P2: Clock Counter (HMS)

60/24 counters with dynamic LED control

Table of Contents

1. P2_CNT60.v

```
module CNT60(RESET, CLK, DEC, CNT6, CNT10, ENABLE, CARRY_in, CARRY_out);

input RESET, CLK, DEC, ENABLE, CARRY_in;

output reg CARRY_out;

output [3:0] CNT10;

output [3:0] CNT6;

reg [3:0] CNT10;

reg [3:0] CNT6;

reg CARRY;


always @(posedge CLK or posedge RESET)

begin

if (RESET == 1'b1)

begin

CNT10 <= 4'h0;

end

else if (ENABLE == 1'b1 && CARRY_in == 1'b1)

// else if (DEC == 1'b1)


if (DEC == 1'b0)

begin

// if (CNT10 == 4'h9)

if (CARRY == 1'b1)

CNT10 <= 4'h0;

else

CNT10 <= CNT10 + 4'h1;

end

else

begin

// if (CNT10 == 4'h0)

if (CARRY == 1'b1)

CNT10 <= 4'h9;

else

CNT10 <= CNT10 - 4'h1;

end

end


always @(CNT10 or DEC or CARRY_in)

begin
```

```
if (DEC == 1'b0)
begin
if (CNT10 == 4'h9 && CARRY_in == 1'b1)
CARRY <= 1'b1;
else
CARRY <= 1'b0;
end
else
begin
if (CNT10 == 4'h0 && CARRY_in == 1'b1)
CARRY <= 1'b1;
else
CARRY <= 1'b0;
end
end

always @(CNT6 or DEC or CARRY)
begin
if (DEC == 1'b0)
begin
if (CNT6 == 4'h5 && CARRY == 1'b1)
CARRY_out <= 1'b1;
else
CARRY_out <= 1'b0;
end
else
begin
if (CNT6 == 4'h0 && CARRY == 1'b1)
CARRY_out<= 1'b1;
else
CARRY_out <= 1'b0;
end
end

always @(posedge CLK or posedge RESET)
begin
if (RESET == 1'b1)
begin
CNT6 <= 3'b000;
```

```
end

else if (ENABLE == 1'b1 && CARRY == 1'b1)

// else if (DEC == 1'b1)

if (DEC == 1'b0)

begin

if (CNT6 == 3'b101)

CNT6 <= 3'b000;

else

CNT6 <= CNT6 + 3'b001;

end

else

begin

if (CNT6 == 3'b000)

CNT6 <= 3'b101;

else

CNT6 <= CNT6 - 3'b001;

end

end


endmodule
```

2. P2_CNT24.v

```
module CNT24(RESET, CLK, DEC, CNT3, CNT10, ENABLE, CARRY_in, CARRY_out);

input RESET, CLK, DEC, ENABLE, CARRY_in;

output reg CARRY_out;

output [3:0] CNT10;

output [3:0] CNT3;

reg [3:0] CNT10;

reg [3:0] CNT3;

reg CARRY;

wire [3:0] max10;


always @(posedge CLK or posedge RESET)

begin

if (RESET == 1'b1)

begin

CNT10 <= 4'h0;

end

else if (ENABLE == 1'b1 && CARRY_in == 1'b1) begin

// else if (DEC == 1'b1)


if (DEC == 1'b0)

begin

// if (CNT10 == 4'h9)

if (CARRY == 1'b1)

CNT10 <= 4'h0;

else

CNT10 <= CNT10 + 4'h1;

end

else begin

// if (CNT10 == 4'h0)

if(CARRY == 1'b1)

CNT10 <= (CNT3 == 4'h0) ? 4'h3 : 4'h9;

else

CNT10 <= CNT10 - 4'h1;

end

end

end


always @(CNT10 or DEC or CARRY_in)
```

```
begin

if (DEC == 1'b0)

begin

if ((CNT10 == 4'h9 || {CNT3,CNT10} == 8'h23) && CARRY_in == 1'b1)

CARRY <= 1'b1;

else

CARRY <= 1'b0;

end

else

begin

if (CNT10 == 4'h0 && CARRY_in == 1'b1)

CARRY <= 1'b1;

else

CARRY <= 1'b0;

end

end


always @(CNT3 or DEC or CARRY)

begin

if (DEC == 1'b0)

begin

if (CNT3 == 4'h2 && CARRY == 1'b1)

CARRY_out <= 1'b1;

else

CARRY_out <= 1'b0;

end

else

begin

if (CNT3 == 4'h0 && CARRY == 1'b1)

CARRY_out<= 1'b1;

else

CARRY_out <= 1'b0;

end

end


always @(posedge CLK or posedge RESET)

begin

if (RESET == 1'b1)

begin
```

```
CNT3 <= 2'b00;
end
else if (ENABLE == 1'b1 && CARRY == 1'b1)
begin
// else if (DEC == 1'b1)
if (DEC == 1'b0)
begin
if (CNT3 == 2'h2)
CNT3 <= 2'h0;
else
CNT3 <= CNT3 + 2'h1;
end
else
begin
if (CNT3 == 2'h0)
CNT3 <= 2'h2;
else
CNT3 <= CNT3 - 2'h1;
end
end
end

endmodule
```

3. P2_DCOUNT.v

```
module DCOUNT(CLK, ENABLE, L1, L2, L3, L4, SA, L);
input CLK, ENABLE;
input [7:0] L1, L2, L3, L4;
output [3:0] SA;
output [7:0] L;

parameter MAX_COUNT = 3'b111;
reg [2:0] sa_count_tmp;
reg [3:0] sa_count;
reg [7:0] L_tmp;

assign SA[3] = (sa_count[3]==1'b1)? 1'b1 : 1'b0;
assign SA[2] = (sa_count[2]==1'b1)? 1'b1 : 1'b0;
assign SA[1] = (sa_count[1]==1'b1)? 1'b1 : 1'b0;
assign SA[0] = (sa_count[0]==1'b1)? 1'b1 : 1'b0;
assign L = L_tmp;

always @(posedge CLK)
begin
if (ENABLE==1'b1)
if (sa_count_tmp==MAX_COUNT)
sa_count_tmp <= 3'b000;
else
sa_count_tmp <= sa_count_tmp + 1'b1;
end

always @(posedge CLK)
begin
if (sa_count_tmp[0]==1'b0)
begin
sa_count <= 4'b0000;L_tmp <= L_tmp;
end
else
case (sa_count_tmp[2:1])
2'b00:begin
sa_count <= 4'b1000;L_tmp <= L4;
end
2'b01:begin
```

```
sa_count <= 4'b0100;L_tmp <= L3;

end

2'b10:begin

sa_count <= 4'b0010;L_tmp <= L2;

end

2'b11:begin

sa_count <= 4'b0001;L_tmp <= L1;

end

default:begin

sa_count <= 4'bxxxx;L_tmp <= 8'bxxxxxxxx;

end

endcase

end


endmodule
```

4. P2_DECODER7.v

```
module DECODER7(COUNT, LED);

input [3:0] COUNT;

output reg [7:0] LED;


always @(COUNT) begin

case (COUNT) // ABCDEFG Dp

4'b0000: LED <= ~8'b0000001_1;

4'b0001: LED <= ~8'b1001111_1;

4'b0010: LED <= ~8'b0010010_1;

4'b0011: LED <= ~8'b0000110_1;

4'b0100: LED <= ~8'b1001100_1;

4'b0101: LED <= ~8'b0100100_1;

4'b0110: LED <= ~8'b0100000_1;

4'b0111: LED <= ~8'b0001101_1;

4'b1000: LED <= ~8'b0000000_1;

4'b1001: LED <= ~8'b0000100_1;

default: LED <= ~8'b0110000_1;

endcase

end

endmodule
```

5. P2_display_switch.v

```
module display_switch( CLK, btn_switch, C1_in, C2_in, C3_in, C4_in,
C5_in, C6_in, C7_in, C8_in,
C1_out, C2_out, C3_out, C4_out);
input btn_switch, CLK;
input [3:0] C1_in, C2_in, C3_in, C4_in, C5_in, C6_in, C7_in, C8_in;
output reg [3:0] C1_out, C2_out, C3_out, C4_out;

always @(posedge CLK)
begin
if (btn_switch == 1'b0)
begin
C1_out <= C1_in;
C2_out <= C2_in;
C3_out <= C3_in;
C4_out <= C4_in;
end
else
begin
C1_out <= C5_in;
C2_out <= C6_in;
C3_out <= C7_in;
C4_out <= C8_in;
end
end

endmodule
```

