Digital System Design Lab Report, Lab3

Author: Ying Yiwen
Number: 12210159

Contents

1 Assignment1

1.1 Introduction

Full adder is a combinational circuit that performs addition of two binary numbers. It is a basic building block of digital logic circuits. It is used in many applications, such as arithmetic, logic, and cryptography.
In this experiment, we will design a full adder using VHDL. At the same time, we write testbench to verify the correctness of the design. Then we can use simulation to know the working principle of full adder.

1.2 VHDL Code

```
-- 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;
```

1.3 Testbench 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 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;
```

1.4 Simulation Result

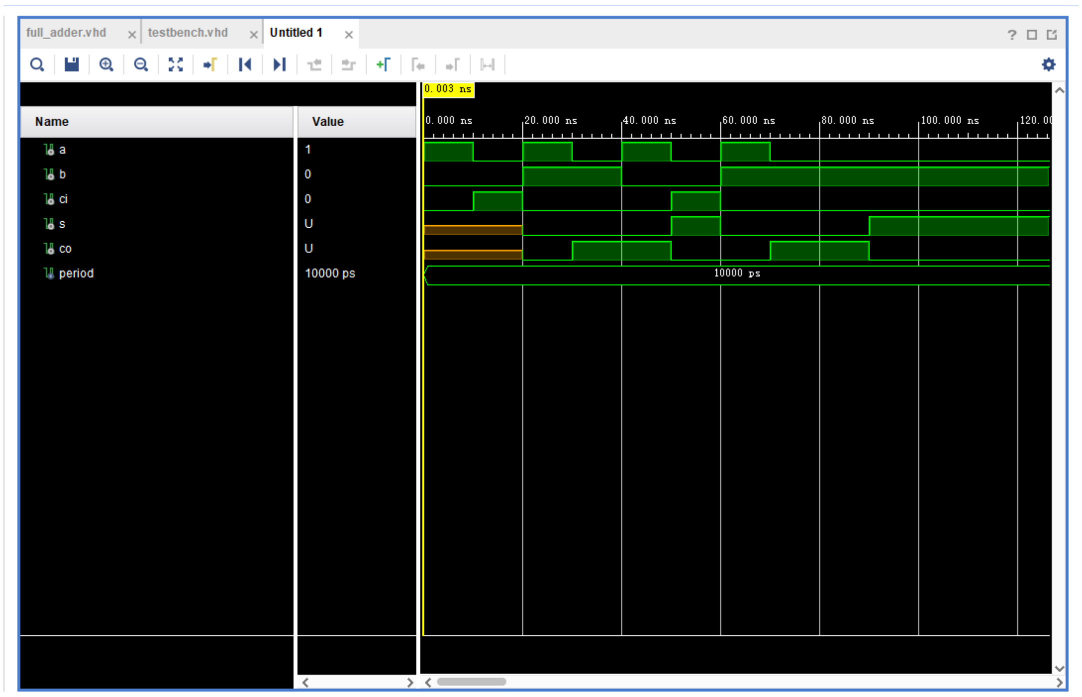


Figure 1: Behavior Simulation Result

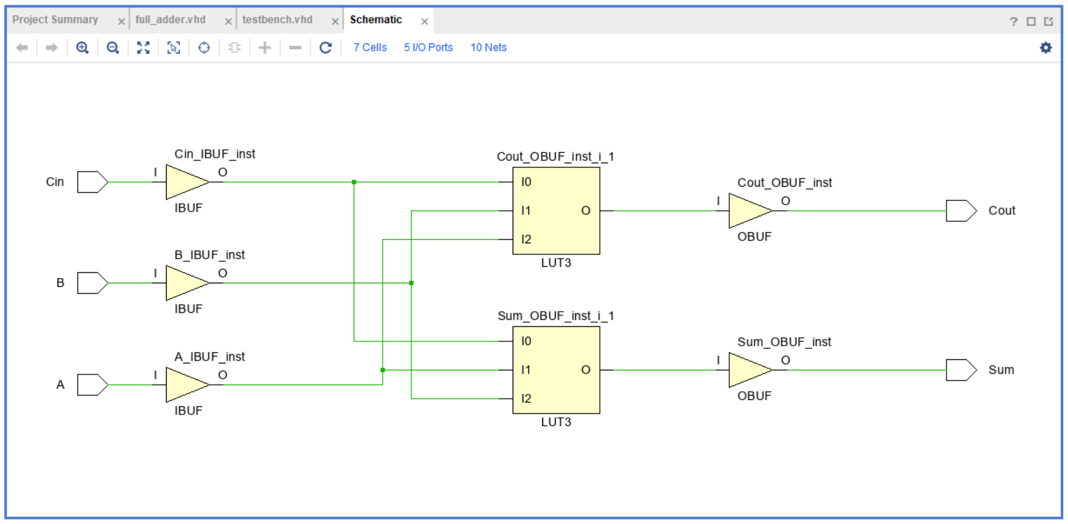
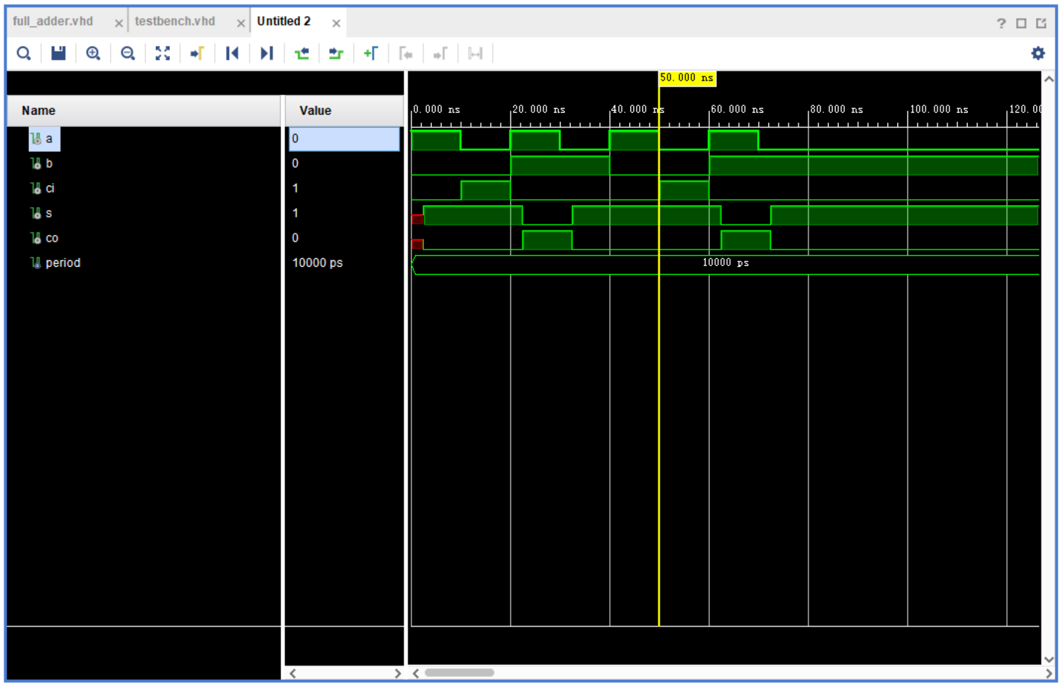
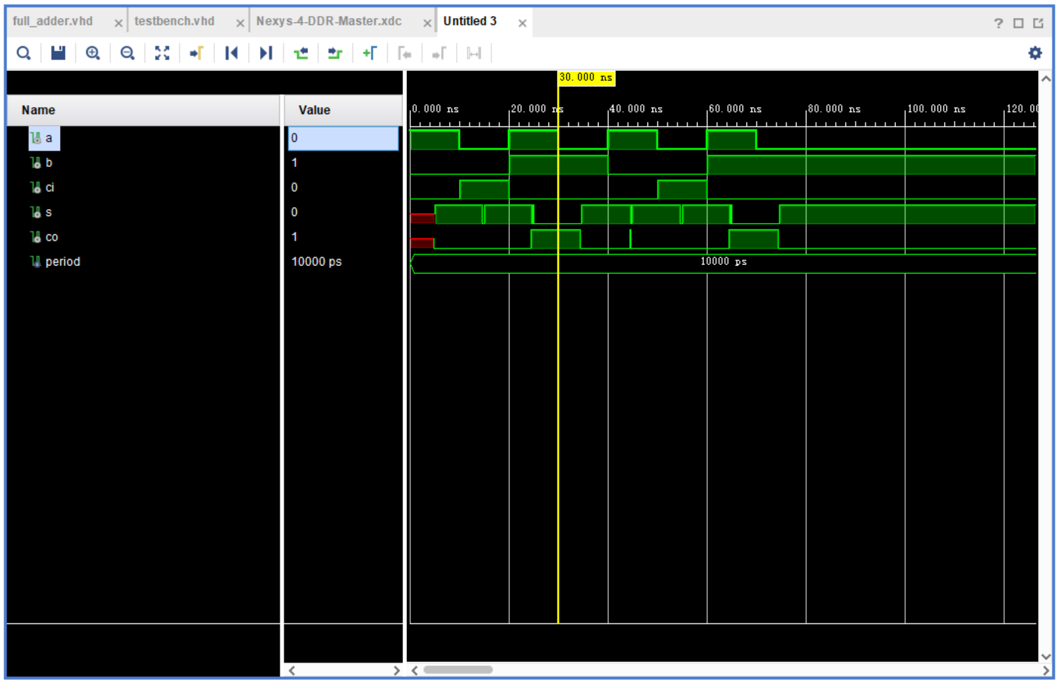


Figure 2: Schematic Simulation Result



Name	Value
a	0
b	0
ci	1
s	1
co	0
period	10000 ps

Figure 3: Post Synthesis Timing Simulation Result



Name	Value
a	0
b	1
ci	0
s	0
co	1
period	10000 ps

Figure 4: Post Implementation Timing Simulation Result

1.5 Analysis

1.6 Conclusion

2 Assignment2

2.1 Introduction

2.2 VHDL Code

```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;
```

2.3 Testbench Code

```vhdl
-- 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;
```

2.4 Simulation Result

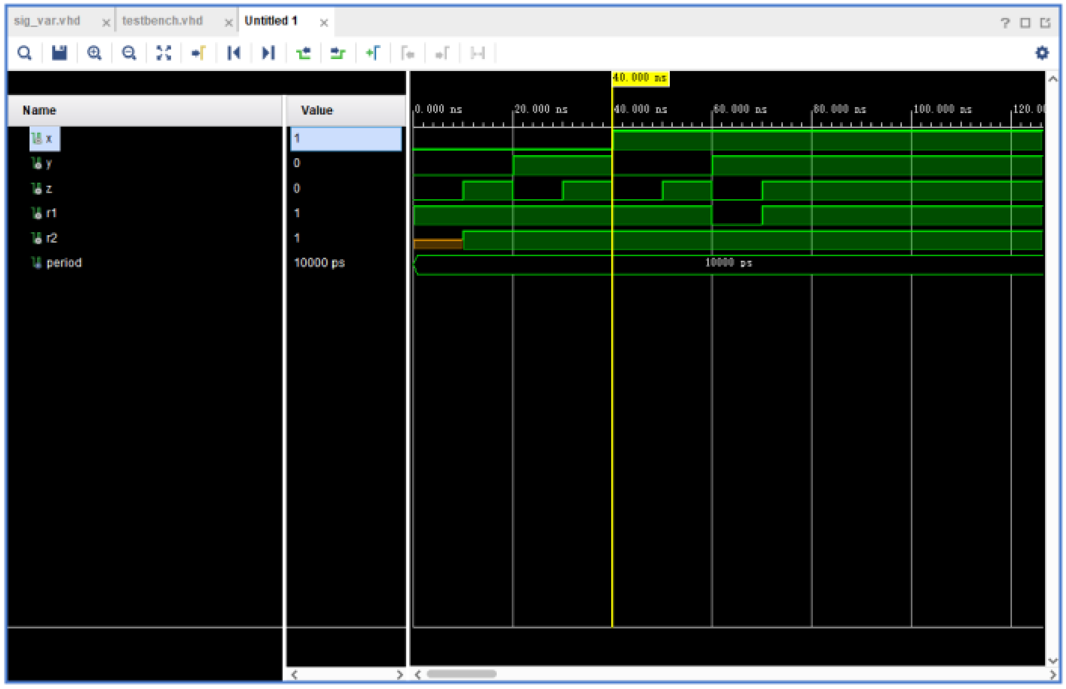


Figure 5: Behavior Simulation Result

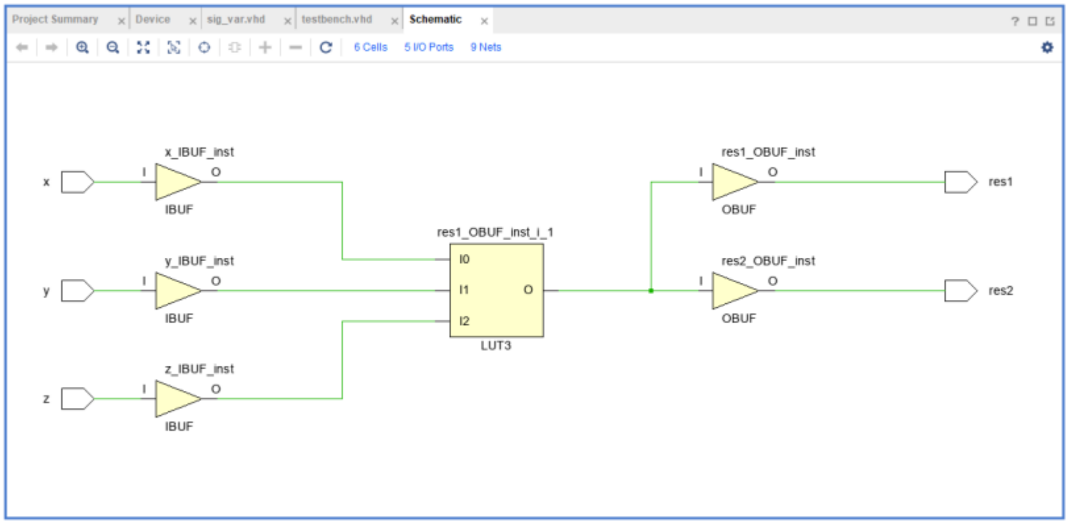
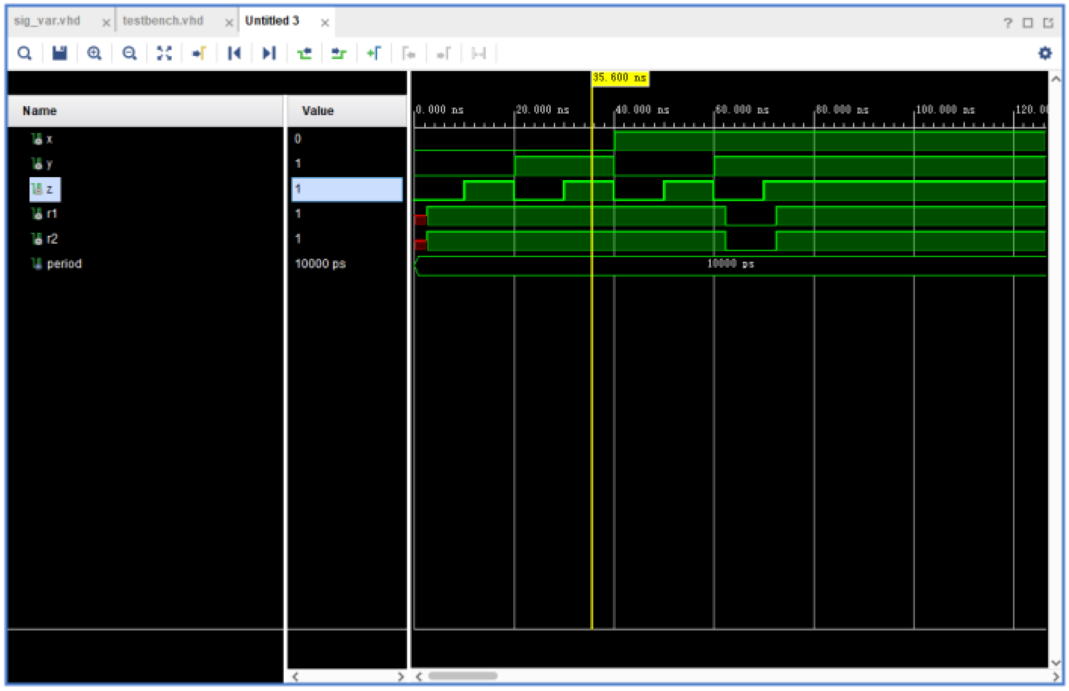
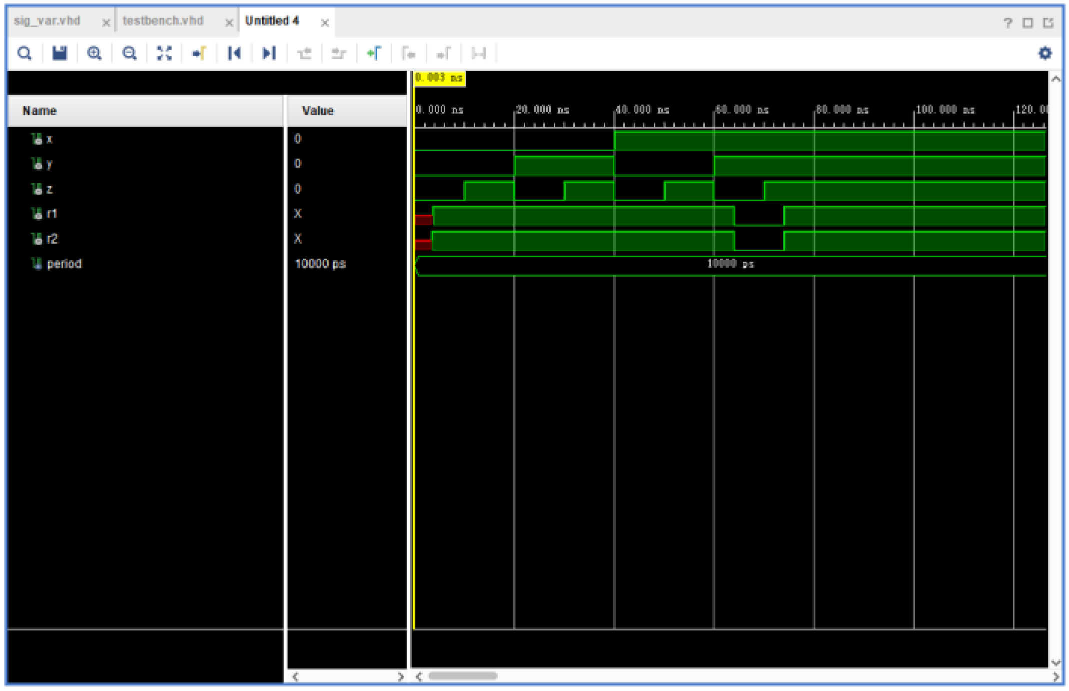


Figure 6: Schematic Simulation Result



Name	Value
x	0
y	1
z	1
r1	1
r2	1
period	10000 ps

Figure 7: Post Synthesis Timing Simulation Result



Name	Value
x	0
y	0
z	0
r1	X
r2	X
period	10000 ps

Figure 8: Post Implementation Timing Simulation Result