SIMULATION AND IMPLEMENTATION OF SEQUENTIAL LOGIC CIRCUITS

AIM:

To simulate and synthesis SR, JK, T, D - FLIPFLOP, COUNTER DESIGN using Xilinx ISE.

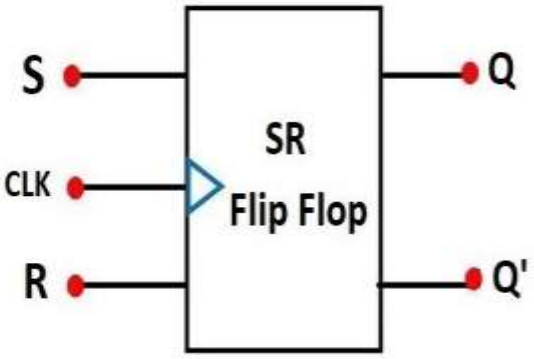
APPARATUS REQUIRED:

Xilinx 14.7 Spartan6 FPGA

LOGIC DIAGRAM:

SR FLIPFLOP:

S	R	Q	Q`
0	0	0	1
0	1	0	1
1	0	1	0
1	1	∞	∞

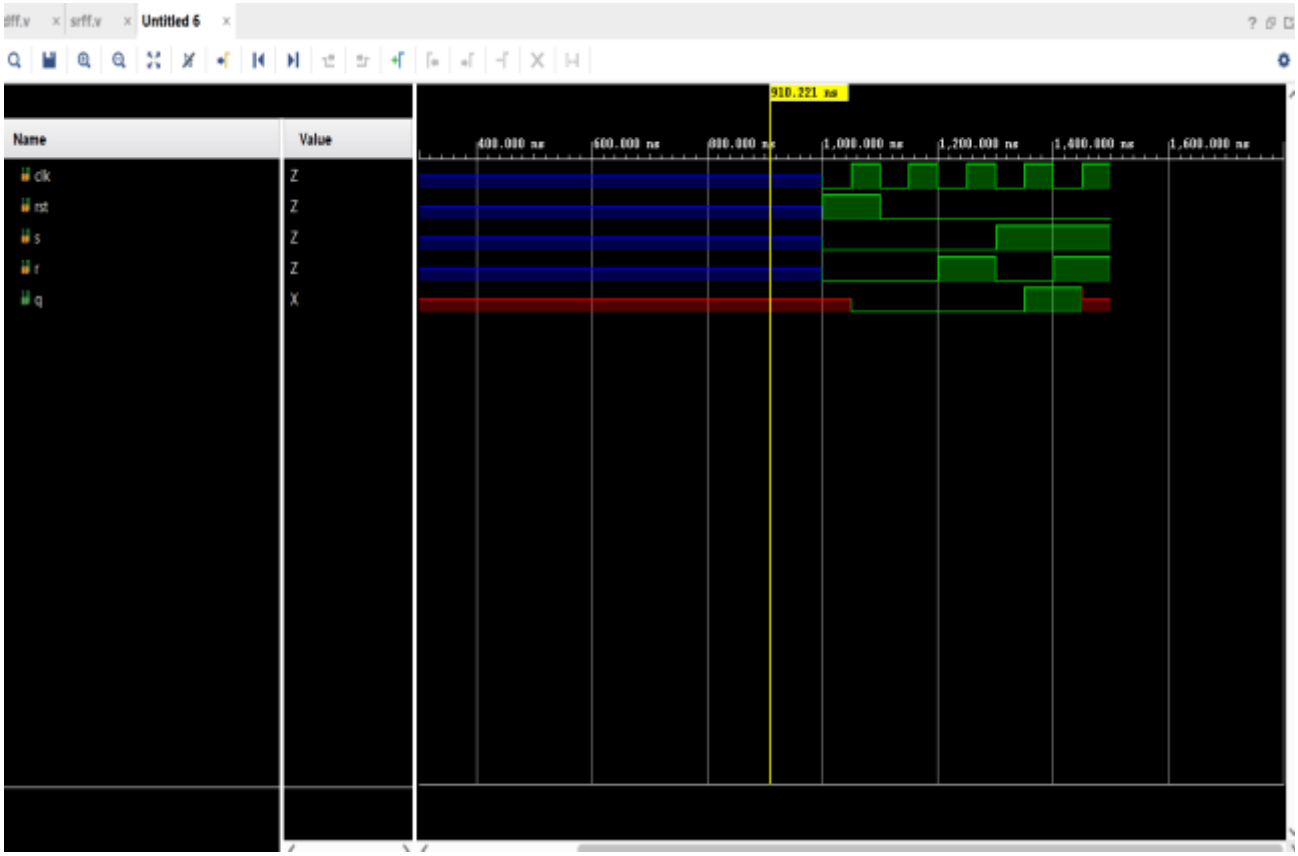


VERILOG CODE:

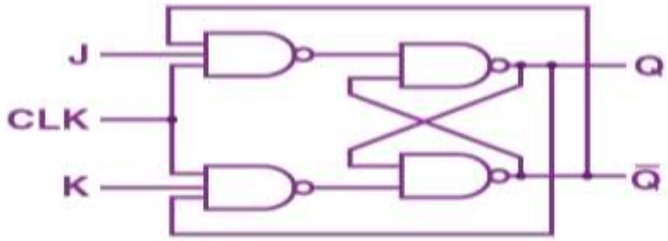
```
