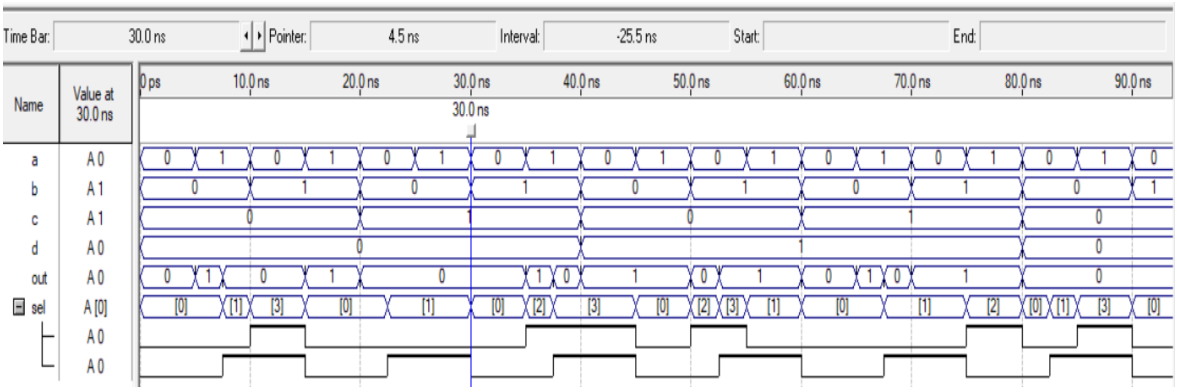
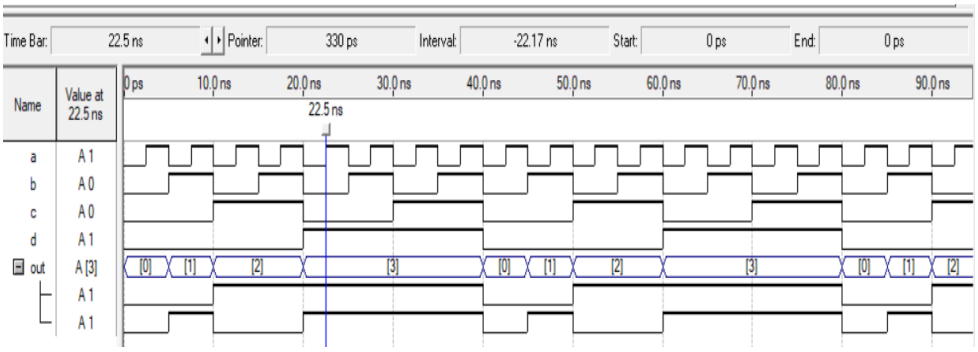
Problem 1:

```
module Prob1 ( input a,
input  b,
input  c,
input  d,
input [1:0] sel,
output  out);
assign out = sel[1] ? (sel[0] ? d : c) : (sel[0] ? b : a);
endmodule
```



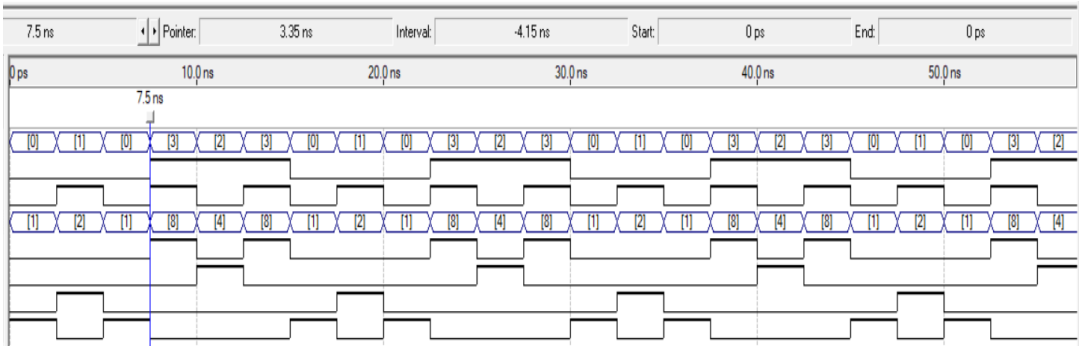
Problem 2 :

```
module Prob2 ( input a,
input  b,
input  c,
input  d,
output[1:0] out);
assign out = d ? 3 : (c ? 2 : (b ? 1 : 0 ) );
endmodule
```



Problem 3:

