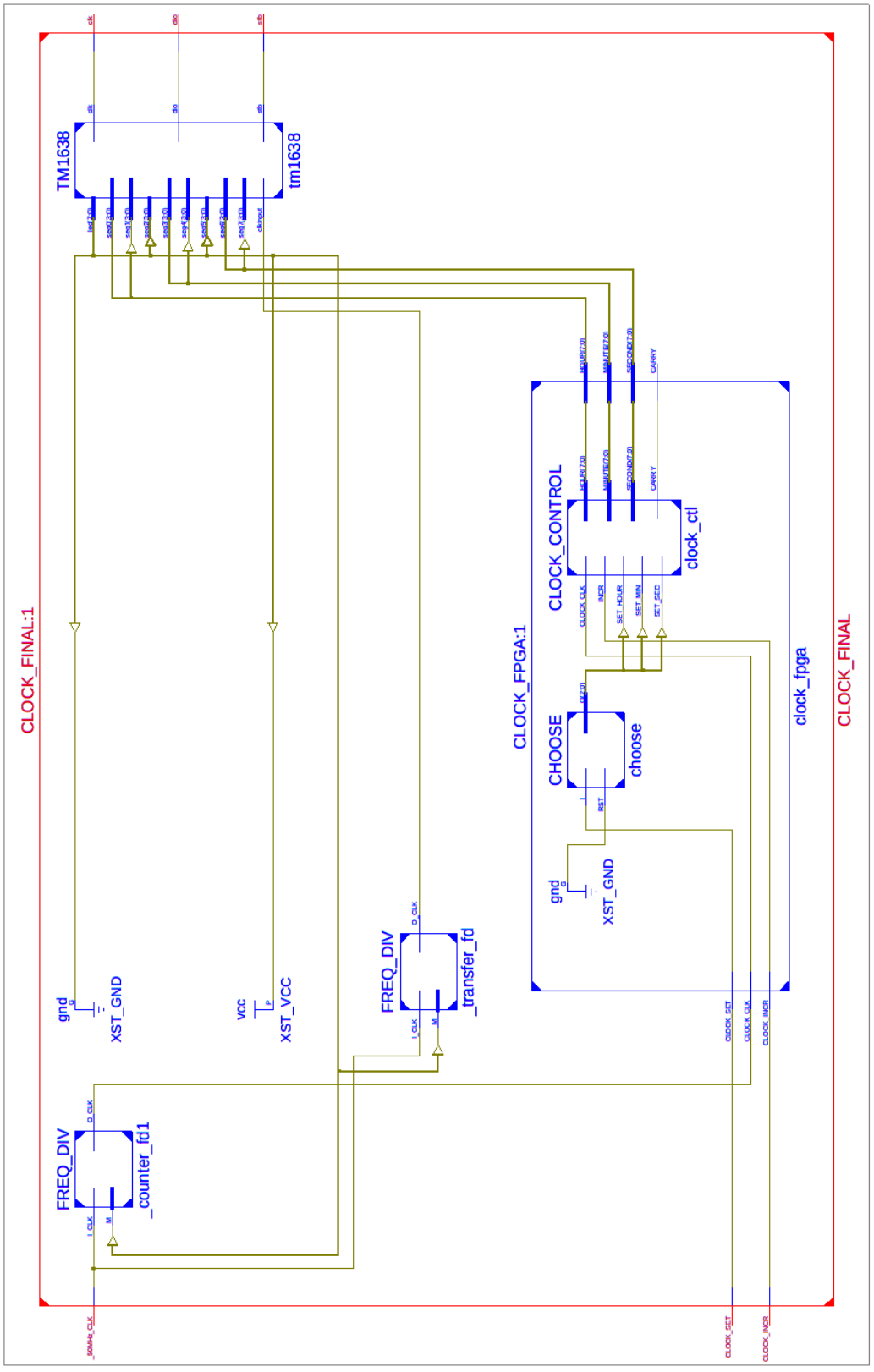
Môn học: TT Thiết kế hệ thống và Vi mạch tích hợp

GVHD: Trần Thị Quỳnh Như

Danh sách thành viên:	
Họ và Tên	MSSV
Nguyễn Thanh Phú	22119211
Trần Thủy Tiên	22119238
Vũ Mai Liên	22119194

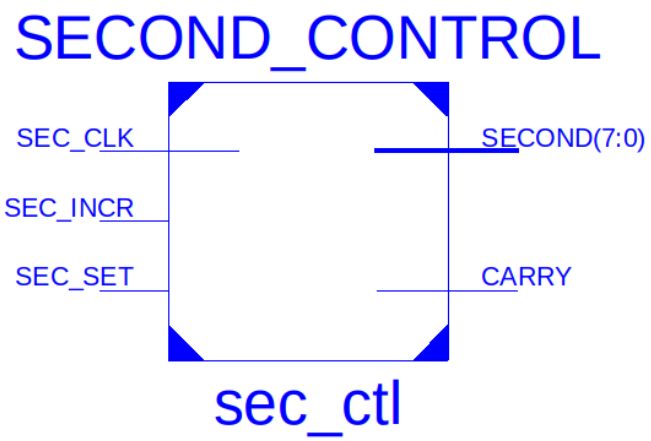
Thiết kế đồng hồ số có điều chỉnh giờ phút

1 - Sơ đồ khối tổng quát (Top-model / level 0)



2 - Khối đếm giây

Sơ đồ khối



Bảng mô tả trạng thái đếm

INPUT			OUTPUT		
SEC_ SET	SEC_ INCR	SEC_CLK	CARRY	SECOND [7:4]	SECOND [3:0]
0	x	x	0	0	0
0	x	↑	0	0	1
0	x	↑	0	0	2
0	x	↑	0	0	3
0	x	↑	0	…	…
0	x	↑	0	5	8
0	x	↑	0	5	9
0	x	↑	1	0	0
0	x	↑	0	0	1
0	x	↑	0	0	2
1	↑	x	0	0	3
1	↑	x	0	…	…
1	↑	x	0	5	8
1	↑	x	0	5	9

S	↑	x	0	0	0
1	↑	x	0	0	1
1	↑	x	0	0	2
1	↑	x	0	0	3
1	↑	x	0	…	…

Verilog HDL

SECOND_CONTROL.v

```
`timescale 1ns / 1ps

module SECOND_CONTROL(
    input SEC_CLK, //  1 pos-edge = +1second
    input SEC_INCR,// 1 pos-edge = +1second
    input SEC_SET, // LOW: (Count CLK) -  HIGH: Set-up (Count INCR)
    output reg CARRY,
    output reg[7:0] SECOND // BCD display <SECOND/10><SECOND%10>
);
    initial begin
        SECOND = 8'H30;
        CARRY = 0;
    end

    wire __triger;
    assign __triger = (SEC_CLK & ~SEC_SET) | (SEC_INCR & SEC_SET);

    always @( posedge __triger ) begin
        // reset CARRY
        CARRY = 0;
        // do nothin'
        if(SECOND[3:0] == 9) begin
            // reset x1
            SECOND[3:0] = 0;
            if(SECOND[7:4] == 5)begin
                // reset x10
                SECOND[7:4] = 0;
                // reset\set CARRY flag
                if(SEC_SET == 0)
                    CARRY = 1;
                else
                    CARRY = 0;
            end
```

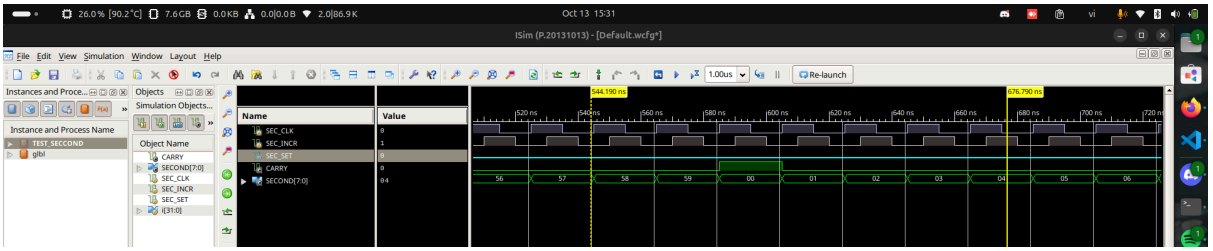
```
            else begin
                // increase x10
                SECOND[7:4] = SECOND[7:4] + 1;
            end
        end
        else begin
            // increase x1
            SECOND[3:0] = SECOND[3:0] + 1;
        end
    end

endmodule
```

