Bilkent University

EE102-02 Lab 6 Report:

Greatest Common Divisor

Mehmet Emre Uncu - 22003884

Purpose:

This lab aims to design a circuit that calculates the greatest common divisor (GCD) of two 8-bit numbers.

Q&A:

- The Euclidian Algorithm is used to find GCD.
- The FSM is used in the design. It is a Moore machine. The FSM is slower and more expensive than a combinational circuit. The combinational circuits cannot store data. The FSM is slower and has more cost because it is clock based.
- It takes 53 clock cycles to calculate the GCD of 140 and 12. If it could be done, it would be quicker to transform the design into a combinational circuit, but we cannot, so we can optimize this design by reducing clock-based operations.

Methodology:

The Euclidean algorithm was used to find GCD (Figure 1). This algorithm subtracts the smaller number from the larger number and replaces the result with the larger number until the two numbers are equal. Each of the two equal numbers obtained due to these operations gives GCD.

A 2-state FSM was used to perform this operation. The system calculates if the start button is pressed and the reset button is not. As a result of the algorithm's calculations, the output is projected over the LEDs.

```
input: x, y
output: out

if x /= y then
    if x > y then
        x = x - y;
    else
        y = y - x;
else
    out <= x
```

Figure 1: The Euclidean Algorithm in VHDL

Design Specifications:

There are five inputs and one output. The clock input is fixed to 100MHz. The reset input is used to clear output. The start input for initiating the algorithm. The other two inputs are 8-bit numbers. The only output is the process result.

A 2-state Moore machine was used in the design:

- Starting state: In this state, if the start button is pressed, the number inputs are assigned as signal and move to the computation state.
- Computation state: Necessary calculations are made in this state and move to the starting state.

When GCD is found, FSM stays in the starting state, and LEDs project output. The schematic of the created design can be seen below:

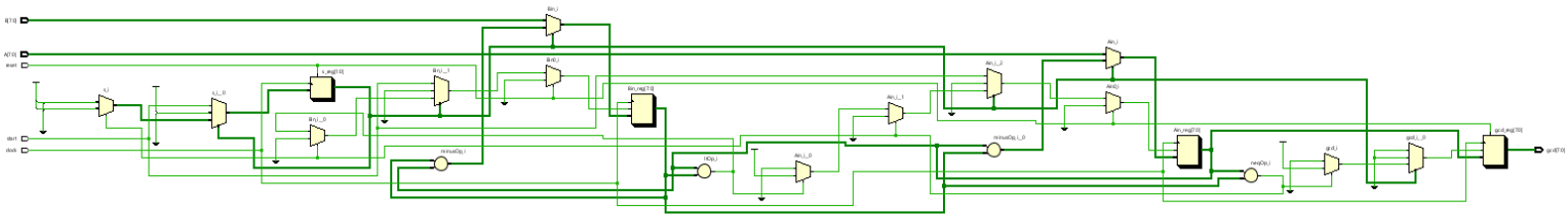

Figure 2: The Schematics of FSM

BASYS3 FPGA was programmed via a constraints file for implementation according to the following assignments:

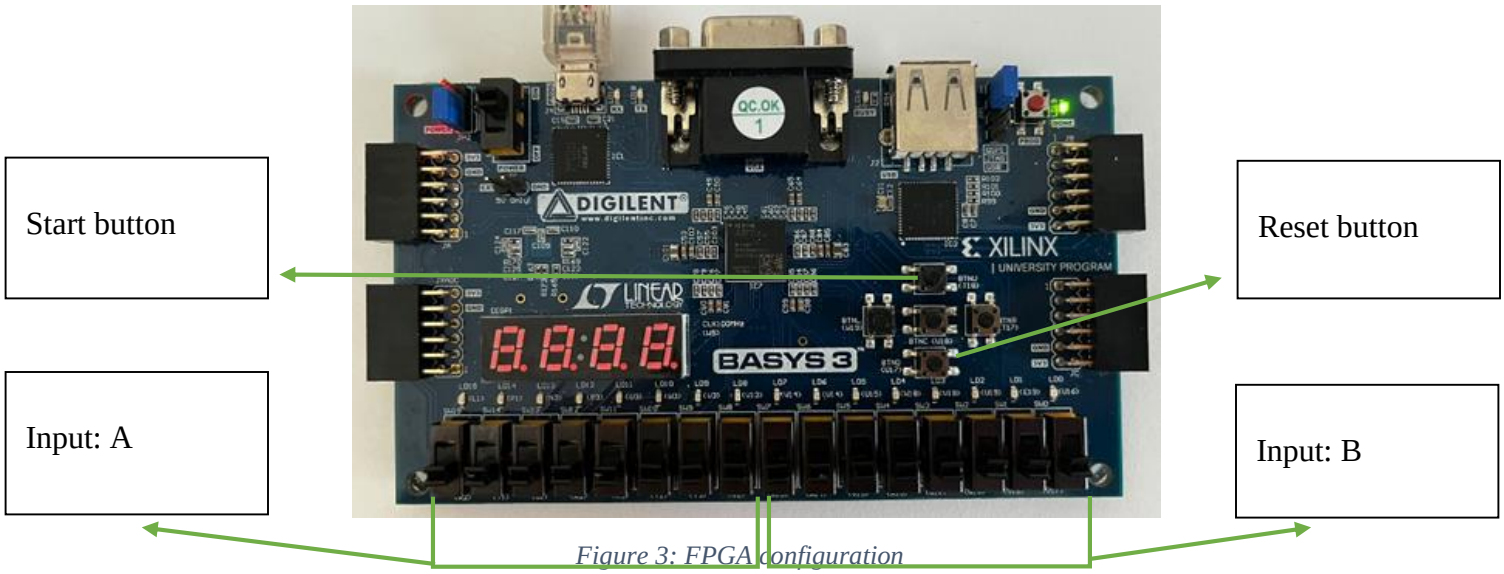

Figure 3: FPGA configuration

Results:

After the design, a testbench was written, and a waveform was created for the top module to check if it worked properly. Input A selected as "10001100" and input B as "00001100". The testbench gave the output "00000100" as a result (Figure 4).

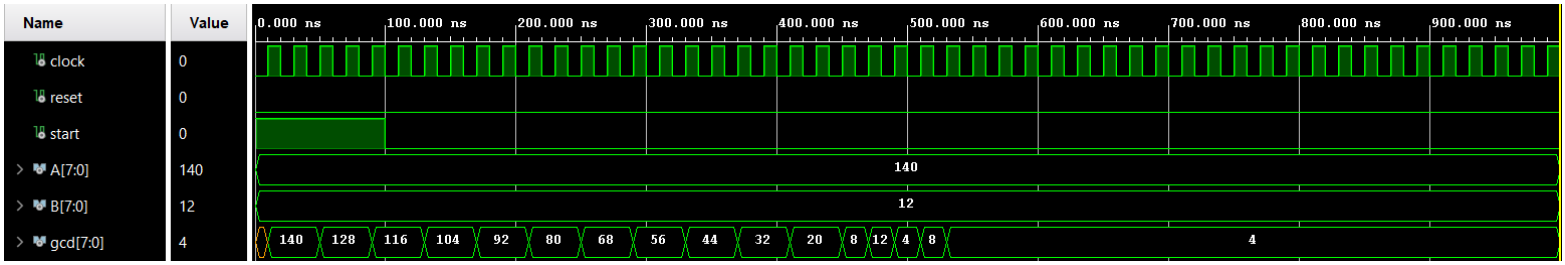

Figure 4: GCD of "10001100" and "00001100"

Lastly, the design was implemented on the FPGA board. Two examples after asserting the start button are as follows:

A = “11100000” = 224

B = “00011100” = 28

OUT = “00011100” = 28

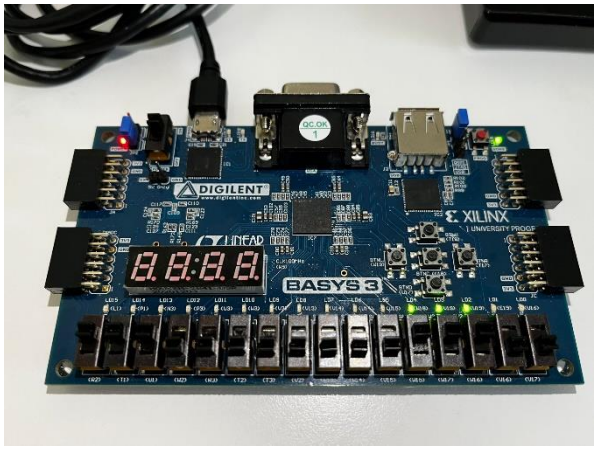

Figure 5: GCD of 224 and 28

A = “11101010” = 234

B = “01101100” = 108

OUT = “00010010” = 18

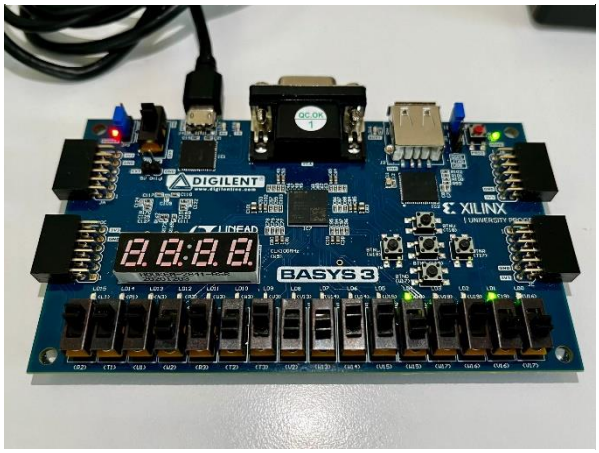

Figure 6: GCD of 234 and 108

Conclusion:

This lab aims to design a circuit that calculates the greatest common divisor (GCD) of two 8-bit numbers. With this lab, register usage and designing of FSM were learned. The outputs are compared with the simulation's waveform and the expected results, so the design is verified. No errors were encountered in this lab, except for human errors that occurred during the design. These errors were fixed by schematizing and simulating with the Vivado.

Appendix:

GCD.vhdl

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use work.all;

entity GCD is
    Port ( A, B : in unsigned(7 downto 0);
           reset, clock, start : in std_logic;
           gcd : out unsigned(7 downto 0) );
end GCD;

architecture Behavioral of GCD is
begin
    process(reset, clock)
    type state is (s0, s1, s2);
    variable s, cs: state;
    variable Ain, Bin: unsigned(7 downto 0);
    begin
        if reset = '1' then
            gcd <= "00000000";
            s := s0;
        elsif rising_edge(clock) then
            case cs is
                when s0 =>
                    if start = '1' then
                        Ain := A;
                        Bin := B;
                        s := s1;
```

```
                else
                    s := s0;
                end if;
            when s1 =>
                if Ain /= Bin then
                    if Ain < Bin then
                        Bin := Bin - Ain;
                    else
                        Ain := Ain - Bin;
                    end if;
                    s := s1;
                else
                    s := s0;
                end if;
        end case;
    end if;
  end process;
gcd <= Ain;
end Behavioral;
```

TESTBENCH

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use work.all;

entity GCDtest is
end GCDtest;

architecture Behavioral of GCDtest is
component gg
  Port ( A, B : in unsigned(7 downto 0);
```

```
        reset, clock, start : in std_logic;
        gcd : out unsigned (7 downto 0) );
end component;
signal A, B : unsigned(7 downto 0) := (others => '0');
signal reset, clock, start : std_logic := '0';
signal gcd :unsigned(7 downto 0);
begin
ggg: gg
   port map (A, B, reset, clock, start, gcd);
process
   begin
   clock <= '0';
   wait for 10 ns;
   clock <= '1';
   wait for 10 ns;
end process;

process
   begin
 start  <= '1';
 A <= "10001100";
 B <= "00001100";
 reset <= '0';
 wait for 100ns;
 start <= '0';
 wait;
end process;
end Behavioral;
```