P4: Complete Clock System

Fully integrated digital clock (101-year operation)

Table of Contents

1. P4_CLOCK_ALL.v

```
module CLOCK_ALL(CLK, RESET, LED, SA, btn_switch, btn_display, btn_inc);

input CLK, RESET, btn_switch, btn_display, btn_inc;

output [7:0] LED;

output [3:0] SA;


reg [26:0] tmp_count;


wire [3:0] SEC_CNT10;

wire [3:0] SEC_CNT6;

wire [3:0] MIN_CNT10;

wire [3:0] MIN_CNT6;

wire [3:0] HOU_CNT10;

wire [3:0] HOU_CNT3;

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

wire ENABLE, ENABLE_kHz;

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

wire CARRY, CARRY_2, CARRY_3, CARRY_4, CARRY_5, CARRY_6, is_leap, TOP_MODE, DIS_MODE, SET_MODE, INC_MODE;

wire [7:0] month;

wire [11:0] year;

//wire [15:0] WEEKDAY_LED;

wire [3:0] DAY_CNT10, DAY_CNT4, MON_CNT10, MON_CNT2, YEA_CNT10_1, YEA_CNT10_2, YEA_CNT2, week_day;

wire [4:0] TOP_CURRENT_STATE;

wire [6:0] DIS_CURRENT_STATE, SET_CURRENT_STATE;

assign month = {MON_CNT2, MON_CNT10};

assign year = {YEA_CNT2, YEA_CNT10_2, YEA_CNT10_1};


parameter SEC1_MAX = 125000000; // 125MHz


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

.CNT10(SEC_CNT10), .CNT6(SEC_CNT6), .SET_CURRENT_STATE({SET_CURRENT_STATE[6], SET_CURRENT_STATE[0]}));

DECODER7 i1(.COUNT(CNT), .LED(LED), .TOP_CURRENT_STATE(TOP_CURRENT_STATE[0]),
.DIS_CURRENT_STATE(DIS_CURRENT_STATE[3:2]), .SA(SA));

CNT60 i2(.CLK(CLK), .RESET(RESET), .ENABLE(ENABLE), .CARRY_in(CARRY), .CARRY_out(CARRY_2), .INC_MODE(INC_MODE),

.CNT10(MIN_CNT10), .CNT6(MIN_CNT6), .SET_CURRENT_STATE({SET_CURRENT_STATE[5], SET_CURRENT_STATE[0]}));

DCOUNT i3(.CLK(CLK), .ENABLE(ENABLE_kHz), .L1(CNT_1), .L2(CNT_2),

.L3(CNT_3), .L4(CNT_4), .SA(SA), .L(CNT));

CNT24 i4(.CLK(CLK), .RESET(RESET), .ENABLE(ENABLE), .CARRY_in(CARRY_2), .CARRY_out(CARRY_3), .INC_MODE(INC_MODE),

.CNT10(HOU_CNT10), .CNT3(HOU_CNT3), .SET_CURRENT_STATE({SET_CURRENT_STATE[4], SET_CURRENT_STATE[0]}));

SEC1 #(.SEC1_MAX(SEC1_MAX)) i5(.CLK(CLK), .RESET(RESET), .ENABLE(ENABLE), .ENABLE_kHz(ENABLE_kHz));
```

```
CNT_DAY i6( .CLK(CLK), .RESET(RESET), .ENABLE(ENABLE), .CARRY_in(CARRY_3), .CARRY_out(CARRY_4), .INC_MODE(INC_MODE),

