ELEC 457

Assignment-4

Sait Berk VARIMLI

141024054

I prepare this assignment in 14 hour.

Problem 1

Design and simulate an **8-bit adder** using `numeric_std` package supplied `+` operator. Compare utilization and timing characteristics with the previous two adders that you designed in assignment 3.

Design Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity adder8b is
  port
  (
    a, b : in unsigned(7 downto 0);
    sum      : out unsigned(7 downto 0);
    carry_out : out std_logic
  );
end entity adder8b;
architecture Behavioral of adder8b is
  signal temp : unsigned(8 downto 0);
begin
  temp <= ("0" & a) + b;
  sum      <= temp(7 downto 0);
  carry_out <= temp(8);
end architecture Behavioral;
```

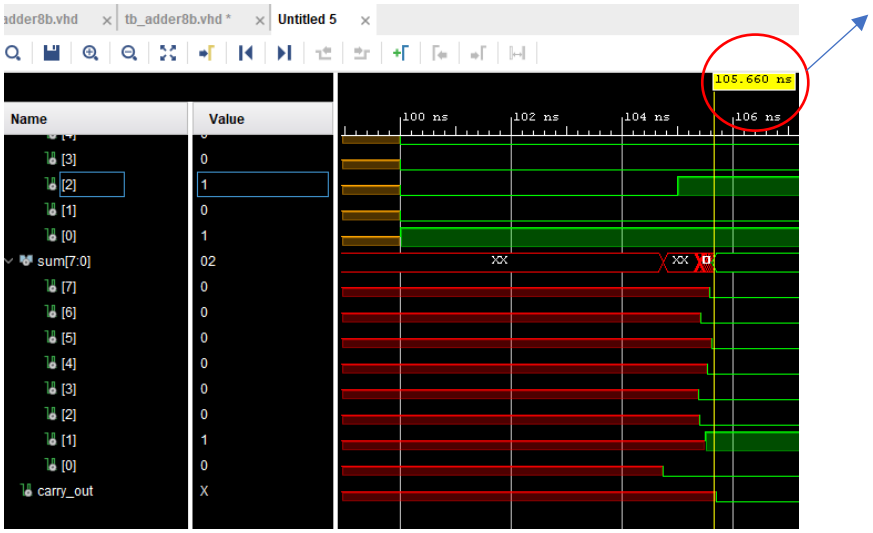
Testbench code:

```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.ALL;
entity tb_adder8b is

END tb_adder8b;

architecture arch12 of tb_adder8b is
component adder8b is
port (a, b : in unsigned(7 downto 0);
      carry_out:out std_logic;
      sum:out unsigned(7 downto 0));
end component;
 signal a,b,sum:unsigned(7 downto 0);
 signal carry_out :std_logic;
begin

 a0 : adder8b port map(
 a => a , b => b , sum => sum, carry_out => carry_out );
 process
 begin
 wait for 100 ns;
 a <= "00000001";
 b <= "00000001";
 wait for 30 ns;
 a <= "00001001";
 b <= "00000101";
 wait for 30 ns;
 a <= "00010001";
 b <= "00001001";
```

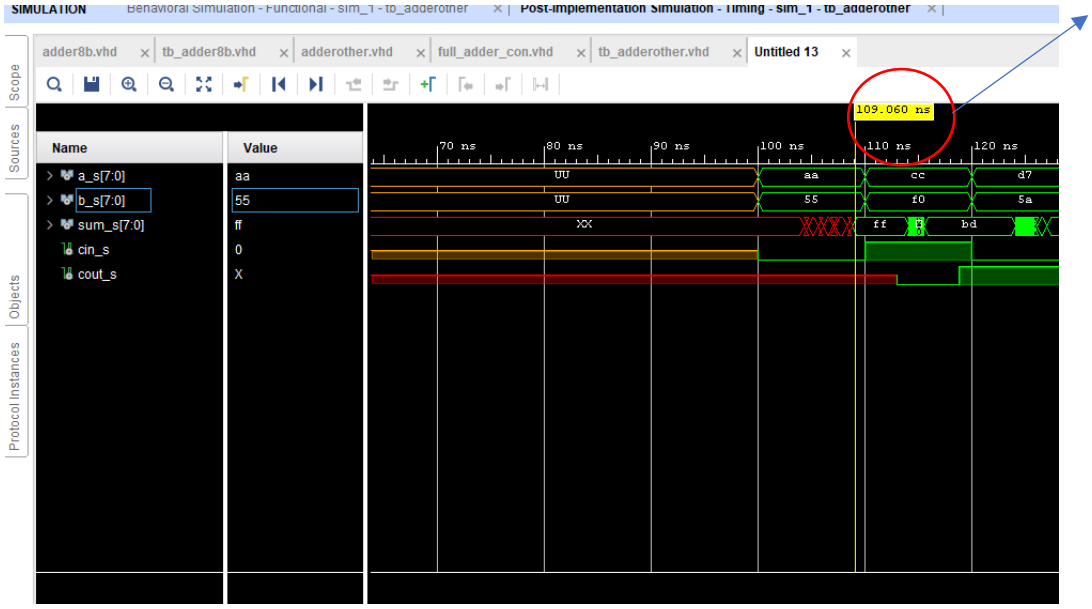
```
  wait for 30 ns;
  a <= "01001101";
  b <= "00110101";
  wait for 30 ns;
  a <= "11011101";
  b <= "11110001";
  wait for 30 ns;
  a <= "11100111";
  b <= "01011101";
  wait for 30 ns;
  a <= "11111111";
  b <= "11111111";
  wait;
  end process;
end arch12;
```

Time Simulation Differences;

With new adder; **Lag is 5.66 ns**



With previous adder; **Lag is 9.06ns**



Utizilation Differences;

With new adder;

Name	Slice LUTs (53200)	Slice (13300)	LUT as Logic (53200)	Bonded IPADs (2)
adder8b	8	3	8	25

With old adder;

Name	Slice LUTs (53200)	Slice (13300)	LUT as Logic (53200)	Bonded IPADs (2)
adderother	8	3	8	26

Problem 2

Design and simulate a **32-bit barrel shifter**. Shift the 32-bit input by the selected amount with the `sel` input, and selected orientation with the `ori` input.

Design Code:

```
library ieee;

use ieee.std_logic_1164.all;

entity barrels is

port (clk,reset,ori :in std_logic;

sel:in std_logic_vector (31 downto 0);

output:out std_logic_vector (31 downto 0));

end barrels;

architecture Barrel of barrels is

begin

process (reset,clk,sel)

variable d: std_logic_vector (31 downto 0);

begin

if (reset = '0') then

d:=(others => '0');

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

if (ori ='1') then

d:=sel;

else

d:= d(0) & d(31 downto 1) ;

end if;

end if;

output<= d;
```

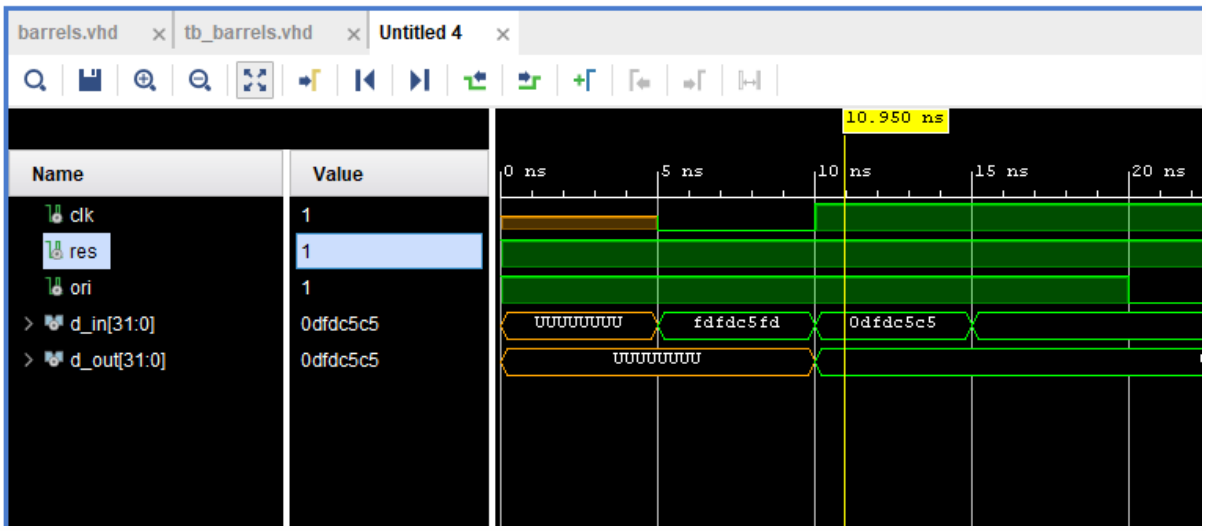
```
end process;

end barrel;
```

Testbench Code:

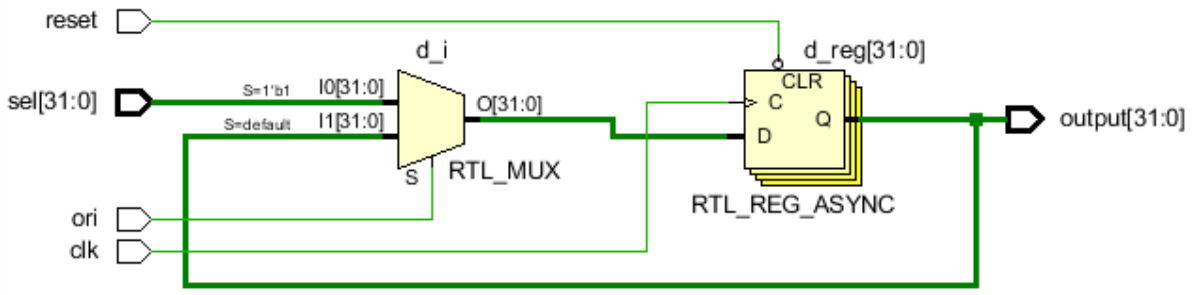
```
library ieee;

use ieee.std_logic_1164.all;

entity tb_Barrels is

end tb_Barrels;

architecture tb_Barrel of tb_Barrels is

component Barrels

port (clk,reset,ori :in std_logic;

sel:in std_logic_vector (31 downto 0);

output:out std_logic_vector (31 downto 0));

end component;

signal clk,res,ori : std_logic;

signal d_in,d_out : std_logic_vector (31 downto 0);

begin

br: Barrels port map (clk,res,ori,d_in,d_out);

process

begin

res <= '1'; ori <= '1';

wait for 5 ns;


ori <= '1';

clk<='0';

d_in<="11111101111111011100010111111101";

wait for 5 ns;
```

```
clk<='1';

d_in<="00001101111111011100010111000101";

wait for 5 ns;

res <= '1';

d_in <= "00000101110001011100010111000101";

ori <= '1';

wait for 5 ns;

ori <= '0';

wait;

end process;


