Name- Moitrish Maity Roll-48 2nd Yr CSE

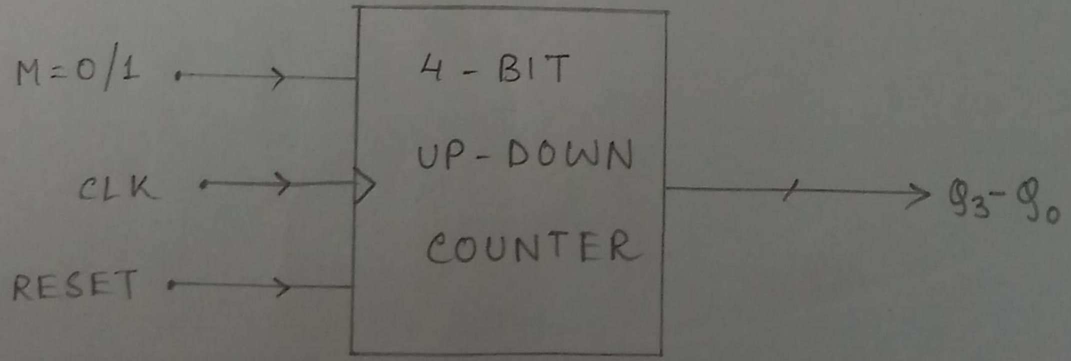
EXPERIMENT-10

- Title: Counters
- Objective: Implementation of Counters.
- Course Outcome: CO3
- Bloom's Level: Analyzing
- Problem Statement: Write verilog programs to design, simulate and test the following counters:

A) 4-bit updown counter
B) Random counter
C) Mod 10 counter
D) Ring Counter
E) Johnson counter.

(A) 4-bit Up-Down Counter

- Theory: 4bit up-down counter is a sequential circuit made up of four T or JK ff's. Depends upon the control signal, (M) counter value increases or decreases at applying of positive edge of clock. Here, we use M=1 for down counter and M=0 for up counting operation.
- Logic Diagram:



Function Table:

M	PRESENT STATE				NEXT STATE			
	Q_3	Q_2	Q_1	Q_0	Q_3^+	Q_2^+	Q_1^+	Q_0^+
L	L	L	L	L	L	L	L	H
L	L	L	L	H	L	L	H	L
L	L	L	H	L	L	L	H	H
L	L	L	H	H	L	H	L	L
L	L	H	L	L	L	H	L	H
H	L	H	L	H	L	H	L	L
H	L	H	L	L	L	L	H	H
H	L	L	H	H	L	L	H	L
H	L	L	H	L	L	L	L	H
H	L	L	L	H	L	L	L	L

M=L $\longrightarrow$ UP   M=H $\longrightarrow$ DOWN

Design Code:

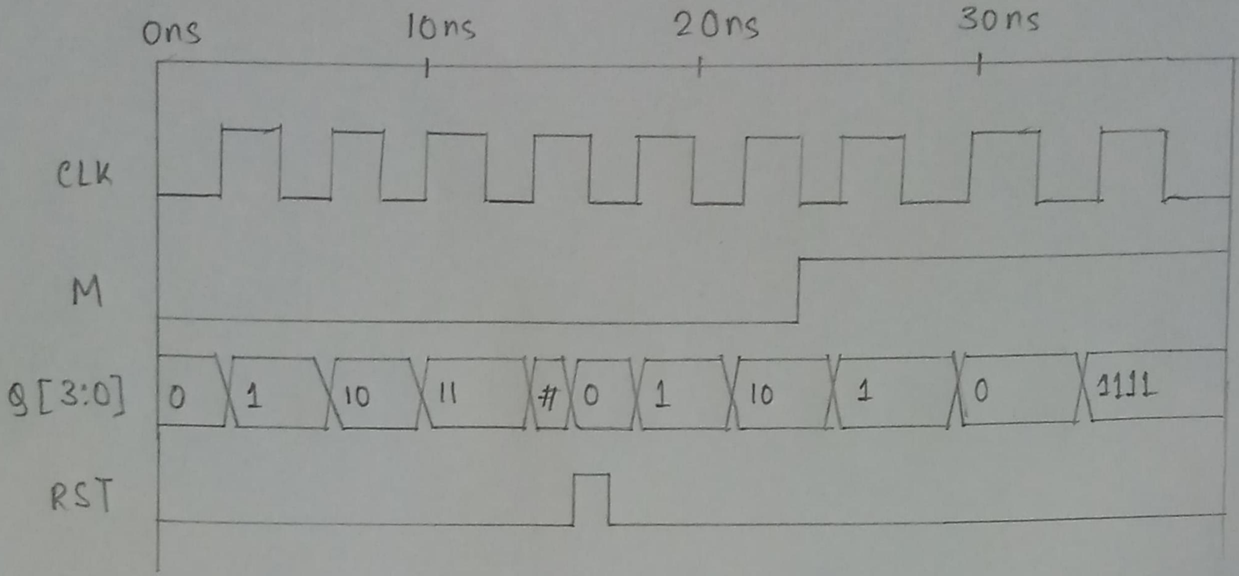
```
module t_ff (input t, input clk, input rst, output q,
                                          output qbar);
   reg q=0;
   assign qbar=~q;
   always @(posedge clk)
      begin
         if (t==1) begin
             q<=~q; end
      end
   always @(posedge rst)
      begin
          q<=0; end
endmodule
```

```
module up-down (input rst , input clk , input m,
                    output [3:0]q , output [3:0]qbar);
  t-ff f1 ( 1, clk , rst , q[0], qbar[0]);
  t-ff f2 (((q[0] & ~m)| (qbar[0] & m)), clk , rst , q[1] ,
                                               qbar[1]);
  t-ff f3 (((q[0] & q[1] & ~m) | (qbar[0] & qbar[1] & m)),
                             clk , rst , q[2] , qbar[2]);
  t-ff f4 ((( q[0] & q[1] & q[2] & ~m) | ( qbar[0] & qbar [1] &
                  qbar[2] & m)) , clk , rst , q[3] , qbar[3]);

endmodule
```

▫ Testbench Code:

```
 module up-down_tb();
   reg RST, CLK, M;
   wire [3:0] Q;
   up-down uut (.rst(RST), .clk(CLK), .m(M), .q(Q));
     initial
       begin
          $ dumpfile ("dump.vcd"); $dumpvars(1);
          CLK = 0;
          RST = 0;
          M = 0; #15;
          RST = 1; #2;
          RST = 0
          M = 0 ; #7;
          M = 1 ; #50;
         $ finish ;
       end
   always  #2 CLK = ~CLK;
 endmodule
```

▫ Timing Diagram:



Ⓑ Random Counter

▫ Theory: Random counter is a sequential circuit made up of four 'T' or 'JK' ff's. In case of random counter, there is no constant sequence of numbers. Here, we can design the sequence of numbers which we want to show. As here, we design a random counter which counts $$\begin{tikzcd} 0 \arrow[r] & 1 \arrow[r] & 3 \arrow[r] & 5 \arrow[r] & 7 \arrow[llll, bend left] \end{tikzcd}$$ sequencing order.

▫ Function Table:

PRESENT STATE				NEXT STATE			
Q_3	Q_2	Q_1	Q_0	Q_3^+	Q_2^+	Q_1^+	Q_0^+
L	L	L	L	L	L	L	H
L	L	L	H	L	L	H	H
L	L	H	H	L	H	L	H
L	H	L	H	L	H	H	H
L	H	H	H	L	L	L	L

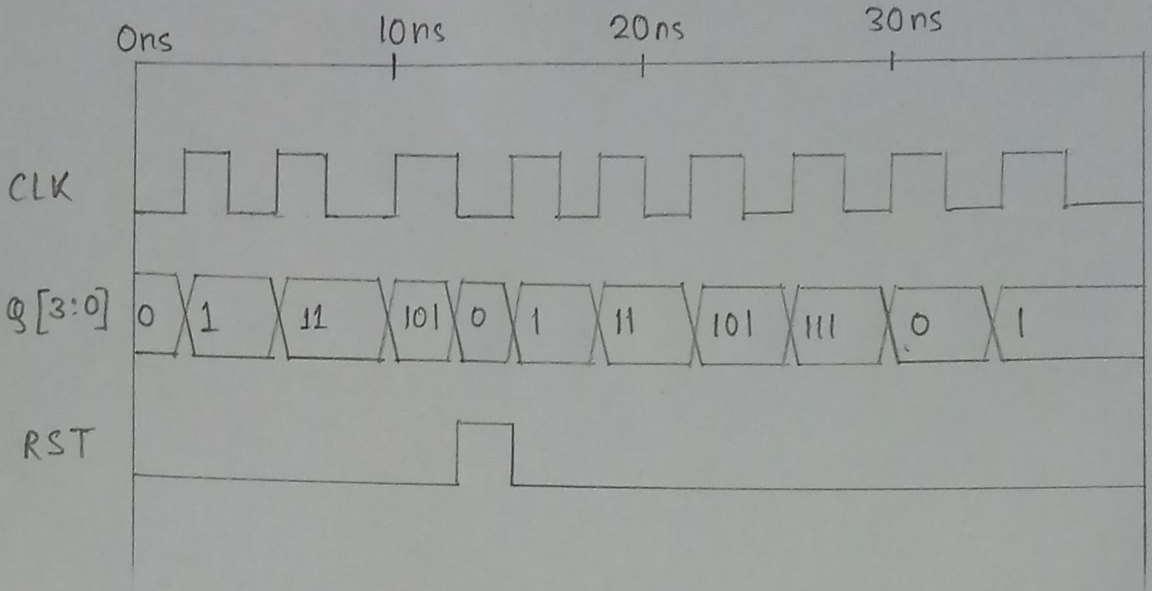
□ Design Code:

```
module random (input rst , input clk , output [3:0] q ,
                                        output [3:0] qbar) ;

    t-ff f1 (((~q[3] & ~q[2] & ~q[1] & ~q[0]) | (~q[3] &
                        q[2] & q[1] & q[0])), clk , rst, q[0], qbar[0]);
    t-ff f2 ((~q[3] & q[0]), clk , rst , q[1], qbar[1]);
    t-ff f3 ((~q[3] & q[1] & q[0]), clk, rst , q[2], qbar[2]);
    t-ff f4 (0, clk , rst , q[3] , qbar [3]) ;

endmodule
```

□ Testbench Code:

```
module random_tb();
    reg RST, CLK ;
    wire [3:0] Q ;
    random uut (.rst(RST), .clk(CLK), q (Q)) ;
      initial begin
          $ dumpfile ("dump.vcd") ; $dumpvars(1) ;
           CLK = 0 ;
           RST = 0  ; #12 ;
           RST = 1  ; # 2 ;
           RST = 0  ; # 50 ;
          $ finish ;
      end
    always #2 CLK = ~CLK ;
endmodule
```

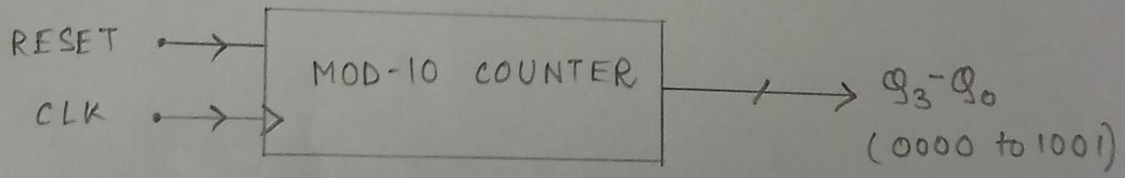
□ Timing Diagram:



© Mod-10 Counter:

□ Theory: MOD counters are cascaded counter circuit which count to a set modulus value before resetting. For Mod 10 counter, 4 T ff's are connected together with a synchronous clock pulse. This counter counts 0000 to 1001. After 1001, it goes again to '0000' and this cycle continues.

□ Logic Diagram:



□ Function Table:

PRESENT STATE				NEXT STATE			
Q_3	Q_2	Q_1	Q_0	Q_3^+	Q_2^+	Q_1^+	Q_0^+
L	L	L	L	L	L	L	H
L	L	L	H	L	L	H	L
L	L	H	L	L	L	H	H
L	L	H	H	L	H	L	L
L	H	L	L	L	H	L	H
L	H	L	H	L	H	H	L
L	H	H	L	L	H	H	H
L	H	H	H	H	L	L	L
H	L	L	L	H	L	L	H
H	L	L	H	L	L	L	L

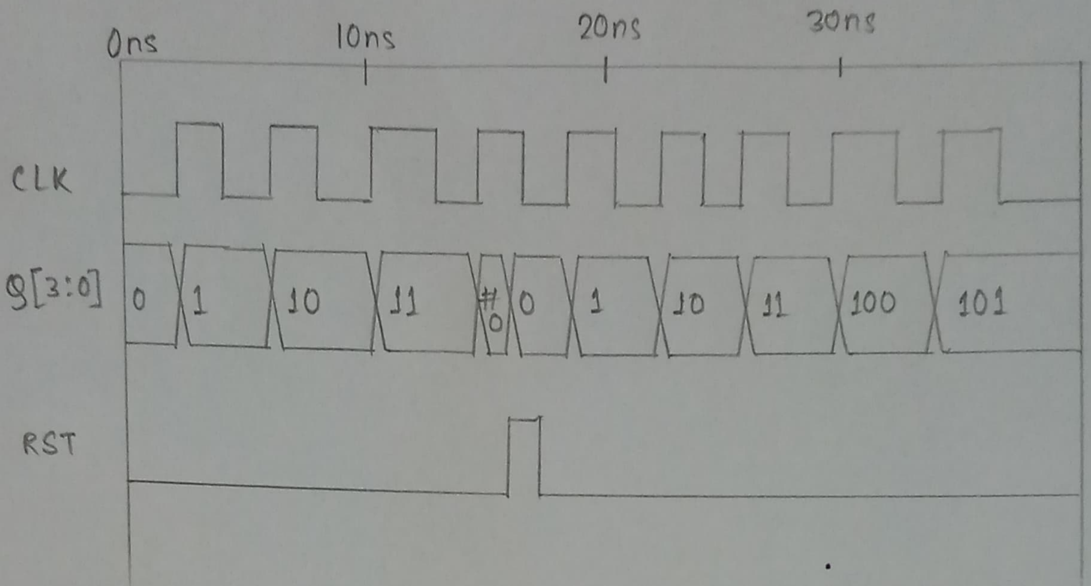
□ Design Code:

```
module mod-10(input rst, input clk, output [3:0]q, output
                                              [3:0]qbar);
  t-ff- f1(1, clk, ((~q[0] & q[1] & ~q[2] & q[3])|rst), q[0],
                                      qbar[0]);
  t-ff f2(q[0], clk, ((~q[0] & q[1] & ~q[2] & q[3]|rst), q[1],
                                      qbar[1]);
  t-ff- f3((q[1] & q[0]), clk, ((~q[0] & q[1] & ~q[2] & q[3]|rst),
                                      q[2], qbar[2]);
  t-ff f4((q[2] & q[1] & q[0]), clk, ((~q[0] & q[1] & ~q[2] &
                                      q[3]|rst), q[3], qbar[3]);
endmodule
```

□ Testbench Code:

```
module mod-10-tb();
    reg RST, CLK;
    wire [3:0] Q;
    mod-10 uut(.rst(RST), .clk(CLK), .q(Q));
      initial begin
            $dumpfile("dump.vcd"); $dumpvars(1);
            CLK=0;
            RST=0; #15;
            RST=1; #2;
            RST=0; #50;
            $finish;
      end
    always #2 CLK=~CLK;
endmodule
```

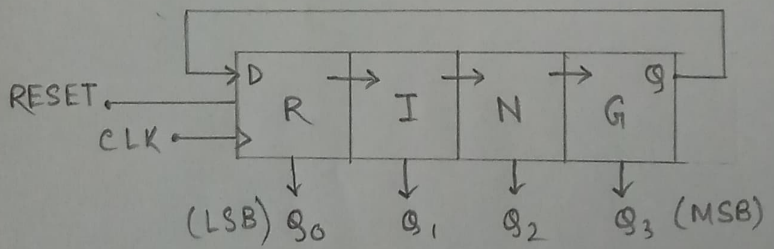
▫ Timing Diagram:



Ⓓ Ring Counter :

▫ Theory: A ring counter is a type of counter composed of flipflops connected into a shift register, with the output of the last flipflop fed to the input of the first, making a ring structure. Here we make a ring counters with outputs 0001→0010→0100→1000→0001.

▫ Logic Diagram:



▫ Function Table:

PRESENT STATE				NEXT STATE			
Q_3	Q_2	Q_1	Q_0	Q_3^+	Q_2^+	Q_1^+	Q_0^+
L	L	L	H	L	L	H	L
L	L	H	L	L	H	L	L
L	H	L	L	H	L	L	L
H	L	L	L	L	L	L	H

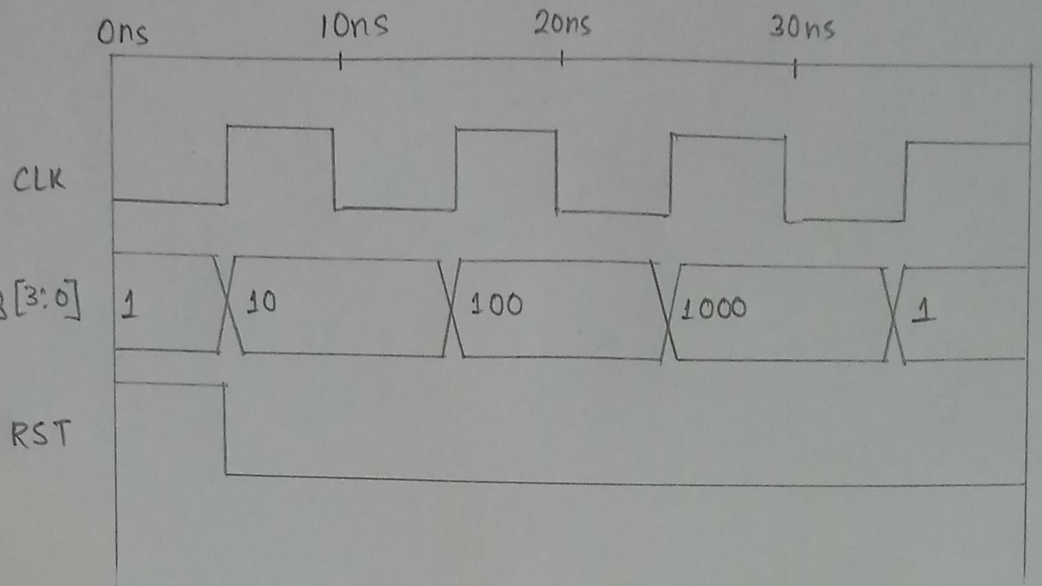
▫ Design Code:

```
module ring-counter( clk, rst, q) ;
    input clk, rst ;
    Output [3:0]q ;
    wire clk, rst ;
    reg [3:0]q = 4'b0001 ;
    always @( posedge clk or posedge rst)
      begin
        if (~rst) begin
          q <= q<<1 ; q[0] <= q[3] ; end
        else
          q <= 4'b0001 ;
      end
endmodule
```

▫ Testbench Code

```
module ring-counter-tb() ;
    reg CLK, RST ;
    wire [3:0] Q ;
    ring-counter uut (.clk(CLK), .rst(RST), .q(Q));
      initial begin
        $dumpfile ("dump.vcd") ;
        $dumpvars (1) ;
        CLK = 0 ;
        RST = 1 ; #5 ;
        RST = 0 ; #50 ;
        RST = 1 ; #5 ;
        RST = 0 ; #20 ;
        $finish ;
      end
    always #5 CLK = ~CLK ;
endmodule
```

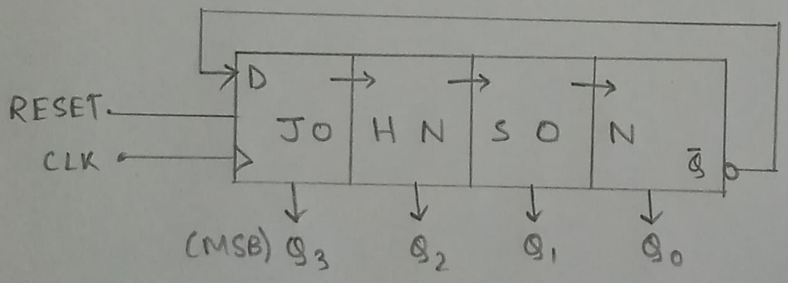
▫ Timing Diagram:



Ⓔ Johnson Counter

▫ Theory: A Johnson Counter is a type of counter composed of ff's connected to a shift register. With the compliment of the output of last ff is fed to the input of the first ff. Johnson counter with 'n' ffs has '2n' no. of states.

▫ Logic Diagram:



▫ Function Table:

PRESENT STATE				NEXT STATE			
Q_3	Q_2	Q_1	Q_0	Q_3^+	Q_2^+	Q_1^+	Q_0^+
L	L	L	L	H	L	L	L
H	L	L	L	H	H	L	L
H	H	L	L	H	H	H	L
H	H	H	L	H	H	H	H

H	H	H	H	L	H	H	H
L	H	H	H	L	L	H	H
L	L	H	H	L	L	L	H
L	L	L	H	L	L	L	L

▫ Design Code:

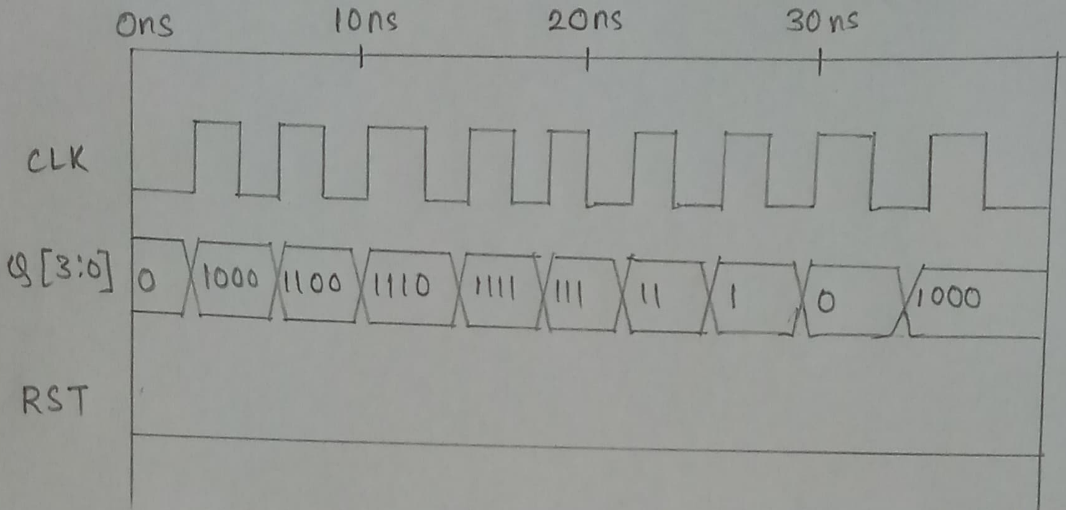
```
module johnson-counter(clk, rst, q);
   input clk, rst;
   output [3:0]q;
   wire clk, rst;
   reg [3:0]q = 4'b0000;
   always @ (posedge clk)
        begin
          if (!rst) begin
             q <= q>> 1;
             q[3] <= ~q[0];

        end
       else
             q <= 4'b 0000; end
 endmodule
```

▫ Testbench Code:

```
module johnson-counter-tb();
   reg CLK, RST;
   wire [3:0] Q;
  johnson-counter uut(.clk(CLK),.rst(RST), .q(Q));
   initial
       begin
          $ dumpfile ("dump.vcd"); $dumpvars(1);
```

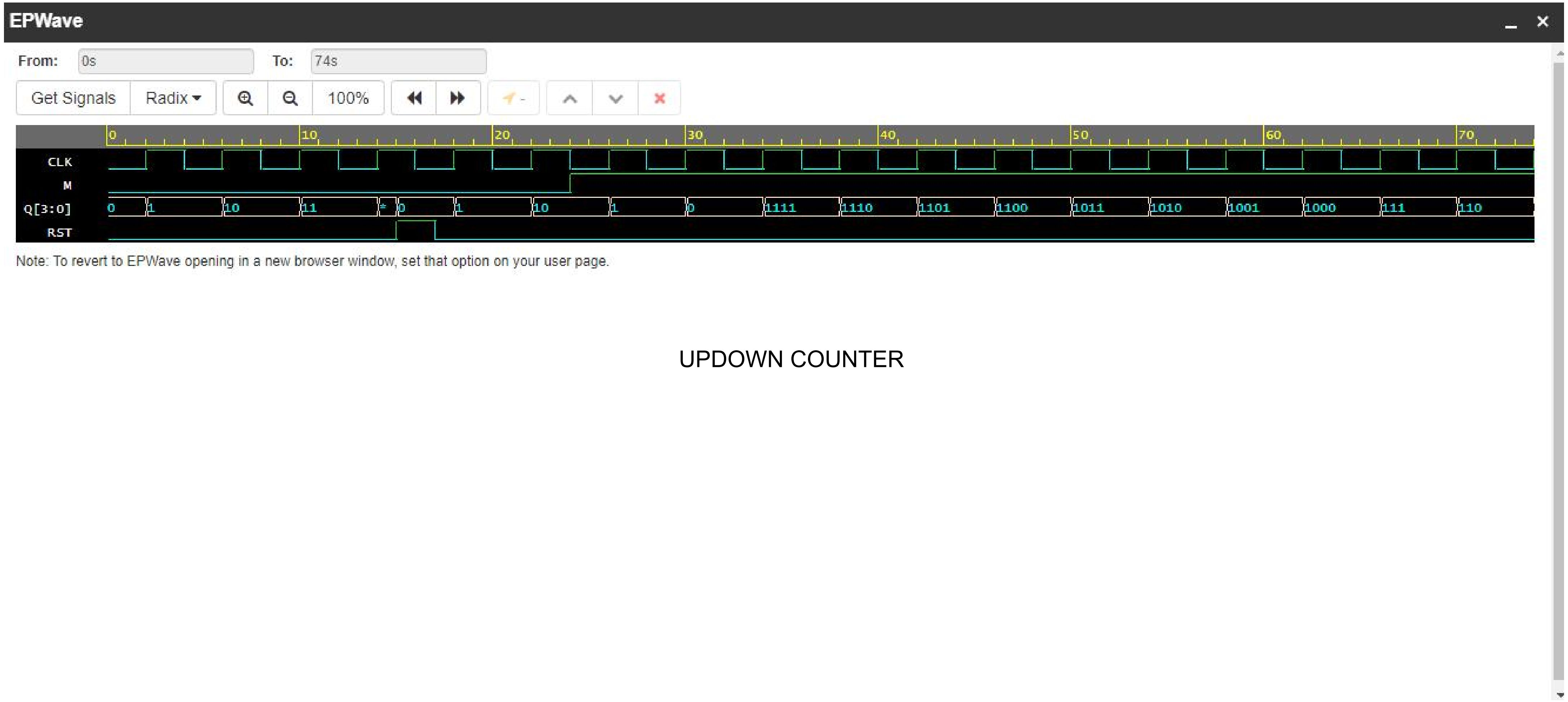
```
        CLK=0 ;
        RST=0; #50;
        $finish;
    end
  always #2 CLK=~CLK;
endmodule
```

▫ Timing Diagram :



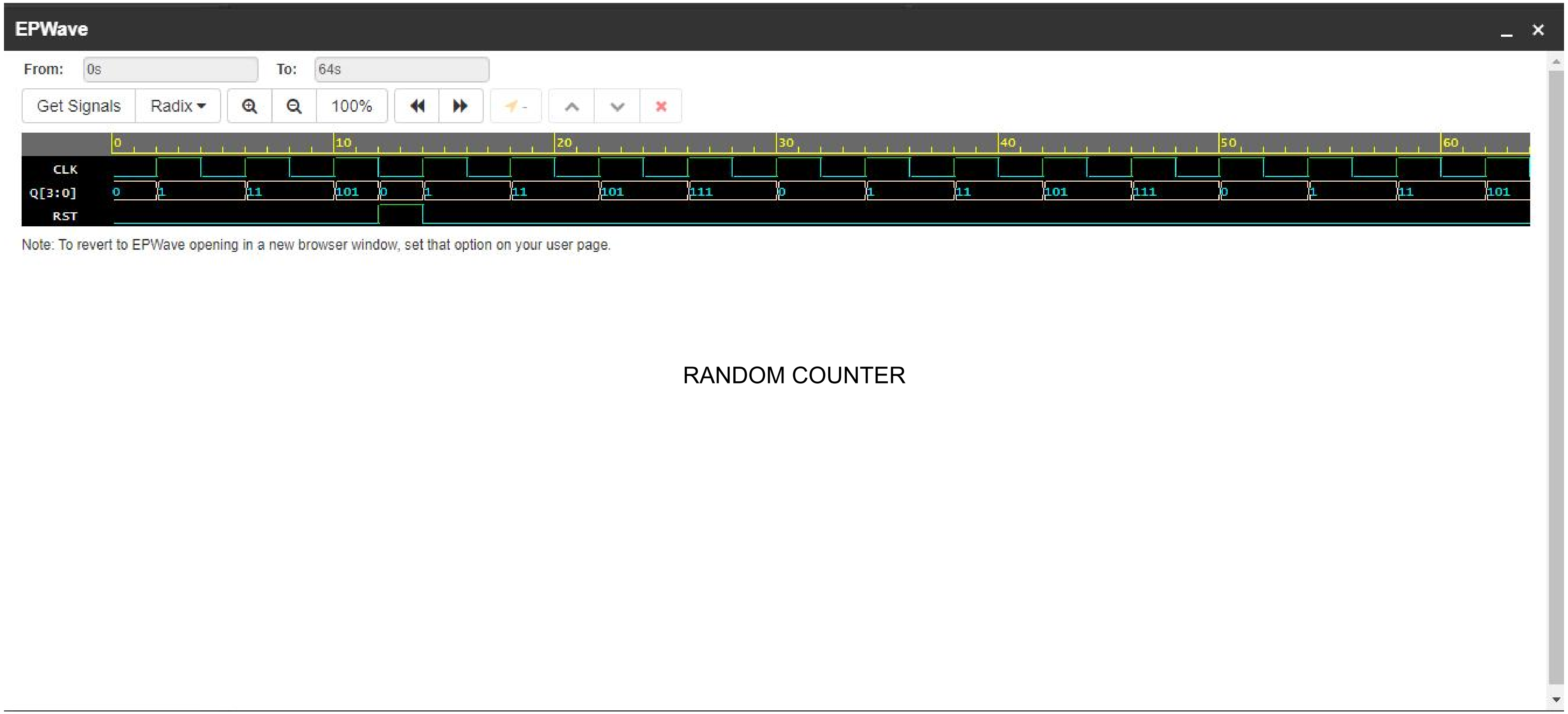
▫ Discussion: In this experiment, we have implemented various types of counters – up-down counter, random counter, ring counter, mod 10 counter and johnson counter. We have also learned various HDL terms.

▫ Justification of CO: In this experiment, we have designed various sequential circuits and used some sequential modules in those circuits, which fulfils the conditions of CO3. So, CO3 is justified.



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

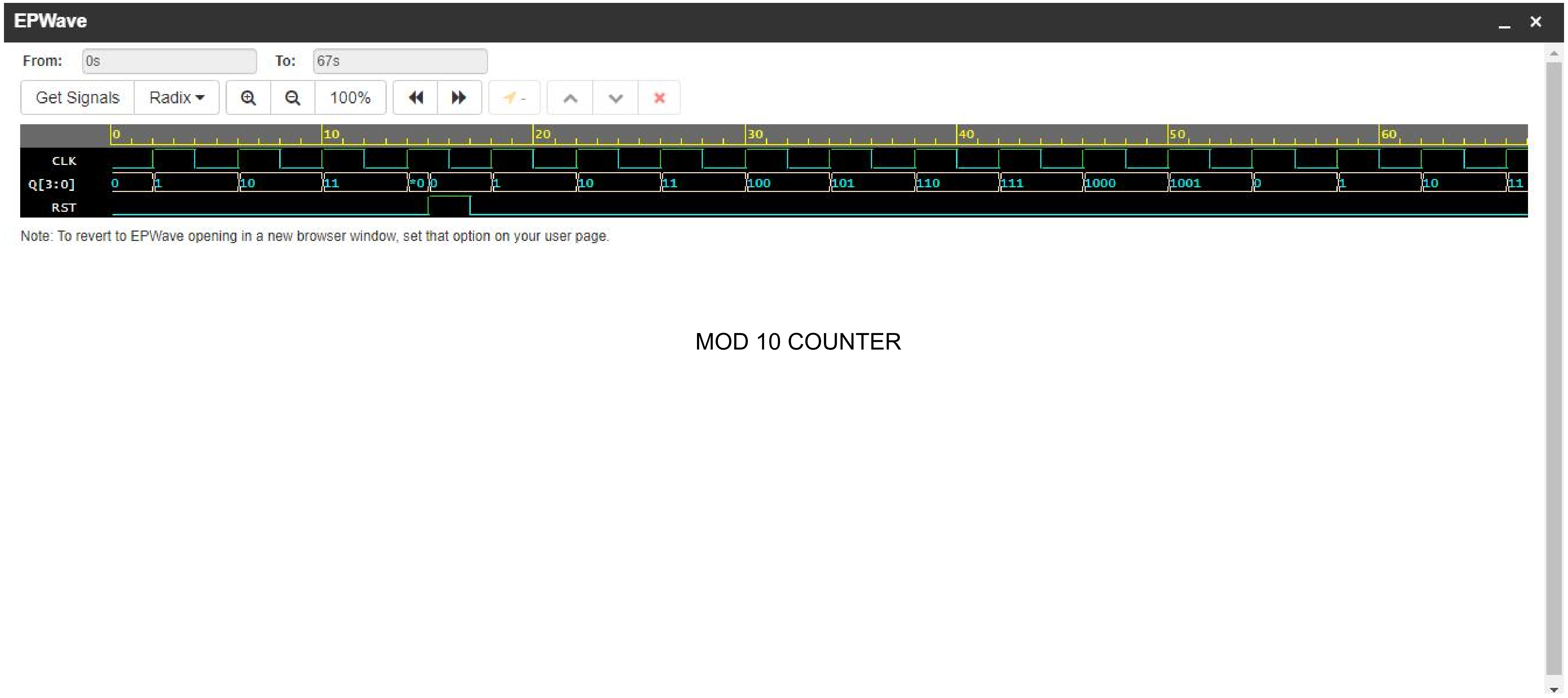
UPDOWN COUNTER



From: 0s
To: 64s

Note: To revert to EPWave opening in a new browser window, set that option on your user page.

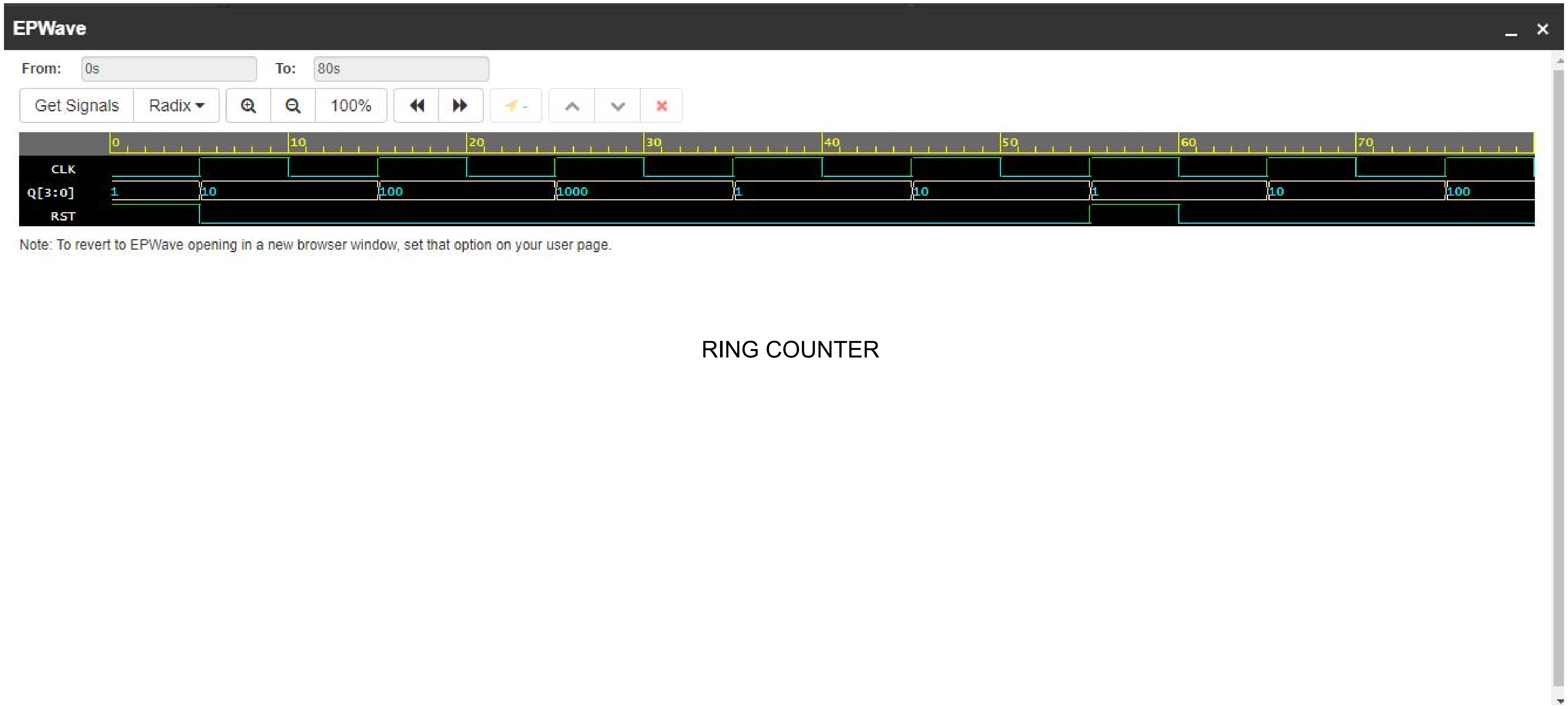
RANDOM COUNTER



From: 0s To: 67s

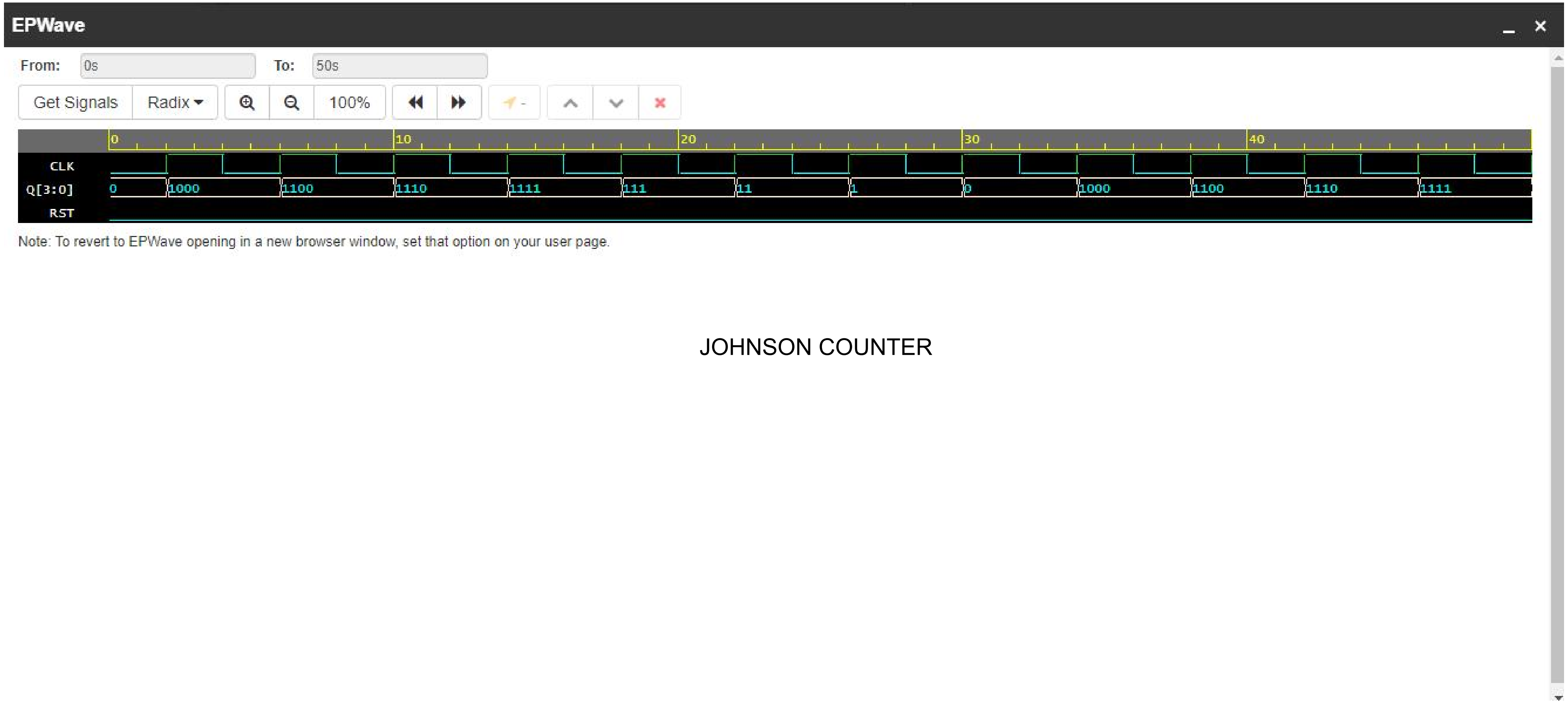
Note: To revert to EPWave opening in a new browser window, set that option on your user page.

MOD 10 COUNTER



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

RING COUNTER



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

JOHNSON COUNTER