

## **Up Down Counter Module using Verilog**

`up_down_counter.v`

```
//up_down_counter module using Verilog
```

```
module up_down_counter(input CLK, ENABLE, UPDN, RST, output wire[3:0] VALUE);
reg [3:0] counter; // reg

always @ (posedge CLK or posedge RST)
begin
    if (RST == 1'b1)
    begin
        counter <= 4'b0000;
    end // if RST occur it will reset counter
    else
    if(ENABLE == 1'b1 && UPDN == 1'b1) // If ENABLE = 1 and UPDN = 1 condition.
    begin
        if (counter == 4'hF)
        begin
            counter <= 4'hF;
        end
        else
        counter <= counter + 1'b1; // The counter counts up incrementing VALUE on
        every clock cycle.
    end
    else if (ENABLE == 1'b1 && UPDN == 1'b0) // If ENABLE = 1 and UPDN = 0 condition.
    begin
        if (counter == 4'h0)
        begin
            counter <= 4'h0;
        end
        else
        counter <= counter - 1'b1; // The counter counts down decrementing VALUE on every clock
        cycle.
    end
    else if (ENABLE == 1'b0 && (UPDN == 1'b0 || 1'b1))
    begin
        counter <= counter;
    end
    else if(ENABLE == 1'b0)
        counter <= counter; // Else counter remains at the present value
    else if (counter == 15)
        counter <= 4'hF;
    else if (counter == 0)
        counter <= 4'h0;
    end
    assign VALUE = counter;
endmodule
```

**Binary to BCD conversion Module Verilog code**  
bin2bcd.v

```
//-----  
// Bin to BCD conversion using double dabble algorithm in verilog  
//-----  
module bin2bcd(input wire [3:0] VALUE, output reg [3:0] TENS, output reg [3:0] ONES);  
  
integer i;  
  
always @(VALUE)  
begin  
    //define TENS and ONES to 0  
    TENS = 4'd0;  
    ONES = 4'd0;  
  
    for (i=3; i>=0; i=i-1)  
    begin  
        // add 3 to columns if >= 5  
        if (TENS >= 5)  
            TENS = TENS + 3;  
        if (ONES >= 5)  
            ONES = ONES + 3;  
        // Shift bits left one  
        TENS = TENS << 1;  
        TENS[0] = ONES[3];  
        ONES = ONES << 1;  
        ONES[0] = VALUE[i];  
    end  
end  
endmodule // bin2bcd
```

## **Up Down Counter Test Bench Verilog code**

`updn_bcd_fixture.v`

```
`include "bin2bcd.v"
`include "up_down_counter.v" //

module updn_bcd_fixture();
    reg clk, en, updn, rst;
    wire [3:0]value;
    wire [3:0]tens;
    wire [3:0]ones;
//instantiation
up_down_counter U1(.CLK(clk), .ENABLE(en), .UPDN(updn), .RST(rst), .VALUE(value));
bin2bcd U2(.VALUE(value), .TENS(tens), .ONES(ones));

integer i;

    reg [0:10] data [0:32];
    reg [0:10] data_bits_reg;
    reg data_bit_rst;
    reg data_bit_en;
    reg data_bit_updn;
    reg [0:3] data_bits_tens;
    reg [0:3] data_bits_ones;
    reg [0:2] Inputsignal;

// read file data
    initial
    begin
        $readmemb("counter_bcd_datafile.txt",data);
    end

//-----
//stimulus signal
//-----
    initial begin
        clk = 0;
        en = 1;
        updn = 1;
        rst = 1;
    end

// stimulate the clock signal
    initial
    begin
        forever #10 clk =~clk;
    end

// finish the simulation at time
    initial
    begin
        #1000 $finish;
    end

    initial begin
        for(i=0;i<32;i=i+1)begin
            data_bits_reg = data[i];
            #10; //20
            data_bits_ones = data_bits_reg[7:10];
            data_bits_tens = data_bits_reg[3:6];
            data_bit_updn = data_bits_reg[2];
            data_bit_en = data_bits_reg[1];
            data_bit_rst = data_bits_reg[0];
            Inputsignal = data_bits_reg[0:2];
        end
    end
endmodule
```

```

rst = data_bit_rst;
en = data_bit_en;
updn = data_bit_updn;
#5; //10

```

```

case(Inputsignal)

```

```

  3'b000:begin

```

```

    #5;

```

```

    if(tens == data_bits_tens && ones == data_bits_ones)begin

```

```

        $display($time, " Output is correct when reset = 0, Enable = 0, Updn = 0");

```

```

        $display($time, " %b %b %b %b %b %b\n",Inputsignal, rst, en, updn, tens, ones,
        data_bits_reg[3:6], data_bits_reg[7:10]);

```

```

    end

```

```

    else begin

```

```

        $display($time, " Output is failed when reset = 0, Enable = 0, Updn = 0");

```

```

        $display($time, " %b %b %b %b %b %b\n",Inputsignal, rst, en, updn, tens, ones,
        data_bits_reg[3:6], data_bits_reg[7:10]);

```

```

    end

```

```

end

```

```

  3'b001:begin

```

```

    #5;

```

```

    if(tens == data_bits_tens && ones == data_bits_ones)begin

```

```

        $display($time, " Output is correct when reset = 0, Enable = 0, Updn = 1");

```

```

        $display($time, " %b %b %b %b %b %b\n",Inputsignal, rst, en, updn, tens, ones,
        data_bits_reg[3:6], data_bits_reg[7:10]);

```

```

    end

```

```

    else begin

```

```

        $display($time, " Output is failed when reset = 0, Enable = 0, Updn = 1");

```

```

        $display($time, " %b %b %b %b %b %b\n",Inputsignal, rst, en, updn, tens, ones,
        data_bits_reg[3:6], data_bits_reg[7:10]);

```

```

    end

```

```

end

```

```

  3'b010:begin

```

```

    #5;

```

```

    if(tens == data_bits_tens && ones == data_bits_ones)begin

```

```

        $display($time, " Output is correct when reset = 0, Enable = 1, Updn = 0");

```

```

        $display($time, " %b %b %b %b %b %b\n",Inputsignal, rst, en, updn, tens, ones,
        data_bits_reg[3:6], data_bits_reg[7:10]);

```

```

    end

```

```

    else begin

```

```

        $display($time, " Output is failed when reset = 0, Enable = 1, Updn = 0");

```

```

        $display($time, " %b %b %b %b %b %b\n",Inputsignal, rst, en, updn, tens, ones,
        data_bits_reg[3:6], data_bits_reg[7:10]);

```

```

    end

```

```

end

```

```

  3'b011:begin

```

```

    #5;

```

```

    if(tens == data_bits_tens && ones == data_bits_ones)begin

```

```

        $display($time, " Output is correct when reset = 0, Enable = 1, Updn = 1");

```

```

        $display($time, " %b %b %b %b %b %b\n",Inputsignal, rst, en, updn, tens, ones,
        data_bits_reg[3:6], data_bits_reg[7:10]);

```

```

    end

```

```

    else begin

```

```

        $display($time, " Output is failed when reset = 0, Enable = 1, Updn = 1");

```

```

        $display($time, " %b %b %b %b %b %b\n",Inputsignal, rst, en, updn, tens, ones,
        data_bits_reg[3:6], data_bits_reg[7:10]);

```

```

    end

```

```

end

```

```

  3'b111:begin

```

```

    #5;

```

```

    if(tens == data_bits_tens && ones == data_bits_ones)begin

```

```
        $display($time, " Output is correct when reset = 1, Enable = 1, Updn = 1");
        $display($time, " %b %b %b %b %b %b\n",Inputsignal, rst, en, updn, tens, ones,
        data_bits_reg[3:6], data_bits_reg[7:10]);
    end
    else begin
        $display($time, " Output is failed when reset = 1, Enable = 1, Updn = 1");
        $display($time, " %b %b %b %b %b %b\n",Inputsignal, rst, en, updn, tens, ones,
        data_bits_reg[3:6], data_bits_reg[7:10]);
    end
end

default:
    $display(" Nothing");
endcase
end
end
endmodule // updn_bcd_fixture
```



**Perl script**  
counter\_bcd.pl

```

#!/usr/bin/perl
#counter_bcd_datafile

#-----
# Variables
#-----
$rst = 0;
$en = 1;
$updn = 1;
$counter = 0;
$tens = 0000;
$ones = 0000;
#-----
# File handler
#-----
$file = 'counter_bcd_datafile.txt'; # File path
open(counter_bcd_datafile, ">$file"); # file handler "open file to write/replace"

#-----
# Test vectors
#-----
for($i=0;$i<32;$i=$i+1)
{
    if($i == 0) # initialize rst, en, updn to 0
    {
        $rst = 0;
        $en = 0;
        $updn = 0;
    }
    else{
        if($i >= 15 && $i <= 17) #reset stimulus condition
        {
            $rst = 1;
        }
        else
        {
            $rst = 0;
        }

        if($i >= 18 && $i <= 19) # enable stimulus condition
        {
            $en = 0;
        }
        else{
            $en = 1;
        }

        if($i <= 25) # updown stimulus condition
        {
            $updn = 1;
        }
        else
        {
            $updn = 0;
        }
    }
}
#-----
# Counter part
#-----
if(!$rst)
{
    if($en == 1)
    {

```

```
    if ($updn == 1)
    {
        $counter = $counter + 1;
    }
    else
    {
        $counter = $counter - 1;
    }
}
else
{
    $counter = $counter;
}
}
else
{
    $counter = 0;
}

#-----
# BCD conversion
#-----
$ones = $counter % 10; # is the remainder
$tens = ($counter - $ones)/10;

#-----
# Write data to the file
#-----
printf counter_bcd_datafile "%01b_%01b_%01b_%04b_%04b\n", $rst, $en, $updn, $tens, $ones;
}
close(counter_bcd_datafile) # flie handler "close file"

#end
```

## **Perl Script Result**

counter\_bcd\_datafile.txt

```
0_0_0_0000_0000
0_1_1_0000_0001
0_1_1_0000_0010
0_1_1_0000_0011
0_1_1_0000_0100
0_1_1_0000_0101
0_1_1_0000_0110
0_1_1_0000_0111
0_1_1_0000_1000
0_1_1_0000_1001
0_1_1_0001_0000
0_1_1_0001_0001
0_1_1_0001_0010
0_1_1_0001_0011
0_1_1_0001_0100
0_1_1_0001_0101
1_1_1_0000_0000
1_1_1_0000_0000
1_1_1_0000_0000
0_0_1_0000_0000
0_0_1_0000_0000
0_1_1_0000_0001
0_1_1_0000_0010
0_1_1_0000_0011
0_1_1_0000_0100
0_1_1_0000_0101
0_1_1_0000_0110
0_1_0_0000_0101
0_1_0_0000_0100
0_1_0_0000_0011
0_1_0_0000_0010
0_1_0_0000_0001
0_1_0_0000_0000
```

## **Test Result**

simulation\_result.txt

Chronologic VCS simulator copyright 1991-2014

Contains Synopsys proprietary information.

Compiler version I-2014.03-2; Runtime version I-2014.03-2; Feb 26 23:34 2019

20 Output is correct when reset = 0, Enable = 0, Updn = 0  
20 000 0 0 0 0000 0000

0 0

40 Output is correct when reset = 0, Enable = 1, Updn = 1  
40 011 0 1 1 0000 0001

0 1

60 Output is correct when reset = 0, Enable = 1, Updn = 1  
60 011 0 1 1 0000 0010

0 2

80 Output is correct when reset = 0, Enable = 1, Updn = 1  
80 011 0 1 1 0000 0011

0 3

100 Output is correct when reset = 0, Enable = 1, Updn = 1  
100 011 0 1 1 0000 0100

0 4

120 Output is correct when reset = 0, Enable = 1, Updn = 1  
120 011 0 1 1 0000 0101

0 5

140 Output is correct when reset = 0, Enable = 1, Updn = 1  
140 011 0 1 1 0000 0110

0 6

160 Output is correct when reset = 0, Enable = 1, Updn = 1  
160 011 0 1 1 0000 0111

0 7

180 Output is correct when reset = 0, Enable = 1, Updn = 1  
180 011 0 1 1 0000 1000

0 8

200 Output is correct when reset = 0, Enable = 1, Updn = 1  
200 011 0 1 1 0000 1001

0 9

220 Output is correct when reset = 0, Enable = 1, Updn = 1  
220 011 0 1 1 0001 0000

1 0

240 Output is correct when reset = 0, Enable = 1, Updn = 1  
240 011 0 1 1 0001 0001

1 1

260 Output is correct when reset = 0, Enable = 1, Updn = 1  
260 011 0 1 1 0001 0010

1 2

280 Output is correct when reset = 0, Enable = 1, Updn = 1  
280 011 0 1 1 0001 0011

1 3

300 Output is correct when reset = 0, Enable = 1, Updn = 1  
300 011 0 1 1 0001 0100

1 4

320 Output is correct when reset = 0, Enable = 1, Updn = 1  
320 011 0 1 1 0001 0101

1 5

340 Output is correct when reset = 1, Enable = 1, Updn = 1  
340 111 1 1 1 0000 0000

0 0

360 Output is correct when reset = 1, Enable = 1, Updn = 1  
360 111 1 1 1 0000 0000

0 0

380 Output is correct when reset = 1, Enable = 1, Updn = 1  
380 111 1 1 1 0000 0000

0 0

400 Output is correct when reset = 0, Enable = 0, Updn = 1  
400 001 0 0 1 0000 0000

0 0

420 Output is correct when reset = 0, Enable = 0, Updn = 1

```

0 0      420 001 0 0 1 0000 0000
         440 Output is correct when reset = 0, Enable = 1, Updn = 1
         440 011 0 1 1 0000 0001
0 1      460 Output is correct when reset = 0, Enable = 1, Updn = 1
         460 011 0 1 1 0000 0010
0 2      480 Output is correct when reset = 0, Enable = 1, Updn = 1
         480 011 0 1 1 0000 0011
0 3      500 Output is correct when reset = 0, Enable = 1, Updn = 1
         500 011 0 1 1 0000 0100
0 4      520 Output is correct when reset = 0, Enable = 1, Updn = 1
         520 011 0 1 1 0000 0101
0 5      540 Output is correct when reset = 0, Enable = 1, Updn = 1
         540 011 0 1 1 0000 0110
0 6      560 Output is correct when reset = 0, Enable = 1, Updn = 0
         560 010 0 1 0 0000 0101
0 5      580 Output is correct when reset = 0, Enable = 1, Updn = 0
         580 010 0 1 0 0000 0100
0 4      600 Output is correct when reset = 0, Enable = 1, Updn = 0
         600 010 0 1 0 0000 0011
0 3      620 Output is correct when reset = 0, Enable = 1, Updn = 0
         620 010 0 1 0 0000 0010
0 2      640 Output is correct when reset = 0, Enable = 1, Updn = 0
         640 010 0 1 0 0000 0001
0 1

```

\$finish called from file "updn\_bcd\_fixture.v", line 51.

\$finish at simulation time 1000

# V C S   S i m u l a t i o n   R e p o r t

Time: 1000

CPU Time: 0.190 seconds; Data structure size: 0.0Mb

Tue Feb 26 23:34:58 2019