TP2: LUSTRE
2016-2017

Exercise 1: Average

1. Copy the following nodes in a file ave.lus

   ```
   node MinMax(X:int) returns (min, max:int);
   let
     min = X -> if (X < pre min) then X else pre min;
     max = X -> if (X > pre max) then X else pre max;
   tel

   node Average(x,y:int) returns (A:int);
   var S:int;
   let
      A = (x+y)/2;
      S=x+y;
   tel

   node MinMaxAverage(x: int) returns (a:int);
   var min, max: int;
   let
     a = Average(min,max);
     min, max = MinMax(x); --several outputs
   tel
   ```

2. Simulate using luciole every node:

   ```
   luciole ave.lus MinMax
   luciole ave.lus Average
   luciole ave.lus MinMaxAverage
   ```

3. Executes every the node MinMaxAverage:

   To compile a Lustre file:

   ```
   lustre ave.lus MinMaxAverage
   ```

   a file MinMaxAverage.oc is generated.

   To generate a C code file use:

   ```
   poc MinMaxAverage.oc -loop
   ```

   two files MinMaxAverage.c and MinMaxAverage_loop.c are generated.

   we can generate an executable program using:

   ```
   gcc MinMaxAverage.c  MinMaxAverage_loop.c -o MinMaxaverage
   ```

4. Give the nodes node Min(X:int) returns (min:int) and the node Max(X:int) returns (max:int) computing the minimum and maximum value of the flow X respectively.

5. Give the node node MinOrMax1(x: int; c:bool) returns (a:int) computing the maximum if c is true and the minimum if c is false of the flow x. Use the nodes Mim and Max in this node. Simulate.

6. Give the node node MinOrMax2(x: int; c:bool) returns (a:int) computing when c is true the maximal value among the values of x for which c is true and the minimal value among the values of x for which c is false otherwise. Use the nodes Mim and Max in this node. Simulate.

Exercise 2: Edge

Give the node **node EDGE(b:bool) return (edge: bool);** detecting the rising edge of *b*. The flow edge detects the points when *b* moves from false to true: edge is false at the beginning and is true if and only if *b* moves from false to true. Simulate and executes this node.

Exercise 3: clock

The goal is to model a clock where:

1. The node clock computed the time "h (hours), m (minute) and s (seconds)", at each tick (global clock) of the node.

2. Th counters h (hours), m (minute) ans s (seconds) of the clock are initialised at 0.

Give the code of the Lustre node **node clock (tt: bool) returns (h, m, s: int);**. Simulate this node. Execute this node.

Exercise 4: Watchdog

A watchdog is a device used to monitor the response time of a system and possibly trigger an alarm if a deadline is not met.

1. We first consider a node watchdog with three inputs:

   (a) The command "set". When the command set is received the watchdog moves to the "active" state.

   (b) The command "reset". When the command reset is received the watchdog moves to the "inactive" state.

   (c) A Boolean variable indicating the occurrence of a deadline.

   The node has a single output "alarm" indicating a deadline miss. The alarm occurs when the watchdog is active and a deadline event occurs.

   The state of the watchdog, active state or incative state computed using a Boolean variable "is_set" such that is_set is true if the watchdog is active and false if not. The variable is_set is initially at true or false depending on whether condition set is true or false.

The commands set and reset can not be present simultaneously (assert not (set and reset))

Implement **node WD1(set, reset, deadline: bool) returns (alarm: bool);** the node modelling this watchdog. Simulate the node.

2. The second version of the watchdog receives the same commands but the alarm is triggered (one time) if the reset event does not occur for some time (delay) after the occurrence of the last set. The time elapsed since the occurrence of a set is measured by a variable REMAIN whose content is decremented each clock cycle. A deadline is produced when the variable REMAIN reaches 0. Use the node edge to detect that REMAIN reaches 0. Implement and simulate this new node: **node WD2(set, reset: bool; delay: int) returns (alarm: bool);**, use the node WD1 in the node WD2.