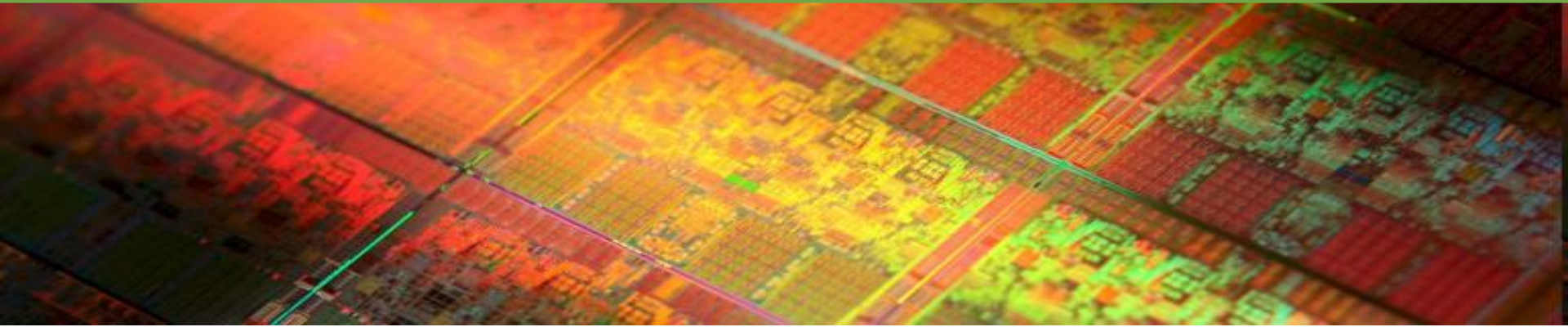


EE 2000

Logic Circuit Design (Fall 2018)



Tutorial 4

8 to 3 Priority Encoder

Instructor: Dr. Ray C.C. Cheung

Objectives

- Learn how to initialize a 8 to 3 encoder.
- Learn how to change it into decoder and extend it with more bits.
- Using Vivado simulator to observe how the encoder works

Priority Encoder

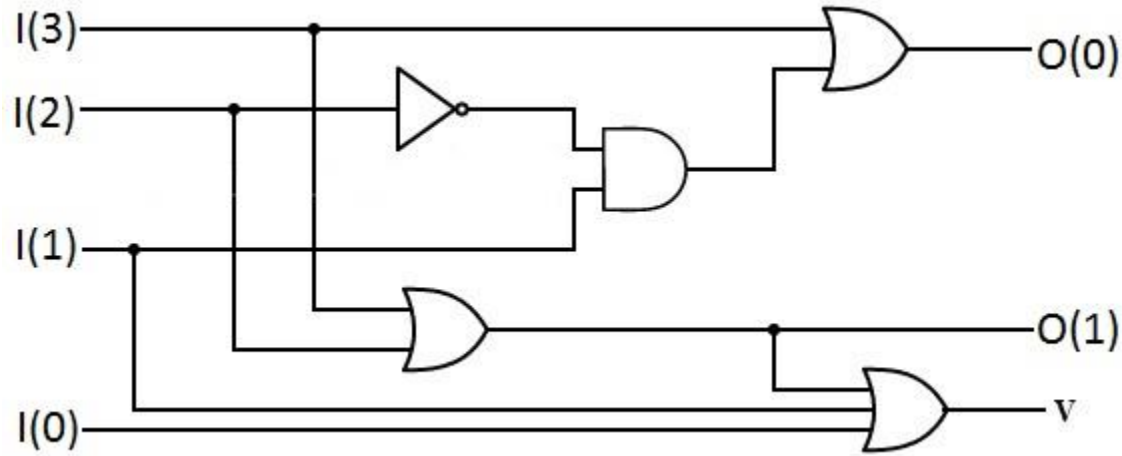


Figure 1. Schematic of 4 to 2 priority encoder

- A **priority encoder** is a circuit or algorithm that compresses multiple binary inputs into a smaller number of outputs.
- It is a typical example of combination logic circuits.
- The output of a priority encoder is the binary representation of the original number starting from zero of the most significant input bit.
- They are often used to control interrupt requests by acting on the highest priority encoder.

Truth Table of Priority Encoder

S(3)	S(2)	S(1)	S(0)	en	O(1)	O(0)	v
-	-	-	-	0	Z	Z	0
0	0	0	1	1	0	0	1
0	0	1	-	1	0	1	1
0	1	-	-	1	1	0	1
1	-	-	-	1	1	1	1
0	0	0	0	1	Z	Z	0

- S: the input data of the encoder
- O : the output data of the encoder
- en: enable port of this encoder
- v : to show that if the encoder has a real output.

State of port in VHDL

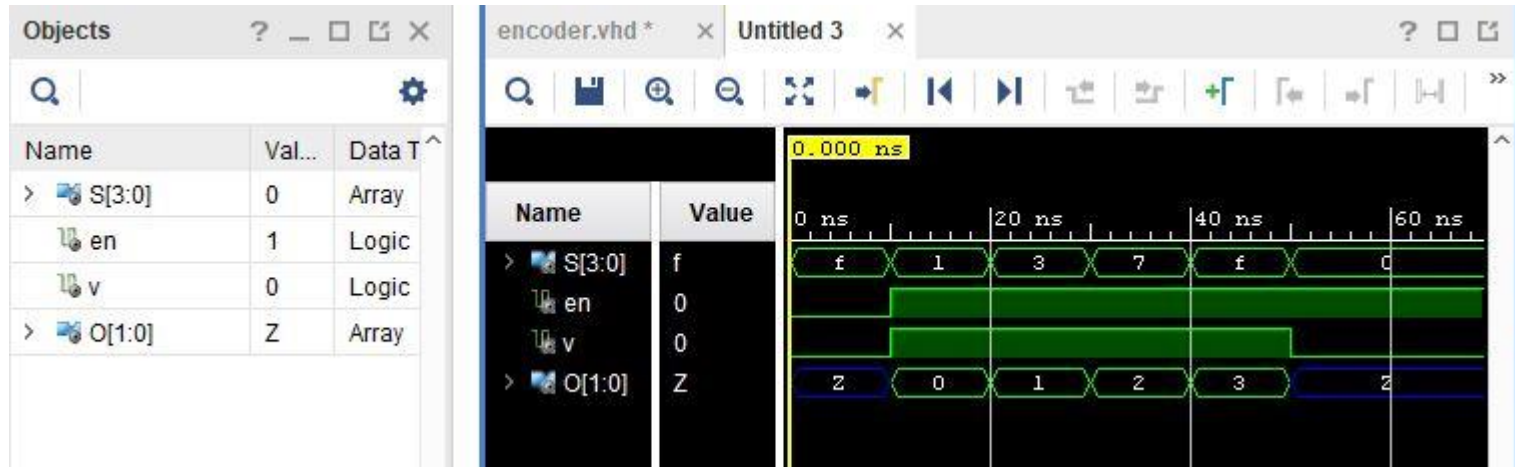
```
type std_ulogic is (  
'U', -- Uninitialized  
'X', -- Forcing Unknown  
'0', -- Forcing 0  
'1', -- Forcing 1  
'Z', -- High Impedance  
'W', -- Weak Unknown  
'L', -- Weak 0  
'H', -- Weak 1  
'-' -- Don't Care );
```

Reference:

https://en.wikipedia.org/wiki/IEEE_1164

<https://standards.ieee.org/standard/1164-1993.html> (for more details)

Simulation result of 4 to 2 Priority encoder



- The higher bit of the input has the priority.
- When en = 0, there is no output; to keep it safe in application, define the output in this occasion as Z.
- If en = 1 but no inputs, there is no output as Z with showing v = 0

How to design an 8 to 3?

S(7)	S(6)	S(5)	S(4)	S(3)	S(2)	S(1)	S(0)	O(2)	O(1)	O(0)	V
0	0	0	0	0	0	0	0	Z	Z	Z	0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	-	0	0	1	1
0	0	0	0	0	1	-	-	0	1	0	1
0	0	0	0	1	-	-	-	0	1	1	1
0	0	0	1	-	-	-	-	1	0	0	1
0	0	1	-	-	-	-	-	1	0	1	1
0	1	-	-	-	-	-	-	1	1	0	1
1	-	-	-	-	-	-	-	1	1	1	1

Step by Step

Create a new project

Revise the entity to achieve 8 to 3 according to truth table.

Save the results in registers and output the results (with pipelined structure in the next tutorial)

Simulate the encoder

Synthesis and Implementation

Create project and sources

- Open tuto4 template project on Canvas
- In the project manager panel, add a new source to create an 8 to 3 priority encoder based on given 4 to 2 priority encoder.
- Define each port of entity to combine the two 4 to 2 priority encoder into one 8 to 3 priority encoder.

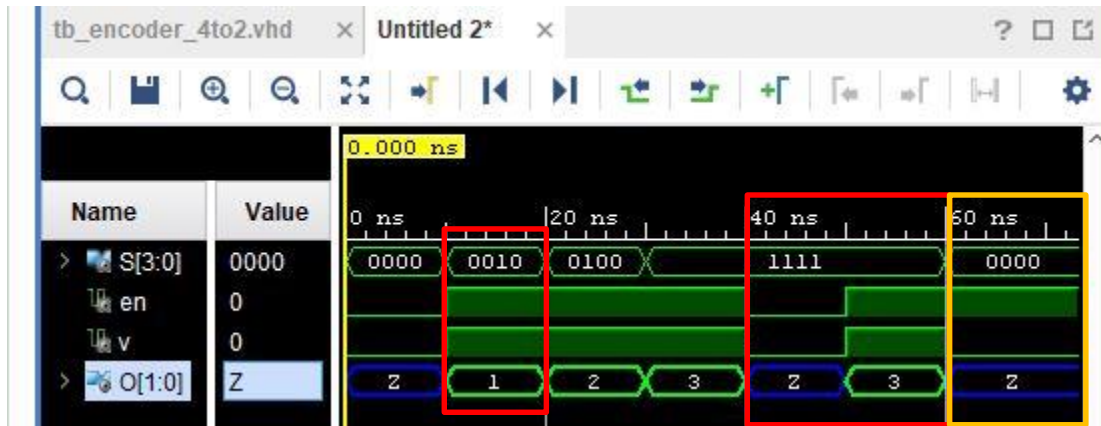
4 to 2 Priority Encoder

Notice the usage of If and elsif. Run behaviour simulation and check the result with truth table.

```
encoder_8to3.vhd x encoder_4to2.vhd * x tb_encoder_4to2.vhd *
[Icons]
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  entity encoder_4to2 is
4      Port (
5          S : in STD_LOGIC_VECTOR(3 DOWNTO 0);
6          en : in STD_LOGIC;
7          v : out STD_LOGIC;
8          O : out STD_LOGIC_VECTOR(1 DOWNTO 0)
9      );
10 end encoder_4to2;
11
12 architecture Behavioral of encoder_4to2 is
13 begin
14     process(en,S)
15     begin
16         if (en = '0') then
17             v <= '1';
18             if (S(3) = '1') then
19                 O <= "11";
```

```
18
19
20         elsif(S(2)='1') then
21             O <= "10";
22         elsif(S(1)='1') then
23             O <= "01";
24         elsif(S(0)='1') then
25             O <= "00";
26         else
27             v <= '0';
28             O <= "ZZ";
29         end if;
30     elsif (en = '1') then
31         v <= '0';
32         O <= "ZZ";
33     end if;
34 end process;
35 end Behavioral;
```

Simulation result of 4 to 2



When the sample is “0000” and enable is ‘1’, the output is high impedance. There is no value so v is 0

When the sample is “0010”, enable is 1, the output is 1 meaning the input is encoded into 1. At the same time, v is 1 means there is a valid result.

When the sample is “1111”, enable is 1, the output is 3 meaning the input is encoded into 3 because of the priority. At the same time, v is 1 means there is a valid result. When enable changes to 0, v is 0 means no results and the output changes into high impedance.

Instructions

1. Add a new source file for 8 to 3 priority encoder
2. Combine two given 4 to 2 encoders, set the ports properly.
3. Write testbench for testing.
4. Run behaviour simulation and check schematic.

Combine two encoders

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity encoder_8to3 is
5      Port (
6          S_1 : in STD_LOGIC_VECTOR(3 DOWNTO 0);
7          S_0 : in STD_LOGIC_VECTOR(3 DOWNTO 0);
8          en : in STD_LOGIC;
9          v : inout STD_LOGIC_VECTOR(1 DOWNTO 0);
10         O_1 : out STD_LOGIC_VECTOR(1 DOWNTO 0);
11         O_0 : out STD_LOGIC_VECTOR(1 DOWNTO 0)
12     );
13 end encoder_8to3;
14
15 architecture Behavioral of encoder_8to3 is
16     COMPONENT encoder_4to2 is
17         Port (
18             S : in STD_LOGIC_VECTOR(3 DOWNTO 0);
19             en : in STD_LOGIC;
20             v : out STD_LOGIC;
21             O : out STD_LOGIC_VECTOR(1 DOWNTO 0)
22         );
23     END COMPONENT;
24     begin
25         uut1 : encoder_4to2 PORT MAP(S_1, en, v(1), O_1 );
26         uut2 : encoder_4to2 PORT MAP(S_0, V(1), v(0), O_0 );
27     end Behavioral;
28
```

- S_1 and S_0 represents high 8 bits and low 8 bits of the samples.
- Respectively, O_1 and O_0 are the output of each 4 to 2 encoder.
- V(1) and V(0) shows the output is read from which encoder

Notice: there are 2 parts should be revised to combine these two encoders:

1. Change the input V into inout port.

Then we can realize when there are inputs from higher 4 bits, the encoder for lower 4 bits will be disabled automatically.

2. Change the enable of each 4 to 2 encoder from high level into low level to make it easier to revise.

Combine two encoders

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity encoder_8to3 is
5      Port (
6          S_1 : in STD_LOGIC_VECTOR(3 DOWNTO 0);
7          S_0 : in STD_LOGIC_VECTOR(3 DOWNTO 0);
8          en   : in STD_LOGIC;
9          v    : inout STD_LOGIC_VECTOR(1 DOWNTO 0);
10         O_1  : out STD_LOGIC_VECTOR(1 DOWNTO 0);
11         O_0  : out STD_LOGIC_VECTOR(1 DOWNTO 0);
12     );
13 end encoder_8to3;
14
15 architecture Behavioral of encoder_8to3 is
16     COMPONENT encoder_4to2 is
17         Port (
18             S : in STD_LOGIC_VECTOR(3 DOWNTO 0);
19             en : in STD_LOGIC;
20             v : out STD_LOGIC;
21             O : out STD_LOGIC_VECTOR(1 DOWNTO 0);
22         );
23     END COMPONENT;
24
25     begin
26         uut1 : encoder_4to2 PORT MAP(S_1, en, v(1), O_1 );
27         uut2 : encoder_4to2 PORT MAP(S_0, V(1), v(0), O_0 );
28     end Behavioral;
```

```
2  architecture Behavioral of encoder_4to2 is
3      begin
4          process(en,S)
5              begin
6                  if (en = '0') then
7                      v <= '1';
8                      if (S(3) = '1') then
9                          O <= "11";
10                     elsif(S(2)='1') then
11                         O <= "10";
12                     elsif(S(1)='1') then
13                         O <= "01";
14                     elsif(S(0)='1') then
15                         O <= "00";
16                     else
17                         v <= '0';
18                         O <= "ZZ";
19                     end if;
20                 elsif (en = '1') then
21                     v <= '0';
22                     O <= "ZZ";
23                 end if;
24             end process;
25         end Behavioral;
```

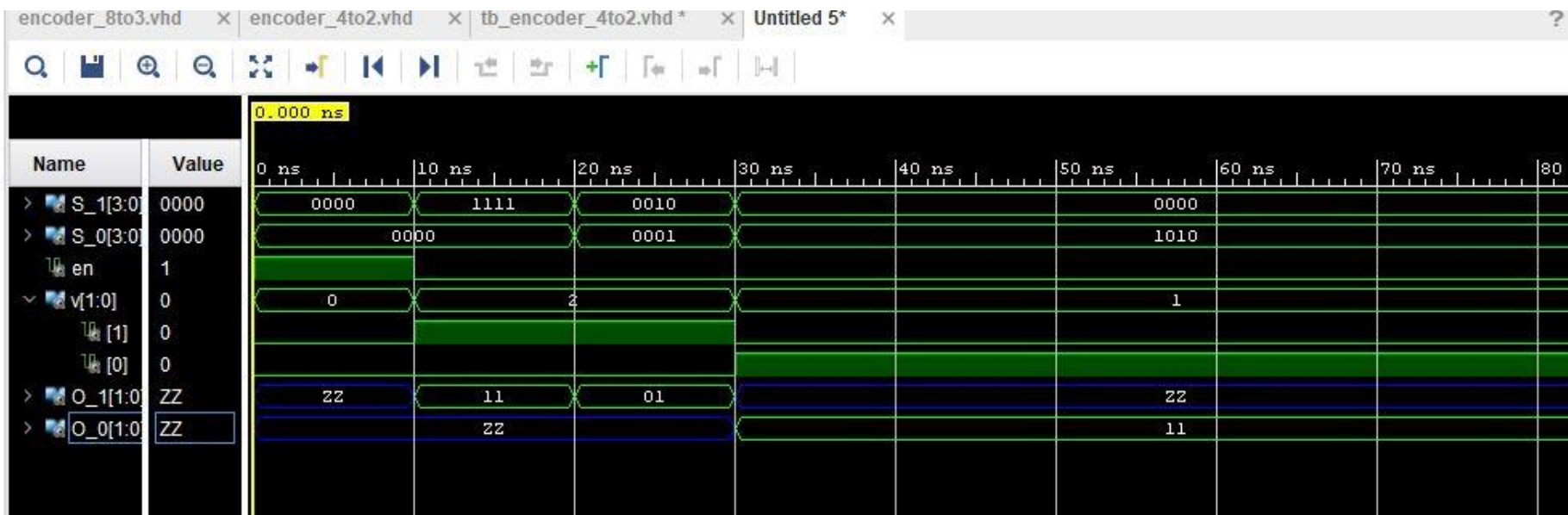
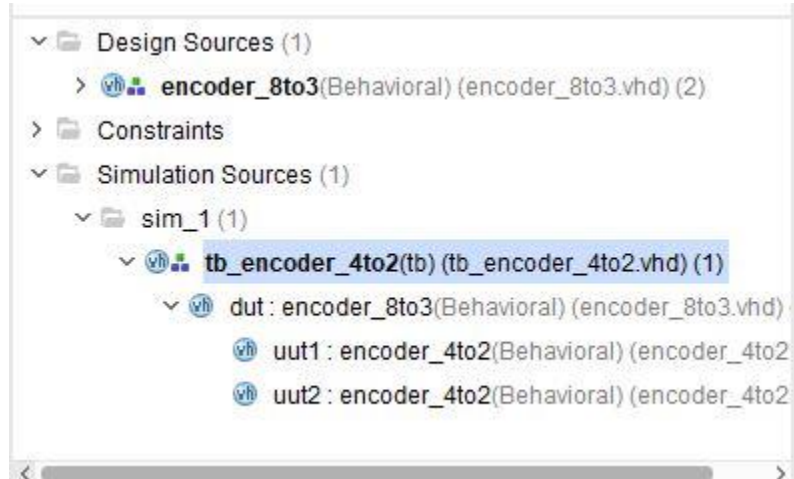
- Notice: there are 2 parts should be revised to combine these two encoders:
1. Change the input V into inout port.
 2. Change the enable of each 4 to 2 encoder from high level into low level.

Write the testbench

```
entity tb_encoder_8to3 is
end tb_encoder_8to3;
architecture tb of tb_encoder_8to3 is
    component encoder_8to3
        port (S_1 : in std_logic_vector (3 downto 0);
              S_0 : in std_logic_vector (3 downto 0);
              en  : in std_logic;
              v   : inout std_logic_vector (1 downto 0);
              O_1 : out std_logic_vector (1 downto 0);
              O_0 : out std_logic_vector (1 downto 0)
            );
    end component;
    signal S_1 : std_logic_vector (3 downto 0);
    signal S_0 : std_logic_vector (3 downto 0);
    signal en  : std_logic;
    signal v   : std_logic_vector (1 downto 0);
    signal O_1 : std_logic_vector (1 downto 0);
    signal O_0 : std_logic_vector (1 downto 0);
```

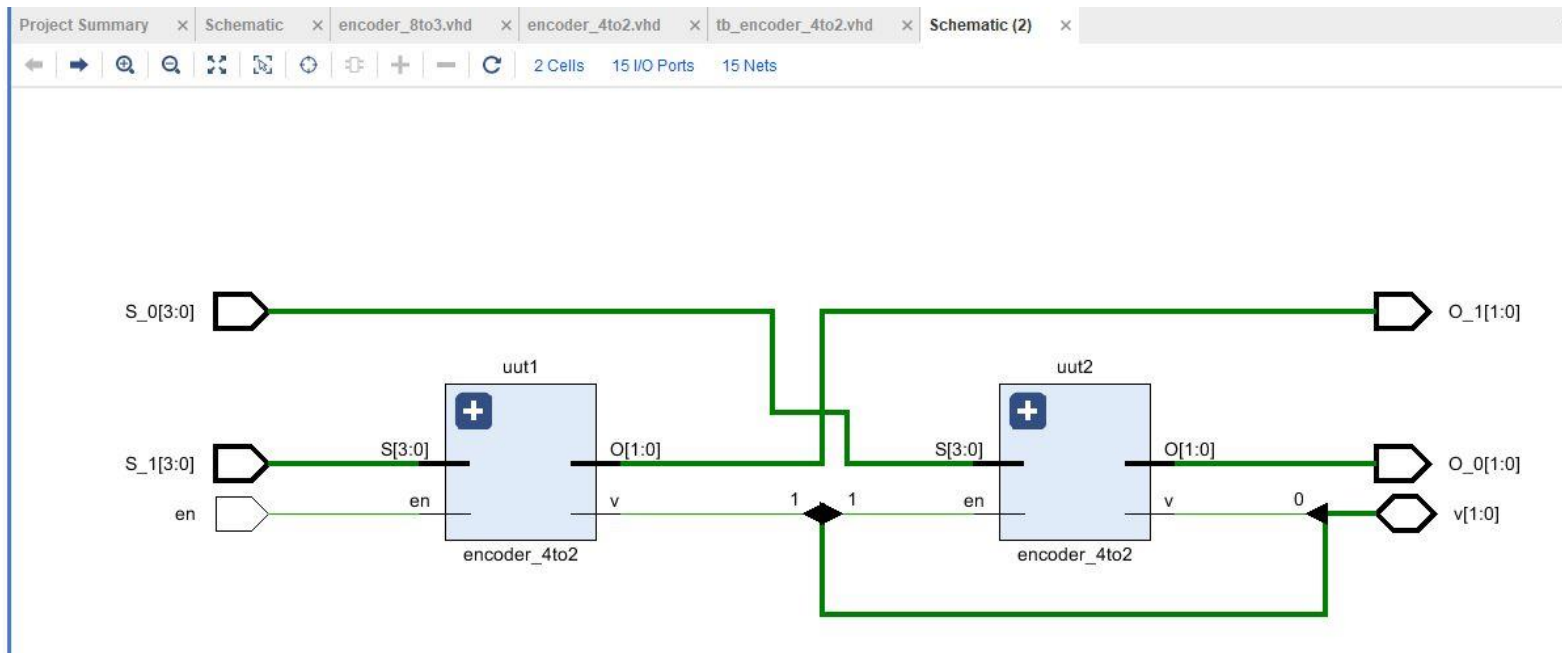
```
begin
    dut : encoder_8to3
        port map (S_1 => S_1,
                  S_0 => S_0,
                  en => en,
                  v  => v,
                  O_1 => O_1,
                  O_0 => O_0);
    stimuli : process
    begin
        S_1 <= "0000";
        S_0 <= "0000";
        en  <= '1';
        wait for 10ns;
        en  <= '0';
        S_1 <= "1111";
        S_0 <= "0000";
        wait for 10ns;
        S_1 <= "0010";
        S_0 <= "0001";
        wait for 10ns;
        S_1 <= "0000";
        S_0 <= "1010";
        wait;
    end process;
end tb;
```

Run simulation and check results



Perform RTL analysis on source file

- Expand the **Open Elaborated Design** entry under the RTL Analysis tasks of the Flow Navigator pane and click on Schematic. And then click **OK**.



- Click “plus” on each entity and you can check the schematic of 4 to 2 encoder.

Additional Video and Tool

- <https://www.youtube.com/watch?v=SSer0c1noBQ&t=183s>
 - This video shows an 8 to 3 priority encoder in ISE (An old software like Vivado, but the VHDL codes are the same) and point out the usage of 'if' and 'elsif'.
- Testbench Generator:
<https://vhdl.lapinoo.net/testbench/>

Tutorial - Remarks

- Please raise your hand if you have questions!
 - 9 course tutors + course Instructors
 - If you finish the tutorial quickly, please try different settings in the Vivado tool.
- You can also refer to the course Canvas homepage.
- Get familiar with the Design tool and VHDL grammar from textbooks or other materials.
- Keep good habits in writing codes and avoid syntax errors.
- Add more functions for your entity, e.g. set the priority by users before it works.
- Pipelined structure will be presented in tutorial 5.