Name: Richard Raymond J. Canda

C.Y.S.: BSCpE – 3A

VIVA QUESTIONS:

1. Implement the following function using VHDL coding. (Try to minimize if you can).

F(A,B,C,D) = (A′+B+C) . (A+B′+D′). (B+C′+D′) . (A+B+C+D)

Answer:

The simplified version:

F(A,B,C,D) = AB + D'(B + C) + A'B'C'D

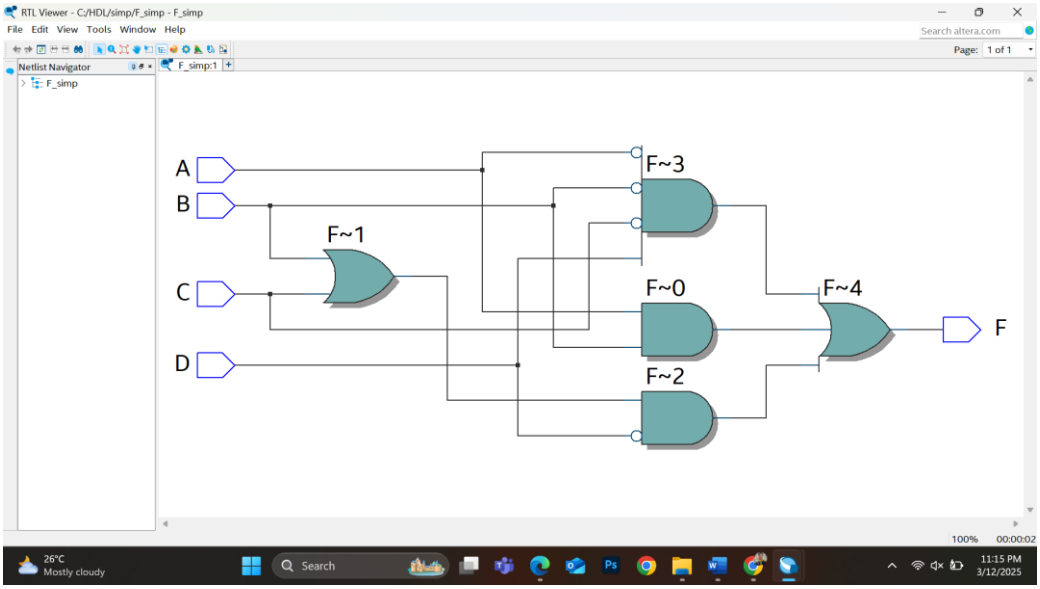
The **CODE**:

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

entity F_simp is
    Port ( A, B, C, D : in STD_LOGIC;
           F : out STD_LOGIC);
end F_simp;

architecture Behavioral of F_simp is
begin
    process(A, B, C, D)
    begin
        F <= (A and B) or (not D and (B or C)) or (not A and not B and not C and D);
    end process;
end Behavioral;
```

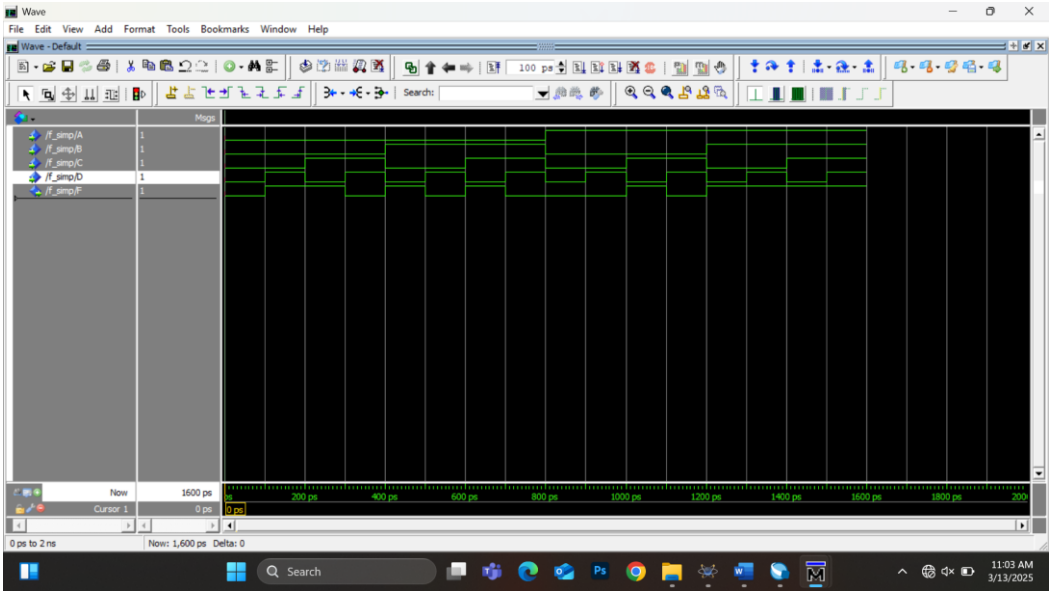
F_simp.vhd | Compilation Report - F_simp

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity F_simp is
5      Port ( A, B, C, D : in STD_LOGIC;
6             F : out STD_LOGIC);
7  end F_simp;
8
9  architecture Behavioral of F_simp is
10 begin
11     process(A, B, C, D)
12     begin
13         F <= (A and B) or (not D and (B or C)) or (not A and not B and not C and D)
14     end process;
15 end Behavioral;
16
```

