به نام خدا

آزمایشگاه طراحی سیستم‌های دیجیتال

گزارش کار سوم

توصیف جریان داده

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

بهار ۱۴۰۱

استاد:

علیرضا اجلالی

دستیار آموزشی:

سحر رضاقلی

گروه:

هیربد بهنام

۹۹۱۷۱۳۳۳

علی نظری

۹۹۱۰۲۴۰۱

عرفان مجیبی

۹۹۱۰۵۷۰۷

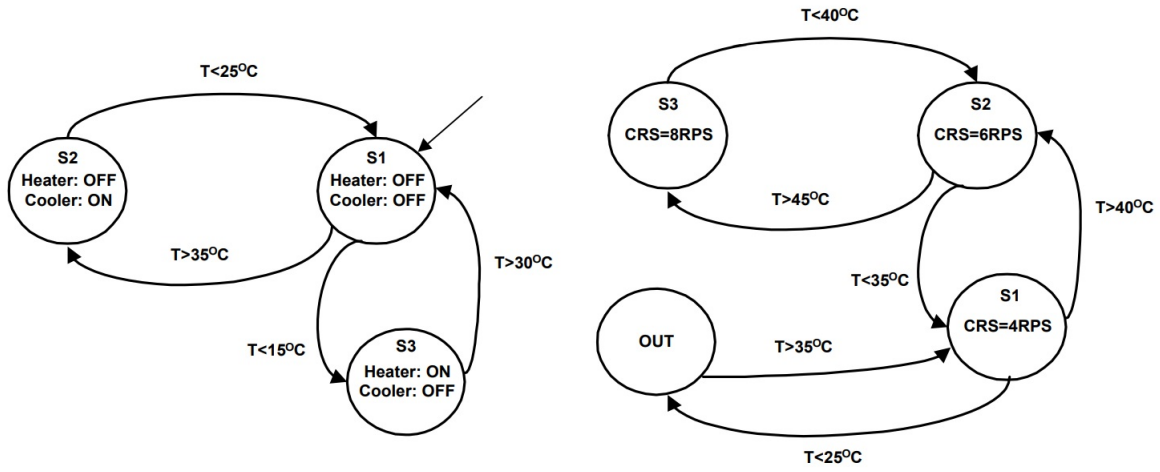
فهرست

مقدمه

در این آزمایش می‌خواهیم واحد کنترلی یک انکوباتور incubator طراحی کنیم که به یک فن و گرم و کننده با توجه به دما دستور دهد که روشن یا خاموش شوند. در صورتی که فن روشن بود باید دورموتور آن را نیز تنظیم کنیم.

سنسور حرارتی هر دقیقه دما را از محیط دریافت می‌کند ولی مدار در لبه‌ی بالارونده‌ی کلاک به فن و گرم‌کننده فرمان‌های مورد نیاز را می‌فرستد.

حالت‌های مختلف سرد کننده و گرم کننده و سرعت فن را می‌توانید در نمودار زیر مشاهده کنید:



گزارش آزمایش

بخش ۱. طراحی یک انکوباتور

کد این آزمایش در زیر آمده است:

```
module Incubator (
    input clk,
    input reset,
    input signed [7:0] T,
    output heater_on,
    output cooler_on,
    output [3:0] cooler_rps
);
    reg [1:0] main_state;
    reg [1:0] cooler_rps_state;
    reg [1:0] next_main_state;
    reg [1:0] next_cooler_rps_state;
    // Just like the chart in lab page
    assign cooler_on = (main_state == 2);
    assign heater_on = (main_state == 3);
    // Cooler rps is zero when it's off
    assign cooler_rps = cooler_rps_state == 0 ? 4'd0 : (cooler_rps_state
    == 1 ? 4'd4 : (cooler_rps_state == 2 ? 4'd6 : 4'd8));
    // Watch for temperature change
    always @(T or main_state or cooler_rps_state) begin
        next_main_state <= main_state;
        // Check main state to decide the next state
        if (main_state == 1) begin
            if (T < 15)
                next_main_state <= 2'd3;
            else if (T > 35)
                next_main_state <= 2'd2;
        end else if (main_state == 2) begin
            if (T < 25)
                next_main_state <= 2'd1;
        end else if (main_state == 3) begin
            if (T > 30)
                next_main_state <= 2'd1;
        end
        // Check the fan state
        next_cooler_rps_state <= cooler_rps_state;
        // Nothing to do if cooler is not on
```

```
        if (next_main_state != 2) begin
            next_cooler_rps_state <= 0; // zero is out
        end else begin
            if (cooler_rps_state == 0) begin
                if (T > 35)
                    next_cooler_rps_state <= 2'd1;
            end else if (cooler_rps_state == 1) begin
                if (T > 40)
                    next_cooler_rps_state <= 2'd2;
                else if (T < 25)
                    next_cooler_rps_state <= 2'd0;
            end else if (cooler_rps_state == 2) begin
                if (T > 45)
                    next_cooler_rps_state <= 2'd3;
                else if (T < 35)
                    next_cooler_rps_state <= 2'd1;
            end else if (cooler_rps_state == 3) begin
                if (T < 40)
                    next_cooler_rps_state <= 2'd2;
            end
        end
    end
    // Wait for clock to assign the states
    always @(posedge clk) begin
        if (reset) begin
            main_state <= 2'b1;
            cooler_rps_state <= 2'b0;
        end else begin
            main_state <= next_main_state;
            cooler_rps_state <= next_cooler_rps_state;
        end
    end
endmodule
```

