Digital System Design Lab Report, Lab3, Homework1

Author: Yiwen Ying
Number: 12210159

Abstract

This paper documents the design, simulation and analysis of the full adder in detail. As a basic building block in digital logic circuits, the correct design of a full adder has a significant impact on the entire digital system. The design is functionally verified by coding the implementation of the full adder using VHDL language and writing the corresponding testbed code. Subsequently, by analyzing the results of different simulation stages (behavioral level simulation, post-synthesis timing simulation, and post-implementation timing simulation), the discrepancies between the simulation results and the theoretical expectations and their causes are explored, which further deepen the understanding of the working principle of the full adder and the design flow of digital circuits.

Contents

1 Introduction

Full adder is a basic combinational logic circuit used to realize binary number addition operation in digital system, which is widely used in arithmetic operation, logic processing and cryptography and other fields.

The purpose of this experiment is to design the full adder circuit through VHDL language, to master its basic principles and implementation methods.
Meanwhile, the design is functionally verified by writing testbed code to ensure the correctness of the circuit.
In addition, by analyzing the simulation results at different stages, we can understand the behavioral characteristics of digital circuits at different abstraction levels, and gain a deeper understanding of the effects of timing delays, signal contention, and other practical hardware factors on the performance of the circuits, so as to lay a foundation for the subsequent design of complex digital systems.

2 VHDL Code

The VHDL code is shown below. It shows the principle of full adder. It's written according to the circuit.

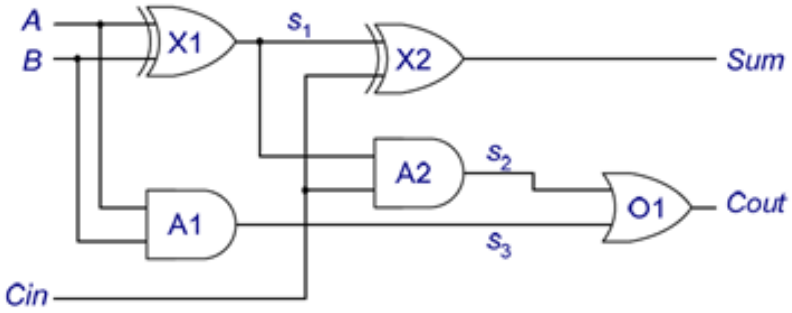

Figure 1: Full Adder Circuit

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

-- entity declaration
entity full_adder is
  Port ( A : in STD_LOGIC;
       B : in STD_LOGIC;
       Cin : in STD_LOGIC;
       Sum : out STD_LOGIC;
       Cout : out STD_LOGIC);
end full_adder;

-- architecture declaration
architecture Behavioral of full_adder is

-- signal and constant declaration
signal s1, s2, s3 : std_logic;
constant gate_delay : time := 10 ns;

-- logic function
begin
```

```
  s1 <= (A xor B) after gate_delay;
  s2 <= (Cin and s1) after gate_delay;
  s3 <= (A and B) after gate_delay;
  sum <= (s1 xor Cin) after gate_delay;
  cout <= (s2 or s3) after gate_delay;
end Behavioral;
```

3 Testbench Code

The testbench code is shown below. It shows the test case of full adder. It can test the correctness of full adder.

```
-- 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 full_adder is
  Port ( A : in STD_LOGIC;
       B : in STD_LOGIC;
       Cin : in STD_LOGIC;
       Sum : out STD_LOGIC;
       Cout : out STD_LOGIC);
end component;

-- signal and constant declaration
signal a, b, ci, s, co : std_logic;
constant period : time := 10ns;

-- logic function
begin
  -- instantiation
  uut:  full_adder port
    map( A => a,
         B => b,
         Cin => ci,
         Sum => s,
         Cout => co);

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

```
  ci <= '0' after period * 0, '1' after period * 1, '0' after period * 2, '0' after
      period * 3, '0' after period * 4, '1' after period * 5, '0' after period * 6,
      '0' after period * 7;

end tb;
```

4 Simulation Result

The simulation results are shown in the figures below.

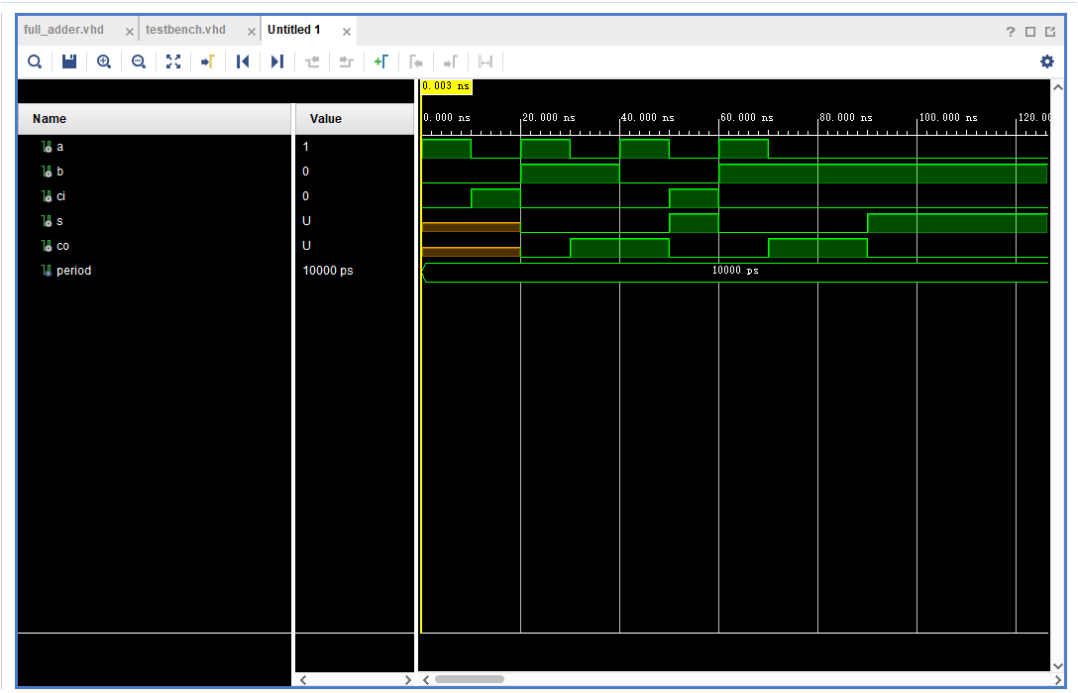

Figure 2: Behavior Simulation Result

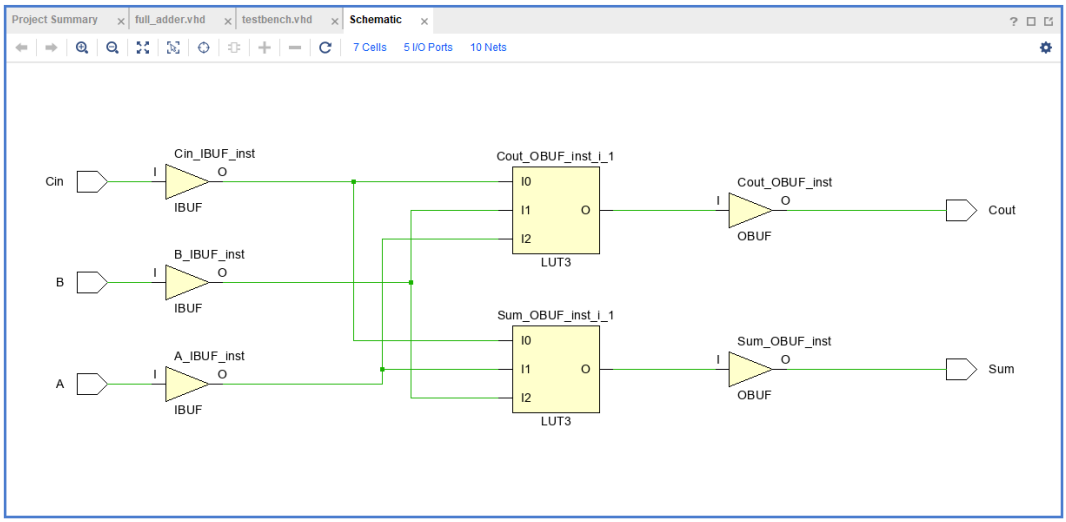

Figure 3: Schematic Simulation Result

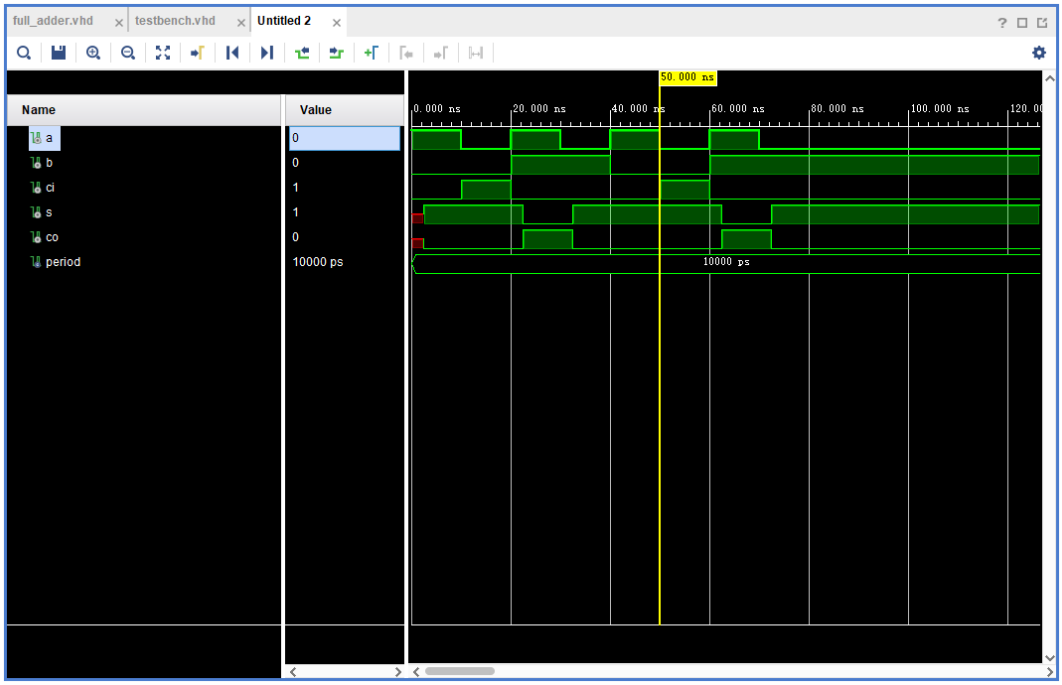

Figure 4: Post Synthesis Timing Simulation Result

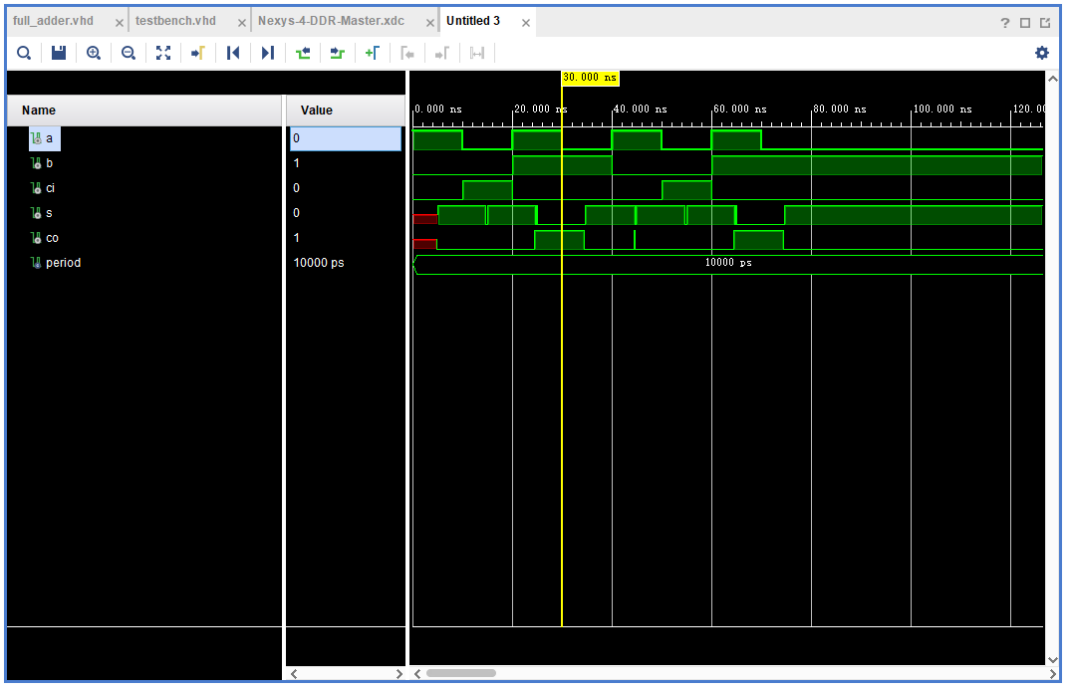

Figure 5: Post Implementation Timing Simulation Result

5 Analysis

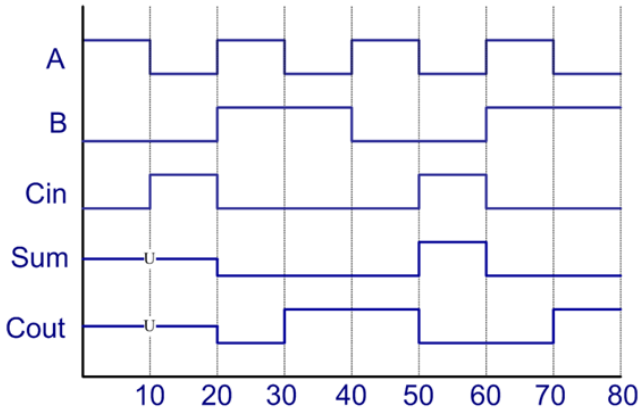

Figure 6: Theoretical Waveform

1. Are the simulation results consistent with those in the theoretical diagrams?

- The results of the behavioral level simulation are basically consistent with the theoretical expectations, and the changes of the input signals a, b, and ci as well as the trends of the output signals s and co match the theoretical waveforms.
- However, in the post-synthesis timing simulation and the post-implementation timing simulation, there are different waveform variations from the behavioral-level simulation, especially some delays and burrs on the output signals s and co, which indicate that there are differences in the results at different simulation stages.

2. Are the results of additive computations consistent with common perceptions?

- In behavioral-level simulation, when the combination of input signals a, b, and ci conforms to the logic of addition, the output s and co results are consistent with expectations. For example, when a=0, b=0, and ci=1, s=1 and co=0, which is consistent with the logic of a half adder.
- However, in the post-implementation timing simulation, due to timing delays and other factors, there may be transient erroneous results, but from the final steady state, the result of the addition calculation is still consistent with the cognition.

3. If there are inconsistencies, what are the reasons for the inconsistencies?

- The inconsistency is mainly due to the different levels of abstraction of the circuit model in different simulation stages. Behavioral level simulation only focuses on the functional implementation without considering the timing delays and physical characteristics of the actual hardware; whereas post-synthesis timing simulation and post-implementation timing simulation add the constraints of the actual hardware, such as gate delays, wiring delays, etc., which can lead to delays and burrs in the signal transmission and thus make the waveforms different from the behavioral level simulation.

4. What is the difference between the 3 types of simulation results?

- Behavioral-level simulation: immediate response to signal changes, clean waveforms, no delays or burrs, and a direct reflection of the functional logic of the design. The unknown state lasts longer because behavioral-level simulation gives less consideration to the timing and delay details of the circuit, and the simulation tool usually defaults to the initial or undefined state of the signal.
- Post-synthesis timing simulation: some delays in signal changes due to gate-level delays introduced by the synthesis tool begin to appear, and a brief undefined state may appear on the waveform. The duration of the unknown state is shorter than in behavioral-level simulation because the post-synthesis simulation adds more constraints from the hardware implementation.
- Post-implementation timing simulation: delays and burrs are more pronounced because the signal variations are more in line with the operation of the real hardware due to the inclusion of a more accurate timing model in the implementation stage, including actual hardware factors such as cabling delays. The duration of the unknown state may be comparable to the post-synthesis simulation, but the exact performance may vary depending on the differences in the actual hardware layout and wiring.

5. Why is there difference?

- Different simulation phases focus on different design details.
- Behavioral simulation focuses on functional correctness, with a more idealized treatment of timing and delay; post-synthesis timing simulation begins to consider the timing constraints of the hardware implementation, and introduces practical factors such as gate delays; and post-implementation timing simulation is an accurate timing verification after the actual hardware layout is routed, with more realistic delays and hardware characteristics.
- In addition, the longer duration of the unknown state in the behavioral level simulation is due to the more simplified way of predicting and handling the circuit behavior at this stage, whereas the post-synthesis and post-implementation simulations handle the unknown state in more detail according to the constraints of the actual hardware, resulting in the possible shorter duration of the unknown state.

6. Do you see some spikes in the results?

- Yes, some spikes (brief high or low level burrs) can be seen in the waveforms of the post-implementation timing simulation, especially on the output signals s and co.

7. What causes these spikes?

- The spikes are caused by competing adventures or different timing delays in the circuit. In digital circuits, when multiple input signals change at the same time but arrive at a gate via different paths, the different path delays may result in transient error states, i.e., spikes, at the output of the gate.
- In addition, synthesis and realization tools may introduce additional logic gates or adjust the circuit structure when dealing with circuit logic, which may also trigger spikes.

6 Conclusion

This experiment successfully realized the design and simulation verification of the full adder. The logic function of the full adder is clearly expressed through the writing of VHDL code, while the test platform is built to comprehensively cover various input combination situations to ensure the functional correctness of the design.

In the behavioral-level simulation phase, the results are highly consistent with theoretical expectations, verifying the correctness of the design in ideal situations.

However, in the synthesis post-timing simulation and realization post-timing simulation, the simulation waveforms show different variations from the behavioral-level simulation, such as signal delays and burr phenomena, due to the introduction of constraints such as timing delays and wiring delays of the actual hardware.

These differences profoundly reflect the different focus on the abstraction degree of the circuit model in different simulation stages, with the behavioral-level simulation focusing on the functional implementation, while the post-synthesis and post-implementation simulations are closer to the actual hardware operation.

The analysis of these phenomena not only deepens the understanding of the working principle of the full adder, but also provides a more comprehensive understanding of the various aspects of the digital circuit design flow. In practical applications, it is necessary to fully consider the impact of hardware characteristics on circuit performance, and reasonably optimize the design to meet specific timing and functional requirements.