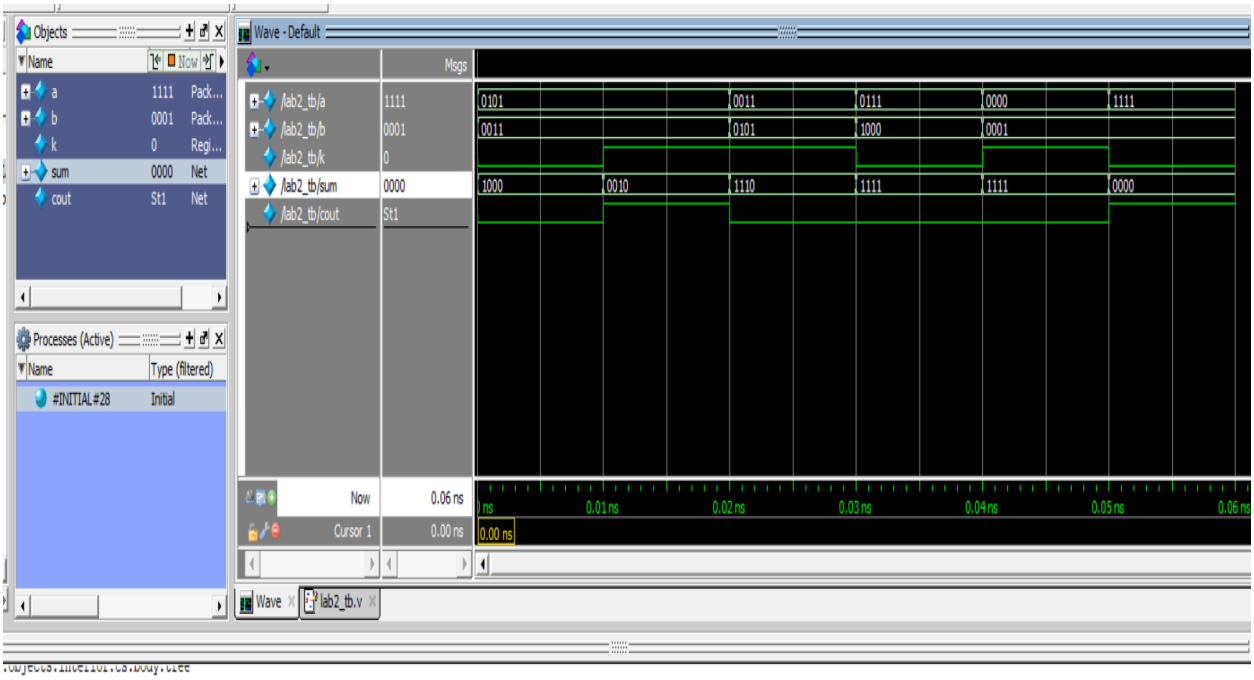
LAB 2:

Date: 10 de mayo de 2025



Alonso Emmanuel López Macias

César Eduardo Inda Ceniceros.

Team: 14

Introduction

This lab focused on the design and implementation of a 4-bit binary adder-subtractor. A binary adder is a fundamental digital circuit that performs binary addition. When extended, it can be used for subtraction through the use of two's complement representation. The goal of this lab was to gain experience with module instantiation, continuous assignment using logical operators (without the '+' operator), and hierarchical circuit design. Ultimately, the circuit was tested using both a testbench and an FPGA, allowing us to observe and verify its functionality.

- # Requirements

  Design a 4-bit binary adder-subtractor using half-adder and full-adder modules.

- Use continuous assignments and logical operators (^, &, |) instead of the + operator.
- Implement the circuit hierarchically using module instantiation.
- Create a testbench to verify functionality.
- Deploy the circuit on an FPGA board and:

  Use SW[17:14] for operand A.

  Use SW[3:0] for operand B.

  Use SW[4] to select between addition and subtraction.

  Display inputs A and B on red LEDs.

  Display the result on green LEDs.

  Show inputs and result on 7-segment displays.

  Represent the result as a signed decimal value with correct polarity.

Development

The circuit was developed in a modular and hierarchical manner:

1. **Half Adder**: Implemented using XOR and AND gates.

```
module lab2_half_adder (
    input a, b,
    output sum, carry
);
    assign sum = a ^ b;
    assign carry = a & b;
endmodule
```

2. **Full Adder**: Built by instantiating two half adders and an OR gate.

```
module lab2_full_adder (
    input a, b, cin,
    output sum, cout
);
    wire s1, c1, c2;
    lab2_half_adder HA1 (.a(a), .b(b), .sum(s1), .carry(c1));
    lab2_half_adder HA2 (.a(s1), .b(cin), .sum(sum), .carry(c2));
    assign cout = c1 | c2;
endmodule
```

3. **4-bit Adder-Subtractor**: Composed of four full adders connected in cascade. The subtraction is achieved by XOR-ing B with the control bit and setting the initial carry-in to that bit.

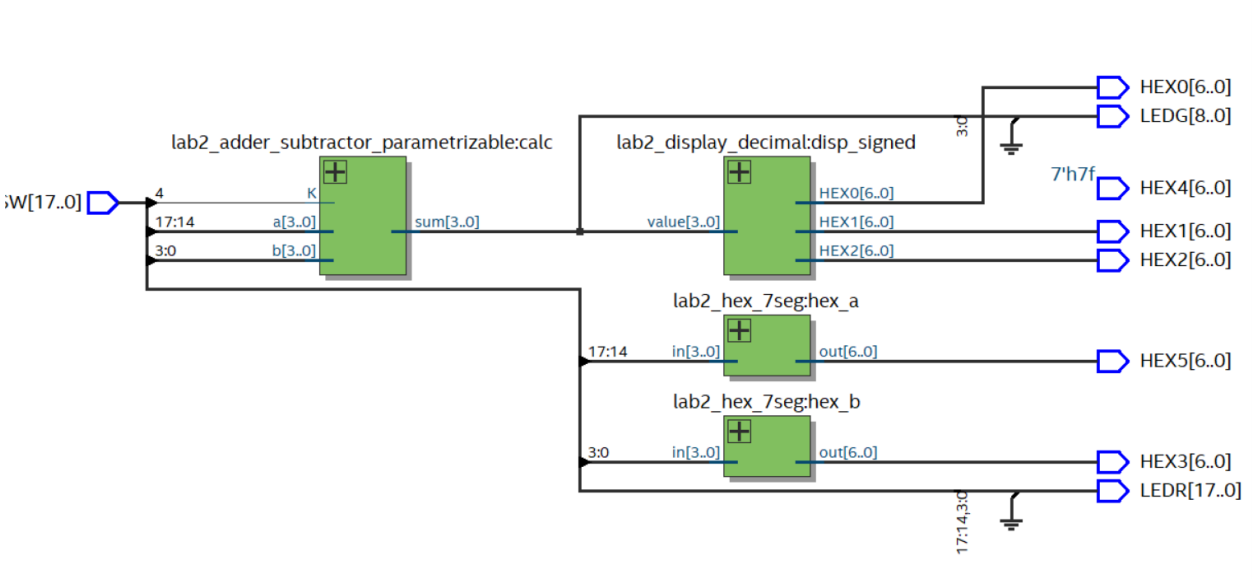
```
module lab2_adder_subtractor_parametrizable #(parameter WIDTH = 4) (
    input [WIDTH-1:0] a, b,
    input K,
    output [WIDTH-1:0] sum,
    output cout
);
    wire [WIDTH-1:0] b_xor;
    wire [WIDTH:0] carry;

    assign b_xor = b ^ {WIDTH{K}};
    assign carry[0] = K;

    genvar i;
    generate
        for (i = 0; i < WIDTH; i = i + 1) begin : gen_add
            lab2_full_adder FA (
                .a(a[i]),
                .b(b_xor[i]),
                .cin(carry[i]),
                .sum(sum[i]),
                .cout(carry[i+1])
            );
        end
    endgenerate

    assign cout = carry[WIDTH];
endmodule
```

4. **Top-level Module**: Connected the logic to FPGA switches, LEDs, and 7-segment displays. A conversion module was used to display the result as a signed decimal.



Testbench

In this testbench, we have the declaration of signals where two 4-bit inputs (a and b) are defined, along with a selector k. This selector determines whether the operation will be addition or subtraction (0 for addition, 1 for subtraction).

Also, we have the result of the operation and an output carry.

```
module lab2_tb;

    reg [3:0] a, b;
    reg k; // 0 = suma, 1 = resta
    wire [3:0] sum;
    wire cout;
```

Next, the testbench signals are connected to the module, which is a parameterized adder/subtractor.

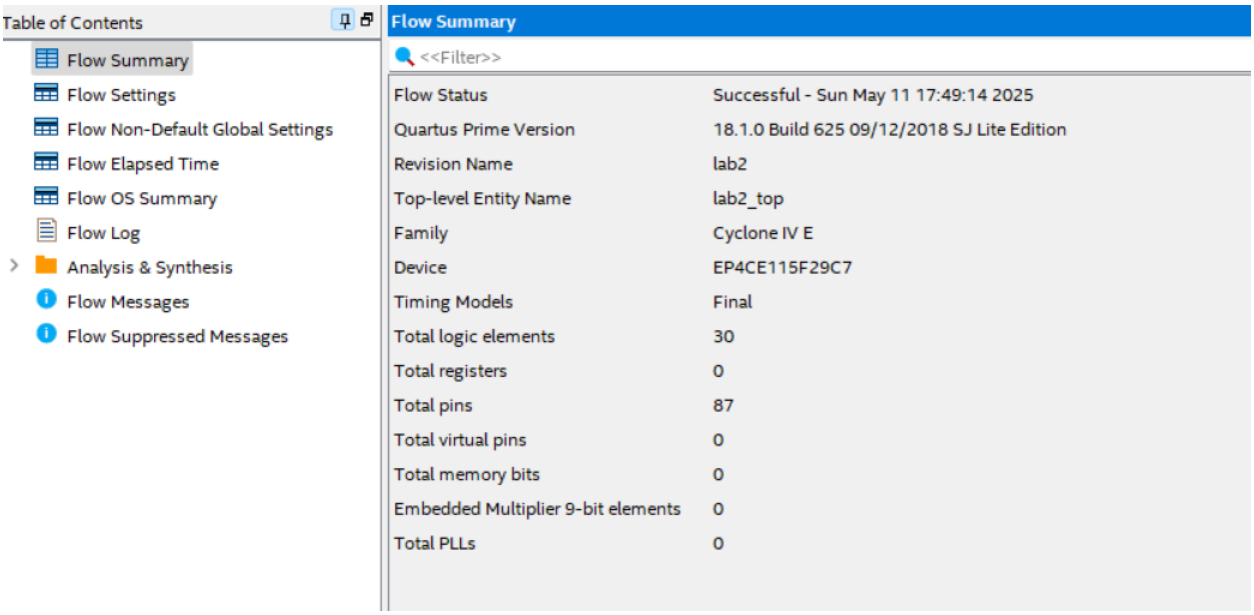
```
adder_subtractor_parametrizable #(4) uut (
    .a(a),
    .b(b),
    .k(k),
    .sum(sum),
    .cout(cout)
);
```

There is also a function that converts the 4-bit result into its signed equivalent in case the value is negative.

```
function signed [4:0] to_signed;
    input [3:0] value;
    begin
        if (value[3] == 1)
            to_signed = {1'b1, value}; // complemento a 2 negativo
        else
            to_signed = {1'b0, value};
    end
endfunction
```

The testbench prints headers for the result table that will appear in the simulation console. Each test block assigns values to a, b, and k, includes a 10 time unit delay (#10), and then prints the final result.

FPGA Implementation



Conclusion/Issues

Cesar Eduardo Inda Ceniceros

We successfully designed and implemented a 4-bit binary adder-subtractor circuit that performs both signed addition and subtraction using logical operators, half-adders, and full-adders. During the development, we concluded that using a switch (SW[4]) to select between addition and subtraction was more practical than a push-button, allowing for consistent and visible operation control. The circuit was verified through simulation and implemented on the DE2-115 FPGA board.

Alonso Emmanuel López Macias.

This lab reinforced hierarchical digital design principles, two's complement arithmetic, and FPGA interfacing. Additionally, we addressed the challenge of displaying negative results by assigning a dedicated 7-segment display to indicate the negative sign, ensuring clarity in signed decimal output. The process further strengthened our Verilog coding skills and debugging using simulation tools and RTL viewers.