📝 گزارش کار تمرین ۶

طراحی سیستم‌های دیجیتال

> *Alireza Habibzadeh* `99109393`
> *July 8, 2022*

برای سنتز و شبیه‌سازی این تمرین از نرم‌افزار Xilinx ISE Design Suite v14.7 و سنتزها برای خانواده‌ی Spartan-3 FPGA Family انجام شده‌اند.

پیوست:

همراه این گزارش کدهای وریلاگ در دایرکتوری `HW6_99109393` و فایل‌های زیر در دایرکتوری `files` پیوست شده‌اند:

- فایل `datapath.pdf`: شکل datapath پیاده شده
- فایل `schematic_all.pdf`: خروجی شماتیک کلی به همراه مموری‌ها
- فایل `schematic_full.pdf`: خروجی شماتیک به همراه پیاده‌سازی داخلی
- فایل `schematic_memory.pdf`: خروجی شماتیک مموری‌ها
- فایل `schematic.pdf`: خروجی شماتیک کلی
- فایل‌های `synthesis_report`: گزارش سنتز
- فایل `synthesis_summary.pdf`: خلاصه‌ی گزارش سنتز
- فایل `wave.pdf` موج خروجی شبیه‌سازی

الف. ASM Chart؟؟

سه state زیر در این مدار وجود دارد:

سرد شد (OFF)
دمای Set Point بیشتر از میانگین یک دقیقه‌ی اخیر سنسور ۲ است و فن و خنک‌کننده خاموش هستند.

گرمه (COOL)
دمای Set Point کمتر از میانگین یک دقیقه‌ی اخیر سنسور ۱ و ۲ است پس خنک‌کننده روشن می‌شود و فن روی دور تند قرار می‌گیرد.

یکم گرمه (FAN)

دمای Set Point بین میانگین سنسور ۱ و ۲ است. در این حالت خنک‌کننده خاموش است و فن روشن. اگر اختلاف دمای میانگین یک دقیقه‌ی اخیر سنسور ۱ و ۲ کمتر از ۳ درجه بود دور کند و اگر بیشتر بود روی دور تند کار می‌کند.

نکته‌ی جالب این است که نیازی به پیاده‌سازی ASM Chart و به کلی state نیست چرا که این مدار به جز حافظه‌ای که برای ذخیره‌ی دماها در نظر گرفته شده نیاز به حافظه‌ی دیگری ندارد. در واقع با داشتن میانگین دماها وضعیت بعدی به طور یکتا بدون اثر حالت قبلی تعیین می‌شود. (حالت بعدی به حالت قبلی وابسته نیست.)

ب. Data-Path

بیشتر این مدار در قسمت controller آن پیاده‌سازی شده است. مقایسه‌های ساده در همان قسمت controller پیاده شده‌اند و در نهایت در همان قسمت سنتز می‌شوند. با این حال می‌توان مقایسه‌کننده‌های دما را به طور جداگانه در قسمت Data-Path قرار داد.

در انتهای فایل شماتیک Data-Path مدار آمده است.

توصیف حافظه

برای این مدار نیاز داریم تا بسته به ورودی‌های کاربر بتوانیم میانگین ۲۰، ۴۰ یا ۶۰ کلاک گذشته را محاسبه کنیم. بنا بر این نمی‌توانیم از حقه‌های میانگین‌گیری استفاده کنیم تا محبور نباشیم همه‌ی اعداد را ذخیره کنیم. برای مثال یک راه خوب برای محاسبه‌ی نوعی میانگین وزن‌دار این بود:

$$M_{n+1} = \frac{M_n + T_n}{2}$$

با این روش دیگر نیازی به ذخیره‌ی همه‌ی مقادیر قبلی نبود و مقادیر جدید با وزنی بالاتر از مقادیر قبلی وارد چرخه می‌شدند. واکنش مدار نسبت به تغییرات دما شبیه تابع نمایی است. (با حل معادله‌ی تفاضلی می‌توان این نکته را ثابت کرد) حتی می‌توان ضرایب $1/2$ و $1/2$ که در نظر گرفته شده را تنظیم کرد. به هر حال اکنون همانطور که در سوال هم اشاره شده نیاز به ذخیره‌ی کل داده‌ها داریم و باید میانگین‌گیری را به شکل کاملا naive بر روی آن‌ها انجام دهیم.

```
module sensor_memory(
    input [7:0] temp,
    input [7:0] duration,
    input clock,
    input reset,
    output reg [7:0] average_temp,
    output latest_temp
```

```
    );
    reg [7:0] temps [0:199];
    reg [7:0] i;
    reg [15:0] sum;
    assign latest_temp = temps[0];
    always @(posedge clock) begin
        if (reset) begin
            average_temp = 0;
            for (i = 0; i < 200; i = i + 1) temps[i] = 0;
        end else begin
            for (i = 0; i < 199; i = i + 1) temps[i + 1] = tem
ps[i];
            temps[0] = temp;
            sum = 0;
            for (i = 0; i < 200; i = i + 1) sum = sum + temps
[i] * (i < duration);
            average_temp = sum / duration;
        end
    end
endmodule
```

پ. توصیف رفتاری controller

این مدار state ندارد بنا بر این توصیف controller آن بسیار ساده است. تنها کافی است هر بار وضعیت ورودی‌ها بررسی شود و بر اساس آن خروجی‌ها تعیین می‌گردند.

البته از دو ماژول حافظه به صورت ساختاری استفاده شده که در حقیقت کار حفظ state بر عهده‌ی آن‌ها است.

```
module ac_controller(
    input [7:0] set_point,
    input clock,
    input reset,
    input [3:0] subcommand,
```

```
    input subcommand_enable,
    input [7:0] sensor1,
    input [7:0] sensor2,
    output reg fan,
    output reg fan_high,
    output reg ac,
    output reg [7:0] subcommand_out
    );
    // subcommands
    parameter reset_memory = 4'b0000, latest1 = 4'b1000, lates
t2 = 4'b1100;
    parameter ma20_1 = 4'b1001, ma40_1 = 4'b1010, ma60_1 = 4'b
1011;
    parameter ma20_2 = 4'b1101, ma40_2 = 4'b1110, ma60_2 = 4'b
1111;
    reg [7:0] duration;
    wire [7:0] average_temp1, average_temp2, latest_temp1, lat
est_temp2;
    wire memory_reset = reset | subcommand_enable & subcommand
== reset_memory;
    sensor_memory memory_1(
        .temp(sensor1),
        .duration(duration),
        .clock(clock),
        .reset(memory_reset),
        .average_temp(average_temp1),
        .latest_temp(latest_temp1)
    );
    sensor_memory memory_2(
        .temp(sensor2),
        .duration(duration),
        .clock(clock),
        .reset(memory_reset),
```

```
        .average_temp(average_temp2),
        .latest_temp(latest_temp2)
    );
    always @(posedge clock) begin
        if (reset) begin
            fan = 0;
            fan_high = 0;
            ac = 0;
            subcommand_out = 0;
            // memory modules are reseted by reset signal asyn
chronously
        end else if (subcommand_enable) begin
            case (subcommand)
                latest1: subcommand_out = latest_temp1;
                latest2: subcommand_out = latest_temp2;
                ma20_1: begin
                    duration = 20;
                    subcommand_out = average_temp1;
                end
                ma40_1: begin
                    duration = 40;
                    subcommand_out = average_temp1;
                end
                ma60_1: begin
                    duration = 60;
                    subcommand_out = average_temp1;
                end
                ma20_2: begin
                    duration = 20;
                    subcommand_out = average_temp2;
                end
                ma40_2: begin
```

```
                    duration = 40;
                    subcommand_out = average_temp2;
                end
                ma60_2: begin
                    duration = 60;
                    subcommand_out = average_temp2;
                end
            endcase
        end else begin
            duration = 60;
            if (set_point > average_temp2) begin
                fan = 0;
                ac = 0;
            end else if (set_point < average_temp1 & set_point < average_temp2) begin
                fan = 1;
                fan_high = 1;
                ac = 1;
            end else if (set_point > average_temp1 & set_point < average_temp2) begin
                fan = 1;
                ac = 0;
                if (average_temp2 - average_temp1 > 3) fan_high = 1;
                else fan_high = 0;
            end
        end
    end
endmodule
```

تست

نهایتا با simulate کردن test-bench در نرم‌افزار و بررسی شکل موج آن از عملکرد درست مدار مطمئن می‌شویم.
کد تست‌بنچ به دلیل طولانی بودن در اینجا نیامده است اما در فایل‌های ارسالی موجود می‌باشد.
شکل موج خروجی در انتهای فایل و در پیوست‌ها آمده است.

سنتز

برای سنتز این قطعه با خطاهایی مواجه شدیم که عموما مربوط به استفاده‌ی همزمان از گزاره‌های blocking و non-blocking بود. پس از رفع این خطاها (البته با در نظر گرفتن منطق درست)، سنتز با موفقیت انجام شد که گزارش و... به پیوست ارسال شده است.