Lab2a.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;



entity lab2a is
        generic (W: integer := 8);
        port(   in0, in1, in2, in3, in4: in STD_LOGIC_VECTOR(W-1 downto 0);
        adrPort: in STD_LOGIC_VECTOR(1 downto 0);
        clock, clear:            in  std_logic;
        out0r, out1r, out2r, out3r, out4r : out STD_LOGIC_VECTOR(W-1 downto
0)
        );
end entity lab2a;

architecture routerNode of lab2a is

signal in0r, in1r, in2r, in3r, in4r :  STD_LOGIC_VECTOR(W-1 downto 0);
signal out0, out1, out2, out3, out4:   STD_LOGIC_VECTOR(W-1 downto 0);

component LUT is --include object LUT
        port( address: in STD_LOGIC_VECTOR(1 downto 0);
                sel_LUT0,sel_LUT1,sel_LUT2,sel_LUT3,sel_LUT4: out
STD_LOGIC_VECTOR(2 downto 0)
        );
end component;


signal sel: STD_LOGIC_VECTOR(14 downto 0);


begin
    --map the output from LUT to signal in switch
    assign_LUT_switch: LUT port map(sel_LUT0 => sel(2 downto 0), sel_LUT1 =>
sel(5 downto 3),
    sel_LUT2 => sel(8 downto 6), sel_LUT3 => sel(11 downto 9),sel_LUT4 =>
sel(14 downto 12), address => adrPort);



    Latch0: process (in0, clock,clear) begin
        if (clear = '1') then
            for i in 0 to W-1 loop
                in0r(i) <= '0';
                -- in0r is the 'registered' or 'latched' version of in0
            end loop;
        elsif (clock'EVENT and clock = '1') then
            in0r <= in0;
        end if;
    end process;
```

```
LATCH1:  process (in1, clock, clear) begin
    if clear = '1' then
        for i in 0 to W-1 loop
            in1r(i) <= '0';
        end loop;
    elsif clock'EVENT and clock = '1' then
        in1r <= in1;
    end if;
end process;

LATCH2:  process (in2, clock, clear) begin
    if clear = '1' then
        for i in 0 to W-1 loop
            in2r(i) <= '0';
        end loop;
    elsif clock'EVENT and clock = '1' then
        in2r <= in2;
    end if;
end process;

LATCH3:  process (in3, clock, clear) begin
    if clear = '1' then
        for i in 0 to W-1 loop
            in3r(i) <= '0';
        end loop;
    elsif clock'EVENT and clock = '1' then
        in3r <= in3;
    end if;
end process;

LATCH4:  process (in4, clock, clear) begin
    if clear = '1' then
        for i in 0 to W-1 loop
            in4r(i) <= '0';
        end loop;
    elsif clock'EVENT and clock = '1' then
        in4r <= in4;
    end if;
end process;
--------------------------------------------------------
my_MUX0: process(in0r, in1r, in2r, in3r,in4r,sel(2 downto 0)) --sel0
begin
    case sel(2 downto 0) is
        when "000" => out0 <= in0r;
        when "001" => out0 <= in1r;
        when "010" => out0 <= in2r;
        when "011" => out0 <= in3r;
        when "100" => out0 <= in4r;
        when others => null;
    end case;
end process;

my_MUX1: process(in0r, in1r, in2r, in3r,in4r,sel(5 downto 3)) --sel1
begin
    case sel(5 downto 3) is
        when "000" => out1 <= in0r;
        when "001" => out1 <= in1r;
```

```vhdl
            when "010" => out1 <= in2r;
            when "011" => out1 <= in3r;
            when "100" => out1 <= in4r;
            when others => null;
        end case;
    end process;

    my_MUX2: process(in0r, in1r, in2r, in3r,in4r,sel(8 downto 6)) --sel2
    begin
        case sel(8 downto 6) is
            when "000" => out2 <= in0r;
            when "001" => out2 <= in1r;
            when "010" => out2 <= in2r;
            when "011" => out2 <= in3r;
            when "100" => out2 <= in4r;
            when others => null;
        end case;
    end process;

    my_MUX3: process(in0r, in1r, in2r, in3r,in4r,sel(11 downto 9)) --sel3
    begin
        case sel(11 downto 9) is

            when "000" => out3 <= in0r;
            when "001" => out3 <= in1r;
            when "010" => out3 <= in2r;
            when "011" => out3 <= in3r;
            when "100" => out3 <= in4r;
            when others => null;
        end case;
    end process;

    my_MUX4: process(in0r, in1r, in2r, in3r,in4r, sel(14 downto 12)) --sel4
    begin
        case sel(14 downto 12) is
            when "000" => out4 <= in0r;
            when "001" => out4 <= in1r;
            when "010" => out4 <= in2r;
            when "011" => out4 <= in3r;
            when "100" => out4 <= in4r;
            when others => null;
        end case;
    end process;
    --------------------------------------------------------
    stage2_Latch0: process (out0, clock,clear) begin
     if (clear = '1') then
            for i in 0 to W-1 loop
                out0r(i) <= '0';
                -- out0r is the ‘registered’ or ‘latched’ version of out0
            end loop;
        elsif (clock'EVENT and clock = '1') then
            out0r <= out0;
        end if;
    end process;

    stage2_Latch1: process (out1, clock,clear) begin
     if (clear = '1') then
```

```
                for i in 0 to W-1 loop
                    out1r(i) <= '0';
                    -- out0r is the ‘registered’ or ‘latched’ version of out0
                end loop;
            elsif (clock'EVENT and clock = '1') then
                out1r <= out1;
            end if;
        end process;

