DESIGN USING VHDL

The asynchronous CPU comprises of number of smaller functional blocks. The VHDL behavioral model for various functional blocks has been written & tested by simulation separately following a hierarchical approach .The smaller functional blocks have been grouped in to bigger functional blocks (say subsystems).

```
----------------------------------------------------------------------------increment_pc.vhd----------------------------------------------------------------------------
-------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity increment_pc is
port( reset: in std_logic;
                        req1  :     out std_logic;
                        req2:        in std_logic;
                        dout :      out std_logic_vector(9 downto 0));
end increment_pc;

architecture Behavioral of increment_pc is

component lso is
port( reset  :                      in std_logic;
                        aclk        :                       out std_logic);
end component;

component Counter  is
port( reset, clk        : in std_logic;
                        dout                            : out std_logic);
end component;

component Adder_in_pc is
PORT(A:             in std_logic_vector(9 downto 0);
            B:                  in std_logic_vector(9 downto 0);
            reset: in std_logic;
            Cin:    in std_logic;
            Cout:   out std_logic;
            O:                  out std_logic_vector(9 downto 0));
end component;

component ecr is
port( din : in std_logic_vector (9 downto 0);
     req : in std_logic ;
                         ack : out std_logic ;
     dout : out std_logic_vector(9 downto 0));
end component;
```

```
signal s0, s1, s2, s3, s4, ack: std_logic;
signal d1, d2: std_logic_vector(9 downto 0);

begin
req1<= s0;
s1<= not s0;
s2 <= reset and s1;
s4 <= s0 and req2;
dout <= d1;

U1: lso port map( reset => s2, aclk => s3);
U2: Counter port map( reset => ack, clk => s3, dout => s0);
U3: Adder_in_pc port map( reset => reset, A => d1, B => "0000000000", Cin => '1', O => d2);
U4: ecr port map( din => d2, req=> s4, ack => ack, dout => d1);
end Behavioral;
```

----------File: adder_IN_PC.vhd----------

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Adder_in_pc is
PORT(A:         in std_logic_vector(9 downto 0);
        B:              in std_logic_vector(9 downto 0);
        reset: in std_logic;
        Cin:    in std_logic;
        Cout:   out std_logic;
        O:              out std_logic_vector(9 downto 0));
end Adder_in_pc;

architecture Behavioral of Adder_in_pc is
component FourBitAdder is
Port(A,B: in std_logic_vector(3 downto 0);
                Cin: in std_logic;
                Cout: out std_logic;
                O: out std_logic_vector(3 downto 0));

end component;

component TwoBitAdder is
Port(A,B: in std_logic_vector(1 downto 0);
Cin: in std_logic;
Cout: out std_logic;
O: out std_logic_vector(1 downto 0));
end component;

signal s1, s2: std_logic;
signal s4: std_logic_vector(9 downto 0);
begin
U1: FourBitAdder port map(A => A(3 downto 0), B=> B(3 downto 0), Cin=> Cin, Cout => s1, O => s4( 3 downto 0));
U2: FourBitAdder port map(A => A(7 downto 4), B=> B(7 downto 4), Cin=> s1, Cout => s2, O => s4( 7 downto 4));
U3: TwoBitAdder port map(A => A(9 downto 8), B=> B(9 downto 8), Cin=> s2, Cout => Cout, O => s4( 9 downto 8));

O <= s4 when reset = '1' else
```

```
                "0000000000" when reset = '0';

end Behavioral;
```
----------File: ecr.vhd----------
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ecr is
port( din : in std_logic_vector(9 downto 0);
      req : in std_logic ;
                          ack : out std_logic ;
      dout : out std_logic_vector(9 downto 0));
end ecr;

architecture Behavioral of ecr is

component OR_gate is
port( inp : in std_logic_vector(19 downto 0);
                          oup: out std_logic);
end component;

component encoder is
port(inp: in std_logic_vector (8 downto 0) ;
    oup : out std_logic_vector(1 downto 0));
end component;

component and_gate is
port( inp : in std_logic;
                          oup: out std_logic);
end component;

component inv_gate is
port( inp : in std_logic;
                          oup: out std_logic);
end component;

component d_ff is
port(inp : in std_logic_vector(9 downto 0);
                oup: out std_logic_vector(9 downto 0);
                clk: in std_logic);
end component;

type dataout is array (9 downto 0) of std_logic_vector(1 downto 0);
signal s1: std_logic;
signal s2: std_logic_vector(9 downto 0);
signal s3: dataout;
signal s4: std_logic;

begin
u1: inv_gate port map( inp=>req, oup=>s1);
u2: inv_gate port map( inp=>din(0), oup=>s2(0));
```

```
u3: inv_gate port map( inp=>din(1), oup=>s2(1));
u4: inv_gate port map( inp=>din(2), oup=>s2(2));
u5: inv_gate port map( inp=>din(3), oup=>s2(3));
u6: inv_gate port map( inp=>din(4), oup=>s2(4));
u7: inv_gate port map( inp=>din(5), oup=>s2(5));
u8: inv_gate port map( inp=>din(6), oup=>s2(6));
u9: inv_gate port map( inp=>din(7), oup=>s2(7));
u10: inv_gate port map( inp=>din(8), oup=>s2(8));
u11: inv_gate port map( inp=>din(9), oup=>s2(9));
u14: encoder port map ( inp(0)=>s1, inp(1)=>din(0), inp(2)=>din(0), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(0), inp(7)=>s2(0),
inp(8)=>req, oup(0)=> s3(0)(0), oup(1) => s3(0)(1));
u15: encoder port map ( inp(0)=>s1, inp(1)=>din(1), inp(2)=>din(1), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(1), inp(7)=>s2(1),
inp(8)=>req, oup(0)=> s3(1)(0), oup(1) => s3(1)(1));
u16: encoder port map ( inp(0)=>s1, inp(1)=>din(2), inp(2)=>din(2), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(2), inp(7)=>s2(2),
inp(8)=>req, oup(0)=> s3(2)(0), oup(1) => s3(2)(1));
u17: encoder port map ( inp(0)=>s1, inp(1)=>din(3), inp(2)=>din(3), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(3), inp(7)=>s2(3),
inp(8)=>req, oup(0)=> s3(3)(0), oup(1) => s3(3)(1));
u18: encoder port map ( inp(0)=>s1, inp(1)=>din(4), inp(2)=>din(4), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(4), inp(7)=>s2(4),
inp(8)=>req, oup(0)=> s3(4)(0), oup(1) => s3(4)(1));
u19: encoder port map ( inp(0)=>s1, inp(1)=>din(5), inp(2)=>din(5), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(5), inp(7)=>s2(5),
inp(8)=>req, oup(0)=> s3(5)(0), oup(1) => s3(5)(1));
u20: encoder port map ( inp(0)=>s1, inp(1)=>din(6), inp(2)=>din(6), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(6), inp(7)=>s2(6),
inp(8)=>req, oup(0)=> s3(6)(0), oup(1) => s3(6)(1));
u21: encoder port map ( inp(0)=>s1, inp(1)=>din(7), inp(2)=>din(7), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(7), inp(7)=>s2(7),
inp(8)=>req, oup(0)=> s3(7)(0), oup(1) => s3(7)(1));
u22: encoder port map ( inp(0)=>s1, inp(1)=>din(8), inp(2)=>din(8), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(8), inp(7)=>s2(8),
inp(8)=>req, oup(0)=> s3(8)(0), oup(1) => s3(8)(1));
u23: encoder port map ( inp(0)=>s1, inp(1)=>din(9), inp(2)=>din(9), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(9), inp(7)=>s2(9),
inp(8)=>req, oup(0)=> s3(9)(0), oup(1) => s3(9)(1));
u26: OR_gate port map ( inp(0)=>s3(0)(0),inp(1)=>s3(0)(1), inp(2)=>s3(1)(0), inp(3)=>s3(1)(1), inp(4)=>s3(2)(0), inp(5)=>s3(2)(1),
inp(6)=>s3(3)(0), inp(7)=>s3(3)(1), inp(8)=>s3(4)(0), inp(9)=>s3(4)(1), inp(10)=>s3(5)(0),inp(11)=>s3(5)(1), inp(12)=>s3(6)(0),
inp(13)=>s3(6)(1),inp(14)=>s3(7)(0),inp(15)=>s3(7)(1), inp(16) => s3(8)(0), inp(17)=> s3(8)(1), inp(18) => s3(9)(0),inp(19)=>s3(9)(1),
oup =>s4);
u27: d_ff port map(inp(0)=> s3(0)(0), inp(1)=> s3(1)(0), inp(2)=> s3(2)(0), inp(3)=> s3(3)(0), inp(4)=> s3(4)(0), inp(5)=>s3(5)(0),
inp(6)=>s3(6)(0), inp(7)=>s3(7)(0), inp(8)=>s3(8)(0), inp(9)=>s3(9)(0), oup=> dout, clk=> s4);
u28: inv_gate port map(inp=> s4, oup=> ack);

end Behavioral;
```

----------File: OR_gate.vhd----------

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity OR_gate is
port( inp : in std_logic_vector(19 downto 0);
                        oup: out std_logic);
end OR_gate;

architecture Behavioral of OR_gate is

begin
oup <= inp(0) or inp(1) or inp(2)or inp(3)or inp(4)or inp(5)or inp(6)or inp(7)or inp(8)or inp(9)or inp(10)or inp(11)or inp(12)or inp(13)or
inp(14)or inp(15)or inp(16)or inp(17)or inp(18)or inp(19);
```

```
end Behavioral;
```
----------**File: d_ff.vhd**----------

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity d_ff is
port(inp : in std_logic_vector(9 downto 0);
                oup: out std_logic_vector(9 downto 0);
                clk: in std_logic);
end d_ff;

architecture Behavioral of d_ff is

begin

process(clk)
begin
if(clk'event and clk = '1') then
oup <= inp;
end if;
end process;

end Behavioral;
```
----------**File: inv_gate.vhd**----------

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity inv_gate is
port( inp : in std_logic;
                        oup : out std_logic);
end inv_gate;

architecture Behavioral of inv_gate is

begin
oup <= not inp;

end Behavioral;
```

----------File: **execute**----------

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

entity execute is
port( instr  :          in std_logic_vector(15 downto 0);
                        ext_din: in std_logic_vector(15 downto 0);
                        jmp     :       out std_logic;
                        addr    :       out std_logic_vector(9 downto 0);
                        req1  :     out std_logic;
                        req2    :       in std_logic;
                        reset   :       in std_logic);
end execute;

architecture Behavioral of execute is

component decoder is
port( reset :           in std_logic;
                        din     :  in std_logic_vector(15 downto 0);
                        inp_a :     in std_logic_vector(15 downto 0);
                        inp_b :     in std_logic_vector(15 downto 0);
                        inp_c :  in std_logic;
                        zf                      :       in std_logic;
                        req1  :     out std_logic;
                        req2    :       in std_logic;
                        dout  :     out std_logic_vector(59 downto 0));
end component;

component ram  is
port( reset :           in std_logic;
                        din     :  in std_logic_vector(59 downto 0);
                        mem_e :   in std_logic;
                        addr_out  :       out std_logic_vector(9 downto 0);
                        rd      :       in std_logic;
                        wr      :  in std_logic;
                        db_sel:   in std_logic_vector(2 downto 0);
                        req1  :     out std_logic;
                        req2    :       in std_logic;
                        dout  :     out std_logic_vector(51 downto 0));
end component;

component Multiplex1 is
port(       din1        : in std_logic_vector(15 downto 0);
                        din2        : in std_logic_vector(15 downto 0);
                        dout        : out std_logic_vector(15 downto 0);
                        sel         : in std_logic);
end component;

component alu is
port(       reset :     in std_logic;
                        din     :  in std_logic_vector(58 downto 0);
                        db_in   :       in std_logic_vector(15 downto 0);
                        rd                      :       out std_logic;
                        wr                      :       out std_logic;
                        mem_en:   out std_logic;
                        db_sel:   out std_logic_vector(2 downto 0);
                        ld_a    :       out std_logic;
                        ld_b    :       out std_logic;
                        ld_acc:   out std_logic;
                        ld_f    :       out std_logic;
                        req1  :     out std_logic;
                        req2    :       in std_logic;
                        dout  :     out std_logic_vector(59 downto 0));
```

```
end component;

component databus is
port(A: in std_logic_vector(15 downto 0);
        B: in std_logic_vector(15 downto 0);
        ACC: in std_logic_vector(15 downto 0);
        Flag: in std_logic_vector(15 downto 0);
        Inp     : in std_logic_vector(15 downto 0);
        sel: in std_logic_vector(2 downto 0);
        data: out std_logic_vector(15 downto 0));
end component;

component Reg_Set is
port( din_A                     : in std_logic_vector(15 downto 0);
            din_ACC             : in std_logic_vector(15 downto 0);
            din_B                       : in std_logic_vector(15 downto 0);
            din_F               : in std_logic_vector(15 downto 0);
            dout_A              : out std_logic_vector(15 downto 0);
            dout_ACC            : out std_logic_vector(15 downto 0);
            dout_B              : out std_logic_vector(15 downto 0);
            dout_F              : out std_logic_vector(15 downto 0);
            ld_flags            : in std_logic;
            flags                       : in std_logic_vector(2 downto 0);
            ld_A                        : in std_logic;
            ld_ACC              : in std_logic;
            ld_B                        : in std_logic;
            ld_F                        : in std_logic);
end component;

signal a_out, b_out, db_out, acc_out, f_out: std_logic_vector(15 downto 0);
signal cin, req_alu, req_dec: std_logic;
signal dec_out: std_logic_vector(58 downto 0);
signal alu_out: std_logic_vector(59 downto 0);
signal ram_out: std_logic_vector(51 downto 0);
signal rd, wr, mem_en, ld_a, ld_b, ld_acc, ld_f: std_logic;
signal zero_flag:std_logic;
signal db_sel: std_logic_vector(2 downto 0);
signal a_in, acc_in : std_logic_vector(15 downto 0);
begin

U1: decoder port map(reset => reset, zf => zero_flag, din => instr, inp_a => a_out, inp_b => b_out, inp_c => cin, req1 => req1, req2 =>
req_dec, dout(59) => jmp, dout(58 downto 0) => dec_out);
U2: alu port map(reset => reset, din => dec_out, db_in => db_out, rd => rd, wr => wr, mem_en => mem_en, db_sel => db_sel, ld_a =>
ld_a, ld_b => ld_b, ld_acc => ld_acc, ld_f => ld_f, req1 => req_dec, req2 => req_alu, dout => alu_out);
U3: ram port map(reset => reset, addr_out => addr,  din => alu_out, rd => rd, wr => wr, mem_e => mem_en, db_sel => db_sel, req1 =>
req_alu, req2 => req2, dout => ram_out);
U4: Reg_set port map( din_A => a_in, din_B => ram_out(15 downto 0), din_ACC => acc_in, din_F => ram_out(15 downto 0), dout_A =>
a_out, dout_B => b_out, dout_ACC => acc_out, dout_F => f_out, ld_A => ld_a, ld_B => ld_b, ld_ACC => ld_acc, ld_F => ld_f, flags =>
ram_out(18 downto 16), ld_flags => ram_out(51));
U5: databus port map( A => a_out, B => b_out, ACC => acc_out, Flag => f_out, Inp => ext_din, sel => db_sel, data => db_out);
U6: Multiplex1 port map( din1 => ram_out(50 downto 35), din2 => ram_out(15 downto 0), sel => ram_out(51), dout => a_in);
U7: Multiplex1 port map( din1 => ram_out(34 downto 19), din2 => ram_out(15 downto 0), sel => ram_out(51), dout => acc_in);

zero_flag <= f_out(1);
cin <= f_out(0);

end Behavioral;
----------File : decoder----------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
```

```
entity decoder is
port( reset :            in std_logic;
                         din          :  in std_logic_vector(15 downto 0);
                         inp_a :      in std_logic_vector(15 downto 0);
                         inp_b :      in std_logic_vector(15 downto 0);
                         inp_c :  in std_logic;
                         zf                    :              in std_logic;
                         req1  :      out std_logic;
                         req2         :              in std_logic;
                         dout  :      out std_logic_vector(59 downto 0));
end decoder;

architecture Behavioral of decoder is

component lso is
port( reset  :                   in std_logic;
                         aclk         :                         out std_logic);
end component;

component Counter  is
port( reset, clk         : in std_logic;
                         dout                              : out std_logic);
end component;

component logic1 is
port(instr               :           in std_logic_vector(15 downto 0);
            db_sel               :           out std_logic_vector(2 downto 0);
            ram_addr :           out std_logic_vector(9 downto 0);
            ram_e                :           out std_logic;
            rd                                :              out std_logic;
            wr                                :  out std_logic;
            alu_e                :           out std_logic;
            alu_sel  :           out std_logic_vector(4 downto 0);
            jmp                               :              out std_logic;
            cf                                :              in std_logic;
            zf                                :              in std_logic;
            ld_a                 :  out std_logic;
            ld_acc               :  out std_logic;
            ld_b                 :  out std_logic;
            ld_f                 :  out std_logic);
end component;

component ecr1 is
port(  din : in std_logic_vector (59 downto 0);
      req      : in std_logic ;
                        ack          : out std_logic ;
      dout : out std_logic_vector (59 downto 0));
end component;

signal s0, s1, s2, s3, s4, ack: std_logic;
signal d1: std_logic_vector(25 downto 0);
signal jmp1: std_logic;
begin
req1<= s0;
s1<= not s0;
s2 <= reset and s1;
s4 <= s0 and req2;

U1: lso port map( reset => s2, aclk => s3);
U2: Counter port map( reset => ack, clk => s3, dout => s0);
```

```
U3: logic1 port map( instr => din, jmp => jmp1, cf => inp_c, zf => zf, db_sel => d1(25 downto 23), ram_addr => d1(22 downto 13), ram_e
=> d1(12), rd => d1(11), wr => d1(10), alu_e => d1(9), alu_sel => d1(8 downto 4), ld_a => d1(3), ld_acc => d1(2), ld_b => d1(1), ld_f =>
d1(0));
U4: ecr1 port map( din(59) => jmp1, din(58 downto 33) => d1, din(32 downto 17) => inp_a, din(16 downto 1) => inp_b, din(0) => inp_c,
req=> s4, ack => ack, dout => dout);
end Behavioral;

`timescale 1ns / 1ps
module lso(aclk, reset);
input reset; output aclk;
reg aclk1, a1, ack; wire osc, a2;
siro s1(osc, reset);
always @(posedge osc or negedge reset)
begin
a1 = ~aclk;
if(osc ==1)
begin
if(reset==0)
begin
aclk1 = 0;
a1 = 1;
ack = 0;
end
if(ack == 1) aclk1=~aclk;
if(a1!= a2) ack =1; else ack = 0; end
end assign a2 = aclk1; assign aclk = aclk1;
endmodule
```

----------file: **lso**----------

```
module lso(aclk, reset);
input reset; output aclk;
reg aclk1, a1, ack; wire osc, a2;
siro s1(osc, reset);
always @(posedge osc or negedge reset)
begin
a1 = ~aclk;
if(osc ==1)
begin
if(reset==0)
begin
aclk1 = 0;
a1 = 1;
ack = 0;
end
if(ack == 1) aclk1=~aclk;
if(a1!= a2) ack =1; else ack = 0; end
end assign a2 = aclk1; assign aclk = aclk1;
endmodule
```

----------file : **siro**----------

```
module siro(osc, reset);
input reset; output osc;
wire osc1;
nand n1(osc, reset, osc1);
assign osc1 = osc;
endmodule
```

----------file: **counter**----------

```
entity Counter is
generic(N: integer := 8);
port( reset, clk            : in std_logic;
            dout: out std_logic);
end Counter;

architecture Behavioral of Counter is
begin

process(clk, reset)
variable s: integer := 0;
begin
if (reset = '0') then
            if (clk'event and clk = '1') then
                        s := s+1;
                        if( s = N) then
                                    s := 0;
                                    dout<='1';
                        else
                                    dout<='0';
                        end if;
            end if;
elsif (reset = '1') then
            s := 0;
            dout <= '0';
end if;
end process;

--process(s, clk)
--begin
--if(clk'event and clk = '1' and s = N) then
--            dout <= '1';
--else
--            dout <= '0';
--end if;
--end process;
end Behavioral;
```

----------file : **logic1**----------

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


entity logic1 is
port(instr                  :           in std_logic_vector(15 downto 0);
            db_sel                  :           out std_logic_vector(2 downto 0);
            ram_addr :          out std_logic_vector(9 downto 0);
            ram_e                   :           out std_logic;
            rd                                  :           out std_logic;
            wr                                  :  out std_logic;
            alu_e                   :           out std_logic;
            alu_sel     :           out std_logic_vector(4 downto 0);
            ld_a                    :  out std_logic;
            ld_acc                  :  out std_logic;
            ld_b                    :  out std_logic;
            jmp                                 :           out std_logic;
            zf                                  :           in std_logic;
            cf                                  :           in std_logic;
            ld_f                    :  out std_logic);
end logic1;

architecture Behavioral of logic1 is
```

```
alias opcode: std_logic_vector(5 downto 0) is instr(15 downto 10);
alias alu_op: std_logic_vector(4 downto 0) is instr(15 downto 11);
alias mem_en: std_logic is instr(10);
alias alu_en: std_logic is instr(14);
alias addr:   std_logic_vector(9 downto 0) is instr(9 downto 0);

begin
ram_addr <= addr;
ram_e <= mem_en;
alu_e <= alu_en;
alu_sel <= alu_op;

jmp <= '1' when opcode = "110011" or (opcode = "110101" and cf = '1') or (opcode = "110111" and zf = '1') else
                                '0';

ld_acc <= '1' when alu_en = '1' or opcode = "010001" or opcode = "010000" or opcode = "010010" or opcode = "011101" or opcode =
"011111" else
                                        '0';

--insert here opcode for MUL instruction
ld_a          <= '1' when opcode = "011011" or opcode = "010100" or opcode = "011000" else
                                        '0';

ld_b          <= '1' when opcode = "011101" or opcode = "010110" or opcode = "011010" else
                                        '0';

db_sel <= "000" when opcode = "010111" or opcode = "010000" else
                                        "001" when opcode = "011001" or opcode = "010010" else
                                        "010" when opcode = "010011" or opcode = "010100" or opcode = "010110" or opcode = "110001" else
                                        "011" when opcode = "011111" else
                                        "100" when opcode = "011011" or opcode = "011101" or opcode = "010001" else
                                        "101" when opcode = "010101" or opcode = "011010" or opcode = "011010" or opcode = "011101";

rd <= '1' when opcode = "010001" or opcode = "011101" or opcode = "011011" else
                               '0';

wr <= '1' when opcode = "010011" or opcode = "010101" or opcode = "010111" or opcode = "011001" else
                               '0';

ld_f <= '1' when opcode = "110001" else
                                '0';


end Behavioral;
------------------------------file :ecr1----------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;

entity ecr1 is
port( din : in std_logic_vector(59 downto 0);
      req : in std_logic ;
                              ack : out std_logic ;
      dout : out std_logic_vector(59 downto 0));
end ecr1;

architecture Behavioral of ecr1 is

component OR_gate1 is
port( inp : in std_logic_vector(119 downto 0);
```

```
                        oup: out std_logic);
end component;

component encoder is
port(inp: in std_logic_vector (8 downto 0) ;
    oup : out std_logic_vector(1 downto 0));
end component;

component and_gate is
port( inp : in std_logic;
                        oup: out std_logic);
end component;

component inv_gate is
port( inp : in std_logic;
                        oup: out std_logic);
end component;

component d_ff1 is
port(inp : in std_logic_vector(59 downto 0);
            oup: out std_logic_vector(59 downto 0);
            clk: in std_logic);
end component;

type dataout is array (59 downto 0) of std_logic_vector(1 downto 0);
signal s1: std_logic;
signal s2: std_logic_vector(59 downto 0);
signal s3: dataout;
signal s4: std_logic;

begin
u1: inv_gate port map( inp=>req, oup=>s1);
u2: inv_gate port map( inp=>din(0), oup=>s2(0));
u3: inv_gate port map( inp=>din(1), oup=>s2(1));
u4: inv_gate port map( inp=>din(2), oup=>s2(2));
u5: inv_gate port map( inp=>din(3), oup=>s2(3));
u6: inv_gate port map( inp=>din(4), oup=>s2(4));
u7: inv_gate port map( inp=>din(5), oup=>s2(5));
u8: inv_gate port map( inp=>din(6), oup=>s2(6));
u9: inv_gate port map( inp=>din(7), oup=>s2(7));
u10: inv_gate port map( inp=>din(8), oup=>s2(8));
u11: inv_gate port map( inp=>din(9), oup=>s2(9));
u12: inv_gate port map( inp=>din(10), oup=>s2(10));
u13: inv_gate port map( inp=>din(11), oup=>s2(11));
u14: inv_gate port map( inp=>din(12), oup=>s2(12));
u15: inv_gate port map( inp=>din(13), oup=>s2(13));
u16: inv_gate port map( inp=>din(14), oup=>s2(14));
u17: inv_gate port map( inp=>din(15), oup=>s2(15));
u18: inv_gate port map( inp=>din(16), oup=>s2(16));
u19: inv_gate port map( inp=>din(17), oup=>s2(17));
u20: inv_gate port map( inp=>din(18), oup=>s2(18));
u21: inv_gate port map( inp=>din(19), oup=>s2(19));
u22: inv_gate port map( inp=>din(20), oup=>s2(20));
u23: inv_gate port map( inp=>din(21), oup=>s2(21));
u24: inv_gate port map( inp=>din(22), oup=>s2(22));
u25: inv_gate port map( inp=>din(23), oup=>s2(23));
u26: inv_gate port map( inp=>din(24), oup=>s2(24));
u27: inv_gate port map( inp=>din(25), oup=>s2(25));
u28: inv_gate port map( inp=>din(26), oup=>s2(26));
u29: inv_gate port map( inp=>din(27), oup=>s2(27));
u30: inv_gate port map( inp=>din(28), oup=>s2(28));
u31: inv_gate port map( inp=>din(29), oup=>s2(29));
```

```
u32: inv_gate port map( inp=>din(30), oup=>s2(30));
u33: inv_gate port map( inp=>din(31), oup=>s2(31));
u34: inv_gate port map( inp=>din(32), oup=>s2(32));
u35: inv_gate port map( inp=>din(33), oup=>s2(33));
u36: inv_gate port map( inp=>din(34), oup=>s2(34));
u37: inv_gate port map( inp=>din(35), oup=>s2(35));
u38: inv_gate port map( inp=>din(36), oup=>s2(36));
u39: inv_gate port map( inp=>din(37), oup=>s2(37));
u40: inv_gate port map( inp=>din(38), oup=>s2(38));
u41: inv_gate port map( inp=>din(39), oup=>s2(39));
u42: inv_gate port map( inp=>din(40), oup=>s2(40));
u43: inv_gate port map( inp=>din(41), oup=>s2(41));
u44: inv_gate port map( inp=>din(42), oup=>s2(42));
u45: inv_gate port map( inp=>din(43), oup=>s2(43));
u46: inv_gate port map( inp=>din(44), oup=>s2(44));
u47: inv_gate port map( inp=>din(45), oup=>s2(45));
u48: inv_gate port map( inp=>din(46), oup=>s2(46));
u49: inv_gate port map( inp=>din(47), oup=>s2(47));
u50: inv_gate port map( inp=>din(48), oup=>s2(48));
u51: inv_gate port map( inp=>din(49), oup=>s2(49));
u52: inv_gate port map( inp=>din(50), oup=>s2(50));
u53: inv_gate port map( inp=>din(51), oup=>s2(51));
u54: inv_gate port map( inp=>din(52), oup=>s2(52));
u55: inv_gate port map( inp=>din(53), oup=>s2(53));
u56: inv_gate port map( inp=>din(54), oup=>s2(54));
u57: inv_gate port map( inp=>din(55), oup=>s2(55));
u58: inv_gate port map( inp=>din(56), oup=>s2(56));
u59: inv_gate port map( inp=>din(57), oup=>s2(57));
u60: inv_gate port map( inp=>din(58), oup=>s2(58));

u122: inv_gate port map( inp=>din(59), oup=>s2(59));

u61: encoder port map ( inp(0)=>s1, inp(1)=>din(0), inp(2)=>din(0), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(0), inp(7)=>s2(0),
inp(8)=>req, oup(0)=> s3(0)(0), oup(1) => s3(0)(1));
u62: encoder port map ( inp(0)=>s1, inp(1)=>din(1), inp(2)=>din(1), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(1), inp(7)=>s2(1),
inp(8)=>req, oup(0)=> s3(1)(0), oup(1) => s3(1)(1));
u63: encoder port map ( inp(0)=>s1, inp(1)=>din(2), inp(2)=>din(2), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(2), inp(7)=>s2(2),
inp(8)=>req, oup(0)=> s3(2)(0), oup(1) => s3(2)(1));
u64: encoder port map ( inp(0)=>s1, inp(1)=>din(3), inp(2)=>din(3), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(3), inp(7)=>s2(3),
inp(8)=>req, oup(0)=> s3(3)(0), oup(1) => s3(3)(1));
u65: encoder port map ( inp(0)=>s1, inp(1)=>din(4), inp(2)=>din(4), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(4), inp(7)=>s2(4),
inp(8)=>req, oup(0)=> s3(4)(0), oup(1) => s3(4)(1));
u66: encoder port map ( inp(0)=>s1, inp(1)=>din(5), inp(2)=>din(5), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(5), inp(7)=>s2(5),
inp(8)=>req, oup(0)=> s3(5)(0), oup(1) => s3(5)(1));
u67: encoder port map ( inp(0)=>s1, inp(1)=>din(6), inp(2)=>din(6), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(6), inp(7)=>s2(6),
inp(8)=>req, oup(0)=> s3(6)(0), oup(1) => s3(6)(1));
u68: encoder port map ( inp(0)=>s1, inp(1)=>din(7), inp(2)=>din(7), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(7), inp(7)=>s2(7),
inp(8)=>req, oup(0)=> s3(7)(0), oup(1) => s3(7)(1));
u69: encoder port map ( inp(0)=>s1, inp(1)=>din(8), inp(2)=>din(8), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(8), inp(7)=>s2(8),
inp(8)=>req, oup(0)=> s3(8)(0), oup(1) => s3(8)(1));
u70: encoder port map ( inp(0)=>s1, inp(1)=>din(9), inp(2)=>din(9), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(9), inp(7)=>s2(9),
inp(8)=>req, oup(0)=> s3(9)(0), oup(1) => s3(9)(1));
u71: encoder port map ( inp(0)=>s1, inp(1)=>din(10), inp(2)=>din(10), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(10),
inp(7)=>s2(10), inp(8)=>req, oup(0)=> s3(10)(0), oup(1) => s3(10)(1));
u72: encoder port map ( inp(0)=>s1, inp(1)=>din(11), inp(2)=>din(11), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(11),
inp(7)=>s2(11), inp(8)=>req, oup(0)=> s3(11)(0), oup(1) => s3(11)(1));
u73: encoder port map ( inp(0)=>s1, inp(1)=>din(12), inp(2)=>din(12), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(12),
inp(7)=>s2(12), inp(8)=>req, oup(0)=> s3(12)(0), oup(1) => s3(12)(1));
u74: encoder port map ( inp(0)=>s1, inp(1)=>din(13), inp(2)=>din(13), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(13),
inp(7)=>s2(13), inp(8)=>req, oup(0)=> s3(13)(0), oup(1) => s3(13)(1));
u75: encoder port map ( inp(0)=>s1, inp(1)=>din(14), inp(2)=>din(14), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(14),
inp(7)=>s2(14), inp(8)=>req, oup(0)=> s3(14)(0), oup(1) => s3(14)(1));
```

u76: encoder port map (inp(0)=>s1, inp(1)=>din(15), inp(2)=>din(15), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(15),
inp(7)=>s2(15), inp(8)=>req, oup(0)=> s3(15)(0), oup(1) => s3(15)(1));
u77: encoder port map (inp(0)=>s1, inp(1)=>din(16), inp(2)=>din(16), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(16),
inp(7)=>s2(16), inp(8)=>req, oup(0)=> s3(16)(0), oup(1) => s3(16)(1));
u78: encoder port map (inp(0)=>s1, inp(1)=>din(17), inp(2)=>din(17), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(17),
inp(7)=>s2(17), inp(8)=>req, oup(0)=> s3(17)(0), oup(1) => s3(17)(1));
u79: encoder port map (inp(0)=>s1, inp(1)=>din(18), inp(2)=>din(18), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(18),
inp(7)=>s2(18), inp(8)=>req, oup(0)=> s3(18)(0), oup(1) => s3(18)(1));
u80: encoder port map (inp(0)=>s1, inp(1)=>din(19), inp(2)=>din(19), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(19),
inp(7)=>s2(19), inp(8)=>req, oup(0)=> s3(19)(0), oup(1) => s3(19)(1));
u81: encoder port map (inp(0)=>s1, inp(1)=>din(20), inp(2)=>din(20), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(20),
inp(7)=>s2(20), inp(8)=>req, oup(0)=> s3(20)(0), oup(1) => s3(20)(1));
u82: encoder port map (inp(0)=>s1, inp(1)=>din(21), inp(2)=>din(21), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(21),
inp(7)=>s2(21), inp(8)=>req, oup(0)=> s3(21)(0), oup(1) => s3(21)(1));
u83: encoder port map (inp(0)=>s1, inp(1)=>din(22), inp(2)=>din(22), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(22),
inp(7)=>s2(22), inp(8)=>req, oup(0)=> s3(22)(0), oup(1) => s3(22)(1));
u84: encoder port map (inp(0)=>s1, inp(1)=>din(23), inp(2)=>din(23), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(23),
inp(7)=>s2(23), inp(8)=>req, oup(0)=> s3(23)(0), oup(1) => s3(23)(1));
u85: encoder port map (inp(0)=>s1, inp(1)=>din(24), inp(2)=>din(24), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(24),
inp(7)=>s2(24), inp(8)=>req, oup(0)=> s3(24)(0), oup(1) => s3(24)(1));
u86: encoder port map (inp(0)=>s1, inp(1)=>din(25), inp(2)=>din(25), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(25),
inp(7)=>s2(25), inp(8)=>req, oup(0)=> s3(25)(0), oup(1) => s3(25)(1));
u124: encoder port map (inp(0)=>s1, inp(1)=>din(26), inp(2)=>din(26), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(26),
inp(7)=>s2(26), inp(8)=>req, oup(0)=> s3(26)(0), oup(1) => s3(26)(1));
u87: encoder port map (inp(0)=>s1, inp(1)=>din(27), inp(2)=>din(27), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(27),
inp(7)=>s2(27), inp(8)=>req, oup(0)=> s3(27)(0), oup(1) => s3(27)(1));
u88: encoder port map (inp(0)=>s1, inp(1)=>din(28), inp(2)=>din(28), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(28),
inp(7)=>s2(28), inp(8)=>req, oup(0)=> s3(28)(0), oup(1) => s3(28)(1));
u89: encoder port map (inp(0)=>s1, inp(1)=>din(29), inp(2)=>din(29), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(29),
inp(7)=>s2(29), inp(8)=>req, oup(0)=> s3(29)(0), oup(1) => s3(29)(1));
u90: encoder port map (inp(0)=>s1, inp(1)=>din(30), inp(2)=>din(30), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(30),
inp(7)=>s2(30), inp(8)=>req, oup(0)=> s3(30)(0), oup(1) => s3(30)(1));
u91: encoder port map (inp(0)=>s1, inp(1)=>din(31), inp(2)=>din(31), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(31),
inp(7)=>s2(31), inp(8)=>req, oup(0)=> s3(31)(0), oup(1) => s3(31)(1));
u92: encoder port map (inp(0)=>s1, inp(1)=>din(32), inp(2)=>din(32), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(32),
inp(7)=>s2(32), inp(8)=>req, oup(0)=> s3(32)(0), oup(1) => s3(32)(1));
u93: encoder port map (inp(0)=>s1, inp(1)=>din(33), inp(2)=>din(33), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(33),
inp(7)=>s2(33), inp(8)=>req, oup(0)=> s3(33)(0), oup(1) => s3(33)(1));
u94: encoder port map (inp(0)=>s1, inp(1)=>din(34), inp(2)=>din(34), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(34),
inp(7)=>s2(34), inp(8)=>req, oup(0)=> s3(34)(0), oup(1) => s3(34)(1));
u95: encoder port map (inp(0)=>s1, inp(1)=>din(35), inp(2)=>din(35), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(35),
inp(7)=>s2(35), inp(8)=>req, oup(0)=> s3(35)(0), oup(1) => s3(35)(1));
u96: encoder port map (inp(0)=>s1, inp(1)=>din(36), inp(2)=>din(36), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(36),
inp(7)=>s2(36), inp(8)=>req, oup(0)=> s3(36)(0), oup(1) => s3(36)(1));
u97: encoder port map (inp(0)=>s1, inp(1)=>din(37), inp(2)=>din(37), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(37),
inp(7)=>s2(37), inp(8)=>req, oup(0)=> s3(37)(0), oup(1) => s3(37)(1));
u98: encoder port map (inp(0)=>s1, inp(1)=>din(38), inp(2)=>din(38), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(38),
inp(7)=>s2(38), inp(8)=>req, oup(0)=> s3(38)(0), oup(1) => s3(38)(1));
u99: encoder port map (inp(0)=>s1, inp(1)=>din(39), inp(2)=>din(39), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(39),
inp(7)=>s2(39), inp(8)=>req, oup(0)=> s3(39)(0), oup(1) => s3(39)(1));
u100: encoder port map (inp(0)=>s1, inp(1)=>din(40), inp(2)=>din(40), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(40),
inp(7)=>s2(40), inp(8)=>req, oup(0)=> s3(40)(0), oup(1) => s3(40)(1));
u101: encoder port map (inp(0)=>s1, inp(1)=>din(41), inp(2)=>din(41), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(41),
inp(7)=>s2(41), inp(8)=>req, oup(0)=> s3(41)(0), oup(1) => s3(41)(1));
u102: encoder port map (inp(0)=>s1, inp(1)=>din(42), inp(2)=>din(42), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(42),
inp(7)=>s2(42), inp(8)=>req, oup(0)=> s3(42)(0), oup(1) => s3(42)(1));
u103: encoder port map (inp(0)=>s1, inp(1)=>din(43), inp(2)=>din(43), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(43),
inp(7)=>s2(43), inp(8)=>req, oup(0)=> s3(43)(0), oup(1) => s3(43)(1));
u104: encoder port map (inp(0)=>s1, inp(1)=>din(44), inp(2)=>din(44), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(44),
inp(7)=>s2(44), inp(8)=>req, oup(0)=> s3(44)(0), oup(1) => s3(44)(1));
u105: encoder port map (inp(0)=>s1, inp(1)=>din(45), inp(2)=>din(45), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(45),
inp(7)=>s2(45), inp(8)=>req, oup(0)=> s3(45)(0), oup(1) => s3(45)(1));

```
u106: encoder port map ( inp(0)=>s1, inp(1)=>din(46), inp(2)=>din(46), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(46),
inp(7)=>s2(46), inp(8)=>req, oup(0)=> s3(46)(0), oup(1) => s3(46)(1));
u107: encoder port map ( inp(0)=>s1, inp(1)=>din(47), inp(2)=>din(47), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(47),
inp(7)=>s2(47), inp(8)=>req, oup(0)=> s3(47)(0), oup(1) => s3(47)(1));
u108: encoder port map ( inp(0)=>s1, inp(1)=>din(48), inp(2)=>din(48), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(48),
inp(7)=>s2(48), inp(8)=>req, oup(0)=> s3(48)(0), oup(1) => s3(48)(1));
u109: encoder port map ( inp(0)=>s1, inp(1)=>din(49), inp(2)=>din(49), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(49),
inp(7)=>s2(49), inp(8)=>req, oup(0)=> s3(49)(0), oup(1) => s3(49)(1));
u110: encoder port map ( inp(0)=>s1, inp(1)=>din(50), inp(2)=>din(50), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(50),
inp(7)=>s2(50), inp(8)=>req, oup(0)=> s3(50)(0), oup(1) => s3(50)(1));
u111: encoder port map ( inp(0)=>s1, inp(1)=>din(51), inp(2)=>din(51), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(51),
inp(7)=>s2(51), inp(8)=>req, oup(0)=> s3(51)(0), oup(1) => s3(51)(1));
u112: encoder port map ( inp(0)=>s1, inp(1)=>din(52), inp(2)=>din(52), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(52),
inp(7)=>s2(52), inp(8)=>req, oup(0)=> s3(52)(0), oup(1) => s3(52)(1));
u113: encoder port map ( inp(0)=>s1, inp(1)=>din(53), inp(2)=>din(53), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(53),
inp(7)=>s2(53), inp(8)=>req, oup(0)=> s3(53)(0), oup(1) => s3(53)(1));
u114: encoder port map ( inp(0)=>s1, inp(1)=>din(54), inp(2)=>din(54), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(54),
inp(7)=>s2(54), inp(8)=>req, oup(0)=> s3(54)(0), oup(1) => s3(54)(1));
u115: encoder port map ( inp(0)=>s1, inp(1)=>din(55), inp(2)=>din(55), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(55),
inp(7)=>s2(55), inp(8)=>req, oup(0)=> s3(55)(0), oup(1) => s3(55)(1));
u116: encoder port map ( inp(0)=>s1, inp(1)=>din(56), inp(2)=>din(56), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(56),
inp(7)=>s2(56), inp(8)=>req, oup(0)=> s3(56)(0), oup(1) => s3(56)(1));
u117: encoder port map ( inp(0)=>s1, inp(1)=>din(57), inp(2)=>din(57), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(57),
inp(7)=>s2(57), inp(8)=>req, oup(0)=> s3(57)(0), oup(1) => s3(57)(1));
u118: encoder port map ( inp(0)=>s1, inp(1)=>din(58), inp(2)=>din(58), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(58),
inp(7)=>s2(58), inp(8)=>req, oup(0)=> s3(58)(0), oup(1) => s3(58)(1));
u125: encoder port map ( inp(0)=>s1, inp(1)=>din(59), inp(2)=>din(59), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(59),
inp(7)=>s2(59), inp(8)=>req, oup(0)=> s3(59)(0), oup(1) => s3(59)(1));

u119: OR_gate1 port map ( inp(0)=>s3(0)(0),inp(1)=>s3(0)(1), inp(2)=>s3(1)(0), inp(3)=>s3(1)(1), inp(4)=>s3(2)(0), inp(5)=>s3(2)(1),
inp(6)=>s3(3)(0), inp(7)=>s3(3)(1), inp(8)=>s3(4)(0), inp(9)=>s3(4)(1), inp(10)=>s3(5)(0),inp(11)=>s3(5)(1), inp(12)=>s3(6)(0),
inp(13)=>s3(6)(1),inp(14)=>s3(7)(0),inp(15)=>s3(7)(1), inp(16) => s3(8)(0), inp(17)=> s3(8)(1), inp(18) => s3(9)(0),inp(19)=>s3(9)(1),
inp(20)=>s3(10)(0), inp(21)=>s3(10)(1), inp(22)=>s3(11)(0),inp(23)=>s3(11)(1),  inp(24)=>s3(12)(0),inp(25)=>s3(12)(1),
inp(26)=>s3(13)(0), inp(27)=>s3(13)(1), inp(28)=>s3(14)(0), inp(29)=>s3(14)(1), inp(30)=>s3(15)(0), inp(31)=>s3(15)(1),
inp(32)=>s3(16)(0), inp(33)=>s3(16)(1), inp(34)=>s3(17)(0),inp(35)=>s3(17)(1), inp(36)=>s3(18)(0),
inp(37)=>s3(18)(1),inp(38)=>s3(19)(0),inp(39)=>s3(19)(1), inp(40) => s3(20)(0), inp(41)=> s3(20)(1), inp(42) =>
s3(21)(0),inp(43)=>s3(21)(1), inp(44)=>s3(22)(0), inp(45)=>s3(22)(1), inp(46)=>s3(23)(0), inp(47)=>s3(23)(1),
inp(48)=>s3(24)(0),inp(49)=>s3(24)(1), inp(50)=>s3(25)(0), inp(51)=>s3(25)(1), inp(52)=>s3(26)(0), inp(53)=>s3(26)(1),
inp(54)=>s3(27)(0), inp(55)=>s3(27)(1), inp(56)=>s3(28)(0), inp(57)=>s3(28)(1), inp(58)=>s3(29)(0),inp(59)=>s3(29)(1),
inp(60)=>s3(30)(0), inp(61)=>s3(30)(1),inp(62)=>s3(31)(0),inp(63)=>s3(31)(1), inp(64) => s3(32)(0), inp(65)=> s3(32)(1), inp(66) =>
s3(33)(0),inp(67)=>s3(33)(1), inp(68)=>s3(34)(0), inp(69)=>s3(34)(1), inp(70)=>s3(35)(0), inp(71)=>s3(35)(1),
inp(72)=>s3(36)(0),inp(73)=>s3(36)(1), inp(74)=>s3(37)(0), inp(75)=>s3(37)(1), inp(76)=>s3(38)(0), inp(77)=>s3(38)(1),
inp(78)=>s3(39)(0), inp(79)=>s3(39)(1), inp(80)=>s3(40)(0), inp(81)=>s3(40)(1), inp(82)=>s3(41)(0),inp(83)=>s3(41)(1),
inp(84)=>s3(42)(0), inp(85)=>s3(42)(1),inp(86)=>s3(43)(0),inp(87)=>s3(43)(1), inp(88) => s3(44)(0), inp(89)=> s3(44)(1), inp(90) =>
s3(45)(0),inp(91)=>s3(45)(1), inp(92)=>s3(46)(0), inp(93)=>s3(46)(1), inp(94)=>s3(47)(0), inp(95)=>s3(47)(1),
inp(96)=>s3(48)(0),inp(97)=>s3(48)(1), inp(98)=>s3(49)(0), inp(99)=>s3(49)(1), inp(100)=>s3(50)(0), inp(101)=>s3(50)(1),
inp(102)=>s3(51)(0), inp(103)=>s3(51)(1), inp(104)=>s3(52)(0), inp(105)=>s3(52)(1), inp(106)=>s3(53)(0),inp(107)=>s3(53)(1),
inp(108)=>s3(54)(0), inp(109)=>s3(54)(1),inp(110)=>s3(55)(0),inp(111)=>s3(55)(1), inp(112) => s3(56)(0), inp(113)=> s3(56)(1), inp(114)
=> s3(57)(0),inp(115)=>s3(57)(1), inp(116)=>s3(58)(0), inp(117)=>s3(58)(1),inp(118)=>s3(59)(0), inp(119)=>s3(59)(1),oup =>s4);
u120: d_ff1 port map(inp(0)=> s3(0)(0), inp(1)=> s3(1)(0), inp(2)=> s3(2)(0), inp(3)=> s3(3)(0), inp(4)=> s3(4)(0), inp(5)=>s3(5)(0),
inp(6)=>s3(6)(0), inp(7)=>s3(7)(0), inp(8)=>s3(8)(0), inp(9)=>s3(9)(0), inp(10)=>s3(10)(0), inp(11)=>s3(11)(0),inp(12)=> s3(12)(0),
inp(13)=> s3(13)(0), inp(14)=> s3(14)(0), inp(15)=> s3(15)(0), inp(16)=> s3(16)(0), inp(17)=>s3(17)(0), inp(18)=>s3(18)(0),
inp(19)=>s3(19)(0), inp(20)=>s3(20)(0), inp(21)=>s3(21)(0), inp(22)=>s3(22)(0), inp(23)=>s3(23)(0),inp(24)=> s3(24)(0), inp(25)=>
s3(25)(0), inp(26)=> s3(26)(0), inp(27)=> s3(27)(0), inp(28)=> s3(28)(0), inp(29)=>s3(29)(0), inp(30)=>s3(30)(0), inp(31)=>s3(31)(0),
inp(32)=>s3(32)(0), inp(33)=>s3(33)(0), inp(34)=>s3(34)(0), inp(35)=>s3(35)(0),inp(36)=> s3(36)(0), inp(37)=> s3(37)(0), inp(38)=>
s3(38)(0), inp(39)=> s3(39)(0), inp(40)=> s3(40)(0), inp(41)=>s3(41)(0), inp(42)=>s3(42)(0), inp(43)=>s3(43)(0), inp(44)=>s3(44)(0),
inp(45)=>s3(45)(0), inp(46)=>s3(46)(0), inp(47)=>s3(47)(0),inp(48)=> s3(48)(0), inp(49)=> s3(49)(0), inp(50)=> s3(50)(0), inp(51)=>
s3(51)(0), inp(52)=> s3(52)(0), inp(53)=>s3(53)(0), inp(54)=>s3(54)(0), inp(55)=>s3(55)(0), inp(56)=>s3(56)(0), inp(57)=>s3(57)(0),
inp(58)=>s3(58)(0),  inp(59)=>s3(59)(0), oup=> dout, clk=> s4);
u121: inv_gate port map(inp=> s4, oup=> ack);

end Behavioral;
```

----------file: **inv_gate**----------

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


entity inv_gate is
port( inp : in std_logic;
                        oup : out std_logic);
end inv_gate;

architecture Behavioral of inv_gate is

begin
oup <= not inp;

end Behavioral;
```

----------file : **encoder**----------

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

entity encoder is
port(inp: in std_logic_vector (8 downto 0) ;
    oup : out std_logic_vector(1 downto 0));
end encoder;

architecture Behavioral of encoder is

component muller_c_elem is
port(i1,i2: in std_logic ;
    oup : out std_logic );
end component;

signal s1, s2, s3, s4, s5, s6, s7, s8: std_logic;
begin
s1 <= inp(0) and inp(1);
s2 <= inp(2) or inp(3);
s3 <= inp(4);
s4 <= inp(5) and inp(6);
s5 <= inp(7) or inp(8);
u1: muller_c_elem port map (i1=>s1, i2=>s2, oup=>s6);
u2: muller_c_elem port map (i1=>s3, i2=>s3, oup=>s7);
u3: muller_c_elem port map (i1=>s4, i2=>s5, oup=>s8);
oup(0) <= s6 and s7;
oup(1) <= s7 and s8;
end Behavioral;
```

----------file : **muller c element**----------

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

entity muller_c_elem is

port(i1:in std_logic;
                i2:in std_logic;
                oup: out std_logic);
end muller_c_elem;
```

```
architecture Behavioral of muller_c_elem is
signal s: std_logic:='0';
begin
s<='1' when i1='1' and i2='1' else
                '0' when i1='0' and i2='0' else
                s;
oup<=s;

end Behavioral;
```

----------file : **Or_gate** ----------

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity OR_gate1 is
port( inp : in std_logic_vector(119 downto 0);
                        oup: out std_logic);
end OR_gate1;

architecture Behavioral of OR_gate1 is

begin
oup <= inp(0) or inp(1) or inp(2)or inp(3)or inp(4)or inp(5)or inp(6)or inp(7)or inp(8)or inp(9)or inp(10)or inp(11)or inp(12)or inp(13)or
inp(14)or inp(15)or inp(16)or inp(17)or inp(18)or inp(19)or inp(20)or inp(21)or inp(22)or inp(23)or inp(24)or inp(25)or inp(26)or inp(27)or
inp(28)or inp(29) or inp(30)or inp(31)or inp(32)or inp(33) or inp(34) or inp(35)or inp(36)or inp(37)or inp(38)or inp(39)or inp(40)or
inp(41)or inp(42)or inp(43)or inp(44)or inp(45)or inp(46)or inp(47)or inp(48)or inp(49)or inp(50)or inp(51)or inp(52)or inp(53)or inp(54)or
inp(55)or inp(56)or inp(57)or inp(58)or inp(59) or inp(60) or inp(61) or inp(62)or inp(63)or inp(64)or inp(65)or inp(66)or inp(67)or
inp(68)or inp(69)or inp(70)or inp(71)or inp(72)or inp(73)or inp(74)or inp(75)or inp(76)or inp(77)or inp(78)or inp(79)or inp(80)or inp(81)or
inp(82)or inp(83)or inp(84)or inp(85)or inp(86)or inp(87)or inp(88)or inp(89) or inp(90)or inp(91)or inp(92)or inp(93) or inp(94) or
inp(95)or inp(96)or inp(97)or inp(98)or inp(99)or inp(100)or inp(101)or inp(102)or inp(103)or inp(104)or inp(105)or inp(106)or inp(107)or
inp(108)or inp(109)or inp(110)or inp(111)or inp(112)or inp(113)or inp(114)or inp(115)or inp(116)or inp(117)or inp(118)or inp(119);

end Behavioral;
```

----------file **: d_ff1**----------

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

entity d_ff1 is
port(inp : in std_logic_vector(59 downto 0);
                oup: out std_logic_vector(59 downto 0);
                clk: in std_logic);
end d_ff1;

architecture Behavioral of d_ff1 is

begin

process(clk)
begin
if(clk'event and clk = '1') then
oup <= inp;
end if;
end process;

end Behavioral;
```

----------file : **alu**----------

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;

entity alu is
port( reset :          in std_logic;
                       din        :  in std_logic_vector(58 downto 0);
                       db_in      :          in std_logic_vector(15 downto 0);
                       rd                    :          out std_logic;
                       wr                    :          out std_logic;
                       mem_en:    out std_logic;
                       db_sel:    out std_logic_vector(2 downto 0);
                       ld_a       :          out std_logic;
                       ld_b       :          out std_logic;
                       ld_acc:    out std_logic;
                       ld_f       :          out std_logic;
                       req1  :    out std_logic;
                       req2       :          in std_logic;
                       dout  :    out std_logic_vector(59 downto 0));
end alu;

architecture Behavioral of alu is

component lso is
port( reset  :                    in std_logic;
                       aclk       :                    out std_logic);
end component;

component Counter  is
port( reset, clk       : in std_logic;
                       dout                         : out std_logic);
end component;

component logic2 is
PORT(P:                in std_logic_vector(15 downto 0);
            en:                in std_logic;
            Q:                 in std_logic_vector(15 downto 0);
            R:                 out std_logic_vector(31 downto 0);
            sel:       in std_logic_vector(4 downto 0);
            Cin:       in std_logic;
            Cout:      out std_logic) ;
end component;

component ecr2 is
port(  din : in std_logic_vector (59 downto 0);
     req     : in std_logic ;
                        ack        : out std_logic ;
     dout : out std_logic_vector (59 downto 0));
end component;

signal s0, s1, s2, s3, s4, ack: std_logic;
signal d1: std_logic_vector(31 downto 0);
signal carry: std_logic;

begin
req1<= s0;
s1<= not s0;
s2 <= reset and s1;
s4 <= s0 and req2;

U1: lso port map( reset => s2, aclk => s3);
U2: Counter port map( reset => ack, clk => s3, dout => s0);
```

```
U3: logic2 port map( P => din(32 downto 17), Q => din(16 downto 1), cin => din(0), en => din(42), sel => din(41 downto 37), R => d1,
cout => carry);
U4: ecr2 port map( din(59 downto 28) => d1, din(27) => carry, din(26 downto 17) => din(55 downto 46), din(16 downto 1) => db_in, din(0)
=> din(42), req=> s4, ack => ack, dout => dout);

rd      <= din(44);
wr      <= din(43);
mem_en <= din(45);
db_sel <= din(58 downto 56);
ld_a <= din(36);
ld_b <= din(35);
ld_acc <= din(34);
ld_f    <= din(33);

end Behavioral;


----------file : logic2----------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity logic2 is
PORT(P:         in std_logic_vector(15 downto 0);
        Q:              in std_logic_vector(15 downto 0);
        en:             in std_logic;
        R:              out std_logic_vector(31 downto 0);
        sel:    in std_logic_vector(4 downto 0);
        Cin:    in std_logic;
        Cout:   out std_logic) ;
end logic2;

architecture Behavioral of logic2 is
component ARITH is
PORT(A:         in std_logic_vector(15 downto 0);
        B:              in std_logic_vector(15 downto 0);
        S:              in std_logic_vector(3 downto 0);
        Cin:    in std_logic;
        Cout:   out std_logic;
        O:              out std_logic_vector(31 downto 0));
end component;

component LOGICAL is
PORT(A:         in std_logic_vector(15 downto 0);
        B:              in std_logic_vector(15 downto 0);
        S:              in std_logic_vector(3 downto 0);
        O:              out std_logic_vector(31 downto 0);
        Cin:    in std_logic;
        Cout:   out std_logic);
end component;

component multiplexer is
PORT(A:         in std_logic_vector(31 downto 0);
        B:              in std_logic_vector(31 downto 0);
        S:              in std_logic;
        O:              out std_logic_vector(31 downto 0));
end component;

component multiplexer2 is
PORT(A:         in std_logic;
        B:              in std_logic;
        S:              in std_logic;
```

```
        O:              out std_logic);
end component;

signal s1,s2,s3:std_logic_vector(31 downto  0);
signal cout1, cout2, c :std_logic;

begin
U1: ARITH port map(A=>P, B=>Q, O=>s1, Cin=>Cin, Cout=> cout1, S=>sel(3 downto 0));
U2: LOGICAL port map(A=>P, B=>Q, O=>s2, S=> sel(3 downto 0), Cin => Cin, Cout => cout2);
U3: multiplexer port map(A=>s1, B=>s2, O=>s3,S=>sel(4));
U4: multiplexer2 port map ( A=> cout1, B=> cout2, S=> sel(4), O=> c);

R <= s3 when en = '1' else
                "ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ" when en = '0';

Cout <= c when en ='1' else
                        'Z' when en = '0';
end Behavioral;
```

----------file : **arith**----------

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

entity ARITH is
PORT(A:         in std_logic_vector(15 downto 0);
        B:              in std_logic_vector(15 downto 0);
        S:              in std_logic_vector(3 downto 0);
        Cin:    in std_logic;
        Cout:   out std_logic;
        O:              out std_logic_vector(31 downto 0));
end ARITH;

architecture Behavioral of ARITH is
component Adder is
PORT(A:         in std_logic_vector(15 downto 0);
        B:              in std_logic_vector(15 downto 0);
        Cin:    in std_logic;
        Cout:   out std_logic;
        O:              out std_logic_vector(15 downto 0));
end component;

component Complement is
PORT(A:         in std_logic_vector(15 downto 0);
        O:              out std_logic_vector(15 downto 0));
end Component;

component SixteenBitMultiplier is
port ( a,b : in std_logic_vector(15 downto 0);
                y: out std_logic_vector(31 downto 0));
end component;

component multiplex_in_alu is
PORT(A:         in std_logic_vector(15 downto 0);
        B:              in std_logic_vector(15 downto 0);
        S:              in std_logic;
        O:              out std_logic_vector(15 downto 0));
end component;

component multiplexer3 is
PORT(A:         in std_logic_vector(31 downto 0);
        B:              in std_logic_vector(31 downto 0);
        C:              in std_logic_vector(31 downto 0);
```

```
        D:          in std_logic_vector(31 downto 0);
        S:          in std_logic_vector(2 downto 0);
        O:          out std_logic_vector(31 downto 0));
end component;

signal s1,s3,s5: std_logic_vector(31 downto  0);
signal s6: std_logic_vector(31 downto 0);

begin

U1: Adder port map(A=>s5(15 downto 0), B=>B, Cin=>Cin, Cout=>Cout, O=>s1(15 downto 0));
U3: Complement port map(A=>A, O=>s3(15 downto 0));
U4: SixteenBitMultiplier port map(a=>A, b=>B, y=> s6);
U5: multiplex_in_alu port map(A=>A,  B=>s3(15 downto 0), O=>s5(15 downto 0), S=>S(2));
U6: multiplexer3 port map(A=>s1, B=>"00000000000000000000000000000000", C=>s3, D=>s6, S=>S(2 downto 0), O=>O);
s1(31 downto 16) <= "0000000000000000";
s3(31 downto 16) <= "0000000000000000";
s5(31 downto 16) <= "0000000000000000";

end Behavioral;
```

----------file : **adder**----------

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


entity Adder is
PORT(A:         in std_logic_vector(15 downto 0);
        B:          in std_logic_vector(15 downto 0);
        Cin:    in std_logic;
        Cout:   out std_logic;
        O:          out std_logic_vector(15 downto 0));
end Adder;

architecture Behavioral of Adder is
component FourBitAdder is
Port(A,B: in std_logic_vector(3 downto 0);
                Cin: in std_logic;
                Cout: out std_logic;
                O: out std_logic_vector(3 downto 0));

end component;

signal s1, s2, s3: std_logic;
begin
U1: FourBitAdder port map(A => A(3 downto 0), B=> B(3 downto 0), Cin=> Cin, Cout => s1, O => O( 3 downto 0));
U2: FourBitAdder port map(A => A(7 downto 4), B=> B(7 downto 4), Cin=> s1, Cout => s2, O => O( 7 downto 4));
U3: FourBitAdder port map(A => A(11 downto 8), B=> B(11 downto 8), Cin=> s2, Cout => s3, O => O( 11 downto 8));
U4: FourBitAdder port map(A => A(15 downto 12), B=> B(15 downto 12), Cin=> s3, Cout => Cout, O => O( 15 downto 12));
end Behavioral;
```

----------file : **four bit adder**----------

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


entity FourBitAdder is
Port(A,B: in std_logic_vector(3 downto 0);
Cin: in std_logic;
Cout: out std_logic;
O: out std_logic_vector(3 downto 0));
```

```
end FourBitAdder;

architecture Behavioral of FourBitAdder is

signal c: std_logic_vector(4 downto 0);
signal p: std_logic_vector(3 downto 0);
signal g: std_logic_vector(3 downto 0);

begin

G1: for i in 0 to 3 generate
p(i) <= A(i) XOR B(i);
g(i) <= A(i) AND B(i);
O(i) <= p(i) XOR c(i);
end generate;

c(0)<= Cin;
c(1)<= (Cin AND p(0)) OR g(0);
c(2)<= (Cin AND p(0) and p(1)) OR (g(0) and p(1)) or g(1);
c(3)<= (Cin AND p(0) and p(1)and p(2)) OR (g(0) and p(1) and p(2)) or (g(1) and p(2)) or g(2);
c(4)<= (Cin AND p(0) and p(1)and p(2) and p(3)) OR (g(0) and p(1) and p(2) and p(3)) or (g(1) and p(2) and p(3)) or (g(2) and p(3)) or
g(3);
Cout<= c(4);
end Behavioral;
```

----------file : **complement**----------

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


entity Complement is
PORT(A:                 in std_logic_vector(15 downto 0);
            O:                      out std_logic_vector(15 downto 0));
end Complement;

architecture Behavioral of Complement is

begin
G: for i in 15 downto 0 generate
O(i)<= not A(i);
end generate;

end Behavioral;
```

----------file :**sixteen bit multiplier**----------

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

entity SixteenBitMultiplier is
port ( a,b : in std_logic_vector(15 downto 0);
                        y: out std_logic_vector(31 downto 0));

end SixteenBitMultiplier;

architecture Behavioral of SixteenBitMultiplier is
```

```
signal init,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s31,s32,s33,s34: std_logic_vector (7 downto 0);
signal s11,s12,s13,s14: std_logic_vector (15 downto 0);
signal s21, s22, s23, s24, s25,s26: std_logic;

component EightBitMultiplier is
port ( a,b : in std_logic_vector(7 downto 0);
                        y: out std_logic_vector(15 downto 0));
end component;

component EightBitAdder is
port(a,b,c : in std_logic_vector(7 downto 0);
                        s: out std_logic_vector(7 downto 0);
                        z: in std_logic;
                        cout1: out std_logic;
                        cout2: out std_logic);
end component;

component EightBitAdderStageTwo is
port(a,b,c : in std_logic_vector(7 downto 0);
                        s: out std_logic_vector(7 downto 0);
                        z: in std_logic;
                        cin: in std_logic;
                        cout1: out std_logic;
                        cout2: out std_logic);
end component;

begin

init<= "00000000";
s1<= a(7 downto 0);
s2<= a(15 downto 8);
s3<= b(7 downto 0);
s4<= b(15 downto 8);
EBM0: EightBitMultiplier port map( a=> s1, b=> s3, y=> s11);
EBM1: EightBitMultiplier port map( a=> s2, b=> s3, y=> s12);
EBM2: EightBitMultiplier port map( a=> s1, b=> s4, y=> s13);
EBM3: EightBitMultiplier port map( a=> s2, b=> s4, y=> s14);
y(7 downto 0) <= s11(7 downto 0);
s5<= s11(15 downto 8);
s6<= s12(7 downto 0);
s7<= s12(15 downto 8);
s8<= s13(7 downto 0);
s9<= s13(15 downto 8);
s10<= s14(7 downto 0);
s31<= s14(15 downto 8);

AB0: EightBitAdder port map (a=>s5, b=>s6 ,c=>s8, s=>s32, z =>'0',  cout1=> s21, cout2 => s22);
AB1: EightBitAdderStageTwo port map (a=>s7, b=>s9 ,c=>s10, s=>s33, z => s22, cin=>s21 , cout1=> s23, cout2 => s24);
AB2: EightBitAdderStageTwo port map (a=>s31, b=>init, c=> init, s=>s34, z => s24, cin=>s23 , cout1=> s25, cout2 => s26);

y(15 downto 8) <= s32;
y(23 downto 16) <= s33;
y(31 downto 24) <= s34;
end Behavioral;


----------------------------------file : EBM-------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity EightBitMultiplier is
```

```
port ( a,b : in std_logic_vector(7 downto 0);
                        y: out std_logic_vector(15 downto 0));
end EightBitMultiplier;

architecture Behavioral of EightBitMultiplier is
signal init,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s31,s32,s33,s34: std_logic_vector (3 downto 0);
signal s11,s12,s13,s14: std_logic_vector (7 downto 0);
signal s21, s22, s23, s24, s25,s26: std_logic;

component VedicMultiplier is
port ( a,b : in std_logic_vector(3 downto 0);
                        y: out std_logic_vector(7 downto 0));
end component;


component FourBitAdder1 is
port(a,b,c : in std_logic_vector(3 downto 0);
                        s: out std_logic_vector(3 downto 0);
                        z: in std_logic;
                        cout1: out std_logic;
                        cout2: out std_logic);
end component;

component FourBitAdderStageTwo is
port(a,b,c : in std_logic_vector(3 downto 0);
                        s: out std_logic_vector(3 downto 0);
                        z: in std_logic;
                        cin: in std_logic;
                        cout1: out std_logic;
                        cout2: out std_logic);
end component;

begin

init<= "0000";
s1<= a(3 downto 0);
s2<= a(7 downto 4);
s3<= b(3 downto 0);
s4<= b(7 downto 4);
VM0: VedicMultiplier port map( a=> s1, b=> s3, y=> s11);
VM1: VedicMultiplier port map( a=> s2, b=> s3, y=> s12);
VM2: VedicMultiplier port map( a=> s1, b=> s4, y=> s13);
VM3: VedicMultiplier port map( a=> s2, b=> s4, y=> s14);
y(3 downto 0) <= s11(3 downto 0);
s5<= s11(7 downto 4);
s6<= s12(3 downto 0);
s7<= s12(7 downto 4);
s8<= s13(3 downto 0);
s9<= s13(7 downto 4);
s10<= s14(3 downto 0);
s31<= s14(7 downto 4);

AB0: FourBitAdder1 port map (a=>s5, b=>s6 ,c=>s8, s=>s32, z =>'0',  cout1=> s21, cout2 => s22);
AB1: FourBitAdderStageTwo port map (a=>s7, b=>s9 ,c=>s10, s=>s33, z => s22, cin=>s21 , cout1=> s23, cout2 => s24);
AB2: FourBitAdderStageTwo port map (a=>s31, b=>init, c=> init, s=>s34, z => s24, cin=>s23 , cout1=> s25, cout2 => s26);

y(7 downto 4) <= s32;
y(11 downto 8) <= s33;
y(15 downto 12) <= s34;
end Behavioral;
```

----------file : **vedic multiplier**----------

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


entity VedicMultiplier is
port ( a,b : in std_logic_vector(3 downto 0);
                        y: out std_logic_vector(7 downto 0));
end VedicMultiplier;

architecture Behavioral of VedicMultiplier is
signal s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19: std_logic ;
signal s20,s21,s22,s23,s24,s25,s26,s27,s28,s29,s30,s31,s32,s33,s34,s35,s36,s37,s38: std_logic ;

component Halfadder is
port ( a,b :in std_logic;
                        cout, sum : out std_logic);
end component;

component Fulladder is
port( a,b,c :in std_logic;
                        cout, sum : out std_logic);
end component;

begin
s0 <= a(0) AND b(0);
s1 <= a(1) AND b(0);
s2 <= a(0) AND b(1);
s3 <= a(2) AND b(0);
s4 <= a(0) AND b(2);
s5 <= a(1) AND b(1);
s6 <= a(3) AND b(0);
s7 <= a(0) AND b(3);
s8 <= a(2) AND b(1);
s9 <= a(1) AND b(2);
s10 <= a(3) AND b(1);
s11 <= a(1) AND b(3);
s12 <= a(2) AND b(2);
s13 <= a(3) AND b(2);
s14 <= a(2) AND b(3);
s15 <= a(3) AND b(3);

y(0) <= s0;
HA0: Halfadder port map(a=> s1, b=>s2, sum=> y(1), cout=> s16);
HA1: Halfadder port map(a=> s3, b=>s16, sum=> s17, cout=> s18);
FA0: Fulladder port map(a=> s4, b=>s5, c=> s17, sum=> y(2), cout=> s19);
HA2: Halfadder port map(a=> s18, b=>s19, sum=> s21, cout=> s20);
FA1: Fulladder port map(a=> s6, b=>s7, c=> s21, sum=> s23, cout=> s22);
FA2: Fulladder port map(a=> s9, b=>s8, c=> s23, sum=> y(3), cout=> s24);
HA3: Halfadder port map(a=> s22, b=>s24, sum=> s26, cout=> s25);
FA3: Fulladder port map(a=> s10, b=>s20, c=> s26, sum=> s27, cout=> s28);
FA4: Fulladder port map(a=> s11, b=>s12, c=> s27, sum=> y(4), cout=> s29);
HA4: Halfadder port map(a=> s29, b=>s28, sum=> s31, cout=> s30);
FA5: Fulladder port map(a=> s13, b=>s14, c=> s31, sum=> s33, cout=> s32);
HA5: Halfadder port map(a=> s25, b=>s33, sum=> y(5), cout=> s34);
HA6: Halfadder port map(a=> s32, b=>s34, sum=> s36, cout=> s35);
FA6: Fulladder port map(a=> s15, b=>s30, c=> s36, sum=> y(6), cout=> s37);
HA7: Halfadder port map(a=> s35, b=>s37, sum=> y(7), cout=> s38);

end Behavioral;
```

----------file : **half adder**----------

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


entity Halfadder is
port ( a,b :in std_logic;
                        cout, sum : out std_logic);
end Halfadder;

architecture Behavioral of Halfadder is

begin
sum <= a XOR b;
cout<= a AND b;

end Behavioral;
```

----------file **: fuller adder**----------

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


entity Fulladder is
port( a,b,c :in std_logic;
                        cout, sum : out std_logic);
end Fulladder;

architecture Behavioral of Fulladder is

begin
sum <= a XOR b XOR c;
cout <= (a AND b) OR (b AND c) OR (c AND a);

end Behavioral;
```

----------file :**4bit adder1**----------

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


entity FourBitAdder1 is
port(a,b,c : in std_logic_vector(3 downto 0);
                        s: out std_logic_vector(3 downto 0);
                        z: in std_logic;
                        cout1: out std_logic;
                        cout2: out std_logic);
end FourBitAdder1;

architecture Behavioral of FourBitAdder1 is

signal s0,s1,s2,s3,s4,s5,s6,s7,s11,s12: std_logic;

component Fulladder is
port( a,b,c :in std_logic;
                        cout, sum : out std_logic);
end component;
```

```
begin

FA0: Fulladder port map(a=> a(0), b=>b(0), c=> c(0), sum=> s0, cout=> s1);
FA1: Fulladder port map(a=> a(1), b=>b(1), c=> c(1), sum=> s2, cout=> s3);
FA2: Fulladder port map(a=> a(2), b=>b(2), c=> c(2), sum=> s4, cout=> s5);
FA3: Fulladder port map(a=> a(3), b=>b(3), c=> c(3), sum=> s6, cout=> s7);
cout1<= s7;
s(0)<= s0;
FA4: Fulladder port map(a=> s1, b=>s2, c=> z, sum=> s(1), cout=> s11);
FA5: Fulladder port map(a=> s3, b=>s4, c=> s11, sum=> s(2), cout=> s12);
FA6: Fulladder port map(a=> s5, b=>s6, c=> s12, sum=> s(3), cout=> cout2);

end Behavioral;
```

----------file : **8 bit adderstage1**----------

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


entity EightBitAdder is
port(a,b,c : in std_logic_vector(7 downto 0);
                        s: out std_logic_vector(7 downto 0);
                        z: in std_logic;
                        cout1: out std_logic;
                        cout2: out std_logic);
end EightBitAdder;

architecture Behavioral of EightBitAdder is
signal s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,s20,s21: std_logic;

component Fulladder is
port( a,b,c :in std_logic;
                        cout, sum : out std_logic);
end component;

begin

FA0: Fulladder port map(a=> a(0), b=>b(0), c=> c(0), sum=> s0, cout=> s1);
FA1: Fulladder port map(a=> a(1), b=>b(1), c=> c(1), sum=> s2, cout=> s3);
FA2: Fulladder port map(a=> a(2), b=>b(2), c=> c(2), sum=> s4, cout=> s5);
FA3: Fulladder port map(a=> a(3), b=>b(3), c=> c(3), sum=> s6, cout=> s7);
FA4: Fulladder port map(a=> a(4), b=>b(4), c=> c(4), sum=> s8, cout=> s9);
FA5: Fulladder port map(a=> a(5), b=>b(5), c=> c(5), sum=> s10, cout=> s11);
FA6: Fulladder port map(a=> a(6), b=>b(6), c=> c(6), sum=> s12, cout=> s13);
FA7: Fulladder port map(a=> a(7), b=>b(7), c=> c(7), sum=> s14, cout=> s15);
cout1 <= s15;
s(0)<= s0;
FA8: Fulladder port map(a=> s1, b=>s2, c=> z, sum=> s(1), cout=> s16);
FA9: Fulladder port map(a=> s3, b=>s4, c=> s16, sum=> s(2), cout=> s17);
FA10: Fulladder port map(a=> s5, b=>s6, c=> s17, sum=> s(3), cout=> s18);
FA11: Fulladder port map(a=> s7, b=>s8, c=> s18, sum=> s(4), cout=> s19);
FA12: Fulladder port map(a=> s9, b=>s10, c=> s19, sum=> s(5), cout=> s20);
FA13: Fulladder port map(a=> s11, b=>s12, c=> s20, sum=> s(6), cout=> s21);
FA14: Fulladder port map(a=> s13, b=>s14, c=> s21, sum=> s(7), cout=> cout2);


end Behavioral;
```

----------file : **8 bitadder stage 2**----------

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

```
entity EightBitAdderStageTwo is
port(a,b,c : in std_logic_vector(7 downto 0);
                        s: out std_logic_vector(7 downto 0);
                        z: in std_logic;
                        cin: in std_logic;
                        cout1: out std_logic;
                        cout2: out std_logic);

end EightBitAdderStageTwo;

architecture Behavioral of EightBitAdderStageTwo is

signal s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,s20,s21,s22: std_logic;

component Fulladder is
port( a,b,c :in std_logic;
                        cout, sum : out std_logic);
end component;

begin

FA0: Fulladder port map(a=> a(0), b=>b(0), c=> c(0), sum=> s0, cout=> s1);
FA1: Fulladder port map(a=> a(1), b=>b(1), c=> c(1), sum=> s2, cout=> s3);
FA2: Fulladder port map(a=> a(2), b=>b(2), c=> c(2), sum=> s4, cout=> s5);
FA3: Fulladder port map(a=> a(3), b=>b(3), c=> c(3), sum=> s6, cout=> s7);
FA4: Fulladder port map(a=> a(4), b=>b(4), c=> c(4), sum=> s8, cout=> s9);
FA5: Fulladder port map(a=> a(5), b=>b(5), c=> c(5), sum=> s10, cout=> s11);
FA6: Fulladder port map(a=> a(6), b=>b(6), c=> c(6), sum=> s12, cout=> s13);
FA7: Fulladder port map(a=> a(7), b=>b(7), c=> c(7), sum=> s14, cout=> s15);
cout1<= s15;
FA8: Fulladder port map(a=> s0, b=>cin, c=> z, sum=> s(0), cout=> s16);
FA9: Fulladder port map(a=> s1, b=>s2, c=> s16, sum=> s(1), cout=> s17);
FA10: Fulladder port map(a=> s3, b=>s4, c=> s17, sum=> s(2), cout=> s18);
FA11: Fulladder port map(a=> s5, b=>s6, c=> s18, sum=> s(3), cout=> s19);
FA12: Fulladder port map(a=> s7, b=>s8, c=> s19, sum=> s(4), cout=> s20);
FA13: Fulladder port map(a=> s9, b=>s10, c=> s20, sum=> s(5), cout=> s21);
FA14: Fulladder port map(a=> s11, b=>s12, c=> s21, sum=> s(6), cout=> s22);
FA15: Fulladder port map(a=> s13, b=>s14, c=> s22, sum=> s(7), cout=> cout2);


end Behavioral;


------------------------------file : logical-----------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity LOGICAL is
PORT(A:             in std_logic_vector(15 downto 0);
        B:                  in std_logic_vector(15 downto 0);
        S:                  in std_logic_vector(3 downto 0);
        O:                  out std_logic_vector(31 downto 0);
        Cin:    in std_logic;
        Cout:   out std_logic);
end LOGICAL;

architecture Behavioral of LOGICAL is


component multiplexer1 is
PORT(A:             in std_logic_vector(15 downto 0);
```

```
		B:		in std_logic_vector(15 downto 0);
		C:		in std_logic_vector(15 downto 0);
		D:		in std_logic_vector(15 downto 0);
		E:		in std_logic_vector(15 downto 0);
		F:		in std_logic_vector(15 downto 0);
		G:		in std_logic_vector(15 downto 0);
		H:		in std_logic_vector(15 downto 0);
		I:		in std_logic_vector(15 downto 0);
		S:		in std_logic_vector(3 downto 0);
		O:		out std_logic_vector(15 downto 0));
end component;

component multiplexer2 is
PORT(A:		in std_logic;
		B:		in std_logic;
		S:		in std_logic;
		O:		out std_logic);
end component;

component AND_BLOCK is
PORT(A:		in std_logic_vector(15 downto 0);
		B:		in std_logic_vector(15 downto 0);
		O:		out std_logic_vector(15 downto 0));
end component;

component OR_BLOCK is
PORT(A:		in std_logic_vector(15 downto 0);
		B:		in std_logic_vector(15 downto 0);
		O:		out std_logic_vector(15 downto 0));
end component;

component SHL is
PORT(A:		in std_logic_vector(14 downto 0);
		O:		out std_logic_vector(15 downto 0));
end component;

component SHR is
PORT(A:		in std_logic_vector(15 downto 1);
		O:		out std_logic_vector(15 downto 0));
end component;


component SAR is
PORT(A:		in std_logic_vector(15 downto 1);
		O:		out std_logic_vector(15 downto 0));
end component;

component RRC is
PORT(A:		in std_logic_vector(15 downto 0);
		Cin:	in std_logic;
		Cout:	out std_logic;
		O:		out std_logic_vector(15 downto 0));
end component;

component RLC is
PORT(A:		in std_logic_vector(15 downto 0);
		Cin:	in std_logic;
		Cout:	out std_logic;
		O:		out std_logic_vector(15 downto 0));
end component;

component RotateLeft is
```

```
PORT(A:             in std_logic_vector(15 downto 0);
        O:                  out std_logic_vector(15 downto 0));
end component;

component RotateRight is
PORT(A:             in std_logic_vector(15 downto 0);
        O:                  out std_logic_vector(15 downto 0));
end component;

signal s1,s2,s3,s4,s5,s6,s7,s8,s9: std_logic_vector(15 downto  0);
signal cout1,cout2: std_logic;
begin
U1: AND_BLOCK port map(A=>A, B=>B, O=>s1);
U2: OR_BLOCK port map(A=>A, B=>B, O=>s2);
U3: SHL port map(A=>A(14 downto 0), O=>s3);
U4: SHR port map(A=>A(15 downto 1), O=>s4);
U5: SAR port map(A=>A(15 downto 1), O=>s5);
U6: RRC port map(A=>A, O=>s6, Cin=> Cin, Cout=> cout1);
U7: RLC port map(A=>A, O=>s7, Cin=> Cin, Cout=> cout2);
U8: RotateLeft port map(A=>A, O=>s8);
U9: RotateRight port map(A=>A,O=>s9);
U11: multiplexer1 port map(A=>s1, B=>s2, C=>s3, D=>s4, E=>s5, F=>s6, G=>s7, H=>s8, I=>s9, S=>S, O=>O(15 downto 0));
U12: multiplexer2 port map(A=>cout1, B=> cout2, O=> Cout, S=> S(0));

O(31 downto 16) <= "0000000000000000";
end Behavioral;


----------file : ecr2----------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
entity ecr2 is
port( din : in std_logic_vector(59 downto 0);
      req : in std_logic ;
                  ack : out std_logic ;
      dout : out std_logic_vector(59 downto 0));
end ecr2;

architecture Behavioral of ecr2 is

component OR_gate2 is
port( inp : in std_logic_vector(119 downto 0);
                  oup: out std_logic);
end component;

component encoder is
port(inp: in std_logic_vector (8 downto 0) ;
    oup : out std_logic_vector(1 downto 0));
end component;

component and_gate is
port( inp : in std_logic;
                  oup: out std_logic);
end component;

component inv_gate is
port( inp : in std_logic;
                  oup: out std_logic);
end component;

component d_ff2 is
```

```
port(inp : in std_logic_vector(59 downto 0);
                oup: out std_logic_vector(59 downto 0);
                clk: in std_logic);
end component;

type dataout is array (59 downto 0) of std_logic_vector(1 downto 0);
signal s1: std_logic;
signal s2: std_logic_vector(59 downto 0);
signal s3: dataout;
signal s4: std_logic;

begin
u1: inv_gate port map( inp=>req, oup=>s1);
u2: inv_gate port map( inp=>din(0), oup=>s2(0));
u3: inv_gate port map( inp=>din(1), oup=>s2(1));
u4: inv_gate port map( inp=>din(2), oup=>s2(2));
u5: inv_gate port map( inp=>din(3), oup=>s2(3));
u6: inv_gate port map( inp=>din(4), oup=>s2(4));
u7: inv_gate port map( inp=>din(5), oup=>s2(5));
u8: inv_gate port map( inp=>din(6), oup=>s2(6));
u9: inv_gate port map( inp=>din(7), oup=>s2(7));
u10: inv_gate port map( inp=>din(8), oup=>s2(8));
u11: inv_gate port map( inp=>din(9), oup=>s2(9));
u12: inv_gate port map( inp=>din(10), oup=>s2(10));
u13: inv_gate port map( inp=>din(11), oup=>s2(11));
u14: inv_gate port map( inp=>din(12), oup=>s2(12));
u15: inv_gate port map( inp=>din(13), oup=>s2(13));
u16: inv_gate port map( inp=>din(14), oup=>s2(14));
u17: inv_gate port map( inp=>din(15), oup=>s2(15));
u18: inv_gate port map( inp=>din(16), oup=>s2(16));
u19: inv_gate port map( inp=>din(17), oup=>s2(17));
u20: inv_gate port map( inp=>din(18), oup=>s2(18));
u21: inv_gate port map( inp=>din(19), oup=>s2(19));
u22: inv_gate port map( inp=>din(20), oup=>s2(20));
u23: inv_gate port map( inp=>din(21), oup=>s2(21));
u24: inv_gate port map( inp=>din(22), oup=>s2(22));
u25: inv_gate port map( inp=>din(23), oup=>s2(23));
u26: inv_gate port map( inp=>din(24), oup=>s2(24));
u27: inv_gate port map( inp=>din(25), oup=>s2(25));
u28: inv_gate port map( inp=>din(26), oup=>s2(26));
u29: inv_gate port map( inp=>din(27), oup=>s2(27));
u30: inv_gate port map( inp=>din(28), oup=>s2(28));
u31: inv_gate port map( inp=>din(29), oup=>s2(29));
u32: inv_gate port map( inp=>din(30), oup=>s2(30));
u33: inv_gate port map( inp=>din(31), oup=>s2(31));
u34: inv_gate port map( inp=>din(32), oup=>s2(32));
u35: inv_gate port map( inp=>din(33), oup=>s2(33));
u36: inv_gate port map( inp=>din(34), oup=>s2(34));
u37: inv_gate port map( inp=>din(35), oup=>s2(35));
u38: inv_gate port map( inp=>din(36), oup=>s2(36));
u39: inv_gate port map( inp=>din(37), oup=>s2(37));
u40: inv_gate port map( inp=>din(38), oup=>s2(38));
u41: inv_gate port map( inp=>din(39), oup=>s2(39));
u42: inv_gate port map( inp=>din(40), oup=>s2(40));
u43: inv_gate port map( inp=>din(41), oup=>s2(41));
u44: inv_gate port map( inp=>din(42), oup=>s2(42));
u45: inv_gate port map( inp=>din(43), oup=>s2(43));
u46: inv_gate port map( inp=>din(44), oup=>s2(44));
u47: inv_gate port map( inp=>din(45), oup=>s2(45));
u48: inv_gate port map( inp=>din(46), oup=>s2(46));
u49: inv_gate port map( inp=>din(47), oup=>s2(47));
u50: inv_gate port map( inp=>din(48), oup=>s2(48));
```

```
u51: inv_gate port map( inp=>din(49), oup=>s2(49));
u52: inv_gate port map( inp=>din(50), oup=>s2(50));
u53: inv_gate port map( inp=>din(51), oup=>s2(51));
u54: inv_gate port map( inp=>din(52), oup=>s2(52));
u55: inv_gate port map( inp=>din(53), oup=>s2(53));
u56: inv_gate port map( inp=>din(54), oup=>s2(54));
u57: inv_gate port map( inp=>din(55), oup=>s2(55));
u58: inv_gate port map( inp=>din(56), oup=>s2(56));
u59: inv_gate port map( inp=>din(57), oup=>s2(57));
u60: inv_gate port map( inp=>din(58), oup=>s2(58));
u122: inv_gate port map( inp=>din(59), oup=>s2(59));
u61: encoder port map ( inp(0)=>s1, inp(1)=>din(0), inp(2)=>din(0), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(0), inp(7)=>s2(0),
inp(8)=>req, oup(0)=> s3(0)(0), oup(1) => s3(0)(1));
u62: encoder port map ( inp(0)=>s1, inp(1)=>din(1), inp(2)=>din(1), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(1), inp(7)=>s2(1),
inp(8)=>req, oup(0)=> s3(1)(0), oup(1) => s3(1)(1));
u63: encoder port map ( inp(0)=>s1, inp(1)=>din(2), inp(2)=>din(2), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(2), inp(7)=>s2(2),
inp(8)=>req, oup(0)=> s3(2)(0), oup(1) => s3(2)(1));
u64: encoder port map ( inp(0)=>s1, inp(1)=>din(3), inp(2)=>din(3), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(3), inp(7)=>s2(3),
inp(8)=>req, oup(0)=> s3(3)(0), oup(1) => s3(3)(1));
u65: encoder port map ( inp(0)=>s1, inp(1)=>din(4), inp(2)=>din(4), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(4), inp(7)=>s2(4),
inp(8)=>req, oup(0)=> s3(4)(0), oup(1) => s3(4)(1));
u66: encoder port map ( inp(0)=>s1, inp(1)=>din(5), inp(2)=>din(5), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(5), inp(7)=>s2(5),
inp(8)=>req, oup(0)=> s3(5)(0), oup(1) => s3(5)(1));
u67: encoder port map ( inp(0)=>s1, inp(1)=>din(6), inp(2)=>din(6), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(6), inp(7)=>s2(6),
inp(8)=>req, oup(0)=> s3(6)(0), oup(1) => s3(6)(1));
u68: encoder port map ( inp(0)=>s1, inp(1)=>din(7), inp(2)=>din(7), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(7), inp(7)=>s2(7),
inp(8)=>req, oup(0)=> s3(7)(0), oup(1) => s3(7)(1));
u69: encoder port map ( inp(0)=>s1, inp(1)=>din(8), inp(2)=>din(8), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(8), inp(7)=>s2(8),
inp(8)=>req, oup(0)=> s3(8)(0), oup(1) => s3(8)(1));
u70: encoder port map ( inp(0)=>s1, inp(1)=>din(9), inp(2)=>din(9), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(9), inp(7)=>s2(9),
inp(8)=>req, oup(0)=> s3(9)(0), oup(1) => s3(9)(1));
u71: encoder port map ( inp(0)=>s1, inp(1)=>din(10), inp(2)=>din(10), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(10),
inp(7)=>s2(10), inp(8)=>req, oup(0)=> s3(10)(0), oup(1) => s3(10)(1));
u72: encoder port map ( inp(0)=>s1, inp(1)=>din(11), inp(2)=>din(11), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(11),
inp(7)=>s2(11), inp(8)=>req, oup(0)=> s3(11)(0), oup(1) => s3(11)(1));
u73: encoder port map ( inp(0)=>s1, inp(1)=>din(12), inp(2)=>din(12), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(12),
inp(7)=>s2(12), inp(8)=>req, oup(0)=> s3(12)(0), oup(1) => s3(12)(1));
u74: encoder port map ( inp(0)=>s1, inp(1)=>din(13), inp(2)=>din(13), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(13),
inp(7)=>s2(13), inp(8)=>req, oup(0)=> s3(13)(0), oup(1) => s3(13)(1));
u75: encoder port map ( inp(0)=>s1, inp(1)=>din(14), inp(2)=>din(14), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(14),
inp(7)=>s2(14), inp(8)=>req, oup(0)=> s3(14)(0), oup(1) => s3(14)(1));
u76: encoder port map ( inp(0)=>s1, inp(1)=>din(15), inp(2)=>din(15), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(15),
inp(7)=>s2(15), inp(8)=>req, oup(0)=> s3(15)(0), oup(1) => s3(15)(1));
u77: encoder port map ( inp(0)=>s1, inp(1)=>din(16), inp(2)=>din(16), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(16),
inp(7)=>s2(16), inp(8)=>req, oup(0)=> s3(16)(0), oup(1) => s3(16)(1));
u78: encoder port map ( inp(0)=>s1, inp(1)=>din(17), inp(2)=>din(17), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(17),
inp(7)=>s2(17), inp(8)=>req, oup(0)=> s3(17)(0), oup(1) => s3(17)(1));
u79: encoder port map ( inp(0)=>s1, inp(1)=>din(18), inp(2)=>din(18), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(18),
inp(7)=>s2(18), inp(8)=>req, oup(0)=> s3(18)(0), oup(1) => s3(18)(1));
u80: encoder port map ( inp(0)=>s1, inp(1)=>din(19), inp(2)=>din(19), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(19),
inp(7)=>s2(19), inp(8)=>req, oup(0)=> s3(19)(0), oup(1) => s3(19)(1));
u81: encoder port map ( inp(0)=>s1, inp(1)=>din(20), inp(2)=>din(20), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(20),
inp(7)=>s2(20), inp(8)=>req, oup(0)=> s3(20)(0), oup(1) => s3(20)(1));
u82: encoder port map ( inp(0)=>s1, inp(1)=>din(21), inp(2)=>din(21), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(21),
inp(7)=>s2(21), inp(8)=>req, oup(0)=> s3(21)(0), oup(1) => s3(21)(1));
u83: encoder port map ( inp(0)=>s1, inp(1)=>din(22), inp(2)=>din(22), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(22),
inp(7)=>s2(22), inp(8)=>req, oup(0)=> s3(22)(0), oup(1) => s3(22)(1));
u84: encoder port map ( inp(0)=>s1, inp(1)=>din(23), inp(2)=>din(23), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(23),
inp(7)=>s2(23), inp(8)=>req, oup(0)=> s3(23)(0), oup(1) => s3(23)(1));
u85: encoder port map ( inp(0)=>s1, inp(1)=>din(24), inp(2)=>din(24), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(24),
inp(7)=>s2(24), inp(8)=>req, oup(0)=> s3(24)(0), oup(1) => s3(24)(1));
```

u86: encoder port map (inp(0)=>s1, inp(1)=>din(25), inp(2)=>din(25), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(25),
inp(7)=>s2(25), inp(8)=>req, oup(0)=> s3(25)(0), oup(1) => s3(25)(1));
u124: encoder port map (inp(0)=>s1, inp(1)=>din(26), inp(2)=>din(26), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(26),
inp(7)=>s2(26), inp(8)=>req, oup(0)=> s3(26)(0), oup(1) => s3(26)(1));
u87: encoder port map (inp(0)=>s1, inp(1)=>din(27), inp(2)=>din(27), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(27),
inp(7)=>s2(27), inp(8)=>req, oup(0)=> s3(27)(0), oup(1) => s3(27)(1));
u88: encoder port map (inp(0)=>s1, inp(1)=>din(28), inp(2)=>din(28), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(28),
inp(7)=>s2(28), inp(8)=>req, oup(0)=> s3(28)(0), oup(1) => s3(28)(1));
u89: encoder port map (inp(0)=>s1, inp(1)=>din(29), inp(2)=>din(29), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(29),
inp(7)=>s2(29), inp(8)=>req, oup(0)=> s3(29)(0), oup(1) => s3(29)(1));
u90: encoder port map (inp(0)=>s1, inp(1)=>din(30), inp(2)=>din(30), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(30),
inp(7)=>s2(30), inp(8)=>req, oup(0)=> s3(30)(0), oup(1) => s3(30)(1));
u91: encoder port map (inp(0)=>s1, inp(1)=>din(31), inp(2)=>din(31), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(31),
inp(7)=>s2(31), inp(8)=>req, oup(0)=> s3(31)(0), oup(1) => s3(31)(1));
u92: encoder port map (inp(0)=>s1, inp(1)=>din(32), inp(2)=>din(32), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(32),
inp(7)=>s2(32), inp(8)=>req, oup(0)=> s3(32)(0), oup(1) => s3(32)(1));
u93: encoder port map (inp(0)=>s1, inp(1)=>din(33), inp(2)=>din(33), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(33),
inp(7)=>s2(33), inp(8)=>req, oup(0)=> s3(33)(0), oup(1) => s3(33)(1));
u94: encoder port map (inp(0)=>s1, inp(1)=>din(34), inp(2)=>din(34), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(34),
inp(7)=>s2(34), inp(8)=>req, oup(0)=> s3(34)(0), oup(1) => s3(34)(1));
u95: encoder port map (inp(0)=>s1, inp(1)=>din(35), inp(2)=>din(35), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(35),
inp(7)=>s2(35), inp(8)=>req, oup(0)=> s3(35)(0), oup(1) => s3(35)(1));
u96: encoder port map (inp(0)=>s1, inp(1)=>din(36), inp(2)=>din(36), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(36),
inp(7)=>s2(36), inp(8)=>req, oup(0)=> s3(36)(0), oup(1) => s3(36)(1));
u97: encoder port map (inp(0)=>s1, inp(1)=>din(37), inp(2)=>din(37), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(37),
inp(7)=>s2(37), inp(8)=>req, oup(0)=> s3(37)(0), oup(1) => s3(37)(1));
u98: encoder port map (inp(0)=>s1, inp(1)=>din(38), inp(2)=>din(38), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(38),
inp(7)=>s2(38), inp(8)=>req, oup(0)=> s3(38)(0), oup(1) => s3(38)(1));
u99: encoder port map (inp(0)=>s1, inp(1)=>din(39), inp(2)=>din(39), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(39),
inp(7)=>s2(39), inp(8)=>req, oup(0)=> s3(39)(0), oup(1) => s3(39)(1));
u100: encoder port map (inp(0)=>s1, inp(1)=>din(40), inp(2)=>din(40), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(40),
inp(7)=>s2(40), inp(8)=>req, oup(0)=> s3(40)(0), oup(1) => s3(40)(1));
u101: encoder port map (inp(0)=>s1, inp(1)=>din(41), inp(2)=>din(41), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(41),
inp(7)=>s2(41), inp(8)=>req, oup(0)=> s3(41)(0), oup(1) => s3(41)(1));
u102: encoder port map (inp(0)=>s1, inp(1)=>din(42), inp(2)=>din(42), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(42),
inp(7)=>s2(42), inp(8)=>req, oup(0)=> s3(42)(0), oup(1) => s3(42)(1));
u103: encoder port map (inp(0)=>s1, inp(1)=>din(43), inp(2)=>din(43), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(43),
inp(7)=>s2(43), inp(8)=>req, oup(0)=> s3(43)(0), oup(1) => s3(43)(1));
u104: encoder port map (inp(0)=>s1, inp(1)=>din(44), inp(2)=>din(44), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(44),
inp(7)=>s2(44), inp(8)=>req, oup(0)=> s3(44)(0), oup(1) => s3(44)(1));
u105: encoder port map (inp(0)=>s1, inp(1)=>din(45), inp(2)=>din(45), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(45),
inp(7)=>s2(45), inp(8)=>req, oup(0)=> s3(45)(0), oup(1) => s3(45)(1));
u106: encoder port map (inp(0)=>s1, inp(1)=>din(46), inp(2)=>din(46), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(46),
inp(7)=>s2(46), inp(8)=>req, oup(0)=> s3(46)(0), oup(1) => s3(46)(1));
u107: encoder port map (inp(0)=>s1, inp(1)=>din(47), inp(2)=>din(47), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(47),
inp(7)=>s2(47), inp(8)=>req, oup(0)=> s3(47)(0), oup(1) => s3(47)(1));
u108: encoder port map (inp(0)=>s1, inp(1)=>din(48), inp(2)=>din(48), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(48),
inp(7)=>s2(48), inp(8)=>req, oup(0)=> s3(48)(0), oup(1) => s3(48)(1));
u109: encoder port map (inp(0)=>s1, inp(1)=>din(49), inp(2)=>din(49), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(49),
inp(7)=>s2(49), inp(8)=>req, oup(0)=> s3(49)(0), oup(1) => s3(49)(1));
u110: encoder port map (inp(0)=>s1, inp(1)=>din(50), inp(2)=>din(50), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(50),
inp(7)=>s2(50), inp(8)=>req, oup(0)=> s3(50)(0), oup(1) => s3(50)(1));
u111: encoder port map (inp(0)=>s1, inp(1)=>din(51), inp(2)=>din(51), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(51),
inp(7)=>s2(51), inp(8)=>req, oup(0)=> s3(51)(0), oup(1) => s3(51)(1));
u112: encoder port map (inp(0)=>s1, inp(1)=>din(52), inp(2)=>din(52), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(52),
inp(7)=>s2(52), inp(8)=>req, oup(0)=> s3(52)(0), oup(1) => s3(52)(1));
u113: encoder port map (inp(0)=>s1, inp(1)=>din(53), inp(2)=>din(53), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(53),
inp(7)=>s2(53), inp(8)=>req, oup(0)=> s3(53)(0), oup(1) => s3(53)(1));
u114: encoder port map (inp(0)=>s1, inp(1)=>din(54), inp(2)=>din(54), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(54),
inp(7)=>s2(54), inp(8)=>req, oup(0)=> s3(54)(0), oup(1) => s3(54)(1));
u115: encoder port map (inp(0)=>s1, inp(1)=>din(55), inp(2)=>din(55), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(55),
inp(7)=>s2(55), inp(8)=>req, oup(0)=> s3(55)(0), oup(1) => s3(55)(1));

```
u116: encoder port map ( inp(0)=>s1, inp(1)=>din(56), inp(2)=>din(56), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(56),
inp(7)=>s2(56), inp(8)=>req, oup(0)=> s3(56)(0), oup(1) => s3(56)(1));
u117: encoder port map ( inp(0)=>s1, inp(1)=>din(57), inp(2)=>din(57), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(57),
inp(7)=>s2(57), inp(8)=>req, oup(0)=> s3(57)(0), oup(1) => s3(57)(1));
u118: encoder port map ( inp(0)=>s1, inp(1)=>din(58), inp(2)=>din(58), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(58),
inp(7)=>s2(58), inp(8)=>req, oup(0)=> s3(58)(0), oup(1) => s3(58)(1));
u123: encoder port map ( inp(0)=>s1, inp(1)=>din(59), inp(2)=>din(59), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(59),
inp(7)=>s2(59), inp(8)=>req, oup(0)=> s3(59)(0), oup(1) => s3(59)(1));

u119: OR_gate2 port map ( inp(0)=>s3(0)(0),inp(1)=>s3(0)(1), inp(2)=>s3(1)(0), inp(3)=>s3(1)(1), inp(4)=>s3(2)(0), inp(5)=>s3(2)(1),
inp(6)=>s3(3)(0), inp(7)=>s3(3)(1), inp(8)=>s3(4)(0), inp(9)=>s3(4)(1), inp(10)=>s3(5)(0),inp(11)=>s3(5)(1), inp(12)=>s3(6)(0),
inp(13)=>s3(6)(1),inp(14)=>s3(7)(0),inp(15)=>s3(7)(1), inp(16) => s3(8)(0), inp(17)=> s3(8)(1), inp(18) => s3(9)(0),inp(19)=>s3(9)(1),
inp(20)=>s3(10)(0), inp(21)=>s3(10)(1), inp(22)=>s3(11)(0),inp(23)=>s3(11)(1),  inp(24)=>s3(12)(0),inp(25)=>s3(12)(1),
inp(26)=>s3(13)(0), inp(27)=>s3(13)(1), inp(28)=>s3(14)(0), inp(29)=>s3(14)(1), inp(30)=>s3(15)(0), inp(31)=>s3(15)(1),
inp(32)=>s3(16)(0), inp(33)=>s3(16)(1), inp(34)=>s3(17)(0),inp(35)=>s3(17)(1), inp(36)=>s3(18)(0),
inp(37)=>s3(18)(1),inp(38)=>s3(19)(0),inp(39)=>s3(19)(1), inp(40) => s3(20)(0), inp(41)=> s3(20)(1), inp(42) =>
s3(21)(0),inp(43)=>s3(21)(1), inp(44)=>s3(22)(0), inp(45)=>s3(22)(1), inp(46)=>s3(23)(0), inp(47)=>s3(23)(1),
inp(48)=>s3(24)(0),inp(49)=>s3(24)(1), inp(50)=>s3(25)(0), inp(51)=>s3(25)(1), inp(52)=>s3(26)(0), inp(53)=>s3(26)(1),
inp(54)=>s3(27)(0), inp(55)=>s3(27)(1), inp(56)=>s3(28)(0), inp(57)=>s3(28)(1), inp(58)=>s3(29)(0),inp(59)=>s3(29)(1),
inp(60)=>s3(30)(0), inp(61)=>s3(30)(1),inp(62)=>s3(31)(0),inp(63)=>s3(31)(1), inp(64) => s3(32)(0), inp(65)=> s3(32)(1), inp(66) =>
s3(33)(0),inp(67)=>s3(33)(1), inp(68)=>s3(34)(0), inp(69)=>s3(34)(1), inp(70)=>s3(35)(0), inp(71)=>s3(35)(1),
inp(72)=>s3(36)(0),inp(73)=>s3(36)(1), inp(74)=>s3(37)(0), inp(75)=>s3(37)(1), inp(76)=>s3(38)(0), inp(77)=>s3(38)(1),
inp(78)=>s3(39)(0), inp(79)=>s3(39)(1), inp(80)=>s3(40)(0), inp(81)=>s3(40)(1), inp(82)=>s3(41)(0),inp(83)=>s3(41)(1),
inp(84)=>s3(42)(0), inp(85)=>s3(42)(1),inp(86)=>s3(43)(0),inp(87)=>s3(43)(1), inp(88) => s3(44)(0), inp(89)=> s3(44)(1), inp(90) =>
s3(45)(0),inp(91)=>s3(45)(1), inp(92)=>s3(46)(0), inp(93)=>s3(46)(1), inp(94)=>s3(47)(0), inp(95)=>s3(47)(1),
inp(96)=>s3(48)(0),inp(97)=>s3(48)(1), inp(98)=>s3(49)(0), inp(99)=>s3(49)(1), inp(100)=>s3(50)(0), inp(101)=>s3(50)(1),
inp(102)=>s3(51)(0), inp(103)=>s3(51)(1), inp(104)=>s3(52)(0), inp(105)=>s3(52)(1), inp(106)=>s3(53)(0),inp(107)=>s3(53)(1),
inp(108)=>s3(54)(0), inp(109)=>s3(54)(1),inp(110)=>s3(55)(0),inp(111)=>s3(55)(1), inp(112) => s3(56)(0), inp(113)=> s3(56)(1), inp(114)
=> s3(57)(0),inp(115)=>s3(57)(1), inp(116)=>s3(58)(0), inp(117)=>s3(58)(1),inp(118)=>s3(59)(0), inp(119)=>s3(59)(1), oup =>s4);
u120: d_ff2 port map(inp(0)=> s3(0)(0), inp(1)=> s3(1)(0), inp(2)=> s3(2)(0), inp(3)=> s3(3)(0), inp(4)=> s3(4)(0), inp(5)=>s3(5)(0),
inp(6)=>s3(6)(0), inp(7)=>s3(7)(0), inp(8)=>s3(8)(0), inp(9)=>s3(9)(0), inp(10)=>s3(10)(0), inp(11)=>s3(11)(0),inp(12)=> s3(12)(0),
inp(13)=> s3(13)(0), inp(14)=> s3(14)(0), inp(15)=> s3(15)(0), inp(16)=> s3(16)(0), inp(17)=>s3(17)(0), inp(18)=>s3(18)(0),
inp(19)=>s3(19)(0), inp(20)=>s3(20)(0), inp(21)=>s3(21)(0), inp(22)=>s3(22)(0), inp(23)=>s3(23)(0),inp(24)=> s3(24)(0), inp(25)=>
s3(25)(0), inp(26)=> s3(26)(0), inp(27)=> s3(27)(0), inp(28)=> s3(28)(0), inp(29)=>s3(29)(0), inp(30)=>s3(30)(0), inp(31)=>s3(31)(0),
inp(32)=>s3(32)(0), inp(33)=>s3(33)(0), inp(34)=>s3(34)(0), inp(35)=>s3(35)(0),inp(36)=> s3(36)(0), inp(37)=> s3(37)(0), inp(38)=>
s3(38)(0), inp(39)=> s3(39)(0), inp(40)=> s3(40)(0), inp(41)=>s3(41)(0), inp(42)=>s3(42)(0), inp(43)=>s3(43)(0), inp(44)=>s3(44)(0),
inp(45)=>s3(45)(0), inp(46)=>s3(46)(0), inp(47)=>s3(47)(0),inp(48)=> s3(48)(0), inp(49)=> s3(49)(0), inp(50)=> s3(50)(0), inp(51)=>
s3(51)(0), inp(52)=> s3(52)(0), inp(53)=>s3(53)(0), inp(54)=>s3(54)(0), inp(55)=>s3(55)(0), inp(56)=>s3(56)(0), inp(57)=>s3(57)(0),
inp(58)=>s3(58)(0), inp(59)=>s3(59)(0), oup=> dout, clk=> s4);
u121: inv_gate port map(inp=> s4, oup=> ack);

end Behavioral;


----------file: ram----------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;


entity ram is
port( reset :       in std_logic;
                    din         :  in std_logic_vector(59 downto 0);
                    mem_e :     in std_logic;
                    addr_out    :           out std_logic_vector(9 downto 0);
                    rd          :           in std_logic;
                    wr          :  in std_logic;
                    db_sel:     in std_logic_vector(2 downto 0);
                    req1  :     out std_logic;
                    req2        :           in std_logic;
                    dout  :     out std_logic_vector(51 downto 0));
end ram;
```

```
architecture Behavioral of ram is

component lso is
port( reset  :                  in std_logic;
                  aclk          :                  out std_logic);
end component;

component Counter  is
port( reset, clk          : in std_logic;
                  dout                        : out std_logic);
end component;

component Multiplex is
port(       din1          : in std_logic_vector(15 downto 0);
                  din2          : in std_logic_vector(15 downto 0);
                  dout          : out std_logic_vector(15 downto 0);
                  sel           : in std_logic_vector(2 downto 0));
end component;

component logic3 is
port(       mem_e :   in std_logic;
                  reset         :           in std_logic;
                  rd            :           in std_logic;
                  wr            :  in std_logic;
                  addr          :           in std_logic_vector(9 downto 0);
                  din           :           in std_logic_vector(15 downto 0);
                  dout          :           out std_logic_vector(15 downto 0));
end component;

component parity_check is
port( din    : in std_logic_vector(15 downto 0);
                  oup           : out std_logic);
end component;

component zero_check is
port( din    : in std_logic_vector(15 downto 0);
                  oup           : out std_logic);
end component;

component ecr3 is
port(  din : in std_logic_vector (51 downto 0);
      req     : in std_logic ;
                   ack          : out std_logic ;
      dout : out std_logic_vector (51 downto 0));
end component;

signal s0, s1, s2, s3, s4, ack: std_logic;
signal d1, d2: std_logic;
signal d3: std_logic_vector(15 downto 0);
signal ram_out: std_logic_vector(15 downto 0);
begin
req1<= s0;
s1<= not s0;
s2 <= reset and s1;
s4 <= s0 and req2;

U1: lso port map( reset => s2, aclk => s3);
U2: Counter port map( reset => ack, clk => s3, dout => s0);
U3: logic3 port map( reset => reset, addr => din(26 downto 17), rd => rd, wr => wr, din => din(16 downto 1), dout => ram_out, mem_e =>
mem_e);
```

```
U4: ecr3 port map( din(51) => din(0), din(50 downto 19) => din(59 downto 28), din(18) => d1, din(17) => d2, din(16) => din(27), din(15
downto 0)=> d3, req=> s4, ack => ack, dout => dout);
U5: parity_check port map( din => din(34 downto 19), oup => d1);
U6: zero_check port map( din => din(34 downto 19), oup => d2);
U7: Multiplex port map(din1 => ram_out, din2 => din(16 downto 1), dout => d3, sel => db_sel);

addr_out <= din(26 downto 17);
end Behavioral;
```

----------------------------------file :**logic3**---

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

entity logic3 is
port( addr: in std_logic_vector(9 downto 0);
                    reset : in std_logic;
                    rd          :          in std_logic;
                    wr          :  in std_logic;
                    mem_e :   in std_logic;
                    din         :          in std_logic_vector(15 downto 0);
                    dout: out std_logic_vector(15 downto 0));
end logic3;

architecture Behavioral of logic3 is

function convert(din: std_logic_vector(9 downto 0))
return integer is
variable result: integer := 0;
begin

for i IN 9 downto 0 loop

if(din(i) = '1') then
result := result*2 + 1;
elsif(din(i) = '0') then
result := result*2;
end if;
end loop;
return result;
end function convert;

signal adint: integer range 0 to 1023:=0;
type RAM is array(0 to 1023) of std_logic_vector(15 downto 0);
signal Ramdata:RAM:=
("0000000000000000","0000001111000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
```

"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",

"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",

"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",

```
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000",
"0000000000000000","0000000000000000","0000000000000000","0000000000000000");

begin

adint<=convert(addr);
dout<=Ramdata(adint) when reset = '1' and rd = '1' else
                    "ZZZZZZZZZZZZZZZZ" when reset = '0';

Ramdata(adint) <= din when reset = '1' and wr = '1';
end Behavioral;
```

----------file: **ecr3**----------

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;

entity ecr3 is
port( din : in std_logic_vector(51 downto 0);
      req : in std_logic ;
                          ack : out std_logic ;
      dout : out std_logic_vector(51 downto 0));
end ecr3;

architecture Behavioral of ecr3 is

component OR_gate3 is
port( inp : in std_logic_vector(103 downto 0);
                          oup: out std_logic);
end component;

component encoder is
port(inp: in std_logic_vector (8 downto 0) ;
    oup : out std_logic_vector(1 downto 0));
end component;

component and_gate is
port( inp : in std_logic;
                          oup: out std_logic);
end component;

component inv_gate is
port( inp : in std_logic;
                          oup: out std_logic);
end component;

component d_ff3 is
port(inp : in std_logic_vector(51 downto 0);
             oup: out std_logic_vector(51 downto 0);
             clk: in std_logic);
end component;

component inv_gate2 is
port( din: in std_logic_vector(51 downto 0);
                          dout : out std_logic_vector(51 downto 0));
end component;

type dataout is array (51 downto 0) of std_logic_vector(1 downto 0);
signal s1: std_logic;
signal s2: std_logic_vector(51 downto 0);
signal s3: dataout;
signal s4: std_logic;

begin
u1: inv_gate port map( inp=>req, oup=>s1);
U2: inv_gate2 port map( din => din, dout => s2);
u61: encoder port map ( inp(0)=>s1, inp(1)=>din(0), inp(2)=>din(0), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(0), inp(7)=>s2(0),
inp(8)=>req, oup(0)=> s3(0)(0), oup(1) => s3(0)(1));
u62: encoder port map ( inp(0)=>s1, inp(1)=>din(1), inp(2)=>din(1), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(1), inp(7)=>s2(1),
inp(8)=>req, oup(0)=> s3(1)(0), oup(1) => s3(1)(1));
```

u63: encoder port map (inp(0)=>s1, inp(1)=>din(2), inp(2)=>din(2), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(2), inp(7)=>s2(2), inp(8)=>req, oup(0)=> s3(2)(0), oup(1) => s3(2)(1));
u64: encoder port map (inp(0)=>s1, inp(1)=>din(3), inp(2)=>din(3), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(3), inp(7)=>s2(3), inp(8)=>req, oup(0)=> s3(3)(0), oup(1) => s3(3)(1));
u65: encoder port map (inp(0)=>s1, inp(1)=>din(4), inp(2)=>din(4), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(4), inp(7)=>s2(4), inp(8)=>req, oup(0)=> s3(4)(0), oup(1) => s3(4)(1));
u66: encoder port map (inp(0)=>s1, inp(1)=>din(5), inp(2)=>din(5), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(5), inp(7)=>s2(5), inp(8)=>req, oup(0)=> s3(5)(0), oup(1) => s3(5)(1));
u67: encoder port map (inp(0)=>s1, inp(1)=>din(6), inp(2)=>din(6), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(6), inp(7)=>s2(6), inp(8)=>req, oup(0)=> s3(6)(0), oup(1) => s3(6)(1));
u68: encoder port map (inp(0)=>s1, inp(1)=>din(7), inp(2)=>din(7), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(7), inp(7)=>s2(7), inp(8)=>req, oup(0)=> s3(7)(0), oup(1) => s3(7)(1));
u69: encoder port map (inp(0)=>s1, inp(1)=>din(8), inp(2)=>din(8), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(8), inp(7)=>s2(8), inp(8)=>req, oup(0)=> s3(8)(0), oup(1) => s3(8)(1));
u70: encoder port map (inp(0)=>s1, inp(1)=>din(9), inp(2)=>din(9), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(9), inp(7)=>s2(9), inp(8)=>req, oup(0)=> s3(9)(0), oup(1) => s3(9)(1));
u71: encoder port map (inp(0)=>s1, inp(1)=>din(10), inp(2)=>din(10), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(10), inp(7)=>s2(10), inp(8)=>req, oup(0)=> s3(10)(0), oup(1) => s3(10)(1));
u72: encoder port map (inp(0)=>s1, inp(1)=>din(11), inp(2)=>din(11), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(11), inp(7)=>s2(11), inp(8)=>req, oup(0)=> s3(11)(0), oup(1) => s3(11)(1));
u73: encoder port map (inp(0)=>s1, inp(1)=>din(12), inp(2)=>din(12), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(12), inp(7)=>s2(12), inp(8)=>req, oup(0)=> s3(12)(0), oup(1) => s3(12)(1));
u74: encoder port map (inp(0)=>s1, inp(1)=>din(13), inp(2)=>din(13), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(13), inp(7)=>s2(13), inp(8)=>req, oup(0)=> s3(13)(0), oup(1) => s3(13)(1));
u75: encoder port map (inp(0)=>s1, inp(1)=>din(14), inp(2)=>din(14), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(14), inp(7)=>s2(14), inp(8)=>req, oup(0)=> s3(14)(0), oup(1) => s3(14)(1));
u76: encoder port map (inp(0)=>s1, inp(1)=>din(15), inp(2)=>din(15), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(15), inp(7)=>s2(15), inp(8)=>req, oup(0)=> s3(15)(0), oup(1) => s3(15)(1));
u77: encoder port map (inp(0)=>s1, inp(1)=>din(16), inp(2)=>din(16), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(16), inp(7)=>s2(16), inp(8)=>req, oup(0)=> s3(16)(0), oup(1) => s3(16)(1));
u78: encoder port map (inp(0)=>s1, inp(1)=>din(17), inp(2)=>din(17), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(17), inp(7)=>s2(17), inp(8)=>req, oup(0)=> s3(17)(0), oup(1) => s3(17)(1));
u79: encoder port map (inp(0)=>s1, inp(1)=>din(18), inp(2)=>din(18), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(18), inp(7)=>s2(18), inp(8)=>req, oup(0)=> s3(18)(0), oup(1) => s3(18)(1));
u80: encoder port map (inp(0)=>s1, inp(1)=>din(19), inp(2)=>din(19), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(19), inp(7)=>s2(19), inp(8)=>req, oup(0)=> s3(19)(0), oup(1) => s3(19)(1));
u81: encoder port map (inp(0)=>s1, inp(1)=>din(20), inp(2)=>din(20), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(20), inp(7)=>s2(20), inp(8)=>req, oup(0)=> s3(20)(0), oup(1) => s3(20)(1));
u82: encoder port map (inp(0)=>s1, inp(1)=>din(21), inp(2)=>din(21), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(21), inp(7)=>s2(21), inp(8)=>req, oup(0)=> s3(21)(0), oup(1) => s3(21)(1));
u83: encoder port map (inp(0)=>s1, inp(1)=>din(22), inp(2)=>din(22), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(22), inp(7)=>s2(22), inp(8)=>req, oup(0)=> s3(22)(0), oup(1) => s3(22)(1));
u84: encoder port map (inp(0)=>s1, inp(1)=>din(23), inp(2)=>din(23), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(23), inp(7)=>s2(23), inp(8)=>req, oup(0)=> s3(23)(0), oup(1) => s3(23)(1));
u85: encoder port map (inp(0)=>s1, inp(1)=>din(24), inp(2)=>din(24), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(24), inp(7)=>s2(24), inp(8)=>req, oup(0)=> s3(24)(0), oup(1) => s3(24)(1));
u86: encoder port map (inp(0)=>s1, inp(1)=>din(25), inp(2)=>din(25), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(25), inp(7)=>s2(25), inp(8)=>req, oup(0)=> s3(25)(0), oup(1) => s3(25)(1));
u124: encoder port map (inp(0)=>s1, inp(1)=>din(26), inp(2)=>din(26), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(26), inp(7)=>s2(26), inp(8)=>req, oup(0)=> s3(26)(0), oup(1) => s3(26)(1));
u87: encoder port map (inp(0)=>s1, inp(1)=>din(27), inp(2)=>din(27), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(27), inp(7)=>s2(27), inp(8)=>req, oup(0)=> s3(27)(0), oup(1) => s3(27)(1));
u88: encoder port map (inp(0)=>s1, inp(1)=>din(28), inp(2)=>din(28), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(28), inp(7)=>s2(28), inp(8)=>req, oup(0)=> s3(28)(0), oup(1) => s3(28)(1));
u89: encoder port map (inp(0)=>s1, inp(1)=>din(29), inp(2)=>din(29), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(29), inp(7)=>s2(29), inp(8)=>req, oup(0)=> s3(29)(0), oup(1) => s3(29)(1));
u90: encoder port map (inp(0)=>s1, inp(1)=>din(30), inp(2)=>din(30), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(30), inp(7)=>s2(30), inp(8)=>req, oup(0)=> s3(30)(0), oup(1) => s3(30)(1));
u91: encoder port map (inp(0)=>s1, inp(1)=>din(31), inp(2)=>din(31), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(31), inp(7)=>s2(31), inp(8)=>req, oup(0)=> s3(31)(0), oup(1) => s3(31)(1));
u92: encoder port map (inp(0)=>s1, inp(1)=>din(32), inp(2)=>din(32), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(32), inp(7)=>s2(32), inp(8)=>req, oup(0)=> s3(32)(0), oup(1) => s3(32)(1));

u93: encoder port map (inp(0)=>s1, inp(1)=>din(33), inp(2)=>din(33), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(33),
inp(7)=>s2(33), inp(8)=>req, oup(0)=> s3(33)(0), oup(1) => s3(33)(1));
u94: encoder port map (inp(0)=>s1, inp(1)=>din(34), inp(2)=>din(34), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(34),
inp(7)=>s2(34), inp(8)=>req, oup(0)=> s3(34)(0), oup(1) => s3(34)(1));
u95: encoder port map (inp(0)=>s1, inp(1)=>din(35), inp(2)=>din(35), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(35),
inp(7)=>s2(35), inp(8)=>req, oup(0)=> s3(35)(0), oup(1) => s3(35)(1));
u96: encoder port map (inp(0)=>s1, inp(1)=>din(36), inp(2)=>din(36), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(36),
inp(7)=>s2(36), inp(8)=>req, oup(0)=> s3(36)(0), oup(1) => s3(36)(1));
u97: encoder port map (inp(0)=>s1, inp(1)=>din(37), inp(2)=>din(37), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(37),
inp(7)=>s2(37), inp(8)=>req, oup(0)=> s3(37)(0), oup(1) => s3(37)(1));
u98: encoder port map (inp(0)=>s1, inp(1)=>din(38), inp(2)=>din(38), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(38),
inp(7)=>s2(38), inp(8)=>req, oup(0)=> s3(38)(0), oup(1) => s3(38)(1));
u99: encoder port map (inp(0)=>s1, inp(1)=>din(39), inp(2)=>din(39), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(39),
inp(7)=>s2(39), inp(8)=>req, oup(0)=> s3(39)(0), oup(1) => s3(39)(1));
u100: encoder port map (inp(0)=>s1, inp(1)=>din(40), inp(2)=>din(40), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(40),
inp(7)=>s2(40), inp(8)=>req, oup(0)=> s3(40)(0), oup(1) => s3(40)(1));
u101: encoder port map (inp(0)=>s1, inp(1)=>din(41), inp(2)=>din(41), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(41),
inp(7)=>s2(41), inp(8)=>req, oup(0)=> s3(41)(0), oup(1) => s3(41)(1));
u102: encoder port map (inp(0)=>s1, inp(1)=>din(42), inp(2)=>din(42), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(42),
inp(7)=>s2(42), inp(8)=>req, oup(0)=> s3(42)(0), oup(1) => s3(42)(1));
u103: encoder port map (inp(0)=>s1, inp(1)=>din(43), inp(2)=>din(43), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(43),
inp(7)=>s2(43), inp(8)=>req, oup(0)=> s3(43)(0), oup(1) => s3(43)(1));
u104: encoder port map (inp(0)=>s1, inp(1)=>din(44), inp(2)=>din(44), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(44),
inp(7)=>s2(44), inp(8)=>req, oup(0)=> s3(44)(0), oup(1) => s3(44)(1));
u105: encoder port map (inp(0)=>s1, inp(1)=>din(45), inp(2)=>din(45), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(45),
inp(7)=>s2(45), inp(8)=>req, oup(0)=> s3(45)(0), oup(1) => s3(45)(1));
u106: encoder port map (inp(0)=>s1, inp(1)=>din(46), inp(2)=>din(46), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(46),
inp(7)=>s2(46), inp(8)=>req, oup(0)=> s3(46)(0), oup(1) => s3(46)(1));
u107: encoder port map (inp(0)=>s1, inp(1)=>din(47), inp(2)=>din(47), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(47),
inp(7)=>s2(47), inp(8)=>req, oup(0)=> s3(47)(0), oup(1) => s3(47)(1));
u108: encoder port map (inp(0)=>s1, inp(1)=>din(48), inp(2)=>din(48), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(48),
inp(7)=>s2(48), inp(8)=>req, oup(0)=> s3(48)(0), oup(1) => s3(48)(1));
u109: encoder port map (inp(0)=>s1, inp(1)=>din(49), inp(2)=>din(49), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(49),
inp(7)=>s2(49), inp(8)=>req, oup(0)=> s3(49)(0), oup(1) => s3(49)(1));
u110: encoder port map (inp(0)=>s1, inp(1)=>din(50), inp(2)=>din(50), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(50),
inp(7)=>s2(50), inp(8)=>req, oup(0)=> s3(50)(0), oup(1) => s3(50)(1));
u111: encoder port map (inp(0)=>s1, inp(1)=>din(51), inp(2)=>din(51), inp(3)=>req, inp(4)=>s1, inp(5)=>s1,inp(6)=>s2(51),
inp(7)=>s2(51), inp(8)=>req, oup(0)=> s3(51)(0), oup(1) => s3(51)(1));

u119: OR_gate3 port map (inp(0)=>s3(0)(0),inp(1)=>s3(0)(1), inp(2)=>s3(1)(0), inp(3)=>s3(1)(1), inp(4)=>s3(2)(0), inp(5)=>s3(2)(1),
inp(6)=>s3(3)(0), inp(7)=>s3(3)(1), inp(8)=>s3(4)(0), inp(9)=>s3(4)(1), inp(10)=>s3(5)(0),inp(11)=>s3(5)(1), inp(12)=>s3(6)(0),
inp(13)=>s3(6)(1),inp(14)=>s3(7)(0),inp(15)=>s3(7)(1), inp(16) => s3(8)(0), inp(17)=> s3(8)(1), inp(18) => s3(9)(0),inp(19)=>s3(9)(1),
inp(20)=>s3(10)(0), inp(21)=>s3(10)(1), inp(22)=>s3(11)(0),inp(23)=>s3(11)(1), inp(24)=>s3(12)(0),inp(25)=>s3(12)(1),
inp(26)=>s3(13)(0), inp(27)=>s3(13)(1), inp(28)=>s3(14)(0), inp(29)=>s3(14)(1), inp(30)=>s3(15)(0), inp(31)=>s3(15)(1),
inp(32)=>s3(16)(0), inp(33)=>s3(16)(1), inp(34)=>s3(17)(0),inp(35)=>s3(17)(1), inp(36)=>s3(18)(0),
inp(37)=>s3(18)(1),inp(38)=>s3(19)(0),inp(39)=>s3(19)(1), inp(40) => s3(20)(0), inp(41)=> s3(20)(1), inp(42) =>
s3(21)(0),inp(43)=>s3(21)(1), inp(44)=>s3(22)(0), inp(45)=>s3(22)(1), inp(46)=>s3(23)(0), inp(47)=>s3(23)(1),
inp(48)=>s3(24)(0),inp(49)=>s3(24)(1), inp(50)=>s3(25)(0), inp(51)=>s3(25)(1), inp(52)=>s3(26)(0), inp(53)=>s3(26)(1),
inp(54)=>s3(27)(0), inp(55)=>s3(27)(1), inp(56)=>s3(28)(0), inp(57)=>s3(28)(1), inp(58)=>s3(29)(0),inp(59)=>s3(29)(1),
inp(60)=>s3(30)(0), inp(61)=>s3(30)(1),inp(62)=>s3(31)(0),inp(63)=>s3(31)(1), inp(64) => s3(32)(0), inp(65)=> s3(32)(1), inp(66) =>
s3(33)(0),inp(67)=>s3(33)(1), inp(68)=>s3(34)(0), inp(69)=>s3(34)(1), inp(70)=>s3(35)(0), inp(71)=>s3(35)(1),
inp(72)=>s3(36)(0),inp(73)=>s3(36)(1), inp(74)=>s3(37)(0), inp(75)=>s3(37)(1), inp(76)=>s3(38)(0), inp(77)=>s3(38)(1),
inp(78)=>s3(39)(0), inp(79)=>s3(39)(1), inp(80)=>s3(40)(0), inp(81)=>s3(40)(1), inp(82)=>s3(41)(0),inp(83)=>s3(41)(1),
inp(84)=>s3(42)(0), inp(85)=>s3(42)(1),inp(86)=>s3(43)(0),inp(87)=>s3(43)(1), inp(88) => s3(44)(0), inp(89)=> s3(44)(1), inp(90) =>
s3(45)(0),inp(91)=>s3(45)(1), inp(92)=>s3(46)(0), inp(93)=>s3(46)(1), inp(94)=>s3(47)(0), inp(95)=>s3(47)(1),
inp(96)=>s3(48)(0),inp(97)=>s3(48)(1), inp(98)=>s3(49)(0), inp(99)=>s3(49)(1), inp(100)=>s3(50)(0), inp(101)=>s3(50)(1),
inp(102)=>s3(51)(0), inp(103)=>s3(51)(1), oup =>s4);
u120: d_ff3 port map(inp(0)=> s3(0)(0), inp(1)=> s3(1)(0), inp(2)=> s3(2)(0), inp(3)=> s3(3)(0), inp(4)=> s3(4)(0), inp(5)=>s3(5)(0),
inp(6)=>s3(6)(0), inp(7)=>s3(7)(0), inp(8)=>s3(8)(0), inp(9)=>s3(9)(0), inp(10)=>s3(10)(0), inp(11)=>s3(11)(0),inp(12)=> s3(12)(0),
inp(13)=> s3(13)(0), inp(14)=> s3(14)(0), inp(15)=> s3(15)(0), inp(16)=> s3(16)(0), inp(17)=>s3(17)(0), inp(18)=>s3(18)(0),
inp(19)=>s3(19)(0), inp(20)=>s3(20)(0), inp(21)=>s3(21)(0), inp(22)=>s3(22)(0), inp(23)=>s3(23)(0),inp(24)=> s3(24)(0), inp(25)=>

```
s3(25)(0), inp(26)=> s3(26)(0), inp(27)=> s3(27)(0), inp(28)=> s3(28)(0), inp(29)=>s3(29)(0), inp(30)=>s3(30)(0), inp(31)=>s3(31)(0),
inp(32)=>s3(32)(0), inp(33)=>s3(33)(0), inp(34)=>s3(34)(0), inp(35)=>s3(35)(0),inp(36)=> s3(36)(0), inp(37)=> s3(37)(0), inp(38)=>
s3(38)(0), inp(39)=> s3(39)(0), inp(40)=> s3(40)(0), inp(41)=>s3(41)(0), inp(42)=>s3(42)(0), inp(43)=>s3(43)(0), inp(44)=>s3(44)(0),
inp(45)=>s3(45)(0), inp(46)=>s3(46)(0), inp(47)=>s3(47)(0),inp(48)=> s3(48)(0), inp(49)=> s3(49)(0), inp(50)=> s3(50)(0), inp(51)=>
s3(51)(0), oup=> dout, clk=> s4);
u121: inv_gate port map(inp=> s4, oup=> ack);

end Behavioral;
```

----------file: **parity check**----------

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


entity parity_check is
port( din    : in std_logic_vector(15 downto 0);
                        oup          : out std_logic);
end parity_check;

architecture Behavioral of parity_check is

begin
oup <= din(0) xor din(1) xor din(2) xor din(3) xor din(4) xor din(5) xor din(6) xor din(7) xor din(8) xor din(9) xor din(10) xor din(11) xor
din(12) xor din(13) xor din(14) xor din(15);

end Behavioral;
```

----------file **: zero check**----------

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


entity zero_check is
port( din    : in std_logic_vector(15 downto 0);
                        oup          : out std_logic);
end zero_check;

architecture Behavioral of zero_check is

begin
oup <= din(0) or din(1) or din(2) or din(3) or din(4) or din(5) or din(6) or din(7) or din(8) or din(9) or din(10) or din(11) or din(12) or
din(13) or din(14) or din(15);

end Behavioral;
```

----------file : **multiplex**----------

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


entity Multiplex is
port(        din1        : in std_logic_vector(15 downto 0);
                        din2        : in std_logic_vector(15 downto 0);
                        dout        : out std_logic_vector(15 downto 0);
                        sel         : in std_logic_vector(2 downto 0));
end Multiplex;

architecture Behavioral of Multiplex is
signal s: std_logic;
begin
```

```
s <= sel(2) and (not sel(1)) and (not sel(0));

dout <= din1 when s = '1' else
                din2 when s = '0';

end Behavioral;
```

----------file: **reg_set**----------

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

entity Reg_Set is
port( din_A                     : in std_logic_vector(15 downto 0);
        din_ACC                 : in std_logic_vector(15 downto 0);
        din_B                   : in std_logic_vector(15 downto 0);
        din_F                   : in std_logic_vector(15 downto 0);
        dout_A                  : out std_logic_vector(15 downto 0);
        dout_ACC                : out std_logic_vector(15 downto 0);
        dout_B                  : out std_logic_vector(15 downto 0);
        dout_F                  : out std_logic_vector(15 downto 0);
        ld_flags                : in std_logic;
        flags                   : in std_logic_vector(2 downto 0);
        ld_A                    : in std_logic;
        ld_ACC                  : in std_logic;
        ld_B                    : in std_logic;
        ld_F                    : in std_logic);
end Reg_Set;

architecture Behavioral of Reg_Set is
component Reg_A is
port(din        : in std_logic_vector(15 downto 0);
        ld_A    : in std_logic;
        dout    : out std_logic_vector(15 downto 0));
end component;

component Reg_ACC is
port(din        : in std_logic_vector(15 downto 0);
        ld_ACC  : in std_logic;
        dout    : out std_logic_vector(15 downto 0));
end component;

component Reg_B is
port(din        : in std_logic_vector(15 downto 0);
        ld_B    : in std_logic;
        dout    : out std_logic_vector(15 downto 0));
end component;

component Reg_Flag is
port(din                : in std_logic_vector(15 downto 0);
        ld_F            : in std_logic;
        ld_flags        : in std_logic;
        flags           : in std_logic_vector(2 downto 0);
        dout            : out std_logic_vector(15 downto 0));
end component;


begin
U1: Reg_A port map(din=>din_A, dout => dout_A, ld_A => ld_A);
U2: Reg_ACC port map(din=>din_ACC, dout => dout_ACC, ld_ACC => ld_ACC);
U3: Reg_B port map(din=>din_B, dout => dout_B, ld_B => ld_B);
U4: Reg_Flag port map(din=>din_F, dout => dout_F, ld_F => ld_F, flags => flags, ld_flags => ld_flags);
```

```
end Behavioral;
```

----------file: **Reg A**----------

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


entity Reg_A is
port(din        : in std_logic_vector(15 downto 0);
     ld_A       : in std_logic;
     dout       : out std_logic_vector(15 downto 0));
end Reg_A;

architecture Behavioral of Reg_A is
signal s: std_logic_vector(15 downto 0);
begin
dout <= s;

s <= din when ld_A = '1' else
                s  when ld_A = '0';

end Behavioral;
```

----------file **: Reg_ACC**----------

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


entity Reg_ACC is
port(din        : in std_logic_vector(15 downto 0);
     ld_ACC     : in std_logic;
     dout       : out std_logic_vector(15 downto 0));
end Reg_ACC;

architecture Behavioral of Reg_ACC is
signal s: std_logic_vector(15 downto 0);
begin
dout <= s;

s <= din when ld_ACC = '1' else
                s  when ld_ACC = '0';

end Behavioral;
```

----------file : **Reg B**----------

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

entity Reg_B is
port(din        : in std_logic_vector(15 downto 0);
     ld_B       : in std_logic;
     dout       : out std_logic_vector(15 downto 0));
end Reg_B;

architecture Behavioral of Reg_B is
signal s: std_logic_vector(15 downto 0);
begin
```

```
dout <= s;

s <= din when ld_B = '1' else
                        s  when ld_B = '0';

end Behavioral;
```

----------file : **Reg_Flag**----------

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

entity Reg_Flag is
port(din                        : in std_logic_vector(15 downto 0);
        ld_F                    : in std_logic;
        ld_flags   : in std_logic;
        flags                   : in std_logic_vector(2 downto 0);
        dout                    : out std_logic_vector(15 downto 0));
end Reg_Flag;

architecture Behavioral of Reg_Flag is
signal s1: std_logic_vector(15 downto 0);
signal s2: std_logic_vector(2 downto 0);
begin
dout <= s1 when ld_F = '1' and ld_flags = '0' else
                         s1(15 downto 3) & s2 when ld_flags = '1' and ld_F = '0';

s1 <= din when ld_F = '1' else
                        s1 when ld_F = '0' ;

s2 <= flags when ld_flags = '1' else
                        s2 when ld_flags = '0';

end Behavioral;
```

----------file : **databus**----------

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

entity databus is
port(A: in std_logic_vector(15 downto 0);
        B: in std_logic_vector(15 downto 0);
        ACC: in std_logic_vector(15 downto 0);
        Flag:in std_logic_vector(15 downto 0);
        Inp: in std_logic_vector(15 downto 0);
        sel: in std_logic_vector(2 downto 0);
        data: out std_logic_vector(15 downto 0));
end databus;

architecture Behavioral of databus is
component mux is
port ( a,b,c,d,e : in std_logic ;
```

```
      y : out std_logic;
                        s : in std_logic_vector ( 2 downto 0 ));
end component;
begin
M0: mux port map(a=>A(0), b=> B(0), c=> ACC(0), e=> Inp(0), d => Flag(0), s=>sel, y=> data(0));
M1: mux port map(a=>A(1), b=> B(1), c=> ACC(1), e=> Inp(1), d => Flag(1), s=>sel, y=> data(1));
M2: mux port map(a=>A(2), b=> B(2), c=> ACC(2), e=> Inp(2), d => Flag(2), s=>sel, y=> data(2));
M3: mux port map(a=>A(3), b=> B(3), c=> ACC(3), e=> Inp(3), d => Flag(3), s=>sel, y=> data(3));
M4: mux port map(a=>A(4), b=> B(4), c=> ACC(4), e=> Inp(4), d => Flag(4), s=>sel, y=> data(4));
M5: mux port map(a=>A(5), b=> B(5), c=> ACC(5), e=> Inp(5), d => Flag(5), s=>sel, y=> data(5));
M6: mux port map(a=>A(6), b=> B(6), c=> ACC(6), e=> Inp(6), d => Flag(6), s=>sel, y=> data(6));
M7: mux port map(a=>A(7), b=> B(7), c=> ACC(7), e=> Inp(7), d => Flag(7), s=>sel, y=> data(7));
M8: mux port map(a=>A(8), b=> B(8), c=> ACC(8), e=> Inp(8), d => Flag(8), s=>sel, y=> data(8));
M9: mux port map(a=>A(9), b=> B(9), c=> ACC(9), e=> Inp(9), d => Flag(9), s=>sel, y=> data(9));
M10: mux port map(a=>A(10), b=> B(10), c=> ACC(10), e=> Inp(10), d => Flag(10), s=>sel, y=> data(10));
M11: mux port map(a=>A(11), b=> B(11), c=> ACC(11), e=> Inp(11), d => Flag(11), s=>sel, y=> data(11));
M12: mux port map(a=>A(12), b=> B(12), c=> ACC(12), e=> Inp(12), d => Flag(12), s=>sel, y=> data(12));
M13: mux port map(a=>A(13), b=> B(13), c=> ACC(13), e=> Inp(13), d => Flag(13), s=>sel, y=> data(13));
M14: mux port map(a=>A(14), b=> B(14), c=> ACC(14), e=> Inp(14), d => Flag(14), s=>sel, y=> data(14));
M15: mux port map(a=>A(15), b=> B(15), c=> ACC(15), e=> Inp(15), d => Flag(15), s=>sel, y=> data(15));


end Behavioral;
```

----------file : **mux**----------

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


entity mux is
port ( a,b,c,d,e: in std_logic ;
      y : out std_logic;
                        s : in std_logic_vector ( 2 downto 0 ));
end mux;

architecture Behavioral of mux is

begin
y <= ( a and (not s(2)) and (not s(1)) and (not s(0))) or ( b and (not s(2)) and (not s(1)) and s(0)) or ( c and (not s(2)) and s(1) and (not s(0)))
or ( d and (not s(2)) and s(1) and s(0)) or ( e and s(2) and (not s(1)) and s(0));

end Behavioral;
```

----------file : multiplex1----------

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

entity Multiplex1 is
port(       din1        : in std_logic_vector(15 downto 0);
                        din2        : in std_logic_vector(15 downto 0);
                        dout        : out std_logic_vector(15 downto 0);
                        sel         : in std_logic);
end Multiplex1;
```

```
architecture Behavioral of Multiplex1 is
begin

dout <= din1 when sel = '1' else
                        din2 when sel = '0';

end Behavioral;
```