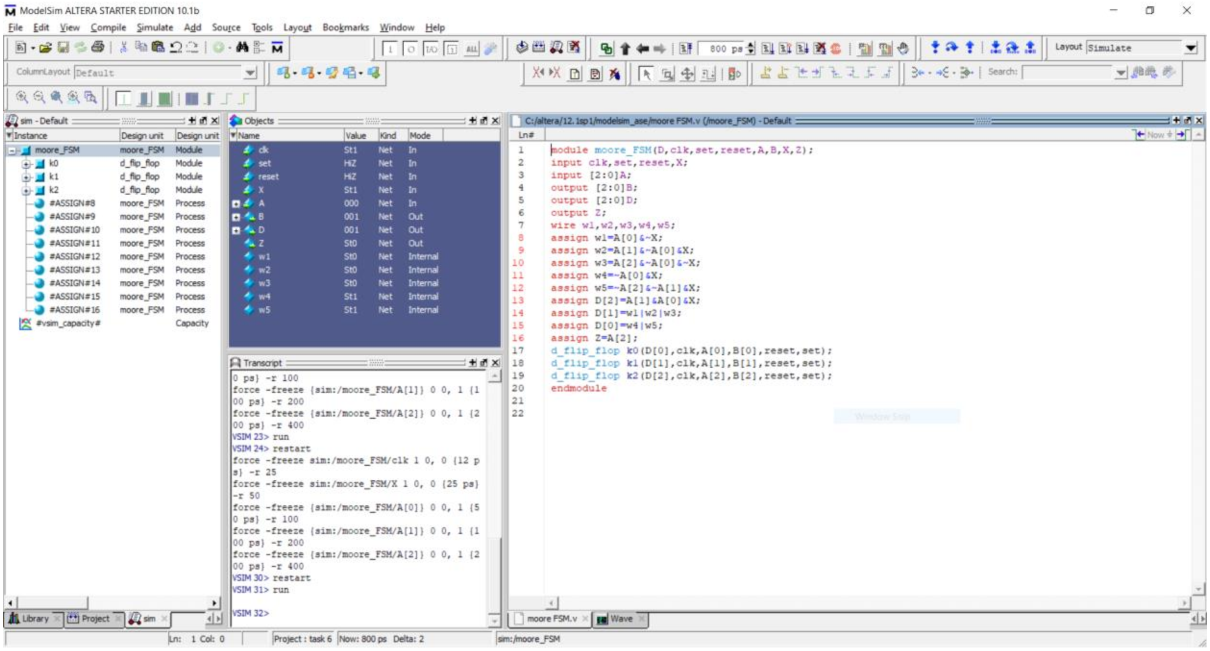
19BEC0358

DATE : / /

PAGE NO. :

19BEC0358 Arpit

```
module moore-FSM(D, Clk, set, reset, A, B, X, Z);
input Clk, set, reset, X;
input [2:0]A;
output [2:0]B;
output [2:0]D;
output Z;
wire w1, w2, w3, w4, w5;
assign w1 = A[0] & ~X;
assign w2 = A[1] & ~A[0] & X;
assign w3 = A[2] & ~A[0] & ~X;
assign w4 = ~A[0] & X;
assign w5 = ~A[2] & ~A[1] & X;
assign D[2] = A[1] & A[0] & X;
assign D[1] = w1 | w2 | w3;
assign D[0] = w4 | w5;
assign Z = A[2];
d_flip_flop K0 (D[0], Clk, A[0], B[0], reset, set);
d_flip_flop K1 (D[1], Clk, A[1], B[1], reset, set);
d_flip_flop K2 (D[2], Clk, A[2], B[2], resel, set);
endmodule
```


ModelSim ALTERA STARTER EDITION 10.1b
File Edit View Compile Simulate Add Source Tools Layout Bookmarks Window Help
ColumnLayout Default
800 ps
Layout Simulate
Search:
sim - Default
Objects
Transcript
Library Project sim
moore FSM.v Wave

