1. (15 points) A barrel shifter is a shift register in which the data can be shifted either by one bit position, as in a normal shift register, or by multiple positions. Design a four-bit barrel shifter that can shift to the right or left by 0, 1, 2, or 3 bits and has clear and parallel load capabilities. If DIR = ‘0’, shift left, else shift right. The binary value of S1 and S0 dictate the amount of shift, for example S1S0 = 10 means shift by 2 bit positions.

(a) (4 points) Write an entity for the barrel shifter. (c) (11 points) Write an architecture for the barrel shifter

entity BARREL is
port (D : in std_logic_vector(3 downto 0);
    CLK, LOAD, CLR, DIR : in std_logic;
    S : in std_logic_vector(1 downto 0);
    Q : out std_logic_vector(3 downto 0));
end BARREL;

architecture BARREL of BARREL is
signal TEMP : std_logic_vector (3 downto 0);
begin
    process (CLK)
    begin
        if (CLK'event and CLK = '1') then
            if (CLR = '0') then TEMP <= "0000";
            elsif (LOAD = '1') then TEMP <= D;
            elsif (DIR = '0') then
                case S is
                    when "00" => TEMP <= TEMP;
                    when "01" => TEMP <= TEMP(2 downto 0) & '0';
                    when "10" => TEMP <= TEMP(1 downto 0) & "00";
                    when "11" => TEMP <= TEMP(0) & "000";
                    when others => TEMP <= TEMP;
                end case;
            else
                case S is
                    when "00" => TEMP <= TEMP;
                    when "01" => TEMP <= '0' & TEMP(3 downto 1);
                    when "10" => TEMP <= "00" & TEMP(3 downto 2);
                    when "11" => TEMP <= "000" & TEMP(3);
                    when others => TEMP <= TEMP;
                end case;
            end if;
        else
            case S is
                when "00" => TEMP <= TEMP;
                when "01" => TEMP <= '0' & TEMP(3 downto 1);
                when "10" => TEMP <= "00" & TEMP(3 downto 2);
                when "11" => TEMP <= "000" & TEMP(3);
                when others => TEMP <= TEMP;
            end case;
        end if;
    end if;
end process;
Q <= TEMP;
end BARREL;
2. (1 point) A function is a primary design unit. (True/False) __False__

3. (20 points). (a) (12 points) Write a VHDL function to compare two IEEE std_logic_vectors to see whether they are equal. Report an error if any bit in either vector is not ‘0’, ‘1’, or ‘-’ (don’t care), or if the lengths of the vectors are not the same. The function call should pass only the vectors. The function should return TRUE if the vectors are equal, else FALSE. When comparing the vectors, consider that ‘0’ = ‘-’ and ‘1’ = ‘-’. Make no assumptions about the index range of the two vectors. (b) (8 points) Show an architecture that includes three calls to the function with the following properties. 1 - returns TRUE, 2 - returns FALSE and 3 - triggers an error message.

```vhdl
package NEED_IT is
  function "=" (l, r : in std_logic_vector) return BOOLEAN;
end NEED_IT;

library ieee;
use ieee.std_logic_1164.all;
package body NEED_IT is
  function "=" (l, r : in std_logic_vector) return BOOLEAN is
    alias L_NEW : std_logic_vector(L'length - 1 downto 0) is L;
    alias R.NEW : std_logic_vector(R'length - 1 downto 0) is R;
    variable MATCH : BOOLEAN;
  begin
    assert (L'length = R'length)
      report "Vectors are not the same length"
        severity ERROR;
    MATCH := TRUE;
    for I in L_NEW'range loop
      if ((L_NEW(I) /= '0' and L_NEW(I) /= '1' and L_NEW(I) /= '-') or
          (R.NEW(I) /= '0' and R.NEW(I) /= '1' and R.NEW(I) /= '-')) then
        report "Vectors have illegal values"
          severity ERROR;
      end if;
      if (L.NEW(I) = '0') then
        if (R.NEW(I) /= '0' and R.NEW(I) /= '-') then
          MATCH := FALSE;
        end if;
      elsif (L.NEW(I) = '1') then
        if (R.NEW(I) /= '1' and R.NEW(I) /= '-') then
          MATCH := FALSE;
        end if;
      end if;
    end loop;
    return MATCH;
  end;
end NEED_IT;

use work.NEED_IT.all;
architecture TEST_EQUAL of TEST_EQUAL is
  signal EQUAL : BOOLEAN;
begin
  process
    variable A  : std_logic_vector(8 downto 3) := "0-1011";
    variable B : std_logic_vector (93 downto 86) := "00000000";
    variable C : std_logic_vector(47 to 52) := "011--1";
    variable D : std_logic_vector(0 to 5) := "011-10";
  begin
    EQUAL <= A = B after 10 ns;
    EQUAL <= transport A = C after 20 ns;
    EQUAL <= transport A = D after 30 ns;
  wait;
  end process;
end TEST_EQUAL;
```
4. (1 point) A process may have both a sensitivity list and wait statements (True/False) ___False___

5. (1 point) A ___configuration__ binds an instantiated component to a library model

6. (10 points) A clocked T flip-flop with synchronous CLEAR and PRESET operates in the following manner: At the falling edge of CLK, Q = ‘0’ and QB = ‘1’ if CLEAR = ‘1’, Q = ‘0’ and QB = ‘1’ if PRESET = ‘1’ and Q = not Q and QB = not QB if T = ‘1’ and Q and QB remain unchanged if T = ‘0’. The delay associated with CLEAR and PRESET is TPCPQ, the delay associated with T is TPTQ. (a) (2 points) Write an entity for this flip-flop. (b) (5 points) Write an architecture for this flip-flop.

```vhdl
entity TFF is
  generic (TPCPQ, TPTQ : time);
  port (CLEAR, PRESET, CLK, T : in std_logic;
    Q, QB : out std_logic);
end TFF;

architecture BEHAV of TFF is
  signal TEMP : std_logic;
begin
  process(CLK)
  begin
    if (CLK = '0' and CLK'event) then
      if (CLEAR = '1') then
        TEMP <= '0' after TPCPQ;
      elsif (PRESET = '1') then
        TEMP <= '1' after TPCPQ;
      elsif (T = '1') then
        TEMP <= not TEMP after TPTQ;
      else
        TEMP <= TEMP;
      end if;
    end if;
  end process;
  Q <= TEMP;
  QB <= not TEMP;
end BEHAV;
```

7. (5 points) Consider the following structural VHDL model.

```vhdl
entity SMODEL is
  port
    (P1 : in BIT;
     P2 : out BIT;
     P3 : inout BIT);
end SMODEL;

architecture STRUCTURE of SMODEL is
  component UNIT
    port (C1, C2, : in BIT; C3 : out BIT);
  end component;
begin
  U1 : UNIT port map (C1 => P1, C2 => P3, C3 => P2);
end STRUCTURE;
```
(a) (3 points) Complete the structural description by giving a legal set of port-to-port connections for entity ports P1, P2, and P3 and component ports C1, C2, and C3.

(b) (2 points) Is there more than one possible set of legal port-to-port connections? Yes

P1, P3, P2 or P3, P1, P2

8. (15 points) For the following VHDL, assume that A changes to ‘1’ at 5 ns and back to ‘0’ at 12 ns. Give the values of A, B, C, D, E, and F each time a change occurs. Carry this out until no further change occurs.

```vhdl
entity prob is
  port (D : inout bit);
end prob;
architecture PROB of prob is
  signal A, B, C, E, F : bit;
begin
  P1: process (A, C)
  begin
    B <= A after 2 ns;
    E <= C after 7 ns;
  end process P1;
  C <= A and B after 6 ns;
  P2: process (C, E)
  begin
    F <= C and E after 4 ns;
  end process P2;
  D <= A or B or C or F;
end PROB;
```

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
<th>Triggered</th>
<th>Scheduled</th>
<th>Event?</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 ns</td>
<td>A 0→1</td>
<td>P1</td>
<td>B ‘1’ 7ns</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E ‘0’ 12 ns</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C</td>
<td>C ‘0’ 12 ns</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D</td>
<td>D ‘1’ 5+Δ ns</td>
<td>Yes</td>
</tr>
<tr>
<td>5+Δns</td>
<td>D 0→1</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>B 0→1</td>
<td>C</td>
<td>C ‘1’ 13 ns</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D</td>
<td>D ‘1’ 7+Δ ns</td>
<td>No</td>
</tr>
<tr>
<td>12</td>
<td>A 1→0</td>
<td>P1</td>
<td>B ‘0’ 14 ns</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E ‘0’ 19 ns</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C</td>
<td>C ‘0’ 18 ns</td>
<td>Overwrites C ‘1’ 13 ns, No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D</td>
<td>D ‘1’ 12+Δ ns</td>
<td>No</td>
</tr>
<tr>
<td>14</td>
<td>B 1→0</td>
<td>C</td>
<td>C ‘0’ 20 ns</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D</td>
<td>D ‘0’ 14+Δ ns</td>
<td>Yes</td>
</tr>
<tr>
<td>14+Δns</td>
<td>D 1→0</td>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9. (1 point) In order to specify edge behavior the _`STABLE_ attribute is used in concurrent statements.

