📝 گزارش کار تمرین ۵

طراحی سیستم‌های دیجیتال

> *Alireza Habibzadeh* `99109393`
> *May 24, 2022*

الف. طراحی ASM Chart

برای طراحی ASM Chart چهار state زیر را پیاده می‌کنیم:

آماده‌سازی (Initialization)
در این مرحله اگر دکمه‌ی `Start` زده نشده بود دوباره به همین مرحله باز می‌گردیم، اگر زده شده بود باید رجیسترهایمان را به مقدار اولیه بازگردانیم و ورودی‌ها را بخوانیم و در رجیسترهای مربوطشان ذخیره کنیم. همچنین باید سیگنال `Ready` را غیر فعال کنیم چون از این لحظه خروجی قبلی نامعتبر است و مقادیر جدیدی برای محاسبه درخواست شده‌اند.

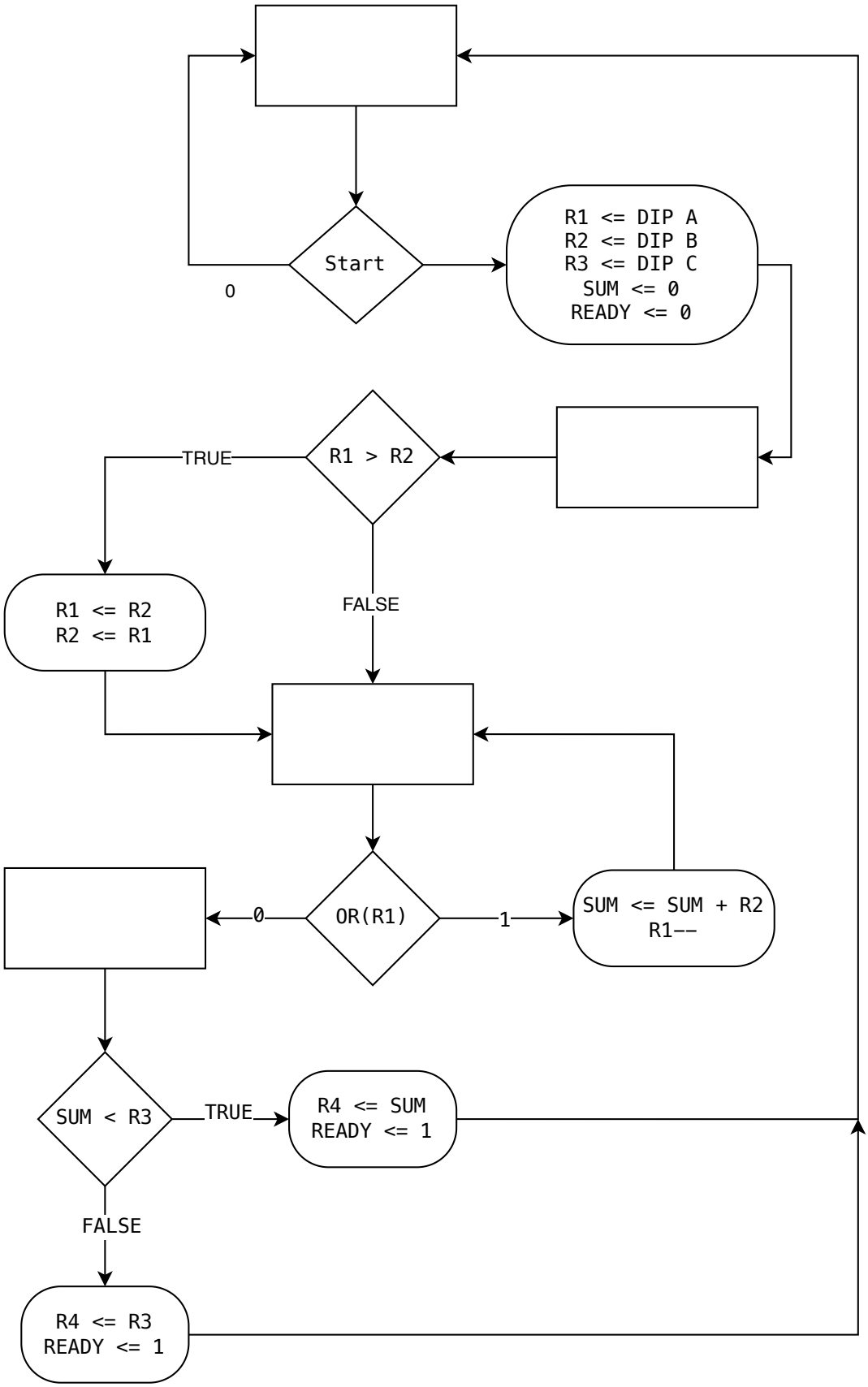
بهینه‌سازی (Optimization)
در این مرحله دو رجیستر جمع‌شونده را مقایسه می‌کنیم و رجیستر کوچک‌تر را به جای رجیستری که شمارنده است قرار می‌دهیم تا عملیات در تعداد کلاک کمتری انجام شود.

جمع متوالی
در این مرحله یکی از رجیستر ها را به `SUM` اضافه می‌کنیم و یکی از رجیستر شمارنده کم می‌کنیم. اگر شمارنده صفر شد به state بعدی و در غیر این صورت دوباره به همین state می‌رویم.

شرط پایانی
در این state شرط پایانی خواسته شده را چک می‌کنیم و پس از آن فلگ `Ready` را روشن می‌کنیم که یعنی نتیجه حاضر شده است. برای عدم نشت مقدار میانی مدار از یک رجیستر جدید برای مقدار نهایی استفاده می‌کنیم.

پیاده‌سازی ASM Chart در صفحه‌ی بعدی آمده.

DSD – HW5
Alireza Habibzadeh 99109393



ب. منابع لازم برای DataPath

برای یافتن منابع لازم همه‌ی جملات داخل باکس‌ها را که وظیفه‌ی بخش DataPath هستند لیست می‌کنیم:

```
1  R1 <= DIP A
2  R2 <= DIP B
3  R3 <= DIP C
4  SUM <= 0
5  READY <= 0
6  R1 > R2
7  R1 <= R2
8  R2 <= R1
9  OR(R1)
10 SUM <= SUM + R2
11 R1--
12 R4 <= SUM
13 READY <= 1
14 SUM < R3
15 R4 <= R3
16 READY <= 1
```

با توجه به لیست بالا منابع زیر مورد نیاز است:

- ۲ عدد مقایسه‌کننده (می‌شد با ControlUnit پیچیده‌تر و MUX بیشتر تعداد مقایسه‌کننده‌ها را به یکی رساند؛ توضیحات زیر لیست را بخوانید)
- ۳ عدد MUX برای تعیین چیزی که در `R1` و `R2` و`R4` لود می‌شود.
- ۱ عدد گیت ۳۲ بیتی OR برای خط ۹
- ۵ عدد رجیستر برای `R1` و `R2` و `R3` و `SUM` و `R4`
- ۱ عدد فلیپ‌فلاپ برای `READY`
- ۱ عدد Adder برای خط ۱۰

می‌شد هنگام مقایسه‌ی دوم، مقادیر `SUM` و `R4` که مقایسه می‌شوند را در رجیسترهای قبلی که مقایسه شدند ریخت تا از همان مقایسه‌کننده برای این مقایسه هم استفاده شود. اما این کار پیچیدگی ControlUnit ما را بیشتر می‌کرد و همچنین به حالت‌های MUX ما اضافه می‌کرد که برای همین این کار انجام نشده.

پ. توصیف رفتاری منابع

توصیف MUX به صورت رفتاری

```
module multiplexer(
    input [31:0] A,
    input [31:0] B,
    output [31:0] Y,
    input sel
    );

    always @(A or B or sel) begin
        case(sel)
            0: Y <= A;
            1: Y <= B;
        endcase
    end
endmodule
```

توصیف Comparator به صورت رفتاری

```
module comparator (
  input [31:0] a,
  input [31:0] b,
  output reg greater,
  output reg equal,
  output reg less
  );

  always @(a,b) begin
    if (a > b) begin
```

```
      greater = 1;
      equal = 0;
      less = 0;
    end else if (a == b) begin
      greater = 0;
      equal = 1;
      less = 0;
    end else begin
      greater = 0;
      equal = 0;
      less = 1;
    end
  end
endmodule
```

توصیف رفتاری گیت OR

(هرچند اصولا بهتر است این را ساختاری توصیف کنیم چون خود ساختار است ولی با توجه به خواسته‌ی سوال این هم رفتاری توصیف شده)

```
module or_gate(
    input a[31:0],
    output c
);
    integer i;

    always @(a) begin
        c <= a[0];
        for (i=1; i<32; i=i+1) begin
            c <= c | a[i];
        end
    end
endmodule
```

توصیف رفتاری جمع کننده

```
module adder(input [31:0] a, input [31:0] b, output [31:0] sum);
    always @(a, b) sum = a + b;
endmodule : adder
```

اتصال ساختاری

قطعات باید مطابق شکل صفحه‌ی بعد به هم متصل شوند.

ت. نمودار حالت طراحی

نمودار حالات ما (state diagram) در صفحه‌ی بعد آمده است.

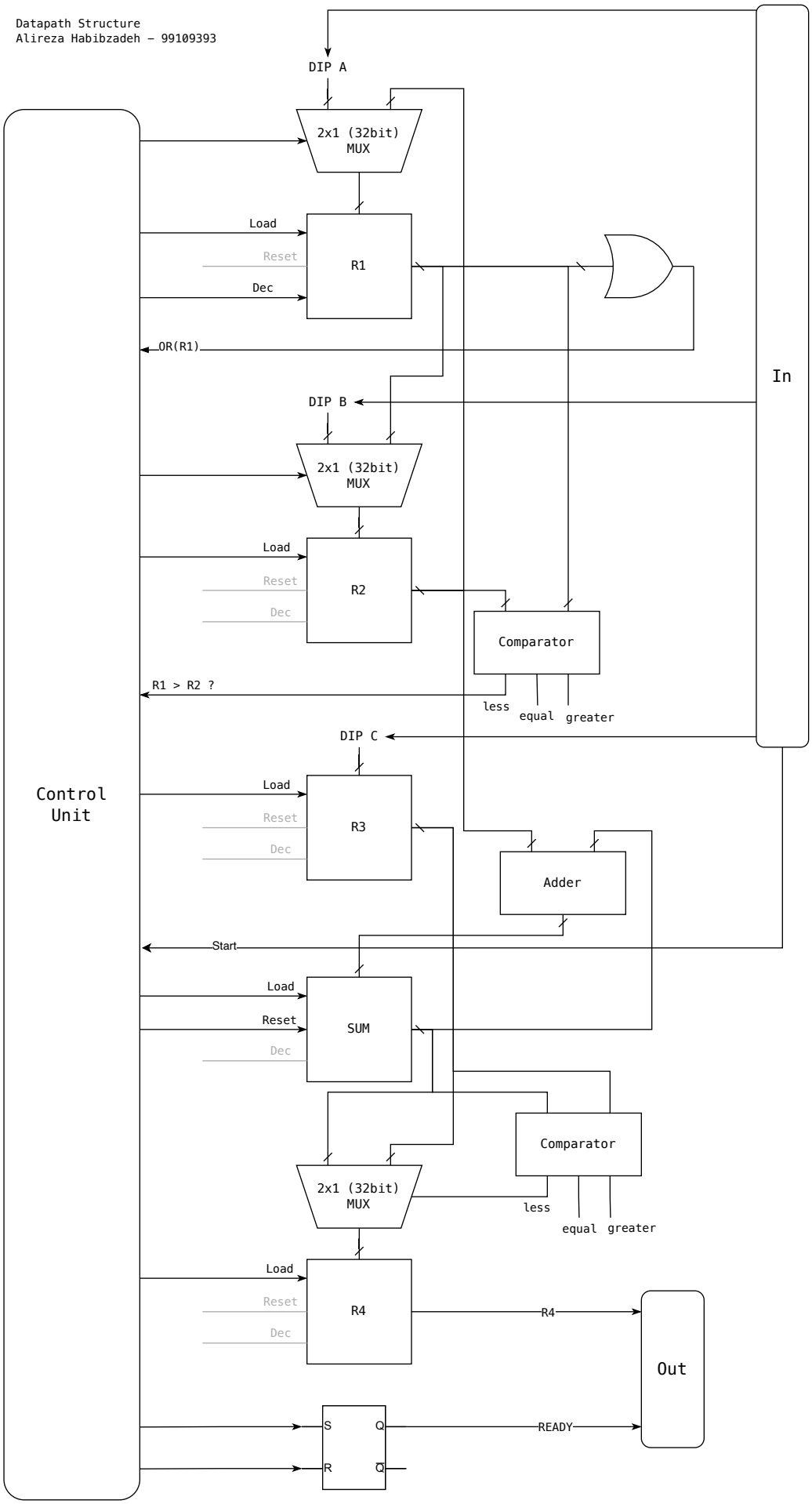
ث. توصیف Control Unit با وریلاگ

توصیف بخش Control Unit به صورت رفتاری در ادامه آمده است. این کدها به صورت ضمیمه هم در فایل ارسال قرار دارند.

```
// control_unit for our ASM
module control_unit(
    input reset, clk, START, OR_R1, CMP_L_R1,
    output reg L_R1, L_R2, L_R3, L_R4, L_SUM, Dec_R1, Sel_R1, Sel_R2, R_SUM, S_READY, R_READY
    );

    reg [1:0] p_state, n_state;
    localparam [1:0] init = 2'b00, cmp = 2'b01, mul = 2'b10, cmp2 = 2'b11;

    always @(p_state or START or OR_R1 or CMP_L_R1) begin
        n_state = init;
        case (p_state)
            init: begin
```

```
            if (START) begin
                n_state <= cmp;
                Sel_R1 <= 1'b0;
                Sel_R2 <= 1'b0;
                L_R1 <= 1'b1;
                L_R2 <= 1'b1;
                L_R3 <= 1'b1;
                S_READY <= 1'b0; R_READY <= 1'b1;
                R_SUM <= 1'b1;
            end
        end
        cmp: begin
            n_state <= mul;
            if (CMP_L_R1) begin
                L_R1 <= 1'b1;
                L_R2 <= 1'b1;
                Sel_R1 <= 1'b1;
                Sel_R2 <= 1'b1;
            end
        end
        mul: begin
            if (OR_R1) begin
                n_state <= mul;
                Dec_R1 <= 1'b1;
                L_SUM <= 1'b1;
            end else n_state <= cmp2;
        end
        cmp2: begin
            n_state <= init;
            L_R4 <= 1'b1;
            S_READY <= 1'b1; R_READY <= 1'b0;
```

```
            end
        endcase
    end

    always @(posedge clk) begin
        if (reset) p_state <= init;
        else p_state <= n_state;
    end
endmodule : control_unit
```


Datapath Structure
Alireza Habibzadeh – 99109393
DIP A
2x1 (32bit)
MUX
Load
Reset
Dec
R1
OR(R1)
In
DIP B
2x1 (32bit)
MUX
Load
Reset
Dec
R2
Comparator
R1 > R2 ?
less
equal
greater
DIP C
Control
Unit
Load
Reset
Dec
R3
Adder
Start
Load
Reset
Dec
SUM
Comparator
2x1 (32bit)
MUX
less
equal
greater
Load
Reset
Dec
R4
R4
Out
S
Q
R
Q
READY

State Diagram
Alireza Habibzadeh – 99109393