Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

16-BIT ALU PROJECT

In this document a 16- bit Arithmetic Logical Unit (ALU) with internal components of arithmetic, logic and shifting will be modeled using VHDL. The ALU will consist of two 16-bit data inputs labeled A and B, a 1-bit Mode input for selecting different operation groups, and a 3-bit input Opcode that will determine which operation each component will be output. The outputs of the ALU are a 16-bit data output and a carry out for arithmetic operations labeled ALU Out and Cout respectively.

For the arithmetic unit, shifting unit and logical unit there is a defined number of operations for each unit. Arithmetic operations for the arithmetic unit are multiplication of two 8-bit numbers, addition of two 16-bit numbers, subtraction of two 16-bit numbers, and an increment of one 16-bit number. When overflow occurs, the carry will output a one otherwise the carry is zero.

The operations for the shifting unit include a shift logical left/right and a rotate left/right. When shifting 16-bit values new bits are replaced with zeros with left shift adding zeros to the least significant bit while shifting to the right replaces the most significant bit with zeros. When rotating left/right new values rotating in are replaced with the values shifted out. For rotating left the most significant bit will rotate to the least significant bit. When rotating to the right will rotate the least significant bit to the most significant bit. This rotation effectively wraps around the bits of a vector.

Lastly, the logical unit carries out the following operations when selected A NOR B, A NAND B, A OR B, A AND B, A XOR B, A XNOR B, NOT A, and NOT B. the behavior of these operations is beyond the scope of this document and therefore will not be described in detail.

The following table will illustrate the manner in which each operation may be selected using the mode and opcode inputs.

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

Mode	Opcode	Operation
0	000	A NOR B
0	001	A NAND B
0	010	A OR B
0	011	A AND B
0	100	A XOR B
0	101	A XNOR B
0	110	NOT A
0	111	NOT B
1	000	A*B
1	001	A+B
1	010	A-B
1	011	Increment A(A+1)
1	100	Shift Left (A,B)
1	101	Shift Right(A,B)
1	110	Rotate Left (A,B)
1	111	Rotate Right (A,B)

Table 1: table of operations and their associated control values

Student Name: Ruben Ramirez

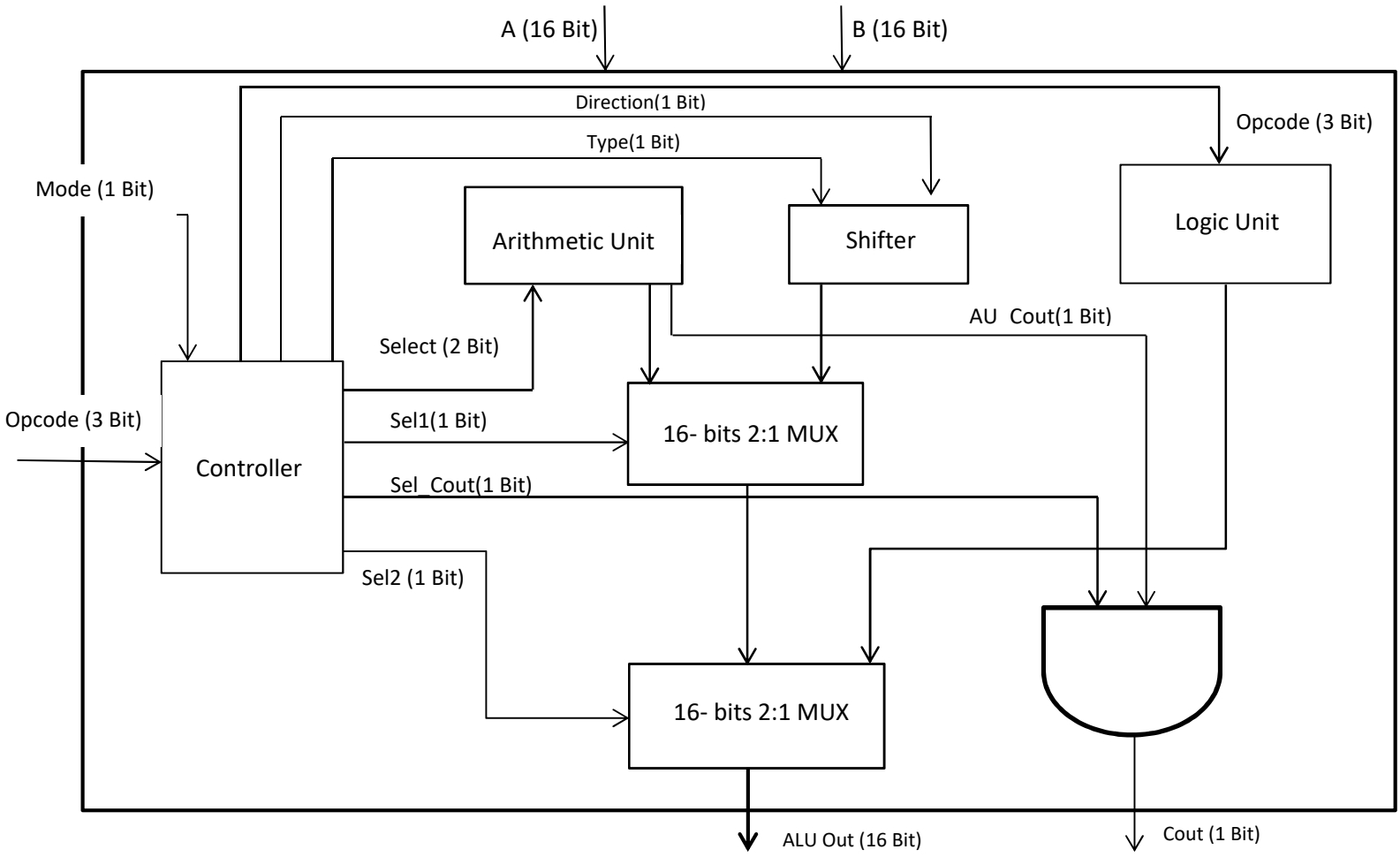
Student Id.: 2694

16-Bit ALU Design

A block diagram of the ALU to be modeled follows;

16-Bit ALU Design

1. Design the 16-bit Arithmetic Logic Unit as shown in the bock diagram below.



Student Name: Ruben Ramirez

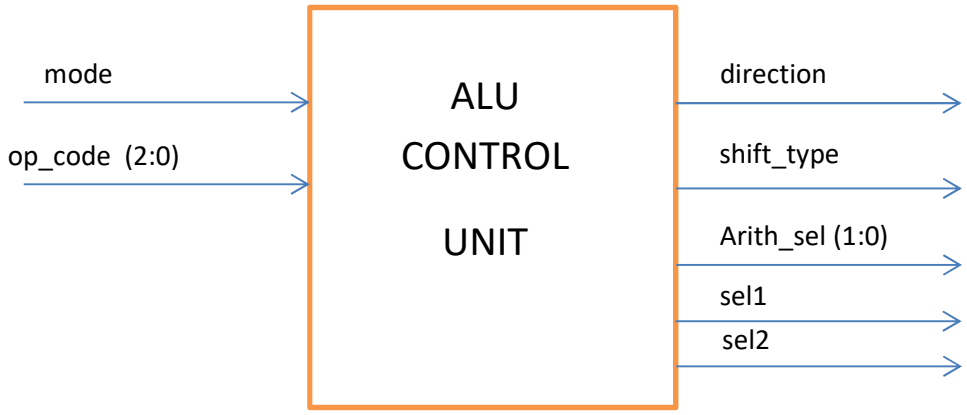
Student Id.: 2694

16-Bit ALU Design

A. ALU Controller

The ALU Control Unit controls the data path for ALU and has inputs mode, and op_code. Mode controls whether the ALU is using the Arithmetic/Shifter unit or the logical unit. When mode is set to one the data path is directed to the Logic unit. When is mode is set to zero the data path is directed to the Arithmetic/Shifter unit. The op_code is used to determine which operation types within respective units will execute. This is shown in the table 1.

1. Block diagram



Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

2. VHDL Code

```
----------------------------------------------------------------------------------
-- Ruben Ramirez
-- 2694
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;



entity control_unit is
    Port (
           Arith_sel : out STD_LOGIC_VECTOR (1 downto 0);
           sel1 : out STD_LOGIC;
           sel2 : out STD_LOGIC;
           sel_cout : out STD_LOGIC;
           direction : out STD_LOGIC;
           shift_type : out STD_LOGIC;
           mode : in STD_LOGIC;
           op_code : in STD_LOGIC_VECTOR (2 downto 0));
end control_unit;

architecture Behavioral of control_unit is

begin




-- pass for direction in shifter unit
direction <= op_code(1);

-- pass for op type in shifter unit
shift_type <= op_code(0);

-- passes selection operation type in arithmetic unit
Arith_sel <= op_code(1 downto 0);

-- determine whether arithmetic unit or shifter unit will output to next mux
sel1 <= op_code(2);
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
-- determine whether output of first mux or logic unit will be output
sel2 <= mode;


cout : process(mode,op_code)

begin
sel_cout <= '0';

-- determine whether carry out from arithmetic unit is output
-- set to 1 if mode = 1 and opcode(2) = 0, else set to 0

if ( mode = '1') and (op_code(2) = '0') then

    sel_cout <= '1';

end if;
end process;


end Behavioral;
```

Test Bench

```
----------------------------------------------------------------------------
--------
-- Ruben Ramirez
-- 2694
----------------------------------------------------------------------------
--------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity control_unit_tb is
 -- Port (
   --      logic_out : out STD_LOGIC_VECTOR (2 downto 0) := (others => '0')
);
```

```
end control_unit_tb;


architecture Behavioral of control_unit_tb is

component control_unit is
    Port (
           Arith_sel : out STD_LOGIC_VECTOR (1 downto 0);
           sel1 : out STD_LOGIC ;
           sel2 : out STD_LOGIC;
           sel_cout : out STD_LOGIC;
           direction : out STD_LOGIC;
           shift_type : out STD_LOGIC;
           mode : in STD_LOGIC;
           op_code : in STD_LOGIC_VECTOR (2 downto 0));
end component;
------------------ signals for output ----------------------
signal     Arith_sel :  STD_LOGIC_VECTOR (1 downto 0);
signal     sel1 :  STD_LOGIC;
signal     sel2 :  STD_LOGIC;
signal     sel_cout :  STD_LOGIC;
signal     direction :  STD_LOGIC;
signal     shift_type :  STD_LOGIC;

------------------- signals for inputs ---------------------------
signal     mode :  STD_LOGIC :='1';
signal     op_code :  STD_LOGIC_VECTOR (2 downto 0) :=(others=>'0');




begin

uut : control_unit port map(

                Arith_sel =>Arith_sel,
                sel1 =>sel1,
                sel2 =>sel2,
                sel_cout => sel_cout,
                direction => direction,
                shift_type => shift_type,
                mode =>mode,
                op_code => op_code);

stim_proc: process
begin
--  hold reset state for 10 ns.
  wait for 10 ns;
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
----------------------test for logical unit ---------------

----------------- DESCRIPTION --------------------------
--  generate signal for logical unit and select which
--  operation will be output for unit
--  mode will remain zero for the following test cases
-------------------------------------------------------
         -- LOGIC UNIT OUTPUTS

         -- TEST CASE: 0
               -- LOGIC UNIT A NOR B SELECT

               mode <= '0';
               op_code <= "000";

               wait for 10 ns;

         -- TEST CASE: 1
      LOGIC UNIT A NAND B SELECT

               mode <= '0';
               op_code <= "001";

               wait for 10 ns;
         -- TEST CASE: 2
               -- LOGIC UNIT A OR B SELECT

               mode <= '0';
               op_code <= "010";

               wait for 10 ns;

         -- TEST CASE: 3
               -- LOGIC UNIT A AND B SELECT

               mode <= '0';
               op_code <= "011";

               wait for 10 ns;

         -- TEST CASE: 4
               -- LOGIC UNIT A XOR B SELECT

               mode <= '0';
               op_code <= "100";

               wait for 10 ns;

         -- TEST CASE: 5
               -- LOGIC UNIT A XNOR B SELECT
```

```
                mode <= '0';
                op_code <= "101";

                wait for 10 ns;

        -- TEST CASE: 6
                -- LOGIC UNIT NOT A  SELECT

                mode <= '0';
                op_code <= "110";

                wait for 10 ns;

        -- TEST CASE: 7
                -- LOGIC UNIT NOT B SELECT

                mode <= '0';
                op_code <= "111";

                wait for 10 ns;

--------------------------------- end of logic unit test ------------------
---------------

----------------------test for arithmetic unit ---------------

----------------- DESCRIPTION --------------------------
--  generate signal for arithmetic unit and select which
--  operation will be output for unit
--  mode will remain one for the following test cases
-------------------------------------------------------

                -- ARITHMETIC UNIT OUTPUTS
        -- TEST CASE: 0
                -- ARITHMETIC UNIT A*B

                mode <= '1';
                op_code <= "000";

                wait for 10 ns;

        -- TEST CASE: 1
                -- ARITHMETIC UNIT A+B

                mode <= '1';
                op_code <= "001";

                wait for 10 ns;

        -- TEST CASE: 2
                -- ARITHMETIC UNIT A-B
```

```
                mode <= '1';
                op_code <= "010";

                wait for 10 ns;

        -- TEST CASE: 3
                -- ARITHMETIC UNIT INCREMENT A

                mode <= '1';
                op_code <= "011";

                wait for 10 ns;

---------------------------------- end of arithmetic unit test ------------
-----------------



----------------------test for shifter unit ---------------

----------------- DESCRIPTION --------------------------
--  generate signal for shifter unit and select which
--  operation will be output for unit
--  mode will remain one for the following test cases
-------------------------------------------------------


                -- SHIFTER UNIT OUTPUTS

                -- TEST CASE: 0
                        -- SHIFTER UNIT SHIFT LEFT

                mode <= '1';
                op_code <= "100";

                wait for 10 ns;

        -- TEST CASE: 1
                -- SHIFTER UNIT SHIFT RIGHT

                mode <= '1';
                op_code <= "101";

                wait for 10 ns;

        -- TEST CASE: 2
                -- SHIFTER UNIT ROTATE LEFT

                mode <= '1';
                op_code <= "110";

                wait for 10 ns;
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
            -- TEST CASE: 3
                -- SHIFTER UNIT ROTATE RIGHT

                mode <= '1';
                op_code <= "111";

                wait for 10 ns;

    -------------------------------- end of shifter unit test ---------------
    --------------




    wait;
    end process;
    end Behavioral;
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

1. Waveform

Comments:

The test bench displays the sixteen possible signal controls of alu controller unit. This will determine which function will be chosen from the arithmetic, logic, or shifter unit.

- 10 to 90ns: generate control signals to logical unit
- 90 to 130ns: generate control signals to arithmetic unit
- 130 to 170ns: generate control signals to shifter unit

Student Name: Ruben Ramirez

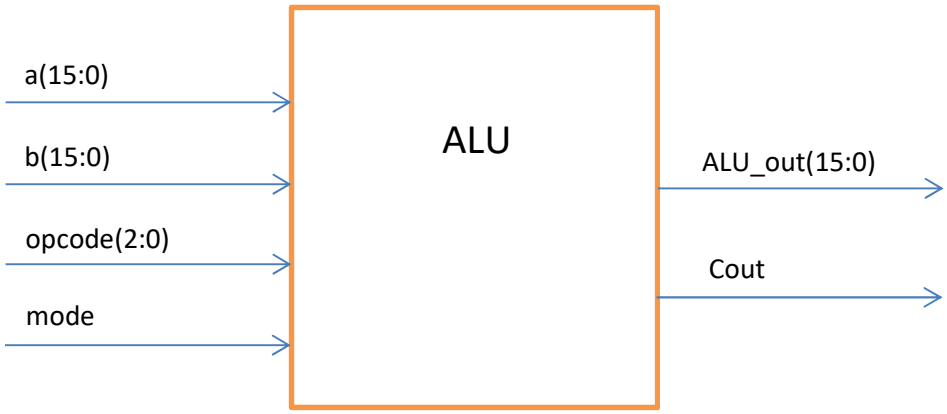
Student Id.: 2694

16-Bit ALU Design

B. ALU

The ALU as described in the opening paragraphs is shown below in a block diagram. With each input and output labeled.

i. Block diagram



ii. VHDL Code

```
--------------------------------------------------------------------------------
-----
-- Ruben Ramirez
-- 2694
--------------------------------------------------------------------------------
-----



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


--use IEEE.NUMERIC_STD.ALL;


entity ALU is
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
  Port (A,B : in std_logic_vector(15 downto 0);
        Opcode : in std_logic_vector(2 downto 0);
        Mode : in std_logic;
        ALUout : out std_logic_vector(15 downto 0);
        Cout : out std_logic
             );
end ALU;



architecture Behavioral of ALU is

-------- alu controller -----------

component control_unit is
    Port (
           Arith_sel : out STD_LOGIC_VECTOR (1 downto 0);
           sel1 : out STD_LOGIC;
           sel2 : out STD_LOGIC;
           sel_cout : out STD_LOGIC;
           direction : out STD_LOGIC;
           shift_type : out STD_LOGIC;
           mode : in STD_LOGIC;
           op_code : in STD_LOGIC_VECTOR (2 downto 0));
end component;

-- shifter unit
component ShiftUnit16Bits is
port(A, B : IN std_logic_vector(15 downto 0):= (others => '0');
  Direction, Op_Type   : IN std_logic := '0';
  ShiftOut   : OUT std_logic_vector(15 downto 0));
end component;

-- arithmetic unit
component ArithmeticUnit16bit is
    Port(A : IN  std_logic_vector(15 downto 0); -- Input
         B : IN  std_logic_vector(15 downto 0); -- Input
         Op_Sel: IN std_logic_vector(1 downto 0);--operation selection
         ArithOut: OUT std_logic_vector(15 downto 0); -- Result
         Cout    : OUT std_logic);  --carry out bit of operation

end component;

-- logic unit
component LogicUnit16bit is
    Port ( a : in STD_LOGIC_VECTOR (15 downto 0);
           b : in STD_LOGIC_VECTOR (15 downto 0);
           Opcode : in STD_LOGIC_VECTOR (2 downto 0);
           logicout : out STD_LOGIC_VECTOR (15 downto 0));
end component;

-- 2-1 16 bit mux
component mux_2_1 is
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
    Port ( a : in STD_LOGIC_VECTOR (15 downto 0); -- this will be low input
for mux
           b : in STD_LOGIC_VECTOR (15 downto 0); -- this will be high input
for mux
           y : out STD_LOGIC_VECTOR (15 downto 0);   -- ouput of mux
dependent on sel
           sel: in STD_LOGIC);      -- input which will determine which input
a or b will pass to output
end component;

------------------------- and gate -------------------------------------
component and_gate is
    port ( A : in std_logic;
           B: in std_logic;
           and_out: out std_logic);
end component;

------------------------------------------------------------------------------
------------------------- output signals --------------------------------
signal muxA_B:std_logic_vector(15 downto 0);
signal shift_out:STD_LOGIC_VECTOR (15 downto 0);
signal logic_out:std_logic_vector(15 downto 0);
signal arith_out:std_logic_vector(15 downto 0);
signal sel_cout: std_logic;
signal arith_select:std_logic_vector(1 downto 0);
signal direction:std_logic;
signal shift_type:std_logic;
signal sel1:std_logic;
signal sel2 :STD_LOGIC;
signal au_Cout:std_logic;
signal and_Cout:std_logic;
signal alu_out:std_logic_vector(15 downto 0);
------------------------------------------------------------------------

begin



----------------- controller port mapping -------------------------
Controller_Comp: control_unit port map(
           Arith_sel => arith_select,
           sel1 => sel1,
           sel2 => sel2,
           sel_cout => sel_cout,
           direction => direction,
           shift_type => shift_type,
           mode => Mode ,
           op_code => Opcode);

--------------------- arithmetic port mapping -------------------
arith_comp: ArithmeticUnit16bit Port map(
         A => A,
         B => B,
```

```
         Op_Sel =>arith_select,
         ArithOut => arith_out,
         Cout =>au_Cout);
---------------------- logic port mapping ----------------------
logic_comp: LogicUnit16bit port map(
           a => A ,
           b => B,
           Opcode => Opcode,
           logicout => logic_out);

---------------------- shifter port mapping ----------------------
shift_comp: ShiftUnit16Bits port map (
           A => A,
           B => B,
           Direction => direction,
           Op_Type => shift_type,
           ShiftOut => shift_out);

------------------------ mux from arithmetic/shifter port mapping -----------
-----------
mux_comp_arith_shift: mux_2_1 Port map ( a => arith_out,
           b => shift_out, -- this will be high input for mux
           y => muxA_B,   -- ouput of mux dependent on sel
           sel =>sel1);

------------------------ mux from logic port mapping ----------------------
mux_comp_out: mux_2_1 Port map ( a => logic_out,
           b => muxA_B, -- this will be high input for mux
           y => ALUout,   -- ouput of mux dependent on sel
           sel =>sel2);

------------------------ and gate port mapping --------------------
and_gate_comp: and_gate Port map (
          A => au_Cout,
          B => sel_cout,
          and_out => Cout);



end Behavioral;
```

iii. Test Bench

```
--------------------------------------------------------------------------------
-----
-- Ruben Ramirez
-- 2694
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
--------------------------------------------------------------------------------
-----

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

USE ieee.numeric_std.ALL;

ENTITY ALU_tb IS
END ALU_tb;

ARCHITECTURE behavior OF ALU_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

component ALU
  Port (A,B : in std_logic_vector(15 downto 0);
        Opcode : in std_logic_vector(2 downto 0);
        Mode : in std_logic;
        ALUout : out std_logic_vector(15 downto 0);
        Cout : out std_logic
             );
end component;


------------- inputs ---------------------
signal A,B: std_logic_vector (15 downto 0);
signal Opcode : std_logic_vector(2 downto 0);
signal Mode :  std_logic;
---------- out puts ----------------------
signal ALUout :  std_logic_vector(15 downto 0);
signal Cout :  std_logic;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
   uut: ALU PORT MAP (
          Mode => Mode,
          Opcode => Opcode,
          A => A,
          B => B,
          ALUout => ALUout,
          Cout => Cout
        );


   stim_proc: process
   begin

     -- test of arithmetic unit

     A <= x"0003";
     B <= x"0002";
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
Opcode <= "000";
mode <= '1';
wait for 10 ns;
Opcode <= "001";
wait for 10 ns;
Opcode <= "010";
wait for 10 ns;
Opcode <= "011";
A <= x"ffff";
wait for 10 ns;
-- test of logic unit
Opcode <= "000";
mode <= '0';
wait for 10 ns;
Opcode <= "001";
A <= x"0007";

wait for 10 ns;
Opcode <= "010";
wait for 10 ns;
Opcode <= "011";
wait for 10 ns;
Opcode <= "100";
wait for 10 ns;
Opcode <= "101";
wait for 10 ns;
Opcode <= "110";
```

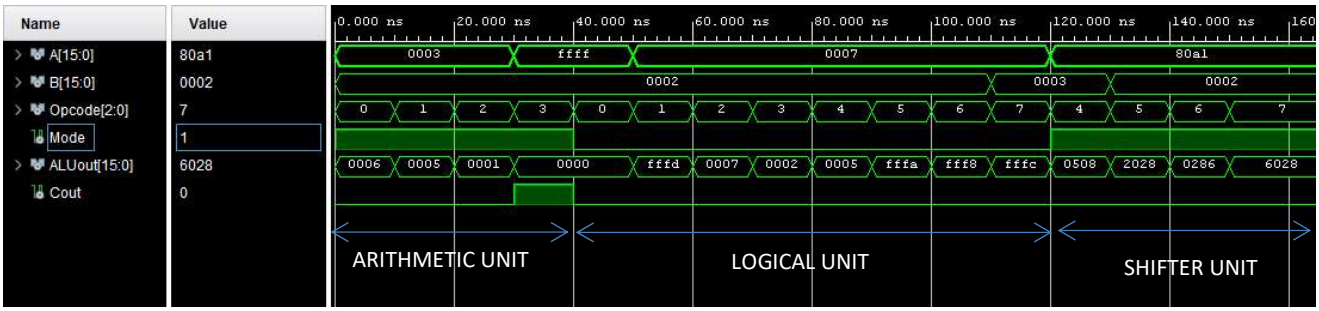
```
      wait for 10 ns;

      B <= x"0003";

      Opcode <= "111";


      wait for 10 ns;
-- test of shifter unit
-----------------------------------------------------

      A <= x"80A1";

      mode <= '1';

      Opcode <= "100";


      wait for 10 ns;
-----------------------------------------------------
      Opcode <= "101";

      B <= x"0002";

      wait for 10 ns;
-----------------------------------------------------
      Opcode <= "110";


      wait for 10 ns;
-----------------------------------------------------
      Opcode <= "111";

      wait for 10 ns;


       wait;
   end process;

END;
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

iv. Waveform



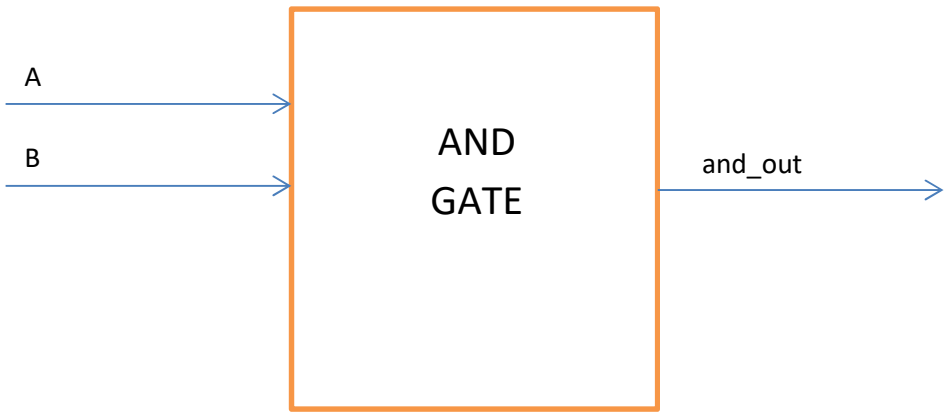
Comments:

The test bench of ALU displays the three main functions of the design: Logical, Shift, and Arithmetic. Each unit within the ALU has a specified set of functions that is determined by the inputs, mode and opcode of the ALU.

- 0 to 40ns: Arithmetic unit and the four operations of the unit
- 40 to 120ns: Logical unit and the eight operations of the unit
- 120 to 160ns: Shifter unit and the four operations of the unit

C. And gate

v. Block diagram



vi. VHDL Code

```
--------------------------------------------------------------------------------
-----
-- Ruben Ramirez
-- 2694
--------------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity and_gate is
  Port (A : in std_logic := '0';
        B : in std_logic := '0';
        and_out : out std_logic );
end and_gate;

architecture Behavioral of and_gate is



begin

and_out <= A and B;
```

```
end Behavioral;
```

vii. TEST BENCH

```
----------------------------------------------------------------------------------
-----
-- Ruben Ramirez
-- 2694
----------------------------------------------------------------------------------
-----



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity and_gate_tb is
--  Port ( );
end and_gate_tb;

architecture Behavioral of and_gate_tb is

component and_gate
  Port (A : in std_logic := '0';
        B : in std_logic := '0';
        and_out : out std_logic  );
end component;

signal A,B :std_logic;
signal and_out: std_logic;


begin

----------------- port map of component and gate ----------------------
uut: and_gate port map(A,B,and_out);

stimulus : process
begin

wait for 10 ns; -- reset period
```

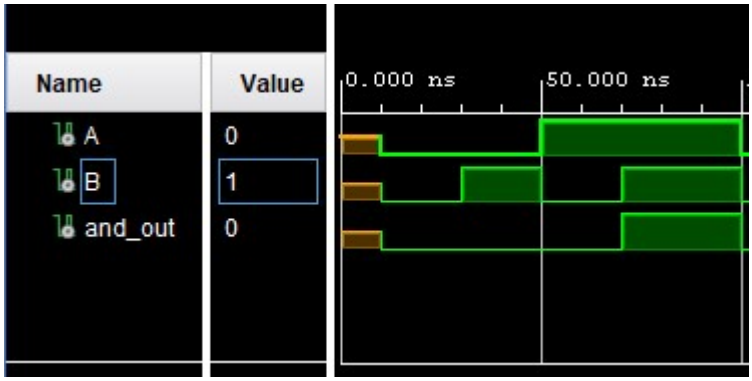
Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
----------------------- test of and gate ---------------------------

----------------- description ----------------------------------------
--  test of  an and gate which will logical and two inputs
--  and then output a 0 or 1 dependent of and gate behavior
-----------------------------------------------------------------------

------------------------ test case 0 ---------------------------------
-- A = 0, B = 0, output should be 0 -----------------------------------

    A <= '0';
    B <= '0';

    wait for 20 ns;
---------------------- end test 0 -----------------------------------

------------------------ test case 1 ---------------------------------
-- A = 0, B = 1, output should be 0 -----------------------------------

    A <= '0';
    B <= '1';

    wait for 20 ns;
---------------------- end test 1 -----------------------------------


------------------------ test case 2 ---------------------------------
-- A = 1, B = 0, output should be 0 -----------------------------------

    A <= '1';
    B <= '0';

    wait for 20 ns;
---------------------- end test 2 -----------------------------------

------------------------ test case 3 ---------------------------------
-- A = 1, B = 1, output should be 1 -----------------------------------

    A <= '1';
    B <= '1';

    wait for 20 ns;
---------------------- end test 3 -----------------------------------



end process;
end Behavioral;
```

viii. Waveform



Comments:

The testbench of the and gate provides the proper model for an and gate circuit. The model will output a one when both A and B inputs are one. Otherwise the output will be a zero.

- 10 to 30ns: inputs are 0 output is 0
- 30 to 50ns: inputs are A = 0, B = 1, output is 0
- 50 to 70ns: inputs are A = 1, B = 0, output is 0
- 70 to 90ns: inputs are A = 1, B = 1, output is 1

VHDL Code for Components

All other components used in the modeling of the ALU which are all described in previous documents.

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
----------------------------------------------------------------------------
-----
-- Ruben Ramirez          ALU CONTROL COMPONENT
-- 2694
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;



entity control_unit is
    Port (
           Arith_sel : out STD_LOGIC_VECTOR (1 downto 0);
           sel1 : out STD_LOGIC;
           sel2 : out STD_LOGIC;
           sel_cout : out STD_LOGIC;
           direction : out STD_LOGIC;
           shift_type : out STD_LOGIC;
           mode : in STD_LOGIC;
           op_code : in STD_LOGIC_VECTOR (2 downto 0));
end control_unit;

architecture Behavioral of control_unit is

begin




-- pass for direction in shifter unit
direction <= op_code(0);

-- pass for op type in shifter unit
shift_type <= op_code(1);

-- passes selection operation type in arithmetic unit
Arith_sel <= op_code(1 downto 0);

-- determine whether arithmetic unit or shifter unit will output to next mux
sel1 <= op_code(2);

-- determine whether output of first mux or logic unit will be output
sel2 <= mode;


cout : process(mode,op_code)
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
begin
sel_cout <= '0';

-- determine whether carry out from arithmetic unit is output
-- set to 1 if mode = 1 and Opcode(2) = 0, else set to 0

if ( mode = '1') and (op_code(2) = '0') then

    sel_cout <= '1';

end if;
end process;


end Behavioral;

----------------------------------------------------------------------------------
-----
-- Ruben Ramirez              ARITHMETIC COMPONENT
-- 2694
----------------------------------------------------------------------------------
-----


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

entity ArithmeticUnit16bit is
    Port(A : IN  std_logic_vector(15 downto 0); -- Input
         B : IN  std_logic_vector(15 downto 0); -- Input
         Op_Sel: IN std_logic_vector(1 downto 0);--operation selection
         ArithOut: OUT std_logic_vector(15 downto 0); -- Result
         Cout    : OUT std_logic); --carry out bit of operation
end ArithmeticUnit16bit;

architecture Behavioral of ArithmeticUnit16bit is


begin



process(A,B,Op_Sel)

variable Val_A:  integer;
variable Val_B: integer;
variable calc : unsigned(15 downto 0);
variable Mult_A: integer;
variable Mult_B: integer;
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
    begin
       Val_A :=  to_integer(unsigned(a));
       Val_B := to_integer(unsigned(b));
       Mult_A :=to_integer(unsigned(A(7 downto 0)));
       Mult_B :=to_integer(unsigned(b(7 downto 0)));


       case Op_Sel is
          when "00" => calc  := to_unsigned((Mult_A*Mult_B),16);      --
multiply A*B
          when "01" => calc  := to_unsigned((Val_A+Val_B),16);        -- add
A+B
          when "10" => calc  :=to_unsigned((Val_A)-(Val_B),16);         --
subtract A-B
          when "11" => calc  := to_unsigned(Val_A+1,16);              --
increment A = A + 1
          when others =>  calc  := to_unsigned(Val_A+1,16);

        end case;

        if Op_Sel ="11" AND A = x"ffff" then
            ArithOut <= x"0000";
            Cout <= '1';
        else
            cout <= '0';
        end if;

        ArithOut <= std_logic_vector(calc);

    end process;

end Behavioral;

----------------------------------------------------------------------------
-----
-- Ruben Ramirez            LOGIC COMPONENT
-- 2694
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;



entity LogicUnit16bit is
    Port ( a : in STD_LOGIC_VECTOR (15 downto 0);
           b : in STD_LOGIC_VECTOR (15 downto 0);
           Opcode : in STD_LOGIC_VECTOR (2 downto 0);
           logicout : out STD_LOGIC_VECTOR (15 downto 0));
end LogicUnit16bit;
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
architecture Behavioral of LogicUnit16bit is

begin
--                  the select statement will determine which logic operation
will be output
    with Opcode select
        logicout <= a NOR b when "000",
                    a NAND b when "001",
                    a OR b when "010",
                    a AND b when "011",
                    a XOR b when "100",
                    a XNOR b when "101",
                    NOT a when "110",
                    NOT b when "111",
                    a when others;


end Behavioral;

----------------------------------------------------------------------------
-----
-- Ruben Ramirez            SHIFT COMPONENT
-- 2694
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;


entity ShiftUnit16Bits is
port(A, B : IN std_logic_vector(15 downto 0):= (others => '0');
  Direction, Op_Type   : IN std_logic := '0';
  ShiftOut    : OUT std_logic_vector(15 downto 0):= (others => '0'));
end ShiftUnit16Bits;

architecture Behavioral of ShiftUnit16Bits is

begin

    process(A,B,Direction,Op_Type)

    variable result :std_logic_vector(15 downto 0):=(others => '0');
    variable shift_choice: std_logic_vector(1 downto 0) ;
    variable move_bits: integer;

    begin

      move_bits := to_integer(unsigned(B));
      shift_choice(1)  := Op_Type;
```

Student Name: Ruben Ramirez

Student Id.: 2694

16-Bit ALU Design

```
      shift_choice(0)    := Direction;
        case shift_choice is

            when "00" =>
               -- shift left operation
               if move_bits <= 15 then
                 result(15 downto move_bits) := A(15-move_bits downto 0);
                 result(move_bits-1 downto 0) := (others => '0');
                 ShiftOut <= result;
               else
                  ShiftOut <= x"0000";
                end if;

             when "01" =>
               -- shift right operation
                if move_bits <= 15 then
                   result(15-move_bits downto 0)   :=  A(15 downto move_bits);
                   result(15 downto 16 - move_bits) := (others => '0');

                   ShiftOut <= result;
                else
                  ShiftOut <= x"0000";
                end if;

              when "10" =>
                   -- rotate left
                if move_bits <= 15 then
                  result(15 downto move_bits) := A(15-move_bits downto 0);

                  result(move_bits-1 downto 0)   :=  A(15 downto 16-
move_bits);

                    ShiftOut <= result;
                else
                   ShiftOut <= A;
                end if;

              when "11" =>
                   -- rotate right
                   if move_bits <= 15 then

                   result(15-move_bits downto 0)   :=  A(15 downto move_bits);

                   result(15 downto 16- move_bits)   :=  A(move_bits-1 downto 0
);

                    ShiftOut <= result;
                else
                   ShiftOut <= A;
                end if;
              when others => ShiftOut <= x"ffff";

              end case;
```

```
    end process;

end Behavioral;


----------------------------------------------------------------------------
-----
--                           2 TO 1 MUX COMPONENT
--
----------------------------------------------------------------------------
-----


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity mux_2_1 is
    Port ( a : in STD_LOGIC_VECTOR (15 downto 0); -- this will be low input
for mux
           b : in STD_LOGIC_VECTOR (15 downto 0); -- this will be high input
for mux
           y : out STD_LOGIC_VECTOR (15 downto 0);   -- ouput of mux
dependent on sel
           sel: in STD_LOGIC);      -- input which will determine which input
a or b will pass to output
end mux_2_1;

architecture Behavioral of mux_2_1 is

begin

process(a,b,sel)

    begin

    if sel = '0' then y <= a;  -- sel input set low output gets a

    else y <= b;                -- -- sel input set high output gets b

    end if;
    end process;

end Behavioral;
```