Hardware Design using VHDL
Lab 2

1. Purpose

The lab should provide training in the use of VHDL for description and simulation of digital circuits.

2. VHDL

VHDL is a language for description of digital circuits. VHDL is a standardized language that allows description of digital circuits behaviour and structure of various abstraction levels and is a very comprehensive language. There are simulators for circuits described in VHDL. Here, we use Active-HDL from Aldec. Synthesizing is compared with simulation more complex. We are using Synplify Pro from Synplicity.

3. Preparation

Read Sjoholm-Lindh: VHDL for Designers chapter 3, 4 and 9. Read the lab-PM.
Perform design work for problem 1-2.

4. Exercises

Laboratory contains two tasks, a combinatorial task and a sequence task.

Problem 1: ALU

Function and performance:

Design a simple ALU using a FPGA. The system has two operands as inputs operands and one operational code. The input data is connected via DIP switches on the system board.
The ALU will receive two 3-bit binary numbers, X = $<x_2, x_1, x_0>$ and Y = $<y_2, y_1, y_0>$.
It also have two control signals, OP = $<op_1, op_0>$, to perform four different operations.
The 3 +3 operand bits and 2 operational bits are taken from the DIP switches on the system board.
The outputs, 7 +7 bits, are connected to the 7-segment displays LED1 and LED2.
A result is calculated depending on the operand values and the operational code. The result will be presented on the LED displays.
As an input to the system will be changed, the results will be changed.

Operational codes:

00	addition of the two 3-bit operands
01	multiplication " "
10	bitwise AND
11	bitwise XOR

The results obtained in binary form. Presentation of the results on the displays both in hexadecimal form, and in decimal form. The choice between hexadecimal and decimal form is made with one of the push button switches.
In order to present the results on the displays in decimal form a conversion from binary code to decimal form take place. Perform this transformation and present the results on the displays.

Problem 2: Codelock

Function:

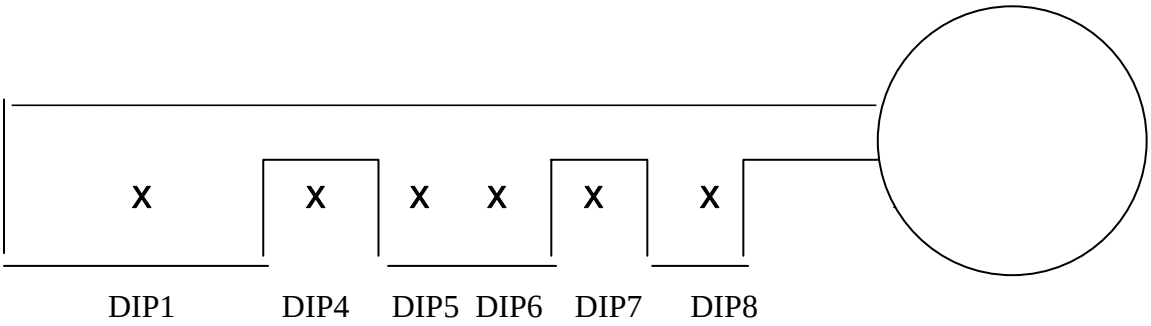
The key (depicted below), will be simulated with six DIP switches. Five DIP switches (DIP4-DIP8) give the code, the sixth DIP switch (DIP1) recognizes that the key is fully inserted in the lock.

When the key is in (DIP1 ='1 ', the signal-key ='1') will open the electronic lock if the code combination is right. Open the lock means that the output from the electronics module will go low (open signal ='0 '). This is shown by the LED (pin A9). After three attempts with invalid code the lock will be inactivated and give an alarm signal. This can be indicated by a segment on one of the displays. In this case it is not possible to open the lock with the correct code. The lock is inactive until it will be activated. The lock is activated with a zero on one input (signal resetn ='0 '). For this, use the Push Button SW6.

The code is assumed to be fixed (built into the logic).

The number of combinations is low (32), so the lock will not be approved by the insurance companies. It is easy to increase the number of fields and thus get a more secure lock.

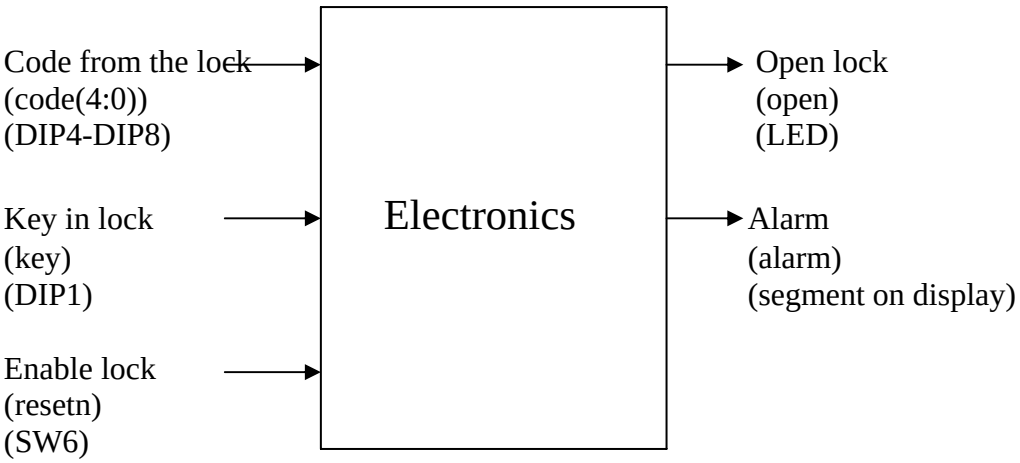
The code in this case is 01101. You may choose the combination of your design. Only zeros or only ones are not allowed.



Execution

The design will be simulated, synthesized and implemented in a gate array and tested on the system board.

Principle layout of the lock inputs and outputs.



The function will be described using a state machine.