Instance	Design unit	Design unit
moore_FSM	moore_FSM	Module
k0	d_flip_flop	Module
k1	d_flip_flop	Module
k2	d_flip_flop	Module
#ASSIGN#8	moore_FSM	Process
#ASSIGN#9	moore_FSM	Process
#ASSIGN#10	moore_FSM	Process
#ASSIGN#11	moore_FSM	Process
#ASSIGN#12	moore_FSM	Process
#ASSIGN#13	moore_FSM	Process
#ASSIGN#14	moore_FSM	Process
#ASSIGN#15	moore_FSM	Process
#ASSIGN#16	moore_FSM	Process
#vsim_capacity#		Capacity

Name	Value	Kind	Mode
clk	St1	Net	In
set	HiZ	Net	In
reset	HiZ	Net	In
X	St1	Net	In
A	000	Net	In
B	001	Net	Out
D	001	Net	Out
Z	St0	Net	Out
w1	St0	Net	Internal
w2	St0	Net	Internal
w3	St0	Net	Internal
w4	St1	Net	Internal
w5	St1	Net	Internal

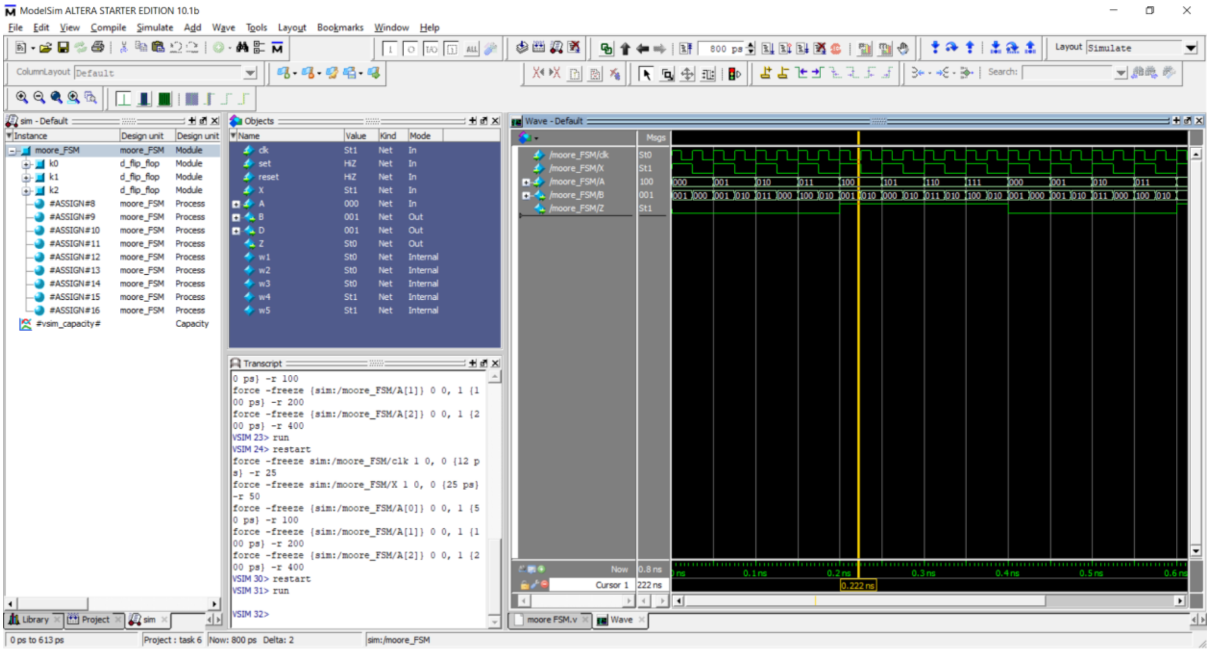
```
0 ps} -r 100
force -freeze {sim:/moore_FSM/A[1]} 0 0, 1 {1
00 ps} -r 200
force -freeze {sim:/moore_FSM/A[2]} 0 0, 1 {2
00 ps} -r 400
VSIM 23> run
VSIM 24> restart
force -freeze sim:/moore_FSM/clk 1 0, 0 {12 p
s} -r 25
force -freeze sim:/moore_FSM/X 1 0, 0 {25 ps}
-r 50
force -freeze {sim:/moore_FSM/A[0]} 0 0, 1 {5
0 ps} -r 100
force -freeze {sim:/moore_FSM/A[1]} 0 0, 1 {1
00 ps} -r 200
force -freeze {sim:/moore_FSM/A[2]} 0 0, 1 {2
00 ps} -r 400
VSIM 30> restart
VSIM 31> run

VSIM 32>
```