Testbench

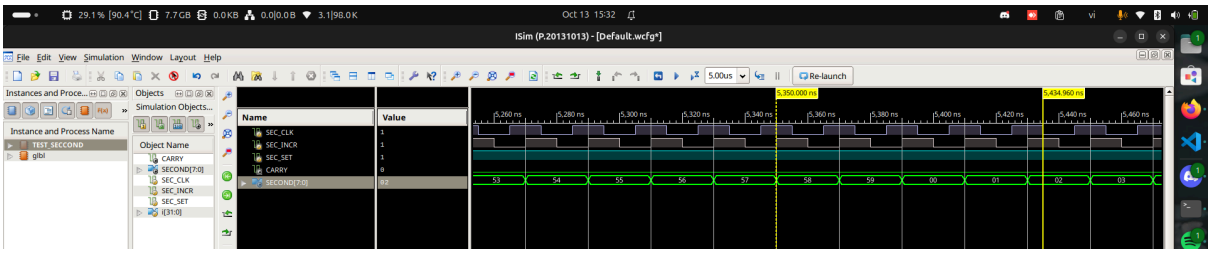
TEST_SECOND.v

```
`timescale 1ns / 1ps
`include "SECOND_CONTROL.v"

module TEST_SECOND;

   // Inputs
   reg SEC_CLK;
   reg SEC_INCR;
   reg SEC_SET;

   // Outputs
   wire CARRY;
   wire [7:0] SECOND;
   integer i;
   // Instantiate the Unit Under Test (UUT)
   SECOND_CONTROL uut (
        .SEC_CLK(SEC_CLK),
        .SEC_INCR(SEC_INCR),
        .SEC_SET(SEC_SET),
        .CARRY(CARRY),
        .SECOND(SECOND)
   );

   initial begin
        // Initialize Inputs
        SEC_CLK = 0;SEC_INCR = 0;SEC_SET = 0;
        SEC_SET = 0;
        for( i = 0; i < 175; i=i+1) begin
            #5 SEC_CLK = ~SEC_CLK; #5 SEC_INCR = ~SEC_INCR;
            #5 SEC_CLK = ~SEC_CLK; #5 SEC_INCR = ~SEC_INCR;
```

```
        end
        SEC_SET = 1;
        for( i = 0; i < 175; i=i+1) begin
            #5 SEC_CLK = ~SEC_CLK; #5 SEC_INCR = ~SEC_INCR;
            #5 SEC_CLK = ~SEC_CLK; #5 SEC_INCR = ~SEC_INCR;
        end
    end

endmodule
```



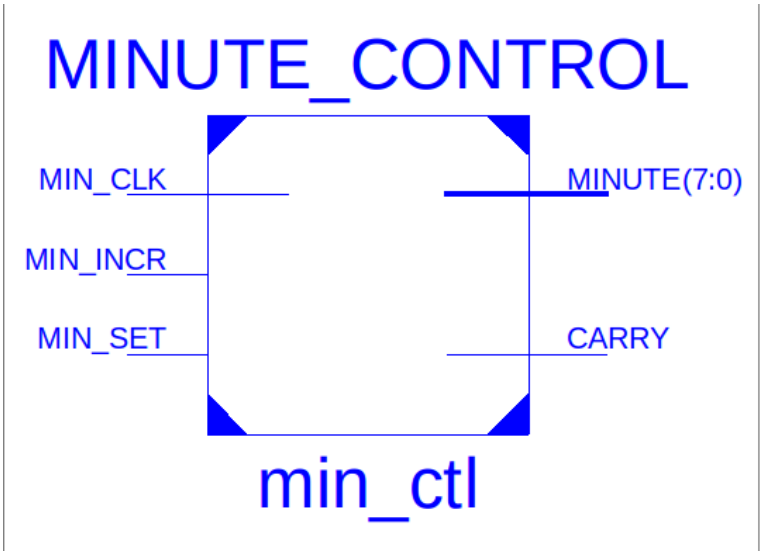
Đếm theo xung CLK (Đếm tự động, có CARRY)



Đếm theo cạnh lên INCR (điều chỉnh thủ công, không có CARRY)

3 - Khối đếm phút

Sơ đồ khối



Bảng mô tả trạng thái đếm

INPUT			OUTPUT		
MIN_ SET	MIN_ INCR	MIN_CLK	CARRY	MINUTE [7:4]	MINUTE [3:0]
0	x	x	0	0	0
0	x	↑	0	0	1
0	x	↑	0	0	2
0	x	↑	0	0	3
0	x	↑	0	…	…
0	x	↑	0	5	8
0	x	↑	0	5	9
0	x	↑	1	0	0
0	x	↑	0	0	1
0	x	↑	0	0	2
1	↑	x	0	0	3
1	↑	x	0	…	…
1	↑	x	0	5	8
1	↑	x	0	5	9
S	↑	x	0	0	0
1	↑	x	0	0	1
1	↑	x	0	0	2
1	↑	x	0	0	3
1	↑	x	0	…	…

Verilog HDL

MINUTE_CONTROL.v

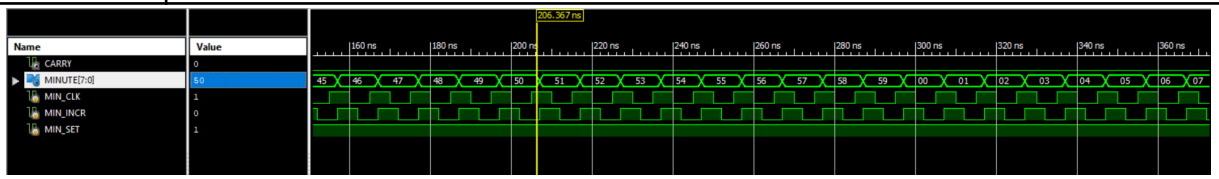
```
`timescale 1ns / 1ps

module MINUTE_CONTROL(
    input MIN_CLK, //  1 pos-edge = +1min
    input MIN_INCR,// 1 pos-edge = +1min
```

```
    input MIN_SET, // LOW: (Count CLK) -  HIGH: Set-up (Count
INCR)
    output reg CARRY,
    output reg[7:0] MINUTE // BCD display <MINUTE/10><MINUTE%10>
);
    initial begin
        MINUTE = 8'H30;
        CARRY = 0;
    end

    wire __triger;
    assign __triger = (MIN_CLK & ~MIN_SET) | (MIN_INCR &
MIN_SET);

    always @( posedge __triger ) begin
        // reset CARRY
        CARRY = 0;
        // do nothin'
        if(MINUTE[3:0] == 9) begin
            // reset x1
            MINUTE[3:0] = 0;
            if(MINUTE[7:4] == 5)begin
                // reset x10
                MINUTE[7:4] = 0;
                // reset\set CARRY flag
                if(MIN_SET == 0)
                    CARRY = 1;
                else
                    CARRY = 0;
            end
            else begin
                // inscrease x10
                MINUTE[7:4] = MINUTE[7:4] + 1;
            end
        end
        else begin
            // increase x1
            MINUTE[3:0] = MINUTE[3:0] + 1;
        end
    end
endmodule
```

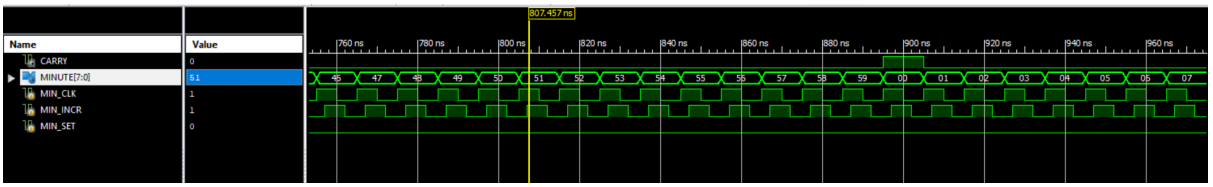
Testbench

TEST_MINUTE.v

```
`timescale 1ns / 1ps
`include "MINUTE_CONTROL.v"

module TEST_MINUTE;
    reg MIN_CLK;
    reg MIN_INCR;
    reg MIN_SET;
    wire CARRY;
    wire [7:0] MINUTE;
    MINUTE_CONTROL uut (
        .MIN_CLK(MIN_CLK),
        .MIN_INCR(MIN_INCR),
        .MIN_SET(MIN_SET),
        .CARRY(CARRY),
        .MINUTE(MINUTE));
    initial begin
        // Initialize Inputs
        MIN_CLK = 0;
        forever #5 MIN_CLK=~MIN_CLK;
    end
    initial begin
        MIN_INCR = 0;#2;
        forever #5 MIN_INCR=~MIN_INCR;
    end
    initial begin
        MIN_SET = 1;
        forever #400 MIN_SET=~MIN_SET;
    end
endmodule
```



