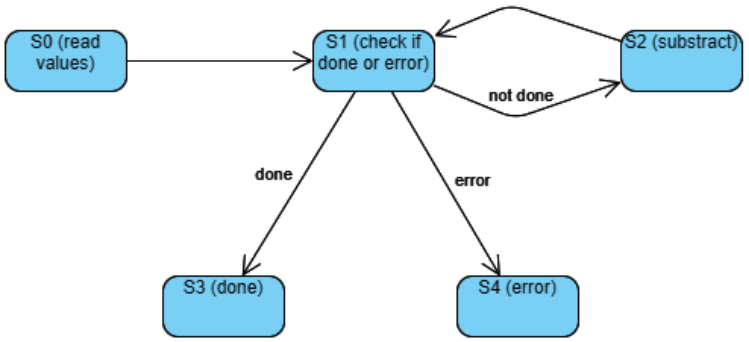
Lab 8 report

Name: Oscar Stark Student ID: F11015127

In this lab I implemented a divider. The 8 switches are used to input the dividend (4 switches) and the divisor (4 switches). The reset button is pressed to calculate the quotient and remainder, which are displayed in 8 led lights. Also another led light is used to display if there was an error.

State Transition Graph



Starting from the state S0, the system will calculate the result, ending either in states S3 or S4. To do another division, the reset button must be pressed.

State Transition Table

Present State (PS)	Next State (NS)			
	reset = 1	reset = 0		
S0	S0	S1		
S1	S0	if(divisor==0) S4	if(dividend>=divisor) S2	if(dividend<divisor) S3
S2	S0	S1		
S3	S0	S3		
S4	S0	S4		

Verilog Code

Main code

```
module task1(CLK, reset, input_divisor, input_dividend, quotient, remainder, error);
    input CLK, reset;
    input [3:0] input_divisor, input_dividend;
    output [3:0] quotient;
    output reg [3:0] remainder;
    output reg error;

    parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011, S4 = 3'b100;
    wire [3:0] divisor, dividend;
    reg [2:0] present_state, next_state;
    reg quotient_CLK, quotient_zero, subs_CLK, subs_read;

    quotient_counter QC(quotient_CLK, quotient_zero, quotient);
    substractor S(subs_CLK, subs_read, input_dividend, input_divisor, dividend, divisor);

    // part 1: initialize to state A and update state register
    always @(posedge CLK, posedge reset)
    begin
        if (reset) present_state <= S0;
        else present_state <= next_state; //update present state
    end

    // part 2: determine next state
    always @(present_state)
    begin
        case (present_state)
            S0: next_state=S1; //read the values from the switches
            S1: begin //check if done or error
                if(divisor==0) next_state=S4;
                else if(dividend>=divisor) next_state=S2;
                else next_state=S3;
            end
            S2: next_state=S1; //substract and quotient+1
            S3: next_state=S3; //done
            S4: next_state=S4; //error
        endcase
    end
```

```
    // part 3: evaluate output function
    always @(present_state)
    begin
        case (present_state)
            S0: begin //read the values from the switches
                subs_read = 1;
                quotient_zero = 1;
                remainder = 0;
                error = 0;
            end
            S1: begin //check if done or error
                subs_read = 0;
                subs_CLK = 0;
                quotient_CLK = 0;
                quotient_zero = 0;
            end
            S2: begin //substract and quotient+1
                subs_CLK = 1;
                quotient_CLK = 1;
            end
            S3: remainder = dividend; //done
            S4: error = 1; //error
        endcase
    end
endmodule
```

Quotient counter

This module is used to keep track of the quotient.

