CENTRO DE ENSEÑANZA TÉCNICA INDUSTRIAL

Materia:

Diseño de Circuitos Integrados Digitales CMOS I I

Nombre de la Tarea:

Practica 2 – Substractor

Nombre del estudiante:

Gonzalez Diaz Ariel

Nombre del Profesor:

Jose Roberto Reyes Baron

Fecha:

28 de Agosto del 2023

Introducción

In this practice we need to create a full Subtractor using our FPGA device and compile with VHDL language, for this is necessary have knowledge about of what is a full Subtractor how we made this configuration:

Subtractor

As their name implies, a **Binary Subtractor** is a decision making circuit that subtracts two binary numbers from each other, for example, X – Y to find the resulting difference between the two numbers.

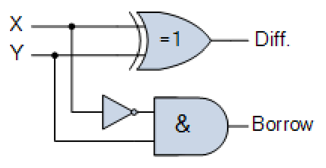
Unlike the Binary Adder which produces a SUM and a CARRY bit when two binary numbers are added together, the *binary subtractor* produces a DIFFERENCE, D by using a BORROW bit, B from the previous column. Then obviously, the operation of subtraction is the opposite to that of addition.

We learnt from our maths lessons at school that the minus sign, "–" is used for a subtraction calculation, and when one number is subtracted from another, a borrow is required if the subtrahend is greater than the minuend. Consider the simple subtraction of the two denary (base 10) numbers below.

Binary Subtraction

Binary Subtraction can take many forms but the rules for subtraction are the same whichever process you use. As binary notation only has two digits, subtracting a "0" from a "0" or a "1" leaves the result unchanged as 0-0 = 0 and 1-0 = 1. Subtracting a "1" from a "1" results in a "0", but subtracting a "1" from a "0" requires a borrow. In other words 0 – 1 requires a borrow.

Symbol

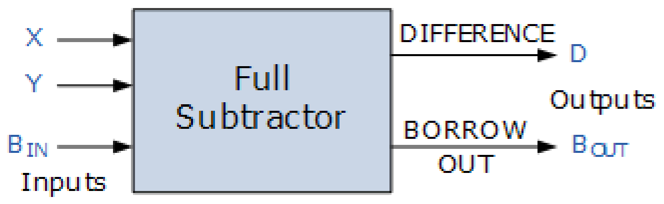


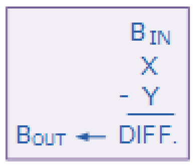
Truth Table

Y	X	DIFFERENCE	BORROW
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	0

Full Binary Subtractor

The main difference between the **Full Subtractor** and the previous **Half Subtractor** circuit is that a full subtractor has three inputs. The two single bit data inputs X (minuend) and Y (subtrahend) the same as before plus an additional *Borrow-in* (B-in) input to receive the borrow generated by the subtraction process from a previous stage as shown below.

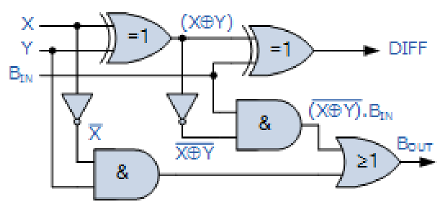




Then the combinational circuit of a “full subtractor” performs the operation of subtraction on three binary bits producing outputs for the difference D and borrow B-out. Just like the binary adder circuit, the full subtractor can also be thought of as two half subtractors connected together, with the first half subtractor passing its borrow to the second half subtractor as follows.

As the full subtractor circuit above represents two half subtractors cascaded together, the truth table for the full subtractor will have eight different input combinations as there are three input variables, the data bits and the *Borrow-in*, B_{IN} input. Also includes the difference output, D and the Borrow-out, B_{OUT} bit.

Full Subtractor Truth Table

Symbol	Truth Table				
	B-in	Y	X	Diff.	B-out
	0	0	0	0	0
	0	0	1	1	0
	0	1	0	1	1
	0	1	1	0	0
	1	0	0	1	1
	1	0	1	0	0
	1	1	0	0	1
	1	1	1	1	1

Equations

$$D = X \oplus Y \ \oplus BIN$$

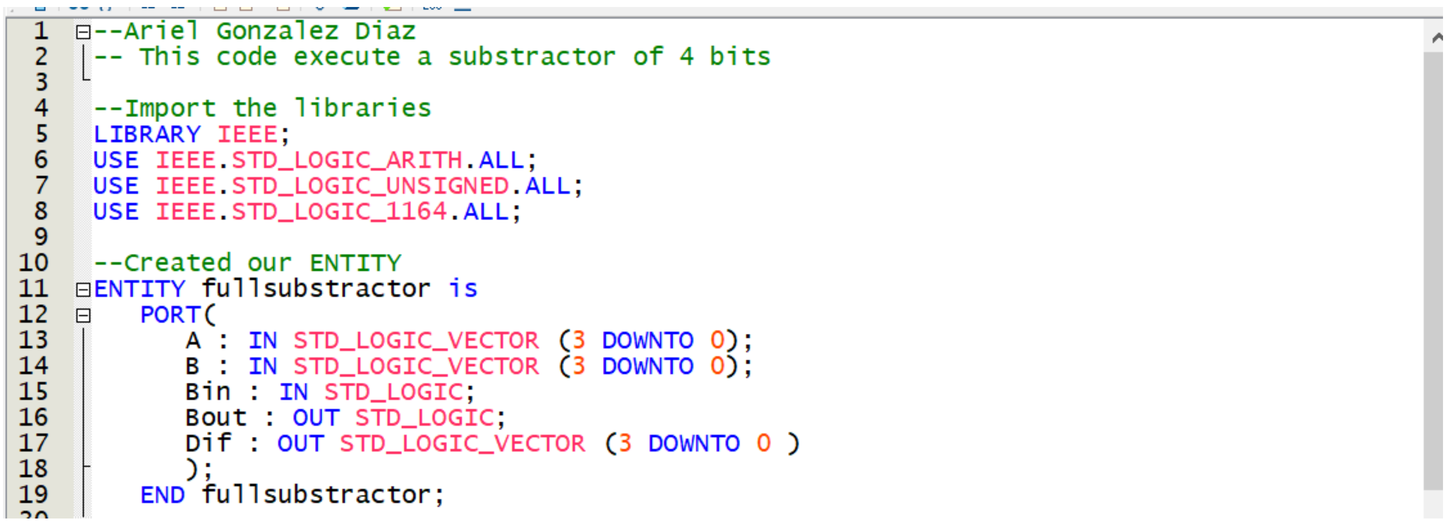
$$Bout = {}' \mathrm{X.Y} + {}'(\mathrm{X} \oplus \mathrm{Y})\mathrm{BIN}$$

Process

Once having this clear we can start with the development of our Full subtractor and analysis his comportment.

Now that we know what we will do we will use our knowledge obtained and develop our vhd file that allows us to declare the inputs and outputs in our design of our code which is as follows:

First is necessary created our project wizard and import the libraries IEEE, now we need to created our entity that's designed our Inputs and Outputs , and say if are a vector. In this case how we can see in the description we note that is necessary a 2 variables and other variable that is de Borrow bit , and in the outputs we have 2 variables that are the result (Diff) and the borrow bit of the Output.



```
--Ariel Gonzalez Diaz
-- This code execute a substractor of 4 bits

--Import the libraries
LIBRARY IEEE;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.STD_LOGIC_1164.ALL;

--Created our ENTITY
ENTITY fullsubstractor is
   PORT(
      A : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
      B : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
      Bin : IN STD_LOGIC;
      Bout : OUT STD_LOGIC;
      Dif : OUT STD_LOGIC_VECTOR (3 DOWNTO 0 )
      );
   END fullsubstractor;
```