6. P2_CNT246060_ALL.v

```
module CNT246060_ALL(CLK, RESET, DEC, LED, SA, btn_switch);

input CLK, RESET, DEC, btn_switch;

output [7:0] LED;

output [3:0] SA;


reg [26:0] tmp_count;


wire [3:0] CNT10;

wire [2:0] CNT6;

wire [3:0] CNT10_2;

wire [2:0] CNT6_2;

wire [3:0] CNT10_3;

wire [1:0] CNT3;

wire [3:0] CNT_1, CNT_2, CNT_3, CNT_4;

wire ENABLE, ENABLE_kHz;

wire [7:0] L1, L2, L3, L4;

wire CARRY, CARRY_2, CARRY_3;


parameter SEC1_MAX = 125000000; // 125MHz


always @(posedge CLK)

begin

if (RESET == 1'b1)

tmp_count <= 27'h000000;

else if (ENABLE == 1'b1)

tmp_count <= 27'h000000;

else

tmp_count <= tmp_count + 27'h1;

end


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

assign ENABLE_kHz = (tmp_count[11:0] == 12'hfff)? 1'b1 : 1'b0;


CNT60 i0(.CLK(CLK), .RESET(RESET), .DEC(DEC), .ENABLE(ENABLE), .CARRY_in(1'b1), .CARRY_out(CARRY),

.CNT10(CNT10), .CNT6(CNT6));

DECODER7 i1(.COUNT(CNT_1), .LED(L1));

DECODER7 i2(.COUNT(CNT_2), .LED(L2));

CNT60 i4(.CLK(CLK), .RESET(RESET), .DEC(DEC), .ENABLE(ENABLE), .CARRY_in(CARRY), .CARRY_out(CARRY_2),

.CNT10(CNT10_2), .CNT6(CNT6_2));
```

```
DECODER7 i5(.COUNT(CNT_3), .LED(L3));

DECODER7 i6(.COUNT(CNT_4), .LED(L4));

DCOUNT i3(.CLK(CLK), .ENABLE(ENABLE_kHz), .L1(L1), .L2(L2),

.L3(L3), .L4(L4), .SA(SA), .L(LED));

CNT24 i7(.CLK(CLK), .RESET(RESET), .DEC(DEC), .ENABLE(ENABLE), .CARRY_in(CARRY_2), .CARRY_out(CARRY_3),

.CNT10(CNT10_3), .CNT3(CNT3));

display_switch i8( .CLK(CLK), .btn_switch(btn_switch),

.C1_in(CNT10), .C2_in({1'b0,CNT6}), .C3_in(CNT10_2), .C4_in({1'b0,CNT6_2}),

.C5_in(CNT10_3), .C6_in({2'b00,CNT3}), .C7_in(4'b0000), .C8_in(4'b0000),

.C1_out(CNT_1), .C2_out(CNT_2), .C3_out(CNT_3), .C4_out(CNT_4));


endmodule
```

7. P2_TEST_CNT246060_ALL.v

```
module TEST_CNT246060_ALL;

parameter MAX_NUM = 60*60*24;

reg clk, reset, dec, btn_switch;

wire [7:0] led;

wire [3:0] sa;

reg [23:0] ref [0:MAX_NUM - 1];

reg [23:0] cnt_value, cnt_value_ref;

integer i;

reg [16:0] ok_count;


parameter CYCLE = 100;

parameter SIM_SEC1_MAX = 4;


CNT246060_ALL #(.SEC1_MAX(SIM_SEC1_MAX)) i1(.RESET(reset), .CLK(clk), .DEC(dec), .btn_switch(btn_switch), .LED(led),
.SA(sa));


always @(posedge clk)

begin

cnt_value = { 2'b0, i1.CNT3, i1.CNT10_3,

1'b0, i1.CNT6_2, i1.CNT10_2,

1'b0, i1.CNT6, i1.CNT10};

end


always #(CYCLE/2)

clk = ~clk;


initial

begin

$readmemh("ref.hex", ref);

end


initial

begin

reset = 1'b1; clk = 1'b0; dec = 1'b0; btn_switch = 1'b0; ok_count = 17'b0;

cnt_value_ref = ref[0];

#CYCLE reset = 1'b0;

@(posedge i1.ENABLE);

@(negedge clk);

if (cnt_value !== cnt_value_ref)begin

$display("Error at step %d: cnt_value=%X expected=%X", 0, cnt_value, ref[0]);
```

```
$display("Total OK steps = %d", ok_count);

$stop;

end

else begin

ok_count = ok_count + 1;

end

for (i = 1; i < MAX_NUM; i = i + 1)

begin

@(negedge i1.ENABLE);

cnt_value_ref = ref[i];

@(posedge i1.ENABLE);

@(negedge clk);

if (cnt_value !== cnt_value_ref)begin

$display("Error at step %d: cnt_value=%X expected=%X", i, cnt_value, ref[i]);

$display("Total OK steps = %d", ok_count);

$stop;

end

else

ok_count = ok_count + 1;

end

$display("Total OK steps = %d", ok_count);


$finish;

end


//initial

// $monitor($time,,"clk=%b reset=%b cnt_value=%b", clk, reset, cnt_value);


endmodule
```

8. P2_CountReference.c

```
#include <stdio.h>

int main(void) {
int h, m, s, hh, mm, ss;

FILE* fp = fopen("ref.hex", "w");
if (!fp) return 1;

for ( h = 0; h < 24; h++) {
for ( m = 0; m < 60; m++) {
for ( s = 0; s < 60; s++) {
hh = ((h / 10) << 4) | (h % 10);
mm = ((m / 10) << 4) | (m % 10);
ss = ((s / 10) << 4) | (s % 10);

fprintf(fp, "%02X%02X%02X\n", hh, mm, ss);
}
}
}

fclose(fp);
return 0;
}
```