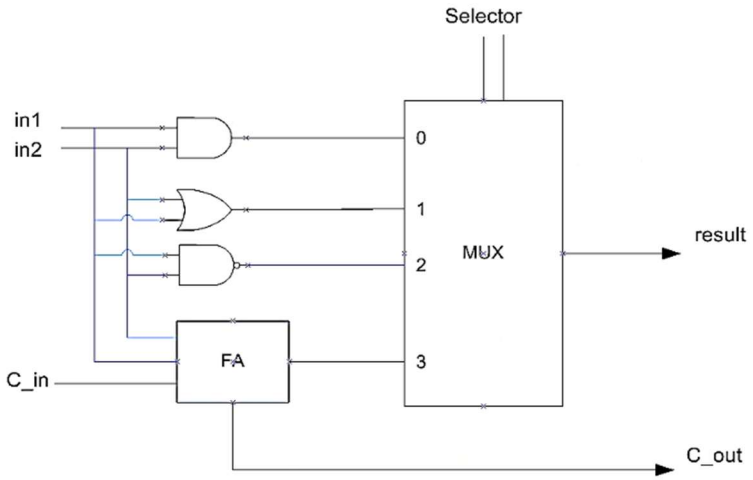
Lab Task_01

ALU implementation of a single bit ALU shown in design 1



Design 1: A simple ALU which performs Logical (AND,OR,NAND) and Arithmetic (ADDER)

ENTITIES:

1:OR gate

```
library ieee;
use ieee.std_logic_1164.all;

entity orGate is
    port (
        A,B : in std_logic;
        X : out std_logic
    );
end orGate;

architecture bhv of orGate is
begin
    X <= A OR B ;
end bhv;
```

1:AND gate

```
library ieee;
use ieee.std_logic_1164.all;

entity andGate is
    port (
        A,B : in std_logic;
        X : out std_logic
    );
end andGate;
```

```
architecture bhv of andGate is
begin
    X <= A AND B ;
end bhv;
```

2:NAND gate

```
library ieee;
use ieee.std_logic_1164.all;

entity nandGate is
    port (
        A,B : in std_logic;
        X : out std_logic
    );
end nandGate;

architecture bhv of nandGate is
begin
    X <= NOT ( A AND B );
end bhv;
```

3:Full Adder

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

entity fullAdder is
    port(
        A, B, Cin : in STD_LOGIC;
        S, C : out STD_LOGIC
    );
end fullAdder;

architecture bhv of fullAdder is
begin
    S <= (A XOR B) XOR Cin;
    C <= (A AND B) OR (Cin AND (A XOR B));
end bhv;
```

4:MUX

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

entity muxx is
    port(
        D0,D1,D2,D3 : in STD_LOGIC;
          S: in STD_LOGIC_VECTOR(1 downto 0);
        X : out STD_LOGIC
    );
end muxx;

architecture bhv of muxx is
begin
```

```
    X <= D0 WHEN S = "00" ELSE
          D1 WHEN S = "01" ELSE
          D2 WHEN S = "10" ELSE
          D3;
end bhv;
```

PACKAGE:

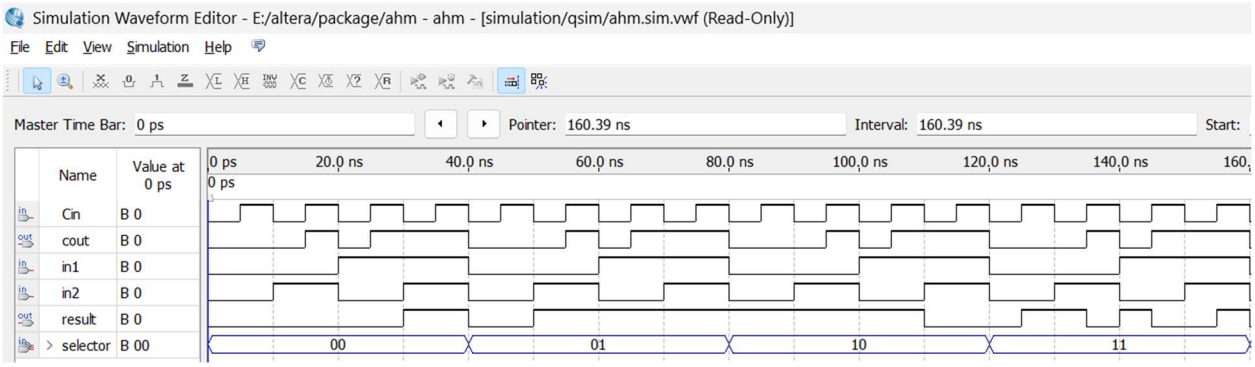
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

package myCompo is
--
component andGate is
    port (
        A,B : in std_logic;
        X : out std_logic
    );
end component;
--
component orGate is
    port (
        A,B : in std_logic;
        X : out std_logic
    );
end component;
--
component nandGate is
    port (
        A,B : in std_logic;
        X : out std_logic
    );
end component;
--
component fullAdder is
    port(
        A, B, Cin : in STD_LOGIC;
        S, C : out STD_LOGIC
    );
end component;
--
component muxx is
    port(
        D0,D1,D2,D3 : in STD_LOGIC;
          S: in STD_LOGIC_VECTOR(1 downto 0);
        X : out STD_LOGIC
    );
end component;
--
end myCompo;
```

TOP-ENTITY(Wrapper File)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use WORK.myCompo.ALL;
```

```
entity ahm is
    port(
        in1,in2,Cin : in std_logic;
          selector : in std_logic_vector(1 downto 0);
          result,cout : out std_logic
    );
end ahm;

architecture bhv of ahm is
    signal and_out, or_out, nand_out, sum_out : std_logic;

begin

    U1: andGate port map (in1,in2,and_out);

    U2: orGate port map (in1,in2,or_out);

    U3: nandGate port map (in1,in2,nand_out);

    U4: fullAdder port map (in1,in2,Cin,sum_out,cout);

    U5: muxx port map (and_out,or_out,nand_out,sum_out,selector,result);


end bhv;
```

Simulation:



Simulation 1: All four components (AND,OR,NAND,SUM) are working on their respective selections.

Control Signal to Operation Mapping:

SELECTION	OPERATION
00	AND GATE
01	OR GATE
10	NAND GATE
11	ADDITION

Table 1: Operation corresponding to selection lines.