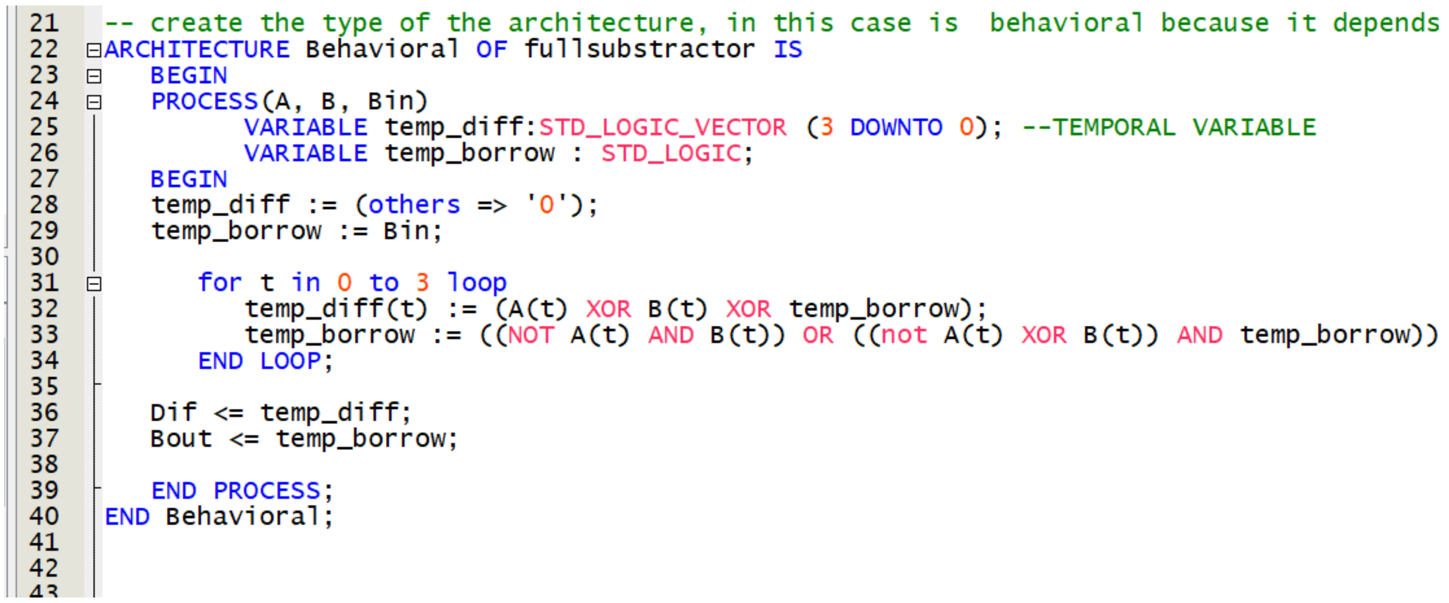
Now we need to continue to the code creating the type of the architecture that in this case is Behavioral because depend of the states for his functions and that why need to created a behavioral architecture , first we need to create a process , that this process depends of the 3 parameters that are the inputs (A, B, Bin)

The first variable is the temporal variable of the difference , that accumulate the result of the difference of the inputs.

The other variable is the temporal variable of the Borrow that accumulates 0 or 1

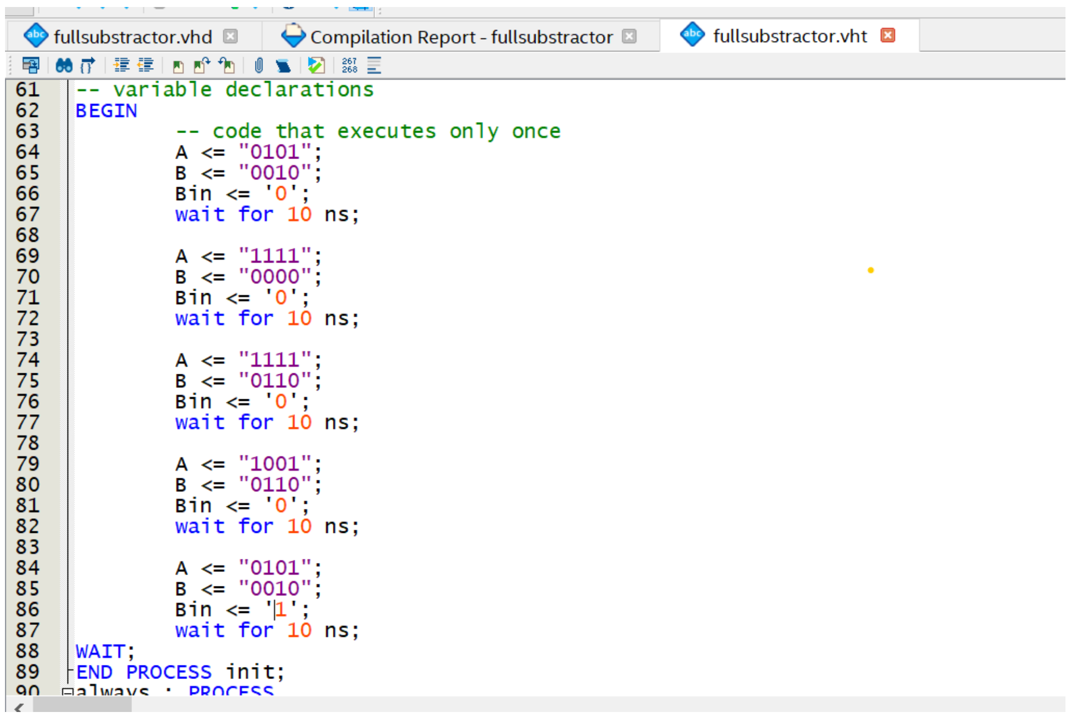
And now we need to create a for sentence because it is a subtractor of 4 bits

And finally, assignment this variables at the outputs and end the Architecture process



```vhdl
-- create the type of the architecture, in this case is  behavioral because it depends
ARCHITECTURE Behavioral OF fullsubstractor IS
   BEGIN
   PROCESS(A, B, Bin)
         VARIABLE temp_diff:STD_LOGIC_VECTOR (3 DOWNTO 0); --TEMPORAL VARIABLE
         VARIABLE temp_borrow : STD_LOGIC;
   BEGIN
   temp_diff := (others => '0');
   temp_borrow := Bin;

      for t in 0 to 3 loop
         temp_diff(t) := (A(t) XOR B(t) XOR temp_borrow);
         temp_borrow := ((NOT A(t) AND B(t)) OR ((not A(t) XOR B(t)) AND temp_borrow))
      END LOOP;

   Dif <= temp_diff;
   Bout <= temp_borrow;

   END PROCESS;
END Behavioral;
```

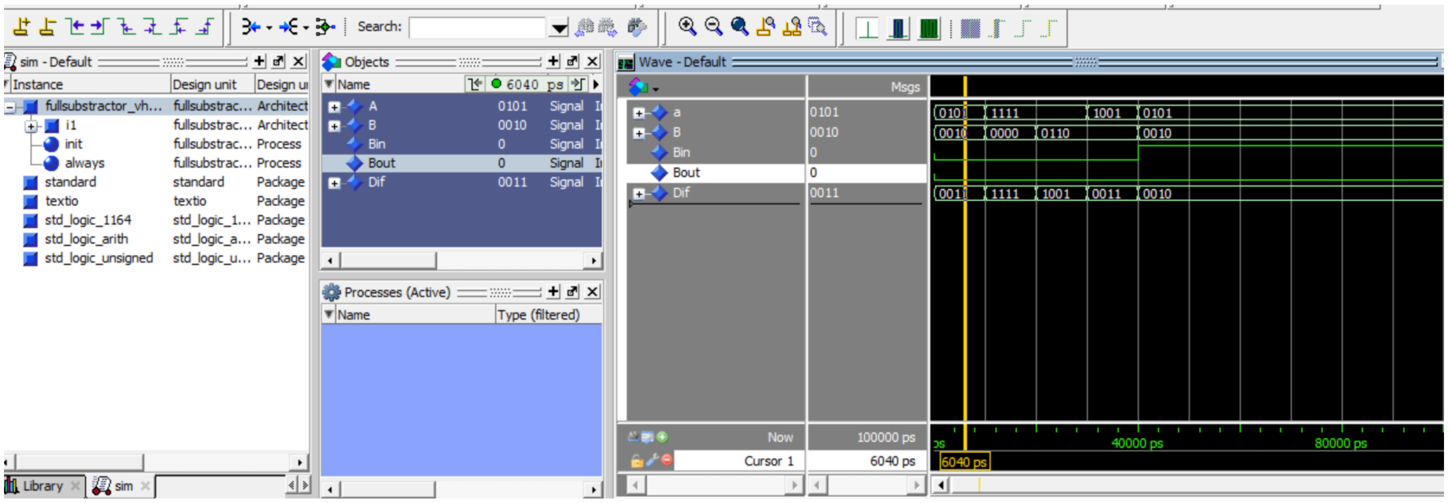
Now this is all the logic that needs to created our Full subtractor and now we need create a Test Bench template Writer and see if the logic is correct , with this file we can probe un a RTL simulation have a view that describes the comportament that have the program.



```vhdl
-- variable declarations
BEGIN
         -- code that executes only once
         A <= "0101";
         B <= "0010";
         Bin <= '0';
         wait for 10 ns;

         A <= "1111";
         B <= "0000";
         Bin <= '0';
         wait for 10 ns;

         A <= "1111";
         B <= "0110";
         Bin <= '0';
         wait for 10 ns;

         A <= "1001";
         B <= "0110";
         Bin <= '0';
         wait for 10 ns;

         A <= "0101";
         B <= "0010";
         Bin <= '1';
         wait for 10 ns;
WAIT;
END PROCESS init;
always : PROCESS
```

In this picture we can see that have 5 possible cases and check with the simulation this states that we describe in the test bench.

If we check in the simulation we have something similar like this



Instance	Design unit	Design u
fullsubstractor_vh...	fullsubstrac...	Architect
i1	fullsubstrac...	Architect
init	fullsubstrac...	Process
always	fullsubstrac...	Process
standard	standard	Package
textio	textio	Package
std_logic_1164	std_logic_1...	Package
std_logic_arith	std_logic_a...	Package
std_logic_unsigned	std_logic_u...	Package

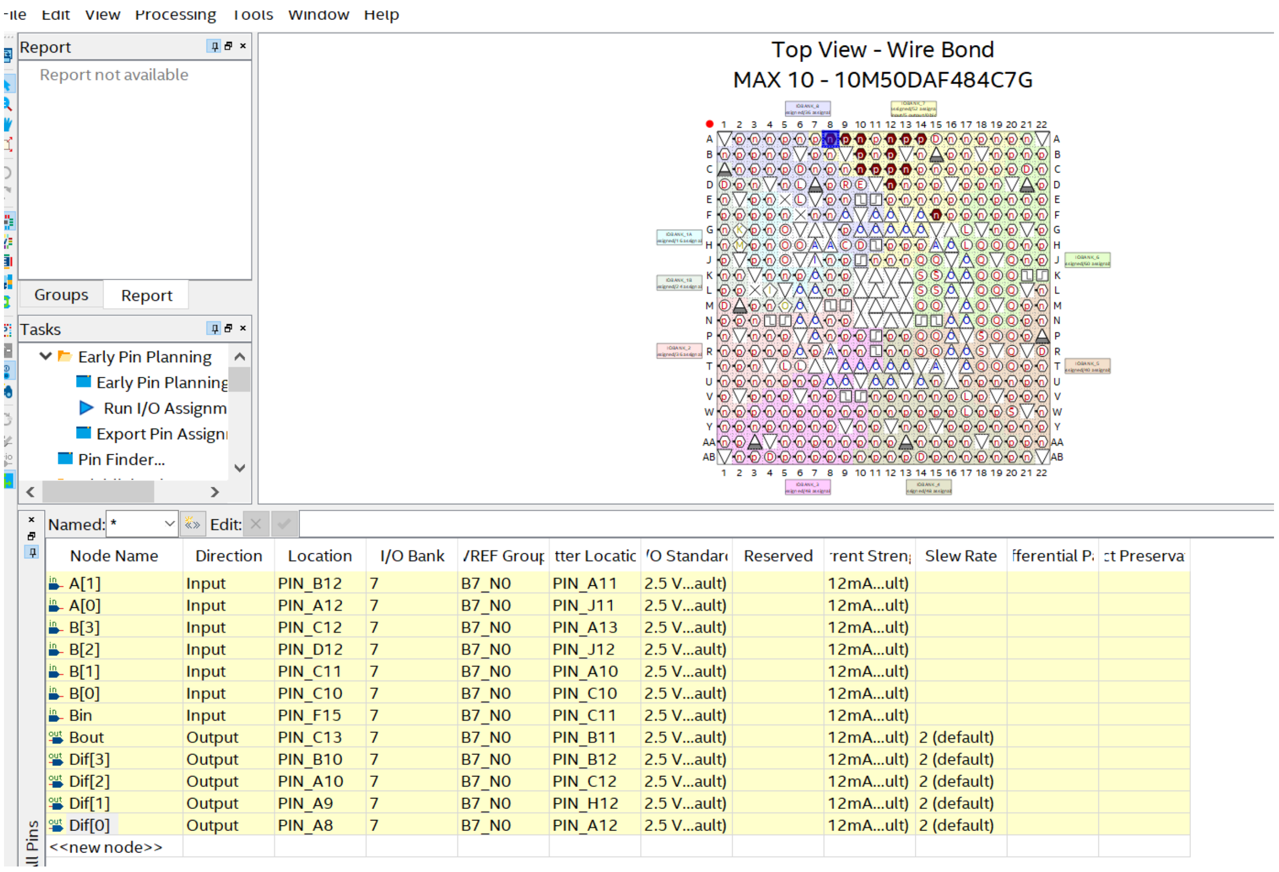
Name	6040 ps	
A	0101	Signal I
B	0010	Signal I
Bin	0	Signal I
Bout	0	Signal I
Dif	0011	Signal I