        stage2_Latch2: process (out2, clock,clear) begin
         if (clear = '1') then
                for i in 0 to W-1 loop
                    out2r(i) <= '0';
                    -- out0r is the ‘registered’ or ‘latched’ version of out0
                end loop;
            elsif (clock'EVENT and clock = '1') then
                out2r <= out2;
            end if;
         end process;

        stage2_Latch3: process (out3, clock,clear) begin
         if (clear = '1') then
                for i in 0 to W-1 loop
                    out3r(i) <= '0';
                    -- out0r is the ‘registered’ or ‘latched’ version of out0
                end loop;
            elsif (clock'EVENT and clock = '1') then
                out3r <= out3;
            end if;
         end process;

         stage2_Latch4: process (out4, clock,clear) begin
         if (clear = '1') then
                for i in 0 to W-1 loop
                    out4r(i) <= '0';
                    -- out0r is the ‘registered’ or ‘latched’ version of out0
                end loop;
            elsif (clock'EVENT and clock = '1') then
                out4r <= out4;
            end if;
         end process;

end architecture routerNode;
```

LUT.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
```

```
entity LUT is
        port( --address: in STD_LOGIC_VECTOR(14 downto 0);
                --sel_LUT
                sel_LUT0,sel_LUT1,sel_LUT2,sel_LUT3,sel_LUT4: out
STD_LOGIC_VECTOR(2 downto 0);
                address : in STD_LOGIC_VECTOR (1 downto 0) -- 4 cases could
be test

        );
end entity LUT;


architecture selMuxBits of LUT is
begin
process (address) begin
        case address is
            when "00" =>
                sel_LUT0  <= "000";
                sel_LUT1  <= "001";
                sel_LUT2  <= "010";
                sel_LUT3  <= "011";
                sel_LUT4  <= "100";
            when "01" =>
                sel_LUT0  <= "001";
                sel_LUT1  <= "010";
                sel_LUT2  <= "011";
                sel_LUT3  <= "100";
                sel_LUT4  <= "000";
            when "10" =>
                sel_LUT0  <= "010";
                sel_LUT1  <= "011";
                sel_LUT2  <= "100";
                sel_LUT3  <= "000";
                sel_LUT4  <= "001";
            when "11" =>
                sel_LUT0  <= "011";
                sel_LUT1  <= "100";
                sel_LUT2  <= "000";
                sel_LUT3  <= "001";
                sel_LUT4  <= "010";
            when
                others => null;
        end case;
end process;

end architecture selMuxBits;
```

test_bench_2a.vhd

```
library ieee;
```

```vhdl
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std;

entity test_bench_2a is

end entity;

architecture test_behaviour of test_bench_2a is

Shared variable endsim : boolean := FALSE;
Shared variable W : integer := 8;

--components
component lab2a is --instantiate a lab2a component
-- Component Declaration for the Unit Under Test (UUT)


generic (W: integer := 8);
port(   in0, in1, in2, in3, in4: in STD_LOGIC_VECTOR(W-1 downto 0);
out0r, out1r, out2r, out3r, out4r : out STD_LOGIC_VECTOR(W-1 downto 0);
clock, clear :           in  std_logic;
adrPort: in STD_LOGIC_VECTOR(1 downto 0));

end component;

------------------------
--define signals below




signal in0_test, in1_test, in2_test, in3_test, in4_test: STD_LOGIC_VECTOR(W-1
downto 0);  --unsigned
signal out0r_test, out1r_test, out2r_test, out3r_test, out4r_test:
STD_LOGIC_VECTOR(W-1 downto 0);
signal clock_test : STD_LOGIC;
signal clear_test: STD_LOGIC;
signal address_test: STD_LOGIC_VECTOR(1 downto 0);


begin


 -- Instantiate the Unit Under Test (UUT)
   uut: lab2a generic map(w) PORT MAP (            --double check which
signals are to be kept
            clock=>clock_test, clear => clear_test,
            in0=>in0_test, in1=>in1_test, in2=>in2_test, in3=>in3_test,
in4=>in4_test,
            out0r=>out0r_test, out1r=>out1r_test, out2r=>out2r_test,
out3r=>out3r_test, out4r=>out4r_test,
            adrPort => address_test
        );


clock_process :process  --50% duty cycle
```

```vhdl
begin
        if (endsim = FALSE) then
          clock_test <= '0';
        wait for 5ns;  --for 0.5 ns signal is '0'.
        clock_test <= '1';
        wait for 5ns;  --for next 0.5 ns signal is '1'.
          else
                wait;
            end if;
end process clock_process;

--Test Design
test_proc: process(clock_test)
variable clk_counter : integer := 0;
begin

        if(rising_edge(clock_test)) then
            in0_test <= "00000000";
            in1_test <= "00000001";
            in2_test <= "00000010";
            in3_test <= "00000011";
            in4_test <= "00000100";
            clk_counter := clk_counter + 1;

            if (clk_counter = 1) then
                address_test <= "00";
            elsif (clk_counter = 5) then
                address_test <= "01";
            elsif (clk_counter = 10) then
                address_test <= "10";
            elsif (clk_counter = 15) then
                address_test <= "11";
            end if;

        end if;

  end process test_proc;


  KillClock: process (clock_test)
  variable counter: integer:=0;
  Begin
        if(RISING_EDGE(clock_test)) then
            If(counter <30) then           --after 30 CC kill simulation
                counter := counter + 1;
            else
                endsim := true;
                counter := 0;
            end if;
        else

        end if;
  end process;



end architecture test_behaviour;
```