ALU 32 Bits

The aim of this project is to construct a 32-Bit ALU using various combinational logic circuits for various arithmetic and logical operations. This ALU then can be used in building a 32-bit processor which require all these necessary operations.

The components that we have used are listed below.

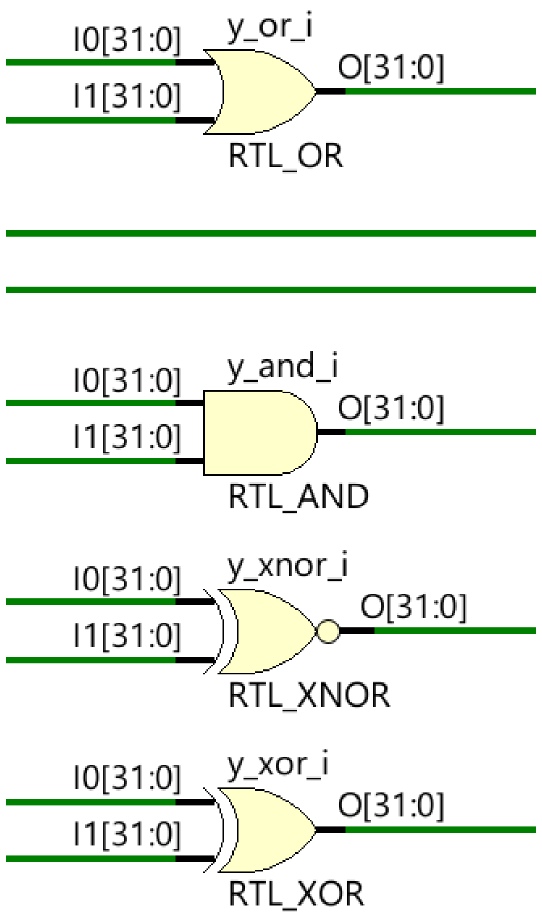
The control signals for various operations of the ALU is as follows: -

Operations	Ainv	Binv	Cin	ALU_Op
A NOT	0	0	0	0000
B NOT	0	0	0	0001
AND	0	0	0	0010
NOR	1	1	0	0010
OR	0	0	0	0011
NAND	1	1	0	0011
XOR	0	0	0	0100
XNOR	0	0	0	0101
ADD	0	0	0	0110
SUB	0	1	1	0110
LLS	0	0	0	0111
LRS	0	0	0	1000
ALS	0	0	0	1001
ARS	0	0	0	1010

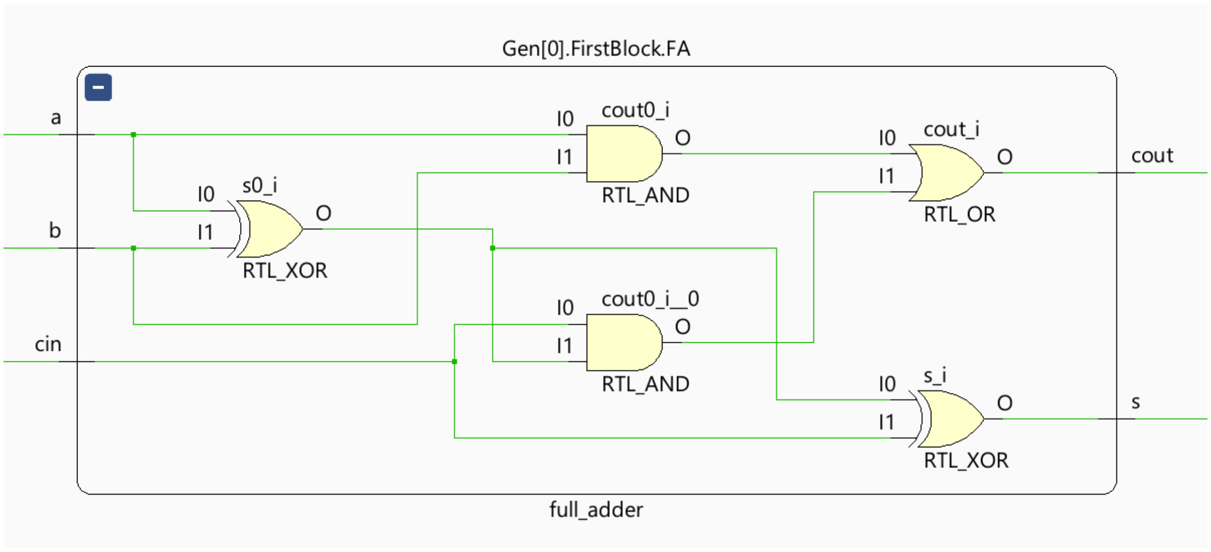
These operations can be performed by the 32Bit ALU by giving the appropriate control Signals.

Components Used

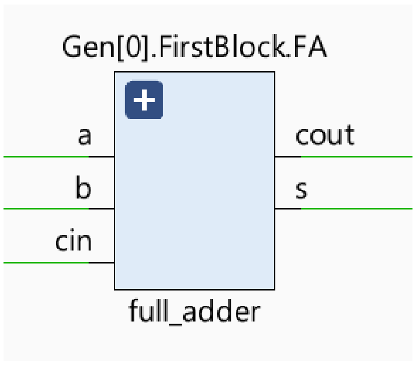
Basic Gates



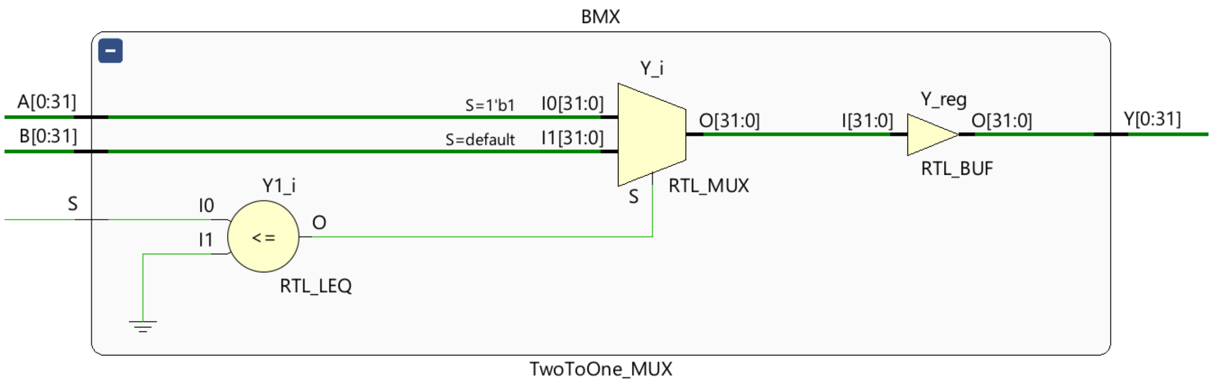
Full Adder



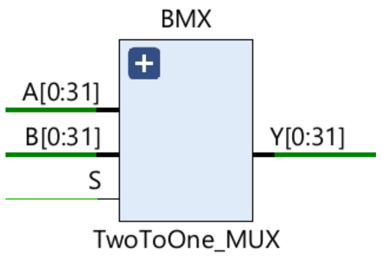
```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY full_adder IS
    PORT (
        a, b, cin : IN STD_LOGIC;
        s, cout : OUT STD_LOGIC

    );
END full_adder;

ARCHITECTURE behavior OF full_adder IS

BEGIN

    s <= a XOR b XOR cin;
    cout <= (a AND b) OR (cin AND (a XOR b));

END behavior;
```



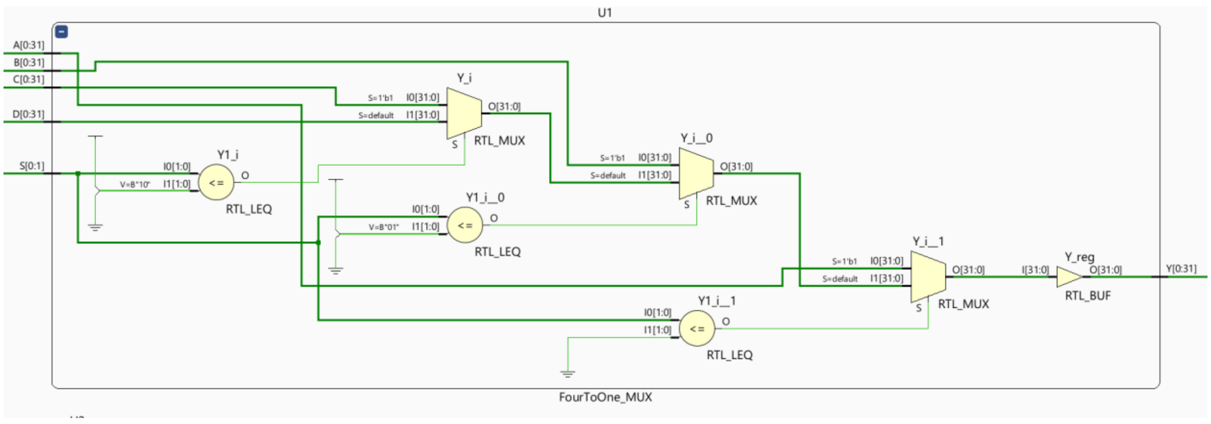
2:1 Mux (BUS)

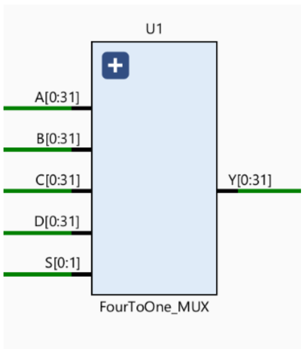


```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY TwoToOne_MUX IS
    PORT (
        A, B : IN STD_LOGIC_VECTOR (0 TO 31);
        S : IN STD_LOGIC;
        Y : OUT STD_LOGIC_VECTOR (0 TO 31)
    );
END TwoToOne_MUX;

ARCHITECTURE DataFlow OF TwoToOne_MUX IS
BEGIN
    PROCESS (A, B, S)
    BEGIN
        IF S <= '0' THEN
            Y <= A;
        ELSIF S <= '1' THEN
            Y <= B;
        END IF;
    END PROCESS;

END DataFlow;
```



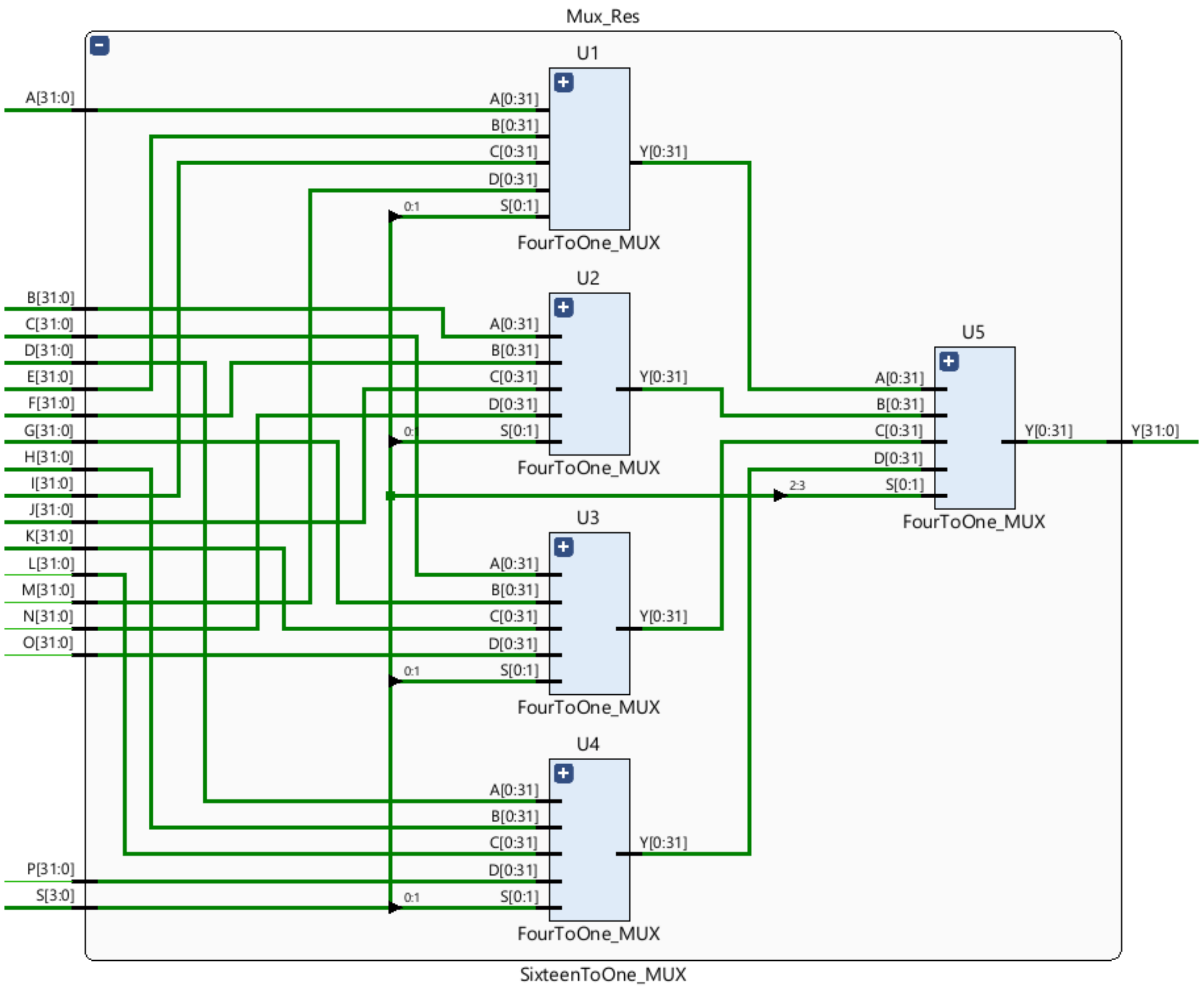
4:1 Mux (Bus)



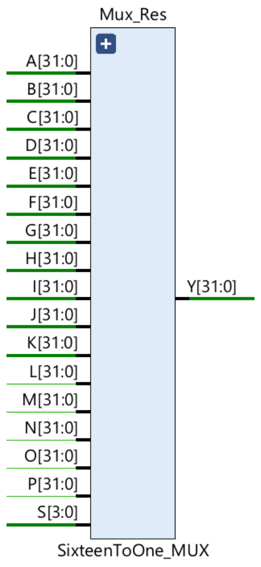


```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY FourToOne_MUX IS
    PORT (
        A, B, C, D : IN STD_LOGIC_VECTOR (0 TO 31);
        S : IN STD_LOGIC_VECTOR (0 TO 1);
        Y : OUT STD_LOGIC_VECTOR (0 TO 31)
    );
END FourToOne_MUX;

ARCHITECTURE DataFlow OF FourToOne_MUX IS
BEGIN

    PROCESS
    BEGIN
        IF S <= "00" THEN
            Y <= A;
        ELSIF S <= "01" THEN
            Y <= B;
        ELSIF S <= "10" THEN
            Y <= C;
        ELSIF S <= "11" THEN
            Y <= D;
        END IF;

    END PROCESS;

END DataFlow;
```

16:1 Mux (Bus)

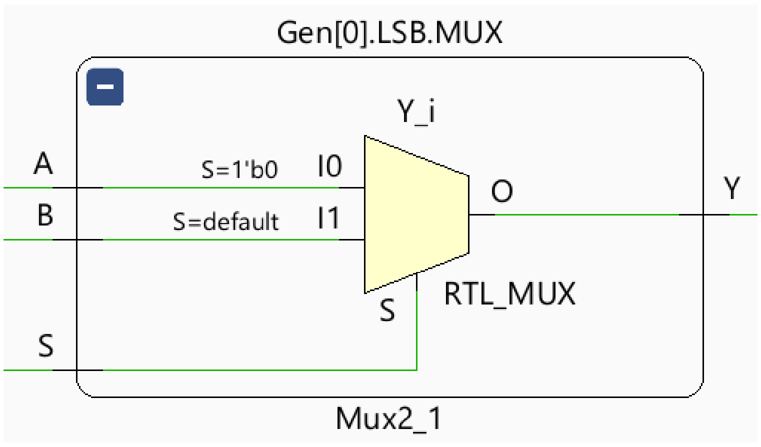


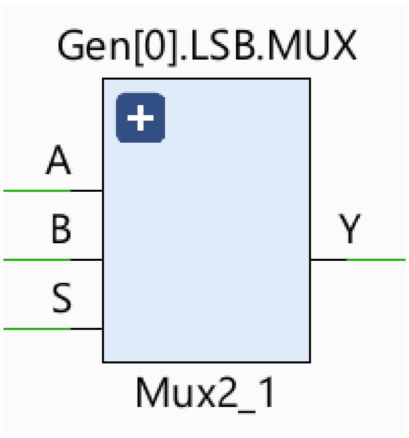
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY SixteenToOne_MUX IS
    PORT (
        A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P
: IN STD_LOGIC_VECTOR (31 DOWNTO 0);
        S : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        Y : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
    );
END SixteenToOne_MUX;

ARCHITECTURE Structural OF SixteenToOne_MUX IS
    COMPONENT FourToOne_MUX IS
        PORT (
            A, B, C, D : IN STD_LOGIC_VECTOR (0 TO 31);
            S : IN STD_LOGIC_VECTOR (0 TO 1);
```



```
            Y : OUT STD_LOGIC_VECTOR (0 TO 31)
        );
    END COMPONENT;
    SIGNAL y0, y1, y2, y3 : STD_LOGIC_VECTOR (0 TO 31);
BEGIN
    U1 : FourToOne_MUX PORT MAP(A => A, B => E, C => I, D => M, S(0) => S(0),
S(1) => S(1), Y => y0);
    U2 : FourToOne_MUX PORT MAP(A => B, B => F, C => J, D => N, S(0) => S(0),
S(1) => S(1), Y => y1);
    U3 : FourToOne_MUX PORT MAP(A => C, B => G, C => K, D => O, S(0) => S(0),
S(1) => S(1), Y => y2);
    U4 : FourToOne_MUX PORT MAP(A => D, B => H, C => L, D => P, S(0) => S(0),
S(1) => S(1), Y => y3);
    U5 : FourToOne_MUX PORT MAP(A => y0, B => y1, C => y2, D => y3, S(0) =>
S(2), S(1) => S(3), Y => Y);

END Structural;
```

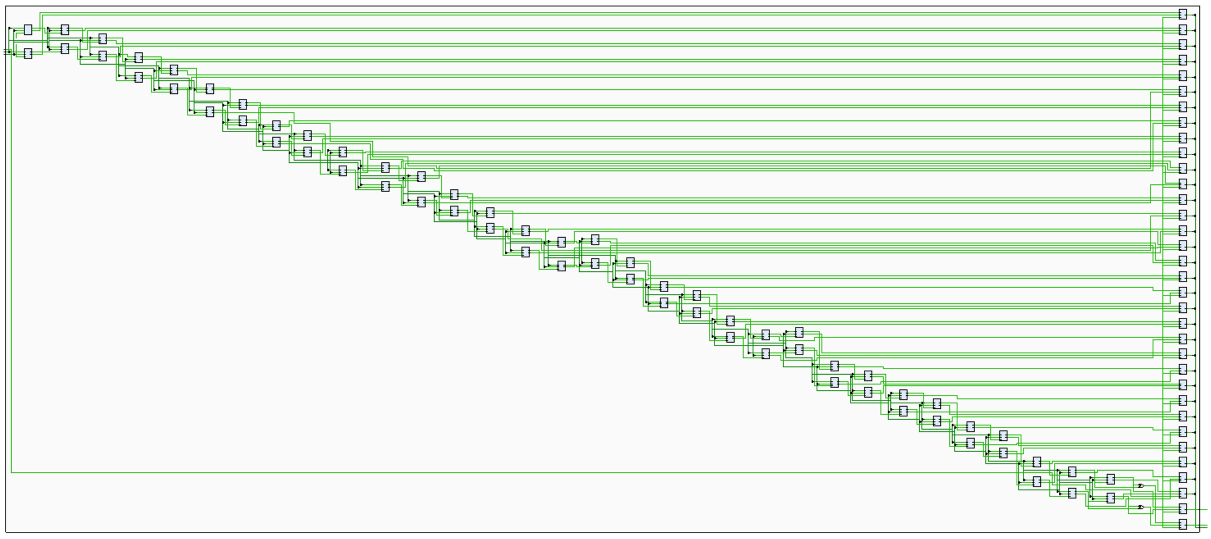
2:1 Mux

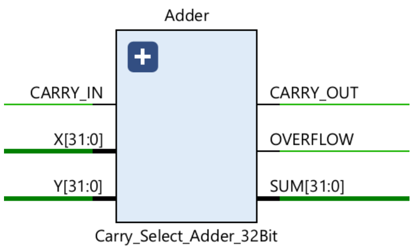




```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY Mux2_1 IS
    PORT (

        A, B : IN STD_LOGIC;
        S : IN STD_LOGIC;
        Y : OUT STD_LOGIC
    );
END Mux2_1;

ARCHITECTURE bhv OF Mux2_1 IS
BEGIN
    PROCESS (A, B, S)
    BEGIN
        IF S = '0' THEN
            Y <= A;
        ELSE
            Y <= B;
        END IF;
    END PROCESS;
END bhv;
```

Carry Select Adder (32 Bit)

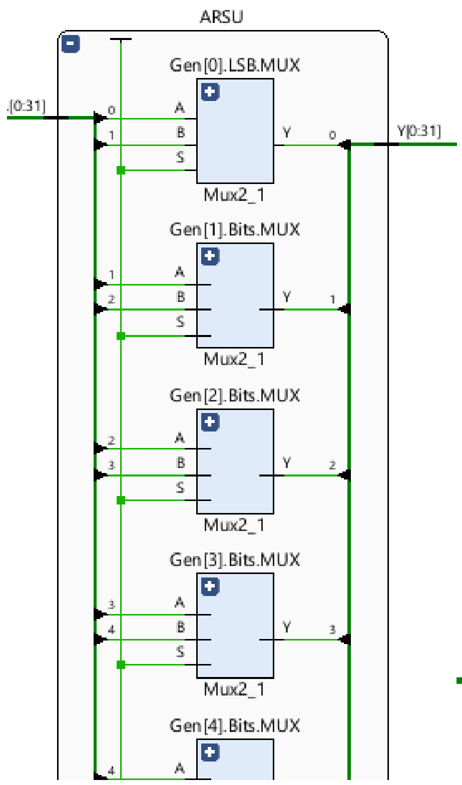


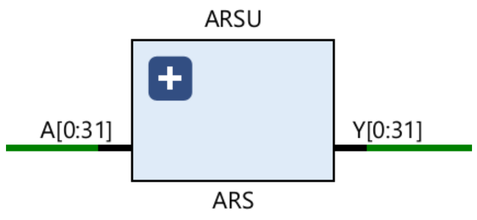


```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE work.ALU_pkg.ALL;

ENTITY Carry_Select_Adder_32Bit IS
    PORT (
        X : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
        Y : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
        CARRY_IN : IN STD_LOGIC;
        SUM : OUT STD_LOGIC_VECTOR (31 DOWNTO 0);
        CARRY_OUT : OUT STD_LOGIC;
        OVERFLOW : OUT STD_LOGIC
    );
END Carry_Select_Adder_32Bit;

ARCHITECTURE Behavioral OF Carry_Select_Adder_32Bit IS

    SIGNAL A, B, C0, C1 : STD_LOGIC_VECTOR(0 TO 31);
    SIGNAL OF0, OF1 : STD_LOGIC;

BEGIN

    Gen : FOR I IN 0 TO 31 GENERATE
        FirstBlock : IF I = 0 GENERATE
            FA : Full_Adder PORT MAP(A => X(I), B => Y(I), Cin => '0', S =>
A(I), Cout => C0(I));
            FA_2 : Full_Adder PORT MAP(A => X(I), B => Y(I), Cin => '1', S =>
B(I), Cout => C1(I));
            Mux : Mux2_1 PORT MAP(A => A(I), B => B(I), S => CARRY_IN, Y =>
SUM(I));
```

```
        END GENERATE FirstBlock;

        LaterBlocks : IF I > 0 GENERATE
            FA : Full_Adder PORT MAP(A => X(I), B => Y(I), Cin => C0(I - 1), S
=> A(I), Cout => C0(I));
            FA_2 : Full_Adder PORT MAP(A => X(I), B => Y(I), Cin => C1(I - 1),
S => B(I), Cout => C1(I));
            Mux : Mux2_1 PORT MAP(A => A(I), B => B(I), S => CARRY_IN, Y =>
SUM(I));
        END GENERATE LaterBlocks;

    END GENERATE Gen;

    MUX_Res : Mux2_1 PORT MAP(A => C0(31), B => C1(31), S => CARRY_IN, Y =>
CARRY_OUT);
    -- Overflow Detection
    OF0 <= C0(31) XOR C0(30);
    OF1 <= C1(31) XOR C1(30);

    MUX_Overflow : Mux2_1 PORT MAP(A => OF0, B => OF1, S => CARRY_IN, Y =>
OVERFLOW);

END Behavioral;
```

Arithmetic Right Shift Unit

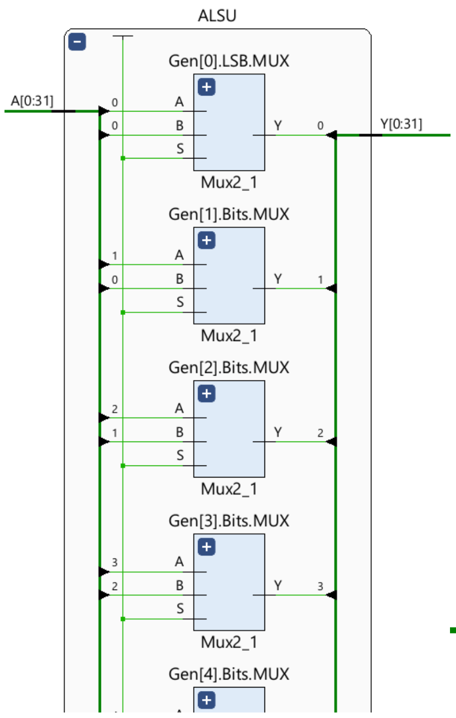


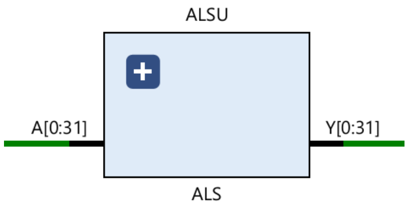


```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE work.ALU_pkg.Mux2_1;

ENTITY ARS IS
    PORT (
        A : IN STD_LOGIC_VECTOR(0 TO 31);
        Y : OUT STD_LOGIC_VECTOR(0 TO 31)
    );
END ARS;

ARCHITECTURE Behavioral OF ARS IS

BEGIN
    Gen : FOR I IN 0 TO 31 GENERATE
        LSB : IF I = 0 GENERATE

            MUX : Mux2_1 PORT MAP(A => A(I), B => A(I + 1), S => '1', Y =>
Y(I));
        END GENERATE LSB;

        Bits : IF I > 0 AND I < 31 GENERATE

            MUX : Mux2_1 PORT MAP(A => A(I), B => A(I + 1), S => '1', Y =>
Y(I));
```

```
            END GENERATE Bits;

            MSB : IF I = 31 GENERATE

                MUX : Mux2_1 PORT MAP(A => A(I), B => A(I), S => '1', Y => Y(I));
            END GENERATE MSB;
        END GENERATE Gen;

    END Behavioral;
```

Arithmetic Left Shift Unit

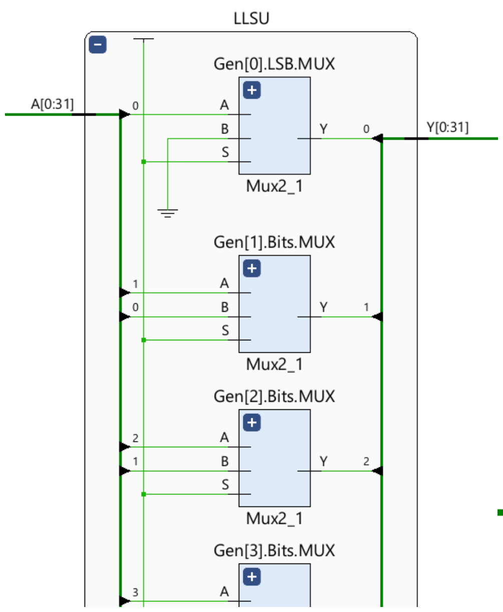


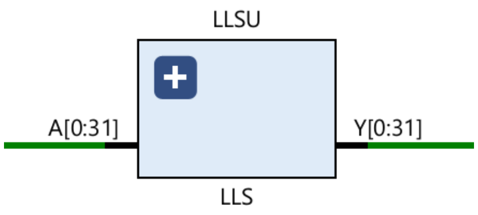


```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE work.ALU_pkg.Mux2_1;

ENTITY ALS IS
    PORT (
        A : IN STD_LOGIC_VECTOR(0 TO 31);
        Y : OUT STD_LOGIC_VECTOR(0 TO 31)
    );
END ALS;

ARCHITECTURE Behavioral OF ALS IS

BEGIN
    Gen : FOR I IN 0 TO 31 GENERATE
        LSB : IF I = 0 GENERATE

            MUX : Mux2_1 PORT MAP(A => A(I), B => A(I), S => '1', Y => Y(I));
        END GENERATE LSB;

        Bits : IF I > 0 AND I < 31 GENERATE

            MUX : Mux2_1 PORT MAP(A => A(I), B => A(I - 1), S => '1', Y =>
Y(I));
        END GENERATE Bits;

        MSB : IF I = 31 GENERATE
```

```
            MUX : Mux2_1 PORT MAP(A => A(I), B => A(I - 1), S => '1', Y =>
Y(I));
        END GENERATE MSB;
    END GENERATE Gen;

END Behavioral;
```

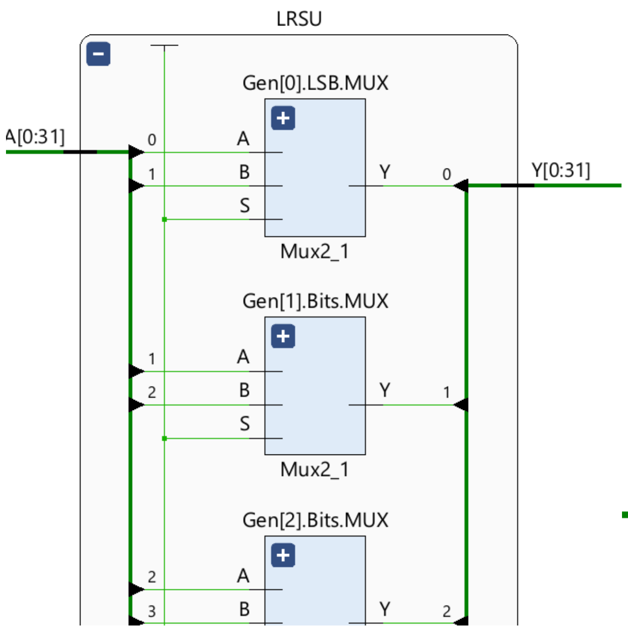
Logical Right Shift Unit

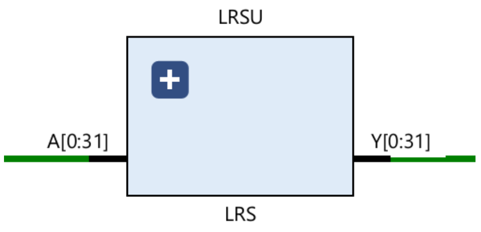




```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE work.ALU_pkg.Mux2_1;

ENTITY LRS IS
    PORT (
        A : IN STD_LOGIC_VECTOR(0 TO 31);
        Y : OUT STD_LOGIC_VECTOR(0 TO 31)
    );
END LRS;

ARCHITECTURE Behavioral OF LRS IS

BEGIN
    Gen : FOR I IN 0 TO 31 GENERATE
        LSB : IF I = 0 GENERATE

            MUX : Mux2_1 PORT MAP(A => A(I), B => A(I + 1), S => '1', Y =>
Y(I));
        END GENERATE LSB;

        Bits : IF I > 0 AND I < 31 GENERATE

            MUX : Mux2_1 PORT MAP(A => A(I), B => A(I + 1), S => '1', Y =>
Y(I));
        END GENERATE Bits;

        MSB : IF I = 31 GENERATE

            MUX : Mux2_1 PORT MAP(A => A(I), B => '0', S => '1', Y => Y(I));
        END GENERATE MSB;
    END GENERATE Gen;

END Behavioral;
```

Logical Left Shift Unit





```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE work.ALU_pkg.Mux2_1;

ENTITY LLS IS
    PORT (
        A : IN STD_LOGIC_VECTOR(0 TO 31);
        Y : OUT STD_LOGIC_VECTOR(0 TO 31)
    );
END LLS;

ARCHITECTURE Behavioral OF LLS IS

BEGIN
    Gen : FOR I IN 0 TO 31 GENERATE
        LSB : IF I = 0 GENERATE

            MUX : Mux2_1 PORT MAP(A => A(I), B => '0', S => '1', Y => Y(I));
        END GENERATE LSB;

        Bits : IF I > 0 AND I < 31 GENERATE

            MUX : Mux2_1 PORT MAP(A => A(I), B => A(I - 1), S => '1', Y =>
Y(I));
        END GENERATE Bits;

        MSB : IF I = 31 GENERATE
```

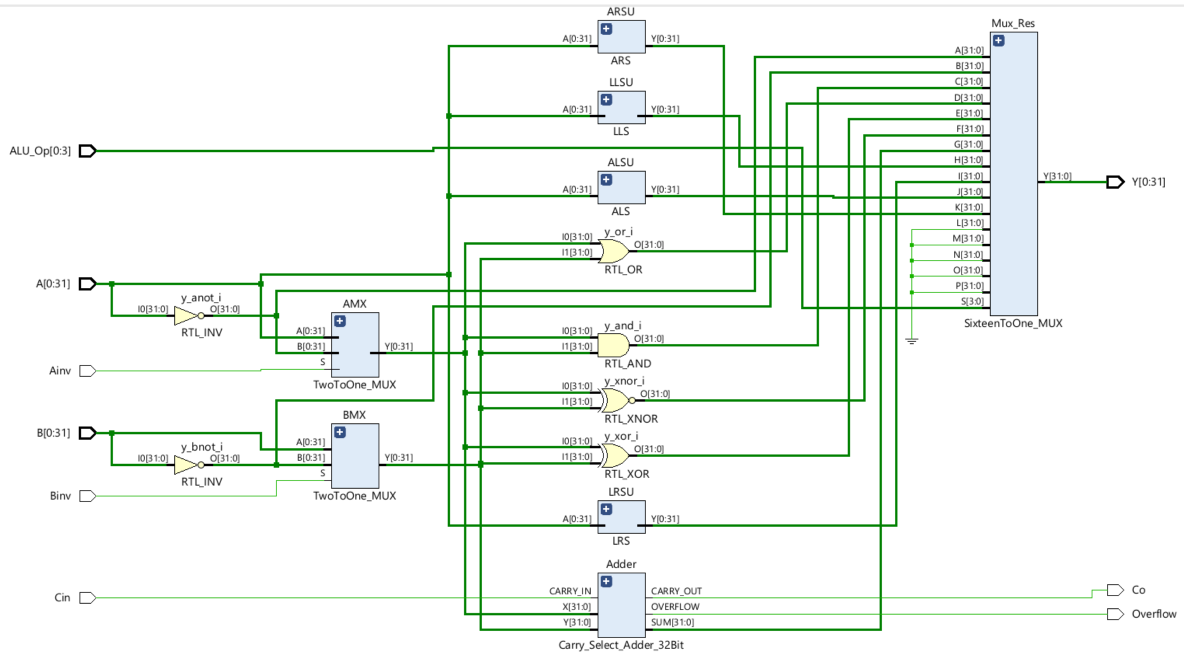
```
            MUX : Mux2_1 PORT MAP(A => A(I), B => A(I - 1), S => '1', Y =>
Y(I));
        END GENERATE MSB;
    END GENERATE Gen;

END Behavioral;
```

We included these components in a package and then used that package in our ALU

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

-- Package Declaration Section
PACKAGE ALU_pkg IS

    -- 2:1 Mux (Bussless)
    COMPONENT Mux2_1 IS
        PORT (

            A, B : IN STD_LOGIC;
            S : IN STD_LOGIC;
            Y : OUT STD_LOGIC
        );
    END COMPONENT;

    -- 2:1 Mux (Bus)
    COMPONENT TwoToOne_Mux IS
        PORT (
            A, B : IN STD_LOGIC_VECTOR (0 TO 31);
            S : IN STD_LOGIC;
            Y : OUT STD_LOGIC_VECTOR (0 TO 31)
        );
    END COMPONENT;

    -- 4:1 Mux
    COMPONENT FourToOne_MUX IS
        PORT (
            A, B, C, D : IN STD_LOGIC_VECTOR (0 TO 31);
            S : IN STD_LOGIC_VECTOR (0 TO 1);
            Y : OUT STD_LOGIC_VECTOR (0 TO 31)
        );
    END COMPONENT;

    -- 16:1 Mux
    COMPONENT SixteenToOne_MUX IS
        PORT (
            A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P : IN
STD_LOGIC_VECTOR (31 DOWNTO 0);
            S : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
            Y : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
        );
    END COMPONENT;

    --Full Adder
```

```vhdl
    COMPONENT Full_Adder IS
        PORT (
            a, b, cin : IN STD_LOGIC;
            s, cout : OUT STD_LOGIC
        );
    END COMPONENT;

    -- Carry Select Adder - 32Bit
    COMPONENT Carry_Select_Adder_32Bit IS
        PORT (
            X : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
            Y : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
            CARRY_IN : IN STD_LOGIC;
            SUM : OUT STD_LOGIC_VECTOR (31 DOWNTO 0);
            CARRY_OUT : OUT STD_LOGIC;
            OVERFLOW : OUT STD_LOGIC);
    END COMPONENT;

    --Logical Right shift
    COMPONENT LRS IS
        PORT (
            A : IN STD_LOGIC_VECTOR(0 TO 31);
            Y : OUT STD_LOGIC_VECTOR(0 TO 31)
        );
    END COMPONENT;

    --Logical Left Shift
    COMPONENT LLS IS
        PORT (
            A : IN STD_LOGIC_VECTOR(0 TO 31);
            Y : OUT STD_LOGIC_VECTOR(0 TO 31)
        );
    END COMPONENT;

    --Arithematic Right Shift
    COMPONENT ARS IS
        PORT (
            A : IN STD_LOGIC_VECTOR(0 TO 31);
            Y : OUT STD_LOGIC_VECTOR(0 TO 31)
        );
    END COMPONENT;

    --Arithematic Left Shift
    COMPONENT ALS IS
        PORT (
            A : IN STD_LOGIC_VECTOR(0 TO 31);
            Y : OUT STD_LOGIC_VECTOR(0 TO 31)
        );
    END COMPONENT;

END PACKAGE;
```

ALU_32 Bit



```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE work.ALU_pkg.ALL;

ENTITY ALU_32Bit IS
    PORT (
        A, B : IN STD_LOGIC_VECTOR(0 TO 31);
        Ainv, Binv : IN STD_LOGIC;
        Cin : IN STD_LOGIC;
        ALU_Op : IN STD_LOGIC_VECTOR(0 TO 3);
        Co : OUT STD_LOGIC;
        Overflow : OUT STD_LOGIC;
        Y : OUT STD_LOGIC_VECTOR(0 TO 31)
    );
END ALU_32Bit;

ARCHITECTURE Behavioral OF ALU_32Bit IS

    SIGNAL y_a, y_b, y_and, y_or, y_xor, y_xnor : STD_LOGIC_VECTOR(0 TO 31);
    SIGNAL y_anot, y_bnot, y_sum, y_lls, y_lrs : STD_LOGIC_VECTOR(0 TO 31);
    SIGNAL y_ars, y_als : STD_LOGIC_VECTOR(0 TO 31);

BEGIN

    -- NOT Operation
    y_anot <= NOT A;
    y_bnot <= NOT B;
    -- Ainv , Binv Operations
    AMX : TwoToOne_Mux PORT MAP(A => A, B => y_anot, S => Ainv, Y => y_a);
    BMX : TwoToOne_Mux PORT MAP(A => B, B => y_bnot, S => Binv, Y => y_b);
    -- AND and OR Gate
    y_and <= y_a AND y_b;
    y_or <= y_a OR y_b;

    -- XOR and XNOR Operations
    y_xor <= y_a XOR y_b;
    y_xnor <= y_a XNOR y_b;

    -- Carry Save Adder with Overflow Detection
    Adder : Carry_Select_Adder_32Bit PORT MAP(
        X => y_a, Y => y_b, CARRY_IN => Cin,
        SUM => y_sum, CARRY_OUT => Co, OVERFLOW => Overflow);

    --Logical Left Shift
    LLSU : LLS PORT MAP(A => A, Y => y_lls);

    --Logical Right Shift
    LRSU : LRS PORT MAP(A => A, Y => y_lrs);

    --Arithematic Left Shift
    ALSU : ALS PORT MAP(A => A, Y => y_als);

    --Arithematic Right Shift
    ARSU : ARS PORT MAP(A => A, Y => y_ars);

    --ALU_Op
    Mux_Res : SixteenToOne_MUX PORT MAP(
        A => y_anot, B => y_bnot, C => y_and,
        D => y_or, E => y_xor, F => y_xnor, G => y_sum, H => y_lls, I =>
y_lrs, J => y_als, K => y_ars,
        L => "00000000000000000000000000000000", M =>
"00000000000000000000000000000000",
        N => "00000000000000000000000000000000", O =>
"00000000000000000000000000000000",
        P => "00000000000000000000000000000000", S => ALU_Op, Y => Y);
END Behavioral;
```