No : Answers **Name : Solutions**
Eskişehir Osmangazi University Faculty of Engineering and Architecture
Department of Electrical Engineering & Electronics 26.01.2022
"*Introduction To VHDL-FPGA*" Final Exam
Note : Books, notes, computers are allowed, communication of all kind is prohibited. 90 minutes.

1. Determine if the VHDL designs involving the following problems require key-debouncers. Explain why/why not?

a) An up/down counter controlled with two buttons named `bUp` & `bDown`; Pressing `bUp` or `bDown` buttons increments or decrements the counter respectively.
Debouncers for both buttons are necessary. Otherwise counter may increment/decrement multiple times when a button is pressed.

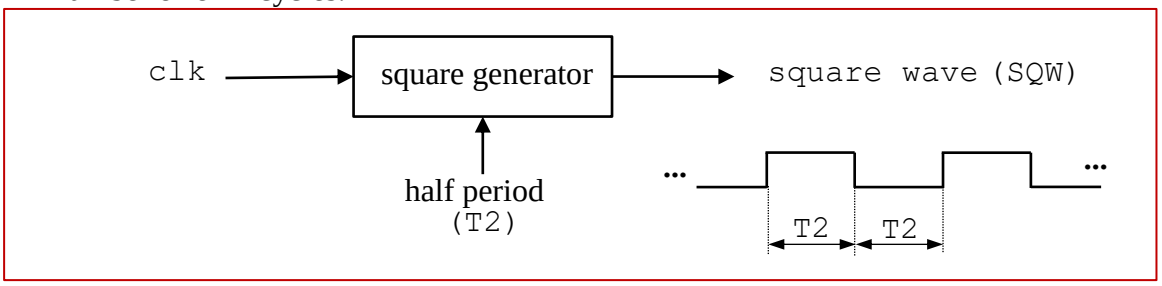
b) A button named `bLight` turns either on or off a light control circuitry; For example, if the light is off, the user presses `bLight` to turn it on and presses again to turn it off.
Debouncer is necessary. Otherwise the toggle circuit may got triggered multiple times.

c) A three button key-entry circuit with button names `b0`, `b1` and `bEnter`; User is supposed to enter binary key sequence using `b0` and `b1` buttons. User is required to press `bEnter` after each `b0/b1` press. For example, if the user is trying to enter "011", the key sequence he/she is expected to press is "`b0,bEnter,b1,bEnter,b1,bEnter`".
Debouncer is not necessary since a `bEnter` key is required between button presses.

d) An up/down counter controlled with two buttons named `bUp` & `bDown`; Pressing one of the buttons and keeping it pressed automatically increments/decrements the counter at 0.5s intervals. For example, pressing `bUp` for 0.9 seconds increments the counter twice; once just after the press event and once at 0.5s after that event. The remaining 0.4s will be forgotton if the user do not press any button for 0.5s afterwards.
Debouncer is not necessary. Because multiple increments/decrements are desired and a delay of 0.5s acts like a debouncer anyway.

e) A circuit with two states is controlled with a switch and a button named as `swS` and `bS` respectively; User is expected to set the `swS` according to the selected state and presses the `bS` afterwards to activate the state.
Debouncer is not necessary, because the switch is not expected to change position when the button is pressed. Multiple transitions of `bS` will cause the reading of the same value on `swS` signal.

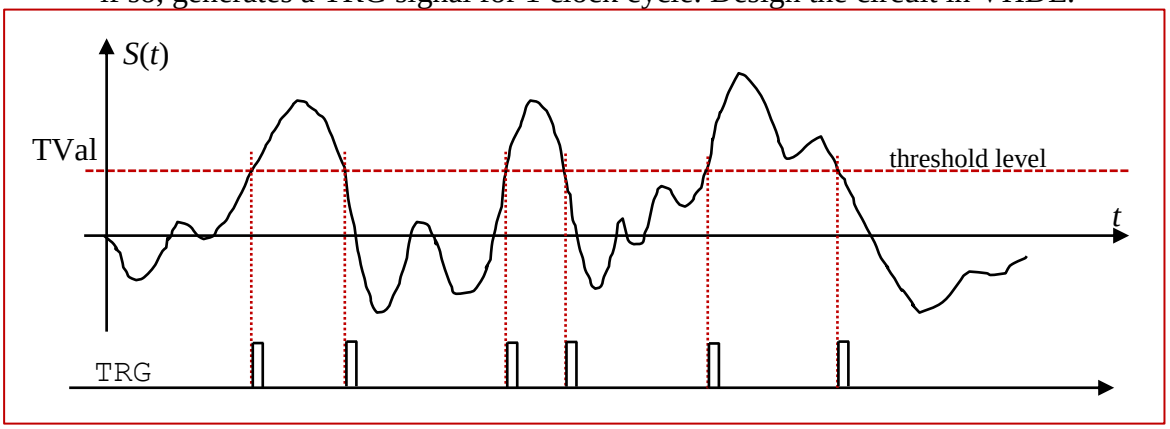
2. Design the following square-wave generator in VHDL. The design should force a gracefull transition from one period selection to another (very important). `T2` is given in number of `clk` cycles.




```
entity SQGen is port (
  clk : in  STD_LOGIC;
  T2  : in  integer;
  SQW : out STD_LOGIC);
end SQGen;
architecture SQGen of SQGen is
  signal sqs : STD_LOGIC := '0';
  signal cntr : integer := 0;

begin
  SQW <= sqs;
  SQWGen:process(clk,T2,cntr,sqs) is begin
    if(rising_edge(clk)) then
      if(cntr>=T2-1) then
        cntr <= 0;
        sqs <= not(sqs);
      else
        cntr <= cntr+1;
      end if;
    end if;
  end process;
end SQGen;
```

3. Sufficiently dense samples of the following stationary signal is given to a trigger generating circuit. The circuit detects if the signal passes through a threshold level and if so, generates a TRG signal for 1 clock cycle. Design the circuit in VHDL.




```
entity TrigGen is port (
  clk   : in  STD_LOGIC;
  S     : in  integer;
  TVal  : in  integer;
  TRG   : out STD_LOGIC);
end TrigGen;
architecture TrigGen of TrigGen is

  signal pS : integer;

begin
  process(clk,S,TVal) is begin
    if(rising_edge(clk)) then
      pS <= S;
      if(((pS<=TVal)and(TVal<=S)) or
         ((pS>=TVal)and(TVal>=S))) then
        TRG <= '1';
      else
        TRG <= '0';
      end if;
    end if;
  end process;
end TrigGen;
```

4. Construct a 128x8 memory using 64x8 memory blocks given. (You may draw a legible block diagram instead).

```
entity M128x8 is port (
  clk   : in  STD_LOGIC;
  WE    : in  STD_LOGIC; -- Write enable
  Adr   : in  STD_LOGIC_VECTOR(6 downto 0);
  Din   : in  STD_LOGIC_VECTOR(7 downto 0);
  Dout  : out STD_LOGIC_VECTOR(7 downto 0));
end M128x8;
architecture M128x8 of M128x8 is
  component M64x8 is port(
    clk   : in  STD_LOGIC;
    WE    : in  STD_LOGIC;  -- Write enable
    Adr   : in  STD_LOGIC_VECTOR(5 downto 0);
    Din   : in  STD_LOGIC_VECTOR(7 downto 0);
    Dout  : out STD_LOGIC_VECTOR(7 downto 0));
  end component;
  signal WE0,WE1: STD_LOGIC;
  signal AdrL: STD_LOGIC_VECTOR(5 downto 0);
  signal Dout0,Dout1: STD_LOGIC_VECTOR(7 downto 0);

begin
  WE0 <= (not Adr(6)) and WE;
  WE1 <= Adr(6) and WE;
  AdrL <= Adr(5 downto 0);
  M0: M64x8 port map(clk,WE0,AdrL,Din,Dout0);
  M1: M64x8 port map(clk,WE1,AdrL,Din,Dout1);
  Dout <= Dout0 when Adr(6)='0' else Dout1;

end M128x8;
```