SPRING 2022
ELEC 204 Digital Design

Project Report
Magical LEDs of Statistics

Group Members:
Demet Tümkaya, 72210
Mahmut Esat Pişkin, 73242

Instructor: Mehmet Cengiz Onbaşlı
Due Date: 8/6/2022

Table of Contents

1. Introduction and objectives

The project aims to calculate certain statistical terms of a set of 3 inputs. These statistical terms are average, median, maximum, minimum, and, accordingly range.

Average can be obtained by dividing the sum of elements into the number of elements in the set. The median is the middle value of the sorted set. Maximum is the greatest and minimum is the smallest number in our case. Lastly, the range can be determined by finding the difference between maximum and minimum numbers.

There are 3 inputs and 5 outputs in the project besides the clock. All inputs are in 4 bits, therefore we can directly set the outputs to 4 bits since the maximum values of average, range and maximum variable is equal to 15, in binary 1111. Additionally, for the sake of simplicity, after getting the inputs in binary form, they are transformed into integers by using the Arithmetic library of IEEE compatible with VHDL design and at the end, all values are transformed to unsigned numbers again to be returned in binary form.

Average is calculated by using the subtraction method instead of the division since division is a type of subtraction but in a simpler way. Logically, if we divide a number into a smaller number, we are consecutively subtracting the divisor from the dividend until we reach 0 or when our dividend is smaller than the divisor. Thus, the number of times that we complete subtraction will be directly equal to the quotient. Our aim is to find the average so we don't need to check for the remainder. On top of that, before starting the division, we need to calculate the sum of all numbers in our set which will be equal to our dividend.

Median is the middle number in the set, and since we have 3 inputs, it is literally the middle number for our design. As we want to determine the minimum and maximum of our input set, we can apply a sorting algorithm to find all of these terms. Finally, the range can be calculated as the difference between these greatest and smallest elements.

To get an error-free design, as our system has many arithmetic calculations, it is better to have intermediate variables and later on assign them to the final outputs. Our project calculates 5 statistical terms and can be extended further to have a wide scope statistical calculator with more inputs.

2. Methods

Inputs	**Size**	**Outputs**	**Size**	**Description**
Clock		result	4 bits	Calculated average
input1	4 bits	mint	4 bits	Minimum number of the set
input2	4 bits	maxt	4 bits	Maximum number of the set
input3	4 bits	medt	4 bits	Median of the set
		ranget	4 bits	Range of the set

Table 1: Inputs and outputs of the statistical calculator

VHDL code must calculate the average of 3 inputs, determine the minimum, the maximum, the median, and the range of 3 inputs of 4 bits. The outputs are directly coherent with the mathematical definitions of average, minimum, maximum, median, and range are shown in Table 1. There are intermediate variables for all inputs and outputs since we get binary data but operate with integers.

a. Calculating average

As it is stated in the objectives of the project, in order to do the division we choose to do subtraction repeatedly because of the arithmetic simplicity in VHDL design. The result of this cascade of operations will be assigned to an intermediate signal of q which will be then assigned to the output of the average at the end. While calculating the average our code waits for rising_edge so that it performs the corresponding operation according to the present condition, then changes the condition to the next state. The designed process is simulated with an example below:

Input 1: 0101
Input 2: 1000
Input 3: 0000
Sum = 0101 + 1000 + 0000 = 1101 = 13

After the calculation of the sum, there is a flag variable that shows that the system is ready to progress to the next step. Average calculation is done by subtracting the count variable, in our case 3 from 13 until the resultant value from the subtraction is smaller than 3 which is the number of elements in our input set.

Operation 1: 13 - 3 =10 (10 > 3, then continue.)
Operation 2: 10 - 3 = 7 (7 > 3, then continue.)
Operation 3: 7 - 3 = 4 (4 > 3, then continue.)
Operation 4: 4 - 3 = 1 (1<3, stop!)
***Average** = 4 = 0100*

As we crosscheck, our final result of the number of operations is equal to the real average of these 3 inputs as 13 divided by 3 is 4.

b. Determining the minimum, maximum, and median

All inputs are assigned to the intermediate shared variables of minimum, median and maximum respectively. Then, the sorting process is started by comparing 3 inputs as pairs and swapping if needed. We can simulate one case concretely for the same inputs:

Input 1: 0101 → Minimum
Input 2: 1000 → Median
Input 3: 0000 → Maximum

Check 1: minimum > median → False → No change in the order
Check 2: median > maximum → True
maximum = median = 1000
median = 0000

Check 3: minimum > median → True
median = minimum = 0101
minimum = 0000

Final Order: minimum = 0000
median = 0101
maximum = 1000

c. *Calculating the range of input set*

Since we know the maximum and minimum from the previous step, we can calculate the range according to the official description of it as the difference between maximum and minimum.

Range = Maximum - Minimum = 1000 - 0000 = 1000

All results can be seen in a real simulation from the VHDL implementation of the design.

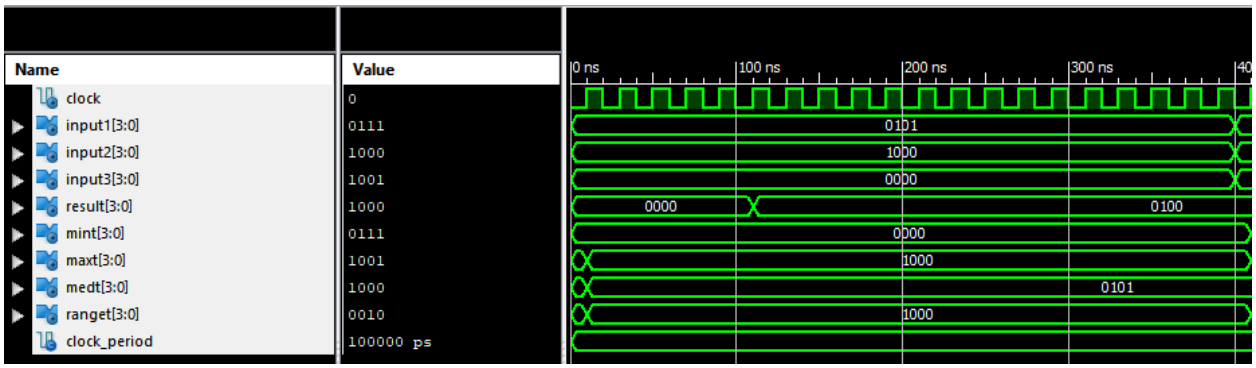


Figure 1: Simulation of design with input 0111, 1000, 1001.

For the safe calculation and obtaining correct outputs, when the code completes the calculation parts, it assigns the resultant values to intermediate variables. After finishing the process, it assigns those variables to logic vector outputs that are given in the beginning.

It can be seen that calculating 5 different variables is highly independent of each other and overall, it has a huge RTL schematics with flip flops, comparators, adders, multiplexers, and combinational logic circuit elements for each set of calculations to finalize the outputs. Additionally, with 3 inputs of 4 bits, we have 15^3 possible cases in our design. Therefore, for the truth table and the state diagram of our project, we tried to represent our design with many different exemplary input sets in Table 2.

	#1	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	**#9**	**#10**
Input1	0010	0001	1010	1101	0101	0110	0101	0100	1100	1011
Input2	0101	0110	0010	0011	0111	1000	1110	1011	1101	1011
Input3	1110	1001	1011	0000	1001	0010	0010	0001	0010	1011
Result	0111	0101	0111	0101	0111	0101	0111	0101	1001	1011
Mint	0010	0001	0010	0000	0101	0010	0010	0001	0010	1011

Maxt	1110	1001	1011	1101	1001	1000	1110	1011	1101	1011
Medt	0101	0110	1010	0011	0111	0110	0101	0100	1100	1011
Ranget	1100	1000	1001	1101	0100	0110	1100	1010	1011	0000

Table 2: Different input sets and respective outputs for statistical terms present in the design

3. Problems encountered, errors, and warnings resolved

We have a fundamental delay problem in our simulation. The values and corresponding results are correct but the change in the *result* variable that represents the average has a delay of approximately 70-120 ns depending on the inputs. We are aware that the reason for this problem is the sequential nature of the consecutive subtractions at each rising clock period while calculating the average of the set.

To get the minimum effect from this desynchronization, we optimized our simulation to have 250 ns between each cycle of input sets instead of having an irregular simulation output like Figure 2. In the case of waiting for the clock period, even before another set of input comes, we cannot get the previous set's statistical terms as final outputs.

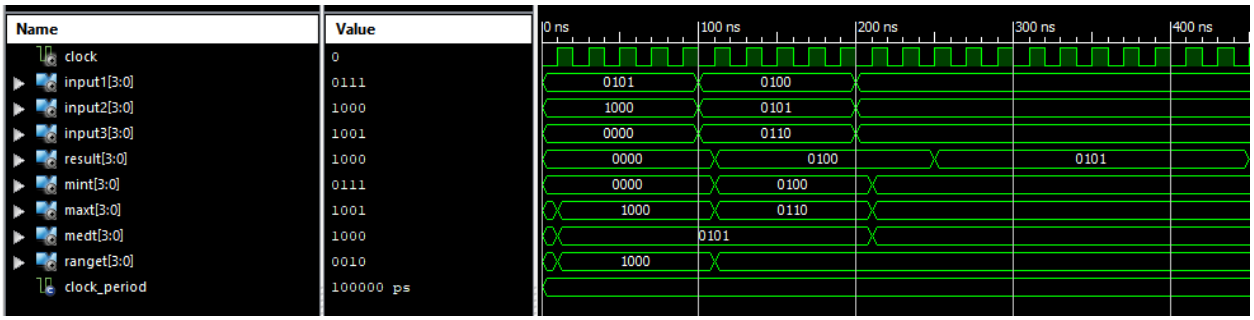


Figure 2: Simulation of inputs with only waiting for a clock period

The number 250 is determined in such a way that we can get a maximum of 15 subtraction operations with the maximum inputs as 1111 and a sum of 45.

Other than that, while writing our code, we encountered a scope problem. When we tried to make changes to intermediate variables in different processes, the Xilinx ISE was giving an error of scope since in different cycles of the clock and rising edges, if we want to use other intermediates, we need to initialize them as global variables. So, we solve this problem by using shared variables instead of variables under the clock-rising sections.

For the synthesis part, there are no errors but since the FPGA part is not available in our project, there are only warnings about connection to hardware.

4. Conclusion

The project was based on different calculations and eventually finding certain statistical terms for an input set of 3. To reach the aim of determining the average, minimum, maximum, median, and range of a set, we have combined different methods like comparison, division by subtraction, clock dependency, and function usage from a library.

Testing the implementation and simulating with many examples were necessary for such design because it has many probabilities within the framework of 3 inputs each in 4 bits and 5 outputs all representing distinct values. We have discovered how to resolve redundant usages and initialize shared variables only once instead of doing it more than once, by getting errors and warnings throughout the process. Moreover, we overcome a design constraint by optimizing our simulation which we have never done before in the scope of this course and this was a brand new experience for us that we see a real sequential process that causes delay.

Unlike the lab assignments, we did the planning, design, implementation, and simulation processes from the very beginning by ourselves, therefore it was a good opportunity to have such a final project that we can relate to its all steps, then use and apply the things that we have learned in perspective.

References

1. https://www.youtube.com/watch?v=2YbelsD79Q4&ab_channel=Eduvance

2. Koç University Department of Electrical & Electronics Engineering. (n.d.). Experiment #3 Design and Implementation of a 4-bit Arithmetic and Logic Unit. In ELEC 204 Digital System Design Laboratory Manual.

3. M. M. Mano and M. D. Ciletti, *Digital design with an introduction to the Verilog HDL, VHDL, and SystemVerilog*. Pearson, 2019.

Appendices

Appendix 1. Project source code

```
----------------------------------------------------------------------------------
-- Company: Koç University
-- Engineer: Demet Tümkaya, Mahmut Esat Pişkin
--
-- Module Name:   final_code - Behavioral
-- Project Name:  ELEC204 - Magical LEDs of Statistics
-- Description: Designed to calculate maximum, minimum, range, median,
--                                        average for 3 inputs of 4 bits.
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity final_code is
        -- inputs: clock, input1, input2, input3
        -- outputs: min, max, median, result (average), range
        Port ( Clock : in STD_LOGIC;
                                input1 : in  STD_LOGIC_VECTOR (3 downto 0);
                                input2 : in  STD_LOGIC_VECTOR (3 downto 0);
                                input3 : in  STD_LOGIC_VECTOR (3 downto 0);
                                result : out  STD_LOGIC_VECTOR (3 downto 0);
                                mint:  out  STD_LOGIC_VECTOR (3 downto 0);
                                maxt:  out  STD_LOGIC_VECTOR (3 downto 0);
                                medt:  out  STD_LOGIC_VECTOR (3 downto 0);
                                ranget:  out  STD_LOGIC_VECTOR (3 downto 0));
end final_code;

architecture Behavioral of final_code is
        -- secondary parameters used in the algorithm
        signal count : integer := 3 ;
        signal average : integer := 0;
        signal q : integer := 0;

        -- intermediate amd temporary variables are initialized as integers
        shared variable range_temp : integer := 0;
        shared variable min_temp : integer := 0;
        shared variable max_temp : integer := 0;
        shared variable med_temp : integer := 0;
        shared variable result_temp : integer := 0;

        shared variable in1_int : integer ;
        shared variable in2_int : integer ;
        shared variable in3_int : integer ;

        shared variable range_int : integer:= 0 ;
        shared variable min_int : integer := 0;
```

```
        shared variable max_int : integer := 0;
        shared variable med_int : integer := 0;
        shared variable result_int : integer := 0;

begin

process(Clock)
        begin
                -- inputs transformed into integers
                in1_int := to_integer(unsigned(input1));
                in2_int := to_integer(unsigned(input2));
                in3_int := to_integer(unsigned(input3));

                if(rising_edge(Clock)) then
                        -- calculates average at rising edge of clock
                        if(average = 0) then
                                -- first sums all inputs and then flags to start calculating average
                                result_int := in1_int + in2_int + in3_int;
                                average <= 1;

                        elsif (average = 1) then
                                -- division is done by subtraction
                                if (result_int > count) then
                                        result_int := result_int - count;
                                        q <= q+1;

                                elsif (result_int = count) then
                                        result_int := result_int - count;
                                        q <= q+1;

                                else
                                        -- final result is the count of steps of subtraction
                                        result_temp := q;
                                        q <= 0;
                                        average <= 0;

                                end if;
                        end if;
                end if;


                if(rising_edge(Clock)) then
                        -- calculates min, max, median, range at rising edge of clock
                        -- min, med and max are assigned to inputs respectively
                        min_int := in1_int;
                        med_int := in2_int;
                        max_int := in3_int;

                        if(min_int > med_int) then
                                -- if any swap is needed between first and second variable
                                med_int := in1_int;
```

