Assignment 2

IT451 : COA LAB

ANINDYA KUNDU
IT, 4th Semester
ID: 510817020 (Hx-19)

11-03-2019

Question 1

Design and simulate the behavioral model of an UP/DOWN counter that is capable of counting up to 10. Use one input count_mode to control the mode of counting i.e. for count_mode = '1' the counter will operate as an UP counter and for count_mode = '0' the counter will behave as a DOWN counter. Use a reset input to reset the counter anytime to 0.

a. Simulate and record the waveform with all possible stimulus waveforms.
b. Synthesize the above circuit for Spartan 3 AN Evaluation Board. Observe and report the hardware utilization with the detailed schematics (both RTL and Technical).
c. Synthesize the same design for Spartan 6 and compare the hardware utilization summary.

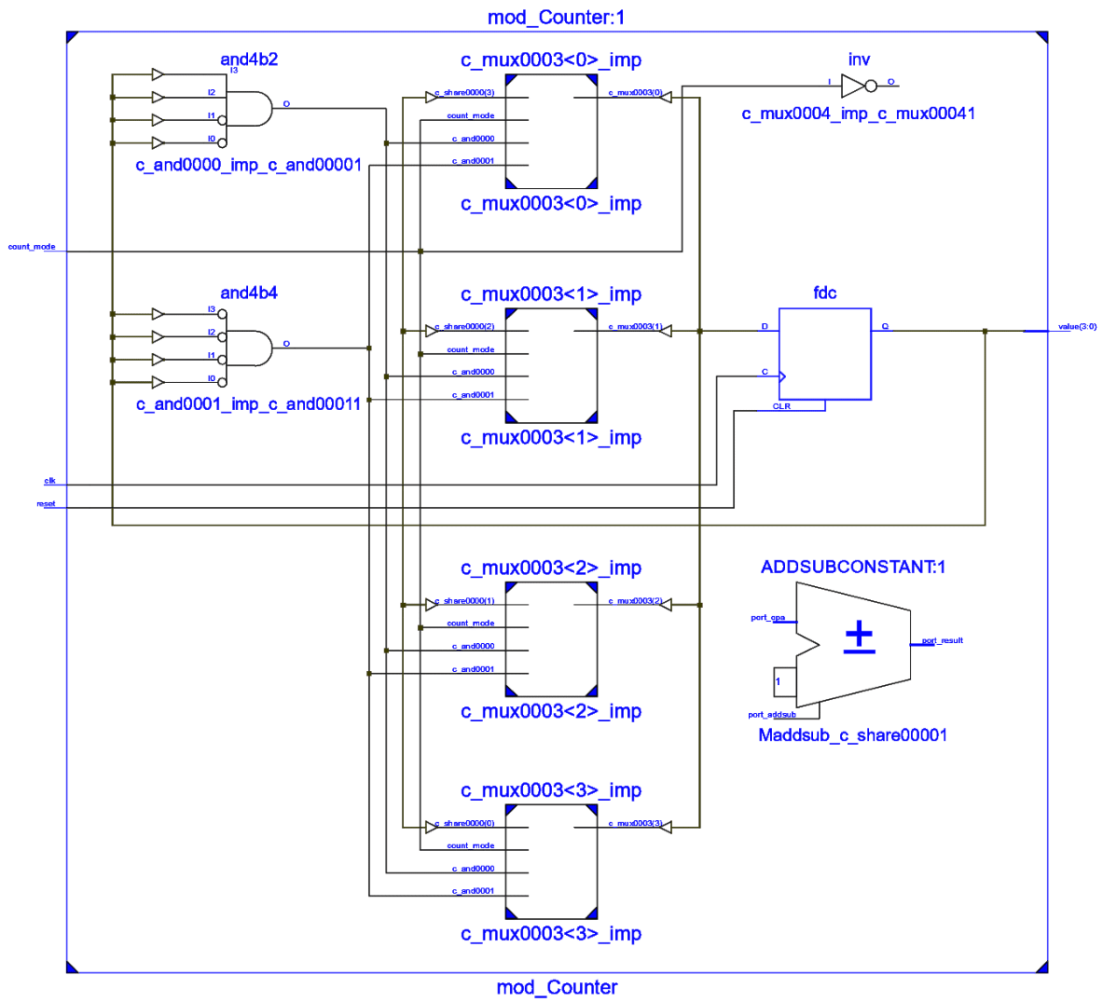
VHDL Module: *mod_Counter.vhd*

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

entity mod_Counter is
    Port ( clk           : in  STD_LOGIC;
           reset         : in  STD_LOGIC;
           count_mode    : in  STD_LOGIC;
           value         : out STD_LOGIC_VECTOR (3 downto 0));
end mod_Counter;

architecture Behavioral of mod_Counter is
      signal c : STD_LOGIC_VECTOR (3 downto 0);
begin

