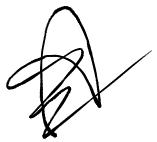
“I certify that this submission is my original work and meets the Faculty’s Expectations of Originality”.

Andre Hei Wang Law, 4017 5600

Monday, November 15, 2021



Experiment 4 (FJ-X)

**1) VHDL Source Code**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_unsigned.all;

use IEEE.numeric\_std.all;

entity registers\_min\_max is

port (

din: in std\_logic\_vector(3 downto 0);

reset: in std\_logic;

clk: in std\_logic;

sel: in std\_logic\_vector(1 downto 0);

max\_out: out std\_logic\_vector(3 downto 0);

min\_out: out std\_logic\_vector(3 downto 0);

reg\_out: out std\_logic\_vector(3 downto 0));

end registers\_min\_max;

architecture reg\_min\_max of registers\_min\_max is

type reg\_array is array (integer range <>) of unsigned(3 downto 0);

signal regist: reg\_array(3 downto 0);

signal max\_regist, min\_regist: unsigned(3 downto 0);

signal save\_max\_regist, save\_min\_regist: unsigned(3 downto 0);

begin

process(regist)

variable max: unsigned(3 downto 0) := "0000";

variable min: unsigned(3 downto 0) := "1111";

begin

for index in regist'high downto 1 loop

if (min > regist(index)) then

min := regist(index);

end if;

if (max < regist(index)) then

max := regist(index);

end if;

end loop;

save\_max\_regist <= max;

save\_min\_regist <= min;

end process;

process(din,reset,clk)

begin

if (reset = '1') then

regist(3) <= "1000";

regist(2) <= "1000";

regist(1) <= "1000";

regist(0) <= "1000";

elsif (rising\_edge(clk)) then

regist(0) <= regist(1);

regist(1) <= regist(2);

regist(2) <= regist(3);

regist(3) <= unsigned(din);

end if;

end process;

with sel select reg\_out <=

std\_logic\_vector(regist(3)) when "00",

std\_logic\_vector(regist(2)) when "01",

std\_logic\_vector(regist(1)) when "10",

std\_logic\_vector(regist(0)) when "11",

"0000" when others;

process(clk,reset,max\_regist,min\_regist,save\_min\_regist,save\_max\_regist)

begin

if (reset = '1') then

max\_regist <= "0000";

min\_regist <= "1111";

elsif rising\_edge(clk) then

if (save\_max\_regist > max\_regist and max\_regist(0) /= 'U') then

max\_regist <= save\_max\_regist;

end if;

if (save\_min\_regist < min\_regist and min\_regist(0) /= 'U') then

min\_regist <= save\_min\_regist;

end if;

end if;

end process;

max\_out <= std\_logic\_vector(max\_regist);

min\_out <= std\_logic\_vector(min\_regist);

end reg\_min\_max;

**2) Xilinx Vivado Implementation Log File**

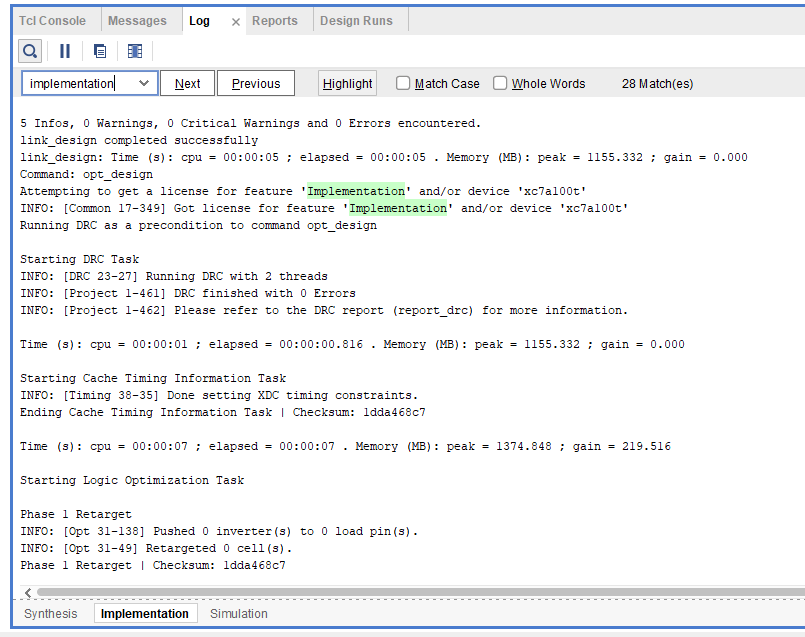


Figure 1. Beginning of the Implementation of the Log File

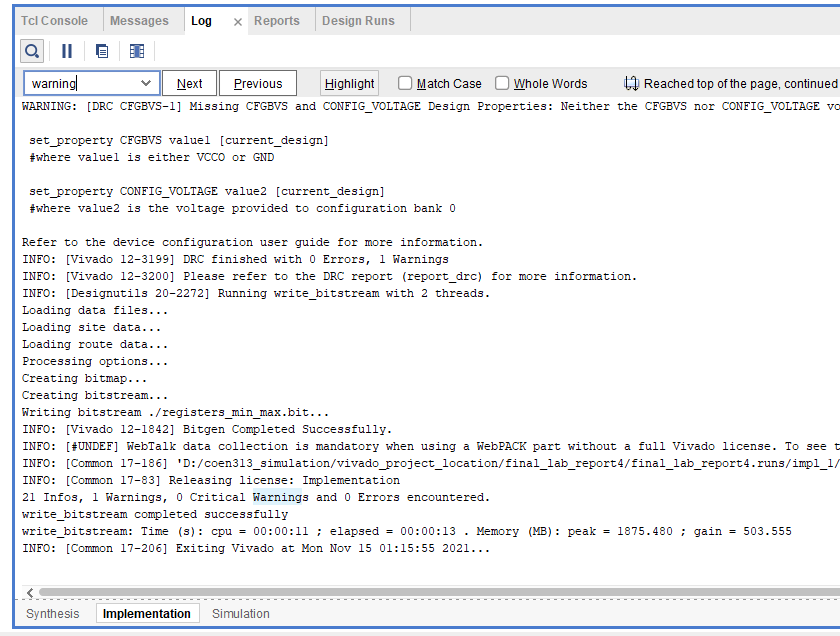


Figure 2. End of the Implementation of the Log File

**3) RTL Elaborated and Implemented Schematic Diagrams as Produced by Xilinx Vivado**

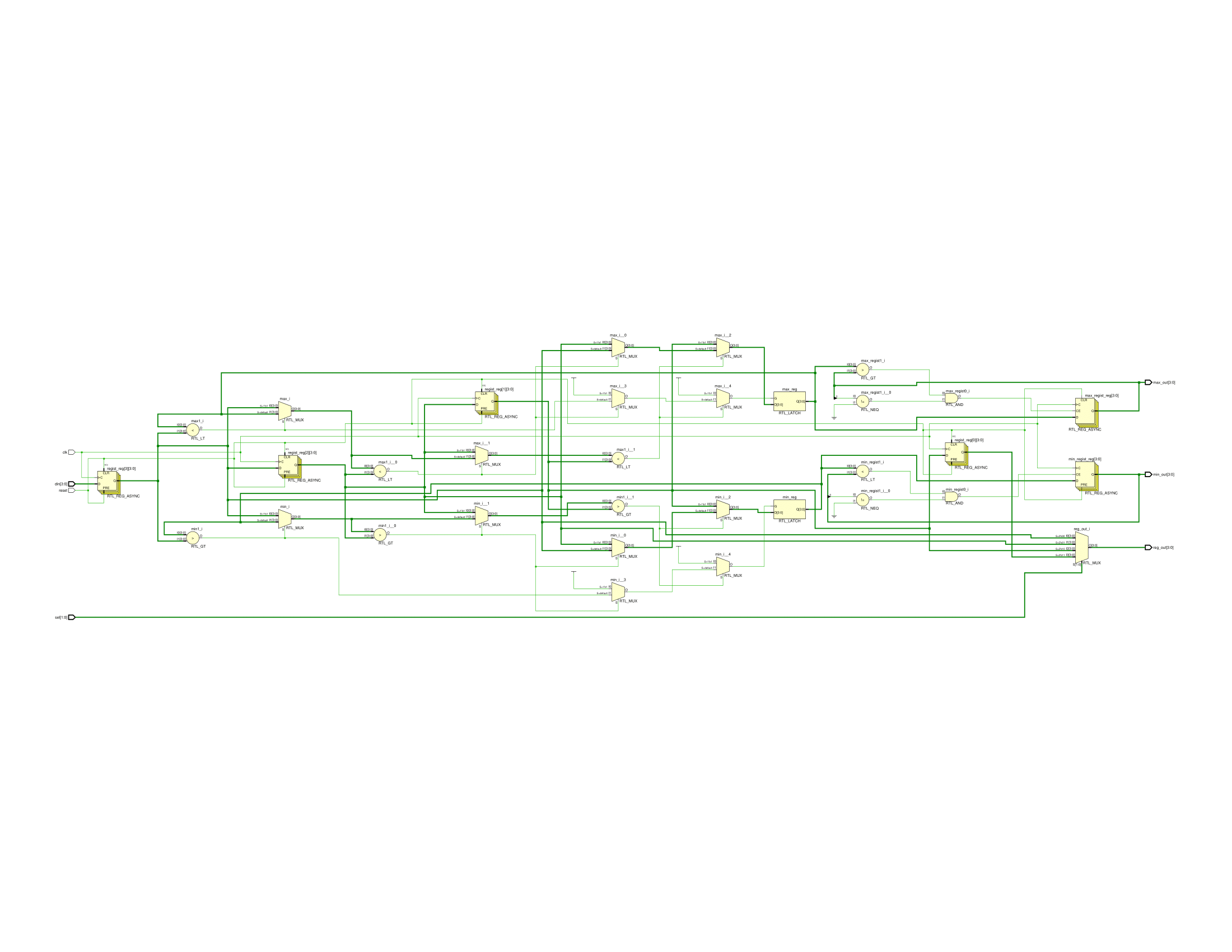


Figure 3. Elaborated Schematic Design of RTL Analysis

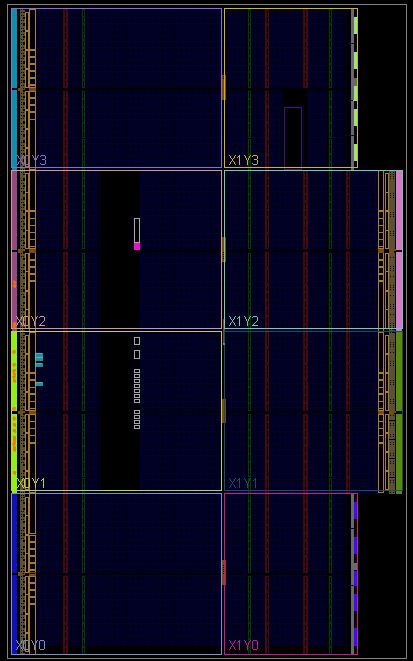
****

Figure 4. Implemented Schematic Diagram Design View

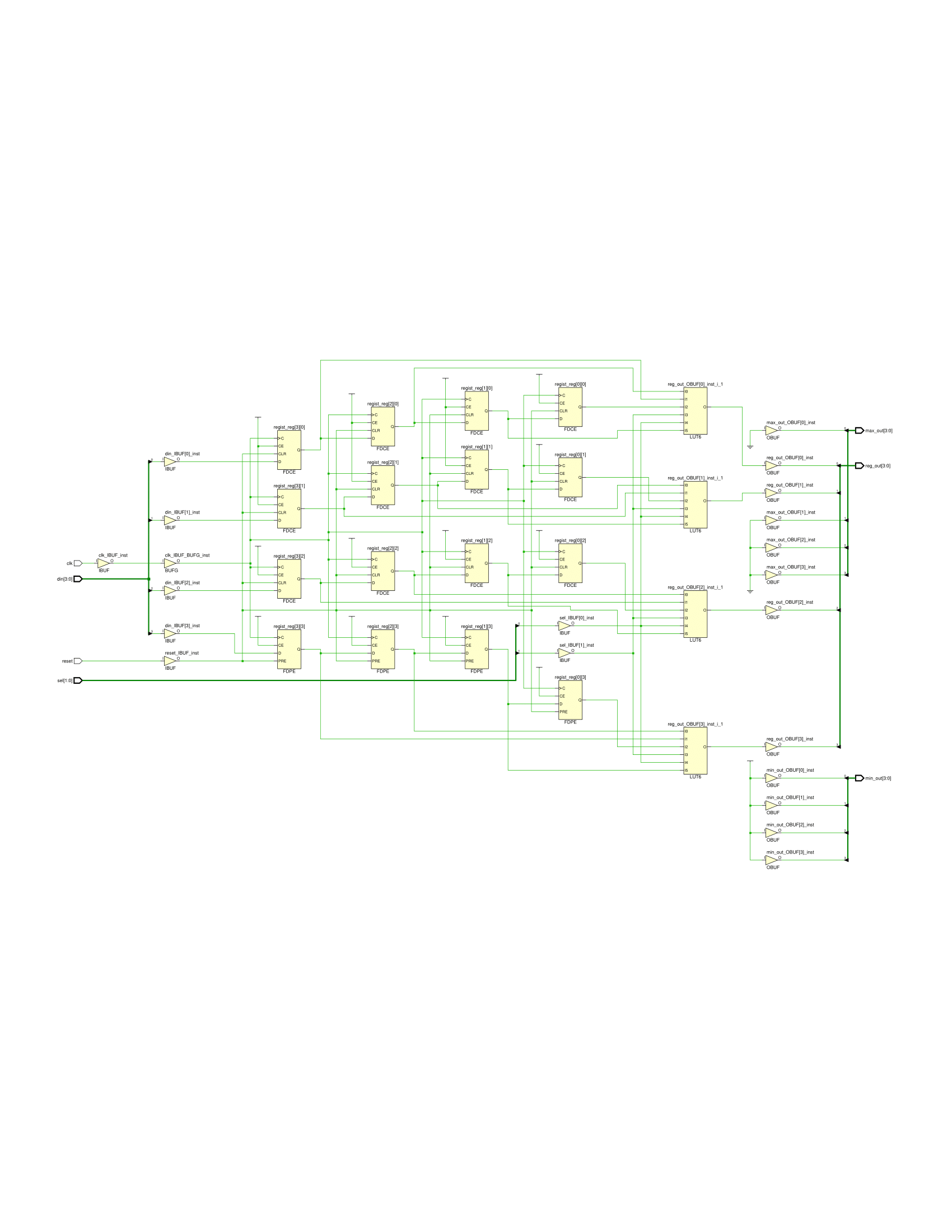


Figure 5. Schematic Diagram Representation of the Implemented Design

