VHDL ASSIGNMENT 5

1- Names

Loïc Duchesne - 261052490
Yassine Mimet - 260980175

2- Executive summary

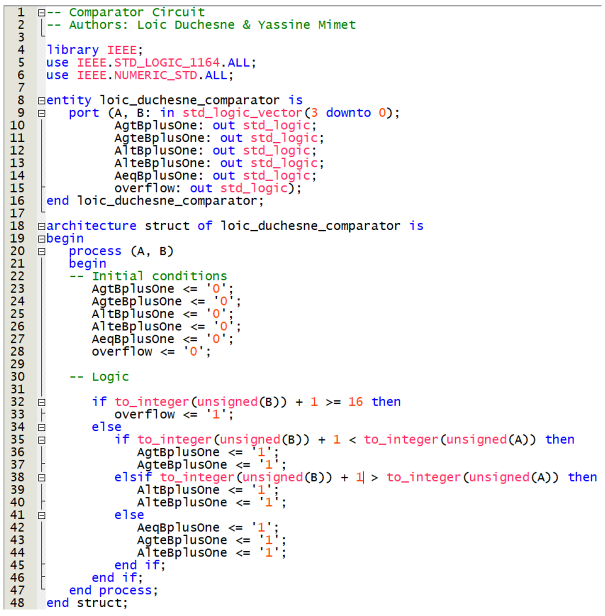
For this lab, we were tasked to write a VHDL program using sequential assignment statements, and then test it using a testbench and the FPGA board. We wrote the code using a behavioral approach and using sequential assignment techniques learned in the lab manual. We then set the input A to 5 as required in the lab manual, and iterated for all values for input B. Afterwards, we ran a timing analysis to find the critical path in our circuit. Finally, we ran the program on our FPGA board using the switches and LED, to observe the behaviors on an actual FPGA board.

3- Questions

1- VHDL explanation

Comparator circuit:

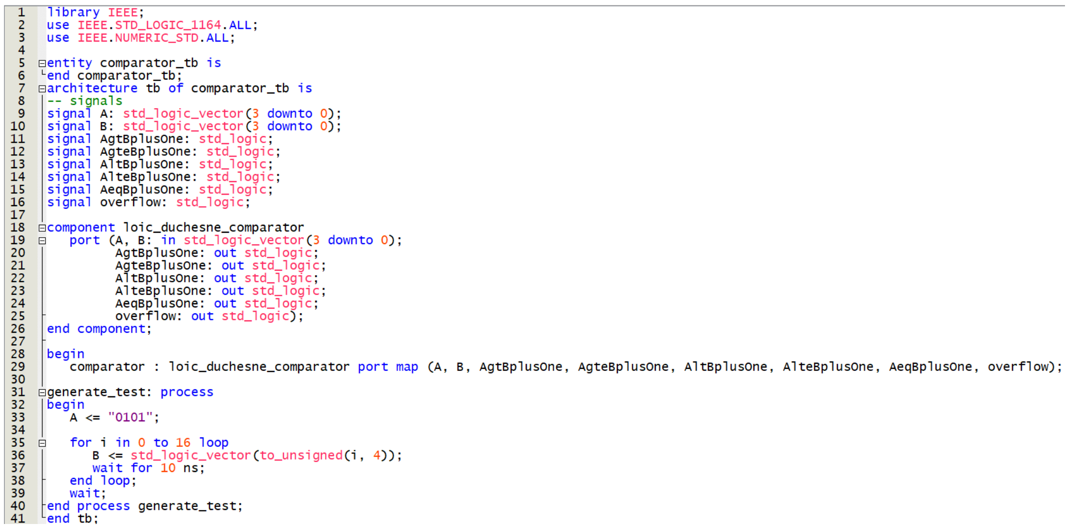
For the entity, we used the code implementation in the lab document. For the architecture, we used the sequential assignment statements. In the sensitivity list, we had to write all our inputs; in this case, it was just A and B. After the process statement, we tried to set the initial conditions of the output signals to ‘0’. For the if / elsif / else, the first if / else statement was just to see if adding ‘1’ to be will have a carry (basically checking if the number is a 5 bits number); if the carry is ‘0’, then we check if B+1 is either bigger smaller or equal to A and we change the output values depending on the condition met.



```vhdl
-- Comparator Circuit
-- Authors: Loic Duchesne & Yassine Mimet

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity loic_duchesne_comparator is
   port (A, B: in std_logic_vector(3 downto 0);
         AgtBplusOne: out std_logic;
         AgteBplusOne: out std_logic;
         AltBplusOne: out std_logic;
         AlteBplusOne: out std_logic;
         AeqBplusOne: out std_logic;
         overflow: out std_logic);
end loic_duchesne_comparator;

architecture struct of loic_duchesne_comparator is
begin
   process (A, B)
   begin
   -- Initial conditions
      AgtBplusOne <= '0';
      AgteBplusOne <= '0';
      AltBplusOne <= '0';
      AlteBplusOne <= '0';
      AeqBplusOne <= '0';
      overflow <= '0';

   -- Logic

      if to_integer(unsigned(B)) + 1 >= 16 then
         overflow <= '1';
      else
         if to_integer(unsigned(B)) + 1 < to_integer(unsigned(A)) then
            AgtBplusOne <= '1';
            AgteBplusOne <= '1';
         elsif to_integer(unsigned(B)) + 1 > to_integer(unsigned(A)) then
            AltBplusOne <= '1';
            AlteBplusOne <= '1';
         else
            AeqBplusOne <= '1';
            AgteBplusOne <= '1';
            AlteBplusOne <= '1';
         end if;
      end if;
   end process;
end struct;
```

Comparator circuit Test-Bench:

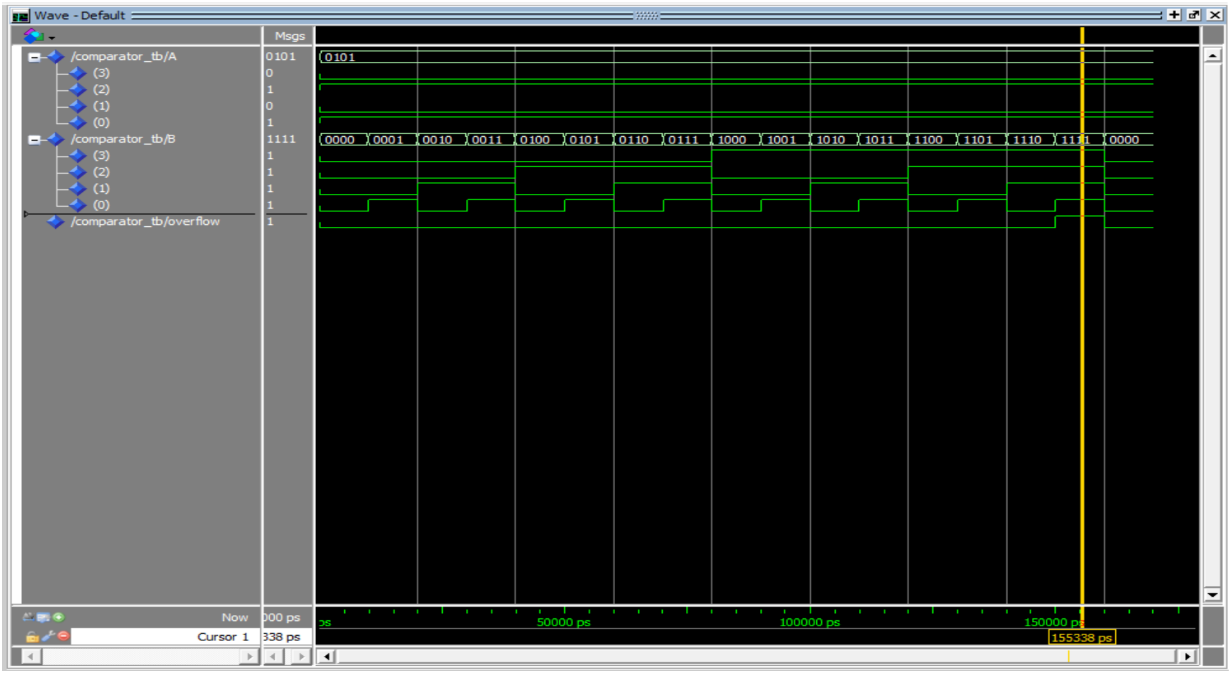
We used the same signals and component declaration as the last labs. In the process, we had to set the value of A to be 0101, as indicated in the lab manual, and for B, we did a for loop to cover all 16 cases.



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

entity comparator_tb is
end comparator_tb;
architecture tb of comparator_tb is
-- signals
signal A: std_logic_vector(3 downto 0);
signal B: std_logic_vector(3 downto 0);
signal AgtBplusOne: std_logic;
signal AgteBplusOne: std_logic;
signal AltBplusOne: std_logic;
signal AlteBplusOne: std_logic;
signal AeqBplusOne: std_logic;
signal overflow: std_logic;

component loic_duchesne_comparator
   port (A, B: in std_logic_vector(3 downto 0);
         AgtBplusOne: out std_logic;
         AgteBplusOne: out std_logic;
         AltBplusOne: out std_logic;
         AlteBplusOne: out std_logic;
         AeqBplusOne: out std_logic;
         overflow: out std_logic);
end component;

begin
   comparator : loic_duchesne_comparator port map (A, B, AgtBplusOne, AgteBplusOne, AltBplusOne, AlteBplusOne, AeqBplusOne, overflow);

generate_test: process
begin
   A <= "0101";

   for i in 0 to 16 loop
      B <= std_logic_vector(to_unsigned(i, 4));
      wait for 10 ns;
   end loop;
   wait;
end process generate_test;
end tb;
```

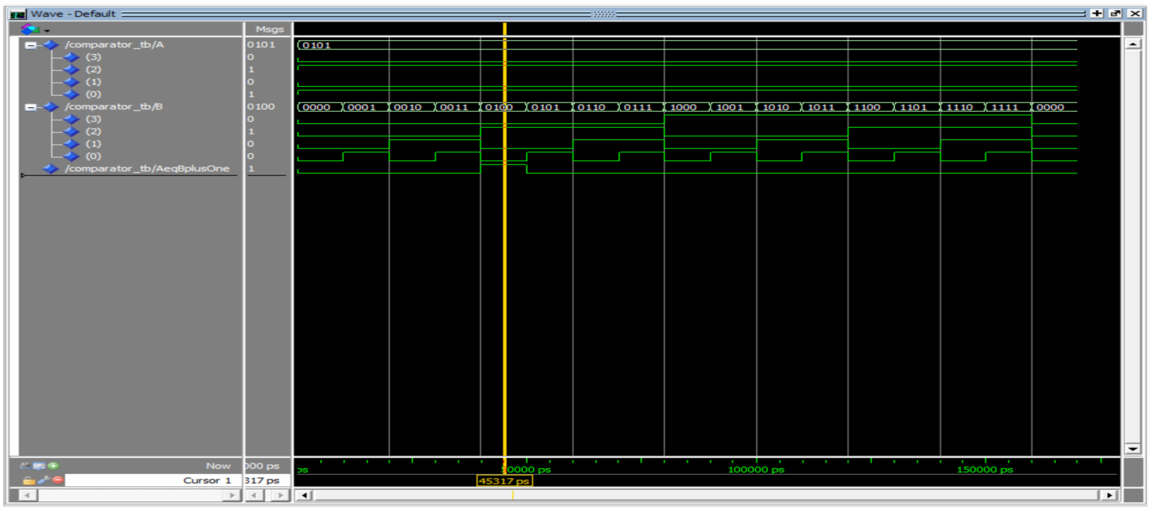
2- Simulation Plots

Overflow:



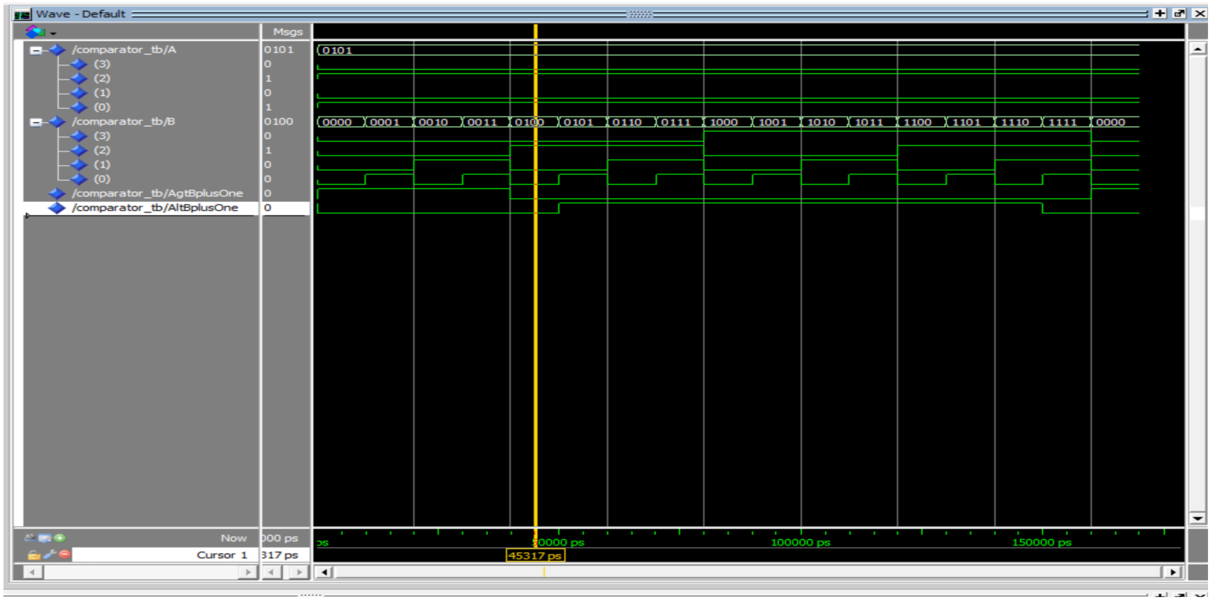
We will get overflow only in the case of B = ‘1111’ because that when the addition of ‘1’ to ‘1111’ will give us a 5-bit number, in other words, the carry is gonna be 1. For the other cases, the overflow will be 0

A = B+1:



The only case where the AeqBplusOne signal is gonna be equal to one is when B+1 = A and that will only happen when B is equal to A-1, in this case, A = ‘0101’ so B has to be ‘0100’, this definitely matches what we have on the graph.

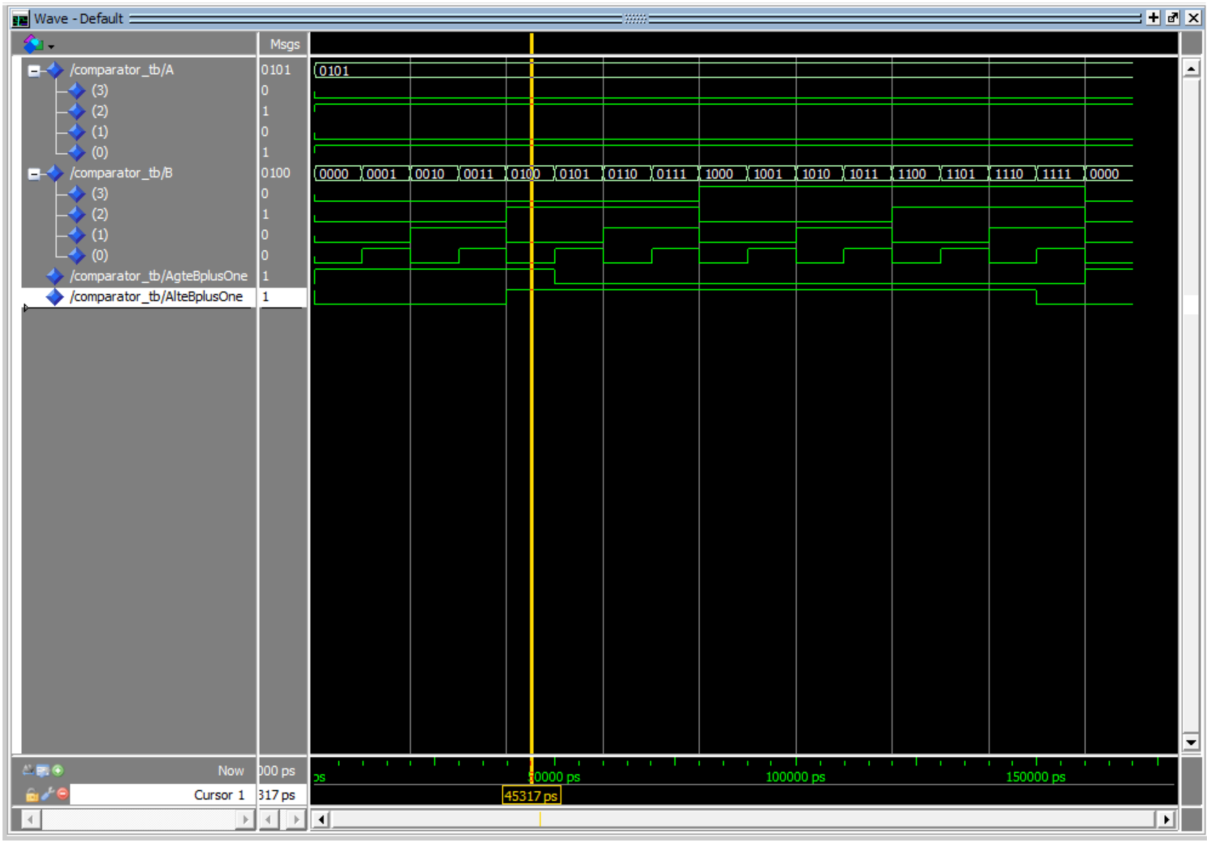
A > B+1 and A < B+1:



For A > B+1, we know that A=B+1 happens when B=’0101’ so A is gonna be bigger than B+1 for all values of B strictly less than ‘0100’ which is just 4 in base 10, so it’s gonna for B=’0000’, B=’0001’, B=’0010’, B=’0011’

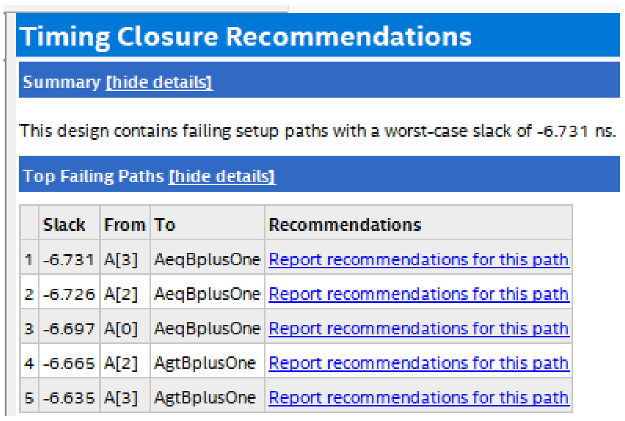
For A < B+1, it’s the same logic you will get the result for values of B strictly bigger than ‘0100’ but not equal to ‘1111’ because you’ll get overflow then.

A >= B+1 and A <= B+1:



It is similar to the A > B+1 and A < B+1, but here you'll have to add the value of B where A = B+1 which is just ‘0100’, you can basically see the overlap between the two graphs, this wasn’t the case in the strictly bigger or less than.

3- Timing Analysis



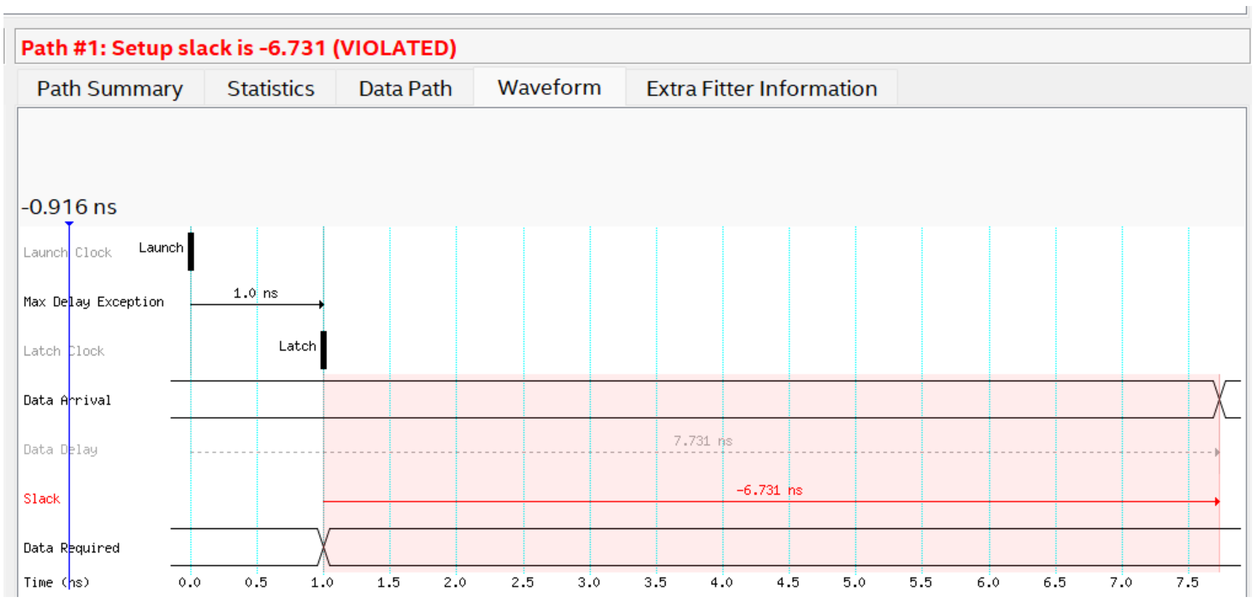
Timing Closure Recommendations

Summary [hide details]

This design contains failing setup paths with a worst-case slack of -6.731 ns.

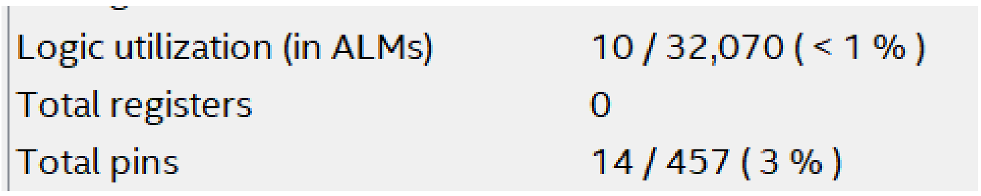
Top Failing Paths [hide details]

	Slack	From	To	Recommendations
1	-6.731	A[3]	AeqBplusOne	Report recommendations for this path
2	-6.726	A[2]	AeqBplusOne	Report recommendations for this path
3	-6.697	A[0]	AeqBplusOne	Report recommendations for this path
4	-6.665	A[2]	AgtBplusOne	Report recommendations for this path
5	-6.635	A[3]	AgtBplusOne	Report recommendations for this path



As seen in the waveform, the Delay for the critical path amounts to 7.731 ns. This was found using an SDC file and Quartus's timing report which returns us to the path with the greatest slack.

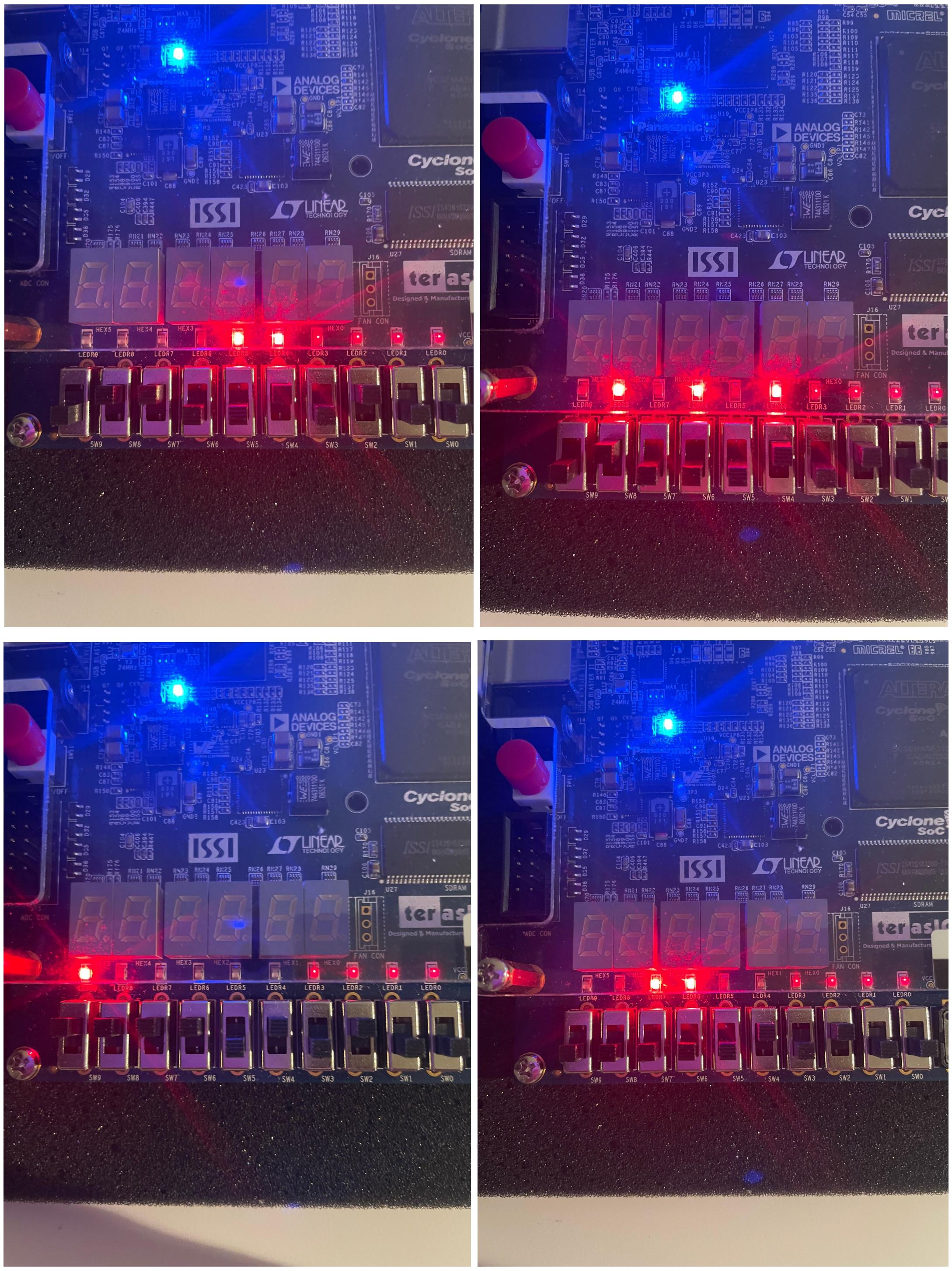
4- Numbers of Pins & Logic modules



Logic utilization (in ALMs)	10 / 32,070 (< 1 %)
Total registers	0
Total pins	14 / 457 (3 %)

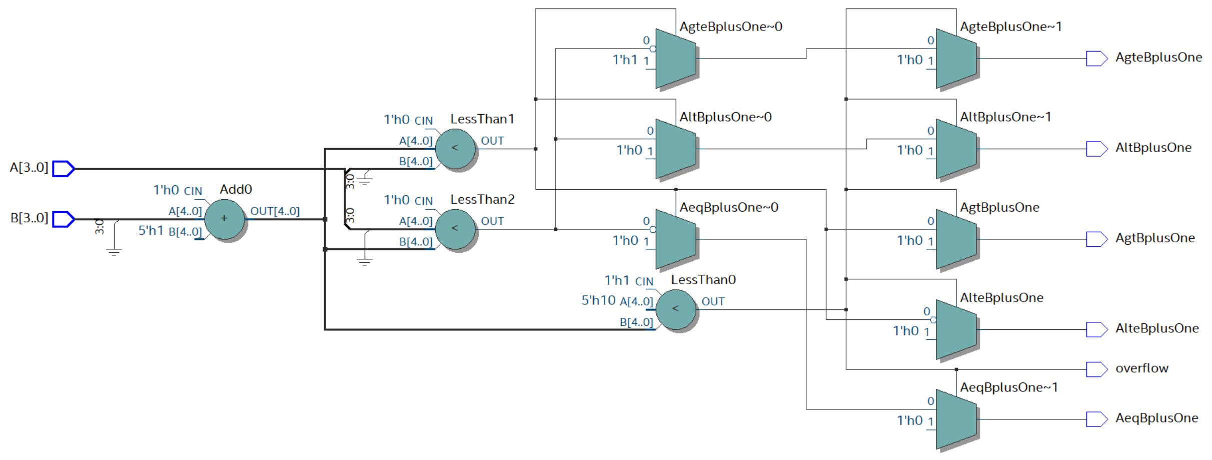
4- Additional Pictures

Implementation of our VHDL program on the FPGA board.



Top left: B is set to 0110 & A set to 0101 (5 in dec.). LED for B+1 > A & B+1 >= A are lit up.
Top right: B is set to 0100 & A set to 0101 (5 in dec.). LED for B+1 = A, B+1 >= A & B+1 <= A are lit up.
Bottom left: B is set to 1111 & A set to 0101 (5 in dec.). LED for Overflow is on.
Bottom right: B is set to 0000 & A set to 0101 (5 in dec.). LED for B+1 <= A & B+1 < A are lit up.

RTL picture of the comparator circuit:



5- Explanation of the results

As you may see in the previous questions, our results correspond with what you would expect from computing the logic statements by hand. You can refer to the previous questions where a thorough explanation of the results has been done for each picture.

6- Conclusion

In conclusion, this lab allowed us to get an idea of how the sequential assignment statements work. We had to write VHDL code for a 4-bit comparator, we tested that using a testbench and we also used the AlteraBoard to test it using the LEDs. We did the pin assignment for the LEDs exactly the same way as last time. Initially, we had difficulties coming up with the comparator code as we were trying to do it in a structural manner. It turned out that would've taken a bit more time than expected, so we tried the behavioral implementation and realized that type of implementation was much more straightforward. Finally, we found it interesting to implement functions that are much more alike regular programming language in VHDL which is far more hardware oriented.