.CNT10(DAY_CNT10), .CNT4(DAY_CNT4), .month(month), .is_leap(is_leap), .SET_CURRENT_STATE({SET_CURRENT_STATE[3],
SET_CURRENT_STATE[0]}));

CNT_MONTH i7( .CLK(CLK), .RESET(RESET), .ENABLE(ENABLE), .CARRY_in(CARRY_4), .CARRY_out(CARRY_5),
.INC_MODE(INC_MODE),

.CNT10(MON_CNT10), .CNT2(MON_CNT2), .SET_CURRENT_STATE({SET_CURRENT_STATE[2], SET_CURRENT_STATE[0]}));

CNT_YEAR i8( .CLK(CLK), .RESET(RESET), .ENABLE(ENABLE), .CARRY_in(CARRY_5), .CARRY_out(CARRY_6),
.INC_MODE(INC_MODE),

.CNT10(YEA_CNT10_1), .CNT10_2(YEA_CNT10_2), .CNT2(YEA_CNT2), .SET_CURRENT_STATE({SET_CURRENT_STATE[1],
SET_CURRENT_STATE[0]}));

leap_year i9( .year_bcd(year), .is_leap(is_leap));

//LED_WEEK i10( .week_day(week_day), .LED(WEEKDAY_LED));

weekday_calc i10(.year_bcd(year), .month_bcd(month), .day_bcd({DAY_CNT4, DAY_CNT10}), .clk(CLK),
.weekday(week_day));

MSE i11(.CLK(CLK), .ENABLE_kHz(ENABLE_kHz), .MODE(TOP_MODE), .MODE_IN(btn_switch));

MAIN_MODE i12(.CLK(CLK), .RESET(RESET), .MODE(TOP_MODE), .CURRENT_STATE(TOP_CURRENT_STATE));

MSE i13(.CLK(CLK), .ENABLE_kHz(ENABLE_kHz), .MODE(DIS_MODE), .MODE_IN(btn_display));

DIS_MODE i14(.CLK(CLK), .RESET(RESET), .MODE(DIS_MODE), .CURRENT_STATE(DIS_CURRENT_STATE),
.main_state_active(TOP_CURRENT_STATE[0]));

MSE i15(.CLK(CLK), .ENABLE_kHz(ENABLE_kHz), .MODE(SET_MODE), .MODE_IN(btn_display));

SET_MODE i16(.CLK(CLK), .RESET(RESET), .MODE(SET_MODE), .CURRENT_STATE(SET_CURRENT_STATE),
.main_state_active(TOP_CURRENT_STATE[1]));

MSE i17(.CLK(CLK), .ENABLE_kHz(ENABLE_kHz), .MODE(INC_MODE), .MODE_IN(btn_inc));

display_switch i18( .TOP_CURRENT_STATE(TOP_CURRENT_STATE[1:0]), .DIS_CURRENT_STATE(DIS_CURRENT_STATE),
.SET_CURRENT_STATE(SET_CURRENT_STATE),

.SEC_CNT10(SEC_CNT10), .SEC_CNT6(SEC_CNT6), .MIN_CNT10(MIN_CNT10), .MIN_CNT6(MIN_CNT6),

.HOU_CNT10(HOU_CNT10), .HOU_CNT3(HOU_CNT3), .DAY_CNT10(DAY_CNT10), .DAY_CNT4(DAY_CNT4), .week_day(week_day),

.MON_CNT10(MON_CNT10), .MON_CNT2(MON_CNT2), .YEA_CNT10_1(YEA_CNT10_1), .YEA_CNT10_2(YEA_CNT10_2),
.YEA_CNT2(YEA_CNT2),

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


endmodule
```

