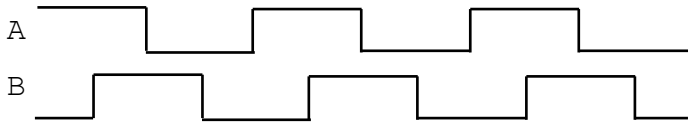
No : Answers **Name : Example Solutions**
Eskişehir Osmangazi University, Faculty of Engineering and Architecture
Department of Electrical Engineering & Electronics 11.01.2023
"*Introduction To FPGA-VHDL*" Final Exam.
Note : Books, notes, computers are allowed, communication of all kind is prohibited. 105 minutes.

1. A serial input signal (`D`) with synchronous clock (`CLK`) is searched for two binary patterns, one following the other but not necessarily consecutively. Let these patterns, for example, be P0=`x"AA"` and P1=`x"CC"`. The circuit has three outputs;
   `A0`: set to 1 when P0 is detected.
   `A1`: set to 1 when P1 is detected.
   `A2`: set to 1 when P1 is detected after the detection of P0.
   Outputs are set to zero when a reset sequence (Pr=`x"01"`) is detected.
   Serial data comes in msb-first. Design the VHDL entity of the circuit.

```
entity SDet is Port (
      D, CLK : in  STD_LOGIC;
  A0, A1, A2 : out STD_LOGIC);
end SDet;

architecture SDet of SDet is

  signal X : STD_LOGIC_VECTOR(7 downto 0);
  constant P0 : STD_LOGIC_VECTOR(7 downto 0) := x"AA";
  constant P1 : STD_LOGIC_VECTOR(7 downto 0) := x"CC";
  constant Pr : STD_LOGIC_VECTOR(7 downto 0) := x"01";
  signal A0x : STD_LOGIC;

begin

  A0 <= A0x;

  process(CLK) is begin
    if(rising_edge(CLK)) then
      X <= X(6 downto 0) & D;
      if(X=P0) then A0x <= '1';
      elsif(X=P1) then A1 <= '1';
        A2 <= A0x; end if;
      elsif(X=Pr) then
        A0x <= '0'; A1 <= '0'; A2 <= '0';
      end if;
    end if;
  end process;

end SDet;
```

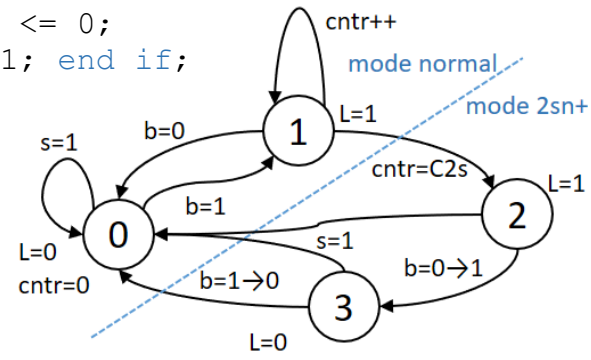
2. A square wave signal pair will be generated with 90° phase difference between them. Frequency is adjustable by two inputs; Inc for increment Dec for decrement. Increment and decrement are performed at the rising edges of these signals. (*A sentence about limit frequencies is removed from here during the exam in order to reduce confusion*). Design the circuit using VHDL.




```
entity Phased is Port (
  Inc, Dec : in  STD_LOGIC;
      A, B : out STD_LOGIC;
       CLK : in  STD_LOGIC); -- 1 MHz
end Phased;

architecture Phased of Phased is
  signal cntrLim, cntr: integer;
  signal pInc, pDec : STD_LOGIC;
  signal AB : STD_LOGIC_VECTOR(0 to 1);
begin

  A <= AB(0); B <= AB(1);
  process(CLK) is begin
    if(rising_edge(CLK)) then
      pInc <= Inc; pDec <= Dec;
      if((pInc&Inc="01")and(cntrLim<500)) then
        cntrLim <= cntrLim +1;
      elsif((pDec&Dec="01")and(cntrLim>0)) then
        cntrLim <= cntrLim -1;
      end if;
      if(cntr>=cntrLim) then
        case AB is
          when "00" => AB <= "10";
          when "10" => AB <= "11";
          when "11" => AB <= "01";
          when others => AB <= "00";
        end case;
        cntr <= 0;
      else cntr <= cntr+1;
      end if;
    end if;
  end process;
end Phased;
```

3. A circuit has two de-bounced button inputs and a logic output (to a LED for example). Pressing and releasing the button `b` activates and deactivates the output respectively. However, keeping the button pressed more than 2 seconds locks it. In that case output stays high even after the button is released. Circuit exits the lock-mode when `b` or `s` button is pressed and the output goes low afterwards.

```
entity MCONT is Port (
  CLK  : in  STD_LOGIC;  -- 1 MHz
  b, s : in  STD_LOGIC;  -- buttons
     L : out STD_LOGIC); -- e.g. LED
end MCONT;

architecture MCONT of MCONT is
  constant C2s : integer := 2000000;
  signal cntr  : integer range 0 to C2s;
  signal pb,Lx : STD_LOGIC := '0';
begin

  process(CLK) is begin
    if(rising_edge(CLK)) then
      pb <= b;
      if(s='1') then           -- reset condition
        cntr <= 0; Lx <= '0';
      elsif(cntr=C2s) then     -- locked case (2sn+)
        if(bp&b="01") then
          Lx <= '0';           -- off but wait for ↓
        elsif(bp&b="10") then -- to reset the counter
          if(Lx='0') then cntr <= 0; end if;
        end if;
      else                     -- normal operation
        Lx <= b;
        if(b='0') then cntr <= 0;
        else cntr <= cntr +1; end if;
      end if;
    end if;
  end process;
  L <= Lx;
end MCONT;
```



two modes of operation

mode 2sn+			
b	**s**	**Lx**	**cntr**
0	0	1	C2s
↑	0	0	C2s
↓	0	0	0
x	1	0	0

normal (cntr<C2s)			
b	**s**	**cntr**	**Lx**
0	0	0	0
1	0	cntr++	1
x	1	0	0

4. A R/W array of records will be created. The record is made of a `std_logic` a `std_logic_vector` of width 8 and an `integer` ranging from 0 to 127.

   a) The record and array type (of size 512) is declared in a package as follows; (do your own type declarations).

```
type rr is record
  flag : STD_LOGIC;
  data : STD_LOGIC_VECTOR(7 downto 0);
  val  : integer range 0 to 127;
end record;
type MType is array (0 to 511) of rr;
```

   b) Array will be used as a single port memory with address, data and WE connections. Complete the design.

```
entity REC is Port (
  CLK  : in  STD_LOGIC;
  Adr  : in  integer range 0 to 511;
  Din  : in  rr;
  Dout : out rr;
  WE   : in  STD_LOGIC);
end REC;

architecture REC of REC is
  signal Mem : MType;
begin

  process(CLK) is begin
    if(rising_edge(CLK)) then
      if(WE='1') then
        Mem(Adr) <= Din;
      end if;
      Dout <= Mem(Adr);
    end if;
  end process;

end REC;
-- note: depending on the synthesizer capability, one may
-- need to convert & combine record members to
-- STD_LOGIC_VECTOR during read/write operations
-- in order to make use of BRAMs.
```