The University of Alabama in Huntsville
ECE Department
CPE 526 01
Midterm Exam
March 2, 2017

Name: ________________________________

1. (18 points) (12 points) Write a VHDL function that has inputs **`vect`** of type **`std_logic_vector`**, **`begin`** of type natural and **`end`** of type natural. You may assume that **`vect`** has the form **`std_logic_vector(vect'length-1 downto 0)`**. The function extracts **`sub`** of the form **`std_logic_vector(begin downto end)`** from **`vect`**. Output an error if **`begin`** is less than **`end`**. Also output an error if begin – end is greater than vect'length – 1. (b)(6 points) Show an architecture that includes three calls to the function with the following properties. 1 - returns a value, 2 – triggers first error message 3 – triggers second error message.

2. (7 points) Modify the following VHDL model by adding a parameter that sets the number of flip-flops in the counter. Also, add an input which is loaded with an asynchronous load input signal which is active low.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity UPCOUNT is
  port ( CLOCK, RESETN, E : in std_logic;
         Q                : out std_logic_vector (3 downto 0));
end UPCOUNT;

architecture BEHAVIOR of UPCOUNT is
  signal COUNT : std_logic_vector (3 downto 0);
begin
  process (CLOCK, RESETN)
  begin
    if RESETN = ‘0’ then
      COUNT <= “0000”;
    elsif (CLOCK’event and CLOCK = ‘1’) then
      if E = ‘1’ then
        COUNT <= COUNT + 1;
      else
         COUNT <= COUNT;
      end if;
  end process
  Q <= COUNT;

end BEHAVIOR;
```

3. (1 point) _______________ is an example of a VHDL attribute.

4. (1 point)_______Multiple Choice: Which of the following cannot occur outside a process?

   (a) Signal Assignment (b) Variable Declaration (c) Signal Declaration

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

6. (1 point) A(n) ______________ occurs when a signal changes value.

7. (1 point).__________________ is an example of a built-in enumerated type.

8. (14 points) Design a circuit using VHDL that resolves priority among eight active-low inputs **i(0)** – **i(7)** where **i(0)** has the highest priority. The circuit must produce active-high address outputs **a(2)**–**a(0)** to indicate the number of the highest-priority asserted input. If at least one input is asserted, then an active-high **avalid** output should be asserted. If multiple outputs are asserted, an active-high output **mul** should be asserted.

Use the following entity:

```
entity priority is
  port (i : in std_logic_vector (0 to 7);
        avalid, mul : out std_logic;
        a : out std_logic_vector (2 downto 0));
end entity priority;
```

9. (13 points) Construct a 5 to 32 decoder with 3 to 8 decoders with enable. If necessary, configure a 3 to 8 decoder to represent any additional logic needed. The entities for the 3 to 8 and 5 to 32 decoders are as follows.

```
entity dec3to8 is
  port (x : in std_logic_vector (2 downto 0);
        en : std_logic;
        y : out std_logic_vector (7 downto 0));
end entity dec3to8;

architecture behave of dec3to8 is
   ...
end behave;

entity dec5to32 is
  port (x : in std_logic_vector (4 downto 0);
        en : std_logic;
        y : out std_logic_vector (31 downto 0));
end entity dec3to8;
```

10. (18 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 : out bit);
end prob;

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

Time	A	B	C	D	E	F
0 ns	0	0	0	0	0	0
5 ns	1	0	0	0	0	0

Time	Event	Processes Triggered	Scheduled Transactions	Event?

Scheduling Rules	Transport	Inertial
New before existing	Overwrite existing	Overwrite existing
New after existing	Append new	If $v_{new} = v_{existing}$, append new Elsif t_{new}-$t_{existing}$ > reject append new Else overwrite existing

11. (15 points) Draw the state diagram for the following state machine. You may omit reset from all the arcs.

```
entity STATE_MACHINE is
  port (CLK, RESET, START, FULL, EMPTY, ZERO: in std_logic;
        HOT, COLD, DRAIN, TURN : out std_logic);
end STATE_MACHINE;

architecture SYNTH of STATE_MACHINE is
  type STATE_TYPE is (IDLE, FILL_1, WASH, DRAIN_1, SPIN_1,
                      FILL_2, RINSE, DRAIN_2, SPIN_2);
  signal STATE : STATE_TYPE;
begin
  process(CLK, RESET, START, FULL, EMPTY)
  begin
    if (RESET = '1') then
      STATE <= IDLE;
    elsif (CLK'event and CLK = '1') then
      HOT <= '0'; COLD <= '0'; DRAIN <= '0'; TURN <= '0';;
      case STATE is
        when IDLE => if (START = '1') then
                       STATE <= FILL_1;
                     else
                       STATE <= IDLE;
                     end if;
        when FILL_1 => if (FULL = '1') then
                         STATE <= WASH; TURN <= '1';
                       else
                         STATE<= FILL_1; HOT <= '1';
                       end if;
        when WASH => if (ZERO = '1') then
                       STATE<= DRAIN_1;
                     else
                       TURN <= '1'; STATE<= WASH;
                     end if;
        when DRAIN_1 => if (EMPTY = '1') then
                          TURN <= '1'; STATE<= SPIN_1;
                        else
                          DRAIN <= '1'; STATE <= DRAIN_1;
                        end if;
        when SPIN_1 => if (ZERO = '1') then
                         STATE <= FILL_2;
                       else
                         STATE <= SPIN_1; DRAIN <= '1'; TURN <= '1';
                       end if;
        when FILL_2 => if (FULL = '1') then
                         TURN <= '1'; STATE <= RINSE;
                       else
                         STATE <= FILL_2; COLD <= '1';
                       end if;
        when RINSE => if (ZERO = '1') then
                         STATE <= DRAIN_2;
                      else
                        TURN <= '1'; STATE <= RINSE;
                      end if;
        when DRAIN_2 => if (EMPTY = '1') then
                          TURN <= '1'; STATE <= SPIN_2;
                        else
                          STATE<= DRAIN_2; DRAIN <= '1';
                        end if;
        when SPIN_2 => if (ZERO = '1') then
                         STATE <= IDLE;
                       else
                         STATE <= SPIN_2; TURN <= '1'; DRAIN <= '1';
                       end if;
      end case;
    end if;
  end process;
end SYNTH;
```

12. (10 points) Write a VHDL entity (3 points) and architecture (7 points) of a synthesizable T flip-flop with synchronous reset that operates on the falling edge of the clock.