The **LOGIC GATES**:



The **WAVEFORMS**:



2. What will be the no. of rows in the truth table of N variables?

Answer:

The number of rows in the truth table of N variables can be determine using the formula **2^n**. Where n is the number of variables. For instance, if the given has 2 variables, then **2^2 = 4 rows**.

3. What are the advantages of VHDL?

Answer:

VHDL (VHSIC Hardware Description Language) offers several advantages in digital circuit design. It allows for designing, simulating, and verifying complex digital systems before actual hardware implementation, reducing development time and cost. VHDL is independent of specific hardware, making it highly portable and reusable across different FPGA and ASIC platforms. It supports both behavioral and structural modeling, enabling designers to describe a system at various abstraction levels. Additionally, VHDL allows concurrent execution, which closely represents real-world hardware behavior. Its strong type-checking and modular approach enhance design reliability and maintainability, making it ideal for large-scale and critical applications.

4. Design Ex-OR gate using behavioral model?

Answer:

THE CODE

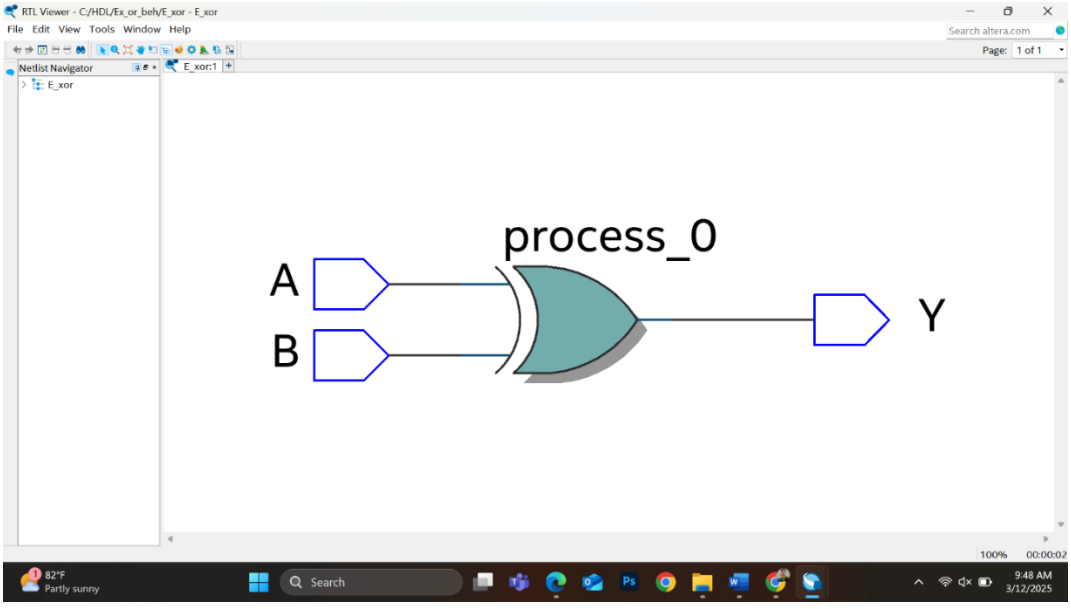
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity E_xor is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           Y : out STD_LOGIC);
end E_xor;

