"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

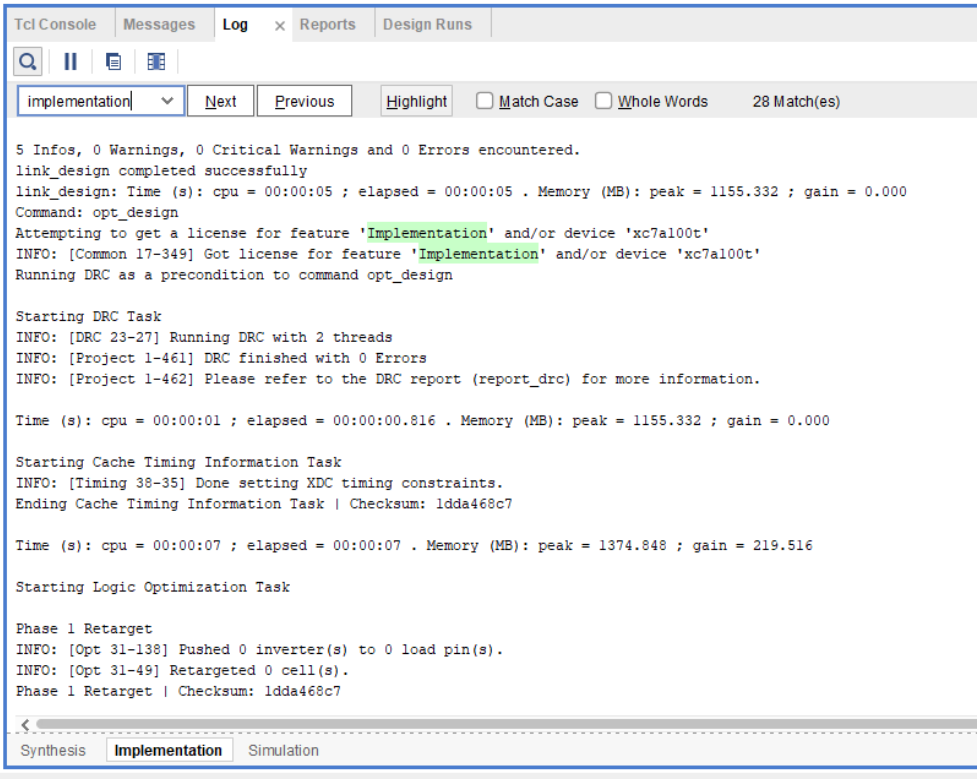


Figure 1. Beginning of the Implementation of the Log File

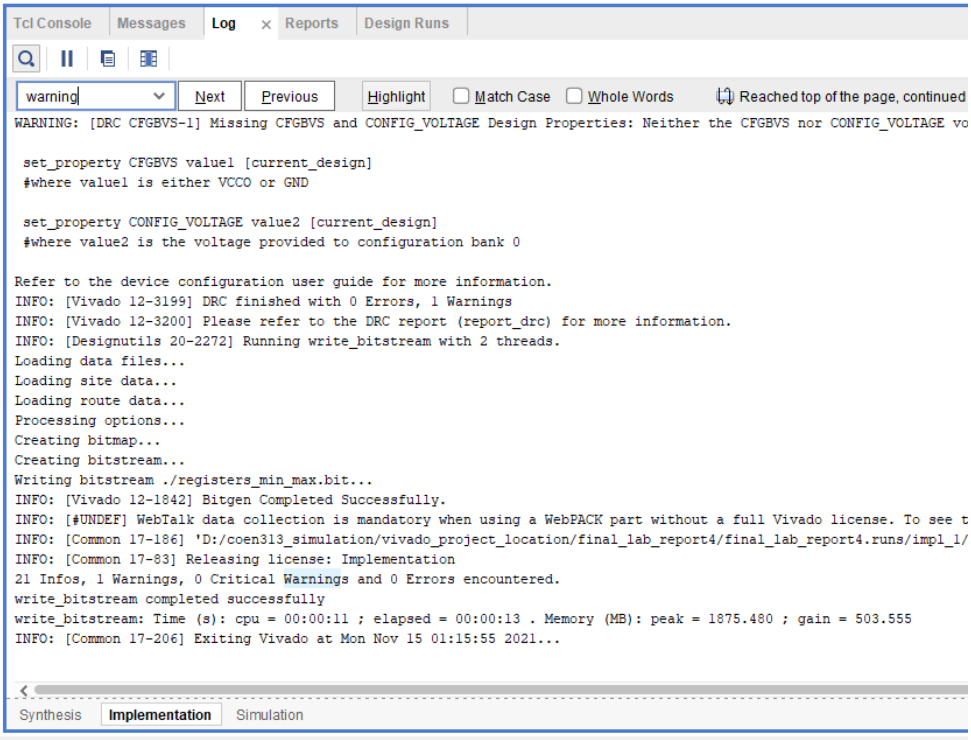


Figure 2. End of the Implementation of the Log File

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

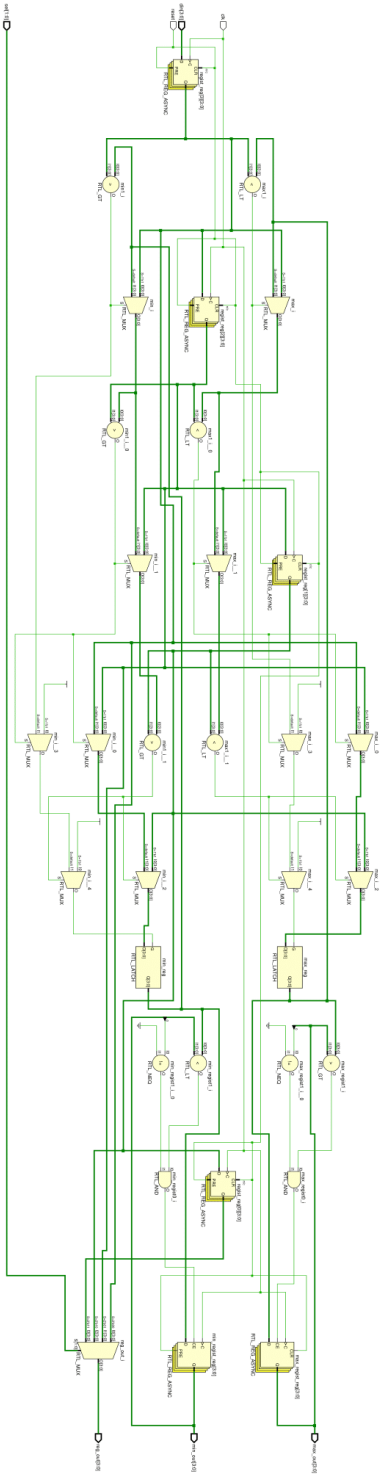

Figure 3. Elaborated Schematic Design of RTL Analysis

