INDEX

PROBLEM STATEMENT:

Design and develop structural VHDL mode for 32-bit Floating Point Shift and Add Multiplier using Top-Down approach ,Simulate to verify the functionality and analyse the performance metrics.

ABSTRACT:

Floating point multiplication is a crucial operation in high power computing applications . In this project Multiplier is designed in VHDL environment and simulation results are obtained by Xiling's tool. IEEE 754 support different floating point formats such as Single Precision format, Double Precision format, Quadruple Precision format etc.

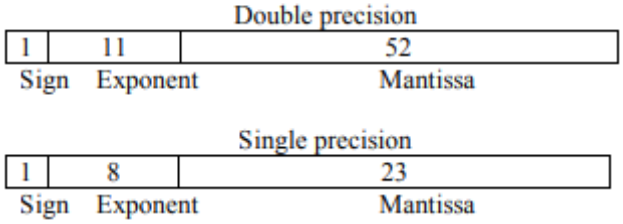


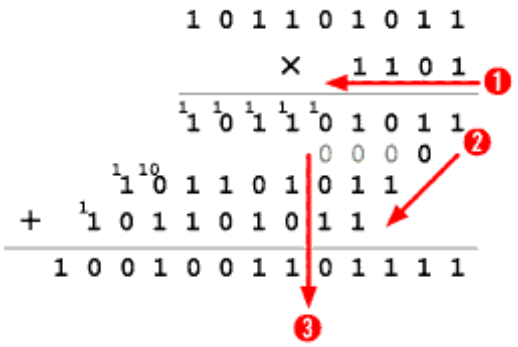
Fig. 1 Floating point formats in the proposed model

Fig. 1 shows the different IEEE 754 floating point formats used commonly. The Single precision format is of 32-bit wide and Double precision format is of 64-bit wide. The Most Significant Bit is the sign bit. The exponent is a signed integer. It is often represented as an unsigned value by adding a bias. In Single precision format, the exponent is of 8-bit wide and the bias is 127, i.e. the exponent has a range of -127 to 128. In Double precision format, the exponent is of 11-bit wide and the bias is 1023, i.e. the exponent has a range of -1023 to 1024. The mantissa or significant of Single precision format is of 23-bit and of double precision format is of 52 bit wide.

ALGORITHM:

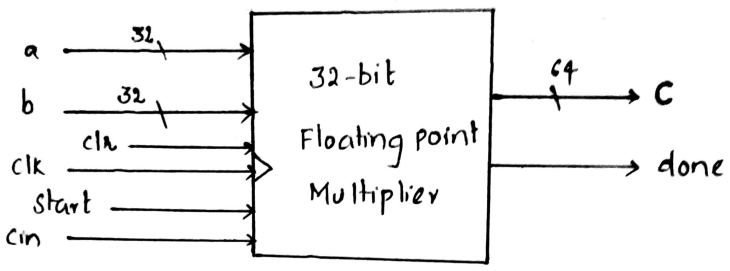
Shift and multiplication:

This algorithm is same as the method we use to multiply two numbers on pen and paper .



To obtain the exponent of the resultant product simply add the exponents of two numbers and check if the position of the most significant 1 of the product is sum of exponent of two numbers plus 1 ,if that is not the case then find the no.of digits in between these two bits (the most significant one and the sum of exponents) and add that number to the exponent . Since the first digit represents the sign , the sign of the resultant operation can be obtained by XOR operation on the sign bits.

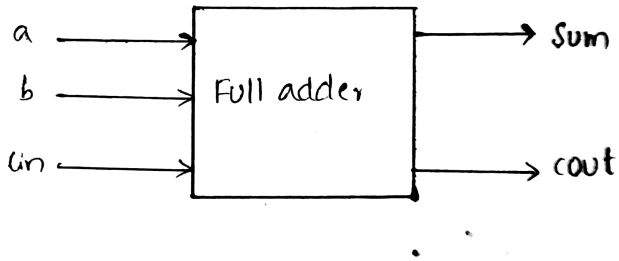
DESIGN ENTITY:



BUILDING BLOCKS:

FULL ADDER:

Design entity:

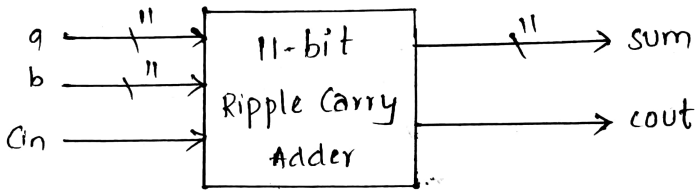


VHDL CODE FOR FULL ADDER:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity fa is

   Port ( a : in STD_LOGIC;

          b : in STD_LOGIC;

          cin : in STD_LOGIC;

          sum : out STD_LOGIC;

          cout : out STD_LOGIC);

end fa;


architecture Behavioral of fa is

begin

sum <= a xor b xor cin;

cout <= (a and b) or (b and cin) or (cin and a);

end Behavioral;
```

11-BIT RIPPLE CARRY ADDER:

Design entity:

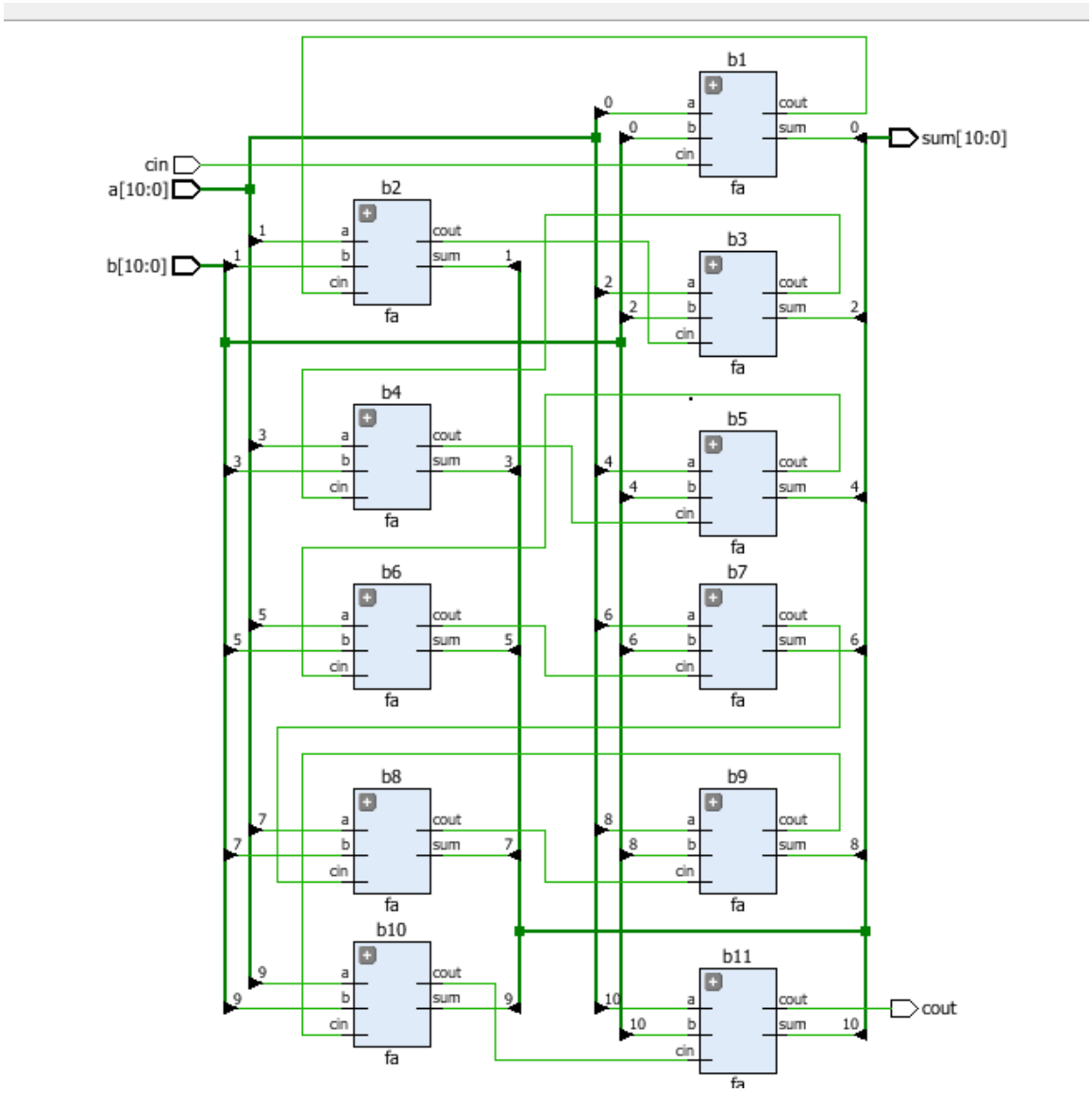


VHDL CODE FOR 11-BIT RIPPLE CARRY ADDER:

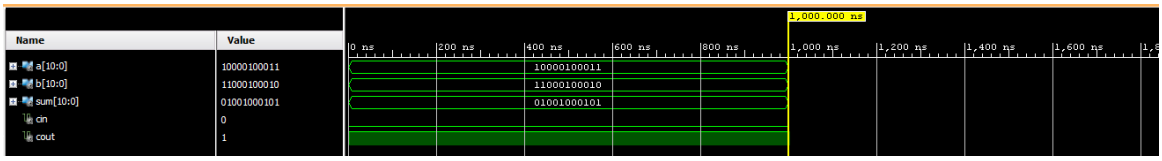
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity rca11 is
   Port ( a : in STD_LOGIC_VECTOR (10 downto 0);
          b : in STD_LOGIC_VECTOR (10 downto 0);
          cin : in STD_LOGIC;
          cout : out STD_LOGIC;
          sum : out STD_LOGIC_VECTOR (10 downto 0));
end rca11;
architecture Behavioral of rca11 is
component fa
   Port ( a : in STD_LOGIC;
      b : in STD_LOGIC;
      cin : in STD_LOGIC;
      sum : out STD_LOGIC;
      cout : out STD_LOGIC);
```

```
end component;
signal c1,c2,c3,c4,c5,c6,c7,c8,c9,c10:std_logic;
begin
b1: fa port map(a(0),b(0),cin,sum(0),c1);
b2: fa port map(a(1),b(1),c1,sum(1),c2);
b3: fa port map(a(2),b(2),c2,sum(2),c3);
b4: fa port map(a(3),b(3),c3,sum(3),c4);
b5: fa port map(a(4),b(4),c4,sum(4),c5);
b6: fa port map(a(5),b(5),c5,sum(5),c6);
b7: fa port map(a(6),b(6),c6,sum(6),c7);
b8: fa port map(a(7),b(7),c7,sum(7),c8);
b9: fa port map(a(8),b(8),c8,sum(8),c9);
b10: fa port map(a(9),b(9),c9,sum(9),c10);
b11: fa port map(a(10),b(10),c10,sum(10),cout);
end Behavioral;
```

RTL Schematic:

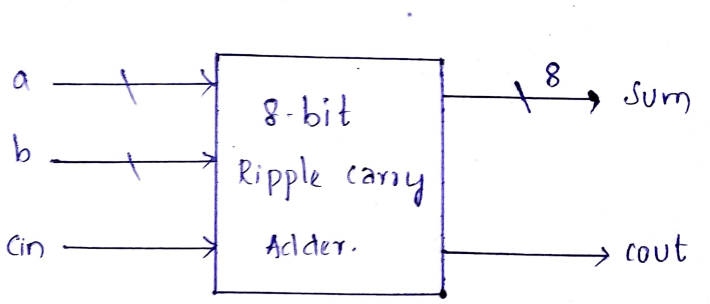


simulation results:



8-**BIT RIPPLE CARRY ADDER:**

Design entity:

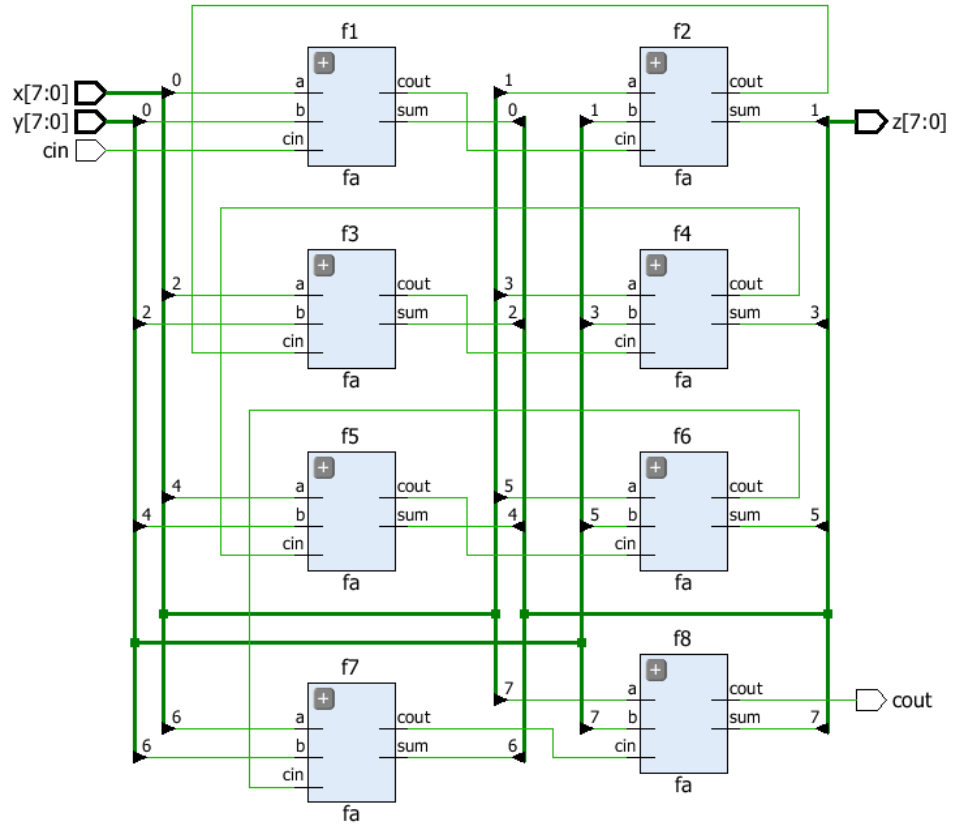


vhdl code for 8-bit ripple carry adder:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity rca8 is
   Port ( x : in STD_LOGIC_VECTOR (7 downto 0);
          y : in STD_LOGIC_VECTOR (7 downto 0);
          cin : in STD_LOGIC;
          cout : out STD_LOGIC;
          z : out STD_LOGIC_VECTOR (7 downto 0));
end rca8;
architecture Behavioral of rca8 is
component fa
   Port ( a : in STD_LOGIC;
     b : in STD_LOGIC;
     cin : in STD_LOGIC;
     sum : out STD_LOGIC;
```

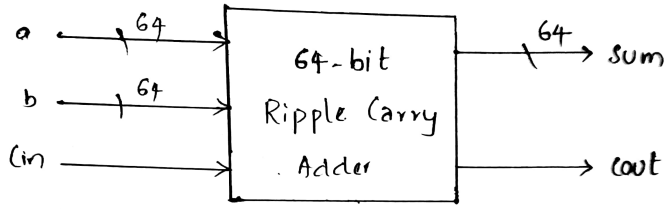
```
        cout : out STD_LOGIC);
end component;
signal c1,c2,c3,c4,c5,c6,c7:std_logic;
begin
f1: fa port map(x(0),y(0),cin,z(0),c1);
f2: fa port map(x(1),y(1),c1,z(1),c2);
f3: fa port map(x(2),y(2),c2,z(2),c3);
f4: fa port map(x(3),y(3),c3,z(3),c4);
f5: fa port map(x(4),y(4),c4,z(4),c5);
f6: fa port map(x(5),y(5),c5,z(5),c6);
f7: fa port map(x(6),y(6),c6,z(6),c7);
f8: fa port map(x(7),y(7),c7,z(7),cout);
end Behavioral;
```

RTL Schematic:



64-BIT RIPPLE CARRY ADDER:

design entity:



VHDL CODE FOR 64-BIT RIPPLE CARRY ADDER:

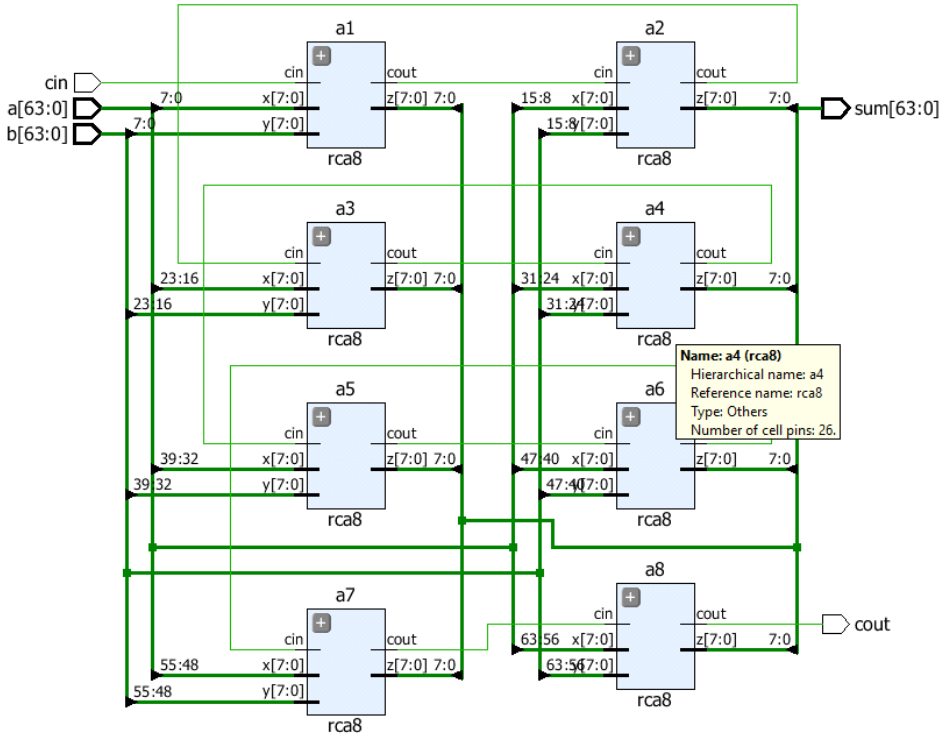
library IEEE;

```
use IEEE.STD_LOGIC_1164.ALL;

entity rca64 is
    Port ( a : in STD_LOGIC_VECTOR (63 downto 0);
           b : in STD_LOGIC_VECTOR (63 downto 0);
           cin : in STD_LOGIC;
           cout : out STD_LOGIC;
           sum : out STD_LOGIC_VECTOR (63 downto 0));
end rca64;

architecture Behavioral of rca64 is

component rca8
    Port ( x : in STD_LOGIC_VECTOR (7 downto 0);
       y : in STD_LOGIC_VECTOR (7 downto 0);
       cin : in STD_LOGIC;
       cout : out STD_LOGIC;
       z : out STD_LOGIC_VECTOR (7 downto 0));
end component;

signal c1,c2,c3,c4,c5,c6,c7:std_logic;

begin

a1: rca8 port map(a(7 downto 0),b(7 downto 0),cin,c1,sum(7 downto 0));

a2: rca8 port map(a(15 downto 8),b(15 downto 8),c1,c2,sum(15 downto 8));

a3: rca8 port map(a(23 downto 16),b(23 downto 16),c2,c3,sum(23 downto 16));

a4: rca8 port map(a(31 downto 24),b(31 downto 24),c3,c4,sum(31 downto 24));

a5: rca8 port map(a(39 downto 32),b(39 downto 32),c4,c5,sum(39 downto 32));

a6: rca8 port map(a(47 downto 40),b(47 downto 40),c5,c6,sum(47 downto 40));

a7: rca8 port map(a(55 downto 48),b(55 downto 48),c6,c7,sum(55 downto 48));
```

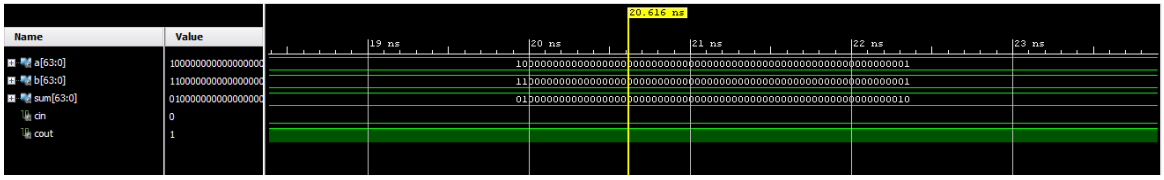
a8: rca8 port map(a(63 downto 56),b(63 downto 56),c7,cout,sum(63 downto 56));

end Behavioral;

RTL Schematic:

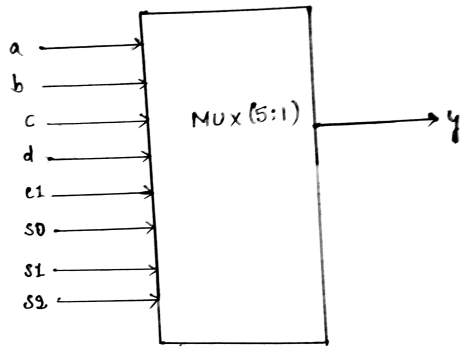


simulation results:



MUX

Design entity:



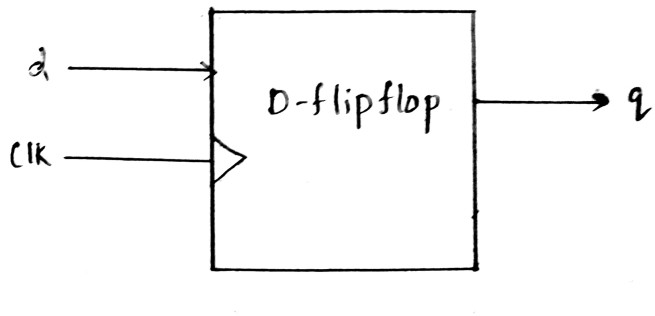
vhdl code for mux:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity mux is
    Port ( a : in STD_LOGIC;
           b : in STD_LOGIC;
           c : in STD_LOGIC;
           d : in STD_LOGIC;
           e1 : in STD_LOGIC;
           s0 : in STD_LOGIC;
           s1 : in STD_LOGIC;
           s2 : in STD_LOGIC;
           y : out STD_LOGIC);
end mux;
architecture Behavioral of mux is
begin
process(s0,s1,s2,a,b,c,d,e1)
```

```
begin
if (s0='0' and s1='0' and s2='0') then y<=a;
elsif(s0='0' and s1='0' and s2='1') then y<=b;
elsif (s0='0' and s1='1' and s2='0') then y<=c;
elsif (s0='0' and s1='1' and s2='1') then y<=d;
else y<=e1;
end if;
end process;
end Behavioral;
```

D-FLIPFLOP:

design entity:



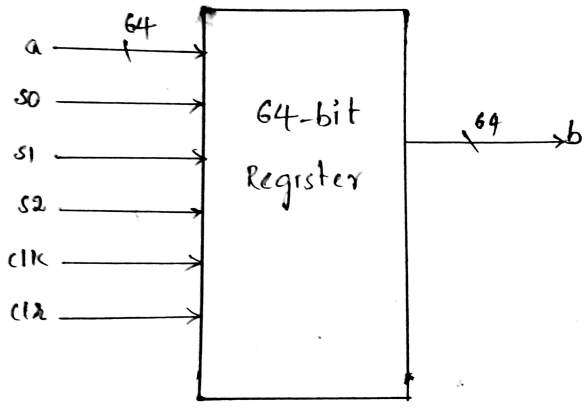
vhdl code for d-flipflop:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity dff is
   Port ( d : in STD_LOGIC;
          clk : in STD_LOGIC;
          q : out STD_LOGIC);
end dff;
```