end tb_Barrel;
```

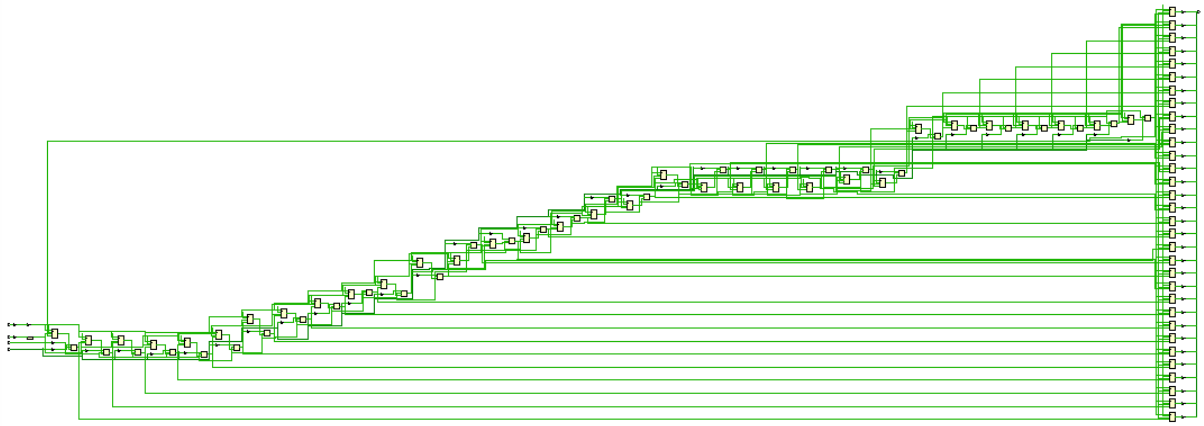
Simulation:



Elaborated Schematic:



Implementation Schematic:



Utizilation Report:

Name	Slice LUTs (53200)	Slice Registers (106400)	Slice (13300)	LUT as Logic (53200)	Block RAM Tile (140)	Bonded IPADs (2)	BUFIO (16)
N barrels	17	64	13	17	64	67	1

Problem 3

Design and simulate a **parity generator** for 8-bit values. Check out the timing and figure out how long it takes. Auto test your results with a *smarter* simulation.

Code:

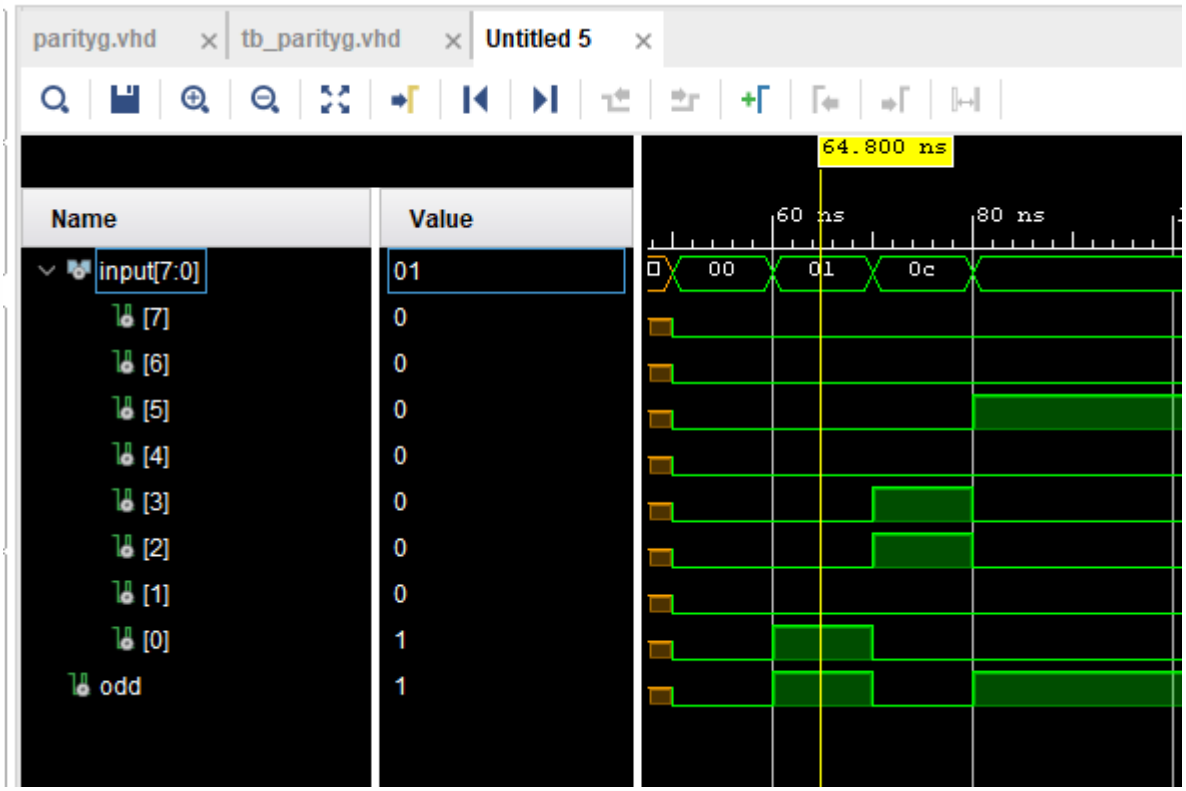
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity parityg is

Port (

input: in std_logic_vector(7 downto 0);

odd : out std_logic);

end parityg;


architecture Behavioral of parityg is

signal s : std_logic_vector(6 downto 0);

begin

s(0) <= input(0) xor input(1);

s(1) <= input(2) xor input(3);

s(2) <= input(4) xor input(5);

s(3) <= input(6) xor input(7);

s(4) <= s(0) xor s(1);

s(5) <= s(2) xor s(3);

odd <= s(4) xor s(5);


end Behavioral;
```

Testbench Code:

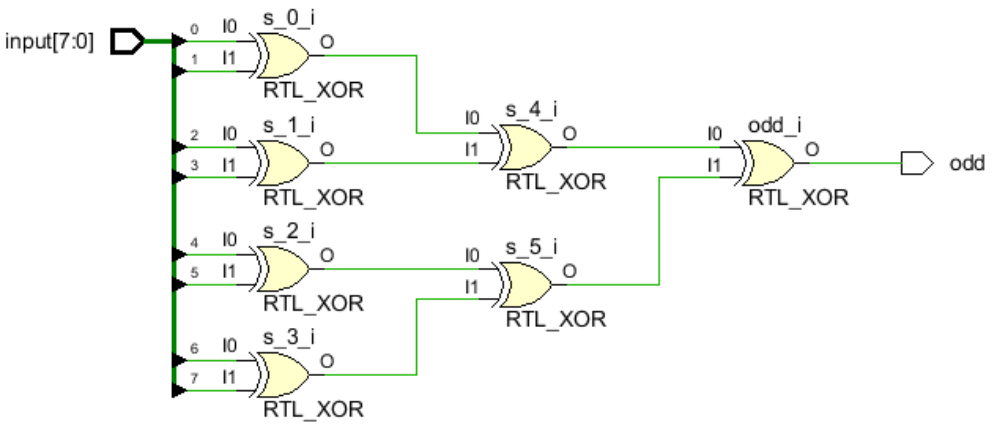
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity tb_parityg is
end tb_parityg;
architecture Behavioral of tb_parityg is
component parityg is
Port (
input: in std_logic_vector(7 downto 0);
odd: out std_logic );
end component;
signal input : std_logic_vector(7 downto 0);
signal odd : std_logic;
begin
s0 : parityg port map(
input => input, odd => odd
);
process
begin
wait for 50 ns;
input <= "00000000";
wait for 10 ns;
input <= "00000001";
wait for 10 ns;
input <= "00001100";
```

```
wait for 10 ns;

input <= "00100000";

wait;

end process;

end Behavioral;
```

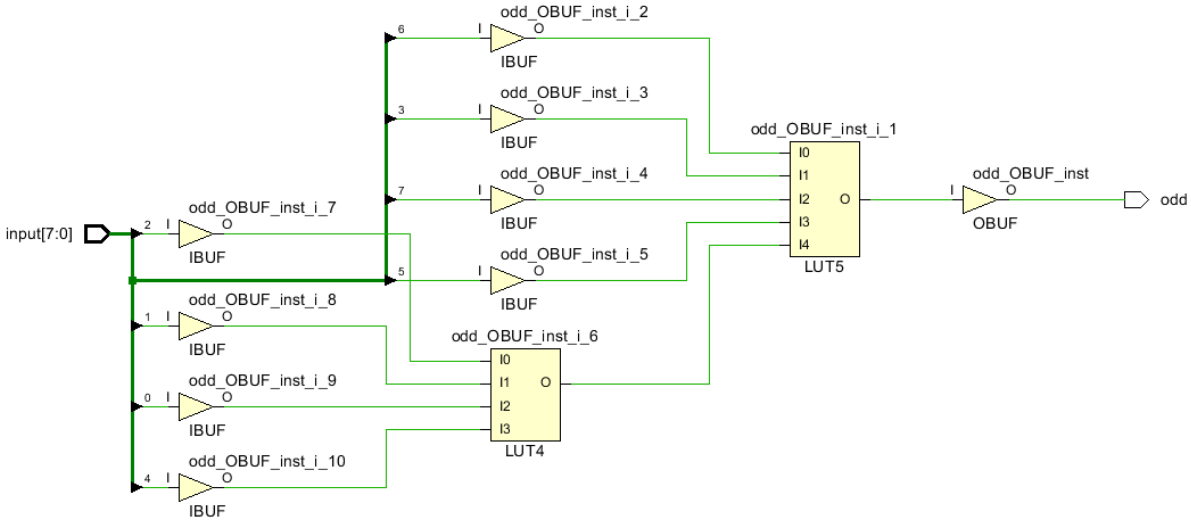
Simulation:



Eloborated Design:



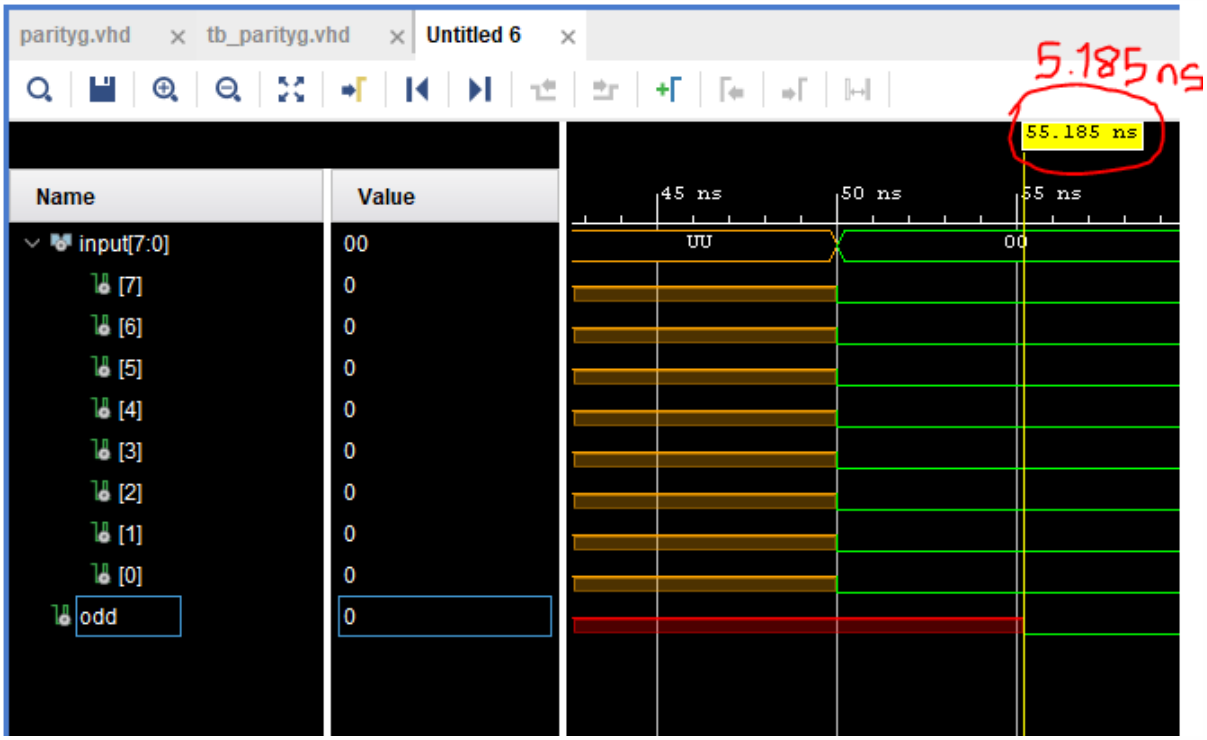
Implementation Design:



Utizilation Report:

Name	Slice LUTs (53200)	Slice (13300)	LUT as Logic (53200)	Bonded IPADs (2)
N parityg	2	2	2	9

Timing Simulation:



Problem 4

Design and simulate an 32-to-5 **priority encoder**. You can design and utilize smaller encoders (structural design). Auto test your results with a *smarter* simulation.

Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity priorty32t5 is
 Port (
 e : in std_logic;
 input : in std_logic_vector(31 downto 0);
 output : out std_logic_vector(4 downto 0)
 );
end priorty32t5;

architecture Behavioral of priorty32t5 is

begin
process (e, input) begin
          output <= "00000";
          if (e = '1') then
             if (input(31 downto 27) = "00001") then
                  output <= "00001";
                  elsif  (input(31 downto 27) = "00010") then
                  output <= "00010";
                  elsif  (input(31 downto 27) = "00011") then
                  output <= "00011";
                  elsif  (input(31 downto 27) = "00100") then
                  output <= "00100";
                  elsif  (input(31 downto 27) = "00101") then
                  output <= "00101";
                  elsif  (input(31 downto 27) = "10010") then
```

```
                    output <= "10010";
                    elsif  (input(31 downto 27) = "10100") then
                    output <= "10100";
                    elsif  (input(31 downto 27) = "11000") then
                    output <= "11000";
                    elsif  (input(31 downto 27) = "11001") then
                    output <= "11001";
                    elsif  (input(31 downto 27) = "11010") then
                    output <= "11010";
                    elsif  (input(31 downto 27) = "11100") then
                    output <= "11100";
                    elsif  (input(31 downto 27) = "11110") then
                    output <= "11110";
                    elsif  (input(31 downto 27) = "11111") then
                    output <= "11111";
                else
                    output <= "00000";
                end if;
            end if;
        end process;
end Behavioral;
```