حال به شرح کد می‌پردازیم. در ابتدا ماژول Incubator را با ورودی و خروجی‌های زیر تعریف می‌کنیم:

- **clk:** کلاک مدار. در هر کلاک state مدار مشخص می‌شود.
- **reset:** در صورتی که ۱ باشد مدار ریست می‌شود. در ابتدا باید مدار را قبل از کار ریست کرد.
- **T:** دما به صورت یک عدد ۸ بیتی مکمل دو علامت‌دار
- **heater_on:** در صورتی که این سیگنال یک باشد باید گرم‌کن را روشن کرد.
- **cooler_on:** در صورتی که این سیگنال یک باشد باید فن خنک‌کن را روشن کرد.

- **cooler_rps:** عددی که این خروجی نشان می‌دهد دور موتور خنک کن است. دقت کنید که زمانی که فن خاموش است این عدد برابر ۰ است. همچنین از آن‌جا که حداکثر عدد حاصل ۸ است، این عدد ۴ بیتی در نظر گرفته شده است.

حال باید به روشی حالت در حال حاضر مدار را ذخیره کنیم. برای این کار از متغیر main_state استفاده شده است که عدد ذخیره شده در آن دقیقا حالت مدار در نمودار کشیده شده در دستور آزمایش را نشان می‌دهد. این عدد بین ۱ تا ۳ است پس به دو بیت برای ذخیره سازی آن نیاز داریم. دقت کنید زمانی که مدار را ریست می‌کنیم مقدار این متغیر را برابر ۱ قرار می‌دهیم. همچنین متغیر دیگه‌ای داریم به اسم next_main_state این متغیر نشان می‌دهد در کلاک بعدی باید مقدار main_state چه باشد. این مقدار را در زمانی که دما عوض می‌شود و یا main_state عوض می‌شود دوباره حساب کرد که در ادامه به صورت مفصل توضیح داده می‌شود.

همچنین دو متغیر دیگر cooler_rps_state و next_cooler_rps_state نیز وجود دارند که کار مشابه main_state و next_main_state را انجام می‌دهد ولی برای نمودار مربوط به سرعت فن.

حال بلاک‌های `always` را بررسی می‌کنیم. اولین بلاک همان بلاکی است که قرار است استیت‌های بعدی را تعیین کنیم. این بلاک را زمانی که دما عوض می‌شود یا زمانی که main_state یا cooler_rps_state عوض می‌شود اجرا می‌کنیم. دلیل اینکه روی main_state یا cooler_rps_state نیز اجرا می‌کنیم این است که تغییر این متغیرها به منزله‌ی کلاک است. فقط دقت کنید که بعد از کلاک این متغیرها عوض می‌شود پس این بلاک بعد از کلاک اجرا می‌شود. در ابتدای این بلاک وضعیت حالت بعدی main_state را با توجه به حالت الان مشخص می‌کنیم. این کار را صرفا با حالت بندی روی حالت فعلی و دما انجام می‌دهیم. دقت کنید که قبل از شروع حالت بندی حالت بعدی را مساوی حالت حال قرار می‌دهیم چرا که ممکن است که اصلا لازم به تغییر حالت بعدی نباشد! در ادامه برای تعیین حالت سرعت فن نیز در ابتدا مقدار فعلی فن را برابر با مقدار فن در کلاک بعدی قرار می‌دهیم برای زمانی که نیازی به تغییر نداشته باشیم. حال در صورتی که حالت بعدی، حالتی بود که نیازی به فن نبود، حالت بعدی فن را بر روی ۰ قرار می‌دهیم. (قرار داد می‌کنیم که ۰ همان out است.) در غیر این صورت با توجه به نمودار و حالت فعلی فن حالت بعدی را مشخص می‌کنیم.

تست‌بنچ

حال برای ماژول مذکور تست کیس می‌نویسیم. این تست کیس‌ها را در ابتدا به صورت هاردکد شده در کد قرار می‌دهیم و در ادامه به صورت رندوم درجه حرارت را تولید می‌کنیم. دقت کنید که بین زمان تغییر درجه حرارت اجازه می‌دهیم که بیش از یک کلاک مدار بخورد چرا که اگر درجه حرارت به عنوان مثال از ۰ به ۶۰ تغییر ناگهانی کند، مدار به بیش از یک کلاک نیاز دارد که state خود را به state درست تغییر دهد (در ابتدا به حالت همه چی خاموش می‌رود و سپس به حالت فن روشن می‌رود و سپس دور موتور تنظیم می‌شود.)

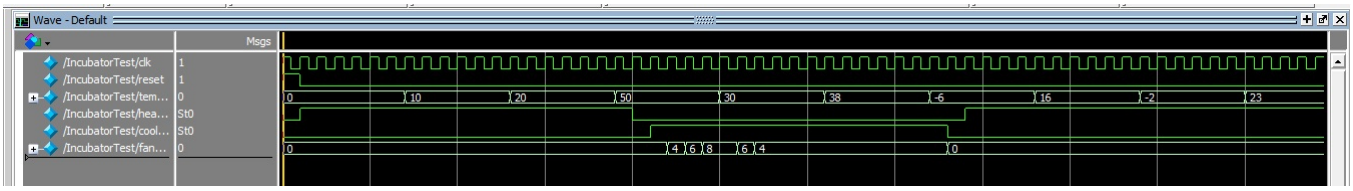
تست کیس ما در زیر آمده است:

```
`include "incubator.v"
module IncubatorTest;
    reg clk;
    reg reset;
    reg signed [7:0] temperature;
    wire heater_on;
    wire cooler_on;
    wire [3:0] fan_speed;
    integer i;

    Incubator incubator(clk, reset, temperature, heater_on, cooler_on,
   fan_speed);
    initial clk = 1'b1;
    always #5 clk= ~clk;
```

```
    initial begin
        reset = 1'b1;
        temperature = 0;
        #10;
        // Simple tests
        reset = 1'b0;
        temperature = 0;
        #60;
        temperature = 10;
        #60;
        temperature = 20;
        #60;
        temperature = 50;
        #60;
        temperature = 30;
        #60;
        for (i = 0; i < 10; i = i + 1) begin
            temperature = ($urandom % 70) - 10;
            #60;
        end
        $stop;
    end
endmodule
```

