4 BIT ARITHMETIC LOGIC UNIT (ALU)

19ECE383 VLSI Design Lab

B. Tech. ECE

Batch: C1

2023-2024 Even Semester

CB.EN.U4ECE21201	ACHANTA KIRAN SAI PAVAN
CB.EN.U4ECE21241	PEDDINENI VENKATA SAI RAJESH
CB.EN.U4ECE21260	KARTHIK VENKAT REDDY T

Faculty Incharges: Dr. Navya Mohan,Dr. Anita J. P.

Signature of the Faculty:

Department of Electronics and Communication Engineering

Amrita School of Engineering

Amrita Vishwa Vidyapeetham

Amritanagar, Coimbatore – 641112

1. **Objective:**

   The main objective of the project is to design and verify different operations of the Arithmetic and Logical Unit (ALU). We have designed a 4-bit ALU that accepts two 4-bit numbers and the code corresponding to the operation that it has to perform from the user. The ALU performs the desired operation and generates the result accordingly. The different operations that we dealt with were arithmetical and logical operations. Arithmetic operations include arithmetic addition, subtraction, and incrementation. Logical operations include AND, OR, XOR, and NOT. To implement ALU, the coding was written in VHDL. The waveforms were obtained successfully. After the coding was done, the synthesis of the code was performed using Basys3. Thereafter, the simulation was done to verify the synthesized code.

2. **Software hardware used:**
   - Modelsim
   - Vivado
   - Basys -3 FPGA Board

3. **Overview of the system:**

   The 4-bit ALU is a digital circuit designed to perform a variety of arithmetic and logical operations on 4-bit binary numbers. The system is constructed using VHDL and synthesized for implementation on a Basys-3 FPGA board. The primary components and their interactions are detailed as follows:

   Inputs and Outputs:
   - *Inputs:* A and B: Two 4-bit binary numbers on which the ALU operations are performed.
   - ALU_Sel: A 4-bit selector signal that determines which operation the ALU performs.
   - *Outputs:* ALU_Out: The 4-bit result of the ALU operation.
   - Carryout: A single-bit output indicating a carry out in arithmetic operations (specifically addition).

   Arithmetic Operations:
   - Addition (ALU_Sel = "0000"): Adds A and B.
   - Subtraction (ALU_Sel = "0001"): Subtracts B from A.
   - Multiplication (ALU_Sel = "0010"): Multiplies A by B.
   - Division (ALU_Sel = "0011"): Divides A by B.

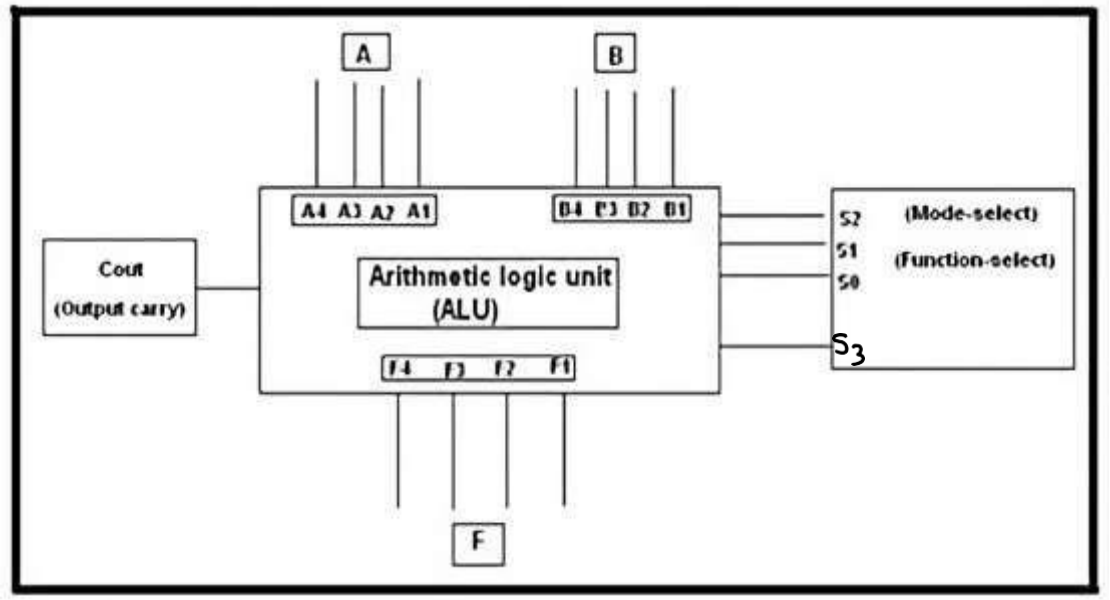
   Shift Operations:
   - Logical Shift Left (ALU_Sel = "0100"): Shifts A left by N bits.
   - Logical Shift Right (ALU_Sel = "0101"): Shifts A right by N bits.
   - Rotate Left (ALU_Sel = "0110"): Rotates A left by N bits.
   - Rotate Right (ALU_Sel = "0111"): Rotates A right by N bits.

   Logical Operations:
   - AND (ALU_Sel = "1000"): Performs bitwise AND on A and B.
   - OR (ALU_Sel = "1001"): Performs bitwise OR on A and B.
   - XOR (ALU_Sel = "1010"): Performs bitwise XOR on A and B.
   - NOR (ALU_Sel = "1011"): Performs bitwise NOR on A and B.
   - NAND (ALU_Sel = "1100"): Performs bitwise NAND on A and B.
   - XNOR (ALU_Sel = "1101"): Performs bitwise XNOR on A and B.

Comparison Operations:

- Greater Comparison (ALU_Sel = "1110"): Checks if A is greater than B.
- Equal Comparison (ALU_Sel = "1111"): Checks if A is equal to B.

4. Block diagram:



5. Introduction:

An Arithmetic Logic Unit (ALU) is a digital circuit that performs arithmetic and logical operations on binary numbers. It is a fundamental building block of the central processing unit (CPU) of a computer, along with the control unit and registers.

The ALU performs arithmetic operations such as addition, subtraction, multiplication, and division, as well as logical operations such as AND, OR, NOT, and XOR. It is responsible for carrying out the instructions of a computer program, manipulating data, and producing the results of those operations.

The ALU works by taking in two binary inputs, performing the specified operation, and outputting the result. It operates at high speeds, enabling the computer to perform calculations and logical operations quickly and efficiently.

Overall, the ALU is a critical component of a computer's architecture, and its design and performance can have a significant impact on the overall speed and efficiency of a computer
system.

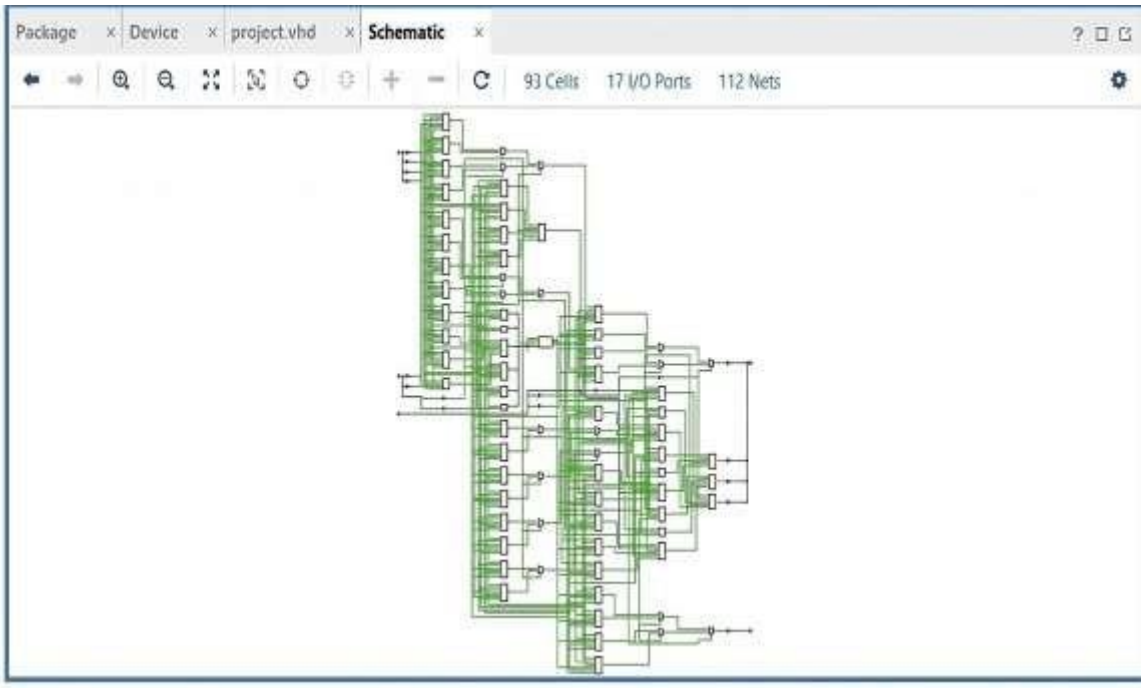
6. Complete Code with comments:

library IEEE;

```
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.NUMERIC_STD.all;

entity ALU is
generic (
constant N: natural := 1 -- number of shifted or rotated bits
);
Port (
A, B : in STD_LOGIC_VECTOR(3 downto 0); -- 2 inputs 4-bit
ALU_Sel : in STD_LOGIC_VECTOR(3 downto 0); -- 1 input 4-bit for selecting
function
ALU_Out : out STD_LOGIC_VECTOR(3 downto 0); -- 1 output 4-bit
Carryout : out std_logic
);
end ALU;

architecture Behavioral of ALU is
signal ALU_Result : std_logic_vector (3 downto 0);
signal tmp: std_logic_vector (4 downto 0);
begin
process(A, B, ALU_Sel)
begin
case ALU_Sel is
when "0000" => -- Addition
ALU_Result <= A + B;
when "0001" => -- Subtraction
ALU_Result <= A - B;
when "0010" => -- Multiplication
ALU_Result <= std_logic_vector(to_unsigned((to_integer(unsigned(A)) *
to_integer(unsigned(B))), 4));
when "0011" => -- Division
ALU_Result <= std_logic_vector(to_unsigned(to_integer(unsigned(A)) /
to_integer(unsigned(B)), 4));
when "0100" => -- Logical shift left
ALU_Result <= std_logic_vector(unsigned(A) sll N);
when "0101" => -- Logical shift right
ALU_Result <= std_logic_vector(unsigned(A) srl N);
when "0110" => -- Rotate left
ALU_Result <= std_logic_vector(unsigned(A) rol N);
when "0111" => -- Rotate right
ALU_Result <= std_logic_vector(unsigned(A) ror N);
when "1000" => -- Logical and
ALU_Result <= A and B;
when "1001" => -- Logical or
ALU_Result <= A or B;
when "1010" => -- Logical xor
ALU_Result <= A xor B;
when "1011" => -- Logical nor
ALU_Result <= A nor B;
when "1100" => -- Logical nand
```

```
ALU_Result <= A nand B;
when "1101" => -- Logical xnor
ALU_Result <= A xnor B;
when "1110" => -- Greater comparison
if (A > B) then
ALU_Result <= "0001"; -- 1 in 4-bit
else
ALU_Result <= "0000"; -- 0 in 4-bit
end if;
when "1111" => -- Equal comparison
if (A = B) then
ALU_Result <= "0001"; -- 1 in 4-bit
else
ALU_Result <= "0000"; -- 0 in 4-bit
end if;
when others =>
ALU_Result <= A + B;
end case;
end process;

ALU_Out <= ALU_Result; -- ALU out
tmp <= ('0' & A) + ('0' & B);
Carryout <= tmp(4); -- Carryout flag
end Behavioral;
```

7. **RTL:**



8. Resource Utilization summary table:

Select Line	Operation
0000	Addition
0001	Subtraction
0010	Multiplication
0011	Division
0100	Logical shift left
0101	Logical shift right
0110	Rotate left
0111	Rotate Right
1000	AND Gate
1001	OR Gate
1010	XOR Gate
1011	NOR Gate
1100	NAND Gate
1101	XNOR Gate
1110	Greater Comparsion
1111	Equal Comparsion

9. Resource Utilization summary table:

10. Output Waveform: