P1: UPDOWN Counter

0-9 up/down counter with 7-segment LED decoder

Table of Contents

1. P1_UPDOWN.v

```
module UPDOWN(RESET, CLK, DEC, COUNT);

input RESET, CLK, DEC;

output reg [3:0] COUNT;


parameter SEC1_MAX = 125000000; // 125MHz


reg [26:0] tmp_count;

wire ENABLE;


always @(posedge CLK or posedge RESET) begin

if (RESET)

tmp_count <= 27'h000000;

else if (ENABLE)

tmp_count <= 27'h000000;

else

tmp_count <= tmp_count + 1;

end


assign ENABLE = (tmp_count == (SEC1_MAX - 1)) ? 1'b1 : 1'b0;


always @(posedge CLK or posedge RESET) begin

if (RESET)

COUNT <= 4'h0;

else if (ENABLE) begin

if (DEC == 1'b0) begin

if (COUNT == 4'h9)

COUNT <= 4'h0;

else

COUNT <= COUNT + 1;

end else begin

if (COUNT == 4'h0)

COUNT <= 4'h9;

else

COUNT <= COUNT - 1;

end

end

end

endmodule
```

2. P1_DECODER7 .v

File not found: /mnt/project/P1_DECODER7 .v

3. P1_UPDOWN_7SEG.v

```
module UPDOWN_7SEG(RESET, CLK, DEC, LED, SA,/* COUNT*/);

input RESET, CLK, DEC;

output [7:0] LED;

output [3:0] SA;

/*output [3:0] COUNT;*/


wire [3:0] COUNT;

parameter SEC1_MAX = 125000000;


UPDOWN #(.SEC1_MAX(SEC1_MAX)) i0(.RESET(RESET), .CLK(CLK), .DEC(DEC), .COUNT(COUNT));

DECODER7 i1(.COUNT(COUNT), .LED(LED), .SA(SA));


endmodule
```

4. P1_TEST_UPDOWN10.v

```
module TEST_UPDOWN10;

reg clk, reset, dec;

wire [7:0] LED;

wire [3:0] sa;

//wire [3:0] count;


parameter CYCLE = 100;

parameter SIM_SEC1_MAX = 4;

integer j;


reg [8*4:1] A_DISP, D_DISP, G_DISP;

reg [8*2:1] B_DISP, C_DISP, E_DISP, F_DISP, Dp_DISP;

parameter TURN_ON = 1'b1;


UPDOWN_7SEG #(.SEC1_MAX(SIM_SEC1_MAX)) i1(/*.COUNT(count),*/ .RESET(reset), .CLK(clk), .DEC(dec), .LED(LED),
.SA(sa));


always #(CYCLE/2) clk = ~clk;


initial begin

reset = 1'b1; clk = 1'b0; dec = 1'b0;

#CYCLE reset = 1'b0;

#(15*CYCLE*SIM_SEC1_MAX) dec = 1'b1;

#(10*CYCLE*SIM_SEC1_MAX) $finish;

end


always @(LED)

begin

for (j = 7; j >= 0; j = j - 1)

begin

case (j)

7 : begin

if (LED[j] === TURN_ON)

A_DISP = "----";

else if (LED[j] === 1'bx)

A_DISP = "xxxx";

else

A_DISP = " ";

end

6 : begin
```

```
if (LED[j] === TURN_ON)
B_DISP = "| ";
else if (LED[j] === 1'bx)
B_DISP = "x ";
else
B_DISP = " ";
end
5 : begin
if (LED[j] === TURN_ON)
C_DISP = "| ";
else if (LED[j] === 1'bx)
C_DISP = "x ";
else
C_DISP = " ";
end
4 : begin
if (LED[j] === TURN_ON)
D_DISP = "----";
else if (LED[j] === 1'bx)
D_DISP = "xxxx";
else
D_DISP = " ";
end
3 : begin
if (LED[j] === TURN_ON)
E_DISP = "| ";
else if (LED[j] === 1'bx)
E_DISP = "x ";
else
E_DISP = " ";
end
2 : begin
if (LED[j] === TURN_ON)
F_DISP = "| ";
else if (LED[j] === 1'bx)
F_DISP = "x ";
else
F_DISP = " ";
```

```
end
1 : begin
if (LED[j] === TURN_ON)
G_DISP = "----";
else if (LED[j] === 1'bx)
G_DISP = "xxxx";
else
G_DISP = " ";
end
0 : begin
if (LED[j] === TURN_ON)
Dp_DISP = " .";
else if (LED[j] === 1'bx)
Dp_DISP = " x";
else
Dp_DISP = " ";
end
endcase
end
#5
// $display("COUNT is %h", count);
$display("");
$display(" %s",A_DISP);
$display(" %s %s",F_DISP, B_DISP);
$display(" %s",G_DISP);
$display(" %s %s",E_DISP, C_DISP);
$display(" %s %s",D_DISP, Dp_DISP);
$display("");
end

initial
$monitor($time,,"clk=%b reset=%b dec=%b LED=%b sa=%b", clk, reset, dec, LED, sa);
endmodule
```