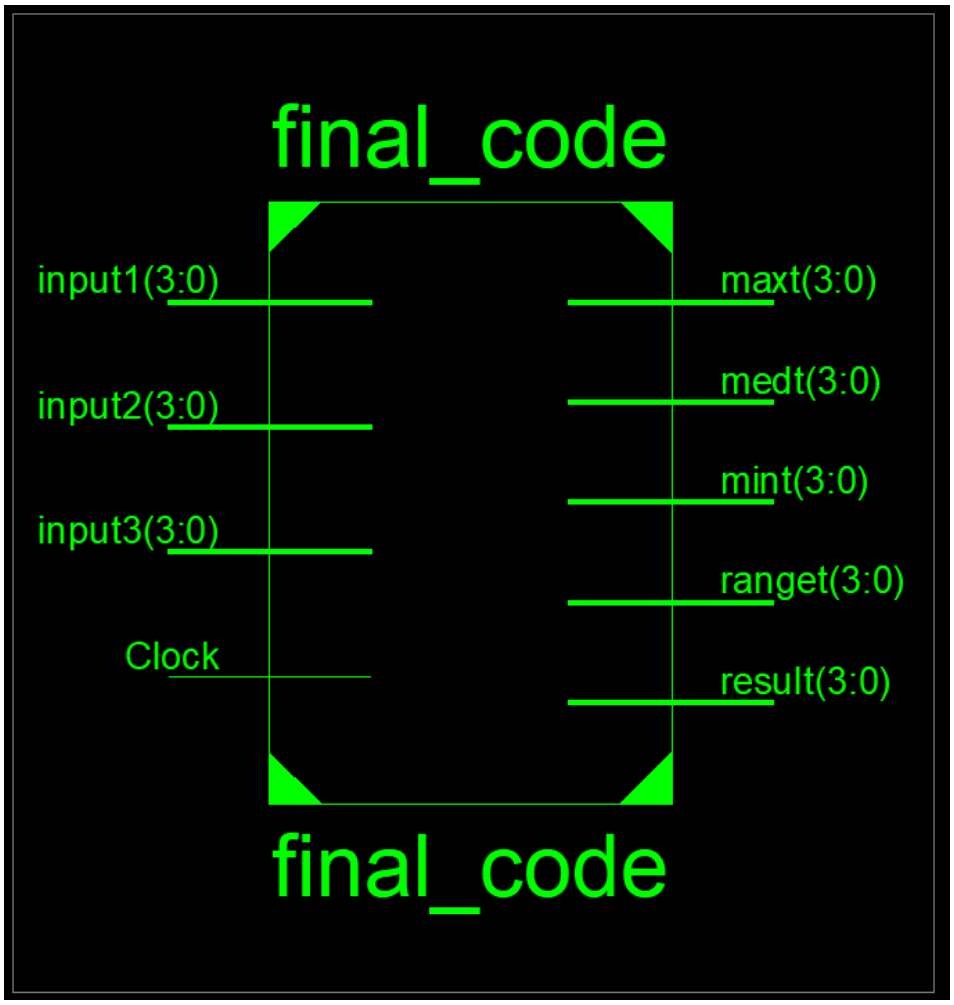
```
                min_int := in2_int;

            end if;

            if(med_int > max_int) then
                -- if any swap needed between second and third variable
                max_int := med_int;
                med_int := in3_int;

                if(min_int > med_int) then
                    -- checks the previous condition again with this change
                    med_int := min_int;
                    min_int:= in3_int;
                end if;
            end if;

            max_temp := max_int;
            min_temp := min_int;
            med_temp := med_int;
            range_int := max_int - min_int;
            range_temp := range_int;
        end if;

        -- finally the integers are transformed into unsigned of 4 bits
        maxt <= std_logic_vector(to_unsigned(max_temp, 4));
        mint <= std_logic_vector(to_unsigned(min_temp, 4));
        medt <= std_logic_vector(to_unsigned(med_temp, 4));
        ranget <= std_logic_vector(to_unsigned(range_temp, 4));
        result <= std_logic_vector(to_unsigned(result_temp, 4));
    end process;

end Behavioral;
```

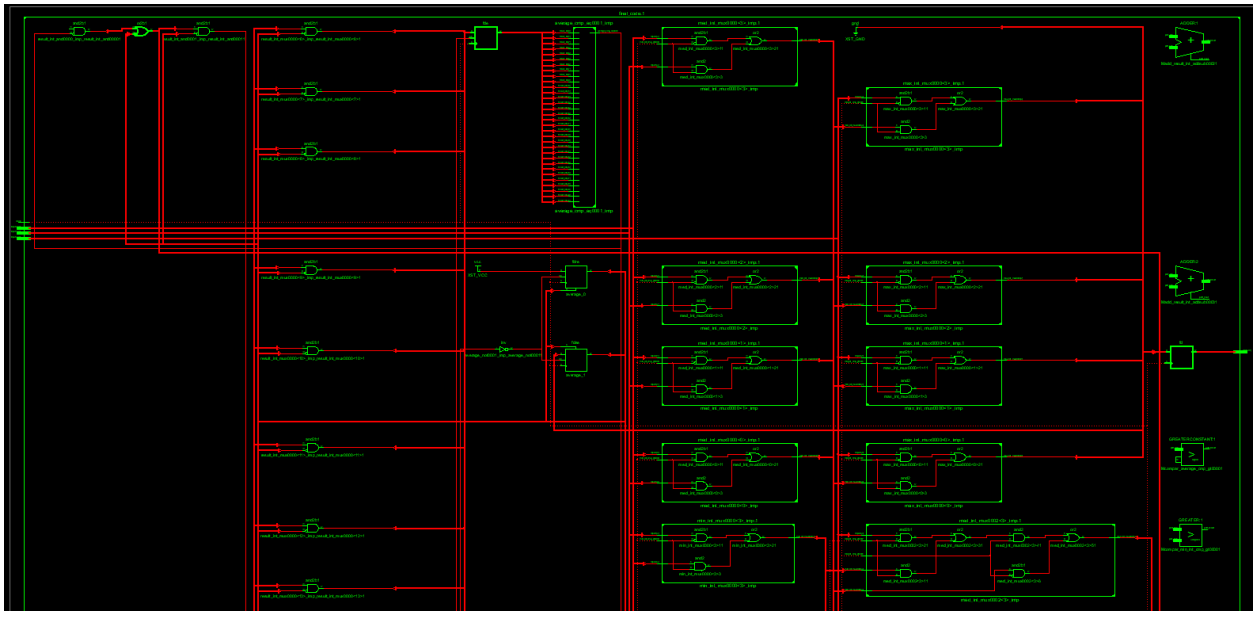
Project Simulation Code

```
--------------------------------------------------------------------------------
-- Company: Koç University
-- Engineer: Demet Tümkaya, Mahmut Esat Pişkin
--
-- Project Name:  ELEC204 - Magical LEDs of Statistics
-- Description: Designed to calculate maximum, minimum, range, median,
--                                          average for 3 inputs of 4 bits.
--
-- VHDL Test Bench Created by ISE for module: final_code
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY final_sim IS
END final_sim;

ARCHITECTURE behavior OF final_sim IS
   -- Component Declaration for the Unit Under Test (UUT)

   COMPONENT final_code
   PORT(
       Clock : IN  std_logic;
       input1 : IN  std_logic_vector(3 downto 0);
       input2 : IN  std_logic_vector(3 downto 0);
       input3 : IN  std_logic_vector(3 downto 0);
       result : OUT  std_logic_vector(3 downto 0);
                           mint : OUT  std_logic_vector(3 downto 0);
                           maxt : OUT  std_logic_vector(3 downto 0);
                           medt : OUT  std_logic_vector(3 downto 0);
                           ranget : OUT  std_logic_vector(3 downto 0)
      );
   END COMPONENT;

  --Inputs
  signal Clock : std_logic := '0';
  signal input1 : std_logic_vector(3 downto 0) := (others => '0');
  signal input2 : std_logic_vector(3 downto 0) := (others => '0');
  signal input3 : std_logic_vector(3 downto 0) := (others => '0');

         --Outputs
  signal result : std_logic_vector(3 downto 0);
         signal mint : std_logic_vector(3 downto 0);
         signal maxt : std_logic_vector(3 downto 0);
```

```
        signal medt : std_logic_vector(3 downto 0);
        signal ranget : std_logic_vector(3 downto 0);

  constant Clock_period : time := 100 ns;

BEGIN
         -- Instantiate the Unit Under Test (UUT)
  uut: final_code PORT MAP (
       Clock => Clock,
       input1 => input1,
       input2 => input2,
       input3 => input3,
       result => result,
                          maxt => maxt,
                          mint => mint,
                          medt => medt,
                          ranget => ranget
      );

  -- Clock process definitions
  Clock_process :process
  begin
                Clock <= '0';
                wait for Clock_period/10;
                Clock <= '1';
                wait for Clock_period/10;
  end process;

  -- Stimulus process
  stim_proc: process
  begin

                input1 <= "0001";
                input2 <= "0010";
                input3 <= "0011";

                wait for 250 ns;

                input1 <= "0100";
                input2 <= "0101";
                input3 <= "0110";

                wait for 250 ns;

                input1 <= "0111";
                input2 <= "1000";
                input3 <= "1001";

    wait;
  end process;
END;
```

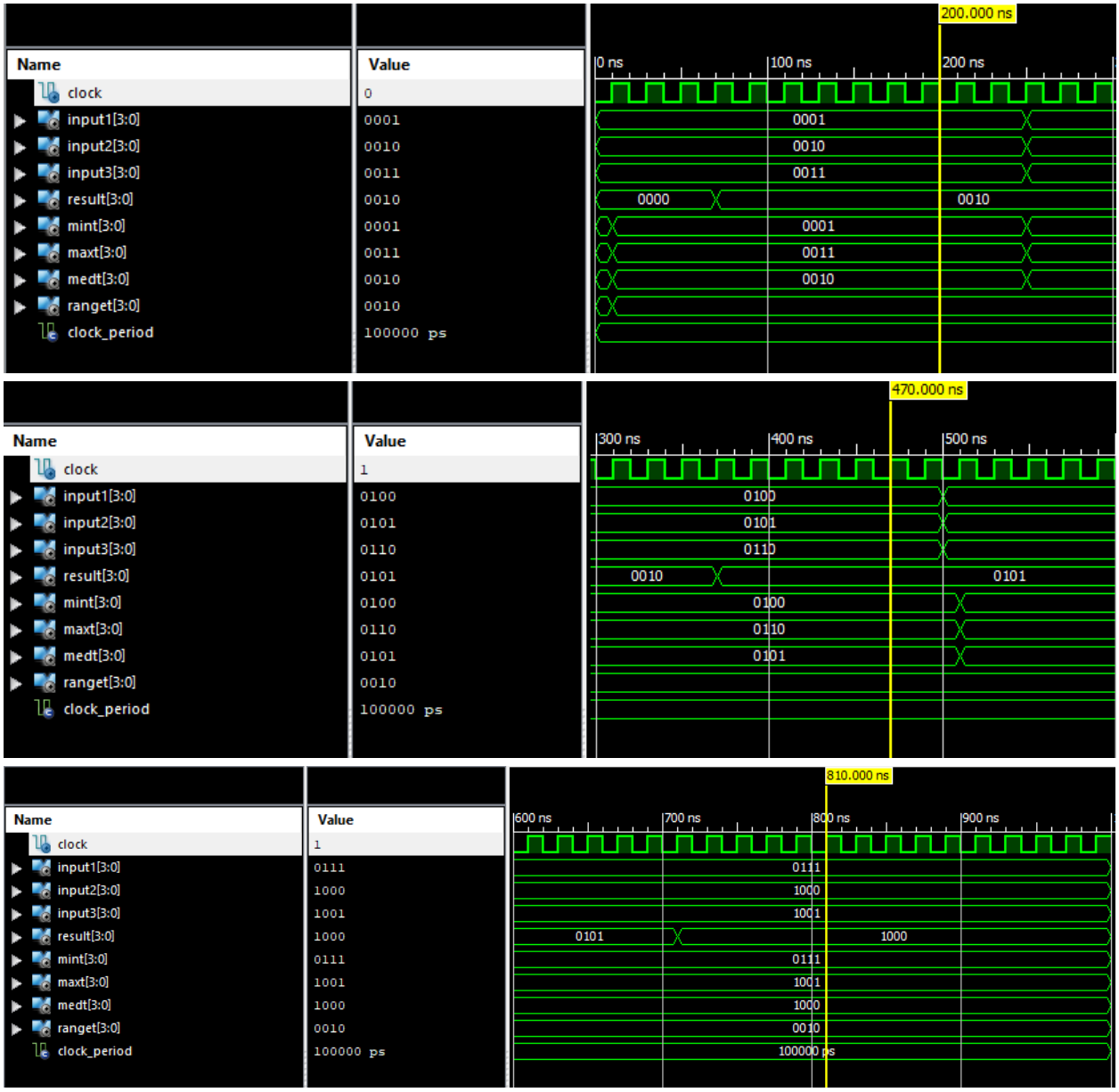
Appendix 2. RTL schematics





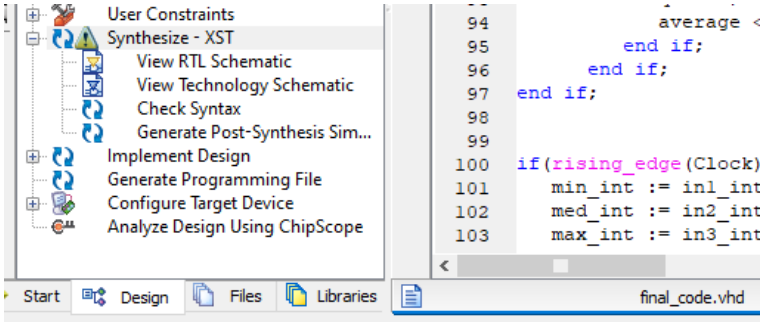
Appendix 3. Photos showing the working code

FPGA board is not available, so we add our simulation here with 3 examples.

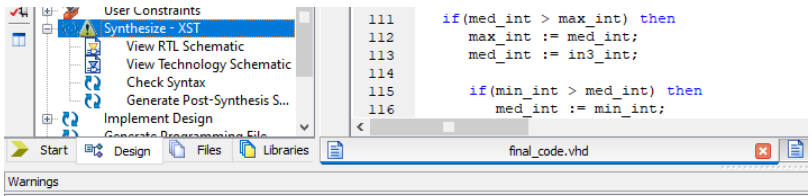


Appendix 4. Screenshots from Xilinx for the errors and other board issues

There are no errors in our code, but we have a number of warnings




```
Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 5.853ns (Maximum Frequency: 170.854MHz)
   Minimum input arrival time before clock: 12.280ns
   Maximum output required time after clock: 5.248ns
   Maximum combinational path delay: No path found

=========================================================================

Process "Synthesize - XST" completed successfully
```




```
Warnings
WARNING:Xst:2677 - Node <q_8> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_9> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_10> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_11> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_12> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_13> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_14> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_15> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_16> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_17> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_18> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_19> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_20> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_21> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_22> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_23> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_24> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_25> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_26> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_27> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_28> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_29> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_30> of sequential type is unconnected in block <final_code>.
WARNING:Xst:2677 - Node <q_31> of sequential type is unconnected in block <final_code>.
```

No errors in the simulation