	Msgs
a	0101
B	0010
Bin	0
Bout	0
Dif	0011
Now	100000 ps
Cursor 1	6040 ps

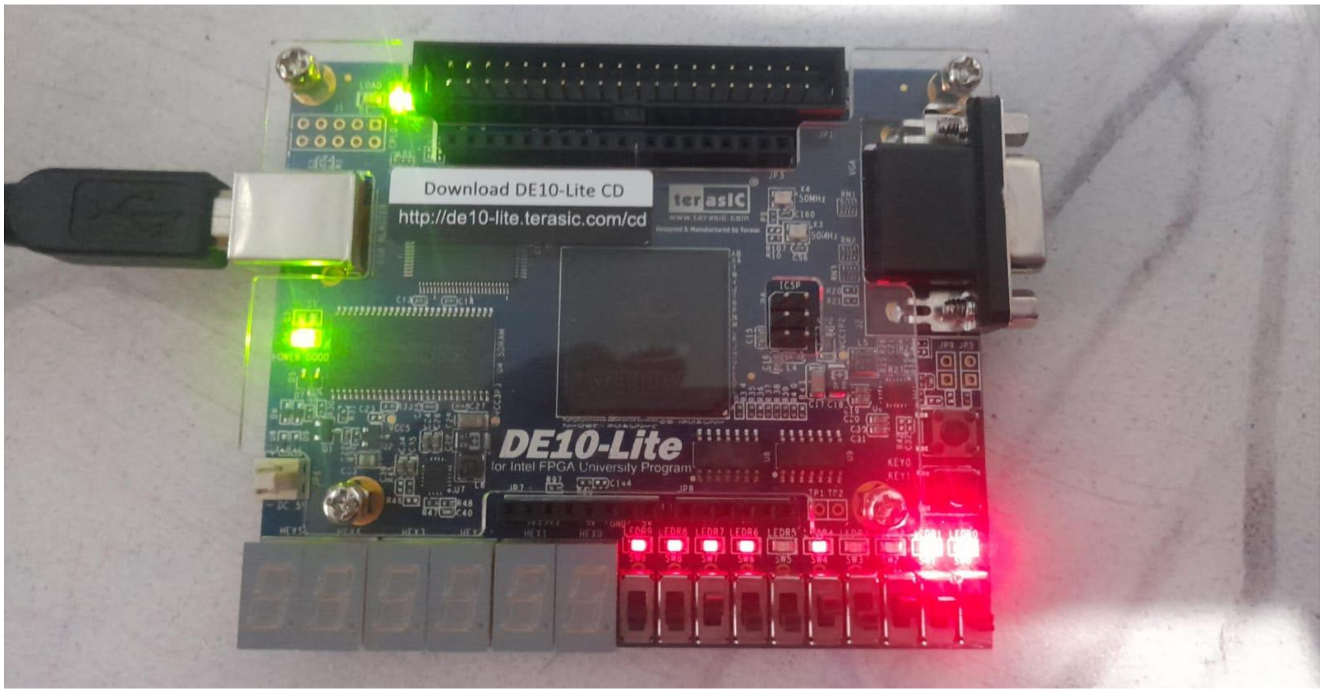
```
a    0101 1111 1001 0101
B    0010 0000 0110 0010
Dif  0011 1111 1001 0011 0010
     ps   40000 ps   80000 ps
     6040 ps
```

How we can see , we have the result that we hope and prove that the code is successful and because all the test are correct

Now the finally step is assigned our inputs and outputs to the FPGA and recompile our program for the fisical test , we need go to the pin planner with Ctrl + Shift + N



Node Name	Direction	Location	I/O Bank	/REF Group	tter Locatic	'O Standard	Reserved	rent Stren;	Slew Rate	fferential P.	ct Preserva
A[1]	Input	PIN_B12	7	B7_N0	PIN_A11	2.5 V...ault)		12mA...ult)			
A[0]	Input	PIN_A12	7	B7_N0	PIN_J11	2.5 V...ault)		12mA...ult)			
B[3]	Input	PIN_C12	7	B7_N0	PIN_A13	2.5 V...ault)		12mA...ult)			
B[2]	Input	PIN_D12	7	B7_N0	PIN_J12	2.5 V...ault)		12mA...ult)			
B[1]	Input	PIN_C11	7	B7_N0	PIN_A10	2.5 V...ault)		12mA...ult)			
B[0]	Input	PIN_C10	7	B7_N0	PIN_C10	2.5 V...ault)		12mA...ult)			
Bin	Input	PIN_F15	7	B7_N0	PIN_C11	2.5 V...ault)		12mA...ult)			
Bout	Output	PIN_C13	7	B7_N0	PIN_B11	2.5 V...ault)		12mA...ult)	2 (default)		
Dif[3]	Output	PIN_B10	7	B7_N0	PIN_B12	2.5 V...ault)		12mA...ult)	2 (default)		
Dif[2]	Output	PIN_A10	7	B7_N0	PIN_C12	2.5 V...ault)		12mA...ult)	2 (default)		
Dif[1]	Output	PIN_A9	7	B7_N0	PIN_H12	2.5 V...ault)		12mA...ult)	2 (default)		
Dif[0]	Output	PIN_A8	7	B7_N0	PIN_A12	2.5 V...ault)		12mA...ult)	2 (default)		
<<new node>>											


Download DE10-Lite CD
http://de10-lite.terasic.com/cd
DE10-Lite
for Intel FPGA University Program

Once that we assignment our inputs and outputs in the FPGA we can programming the FPGA and executes the code.

Conclusion

In this practice we were able to observe the operation of a complete subtractor of 4 bits and how to obtain the equations to make it work, in addition to understanding in a better way how to use our FPGA and have one of the multiple functions that it has.

https://www.electronics-tutorials.ws/combination/binary-subtractor.html

Full Subtractor in Digital Logic. (2017, October 10). GeeksforGeeks; GeeksforGeeks. https://www.geeksforgeeks.org/full-subtractor-in-digital-logic/

Full Subtractor | Definition | Circuit Diagram | Truth Table | Gate Vidyalay. (n.d.). https://www.gatevidyalay.com/full-subtractor/

BYJU'S Exam Prep. (2022). @Gradeupapp. https://byjusexamprep.com/full-subtractor-i

Virtual Labs. (2023). Vlabs.ac.in. https://de-iitr.vlabs.ac.in/exp/half-full-subtractor/theory.html