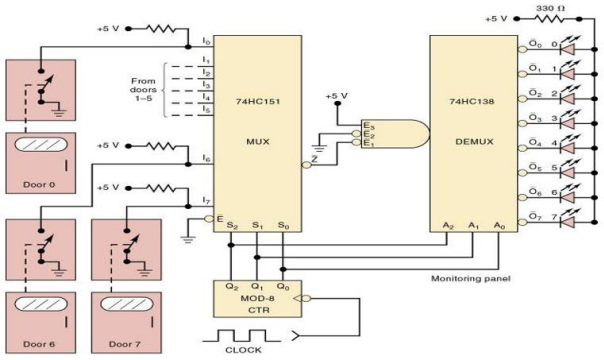
Lab 4 LMP Model -Multiplexer and Demultiplexer in Embedded System

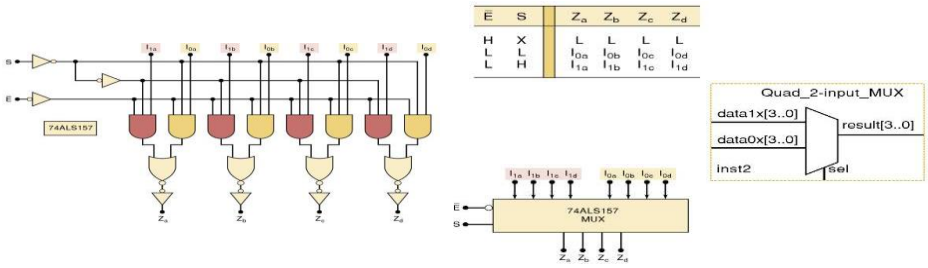
– Implemented by FPGAs

A multiplexer is a device that acts like a selector switch for digital signals. The select inputs are used to specify the input channel that is to be "connected" to the output pins. A demultiplexer works in the opposite direction by taking a digital signal as an input and distributing it to one of its outputs. The 74ALS157 contains four two-input multiplexers.

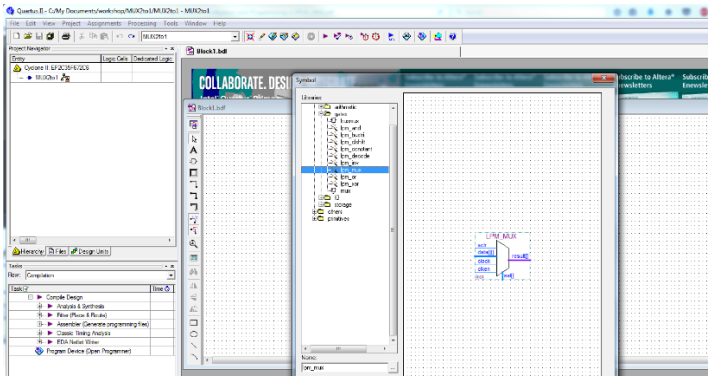
- Security monitoring system using the 74ALS138



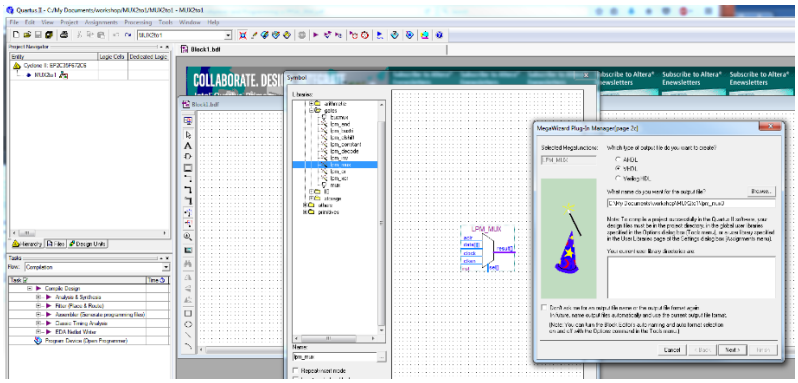
The system shown can handle eight doors, but can be expanded to any number. The door switches are data inputs to the MUX. They produce a HIGH when a door is open and a LOW when it is closed.



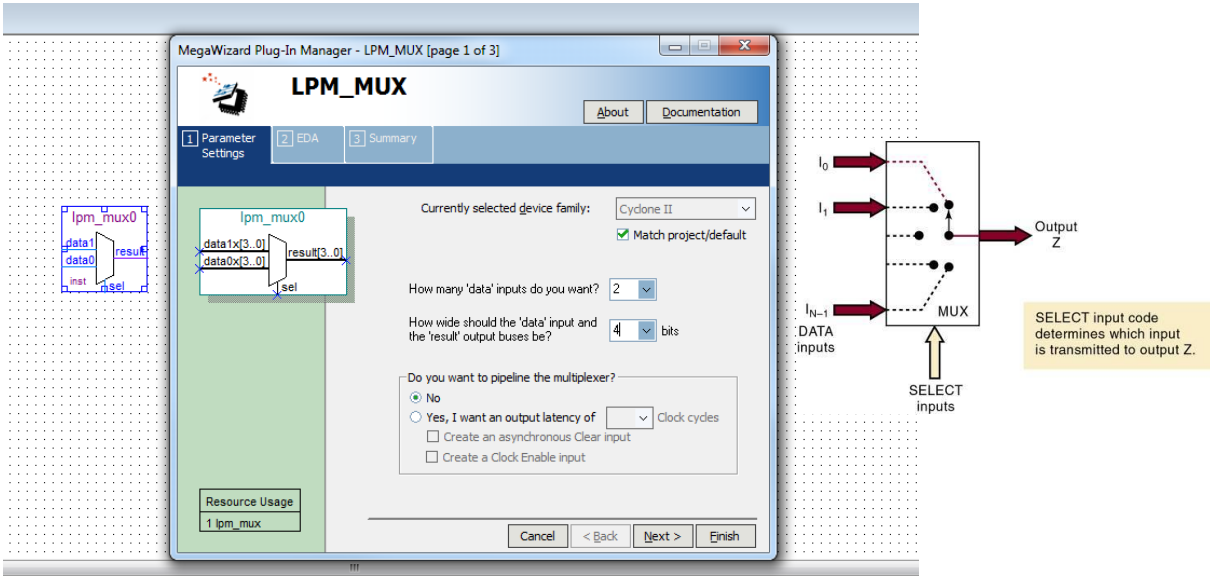
1. To ease the design process for commonly used system, such as adders, the Quartus II provide a build-in **Library of Parameterized Modules (LPM)**. For example, the *bdf* file can utilize four two-input multiplexer *lpm_mux* module.



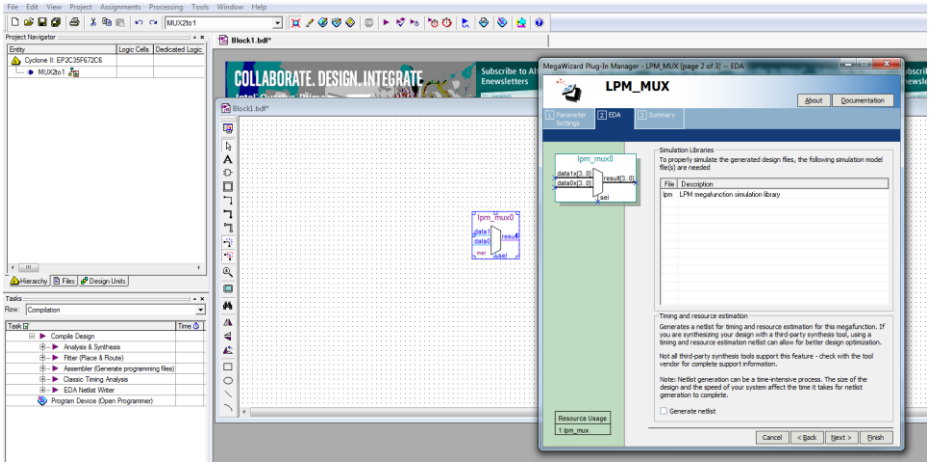
2. Configure *lpm_mux* module to have VHDL as output prograaming code.



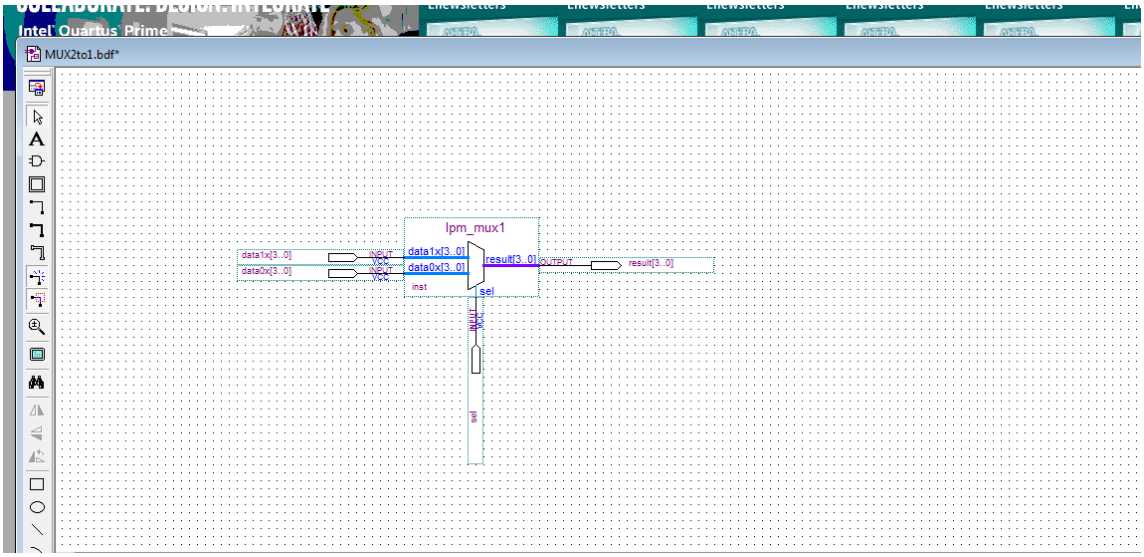
3. Configure multiplexer to select 1 of 2 inputs data source and transmit the selected data to a single output. Each of inputs has four bits.



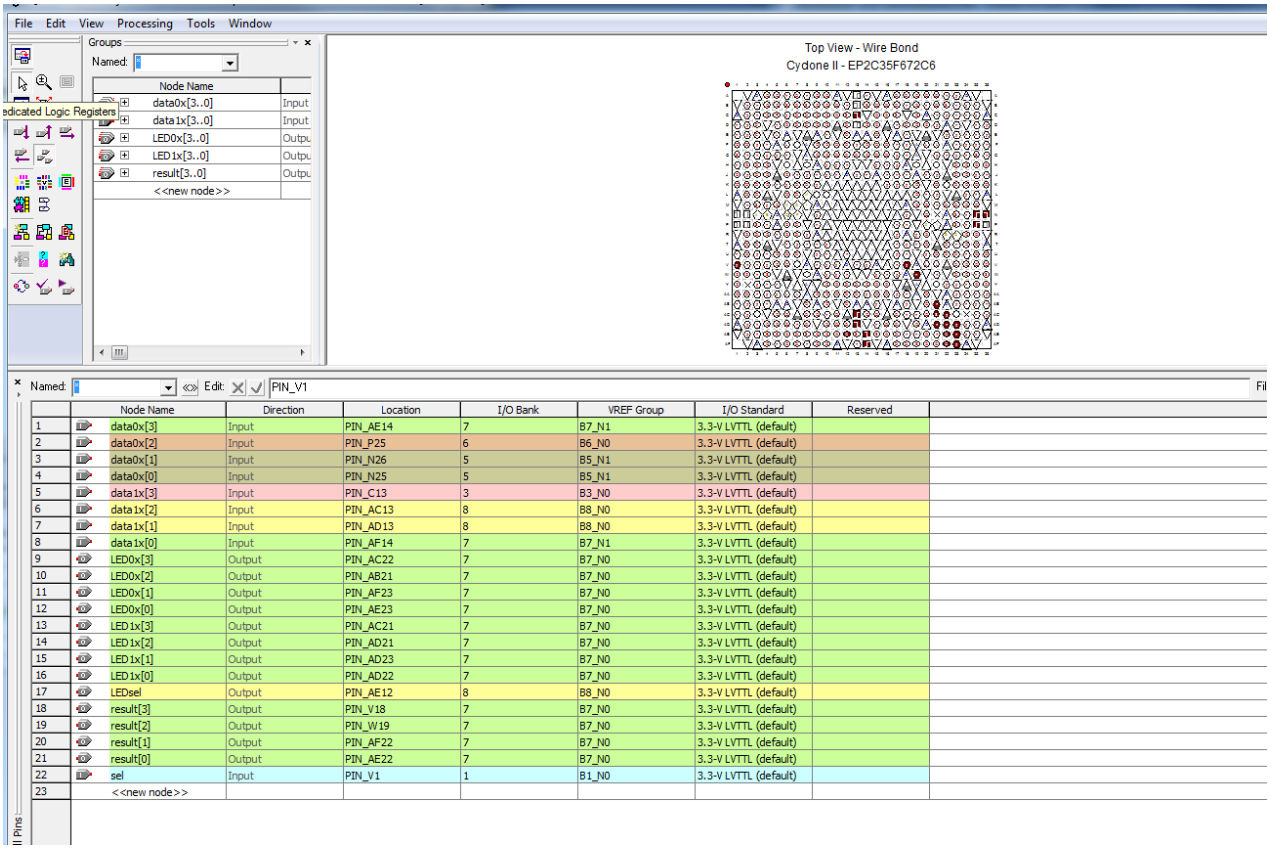
4. Click finish



5. Click finish again

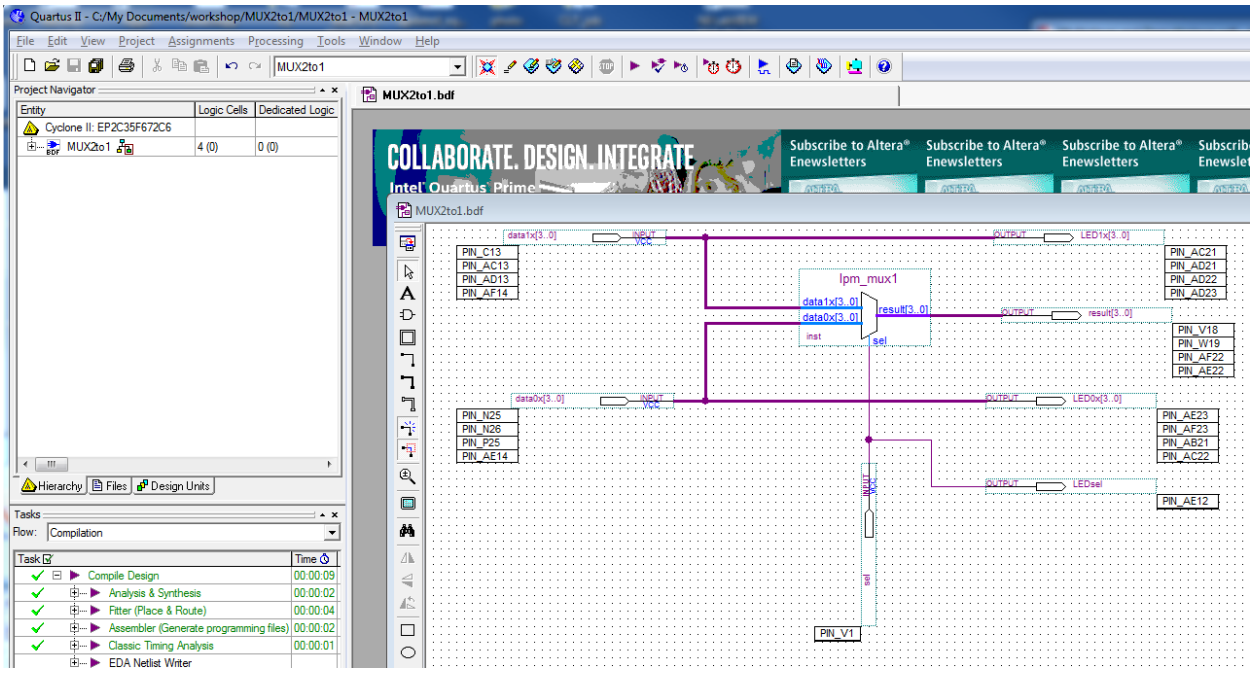


6. Compilation and see if any mistakes.
7. Pin planner-pin assignment



	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	data0x[3]	Input	PIN_AE14	7	B7_N1	3.3-V LVTTL (default)	
2	data0x[2]	Input	PIN_P25	6	B6_N0	3.3-V LVTTL (default)	
3	data0x[1]	Input	PIN_N26	5	B5_N1	3.3-V LVTTL (default)	
4	data0x[0]	Input	PIN_N25	5	B5_N1	3.3-V LVTTL (default)	
5	data1x[3]	Input	PIN_C13	3	B3_N0	3.3-V LVTTL (default)	
6	data1x[2]	Input	PIN_AC13	8	B8_N0	3.3-V LVTTL (default)	
7	data1x[1]	Input	PIN_AD13	8	B8_N0	3.3-V LVTTL (default)	
8	data1x[0]	Input	PIN_AF14	7	B7_N1	3.3-V LVTTL (default)	
9	LED0x[3]	Output	PIN_AC22	7	B7_N0	3.3-V LVTTL (default)	
10	LED0x[2]	Output	PIN_AB21	7	B7_N0	3.3-V LVTTL (default)	
11	LED0x[1]	Output	PIN_AF23	7	B7_N0	3.3-V LVTTL (default)	
12	LED0x[0]	Output	PIN_AE23	7	B7_N0	3.3-V LVTTL (default)	
13	LED1x[3]	Output	PIN_AC21	7	B7_N0	3.3-V LVTTL (default)	
14	LED1x[2]	Output	PIN_AD21	7	B7_N0	3.3-V LVTTL (default)	
15	LED1x[1]	Output	PIN_AD23	7	B7_N0	3.3-V LVTTL (default)	
16	LED1x[0]	Output	PIN_AD22	7	B7_N0	3.3-V LVTTL (default)	
17	LEDsel	Output	PIN_AE12	8	B8_N0	3.3-V LVTTL (default)	
18	result[3]	Output	PIN_V18	7	B7_N0	3.3-V LVTTL (default)	
19	result[2]	Output	PIN_W19	7	B7_N0	3.3-V LVTTL (default)	
20	result[1]	Output	PIN_AF22	7	B7_N0	3.3-V LVTTL (default)	
21	result[0]	Output	PIN_AE22	7	B7_N0	3.3-V LVTTL (default)	
22	sel	Input	PIN_V1	1	B1_N0	3.3-V LVTTL (default)	
23	<<new node>>						

8. Your final design view



9. Compile again to include FPGA device information into your design

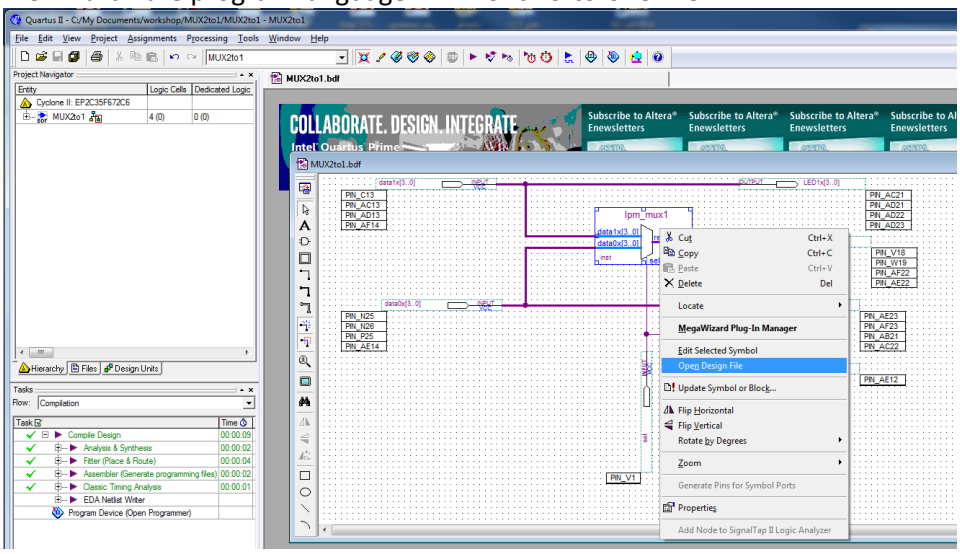
10. Download the design to FPGA IC. Observe the output LEDs and inputs Switches
11. Explore different led lights by your own.
12. Flip SW 16 to observe input (red led light)/output (green light).

    If SW16 is in down position (logic 0), them output (result=data0x);

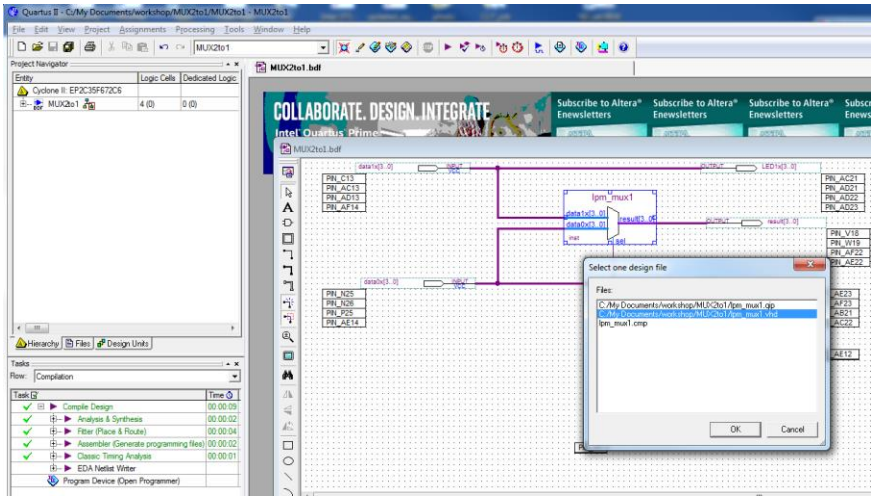
    If SW16 is in up position (logic 1), them output (result=data1x);

selector	Input group 1 (data1x)	Input group 0 (data0x)	Output (result)
SW16	LEDR7, LEDR6, LEDR5, LEDR4	LEDR3, LEDR2, LEDR1, LEDR0	LEDG3, LEDG2, LEDG1,LEDG0
0 (down)	SW7,SW6,SW5,SW4	SW3,SW2,SW1,SW0	SW3,SW2,SW1,SW0
1 (up)	SW7,SW6,SW5,SW4	SW3,SW2,SW1,SW0	SW7,SW6,SW5,SW4

13. View hardware program language VHDL of two-to-one MUX



14. Explore the VHDL (Hardware programming) design source code



15. VHDL source code generated

```vhdl
USE ieee.std_logic_1164.all;
LIBRARY lpm;
USE lpm.lpm_components.all;

ENTITY lpm_mux1 IS
    PORT
    (
        data0x      : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        data1x      : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        sel     : IN STD_LOGIC ;
        result      : OUT STD_LOGIC_VECTOR (3 DOWNTO 0)
    );
END lpm_mux1;

ARCHITECTURE SYN OF lpm_mux1 IS

--  type STD_LOGIC_2D is array (NATURAL RANGE <>, NATURAL RANGE <>) of STD_LOGIC;

    SIGNAL sub_wire0    : STD_LOGIC_VECTOR (3 DOWNTO 0);
    SIGNAL sub_wire1    : STD_LOGIC ;
    SIGNAL sub_wire2    : STD_LOGIC_VECTOR (0 DOWNTO 0);
    SIGNAL sub_wire3    : STD_LOGIC_VECTOR (3 DOWNTO 0);
    SIGNAL sub_wire4    : STD_LOGIC_2D (1 DOWNTO 0, 3 DOWNTO 0);
    SIGNAL sub_wire5    : STD_LOGIC_VECTOR (3 DOWNTO 0);

BEGIN
    sub_wire5    <= data0x(3 DOWNTO 0);
    result    <= sub_wire0(3 DOWNTO 0);
    sub_wire1    <= sel;
    sub_wire2(0)    <= sub_wire1;
    sub_wire3    <= data1x(3 DOWNTO 0);
    sub_wire4(1, 0)    <= sub_wire3(0);
    sub_wire4(1, 1)    <= sub_wire3(1);
    sub_wire4(1, 2)    <= sub_wire3(2);
    sub_wire4(1, 3)    <= sub_wire3(3);
    sub_wire4(0, 0)    <= sub_wire5(0);
    sub_wire4(0, 1)    <= sub_wire5(1);
    sub_wire4(0, 2)    <= sub_wire5(2);
    sub_wire4(0, 3)    <= sub_wire5(3);

    lpm_mux_component : lpm_mux
    GENERIC MAP (
        lpm_size => 2,
        lpm_type => "LPM_MUX",
        lpm_width => 4,
        lpm_widths => 1
    )
    PORT MAP (
        sel => sub_wire2,
        data => sub_wire4,
        result => sub_wire0
    );

END SYN;
```