1. (15 points) (a) (4 points) Create a VHDL entity named `mux_16_to_1` that represents a 16 to 1 multiplexor. (b) (11 points) Create a VHDL architecture representing a structural model of the 16 to 1 mux using as many `mux_4_to_1` muxes as are needed. You do not need to write an entity or an architecture for `mux_4_to_1`. You may also assume that a component has already been declared and that no configuration statement is required.

```vhdl
entity MUX_16_TO_1 is
    port (I : in std_logic_vector(15 downto 0);
          SEL : in std_logic_vector(3 downto 0);
          O : out std_logic);
end MUX_16_TO_1;

architecture STRUCTURAL of MUX_16_TO_1 is
    component MUX_4_TO_1 is
        port (I : in std_logic_vector(3 downto 0);
              SEL : in std_logic_vector(1 downto 0);
              O : out std_logic);
    end component;
    signal INTERNAL : std_logic_vector(3 downto 0);
    begin
        U1 : MUX_4_TO_1
            port map(I => I(15 downto 12), SEL => SEL(1 downto 0),
                     O => INTERNAL(3));
        U2 : MUX_4_TO_1
            port map(I => I(11 downto 8), SEL => SEL(1 downto 0),
                     O => INTERNAL(2));
        U3 : MUX_4_TO_1
            port map(I => I(7 downto 4), SEL => SEL(1 downto 0),
                     O => INTERNAL(1));
        U4 : MUX_4_TO_1
            port map(I => I(3 downto 0), SEL => SEL(1 downto 0),
                     O => INTERNAL(0));
        U5 : MUX_4_TO_1
            port map(I(3) => INTERNAL(3), I(2) => INTERNAL(2),
                     I(1) => INTERNAL(1), I(0) => INTERNAL(0),
                     SEL => SEL(3 downto 2), O => O);
    end STRUCTURAL;
```

2. (1 point) The synthesizable subset of VHDL is standard. (True/False) __False__

3. (20 points) (a) (12 points) Write a VHDL function that will take two integer vectors, A and B, and find the dot product \( C = \sum a_i \cdot b_i \). The function call should be of the form `DOT(A,B)`, where A and B are integer vector signals. Make no assumptions about the high and low values of the ranges. For example, 
   
   \[
   A(3 \text{ downto } 1) = (1,2,3), B(3 \text{ downto } 1) = (4,5,6), C = 3*6 + 2*5 + 1*4 = 32.
   \]
   
   \[
   A(0 \text{ to } 4) = (1,3,5,7,9), B(9 \text{ downto } 5) = (2,4,6,8,10) = 1*2 + 3*4 + 5*6 + 7*8 + 9*10 = 190
   \]
   
   Output a warning if the ranges are not the same.

   (b) (8 points) Show an architecture that includes two calls to the function with the following properties. 1 - returns a value, 2 – triggers a warning message.
package MINE is
    type INTEGER_VECTOR is array (NATURAL RANGE <>) of integer;
    function DOT_PRODUCT (L, R : INTEGER_VECTOR) return integer;
end MINE;

package BODY MINE is
    function DOT_PRODUCT (L, R : INTEGER_VECTOR) return integer is
        variable TEMP : integer := 0;
        alias new_l : INTEGER_VECTOR(L'LENGTH -1 downto 0) is L;
        alias new_r : INTEGER_VECTOR(L'LENGTH -1 downto 0) is R;
    begin
        assert (L'LENGTH = R'LENGTH)
        report "Ranges of operands are not the same"
        severity WARNING;
        for I in new_l'RANGE loop
            temp := temp + new_l(I)*new_r(I);
        end loop;
        return TEMP;
    end DOT_PRODUCT;
end MINE;

entity DOT_PRODUCT_CALL is
end DOT_PRODUCT_CALL;

architecture CALL of DOT_PRODUCT_CALL is
    signal A : INTEGER_VECTOR(2 to 8) := (1,3,5,6,4,2,0);
    signal B : INTEGER_VECTOR(24 downto 18) := (1,2,3,4,5,6,7);
    signal C : INTEGER_VECTOR(52 to 55) := (1,3,5,6);
    signal D : integer;
    signal E : integer;
begin
    D <= DOT_PRODUCT(A, B);
    E <= DOT_PRODUCT(A, C);
end CALL;

4. (1 point) All processes are executed at initialization. (True/False) ___True___

5. (1 point) A ___procedure___ is used when you have multiple return values.

6. (4 points) Translate the following statement to an if-then-else statement:

```plaintext
transmit <= signal_a when state = idle else
    signal_b when state = incoming else
    signal_c when state = outgoing else
    signal_d;
```

if (state = idle) transmit <= signal_a;
elsif (state = incoming) transmit <= signal_b;
elsif (state = outgoing) transmit <= signal_c;
else transmit <= signal_d;

7. (1 point) For every process, there is an equivalent concurrent statement. (True/False) _False_

8. (4 points) (a) (2 points) Specify a CLASSIFICATION enumeration data type that spells out the various classifications for undergraduate students.(b) (2 points) Write a variable declaration MY_CLASS that has a value equal to the rightmost element of the type.
type CLASSIFICATION is (FRESHMAN, SOPHOMORE, JUNIOR, SENIOR);
variable MY_CLASS : CLASSIFICATION := SENIOR;

9. (1 point) Multiple architectures can exist for a single entity. (True/False) _True_

10. (1 point) Multiple Choice: _a_ is the default delay in VHDL. (a) Inertial (b) Transport

11. (6 points) (a) (4 points) Write a declaration of an array that can be used to hold the student numbers of the students in this class. (b) (2 points) Initialize the first element of this array with your student number.

   type CLASS_ANUMBERS is array (0 to 15) of string(1 to 9);
   variable CPE_426_526 : CLASS_ANUMBERS (0 => A12345678, others => A00000000);

12. (20 points) Given the following VHDL, indicate all transactions and events. Give the values of A, B, C, D, E, and F each time a change occurs. Carry this out until no further change occurs.

   entity prob is
     port (D : inout bit);
   end prob;

   architecture PROB of PROB is
     signal A, B, C, E, F : bit;
     begin
       process
         A <= '1' after 5 ns,
             '0' after 12 ns;
       wait;
     end process;
     P1: process (A, C)
     begin
       B <= A after 2 ns;
       E <= C after 7 ns;
     end process P1;
     C <= transport A and B after 6 ns;
     P2: process (C, E)
     begin
       F <= C or E after 4 ns;
     end process P2;
     D <= A or B or C or F after 1 ns;
   end PROB;

<table>
<thead>
<tr>
<th>Time</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ns</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5 ns</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6 ns</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7 ns</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12 ns</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13 ns</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14 ns</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17 ns</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>18 ns</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>22 ns</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23 ns</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

   Time | Event | Processes Triggered | Scheduled Transactions | Event? |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5 ns</td>
<td>A 0→1</td>
<td>P1</td>
<td>B ‘1’ 7 ns</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E ‘0’ 12 ns</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C ‘0’ 11 ns</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D ‘1’ 6 ns</td>
<td>Y</td>
</tr>
<tr>
<td>6 ns</td>
<td>D 0→1</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 ns</td>
<td>B 0→1</td>
<td>C</td>
<td>C ‘1’ 13 ns</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D</td>
<td>D ‘1’ 8 ns</td>
<td>N</td>
</tr>
<tr>
<td>12 ns</td>
<td>A 1→0</td>
<td>P1</td>
<td>B ‘0’ 14 ns</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>E ‘0’ 19 ns</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C ‘0’ 18 ns</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>D ‘1’ 13 ns</td>
<td>N</td>
</tr>
<tr>
<td>13 ns</td>
<td>C 0→1</td>
<td>P1</td>
<td>B ‘0’ 15 ns</td>
<td>N</td>
</tr>
</tbody>
</table>

   Time | Event | Processes Triggered | Scheduled Transactions | Event? |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15 ns</td>
<td>C 0→1</td>
<td>P1</td>
<td>B ‘0’ 15 ns</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(already have B ‘0’ 14 ns)</td>
<td></td>
</tr>
</tbody>
</table>
13. (15 points) Design a priority encoder that is described by the following truth table. (d is for don’t care)

(a) (3 points) Write a VHDL entity.
(b) (6 points) Use concurrent signal assignments to implement the architecture.
(c) (6 points) Use sequential statements to implement the architecture. Include any necessary library references.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0 D1 D2 D3</td>
<td>X Y V</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>Z Z 0</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>d 1 0 0</td>
<td>0 1 1</td>
</tr>
<tr>
<td>d d 1 0</td>
<td>1 0 1</td>
</tr>
<tr>
<td>d d d 1</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

entity P_ENCODE is
  port (D : in std_logic_vector(3 downto 0);
        X, Y, V : out std_logic);
end P_ENCODE;

architecture CONCURRENT of P_ENCODE is
begin
  V <= '1' when (D(3) = '1') else '1' when (D(2) = '1') else '1' when (D(1) = '1') else '1' when (D(0) = '1') else '0' when (D = "0000") else 'Z';
  X <= '1' when (D(3) = '1') else '1' when (D(2) = '1') else '0' when (D(1) = '1') else '0' when (D(0) = '1') else 'Z';
  Y <= '1' when (D(3) = '1') else '0' when (D(2) = '1') else '0' when (D(1) = '1') else '0' when (D(0) = '1') else 'Z';
end CONCURRENT;

architecture SEQUENTIAL of P_ENCODE is
begin
  process(D)
  begin
    variable TEMP : std_logic_vector(2 downto 0);
    begin
      if (D(3) = '1') then TEMP := "111";
      elsif (D(2) = '1') then TEMP := "101";
      elsif (D(1) = '1') then TEMP := "011";
      elsif (D(0) = '1') then TEMP := "001";
      elsif (D = "0000") then TEMP := "ZZ0";
      else TEMP := "ZZZ";
    end if;
    X <= TEMP(2);
    Y <= TEMP(1);
    V <= TEMP(0);
  end process;
end SEQUENTIAL;
14. (10 points) Draw the state diagram for the following state machine. Is it a Moore machine or a Mealy

machine? Moore

ENTITY state_machine IS
PORT (sig_in : IN BIT; clk, rst : 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 <= c;
CASE current_state
WHEN a =>
   IF sig_in = '0' THEN
       next_state <= a;
       sig_out <= '1';
   ELSE
       next_state <= d;
       sig_out <= '1';
   END IF;
WHEN b =>
   IF sig_in = '0' THEN
       next_state <= b;
   ELSE
       next_state <= c;
   END IF;
WHEN c =>
   IF sig_in = '1' THEN
       sig_out <= '1';
       next_state <= a;
   ELSE
       next_state <= b;
   END IF;
WHEN d =>
   IF sig_in = '0' THEN
       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 (rst = '0') then
      current_state <= a;
   ELSIF (clk'EVENT AND clk = '1') THEN
      current_state <= next_state;
   END IF;
END PROCESS;
END state_machine;