Saif Ba Madhaf - 527028

VHDL Week 1

self-test chapter 2:

1. Type standard logic, STD_LOGIC is normally used to model a logic bit. What are the possible values for this type?

There are nine possible values for STD_LOGIC, U, X, 0, 1, Z, W, L, H, and "-". U is uninitialized and X is forced unknown. Z is high impedance. L and H are both are weak 0 and 1. "-" is don’t care.

2. How do you represent a bit in STD_LOGIC?

To represent a bit in STD_LOGIC you assign a value to it, such as assigning a 1 for high and 0 for low.

3. How do you represent a group (for example a word) in STD_LOGIC?

To represent a group like words you need to use STD_LOGIC_VECTOR, which is a one-dimensional array of STD_LOGIC bits.

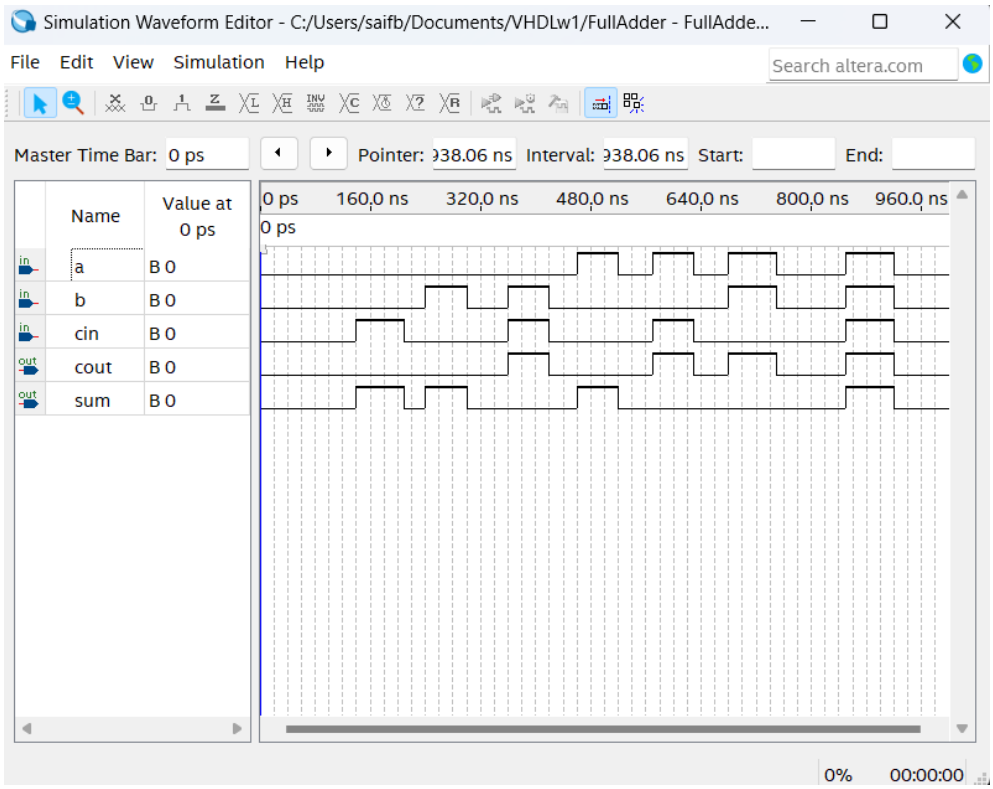
4. What is the essential difference between a HDL and a computer language?

The difference between them is that HDL is used for simulating electronic circuits, and computer language is used to write software instruction to the processor.

5. Why are some basic mathematical operators not easy for implementing in a chip?

They are difficult to implement because they require a lot of hardware resources. They additionally require a lot of complex logic gates compared to addition and subtraction. There algorithms are also very difficult to do.

Exercise 1: Full Adder



This is the full adder simulation in Quartus. In the picture above we can see all possible outcomes. As we know a full adder adds three binary inputs in our case A, B, and Cin it the produces 2 outputs our sum and carry out. An example of adding three binary inputs would be adding three ones:

1 1(carry out)

1

1

-

1 (sum)

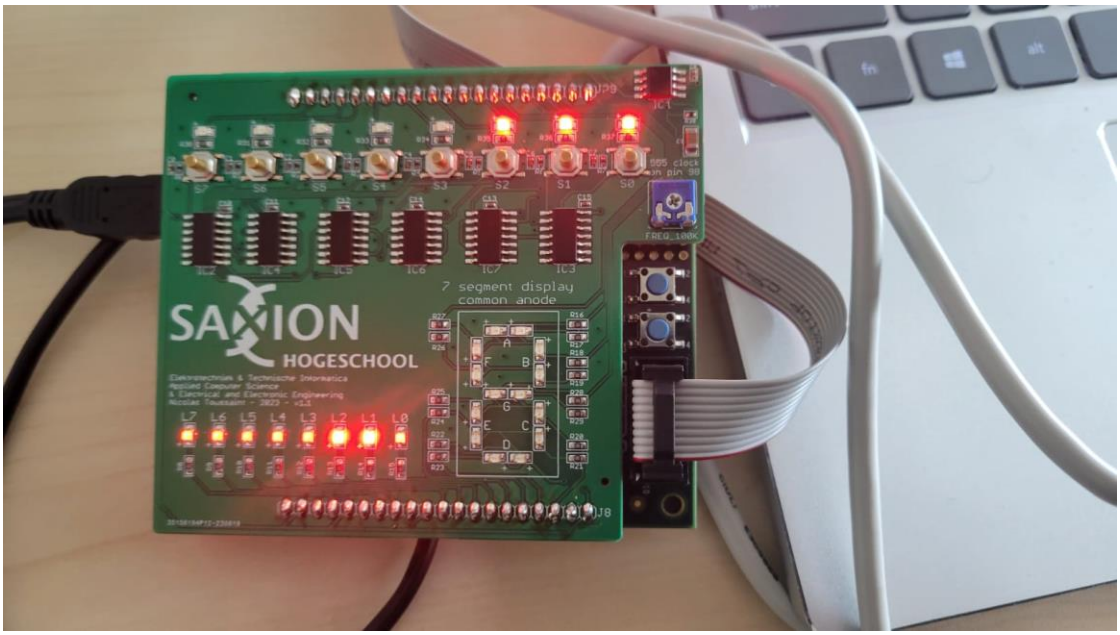
Adding two ones in binary is zero with a carry out 1. In the example above we are adding three ones which is 1(sum) and 1 carry out.

Table Full Adder

Inputs			Outputs	
A	B	C – IN	Sum	C - Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

FPGA

In the FPGA, we stored A as S0, B as S1, and Carry in as S2. My outputs are L1 (LED 1) carry out and L2 (LED 2) sum. As we can see A and B are both ones and carry in is 0, which means that the sum will be 0 be zero and carry out will be one as shown in the table above.



In this example, all inputs are 1 meaning that both carry out and sum should be 1 as shown in the table above.

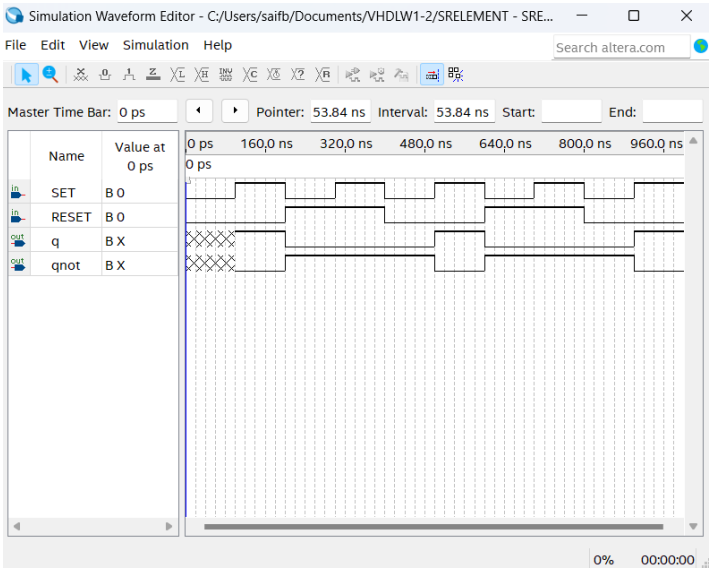
Code for exercise 1:

```
library ieee;
use ieee.std_logic_1164.all;

ENTITY FullAdder IS
    PORT (
        a, b, cin : IN std_logic;
        sum, cout : OUT std_logic
    );
END FullAdder;

ARCHITECTURE behaviour OF FullAdder IS
    signal c1, c2: std_logic;
BEGIN
    c1 <= (a AND b) OR (cin AND (a XOR b));
    sum <= a XOR b XOR cin;
    c2 <= (a AND b);
    cout <= c1 OR c2;
END behaviour;
```

Exercise 2: S/R Element



Above we can see the simulation for the s/r element. All possible outcomes are displayed. In the s/r element if the set is 1 and the reset is 0 the q would be 1 and the qnot is always the opposite of the q which is in this case 0. If set and reset have the same values ie 1 and 1 or 0 and 0 the q and qnot are what they were previously.

Code

```
library ieee;
use ieee.std_logic_1164.all;

ENTITY SRELEMENT IS

PORT(
       SET, RESET : IN std_logic;
       q, qnot : OUT std_logic
       );
END SRELEMENT;

ARCHITECTURE behaviour OF SRELEMENT IS

BEGIN

process(SET, RESET)
    begin
        if (SET = '1' and RESET = '0') then
            q <= '1';
            qnot <= '0';
        elsif (SET = '0' and RESET = '1') then
            q <= '0';
            qnot <= '1';
         elsif (SET = '0' and RESET = '0') then
         elsif (SET = '0' and RESET = '1') then

         end if;
    end process;
END behaviour;
```

FPGA

Set is s0 and our reset is s1. Led 0 is our q and led 1 is our qnot.

As we can see here, when we set our set 1 and reset to 0 our q is on and qnot 0.

Over here we can see that both set and reset are 1, which means that it is invalid. It automatically sets the q and qnot to what they previously were as seen in the image above this one.

References

GfG. (2023, August 7). *Full Adder in digital logic*. GeeksforGeeks. https://www.geeksforgeeks.org/full-adder-in-digital-logic/