ECE 440

Mealy FSM GCD

Lab #1 Report

Thomas Schmidt

March 3rd, 2025

Contents

1.1 Introduction

The purpose of this lab is to implement a Mealy FSM to compute the Greatest Common Divisor (GCD) of two 8-bit integers. The implementation utilizes Vivado 2016.4 for the design and simulation. The FSM progresses through different states, including loading input values, performing GCD computation, and providing the result.

1.2 Implementation

The FSM consists of four states: IDLE, LOAD$_A$, $LOAD_B, and COMPUTE. In the IDLE state, the FSM waits for the loadsign$

The following is the Verilog source code for the Mealy FSM GCD implementation. The FSM calculates the greatest common divisor (GCD) of two 8-bit numbers using the Euclidean algorithm.

```
`timescale 1ns / 1ps

module gcd_thread (
    input wire clk,
    input wire rst,
    input wire load,
    input wire [7:0] val_in,
    output reg [7:0] val_out,
    output reg done
);

    reg [7:0] A, B;
    reg [1:0] state;

    localparam IDLE = 2'b00, LOAD_A = 2'b01, LOAD_B = 2'b10, COMPUTE = 2'b11;

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            state <= IDLE;
            A <= 0;
            B <= 0;
            done <= 0;
            val_out <= 0;
        end else begin
            case (state)
                IDLE: begin
                    done <= 0;
                    if (load) state <= LOAD_A;
                end

                LOAD_A: begin
                    A <= val_in;
                    state <= LOAD_B;
                end

                LOAD_B: begin
                    B <= val_in;
                    state <= COMPUTE;
                end

                COMPUTE: begin
                    if (B == 0) begin
```

```
                        val_out <= A;
                        done <= 1;
                        state <= IDLE;
                    end else begin
                        A <= B;
                        B <= A % B;
                    end
                end
            endcase
        end
    end

endmodule
```

Listing 1.1: Mealy FSM GCD Implementation

1.3 Testing and Verification

The design was tested using a testbench that applies various input pairs to the FSM and monitors the output. The FSM was tested with multiple GCD pairs, such as (8, 20), (18, 45), and (28, 49), and the expected GCD results were verified. Behavioral simulation was performed in Vivado to observe the state transitions and output values. The state machine was also tested for edge cases, such as when one of the inputs is zero.

The following is the Verilog testbench used to verify the GCD FSM implementation. The testbench applies various input pairs and verifies the correct output of the GCD computation.

```
`timescale 1ns / 1ps

module simple_tb(

    );

    parameter CLK_PRD = 10;
    parameter HLD_TIME = 2;

    logic clk;
    initial begin
        clk = 0;
        forever #(CLK_PRD/2) clk = ~clk;
    end

    initial begin #(1000*CLK_PRD) $finish; end


    logic rst, load, done;
    logic [7:0] val_in;
    logic [7:0] val_out;
    gcd_thread dut(
    .clk(clk),
    .rst(rst),
    .val_in(val_in),
    .load(load),
    .val_out(val_out),
    .done(done));

    //gcd_thread dut(.*);
```

```
31
32    initial
33    begin
34        rst = 1;
35        load = 0;
36        val_in = 0;
37        #100;
38
39        @(posedge clk);
40        #HLD_TIME;
41        rst = 0;
42
43        #CLK_PRD;
44
45        load = 1;
46
47        #CLK_PRD;
48
49        load = 0;
50
51        val_in = 8;
52
53        #CLK_PRD;
54
55        val_in = 20;
56
57        #CLK_PRD;
58
59        val_in = 0;
60
61        @ (posedge done);
62
63        $display("The value of the GCD for 8 and 20 is %0d", val_out);
64
65        test_gcd(18, 45);
66
67        test_gcd(28, 49);
68
69        $finish;
70    end
71
72
73  task test_gcd(input [7:0] a, b);
74    @(posedge clk);
75    #HLD_TIME;
76    load = 1;
77    #CLK_PRD;
78    val_in = a;
79    load = 0;
80    #CLK_PRD;
81    val_in = b;
82    #CLK_PRD;
83    val_in = 0;
84    @(posedge done);
85    $display("The value of the GCD for %0d and %0d is %0d ", a, b, val_out);
86  endtask
87
```

```
88 endmodule
```

Listing 1.2: Testbench for Mealy FSM GCD

The following screenshots show the results of the behavioral simulation and post-synthesis simulation.

1.3.1 Behavioral Simulation

The behavioral simulation shows the state transitions and output values of the FSM during simulation. The state variables are visible in symbolic form.

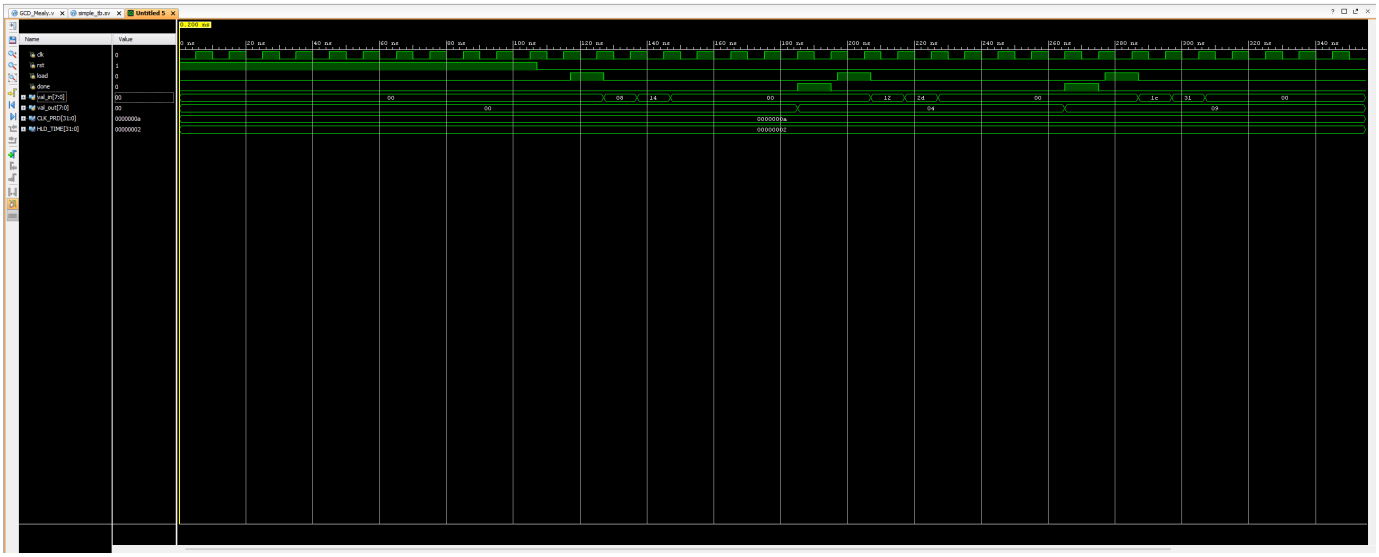

Figure 1.1: Behavioral Simulation showing state transitions and output values.

1.3.2 Post-Synthesis Simulation

The post-synthesis simulation shows the functionality of the GCD FSM after synthesis, ensuring that the design works as expected on the hardware.

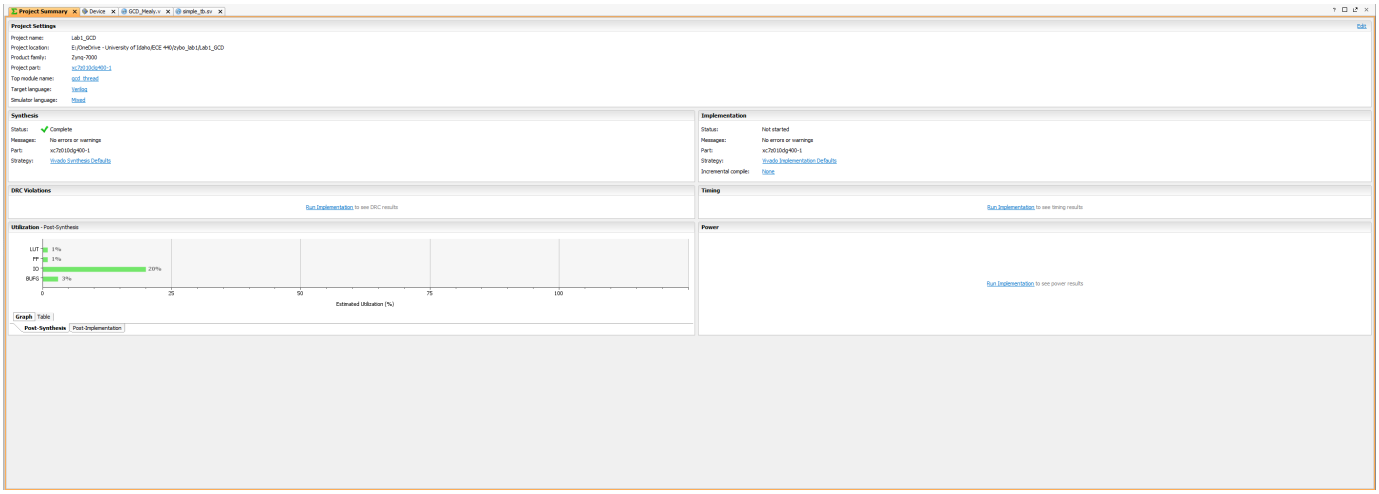

Figure 1.2: Post-synthesis simulation verifying the functionality of the design.

1.4 Conclusion

The Mealy FSM was successfully implemented and tested using Vivado 2016.4. All test cases passed, and the correct GCD values were produced. The FSM correctly transitioned between states, and the design was verified using behavioral and post-synthesis simulations. Future work could focus on optimizing the FSM for larger inputs or implementing a different algorithm for GCD computation.