Contents

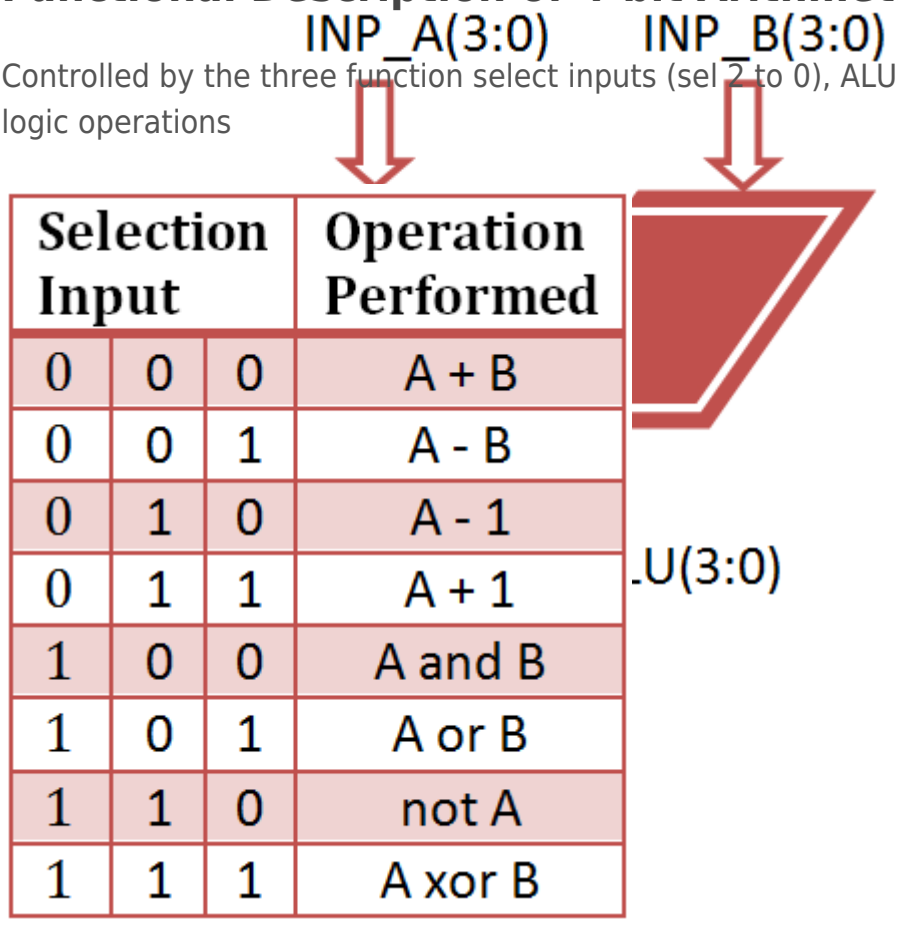
ALU

ALU's comprise the combinational logic that implements logic operations such as AND, OR, NOT gate and arithmetic operations, such as Adder, Subtractor.

Functionally, the operation of typical ALU is represented as shown in diagram below,

Functional Description of 4-bit Arithmetic Logic Unit

Controlled by the three function select inputs (sel 2 to 0), ALU can perform all the 8 possible logic operations



Selection Input			Operation Performed
0	0	0	A + B
0	0	1	A - B
0	1	0	A - 1
0	1	1	A + 1
1	0	0	A and B
1	0	1	A or B
1	1	0	not A
1	1	1	A xor B

VHDL Code for 4-bit ALU

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

entity alu is
 Port ( inp_a : in signed(3 downto 0);
 inp_b : in signed(3 downto 0);
 sel : in STD_LOGIC_VECTOR (2 downto 0);
 out_alu : out signed(3 downto 0));
end alu;

architecture Behavioral of alu is
begin
process(inp_a, inp_b, sel)
begin
case sel is
 when "000" =>
 out_alu<= inp_a + inp_b; --addition
 when "001" =>
 out_alu<= inp_a - inp_b; --subtraction
 when "010" =>
 out_alu<= inp_a - 1; --sub 1
 when "011" =>
 out_alu<= inp_a + 1; --add 1
 when "100" =>
 out_alu<= inp_a and inp_b; --AND gate
 when "101" =>
 out_alu<= inp_a or inp_b; --OR gate
 when "110" =>
 out_alu<= not inp_a ; --NOT gate
 when "111" =>
```

```
 out_alu<= inp_a xor inp_b; --XOR gate
 when others =>
 NULL;
end case;

end process;

end Behavioral;
```

Testbench VHDL Code for 4-Bit ALU

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

ENTITY Tb_alu IS
END Tb_alu;

ARCHITECTURE behavior OF Tb_alu IS

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

 COMPONENT alu
 PORT(
 inp_a : IN signed(3 downto 0);
 inp_b : IN signed(3 downto 0);
 sel : IN std_logic_vector(2 downto 0);
 out_alu : OUT signed(3 downto 0)
 );
 END COMPONENT;
```

```
 --Inputs
 signal inp_a : signed(3 downto 0) := (others => '0');
 signal inp_b : signed(3 downto 0) := (others => '0');
 signal sel : std_logic_vector(2 downto 0) := (others => '0');

 --Outputs
 signal out_alu : signed(3 downto 0);

BEGIN

 -- Instantiate the Unit Under Test (UUT)
 uut: alu PORT MAP (
 inp_a => inp_a,
 inp_b => inp_b,
 sel => sel,
 out_alu => out_alu
 );

 -- Stimulus process
 stim_proc: process
 begin
 -- hold reset state for 100 ns.
 wait for 100 ns;

 -- insert stimulus here

 inp_a <= "1001";
 inp_b <= "1111";

 sel <= "000";
 wait for 100 ns;
 sel <= "001";
 wait for 100 ns;
 sel <= "010";
```

```
   wait for 100 ns;
   sel <= "011";
   wait for 100 ns;
   sel <= "100";
   wait for 100 ns;
   sel <= "101";
   wait for 100 ns;
   sel <= "110";
   wait for 100 ns;
   sel <= "111";
   end process;

  END;
```

Simulation Result for 4-bit ALU