EEE 102 Digital Design - GCD Calculator

Introduction:

In this lab, I implemented Stein's algorithm for two 8-bit numbers in order to calculate their "Greatest Common Divisor". The circuit I've designed can be considered a Moore Machine as it only depends on the initial inputs and not the current inputs.

Methodology:

The design of this system is based on three registers. Two of these registers are for the input signals and the third is for the output. These registers are held in place or updated or loaded according to the algorithm. I have summarised the algorithm and my implementation as follows:

If both signals are even:

The registers shift right by one and a '0' is loaded into the result register.

If one of them is even:

Only the even signal's register shifts right and other registers are held.

If both are odd and equal:

The signals that are equal are shifted and as they are outputted from their respective registers, they are being inputted to the output's register.

If both are odd but not even:

The lesser signal is subtracted from the greater signal and all the registers are being held.

If both signals are 0:

The system holds all registers indefinitely and the digits inside the register are processed to receive the GCD signal.

Questions and Answers

- The algorithm used to calculate the greatest common divisor of two 8-bit numbers is "Stein's Algorithm".
- This circuit can be considered a Moore machine as it does not depend on the current inputs rather it only depends on the initial inputs.
- Using my implementation, it took 17 cycles to calculate the greatest common divider of 140 and 12. This could have been optimized because my algorithm also takes a clock cycle to subtract the signals and another cycle to load them into the registers. This could be done without taking a clock cycle.

Results:

In the end, the circuit takes two 8-bit signals(x1,x2) and outputs two signals (out1,out2). “out1” is an 8-bit output and gives the GCD of these signals. “out2” is for debugging purposes and has no specific purpose in the calculation process.

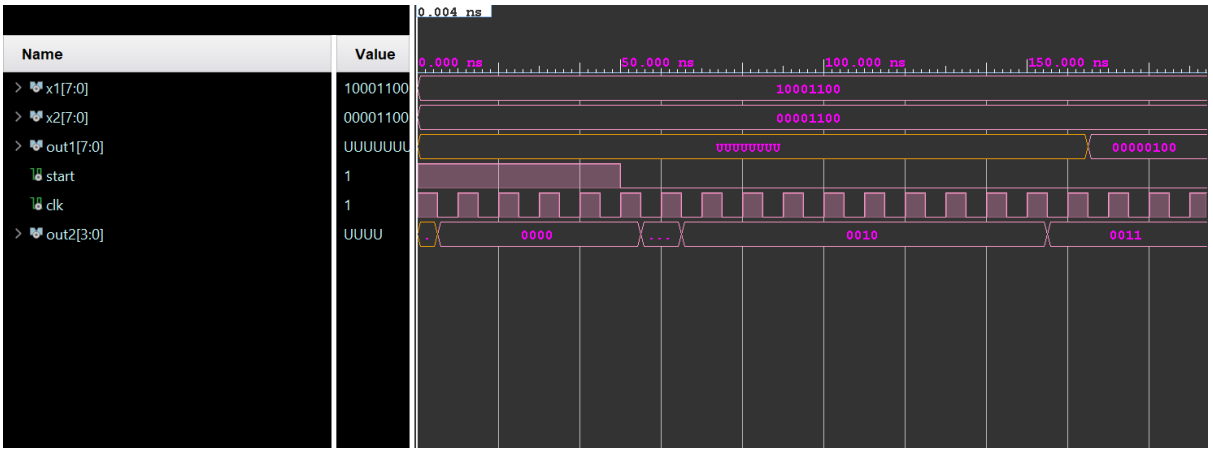

Figure 1: Testbench gcd(140,12)

Main.vhd:

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


entity main is
  Port (x1,x2 : in std_logic_vector(7 downto 0);
        out1  : out std_logic_vector(7 downto 0);
        out2,out3 : out std_logic_vector(3 downto 0);
        start,clk : in std_logic
        );
end main;

architecture Behavioral of main is
    component reg is
        Port (L : in std_logic_vector(7 downto 0);
              out1 : out std_logic;
              out2 : out std_logic_vector(7 downto 0);
              slk,s,clk : in std_logic);
    end component;

    component subtractor is
        Port (x,y : in std_logic_vector(7 downto 0);
              out1 : out std_logic_vector(7 downto 0)
```