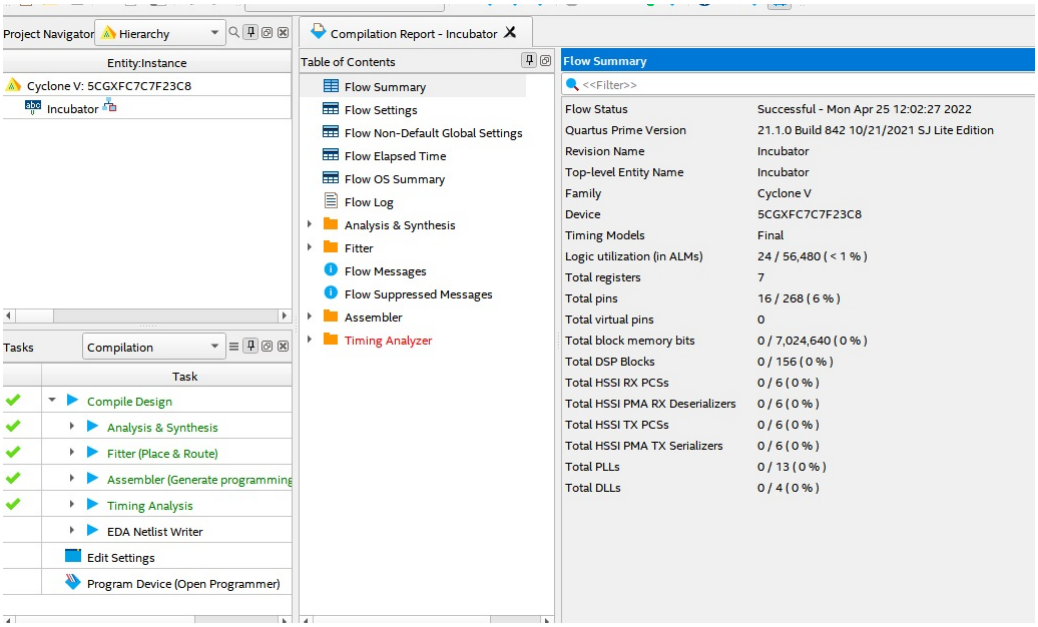
و نتیجه‌ی آن نیز در عکس زیر آمده است:



در تست بنچ، ابتدا مدار را ریست می‌کنیم و دما را بر روی صفر تنظیم می‌کنیم. پس در کلاک بعد در حالت S3 می‌رویم و heater روشن می‌شود. سپس با تغییر دما به ۲۰ تغییری در حالت رخ نمی‌دهد و heater روشن می‌ماند. سپس با افزایش دما به ۵۰، در ابتدا به حالت ۱ می‌رویم و سپس در کلاک بعدی به حالت ۲ می‌رویم و فن را روشن می‌کنیم. همچنین دقت کنید که سرعت فن تا 8 rps بالا می‌رود. سپس دما به ۳۰ درجه افت می‌کند و سرعت فن کم می‌شود. (ولی خاموش نمی‌شود.) از اینجا به بعد وارد درجه حرارت‌های رندوم می‌شویم. با توجه به نمودار داده شده در دستور کار می‌توان از صحت درست کار کردن مدار اطمینان حاصل کرد.

سنتز

سنتز را به کمک Quartus انجام می‌دهیم. دقت کنید که خط `include` را باید از فایل تست بنچ برداشت و سپس شروع به سنتز کرد. نتیجه‌ی سنتز در عکس زیر آمده‌است که همان طور که مشاهده می‌شود مشکلی ندارد:


Project Navigator
Hierarchy
Entity:Instance
Cyclone V: 5CGXFC7C7F23C8
Incubator
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming
Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)
Compilation Report - Incubator
Table of Contents
Flow Summary
Flow Settings
Flow Non-Default Global Settings
Flow Elapsed Time
Flow OS Summary
Flow Log
Analysis & Synthesis
Fitter
Flow Messages
Flow Suppressed Messages
Assembler
Timing Analyzer
Flow Summary
<<Filter>>
Flow Status Successful - Mon Apr 25 12:02:27 2022
Quartus Prime Version 21.1.0 Build 842 10/21/2021 SJ Lite Edition
Revision Name Incubator
Top-level Entity Name Incubator
Family Cyclone V
Device 5CGXFC7C7F23C8
Timing Models Final
Logic utilization (in ALMs) 24 / 56,480 (< 1 %)
Total registers 7
Total pins 16 / 268 (6 %)
Total virtual pins 0
Total block memory bits 0 / 7,024,640 (0 %)
Total DSP Blocks 0 / 156 (0 %)
Total HSSI RX PCSs 0 / 6 (0 %)
Total HSSI PMA RX Deserializers 0 / 6 (0 %)
Total HSSI TX PCSs 0 / 6 (0 %)
Total HSSI PMA TX Serializers 0 / 6 (0 %)
Total PLLs 0 / 13 (0 %)
Total DLLs 0 / 4 (0 %)

نتیجه‌گیری

در این آزمایش توانستیم مدار ترتیبی واحد کنترلی یک انکوباتور را طراحی کنیم. نکته‌ای که در طراحی این مدار وجود دارد این است که نمی‌توانیم در یک کلاک بیش از یک state پیاپی عوض کنیم و حتما در صورت نیاز باید چند کلاک بگذرد.