module srff(clk,rst,s,r,q);
input clk,rst,s,r;
output reg q;
always @(posedge clk)
begin
 if(rst)
  q<=1'b0;
 else
  begin
   case({s,r})
```

```
    2'b00:q<=q;
    2'b01:q<=1'b0;
    2'b10:q<=1'b1;
    2'b11:q<=1'bx;
    default:q<=1'bx;
   endcase
  end
 end
endmodule
```

OUTPUT:



JK FLIPFLOP:



Truth Table

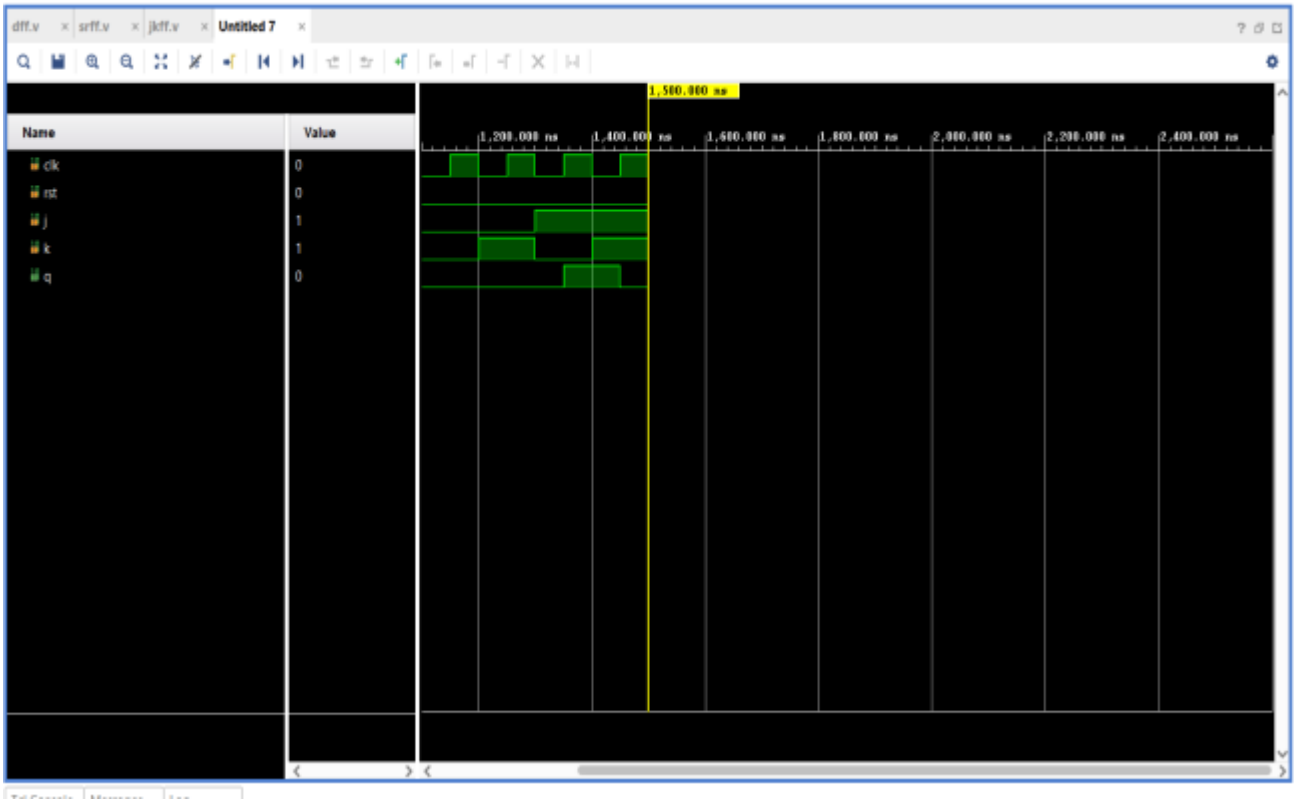
J	K	Q_N	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

