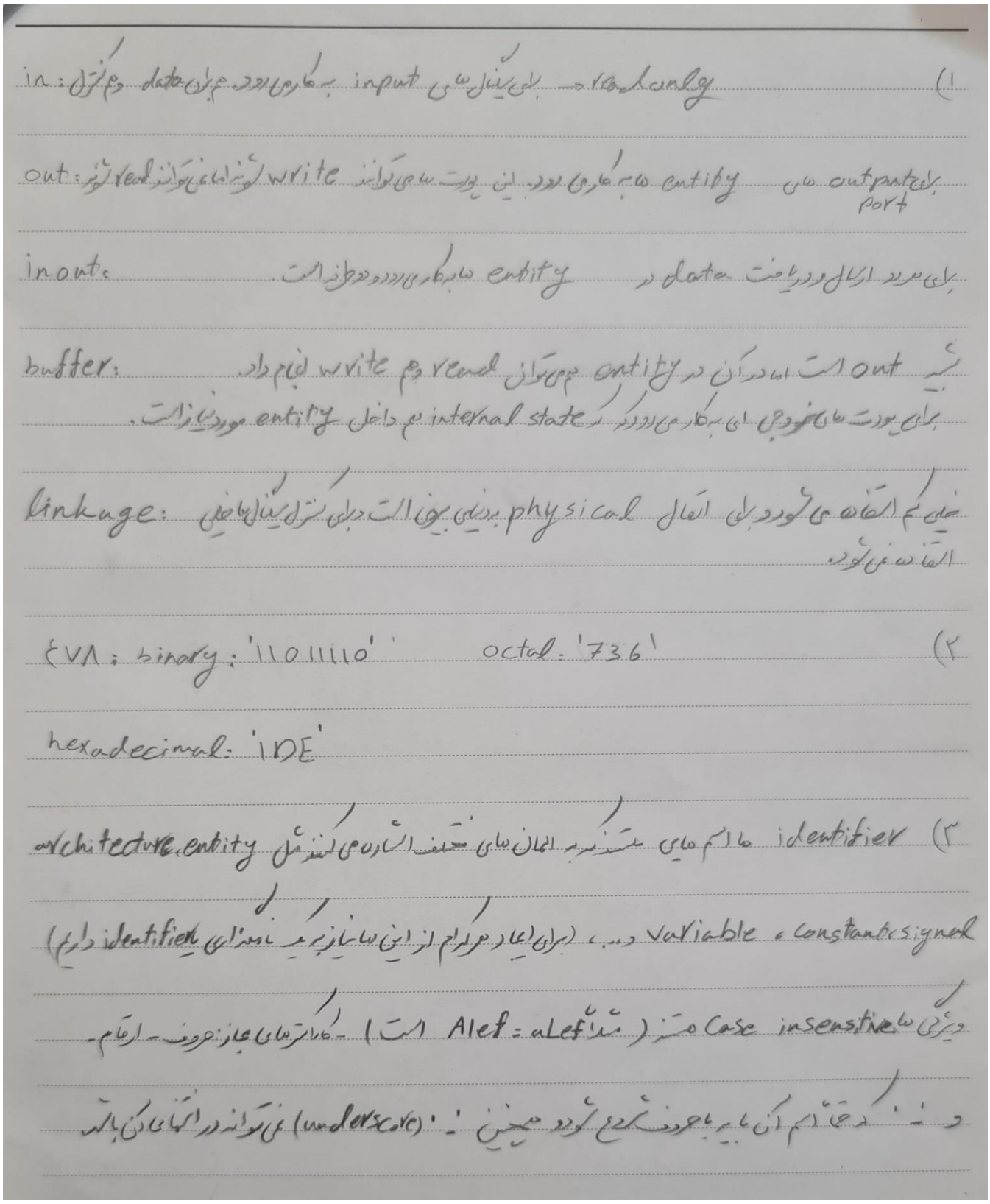
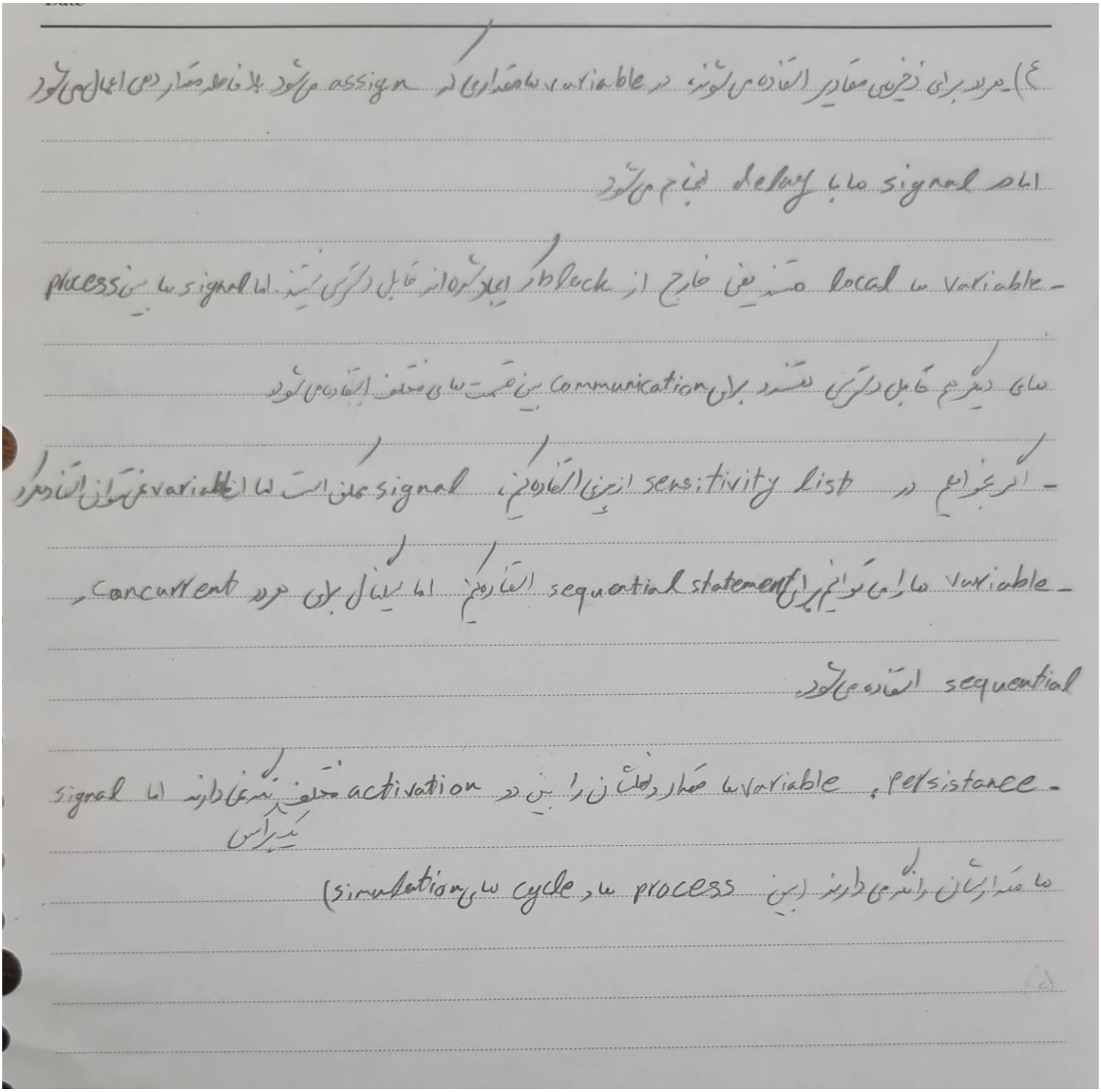
طراحی سیستم های دیجیتال 1402-1403

تمرین1

سیده شکیبا انارکی فیروز 99442047

سوال1)





(5-1

1.Parameterizing Hardware Characteristics

Generics can be used to define key characteristics of hardware components, such as the width of data buses, memory size, or the number of interface ports. This allows a single entity design to be easily adapted for different hardware configurations.

Example:

```
entity FIFO_Buffer is
generic (
 DATA_WIDTH : integer := 8; -- Number of bits in data bus
 BUFFER_SIZE : integer := 64 -- Size of the FIFO buffer );
port ( ... );
end FIFO_Buffer;
```

2. Configuring Operational Parameters

Operational parameters like clock frequencies, timing constraints, or specific operational modes can be parameterized using generics. This enables the tuning of components for optimal performance in different environments.

Example:

```
entity Clock_Divider is
generic (
DIVIDE_BY : integer := 2; -- Factor by which to divide the input clock
); port ( ... );
end Clock_Divider;
```

3. Setting Initial Conditions

Generics can set initial conditions or default states for a design, such as initial values for registers or the default state for a state machine.

Example:

```
entity Counter is
generic (
INITIAL_VALUE : integer := 0; -- Initial value of the counter
); port ( ... );
end Counter;
```

4. Adapting to Different Technology Libraries

When targeting different FPGA or ASIC technology libraries, generics can be used to abstract differences in implementation details, such as logic cell configurations, making it easier to port VHDL code across platforms.

Example:

```
entity Generic_Adder is
generic (
TECH_LIBRARY : string := "ASIC_45nm" -- Target technology library
); port ( ... );
end Generic_Adder;
```

5. Feature Enablement and Experimental Configurations

Generics can enable or disable features of a component, such as debug modes, test features, or optional functionality. This allows a single design to be used in both development and production with different settings.

Example:

```
entity Processor is
generic (
DEBUG_MODE_ENABLED : boolean := false; -- Enable or disable debug mode
); port ( ... );
end Processor;
```

6. Simulation-Specific Parameters

For simulation purposes, generics can specify behaviors or parameters that are only relevant during simulation, such as simulating different operational conditions or hardware faults.

Example:

```
entity Sensor_Simulator is
generic (
FAULT_MODE : boolean := false; -- Simulate a fault condition );
port ( ... );
end Sensor_Simulator;
```

(6-1

کد اصلاح شده آورده شده است و اشکالات به رنگ قرمز هستند:

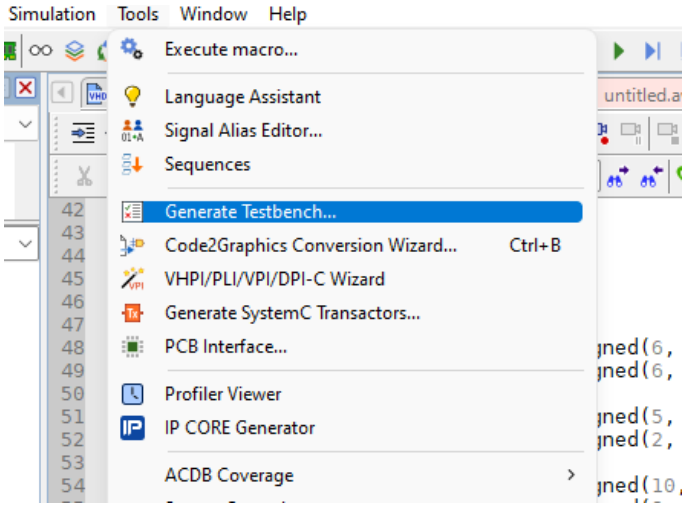
```
entity Example is
   Port (
      input_signal : in STD_LOGIC;
      output_signal : out STD_LOGIC
   );
end Example ;

architecture Behavioral of Example is
   signal internal_signal : STD_LOGIC;
begin
   process
   begin
      if input_signal = '1' then
         internal_signal <= '1';
      else
         internal_signal <= '0';
          end if ;
   end process;

   output_signal <= internal_signal ;
end Behavioral ;
```

سوال2)

روند انجام : ابتدا در نرم افزار active hdl یک workspace ساخته و بعد یک فایل vhd. برای design اضافه می کنیم. سپس کد مقایسه کننده را در آن نوشته و کامپایل می کنیم.

```
--entity declaration with port definitions
entity compare is
port(    num1 : in std_logic_vector(3 downto 0);  --input 1
           num2 :      in std_logic_vector(3 downto 0);  --input 2
           less :              out std_logic;   -- indicates first number is small
          equal :          out std_logic;   -- both are equal
          greater :   out std_logic    -- indicates first number is bigger
);
end compare;

--architecture of entity
architecture Behavioral of compare is

begin
process(num1,num2)
begin
if (num1 > num2 ) then  --checking whether num1 is greater than num2
less <= '0';
equal <= '0';
greater <= '1';
elsif (num1 < num2) then    --checking whether num1 is less than num2
less <= '1';
equal <= '0';
greater <= '0';
else     --checking whether num1 is equal to num2
less <= '0';
equal <= '1';
greater <= '0';
end if;
end process;

end Behavioral;
```

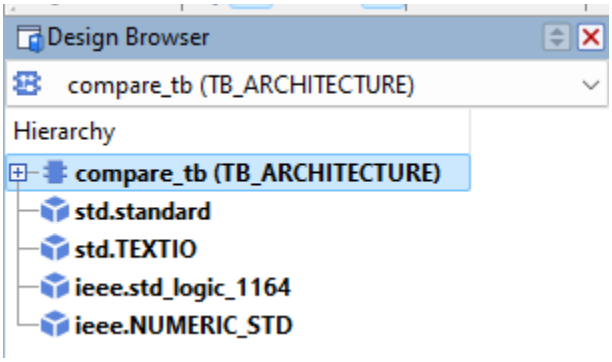
سپس از قسمت tools => generate testbench برای آن فایل testbench ایجاد می کنیم



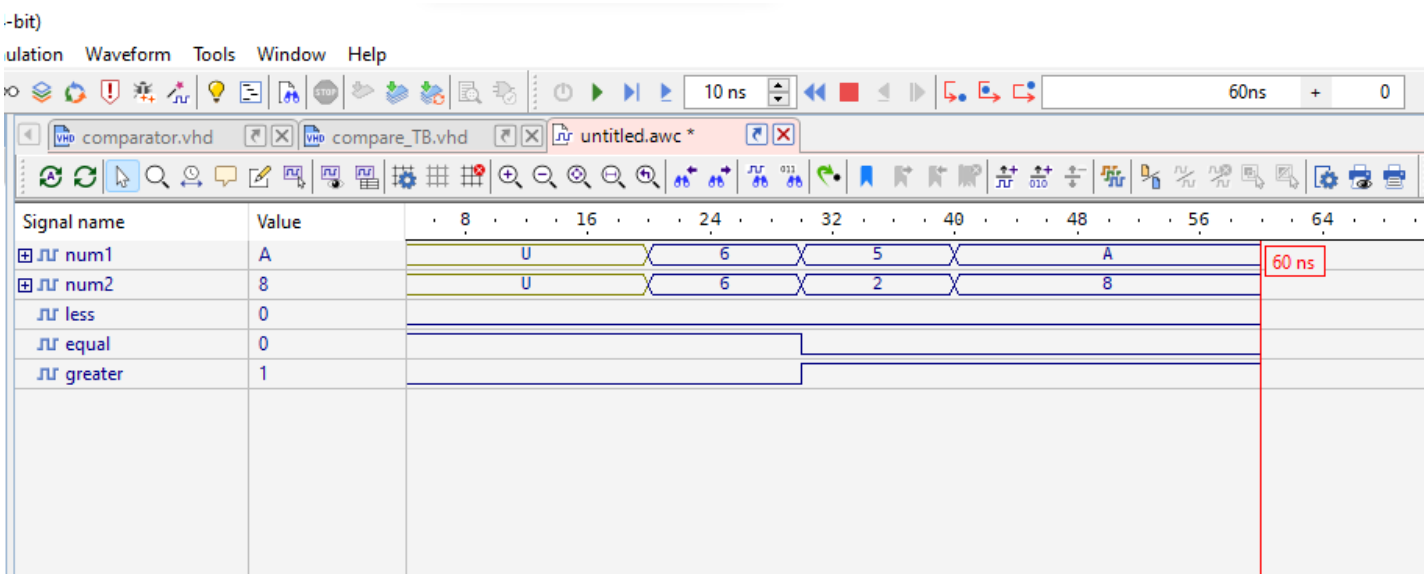
و بعد از اضافه کردن کد قسمت process آن waveform ایجاد می کنیم.

```
    -- Add your stimulus here ...
       process
   begin
       wait for 20 ns;
       num1 <= std_logic_vector(to_unsigned(6, 4));
       num2 <= std_logic_vector(to_unsigned(6, 4));
       wait for 10 ns;
       num1 <= std_logic_vector(to_unsigned(5, 4));
       num2 <= std_logic_vector(to_unsigned(2, 4));
       wait for 10 ns;
       num1 <= std_logic_vector(to_unsigned(10, 4));
       num2 <= std_logic_vector(to_unsigned(8, 4));
       wait for 10 ns;

       -- for i in 0 to 15 loop
--             for j in 0 to 15 loop
--               A <= std_logic_vector(to_unsigned(i, 4));
--                 B <= std_logic_vector(to_unsigned(j, 4));
--                 wait for 10 ns;
--             end loop;
--         end loop;

        wait;
   end process;
```

سپس با انتخاب معماری آن در منوی waveform می توانیم شروع به ران کردن تست کنیم.



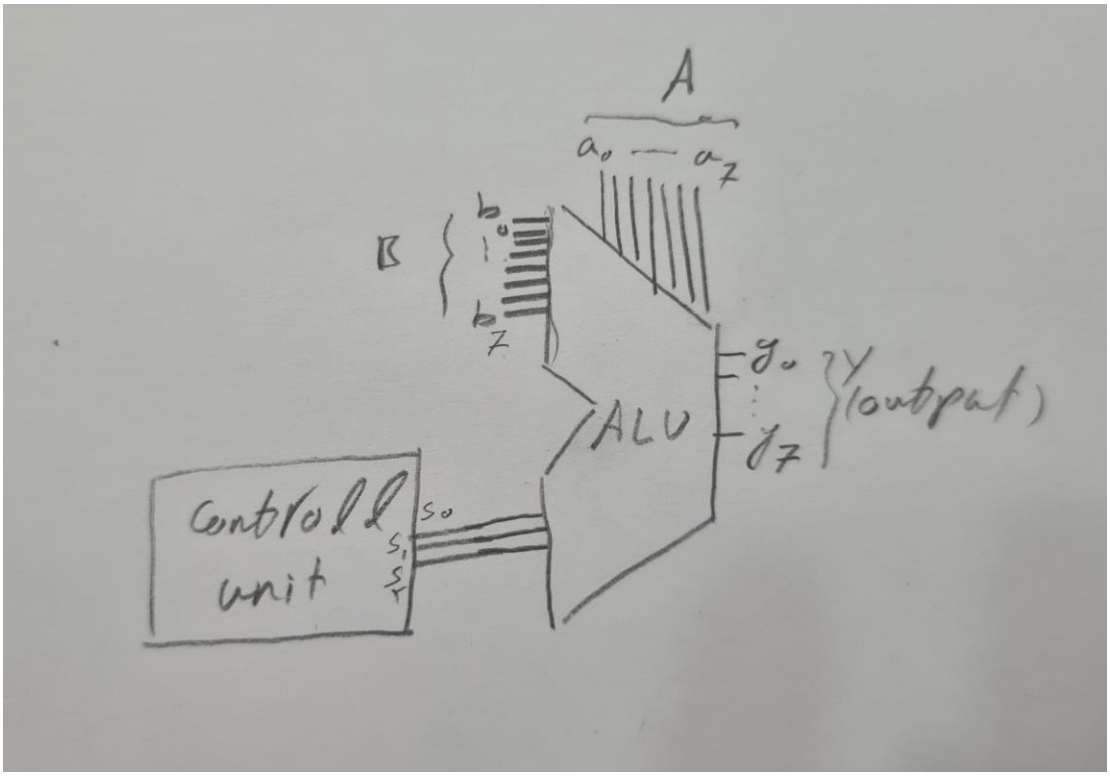
نمونه waveform ران شدن testbench برای سوال دوم:



سوال3)

روند ایجاد پروژه و کد ها را مشابه سوال دوم انجام می دهیم.

تصویر شماتیک:



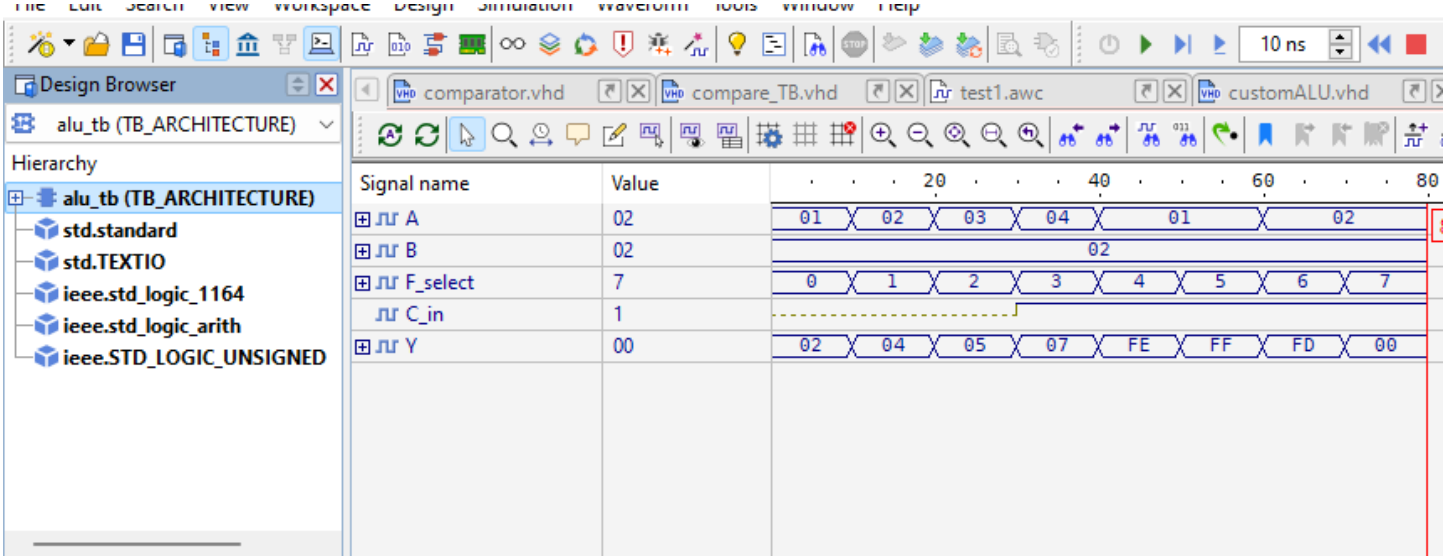
کد entity:

```
architecture Behavioral of ALU is
begin
    process(A, B, F_select, C_in)
    -- the result may have overflow . so we will use 8 bit variable for result
    variable temp_result: STD_LOGIC_VECTOR(8 downto 0);
    begin
        case F_select is
            when "000" =>
            -- A +1
                temp_result := ('0' & A) + 1;
                Y <= temp_result(7 downto 0);
            when "001" =>
                Y <= A + B;
            when "010" =>
                Y <= signed(A) + signed(B);
            when "011" =>
                temp_result := ('0' & A) + ('0' & B) + C_in;
                Y <= temp_result(7 downto 0);
            when "100" =>
                Y <= not A;
            when "101" =>
                Y <= not A + 1;
            when "110" =>
                Y <= A nand B;
            when "111" =>
                Y <= A xor B;
            when others =>
                Y <= (others => '0'); -- default case
        end case;
    end process;
end Behavioral;
```

کد testbench:

```
    -- Add your stimulus here ...
    process
    begin
        -- Test each function in sequence
        -- Increment
        F_select <= "000"; A <= "00000001"; wait for 10 ns;
        -- Unsigned Add
        F_select <= "001"; A <= "00000010"; B <= "00000010"; wait for 10 ns;
        -- Signed Add
        F_select <= "010";
        F_select <= "001"; A <= "00000011"; B <= "10000010"; wait for 10 ns;
        -- Add with carry
        F_select <= "011"; A <= "00000100"; B <= "00000010";
        C_in <= '1'; wait for 10 ns;
        -- complement
        F_select <= "100"; A <= "00000001"; wait for 10 ns;
        -- 2's complement
        F_select <= "101"; A <= "00000001"; wait for 10 ns;
        -- NAND
        F_select <= "110"; A <= "00000010"; B <= "00000010"; wait for 10 ns;
        -- XOR
        F_select <= "111"; A <= "00000010"; B <= "00000010"; wait for 10 ns;

        wait;
    end process;
```

نمونه waveform ران شدن testbench برای سوال سوم:



سوال4)

پیاده سازی به روش when else:

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

entity PriorityEncoder_OPERATORS is
    Port (
        inp : in std_logic_vector(6 downto 0);
        outp : out std_logic_vector(2 downto 0)
    );
end PriorityEncoder_OPERATORS;

architecture Behavioral of PriorityEncoder_OPERATORS is
begin
    outp <= "000" when inp = "0000000" else
            "001" when inp(0) = '1' else
            "010" when inp(1) = '1' else
            "011" when inp(2) = '1' else
            "100" when inp(3) = '1' else
            "101" when inp(4) = '1' else
            "110" when inp(5) = '1' else
            "111";
end Behavioral;
```

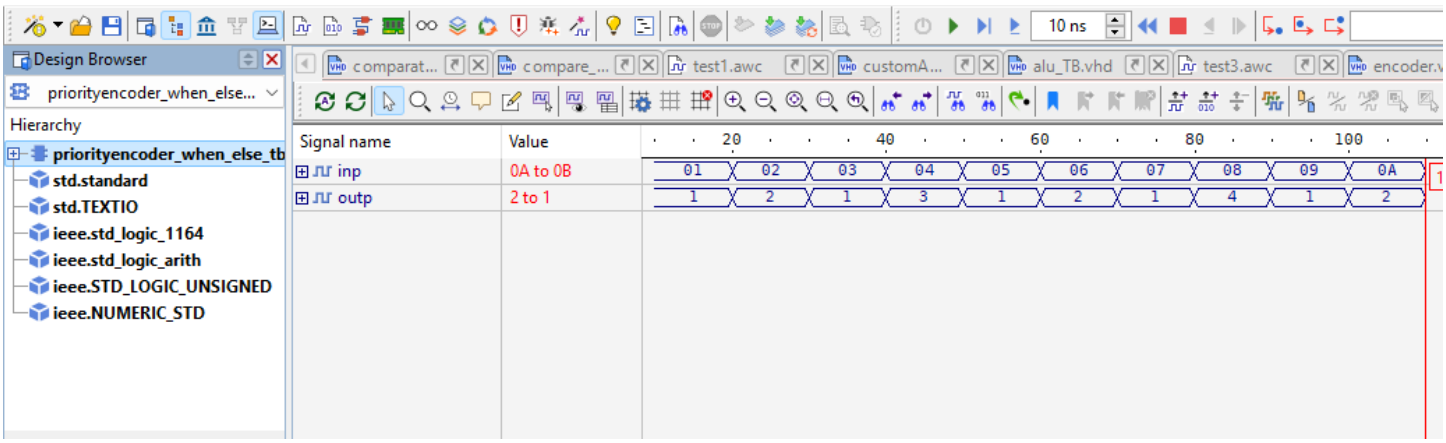
پیاده سازی با گیت های منطقی:

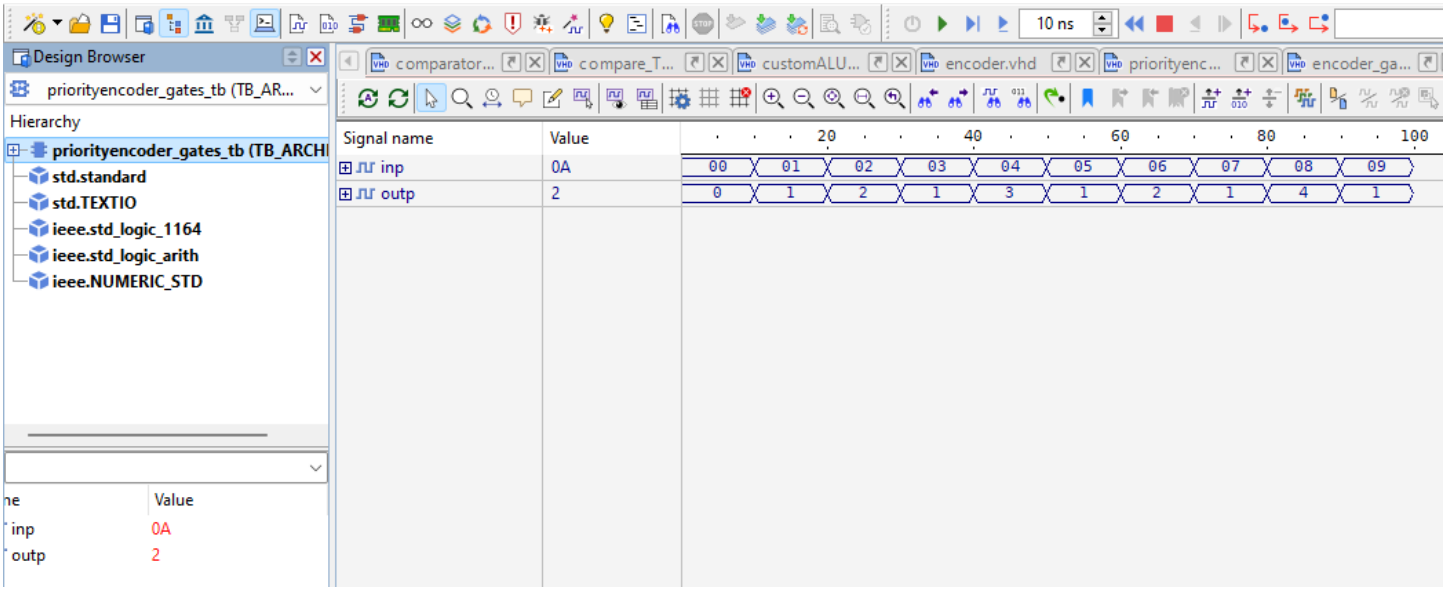
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity PriorityEncoder_gates is
    Port (
        inp : in std_logic_vector(6 downto 0);
        outp : out std_logic_vector(2 downto 0)
    );
end PriorityEncoder_gates;

architecture Behavioral_gates of PriorityEncoder_gates is
begin
   outp(0) <= inp(0) OR
       (NOT inp(0) AND NOT inp(1) AND inp(2)) OR
       (NOT inp(0) AND NOT inp(1) AND NOT inp(2) AND NOT inp(3) AND inp(4)) OR
       (NOT inp(0) AND NOT inp(1) AND NOT inp(2) AND NOT inp(3) AND NOT inp(4) AND NOT inp(5) AND inp(6));

   outp(1) <= (NOT inp(0) AND inp(1)) OR
       (NOT inp(0) AND NOT inp(1) AND inp(2)) OR
       (NOT inp(0) AND NOT inp(1) AND NOT inp(2) AND NOT inp(3) AND NOT inp(4) AND inp(5)) OR
       (NOT inp(0) AND NOT inp(1) AND NOT inp(2) AND NOT inp(3) AND NOT inp(4) AND NOT inp(5) AND inp(6));

    outp(2) <= (NOT inp(0) AND NOT inp(1) AND NOT inp(2) AND inp(3)) OR
       (NOT inp(0) AND NOT inp(1) AND NOT inp(2) AND NOT inp(3) AND inp(4)) OR
       (NOT inp(0) AND NOT inp(1) AND NOT inp(2) AND NOT inp(3) AND NOT inp(4) AND inp(5)) OR
       (NOT inp(0) AND NOT inp(1) AND NOT inp(2) AND NOT inp(3) AND NOT inp(4) AND NOT inp(5) AND inp(6));

end Behavioral_gates;
```

نمونه فایل testbench:

```vhdl
begin

    -- Unit Under Test port map
    UUT : priorityencoder_when_else
        port map (
            inp => inp,
            outp => outp
        );

    -- Add your stimulus here ...
process
    begin
        -- Test all possible combinations
        for i in 0 to 127 loop
            inp <= std_logic_vector(to_unsigned(i, 7));
            wait for 10 ns;
        end loop;
        wait;
    end process;

end TB_ARCHITECTURE;
```

نمونه های waveform ران شدن testbench برای سوال چهارم:




Design Browser
prioritytencoder_gates_tb (TB_AR...
Hierarchy
prioritytencoder_gates_tb (TB_ARCHI
std.standard
std.TEXTIO
ieee.std_logic_1164
ieee.std_logic_arith
ieee.NUMERIC_STD
Value
inp
0A
outp
2
comparator...
compare_T...
customALU...
encoder.vhd
priorityenc...
encoder_ga...
10 ns
Signal name
Value
20
40
60
80
100
inp
0A
00 01 02 03 04 05 06 07 08 09
outp
2
0 1 2 1 3 1 2 1 4 1