10. (4 points) (a) (2 points) Specify a DAY_OF_WEEK enumeration data type.(b) (2 points) Write a variable declaration CURRENT_DAY that has a value equal to the current day of the week.

```vhdl
package TRY is
  type DAY_OF_WEEK is (SU, MO, TU, WE, TH, FR, SA);
  variable CURRENT_DAY : DAY_OF_WEEK := TH;
end TRY;
```

11. (10 points) Design a 2 to 4 decoder with enable. All outputs are tristated when the enable input = ‘0’. When the enable input = ‘1’, one of the four output D0, D1, D2, D3 is selected based on the binary value of the two select inputs S1 and S0. (a) (2 points) Write a VHDL entity. (b) (4 points) Use
concurrent signal assignments to implement the architecture. (c) (4 points) Use sequential statements to implement the architecture. Include any necessary library references.

library ieee;
use ieee.std_logic_1164.all;
entity TWO_TO_FOUR is
port (I : in std_logic_vector(1 downto 0);
    EN : in std_logic;
    O : out std_logic_vector(3 downto 0));
end TWO_TO_FOUR;
architecture CONCURRENT of TWO_TO_FOUR is
    signal TEMP : std_logic_vector(3 downto 0);
begin
    with I select
    TEMP <= "0001" when "00", "0010" when "01",
             "0100" when "10", "1000" when "11",
             "0000" when others;
    O <= TEMP when EN = '1' else "ZZZZ";
end CONCURRENT;
architecture SEQUENTIAL of TWO_TO_FOUR is
begin
    process(I)
    variable TEMP : std_logic_vector(3 downto 0);
    begin
        case I is
            when "00" => TEMP := "0001";
            when "01" => TEMP := "0010";
            when "10" => TEMP := "0100";
            when "11" => TEMP := "1000";
            when others => TEMP := "0000";
        end case;
        if (EN = '1') then  O <= TEMP;
        else O <= "ZZZZ";
        end if;
    end process;
end SEQUENTIAL;

12. (1 point) ___Communication__ is the hardest problem.

13. (6 points) (a) (4 points) Write a declaration of a record data type, PERSONNEL, with fields for last name (LAST) (up to twenty characters); first name (FIRST) (up to twenty characters); middle initial (MID); and social security number (SOC_SEC).(b) (2 points) Write a declaration for a constant MY_PERSONNEL_RECORD that defines your own data, consistent with type PERSONNEL.

package MINE is
    type PERSONNEL is record
        LAST : string(20 downto 1);
        FIRST : string(20 downto 1);
        MID : character;
        SOC_SEC : integer range 0 to 999999999;
    end record;
    constant MY_PERSONNEL_RECORD : PERSONNEL := ("Gaede","Rhonda","K",999274819);
end MINE;
14. (10 points) Draw the state diagram for the following state machine. Is it a Moore machine or a Mealy machine?

Mealy

ENTITY state_machine IS
  PORT (sig_in : IN BIT; clk : IN BIT;
        sig_out : OUT BIT);
END state_machine;

ARCHITECTURE state_machine OF state_machine IS
  TYPE state_type IS (a, b, c, d, e);
  SIGNAL current_state, next_state : state_type;
BEGIN
  PROCESS (sig_in, current_state)
  BEGIN
    sig_out <= '0';
    next_state <= e;
    CASE current_state
    WHEN a =>
      IF sig_in = '0' THEN
        next_state <= a;
        sig_out <= '1';
      ELSE
        next_state <= d;
      END IF;
    WHEN b =>
      IF sig_in = '0' THEN
        next_state <= b;
      ELSE
        next_state <= c;
        sig_out <= '1';
      END IF;
    WHEN c =>
      IF sig_in = '1' THEN
        sig_out <= '1';
        next_state <= a;
      ELSE
        next_state <= e;
      END IF;
    WHEN d =>
      IF sig_in = '0' THEN
        sig_out <= '1';
        next_state <= e;
      END IF;
    WHEN e =>
      IF sig_in = '1' THEN
        next_state <= c;
      END IF;
    END CASE;
  END PROCESS;
  PROCESS (clk)
  BEGIN
    IF (clk'EVENT AND clk = '1') THEN
      current_state <= next_state;
    END IF;
  END PROCESS;
END state_machine;