C:/altera/12.1sp1/modelsim_ase/moore FSM.v (/moore_FSM) - Default

```
module moore_FSM(D,clk,set,reset,A,B,X,Z);
input clk,set,reset,X;
input [2:0]A;
output [2:0]B;
output [2:0]D;
output Z;
wire w1,w2,w3,w4,w5;
assign w1=A[0]&~X;
assign w2=A[1]&~A[0]&X;
assign w3=A[2]&~A[0]&~X;
assign w4=~A[0]&X;
assign w5=~A[2]&~A[1]&X;
assign D[2]=A[1]&A[0]&X;
assign D[1]=w1|w2|w3;
assign D[0]=w4|w5;
assign Z=A[2];
d_flip_flop k0(D[0],clk,A[0],B[0],reset,set);
d_flip_flop k1(D[1],clk,A[1],B[1],reset,set);
d_flip_flop k2(D[2],clk,A[2],B[2],reset,set);
endmodule
```

Ln: 1 Col: 0 | Project : task 6 | Now: 800 ps Delta: 2 | sim:/moore_FSM



Instance	Design unit	Design unit
moore_FSM	moore_FSM	Module
k0	d_flip_flop	Module
k1	d_flip_flop	Module
k2	d_flip_flop	Module
#ASSIGN#8	moore_FSM	Process
#ASSIGN#9	moore_FSM	Process
#ASSIGN#10	moore_FSM	Process
#ASSIGN#11	moore_FSM	Process
#ASSIGN#12	moore_FSM	Process
#ASSIGN#13	moore_FSM	Process
#ASSIGN#14	moore_FSM	Process
#ASSIGN#15	moore_FSM	Process
#ASSIGN#16	moore_FSM	Process
#vsim_capacity#		Capacity

Name	Value	Kind	Mode
clk	St1	Net	In
set	HiZ	Net	In
reset	HiZ	Net	In
X	St1	Net	In
A	000	Net	In
B	001	Net	Out
D	001	Net	Out
Z	St0	Net	Out
w1	St0	Net	Internal
w2	St0	Net	Internal
w3	St0	Net	Internal
w4	St1	Net	Internal
w5	St1	Net	Internal

```
0 ps} -r 100
force -freeze {sim:/moore_FSM/A[1]} 0 0, 1 {1
00 ps} -r 200
force -freeze {sim:/moore_FSM/A[2]} 0 0, 1 {2
00 ps} -r 400
VSIM 23> run
VSIM 24> restart
force -freeze sim:/moore_FSM/clk 1 0, 0 {12 p
s} -r 25
force -freeze sim:/moore_FSM/X 1 0, 0 {25 ps}
-r 50
force -freeze {sim:/moore_FSM/A[0]} 0 0, 1 {5
0 ps} -r 100
force -freeze {sim:/moore_FSM/A[1]} 0 0, 1 {1
00 ps} -r 200
force -freeze {sim:/moore_FSM/A[2]} 0 0, 1 {2
00 ps} -r 400
VSIM 30> restart
VSIM 31> run

VSIM 32>
```

0 ps to 613 ps | Project : task 6 | Now: 800 ps Delta: 2 | sim:/moore_FSM

Here A is initial state, B is final state, X is input and Z is output.