State machine

Mealy and **Moore** are two basic types of sequential networks. In a **Mealy network**, the outputs depend on both present state and the present inputs. In a **Moore network**, the outputs depend only on the present state. A general model of a mealy sequential network needs one combinational network to generate the outputs and the next state and a state register to hold the present state. The state register normally consist of **D flip-flops**. In this exercise, we will design a code converter as an example of a Mealy sequential network. This converter has to convert an **8-4-2-1 binary-coded-decimal (BCD)** digit to an **exess-3-coded** decimal digit.

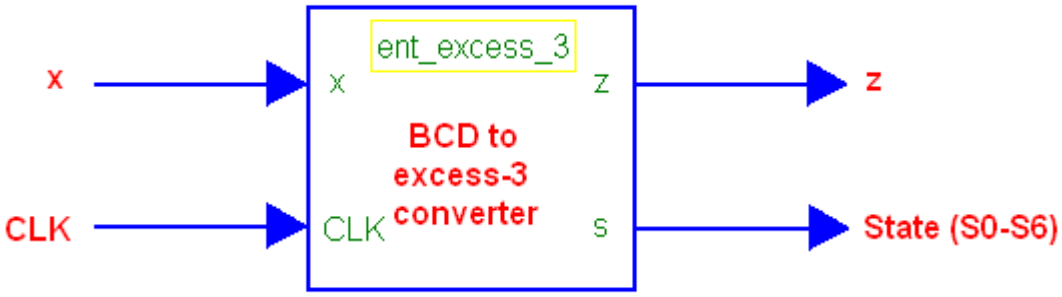
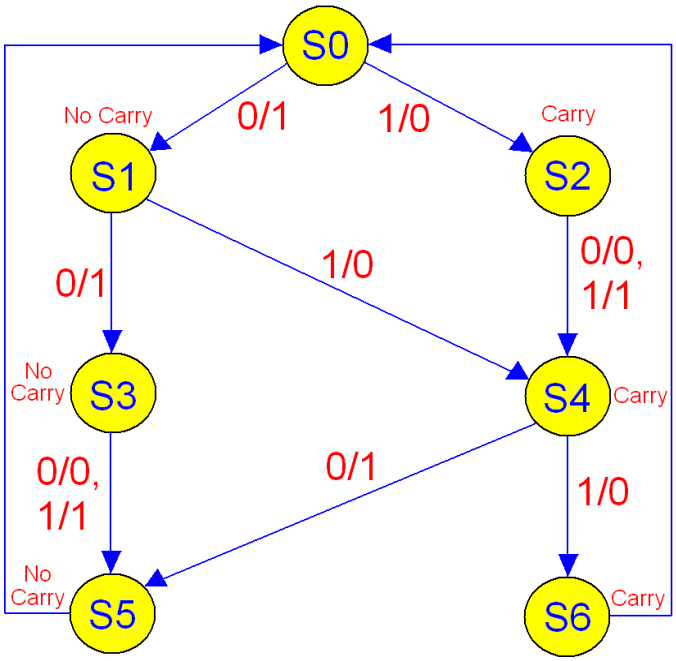


Fig.-1: BCD to excess-3 converter



Present State	Next State		z	
	x = 0	x = 1	x = 0	x = 1
S0	S1	S2	1	0
S1	S3	S4	1	0
S2	S4	S4	0	1
S3	S5	S5	0	1
S4	S5	S6	1	0
S5	S0	S0	0	1
S6	S0	-	1	-

Fig.-2: State Graph and State Table for BCD to excess-3 code converter

VHDL CODE

```
================================================================================
Excess_3.vhdl
================================================================================
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Hendra Kesuma

entity ent_excess_3 is
    Port ( x      : in std_logic;
           CLK  : in std_logic;
           z      : out std_logic;
           s      : out std_logic_vector (2 downto 0));
end ent_excess_3;

architecture arch_ent_excess_3 of ent_excess_3 is
signal State, NextState : integer := 0;
begin
  P1: process(State,x)      --Combinational Network
  begin
   case State is
     when 0 =>
       if   x ='0' then z<='1' ; NextState <= 1; s <= "000";
       else z<='0' ; NextState <= 2; s <= "000";end if;
     when 1 =>
       if   x ='0' then z<='1' ; NextState <= 3; s <= "001";
       else z<='0' ; NextState <= 4; s <= "001"; end if;
     when 2 =>
       if   x ='0' then z<='0' ; NextState <= 4; s <= "010";
       else z<='1' ; NextState <= 4; s <= "010"; end if;
     when 3 =>
       if   x ='0' then z<='0' ; NextState <= 5; s <= "011";
       else z<='1' ; NextState <= 5; s <= "011"; end if;
     when 4 =>
       if   x ='0' then z<='1' ; NextState <= 5; s <= "100";
       else z<='0' ; NextState <= 6; s <= "100"; end if;
     when 5 =>
       if   x ='0' then z<='0' ; NextState <= 0; s <= "101";
       else z<='1' ; NextState <= 0; s <= "101"; end if;
     when 6 =>
       if   x ='0' then z<='1' ; NextState <= 0; s <= "110";
       else z<='0' ; NextState <= 0; s <= "110"; end if;
     when others =>
           z<='0' ; NextState <= 0; s <= "000";
   end case;
end process;
```

```
P2:process(CLK)      --state register
  begin
  if clk='1' and clk'EVENT then
    State <= NextState;
  END IF;
end process;

end arch_ent_excess_3;
```

TESTBENCH CODE

==========

tb_excess_3.vhd

==========

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use work.comp_bit.ALL; -- Include package

-- Hendra Kesuma

ENTITY ent_excess_3_tb_excess_3_vhd_tb IS
END ent_excess_3_tb_excess_3_vhd_tb;

ARCHITECTURE behavior OF ent_excess_3_tb_excess_3_vhd_tb IS

        COMPONENT ent_excess_3
        PORT(
                x     : IN std_logic;
                CLK : IN std_logic;
                z     : OUT std_logic;
                s     : OUT std_logic_vector(2 downto 0)
                );
        END COMPONENT;

        SIGNAL x     :  std_logic;
        SIGNAL CLK :  std_logic;
        SIGNAL z     :  std_logic;
        SIGNAL s     :  std_logic_vector(2 downto 0);

BEGIN
        uut: ent_excess_3 PORT MAP(
                x     => x,
                CLK => CLK,
                z     => z,
                s     => s     );
```

```
-- *** Test Bench - User Defined Section ***
   tb1: PROCESS
   BEGIN
   ApplyData_Procedure(x, "0000110000000000", 10 ns);
      wait; -- will wait forever
   END PROCESS;

   tb2 : PROCESS
   BEGIN
     loop
       CLK <= '0';
     wait for 5 ns;
       CLK <= '1';
     wait for 5 ns;
     end loop;
     wait; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

SIMULATION