Computation II: embedded system design (5EIB0)

Description | Submission view

Part 2f: 2020-04-16 Finite State Machine (Design an Electronic Combination Lock - 7pt)

Avaliable from: Thursday, 16 April 2020, 3:35 PM
Due date: Thursday, 16 April 2020, 5:30 PM
Requested files: ecl.v, ecl_top.v (Download)
Type of work: Individual work

Introduction

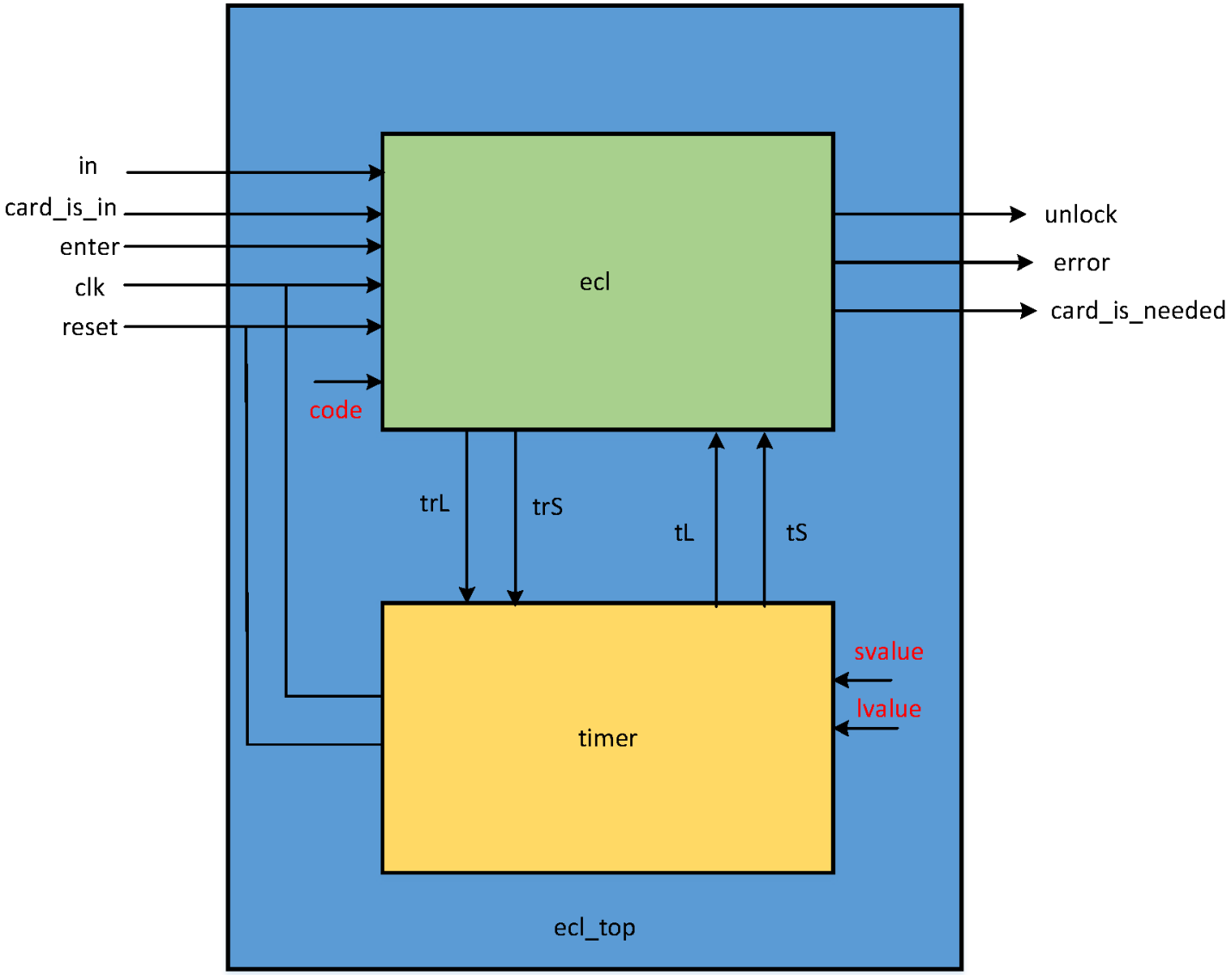
In this assignment, you will create a Verilog implementation of a 3-bit **electronic combination lock (ecl)**. The correct password code is **010**. You have two chances to try the password. When a wrong password is entered for the first time, you have to wait for **5 cycles** before you can try again. The second time you have to wait **10 cycles**, and the third time an **administrator card** is required for trying another combination.

Because these are a lot of requirements, it has been decided to split the design into two separate Verilog modules.

- The first module **timer** is the one from the previous exam question. For this question, however, it has been provided for you (but is not visible). This module is used to count the number of timeout cycles after a wrong password was entered.
- The second module **ecl** contains the finite state machine (FSM) for the lock. This is the module you are going to implement.

The two modules are glued together in a top-level module called **ecl_top**. This module has been mostly provided for you, however, *you still have to define the correct parameters*, namely the amount of wait cycles (**5** for the short delay and **10** for the long delay) and the correct password code (**010**).

A block diagram is shown below:

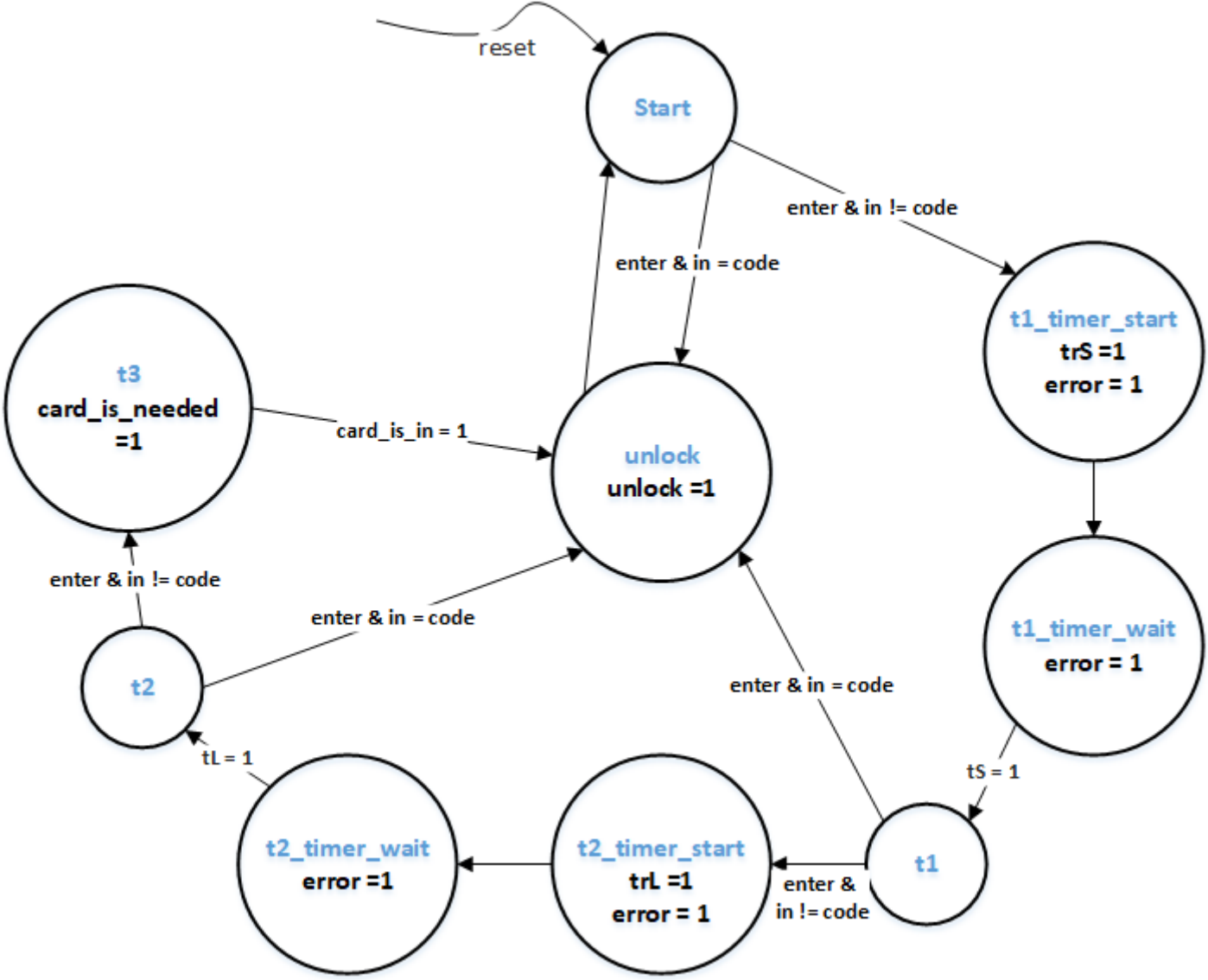


The details of the signals are given below:

- **in (3 bit) –** combinational input
- **enter(1 bit) -** signal indicating new input is entered
- **card_is_in (1 bit) -** signal indicating administrator's card is inserted
- **trL(1 bit) -** trigger signal to start counting for **lvalue** cycles
- **trS (1 bit) -** trigger signal to start counting for **svalue** cycles
- **tL (1 bit) -** signal indicating expiration of a time interval of **lvalue** cycles
- **tS (1 bit) -** signal indicating expiration of a time interval of **svalue** cycles
- **unlock (1 bit) -** signal indicating electronic combination lock is unlocked
- **error (1 bit) -** signal indicating password is wrong
- **card_is_needed (1 bit) -** signal indicating administrator's card is required

The finite state machine

One way to implement the FSM is as a Moore FSM with the following state transition diagram:



The states in the diagram have the following meaning:

- **start**: start-up state; wait until user enters code (first attempt)
- **unlock**: code was correct; unlock the lock for one cycle and reset the number of wrong attempts
- **t1_timer_start**: first attempt was wrong; start short timer
- **t1_timer_wait**: first attempt was wrong; wait until short timer expires
- **t1**: first attempt was wrong; wait until user enters another code (second attempt)
- **t2_timer_start**: second attempt was wrong; start long timer
- **t2_timer_wait**: second attempt was wrong; wait until long timer expires
- **t2**: second attempt was wrong; wait until user enters another code (third attempt)
- **t3**: third attempt was wrong; no more attempts are allowed until administrator card is inserted

Assignment

1. Define the parameters *svalue*, *lvalue* and *code* for the **ecl_top** module by inserting the correct values in the indicated locations in the **ecl_top.v** file.
   Do not change anything else in this file!
2. Implement the **ecl** module (i.e. above FSM) by inserting your code in the indicated location in the **ecl.v** file.
   It is highly recommended to base your implementation on above Moore state diagram, but other approaches (e.g. a Mealy FSM) are also accepted, as long as they are synthesizable to hardware and functionally equivalent to the provided Moore state diagram.

Debug

Click on the symbol marked below to see the waveforms produced by your design. Please note that if your code has an error that prevents it being simulated it will not produce any waveforms.

Requested files

ecl.v

```
`timescale 1ns / 1ps

/* FSM module for electronic combination lock */
module ecl (
        // Clock (positive edge polarity)
        input clk,

        // Reset (active-high synchronous)
        input reset,

        // Combination input
        input [2:0] in,

        // High when administrator's card is inserted
        input card_is_in,

        // High for one cycle when new combination input is entered
        input enter,

        // High when electronic combination lock is unlocked
        output unlock,

        // High when password is wrong
        output error,

        // High when administrator's card is required
        output card_is_needed,

        // Trigger to start counting lvalue cycles
        output trL,

        // Trigger to start counting svalue cycles
        output trS,

        // High when lvalue interval has passed
        input tL,

        // High when svalue interval has passed
        input tS
    );

    parameter [2:0] code = 3'bxxx;

    // <<Insert your own code here>>

endmodule
```

ecl_top.v

```
`timescale 1ns / 1ps

/* Top-level module for electronic combination lock */
module ecl_top (
        // Clock (positive edge polarity)
        input clk,

        // Reset (active-high synchronous)
        input reset,

        // Combination input
        input [2:0] in,

        // High when administrator's card is inserted
        input card_is_in,

        // High for one cycle when new combination input is entered
        input enter,

        // High when electronic combination lock is unlocked
        output unlock,

        // High when password is wrong
        output error,

        // High when administrator's card is required
        output card_is_needed
    );

    // Number of cycles to wait for on first wrong password
    parameter svalue = /* <<Insert correct value here>> */;

    // Number of cycles to wait for on second wrong password
    parameter lvalue = /* <<Insert correct value here>> */;

    // Correct code for combination lock
    parameter [2:0] code = /* <<Insert correct value here>> */;

    // Trigger to start counting lvalue cycles
    wire trL;

    // Trigger to start counting svalue cycles
    wire trS;

    // High when lvalue interval has passed
    wire tL;

    // High when svalue interval has passed
    wire tS;

    /* Instantiate timer module (provided for you) */
    timer #(
        .svalue(svalue),
        .lvalue(lvalue)
    )
    timer_inst(
        .clk(clk),
        .reset(reset),
        .trL(trL),
        .trS(trS),
        .tL(tL),
        .tS(tS)
    );

    /* Instantiate FSM module (which you have to implement in ecl.v) */
    ecl #(
        .code(code)
    )
    ecl_inst (
        .clk(clk),
        .reset(reset),
        .in(in),
        .card_is_in(card_is_in),
        .enter(enter),
```

```
75          .unlock(unlock),
76          .error(error),
77          .card_is_needed(card_is_needed),
78          .trL(trL),
79          .trS(trS),
80          .tL(tL),
81          .tS(tS)
82      );
83
84  endmodule
```

VPL

◄ Part 2e: 2020-04-16 Finite State Machine (Design a Parameterized Timer - 7pt) EDITABLE

Part 2f: 2020-04-16 Finite State Machine (Design an Electronic Combination Lock - 7pt) EDITABLE ►

Return to: Final Exam 2020...