2. P4_CNT60.v

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

input RESET, CLK, ENABLE, CARRY_in, INC_MODE;

input [1:0] SET_CURRENT_STATE;

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) && SET_CURRENT_STATE[0] == 1'b1) ||

(SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 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 CARRY_in or SET_CURRENT_STATE or INC_MODE)

begin

// if (DEC == 1'b0)

begin

if (CNT10 == 4'h9 && (CARRY_in == 1'b1 || (SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 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 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 ((CARRY == 1'b1) && ((ENABLE == 1'b1 && SET_CURRENT_STATE[0] == 1'b1) ||

(SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 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
```

3. P4_CNT24.v

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

input RESET, CLK, ENABLE, CARRY_in, INC_MODE;

input [1:0] SET_CURRENT_STATE;

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) && SET_CURRENT_STATE[0] == 1'b1) ||

(SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 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 CARRY_in or SET_CURRENT_STATE or INC_MODE)

begin

// if (DEC == 1'b0)

begin

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

(CARRY_in == 1'b1 || (SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 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 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 ((CARRY == 1'b1) && ((ENABLE == 1'b1 && SET_CURRENT_STATE[0] == 1'b1) ||

(SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 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
```

4. P4_CNT_DAY.v

```
module CNT_DAY( RESET, CLK, CNT4, CNT10, ENABLE,

CARRY_in, CARRY_out, month, is_leap, SET_CURRENT_STATE, INC_MODE);

input RESET, CLK, ENABLE, CARRY_in, is_leap, INC_MODE;

input [1:0] SET_CURRENT_STATE;

input [7:0] month;


output reg CARRY_out;

output [3:0] CNT10;

output [3:0] CNT4;

reg [3:0] CNT10;

reg [3:0] CNT4;

reg CARRY;



always @(posedge CLK or posedge RESET)

begin

if (RESET == 1'b1) begin

CNT10 <= 4'h1;

end

else if (((ENABLE == 1'b1 && CARRY_in == 1'b1) && SET_CURRENT_STATE[0] == 1'b1) ||

(SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 1'b1)) begin

if (CARRY_out == 1'b1)

CNT10 <= 4'h1;

else if(CARRY == 1'b1)

CNT10 <= 4'h0;

else begin

CNT10 <= CNT10 + 4'h1;

end

end

end


always @(CNT10 or CARRY_in or CNT4 or month or is_leap or SET_CURRENT_STATE or INC_MODE)

begin


if ((((month == 8'h04 || month == 8'h06 || month == 8'h09 || month == 8'h11) && ({CNT4, CNT10} == 8'h30)) ||

((month == 8'h02) && ((is_leap == 1'b0) && ({CNT4, CNT10} == 8'h28))) ||

(({CNT4, CNT10} == 8'h31) || (CNT10 == 4'h9))) &&

(CARRY_in == 1'b1 || (SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 1'b1)))

CARRY <= 1'b1;
```

```
else

CARRY <= 1'b0;


end


always @(CNT4 or CARRY or CNT10 or is_leap)

begin


if (CARRY == 1'b1 && ((CNT10 != 4'h9) || (((month == 8'h02) && (is_leap == 1'b1) && ({CNT4, CNT10} == 8'h29)))))

CARRY_out <= 1'b1;

else

CARRY_out <= 1'b0;


end


always @(posedge CLK or posedge RESET)

begin

if (RESET == 1'b1)

begin

CNT4 <= 4'h0;

end

else if ((CARRY == 1'b1) && ((ENABLE == 1'b1 && SET_CURRENT_STATE[0] == 1'b1) ||

(SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 1'b1)))

if(CARRY_out == 1'b1)

CNT4 <= 4'h0;

else

CNT4 <= CNT4 + 4'h1;

end



endmodule
```

5. P4_CNT_MONTH.v

```
module CNT_MONTH( RESET, CLK, CNT2, CNT10, ENABLE,

CARRY_in, CARRY_out, SET_CURRENT_STATE, INC_MODE);

input RESET, CLK, ENABLE, CARRY_in, INC_MODE;

input [1:0] SET_CURRENT_STATE;

output reg CARRY_out;

output [3:0] CNT10;

output [3:0] CNT2;

reg [3:0] CNT10;

reg [3:0] CNT2;

reg CARRY;


always @(posedge CLK or posedge RESET)

begin

if (RESET == 1'b1) begin

CNT10 <= 4'h1;

end

else if (((ENABLE == 1'b1 && CARRY_in == 1'b1) && SET_CURRENT_STATE[0] == 1'b1) ||

(SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 1'b1)) begin

if (CARRY_out == 1'b1)

CNT10 <= 4'h1;

else if (CARRY == 1'b1)

CNT10 <= 4'h0;

else

CNT10 <= CNT10 + 4'h1;

end

end


always @(CNT10 or CARRY_in or CNT2 or SET_CURRENT_STATE or INC_MODE)

begin


if ((CNT10 == 4'h9 || {CNT2,CNT10} == 8'h12) &&

(CARRY_in == 1'b1 || (SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 1'b1)))

CARRY <= 1'b1;

else

CARRY <= 1'b0;


end


always @(CNT2 or CARRY)
```

```
begin

if (CNT2 == 4'h1 && CARRY == 1'b1)

CARRY_out <= 1'b1;

else

CARRY_out <= 1'b0;

end

always @(posedge CLK or posedge RESET)

begin

if (RESET == 1'b1)

begin

CNT2 <= 4'h0;

end

else if ((CARRY == 1'b1) && ((ENABLE == 1'b1 && SET_CURRENT_STATE[0] == 1'b1) ||

(SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 1'b1)))

begin

if (CARRY_out == 1'b1)

CNT2 <= 4'h0;

else

CNT2 <= CNT2 + 4'h1;

end

end

endmodule
```