process(clk, count_mode)
begin
      if (reset = '1') then c <= "0000";
      elsif rising_edge(clk) then
            if(count_mode = '1') then
                  if    (c /= "1001") then
                        c <= std_logic_vector(unsigned(c) + 1);
                  else  c <= std_logic_vector(unsigned(c) + 7);
                  end if;
            else
                  if    (c /= "0000") then
                        c <= std_logic_vector(unsigned(c) - 1);
                  else  c <= std_logic_vector(unsigned(c) + 9);
                  end if;
            end if;
      end if;
```

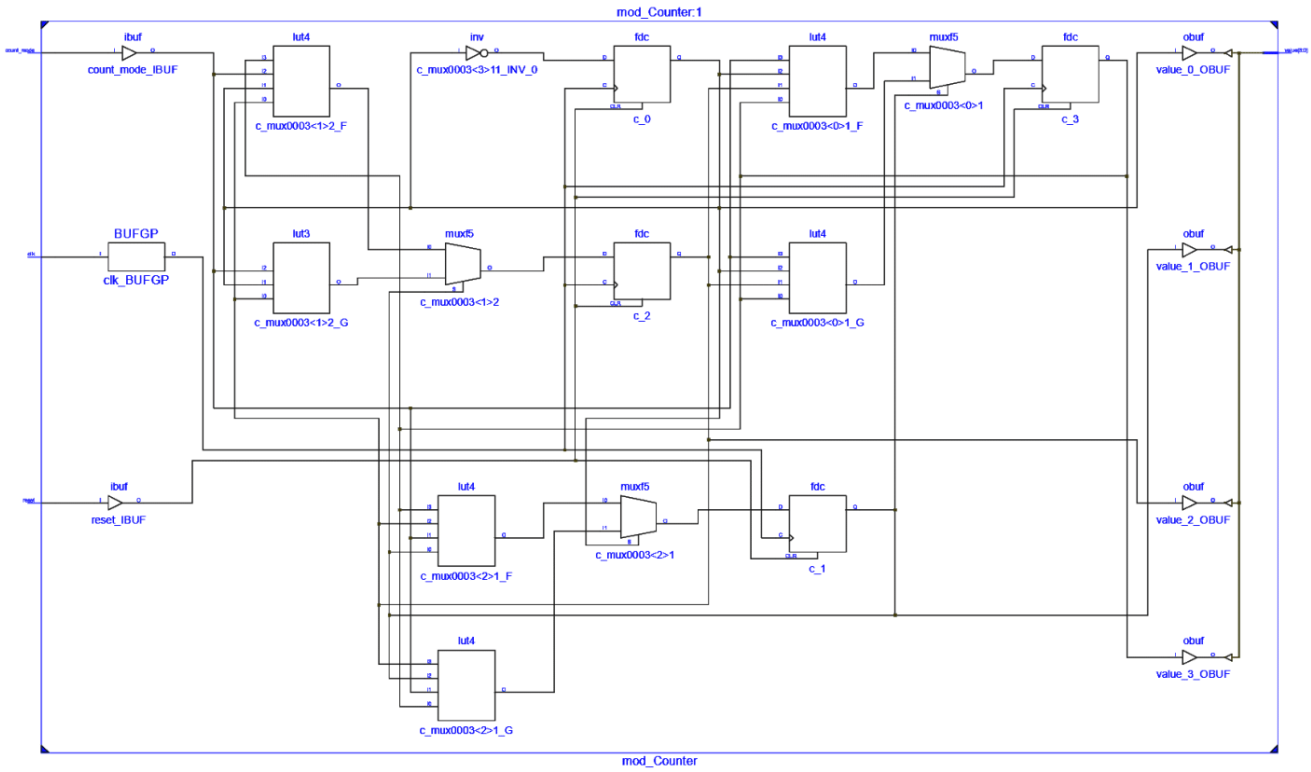
```
end process;
      value <= c;

end Behavioral;
```

SPARTAN 3AN

RTL Schematic:



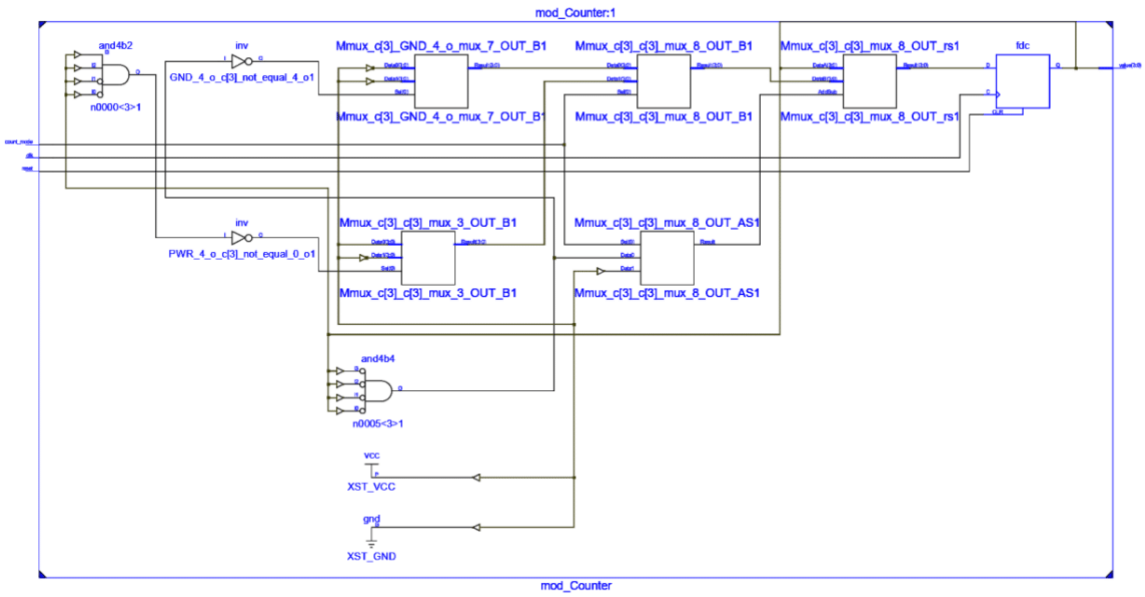
Technology Schematic:



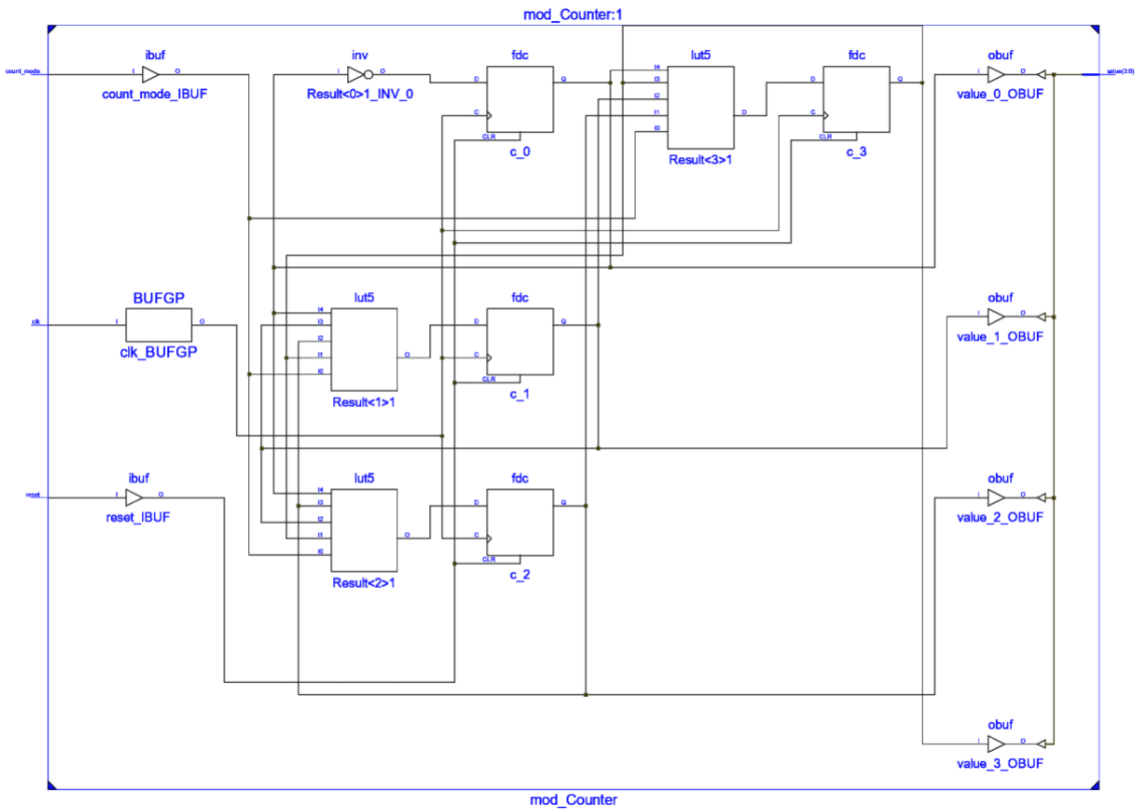
Device Utilisation Summary:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	4	5888	0%
Number of Slice Flip Flops	4	11776	0%
Number of 4 input LUTs	7	11776	0%
Number of bonded IOBs	7	372	1%
Number of GCLKs	1	24	4%

RTL Schematic:



Technology Schematic:



Device Utilisation Summary:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	4	54576	0%
Number of Slice LUTs	4	27288	0%
Number of fully used LUT-FF pairs	0	8	0%
Number of bonded IOBs	7	296	2%
Number of BUFG/BUFGCTRLs	1	16	6%

VHDL Test Bench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_Counter IS
END tb_Counter;

ARCHITECTURE behavior OF tb_Counter IS
   COMPONENT mod_Counter
   PORT(
        clk             : IN  std_logic;
        reset           : IN  std_logic;
        count_mode: IN  std_logic;
        value           : OUT std_logic_vector(3 downto 0)
       );
   END COMPONENT;

   signal clk                : std_logic := '0';
   signal reset         : std_logic := '0';
   signal mode               : std_logic := '0';
   signal value         : std_logic_vector(3 downto 0);

   constant clk_period : time := 75 ns;

BEGIN
   uut: mod_Counter PORT MAP (
          clk                => clk,
          reset         => reset,
          count_mode    => mode,
          value         => value
        );

   clk_process :process
```

```
    begin
        clk <= '1';
        wait for clk_period/2;
        clk <= '0';
        wait for clk_period/2;
    end process;

    stim_proc: process
    begin
        reset <= '1', '0' after 75ns;
        mode  <= '1', '0' after 900ns;
        wait;
    end process;
END;
```

Simulation:

Question 2

Design and simulate the behavioral model of a 4 bit shift register that can support the following modes of data transfer.

a. Serial in parallel out (SIPO)
b. Serial in serial out (SISO)
c. Parallel in serial out (PISO)
d. Parallel in parallel out (PIPO).

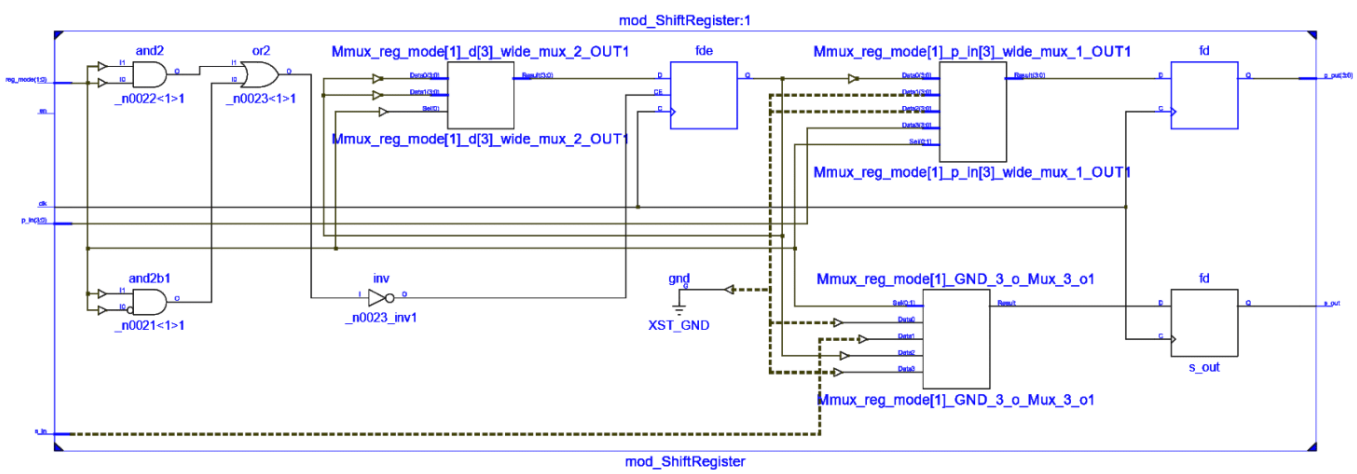
Use case statement for this design. Use one input reg_mode [std_logic_vector (1 downto 0)] to control the nature of behavior of the register. Use "00" for SIPO, "01" for SISO and so on.

VHDL Module: *mod_ShiftRegister.vhd*

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

entity mod_ShiftRegister is
    Port ( clk          : in  STD_LOGIC;
           reg_mode     : in  STD_LOGIC_VECTOR (1 downto 0);
            en          : in  STD_LOGIC;
           s_in         : in  STD_LOGIC;
           p_in         : in  STD_LOGIC_VECTOR (3 downto 0);
           s_out        : out STD_LOGIC;
           p_out        : out STD_LOGIC_VECTOR (3 downto 0));
end mod_ShiftRegister;

architecture Behavioral of mod_ShiftRegister is
      signal d : STD_LOGIC_VECTOR (3 downto 0) := "0000";

begin

process(clk)
begin
      if rising_edge(clk) then
            case reg_mode is
                  when "00" =>
                        p_out(0) <= d(1); p_out(1) <= d(2);
                        p_out(2) <= d(3); p_out(3) <= s_in;
                        d(0) <= d(1); d(1) <= d(2);
                        d(2) <= d(3); d(3) <= s_in;
                        s_out <= '0';
                  when "01" =>
                        s_out <= s_in;
                        p_out <= "0000";
```