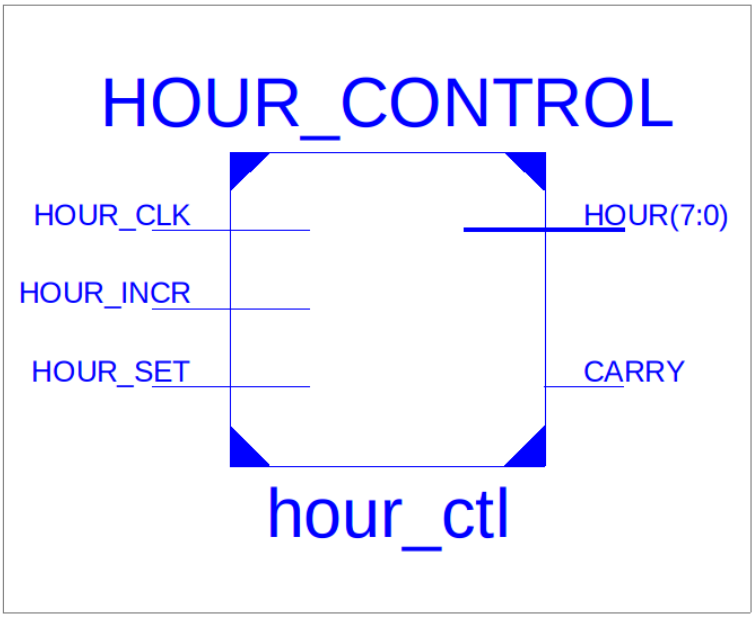
Đếm theo cạnh lên INCR (điều chỉnh thủ công, không có CARRY)



Đếm theo xung CLK (Đếm tự động, có CARRY)

4 - Khối đếm giờ

Sơ đồ khối



Bảng mô tả trạng thái đếm

INPUT			OUTPUT		
HOUR _SET	HOUR _INCR	HOUR_ CLK	CARRY	HOUR [7:4]	HOUR [3:0]
0	x	x	0	0	0
0	x	↑	0	0	1
0	x	↑	0	0	2
0	x	↑	0	0	3
0	x	↑	0	…	…
0	x	↑	0	2	2
0	x	↑	0	2	3
0	x	↑	1	0	0
0	x	↑	0	0	1
0	x	↑	0	0	2
1	↑	x	0	0	3
1	↑	x	0	…	…

1	↑	x	0	2	2
1	↑	x	0	2	3
S	↑	x	0	0	0
1	↑	x	0	0	1
1	↑	x	0	0	2
1	↑	x	0	0	3
1	↑	x	0	…	…

Verilog HDL

HOUR_CONTROL.v

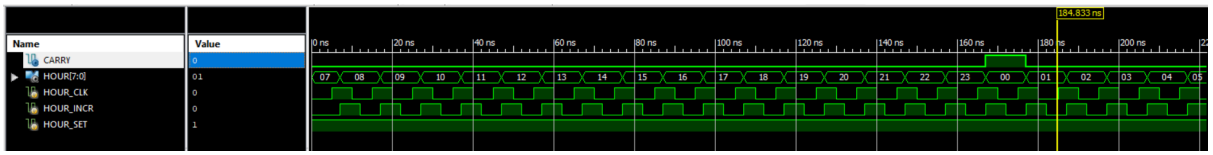
```
`timescale 1ns / 1ps

module HOUR_CONTROL(
    input HOUR_CLK, //  1 pos-edge = +1hour
    input HOUR_INCR,// 1 pos-edge = +1hour
    input HOUR_SET, // LOW: (Count CLK) -  HIGH: Set-up (Count INCR)
    output reg CARRY,
    output reg[7:0] HOUR // BCD display <HOUR/10><HOUR%10>
);
    initial begin
        HOUR = 8'H07;
        CARRY = 0;
    end

    wire __triger;
    assign __triger = (HOUR_CLK & ~HOUR_SET) | (HOUR_INCR & HOUR_SET);

    always @( posedge __triger ) begin
        // reset CARRY
        CARRY = 0;
        // do nothin'
        if(HOUR[3:0] == 4) begin
            // reset x1
            HOUR[3:0] = 0;
            if(HOUR[7:4] == 2)begin
                // reset x10
                HOUR[7:4] = 0;
                // reset\set CARRY flag
                if(HOUR_SET == 0)
                    CARRY = 1;
                else
```

```
                    CARRY = 0;
                end
                else begin
                    // increase x10
                    HOUR[7:4] = HOUR[7:4] + 1;
                end
            end
            else begin
               // increase x1
                HOUR[3:0] = HOUR[3:0] + 1;
            end
        end
    endmodule
```

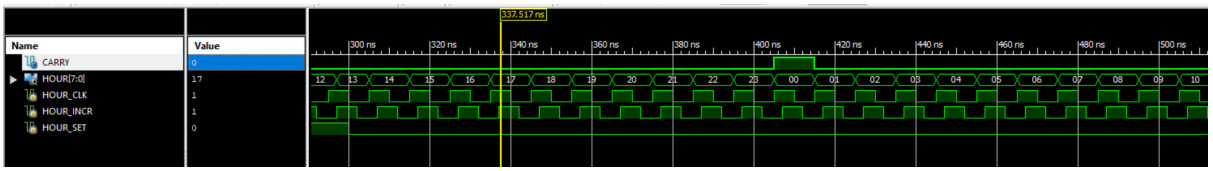
Testbench

TEST_HOUR.v

```
`timescale 1ns / 1ps
`include "HOUR_CONTROL.v"

module TEST_HOUR;
    reg HOUR_CLK;
    reg HOUR_INCR;
    reg HOUR_SET;
    wire CARRY;
    wire [7:0] HOUR;
    HOUR_CONTROL uut (
        .HOUR_CLK(HOUR_CLK),
        .HOUR_INCR(HOUR_INCR),
        .HOUR_SET(HOUR_SET),
        .CARRY(CARRY),
        .HOUR(HOUR));
    initial begin
        HOUR_CLK = 0;
        forever #5 HOUR_CLK=~HOUR_CLK;
    end
    initial begin
        HOUR_INCR = 0;#2;
        forever #5 HOUR_INCR=~HOUR_INCR;
    end
    initial begin
        HOUR_SET = 1;
        forever #300 HOUR_SET=~HOUR_SET;
    end
endmodule
```



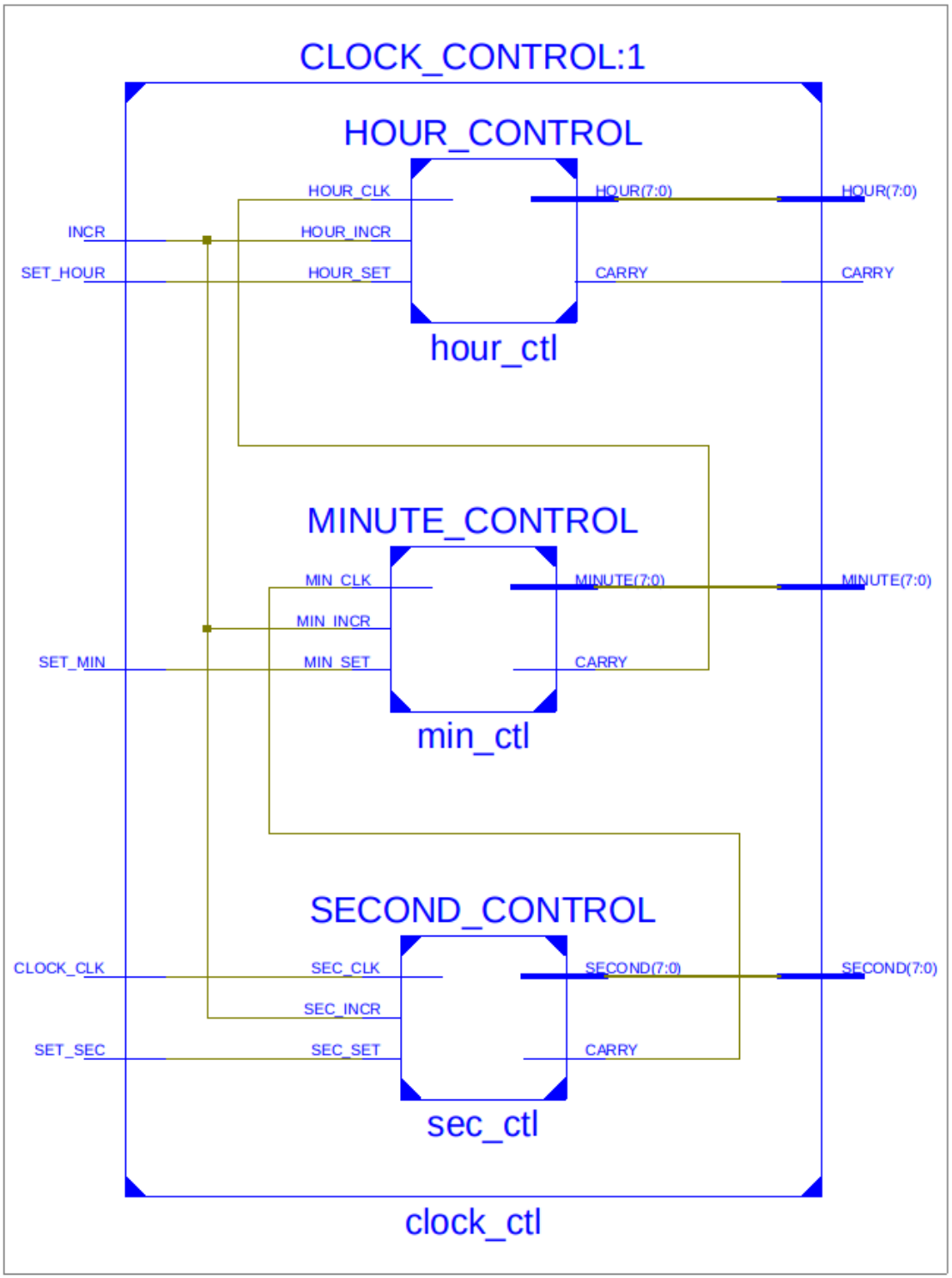
Đếm theo cạnh lệnh INCR (Điều chỉnh thủ công)



Đếm theo xung CLK (Đếm tự động)

5 - Khối đồng hồ 3 ngõ vào chọn chế độ (Giờ + Phút + Giây)

Sơ đồ khối



Bảng mô tả trạng thái đếm

SET_HOUR	SET_MIN	SET_SEC	HOUR	MINUTE	SECOND
0	0	0	Đếm/CLK	Đếm/CLK	Đếm/CLK
x	x	1	Không đổi	Không đổi	Đếm/INCR
x	1	0	Không đổi	Đếm/INCR	Đếm/CLK
1	0	0	Không đổi	Đếm/CLK	Đếm/CLK

Đếm/CLK:
+ Đếm lên MOD60, theo xung cạnh lên CLK với SECOND;
+ Đếm lên MOD60, theo xung cạnh lên CARRY của SECOND với MINUTE;
+ Đếm lên MOD60, theo xung cạnh lên CARRY của MINUTE với HOUR.

Đếm/INCR:
+ Đếm theo xung INCR.

Verilog HDL

CLOCK_CONTROL.v

```
`timescale 1ns / 1ps
`include "SECOND_CONTROL.v"
`include "MINUTE_CONTROL.v"
`include "HOUR_CONTROL.v"

module CLOCK_CONTROL(
    input CLOCK_CLK,
    input SET_SEC,
    input SET_MIN,
    input SET_HOUR,
    input INCR,
    output [7:0] SECOND,
    output [7:0] MINUTE,
    output [7:0] HOUR,
    output CARRY
);
    wire _sec_carry, _min_carry;

    SECOND_CONTROL sec_ctl(
        .SEC_CLK(CLOCK_CLK),
        .SEC_INCR(INCR),
        .SEC_SET(SET_SEC),
        .CARRY(_sec_carry),
        .SECOND(SECOND)
    );
    MINUTE_CONTROL min_ctl(
        .MIN_CLK(_sec_carry),
        .MIN_INCR(INCR),
```

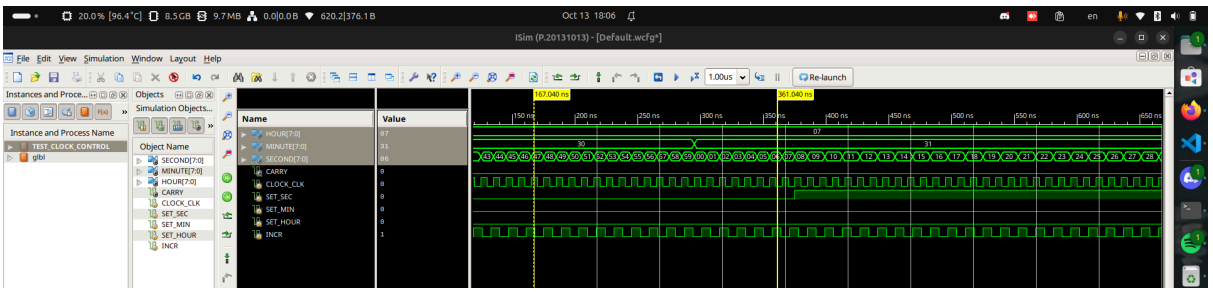
```
        .MIN_SET(SET_MIN),
        .CARRY(_min_carry),
        .MINUTE(MINUTE)
    );
    HOUR_CONTROL hour_ctl(
        .HOUR_CLK(_min_carry),
        .HOUR_INCR(INCR),
        .HOUR_SET(SET_HOUR),
        .HOUR(HOUR),
        .CARRY(CARRY)
    );

endmodule
```

Testbench

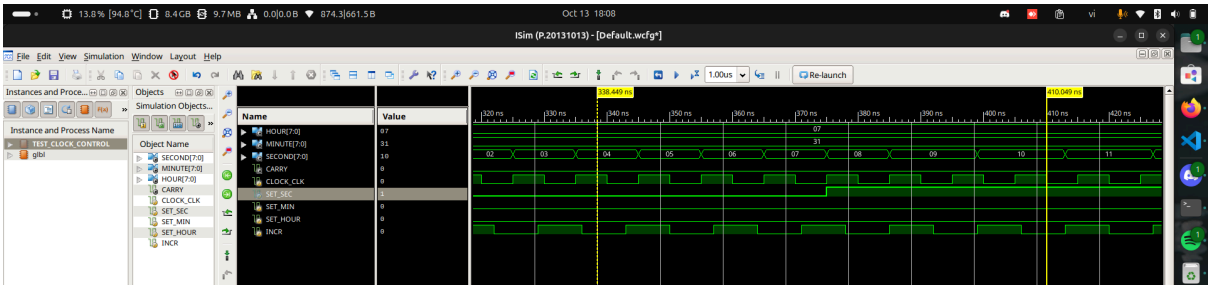
TEST_CLOCK_CONTROL.v

```
`timescale 1ns / 1ps
`include "CLOCK.v"
module TEST_CLOCK_CONTROL;

    // Inputs
    reg CLOCK_CLK;
    reg SET_SEC;
    reg SET_MIN;
    reg SET_HOUR;
    reg INCR;

    // Outputs
    wire [7:0] SECOND;
    wire [7:0] MINUTE;
    wire [7:0] HOUR;
    wire CARRY;

    // Instantiate the Unit Under Test (UUT)
    CLOCK_CONTROL uut (
        .CLOCK_CLK(CLOCK_CLK),
        .SET_SEC(SET_SEC),
        .SET_MIN(SET_MIN),
        .SET_HOUR(SET_HOUR),
        .INCR(INCR),
        .SECOND(SECOND),
        .MINUTE(MINUTE),
        .HOUR(HOUR),
        .CARRY(CARRY)
    );
```

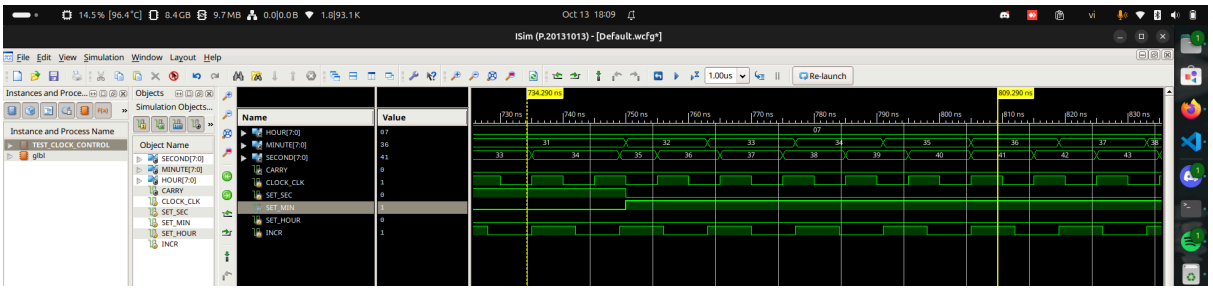
```
    initial begin
        // Initialize Inputs
        CLOCK_CLK = 0;
        SET_SEC = 0;        SET_MIN = 0;        SET_HOUR = 0;
        INCR = 0;
        // Add stimulus here
        SET_SEC = 0;        SET_MIN = 0;        SET_HOUR = 0;
        #375;
        SET_SEC = 1;        SET_MIN = 0;        SET_HOUR = 0;
        #375;
        SET_SEC = 0;        SET_MIN = 1;        SET_HOUR = 0;
        #375;
        SET_SEC = 0;        SET_MIN = 0;        SET_HOUR = 1;
        #375;
    end

    initial begin
        forever begin
            #5 CLOCK_CLK = ~ CLOCK_CLK;
        end
    end

    initial begin
        forever begin
            #7 INCR = ~ INCR;
        end
    end

endmodule
```



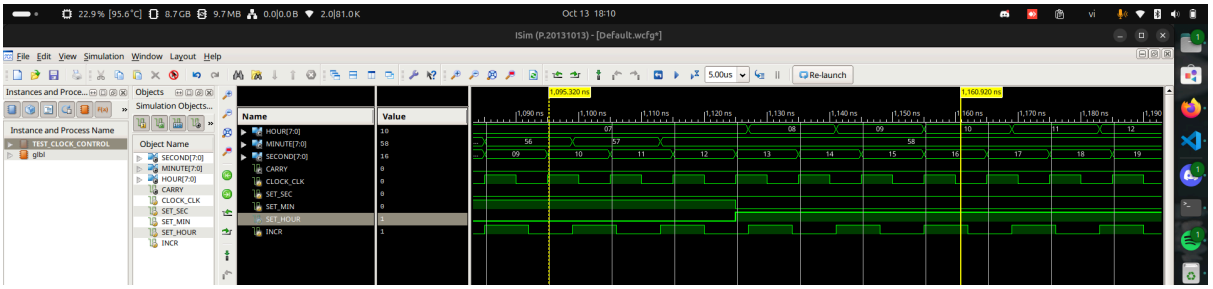
Đếm theo cạnh lên của CLK (Chế độ bình thường)



Đếm theo cạnh lên của INCR (Chế độ điều chỉnh giây)



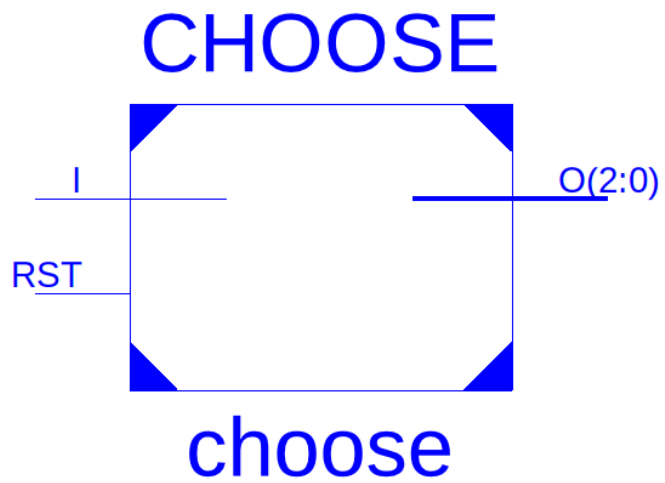
Đếm theo cạnh lên của INCR (Chế độ điều chỉnh phút)



Đếm theo cạnh lên của INCR (Chế độ điều chỉnh giờ)

6 - Khối chọn chế độ chỉnh sửa

Sơ đồ khối



Bảng mô tả trạng thái đếm

I	RST	O[2:0]
x	1	000
↑	0	001
↑	0	010
↑	0	100
↑	0	000

Verilog HDL

CHOOSE.v

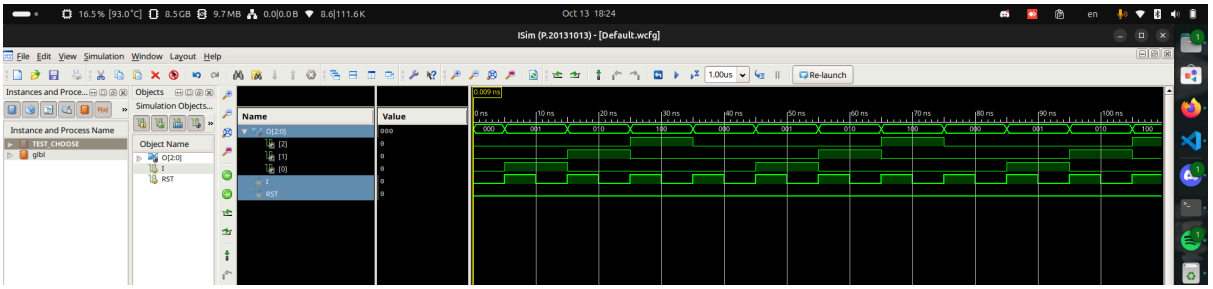
```
`timescale 1ns / 1ps

module CHOOSE(
   input I, // 1 pos-edge = shift-left x1
   input RST,
   output reg [2:0] O // Initial state O = 0;
);
   initial begin
       O = 0;
   end
   always @(posedge I) begin
       if(RST == 1) begin
           O = 0;
       end
       else begin
           if(O == 0)
               O = 1;
           else
               O = O << 1;
       end
   end

endmodule
```

Testbench

TEST_CHOOSE.v

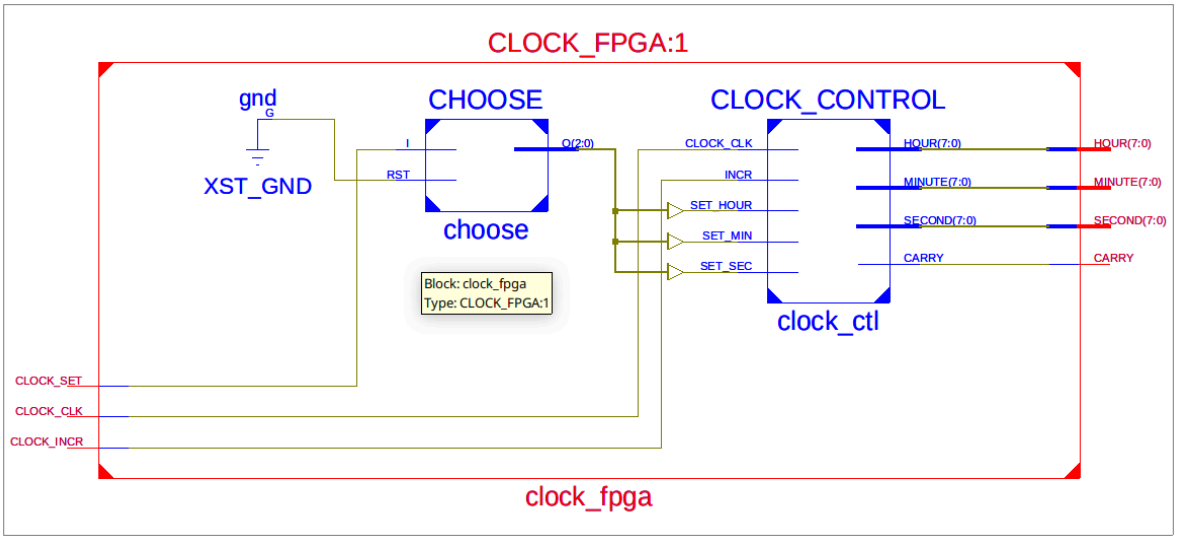
```
`timescale 1ns / 1ps
`include "CHOOSE.v"
module TEST_CHOOSE;
```

```
    // Inputs
    reg I;
    reg RST;

    // Outputs
    wire [2:0] O;

    CHOOSE uut (
        .I(I),
        .RST(RST),
        .O(O)
    );

    initial begin
        I = 0;
        RST = 0;
        #10;
    end

    initial begin
        forever #5 I = ~I;
    end
endmodule
```



7 - Khối đồng hồ một ngõ vào chọn chế độ

Sơ đồ khối



Bảng trạng thái

CLOCK _SET	CLOCK _INCR	CLOCK _CLK	HOUR [7:0]	MINUTE [7:0]	SECOND [7:0]	CARRY
0	x	↑	UP/CLK	UP/CLK	UP/CLK	0 (1 - khi HOUR đếm từ 23->0)
↑	↑	x	NC	NC	UP/INCR	0
↑	↑	x	NC	UP/INCR	UP/CLK	0
↑	↑	x	UP/INCR	UP/CLK	UP/CLK	
↑	x	↑	UP/CLK	UP/CLK	UP/CLK	0 (1 - khi HOUR đếm từ 23->0)

(Điều khiện: thứ tự chuyển trạng thái từ cột đầu tiên đến cột thứ ba)
UP/CLK:
+ Đếm lên MOD60, theo xung cạnh lên CLK với SECOND;
+ Đếm lên MOD60, theo xung cạnh lên CARRY của SECOND với MINUTE;
+ Đếm lên MOD60, theo xung cạnh lên CARRY của MINUTE với HOUR.
UP/INCR:
+ Đếm theo xung INCR.
NC:
+ Không đổi trạng thái.

Verilog HDL

CLOCK_FPGA.v

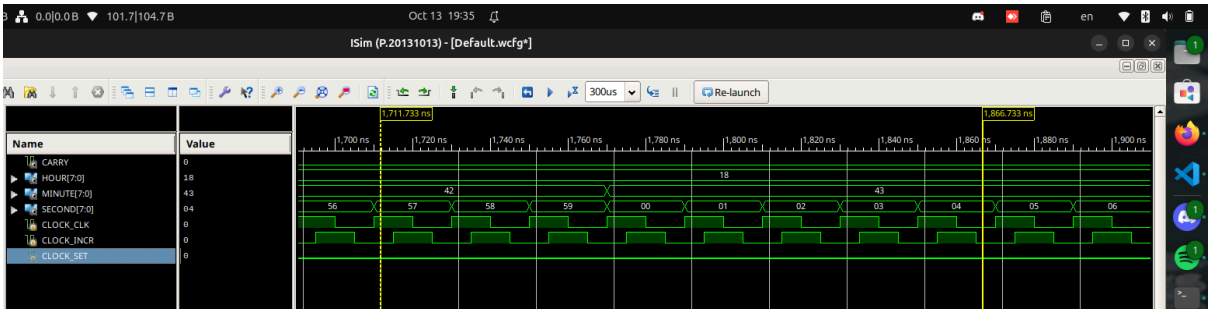
```
`timescale 1ns / 1ps
`include "CLOCK.v"
`include "CHOOSE.v"


module CLOCK_FPGA(
   input CLOCK_CLK, CLOCK_SET, CLOCK_INCR,
    output CARRY,
   output [7:0] MINUTE, SECOND, HOUR
);
   wire [2:0] _set;

   CHOOSE choose(.I(CLOCK_SET), .RST(GND), .O(_set));

   CLOCK_CONTROL clock_ctl(
        .CLOCK_CLK(CLOCK_CLK),
        .SET_SEC(_set[0]),
        .SET_MIN(_set[1]),
        .SET_HOUR(_set[2]),
        .INCR(CLOCK_INCR),
        .SECOND(SECOND),
        .MINUTE(MINUTE),
        .HOUR(HOUR),
          .CARRY(CARRY)
   );

endmodule
```

Testbench

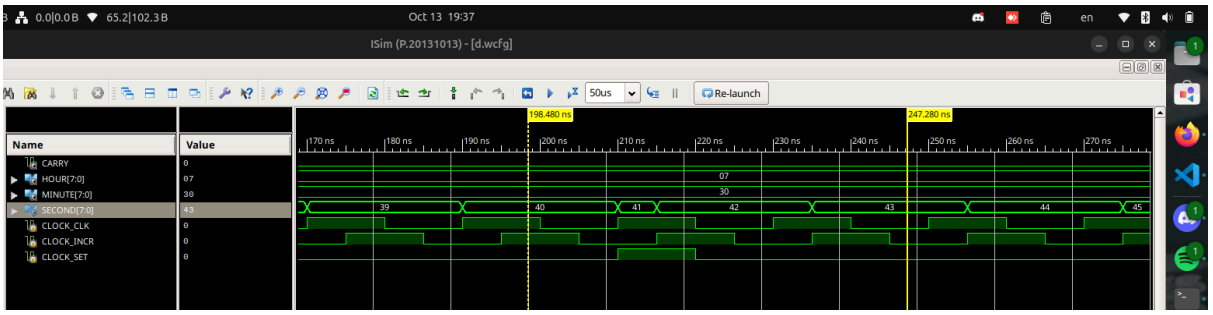
TEST_CLOCK_FPGA.v

```
`timescale 1ns / 1ps
`include "CLOCK_FPGA.v"
module TEST_CLOCK_FPGA;

   // Inputs
   reg CLOCK_CLK;
   reg CLOCK_SET;
   reg CLOCK_INCR;

   // Outputs
   wire CARRY;
   wire [7:0] MINUTE;
   wire [7:0] SECOND;
```

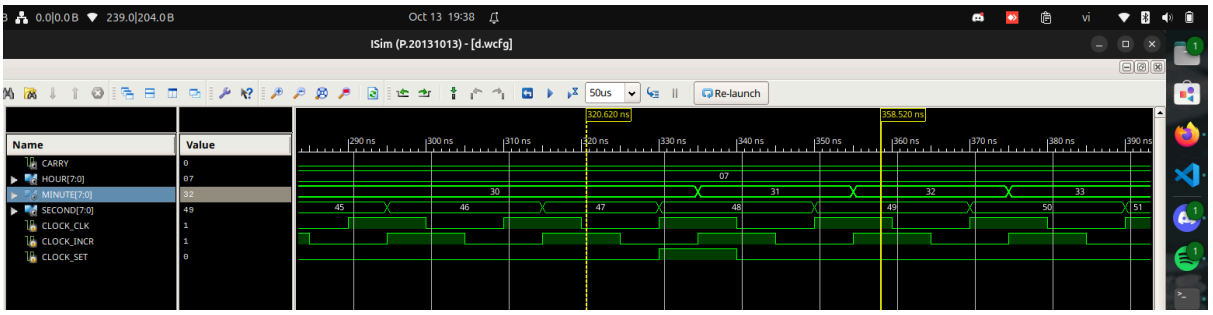
```
    wire [7:0] HOUR;

    // Instantiate the Unit Under Test (UUT)
    CLOCK_FPGA uut (
        .CLOCK_CLK(CLOCK_CLK),
        .CLOCK_SET(CLOCK_SET),
        .CLOCK_INCR(CLOCK_INCR),
        .CARRY(CARRY),
        .MINUTE(MINUTE),
        .SECOND(SECOND),
        .HOUR(HOUR)
    );

    initial begin
        // Initialize Inputs
        CLOCK_CLK = 0;
        CLOCK_SET = 0;
        CLOCK_INCR = 0;

        // Wait 100 ns for global reset to finish
        #200;
        #10 CLOCK_SET = 1; #10 CLOCK_SET = 0;
        #100;
        #10 CLOCK_SET = 1; #10 CLOCK_SET = 0;
        #200;
        #10 CLOCK_SET = 1; #10 CLOCK_SET = 0;
        #200;
        #10 CLOCK_SET = 1; #10 CLOCK_SET = 0;
    end

    initial begin
        #5
        forever #10 CLOCK_INCR = ~CLOCK_INCR;
    end

    initial begin
        forever #10 CLOCK_CLK = ~CLOCK_CLK;
    end
endmodule
```



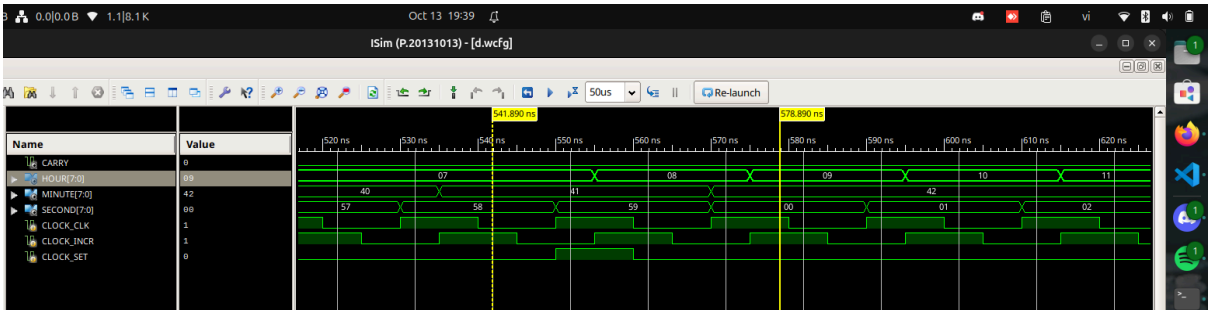
Chế độ đếm bình thường, theo xung CLK.



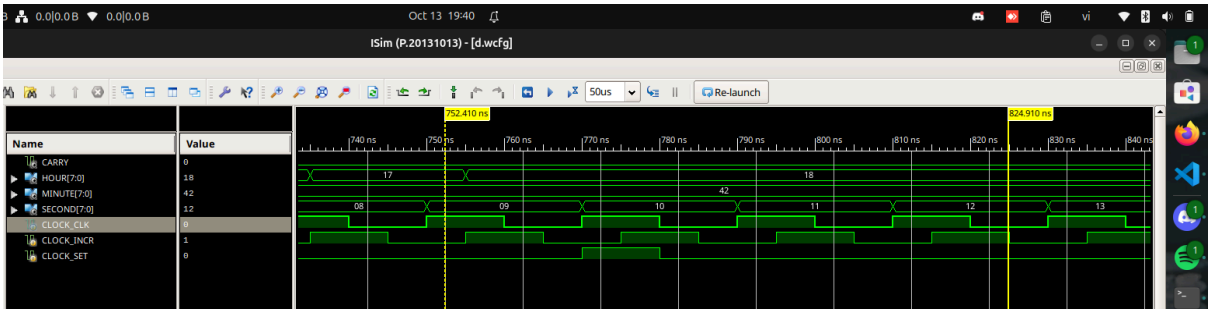
Chế độ điều chỉnh giây, theo xung INCR (Xung pos-edge CLOCK_SET thứ nhất)



Chế độ chỉnh phút, theo xung INCR (Xung pos-edge CLOCK_SET thứ hai)



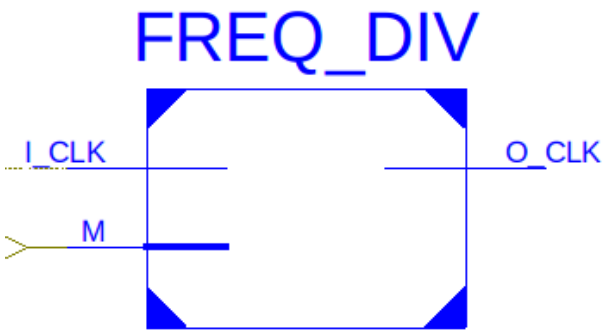
Chế độc chỉnh giờ, theo xung INCR (Xung pos-edge CLOCK_SET thứ ba)



Hoạt động ở chế độ bình thường theo xung CLK (Xung pos-edge CLOCK_SET thứ tư)

8 - Khối chia tần số

Sơ đồ khối



Verilog HDL

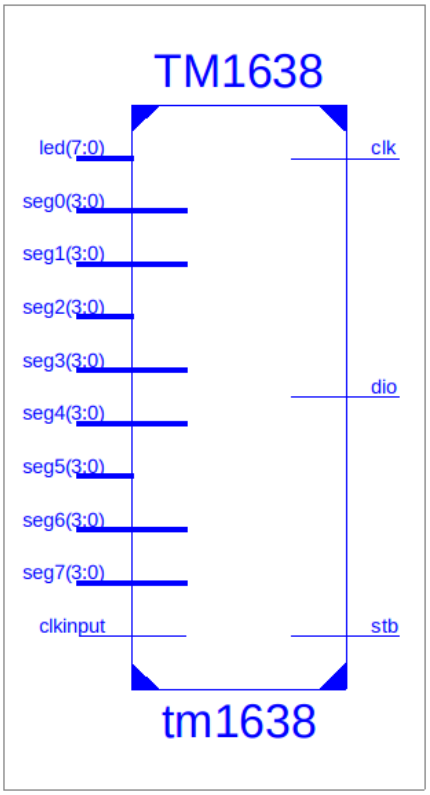
FREQ_DIV.v

```
`timescale 1ns / 1ps
`include "FREQ_DIV.v"
module FREQ_DIV(
    input I_CLK,
    input M,
    output reg O_CLK
);
     reg [31:0] COUNT;
     initial begin
        COUNT = 0;
        O_CLK = 0;
     end
     always @(negedge I_CLK) begin
         if( M == 1 ) begin
                 if( COUNT == 12_000_000 ) begin
                     O_CLK = ~O_CLK;
                     COUNT = 0;
```

```
                end else
                    COUNT = COUNT + 1;
        end
        else begin
                if( COUNT > 10000 )
                    COUNT = 0;
                if( COUNT == 1000) begin
                    O_CLK = ~O_CLK;
                    COUNT = 0;
                end else
                    COUNT = COUNT + 1;
        end
    end
endmodule
```

9 - Khối giao tiếp bảng mạch mở rộng TM1638

Sơ đồ khối



Verilog HDL

TM1638.v

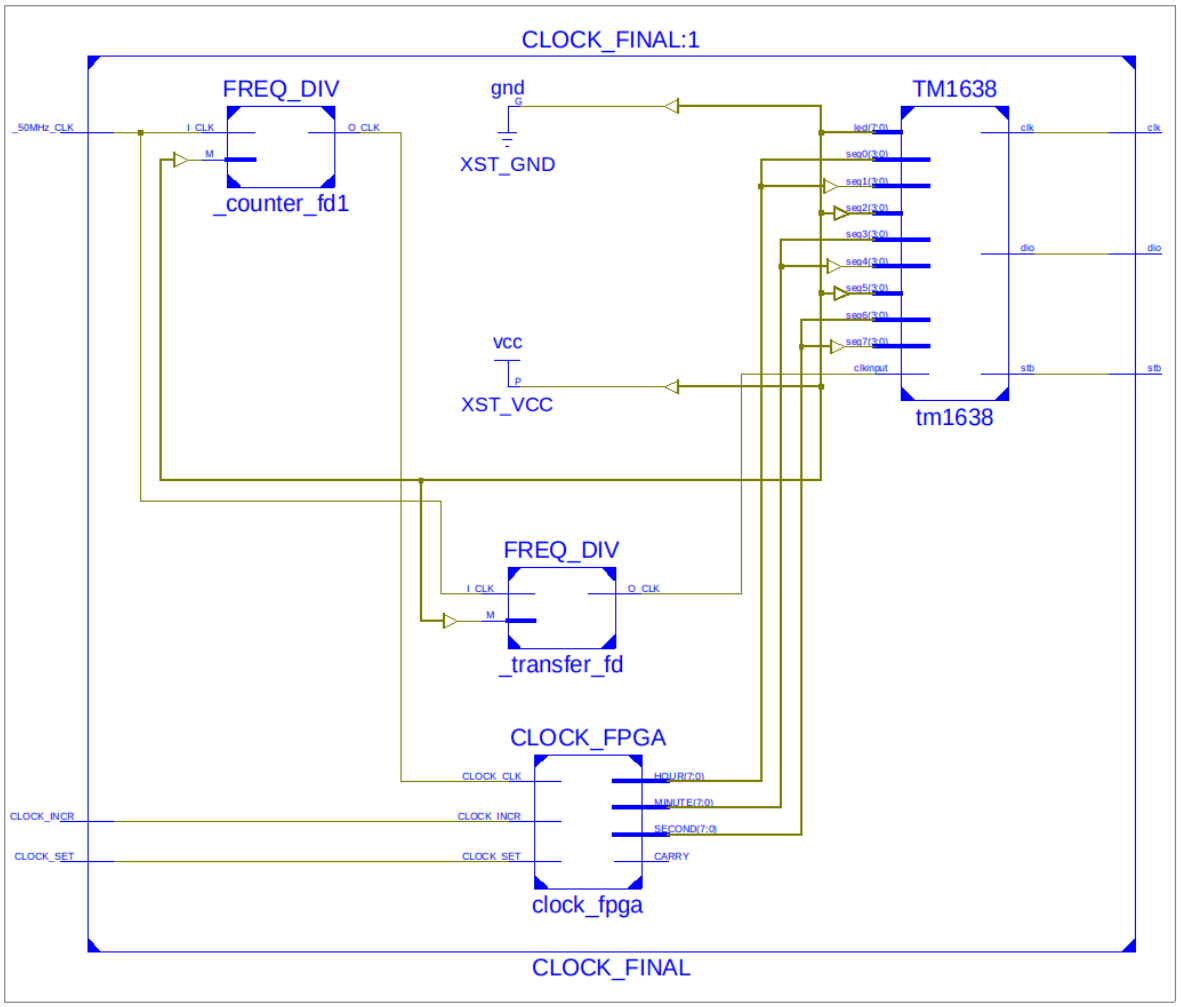
```
/* TM1638 driver
Author: Mr. Son
input clock: tested for 200KHz ( 50H - free counter ->
bit 5
*/
module TM1638(
    input wire [7:0] led , // 8 leds
    input wire [3:0]
seg7,seg6,seg5,seg4,seg3,seg2,seg1,seg0 ,//4 bit data for
cathode commond LED
    input clkinput,
    output reg clk,
    output reg stb,
    output reg dio
);
    /* Hex-Digit to seven segment LED decoder
    Author: Mr. Son
    */
    function [7:0] sseg;
        input [3:0] hex;
        begin
            case (hex)
                4'h0: sseg[7:0] = 8'b0111111;
                4'h1: sseg[7:0] = 8'b0000110;
                4'h2: sseg[7:0] = 8'b1011011;
                4'h3: sseg[7:0] = 8'b1001111;
                4'h4: sseg[7:0] = 8'b1100110;
                4'h5: sseg[7:0] = 8'b1101101;
                4'h6: sseg[7:0] = 8'b1111101;
                4'h7: sseg[7:0] = 8'b0000111;
                4'h8: sseg[7:0] = 8'b1111111;
                4'h9: sseg[7:0] = 8'b1101111;
                4'hA: sseg[7:0] = 8'b1110111;
                4'hB: sseg[7:0] = 8'b1111100;
                4'hC: sseg[7:0] = 8'b1011000;
                4'hD: sseg[7:0] = 8'b1011110;
                4'hE: sseg[7:0] = 8'h40; // dau tru
                default : sseg[7:0] = 8'b0000000; // 4'hF
            endcase
        end
    endfunction
```

```
    integer cs = 0;
    integer i ;
    reg [7:0] command1 =8'h40, command2 =8'hC0,command3
=8'h8F;
    wire [127:0] leddata; // 1,3,5,7,9,11,13,15: single
led; 0,2,4,6,8,10,12,14: seg LED (common cathode)
    reg [127:0] leddatahold;
    assign leddata[0*8+7:0*8+0] = sseg(seg0);
    assign leddata[2*8+7:2*8+0] = sseg(seg1);
    assign leddata[4*8+7:4*8+0] = sseg(seg2);
    assign leddata[6*8+7:6*8+0] = sseg(seg3);
    assign leddata[8*8+7:8*8+0] = sseg(seg4);
    assign leddata[10*8+7:10*8+0] = sseg(seg5);
    assign leddata[12*8+7:12*8+0] = sseg(seg6);
    assign leddata[14*8+7:14*8+0] = sseg(seg7);
    assign leddata[1*8+7:1*8+0] = led[0] ;
    assign leddata[3*8+7:3*8+0] = led[1] ;
    assign leddata[5*8+7:5*8+0] = led[2] ;
    assign leddata[7*8+7:7*8+0] = led[3] ;
    assign leddata[9*8+7:9*8+0] = led[4] ;
    assign leddata[11*8+7:11*8+0] = led[5] ;
    assign leddata[13*8+7:13*8+0] = led[6] ;
    assign leddata[15*8+7:15*8+0] = led[7] ;
    initial begin
        clk = 1 ;
        stb = 1 ;
        dio = 0 ;
    end
    always @(posedge clkinput)
    begin
        if (cs==0)begin
            stb = 0; // initial tm1638
            command1 =8'h40; command2 =8'hC0;command3
=8'h8F;
            leddatahold=leddata ;
        end
        else if ((cs >=1)&&(cs<=16))
            begin
                dio = command1[0];
                clk = ~clk ;
            if (clk) command1=command1>>1 ;
        end
```

```
        else if (cs==17)
            stb = 1; // stop tm1638

        else if (cs==18)
            stb = 0; // ready to send the second command
        // send second command
        else if ((cs >=19)&&(cs<=34))
            begin
                dio = command2[0];
                clk = ~clk ;
                if (clk) command2=command2>>1 ;
            end

        else if ((cs >=35)&&(cs<=290))
        begin
                dio = leddatahold[0];
                clk = ~clk ;
                if (clk) leddatahold=leddatahold>>1 ;
        end

        else if (cs==291)
            stb = 1; // stop tm1638 for end of data

        else if (cs==292)
            stb = 0; // ready to send the third command
        // send last command
        else if ((cs >=293)&&(cs<=308))
            begin
                dio = command3[0];
                clk = ~clk ;
                if (clk) command3=command3>>1 ;
            end
        else if (cs==309)
            stb = 1; // End
        else if (cs==310)
            cs = -1 ; //repeat
        // update cs
        cs=cs+1;
    end
endmodule
```

10- Khối tổng quát (Top-model / level 0)

Sơ đồ khối



Verilog HDL

CLOCK_FINAL.v	`timescale 1ps / 1ps `include "CLOCK_FPGA.v" `include "FREQ_DIV.v" `include "TM1638.v" module CLOCK_FINAL(input _50MHz_CLK, CLOCK_SET, CLOCK_INCR, output clk, output stb, output dio); wire VCC, GND; wire _transfer_freq_0;

```
    wire _counter_freq_1;

    wire [3:0] LED7SEG_0, LED7SEG_1, LED7SEG_2,
               LED7SEG_3, LED7SEG_4, LED7SEG_5,
               LED7SEG_6, LED7SEG_7;
    wire [7:0] LED;
    wire [7:0] __minute, __second, __hour;

    FREQ_DIV
_transfer_fd(.I_CLK(_50MHz_CLK),.M(GND),.O_CLK(_transfer_freq
_0));
    FREQ_DIV
_counter_fd1(.I_CLK(_50MHz_CLK),.M(VCC),.O_CLK(_counter_freq_
1));

    TM1638  tm1638(
        .led(LED),
        .seg0(LED7SEG_0),
        .seg1(LED7SEG_1),
        .seg2(LED7SEG_2),
        .seg3(LED7SEG_3),
        .seg4(LED7SEG_4),
        .seg5(LED7SEG_5),
        .seg6(LED7SEG_6),
        .seg7(LED7SEG_7),
        .clkinput(_transfer_freq_0),
        .clk(clk),
        .stb(stb),
        .dio(dio)
    );

    CLOCK_FPGA clock_fpga(
        .CLOCK_CLK(_counter_freq_1),
        .CLOCK_SET(CLOCK_SET),
        .CLOCK_INCR(CLOCK_INCR),
        .SECOND(__second),
        .MINUTE(__minute),
        .HOUR(__hour)
    );

    assign LED = 8'B1010_1010;
    assign LED7SEG_0 = __hour[7:4] ;
    assign LED7SEG_1 = __hour[3:0] ;
    assign LED7SEG_2 = 4'H0E;
    assign LED7SEG_3 = __minute[7:4];
    assign LED7SEG_4 = __minute[3:0];
    assign LED7SEG_5 = 4'h0E;
    assign LED7SEG_6 = __second[7:4];
    assign LED7SEG_7 = __second[3:0];
```

```
   assign VCC = 1;
   assign GND = 0;
endmodule
```