**4) Modelsim Simulation Results with Judicious Choice of Test Values**

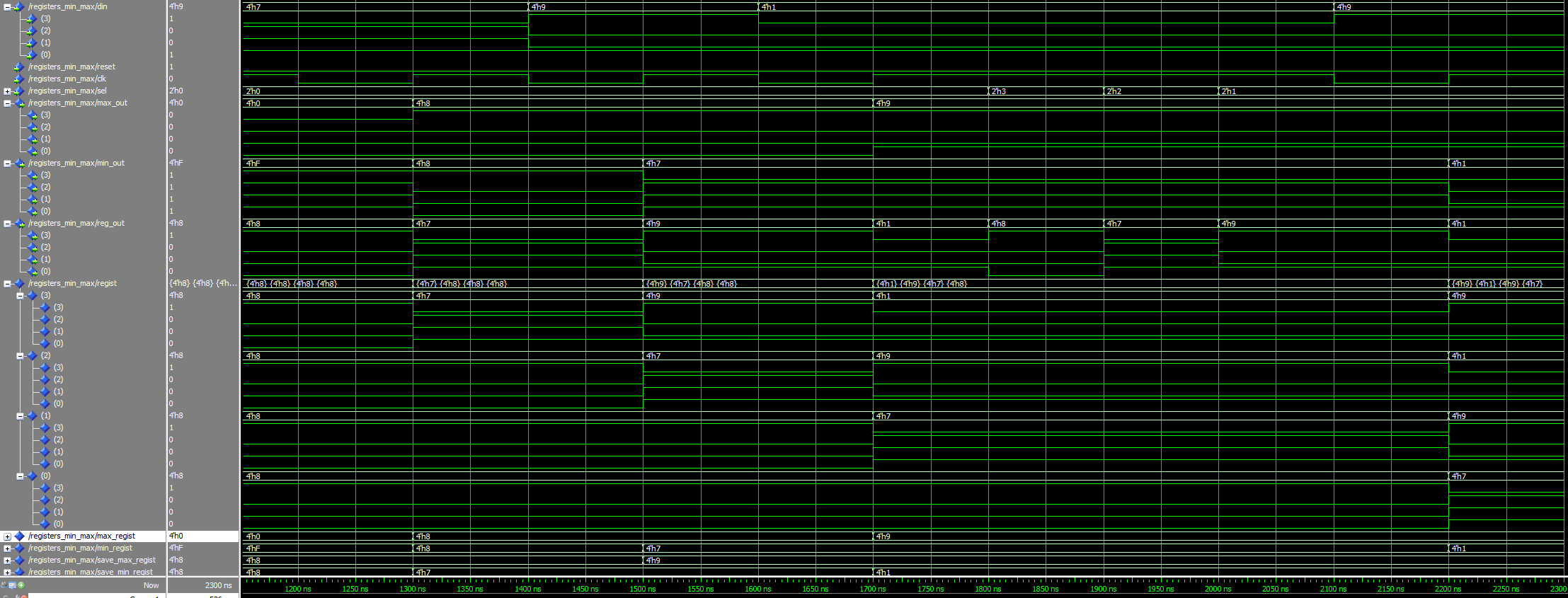


Figure 6. Modelsim Simulation

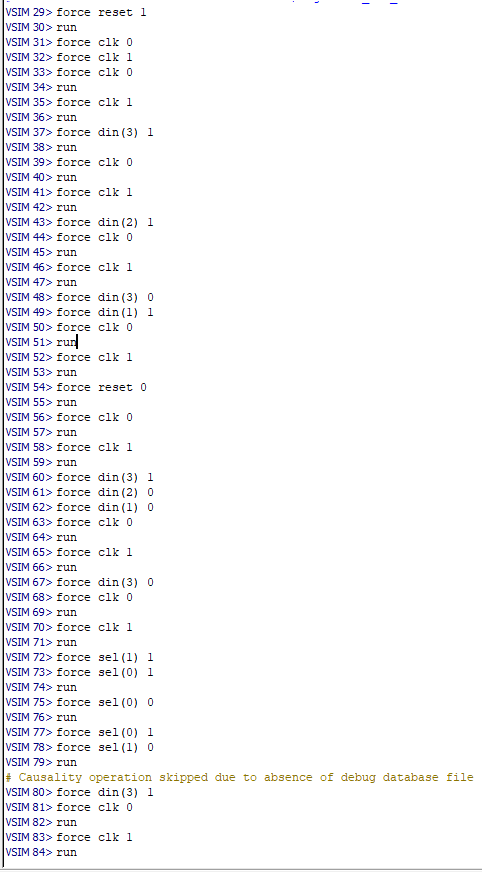


Figure 7. Example of the Transcript Do File Simulation

**5) Questions**

**5.1 To which signals should a clocked process be sensitive to if the register is to have a synchronous reset?**

For a synchronous reset sensitive logic, the main signals that needs to be in the sensitivity list are the clock and the reset. This is due to the fact that the values that we are most concerned about are the signals other than the clock which changes when there is a clock edge. Usually, the analysis focuses on signal other than the clock.

**2. What will happen if the following VHDL code is simulated?**

When simulated, the design will have a repeated segment where the two processes will have their own section. More specifically, the first process design will reset mick and keith to “0000” if reset = ‘1’ while it will make mick equal to input 1 (din1) when there is a rising edge of the clock. The second process design will behave similarly, except that when there is an event clock = ‘1’, it will be made keith equal to the second input (din2). This repetition will be kept as is during simulation.

**3. What will happen if the above code is synthesised?**

On the other hand, when this vhdl code is synthesised, the circuit will simplify itself so that the two processes will combines together. The behavior will be the same, but the synthesised schematic design will be more compact. The final result will have a singular component that reset both mick and keith to “0000” when reset = ‘1’, while it will make keith equal to input 2 and mick equal to input 1 when the event clk = ‘1’ is triggered. Overall, the above code will simplify greatly from using two processes to only one.