دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش پروژه آزمایشگاه مدار منطقی و معماری کامپیوتر

نحوه اجرا کدهای دستوری

دانشجو:

علی خورانی(۴۰۱۲۱۴۴۱۰۵۴۲۵۳)

استاد:

مهندس بهروز طاهری

ترم بهمن۱۴۰۲

به نام خدا

ابتدا یک ماژول full_adder تعریف می کنیم که دارای ورودی های a&b&carry in ودارای خروجی های result &carry_outمی باشد.

ساختارassign:دراین ساختار می توان از اُپراتور جبری جمع +، تفریق -، ضرب *، شیفت به راست و چپ <<و >> و= استفاده کرد که باعث ساده سازی کد دستوری به زبان ریاضی و فهم ودرک ساده تر کاربر می شود و کمک شایانی در کاهش خطوط برنامه می کند.

در ادامه سه ورودی a&b&carry_inباهم XORمی شود.

و دوباره سه ورودی a&b&carry_inباهم باید OR شودکه قبل ازآن یک بار ANDمی شود.

```
// ماژول جمع‌کننده کامل
module full_adder(
                input a,
                input b,
                input carry_in,
                output result,
                output carry_out
                );

                    assign result= a ^ b ^ carry_in;
                        assign carry_out = (a & b) | (b & carry_in) | (a & carry_in);

endmodule
```

سپس به ماژول اصلی برنامه ماژول adder_subtractor_4bit می رویم دوعدد چهار بیتی به عنوانورودی و ورودی add_subکه اگر صفر باشد عملیات جمع دوعدد چهاربیتی اتفاق می افتد و اگر یک باشد عملیات تفریق انجام می شود.

و دارای خروجی های result۵بیتی و carry-out می باشد.

سپس wireهای b_xorرا که برای عملیات تفریق کاربرد دارد و carry_out &&carry_inرا را بصورت وایر تعریف کرده و مقدارb_xorبرابر است با xor bچهاربیت آن با add_subودر ابتدا مقدارcarry_inبرابراست با add_sub.

```
module adder_subtractor_4bit(
                        input [3:0] a,
                        input [3:0] b,
                        input add_sub,
                        output [4:0] result
                        output carry
                        );

    wire [3:0] b_xor;
        assign b_xor = b ^ {4{add_sub}};
            wire [3:0] sum;
             wire carry_out;
                wire carry_in;
                   assign carry_in = add_sub; // برای تفریق، کری اولیه باید 1 باشد
```

در ادامه برای تک تک بیت های ورودی از ماژول full_adderبصورت فراخوانی در چهار مرحله استفاده می شود .

```
// اجرای جمع 4 بیتی
full_adder fa0 (
.a(a[0]),
.b(b_xor[0]),
.carry_in(carry_in),
.result(result[0]),
.carry_out(carry_out0)
);
    full_adder fa1 (
    .a(a[1]),
    .b(b_xor[1]),
    .carry_in(carry_out0),
    .result(result[1]),
    .carry_out(carry_out1)
    );
        full_adder fa2 (
        .a(a[2]),
        .b(b_xor[2]),
        .carry_in(carry_out1),
        .result(result[2]),
        .carry_out(carry_out2)
        );
            full_adder fa3 (
            .a(a[3]),
            .b(b_xor[3]),
            .carry_in(carry_out2),
            .result(result[3]),
            .carry_out(carry_out)
            );
```

در انتها مقدار خروجی full_adder۴درون resultریخته می شودو اگرکری نیز داشته باشیم برابر است با آخرین کری خروجی.

```
        assign result =result;
        assign carry = carry_out; // کری برای جمع

endmodule
```

TTB:

TTBدرواقع یک ساختاری به نام تست بنچ است برای تست کردن ماژول های نوشته شده است خود ماژول تست بنچ بدون ورودی و خروجی می باشد اما در ادامه ورودی هارا بصورتREGوخروجی هارا بصورتWIREتعریف می کنیم و در ادامه ماژولی که برای تست به آن نیاز داریم را یکبار فراخوانی می کنیم سوال: آیا باید ماژول اصلی را در فایل تست بنچ دوباره تعریف کرد؟ پاسخ: خیر. نیازیبه این کار نیست تنها با وارد کردن آن فایل به این فایل کار انجام می شود. بصورت زیر:

```
`include "./project.v"
```

سپس در قسمت intialیک begin&endتعریف می کنیم و مقدار های شبیه سازی را در هر کیس به ورودی مدنظر داده و در پایان زمان انجام آن را مشخص می کنیم مانند:

```
    a =0;
    b = 2;
    add_sub = 0;
    #10;
```

ودر انتها ازکلمه$finishبه منظور پایان شبیه سازی استفاده می شود.

```
$finish;
```

درادامه مثالی ازکد دستوری پروژه جمع کنندهو تفریق کننده چهاربیتی را مشاهده می کنیم.

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


module test_pr();

            reg [3:0] a;
            reg [3:0] b;
            reg add_sub;
            wire[4:0]result;
            wire carry;

adder_subtractor_4bit uf(
```

```
                        .a(a),
                        .b(b),
                        .add_sub(add_sub),
                        .result(result),
                        .carry(carry)
                        );
initial
  begin
          $dumpfile("TBB.vcd");
          $dumpvars(0 ,test_pr);
    // Test Case 1: Addition
    a =0;
    b = 2;
    add_sub = 0;
    #10;

    // Test Case 2: Subtraction
    a = 10;
    b = 1;
    add_sub = 1;
    #10;
    // test case 3: Addition
    a=7;
    b=6;
    add_sub =0;
    #10;
    //you can test more case of Addition &&  Subtraction



    $finish;


  end


endmodule
```

درنهایت از استاد مهندس بهروز طاهری کمال تقدیر و تشکر را بابت همکاری و راهنمایی پروژه می نماییم.

پایان