```
architecture Behavioral of dff is

begin

process(clk)

begin

if (clk='1' and clk'event) then q<=d ;

end if;

end process;

end Behavioral;
```

64-BIT REGISTER:

design entity:



VHDL CODE FOR 64 BIT REGISTER:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity usr64 is

   Port ( a : in STD_LOGIC_VECTOR (63 downto 0);

          b : out STD_LOGIC_VECTOR (63 downto 0);

          s0 : in STD_LOGIC;
```

```
        s1 : in STD_LOGIC;
        s2 : in STD_LOGIC;
        clk : in STD_LOGIC;
        clr : in STD_LOGIC);
end usr64;
architecture Behavioral of usr64 is
component dff
   Port ( d : in STD_LOGIC;
     clk : in STD_LOGIC;
     q : out STD_LOGIC);
end component;
component mux
   Port ( a : in STD_LOGIC;
     b : in STD_LOGIC;
     c : in STD_LOGIC;
     d : in STD_LOGIC;
     e1 : in STD_LOGIC;
     s0 : in STD_LOGIC;
     s1 : in STD_LOGIC;
     s2 : in STD_LOGIC;
     y : out STD_LOGIC);
end component;
signal
m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16,m17,m18,m19,m20,m21,
m22,m23,m24,m25,m26,m27,m28,m29,m30,m31,m32,m33,m34,m35,m36,m37,m38,m39,m40
,m41,m42,m43,m44,m45,m46,m47,m48,m49,m50,
m51,m52,m53,m54,m55,m56,m57,m58,m59,m60,
```

m61,m62,m63,m64,q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11,q12,q13,q14,q15,q16,q17,q18,q19,q20
,q21,q22,q23,q24,q25,q26,q27,q28,q29,q30,q31,q32,q33,q34,q35,q36,q37,q38,q39,q40,q41,q4
2,q43,q44,q45,q46,q47,q48,q49,q50,q51,q52,q53,q54,q55,q56,q57,q58,q59,q60,q61,q62,q63,q
64:std_logic;

begin

mu1: mux port map(clr,a(0),clr,q2,q1,s0,s1,s2,m1);

mu2: mux port map(clr,a(1),q1,q3,q2,s0,s1,s2,m2);

mu3: mux port map(clr,a(2),q2,q4,q3,s0,s1,s2,m3);

mu4: mux port map(clr,a(3),q3,q5,q4,s0,s1,s2,m4);

mu5: mux port map(clr,a(4),q4,q6,q5,s0,s1,s2,m5);

mu6: mux port map(clr,a(5),q5,q7,q6,s0,s1,s2,m6);

mu7: mux port map(clr,a(6),q6,q8,q7,s0,s1,s2,m7);

mu8: mux port map(clr,a(7),q7,q9,q8,s0,s1,s2,m8);

mu9: mux port map(clr,a(8),q8,q10,q9,s0,s1,s2,m9);

mu10: mux port map(clr,a(9),q9,q11,q10,s0,s1,s2,m10);

mu11: mux port map(clr,a(10),q10,q12,q11,s0,s1,s2,m11);

mu12: mux port map(clr,a(11),q11,q13,q12,s0,s1,s2,m12);

mu13: mux port map(clr,a(12),q12,q14,q13,s0,s1,s2,m13);

mu14: mux port map(clr,a(13),q13,q15,q14,s0,s1,s2,m14);

mu15: mux port map(clr,a(14),q14,q16,q15,s0,s1,s2,m15);

mu16: mux port map(clr,a(15),q15,q17,q16,s0,s1,s2,m16);

mu17: mux port map(clr,a(16),q16,q18,q17,s0,s1,s2,m17);

mu18: mux port map(clr,a(17),q17,q19,q18,s0,s1,s2,m18);

mu19: mux port map(clr,a(18),q18,q20,q19,s0,s1,s2,m19);

mu20: mux port map(clr,a(19),q19,q21,q20,s0,s1,s2,m20);

mu21: mux port map(clr,a(20),q20,q22,q21,s0,s1,s2,m21);

mu22: mux port map(clr,a(21),q21,q23,q22,s0,s1,s2,m22);

```
mu23: mux port map(clr,a(22),q22,q24,q23,s0,s1,s2,m23);

mu24: mux port map(clr,a(23),q23,q25,q24,s0,s1,s2,m24);

mu25: mux port map(clr,a(24),q24,q26,q25,s0,s1,s2,m25);

mu26: mux port map(clr,a(25),q25,q27,q26,s0,s1,s2,m26);

mu27: mux port map(clr,a(26),q26,q28,q27,s0,s1,s2,m27);

mu28: mux port map(clr,a(27),q27,q29,q28,s0,s1,s2,m28);

mu29: mux port map(clr,a(28),q28,q30,q29,s0,s1,s2,m29);

mu30: mux port map(clr,a(29),q29,q31,q30,s0,s1,s2,m30);

mu31: mux port map(clr,a(30),q30,q32,q31,s0,s1,s2,m31);

mu32: mux port map(clr,a(31),q31,q33,q32,s0,s1,s2,m32);

mu33: mux port map(clr,a(32),q32,q34,q33,s0,s1,s2,m33);

mu34: mux port map(clr,a(33),q33,q35,q34,s0,s1,s2,m34);

mu35: mux port map(clr,a(34),q34,q36,q35,s0,s1,s2,m35);

mu36: mux port map(clr,a(35),q35,q37,q36,s0,s1,s2,m36);

mu37: mux port map(clr,a(36),q36,q38,q37,s0,s1,s2,m37);

mu38: mux port map(clr,a(37),q37,q39,q38,s0,s1,s2,m38);

mu39: mux port map(clr,a(38),q38,q40,q39,s0,s1,s2,m39);

mu40: mux port map(clr,a(39),q39,q41,q40,s0,s1,s2,m40);

mu41: mux port map(clr,a(40),q40,q42,q41,s0,s1,s2,m41);

mu42: mux port map(clr,a(41),q41,q43,q42,s0,s1,s2,m42);

mu43: mux port map(clr,a(42),q42,q44,q43,s0,s1,s2,m43);

mu44: mux port map(clr,a(43),q43,q45,q44,s0,s1,s2,m44);

mu45: mux port map(clr,a(44),q44,q46,q45,s0,s1,s2,m45);

mu46: mux port map(clr,a(45),q45,q47,q46,s0,s1,s2,m46);

mu47: mux port map(clr,a(46),q46,q48,q47,s0,s1,s2,m47);
```

```
mu48: mux port map(clr,a(47),q47,q49,q48,s0,s1,s2,m48);
mu49: mux port map(clr,a(48),q48,q50,q49,s0,s1,s2,m49);
mu50: mux port map(clr,a(49),q49,q51,q50,s0,s1,s2,m50);
mu51: mux port map(clr,a(50),q50,q52,q51,s0,s1,s2,m51);
mu52: mux port map(clr,a(51),q51,q53,q52,s0,s1,s2,m52);
mu53: mux port map(clr,a(52),q52,q54,q53,s0,s1,s2,m53);
mu54: mux port map(clr,a(53),q53,q55,q54,s0,s1,s2,m54);
mu55: mux port map(clr,a(54),q54,q56,q55,s0,s1,s2,m55);
mu56: mux port map(clr,a(55),q55,q57,q56,s0,s1,s2,m56);
mu57: mux port map(clr,a(56),q56,q58,q57,s0,s1,s2,m57);
mu58: mux port map(clr,a(57),q57,q59,q58,s0,s1,s2,m58);
mu59: mux port map(clr,a(58),q58,q60,q59,s0,s1,s2,m59);
mu60: mux port map(clr,a(59),q59,q61,q60,s0,s1,s2,m60);
mu61: mux port map(clr,a(60),q60,q62,q61,s0,s1,s2,m61);
mu62: mux port map(clr,a(61),q61,q63,q62,s0,s1,s2,m62);
mu63: mux port map(clr,a(62),q62,q64,q63,s0,s1,s2,m63);
mu64: mux port map(clr,a(63),q63,clr,q64,s0,s1,s2,m64);
d1: dff port map(m1,clk,q1);
d2: dff port map(m2,clk,q2);
d3: dff port map(m3,clk,q3);
d4: dff port map(m4,clk,q4);
d5: dff port map(m5,clk,q5);
d6: dff port map(m6,clk,q6);
d7: dff port map(m7,clk,q7);
d8: dff port map(m8,clk,q8);
```

```
d9: dff port map(m9,clk,q9);

d10: dff port map(m10,clk,q10);

d11: dff port map(m11,clk,q11);

d12: dff port map(m12,clk,q12);

d13: dff port map(m13,clk,q13);

d14: dff port map(m14,clk,q14);

d15: dff port map(m15,clk,q15);

d16: dff port map(m16,clk,q16);

d17: dff port map(m17,clk,q17);

d18: dff port map(m18,clk,q18);

d19: dff port map(m19,clk,q19);

d20: dff port map(m20,clk,q20);

d21: dff port map(m21,clk,q21);

d22: dff port map(m22,clk,q22);

d23: dff port map(m23,clk,q23);

d24: dff port map(m24,clk,q24);

d25: dff port map(m25,clk,q25);

d26: dff port map(m26,clk,q26);

d27: dff port map(m27,clk,q27);

d28: dff port map(m28,clk,q28);

d29: dff port map(m29,clk,q29);

d30: dff port map(m30,clk,q30);

d31: dff port map(m31,clk,q31);

d32: dff port map(m32,clk,q32);

d33: dff port map(m33,clk,q33);
```

```
d34: dff port map(m34,clk,q34);

d35: dff port map(m35,clk,q35);

d36: dff port map(m36,clk,q36);

d37: dff port map(m37,clk,q37);

d38: dff port map(m38,clk,q38);

d39: dff port map(m39,clk,q39);

d40: dff port map(m40,clk,q40);

d41: dff port map(m41,clk,q41);

d42: dff port map(m42,clk,q42);

d43: dff port map(m43,clk,q43);

d44: dff port map(m44,clk,q44);

d45: dff port map(m45,clk,q45);

d46: dff port map(m46,clk,q46);

d47: dff port map(m47,clk,q47);

d48: dff port map(m48,clk,q48);

d49: dff port map(m49,clk,q49);

d50: dff port map(m50,clk,q50);

d51: dff port map(m51,clk,q51);

d52: dff port map(m52,clk,q52);

d53: dff port map(m53,clk,q53);

d54: dff port map(m54,clk,q54);

d55: dff port map(m55,clk,q55);

d56: dff port map(m56,clk,q56);

d57: dff port map(m57,clk,q57);

d58: dff port map(m58,clk,q58);
```

```
d59: dff port map(m59,clk,q59);

d60: dff port map(m60,clk,q60);

d61: dff port map(m61,clk,q61);

d62: dff port map(m62,clk,q62);

d63: dff port map(m63,clk,q63);

d64: dff port map(m64,clk,q64);

b(0)<=q1;

b(1)<=q2;

b(2)<=q3;

b(3)<=q4;

b(4)<=q5;

b(5)<=q6;

b(6)<=q7;

b(7)<=q8;

b(8)<=q9;

b(9)<=q10;

b(10)<=q11;

b(11)<=q12;

b(12)<=q13;

b(13)<=q14;

b(14)<=q15;

b(15)<=q16;

b(16)<=q17;

b(17)<=q18;

b(18)<=q19;
```

b(19)<=q20;

b(20)<=q21;

b(21)<=q22;

b(22)<=q23;

b(23)<=q24;

b(24)<=q25;

b(25)<=q26;

b(26)<=q27;

b(27)<=q28;

b(28)<=q29;

b(29)<=q30;

b(30)<=q31;

b(31)<=q32;

b(32)<=q33;

b(33)<=q34;

b(34)<=q35;

b(35)<=q36;

b(36)<=q37;

b(37)<=q38;

b(38)<=q39;

b(39)<=q40;

b(40)<=q41;

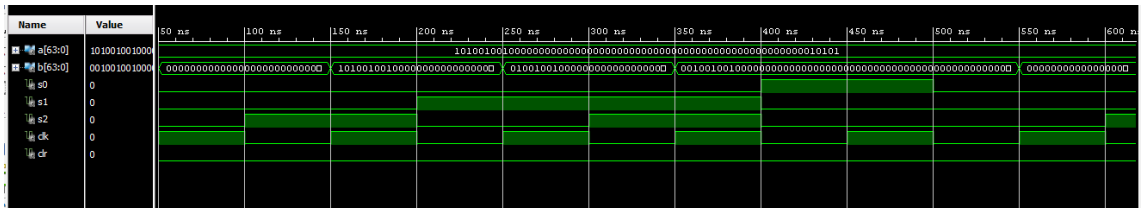
b(41)<=q42;

b(42)<=q43;

b(43)<=q44;

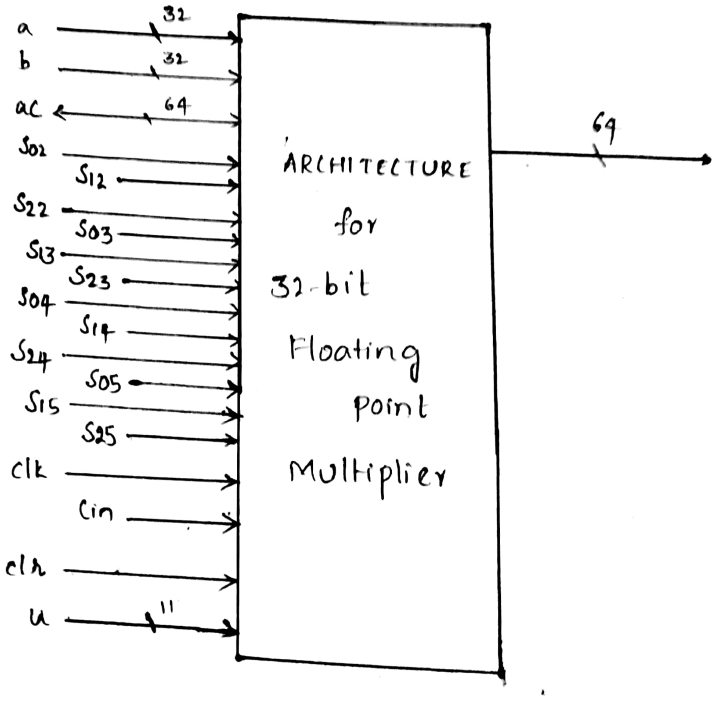
```
b(44)<=q45;

b(45)<=q46;

b(46)<=q47;

b(47)<=q48;

b(48)<=q49;

b(49)<=q50;

b(50)<=q51;

b(51)<=q52;

b(52)<=q53;

b(53)<=q54;

b(54)<=q55;

b(55)<=q56;

b(56)<=q57;

b(57)<=q58;

b(58)<=q59;

b(59)<=q60;

b(60)<=q61;

b(61)<=q62;

b(62)<=q63;

b(63)<=q64;

end Behavioral;
```

simulation results:



ARCHITECTURE:

Design entity:



Vhdl code for architecture:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity architech is

    Port ( a : in STD_LOGIC_VECTOR (31 downto 0);

           b : in STD_LOGIC_VECTOR (31 downto 0);

           c : out STD_LOGIC_VECTOR (63 downto 0);
```

```
          ac : inout STD_LOGIC_VECTOR (63 downto 0);
           u :in std_logic_vector(10 downto 0);
           s02 : in STD_LOGIC;
           s12 : in STD_LOGIC;
           s22 : in STD_LOGIC;
           s03 : in STD_LOGIC;
           s13 : in STD_LOGIC;
           s23 : in STD_LOGIC;
           s04 : in STD_LOGIC;
           s14 : in STD_LOGIC;
           s24 : in STD_LOGIC;
           s05 : in STD_LOGIC;
           s15 : in STD_LOGIC;
           s25 : in STD_LOGIC;
           clk : in STD_LOGIC;
           clr : in STD_LOGIC;
           cin : in std_logic );
end architech;
architecture Behavioral of architech is
component usr64
    Port ( a : in STD_LOGIC_VECTOR (63 downto 0);
       b : out STD_LOGIC_VECTOR (63 downto 0);
       s0 : in STD_LOGIC;
       s1 : in STD_LOGIC;
       s2 : in STD_LOGIC;
```

```
      clk : in STD_LOGIC;
      clr : in STD_LOGIC);
end component;
component rca11
   Port ( a : in STD_LOGIC_VECTOR (10 downto 0);
      b : in STD_LOGIC_VECTOR (10 downto 0);
      cin : in STD_LOGIC;
      cout : out STD_LOGIC;
      sum : out STD_LOGIC_VECTOR (10 downto 0));
end component;
component rca64
   Port ( a : in STD_LOGIC_VECTOR (63 downto 0);
      b : in STD_LOGIC_VECTOR (63 downto 0);
      cin : in STD_LOGIC;
      cout : out STD_LOGIC;
      sum : out STD_LOGIC_VECTOR (63 downto 0));
end component;
signal w,z,q,ac1,p : std_logic_vector(63 downto 0);
signal x,y,s,r,t,v : std_logic_vector(10 downto 0);
signal cout1,cout2,cout3,cout4: std_logic;
begin
r<="000"&a(30 downto 23);
s<="000"&b(30 downto 23);
t<="01100000001";
w<="00000000000000000000000000000000000000001"&a(22 downto 0);
```

rca1: rca11 port map(r,s,cin,cout1,x);

rca2: rca11 port map(x,t,cin,cout2,y);

rca4: rca11 port map(u,y,cin,cout4,v);

usr2: usr64 port map(z,ac1,s02,s12,s22,clk,clr);

usr3: usr64 port map(w,q,s03,s13,s23,clk,clr);

usr4: usr64 port map(ac1,p,s04,s14,s24,clk,clr);

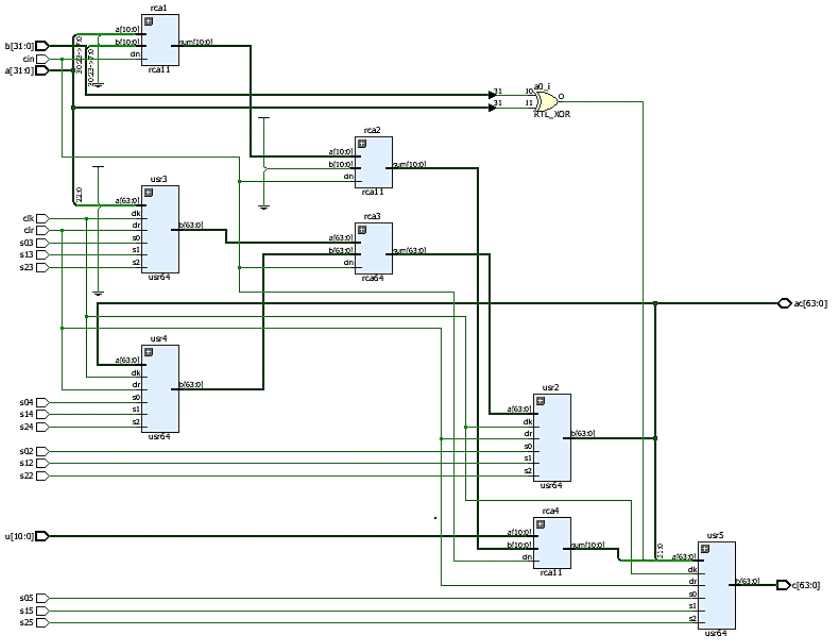
rca3: rca64 port map(q,p,cin,cout3,z);

ac<=ac1;

usr5: usr64 port map((b(31) xor a(31))&v&ac(51 downto 0),c,s05,s15,s25,clk,clr);

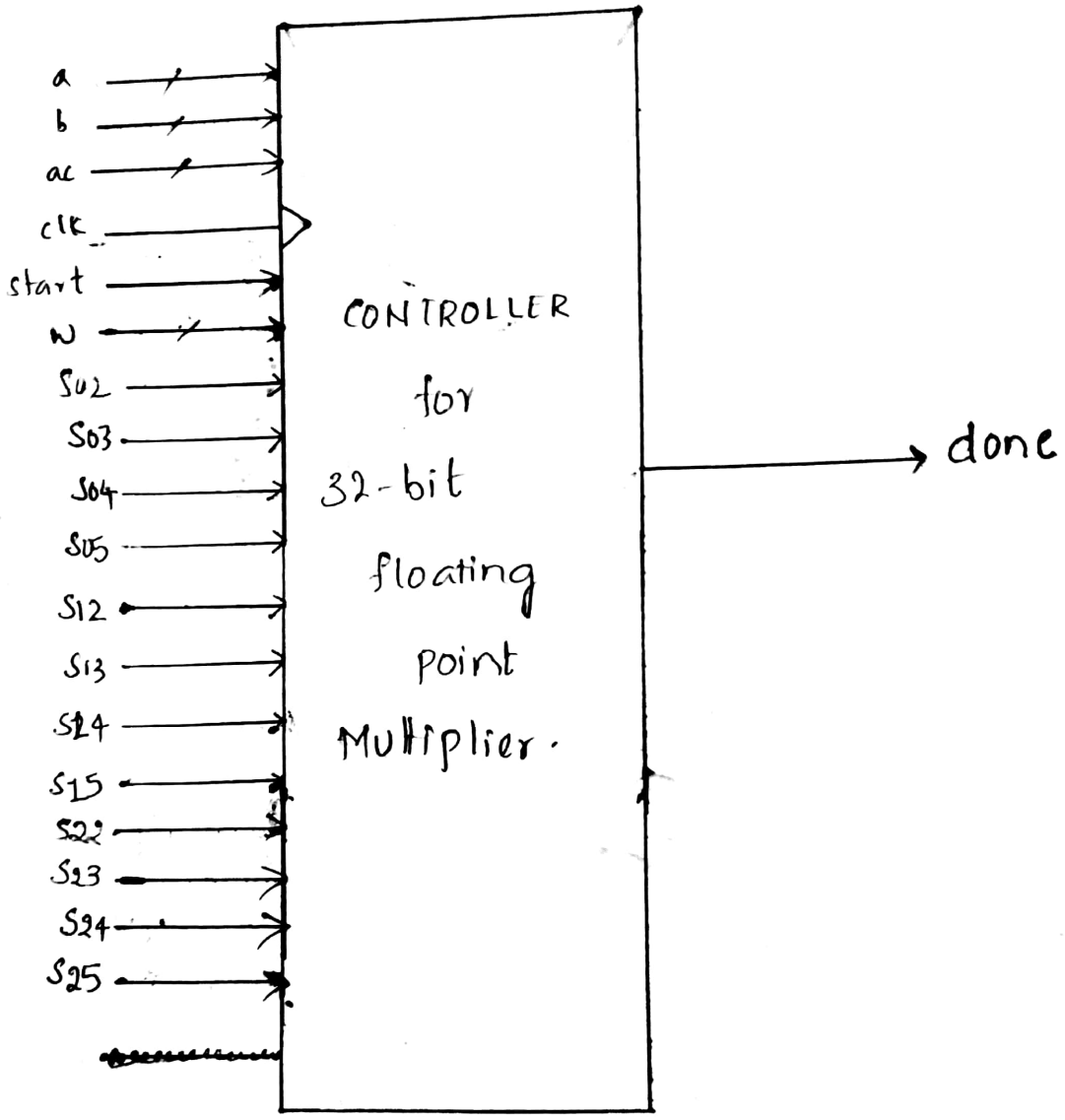
end Behavioral;

RTL Schematic:

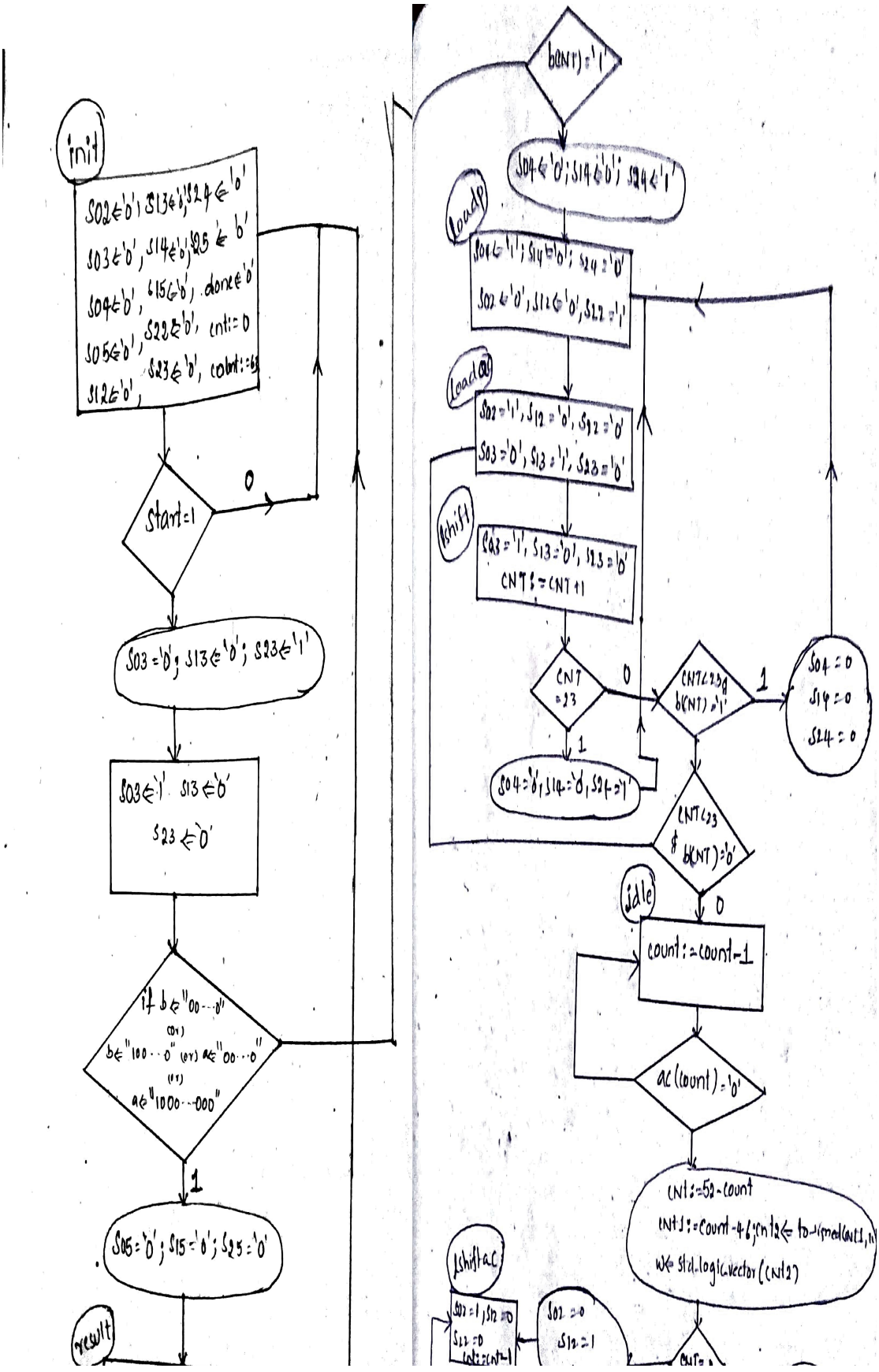


CONTROLLER:

Design entity:



design:


init
S02⇐'0'; S13⇐'0'; S24⇐'0'
S03⇐'0'; S14⇐'0'; S25⇐'0'
S04⇐'0'; b15⇐'0'; done⇐'0'
S05⇐'0'; S22⇐'0'; cnt:=0
S12⇐'0'; S23⇐'0'; count:=52
start=1
0
S03='0'; S13⇐'0'; S23⇐'1'
S03⇐'1' S13⇐'0'
S23⇐'0'
if b⇐"00···0"
or)
b⇐"100···0" (or) a⇐"00···0"
(or)
a⇐"1000···000"
1
S05='0'; S15='0'; S25='0'
result
b(NT)='1'
S04⇐'0'; S14⇐'0'; S24⇐'1'
loadB
S04='1'; S14='0'; S24='0'
S02⇐'0'; S12⇐'0'; S22='1'
loadA
S02='1'; S12='0'; S22='0'
S03='0'; S13='1'; S23='0'
shift
S03='1'; S13='0'; S23='0'
CNT:=CNT+1
CNT=23
0
1
S04='0'; S14='0'; S24='1'
CNT<23 & b(NT)='1'
1
S04=0
S14=0
S24=0
CNT<23 & b(NT)='0'
0
idle
count:=count-1
ac(count)='0'
CNT:=52-count
cnt1:=count-46; cnt2⇐ to_unsigned(cnt1,n)
w⇐ std_logic_vector(CNT2)
shiftac
S02='1'; S12='0'
S12='0'
cnt:=cnt-1
S02='0'
S12='1'

Vhdl code for controller:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity controller is
    Port ( a : in STD_LOGIC_VECTOR (31 downto 0);
           b : in STD_LOGIC_VECTOR (31 downto 0);
           ac : in STD_LOGIC_VECTOR (63 downto 0);
           clk : in STD_LOGIC;
           start : in STD_LOGIC;
            w: out std_logic_vector(10 downto 0);
           s02 : out STD_LOGIC;
           s03 : out STD_LOGIC;
           s04 : out STD_LOGIC;
           s05 : out STD_LOGIC;

           s12 : out STD_LOGIC;
           s13 : out STD_LOGIC;
           s14 : out STD_LOGIC;
           s15 : out STD_LOGIC;

           s22 : out STD_LOGIC;
           s23 : out STD_LOGIC;
           s24 : out STD_LOGIC;
```

```
          s25 : out STD_LOGIC;
         done : out STD_logic);
end controller;

architecture Behavioral of controller is
type state is (init,loadr,loadp,loadac,lshift,lshiftac,idle,loadc,result);
signal ps,ns : state;
shared variable cnt,count,cnt1:integer;
signal cnt2 : signed(10 downto 0);
begin
ns_logic:process(ps,clk)
begin
   case ps is
      when init=>
            s02<='0';
            s03<='0';
            s04<='0';
            s05<='0';

            s12<='0';
            s13<='0';
            s14<='0';
            s15<='0';

            s22<='0';
```

```
        s23<='0';
        s24<='0';
        s25<='0';
        cnt:=0;
        count:=63;
        done<='0';
        if start ='1' then


          s03<='0';
          s13<='0';
          s23<='1';
          ns<=loadr;

      else ns<=init;
      end if;
          if (clk='1' and clk'event) then
          ps<=ns;
          end if;
when loadr =>
      s03<='1';

      s13<='0';

      s23<='0';
```

```
            if (b="00000000000000000000000000000000") or
(b="10000000000000000000000000000000") or (a="00000000000000000000000000000000")
or (a="10000000000000000000000000000000")   then

              s05<='0';

              s15<='0';

              s25<='0';

              ns<=result;

            elsif b(cnt)='1' then

              s04<='0';

              s14<='0';

              s24<='1';

              ns<=loadp;

            else ns<=loadac;

           end if;

        if (clk='1' and clk'event) then

           ps<=ns;

        end if;

    when loadp => s04<='1';

           s14<='0';

           s24<='0';

           s02<='0';

           s12<='0';

           s22<='1';

           ns<=loadac;

              if (clk='1' and clk'event) then

              ps<=ns;
```

```
        end if;
when loadac => s02<='1';
        s12<='0';
        s22<='0';
        s03<='0';
        s13<='1';
        s23<='0';
        ns<=lshift;
           if (clk='1' and clk'event) then
           ps<=ns;
           cnt:=cnt+1;
           end if;
when lshift => s03<='1';
        s13<='0';
        s23<='0';
        if cnt=23 then
           s04<='0';
           s14<='0';
           s24<='1';
           ns<=loadp;
       elsif (cnt<23 and b(cnt)='1') then
              s04<='0';
              s14<='0';
              s24<='1';
              ns<=loadp;
```

```
        elsif (cnt<23 and b(cnt)='0') then
                ns<=loadac;
         else
         ns<=idle;
         end if;
             if (clk='1' and clk'event) then
         ps<=ns;
         end if;
when idle => if ac(count)='0'  then
               ns<=idle;
              if (clk='1' and clk'event) then
                 count := count-1;
                ps<=ns;
                end if;
             else
                 cnt := 52-count;
                 cnt1 := count-46;
                 cnt2 <= to_signed(cnt1,11);
                 w<= std_logic_vector(cnt2);
                  if cnt=0 then
                       s05<='0';
                       s15<='0';
                       s25<='1';
                         ns<=loadc;
```

```
                else s02<='0';
                    s12<='1';
                    s22<='0';
                       ns<=lshiftac;
                end if;
             if (clk='1' and clk'event) then
                count := count-1;
               ps<=ns;
            end if;
            end if;
when lshiftac =>  s02<='1';
             s12<='0';
             s22<='0';

             if cnt=1 then
                s05<='0';
                s15<='0';
                s25<='1';
                ns<=loadc;
             else s02<='0';
                s12<='1';
                s22<='0';
                ns<=lshiftac;
             end if;
             if (clk='1' and clk'event) then
```

```
                cnt :=cnt-1;
                ps<=ns;
                end if;
    when loadc => done<='1';
                ns<=init;
                   if (clk='1' and clk'event) then
                   ps<=ns;
                end if;
    when result=> done<='1';
                ns<=init;
                   if (clk='1' and clk'event) then
                   ps<=ns;
                end if;
    when others =>
                if (clk='1' and clk'event) then
                  ps<=init;
                 end if;
   end case;
end process;
end Behavioral;
```

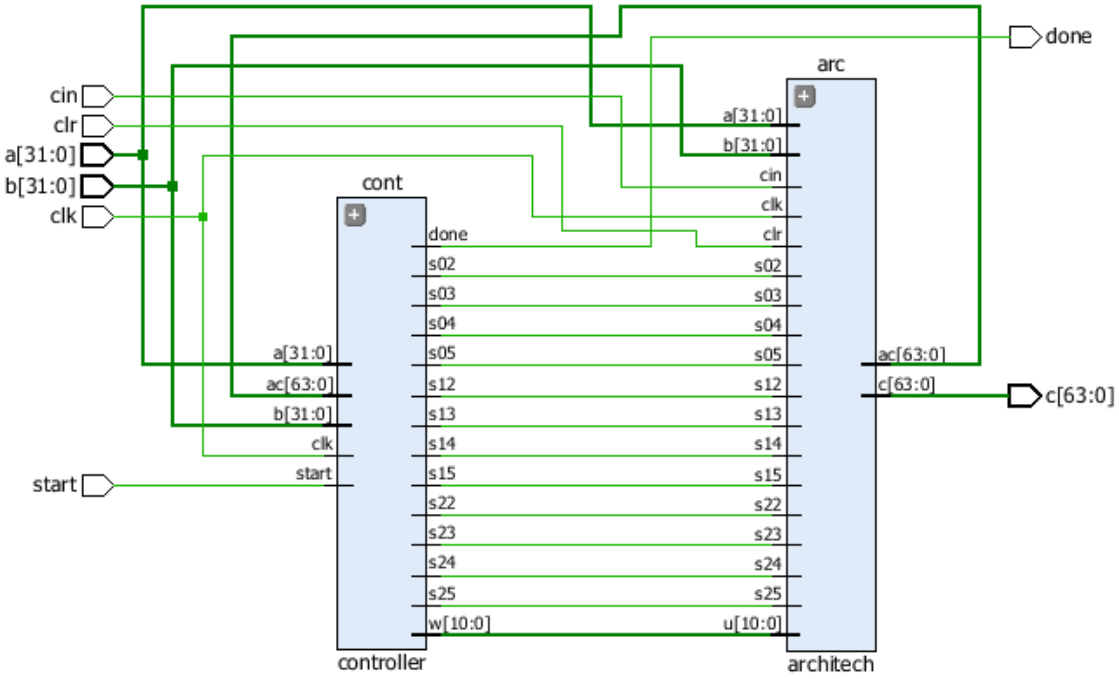
VHDL CODE FOR 32-BIT FLOATING POINT MULTIPLIER:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity fpm32 is
   Port ( a : in STD_LOGIC_VECTOR (31 downto 0);
          b : in STD_LOGIC_VECTOR (31 downto 0);
          c : out STD_LOGIC_VECTOR (63 downto 0);
          start : in STD_LOGIC;
          done : out STD_LOGIC;
          clk : in STD_LOGIC;
          cin : in STD_LOGIC;
          clr : in std_logic);
end fpm32;
architecture Behavioral of fpm32 is
component architech
   Port ( a : in STD_LOGIC_VECTOR (31 downto 0);
          b : in STD_LOGIC_VECTOR (31 downto 0);
          c : out STD_LOGIC_VECTOR (63 downto 0);
         ac : inout STD_LOGIC_VECTOR (63 downto 0);
          u :in std_logic_vector(10 downto 0);
          s02 : in STD_LOGIC;
          s12 : in STD_LOGIC;
          s22 : in STD_LOGIC;
          s03 : in STD_LOGIC;
          s13 : in STD_LOGIC;
```

```
        s23 : in STD_LOGIC;
        s04 : in STD_LOGIC;
        s14 : in STD_LOGIC;
        s24 : in STD_LOGIC;
        s05 : in STD_LOGIC;
        s15 : in STD_LOGIC;
        s25 : in STD_LOGIC;
        clk : in STD_LOGIC;
        clr : in STD_LOGIC;
        cin : in std_logic);
end component;
component controller
    Port ( a : in STD_LOGIC_VECTOR (31 downto 0);
        b : in STD_LOGIC_VECTOR (31 downto 0);
        ac : in STD_LOGIC_VECTOR (63 downto 0);
        clk : in STD_LOGIC;
        start : in STD_LOGIC;
         w: out std_logic_vector(10 downto 0);
        s02 : out STD_LOGIC;
        s03 : out STD_LOGIC;
        s04 : out STD_LOGIC;
        s05 : out STD_LOGIC;
        s12 : out STD_LOGIC;
        s13 : out STD_LOGIC;
        s14 : out STD_LOGIC;
```

```
        s15 : out STD_LOGIC;
        s22 : out STD_LOGIC;
        s23 : out STD_LOGIC;
        s24 : out STD_LOGIC;
        s25 : out STD_LOGIC;
        done : out STD_logic );
 end component;
 signal s02,s12,s22,s03,s13,s23,s04,s14,s24,s05,s15,s25:std_logic;
 signal ac :STD_LOGIC_VECTOR (63 downto 0);
 signal w:std_logic_vector(10 downto 0);
begin
arc: architech port map(a,b,c,ac,w,s02,s12,s22,s03,s13,s23,s04,s14,s24,s05,s15,s25,clk,clr,cin);
cont: controller port
map(a,b,ac,clk,start,w,s02,s03,s04,s05,s12,s13,s14,s15,s22,s23,s24,s25,done);
end Behavioral;
```

RTL Schematic:



simulation results: