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

entity selector is port(
    a: in std_logic_vector(15 downto 0);
    sel: in unsigned(3 downto 0);
    x : out std_logic_vector(15 downto 0);
    y : out std_logic);
end selector;

architecture rtl of selector is
begin
    y <= a(to_integer(sel));

    process(sel)
    begin
        x <= (others => '0');
        x(to_integer(sel)) <= '1';
    end process;

end rtl;
```

```vhdl
library ieee; --test bench for selector to analyze any errors
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity tbselector is
end tbselector;

architecture beh of tbselector is
    component selector port(
        a: in std_logic_vector(15 downto 0);
        sel: in unsigned(3 downto 0);
        x : out std_logic_vector(15 downto 0);
        y : out std_logic);
end component;

signal a: std_logic_vector(15 downto 0);
signal sel: unsigned(3 downto 0);
signal x, x1 : std_logic_vector(15 downto 0);
signal y, y1: std_logic; --y is the computed out while y1 is the out from component for comparison

constant period : time := 50 ns;
constant strobe : time := 45 ns;

begin
    p0: process
        variable cnt : unsigned(4 downto 0);
    begin
        for j in 0 to 31 loop
            cnt := to_unsigned(j,5);
            sel <= cnt(3 downto 0);--bit 3-0 is SEL signal.
            x <= (others => '0');
            x(to_integer(cnt(3 downto 0))) <= not cnt(4);
            y <= cnt(4);--bit 4 is the expected value of Y.
            --the correct index of the 4-bit binary number in signal a is changed to the expected value.
            --the result is that only the correct index of signal a is the same as the expected values; the
other vaules in a are reversed.
            --the first two types of errors will be caught.
            a <= (a'range => not cnt(4));-- a <= (all of a =  not bit 4). assigns all values to be complement of
the expected value.
            a(to_integer(cnt(3 downto 0))) <= cnt(4);--a(index) <= bit 4
            wait for period;--wait for period before it changes another set of inputs
        end loop;
        wait;
    end process;

    --check: process compares the results Y1 with the expected Y. Wait for strobe
    --time after the inputs are changed to compare the results. Note that input
    --signals are changed every period(50ns). the results are compared 5ns before
    --the inputs are changed so that the strobe is declared as 45ns. This means that
    --after the inputs are changed, we wait for 45ns for the combinational circuits to
    --settle down before the output values are compared. after they are compared, the next
    --set of input values can be used 5ns later. in a synchronous design, the strobe is usually
    --done just before the rising edge of the clock.

    check: process
        variable err_cnt : integer := 0;
    begin
        wait for strobe;
        for j in 0 to 31 loop
            assert false report "comparing..." severity note;
            if(x /= x1) then
                assert false report "x not compared" severity warning;
                err_cnt := err_cnt +1;
            end if;
            if(y /= y1) then
                assert false report "y not compared" severity warning;
                err_cnt := err_cnt +1;
            end if;
            wait for period;
        end loop;
        assert (err_cnt = 0) report "test failed" severity error;
        assert (err_cnt /= 0) report "test passed" severity note;
        wait;
    end process;
    selector1 : selector
        port map(a => a, sel => sel, y => y1, x => x1);
end beh;

configuration cfg_tbselector of tbselector is
    for beh
    for selector1 : selector
    use entity work.selector(rtl);
end for;
end for;
end cfg_tbselector;
```