```
                when "10" =>
                      if en = '1' then d <= p_in;
                      end if;
                      s_out <= d(0);
                      d(0) <= d(1); d(1) <= d(2);
                      d(2) <= d(3); d(3) <= '0';
                      p_out <= "0000";
                when "11" =>
                      p_out <= p_in;
                      s_out <= '0';
                when others =>
          end case;
      end if;
end process;

end Behavioral;
```

RTL Schematic:

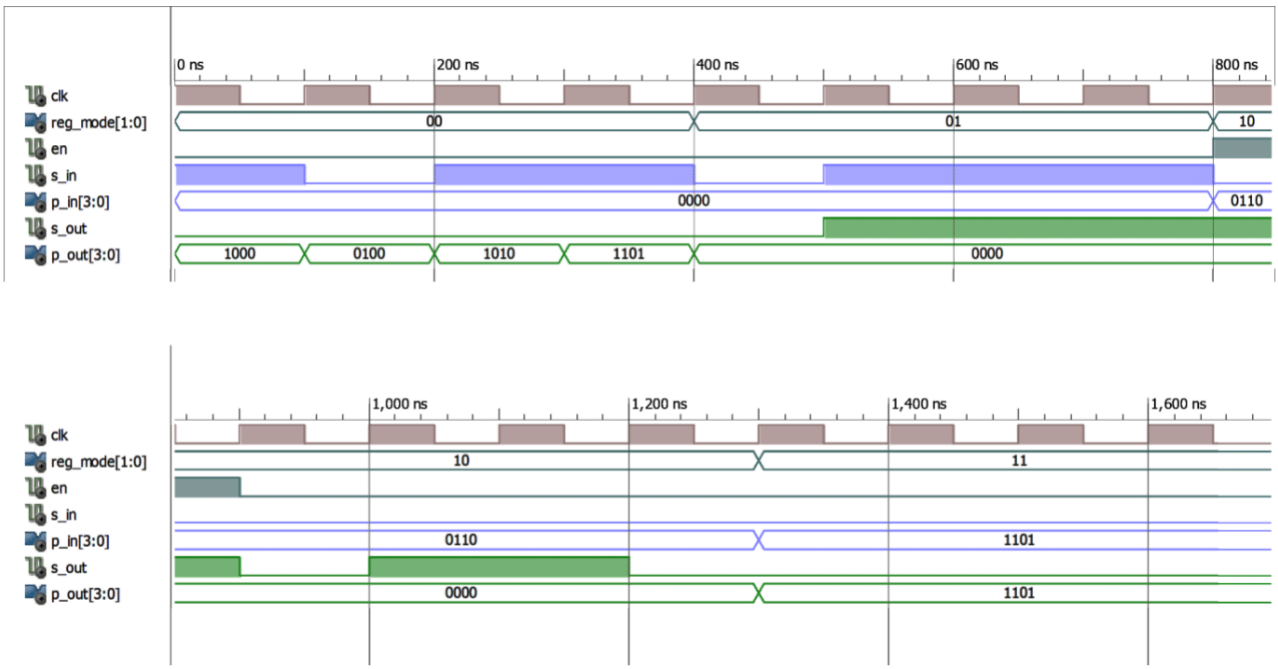


VHDL Test Bench: *tb_ShiftRegister.vhd*

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_ShiftRegister IS
END tb_ShiftRegister;

ARCHITECTURE behavior OF tb_ShiftRegister IS
    COMPONENT mod_ShiftRegister
    PORT(
         clk            : IN  std_logic;
         reg_mode       : IN  std_logic_vector(1 downto 0);
         en             : IN  std_logic;
         s_in           : IN  std_logic;
         p_in           : IN  std_logic_vector(3 downto 0);
         s_out          : OUT std_logic;
         p_out          : OUT std_logic_vector(3 downto 0)
        );
    END COMPONENT;

   signal clk            : std_logic := '0';
   signal reg_mode       : std_logic_vector(1 downto 0) := (others => '0');
   signal en             :    std_logic := '0';
   signal s_in           : std_logic := '0';
   signal p_in           : std_logic_vector(3 downto 0) := (others => '0');

   signal s_out    : std_logic;
   signal p_out    : std_logic_vector(3 downto 0);

   constant clk_period : time := 100 ns;

BEGIN
   uut: mod_ShiftRegister PORT MAP (
          clk           => clk,
          reg_mode      => reg_mode,
          en            => en,
          s_in          => s_in,
          p_in          => p_in,
          s_out         => s_out,
          p_out         => p_out
        );

   clk_process :process
   begin
```

```
      clk <= '1';
      wait for clk_period/2;
      clk <= '0';
      wait for clk_period/2;
   end process;

stim_proc: process
   begin
      reg_mode <= "00", "01" after 400ns, "10" after 800ns, "11" after 1300ns;
      s_in <= '1', '0' after 100ns, '1' after 200ns, '0' after 400ns, '1' after
      500ns, '0' after 800ns;
      en   <= '0', '1' after 800ns, '0' after 900ns;
      p_in <= "0000", "0110" after 800ns, "1101" after 1300ns;
      wait;
   end process;
END;
```

Simulation:



Question 3

Design the followings:

a. Design and simulate the behavioral model of an 8 bit even parity generator.
b. Design and simulate the behavioral model an 8 bit even parity checker.
c. Design a top level module called parity and use the previously designed two modules as components to check whether the functionality of the two modules are in synchronization or not.

VHDL Module:

mod_genParity.vhd

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

entity mod_genParity is
    Port ( d : in  STD_LOGIC_VECTOR (7 downto 0);
           p : out STD_LOGIC);
end mod_genParity;

architecture Behavioral of mod_genParity is

begin
      p <= d(0) xor d(1) xor d(2) xor d(3) xor d(4) xor d(5) xor d(6) xor d(7);
end Behavioral;
```

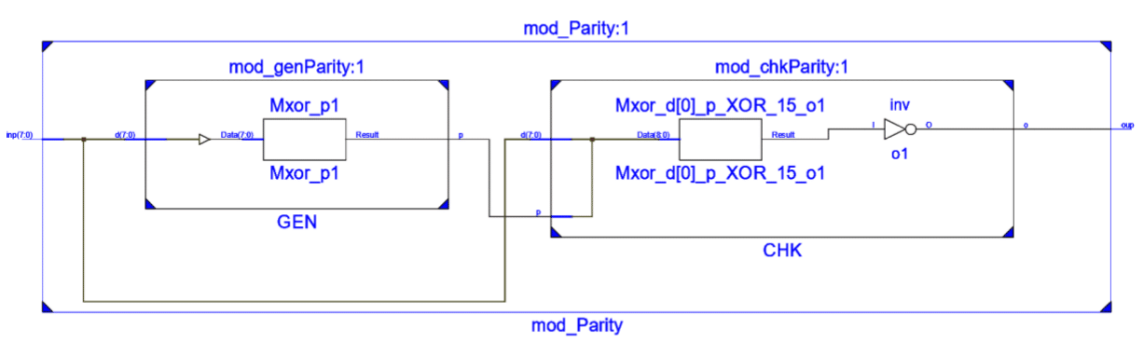
mod_chkParity.vhd

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

entity mod_chkParity is
    Port ( d : in  STD_LOGIC_VECTOR (7 downto 0);
                   p : in  STD_LOGIC;
           o : out STD_LOGIC);
end mod_chkParity;

architecture Behavioral of mod_chkParity is

begin
      o <= not (d(0) xor d(1) xor d(2) xor d(3) xor d(4) xor d(5) xor d(6) xor d(7)
      xor p);
end Behavioral;
```

mod_Parity.vhd

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

entity mod_Parity is
    Port ( inp : in  STD_LOGIC_VECTOR (7 downto 0);
           oup : out STD_LOGIC);
end mod_Parity;

architecture Behavioral of mod_Parity is
      Component mod_genParity is
      Port ( d : in  STD_LOGIC_VECTOR (7 downto 0);
             p : out STD_LOGIC);
      end Component;

      Component mod_chkParity is
      Port ( d : in  STD_LOGIC_VECTOR (7 downto 0);
             p : in  STD_LOGIC;
             o : out STD_LOGIC);
      end Component;
      signal par : STD_LOGIC := '0';

begin
      GEN : mod_genParity port map (inp, par);
      CHK : mod_chkParity port map (inp, par, oup);
end Behavioral;
```

RTL Schematic:

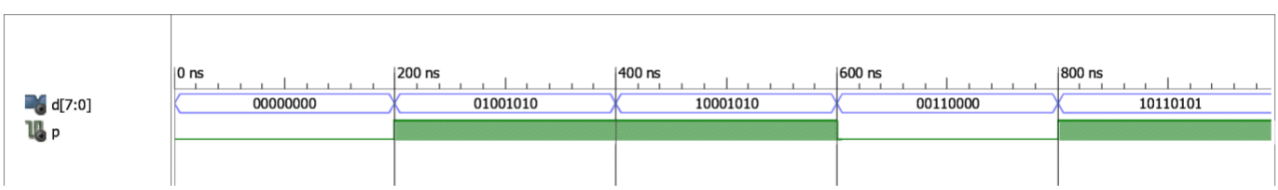


VHDL Test Bench:

tb_genParity.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_genParity IS
END tb_genParity;

ARCHITECTURE behavior OF tb_genParity IS
    COMPONENT mod_genParity
    PORT(
         d : IN  std_logic_vector(7 downto 0);
         p : OUT  std_logic
        );
    END COMPONENT;
   signal d : std_logic_vector(7 downto 0) := (others => '0');
   signal p : std_logic;

BEGIN
   uut: mod_genParity PORT MAP (
          d => d,
          p => p
        );
   stim_proc: process
   begin
            d <= "00000000";
            wait for 200ns;
            d <= "01001010";
            wait for 200ns;
            d <= "10001010";
            wait for 200ns;
            d <= "00110000";
            wait for 200ns;
            d <= "10110101";
      wait;
   end process;
END;
```

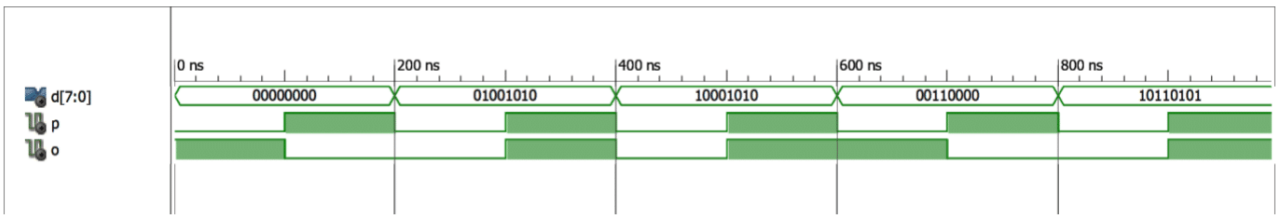
Simulation:



tb_chkParity.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_chkParity IS
END tb_chkParity;

ARCHITECTURE behavior OF tb_chkParity IS
    COMPONENT mod_chkParity
    PORT(
         d : IN  std_logic_vector(7 downto 0);
         p : IN  std_logic;
         o : OUT  std_logic
        );
    END COMPONENT;
   signal d : std_logic_vector(7 downto 0) := (others => '0');
   signal p : std_logic := '0';
   signal o : std_logic;

