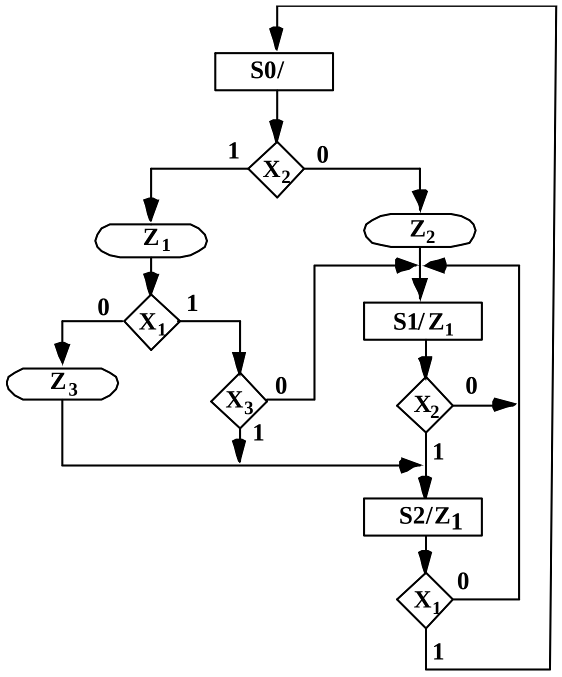
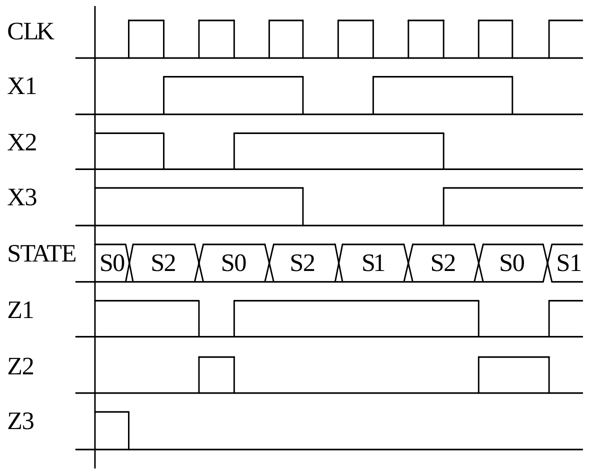
The University of Alabama in Huntsville
Electrical and Computer Engineering
CPE/EE 422/522
Spring 2005
Homework #6 Solution

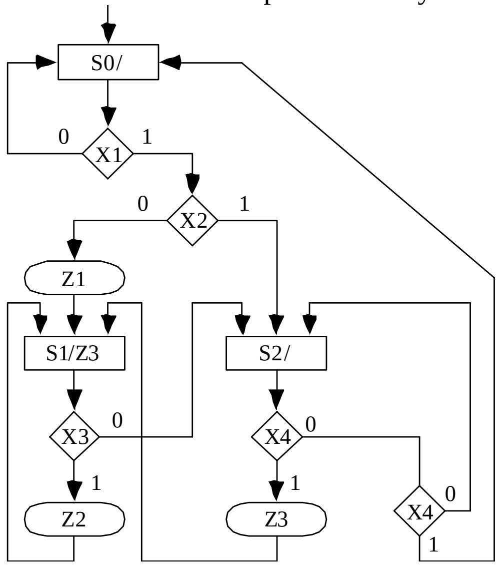
5.3(a)(20), 5.6(c)(20), 5.12(20), 8.2(20), 8.8(20)

5.3 (a) For the following SM chart: Draw a timing chart that shows the clock, the state (S0, S1 or S2), the inputs (X1, X2 and X3) and the outputs. The input sequence is X1 X2 X3 = 011, 101, 111, 010, 110, 101, 001. Assume that all state changes occur on the rising edge of the clock, and the inputs change on the falling edge of the clock.





5.6(c) For the given SM chart: Write a VHDL description of the system.

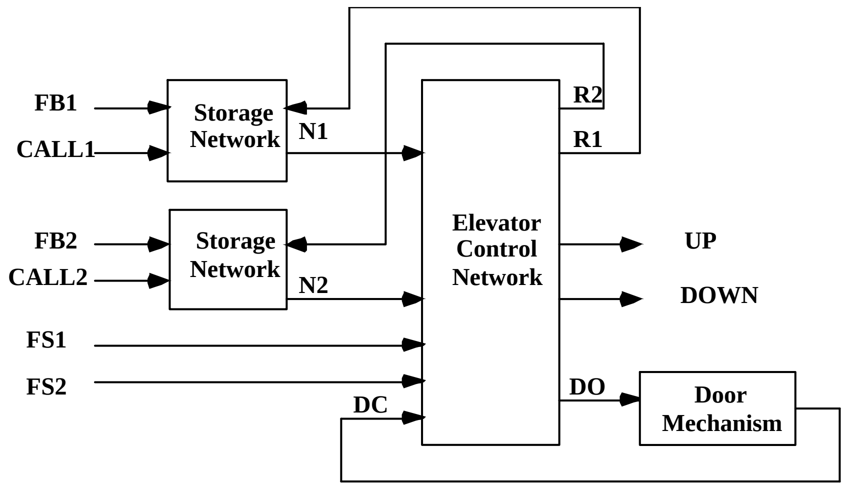


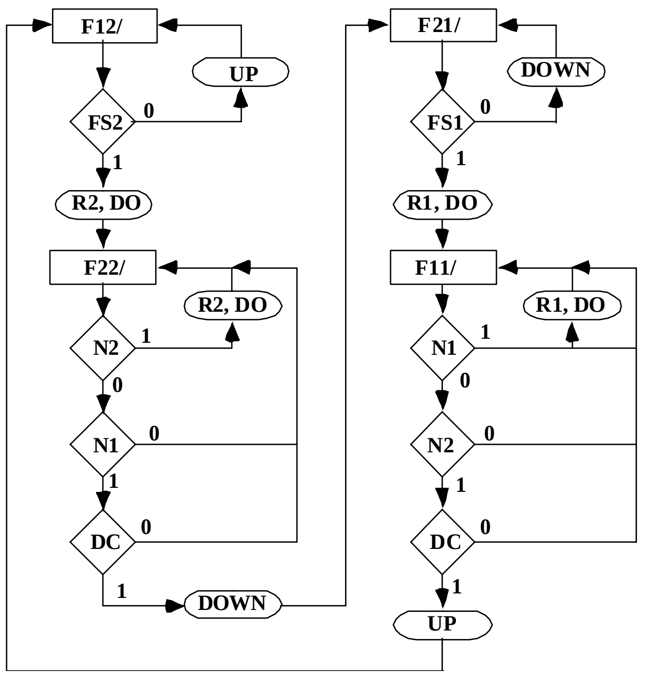
```
library ieee;
use ieee.std_logic_1164.all;

entity SM_CHART5_6 is
  port (X1, X2, X3, X4, X5 : in std_logic;
        Z1, Z2, Z3 : out std_logic;
        CLK, RESET : in std_logic);
end SM_CHART5_6;

architecture BEHAVE of SM_CHART5_6 is
  type STATE_TYPE is (S0, S1, S2);
  signal STATE, NEXT_STATE : STATE_TYPE;
begin
  process(X1, X2, X3, X4, X5, STATE)
  begin
    Z1 <= '0'; Z2 <= '0'; Z3 <= '0';
    case STATE is
      when S0 => if (X1 = '0') then
                   NEXT_STATE <= S0;
                 else
                   if (X2 = '0') then
                     Z1 <= '1';
                     NEXT_STATE <= S1;
                   else
                     NEXT_STATE <= S2;
                   end if;
                 end if;
      when S1 => Z3 <= '1';
                 if (X3 = '0') then
                   NEXT_STATE <= S2;
                 else
                   Z2 <= '1';
                   NEXT_STATE <= S1;
                 end if;
      when S2 => if (X4 = '0') then
                   if (X5 = '0') then
                     NEXT_STATE <= S2;
                   else
                     NEXT_STATE <= S0;
                   end if;
                 else
                   Z3 <= '1';
                   NEXT_STATE <= S1;
                 end if;
    end case;
  end process;
  process(CLK, RESET)
  begin
    if (RESET = '1') then
      STATE <= S0;
    elsif (CLK = '1' and CLK'event) then
      STATE <= NEXT_STATE;
    end if;
  end process;
end BEHAVE;
```

5.12 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. Draw an SM chart for the elevator controller (four states).





8.2 Design an address decoder. One input to the address decoder is an 8-bit address, which can have any range with a length of 8, for example: std_logic_vector addr(8 to 15). The second input is check : std_logic_vector(5 down to 0). The address decoder will output Sel = ‘1’ if the upper 6 bits of the 8-bit address match the 6-bit check vector. For example, if addr = “10001010” and check = “1000--“ then Sel = ‘1’. Only the 6 leftmost bits of addr will be compared; the remaining bits are ignored. An ‘-‘ in the check vector is a don’t care.

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

entity address_decoder is
  port (ADDRESS : in std_logic_vector;
        CHECK : in std_logic_vector(5 downto 0);
        SEL : out std_logic);
end ADDRESS_DECODER;

architecture BEHAV of ADDRESS_DECODER is
  alias ADDR : std_logic_vector(ADDRESS'length-1 downto 0) is ADDRESS;
begin
  process(ADDRESS, CHECK)
   variable MATCH : boolean;
  begin
    MATCH := TRUE;
    for I in ADDRESS'length-1 downto ADDRESS'length-1-5 loop
      if ((ADDR(I) /= CHECK(I-2)) and (CHECK(I-2) /= '-')) then
        MATCH := FALSE;
      end if;
    end loop;
    if (MATCH) then
      SEL <= '1';
    else
      SEL <= '0';
    end if;
  end process;
end BEHAV;
```

8.8 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 vector 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.

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

package MINE is
  function "=" (L, R : std_logic_vector) return boolean;
end MINE;
```

```
package body MINE is
  function "=" (L, R : std_logic_vector) return boolean is
    variable EQUAL : boolean;
    alias LEFT : std_logic_vector(L'length-1 downto 0) is L;
    alias RIGHT : std_logic_vector(L'length-1 downto 0) is R;
  begin
    assert L'length = R'length
    report "Vectors are not the same length"
    severity error;
    EQUAL := TRUE;
    for I in LEFT'range loop
      case LEFT(I) is
        when '0' => case RIGHT(I) is
                      when '0'|'-' => null;
                      when others => EQUAL := FALSE;
                    end case;
        when '1' => case RIGHT(I) is
                      when '1'|'-' => null;
                      when others => EQUAL := FALSE;
                    end case;
        when '-' => case RIGHT(I) is
                      when '0'|'1' => null;
                      when others => EQUAL := FALSE;
                    end case;
        when others => assert FALSE
                       report "First vector has invalid value"
                       severity error;
      end case;
      assert (RIGHT(I)='0' or RIGHT(I)='1' or RIGHT(I)='-')
      report "Second vector has invalid value"
      severity error;
    end loop;
    return EQUAL;
  end "=";
end MINE;
```