VERILOG CODE:

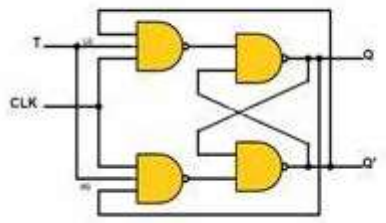
```
module jkff(clk,rst,j,k,q);
input clk,rst,j,k;
output reg q;
always @(posedge clk)
begin
 if(rst)
  q<=1'b0;
 else
  begin
   case({j,k})
    2'b00:q<=q;
```

```
    2'b01:q<=1'b0;
    2'b10:q<=1'b1;
    2'b11:q<=~q;
    default:q<=1'bx;
   endcase
  end
 end
endmodule
```

OUTPUT:



T FLIPFLOP:

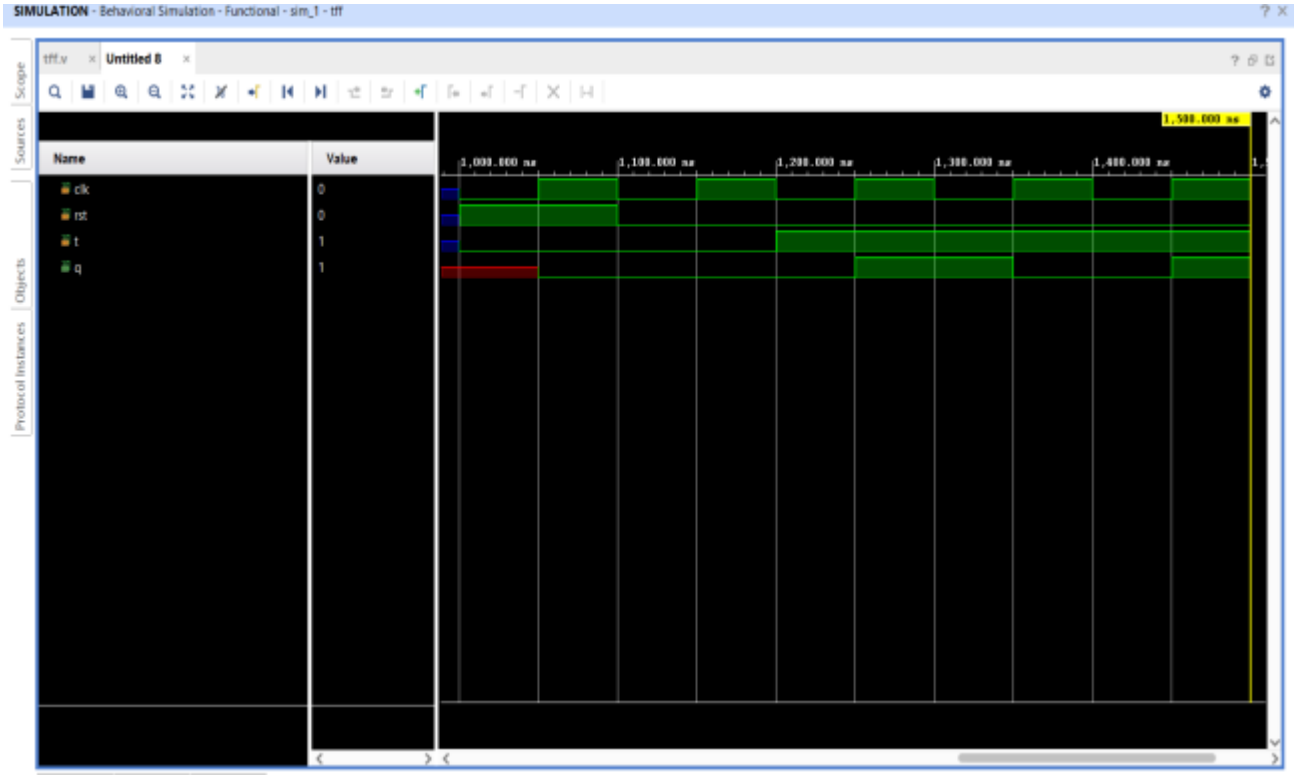


T	Q	Q'
0	0	0
1	0	1
0	1	0
1	1	0

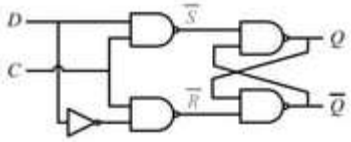
