The University of Alabama in Huntsville
Electrical and Computer Engineering
CPE/EE 422/522
Spring 2005
Homework #7
Solution

From the book,
10.1 (20 points) (a) & (b) for q s-a-0 and d s-a-1
(a) Determine the necessary inputs to the following network to test for q s-a-0 and d s-a-1.
(b) For this set of inputs, determine which other stuck-at faults can be tested.

q s-a-0 ABCD = 1000 detects q s-a-0 and also detects c s-a-1, d s-a-1, r s-a-0, t s-a-1, v s-a-0, and F s-a-0
ABCD = 1100 detects q s-a-0 and also detects a s-a-0, r s-a-0, t s-a-1, v s-a-0, and F s-a-0
d s-a-1 ABCD = 1000 detects d s-a-1 and also detects q s-a-0, c s-a-1, r s-a-0, t s-a-1, v s-a-0, and F s-a-0
ABCD = 0000 detects d s-a-1 and also detects c s-a-1, p s-a-1, q s-a-0, r s-a-0, s s-a-0, and t s-a-1
ABCD = 0100 detects d s-a-1 and also detects a s-a-1, c s-a-1, p s-a-0, r s-a-1, t s-a-1, u s-a-1, v s-a-1, and F s-a-0

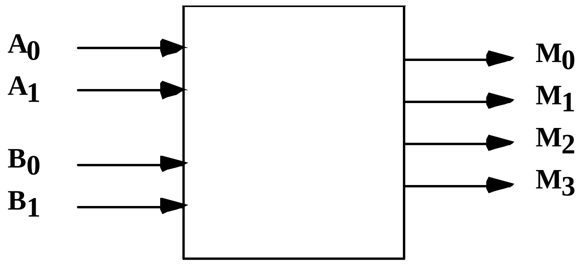
10.2 (20 points) Find a minimum set of tests that will test all single stuck-at-0 and stuck-at-1 faults in the following network. For each test, specify which faults are tested for s-a-1 and s-a-1.
abcdefi = 0001001 a s-a-1, b s-a-1, c s-a-1, g s-a-1
abcdefi = 1000001 d s-a-1, e s-a-1, f s-a-1, h s-a-1
abcdefi = 1001001 g s-a-0, h s-a-0, i s-a-0, Z s-a-0
abcdefi = 0101001 b s-a-0
abcdefi = 0011001 c s-a-0
abcdefi = 1000101 e s-a-0
abcdefi = 1000011 f s-a-0
abcdefi = 1001000 i s-a-1, Z s-a-1

10.7 (10 points) State graphs for two sequential machines are given below. The first graph represents a correctly functioning machine, and the second represents the same machine with a malfunction. Assuming that the two machines can be reset to their starting states (S0 and T0), determine the shortest input sequence that will distinguish the two machines.

Input:	0	0	0	1	1	1	1
Correct Output:	1	1	1	0	0	0	1
Incorrect Output:	1	1	1	0	0	0	0

From other sources

(25 points) Design a hardware multiplier circuit (M) that computes the product of two, positive 2-bit binary numbers.



A_1A_0 represents one 2-bit number, B_1B_0 represents the second 2-bit number and $M_3M_2M_1M_0$ represents the 4-bit product. For example, if A_1A_0 = 10 and $B_2B_1B_0$ = 11, then $M_3M_2M_1M_0$ –0110. Model your circuit by performing the following steps.

a. Develop a VHDL entity declaration for the multiplier.
b. Develop an algorithmic behavioral architectural body for the multiplier.
c. Simulate to verify the correctness of your model.

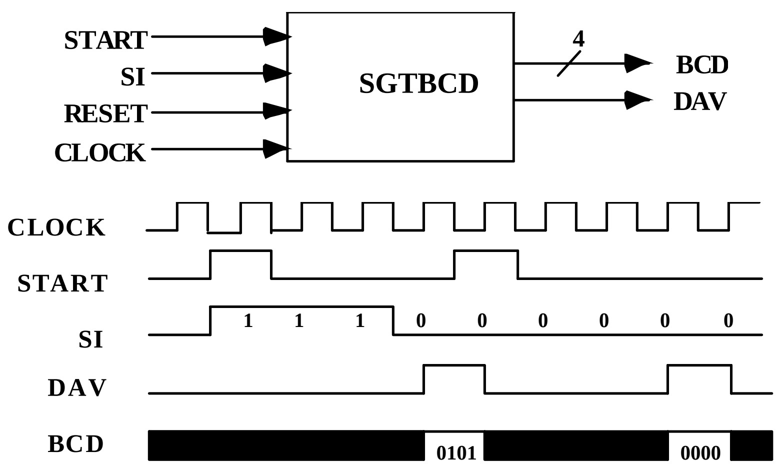
```
entity MULTIPLIER is
  port (A:in bit_vector (1 downto 0);
        B: in bit_vector (1 downto 0);
        M: out bit_vector (3 downto 0));
end MULTIPLIER;

architecture BEHAVE of MULTIPLIER is
begin
  process(A,B)
    variable temp : bit_vector (3 downto 0);
  begin
    temp := A & B;
    case TEMP is
      when "0000"|"0001"|"0010"|"0011"|"0100"|"1000"|"1100"  => M <= "0000";
      when "0101" => M <= "0001";
      when "0110"|"1001" => M <= "0010";
      when "0111"|"1101" => M <= "0011";
      when "1010" => M <= "0100";
      when "1011"|"1110" => M <= "0110";
      when "1111" => M <= "1001";
    end case;
  end process;
end BEHAVE;
```

(25 points) Design a sequential circuit that converts a 4-bit Gray code into a 4-bit BCD code. The inputs and outputs are timed by the same system clock. Assume that the device receives a START pulse coincident with the left most bit of the Gray code. The remaining bits of the Gray code are received one bit at a time (from left to right). At the end of each 4-bit Gray code, the device outputs the corresponding 4-bit BCD code in parallel along with a DAV signal. The next input could start during the clock period following the last input bit or at any time after that. The device must have a RESET input that initializes the device to the correct starting state. The figure given shows a block diagram and sample timing diagram. Use the table below for the Gray and BCD codes.

Decimal Digit	Gray Code	BCD Code
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	1110	0101
6	1010	0110
7	1011	0111
8	1001	1000
9	1000	1001

Model the device using VHDL and do a simulation to verify your model.




```
entity SGRAY2BCD is
  port (R, SI, START, CLK: in BIT;
        BCD: out BIT_VECTOR(3 downto 0);
        DAV: out BIT);
end SGRAY2BCD;

architecture FSM_RTL of SGRAY2BCD is
  type STATE_TYPE is (S0, S1, S2, S3, S4);
  signal STATE: STATE_TYPE;
  signal SHIFT_REG: BIT_VECTOR (3 downto 0);
begin
-- Process to update state at end of each clock period.
```

```
NEXT_STATE: process (R, CLK)
  begin
    if (R = '0') then
       STATE <= S0;
    elsif (CLK='1'and CLK'event) then
      case STATE is
        when S0 =>
            if (START = '1') then
              STATE <= S1;
              SHIFT_REG <= SHIFT_REG(2 downto 0) &SI;
            end if;
        when S1 =>
            SHIFT_REG <= SHIFT_REG(2 downto 0) & SI;
            STATE <= S2;
        when S2 =>
            SHIFT_REG <= SHIFT_REG(2 downto 0) & SI;
            STATE <= S3;
        when S3 =>
            SHIFT_REG <= SHIFT_REG(2 downto 0) & SI;
            STATE <= S4;
        when S4 =>
            if (START = '1') then
              STATE <= S1;
              SHIFT_REG <= SHIFT_REG(2 downto 0) & SI;
            else
              STATE <= S0;
            end if;
      end case;
    end if;
  end process NEXT_STATE;
--
-- Output process
--
  OUTPUT: process (STATE)
  Begin
    case STATE is
      when S0|S1|S2|S3 =>
        BCD <= "0000";
        DAV <= '0';
      when S4 =>
        DAV <= '1';
        case SHIFT_REG is
          when "0000" => BCD <= "0000";
          when "0001" => BCD <= "0001";
          when "0011" => BCD <= "0010";
          when "0010" => BCD <= "0011";
          when "0110" => BCD <= "0100";
          when "1110" => BCD <= "0101";
          when "1010" => BCD <= "0110";
          when "1011" => BCD <= "0111";
          when "1001" => BCD <= "1000";
          when "1000" => BCD <= "1001";
          when others => BCD <= "0000";
        end case;
    end case;
  end process OUTPUT;
end FSM_RTL;
```