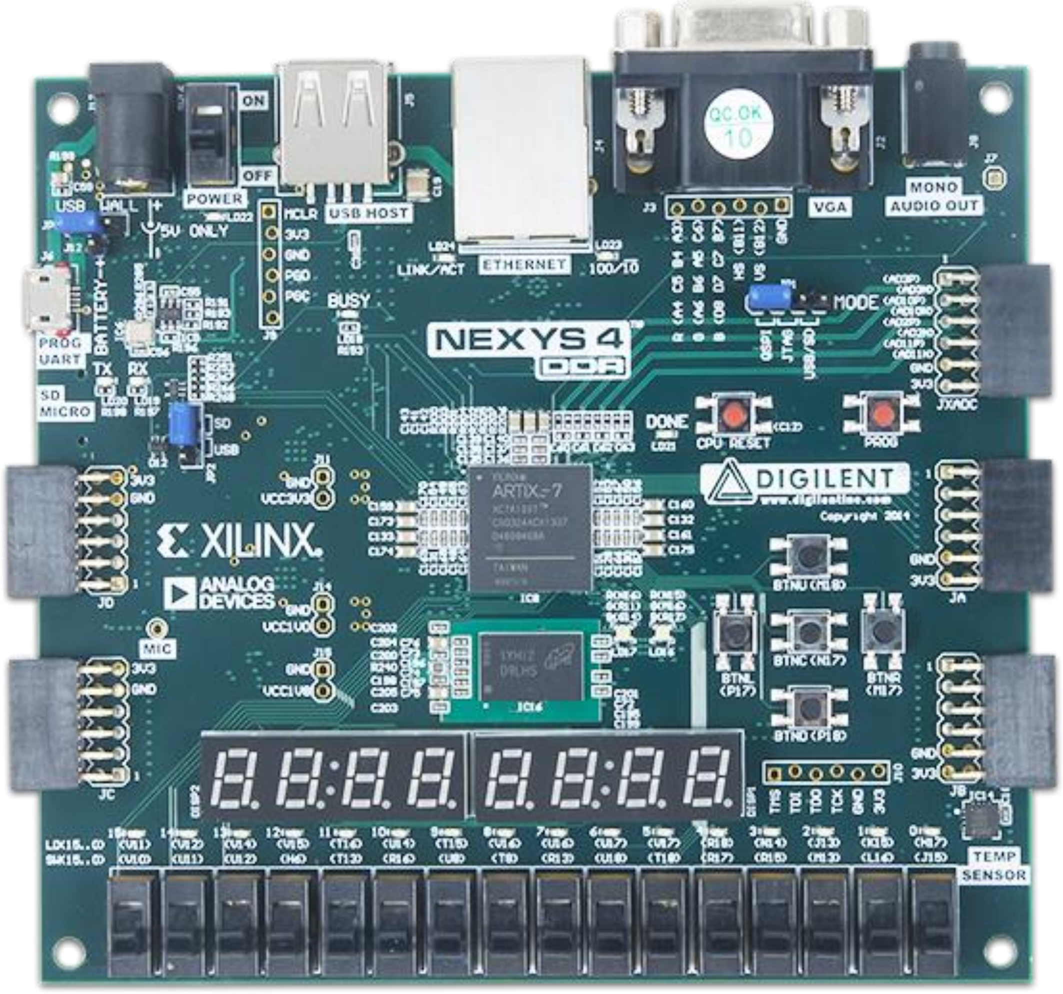
CMPS 375
Computer Architecture
Lab 3 (4-bit ALU)
Arithmetic Logic Unit (ALU)

Lab Objectives

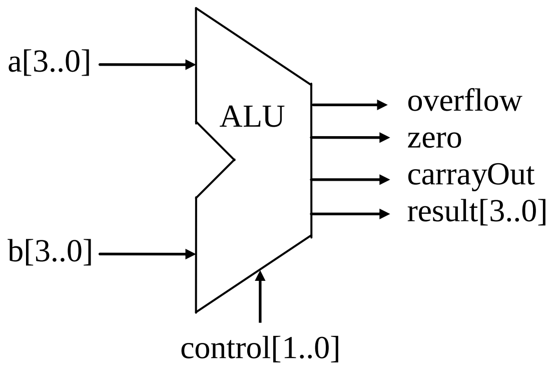
The purpose of this lab is to build combinational logic circuits in Arithmetic Logic Unit (ALU) running on the Nexys4 DDR board. You learn how to create ALU, a fundamental building block of the CPU of a computer.



Xilinx Nexys4 DDR Board: Artix-7 FPGA (XC7A100TCSG324-1)

Problem Description

You are required to create a 4-bit Arithmetic Logic Unit (ALU) in VHDL. ALU is a digital circuit that performs arithmetic and logical operations. The ALU logic block diagram and its operations shown as following:



ALU Block Diagram

ALU Control Lines and Operations

ALU Control	Operations
00	And
01	Or
10	Add
11	Subtract

The hierarchical design of the 4-bit ALU can be automatically imported from 1-bit adders and 7-segment displays in the previous labs. You need to demonstrate logical operations (AND and OR) and arithmetic operations (Add, Subtract). This 4-bit ALU uses slide switches (4-bit a, 4-bit b, and 2-bit control lines) as the inputs, LEDs (overflow, zero, carryOut) and a 7-segment display (4-bit result) as the outputs on the Nexys4 DDR board.

Lab Submission

- Zip the **entire** DispAlu4 project folder including its sub-folders into a single file name: **DispAlu4.zip**
- Check the following 8 files under DispAlu4 sub-folders before your submission
  1. **DispAlu4.vhd**: Display 4-bit Alu, design file
  2. **DispAlu4_tb.vhd**: Display 4-bit Alu, test bench
  3. **DispAlu4.xdc**: Display 4-bit Alu, constraints file
  4. **Alu4.vhd**: 4-bit Alu, design file
  5. **Adder4_tb.vhd**: 4-bit Alu, test bench
  6. **Alu1.vhd**: 1-bit Alu, design file
  7. **Bin2Hex.vhd**: 4-bit binary inputs to 7-segment hex display
  8. **Adder1.vhd**: 1-bit adder, design file
- Submit **DispAlu4.zip** to Moodle as your lab3.

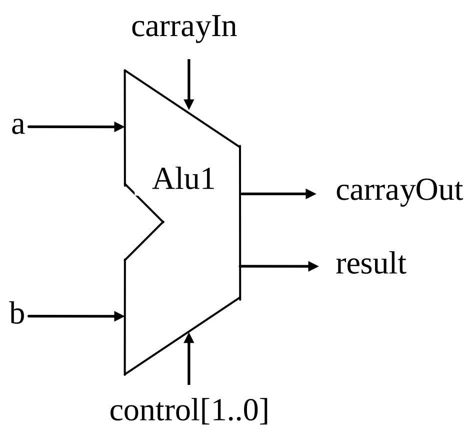
Lab Exercise

There are three steps to complete this lab exercise:

- Step 1: Create 1-bit ALU
- Step 2: Create 4-bit ALU
- Step 3: Display 4-bit ALU

Step 1: Create 1-bit ALU in VHDL Language

1. Create a project name, **DispAlu4**. Assign the device to Artix-7 FPGA **XC7A100TCSG324-1**
2. Copy **Adder1.vhd** from Lab1 into this project.
    - 1-bit ALU needs 1-bit adder (Adder1.vhd) from previous labs
    - In the *Project Manager* tasks of the *Flow Navigator* pane, click the *Add Sources*, select *Add or create design source*, then select **Adder1.vhd**.
3. 1-bit ALU block diagram and its operations as following:



1-bit ALU Block Diagram

ALU Control Lines and Operations

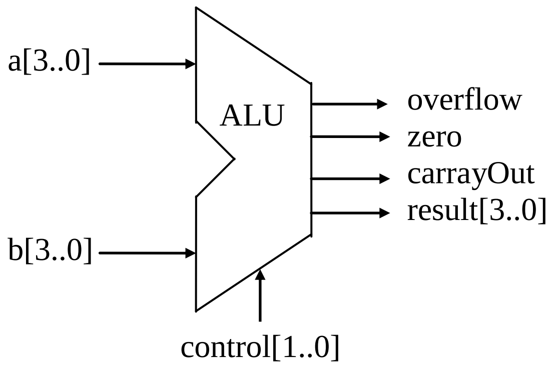
ALU Control	Operations
00	And
01	Or
10	Add
11	Subtract

4. Create **Alu1.vhd** and enter the following VHDL code.
   - In the *Project Manager* tasks of the *Flow Navigator* pane, click the *Add Sources*, select *Add or create design sources*, then *Create File*.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY Alu1 IS
    PORT(
            a, b:       IN STD_LOGIC;
            carryIn:    IN STD_LOGIC;
            control:    IN STD_LOGIC_VECTOR(1 DOWNTO 0);
            carryOut:   OUT STD_LOGIC;
            result:     OUT STD_LOGIC
     );
END Alu1;

ARCHITECTURE behavioral OF Alu1 IS
  COMPONENT Adder1
        PORT (
                a, b, cIn :     IN  std_logic;
                cOut, sum :     OUT std_logic);
  END COMPONENT;

  SIGNAL a_sig, b_sig: STD_LOGIC;
  SIGNAL sum_sig, carryOut_sig: STD_LOGIC;
  SIGNAL result_sig: STD_LOGIC;
BEGIN

  a_sig <= a;
  b_sig <= NOT b WHEN control = "11" ELSE    -- SUBTRACT (1's complement)
              b;

  co: Adder1 PORT MAP(a_sig, b_sig, carryIn, carryOut_sig, sum_sig);

  -- ADD and SUBTRACT operations do the same thing (ADD)
  WITH control SELECT
    result_sig <=
          a AND b    WHEN "00",  -- AND
          a OR  b    WHEN "01",  -- OR
          sum_sig     WHEN "10",  -- ADD
          sum_sig    WHEN "11",  -- SUBTRACT
          sum_sig    WHEN Others;

  result <= result_sig;

  -- carryOut = '0' when AND or OR operations
  carryOut  <= '0' WHEN control = "00" OR control = "01" ELSE
                   carryOut_sig;

END behavioral;
```

Step 2: Create 4-bit ALU in VHDL Language

1. 4-bit ALU needs 1-bit ALU (**Alu1.vhd**)
2. 4-bit ALU block diagram and its operations as following:



4-bit ALU Block Diagram

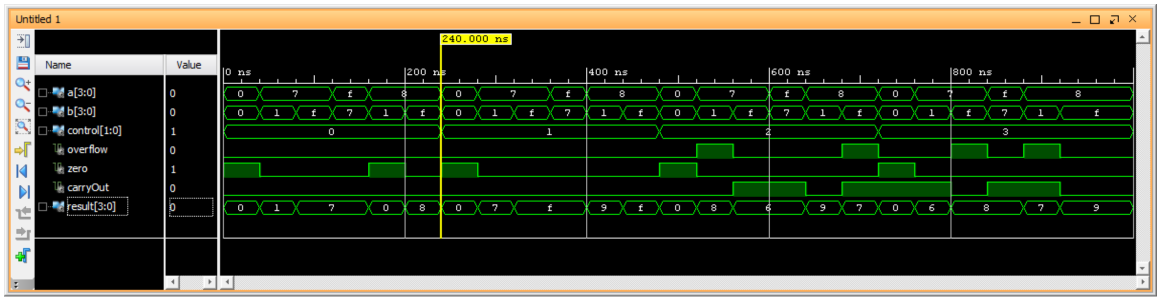
ALU Control Lines and Operations

ALU Control	Operations
00	And
01	Or
10	Add
11	Subtract

3. Create **Alu4.vhd** and complete the following VHDL code. Note: You may get help from “Generate Statements” in the VHDL Cookbook. You write VHDL like GENERATE … PORT MAP …

```
IBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY Alu4 IS
    GENERIC(CONSTANT N: INTEGER := 4;                                   -- 4 bits ALU
                CONSTANT Z: STD_LOGIC_VECTOR(3 DOWNTO 1) := "000"      -- 3 Zeros
        );
    PORT(
            a, b:        IN  STD_LOGIC_VECTOR(N-1 DOWNTO 0);
            control:     IN  STD_LOGIC_VECTOR(1 DOWNTO 0);
            overflow:  OUT STD_LOGIC;
            zero:        OUT STD_LOGIC;
            carryOut:  OUT STD_LOGIC;
            result:    OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0)
      );
END Alu4;

ARCHITECTURE behavioral OF Alu4 IS
    COMPONENT Alu1
        PORT(
            a, b:      IN STD_LOGIC;
            carryIn:  IN STD_LOGIC;
            control:  IN STD_LOGIC_VECTOR(1 DOWNTO 0);
            carryOut: OUT STD_LOGIC;
            result:  OUT STD_LOGIC
        );
    END COMPONENT;

    SIGNAL carry_sig:  STD_LOGIC_VECTOR(N DOWNTO 0);  -- carry_sig(N) = MSB carryOut
    SIGNAL result_sig: STD_LOGIC_VECTOR(N-1 DOWNTO 0);
 BEGIN

    -- Write Your Code Here

END behavioral;
```

4. Create a simulation file **Adder4_tb.vhd** if no compilation errors. Simulate your design: the results like this figure (Simulation Time: 1000 ns; wait for 40 ns).

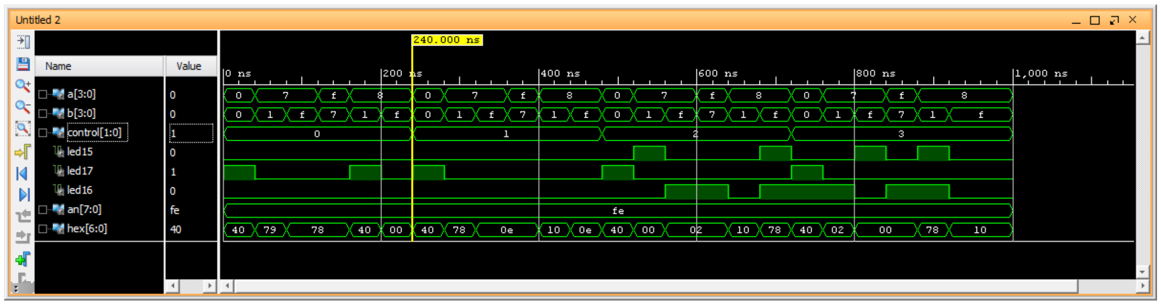


Step 3: Display 4-bit ALU on LEDs and 7-Segment Displays

1. Copy **Bin2Hex.vhd** from previous labs into this project.
   - Display 4-bit ALU needs a 7-segment display from previous lab.
   - In the *Project Manager* tasks of the *Flow Navigator* pane, click the *Add Sources*, select *Add or create design source*, then select **Bin2Hex.vhd**.
2. Create a VHDL file named **DispAlu4.vhd**. Enter and then complete the following VHDL code.
   - Note: You need "PORT MAP"

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY DispAlu4 IS
    port(
        a:          IN  STD_LOGIC_VECTOR(3 DOWNTO 0); -- Input     SW[7..4]: a[3..0]
        b:          IN  STD_LOGIC_VECTOR(3 DOWNTO 0); -- Input     SW[3..0]: b[3..0]
        control:    IN  STD_LOGIC_VECTOR(1 DOWNTO 0); -- Input     SW[15..14]: control[1..0]
        led15:      OUT STD_LOGIC;                      -- Output  LED[15]:  overflow
        led17:      OUT STD_LOGIC;                      -- Output  LED[17]:  zero
        led16:      OUT STD_LOGIC;                      -- Output  LED[16]:  carryOut
        an:         OUT STD_LOGIC_VECTOR(7 DOWNTO 0);   -- Output  AN[7..0]: '0' enabled
        hex:        OUT STD_LOGIC_VECTOR(6 DOWNTO 0)    -- Output  HEX[6..0]: result[3..0]
    );
END DispAlu4;

ARCHITECTURE behavioral OF DispAlu4 IS
    COMPONENT Alu4
        PORT(
                a, b:       IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
                control:    IN  STD_LOGIC_VECTOR(1 DOWNTO 0);
                overflow:   OUT STD_LOGIC;
                zero:       OUT STD_LOGIC;
                carryOut :  OUT STD_LOGIC;
                result:     OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
         );
    END COMPONENT;
    COMPONENT Bin2Hex
        PORT(
                bin:    IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
                hex:    OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
        );
    END COMPONENT;
    SIGNAL overflow_sig:   STD_LOGIC;
    SIGNAL zero_sig:       STD_LOGIC;
    SIGNAL carry_sig:      STD_LOGIC;
    SIGNAL result_sig:     STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN

     -- Write Your Code Here

END behavioral;
```

3. Create a simulation file **DispAlu4_tb.vhd** if no compilation errors. Simulate your design: the results like this figure (Simulation Time: 1000 ns; wait for 40 ns).

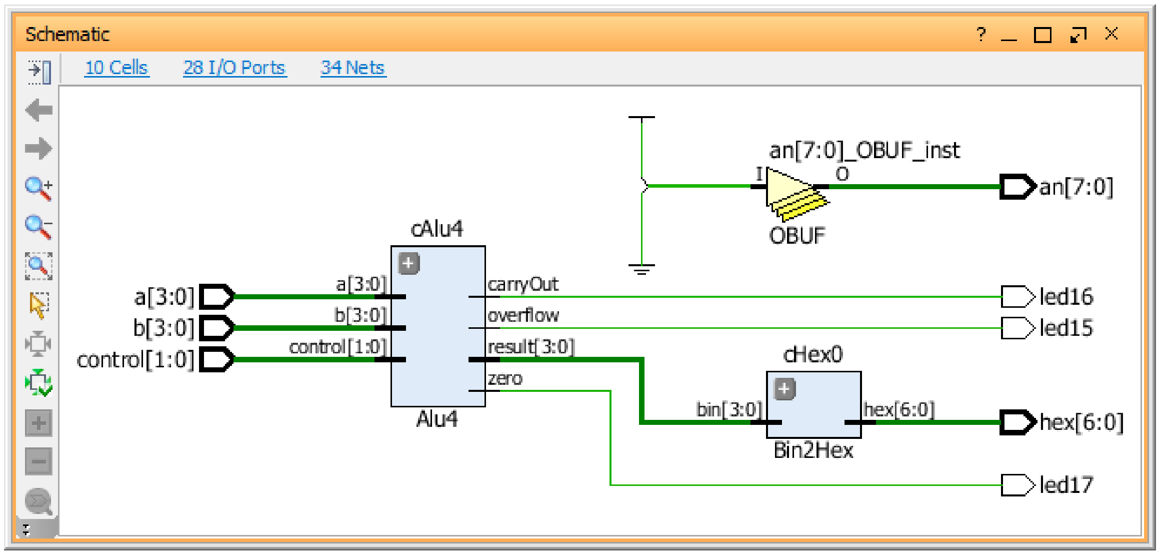


4. Copy Nexys4DDR_Master.xdc from previous labs into this project and rename to **DispAlu4.xdc**. Edited the port names and assign the FPGA I/O pins.

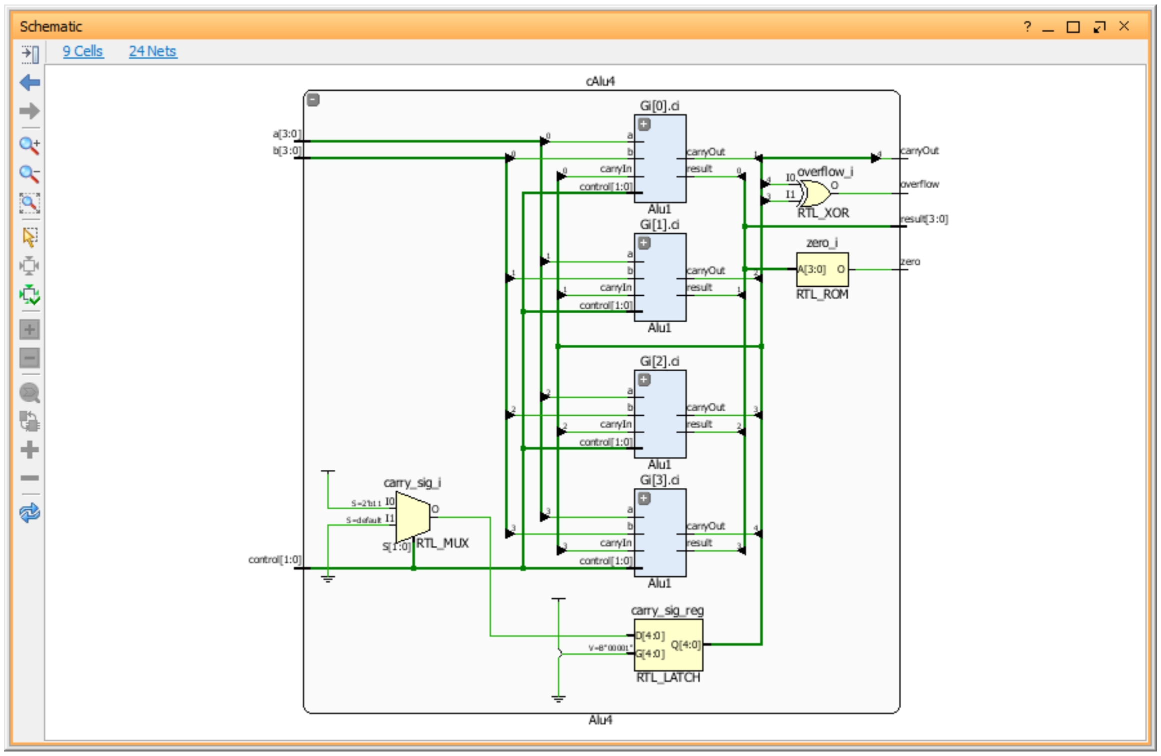
Scalar Ports	***Direction***	***Package Pin***	***Name***
b[0]	IN	J15	SW[0]
b[1]	IN	L16	SW[1]
b[2]	IN	M13	SW[2]
b[3]	IN	R15	SW[3]
a[0]	IN	R17	SW[4]
a[1]	IN	T18	SW[5]
a[2]	IN	U18	SW[6]
a[3]	IN	R13	SW[7]
control[0]	IN	U11	SW[14]
control[1]	IN	V10	SW[15]
led15	OUT	V11	LED15
led17	OUT	N16	LED17_R
led16	OUT	R12	LED16_B
an[0]	OUT	J17	AN[0]
an[1]	OUT	J18	AN[1]
an[2]	OUT	T9	AN[2]
an[3]	OUT	J14	AN[3]
an[4]	OUT	P14	AN[4]
an[5]	OUT	T14	AN[5]
an[6]	OUT	K2	AN[6]
an[7]	OUT	U13	AN[7]
hex[0]	OUT	T10	CA
hex[1]	OUT	R10	CB
hex[2]	OUT	K16	CC
hex[3]	OUT	K13	CD
hex[4]	OUT	P15	CE
hex[5]	OUT	T11	CF
hex[6]	OUT	L18	CG

5. Synthesize, implement the design, and then generate the bitstream.
6. Program and download the design to FPGA: choose the download file **DispAlu4.bit.**
7. Test and verify functionality of the designed circuit.

- You may get help from schematics: DispAlu4 and Alu4



DispAlu4 Schematic



Alu4 Schematic