Calaculator

Parsa Ghafari(401102178)

January 29, 2024

1 code

1.1 main

we define raw-first-input for our first input and second input and out for output.

Debounce process : We use debouncing for noise cancelling.we input one number for each clock cycle the input

$$rawfirstinput + 1$$

to debounce it.

Main logic: this part is doing the process. we assign specific number for each operator with case.

3'b000 : addition

3'b001 : subtraction

3'b010 : multiplication

3'b011 : fraction (it isn't in project) but it shows only integer output

if we want to create calculator without debounce we can use only the main logic at code that we write that with case and assign each number to specific operator.

1.2 Test bench part

my test bench code has the normal form of type that ready in verilog.

in the initial part we assign the number to input and output.

first we assign

$$rawfirstinput = 100$$

$$secondinput = 50$$

and

$$operatorinput = 3'b000 = addition$$

but we assume that we have 5 clock cycle . in result raw-first-input changes to 105.

and we have

$$rawfirstinput = 105$$

$$secondinput = 50$$

and

$$operatorinput = 3'b000 = addition$$

1.3 Output part

```
module cal_deb (
    input wire clk,
    input wire [19:0] raw_first_input,
    input wire [2:0] operator_input,
    input wire [19:0] second_input,
    output wire [19:0] out);
    reg [19:0] result;
    reg [19:0] debounced_first_input;
    reg [3:0] debounce_counter;
    reg debounce_done_internal;
    // Debounce process
    always @(posedge clk) begin
        if (debounce_counter < 4) begin
            debounce_counter <= debounce_counter + 1;
            debounced_first_input <= debounced_first_input;
            debounce_done_internal <= 0; // Reset internal signal
        end else begin
            debounce_counter <= 0;
            debounced_first_input <= raw_first_input;
            debounce_done_internal <= 1;
        end
    end
    // Main logic
    always @(posedge clk) begin
        if (debounce_done_internal) begin
            case (operator_input)
                3'b000: result <= debounced_first_input + second_input; // addition
                3'b001: result <= debounced_first_input - second_input; // subtraction
                3'b010: result <= debounced_first_input * second_input; // multiplication
                3'b011: if (second_input != 0) result <= debounced_first_input / second_input; // division
                default: result <= result; // equals
            endcase
        end
    end
    assign out = result;
endmodule
```

Figure 1: Main part.

```
module cal_deb_tb;
    reg clk;
    reg [19:0] raw_first_input; //first input
    reg [2:0] operator_input; // operator
    reg [19:0] second_input; // seconf input
    wire [19:0] out; // uotput

    cal_deb uut (
        .clk(clk),
        .raw_first_input(raw_first_input),
        .operator_input(operator_input),
        .second_input(second_input),
        .out(out)
    );

    initial begin
        clk = 0;
        raw_first_input = 20'd100; // assign number
        operator_input = 3'b000; // assign number
        second_input = 20'd50; // assign number

        // Apply debounced input after a few clock cycles
        #5 raw_first_input = 20'd105;

        // Monitor output
        $monitor(" out=%d", out);

        // Wait for a few clock cycles
        #50 $finish;
    end

    always #10 clk = ~clk;

endmodule
```

Figure 2: test bench.

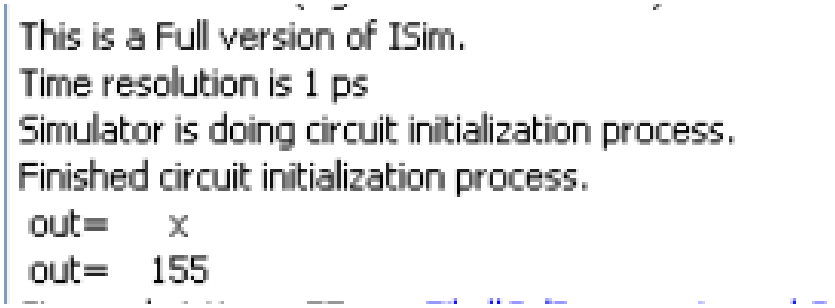

Figure 3: 105+50=155.

Figure 4: 105-50=55.

Figure 5: 5250=105*50.

Figure 6: $2{=}105_{\overline{50}}$.