ENGG*3380

Dr. Abou El Nasr

February 27th, 2023

Lab 4: Arithmetic Logic Unit (ALU) Design

Group 6

Yasir Al-Obaidi (1145544)

Pratham Patel (1140832)

Problem Statement

In this lab, we were to design a generic Arithmetic Logic Unit circuit and verify the design through simulations and testbenches.

Assumptions and Constraints

For the higher bit ALUs, we were restricted to using structural design concepts and a 1-bit ALU as our foundational building block.

System Overview

Part 1

For part 1 of the lab, we had to make a 1 bit ALU based on the full adder we built in lab 1. This ALU required addition, subtraction and Logical AND and OR functionalities. Using this 1-bit ALU, we had to then design a 16-bit ALU and write a testbench for it to ensure that it functions without any issues. Below are the codes for the 1-bit and 16-bit ALUs.

ALU 1 -bit Code

```
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 OneBitALU is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           Cin : in STD_LOGIC;
           S : in STD_LOGIC_VECTOR (1 downto 0);
           Cout : out STD_LOGIC;
           Sout : out STD_LOGIC);
end OneBitALU;

architecture Behavioral of OneBitALU is

begin


end Behavioral;
```

ALU 16-bit Code

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;

Entity ALU_16Bit is
port(
        A       :   in      std_logic_vector(15 downto 0);
        B       :   in      std_logic_vector(15 downto 0);
        Cin     :   in      std_logic;
        S       :   in      std_logic_vector(1 downto 0);
        Sout    :   out     std_logic_vector(15 downto 0);
        Cout    :   out std_logic
);
End;

Architecture behavior of ALU_16Bit is
    COMPONENT ALU
    port(
        A       :   in      std_logic;
        B       :   in      std_logic;
        Cin :   in      std_logic;
        S       :   in      std_logic_vector(1 downto 0);
        Sout    :   out     std_logic;
        Cout    :   out std_logic
    );
    END COMPONENT;

    signal Carry        :   std_logic_vector(14 downto 0);

Begin
    alu00   :   ALU port map(A(0),      B(0),       S(0),           S,  Sout(0),        Carry(0));
    alu01   :   ALU port map(A(1),      B(1),       Carry(0),   S,  Sout(1),        Carry(1));
    alu02   :   ALU port map(A(2),      B(2),       Carry(1),   S,  Sout(2),        Carry(2));
    alu03   :   ALU port map(A(3),      B(3),       Carry(2),   S,  Sout(3),        Carry(3));

    alu04   :   ALU port map(A(4),      B(4),       Carry(3),   S,  Sout(4),        Carry(4));
    alu05   :   ALU port map(A(5),      B(5),       Carry(4),   S,  Sout(5),        Carry(5));
    alu06   :   ALU port map(A(6),      B(6),       Carry(5),   S,  Sout(6),        Carry(6));
    alu07   :   ALU port map(A(7),      B(7),       Carry(6),   S,  Sout(7),        Carry(7));

    alu08   :   ALU port map(A(8),      B(8),       Carry(7),   S,  Sout(8),        Carry(8));
    alu09   :   ALU port map(A(9),      B(9),       Carry(8),   S,  Sout(9),        Carry(9));
    alu10   :   ALU port map(A(10), B(10),  Carry(9),   S,  Sout(10),   Carry(10));
    alu11   :   ALU port map(A(11), B(11),  Carry(10),  S,  Sout(11),   Carry(11));

    alu12   :   ALU port map(A(12), B(12),  Carry(11),  S,  Sout(12),   Carry(12));
    alu13   :   ALU port map(A(13), B(13),  Carry(12),  S,  Sout(13),   Carry(13));
    alu14   :   ALU port map(A(14), B(14),  Carry(13),  S,  Sout(14),   Carry(14));
    alu15   :   ALU port map(A(15), B(15),  Carry(14),  S,  Sout(15),   Cout);
End;
```

Part 2

For the second part of the lab, we had to write a 8-bit ALU that had 8 different operations. The code for the ALU is provided below. This time it had to be behavioral and not structural.

8Bit ALU Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library UNISIM;
use UNISIM.VComponents.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity ALU_16Bit_Part2 is
    generic (Dwidth : integer := 8);
    port ( IN1,IN2  : in STD_LOGIC_VECTOR (Dwidth-1 downto 0);
           ALU_OUT : out STD_LOGIC_VECTOR (Dwidth-1 downto 0);
           SEL     : in STD_LOGIC_VECTOR (2 downto 0);
           Cin     : in STD_LOGIC;
           Zero, OVF   : out STD_LOGIC);
end ALU_16Bit_Part2;
architecture Behavioral of ALU_16Bit_Part2 is
    signal add_result : unsigned(8 downto 0);
    signal and_result : std_logic_vector(7 downto 0);
    signal or_result : std_logic_vector(7 downto 0);
    signal xor_result : std_logic_vector(7 downto 0);
    signal slt_result : std_logic;
    signal beq_result : std_logic;
begin
process (IN1,IN2, SEL, Cin)
begin
    add_result <= unsigned(in1) + unsigned(in2);
    and_result <= in1 and in2;
    or_result  <= in1 or in2;
    xor_result <= in1 xor in2;
    if (in1 < in2) then
        slt_result <= '1';
    else
        slt_result <= '0';
    end if;
    if (in1 = in2) then
        beq_result <= '1';
    else
        beq_result <= '0';
    end if;
        case SEL is
            when "000" =>
                ALU_OUT <= std_logic_vector(add_result(7 downto 0));
                zero <= '0';
                ovf <= '1' when add_result(8) /= in1(7) and add_result(8) /= in2(7) else '0';
            when "001" =>
                ALU_OUT <= std_logic_vector(add_result(7 downto 0));
                zero <= '0';
                ovf <= '1' when add_result(8) /= in1(7) and add_result(8) /= in2(7) else '0';
            when "010" =>
                ALU_OUT <= and_result;
                zero <= '1' when and_result = "00000000" else '0';
                ovf <= '0';
            when "011" =>
                ALU_OUT <= or_result;
                zero <= '1' when or_result = "00000000" else '0';
                ovf <= '0';
            when "100" =>
                ALU_OUT <= xor_result;
                zero <= '1' when xor_result = "00000000" else '0';
                ovf <= '0';
            when "101" =>
                ALU_OUT <= (others => '0');
                zero <= '1' when slt_result = '1' else '0';
                ovf <= '0';
            when "110" =>
                ALU_OUT <= (others => '0');
                zero <= '1' when beq_result = '1' else '0';
                ovf <= '0';
            when others =>
                ALU_OUT <= (others => '0');
                zero <= '0';
                ovf <= '0';
        end case;
    end process;
end architecture;
```

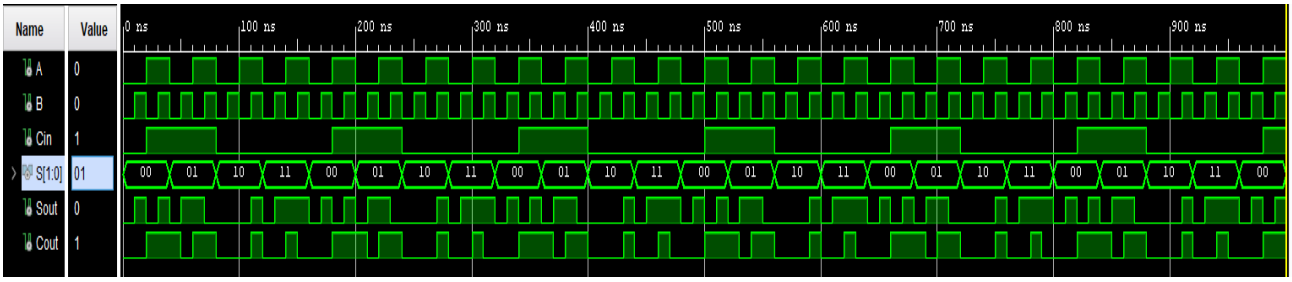
Here the different operations are calculated in the cases of SEL. The SEL input is the 3-bit selector that dictates which of the 8 operations is chosen. The overflow and zero variables are also calculated in the cases where they are relevant.

Verification

Part 1

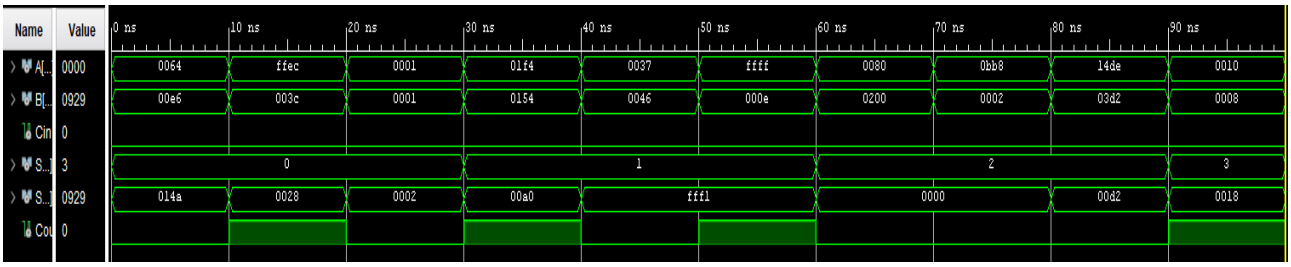
To verify the functionality of part 1, we wrote a testbench that would contain test cases for each of the operations of the ALU. The code for this testbench is provided in appendix A. The wave form and table of results is shown below.

Waveform for 1-Bit ALU



Here you can see the inputs and outputs of various cases in a 1-bit ALU.

Waveform for 16-Bit ALU



Here you can see the inputs and outputs of various cases in a 16-bit ALU. The cases that were used are shown in more detail below. The table below showcases the inputs (S1, S0, A, and B) and outputs (Sout, Cout).

Results Table from 16-Bit ALU

S1	S0	A	B	Sout	Cout
0	0	100	230	330	0
0	0	-20	60	40	1
0	0	1	1	2	0
0	1	500	340	160	1
0	1	55	70	-15	0
0	1	-1	14	-15	1
1	0	128	512	0	0
1	0	3000	2	0	0
1	0	5342	978	210	0
1	1	16	8	24	1
1	1	0	2345	2345	1
1	1	-1	5	-6	1

Part 2

For part 2, we wanted to test the program and all of the operations that were a part of the ALU. The code for the testbench is provided below.

ALU Testbench

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity ALU_16BitPart2_tb is
--  Port ( );
end ALU_16BitPart2_tb;

architecture Behavioral of ALU_16BitPart2_tb is
    component ALU_16Bit_Part2 is
    generic (Dwidth : integer := 8);
    port ( IN1,IN2  : in STD_LOGIC_VECTOR (Dwidth-1 downto 0);
           ALU_OUT : out STD_LOGIC_VECTOR (Dwidth-1 downto 0);
           SEL     : in STD_LOGIC_VECTOR (2 downto 0);
           Cin     : in STD_LOGIC;
           Zero, OVF   : out STD_LOGIC);
end component;

    signal IN1, IN2 : STD_LOGIC_VECTOR (7 downto 0) := x"00";
    signal SEL  : std_logic_vector (2 downto 0);
    signal Cin  : std_logic := '0';

    signal ALU_OUT : STD_LOGIC_VECTOR (7 downto 0) := x"00";
    signal Zero, OVF    : std_logic;
begin
    uut: ALU_16Bit_Part2 port map (
        IN1 => IN1,
        IN2 => IN2,
        Cin => Cin,
        SEL => SEL,
        ALU_OUT => ALU_OUT,
        Zero => Zero,
        OVF => OVF
        );

    stim_proc: process
    begin
        IN1     <=  x"10";  -- 100
        IN2     <=  x"08";  -- 230
        Cin <=  '0';
        SEL     <=  "000";      -- Add
        wait for 10 ns;

        IN1     <=  x"00";  -- 100
        IN2     <=  x"29";  -- 230
        Cin <=  '0';
        SEL     <=  "000";      -- Add
        wait for 10 ns;

        IN1     <=  x"00";  -- 100
        IN2     <=  x"29";  -- 230
        Cin <=  '0';
        SEL     <=  "001";      -- Add
        wait for 10 ns;

        IN1     <=  x"FF";  -- 100
        IN2     <=  x"05";  -- 230
        Cin <=  '0';
        SEL     <=  "001";      -- Add
        wait for 10 ns;

        IN1     <=  x"64";  -- 100
        IN2     <=  x"E6";  -- 230
        Cin <=  '0';
        SEL     <=  "010";      -- Add
        wait for 10 ns;
```

```
IN1     <=  x"EC";  -- 100
IN2     <=  x"3C";  -- 230
Cin <=  '0';
SEL     <=  "010";      -- Add
wait for 10 ns;

IN1     <=  x"00";  -- 100
IN2     <=  x"29";  -- 230
Cin <=  '0';
SEL     <=  "011";      -- Add
wait for 10 ns;

IN1     <=  x"00";  -- 100
IN2     <=  x"29";  -- 230
Cin <=  '0';
SEL     <=  "011";      -- Add
wait for 10 ns;

IN1     <=  x"64";  -- 100
IN2     <=  x"E6";  -- 230
Cin <=  '0';
SEL     <=  "100";      -- Add
wait for 10 ns;

IN1     <=  x"EC";  -- 100
IN2     <=  x"3C";  -- 230
Cin <=  '0';
SEL     <=  "100";      -- Add
wait for 10 ns;

IN1     <=  x"F4";  -- 100
IN2     <=  x"54";  -- 230
Cin <=  '0';
SEL     <=  "101";      -- Add
wait for 10 ns;

IN1     <=  x"FF";  -- 100
IN2     <=  x"0E";  -- 230
Cin <=  '0';
SEL     <=  "101";      -- Add
wait for 10 ns;

IN1     <=  x"00";  -- 100
IN2     <=  x"29";  -- 230
Cin <=  '0';
SEL     <=  "110";      -- Add
wait for 10 ns;

IN1     <=  x"29";  -- 100
IN2     <=  x"00";  -- 230
Cin <=  '0';
SEL     <=  "110";      -- Add
wait for 10 ns;

IN1     <=  x"29";  -- 100
IN2     <=  x"29";  -- 230
Cin <=  '0';
SEL     <=  "111";      -- Add
wait for 10 ns;

IN1     <=  x"29";  -- 100
IN2     <=  x"00";  -- 230
Cin <=  '0';
SEL     <=  "111";      -- Add
wait for 10 ns;
```

Errors and Issues

For Part 1, we did not have any issues. For part 2, however, the group could not figure out the overflows and were struggling to get a simulation to show. The group did not manage to figure it out by the due date. Due to this our demonstration was not fully complete and we lost 0.5 marks on this.

Summary

The lab overall was good, just the end of part 2 was difficult. The group should have allocated more time to this issue and found a way to fix it. However, this is just a learning opportunity for future labs.

Appendix A – Part 1 Testbench

```
        LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;


ENTITY ALU_16Bit_test IS
END ALU_16Bit_test;

ARCHITECTURE behavior OF ALU_16Bit_test IS

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


   COMPONENT ALU_16Bit
      port(
         A       :    in       std_logic_vector(15 downto 0);
         B       :    in       std_logic_vector(15 downto 0);
         Cin     :    in       std_logic;
         S       :    in       std_logic_vector(1 downto 0);
         Sout    :    out    std_logic_vector(15 downto 0);
         Cout    :    out    std_logic
```

```
        );
    END COMPONENT;


   --Inputs
   signal A     : std_logic_vector(15 downto 0) := x"0000";
   signal B     : std_logic_vector(15 downto 0) := x"0000";
   signal Cin     : std_logic := '0';
   signal S       : std_logic_vector(1 downto 0);

    --Outputs
   signal Sout    : std_logic_vector(15 downto 0);
   signal Cout    : std_logic;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
   uut: ALU_16Bit PORT MAP (
          A       =>    A,
          B       =>    B,
          Cin    =>    Cin,
          S       => S,
          Sout    =>    Sout,
```

```
        Cout    =>    Cout
    );



-- Stimulus process
stim_proc: process
begin
    A       <=    x"0064";    -- 100
    B       <=    x"00E6";    -- 230
    Cin     <=    '0';
    S       <=    "00";       -- Add
    wait for 10 ns;

    A       <=    x"FFEC";    -- 100
    B       <=    x"003C";    -- 230
    Cin     <=    '0';
    S       <=    "00";       -- Add
    wait for 10 ns;

    A       <=    x"0001";    -- 100
    B       <=    x"0001";    -- 230
    Cin     <=    '0';
    S       <=    "00";       -- Add
```

```
wait for 10 ns;

A       <=    x"01F4";    -- 100
B       <=    x"0154";    -- 230
Cin     <=    '0';
S       <=    "01";       -- Add
wait for 10 ns;

A       <=    x"0037";    -- 100
B       <=    x"0046";    -- 230
Cin     <=    '0';
S       <=    "01";       -- Add
wait for 10 ns;

A       <=    x"FFFF";    -- 100
B       <=    x"000E";    -- 230
Cin     <=    '0';
S       <=    "01";       -- Add
wait for 10 ns;

A       <=    x"0080";    -- 100
B       <=    x"0200";    -- 230
Cin     <=    '0';
```

```
S       <=    "10";       -- Add
wait for 10 ns;

A       <=    x"0BB8";    -- 100
B       <=    x"0002";    -- 230
Cin     <=    '0';
S       <=    "10";       -- Add
wait for 10 ns;

A       <=    x"14DE";    -- 100
B       <=    x"03D2";    -- 230
Cin     <=    '0';
S       <=    "10";       -- Add
wait for 10 ns;

A       <=    x"0010";    -- 100
B       <=    x"0008";    -- 230
Cin     <=    '0';
S       <=    "11";       -- Add
wait for 10 ns;

A       <=    x"0000";    -- 100
B       <=    x"0929";    -- 230
```

```
        Cin    <=    '0';
        S      <=    "11";       -- Add
        wait for 10 ns;

        A      <=    x"FFFF";    -- 100
        B      <=    x"0005";    -- 230
        Cin    <=    '0';
        S      <=    "11";       -- Add
        wait for 10 ns;

        wait;
    end process;

END;
```