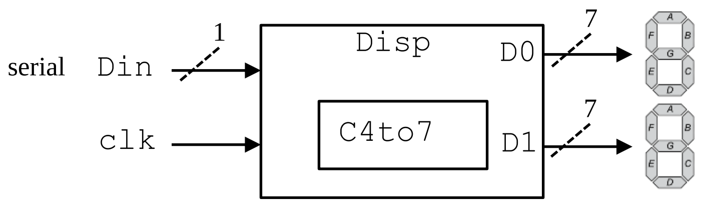
No : Answers **Name : Solutions**
Eskişehir Osmangazi University Faculty of Engineering and Architecture
Department of Computer Engineering 06.01.2020
"*Introduction To VHDL-FPGA*" Final
Note : Books, notes, computers are allowed, communication of all kind is prohibited. 90 minutes.

1. The following display circuit receives 5 bit words serially (MSB first) and continuously. MSB is the display number, the rest is a BCD code of the number to be displayed on the corresponding 7-segment LED. Design the circuit in VHDL with minimum number of elements. Outputs should be stored in two 7-bit registers when the last bit of 5-bit word is received, so that no scanning is necessary.



Use the 4-to-7 combinatorial decoder given.

```
entity DISP is Port (
    CLK, Din :  in STD_LOGIC;
    D0, D1   : out STD_LOGIC_VECTOR(6 downto 0));
end DISP;
architecture DISP of DISP is
  component C4to7 is Port(
    D : in  STD_LOGIC_VECTOR(3 downto 0);
    S : out STD_LOGIC_VECTOR(6 downto 0));
  end component;
  signal R: STD_LOGIC_VECTOR(3 downto 0);
  signal Dx : STD_LOGIC_VECTOR(6 downto 0);
  signal cntr: integer range 0 to 4;


begin
  process(CLK,Din,cntr,R,Dx) is begin
    if(rising_edge(CLK)) then
      if(cntr=3) then
        if(R(3)='0') then D0<=Dx;
        else D1<=Dx; end if;
        cntr<=0;
      else
        R(3)<=R(2); R(2)<=R(1); -- could use loop
        R(1)<=R(0); R(0)<=Din;  -- or R<=R(2 downto 0)&Din;
        cntr<= cntr+1;
      end if;
    end if;
  end process;
end DISP;

CNV: C4to7 port map (
  D=> R(2 downto 0) & Din; -- R3 is the output number
  D=> Dx
);
```

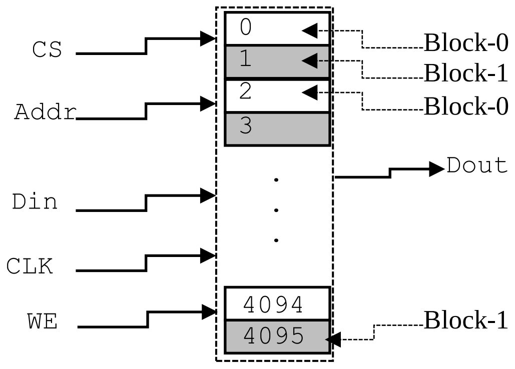
2. Design a GENERIC Watch-Dog timer. Watch-Dog timer is a circuit that activates an output when its input does not show any activity within a specific time duration given with the generic parameter. Generic parameter in this case is the number of clock cycles that is allowed to pass before the output is set high. When the system is Reset this output is also reset to zero synchronously.

```
entity WDog is
  Generic (TM : in integer);
  Port (
    CLK  : in  STD_LOGIC;
    Tin  : in  STD_LOGIC;
    Rst  : in  STD_LOGIC;
    Tovr : out STD_LOGIC);
end WDog;

architecture WDog of WDog is
  signal cntr : integer;
  signal PTin : STD_LOGIC;
begin
  process(CLK,Tin,PTin,Rst) is begin
    if(rising_edge(CLK)) then
      if((Rst='1')or(PTin/=Tin)) then
        cntr<=0; Tovr<='0';
      elsif(cntr=TM) then
        Tovr <= '1';
      else
        cntr<=cntr+1;
      end if;
      PTin<=Tin;
    end if;
  end process;

end WDog;
```

3. We would like to have a 4Kx8 memory using two 2Kx8 memory blocks. However, we also want even addresses physically map into one block and odd addresses into the other. Solve the problem using the given components.



```
entity MEM4K is Port (
  CLK,CS,WE : in  STD_LOGIC;
       Addr : in  STD_LOGIC_VECTOR(11 downto 0);
        Din : in  STD_LOGIC_VECTOR(7 downto 0);
       Dout : out STD_LOGIC_VECTOR(7 downto 0));
);
end MEM4K;
architecture MEM4K of MEM4K is
  component MEM2K is Port(
    CLK,CS,WE : in  STD_LOGIC;
         Addr : in  STD_LOGIC_VECTOR(10 downto 0);
          Din : in  STD_LOGIC_VECTOR(7 downto 0);
         Dout : out STD_LOGIC_VECTOR(7 downto 0));
  end component;
  signal Dout0,Dout1: STD_LOGIC_VECTOR(7 downto 0);
  signal CS0,CS1: STD_LOGIC;
begin
  CS0<= CS and not Addr(0);
  CS1<= CS and Addr(0);
  M0: MEM2K port map(
    CLK   => CLK,
    CS    => CS0,
    WE    => WE,
    Addr => Addr(11 downto 1),
    Din   => Din,
    Dout => Dout0
  );
  M1: MEM2K port map(
    CLK   => CLK,
    CS    => CS1,
    WE    => WE,
    Addr => Addr(11 downto 1),
    Din   => Din,
    Dout => Dout1
  );
  Dout <= Dout0 when Addr(0)='0' else Dout1;

end MEM4K;
```