BEGIN
   uut: mod_chkParity PORT MAP (
          d => d,
          p => p,
          o => o
        );
   stim_proc: process
   begin
      d <= "00000000"; p <= '0';
      wait for 100ns;  p <= '1';
      wait for 100ns;
      d <= "01001010"; p <= '0';
      wait for 100ns;  p <= '1';
      wait for 100ns;
      d <= "10001010"; p <= '0';
      wait for 100ns;  p <= '1';
      wait for 100ns;
      d <= "00110000"; p <= '0';
      wait for 100ns;  p <= '1';
      wait for 100ns;
      d <= "10110101"; p <= '0';
      wait for 100ns;  p <= '1';
      wait;
   end process;
END;
```

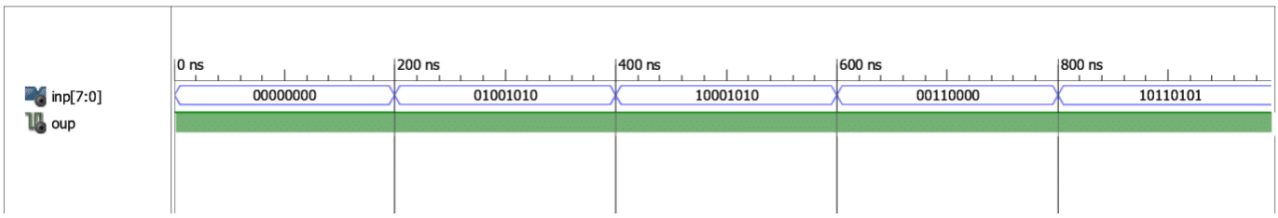
Simulation:



tb_Parity.vhd

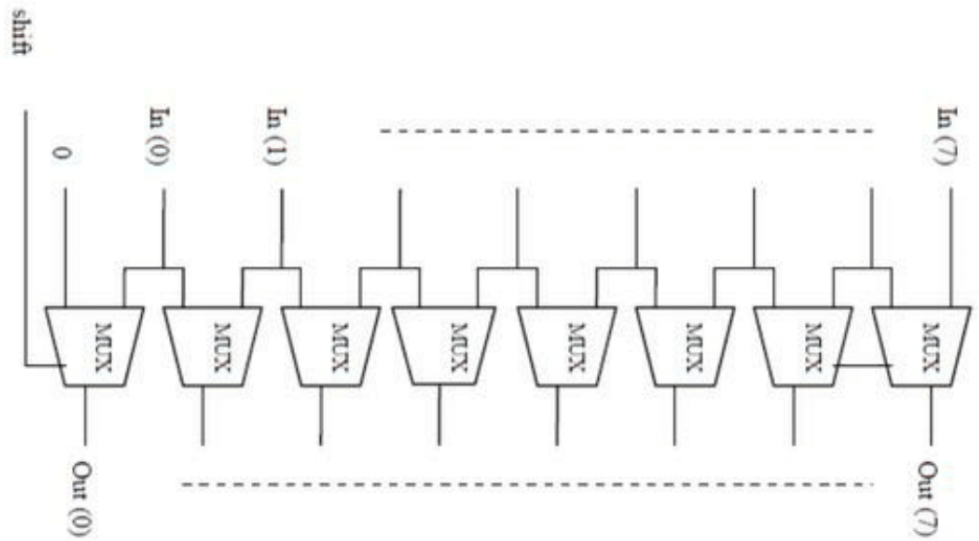
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_Parity IS
END tb_Parity;

ARCHITECTURE behavior OF tb_Parity IS
   COMPONENT mod_Parity
   PORT(
        inp : IN  std_logic_vector(7 downto 0);
        oup : OUT std_logic
       );
   END COMPONENT;
   signal inp : std_logic_vector(7 downto 0) := (others => '0');
   signal oup : std_logic;

BEGIN
   uut: mod_Parity PORT MAP (
          inp     => inp,
          oup     => oup
        );
   stim_proc: process
   begin
      inp <= "00000000";
      wait for 200ns;
      inp <= "01001010";
      wait for 200ns;
      inp <= "10001010";
      wait for 200ns;
      inp <= "00110000";
      wait for 200ns;
      inp <= "10110101";
      wait;
   end process;
END;
```

Simulation:


0 ns
200 ns
400 ns
600 ns
800 ns
inp[7:0]
oup
00000000
01001010
10001010
00110000
10110101

Question 4

Design a 8 bit barrel shifter. The circuit must shift the input vector of size 8 either 0 or 1 position to the left. When actually shifted (shift = 1), the LSB bit must be filled with '0'. If shift = 0, then out=in; else, if shift = 1, then out(0)='0' and out(i)=in(i-1), for 1≤i≤7. Write a concurrent code for this circuit and verify its behavior.



VHDL Module:

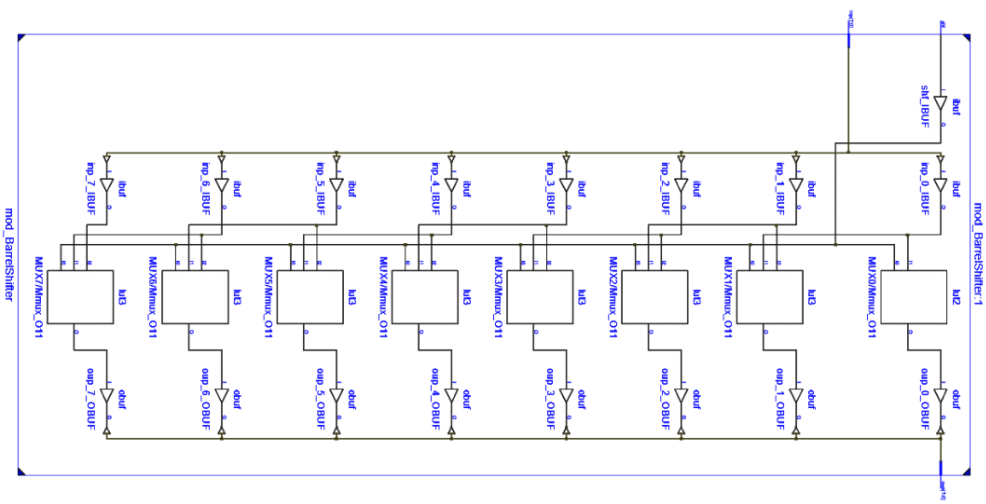
modMux.vhd

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

entity mod_mux is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           S : in  STD_LOGIC;
           O : out STD_LOGIC);
end mod_mux;

architecture Behavioral of mod_mux is

begin
      O <= A when S = '0' else B;
end Behavioral;
```

mod_BarrelShifter.vhd

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

entity mod_BarrelShifter is
    Port ( shf : in  STD_LOGIC;
           inp : in  STD_LOGIC_VECTOR (7 downto 0);
           oup : out STD_LOGIC_VECTOR (7 downto 0));
end mod_BarrelShifter;

architecture Behavioral of mod_BarrelShifter is
      component mod_mux is
            Port ( A : in  STD_LOGIC;
                   B : in  STD_LOGIC;
                   S : in  STD_LOGIC;
                   O : out STD_LOGIC);
      end component;

begin
      MUX0 : mod_mux port map ('0',    inp(0), shf, oup(0));
      MUX1 : mod_mux port map (inp(0), inp(1), shf, oup(1));
      MUX2 : mod_mux port map (inp(1), inp(2), shf, oup(2));
      MUX3 : mod_mux port map (inp(2), inp(3), shf, oup(3));
      MUX4 : mod_mux port map (inp(3), inp(4), shf, oup(4));
      MUX5 : mod_mux port map (inp(4), inp(5), shf, oup(5));
      MUX6 : mod_mux port map (inp(5), inp(6), shf, oup(6));
      MUX7 : mod_mux port map (inp(6), inp(7), shf, oup(7));
end Behavioral;
```

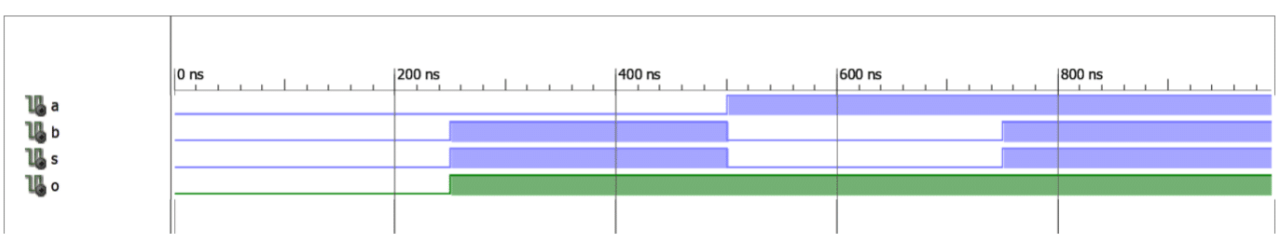
Technology Schematic:



VHDL Test Bench: *tb_mux.vhd*

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_mux IS
END tb_mux;

ARCHITECTURE behavior OF tb_mux IS
    COMPONENT mod_mux
    PORT(
         A : IN  std_logic;
         B : IN  std_logic;
         S : IN  std_logic;
         O : OUT  std_logic
        );
    END COMPONENT;
   signal A : std_logic := '0';
   signal B : std_logic := '0';
   signal S : std_logic := '0';
   signal O : std_logic;

BEGIN
   uut: mod_mux PORT MAP (
          A => A,
          B => B,
          S => S,
          O => O
        );
   stim_proc: process
   begin
            A <= '0', '1' after 500ns;
            B <= '0', '1' after 250ns, '0' after 500ns, '1' after 750ns;
            S <= '0', '1' after 250ns, '0' after 500ns, '1' after 750ns;
      wait;
   end process;
END;
```

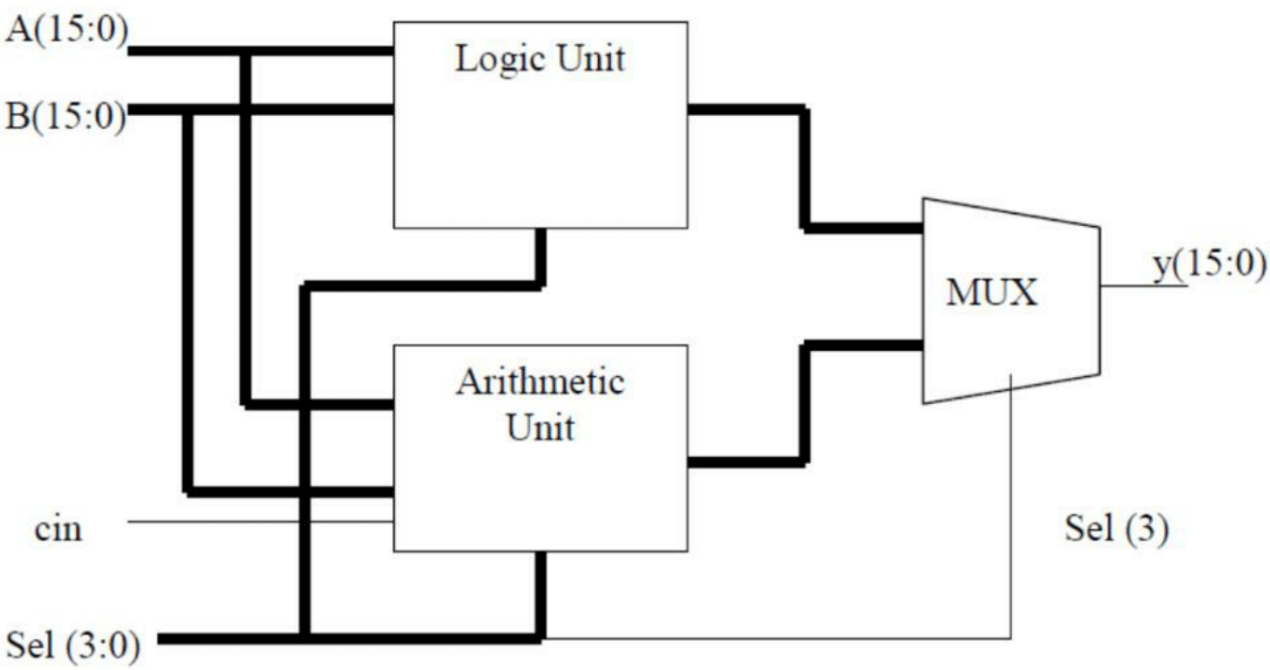
Simulation:



Question 5

Design an 16 bit ALU (Arithmetic Logic Unit) as shown in the following figure and verify it's behavior with simulation

Unit	Function	Sel
Arithmetic	Transfer a	0000
	Increment a	0001
	Decrement a	0010
	Transfer b	0011
	Increment b	0100
	Decrement b	0101
	Add a and b	0110
	Add a and b with carry	0111
Logical	Complement a	1000
	Complement b	1001
	AND	1010
	OR	1011
	NAND	1100
	NOR	1101
	XOR	1110
	XNOR	1111



VHDL Module:

arithUnit.vhd

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

entity arithUnit is
    Port ( S : in  STD_LOGIC_VECTOR ( 3 downto 0);
           A : in  STD_LOGIC_VECTOR (15 downto 0);
           B : in  STD_LOGIC_VECTOR (15 downto 0);
           C : in  STD_LOGIC;
           Y : inout  STD_LOGIC_VECTOR (15 downto 0));
end arithUnit;

architecture Behavioral of arithUnit is
begin

process(S, A, B, C)
begin
      if(S(3) = '0') then
            case S is
                  when "0000" => Y <= A;
                  when "0001" =>
                        Y <= STD_LOGIC_VECTOR(unsigned(A) + 1);
                  when "0010" =>
                        Y <= STD_LOGIC_VECTOR(unsigned(A) - 1);
                  when "0011" => Y <= B;
                  when "0100" =>
                        Y <= STD_LOGIC_VECTOR(unsigned(B) + 1);
                  when "0101" =>
                        Y <= STD_LOGIC_VECTOR(unsigned(B) - 1);
                  when "0110" =>
                        Y <= STD_LOGIC_VECTOR(unsigned(A) + unsigned(B));
                  when "0111" =>
                        Y <= STD_LOGIC_VECTOR(unsigned(A) + unsigned(B));
                        if(C = '1') then Y <= STD_LOGIC_VECTOR(unsigned(Y) + 1);
                        end if;
                  when others => Y <= "0000000000000000";
            end case;
      end if;
end process;

end Behavioral;
```

logicUnit.vhd

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

entity logicUnit is
    Port ( S : in       STD_LOGIC_VECTOR ( 3 downto 0);
           A : in       STD_LOGIC_VECTOR (15 downto 0);
           B : in       STD_LOGIC_VECTOR (15 downto 0);
           Y : inout    STD_LOGIC_VECTOR (15 downto 0));
end logicUnit;

architecture Behavioral of logicUnit is

begin

process(S, A, B)
begin
      if(S(3) = '1') then
            case S is
                  when "1000" => Y <= not A;
                  when "1001" => Y <= not B;
                  when "1010" => Y <= A and  B;
                  when "1011" => Y <= A or   B;
                  when "1100" => Y <= A nand B;
                  when "1101" => Y <= A nor  B;
                  when "1110" => Y <= A xor  B;
                  when "1111" => Y <= A xnor B;
                  when others => Y <= "0000000000000000";
            end case;
      end if;
end process;

end Behavioral;
```

selMux.vhd

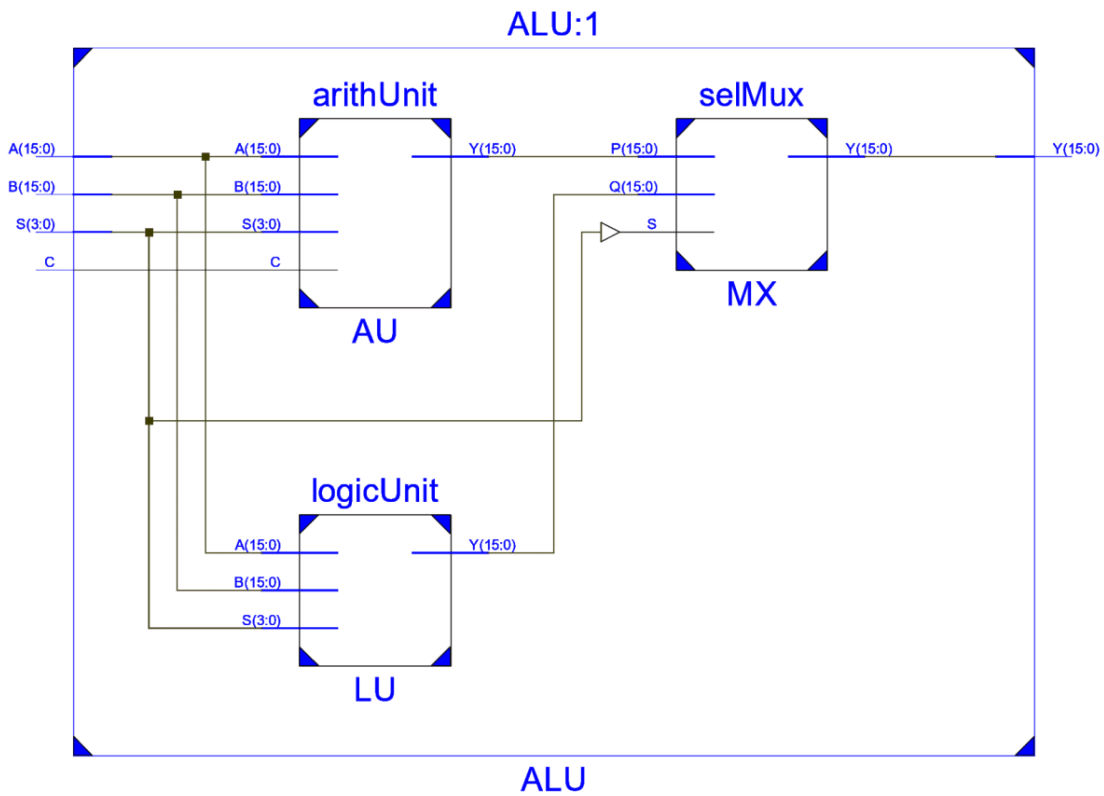
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity selMux is
    Port ( S : in  STD_LOGIC;
                   P : in  STD_LOGIC_VECTOR (15 downto 0);
           Q : in  STD_LOGIC_VECTOR (15 downto 0);
           Y : out STD_LOGIC_VECTOR (15 downto 0));
end selMux;
```

```
architecture Behavioral of selMux is

begin
      Y <= P when S = '0' else Q;
end Behavioral;
```

ALU.vhd

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

entity ALU is
    Port ( A : in  STD_LOGIC_VECTOR (15 downto 0);
           B : in  STD_LOGIC_VECTOR (15 downto 0);
           C : in  STD_LOGIC;
           S : in  STD_LOGIC_VECTOR ( 3 downto 0);
           Y : out STD_LOGIC_VECTOR (15 downto 0));
end ALU;

architecture Structural of ALU is
      signal M, N, O : STD_LOGIC_VECTOR (15 downto 0);

begin

AU : entity work.arithUnit port map(S, A, B, C, M);
LU : entity work.logicUnit port map(S, A, B, N);
MX : entity work.selMux       port map(S(3), M, N, Y);

end Structural;
```

RTL Schematic:



VHDL Test Bench: *tb_ALU.vhd*

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY tb_ALU IS
END tb_ALU;

ARCHITECTURE behavior OF tb_ALU IS
	COMPONENT ALU
		PORT ( A : in  STD_LOGIC_VECTOR (15 downto 0);
		       B : in  STD_LOGIC_VECTOR (15 downto 0);
		       C : in  STD_LOGIC;
		       S : in  STD_LOGIC_VECTOR ( 3 downto 0);
		       Y : out STD_LOGIC_VECTOR (15 downto 0));
	END COMPONENT;

SIGNAL a : STD_LOGIC_VECTOR (15 downto 0) := (others => '0');
SIGNAL b : STD_LOGIC_VECTOR (15 downto 0) := (others => '0');
SIGNAL c : STD_LOGIC := '0';
SIGNAL s : STD_LOGIC_VECTOR ( 3 downto 0) := (others => '0');
```

```
SIGNAL y : STD_LOGIC_VECTOR (15 downto 0);

BEGIN
	uut: ALU PORT MAP(
		A => a,
		B => b,
		C => c,
		S => s,
		Y => y
	);

process
begin
	a <= "0101001010111010";
	b <= "1001001011011011";
	c <= '1';
	s <= "0000";
	while s >= "0000" loop
		wait for 100ns;
		s <= STD_LOGIC_VECTOR(unsigned(s) + 1);
	end loop;
	wait;
end process;
END;
```

Simulation: