Θέμα 1

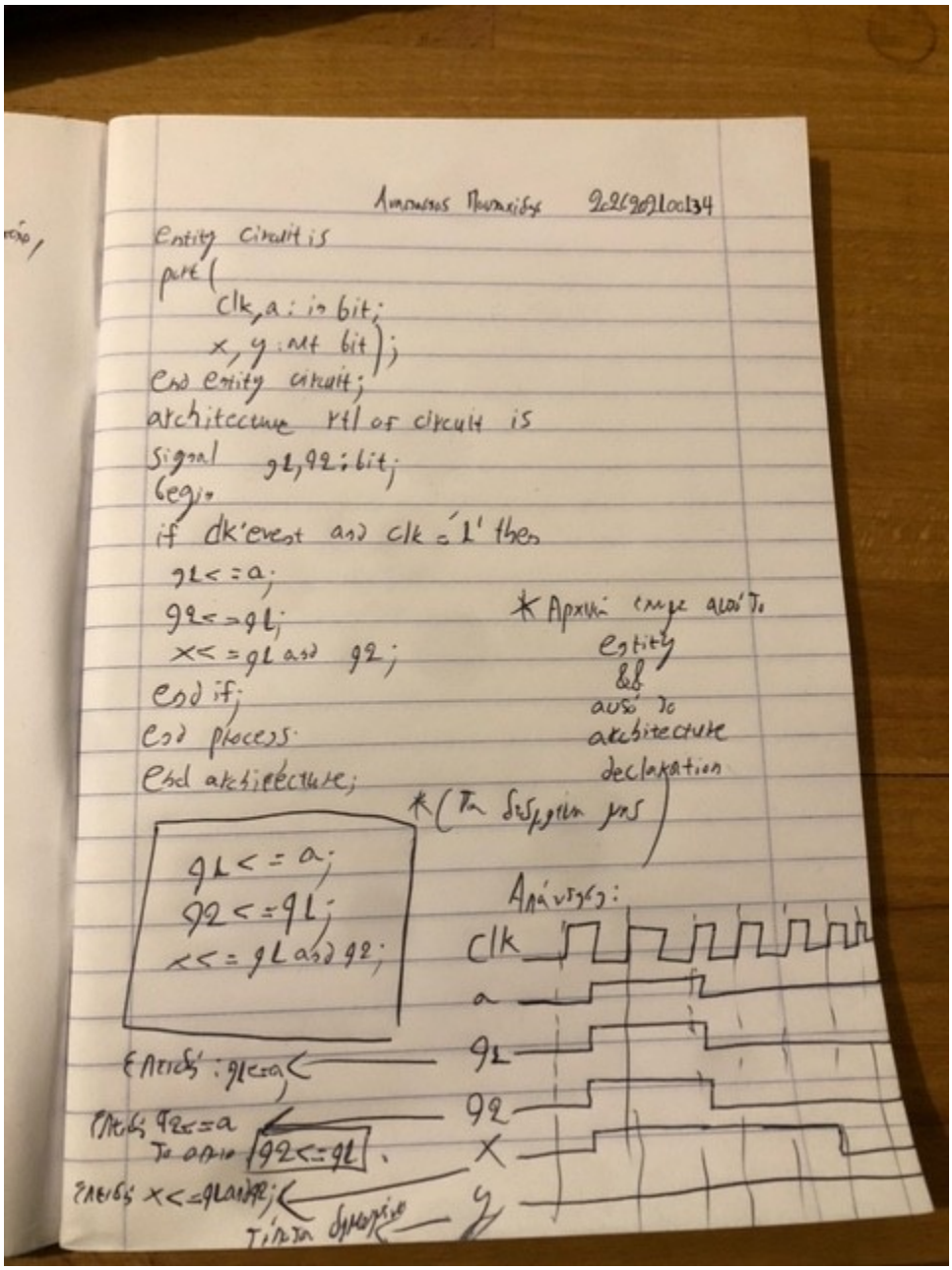
```
library ieee;
use ieee.std_logic_1164.all;
library std;
use std.standard.all;
library work;
use work.all;

--Define Entity of the alarm:
entity AlarmSystem is
   -- Οι είσοδοι και οι έξοδοι του συστήματος
   Port ( sensors : in  STD_LOGIC_VECTOR (3 downto 0);  -- 4 inputs
          zone    : out STD_LOGIC_VECTOR (1 downto 0);  -- output where output the zone
          alarm   : out STD_LOGIC);                     -- Έξοδος που ενεργοποιείται όταν υπάρχει παραβίαση
end AlarmSystem;

-- Definition of the architecture:
architecture Behavior of AlarmSystem is
begin
   -- Η λογική του συστήματος συναγερμού
   process(sensors)
   begin
      -- Ελέγχουμε αν υπάρχει παραβίαση σε κάθε ζώνη
      if sensors = "0001" then
         zone <= "00";  -- Ζώνη 1
         alarm <= '1';  -- Actived!
      elsif sensors = "0010" then
         zone <= "01";  -- Ζώνη 2
         alarm <= '1';  -- Actived!
      elsif sensors = "0100" then
         zone <= "10";  -- Ζώνη 3
         alarm <= '1';  -- Actived!
      elsif sensors = "1000" then
         zone <= "11";  -- Ζώνη 4
         alarm <= '1';  -- Actived!
      else
         zone <= "00";  -- Καμία ζώνη
         alarm <= '0';  --Deactived!
      end if;
   end process;
end Behavior;
```

**Συνέχεια Θεμάτων σε επόμενη σελίδα

Θέμα 2



**Συνέχεια Θεμάτων σε επόμενη σελίδα

Θέμα 3

--ieee library:

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

--Entity Already Defined in the question*
entity shiftreg4 is
    port(
        clk : in bit;  -- Το ρολόι
        sin : in bit;  -- Το εισερχόμενο bit
        dout : out bit_vector(3 downto 0)  -- Τα εξερχόμενα bits
    );
end entity shiftreg4;

-- Architecture for shift register:
architecture behavior of shiftreg4 is
begin
    process(clk)
        variable temp : bit_vector(3 downto 0);  -- temp variable to save 4 bits
    begin
        if rising_edge(clk) then  -- clock rising edge
            temp := sin & dout(3 downto 1);  --Input new bit in the starting position to shift the bits to right
            dout <= temp;  --STORE IT!
        end if;
    end process;
end architecture behavior;
```

**Συνέχεια Θεμάτων σε επόμενη σελίδα

Θέμα 4

```
--libraries needed**
library ieee;
use ieee.std_logic_1164.all;

--entity definition of the counter**
entity counter is
   port(
      clk : in std_logic;  -- clock
      reset : in std_logic;  --
      count : out std_logic_vector(3 downto 0)  --output of counter
   );
end entity counter;

-- the architecture needed
architecture behavior of counter is
   signal temp : std_logic_vector(3 downto 0);  -- temp variable saves the count variable
begin
   process(clk, reset)
   begin
      if reset = '1' then  -- reset=1...
         temp <= "0000";  -- declaring with zeros….
      elsif rising_edge(clk) then  --rised..
         if temp = "0101" then  -- case is 5
            temp <= "1010";  -- go to 10
         elsif temp = "1111" then  -- case is 15 (F) hexadecimal
            temp <= "0000";  -- go to zero….
         else
            temp <= temp + 1;  --OtherWise add 1 to the value of temp!
         end if;
      end if;
   end process;

   count <= temp;  -- STORE IT!
end architecture behavior;
```

**Συνέχεια Θεμάτων σε επόμενη σελίδα

Θέμα 5

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

-- Entity definiton
entity combinational_circuit is
    port ( a : in std_logic_vector(31 downto 0);
           b : in std_logic_vector(31 downto 0);
           function : in std_logic_vector(1 downto 0);
           y : out std_logic_vector(31 downto 0));
end combinational_circuit;

-- Architecture and border of the exercises description in logic circuit
architecture Behavioral of combinational_circuit is
begin
    -- Η διαδικασία που εκτελείται για κάθε αλλαγή στις εισόδους
    process(A, B, function)
    begin
        case function is
            when "00" => -- *AND Operation
                Y <= A and B;
            when "01" => -- *OR Operation
                Y <= A or B;
            when "10" => -- *XOR Operation
                Y <= A xor B;
            when "11" => -- *XNOR Operation
                Y <= not(A xor B);
            when others => -- OthersCases….
                Y <= (others => '0'); -- Default set to zero
        end case;
    end process;
end Behavioral;
```

ΤΕΛΟΣ ΘΕΜΆΤΩΝ