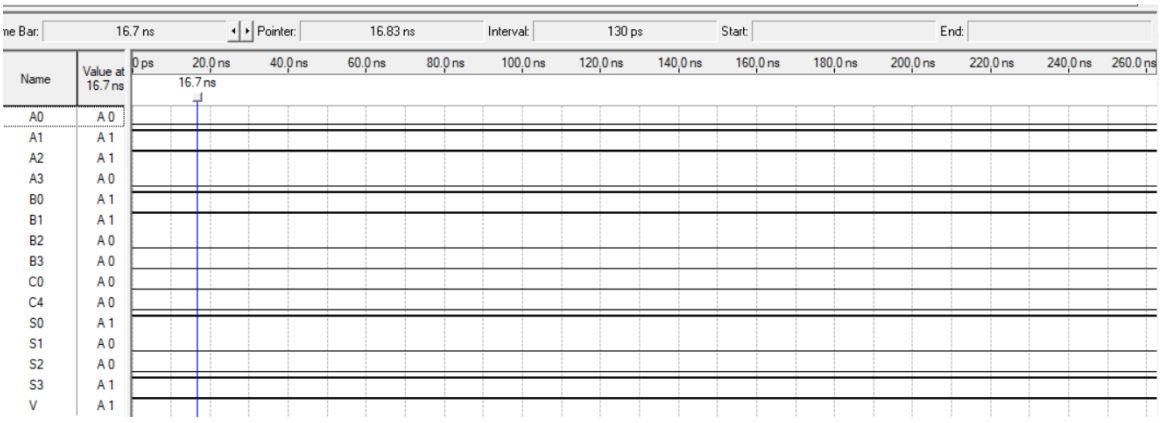
```
module Prob3 ( input [1:0] a,
output [3:0] out );
assign out = a[1] ? (a[0] ? 8 : 4) : (a[0] ? 2 : 1);
endmodule
```



Problem 4:

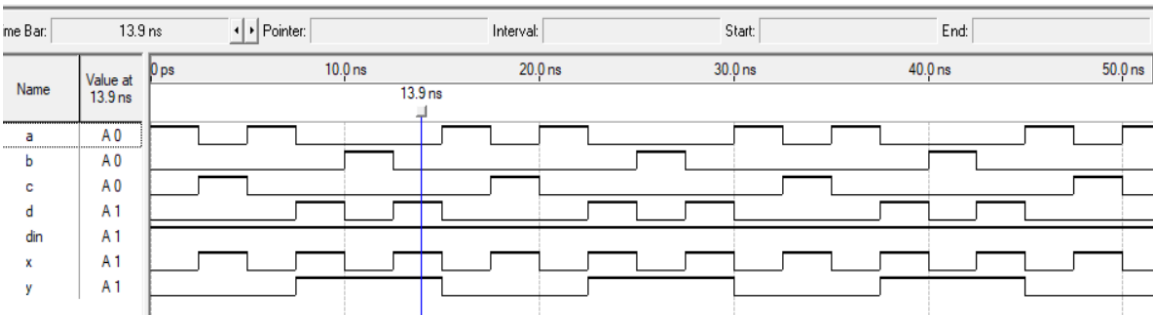
```
module Prob4(A0,A1,A2,A3,B0,B1,B2,B3,S0,S1,S2,S3,C0,C4,V);
input A0,A1,A2,A3;
input B0,B1,B2,B3;
input C0;
output S0,S1,S2,S3;
output C4,V;
wire A0,A1,A2,A3;
wire B0,B1,B2,B3;
```

```
wire C0;
wire S0,S1,S2,S3;
wire C4,V;
wire C1,C2,C3;
assign X0 = C0^B0;
assign X1 = C0^B1;
assign X2 = C0^B2;
assign X3 = C0^B3;
assign V = C3^C4;
fulladder Y0(C0,X0,A0,S0,C1);
fulladder Y1(C1,X1,A1,S1,C2);
fulladder Y2(C2,X2,A2,S2,C3);
fulladder Y3(C3,X3,A3,S3,C4);
endmodule
module fulladder(a,b,cin,sum,cout);
input a,b,cin;
output sum,cout;
assign {cout,sum} = a+b+cin;
endmodule
```



Problem 5(a):

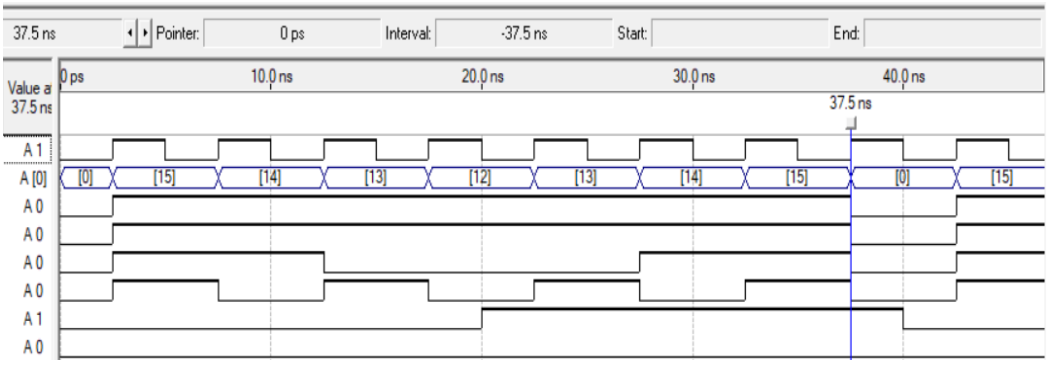
```
module Prob5a ( din ,x ,y ,a ,b ,c ,d );
output a ;
output b ;
output c ;
output d ;
input din ;
input x ;
input y ;
assign a = din & (~x) & (~y);
assign b = din & (~x) & y;
assign c = din & x & (~y);
assign d = din & x & y;
endmodule
```



Problem5(b):

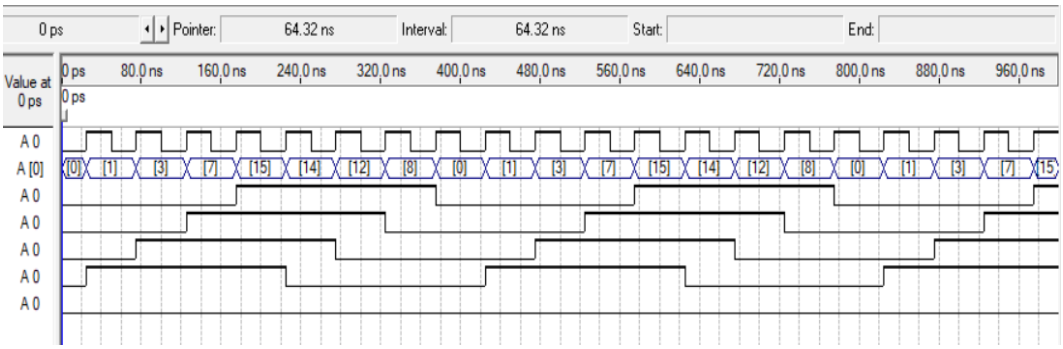
```
module Prob5b(
Clk,
reset,
UpOrDown,  //high for UP counter and low for Down counter
Count
);
```

```
input Clk,reset,UpOrDown;
output [3 : 0] Count;
reg [3 : 0] Count = 0;
always @(posedge(Clk) or posedge(reset))
begin
if(reset == 1)
Count <= 0;
else
if(UpOrDown == 1)   //Up mode selected
if(Count == 15)
Count <= 0;
else
Count <= Count + 1; //Incremend Counter
else  //Down mode selected
if(Count == 0)
Count <= 15;
else
Count <= Count - 1; //Decrement counter
end
endmodule
```



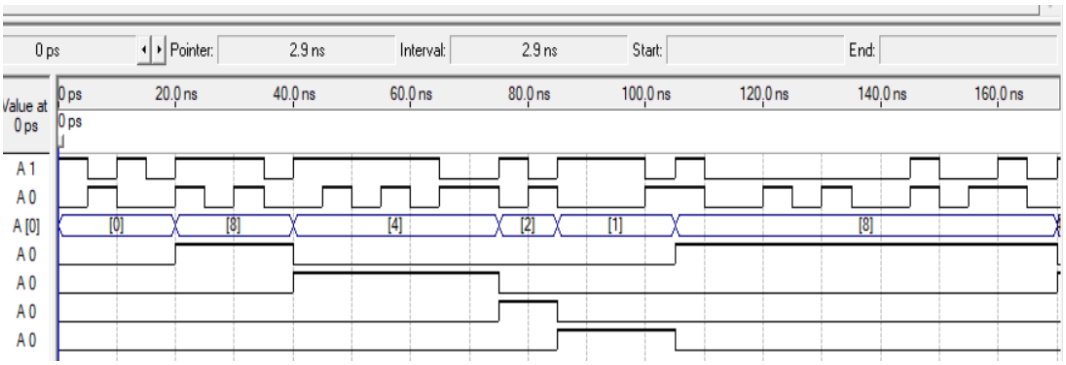
Problem 6 Johnson:

```
module Prob6johnson(
Clock,
Reset,
Count_out);
input Clock;
output [3:0] Count_out;
reg [3:0] Count_temp;
always @(posedge(Clock) or posedge(Reset))
begin
if(Reset == 1'b1)   begin
Count_temp = 4'b0000;   end
else if(Clock == 1'b1)  begin
Count_temp = {Count_temp[2:0],~Count_temp[3]};  end
end
assign Count_out = Count_temp;
endmodule
```



Problem 6 Ring Counter:

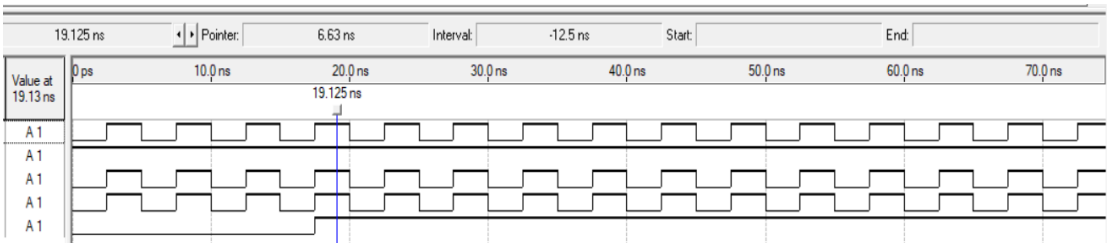
```
module Prob6ring(q,clk,clr);
input clk,clr;
output [3:0] q;
reg [3:0] a;
always @(posedge clk)
if(clr==1)
a<=4'b1000;
else
begin
a[3]<=a[0];
a[2]<=a[3];
a[1]<=a[2];
a[0]<=a[1];
end
assign q = a;
endmodule
```



Problem 7:

```
module Prob7 (Clock, Resetn, a,b, z);
input Clock, Resetn, a,b;
output z;
assign w=a^b;
reg [2:0] y;
parameter [2:0] S0=0, S1=1, S2=2, S3=3, S4=4;
always @(posedge Clock, negedge Resetn)
begin
if (Resetn == 0) y <= S0;
else
begin
case (y)
S0: if (w) y <= S0;
else y <= S1;
S1:
if (w) y <= S0;
else y<=S2;
S2:
if (w) y <= S0;
```

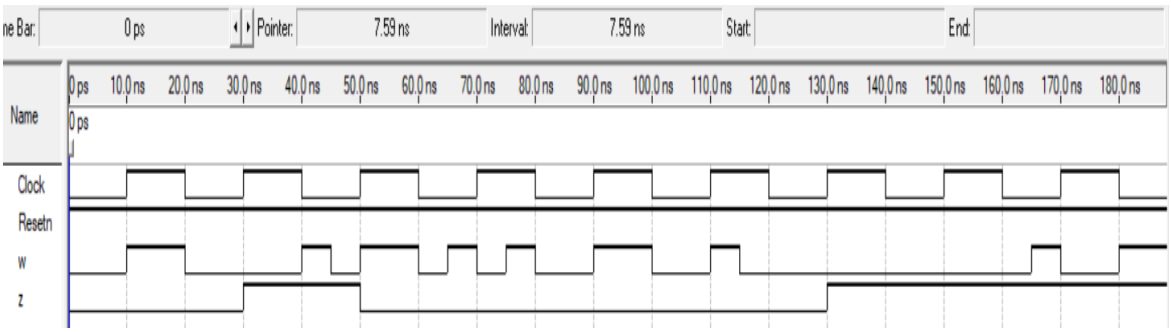
```
else y <= S3;
S3:
if (w) y <= S0;
else y <= S4;
S4:
if (w) y <= S0;
else y <= S4;
endcase
end
end
assign z=(y==S4);
endmodule
```



Problem 8:

```
module Prob8(Clock, Resetn, w, z);
input Clock, Resetn, w;
output z;
reg [2:0] y;
parameter [2:0] S0=0, S1=1, S2=2, S3=3;
always @(posedge Clock, negedge Resetn)
begin
```

```
if (Resetn == 0) y <= S0;
else
begin
case (y)
S0: if (w) y <= S3;
else y <= S2;
S1:
if (w) y <= S0;
else y<=S1;
S2:
if (w) y <= S0;
else y <= S3;
S3:
if (w) y <= S1;
else y <= S1;
endcase
end
end
assign z=(y==S1);
endmodule
```



State Diagram: