Lab 3 – Implementing & Verifying a Door Lock

In this lab you will be specifying, implementing, and verifying parts of a classical code-based door lock system. The system uses 4-digit PINs entered sequentially by pressing the buttons of a keypad, unlocks the door for 3 seconds if a correct code is entered, and has functionality to update the stored code using a Master key.

Requirements:

- Familiarity with Verilog.
- Knowledge about SystemVerilog assertions.

Tools:

- We will use the EBMC tool for bounded model checking and *k*-induction. EBMC can either be downloaded from http://www.cprover.org/ebmc/, or invoked through the web interface http://logicrunch.it.uu.se:4096/~wv/ebmc/

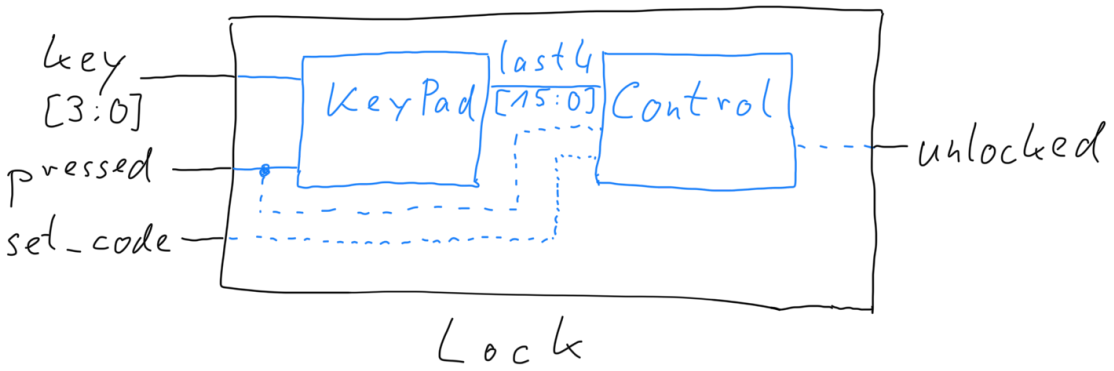
Intended Learning outcomes:

- Translation of specifications to SystemVerilog assertions.
- Implementation according to specification.
- Verification and debugging of Verilog designs with the help of static analysis.

Assessment:

- Verify your implementation against translated and provided requirements using EBMC.
  Help with the lab is provided during the following lab sessions:
  - Thursday May 7th, 13:15 – 17:00.
  - Monday May 11th, 10:15 – 12:00.
  - Thursday, May 14th, 13:15 – 17:00.
- Before submitting your solution, convince yourself through verification or test that it is working correctly; format your code properly, and add comments where necessary.
- To pass the lab, your solution has to be checked and accepted by the submission system by May 14:
  http://logicrunch.it.uu.se:4096/~wv/verilog-verif-lab/

Overview of the Door Lock System



The diagram shows the structure of the Verilog door lock design. The door lock has an overall module `Lock` with three inputs (plus the clock input) and one output:

- `key`: the binary-coded digit for a key, a number in the range [0, 10);
- `pressed`: a Boolean signal telling whether a key was pressed in the current cycle;
- `set_code`: a Boolean signal telling whether the door code stored in the system should be updated to the code entered last (Master key was used);
- `unlocked`: a Boolean signal telling whether the door is unlocked.

```
module Lock(
           output unlocked,
           input [3:0] key,
           input pressed,
           input set_code,
           input clk);
  // ...
endmodule
```

You can find a skeleton of the door lock design on the student portal. The skeleton includes all required modules, but the right behavioural implementations have to be added to get a working system.

To make sure that the door lock has been implemented correctly, we formulate a set of simple requirements.

Overall Requirements of the Lock:

- **R1:** If no key is pressed and the door is locked, then the door stays locked.
- **R2:** If the door is unlocked and no key is pressed, the door is locked within 10 cycles.

Assignment 1: Implementation and Verification of the KeyPad

The `KeyPad` is the component (the Verilog module) receiving physical key presses, and encoding the last four digits as a 16-bit bitvector `last4`:

```
module KeyPad(
          output [15:0] last4,
          input [3:0] key,
          input pressed,
          input clk);
  // ...
endmodule
```

For instance:

- Initially, the output `last4` is 0.
- After pressing key 1, the output becomes 1.
- After pressing key 2, the output becomes 12 (decimal notation)
- After pressing key 3, the output becomes 123
- After pressing key 4, the output becomes 1234
- After pressing key 5, the output becomes 2345
- *Etc.*

NB: After a keypress, the next `last4` output can be computed using an assignment `x <= (x % 1000) * 10`. The alternative formula `x <= (x * 10) % 10000` should be avoided, since it leads to integer overflows.

The keypad is supposed to satisfy the following requirements:

Requirements of the KeyPad:

- **R4:** The output `last4` is an integer in the interval [0, 10000).
- **R5:** If a key is pressed, the output `last4` is updated within 1 cycle, and the remainder of the new output modulo 10 will be the code of the pressed key.
- **R6:** If no key is pressed, the output stays the same in the next cycle.

To establish those requirements, the keypad can rely on an assumption about the inputs:

- **E1:** the input `key` is a number in the interval [0, 10).

In this assignment, you have to do the following things:

- Starting from the code template on the student portal, implement the keypad in such a way that it satisfies **R4**-**R6**, assuming **E1**.
- Formally verify using EBMC that your implementation satisfies requirements **R4**-**R6**. For this you have to express **R4**-**R6** as SystemVerilog assertions, and **E1** as a SystemVerilog assumption, and put all expressions in the observer module `ReqKeyPad`. Verify the requirements up to a bound of at least 20 using BMC.

Assignment 2: Implementation and Verification of the Controller

`Control` is the component that decides whether the door should be unlocked. The component stores the currently set code of the lock and continuously compares it to the number entered through the keypad. Whenever the right code was entered, the controller will set its output to true.

```
module Control(
          output code_matches,
          input pressed,
          input [15:0] last4,
          input set_code,
          input clk);
  // ...
endmodule
```

Requirements of the Controller:

- **R7:** If no code has been set using the `set_code` input, the door will not be unlocked.
- **R8:** The controller will unlock the door only if a key press occurred in the previous cycle.

To establish those requirements, also the controller can rely on an assumption about the inputs. The assumption can be made due to requirement **R4** of the keypad:

- **E2:** the input `last4` is a number in the interval [0, 10000).

Like in assignment 1, your task is to:

- Implement the controller in such a way that it satisfies **R7**-**R8**, assuming **E1**.
- Formally verify using EBMC that your implementation satisfies requirements **R7**-**R8**. Verify the requirements up to a bound of at least 20 using BMC.

As a bonus assignment, you can also try to formalise and verify the following requirement:

- **R9:** If the door code has been set to a number C, if the input `set_code` has not been true since then, and if C is entered again, the door will be unlocked in the next cycle.

Assignment 3: Verification of the overall Door Lock System

This final assignment is about the verification of the overall door lock system, including the two modules that you have implemented in the previous assignments. The implementation of the `Lock` module itself is already provided as part of the code skeleton on the student portal. The module internally instantiates the modules `KeyPad` and `Control`, and takes care of keeping the door unlocked for a certain number of cycles when the controller decides that the door can be unlocked.

Formally verify using EBMC that your implementation satisfies requirements **R1**-**R2** (from the beginning of the document). Verify the requirements up to a bound of at least 20 using BMC.

To establish the requirements, we might need to add the same environment assumption as for the keypad:

- **E1:** the input `key` is a number in the interval [0, 10).