architecture Behavioral of E_xor is
begin
    process(A, B)
    begin
```

```
        if A = B then
            Y <= '0';
        else
            Y <= '1';
        end if;
    end process;
end Behavioral;
```

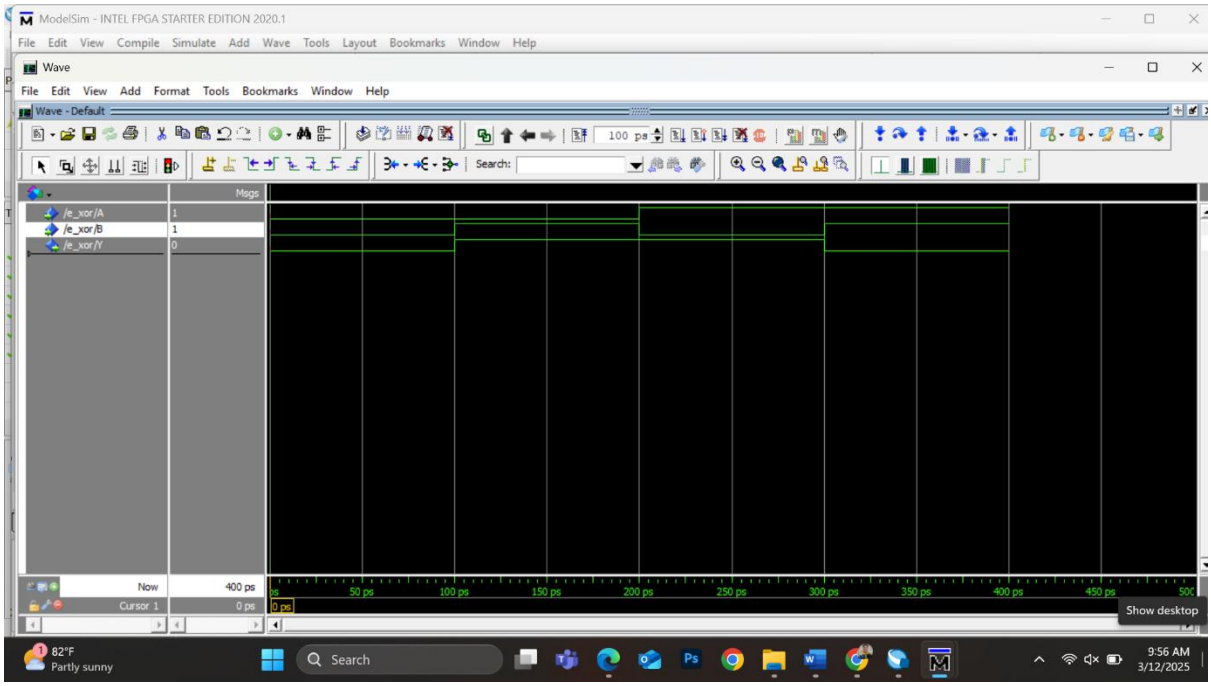
E_xor.vhd | Compilation Report - E_xor

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity E_xor is
5      Port ( A : in STD_LOGIC;
6             B : in STD_LOGIC;
7             Y : out STD_LOGIC);
8  end E_xor;
9
10 architecture Behavioral of E_xor is
11 begin
12     process(A, B)
13     begin
14         if A = B then
15             Y <= '0';
16         else
17             Y <= '1';
18         end if;
19     end process;
20 end Behavioral;
21
```

THE LOGIC GATE:



THE WORKBENCH RESULTS:

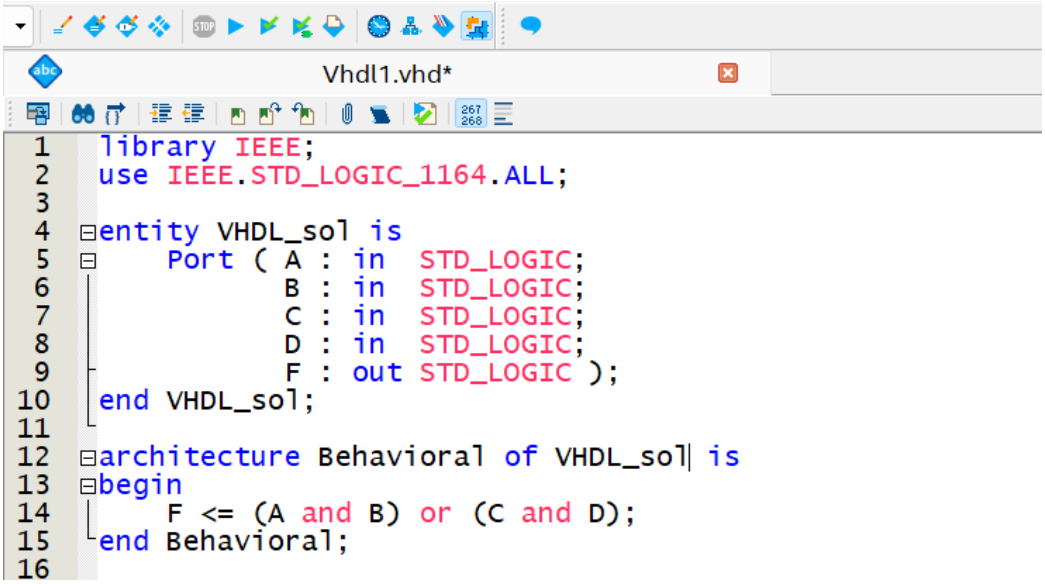


5. Implement the following function using VHDL code

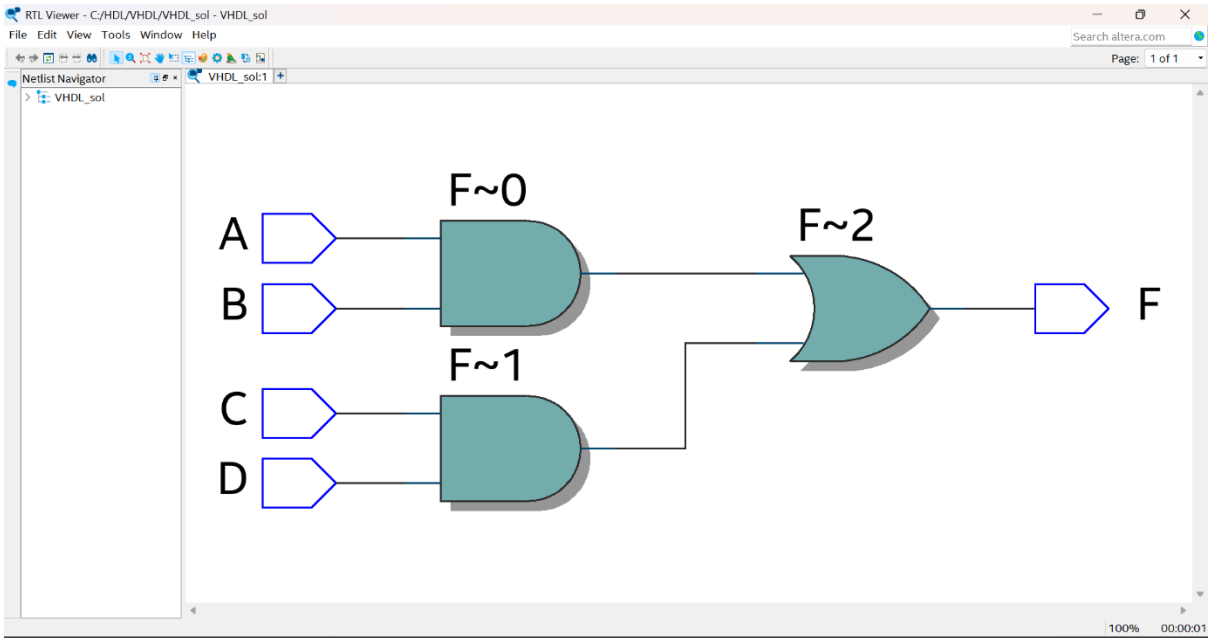
F = AB + CD.

Answer:

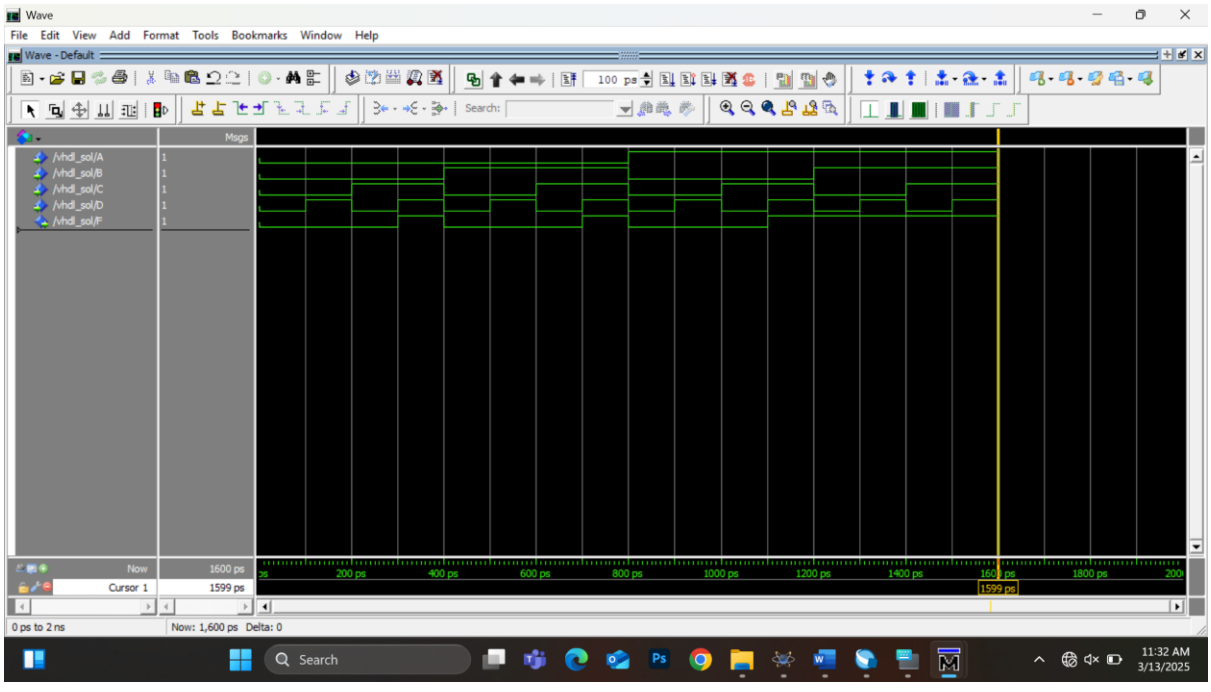
The **CODE**:



THE LOGIC GATE:



THE WORKBENCH RESULTS:



6. What are the differences between half adder and full adder?

Answer:

A half adder adds two binary digits (A and B) and produces a sum and a carry-out, but it cannot handle carry input from a previous stage. This makes it suitable for simple addition but not for multi-bit operations. In contrast, a full adder adds three inputs: two binary digits (A and B) and an additional carry-in from a previous stage. This allows it to be used in multi-bit binary addition, such as in ripple carry adders. The half adder consists of an XOR gate for the sum and an AND gate for the carry, while the full adder requires two XOR gates, two AND gates, and an OR gate to handle the additional carry-in.