6. P4_CNT_YEAR.v

```
module CNT_YEAR( RESET, CLK, CNT10_2, CNT10, CNT2, ENABLE,

CARRY_in, CARRY_out, SET_CURRENT_STATE, INC_MODE);

input RESET, CLK, ENABLE, CARRY_in, INC_MODE;

input [1:0] SET_CURRENT_STATE;

output reg CARRY_out;

output [3:0] CNT10;

output [3:0] CNT10_2;

output [3:0] CNT2;

reg [3:0] CNT10;

reg [3:0] CNT10_2;

reg [3:0] CNT2;

reg CARRY, CARRY_2;


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) && SET_CURRENT_STATE[0] == 1'b1) ||

(SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 1'b1)) begin

if (CARRY == 1'b1)

CNT10 <= 4'h0;

else

CNT10 <= CNT10 + 4'h1;

end

end


always @(CNT10 or CARRY_in or SET_CURRENT_STATE or INC_MODE or CNT10_2 or CNT2)

begin


if ((CNT10 == 4'h9 || {CNT2, CNT10_2, CNT10} == 12'h100) && (CARRY_in == 1'b1 || (SET_CURRENT_STATE[1] == 1'b1 &&
INC_MODE == 1'b1)))

CARRY <= 1'b1;

else

CARRY <= 1'b0;


end


always @(CNT10_2 or CARRY or CNT2 or CNT10)
```

```
begin

if ((CNT10_2 == 4'h9 || {CNT2, CNT10_2, CNT10} == 12'h100) && CARRY == 1'b1 )
CARRY_2 <= 1'b1;
else
CARRY_2 <= 1'b0;

end

always @(posedge CLK or posedge RESET)
begin
if (RESET == 1'b1)
begin
CNT10_2 <= 4'h0;
end
else if ((CARRY == 1'b1) && ((ENABLE == 1'b1 && SET_CURRENT_STATE[0] == 1'b1) ||
(SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 1'b1)))
begin
if (CARRY_2 == 1'b1)
CNT10_2 <= 4'h0;
else
CNT10_2 <= CNT10_2 + 4'h1;
end

end

always @(CNT10_2 or CARRY_2 or CNT2 or CNT10)
begin

if ({CNT2, CNT10_2, CNT10} == 12'h100 && CARRY_2 == 1'b1 )
CARRY_out <= 1'b1;
else
CARRY_out <= 1'b0;

end

always @(posedge CLK or posedge RESET)
begin
if (RESET == 1'b1)
begin
CNT2 <= 4'h0;
end
```

```
else if ((CARRY_2 == 1'b1) && ((ENABLE == 1'b1 && SET_CURRENT_STATE[0] == 1'b1) ||

(SET_CURRENT_STATE[1] == 1'b1 && INC_MODE == 1'b1)))

begin

if (CARRY_out == 1'b1)

CNT2 <= 4'h0;

else

CNT2 <= CNT2 + 4'h1;

end


end


endmodule
```

7. P4_DCOUNT.v

```
module DCOUNT(CLK, ENABLE, L1, L2, L3, L4, SA, L);

input CLK, ENABLE;

input [3:0] L1, L2, L3, L4;

output [3:0] SA;

output [3:0] L;


parameter MAX_COUNT = 3'b111;

reg [2:0] sa_count_tmp;

reg [3:0] sa_count;

reg [3:0] 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
```

