Report to Lab 6: Greatest Common Divisor

Arda Özkut
22101727
Section 2
Physics Department, Ihsan Doğramacı Bilkent University
(Dated: November 28, 2022)

I. PURPOSE OF THE LAB ASSIGNMENT

Designing a logic circuit that inputs two 8 bit numbers and displays the greatest common divisor using a combination of sequential and circuit.

II. METHODOLOGY

The design is based upon Euler's Algorithm which basically states that $gcd(a, b) = gcd(a, b - a)$. Bellow is the algorithm for finding the GCD.

Algorithm 1 Euler's Algorithm

```
procedure COMPARE($a, b$)                              ▷ Check for Bigger Number
    if $a = b$ then
        $a \to a, b \to b$
    else if $a < b$ then
        $a \to b, b \to a$
    else
        $a \to a, b \to b$                             ▷ now we know that $a > b$
procedure FINITE SUBTRACTION(a,b)
    while $b \neq 0$ do
        $a - b \to b, b \to a$
        if b = 0 then
            a = GCD
```

To implement this algorithm I designed finite state Moore Machine with states no-swap, swap and subtract. I included a "set" and "reset" button. Reset, when pressed resets the the state and set starts the calculation with the inserted 8 bit numbers "a" and "b". The resulting number is than displayed on the LEDs.

I included an "OK" indicator that lights up when the machine is finished or hasn't started calculating.

III. DESIGN SPECIFICATIONS

Besides the clock which is fixed at 100Mhz (no under-clocking), the circuit has 4 input and 2 output signals. Other Inputs are "A" and "B", both 8 bit numbers, the reset signal and the signal for setting the numbers that are selected by the user. The RTL schematic can be seen in figure 1.

For the design I used 3 submodules that define the switching of the numbers and subtracting operations, and 1 top module that describes the registers.
The machine designed has 3 states.

- No-Swap : If a = b or a > b, no swapping operation is needed as the subtraction operation will not yield a negative number. Machine moves to the subtract state after this operation.
- Swap : If a = b machine moves to the No-Swap state. If a < b, swapping operation is needed as the subtraction operation will result in a negative operation. Machine moves to the subtract state after swapping operation.
- Subtract : After this state machine moves to the swap state.

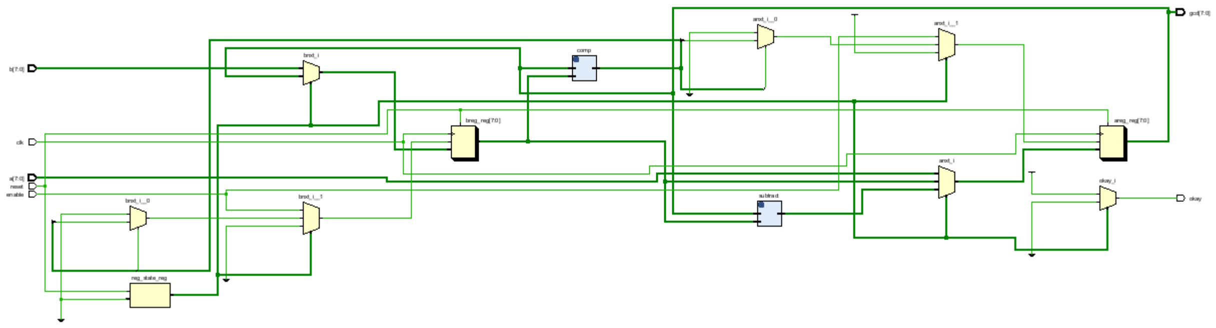

FIG. 1: RTL Schematic

To design the 8 bit subtractor [FIG.2] I borrowed my own code from the 6th Lab and used 8 of full subtractors.

