Lab Task_01

ALU implementation using your own library components

ENTITIES:

1:OR gate

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

entity Orgate is
    port(
        or_in1, or_in2 : in STD_LOGIC;
        or_out : out STD_LOGIC
    );
end Orgate;

architecture bhv of Orgate is
begin
    or_out <= or_in1 OR or_in2;
end bhv;
```

1:AND gate

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

entity Andgate is
    port(
        and_in1, and_in2 : in STD_LOGIC;
        and_out : out STD_LOGIC
    );
end Andgate;

architecture bhv of Andgate is
begin
    and_out <= and_in1 AND and_in2;
end bhv;
```

2:NAND gate

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

entity Nandgate is
    port(
        nand_in1, nand_in2 : in STD_LOGIC;
        nand_out : out STD_LOGIC
    );
end Nandgate;

architecture bhv of Nandgate is
begin
```

```
    nand_out <= NOT (nand_in1 AND nand_in2);
end bhv;
```

3:Full Adder

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

entity Fulladder is
    port(
        adder_inA, adder_inB, adder_inC : in STD_LOGIC;
        adder_outS, adder_outC : out STD_LOGIC
    );
end Fulladder;

architecture bhv of Fulladder is
begin
    adder_outS <= (adder_inA XOR adder_inB) XOR adder_inC;
    adder_outC <= (adder_inA AND adder_inB) OR (adder_inC AND (adder_inA XOR
adder_inB));
end bhv;
```

4:MUX

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

entity Mux is
    port ( mux_inI0, mux_inI1, mux_inI2, mux_inI3 : in STD_LOGIC;
           mux_Sel : in STD_LOGIC_VECTOR(1 downto 0);
           mux_outY : out STD_LOGIC);
end Mux;

architecture bhv of Mux is
begin
    process (mux_Sel,  mux_inI0, mux_inI1, mux_inI2, mux_inI3)
    begin
        case mux_Sel is
            when "00" => mux_outY <= mux_inI0;
            when "01" => mux_outY <= mux_inI1;
            when "10" => mux_outY <= mux_inI2;
            when "11" => mux_outY <= mux_inI3;
            when others => mux_outY <= '0';
        end case;
    end process;
end bhv;
```

PACKAGE:

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

package alu_components is

    component Orgate
        port ( or_in1, or_in2 : in STD_LOGIC;
               or_out : out STD_LOGIC);
```

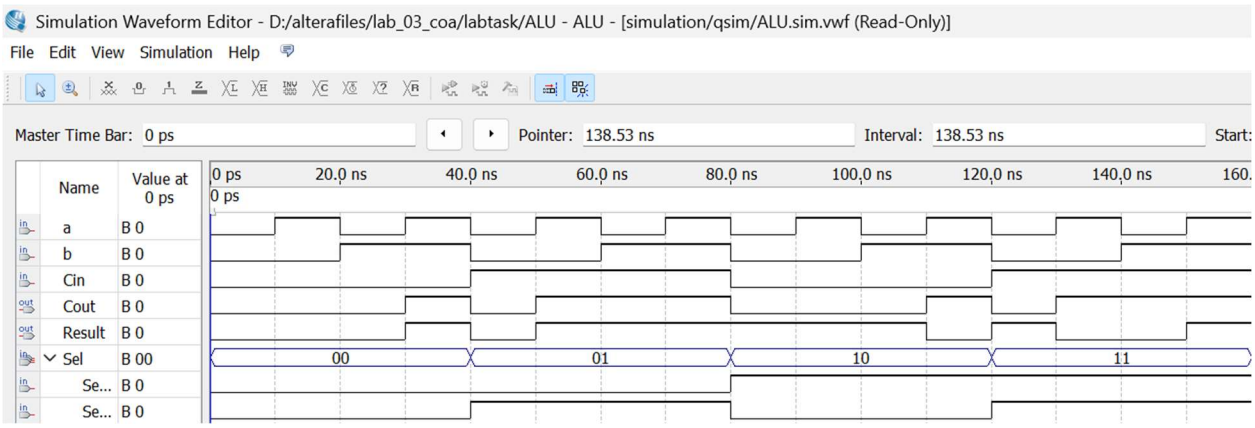
```
    end component;

    component Andgate
        port ( and_in1, and_in2 : in STD_LOGIC;
               and_out : out STD_LOGIC);
    end component;

    component Nandgate
        port ( nand_in1, nand_in2 : in STD_LOGIC;
               nand_out : out STD_LOGIC);
    end component;

    component Fulladder
        port ( adder_inA, adder_inB, adder_inC : in STD_LOGIC;
               adder_outS, adder_outC : out STD_LOGIC);
    end component;

    component Mux
        port ( mux_inI0, mux_inI1, mux_inI2, mux_inI3 : in STD_LOGIC;
               mux_Sel : in STD_LOGIC_VECTOR(1 downto 0);
               mux_outY : out STD_LOGIC);
    end component;
end alu_components;
```

TOP-ENTITY(Wrapper File)

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

entity ALU is
    port (
        a, b, Cin : in STD_LOGIC;
        Sel : in STD_LOGIC_VECTOR(1 downto 0);
        Result, Cout : out STD_LOGIC
    );
end ALU;

architecture struct of ALU is
    signal and_out, or_out, nand_out, sum_out, carry_out: STD_LOGIC;
begin

    U1: Andgate port map(and_in1 => a, and_in2 => b, and_out => and_out);

    U2: Orgate port map(or_in1 => a, or_in2 => b, or_out => or_out);

    U3: Nandgate port map(nand_in1 => a, nand_in2 => b, nand_out =>
nand_out);

    U4: Fulladder port map(adder_inA => a, adder_inB => b, adder_inC => Cin,
adder_outS => sum_out, adder_outC => carry_out);

    U5: Mux port map(
        mux_inI0 => and_out,
        mux_inI1 => or_out,
        mux_inI2 => nand_out,
```

```
        mux_inI3 => sum_out,
        mux_Sel => Sel,
        mux_outY => Result
    );

    Cout <= carry_out;
end struct;
```

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.