The University of Alabama in Huntsville
ECE Department
CPE 426 01
Midterm Exam Solution
Spring 2014

1. (20 points) Write a function, weaken, that maps a standard-logic value to the same value, but with weak drive strength. Thus, '0' and 'L' are mapped to 'L', '1' and 'H' are mapped to 'H', 'X' and 'W' are mapped to 'W' and all other values are unchanged.

```
library ieee;
use ieee.std_logic_1164.all;

package MINE is
  function WEAKEN (I : std_logic) return std_logic;
end package MINE;

package body MINE is
  function WEAKEN (I : std_logic) return std_logic is
    variable TEMP : std_logic;
  begin
    case I is
      when '0' | 'L' => TEMP := 'L';
      when '1' | 'H' => TEMP := 'H';
      when 'X' | 'W' => TEMP := 'W';
      when others => TEMP := I;
    end case;
    return TEMP;
   end WEAKEN;
end MINE;
```

2. (10 points) Write the equivalent process for the conditional signal assignment statement

```
with bit_vector'(s, r) select
  Q <= unaffected when "00",
       '0' when "01",
       '1' when "10" | "11";

entity SR is
  port (S, R : in bit;
        Q : out bit);
end SR;

architecture SEQUENTIAL of SR is
begin
  process (S, R)
  begin
    case bit_vector'(S, R) is
      when "01" => Q <= '0';
      when "10" | "11" => Q <= '1';
      when others => null;
    end case;
  end process;
end SEQUENTIAL;
```

3. (1 point) _**Inertial**_ delay is the delay which represents gate delay in VHDL

4. (1 point) A process is triggered whenever an event occurs on a signal that is in the _**sensitivity list**_ of the process.

5. (1 point) In order to specify edge behavior the _**'STABLE**_ attribute is used in concurrent statements.

6. (1 point) _**False**_(True or False) A D flip-flop and a D latch have the same behavior.

7. (1 point) _**True**_ (True or False) It is possible to make aggregate assignments in VHDL.

8. (15 points) A 4-bit magnitude comparator chip compares two unsigned 4-bit numbers A and B and produces outputs to indicate whether A < B, A = B, or A > B. There are three output signals to indicate each of the above conditions. Note that exactly one of the output lines will be high and the other two lines will be low at any time. Write a behavioral VHDL model for the 4-bit comparator.

```
entity COMPARE is
  port (A : in bit_vector (3 downto 0);
        B : in bit_vector (3 downto 0);
        LT, EQ, GT : out bit);
end entity COMPARE;

architecture BEHAV2 of COMPARE is
begin
  process (A, B)
  begin
    LT <= '0';
    GT <= '0';
    EQ <= '0';
    if (A = B) then
      EQ <= '1';
    elsif (A < B) then
      LT <= '1';
    else
      GT <= '1';
    end if;
  end process;
end BEHAV2;

architecture BEHAV3 of COMPARE is
begin
    GT <= '1' when A > B else '0';
    LT <= '1' when A < B else '0';
    EQ <= '1' when A = B else '0';
end BEHAV3;
```

9. (20 points) Develop a VHDL model of a 14-bit counter with parallel load inputs using instances of the 4-bit counter whose entity is given. Ensure that any unused inputs are properly connected to a constant driving value.

```
entity COUNTER is
  port (CLK_N, LOAD_EN, RESET : in std_ulogic;
        D : in std_ulogic_vector (3 downto 0);
        Q : out std_ulogic_vector (3 downto 0));
end entity COUNTER;

library ieee;
use ieee.std_logic_1164.all;
use WORK.all;

entity COUNTER_14 is
  port (D : in std_ulogic_vector (13 downto 0);
        CLK_N : in std_ulogic;
        LOAD_EN : in std_ulogic;
        RESET : in std_ulogic;
        Q : out std_ulogic_vector (13 downto 0));
end COUNTER_14;

architecture STRUCT of COUNTER_14 is
  signal TEMP : std_ulogic_vector (13 downto 0);
  signal LOAD : std_ulogic;
  signal LOAD_DATA : std_ulogic_vector (13 downto 0);
  signal USELESS : std_ulogic_vector (1 downto 0);
begin
  C3 : entity COUNTER
         port map (CLK_N => TEMP(11), LOAD_EN => LOAD, RESET => RESET,
                   Q(3 downto 2) => USELESS,
                   Q(1 downto 0) => TEMP(13 downto 12),
                   D(3) => '0', D(2) => '0',
                   D(1 downto 0) => LOAD_DATA(13 downto 12));
  C2 : entity COUNTER
         port map (CLK_N => TEMP(7), LOAD_EN => LOAD, RESET => RESET,
                   Q => TEMP(11 downto 8),
                   D => LOAD_DATA(11 downto 8));
  C1 : entity COUNTER
         port map (CLK_N => TEMP(3), LOAD_EN => LOAD, RESET => RESET,
                   Q => TEMP(7 downto 4),
                   D => LOAD_DATA(7 downto 4));
  C0 : entity COUNTER
         port map (CLK_N => CLK_N, LOAD_EN => LOAD, RESET => RESET,
                   Q => TEMP(3 downto 0),
                   D => LOAD_DATA(3 downto 0));
  LOAD <= '1' when RESET = '1' else
          '1' when TEMP(13 downto 0) = "11111111111111" else
          LOAD_EN;
  LOAD_DATA <= "00000000000000" when RESET = '1' else
        "00000000000000" when TEMP(13 downto 0) = "11111111111111" else
        LOAD_DATA;
  Q <= TEMP;
end STRUCT;
```

10. (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;
  C1: C <= transport A and B after 6
ns;
  P2: process (C, E)
  begin
    F <= reject 3 ns inertial C and E
         after 5 ns;
  end process P2;
  C2: D <= A or B or C or F after 2 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
7 ns	1	1	0	1	0	0
12 ns	0	1	0	1	0	0
13 ns	0	1	1	1	0	0
14 ns	0	0	1	1	0	0
18 ns	0	0	0	1	0	0
20 ns	0	0	0	0	0	0

Time	Event	Processes Triggered	Scheduled Transactions	Event?
5 ns	A → '1'	P1	B ('1', 7 ns)	Y
			E ('0' 12 ns)	N
		C1	C ('0' 11ns)	N
		C2	D ('1', 7 ns)	Y
7 ns	B → '1'	C1	C ('1', 13 ns) appended	Y
		C2	D ('1', 9 ns)	N
	D → '1'	none		
12 ns	A → '0'	P1	B ('0', 14 ns)	Y
			~~E ('0', 19 ns)~~ overwritten by E ('1', 20ns)	N
		C1	C ('0', 18 ns) appended	Y
		C2	D ('1', 14 ns)	N
13 ns	C → '1'	P1	B ('0', 15 ns)	N
			~~E ('1', 20 ns)~~ overwritten by E ('0', 25 ns)	Y
		P2	F ('0', 18 ns)	N
		C2	D ('1', 15 ns)	N
14 ns	B → '0'	C1	C ('0', 20ns) appended	N
		C2	D ('1', 16 ns)	N
18 ns	C → '0'	P1	B ('0', 20 ns)	N
			E ('0', 25 ns)	N
		P2	F ('0', 23 ns) appended	N
		C2	D ('0', 20 ns) appended	Y
20 ns	D → '0'	none		

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. (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, 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 <= C;
          SIG_OUT <= '1';
        else
          NEXT_STATE <= D;
        end if;
      when B =>
        if SIG_IN = '0' then
          NEXT_STATE <= B;
        else
          NEXT_STATE <= C;
        end if;
          SIG_OUT <= '1';
      when C =>
        if SIG_IN = '1' then
          SIG_OUT <= '1';
          NEXT_STATE <= A;
        else
          NEXT_STATE <= B;
        end if;
          SIG_OUT <= '1';
      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;
```