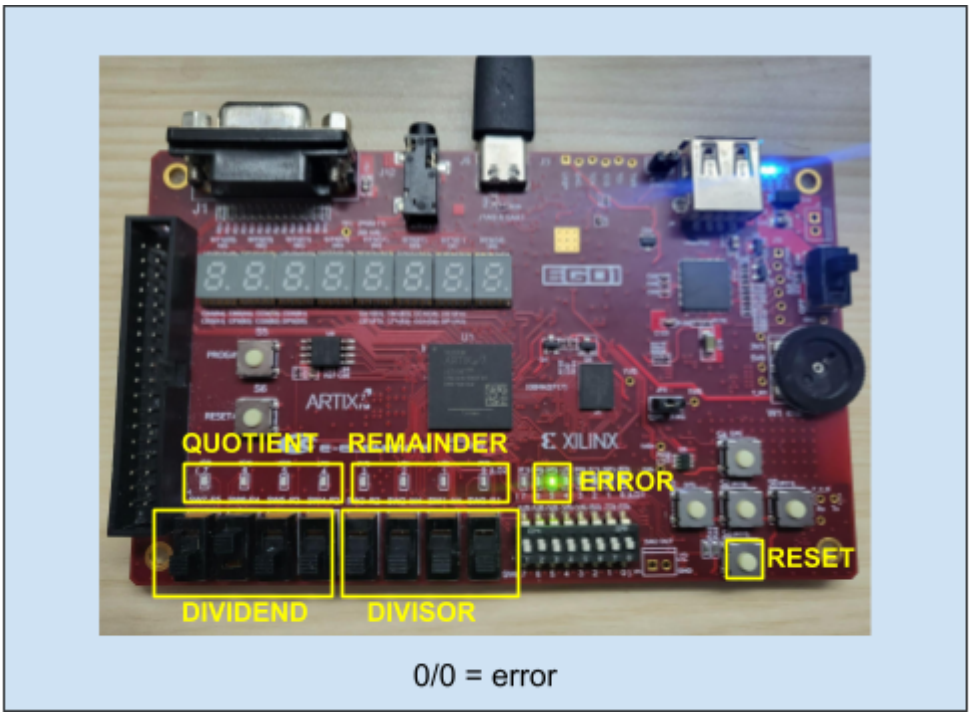
```
module quotient_counter(CLK, zero, quotient);
    input CLK, zero; //CLK adds 1 to the quotient
                     //zero sets quotient = 0
    output reg [3:0] quotient;

    always @(posedge CLK, posedge zero)
    begin
        if(zero) quotient = 0;
        else quotient = quotient+1;
    end
endmodule
```

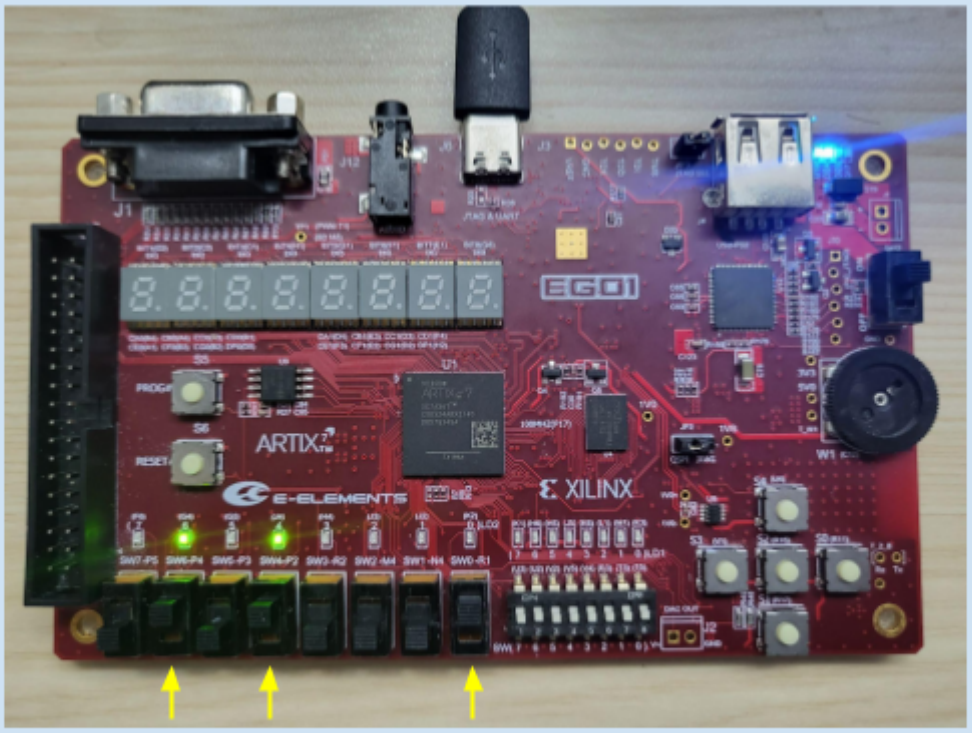
Subtractor

This module is used to read the values from the switches and to subtract the divisor from the dividend.

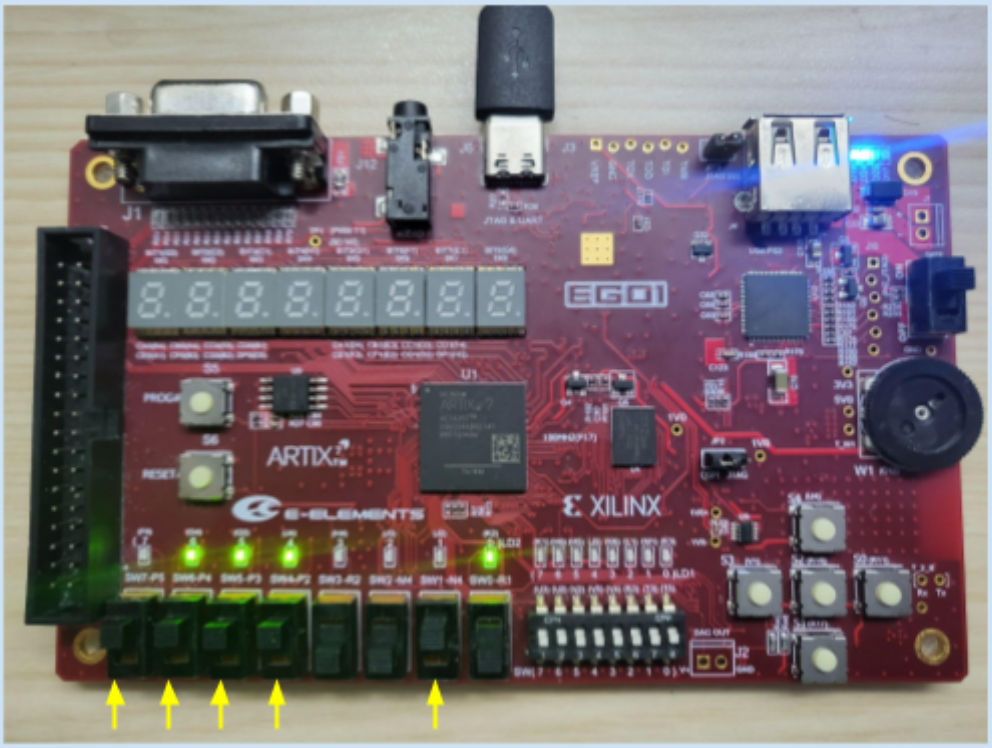
```
module substractor(CLK, read, input_dividend, input_divisor, dividend, divisor);
    input CLK, read; //CLK substracts
                     //read reads the values from the switches
    input [3:0] input_dividend, input_divisor; //these are the switches' values
    output reg [3:0] dividend, divisor; //these are the stored values

    always @(posedge CLK, posedge read)
    begin
        if(read)
        begin
            dividend = input_dividend;
            divisor = input_divisor;
        end
        else dividend = dividend - divisor;
    end
endmodule
```

FPGA Results





5/1 = 5



15/2 = 7, remainder = 1