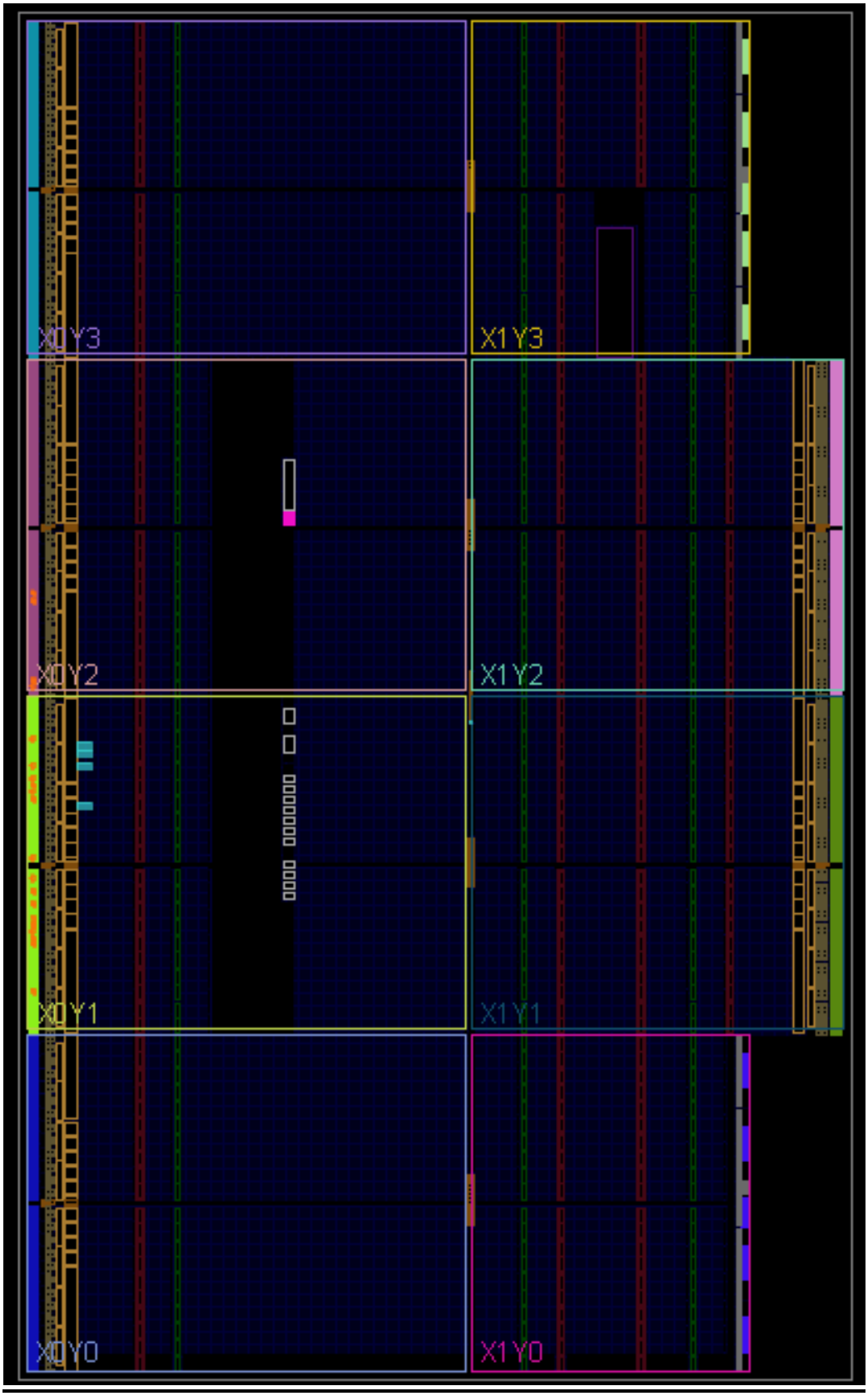


Figure 4. Implemented Schematic Diagram Design View

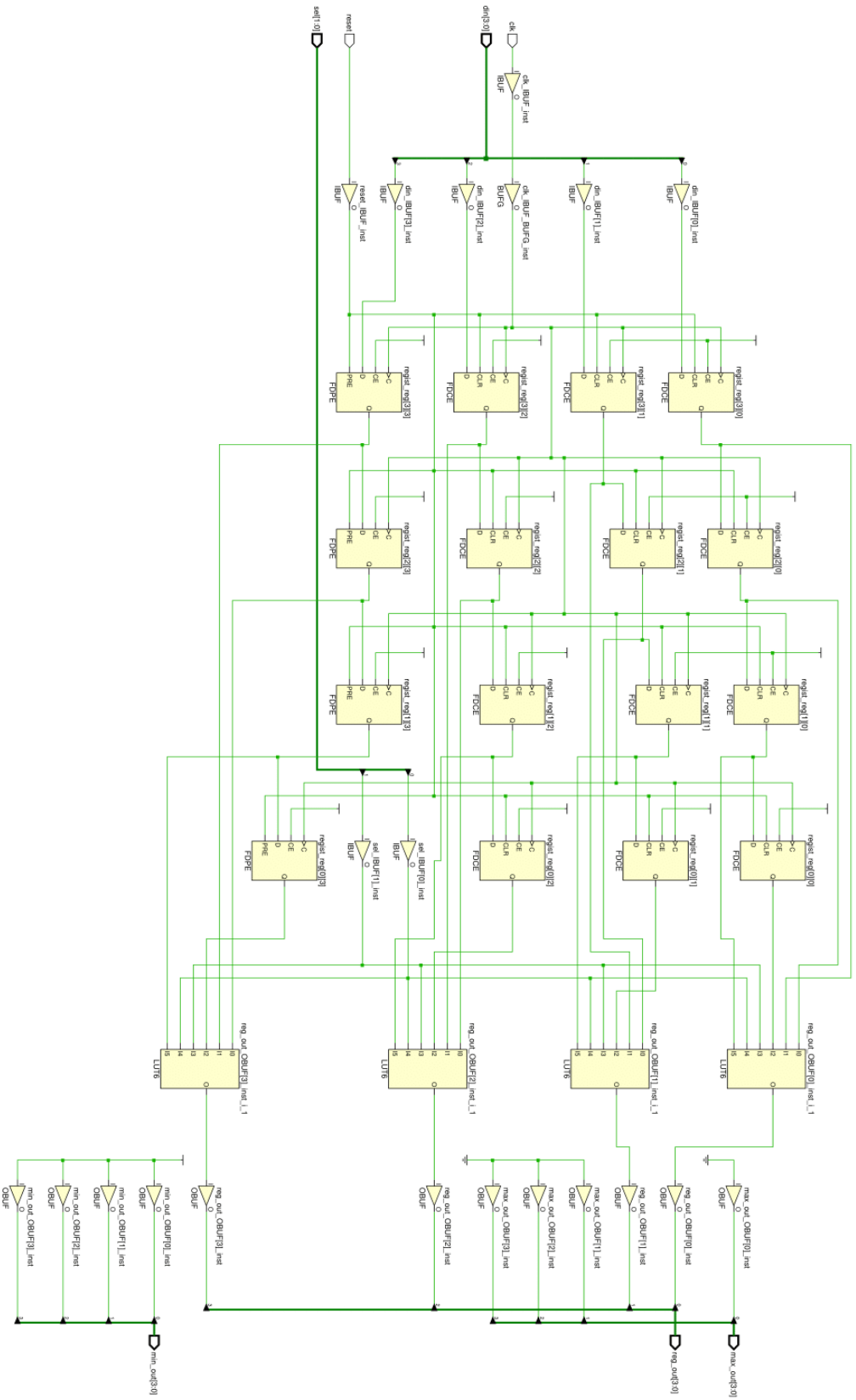

Figure 5. Schematic Diagram Representation of the Implemented Design

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

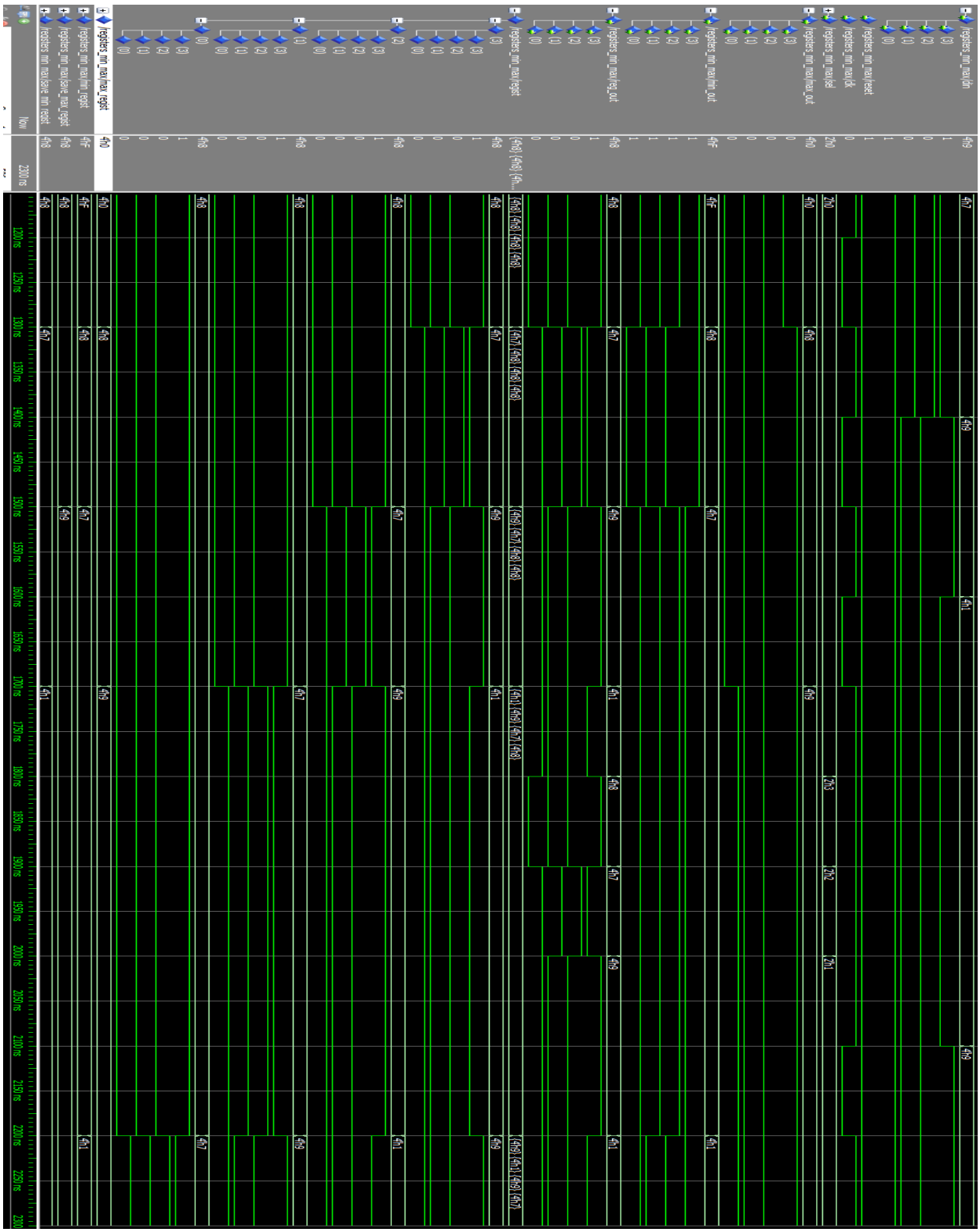

Figure 6. Modelsim Simulation

```
VSIM 29> force reset 1
VSIM 30> run
VSIM 31> force clk 0
VSIM 32> force clk 1
VSIM 33> force clk 0
VSIM 34> run
VSIM 35> force clk 1
VSIM 36> run
VSIM 37> force din(3) 1
VSIM 38> run
VSIM 39> force clk 0
VSIM 40> run
VSIM 41> force clk 1
VSIM 42> run
VSIM 43> force din(2) 1
VSIM 44> force clk 0
VSIM 45> run
VSIM 46> force clk 1
VSIM 47> run
VSIM 48> force din(3) 0
VSIM 49> force din(1) 1
VSIM 50> force clk 0
VSIM 51> run
VSIM 52> force clk 1
VSIM 53> run
VSIM 54> force reset 0
VSIM 55> run
VSIM 56> force clk 0
VSIM 57> run
VSIM 58> force clk 1
VSIM 59> run
VSIM 60> force din(3) 1
VSIM 61> force din(2) 0
VSIM 62> force din(1) 0
VSIM 63> force clk 0
VSIM 64> run
VSIM 65> force clk 1
VSIM 66> run
VSIM 67> force din(3) 0
VSIM 68> force clk 0
VSIM 69> run
VSIM 70> force clk 1
VSIM 71> run
VSIM 72> force sel(1) 1
VSIM 73> force sel(0) 1
VSIM 74> run
VSIM 75> force sel(0) 0
VSIM 76> run
VSIM 77> force sel(0) 1
VSIM 78> force sel(1) 0
VSIM 79> run
# Causality operation skipped due to absence of debug database file
VSIM 80> force din(3) 1
VSIM 81> force clk 0
VSIM 82> run
VSIM 83> force clk 1
VSIM 84> run
```

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.