8. P4_DECODER7.v

```
module DECODER7(COUNT, LED, TOP_CURRENT_STATE, DIS_CURRENT_STATE, SA);

input TOP_CURRENT_STATE;

input [1:0] DIS_CURRENT_STATE;

input [3:0] COUNT, SA;

output reg [7:0] LED;


always @(COUNT or TOP_CURRENT_STATE or DIS_CURRENT_STATE or SA) begin

if(TOP_CURRENT_STATE == 1'b1 && DIS_CURRENT_STATE[0] == 1'b1 && SA[3:2] == 2'b00) begin

case(SA[1:0])

2'b01: case(COUNT)

4'b0000: LED <= 8'b0011100_0;

4'b0001: LED <= 8'b0011101_0;

4'b0010: LED <= 8'b0011100_0;

4'b0011: LED <= 8'b1001111_0;

4'b0100: LED <= 8'b0010111_0;

4'b0101: LED <= 8'b0000101_0;

4'b0110: LED <= 8'b1110111_0;

default: LED <= 8'b0011100_0;

endcase

2'b10: case(COUNT)

4'b0000: LED <= 8'b0011011_0;

4'b0001: LED <= 8'b1110110_0;

4'b0010: LED <= 8'b0001111_0;

4'b0011: LED <= 8'b0101010_0;

4'b0100: LED <= 8'b0001111_0;

4'b0101: LED <= 8'b1000111_0;

4'b0110: LED <= 8'b0011011_0;

default: LED <= 8'b0011011_0;

endcase

default: LED <= 8'b0000000_0;

endcase

end else if(TOP_CURRENT_STATE == 1'b1 && DIS_CURRENT_STATE[1] == 1'b1 && SA[1:0] == 2'b00) begin

case(SA[3:2])

2'b01: case(COUNT)

4'b0000: LED <= 8'b0011100_0;

4'b0001: LED <= 8'b0011101_0;

4'b0010: LED <= 8'b0011100_0;
```

```
4'b0011: LED <= 8'b1001111_0;

4'b0100: LED <= 8'b0010111_0;

4'b0101: LED <= 8'b0000101_0;

4'b0110: LED <= 8'b1110111_0;

default: LED <= 8'b0011100_0;

endcase

2'b10: case(COUNT)

4'b0000: LED <= 8'b0011011_0;

4'b0001: LED <= 8'b1110110_0;

4'b0010: LED <= 8'b0001111_0;

4'b0011: LED <= 8'b0101010_0;

4'b0100: LED <= 8'b0001111_0;

4'b0101: LED <= 8'b1000111_0;

4'b0110: LED <= 8'b0011011_0;

default: LED <= 8'b0011011_0;

endcase

default: LED <= 8'b0000000_0;

endcase

end else 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

end

endmodule
```