To design the magnitude comparator [FIG. 3], I used 2 of cascadable comparators. I learned the design of the

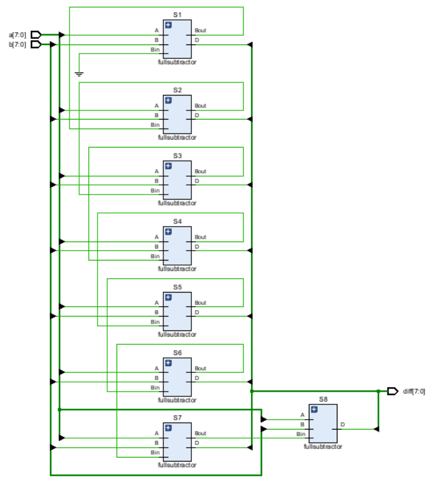

FIG. 2: 8 Bit Subtractor with 8 Full Subtractors

4 bit cascadable magnitude comparator from the data sheet [1] of an IC (SN54S85) that is manufactured by Texas Instruments. The circuit is designed so that if the two numbers are equal in magnitude the output is "010", if A is bigger than B : "100", if B is bigger than A : "001".

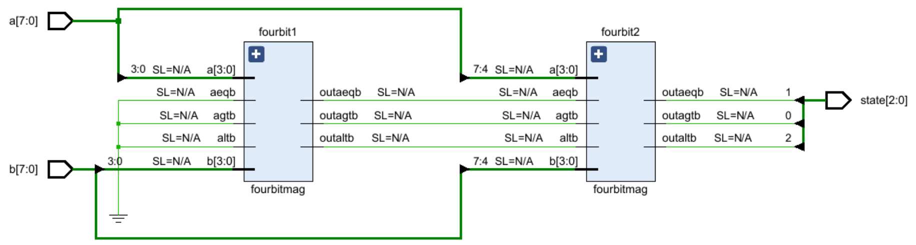

FIG. 3: 2x 4bit Cascadable Comparators

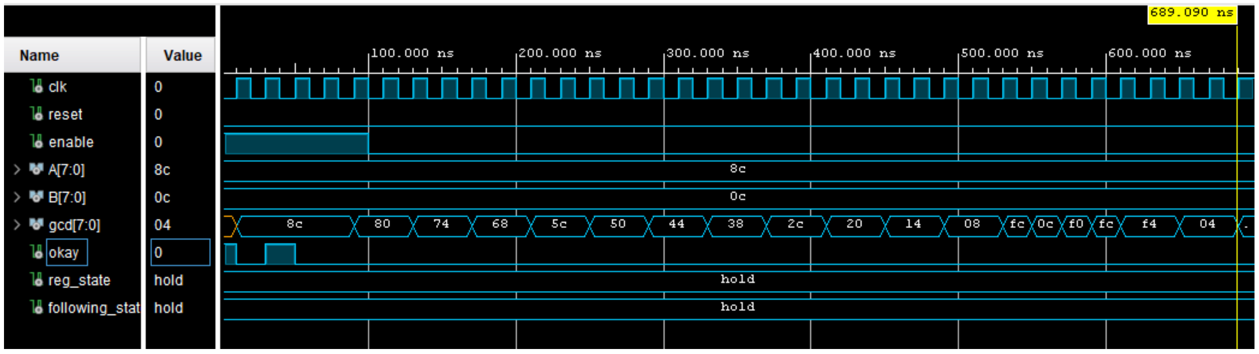

FIG. 4: Test Bench
Initial Values:
A:10001100, B:00001100
Out: 00000100

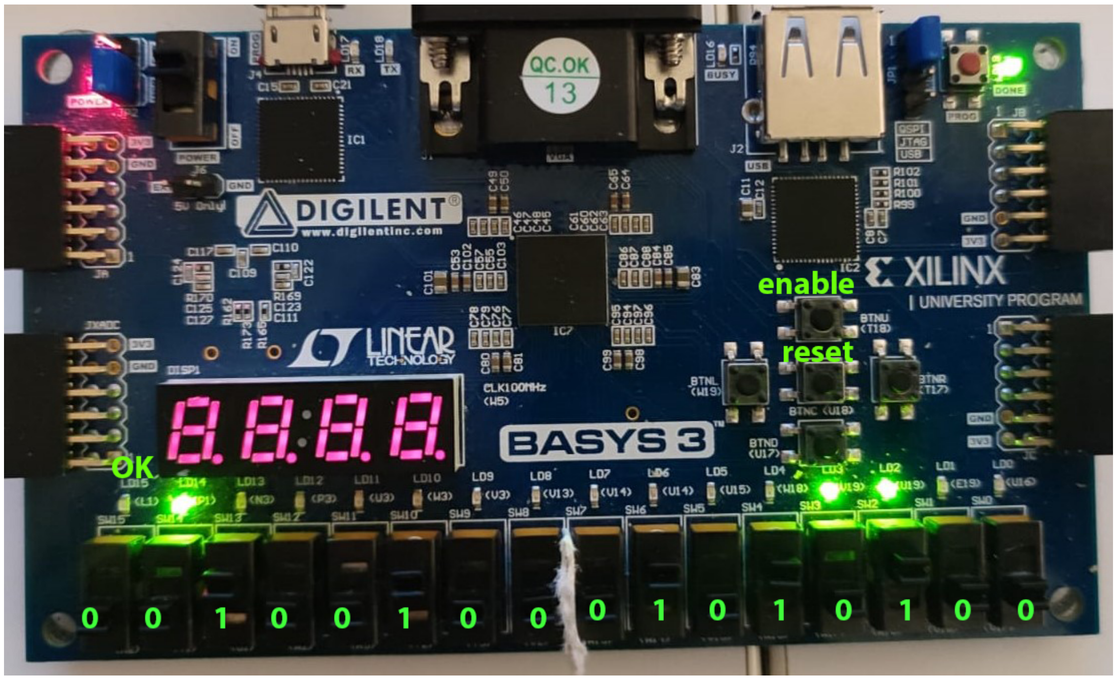

FIG. 5: Pin Configurations,
A:00100100, B:01010100
Out:00001100

IV. RESULTS

I have written a test bench for inputs 10001100 and 00001100. Calculated value for number of clock cycles taken for 140 and 12 is 26, however because the clock cycles in the test bench is not the same as the BASYS 3 board, there is different number of cycles in the test bench as can be observed in figure 4, specifically there is 32 cycles of the clock in the test bench. The number of clock cycles required for many calculations may be reduced if we use a divider(modulus) instead of a subtractor. The configuration can be seen in figure 5 and the pin configuration with the same inputs as the test bench is shown in figure 6.

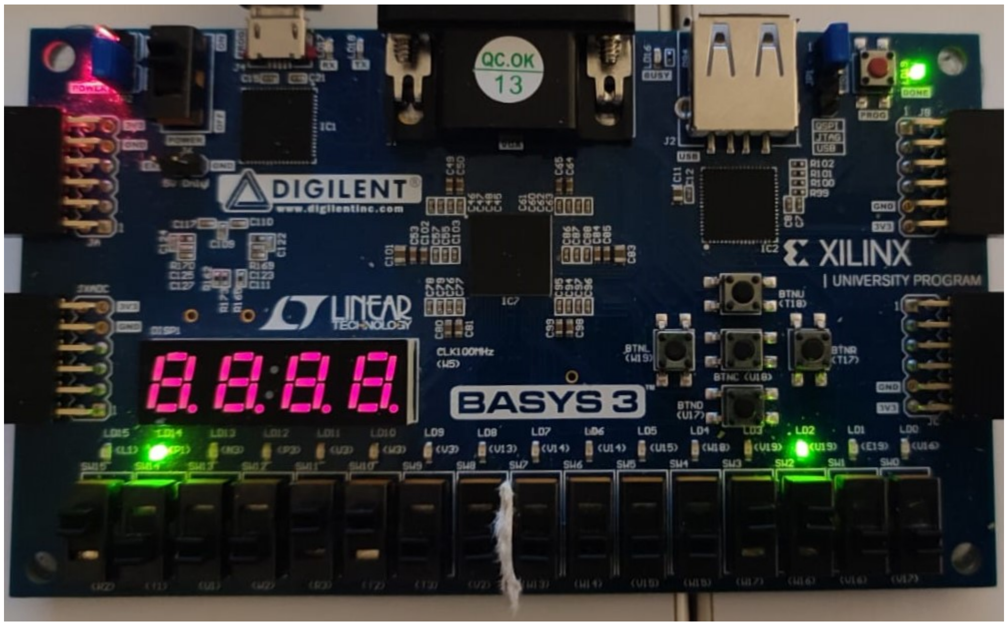

FIG. 6: A:10001100, B:00001100
Out: 00000100

V. CONCLUSION

The purpose of this lab was to design a finite state machine to fing the greatest common divisor of two 8-bit numbers. For designing such a circuit I used the simplified version of Euclid's Algorithm for finding the greatest common divisor. The implementation was designed as a sequential circuit, however a combinational design approach would be faster and better in time performance compared to a finite state machine. Problem with the combinational approach would be that since combinational circuits cannot maintain voltage levels (hold data) the designing of such a circuit for multiple inputs (such as 2x 8-bit numbers and 2 signals, as in the case of this lab) would be really hard, time consuming and also it would take more space on the chip and would be more expensive. This is why for less input operations such as subtraction and multiplication, a combinational design is chosen since it it faster.

Implementing the same functionality as a Meally machine would do the same task in a smaller time compared to the Moore machine.

VI. REFERENCES

[1] - "4-bit magnitude comparators datasheet - texas instruments." [Online].
Available: https://www.ti.com/lit/ds/symlink/sn54ls85.pdf. [Accessed: 28-Nov-2022].

VII. APPENDIX

A. Code

1. *Top Level (main_gcd.vhd)*

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
entity main_gcd is
    port(
        clk, reset: in std_logic;
        enable: in std_logic;
        a, b: in std_logic_vector(7 downto 0);
        okay: out std_logic;
        gcd: out std_logic_vector(7 downto 0)
    );
end main_gcd ;
architecture rtl_gcd of main_gcd is
    type state_type is (hold, replace, substract);
    signal register_state, next_state : state_type;

    signal register_a, register_b, next_a, next_b: std_logic_vector(7 downto 0);
    signal a_compared,b_compared : std_logic_vector(7 downto 0);

    signal subs_b, subs_a, subs_out: std_logic_vector(7 downto 0);
    signal compared: std_logic_vector(2 downto 0);

    component comparator is
        port( a, b : in  std_logic_vector ( 7 downto 0);
             state : buffer std_logic_vector ( 2 downto 0));
    end component;

    component subtractor is
        port( a, b : in  std_logic_vector ( 7 downto 0);
             diff : buffer std_logic_vector ( 7 downto 0));
    end component;
begin

    comp: comparator
        port map(a => a_compared, b=> b_compared, state=> compared);
    subt: subtractor
        port map(a => subs_a, b => subs_b, diff => subs_out);

    process(clk,reset)
    begin
        if reset='1' then
            register_state <= hold;
            register_a <= (others=>'0');
            register_b <= (others=>'0');
        elsif (rising_edge(clk)) then
            register_state <= next_state;
            register_a <= next_a;
            register_b <= next_b;
        end if;
    end process;
    process(register_state,register_a,register_b,enable,a,b)
    begin
        next_a <= register_a;
        next_b <= register_b;
        a_compared <= register_a;
        b_compared <= register_b;
        subs_a <= std_logic_vector(register_a);
        subs_b <= std_logic_vector(register_b);

        case register_state is
            when hold =>
                if enable = '1' then
                    next_a <= std_logic_vector(a);
                    next_b <= std_logic_vector(b);
                    next_state <= replace;
                else
                    next_state <= hold;
                end if;
            when replace =>
                if (compared(1) = '1') then
                    next_state <= hold;
                else
                    if(compared(2) = '1') then
                        next_a <= register_b;
                        next_b <= register_a;
                    end if;
                    next_state <= substract;
                end if;
            when substract =>
                next_a <= std_logic_vector(subs_out);
                next_state <= replace;
        end case;
    end process;
    okay <= '1' when register_state = hold else '0';
    gcd <= std_logic_vector(register_a);
end rtl_gcd;
```

2. Comparator

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity comparator_module is
    port( a,b : in std_logic_vector(7 downto 0);
          state : out std_logic_vector(2 downto 0)
        );

end comparator_module;

architecture compare of comparator_module is
    signal eq1, lt1, gt1, eq2, lt2, gt2 : std_logic;
begin
    fourbit1 : entity work.fourbitmag(magnitude)
        port map (
            a(3 downto 0),
            b(3 downto 0),
            '0',
            '0',
            '0',
            eq1,
            lt1,
            gt1
        );

    fourbit2 : entity work.fourbitmag(magnitude)
        port map(
            a(7 downto 4),
            b(7 downto 4),
            eq1,
            lt1,
            gt1,
            eq2,
            lt2,
            gt2
        );

    state(1) <= eq2;
    state(2) <= lt2;
    state(0) <= gt2;

end compare;
```

3. 4-bit Comparator

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity fourbitmag is
    port(a,b : in std_logic_vector(3 downto 0);
         aeqb, altb, agtb : in std_logic;
         outaeqb, outaltb, outagtb : buffer std_logic
        );
end fourbitmag;

architecture magnitude of fourbitmag is
    signal i0,i1,i2,i3 : std_logic;
    signal saeqb, saltb, sagtb : std_logic;
begin
    i0 <= a(0) xor b(0);
    i1 <= a(1) xor b(1);
    i2 <= a(2) xor b(2);
    i3 <= a(3) xor b(3);

    saeqb <= i3 and i2 and i1 and i0;
    sagtb <= (a(0) and not b(0) and i3 and i2 and i1)
    or (a(1) and not b(1) and i2 and i3) or (not b(2) and
    a(2) and i3) or (a(3) and not b(3));
    saltb <= saeqb nand sagtb;
    outaeqb <= saeqb and aeqb;
    outagtb <= (saeqb and agtb) or sagtb;
    outaltb <= (saeqb and altb) or saltb;

end magnitude;
```

4. Subtractor

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

entity substractor is
    port( a,b : in std_logic_vector(7 downto 0);
         diff : out std_logic_vector(7 downto 0));
end substractor;

architecture substract of substractor is
    component fullsubtractor is
        Port ( A : in STD_LOGIC;
             B : in STD_LOGIC;
             Bin : in STD_LOGIC;
             D : out STD_LOGIC;
             Bout : out STD_LOGIC);
    end component;
signal b0,b1,b2,b3,b4,b5,b6,b7 : std_logic;
begin
S1: fullsubtractor port map(a(0),b(0),'0',diff(0),b0);
S2: fullsubtractor port map(a(1),b(1),b0,diff(1),b1);
S3: fullsubtractor port map(a(2),b(2),b1,diff(2),b2);
S4: fullsubtractor port map(a(3),b(3),b2,diff(3),b3);
S5: fullsubtractor port map(a(4),b(4),b3,diff(4),b4);
S6: fullsubtractor port map(a(5),b(5),b4,diff(5),b5);
S7: fullsubtractor port map(a(6),b(6),b5,diff(6),b6);
S8: fullsubtractor port map(a(7),b(7),b6,diff(7),b7);


end substract;
```