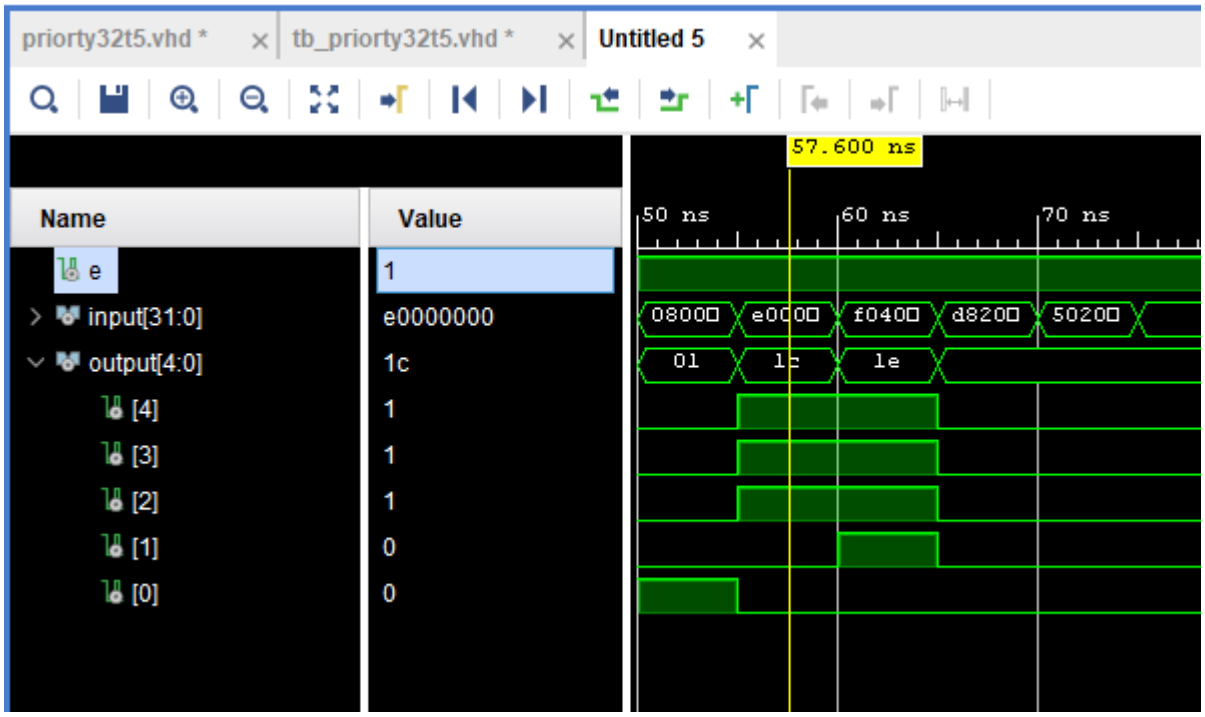
Testbench:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity tb_priorty32t5 is
end tb_priorty32t5;

architecture Behavioral of tb_priorty32t5 is
component priorty32t5 is
Port (
 e : in std_logic;
```

```
 input : in std_logic_vector(31 downto 0);
 output : out std_logic_vector(4 downto 0)
 );
end component;
signal e : std_logic;
signal input : std_logic_vector(31 downto 0);
signal output : std_logic_vector(4 downto 0);
begin
s0 : priorty32t5 port map(
e => e, input => input, output => output
);
process
begin
wait for 50 ns;
e <= '1';
input <= "00001000000000000000000000000000";
wait for 5 ns;
e <= '1';
input <= "11100000000000000000000000000000";
wait for 5 ns;
e <= '1';
input <= "11110000010000000000000000000000";
wait for 5 ns;
e <= '1';
input <= "11011000001000000000000000000000";
wait for 5 ns;
e <= '1';
input <= "01010000001000000000000000000000";
wait for 5 ns;
e <= '1';
input <= "00000000001000000000000000000000";
wait for 5 ns;
```

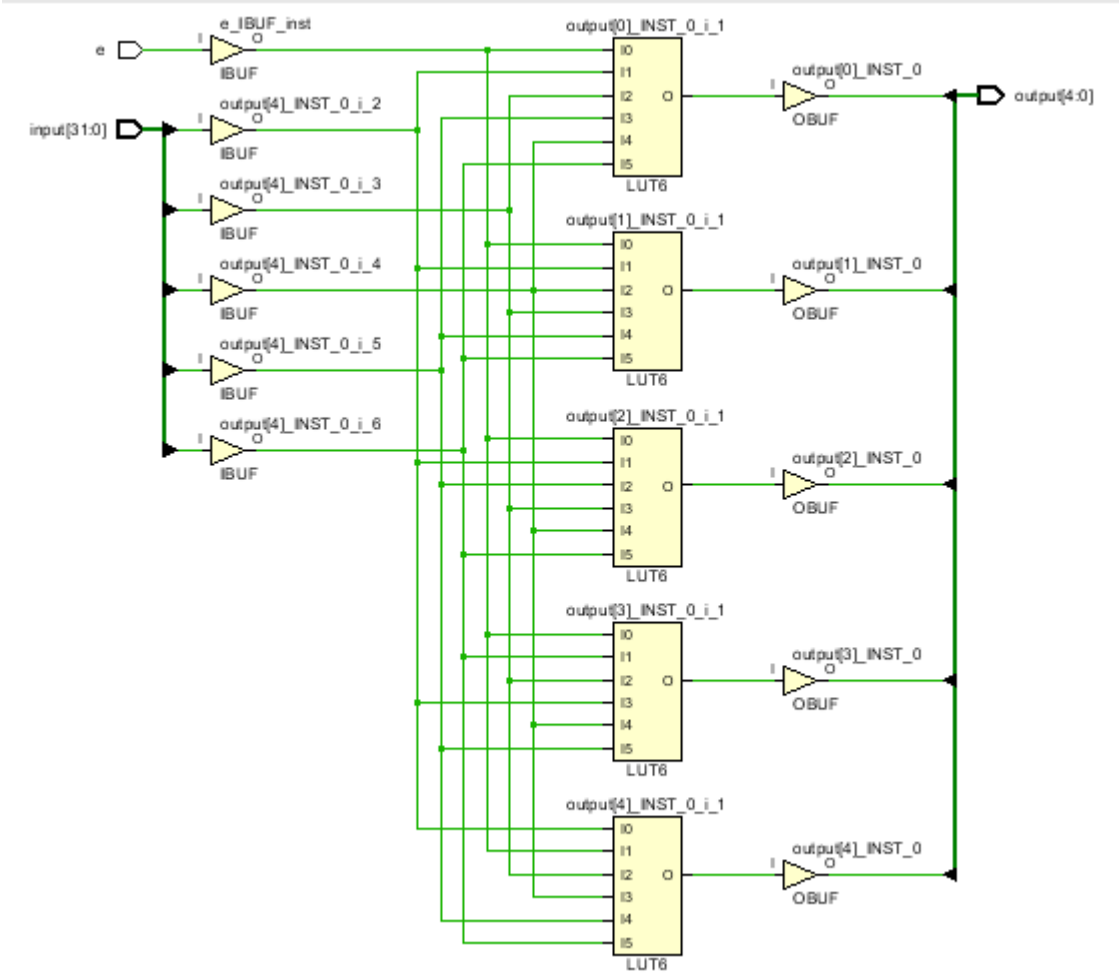
```
end process;

end Behavioral;
```

Simulation:



Elaborated Schematic:

Implementation Schematic:



Utizilation Report:

Name	Slice LUTs (53200)	Slice (13300)	LUT as Logic (53200)	Bonded IPADs (2)
priorty32t5	5	2	5	11

Previous Homework

Problem 5

Design and simulate an **RGB to YUV converter**. Your circuit should have a 24-bit input consisting of 8-bit red, 8-bit green and 8-bit blue values in that order. You should have 24-bit YUV output. You can use the following formulation;

For simulation, just pick 16 random colors with 5 of them being black, white, just blue, just red and just green.

For this problem,firstly i understood that i had to do with fixed_pkg library but i couldn't use this library. When i tried to add this library i got some errrors and i couldn't setup it. After that without fixed_pkg library, my method for float number was first multiple those numbers(0.299, etc.) with 2^17 and make them as integer, after calculation with equation, divide every result with 2^17. I wrote some code but i couldn't divide them.

My design code:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use ieee.numeric_std.all;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity rgbtoyuv is

Port (

rxy : in integer:=39191; --0.299 * 2^17 etc.

gyy : in integer:=76939;

bzy : in integer:=14942;

rxu : in integer:=19268;

gyu : in integer:=37880;

bzu : in integer:=57148;
```

```
rxv : in integer:=80610;

gyv : in integer:=67502;

bzv : in integer:=13107;

rgb : in unsigned(23 downto 0):="000010001100011010011001";--example

y,u,v : out std_logic_vector(7 downto 0)

 );

end rgbtoyuv;

architecture Behavioral of rgbtoyuv is

begin

process

variable intr,intg,intb, inty,intu,intv : integer;

variable cons : std_logic_vector(23 downto
0):="000000100000000000000000"; --it is 2^17 in binary

begin

intr := to_integer(rgb(23 downto 16));

intg := to_integer(rgb(15 downto 8));

intb := to_integer(rgb(7 downto 0));

inty := ((rxy * intr) + (gyy * intg) + (bzy * intb)) ;

intu := ((-rxu * intr) + (-gyu * intg) + (bzu * intb));

intv := ((rxv * intr) + (-gyv * intg) + (-bzv * intb));

y <= std_logic_vector(to_signed(inty,8));

u <= std_logic_vector(to_signed(intu,8));

v <= std_logic_vector(to_signed(intv,8));

end process;

end Behavioral;
```

Problem 6

Design and simulate a **hamming distance** finder. Your circuit should have a generic for input bit with, two n-bit inputs to calculate the hamming distance and an 8-bit output to send the result. Find a suitable algorithm. Test 16 random selected pairs for the simulation.

Design Code:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use ieee.numeric_std.all;

entity hamming is

generic ( N : integer := 32);

port(

i_a, i_b : in std_logic_vector(N-1 downto 0);

o_dist : out std_logic_vector(7 downto 0));

end hamming;

architecture Behavioral of hamming is

signal difference : std_logic_vector(N-1 downto 0) := (others => '0');

function sum ( s : std_logic_vector ) return natural is

    variable sum : integer range 0 to s'length := 0;

begin

for x in s'range loop

if s(x) = '1' then

sum := sum+1;

else

sum := sum;

end if;

end loop;

return sum;

end function;

begin
```

```
difference <= i_a xor i_b ;
o_dist <= std_logic_vector(to_unsigned(sum(difference),8));
end Behavioral;
```

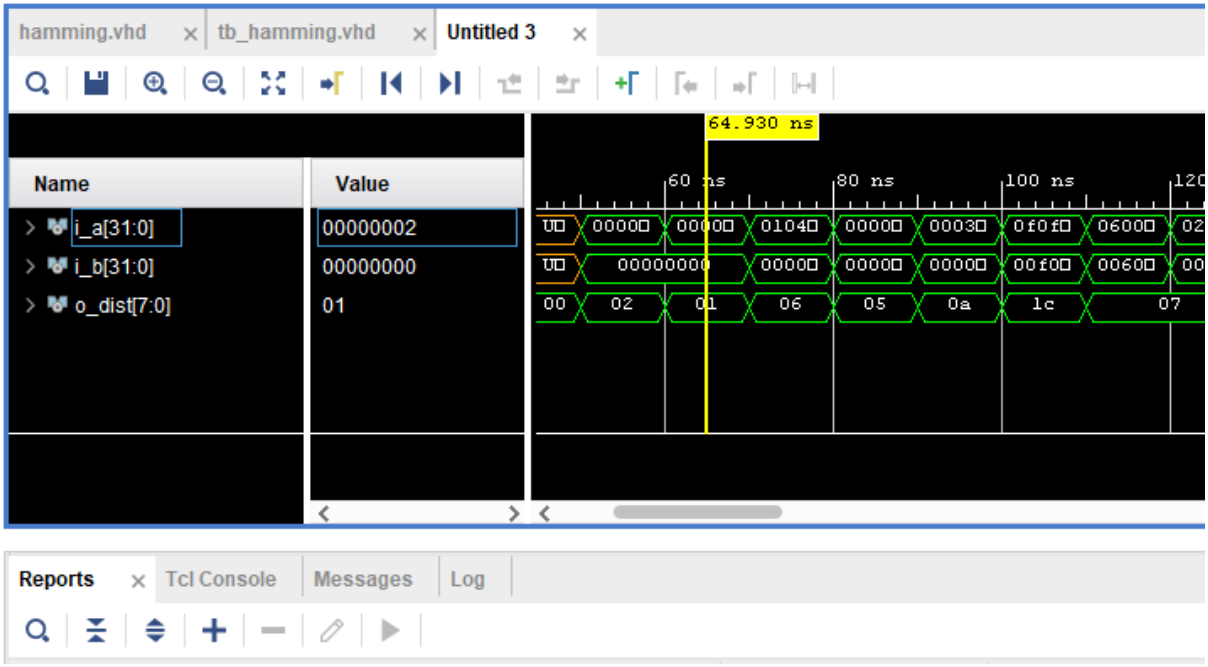
Testbench Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity tb_hamming is
end tb_hamming;
architecture Behavioral of tb_hamming is
component hamming is
generic ( N : integer := 32);
port(
i_a, i_b : in std_logic_vector(N-1 downto 0);
o_dist : out std_logic_vector(7 downto 0));
end component;
signal i_a,i_b : std_logic_vector(31 downto 0);
signal o_dist  : std_logic_vector(7 downto 0);

begin
h0: hamming port map(
i_a => i_a, i_b=>i_b, o_dist=>o_dist);
process
begin
wait for 50 ns;
i_a <= "00000000000000000000000000000110";
i_b <= "00000000000000000000000000000000";
wait for 10 ns;
i_a <= "00000000000000000000000000000010";
i_b <= "00000000000000000000000000000000";
wait for 10 ns;
i_a <= "00000001000001000000000000000110";
i_b <= "00000000000000000100000100000000";
wait for 10 ns;
```

```
i_a <= "00000000000000000100000000000110";
i_b <= "00000000000000000001000010000000";
wait for 10 ns;
i_a <= "00000000000000111100000000000111";
i_b <= "00000000000000000000000000111000";
wait for 10 ns;
i_a <= "00001111000011110000111100001111";
i_b <= "00000000111100001111000011110000";
wait for 10 ns;
i_a <= "00000110000000000000000000000111";
i_b <= "00000000011000000000000000000000";
wait for 10 ns;
i_a <= "00000010100100000000000000000111";
i_b <= "00000000011100000000000000000000";
wait for 10 ns;
i_a <= "11000000000000000000000000000111";
i_b <= "11001010000000000000000000000000";
wait for 10 ns;
i_a <= "00000000000000000000000000000111";
i_b <= "10010010010010010010010010100000";
wait for 10 ns;
i_a <= "00000001010101010101101010100111";
i_b <= "00000000000000000000000000000000";
wait for 10 ns;
i_a <= "10000000000000000000000000000111";
i_b <= "01100000000000000000000000000000";
wait for 10 ns;
i_a <= "00001001000000000000000000000111";
i_b <= "00000000000001110000000000000000";
wait for 10 ns;
i_a <= "00000000000011100000000000000111";
i_b <= "00000011100000000000000000000000";
wait for 10 ns;
i_a <= "11111111111111111111111111111111";
```

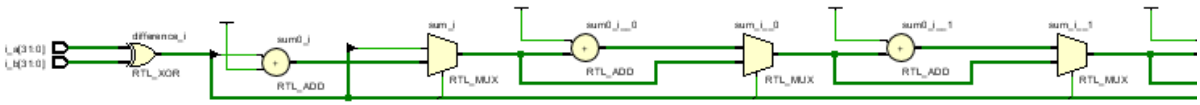
```
i_b <= "11111111111111111111111111111111";
wait for 10 ns;
i_a <= "11111111111111111111111111111111";
i_b <= "00000000000000000000000000000000";
wait for 10 ns;
wait;
end process;
end Behavioral;
```

Simulation:

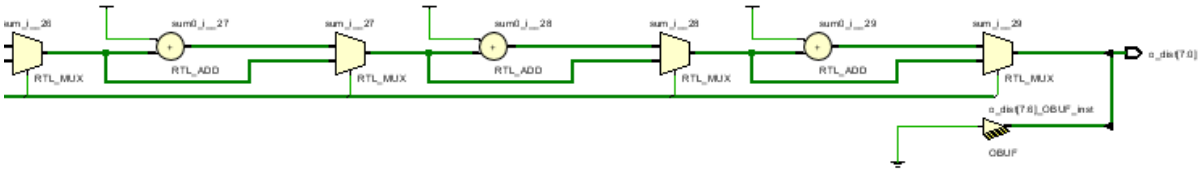


Elaborated Schematic:

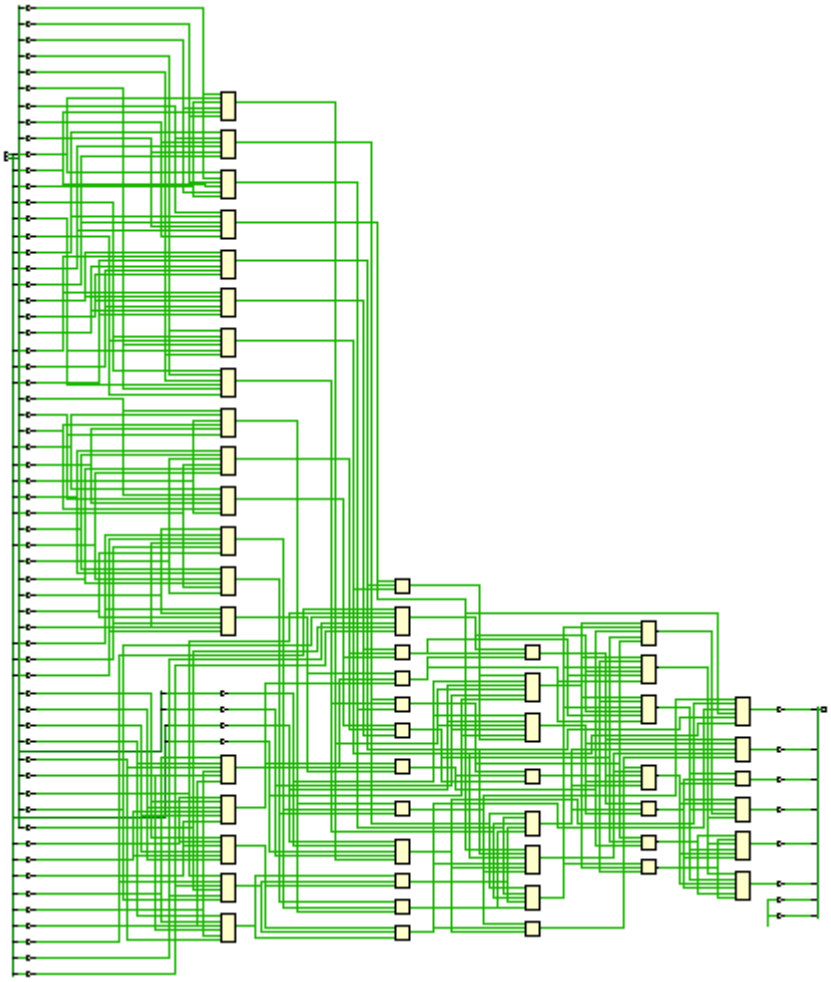
Beginning



End



Implementation Schematic:



Utizilation Report:

Name	Slice LUTs (53200)	Slice (13300)	LUT as Logic (53200)	Bonded IPADs (2)
N hamming	44	12	44	72