**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。

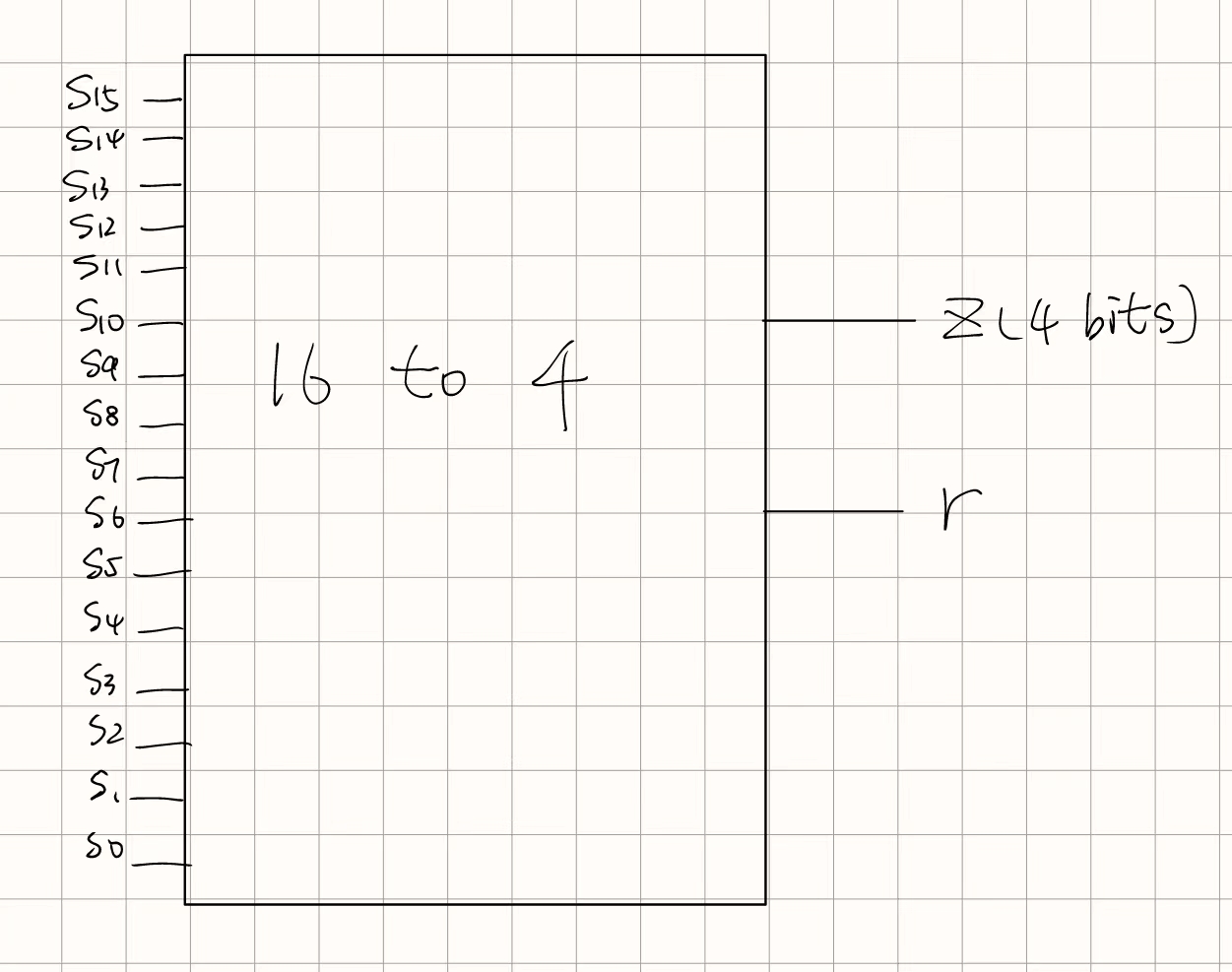
****

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：

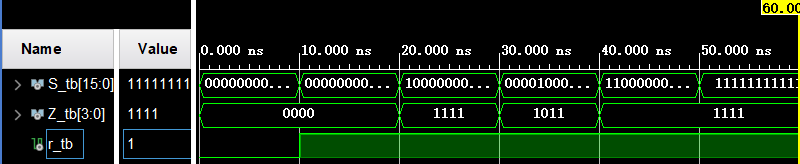


图 2 Prelab Behavioral Simulation结果

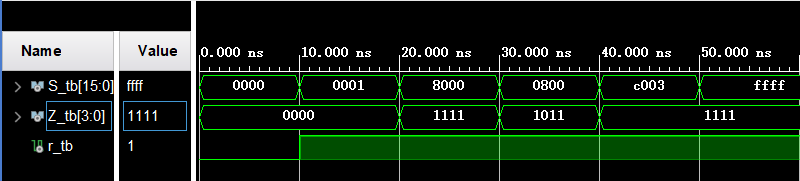


图 3 Prelab Post-Synthesis Functional Simulation结果

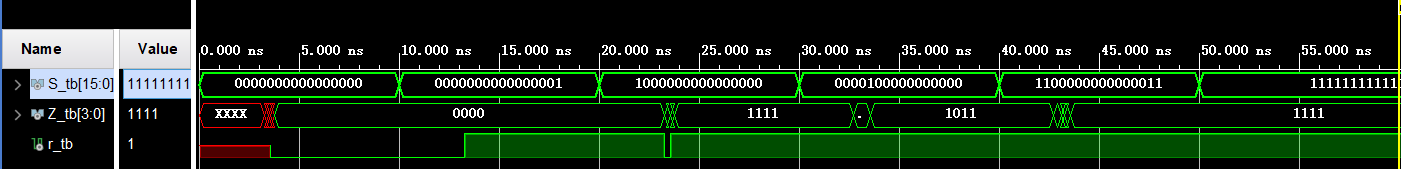


图 4 Prelab Post-Synthesis Timing Simulation结果

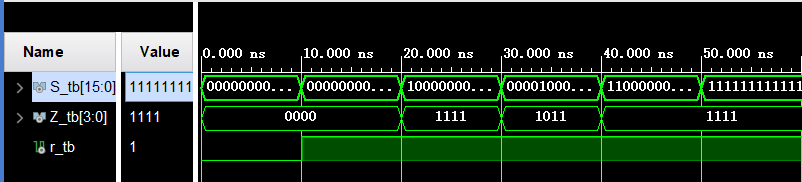


图 5 Prelab Post-Implementation Functional Simulation结果

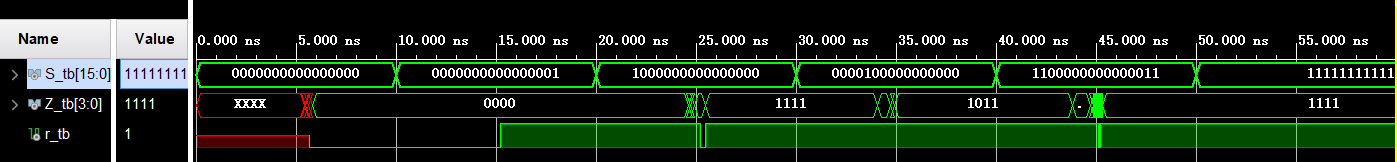


图 6 Prelab Post-Implementation Timing Simulation结果

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。

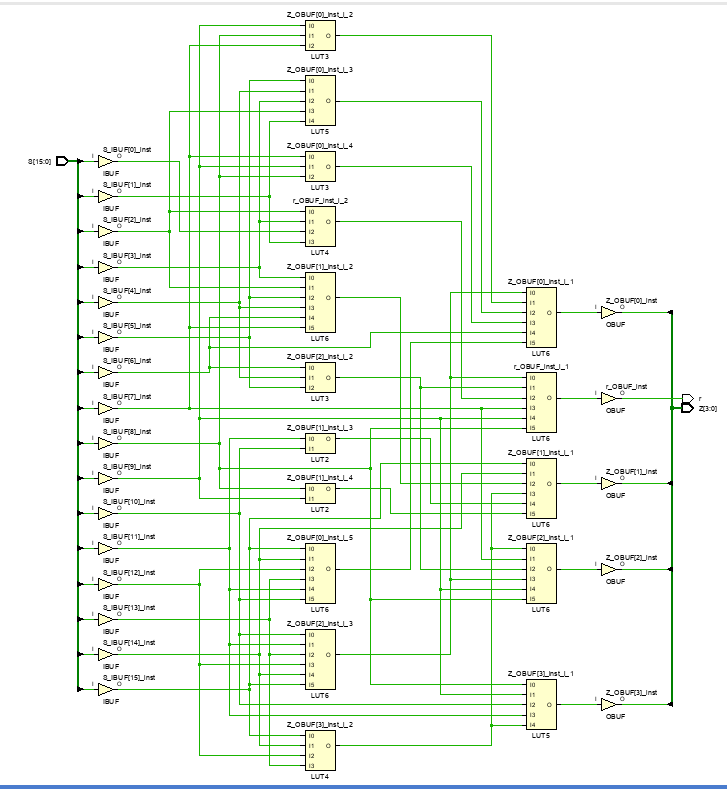


图 7 prelab逻辑电路图

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。

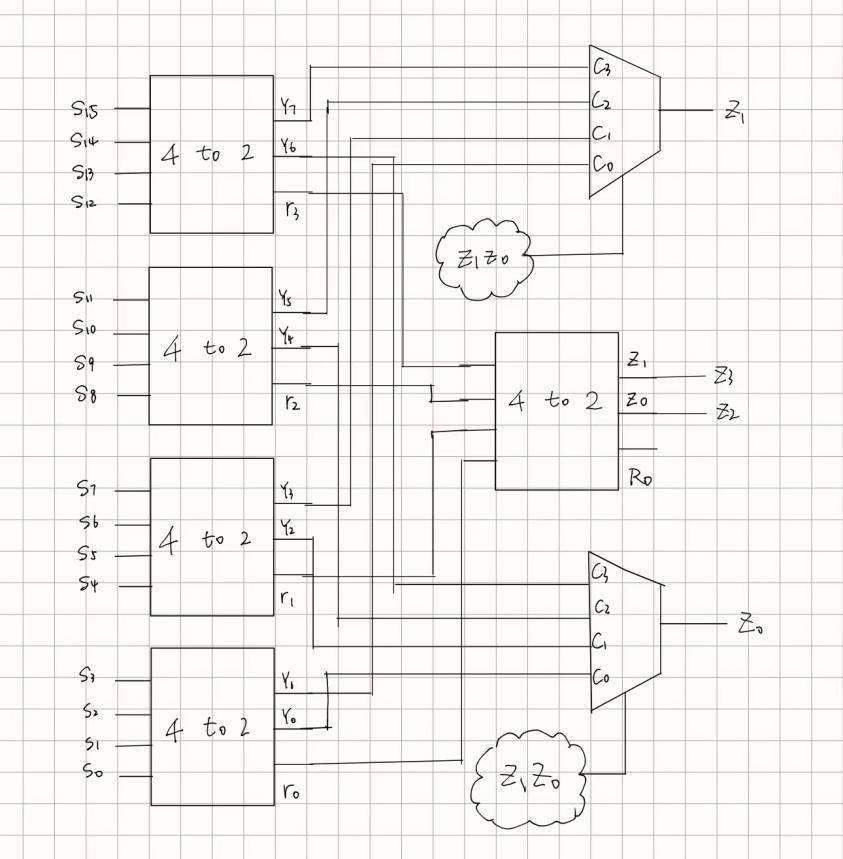


图 8 树状结构概念图

然后和prelab相同，用testbench测试六种情况，仿真结果如下：

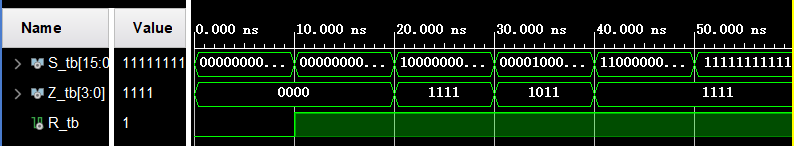


图 9 树状结构Behavioral Simulation

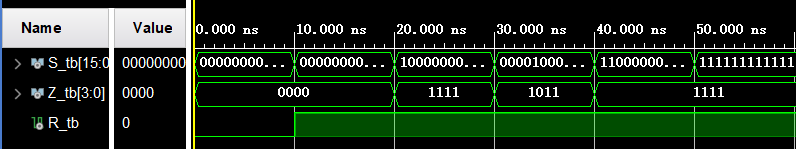


图 10 树状结构Post-Synthesis Functional Simulation

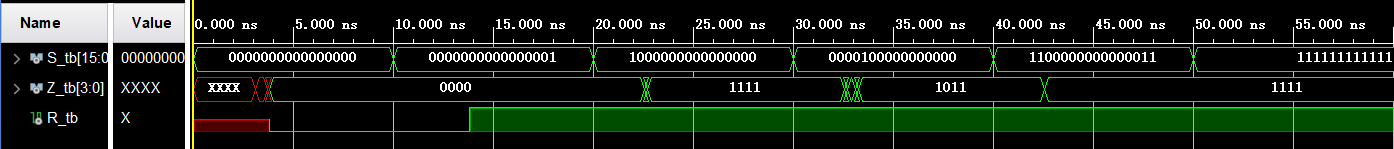


图 11 树状结构Post-Synthesis Timing Simulation

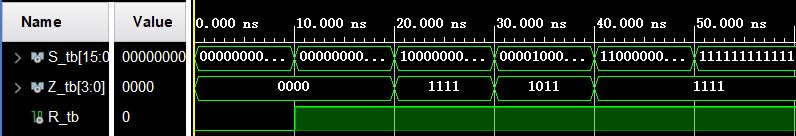


图 12 树状结构Post-Implementation Functional Simulation

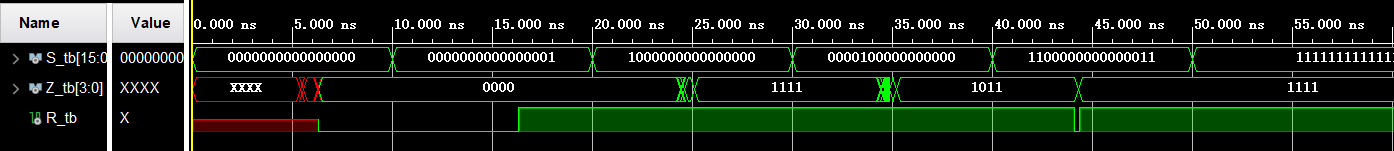


图 13 树状结构Post-Implementation Timing Simulation

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。

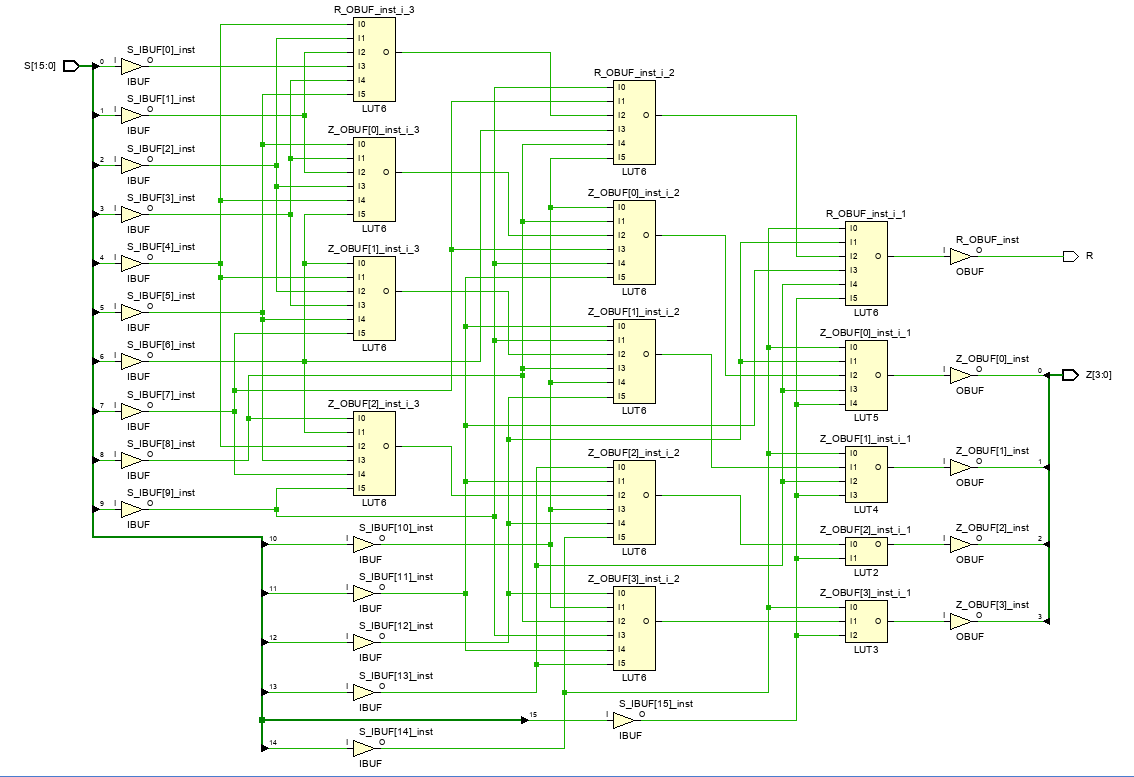


图 14 树状结构逻辑电路图

**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;