```
        );
    end component;

    signal L1,L2,L3 : std_logic_vector(7 downto 0);
    signal slk1,slk2,slk3 : std_logic;
    signal clk1,clk2,clk3 : std_logic;
    signal d1,d2,d3,d3_in : std_logic;
    signal b1,b2,b3 : std_logic_vector(7 downto 0);
    signal hold1,hold2,hold3 : std_logic;
    signal sub_b : std_logic_vector(7 downto 0);
    signal count,count2 : std_logic_vector(3 downto 0); –debugging
    signal sol : std_logic_vector(7 downto 0);
    signal ekz: std_logic; –debugging
begin
    reg1 : reg port map (L => L1 , slk => slk1 , clk => clk1 , out1 => d1 , out2 => b1 , s => '0');
    reg2 : reg port map (L => L2 , slk => slk2 , clk => clk2 , out1 => d2 , out2 => b2 , s => '0');
    reg3 : reg port map (L => L3 , slk => slk3 , clk => clk3 , out1 => d3 , out2 => b3 , s => d3_in);
    sub1 : subtractor port map(x => b1, y => b2, out1 => sub_b);

    clk1 <= clk and not hold1;
    clk2 <= clk and not hold2;
    clk3 <= clk and not hold3;

    process(clk) begin
    if falling_edge(clk) then
       if start = '1' then
          slk1 <= '1';
          slk2 <= '1';
          slk3 <= '1';
          L1 <= x1;
          L2 <= x2;
          count <= "0000";
          count2 <= "0000";
          L3 <= "11111111";
          hold1 <= '0';
          hold2 <= '0';
          hold3 <= '0';
          ekz <= '0';
       else


          if (b1 = "00000000") and (b2 = "00000000") then
          hold1 <= '1'; hold2 <= '1'; hold3 <= '1';
          slk1  <= '0'; slk2  <= '0'; slk3  <= '0';

          case count is
             when "0000" => out1 <= "00000000";
             when "0001" => out1 <= "0000000" & b3(7);
             when "0010" => out1 <= "000000" & b3(7 downto 6);
```

```
            when "0011" => out1 <= "00000" & b3(7 downto 5);
            when "0100" => out1 <= "0000" & b3(7 downto 4);
            when "0101" => out1 <= "000" & b3(7 downto 3);
            when "0110" => out1 <= "00" & b3(7 downto 2);
            when "0111" => out1 <= "0" & b3(7 downto 1);
            when others => out1 <= b3;
        end case;

        elsif (d1 = '0' and d2 = '0') then
        hold1 <= '0'; hold2 <= '0'; hold3 <= '0';
        slk1  <= '0'; slk2  <= '0'; slk3  <= '0';
        d3_in <= '0';
        count2 <= count2 + '1';
        count <= count + '1';

        elsif (d1 = '0' and d2 = '1') then
        hold1 <= '0'; hold2 <= '1'; hold3 <= '1';
        slk1  <= '0'; slk2  <= '0'; slk3  <= '0';

        elsif (d1 = '1' and d2 = '0') then
        hold1 <= '1'; hold2 <= '0'; hold3 <= '1';
        slk1  <= '0'; slk2  <= '0'; slk3  <= '0';

        elsif b1 = b2 then
        hold1 <= '0'; hold2 <= '0'; hold3 <= '0';
        slk1  <= '0'; slk2  <= '0'; slk3  <= '0';
        d3_in <= d1;
        count <= count + '1';

        elsif b1 > b2 then
        hold1 <= '0'; hold2 <= '1'; hold3 <= '1';
        slk1  <= '1'; slk2  <= '0'; slk3  <= '0';
        L1 <= sub_b;


        elsif b1 < b2 then
        hold1 <= '1'; hold2 <= '0'; hold3 <= '1';
        slk1  <= '0'; slk2  <= '1'; slk3  <= '0';
        L2 <= sub_b;

        end if;
      end if;
   end if;
   end process;
   out3 <= count2;
   out2 <= count;
end Behavioral;
```

Cons.xdl

```
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

set_property PACKAGE_PIN W19 [get_ports start]
set_property IOSTANDARD LVCMOS33 [get_ports start]

set_property PACKAGE_PIN V17 [get_ports {x1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x1[0]}]
set_property PACKAGE_PIN V16 [get_ports {x1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x1[1]}]
set_property PACKAGE_PIN W16 [get_ports {x1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x1[2]}]
set_property PACKAGE_PIN W17 [get_ports {x1[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x1[3]}]
set_property PACKAGE_PIN W15 [get_ports {x1[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x1[4]}]
set_property PACKAGE_PIN V15 [get_ports {x1[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x1[5]}]
set_property PACKAGE_PIN W14 [get_ports {x1[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x1[6]}]
set_property PACKAGE_PIN W13 [get_ports {x1[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x1[7]}]

set_property PACKAGE_PIN V2 [get_ports {x2[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x2[0]}]
set_property PACKAGE_PIN T3 [get_ports {x2[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x2[1]}]
set_property PACKAGE_PIN T2 [get_ports {x2[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x2[2]}]
set_property PACKAGE_PIN R3 [get_ports {x2[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x2[3]}]
set_property PACKAGE_PIN W2 [get_ports {x2[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x2[4]}]
set_property PACKAGE_PIN U1 [get_ports {x2[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x2[5]}]
set_property PACKAGE_PIN T1 [get_ports {x2[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x2[6]}]
set_property PACKAGE_PIN R2 [get_ports {x2[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {x2[7]}]
# LEDs

set_property PACKAGE_PIN U16 [get_ports {out1[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out1[0]}]
set_property PACKAGE_PIN E19 [get_ports {out1[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out1[1]}]
set_property PACKAGE_PIN U19 [get_ports {out1[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out1[2]}]
set_property PACKAGE_PIN V19 [get_ports {out1[3]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {out1[3]}]
set_property PACKAGE_PIN W18 [get_ports {out1[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out1[4]}]
set_property PACKAGE_PIN U15 [get_ports {out1[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out1[5]}]
set_property PACKAGE_PIN U14 [get_ports {out1[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out1[6]}]
set_property PACKAGE_PIN V14 [get_ports {out1[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out1[7]}]

set_property PACKAGE_PIN L1 [get_ports {out2[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out2[3]}]
set_property PACKAGE_PIN P1 [get_ports {out2[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out2[2]}]
set_property PACKAGE_PIN N3 [get_ports {out2[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out2[1]}]
set_property PACKAGE_PIN P3 [get_ports {out2[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out2[0]}]


set_property PACKAGE_PIN U3 [get_ports {out3[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out3[3]}]
set_property PACKAGE_PIN W3 [get_ports {out3[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out3[2]}]
set_property PACKAGE_PIN V3 [get_ports {out3[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out3[1]}]
set_property PACKAGE_PIN V13 [get_ports {out3[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out3[0]}]
```

---

main_tb.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity main_tb is

end main_tb;

architecture Behavioral of main_tb is
    component main is
     Port (x1,x2 : in std_logic_vector(7 downto 0);
        out1  :out std_logic_vector(7 downto 0);
        out2,out3  : out std_logic_vector(3 downto 0);
        start,clk : in std_logic
        );
    end component;

    signal x1,x2,out1 : std_logic_vector(7 downto 0);
    signal start, clk : std_logic;
    signal out2 : std_logic_vector(3 downto 0);
```

```
begin
   m1 : main port map (x1 => x1, x2 => x2, out1 => out1, start => start, clk => clk , out2 => out2);

   process begin
   clk <= '1';
   wait for 5ns;
   clk <= '0';
   wait for 5ns;
   end process;

   process begin


   x1 <= "10001100";
   x2 <= "00001100";
   start <= '1';
   wait for 50ns;
   start <= '0';
   wait for 500ns;

   end process;

end Behavioral;
```