Lab 2 Exercise 3: Design a tree 16-to-4 priority encoder

姓名：王宇　　学号：12112725

Prelab:

根据要求，画出的概念草图如下，根据输入的优先级，直接产生四位的输出和一位活动标志。

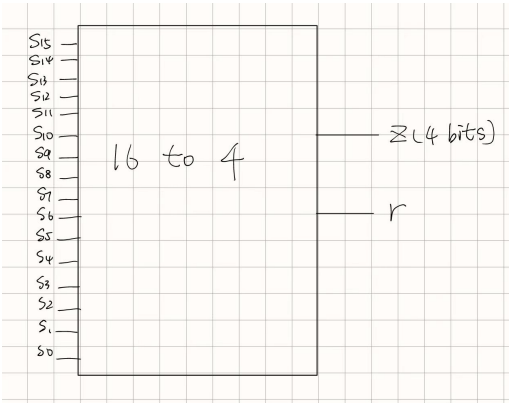

图 1 Prelab 概念图

Testbench 中，共测试了六种情况，分别是：全为 0，仅 S0 为 1，仅 S15 为 1，仅 S11 为 1，多位为 1，全为 1。共跑了五种仿真结果，如下图所示：

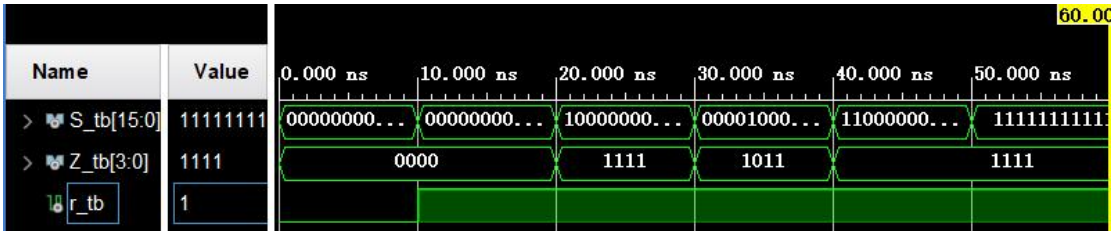

图 2 Prelab Behavioral Simulation 结果

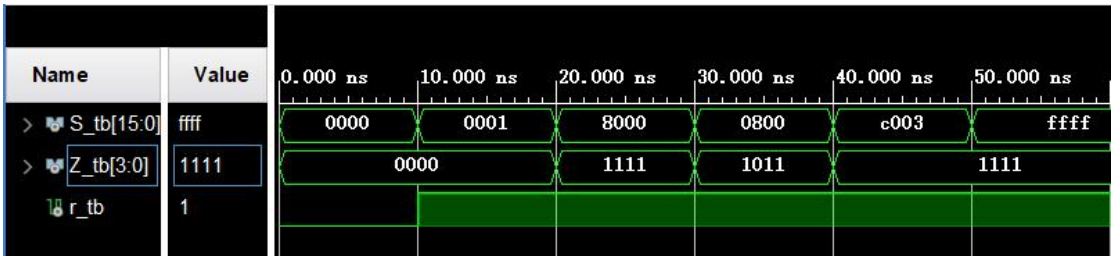

图 3 Prelab Post-Synthesis Functional Simulation 结果

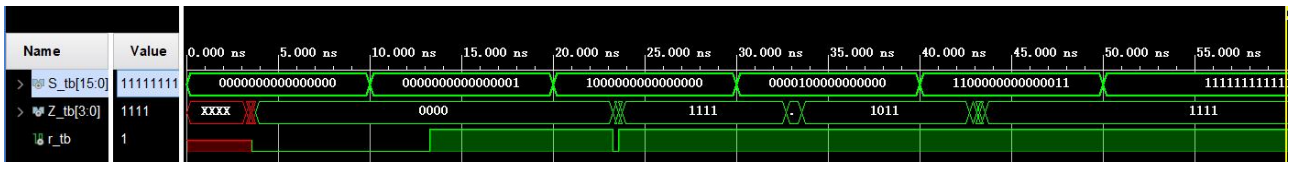

图 4 Prelab Post-Synthesis Timing Simulation 结果

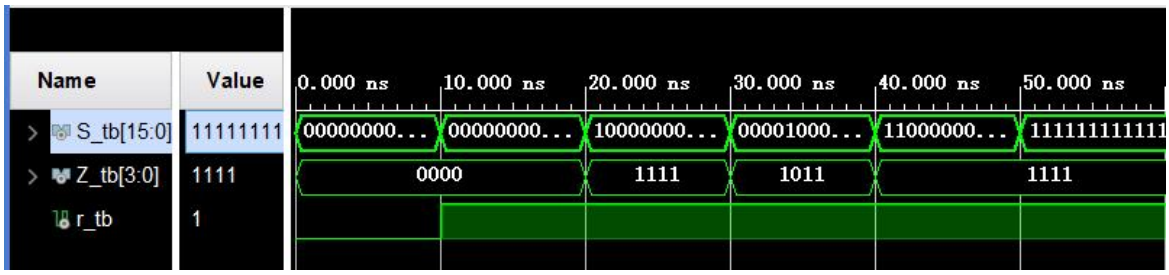

图 5 Prelab Post-Implementation Functional Simulation 结果

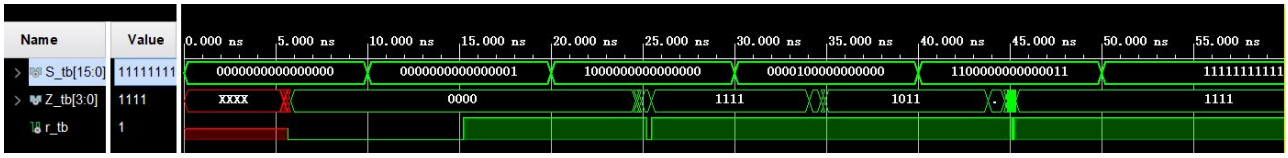

图 6 Prelab Post-Implementation Timing Simulation 结果

从图中可以看出，代码的功能仿真上没有问题，结果与预期一致。而在时序仿真中，在开始时存在明显的时延，并且每次信号变化时也会产生一定的时延。根据 Schematic 产生的电路图来看，最大延时路径应该是 IBUF 到 OBUF 之间的路径，在 LUT 中实现逻辑功能时需要花费较多时间。

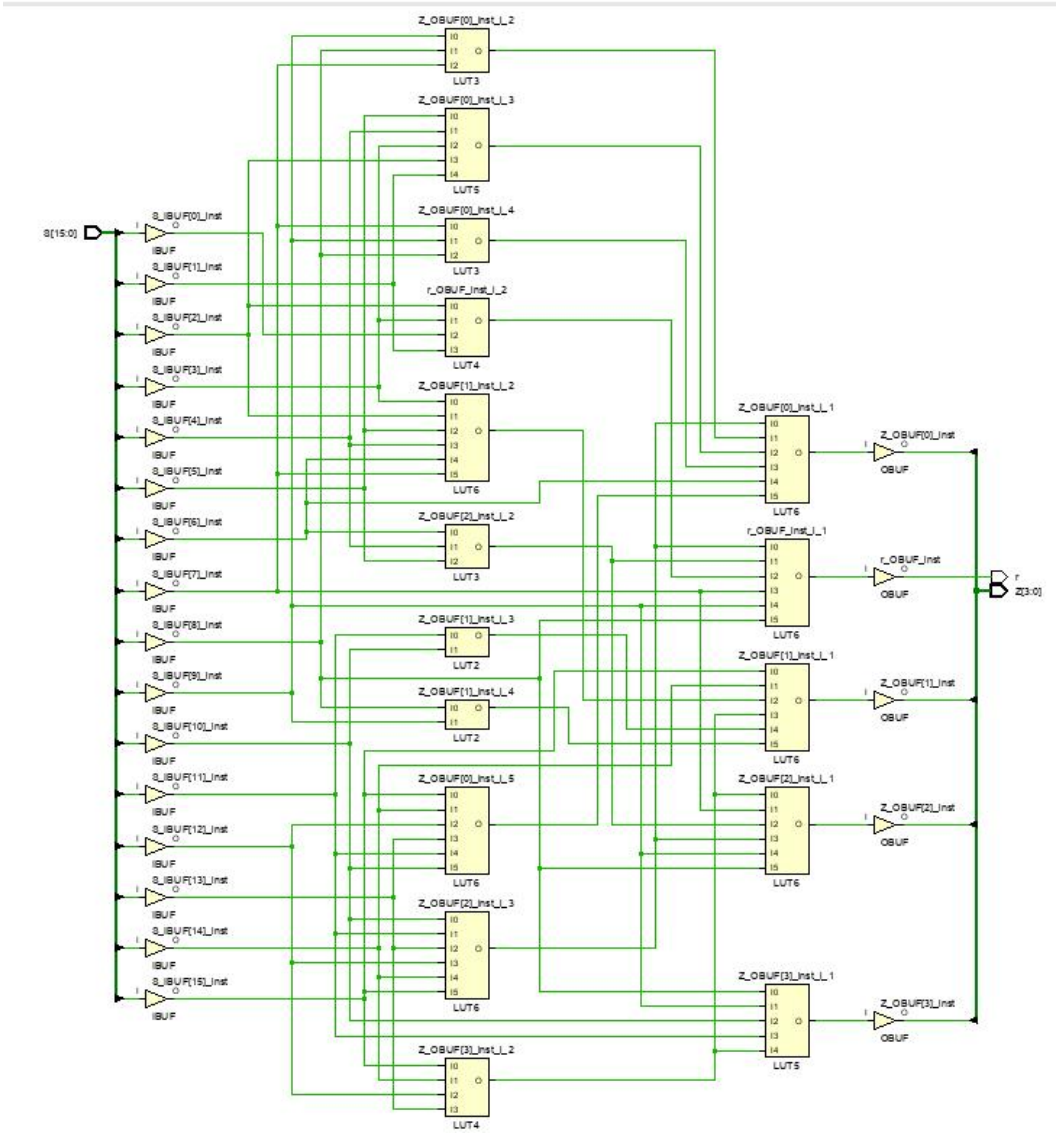

图 7 prelab 逻辑电路图

根据时序仿真图来看，开始时的时延大概是在 5ns 左右。

During the lab:

根据要求，电路应为树状结构，我采用了四个 4-to-2 优先编码器获取输入，然后根据他们的活动位，确定优先的输入在四个编码器的哪一个，以确定最终输出的前两位，然后再获取对应 4-to-2 编码器的输出，以确定最终输出的后两位。

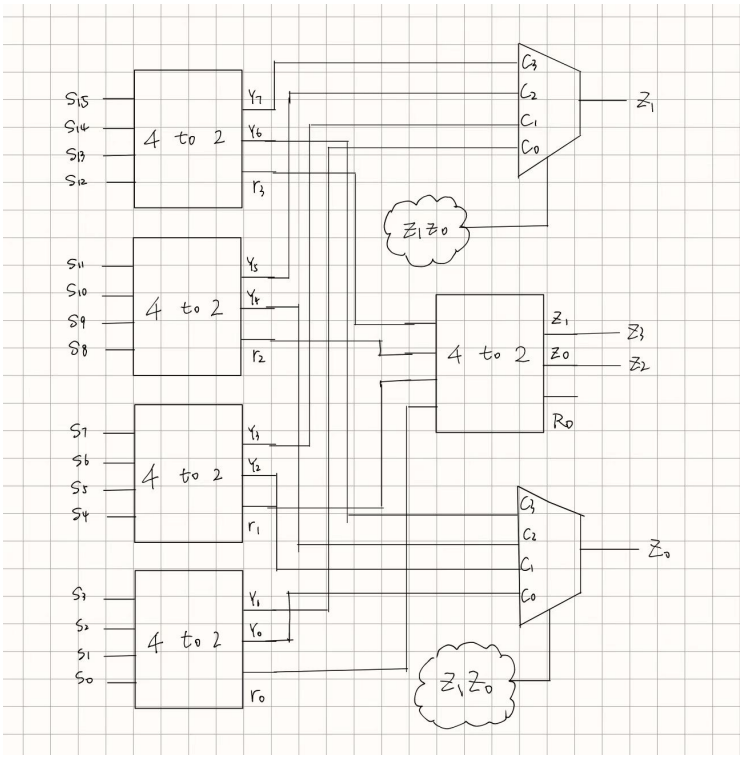

图 8 树状结构概念图

然后和 prelab 相同，用 testbench 测试六种情况，仿真结果如下：

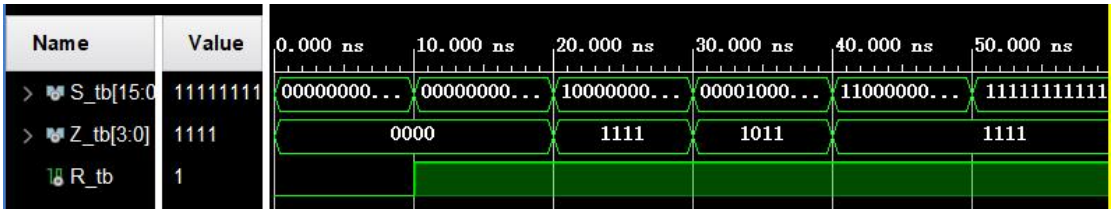

图 9 树状结构 Behavioral Simulation

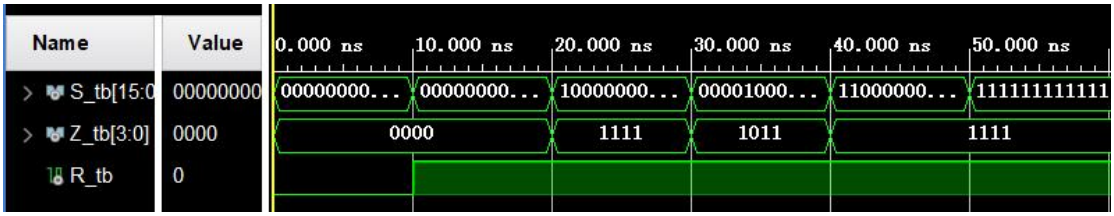

图 10 树状结构 Post-Synthesis Functional Simulation

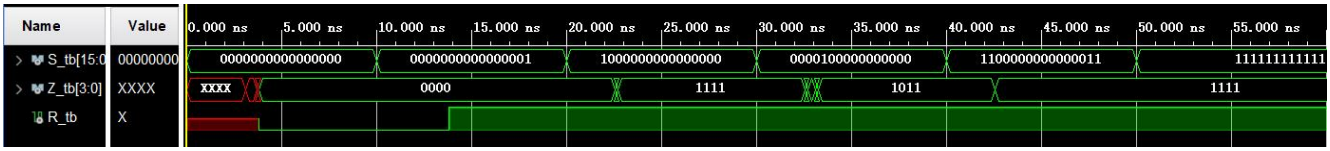

图 11 树状结构 Post-Synthesis Timing Simulation

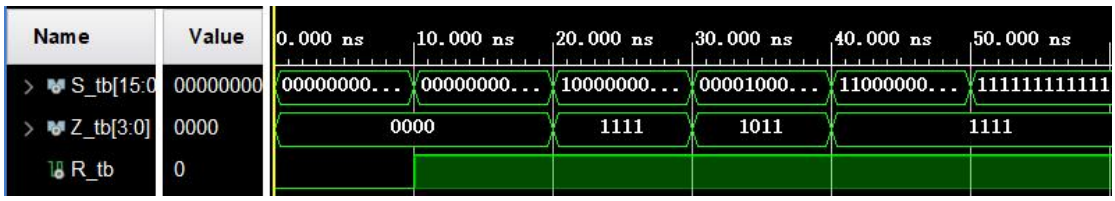

图 12 树状结构 Post-Implementation Functional Simulation

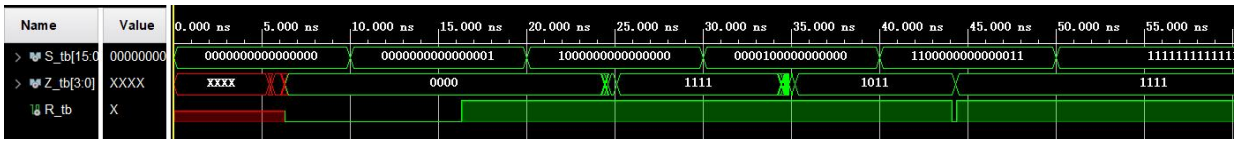

图 13 树状结构 Post-Implementation Timing Simulation

从图中可以看出，代码的功能仿真与 prelab 结果一致，说明代码逻辑上不存在问题。从时序仿真上来看，开始时的时延与 prelab 近似，甚至略大于先前产生的时延，但对于信号变换产生的时延似乎有一定的改善，但也并没有明显差别。根据电路图我们可以发现，先前的电路最多经过两个 LUT，但在我设计的树状结构中，最多会经过 3 个 LUT，因此时延比起之前会有一定的增加。

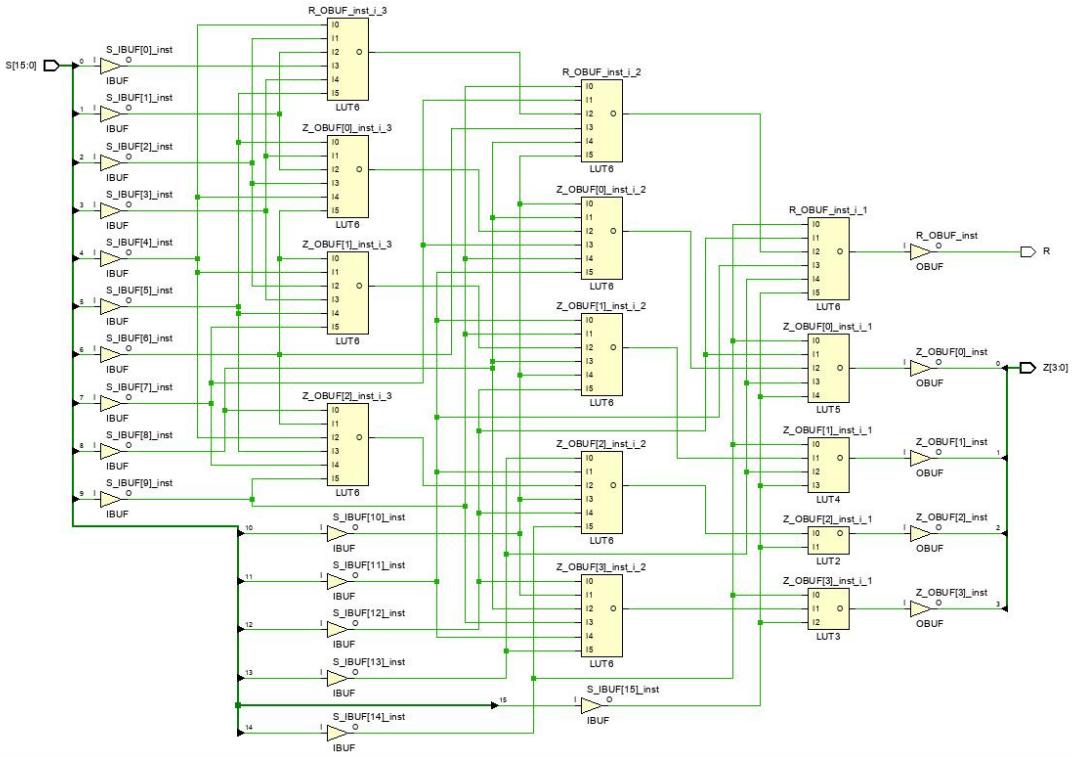

图 14 树状结构逻辑电路图

Conclusion：

原先我认为树状结构的时延会优于原结构的时延，但是事实上我设计的树状结构并没有起到优化效果。也许是因为我设计的结构并没有达到简化电路的效果，对于较少的位数来说，我设计的树状结构需要经过更多的逻辑判断，反而不如直接进行筛选来的更快。

Coding：

Prelab 部分：

VHDL：

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

entity pr_encoder is
port (S: in std_logic_vector(15 downto 0);
        Z : out std_logic_vector (3 downto 0);
        r : out std_logic);
end entity pr_encoder;

architecture Behavioral of pr_encoder is
begin
      Z <= "1111" when S(15) = '1' else
              "1110" when S(14) = '1' else
              "1101" when S(13) = '1' else
              "1100" when S(12) = '1' else
              "1011" when S(11) = '1' else
              "1010" when S(10) = '1' else
              "1001" when S(9) = '1' else
              "1000" when S(8) = '1' else
              "0111" when S(7) = '1' else
              "0110" when S(6) = '1' else
              "0101" when S(5) = '1' else
              "0100" when S(4) = '1' else
              "0011" when S(3) = '1' else
              "0010" when S(2) = '1' else
              "0001" when S(1) = '1' else
              "0000";
     r <= S(15) or S(14) or S(13) or S(12) or S(11) or S(10) or S(9) or S(8)
            or S(7) or S(6) or S(5) or S(4) or S(3) or S(2) or S(1) or S(0);
end Behavioral;
```

Testbench：

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

entity pr_encoder_tb is
end pr_encoder_tb;

architecture behavior of pr_encoder_tb is

      signal s_tb : std_logic_vector(15 downto 0);
```

```
    signal z_tb : std_logic_vector(3 downto 0);
    signal r_tb : std_logic;

    component pr_encoder
    port(
         s : in    std_logic_vector(15 downto 0);
         z : out std_logic_vector(3 downto 0);
         r : out std_logic
        );
    end component;

begin
    uut: pr_encoder port map (
          s => s_tb,
          z => z_tb,
          r => r_tb
        );
    process
    begin
        s_tb <= (others => '0');
        wait for 10 ns;

        s_tb <= "0000000000000001";
        wait for 10 ns;

        s_tb <= "1000000000000000";
        wait for 10 ns;

        s_tb <= "0000100000000000";
        wait for 10 ns;

        s_tb <= "1100000000000011";
        wait for 10 ns;

        s_tb <= (others => '1');
        wait for 10 ns;

        wait;
    end process;

end behavior;
```

树状结构部分：

VHDL：

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

entity pr_encoder is
port (S: in std_logic_vector(15 downto 0);
        Z : out std_logic_vector (3 downto 0);
        R : out std_logic
        );
end entity pr_encoder;

architecture Behavioral of pr_encoder is
    signal group_priority: std_logic_vector(7 downto 0);
    signal valid_bits: std_logic_vector(3 downto 0);
    signal selected_group: std_logic_vector(1 downto 0);

begin

process(S)
begin
    for i in 0 to 3 loop
        if S(i*4+3 downto i*4) /= "0000" then
            valid_bits(i) <= '1';

            if S(i*4+3) = '1' then
                group_priority(i*2+1 downto i*2) <= "11";
            elsif S(i*4+2) = '1' then
                group_priority(i*2+1 downto i*2) <= "10";
            elsif S(i*4+1) = '1' then
                group_priority(i*2+1 downto i*2) <= "01";
            else
                group_priority(i*2+1 downto i*2) <= "00";
            end if;
        else
            valid_bits(i) <= '0';
            group_priority(i*2+1 downto i*2) <= "00";
        end if;
    end loop;
end process;

process(valid_bits)
begin
    selected_group <= "00";
    R <= '0';
```

```
        if valid_bits(0) = '1' then
            selected_group <= "00";
            R <= '1';
        end if;
        if valid_bits(1) = '1' then
            selected_group <= "01";
            R <= '1';
        end if;
        if valid_bits(2) = '1' then
            selected_group <= "10";
            R <= '1';
        end if;
        if valid_bits(3) = '1' then
            selected_group <= "11";
            R <= '1';
        end if;
end process;

process(selected_group, group_priority)
begin
        case selected_group is
            when "00" =>
                Z <= selected_group & group_priority(1 downto 0);
            when "01" =>
                Z <= selected_group & group_priority(3 downto 2);
            when "10" =>
                Z <= selected_group & group_priority(5 downto 4);
            when "11" =>
                Z <= selected_group & group_priority(7 downto 6);
            when others =>
                Z <= "0000";

        end case;
end process;
end Behavioral;
```

Testbench：

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

entity pr_encoder_tb is
end pr_encoder_tb;

architecture behavior of pr_encoder_tb is
```

```
    signal s_tb : std_logic_vector(15 downto 0);
    signal z_tb : std_logic_vector(3 downto 0);
    signal r_tb : std_logic;

    component pr_encoder
    port(
         s : in    std_logic_vector(15 downto 0);
         z : out std_logic_vector(3 downto 0);
         r : out std_logic
        );
    end component;

begin

    uut: pr_encoder port map (
          s => s_tb,
          z => z_tb,
          r => r_tb
        );

    process
    begin
        s_tb <= (others => '0');
        wait for 10 ns;

        s_tb <= "0000000000000001";
        wait for 10 ns;

        s_tb <= "1000000000000000";
        wait for 10 ns;

        s_tb <= "0000100000000000";
        wait for 10 ns;

        s_tb <= "1100000000000011";
        wait for 10 ns;

        s_tb <= (others => '1');
        wait for 10 ns;

        wait;
    end process;
```

end behavior;