7. What are the advantages of minimizing the logical expressions?

Answer:

A minimized Logical expression is more simple ang easier to implement. When the logical expressions are minimized, complexity is alleviated. Another benefit ss it reduces the number of logic gates required, leading to lower hardware costs and simpler circuit implementation. A minimized expression also decreases power consumption since fewer gates mean lower energy usage. Additionally, it improves the circuit's speed by reducing propagation delay, which is crucial for high-speed applications. Simplified circuits are also more reliable and easier to troubleshoot, as they contain fewer components that could fail. Furthermore, minimizing logical expressions helps in optimizing space on an integrated circuit, making the design more efficient for compact devices.

8. What does a combinational circuit mean?

Answer:

A combinational circuit is a type of digital circuit where the output is determined solely by the current inputs, without any dependence on previous inputs or stored data. These circuits do not have memory elements and operate purely based on logic gates such as AND, OR, NOT, XOR, NAND, and NOR. Since they do not store past values, their response to a given set of inputs is immediate and always the same. Some common examples of combinational circuits include adders, multiplexers, demultiplexers, encoders, decoders, and comparators, all of which perform specific logical or arithmetic functions based on the given inputs.

9. Implement the half adder using VHDL code?

Answer:

The **CODE**:

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

entity Half_add is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           Sum : out STD_LOGIC;
           Carry : out STD_LOGIC);
end Half_add;

architecture Behavioral of Half_add is
begin
    -- Sum is the XOR of A and B
    Sum <= A XOR B;

    -- Carry is the AND of A and B
    Carry <= A AND B;
end Behavioral;
```

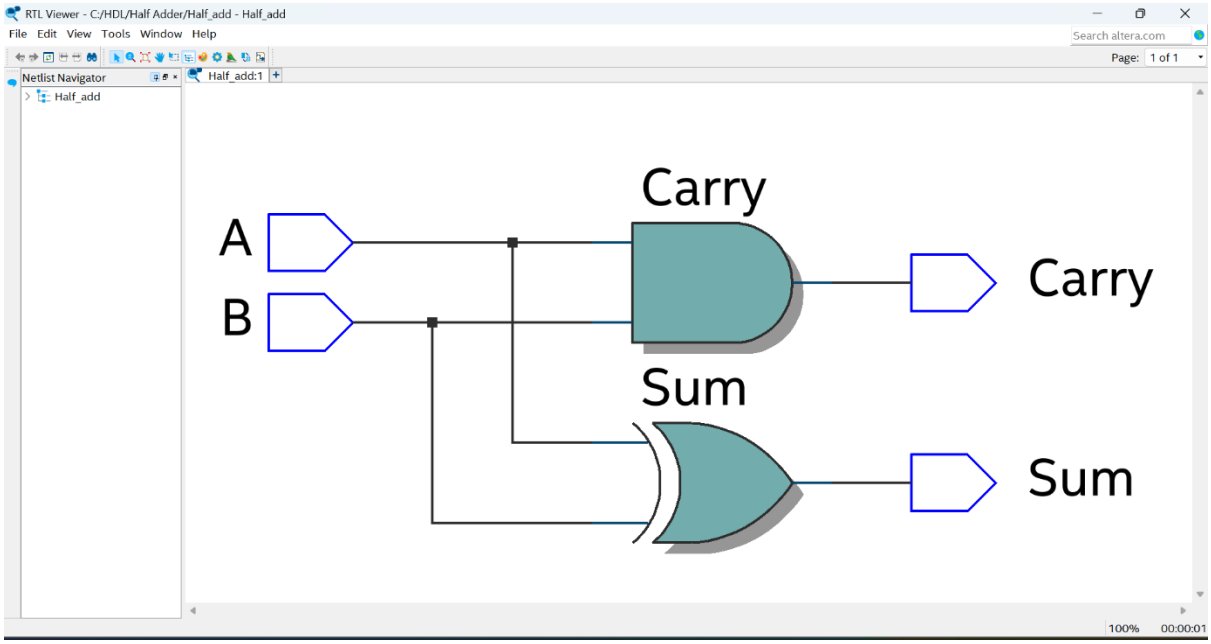
THE **CODE**:

```
sing  Tools  Window  Help
```

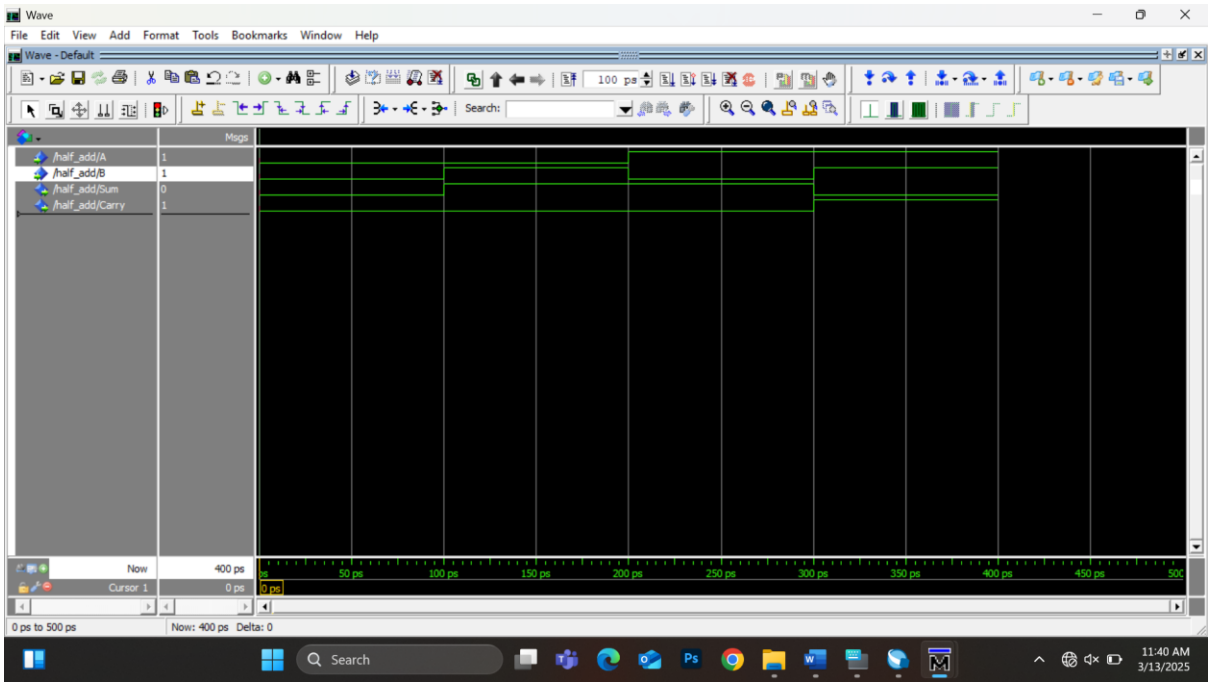
Half_add.vhd

```
1  -- Half Adder VHDL Implementation
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5  entity Half_add is
6      Port ( A : in  STD_LOGIC;
7             B : in  STD_LOGIC;
8             Sum : out STD_LOGIC;
9             Carry : out STD_LOGIC);
10 end Half_add;
11
12 architecture Behavioral of Half_add is
13 begin
14     -- Sum is the XOR of A and B
15     Sum <= A XOR B;
16
17     -- Carry is the AND of A and B
18     Carry <= A AND B;
19 end Behavioral;
20
```

THE LOGIC GATE:



THE WORKBENCH RESULTS:



10. Implement the full adder using two half adders and write VHDL program in structural model?

Answer:

THE **CODE**:

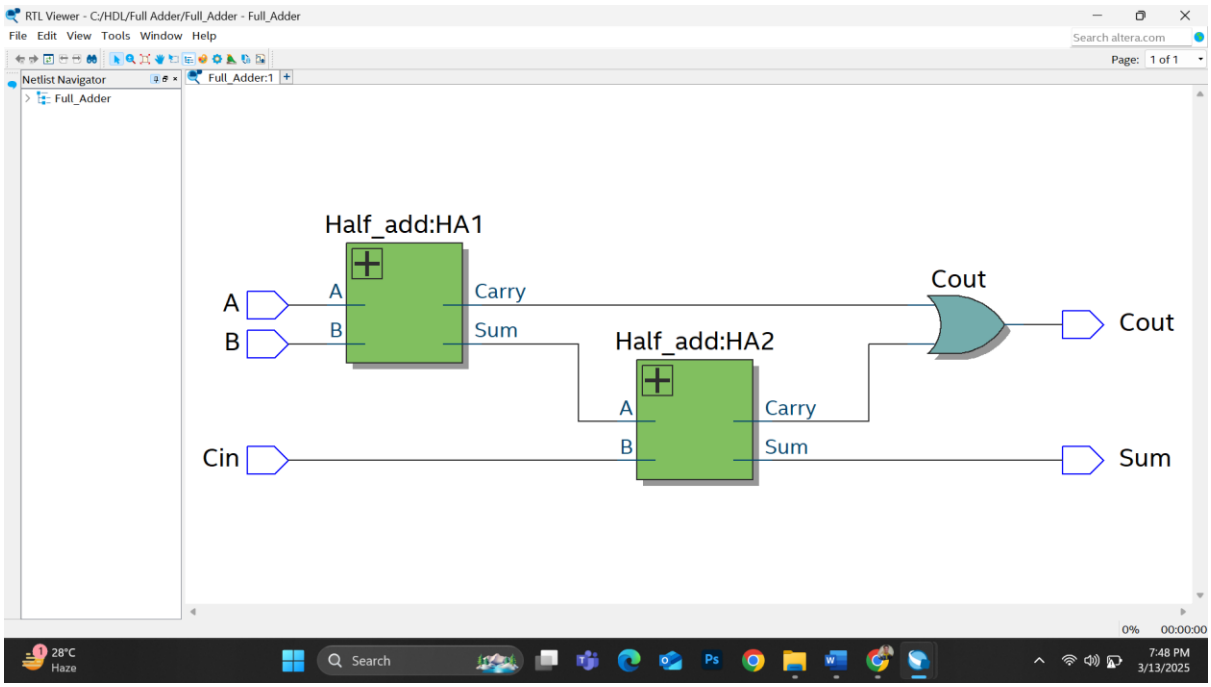
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Full_Adder is
    Port (
        A, B, Cin : in  STD_LOGIC;
        Sum, Cout : out  STD_LOGIC
    );
end Full_Adder;
architecture Structural of Full_Adder is
    -- Component declaration for the Half Adder
    component Half_add
```

```
    Port (
        A, B : in  STD_LOGIC;
        Sum, Carry : out  STD_LOGIC
    );
end component;
-- Internal signals to connect Half Adders
signal Sum1, Carry1, Carry2 : STD_LOGIC;
begin
-- First Half Adder: Adds A and B
HA1: Half_add port map(A => A, B => B, Sum => Sum1, Carry => Carry1);

-- Second Half Adder: Adds Sum1 and Cin
HA2: Half_add port map(A => Sum1, B => Cin, Sum => Sum, Carry => Carry2);

-- OR gate to generate Cout
Cout <= Carry1 or Carry2;
end Structural;
```

Full_Adder.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Full_Adder is
5      Port (
6          A, B, Cin : in  STD_LOGIC;
7          Sum, Cout : out  STD_LOGIC
8      );
9  end Full_Adder;
10
11 architecture Structural of Full_Adder is
12     -- Component declaration for the Half Adder
13     component Half_add
14         Port (
15             A, B : in  STD_LOGIC;
16             Sum, Carry : out  STD_LOGIC
17         );
18     end component;
19
20     -- Internal signals to connect Half Adders
21     signal Sum1, Carry1, Carry2 : STD_LOGIC;
22
23 begin
24     -- First Half Adder: Adds A and B
25     HA1: Half_add port map(A => A, B => B, Sum => Sum1, Carry => Carry1);
26
27     -- Second Half Adder: Adds Sum1 and Cin
28     HA2: Half_add port map(A => Sum1, B => Cin, Sum => Sum, Carry => Carry2);
29
```

THE LOGIC GATE:



THE WORKBENCH RESULTS: