Combination Lock

Requirements for the exercise (issues and skills necessary to complete the task):

- representation of numbers in decimal, binary and hexadecimal systems;
- setting up a new project in Quartus Prime;
- creating a hardware module (symbol) in Quartus Prime based on a schematic file (*.bdf);
- creating a hardware module (symbol) in Quartus Prime based on a source code file (eg. *.vhd);
- ability to simplify a logical expressions using the Karnaugh Map method;
- ability to implement a scheme with logic gates based on an algebraic equation;
- **ability to implement a Moore machine;**

1. Task

Design a combination lock in the form of a Moore machine. In the nominal state (S0), the lock is closed and waits for a bit sequence to be entered. To open the lock, a correct 6-bit sequence must be entered through the single-bit input X. Any incorrect bit entered during the sequence immediately causes the system to return to the initial state (S0). The lock operates in sync with a clock signal — each rising edge of the CLK signal causes the current value of input X (i.e., the next bit in the sequence) to be read. After correctly entering all 6 bits of the sequence, the lock opens. When in the open state, the lock can be closed again by entering the sequence 11, which returns it to the nominal state (S0). The state of the lock (open or closed) should be indicated by displaying the text OPEN or CLOSED on the 7-segment displays. The input X should be connected to the switch SW[0], while the clock signal CLK should be obtained from the KEY[0] button (after a debouncing circuit). Each press of the KEY[0] button generates one rising edge and causes the bit from input X to be read. The state machine should be implemented using two different types of flip-flops. For easier debugging, in addition to the output controlling the lock state (open/closed), output signals representing the state bits (q2, q1, q0) should also be provided (Total: 2.0 points).

a) Implementation of a module displaying the text OPEN/CLOSED on the FPGA: 0.5 pts.
b) Correctly drawn state diagram for the combination lock automaton: 0.5 pts.
c) Implementation of the lock in the FPGA: 0.5 pts.
d) Use of two different types of flip-flops in the implementation: +0.25 pts.
e) Implementation of the circuit as a hardware block (symbol): +0.25 pts.

The table below presents the individual 6-bit unlocking sequences assigned to each team.

GROUP 1	GROUP 2
Team 1 – opening sequence: 100001	Team 1 – opening sequence: 111010
Team 2 – opening sequence: 100010	Team 2 – opening sequence: 111011
Team 3 – opening sequence: 100011	Team 3 – opening sequence: 111100
Team 4 – opening sequence: 100100	Team 4 – opening sequence: 111101
Team 5 – opening sequence: 100101	Team 5 – opening sequence: 100010
Team 6 – opening sequence: 100110	Team 6 – opening sequence: 100100
Team 7 – opening sequence: 100111	Team 7 – opening sequence: 100101
Team 8 – opening sequence: 101000	Team 8 – opening sequence: 100110
GROUP 3	GROUP 4
Team 1 – opening sequence: 101001	Team 1 – opening sequence: 101100
Team 2 – opening sequence: 101010	Team 2 – opening sequence: 101101
Team 3 – opening sequence: 101011	Team 3 – opening sequence: 101110
Team 4 – opening sequence: 101100	Team 4 – opening sequence: 110000
Team 5 – opening sequence: 101101	Team 5 – opening sequence: 110001

Team 6 – opening sequence: 101110	Team 6 – opening sequence: 110010
Team 7 – opening sequence: 110000	Team 7 – opening sequence: 110011
Team 8 – opening sequence: 110001	Team 8 – opening sequence: 110100
GROUP 5	GROUP 6
Team 1 – opening sequence: 110010	Team 1 – opening sequence: 110101
Team 2 – opening sequence: 110011	Team 2 – opening sequence: 110110
Team 3 – opening sequence: 110100	Team 3 – opening sequence: 110111
Team 4 – opening sequence: 110101	Team 4 – opening sequence: 111000
Team 5 – opening sequence: 110110	Team 5 – opening sequence: 111001
Team 6 – opening sequence: 110111	Team 6 – opening sequence: 100010
Team 7 – opening sequence: 111000	Team 7 – opening sequence: 100011
Team 8 – opening sequence: 111001	Team 8 – opening sequence: 100100

2. Debouncer

A signal coming directly from mechanical buttons contains disturbances that occur at the moments of pressing and releasing — that is, at the times when a single rising or falling edge of the signal should appear. These disturbances result from mechanical contact bounce. Feeding such a noisy signal directly into a digital circuit can cause malfunctions, due to the presence of multiple rising and falling edges instead of just one. To eliminate this phenomenon, a debouncing circuit is used. It can be implemented either: analogously, using low-pass filters, or digitally, by measuring the time during which a stable high or low signal level is maintained.

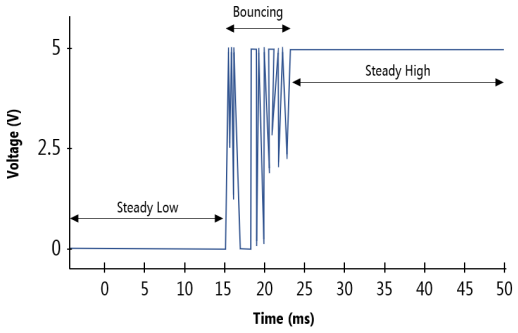

Fig. 1: Button state change

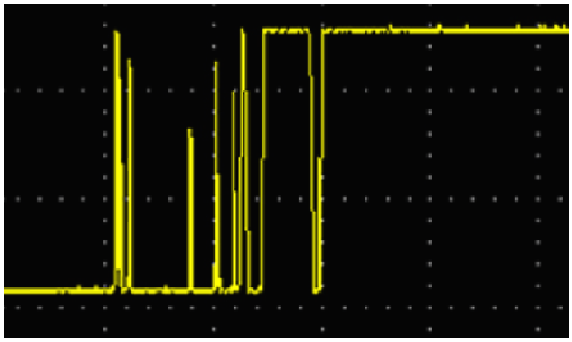

Fig. 2: Signal occurring when the button state changes

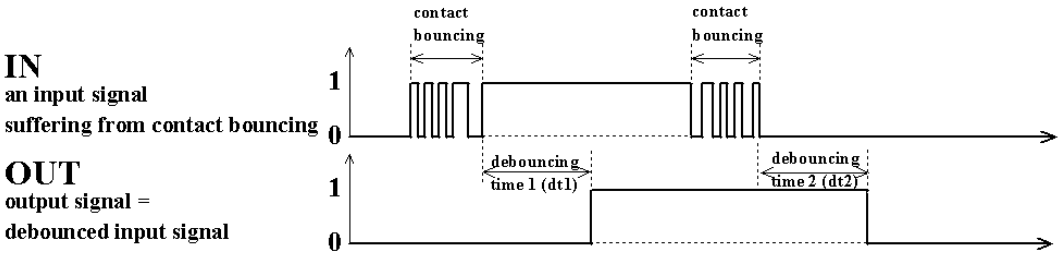

Fig. 3: Operation of a debouncer circuit,
input signal with noise, and output signal without noise

3. Module displaying a message

Module displaying a message: CLOSED / OPEN

OUTPUTS	INPUT STATE Z		LOGIC FUNCTION
	CLOSED (0)	OPEN (1)	
HEX0[0]	1	0	~ Z
HEX0[1]	0	0	0
HEX0[2]	0	0	0
HEX0[3]	0	1	Z
HEX0[4]	0	0	0
HEX0[5]	1	0	~ Z
HEX0[6]	0	1	Z
HEX1[0]	0	0	0
HEX1[1]	1	1	1
HEX1[2]	1	1	1
HEX1[3]	0	0	0
HEX1[4]	0	0	0
HEX1[5]	0	0	0
HEX1[6]	0	0	0
HEX2[0]	0	0	0
HEX2[1]	1	0	~ Z
HEX2[2]	0	1	Z
HEX2[3]	0	1	Z
HEX2[4]	1	0	~ Z
HEX2[5]	0	0	0
HEX2[6]	0	0	0
HEX3[0]	0	0	0
HEX3[1]	0	0	0
HEX3[2]	0	0	0
HEX3[3]	0	0	0
HEX3[4]	0	0	0
HEX3[5]	0	0	0
HEX3[6]	1	1	1
HEX4[0]	1	1	1
HEX4[1]	1	1	1
HEX4[2]	1	1	1
HEX4[3]	0	1	Z
HEX4[4]	0	1	Z
HEX4[5]	0	1	Z
HEX4[6]	1	1	1
HEX5[0]	0	1	Z
HEX5[1]	1	1	1
HEX5[2]	1	1	1
HEX5[3]	0	1	Z
HEX5[4]	0	1	Z
HEX5[5]	0	1	Z
HEX5[6]	1	1	1

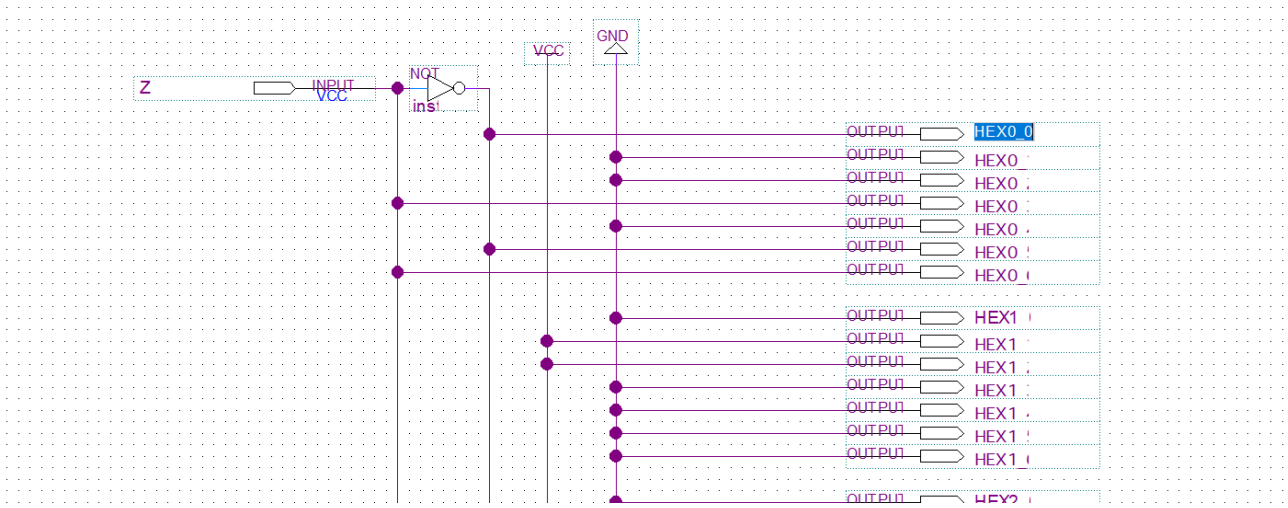

Fig. 4: Connection of control signals to the outputs of the HexOpenClosed module

4. Test circuit

Below is a circuit for testing the combination lock, consisting of a Debouncer (ready-made VHDL module), the combination lock state machine, and a module displaying the open/closed message.

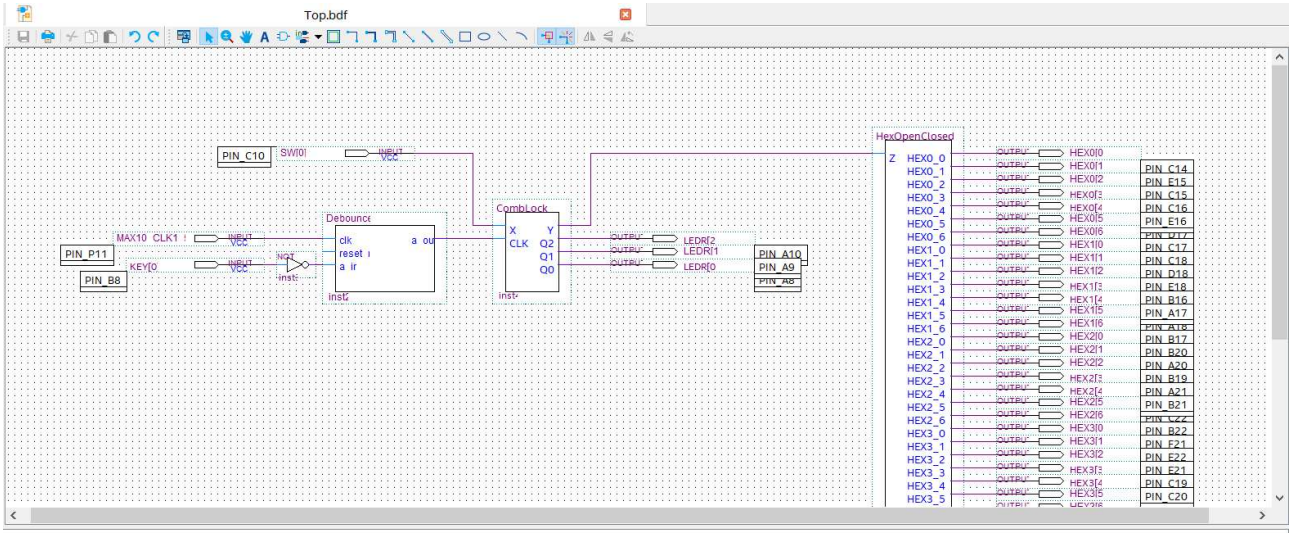

Fig. 5: Test circuit of the Combination Lock