Inverter Explanation

Harsha

May 22, 2025

1 Basic Inverter Code

This inverter uses a NOT gate to invert the input signal `a` and produce the output signal `y`.

```
entity test is
    port(
        a : in std_logic;
        y : out std_logic
    );
end entity test;

architecture behavior of test is
begin
    y <= not a;
end architecture behavior;
```

The **entity** declaration in VHDL defines the external interface of the circuit, including the input and output ports. It tells the compiler what signals the module will receive and produce.

The **architecture** describes the internal implementation or behavior of the circuit declared in the entity. In this example, it specifies how the output `y` is assigned the inverted value of input `a`.

Note: The VHDL file in the project name must match the entity name (e.g., `test.vhd` should define `entity test`).

2 Structural Modelling

Structural modelling in VHDL describes a digital system by specifying how components are connected together. It is similar to drawing a schematic where various gates or modules are wired together. Each component must be declared, instantiated, and connected using port maps.

Below is an example showing the use of structural modelling using a custom NAND gate component to create an AND gate.

2.1

Package Declaration Defines the reusable component:

```
library ieee;
use ieee.std\_logic\_1164.all;

package Gates is
component NAND\_2 is
port (A, B: in std\_logic; Y: out std\_logic);
end component NAND\_2;
end package Gates;
```

2.2

NAND Gate Entity and Architecture This is the implementation of the NAND gate.

```
library ieee;
use ieee.std\_logic\_1164.all;

entity NAND\_2 is
port (A, B: in std\_logic; Y: out std\_logic);
end entity NAND\_2;

architecture Equations of NAND\_2 is
begin
Y <= not (A and B);
end Equations;
```

2.3

Using Structural Modelling to Build AND Gate with NAND Gates This example uses two NAND gates to create an AND gate:

```
entity and\_nand is
port( a, b : in std\_logic;
y : out std\_logic);
end entity;

architecture internal of and\_nand is
signal s1: std\_logic;
begin
inst1: NAND\_2 port map(a => a, b => b, y => s1);
inst2: NAND\_2 port map(a => s1, b => s1, y => y);
end architecture;
```

Explanation: This structure instantiates the same NAND gate component twice. The first gate produces an intermediate result `s1`, which is then fed to both inputs of the second NAND gate to produce the final output `y`. This results in an AND gate behavior using only NAND components.

3 Try the following things out

- Make an OR gate, XOR gate, AND gate starting from defining a NAND gate as package and through port mapping only! (The VHDL file for this package is available on repo, .txt file named package)
- Design a 2-1 MUX
- Design a 4-1 MUX using the above 2-1 MUX as a component
- Design a 5 input Majority circuit which is 1 if 3 or more inputs are high