Assignment 4: Lustre and other models of computation

1DT059: Model-based Design of Embedded Software
2022

December 13, 2022

This is a smaller laboratory assignment. You produce programs and text. Your solutions to each exercise exercise should be implemented in a .lus-file, .txt-file and/or .pdf-file that is named after the exercise. Include your name at the top of all submitted files.

Assignments are to be solved by students individually. You can (and are encourage to) discuss ideas and concepts with fellow students, but it is absolutely forbidden to share or copy (even parts of) solutions, lines of code, or similar, from each other, or from models found on internet.

Existing problems that have been used

Problem 4: Elevator Controller In this problem, you will construct a controller for a simple elevator in Lustre. The requirements for the elevator are a simplified form of those for the elevator controller in the project. Below are the simplified requirements, adapted to the context of a Lustre program.

The elevator serves four floors (numbered 1, 2, 3, 4). Its control unit is controlled by users using four buttons, numbered 1, 2, 3, 4, corresponding to four floors, that can be pressed by users (for simplicity, we do not distinguish between buttons at floors and buttons inside the elevator, there is just one button for each floor). Whenever a button for floor n has been pressed, the elevator must eventually (possibly after passing and stopping at some other floors) arrive to floor n. When this happens, a yellow lamp lights up (meaning that doors open). The elevator then stays at this floor until the control unit orders it to move to some other floor. However, once the yellow lamp is on, it must stay on for at least 5 seconds. The elevator must put off the yellow lamp (close doors) before leaving a floor. That is, once the elevator has stopped at a floor, it must remain there for at least 5 seconds (or time units).

The control unit controls the elevator itself via an output signal *speed*, which is the speed of the elevator cart. We use the unit *floors*/*s*, where 0 speed means "stopped", positive speed means "going up", and negative speed means "going down". The absolute speed of the elevator when it is moving should be is 0.2*floors*/*s* (i.e., it takes 5 seconds to go from one floor to the next). Thus, the value of the signal *speed* is either 0, 0.2 or -0.2.

The elevator informs the control unit about its position via the signal *position* (an input to the controller), whose value is a real number, which under normal operation should be between 1 and 4 (i.e., the unit is *floors*). Apart from the input signal *position*, the controller has an input signal corresponding to each button. These are the above mentioned buttons for each floor (*floorn* for $n = 1, 2, 3, 4$).

In order to make this assignment more tutorial-like, it is structured as a sequences of subproblems, which step by step builds a Lustre model of the elevator and the controller.

a) (10p) Make a Lustre node which models the physical elevator. This node has as input the signal *speed*, and as output the signal *position*. At each time step, the signal *position* just moves according to whatever the signal *speed* says. For the purposes of this exercises, we take the Lustre time units to be *seconds*. That is, at each time step (each second), the signal *speed* has some value, which determines how much *position* will change from this time step to the next. Thus, if *speed* is 0.2, then *position* increases by 0.2. An additional input signal (called, e.g., *init_position*) gives the initial position.

b) (20p) Extend your model which a Lustre node which is the simplest form of controller, which has four input signals (*floorn* for $n = 1, 2, 3, 4$) corresponding to the buttons for each floor, one input signal *position*, and output signal *speed* and *doors_open*. The simple controller, together with the elevator node constructed in a), should ensure that, whenever the button for floor n has been pressed, the elevator arrives to floor n within some reasonable time. The elevator then stays at this floor for at least 5 seconds, before leaving for another floor.

   There are of course, many ways to structure this design. Think about how to do it in your way. If you want some inspiration, here is an outline of a possible approach (which, of course, can be varied in a lot of ways; see this just as some possible starting suggestion).

   – As already said, for each floor there is an input signal to the controller. You can use this signal to generate another signal, which at any time instant says whether the elevator still needs to go to the corresponding floor. I.e., this signal is true for floor n if the elevator has not reached floor n after (or also at the same time as, if you like) the last press of button n.

   – Given the derived signals in the previous item, you can then generate a signal, which at any moment gives a direction for the elevator, based on where it is and what buttons it must still serve.

   – You can also derive a signal which is true if the elevator should stop at its current position, in order to serve a yet unserved button. This signal can also be used to drive the signal for the yellow lamp.

   – After that you can let the output *speed* of the controller be guided by the direction, provided above, which may be modified if there is a need to remain at the current floor for sufficiently long (the previous item).

Bundle together your controller and the physical model into one Lustre node, which inputs the buttons, and outputs the position.

Of course, you should simulate your Lustre model to check that it works as intended.

d) (10p) Formulate, in Lustre, requirements for your Elevator model. These requirements are similar as in Assignment 3, but this time in Lustre. You can think about this as producing monitors.

R1: The doors are never open when the elevator is not at some floor. This requirement can be formulated in terms of the signal to the yellow lamp, and the signal *position* from the elevator.

R2: When the yellow lamp has been switched on, it remains on for at least 5 time units.

After formulating the requirements, attempt to verify them using Kind2 (e.g., its web interface).

Suggestions: You are encouraged to look at and — when applicable — use the "temporal patterns" that can be succinctly defined in Lustre, such as, e.g., `RisingEdge`, `HasHappened`, `Sofar`, and `Since` from the examples in the Lustre Lectures.

Other

Within some short time, this text will be updated with some small links and tips regarding Kind2 and Lustre.

Submission

Solutions (all files) to this assignment are to be submitted via the Student Portal by
Sunday, January 15, 2023