📝 گزارش آزمایش ۸

آزمایشگاه طراحی سیستم‌های دیجیتال

Alireza Habibzadeh `99109393`
June 7, 2022

مقدمه

در این آزمایش قصد داریم یک ALU برای اعداد مختلط درست کنیم. البته با صفر گذاشتن قسمت موهومی می‌توان از آن برای عملیات روی اعداد عادی هم استفاده کرد.

ماژول‌های حافظه

دو ماژول حافظه یکی برای دستورات و یکی برای داده‌ها در نظر گرفته شده است. پیاده سازی ماژول حافظه‌ی داده‌ها به شکل زیر است:

```
// data memory
module data_mem(
    input clock,
    input reset,
    input write_en,
    input [4:0] address,
    input [7:0] data_in,
    output reg [7:0] data_out
    );
    reg [7:0] data_reg [0:31];

    always @(negedge clock or posedge reset) begin
        if(reset) begin
            data_out = 8'b0;
        end
        else begin
            if(write_en) data_reg[address] <= data_in;
            data_out <= data_reg[address];
        end
    end
```

```
endmodule
```

پیاده‌سازی ماژول حافظه‌ی دستورات هم مشابه همین است.

جمع و ضرب اعداد مختلط

برای جمع و ضرب دو عدد مختلط ابتدا قسمت‌های حقیقی و موهومی آن‌ها را جدا می‌کنیم و سپس مطابق روابط زیر چند عملیات ساده جمع و ضرب انجام می‌دهیم. (که یعنی نتیجه‌ی ما در صورت نداشتن جمع‌کننده/ضرب‌کننده‌های متعدد در یک کلاک آماده نمی‌شود)

$$\tilde{C}_1 + \tilde{C}_2 = (a_1 + ib_1) + (a_2 + ib_2) = (a_1 + a_2) + i(b_1 + b_2)$$
$$\tilde{C}_1 \cdot \tilde{C}_2 = (a_1 + ib_1)(a_2 + ib_2) = (a_1a_2 - b_1b_2) + i(a_1b_2 + a_2b_1)$$

```
// add or subtract two complex numbers
module complex_add_subtract(
    input wire [7:0] a,
    input wire [7:0] b,
    input wire op,
    output wire [7:0] c
    );

    wire [3:0] a_real, a_imag, b_real, b_imag, c_real, c_imag;
    assign a_real = a >> 4;
    assign a_imag = a;
    assign b_real = b >> 4;
    assign b_imag = b;

    assign c_real = op ? a_real + b_real : a_real - b_real;
    assign c_imag = op ? a_imag + b_imag : a_imag - b_imag;

    assign c = {c_real, c_imag};
endmodule
```

```
// multiply two complex numbers
module complex_mul(
    input wire [7:0] a,
    input wire [7:0] b,
    output wire [7:0] c
```

```
    );

    wire [3:0] a_real, a_imag, b_real, b_imag, c_real, c_imag;
    assign a_real = a >> 4;
    assign a_imag = a;
    assign b_real = b >> 4;
    assign b_imag = b;

    assign c_real = a_real * b_real - a_imag * b_imag;
    assign c_imag = a_real * b_imag + a_imag * b_real;

    assign c = {c_real, c_imag};
endmodule
```

آپکدها

از جدول زیر برای تعیین عملکرد opcodeها استفاده شده است. این‌ها طوری تعیین شده‌اند که به سادگی بتوان بیت کم‌ارزش opcode را به opcode جمع‌کننده/تفریق‌کننده داد.

opcode	operation
00	nop (skip)
01	multiply
10	subtract
11	add