9. P4_display_switch.v

```
module display_switch( TOP_CURRENT_STATE, DIS_CURRENT_STATE, SET_CURRENT_STATE,

SEC_CNT10, SEC_CNT6, MIN_CNT10, MIN_CNT6, HOU_CNT10, HOU_CNT3, week_day,

DAY_CNT10, DAY_CNT4, MON_CNT10, MON_CNT2, YEA_CNT10_1, YEA_CNT10_2, YEA_CNT2,

C1_out, C2_out, C3_out, C4_out);

input [1:0] TOP_CURRENT_STATE;

input [6:0] DIS_CURRENT_STATE, SET_CURRENT_STATE;

input [3:0] SEC_CNT10, SEC_CNT6, MIN_CNT10, MIN_CNT6, HOU_CNT10, HOU_CNT3, week_day,

DAY_CNT10, DAY_CNT4, MON_CNT10, MON_CNT2, YEA_CNT10_1, YEA_CNT10_2, YEA_CNT2;

output reg [3:0] C1_out, C2_out, C3_out, C4_out;


parameter L1 = 7'b0000001,

L2 = 7'b0000010,

L3 = 7'b0000100,

L4 = 7'b0001000,

L5 = 7'b0010000,

L6 = 7'b0100000,

L7 = 7'b1000000;


always @(TOP_CURRENT_STATE or DIS_CURRENT_STATE or SET_CURRENT_STATE or

SEC_CNT10 or SEC_CNT6 or MIN_CNT10 or MIN_CNT6 or HOU_CNT10 or HOU_CNT3 or week_day or

DAY_CNT10 or DAY_CNT4 or MON_CNT10 or MON_CNT2 or YEA_CNT10_1 or YEA_CNT10_2 or YEA_CNT2)

begin

if (TOP_CURRENT_STATE[0] == 1'b1 || (TOP_CURRENT_STATE[1] == 1'b1 && SET_CURRENT_STATE[0] ==1'b1))

begin

case(DIS_CURRENT_STATE)

L1: begin

C1_out = YEA_CNT10_1;

C2_out = YEA_CNT10_2;

C3_out = YEA_CNT2;

C4_out = 4'h2;

end

L2: begin

C1_out = MON_CNT10;

C2_out = MON_CNT2;

C3_out = YEA_CNT10_1;

C4_out = YEA_CNT10_2;

end
```

```
L3: begin

C1_out = week_day;

C2_out = week_day;

C3_out = MON_CNT10;

C4_out = MON_CNT2;

end

L4: begin

C1_out = DAY_CNT10;

C2_out = DAY_CNT4;

C3_out = week_day;

C4_out = week_day;

end

L5: begin

C1_out = HOU_CNT10;

C2_out = HOU_CNT3;

C3_out = DAY_CNT10;

C4_out = DAY_CNT4;

end

L6: begin

C1_out = MIN_CNT10;

C2_out = MIN_CNT6;

C3_out = HOU_CNT10;

C4_out = HOU_CNT3;

end

L7: begin

C1_out = SEC_CNT10;

C2_out = SEC_CNT6;

C3_out = MIN_CNT10;

C4_out = MIN_CNT6;

end

default: begin

C1_out = YEA_CNT10_1;

C2_out = YEA_CNT10_2;

C3_out = YEA_CNT2;

C4_out = 4'h2;

end

endcase

end
```

```
else if (TOP_CURRENT_STATE[1] == 1'b1)

begin

case(SET_CURRENT_STATE)

L2: begin

C1_out = YEA_CNT10_1;

C2_out = YEA_CNT10_2;

C3_out = YEA_CNT2;

C4_out = 4'h2;

end

L3: begin

C1_out = MON_CNT10;

C2_out = MON_CNT2;

C3_out = YEA_CNT10_1;

C4_out = YEA_CNT10_2;

end

L4: begin

C1_out = DAY_CNT10;

C2_out = DAY_CNT4;

C3_out = MON_CNT10;

C4_out = MON_CNT2;

end

L5: begin

C1_out = HOU_CNT10;

C2_out = HOU_CNT3;

C3_out = DAY_CNT10;

C4_out = DAY_CNT4;

end

L6: begin

C1_out = MIN_CNT10;

C2_out = MIN_CNT6;

C3_out = HOU_CNT10;

C4_out = HOU_CNT3;

end

L7: begin

C1_out = SEC_CNT10;

C2_out = SEC_CNT6;

C3_out = MIN_CNT10;

C4_out = MIN_CNT6;
```

```
end

default: begin

C1_out = YEA_CNT10_1;

C2_out = YEA_CNT10_2;

C3_out = YEA_CNT2;

C4_out = 4'h2;

end

endcase

end

end


endmodule
```

10. P4_leap_year.v

```
module leap_year(year_bcd, is_leap);

input [11:0] year_bcd;

output is_leap;

wire [3:0] ones;

wire [3:0] tens;

wire [3:0] hunds;

wire ones_div4, tens_even, div4, ones_div4_2, tens_odd;

assign ones = year_bcd[3:0];

assign tens = year_bcd[7:4];

assign hunds = year_bcd[11:8];

assign ones_div4 = (ones == 4'h0) || (ones == 4'h4) || (ones == 4'h8);

assign tens_even = ~tens[0];

assign ones_div4_2 = (ones == 4'h2) || (ones == 4'h6);

assign tens_odd = tens[0];

assign div4 = ((ones_div4 && tens_even) || (ones_div4_2 && tens_odd));

assign is_leap = div4 && ~(hunds == 4'h1);

endmodule
```

11. P4_weekday_calc.v

```
module weekday_calc(
input wire [11:0] year_bcd, // ■: 3■BCD
input wire [7:0] month_bcd, // ■: 2■BCD
input wire [7:0] day_bcd, // ■: 2■BCD
input wire clk,
output reg [3:0] weekday // ■■ 0=■■, ..., 6=■■
);

// BCD → ■■■■■■ (■■■■■)
wire [11:0] year_bin;
wire [7:0] month_bin;
wire [7:0] day_bin;

assign year_bin = ((year_bcd[11:8] << 6) + (year_bcd[11:8] << 5) + (year_bcd[11:8] << 2) + // ■■■ *100 = 64+32+4
(year_bcd[7:4] << 3) + (year_bcd[7:4] << 1) + // ■■■ *10 = 8+2
year_bcd[3:0]); // ■■■

assign month_bin = ((month_bcd[7:4] << 3) + (month_bcd[7:4] << 1) + month_bcd[3:0]); // ■■■*10 + ■■■
assign day_bin = ((day_bcd[7:4] << 3) + (day_bcd[7:4] << 1) + day_bcd[3:0]); // ■■■*10 + ■■■

// BRAM■■■■■■ (year*12 + (month-1)) ■■■■■
wire [11:0] bram_addr;
assign bram_addr = (year_bin << 3) + (year_bin << 2) + (month_bin - 1);

// BRAM■■■■■■■■■■■■■
wire [3:0] bram_dout;
blk_mem_gen_0 weekday_bram (
.clka(clk),
.ena(1'b1),
.wea(1'b0),
.addra(bram_addr),
.douta(bram_dout)
);


// ■■■■■BRAM■1■■■■ + day-1 ■7■■■■■■■
always @(posedge clk) begin
weekday <= (bram_dout + day_bin - 1) % 7;
end
```

```
endmodule
```

12. P4_SEC1.v

```
module SEC1(CLK, RESET, ENABLE, ENABLE_kHz);

input CLK, RESET;

output ENABLE, ENABLE_kHz;

reg [26:0] tmp_count;

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;

endmodule
```

13. P4_MAIN_MODE.v

```
module MAIN_MODE(CLK, RESET, MODE, CURRENT_STATE);

input CLK, RESET, MODE;

output reg [4:0] CURRENT_STATE;


parameter L1 = 5'b00001,

L2 = 5'b00010,

L3 = 5'b00100,

L4 = 5'b01000,

L5 = 5'b10000;


reg [4:0] NEXT_STATE;


always @(CURRENT_STATE or MODE) begin

case (CURRENT_STATE)

L1:if(MODE == 1'b1)

NEXT_STATE <= L2;

else

NEXT_STATE <= L1;

L2:if(MODE == 1'b1)

NEXT_STATE <= L3;

else

NEXT_STATE <= L2;

L3:if(MODE == 1'b1)

NEXT_STATE <= L4;

else

NEXT_STATE <= L3;

L4:if(MODE == 1'b1)

NEXT_STATE <= L5;

else

NEXT_STATE <= L4;

L5:if(MODE == 1'b1)

NEXT_STATE <= L1;

else

NEXT_STATE <= L5;

default:NEXT_STATE <= L1;

endcase

end

always@(posedge RESET or posedge CLK) begin
```

```
if(RESET == 1'b1)

CURRENT_STATE <= L1;

else

CURRENT_STATE <= NEXT_STATE;

end


endmodule
```

14. P4_DIS_MODE.v

```
module DIS_MODE(CLK, RESET, MODE, CURRENT_STATE, main_state_active);

input CLK, RESET, MODE, main_state_active;

output reg [6:0] CURRENT_STATE;


parameter L1 = 7'b0000001,

L2 = 7'b0000010,

L3 = 7'b0000100,

L4 = 7'b0001000,

L5 = 7'b0010000,

L6 = 7'b0100000,

L7 = 7'b1000000;


reg [6:0] NEXT_STATE;


always @(CURRENT_STATE or MODE or main_state_active) begin

case (CURRENT_STATE)

L1:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L2;

else

NEXT_STATE <= L1;

L2:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L3;

else

NEXT_STATE <= L2;

L3:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L4;

else

NEXT_STATE <= L3;

L4:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L5;

else

NEXT_STATE <= L4;

L5:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L6;

else

NEXT_STATE <= L5;

L6:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L7;
```

```
else

NEXT_STATE <= L6;

L7:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L1;

else

NEXT_STATE <= L7;

default:NEXT_STATE <= L1;

endcase

end

always@(posedge RESET or posedge CLK) begin

if(RESET == 1'b1)

CURRENT_STATE <= L1;

else

CURRENT_STATE <= NEXT_STATE;

end


endmodule
```

15. P4_SET_MODE.v

```
module SET_MODE(CLK, RESET, MODE, CURRENT_STATE, main_state_active);

input CLK, RESET, MODE, main_state_active;

output reg [6:0] CURRENT_STATE;

reg [1:0] ED;

wire MAIN_STATE_MODE;


parameter L1 = 7'b0000001,

L2 = 7'b0000010,

L3 = 7'b0000100,

L4 = 7'b0001000,

L5 = 7'b0010000,

L6 = 7'b0100000,

L7 = 7'b1000000;


reg [6:0] NEXT_STATE;


always @(CURRENT_STATE or MODE or main_state_active) begin

case (CURRENT_STATE)

L1:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L2;

else

NEXT_STATE <= L1;

L2:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L3;

else

NEXT_STATE <= L2;

L3:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L4;

else

NEXT_STATE <= L3;

L4:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L5;

else

NEXT_STATE <= L4;

L5:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L6;

else

NEXT_STATE <= L5;
```

```
L6:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L7;

else

NEXT_STATE <= L6;

L7:if(MODE == 1'b1 && main_state_active == 1'b1)

NEXT_STATE <= L1;

else

NEXT_STATE <= L7;

default:NEXT_STATE <= L1;

endcase

end


always@(posedge RESET or posedge CLK) begin

if(RESET == 1'b1 || MAIN_STATE_MODE == 1'b1)

CURRENT_STATE <= L1;

else

CURRENT_STATE <= NEXT_STATE;

end


always @(posedge CLK) begin

ED <= {ED[0], main_state_active};

end


assign MAIN_STATE_MODE = ~ED[0] & ED[1];


endmodule
```

16. P4_MSE.v

```
module MSE(CLK, MODE_IN, MODE, ENABLE_kHz);

input CLK, MODE_IN, ENABLE_kHz;

reg [1:0] META;

reg [3:0] SFT;

wire CHATA;

reg [1:0] ED;

output MODE;


always @(posedge CLK) begin

META <= {META[0], MODE_IN};

end


always @(posedge CLK) begin

if(ENABLE_kHz == 1'b1)

SFT <= {SFT[2:0], META[1]};

end


assign CHATA =& SFT;


always @(posedge CLK) begin

ED <= {ED[0], CHATA};

end


assign MODE = ED[0] & ~ED[1];

endmodule
```

17. P4_weekday.c

```
#include <stdio.h>

int zeller(int, int, int);

int isleap(int);

int daysofmonth(int, int);

int main(void) {

int year, month, w;

FILE* fp = fopen("weekday.txt", "w");

for (year = 2000; year <= 2100; year++) {

for (month = 1; month <= 12; month++) {

w = zeller(year, month, 1);

fprintf(fp, "%X\n", w);

}

}

fclose(fp);

return 0;

}

int zeller(int y, int m, int d) {

int i, j, goukei, w;

goukei = 0;

for (i = 1; i < y; i++) {

for (j = 1;j <= 12;j++) {

goukei += daysofmonth(i, j);

}

}

for (j = 1; j < m; j++) {

goukei += daysofmonth(y, j);

}

w = (goukei + d) % 7;

return w;

}

int daysofmonth(int year, int month)

{
```

```
int day;

switch(month){

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12: day = 31;

break;

case 4:

case 6:

case 9:

case 11: day = 30;

break;

case 2: day = 28 + isleap(year);

break;

}

return day;

}

int isleap(int year)

{

return (year % 4 == 0 && year % 100 != 0 || year % 400 == 0);

}
```

18. P4_weekday_coe.c

```
#include <stdio.h>

int main(void)

{

int val, count;

count = 0;

FILE *fin = fopen("weekday.txt", "r");

FILE *fout = fopen("weekday.coe", "w");

fprintf(fout, "memory_initialization_radix=16;\n");

fprintf(fout, "memory_initialization_vector=\n");

while (fscanf_s(fin, "%d", &val) == 1) {

fprintf(fout, "%d", val);

count++;

if (count < 1212) {

fprintf(fout, ",");

}

if (count % 16 == 0) fprintf(fout, "\n");

}

fprintf(fout, ";\n");

fclose(fin);

fclose(fout);

return 0;

}
```