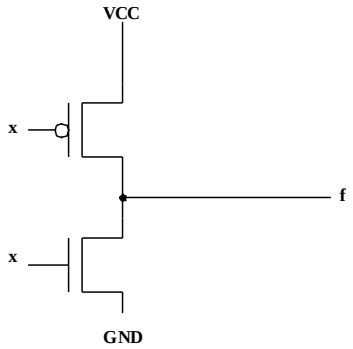
The University of Alabama in Huntsville
ECE Department
CPE 426 01
Final Exam Solution
Spring 2009

1. (5 points) Draw the transistor-level diagram of a CMOS inverter.



2. (5 points) Consider the following structural VHDL model.

```
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 => ?, C2 => ?, C3 => ?);
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?

(a) U1 : UNIT port map (C1 => P1, C2 => P3, C3 => P2);
(b) Yes, U1 : UNIT port map (C1 => P3, C2 => P1, C3 =>);

3. (15 points) For the following VHDL architecture, give the values of A, B, C, D, E, and F each time a change occurs. Carry this out until no further change occurs. I

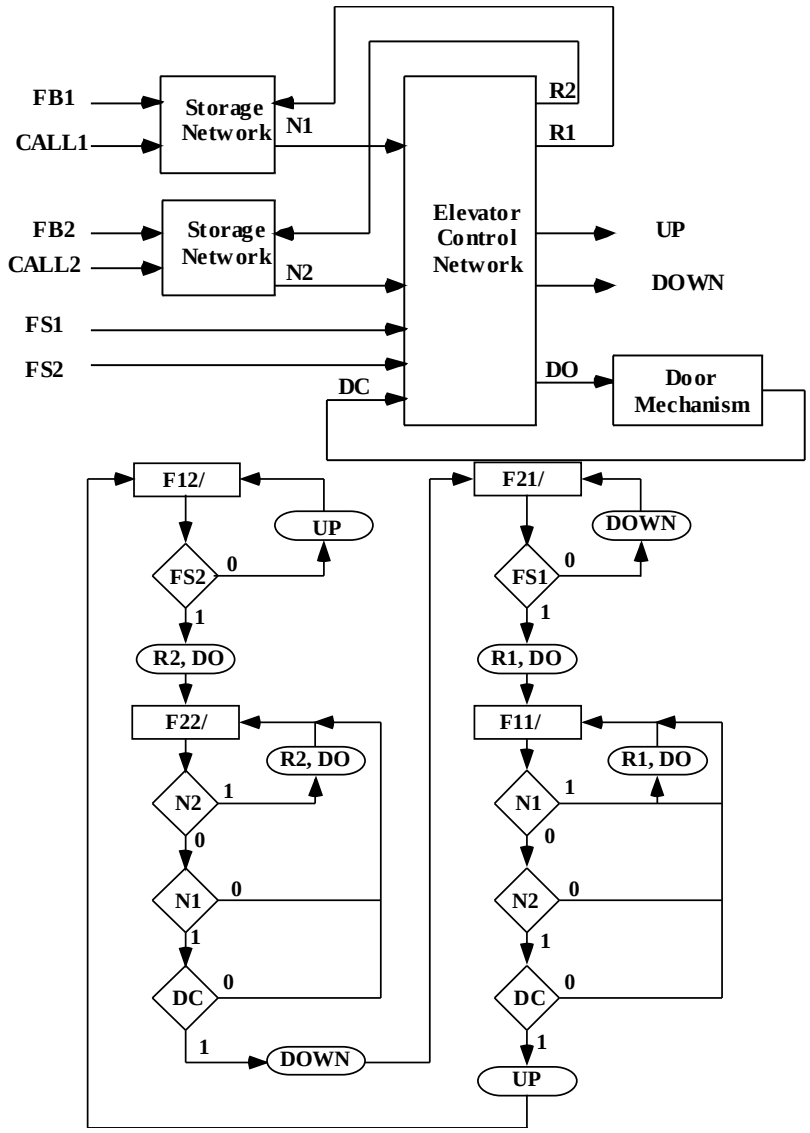
```
entity prob is
  port (D : inout bit);
end prob;

architecture PROB of PROB is
  signal A, B, C, E, F : bit;
begin
  P1: process (A, C)
  begin
    B <= A after 3 ns;
    E <= C after 5 ns;
  end process P1;
  C <= A after 10 ns;
  P2: process (C, E)
  begin
    F <= C and E after 4 ns;
  end process P2;
  D <= A or B or C or F after 1 ns;
  process
    A <= '1' after 5 ns;
    wait;
  end process'
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
6 ns	1	0	0	1	0	0
8 ns	1	1	0	1	0	0
9 ns	1	1	0	0	0	0
11 ns	1	0	0	0	0	0
18 ns	1	0	0	0	1	0
22 ns	1	0	0	0	1	1

Time	Event	Processes Triggered	Scheduled Transactions	Event?
5 ns	A→1	C1	C '1' 11 ns	Y
		C2	D '1' 6 ns	Y
6 ns	D→1	P1	B '1' 8 ns	Y
		P1	E '0' 13 ns	N
8 ns	B→1	C2	D '0' 9 ns	Y
9 ns	D→0	P1	B '0' 11 ns	Y
		P1	E '0' 16 ns	N
11 ns	B→0	C2	D '0' 12 ns	N
	C→1	P1	B '0' 13 ns	N
		P1	E '1' 18 ns	Y
		P2	F '0' 15 ns	N
18 ns	E→1	C1	C '1' 24 ns	N
		P2	F '1' 22 ns	Y
22 ns	F→1	none		

4. (15 points) The block diagram for an elevator controller for a building with two floors is shown below. The inputs FB1 and FB2 are floor buttons in the elevator. The inputs CALL1 and CALL2 are call buttons in the hall. The inputs FS1 and FS2 are floor switches that output a 1 when the elevator is at the first or second floor landing. Outputs UP and DOWN control the motor, and the elevator is stopped when UP = DOWN = 0. N1 and N2 are flip-flops that indicate when the elevator is needed on the first or second floor. R1 and R2 are signals that reset these flip-flops. DO = 1 causes the door to open, and DC = 1 indicates that the door is closed. (a) Write a VHDL entity for the controller. (b) Write a VHDL architecture that models the functionality of the controller.




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

entity ELEVATOR_CONTROL is
  port(FB1, CALL1, FB2, CALL2, FS1, FS2, DC : in std_logic;
       N1, N2 , CLK: in std_logic;
       R1, R2, UP, DOWN, DO : out std_logic);
end ELEVATOR_CONTROL;
```

```
architecture BEHAV of ELEVATOR_CONTROL is
  type STATE_TYPE is (F12, F22, F21, F11);
  signal CURRENT_STATE, NEXT_STATE : STATE_TYPE;
begin
  process (FB1, CALL1, FB2, CALL2, FS1, FS2, DC,
           N1, N2, CURRENT_STATE)
  begin
    if (CURRENT_STATE = F12) then
      if (FS2 = '0') then
        UP <= '1';
        NEXT_STATE <= F12;
      else
        R2 <= '1';
        DO <= '1';
        NEXT_STATE <= F22;
      end if;
    elsif (CURRENT_STATE = F22) then
      NEXT_STATE <= F22;
      if (N2 = '1') then
        R2 <= '1';
        DO <= '1';
      elsif (N1 = '1') then
        if (DC = '1') then
          DOWN <= '1';
          NEXT_STATE <= F21;
        end if;
      end if;
    elsif (CURRENT_STATE = F21) then
      if (FS1 = '1') then
        R1 <= '1';
        DO <= '1';
        NEXT_STATE <= F11;
      else
        DOWN <= '1';
        NEXT_STATE <= F21;
      end if;
    elsif (CURRENT_STATE = F11) then
      NEXT_STATE <= F11;
        if (N1 = '1') then
          R1 <= '1';
          DO <= '1';
        elsif (N2 = '1') then
          if (DC = '1') then
            UP <= '1';
            NEXT_STATE <= F12;
          end if;
        end if;
    end if;
  end process;
  process (CLK)
  begin
    if (CLK = '1' and CLK'event) then
      CURRENT_STATE <= NEXT_STATE;
    end if;
  end process;
end BEHAV;
```

5. (5 points) If the NRE costs for FPGA and ASIC circuits are $30,000 and $750,000, respectively, and the cost of individual parts for FPGA and ASIC circuits are $30 and $5, respectively, what is the break-even manufacturing volume for these two types of circuits?

x – number of units **21000 + 15x = 187000 + 7x, 8x = 166000, x = 20750**

6. (10 points) Write a VHDL procedure that will add two n x m matrices of integers, `C <= A + B`. The procedure call should be of the form `addM (A, B, C)`. The procedure should report an error if the number of rows in A and B are not the same or if the columns in A and B are not the same. Make no assumptions about the high and low values or direction of the ranges for either dimension.

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

package MINE is
  type INTEGER_MATRIX is array (NATURAL RANGE <>, NATURAL RANGE <>) of integer;
  procedure addM (L, R : in INTEGER_MATRIX; RESULT : out INTEGER_MATRIX);
end MINE;
package BODY MINE is
  procedure addM (L, R : in INTEGER_MATRIX; RESULT : out INTEGER_MATRIX) is
    variable TEMP : INTEGER_MATRIX(L'RANGE(1), L'RANGE(2));
  begin
    if (L'LENGTH(1) /= R'LENGTH(1) or L'LENGTH(2) /= R'LENGTH(2)) then
      report "Ranges of operands are not the same"
      severity WARNING;
      return;
    end if;
    for I in  L'RANGE(1) loop
      for J in L'RANGE(2) loop
        TEMP(I, J) := L(I, J) + R(I, J);
      end loop;
    end loop;
    return;
  end addM;
end MINE;
```

7. (10 points) Rewrite the following VHDL to use processes instead of blocks. The processes should have the exact same behavior as the blocks.

```
use WORK.TSL.all,WORK.SYS.all;

entity I8212 is
 generic (GDEL,FFDEL,BUFDEL: TIME);
 port (DI: in WORD;
       DO: out WORD;
       NDS1,DS2,MD,STB,NCLR: in BIT;
       NINT: out BIT := '1');
end I8212;
```

```
architecture BEHAVIOR of I8212 is
begin
I8212_BLK: block

signal S0,S1,S2,S3,S4: BIT;
signal SRQ: BIT;
signal Q: WORD;
begin

INT_BLK:
block (S1='1' and S4='1')
begin
 Q <= guarded DI after FFDEL;
 Q <="00000000" after FFDEL when (S1='0' and S4='0') else
      Q;
 DO <= Q after BUFDEL when (S3 = '1') else
      "ZZZZZZZZ" after BUFDEL;
end block INT_BLK;

S0 <= not NDS1 and DS2 after GDEL;
S1 <= (S0 and MD) or (STB and not MD) after (2*GDEL);
S2 <= (S0 nor (not S4)) after GDEL;
S3 <= (S0 or MD) after GDEL;
S4 <= (S1 OR NCLR) after GDEL;
SRQ <= '1' after FFDEL when (S2= '0') else
       '0' after FFDEL when (S2= '1') and (not STB'STABLE) and (STB='0') else
        SRQ;
NINT <= not SRQ nor S0 after GDEL;
end block I8212_BLK;
end BEHAVIOR;
```

```
process (S1, S4, DI, S3, Q)
  begin
    if (S1 = '1' and S4 = '1') then
      Q <= DI after FFDEL;
    elsif (S1 = '0' and S4 = '0') then
      Q <="00000000" after FFDEL;
    end if;
    if (S3 = '1') then
      DO <= Q after BUFDEL;
    else
      DO <= "ZZZZZZZZ" after BUFDEL;
    end if;
  end process;
```

8. (10 points) Create a VHDL entity named `en_dec_328` that represents a 3-to-8 decoder with an active-low enable input which has an architecture which uses a case statement to represent the functionality of the decoder. Create a second entity and its accompanying architecture that represents a 4-to-16 decoder by using two instances of the `en_dec_328` entity.

```
entity EN_DEC_3TO8 is
  port (EN : in std_logic;
        I : in std_logic_vector(2 downto 0);
        O : out std_logic_vector(7 downto 0));
end EN_DEC_3TO8;
```

```
architecture BEHAV of EN_DEC_3TO8 is
begin
  process(EN, I)
  begin
    case EN is
      when '0' =>
        case I is
          when "000" => O <= "00000001";
          when "001" => O <= "00000010";
          when "010" => O <= "00000100";
          when "011" => O <= "00001000";
          when "100" => O <= "00010000";
          when "101" => O <= "00100000";
          when "110" => O <= "01000000";
          when "111" => O <= "10000000";
          when others => O <= "00000000";
        end case;
      when others =>
        O <= "00000000";
    end case;
  end process;
end BEHAV;

entity DEC_4TO16 is
  port (I : in std_logic_vector(3 downto 0);
        O : out std_logic_vector(15 downto 0));
end DEC_4TO16;

architecture STRUCT of DEC_4TO16 is
  signal I3BAR : std_logic;
  component EN_DEC_3TO8C
    port (EN : in std_logic;
          I : in std_logic_vector(2 downto 0);
          O : out std_logic_vector(7 downto 0));
  end component;
  for all : EN_DEC_3TO8C use entity work.EN_DEC_3TO8(BEHAV);
begin
  I3BAR <= not I(3);
  U1 : EN_DEC_3TO8C
         port map (I(3), I(2 downto 0), O(7 downto 0));
  U2 : EN_DEC_3TO8C
         port map(I3BAR, I(2 downto 0), O(15 downto 8));
end;
```

9. (8 points) (a) (5 points) Write a single VHDL model which represents an AND gate with an arbitrary number of inputs, N. (b) (3 points) Use that model as a component in an entity that represents a four input AND gate with inputs a, b, c, d and output f

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

entity GENERIC_AND is
  generic (N : integer := 2);
  port (I : in std_logic_vector(N-1 downto 0);
        O : out std_logic);
end GENERIC_AND;
```

```
architecture BEHAV of GENERIC_AND is
begin
  process(I)
    variable TEMP : std_logic := '1';
  begin
    for J in I'RANGE loop
      TEMP := TEMP and I(J);
    end loop;
    O <= TEMP;
  end process;
end BEHAV;

library ieee;
use ieee.std_logic_1164.all;

entity AND_4 is
  port (A, B, C, D : in std_logic;
        F : out std_logic);
end AND_4;

architecture STRUCT of AND_4 is
begin
  U1 : entity work.GENERIC_AND
         generic map (4)
         port map (I(3) => A, I(2) => B, I(1)=> C,
                   I(0) => D, O => F);
end STRUCT;
```

10. (5 points) What kind of hardware element will be inferred by a synthesis tool from the following model?

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

entity WIDGET is
  Port (A, B : in SIGNED (0 to 2);
        CLK, RESET : in std_logic;
        Z : out SIGNED(0 to 2));
end WIDGET;

architecture EXAMPLE of WIDGET is
begin
  process (CLK, RESET)
  begin
    if (RESET = '1') then
      Z <= '0';
    elsif (CLK = '1') then
        Z <= A nor B;
    end if;
  end process;
end EXAMPLE;
```

a latch

11. (2 points)__**Functional coverage**__ measures the progress of all tests in fulfilling the verification plan requirements

12. (2 points) An __**interface**__ is a construct that represents a bundle of wires but also has intelligence such as synchronization and functional code. __**Modports**__ allow a module to easily tap a subset of signals from an __interface__(same as first blank) and can be used to check signal direction.

13. (4 points) A constrained random test (CRT) is made of two parts
    a. __**test code**_____
    b. __**seed to the PRNG**__

14. (2 points) __False__(True or False)The SystemVerilog standard specifies the meaning of constraint expressions, the legal values that are created, and the precise order in which the solver should operate.

15. (2 points) Which of the following are true? __**b**__ Multiple choice
    a. Constraints execute from top to bottom.
    b. Constraints are bidirectional, meaning that the constraints on all random variables are solved concurrently.
    c. both a and b
    d. neither a nor b