Digital System Design Lab Report, Lab3, Homework2

Author: Ying Yiwen
Number: 12210159

Abstract

This lab report delves into the differences between signal and variable types in VHDL by designing and simulating a digital circuit. In the experiment, we implemented the same logic function using signals and variables respectively and thoroughly analyzed the behavior of the circuit through behavioral simulation, schematic simulation, post-synthesis timing simulation and post-implementation timing simulation. The results show that variables are updated immediately during the process and are suitable for internal computation, while signals are updated at the end of the simulation cycle and are suitable for inter-component communication and hardware connection modeling. The experiments verify the correctness of the VHDL code and reveal the different characteristics and application scenarios of signals and variables in digital system design, providing a practical basis for understanding and applying these two data types.

Contents

1 Introduction

In digital system design, particularly when working with hardware description languages like VHDL, understanding the distinction between signals and variables is crucial. This experiment delves into the practical application of VHDL, focusing on the differences between signal and variable types through the implementation and simulation of a digital circuit.

Signals in VHDL are used for communication between different parts of a design. They maintain their values until explicitly updated and are suitable for representing hardware connections. Signals are updated at the end of a simulation cycle, making them ideal for modeling hardware components that maintain their state until new values are assigned.

Variables, on the other hand, are used within processes and retain their values only during the execution of the process. They are updated immediately within the process and are ideal for internal calculations and temporary storage within a process. Variables are not visible outside the process they are declared in, making them suitable for local computations.

This lab aims to clarify these distinctions by implementing a digital circuit using both signal and variable types. The VHDL code for the circuit, along with a testbench for verification, is provided. Theoretical analysis of the circuit's behavior is conducted, and simulation results from various stages—behavioral simulation, schematic simulation, post-synthesis timing simulation, and post-implementation timing simulation—are presented and compared. This comprehensive approach not only reinforces theoretical knowledge but also enhances practical skills in VHDL coding and digital system simulation.

2 VHDL Code

The VHDL code is as follows:

```vhdl
-- import library
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- entity declaration
entity sig_var is
Port ( x : in STD_LOGIC;
        y : in STD_LOGIC;
        z : in STD_LOGIC;
        res1 : out STD_LOGIC;
        res2 : out STD_LOGIC);
end sig_var;

-- architecture declaration
architecture Behavioral of sig_var is

-- signal declaration
signal sig_s1, sig_s2 : std_logic;

-- logic function
begin

-- process declaration, variable type
proc1: process (x, y, z) is
variable var_s1, var_s2 : std_logic;
```

```
begin
    var_s1 := x and y;
    var_s2 := var_s1 xor z;
    res1 <= var_s1 nand var_s2;
end process proc1;

-- process declaration, signal type
proc2: process (x, y, z) is
begin
    sig_s1 <= x and y;
    sig_s2 <= sig_s1 xor z;
    res2 <= sig_s1 nand sig_s2;
end process proc2;

end Behavioral;
```

3 Testbench Code

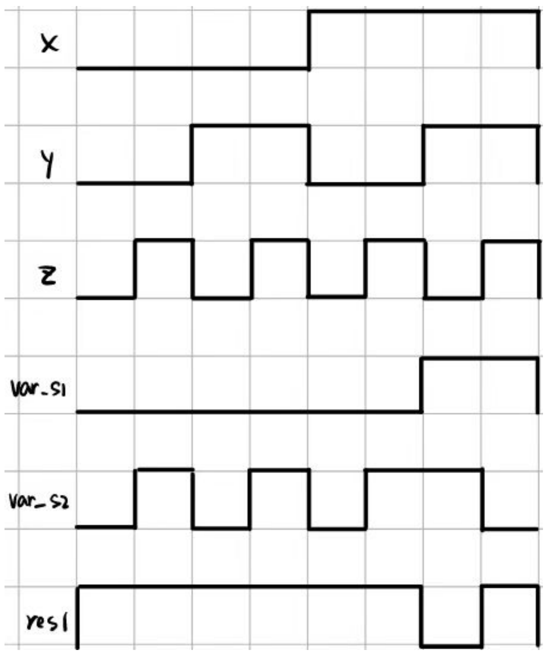
The testbench code is as follows, and it can test the correctness of the VHDL code.

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

-- entity declaration
entity testbench is
--  Port ( );
end testbench;

-- architecture declaration
architecture tb of testbench is
-- component declaration
component sig_var is
Port ( x : in STD_LOGIC;
        y : in STD_LOGIC;
        z : in STD_LOGIC;
        res1 : out STD_LOGIC;
        res2 : out STD_LOGIC);
end component;

-- signal and constant declaration
signal x, y, z, r1, r2 : std_logic ;
constant period: time := 10ns;

-- logic function
begin
-- instantiation
uut: sig_var port
    map( x => x,
            y => y,
            z => z,
            res1 => r1,
            res2 => r2);
```

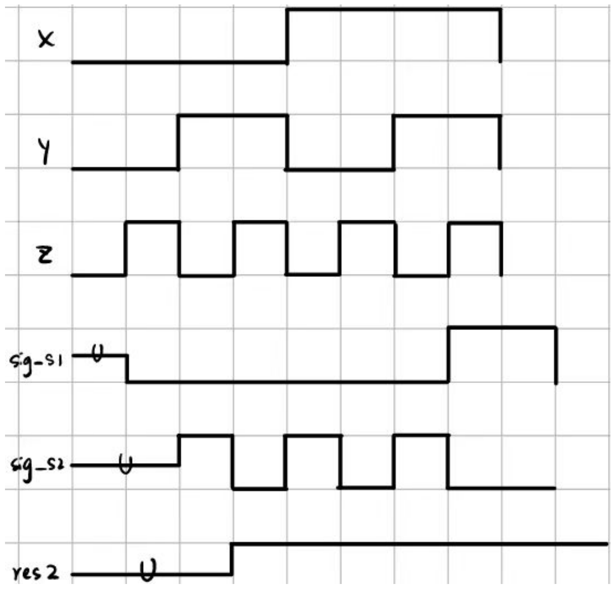
```
-- test case
x <= '0' after period * 0, '1' after period * 4;
y <= '0' after period * 0, '1' after period * 2, '0' after period * 4, '1' after
   period * 6;
z <= '0' after period * 0, '1' after period * 1, '0' after period * 2, '1' after
   period * 3,
        '0' after period * 4, '1' after period * 5, '0' after period * 6, '1' after
           period * 7;

end tb;
```

4 Therotical Analysis

The theoretical analysis is as follows:



(a) register1



(b) register2

Figure 1: Theoretical Analysis

In VHDL, variables and signals are both used to store data, but they have distinct characteristics and applications, especially evident in digital system design and simulation.

Definition and Declaration
Variables are declared within a process and are used for internal calculations and temporary storage within that process. They are assigned values using the ":=" operator. Signals, declared in the architecture declarative part, represent hardware connections and are updated using the "<=" operator.

Scope and Visibility
Variables are only visible within the process they are declared in. This limited scope makes them suitable for local computations where data needs to be stored temporarily during process execution. Signals, however, are visible throughout the entire design unit, allowing different components and processes to communicate by sharing signal

values.

Update Mechanism
Variables are updated immediately within the process. When a variable is assigned a new value, it is available for use in subsequent statements within the same process execution. Signals are updated at the end of a simulation cycle. The new value assigned to a signal doesn't take effect until the current simulation cycle completes, which models the behavior of hardware components that maintain their state until new values are explicitly assigned.

Memory and Resource Allocation
Variables don't occupy hardware resources like flip-flops or latches. They are just temporary storage locations used during process execution. Signals, especially those representing registers or memory elements, do occupy hardware resources. The synthesis tool maps signals to actual hardware components based on their usage in the design.

Simulation Behavior
In behavioral simulation, variables show their advantage in internal calculations with immediate updates, while signals model hardware connections with delayed updates. In post-synthesis and post-implementation timing simulations, signals experience delays due to gate and routing delays, reflecting real hardware behavior. Variables remain unaffected by these delays within processes.

Application Scenarios
Variables are ideal for complex calculations within a process where intermediate results need to be stored and reused quickly. They provide efficient temporary storage without the overhead of hardware resources. Signals are essential for inter-component communication and maintaining state information in hardware descriptions. They are used to connect different parts of a design and model the actual hardware behavior with appropriate delays.

5 Simulation Result

In behavior simulation, the unknown signal is of the length of basic period, but in post-synthesis timing simulation and post-implementation timing simulation, the unknown time is according to the gate delay. So all of the signal is delayed by the gate delay in post-synthesis simulation and post-implementation timing simulation. And the delay in post-implementation timing simulation is longer than that in post-synthesis simulation.

Behavioral Simulation

- Objective: Performed early in the design phase to check for syntax errors and verify that the code implements the intended functionality.
- Simulation Process: Based solely on the RTL code without considering timing information, physical implementation details, or layout.
- Advantages and Disadvantages: Fast simulation speed allows for quick iteration and validation of design ideas, but it cannot detect issues caused by timing, layout, or implementation details.

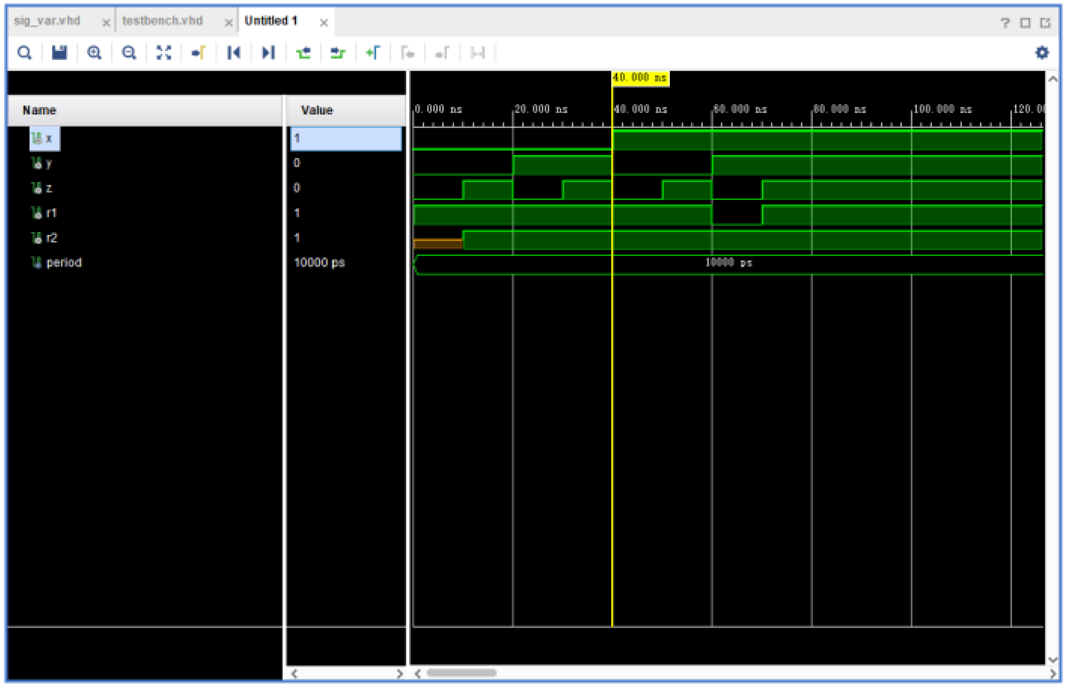


Figure 2: Behavior Simulation Result

The behavior simulation is the same as the theoretical analysis.

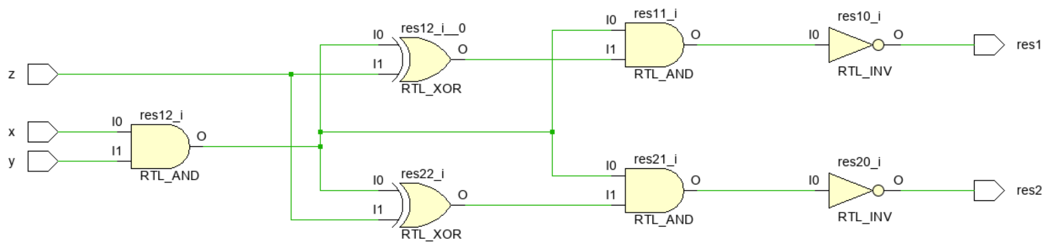


Figure 3: Behavior Schematic Simulation Result

Post-Synthesis Timing Simulation

- Objective: Ensure the design meets timing requirements by incorporating delay information from the synthesis process.
- Simulation Process: Utilizes an SDF file generated by the synthesis tool that contains delay information of the synthesized design elements to simulate timing behavior.
- Advantages and Disadvantages: Can identify potential issues caused by timing constraints, but the simulation speed is slower due to the need to model more timing details.

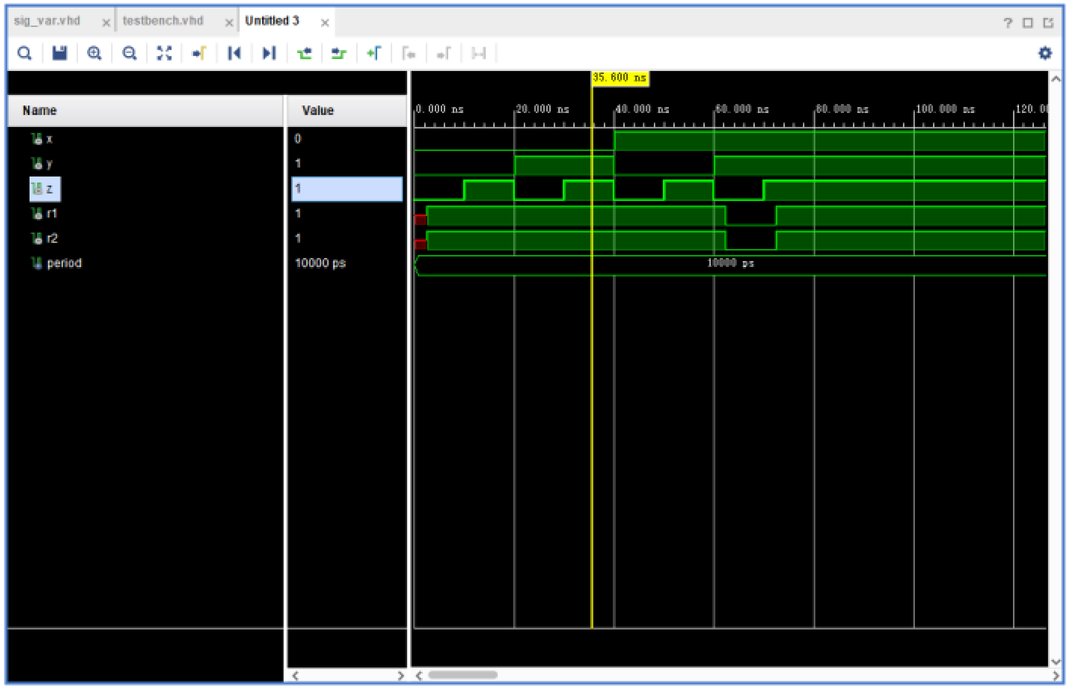


Figure 4: Post Synthesis Timing Simulation Result

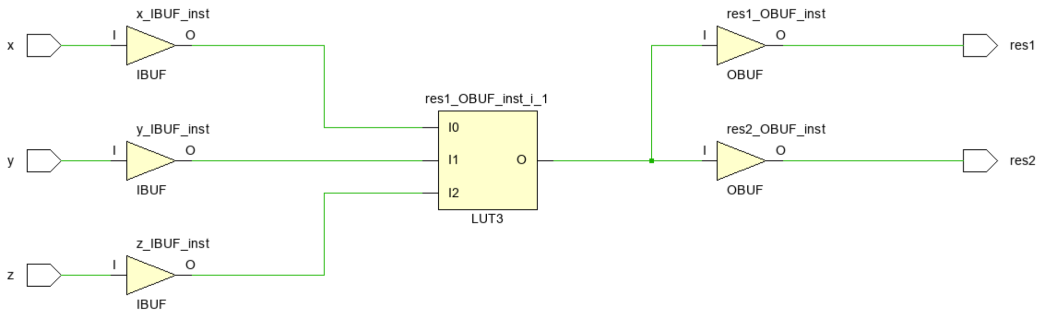


Figure 5: Synthesis Schematic Simulation Result

Post-Implementation Timing Simulation

- Objective: Ensure the design meets timing requirements after place and route by incorporating actual hardware delay factors.
- Simulation Process: Considers delays such as gate delay and routing delay to verify timing constraints.
- Advantages and Disadvantages: The simulation result is the closest to the actual hardware operation and can effectively identify timing-related issues, but it has the slowest simulation speed.

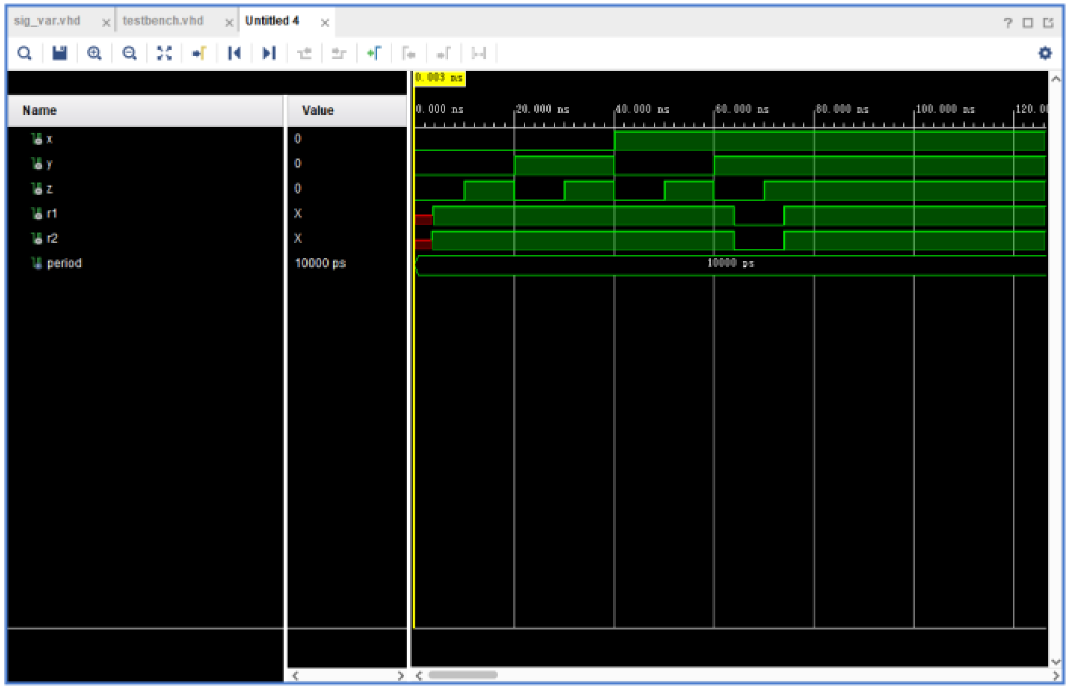


Figure 6: Post Implementation Timing Simulation Result

6 Conclusion

This experiment successfully demonstrated the differences between signal and variable types in VHDL through the design and simulation of a digital circuit. The VHDL code was correctly implemented, and the testbench effectively verified its functionality. The theoretical analysis aligned with the simulation results, confirming the proper operation of the circuit. Through this lab, the importance of understanding the distinct behaviors of signals and variables in digital system design was highlighted. Signals are suitable for inter-component communication and maintaining state information, while variables are ideal for internal calculations within processes.