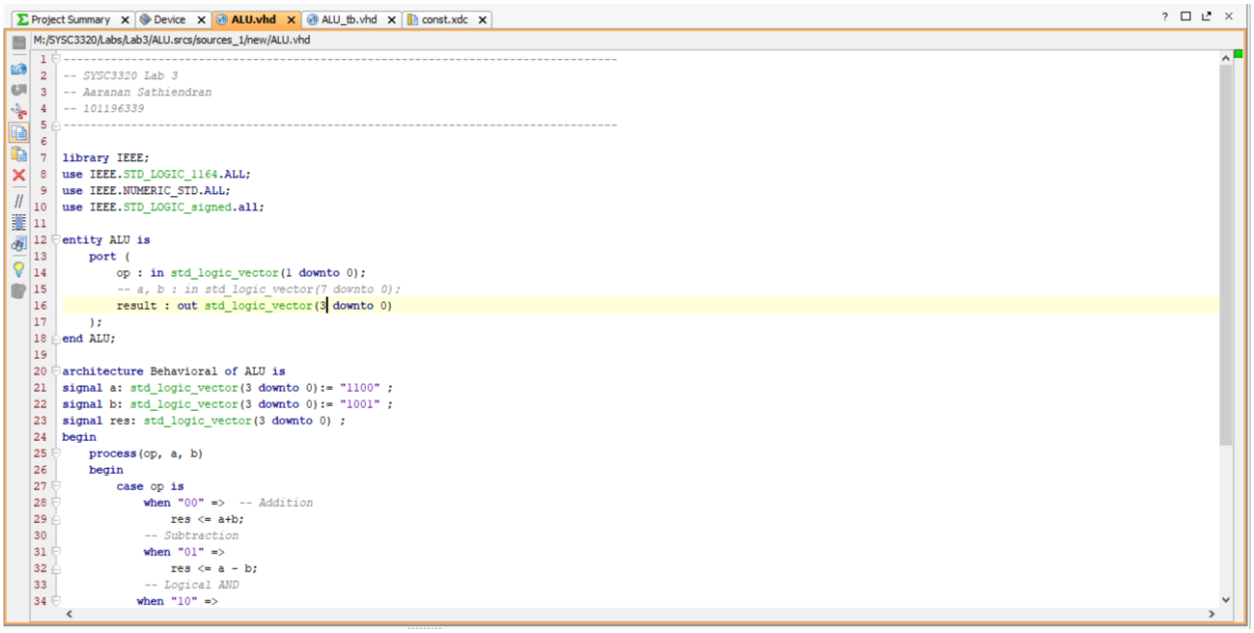
# Department of
# Systems and Computer Engineering

## SYSC-3320: Laboratory 3

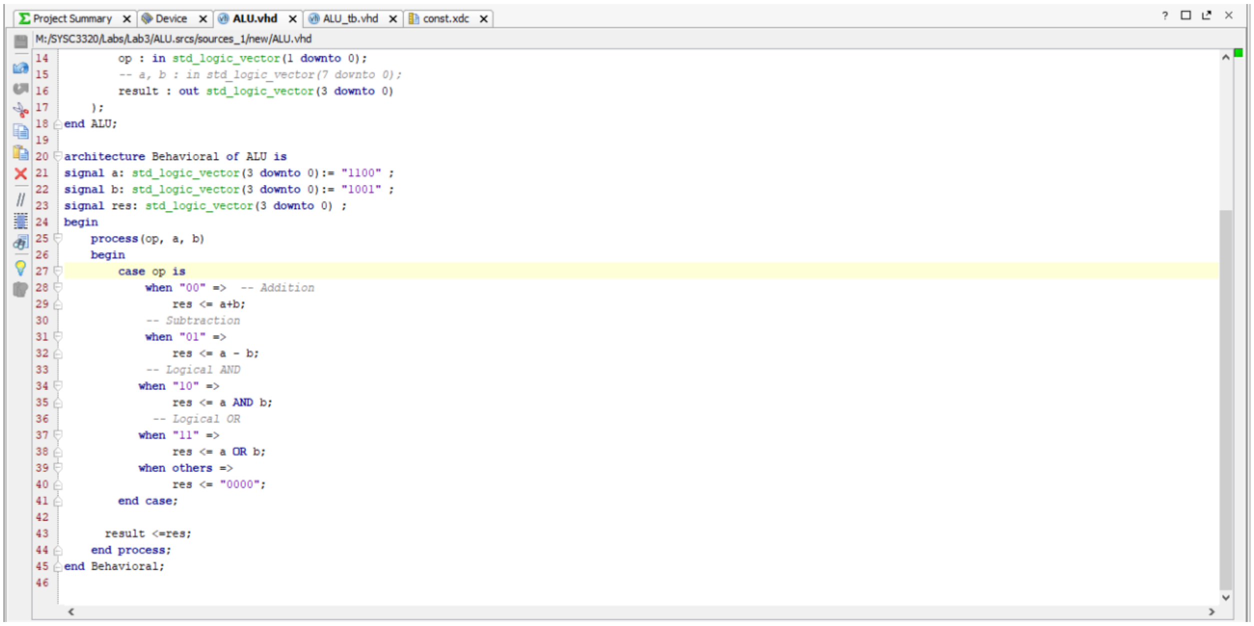
## An 8-Bit ALU Design and Implementation using VHDL and Zynq-7000 SoC boards family

Completed by: Aaranan Sathiendran (101196339)

## 8-bit ALU design (modified to 4 bits in order to work with LEDs):

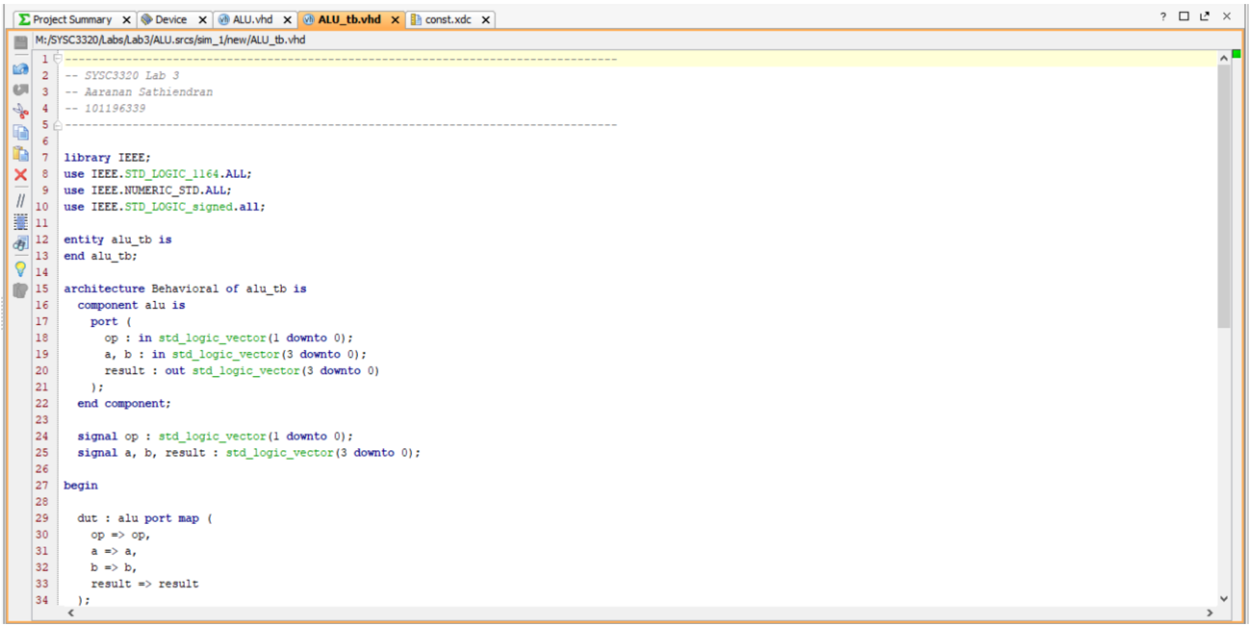


```
--------------------------------------------------------------------------------
-- SYSC3320 Lab 3
-- Aaranan Sathiendran
-- 101196339
--------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_signed.all;

entity ALU is
    port (
        op : in std_logic_vector(1 downto 0);
        -- a, b : in std_logic_vector(7 downto 0);
        result : out std_logic_vector(3 downto 0)
    );
end ALU;

architecture Behavioral of ALU is
signal a: std_logic_vector(3 downto 0):= "1100" ;
signal b: std_logic_vector(3 downto 0):= "1001" ;
signal res: std_logic_vector(3 downto 0) ;
begin
    process(op, a, b)
    begin
        case op is
            when "00" =>  -- Addition
                res <= a+b;
            -- Subtraction
            when "01" =>
                res <= a - b;
            -- Logical AND
           when "10" =>
```

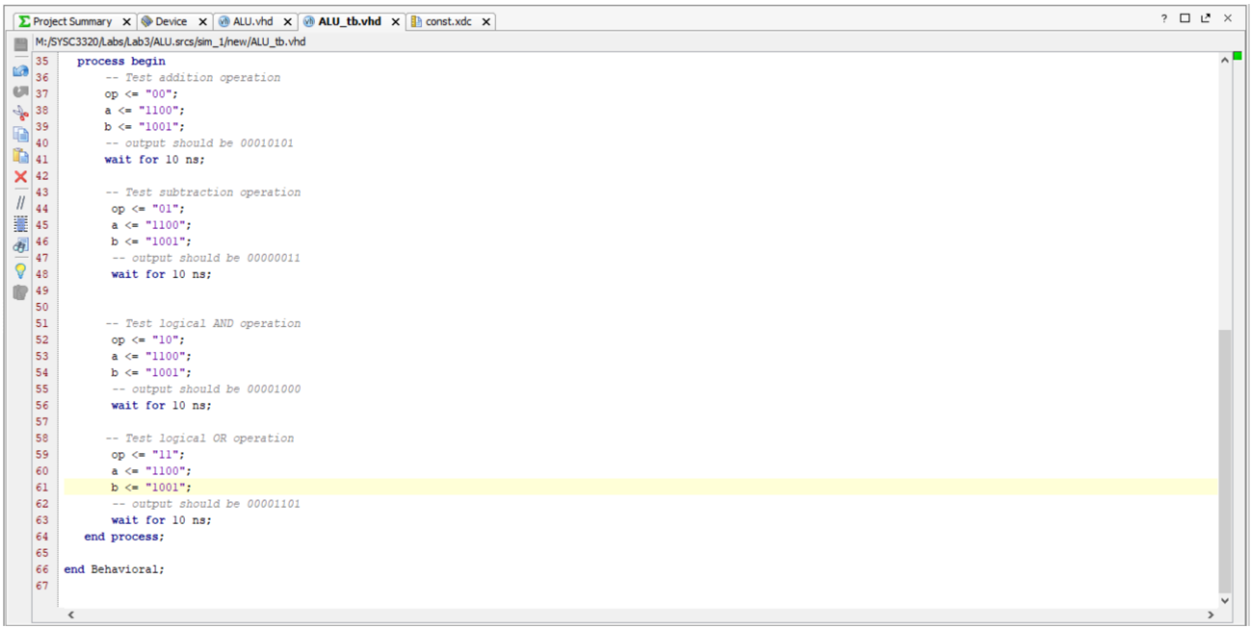


```
        op : in std_logic_vector(1 downto 0);
        -- a, b : in std_logic_vector(7 downto 0);
        result : out std_logic_vector(3 downto 0)
    );
end ALU;

architecture Behavioral of ALU is
signal a: std_logic_vector(3 downto 0):= "1100" ;
signal b: std_logic_vector(3 downto 0):= "1001" ;
signal res: std_logic_vector(3 downto 0) ;
begin
    process(op, a, b)
    begin
        case op is
            when "00" =>  -- Addition
                res <= a+b;
            -- Subtraction
            when "01" =>
                res <= a - b;
            -- Logical AND
           when "10" =>
                res <= a AND b;
             -- Logical OR
           when "11" =>
                res <= a OR b;
           when others =>
                res <= "0000";
        end case;

      result <=res;
    end process;
end Behavioral;
```

**Figure 1: VHDL Code for 8-bit ALU**



```vhdl
--------------------------------------------------------------------------------
-- SYSC3320 Lab 3
-- Aaranan Sathiendran
-- 101196339
--------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_signed.all;

entity alu_tb is
end alu_tb;

architecture Behavioral of alu_tb is
  component alu is
    port (
      op : in std_logic_vector(1 downto 0);
      a, b : in std_logic_vector(3 downto 0);
      result : out std_logic_vector(3 downto 0)
    );
  end component;

  signal op : std_logic_vector(1 downto 0);
  signal a, b, result : std_logic_vector(3 downto 0);

begin

  dut : alu port map (
    op => op,
    a => a,
    b => b,
    result => result
  );
```



```vhdl
  process begin
      -- Test addition operation
      op <= "00";
      a <= "1100";
      b <= "1001";
      -- output should be 00010101
      wait for 10 ns;

      -- Test subtraction operation
       op <= "01";
       a <= "1100";
       b <= "1001";
       -- output should be 00000011
       wait for 10 ns;


      -- Test logical AND operation
       op <= "10";
       a <= "1100";
       b <= "1001";
       -- output should be 00001000
       wait for 10 ns;

      -- Test logical OR operation
       op <= "11";
       a <= "1100";
       b <= "1001";
       -- output should be 00001101
       wait for 10 ns;
   end process;

end Behavioral;
```

**Figure 2: VHDL Code for 8-bit ALU testbench**

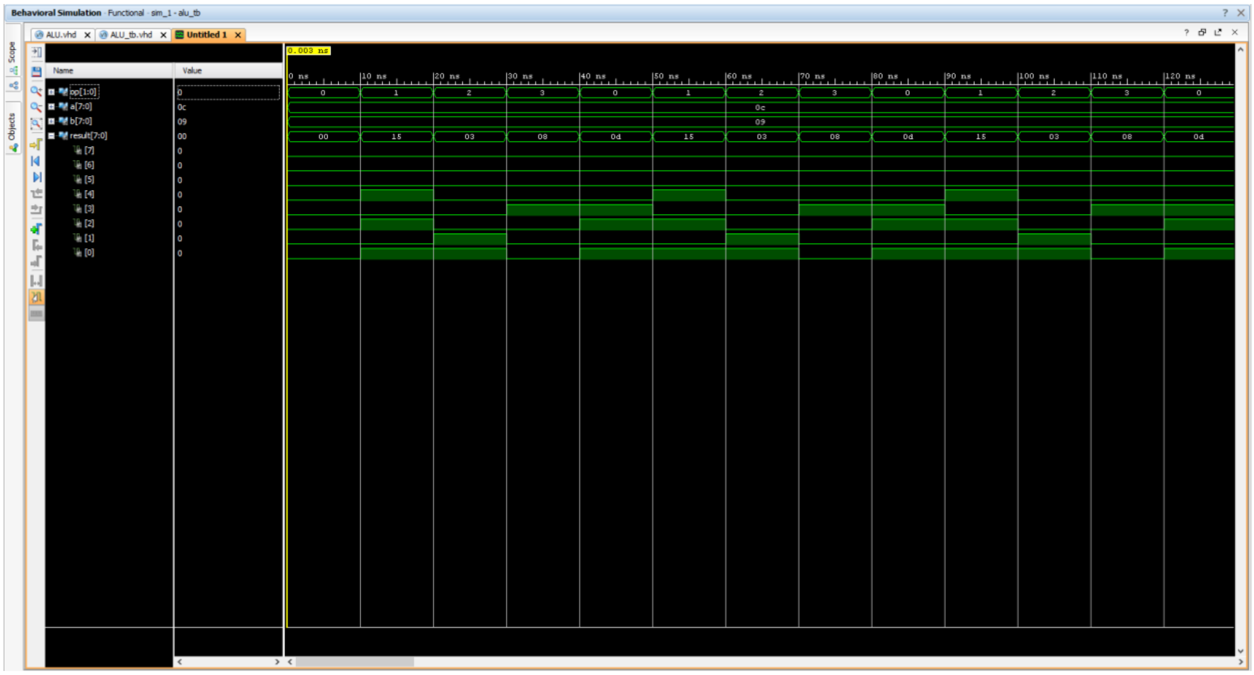
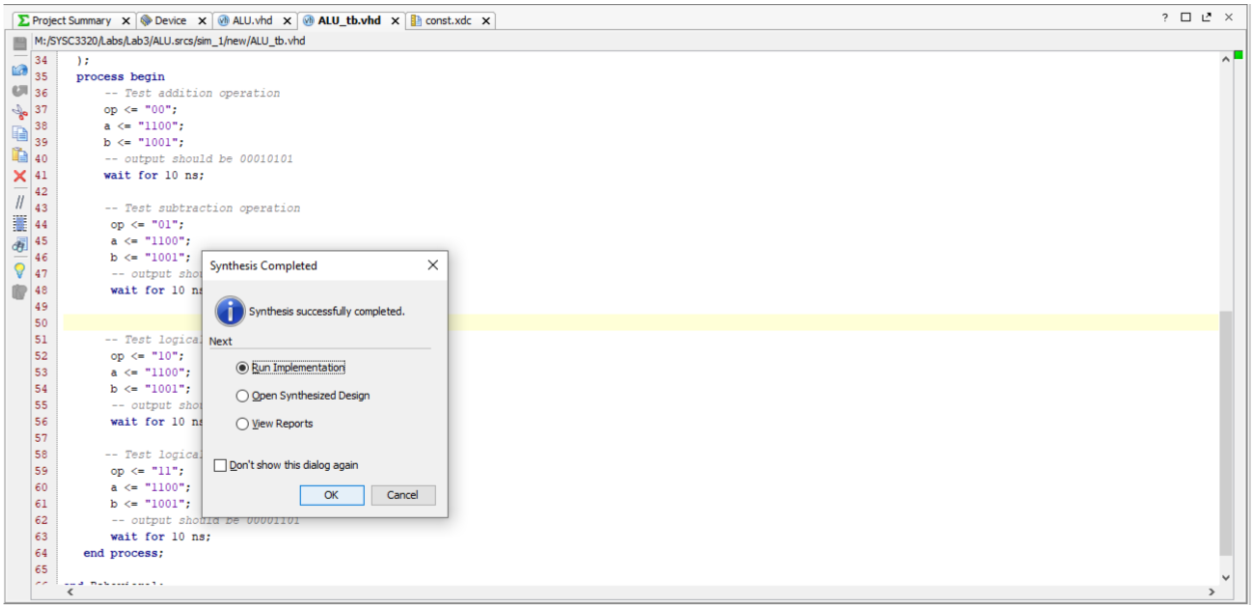


**Figure 3: Resulting Simulation Waveform for 8-bit ALU**

Figure 3 shows that the implemented 8-bit ALU follows the expected behaviour. Using fixed inputs a = “00001100” (12 in decimal) and b = “00001001” (9 in decimal), when the opcode is set to “00” from 10 – 20 ns to select the addition operation, the output is 00010101 (21). When the opcode is set to “01” from 20 – 30 ns to select the subtraction operation, the output is “00000011” (3). When the opcode is set to “10” from 30 – 40 ns to select the AND operation, the output is “00001000”. Lastly, when the opcode is set to “11” from 40 – 50 ns to select the OR operation, the output is “00001101”.



```
    );
    process begin
        -- Test addition operation
        op <= "00";
        a <= "1100";
        b <= "1001";
        -- output should be 00010101
        wait for 10 ns;

         -- Test subtraction operation
         op <= "01";
         a <= "1100";
         b <= "1001";
         -- output sho
         wait for 10 ns


         -- Test logica
         op <= "10";
         a <= "1100";
         b <= "1001";
         -- output sho
         wait for 10 ns

         -- Test logica
         op <= "11";
         a <= "1100";
         b <= "1001";
         -- output should be 00001101
         wait for 10 ns;
     end process;

end Behavioral;
```

Synthesis Completed

Synthesis successfully completed.

Next

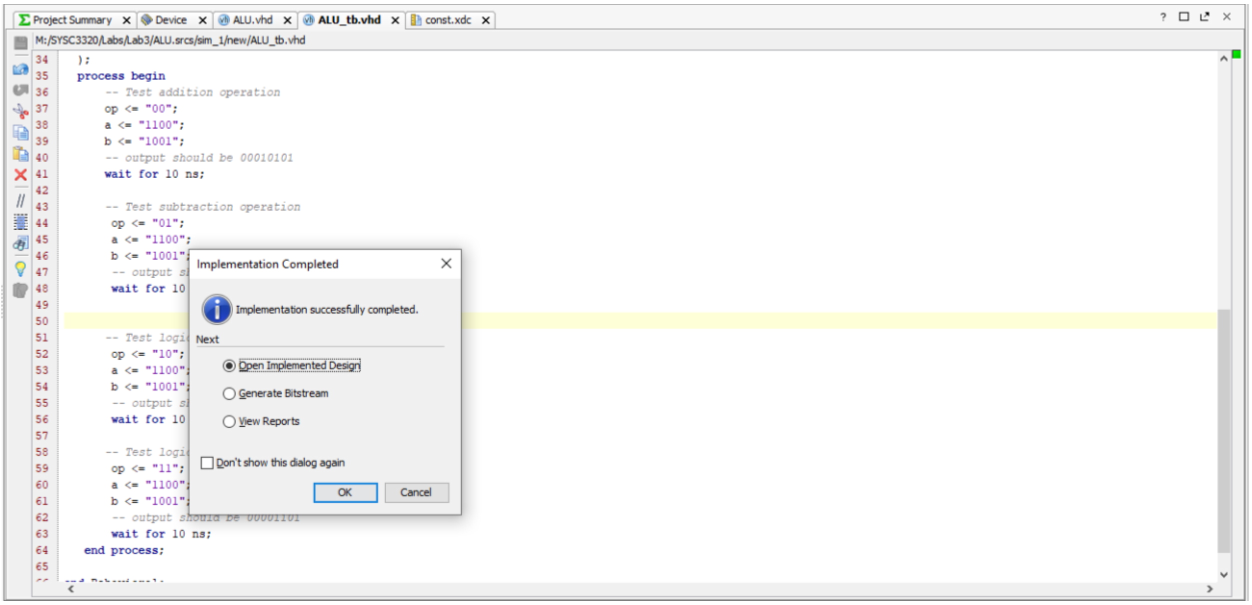
● Run Implementation

○ Open Synthesized Design

○ View Reports

☐ Don't show this dialog again

## Figure 4: Successful Synthesis of 8-bit ALU



```
    );
    process begin
        -- Test addition operation
        op <= "00";
        a <= "1100";
        b <= "1001";
        -- output should be 00010101
        wait for 10 ns;

        -- Test subtraction operation
         op <= "01";
         a <= "1100";
         b <= "1001";
         -- output s
         wait for 10


        -- Test logi
         op <= "10";
         a <= "1100";
         b <= "1001";
         -- output s
         wait for 10

        -- Test logi
         op <= "11";
         a <= "1100";
         b <= "1001";
         -- output should be 00001101
         wait for 10 ns;
     end process;

end Behavioral;
```

Implementation Completed

Implementation successfully completed.

Next

● Open Implemented Design

○ Generate Bitstream

○ View Reports

☐ Don't show this dialog again

## Figure 5: Successful Implementation of 8-bit ALU

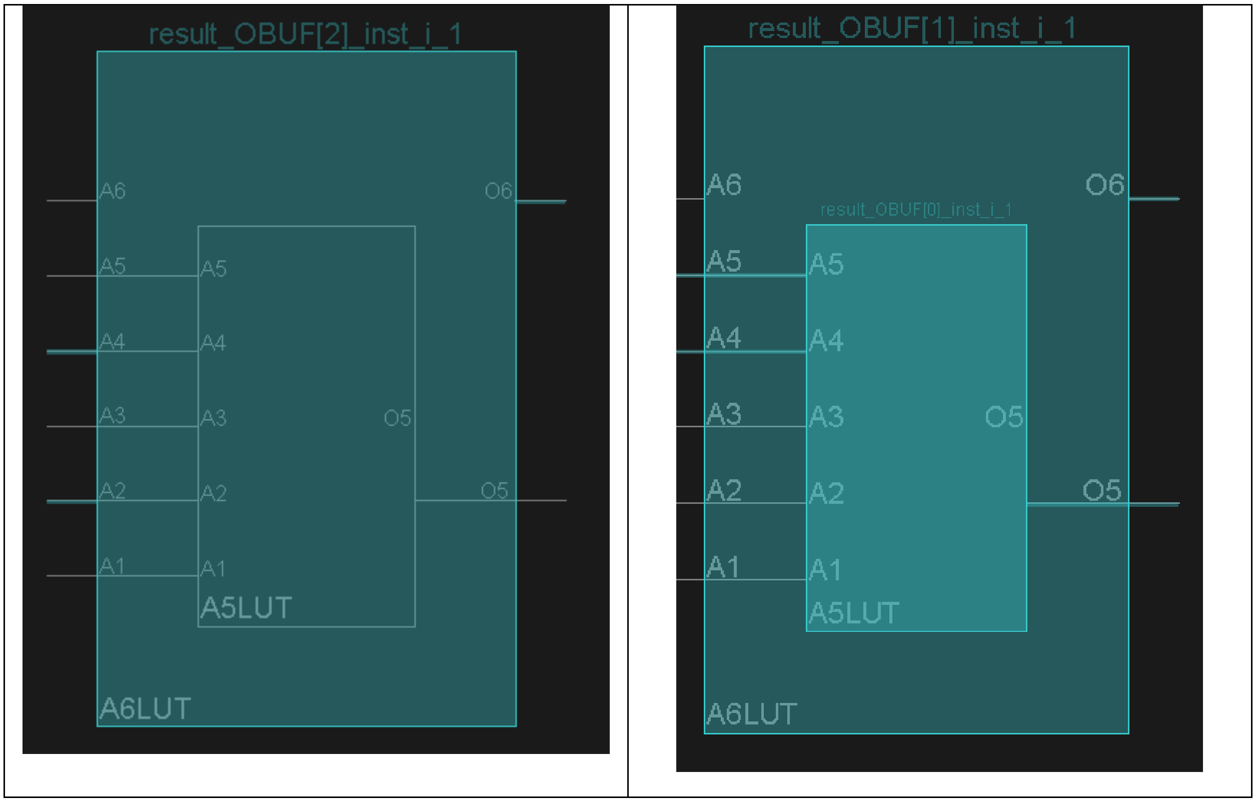
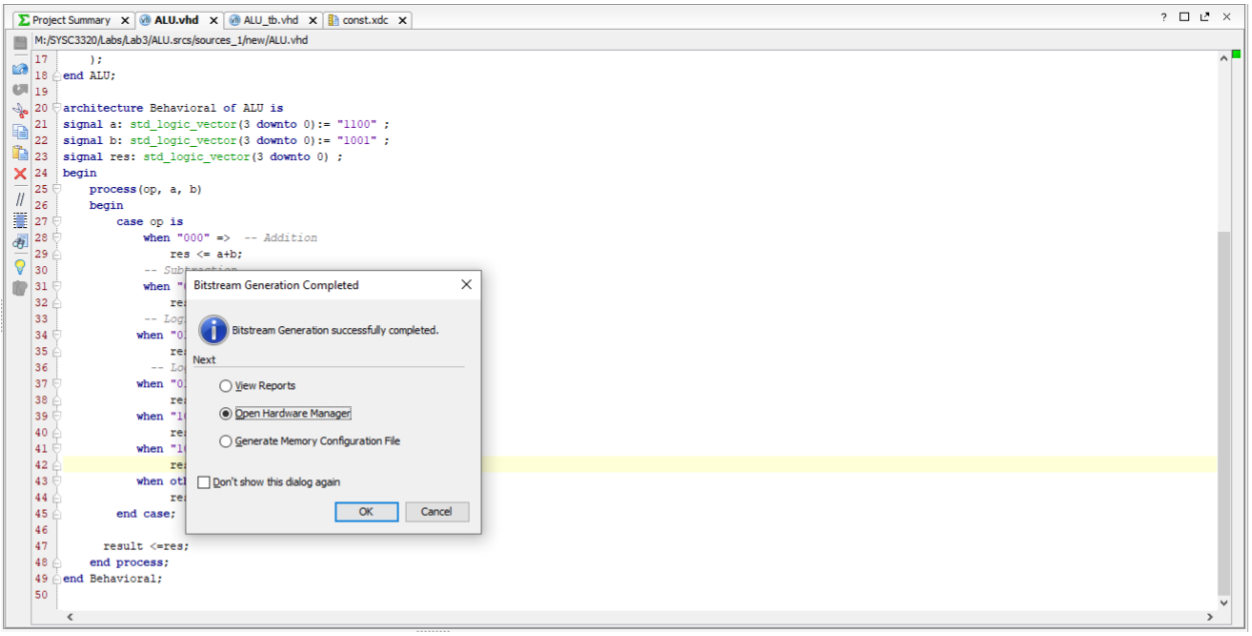


**Figure 6: Physical Implementation of 8-bit ALU on Zybo Board**



M:/SYSC3320/Labs/Lab3/ALU.srcs/sources_1/new/ALU.vhd

```
    );
end ALU;

architecture Behavioral of ALU is
signal a: std_logic_vector(3 downto 0):= "1100" ;
signal b: std_logic_vector(3 downto 0):= "1001" ;
signal res: std_logic_vector(3 downto 0) ;
begin
    process(op, a, b)
    begin
        case op is
            when "000" =>  -- Addition
                res <= a+b;
            -- Subtraction
            when "
                re
            -- Log
           when "0
                re
             -- Lo
           when "0
                re
           when "1
                re
           when "1
                re
           when ot
                re
        end case;

      result <=res;
    end process;
end Behavioral;
```

Bitstream Generation Completed

Bitstream Generation successfully completed.

Next

○ View Reports

● Open Hardware Manager

○ Generate Memory Configuration File

□ Don't show this dialog again

OK   Cancel

**Figure 7: Successful Bitstream Generation for 8-bit ALU**

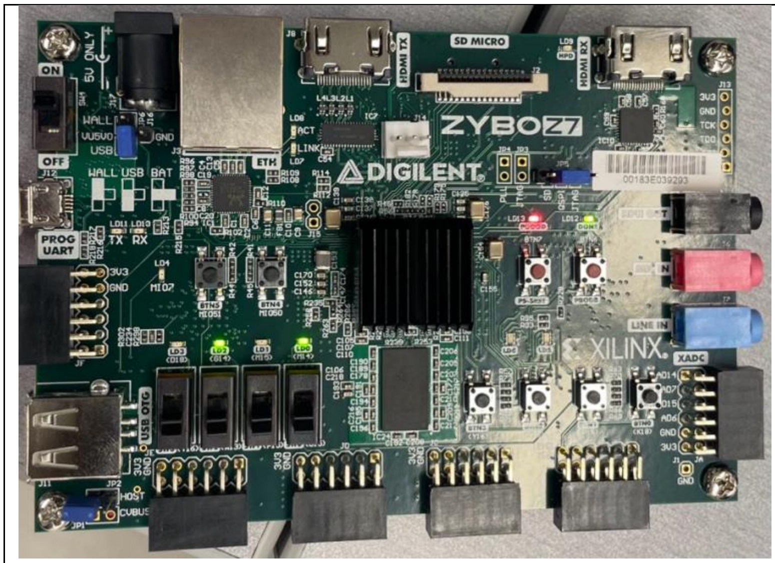


**Figure 8: Physical implementation of ALU on Zybo board showing successful addition**

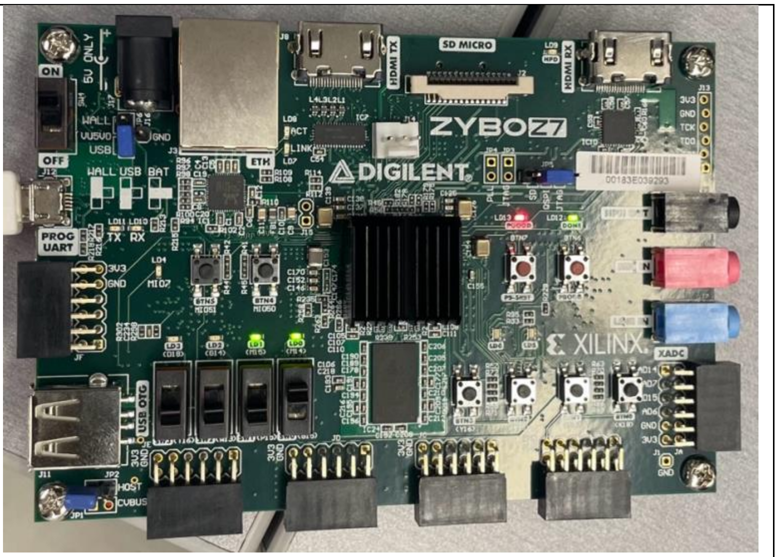


**Figure 9: Physical implementation of ALU on Zybo board showing successful subtraction**

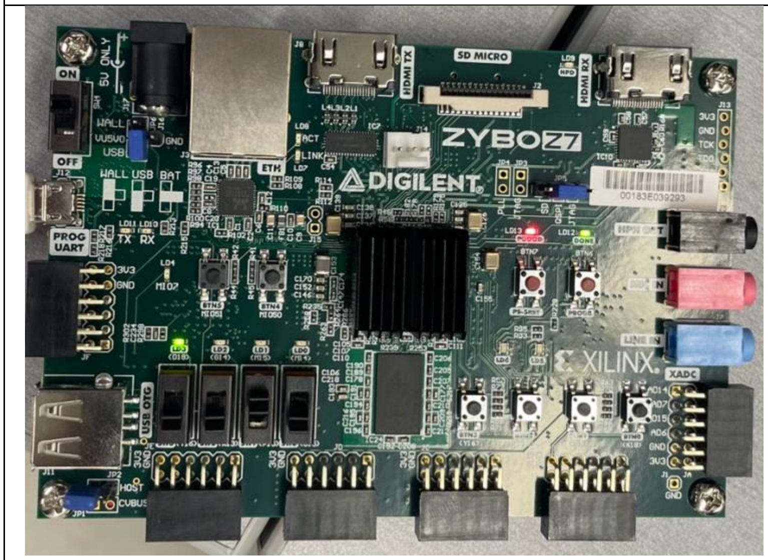


**Figure 10: Physical implementation of ALU on Zybo board showing successful AND operation**

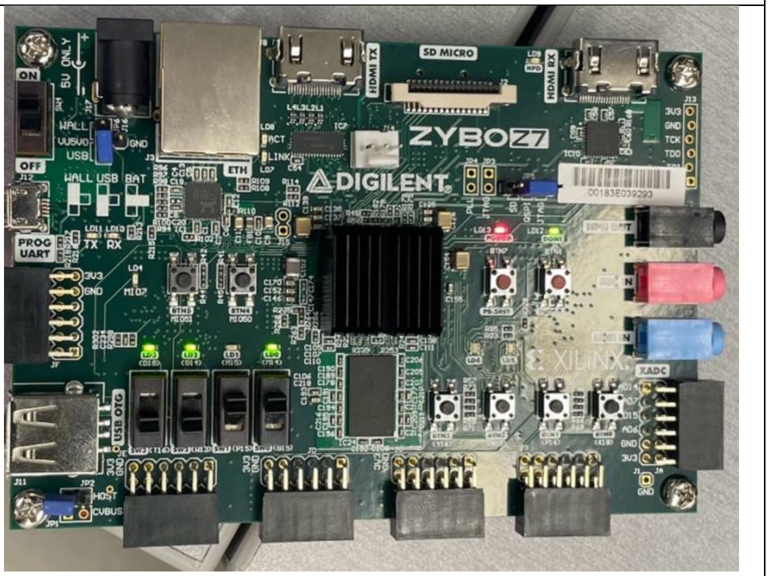
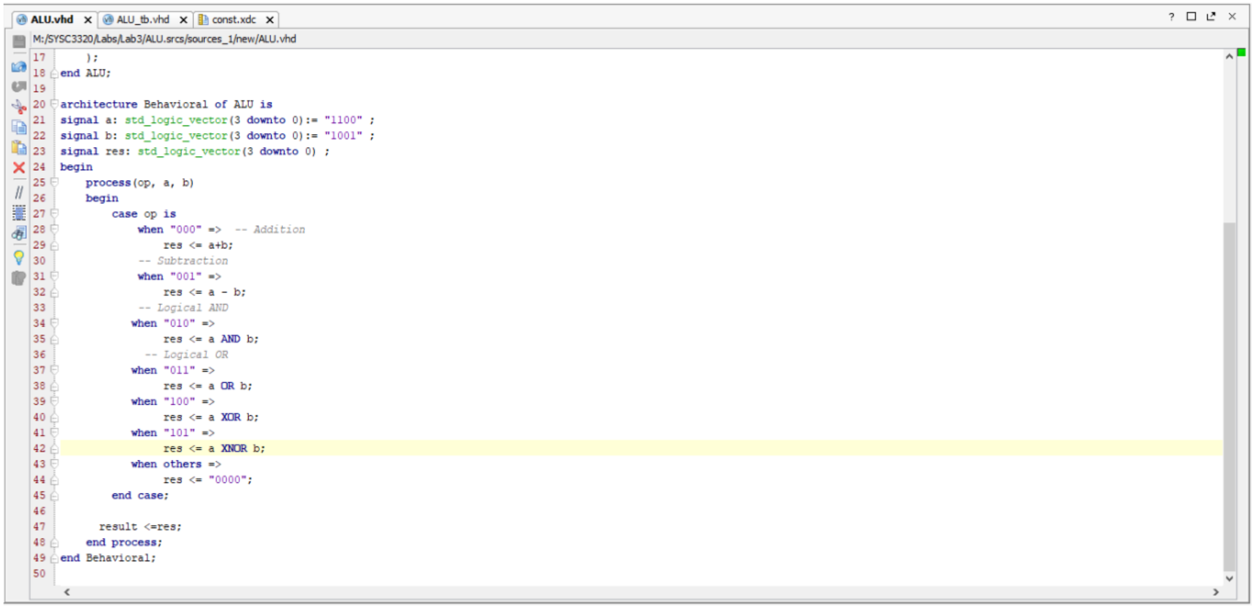


**Figure 11: Physical implementation of ALU on Zybo board showing successful OR operation**

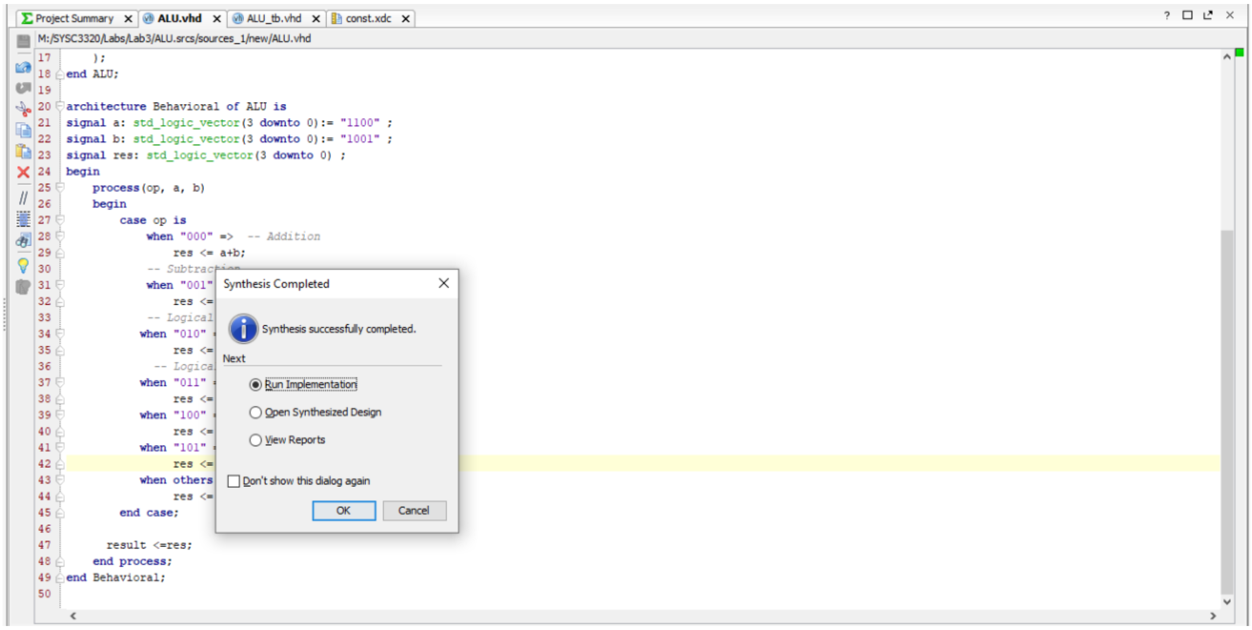
Figures 8 – 11 show that the implemented ALU follows the expected behaviour as per the simulation in Figure 3. Using fixed inputs a = “00001100” (12 in decimal) and b = “00001001” (9 in decimal), when the addition operation is selected by turning switch 0 and 1 off (“00”) as in Figure 8, LEDs 0 and 2 are on (representing “0101”. In Figure 9, when the subtraction operation is selected by turning switch 0 on (“01”), LEDs 0 and 1 are on, representing “0011”. In Figure 10, when the AND operation is selected by turning switch 1 on and switch 0 off (“10”), LED 3 is on, representing “1000”. Lastly, in Figure 11, when the OR operation is selected by turning switch 1 and 0 on (“11”), LEDs 3, 2, and 0 are on, representing “1101”.

## 8-bit ALU design with additional XOR and XNOR operations implemented:



```
    );
end ALU;

architecture Behavioral of ALU is
signal a: std_logic_vector(3 downto 0):= "1100" ;
signal b: std_logic_vector(3 downto 0):= "1001" ;
signal res: std_logic_vector(3 downto 0) ;
begin
    process(op, a, b)
    begin
        case op is
            when "000" =>  -- Addition
                res <= a+b;
            -- Subtraction
            when "001" =>
                res <= a - b;
            -- Logical AND
           when "010" =>
                res <= a AND b;
             -- Logical OR
           when "011" =>
                res <= a OR b;
           when "100" =>
                res <= a XOR b;
           when "101" =>
                res <= a XNOR b;
           when others =>
                res <= "0000";
        end case;

      result <=res;
    end process;
end Behavioral;
```

**Figure 12: VHDL Code for 8-bit ALU with XOR and XNOR operations implemented**



```
    );
end ALU;

architecture Behavioral of ALU is
signal a: std_logic_vector(3 downto 0):= "1100" ;
signal b: std_logic_vector(3 downto 0):= "1001" ;
signal res: std_logic_vector(3 downto 0) ;
begin
    process(op, a, b)
    begin
        case op is
            when "000" =>  -- Addition
                res <= a+b;
            -- Subtrac
            when "001"
                res <=
            -- Logical
           when "010"
                res <=
             -- Logica
           when "011"
                res <=
           when "100"
                res <=
           when "101"
                res <=
           when others
                res <=
        end case;

      result <=res;
    end process;
end Behavioral;
```

Synthesis Completed

Synthesis successfully completed.

Next

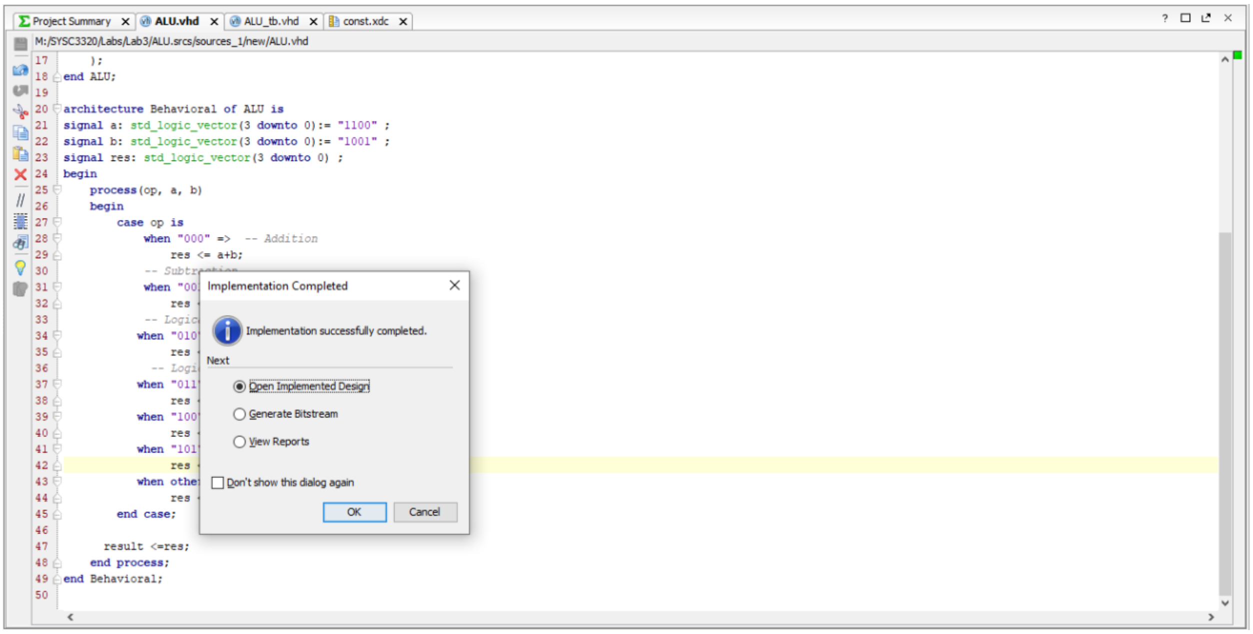
● Run Implementation

○ Open Synthesized Design

○ View Reports

□ Don't show this dialog again

**Figure 13: Successful Synthesis of 8-bit ALU with XOR and XNOR operations implemented**



M:/SYSC3320/Labs/Lab3/ALU.srcs/sources_1/new/ALU.vhd

```vhdl
    );
end ALU;

architecture Behavioral of ALU is
signal a: std_logic_vector(3 downto 0):= "1100" ;
signal b: std_logic_vector(3 downto 0):= "1001" ;
signal res: std_logic_vector(3 downto 0) ;
begin
    process(op, a, b)
    begin
        case op is
            when "000" =>  -- Addition
                res <= a+b;
            -- Subtr
            when "00
                res
            -- Logic
           when "010
                res
             -- Logi
           when "011
                res
           when "100
                res
           when "101
                res
           when othe
                res
        end case;

      result <=res;
    end process;
end Behavioral;
```

Implementation Completed

Implementation successfully completed.

Next

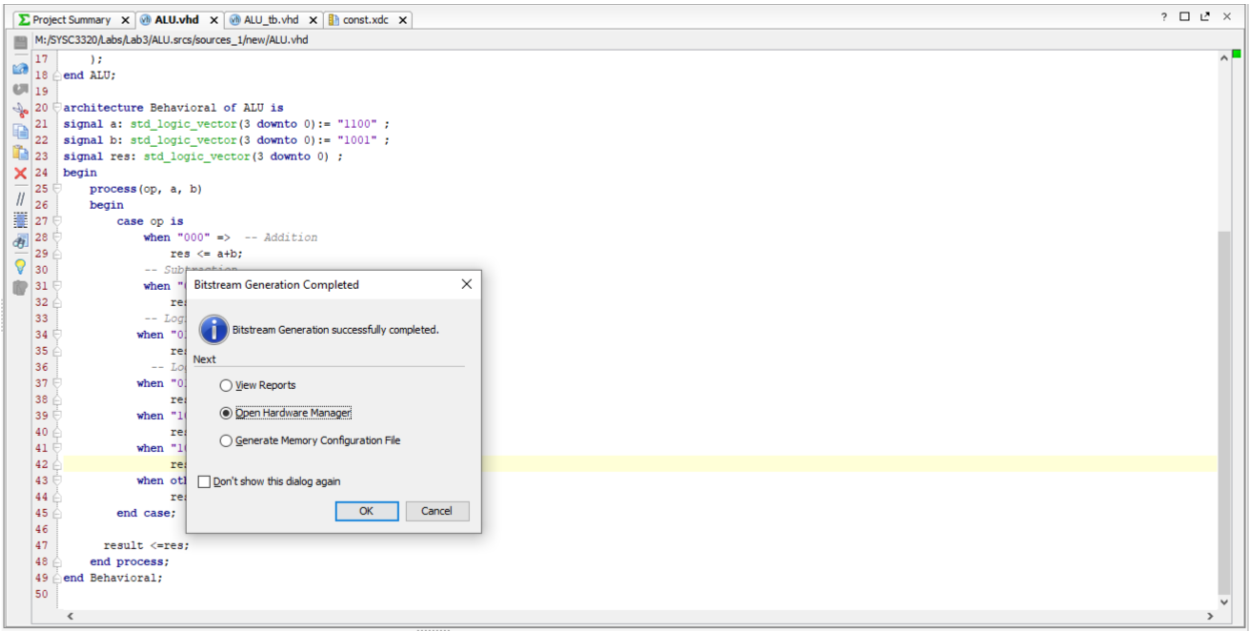
● Open Implemented Design

○ Generate Bitstream

○ View Reports

☐ Don't show this dialog again

**Figure 14: Successful Implementation of 8-bit ALU with XOR and XNOR operations implemented**



M:/SYSC3320/Labs/Lab3/ALU.srcs/sources_1/new/ALU.vhd

```vhdl
    );
end ALU;

architecture Behavioral of ALU is
signal a: std_logic_vector(3 downto 0):= "1100" ;
signal b: std_logic_vector(3 downto 0):= "1001" ;
signal res: std_logic_vector(3 downto 0) ;
begin
    process(op, a, b)
    begin
        case op is
            when "000" =>  -- Addition
                res <= a+b;
            -- Sub
            when "
                re
            -- Log
           when "0
                re
             -- Lo
           when "0
                re
           when "1
                re
           when "1
                re
           when ot
                re
        end case;

      result <=res;
    end process;
end Behavioral;
```

Bitstream Generation Completed

Bitstream Generation successfully completed.

Next

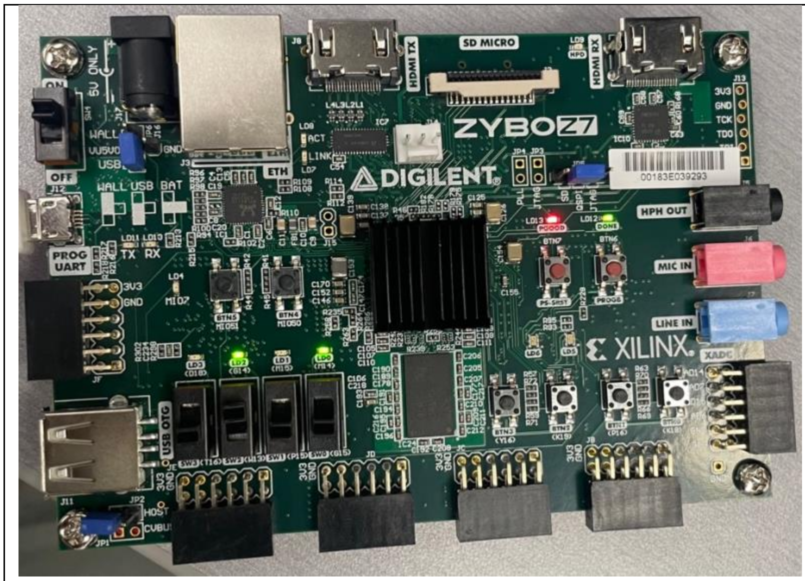
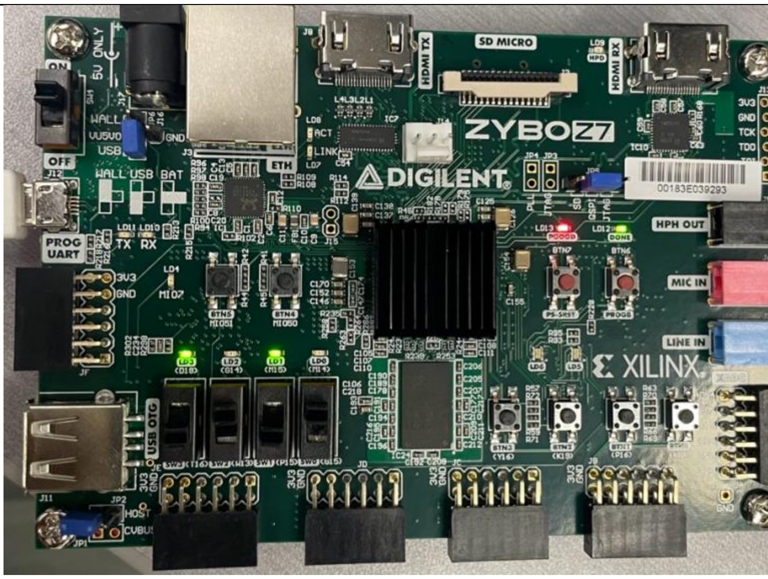
○ View Reports

● Open Hardware Manager

○ Generate Memory Configuration File

☐ Don't show this dialog again

**Figure 15: Successful bitstream generation for 8-bit ALU with XOR and XNOR operations implemented**

| | |
|---|---|
| <br>**Figure 16: Physical implementation of ALU on Zybo board showing successful XOR operation** | <br>**Figure 17: Physical implementation of ALU on Zybo board showing successful XNOR operation** |

Figures 16 and 17 show that the additional operations for the ALU are implemented correctly on the Zybo board. Using fixed inputs a = “00001100” (12 in decimal) and b = “00001001” (9 in decimal), when the XOR operation is selected by turning switch 2 on and switches 1 and 0 off (“100”) as in Figure 16, LEDs 1 and 0 are on, representing “0101”.  In figure 17, when the XNOR operation is selected by turning on switch 0 and 2 and turning off switch 1 (“101”), LEDs 3 and 1 are on, representing “1010” (the opposite of Figure 16, as expected).