VERILOG CODE:

```
module tff(clk,rst,t,q);
input clk,rst,t;
output reg q;
always @(posedge clk)
  begin
   if(rst)
    q<=0;
   else if(t)
    q<=~q;
   else
    q<=q;
  end
endmodule
```

OUTPUT:



D FLIPFLOP:



C	D	Q	
0	x	Q_0	No change
1	0	0	Reset
1	1	1	Set

VERILOG CODE:

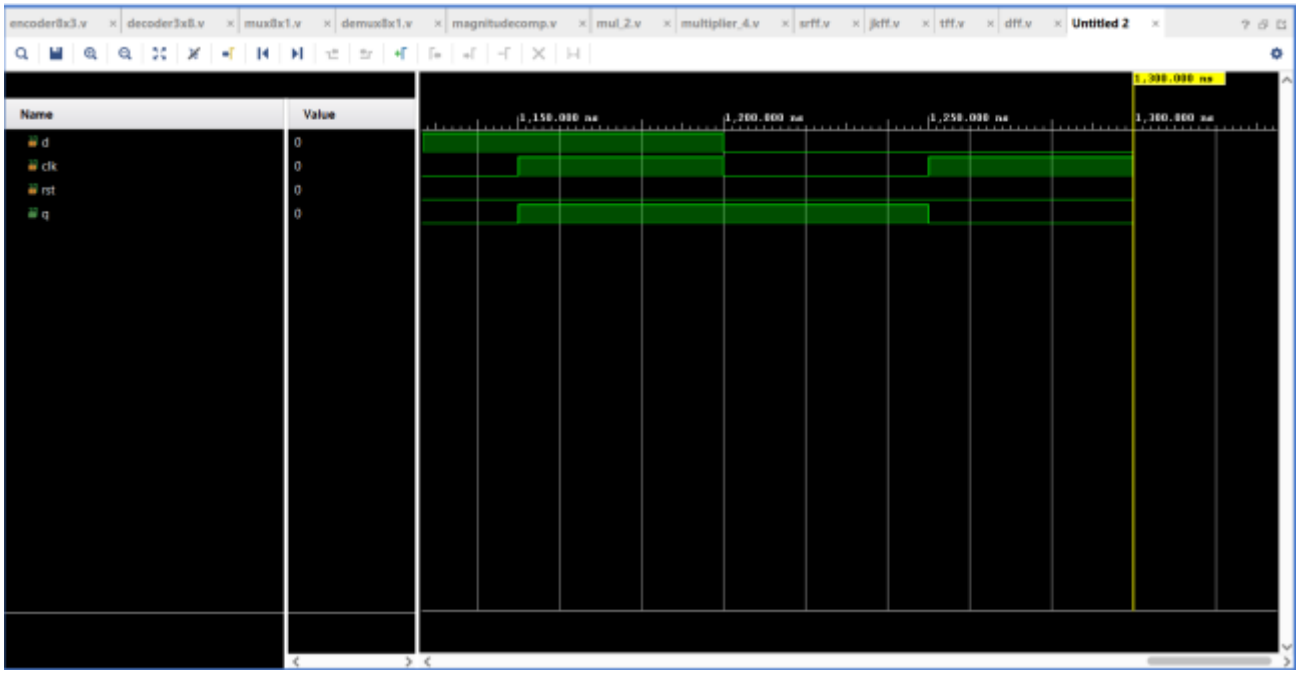
```
module dff(d,clk,rst,q);

input d,clk,rst;

output reg q;
```

```
always @(posedge clk)
begin
 if(rst)
   q <=1'b0;
 else
   q <= d;
end
endmodule
```

OUTPUT:



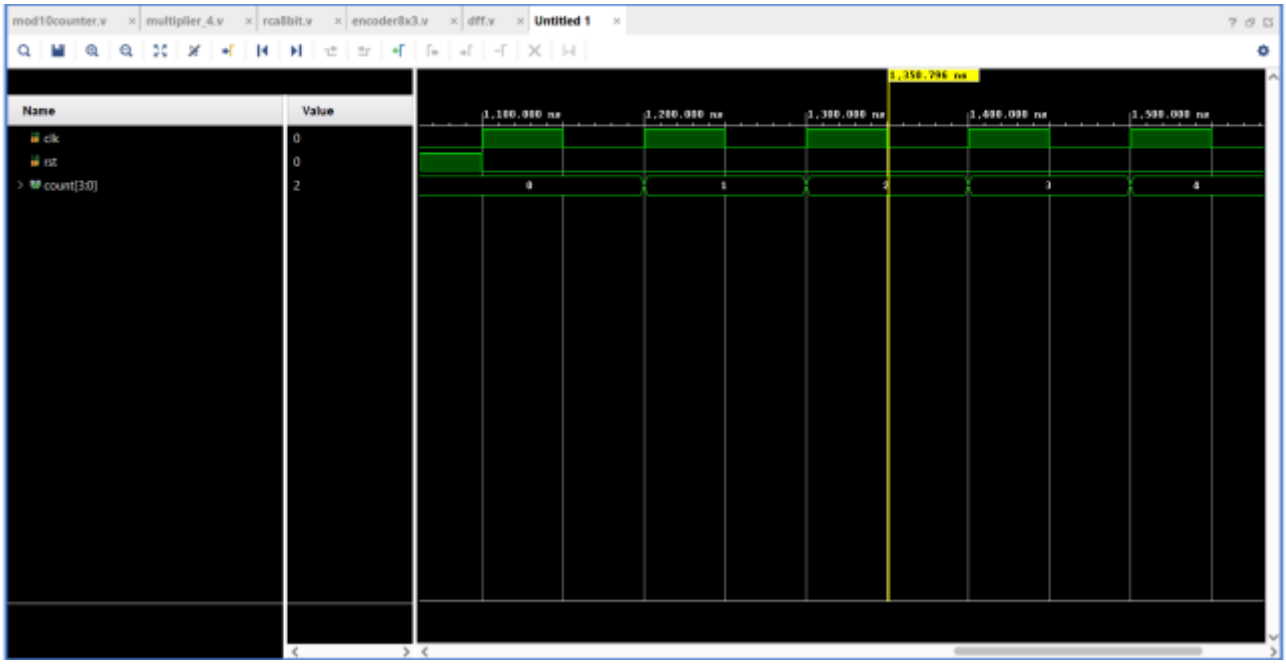
COUNTER

MOD10COUNTER:

VERILOG CODE:

```
module mod10counter(clk,rst,count);
input clk,rst;
output[3:0]count;
reg[3:0]count;
always@(posedge clk)
begin
if(rst|count==4'b1001)
count<=4'b0;
else
count<=count+1;
end
endmodule
```

OUTPUT:

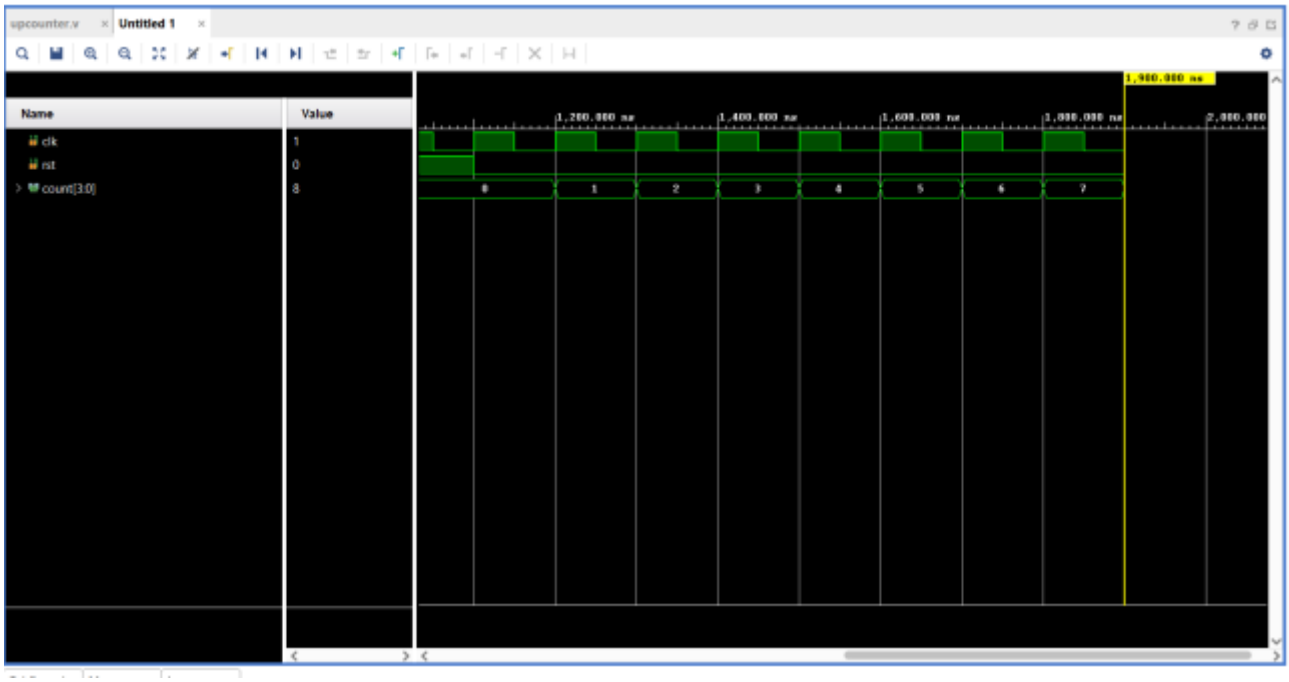


UPCOUNTER:

VERILOG CODE:

```
module upcounter(clk,rst,count);
input clk,rst;
output[3:0]count;
reg[3:0]count;
always@(posedge clk)
begin
if(rst)
count<=4'b0;
else
count<=count+1;
end
endmodule
```

OUTPUT:

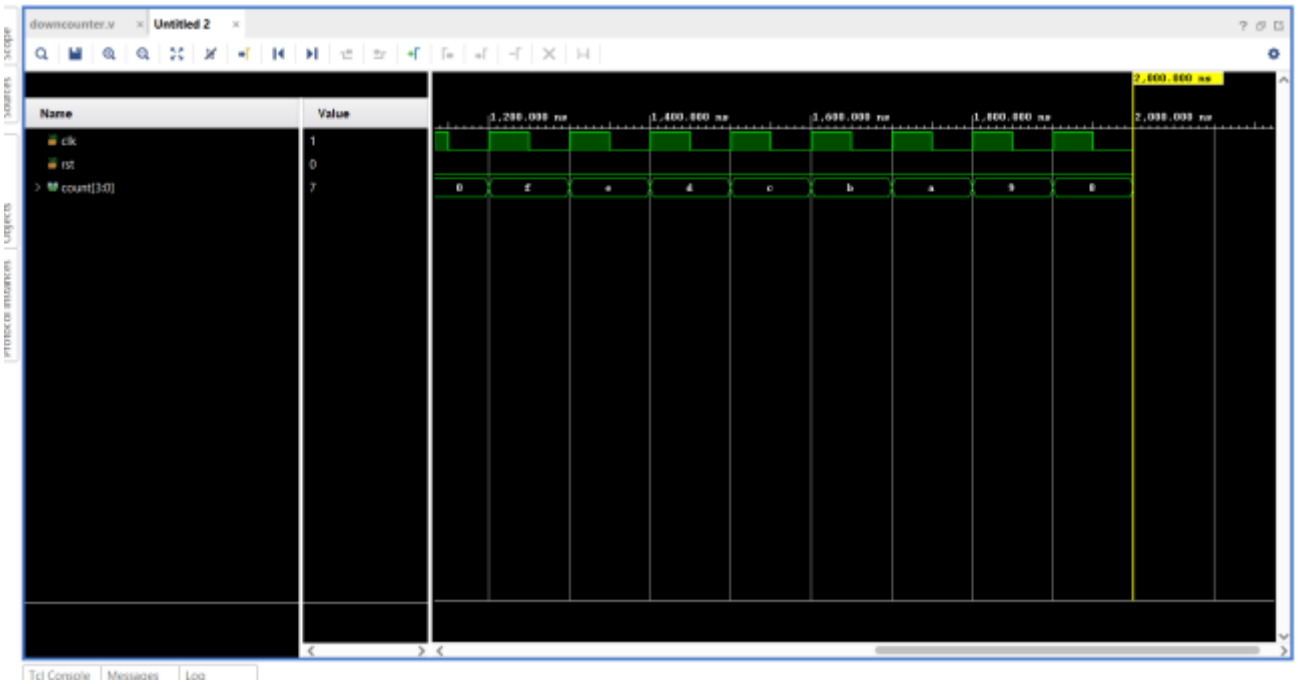


DOWN COUNTER:

VERILOG CODE:

```
module downcounter(clk,rst,count);
input clk,rst;
output[3:0]count;
reg[3:0]count;
always@(posedge clk)
begin
if(rst)
count<=4'b0;
else
count <=count-1;
end
endmodule
```

OUTPUT:



RIPPLE COUNTER:

VERILOG CODE:

```
module jkff(j,k,clock,reset,q,qb);
input j,k,clock,reset;
output reg q,qb;
always@(negedge clock)
begin
case({reset,j,k})
3'b100 :q=q;
3'b101 :q=0;
3'b110 :q=1;
3'b111 :q=~q;
default :q=0;
endcase
qb<=~q;
```

```
end
endmodule

module ripple_count(j,k,clock,reset,q,qb);
input j,k,clock,reset;
output wire [3:0]q,qb;

jkff JK1(j,k,clock,reset,q[0],qb[0]);
jkff JK2(j,k,q[0],reset,q[1],qb[1]);
jkff JK3(j,k,q[1],reset,q[2],qb[2]);
jkff JK4(j,k,q[2],reset,q[3],qb[3]);

endmodule
```

OUTPUT: