**4-bit synchronous up-down counter:**

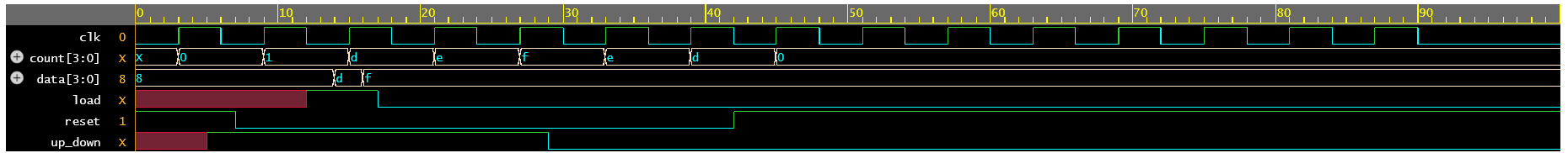
**Verilog Code:**

|  |
| --- |
| // Code your design here  module counter(clk, reset, up\_down, load, data, count);    //define input and output ports  input clk, reset, load, up\_down;  input [3:0] data;  output reg [3:0] count;    //always block will be executed at each and every positive edge of the clock  always@(posedge clk)  begin  if(reset) count <= 0; //Set Counter to Zero  else if(load) count <= data; //load the counter with data value  else if(up\_down) count <= count + 1; //count up  else count <= count - 1; //count down  end    endmodule :counter |

**Testbench:**

|  |
| --- |
| // Code your testbench here  // or browse Examples  module counter\_tb;  reg clk,reset,load,up\_down;  reg [3:0] data;  wire [3:0] count;    // instance counter design  counter ct\_1(.up\_down(up\_down),.\*);    //clock generator  initial  begin  clk = 1'b0;  repeat(30) #3  clk= ~clk;  end    //insert all the input signal  initial  begin  reset=1'b1; #7  reset=1'b0; #35  reset=1'b1;  end    initial  begin  #12  load=1'b1; #5  load=1'b0;  end    initial  begin  #5  up\_down=1'b1; #24  up\_down=1'b0;  end    initial  begin  data=4'b1000; #14  data=4'b1101; #2  data=4'b1111;  end    //monitor all the input and output ports at times when any inputs changes its state  initial  begin  $monitor("time=%0d, reset=%b, load=%b, up\_down=%b, data=%d, count=%d",  $time, reset, load, up\_down, data, count);  end    initial  begin  $dumpfile("dump.vcd");  $dumpvars();  #100  $finish;  end    endmodule :counter\_tb |

**Waveform:**

****

**Update-1: 4-bit asynchronous up-down counter**

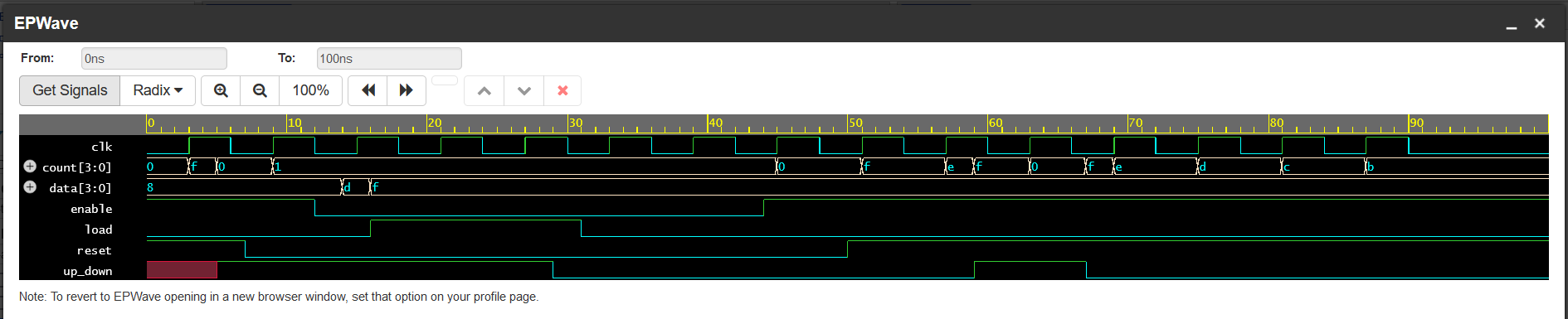
**Verilog:**

|  |
| --- |
| // Code your design here  module counter(clk, reset, up\_down, load, data, count, enable);    //define input and output ports  input clk, reset, load, up\_down, enable;  input [3:0] data;  output reg [3:0] count;    always@(posedge reset)  if(enable)  begin  count<=0;  end  always@(posedge load)  if(enable)  begin  count<=data;  end    always@(posedge up\_down or negedge up\_down)  if(enable)  begin  if(up\_down) count<=count+1;  else count<=count-1;  end    //always block will be executed at each and every positive edge of the clock  always@(posedge clk)  if(enable)  begin  //if(reset) count <= 0; //Set Counter to Zero  //else if(load) count <= data; //load the counter with data value  if(up\_down) count <= count + 1; //count up  else count <= count - 1; //count down  end    endmodule |

**Testbench:**

|  |
| --- |
| // Code your testbench here  // or browse Examples  // Code your testbench here  // or browse Examples  module counter\_tb;  reg clk,reset,load,up\_down, enable;  reg [3:0] data;  wire [3:0] count;    // instance counter design  counter ct\_1(.clk(clk), .reset(reset), .load(load), .up\_down(up\_down), .enable(enable), .data(data), .count(count));    //clock generator  initial  begin  clk = 1'b0;  repeat(30) #3  clk= ~clk;  end    //insert all the input signal  initial  begin  reset=1'b1; #7  reset=1'b0; #43  reset=1'b1;  end    initial  begin  enable=1'b1; #12  enable=1'b0; #32  enable=1'b1;  end    initial  begin  load = 1'b0; #16  load=1'b1; #15  load=1'b0;  end    initial  begin  #5  up\_down=1'b1; #24  up\_down=1'b0; #30  up\_down=1'b1; #8  up\_down=1'b0;  end    initial  begin  data=4'b1000; #14  data=4'b1101; #2  data=4'b1111;  end    //monitor all the input and output ports at times when any inputs changes its state  initial  begin  $monitor("time=%0d, reset=%b, load=%b, up\_down=%b, data=%d, count=%d",  $time, reset, load, up\_down, data, count);  end    initial  begin  $dumpfile("dump.vcd");  $dumpvars();  #100  $finish;  end    endmodule :counter\_tb |

**Waveform:**

****

**Update-2: 4-bit asynchronous up-down counter**

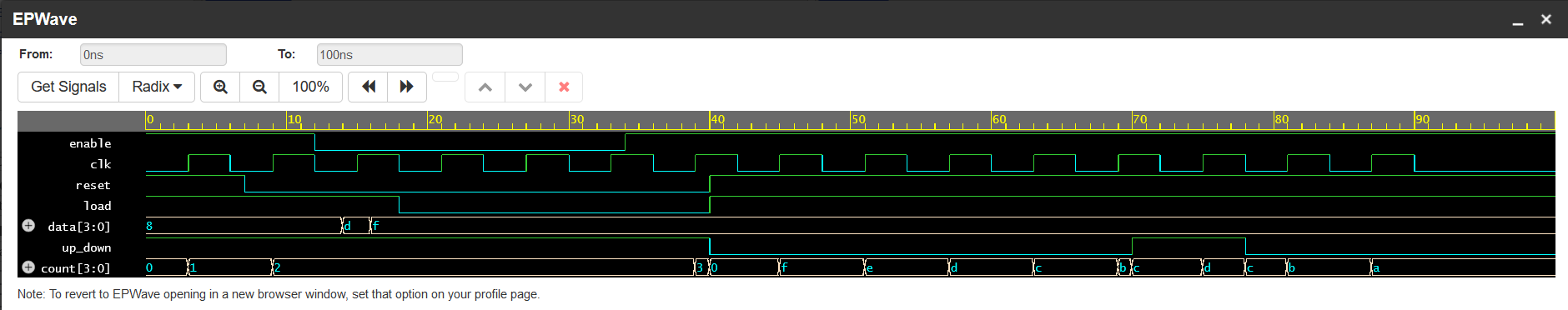
**Verilog:**

|  |
| --- |
| // Code your design here  module counter(clk, reset, up\_down, load, data, count, enable);    //define input and output ports  input clk, reset, load, up\_down, enable;  input [3:0] data;  output reg [3:0] count;  integer r=0;  integer l=0;  integer ud=up\_down;      //always block will be executed at each and every positive edge of the clock  always@(posedge clk or posedge reset or posedge load or posedge up\_down or negedge up\_down)  begin  if(enable)  begin  if(reset==1 & r==0)  begin  count<=0;  r=1;  end  else if(load==1 & l==0)  begin  count<=data;  l=1;  end  else if(up\_down==1 & ud==0)  begin  count<=count+1;  ud=1;  end  else if(up\_down==0 & ud==1)  begin  count<=count-1;  ud=0;  end  else if(up\_down==1)  begin  count<=count+1;  ud=1;  end    else if(up\_down==0)  begin  count<=count-1;  ud=0;  end    if(reset==0) r=0;  else r=1;  if(load==0) l=0;  else l=1;  end    else count<=count;    end  endmodule |

**Testbench:**

|  |
| --- |
| // Code your testbench here  // or browse Examples  // Code your testbench here  // or browse Examples  module counter\_tb;  reg clk,reset,load,up\_down, enable;  reg [3:0] data;  wire [3:0] count;    // instance counter design  counter ct\_1(.clk(clk), .reset(reset), .load(load), .up\_down(up\_down), .enable(enable), .data(data), .count(count));    //clock generator  initial  begin  clk = 1'b0;  repeat(30) #3  clk= ~clk;  end    //insert all the input signal  initial  begin  reset=1'b1; #7  reset=1'b0; #33  reset=1'b1;  end    initial  begin  enable=1'b1; #12  enable=1'b0; #22  enable=1'b1;  end    initial  begin  load = 1'b1; #18  load=1'b0; #22  load=1'b1; #8  load=1'b0; #12  load=1'b1;  end    initial  begin  up\_down=1'b1; #40  up\_down=1'b0; #20  up\_down=1'b1; #8  up\_down=1'b0; #10  up\_down=1'b1;  end    initial  begin  data=4'b1000; #14  data=4'b1101; #2  data=4'b1111;  end    //monitor all the input and output ports at times when any inputs changes its state  initial  begin  $monitor("time=%0d, reset=%b, load=%b, up\_down=%b, data=%d, count=%d", $time, reset, load, up\_down, data, count);  end    initial  begin  $dumpfile("dump.vcd");  $dumpvars();  #100  $finish;  end    endmodule |

**Waveform:**

****

**Update-3: 12-bit asynchronous up-down counter**

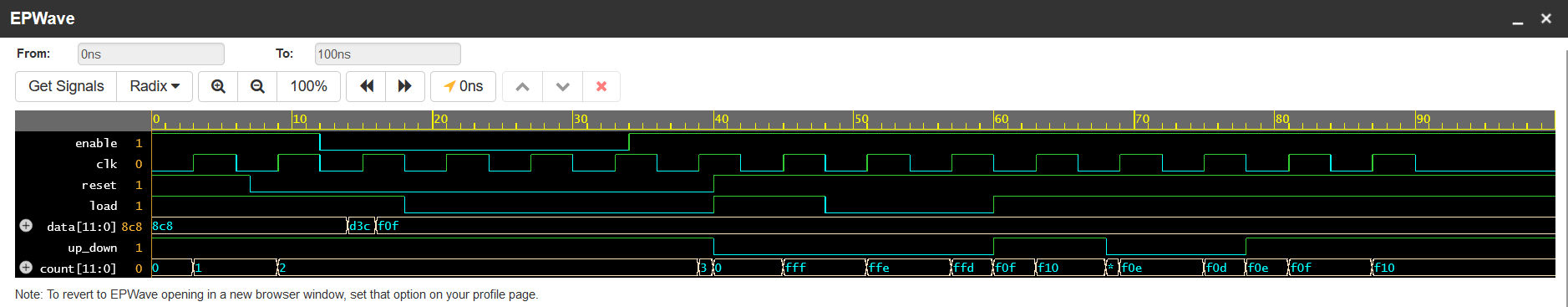
**Verilog:**

|  |
| --- |
| // Code your design here  module counter(clk, reset, up\_down, load, data, count, enable);    //define input and output ports  input clk, reset, load, up\_down, enable;  input [11:0] data;  output reg [11:0] count;  logic r=0;  logic l=0;  logic ud=up\_down;      //always block will be executed at each and every positive edge of the clock  always@(posedge clk or posedge reset or posedge load or posedge up\_down or negedge up\_down)  begin  if(enable)  begin  if(reset==1 & r==0)  begin  count<=0;  r=1;  end  else if(load==1 & l==0)  begin  count<=data;  l=1;  end  else if(up\_down==1 & ud==0)  begin  count<=count+1;  ud=1;  end  else if(up\_down==0 & ud==1)  begin  count<=count-1;  ud=0;  end  else if(up\_down==1)  begin  count<=count+1;  ud=1;  end    else if(up\_down==0)  begin  count<=count-1;  ud=0;  end    if(reset==0) r=0;  else r=1;  if(load==0) l=0;  else l=1;  end    else count<=count;    end  endmodule |

**Testbench:**

|  |
| --- |
| // Code your testbench here  // or browse Examples  // Code your testbench here  // or browse Examples  module counter\_tb;  reg clk,reset,load,up\_down, enable;  reg [11:0] data;  wire [11:0] count;    // instance counter design  counter ct\_1(.clk(clk), .reset(reset), .load(load), .up\_down(up\_down), .enable(enable), .data(data), .count(count));    //clock generator  initial  begin  clk = 1'b0;  repeat(30) #3  clk= ~clk;  end    //insert all the input signal  initial  begin  reset=1'b1; #7  reset=1'b0; #33  reset=1'b1;  end    initial  begin  enable=1'b1; #12  enable=1'b0; #22  enable=1'b1;  end    initial  begin  load=1'b1; #18  load=1'b0; #22  load=1'b1; #8  load=1'b0; #12  load=1'b1;  end    initial  begin  up\_down=1'b1; #40  up\_down=1'b0; #20  up\_down=1'b1; #8  up\_down=1'b0; #10  up\_down=1'b1;  end    initial  begin  data=12'b100011001000; #14  data=12'b110100111100; #2  data=12'b111100001111;  end    //monitor all the input and output ports at times when any inputs changes its state  initial  begin  $monitor("time=%0d, reset=%b, load=%b, up\_down=%b, data=%d, count=%d", $time, reset, load, up\_down, data, count);  end    initial  begin  $dumpfile("dump.vcd");  $dumpvars();  #100  $finish;  end    endmodule |

**Waveform:**

****