Exercise 1

Minimize the Boolean equation for the control of the segment 1 in the following seven-segment display, using Karnaugh maps.

```
  - 0 -
|       |
5       1
|       |
  - 6 -
|       |
4       2
|       |
  - 3 -
```

The inputs of the circuit are the four variables ***a b c d*** representing the bits of the number from 0 to 9 to be represented, with ***a*** the most significant bit and ***d*** the least significant bit. The output of the circuit is the output ***Z*** which needs to be equal to ‘1’ when the segment 1 in the figure above needs to be illuminated and ‘0’ otherwise.

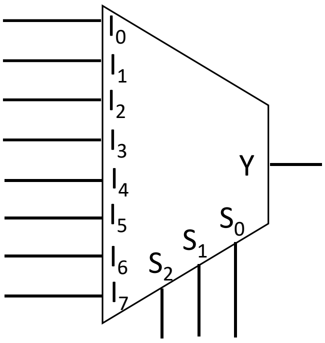
Highlight the following steps:

a) Find the truth table of the required logic function

b) Minimize the function in Sum of Products (SOP) form using Karnaugh map method and properly considering the don’t care conditions

c) Draw the corresponding logic circuit

d) Realize the logic function through the 8-to-1 line-multiplexer in the figure. Clearly indicate how *a, b, c, d*, and *Z* should be connected to the mux.

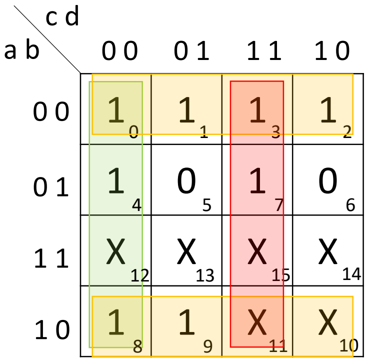


Solution Exercize 1:

a) Truth table

a	B	c	d	Z
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

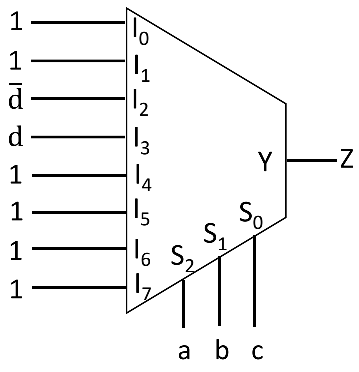
b) Minimized function in Sum of Products (SOP) form:



a b \ c d	0 0	0 1	1 1	1 0
0 0	1_{0}	1_{1}	1_{3}	1_{2}
0 1	1_{4}	0_{5}	1_{7}	0_{6}
1 1	X_{12}	X_{13}	X_{15}	X_{14}
1 0	1_{8}	1_{9}	X_{11}	X_{10}

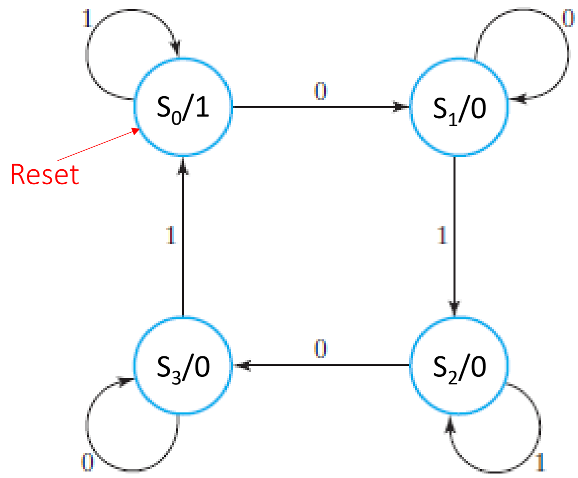
Z = $\mathrm{cd} + \bar{\mathrm{c}}\bar{\mathrm{d}} + \bar{\mathrm{b}}$

d) Implementation with 8-to-1 line-mux (note: due to don't care conditions, there are multiple correct implementations)



Exercise 2

Consider the sequential system described by the state diagram below, with input x and output z.



Answer to the following questions:

a) Is this a Mealy or a Moore machine? Motivate your answer

b) Find the state transition and output table

c) What is the minimum number of flip-flops needed to implement the system? How many flip-flops are needed to implement the system using a 1-hot encoding for the states?

d) Describe the system in VHDL, using positive edge triggered D-Flip-Flops.

Solution Exercize 2:

a) It's a Moore machine, as the output depends only on the current state and not on the input.

b) State transitions and output table

Present state	Future state		Output Z
	X = 0	X = 1	
S_0	S_1	S_0	1
S_1	S_1	S_2	0
S_2	S_3	S_2	0
S_3	S_3	S_0	0

c) The minimum number of bits needed to represent the states of the system is 2. 4 filp-flops are needed to implement the system using a 1-hot encoding representation for the states.

d) A possible VHDL description is a behavioral one:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity seq is
   port (clk, rst, X: in std_logic;
                    Z: out std_logic);
end seq;

architecture beh of seq is
   type state_type is (S0, S1, S2, S3);
   signal state, next_state: state_type;
begin

-- Process 1: Updates state and implements asynchronous reset
   state_register: process (rst, clk)
   begin
         if (rst = '1') then
               state <= S0;
         elsif (clk'event and clk = '1') then
               state <= next_state;
         end if;
   end process;

-- Process 2: Computes the next state (function of input and present state)
   next_state_comb: process (X, state)
   begin
   case state is
         when S0 =>
               if X = '0' then
                     next_state <= S1;
               else
                     next_state <= S0;
               end if;

         when S1 =>
               if X = '0' then
                     next_state <= S1;
               else
                     next_state <= S2;
               end if;

         when S2 =>
               if X='0' then
                     next_state <= S3;
```

```
                 else
                       next_state <= S2;
                 end if;

          when S3 =>
                 if X='0' then
                       next_state <= S3;
                 else
                       next_state <= S0;
                 end if;
   end case;
   end process;

-- Process 3: calculates the output (function of the state)
   output_comb: process (state)
   begin
          if (state = S0) then
                 Z <= '1'
          else
                 Z <= '0';
          end if;
   end process;

end beh;
```