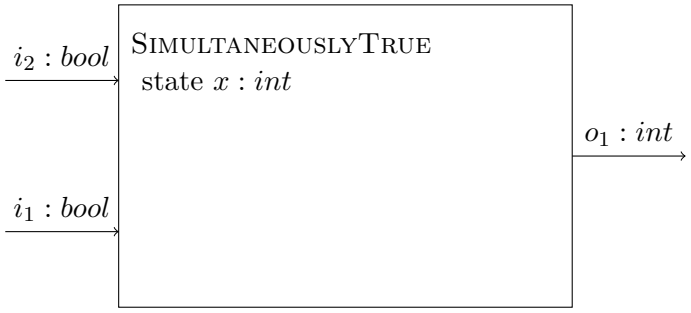
Verification and Control of CPS: Assignment 1
Due Date: Thursday, September 11, 2014 (in class)

P1. Consider the synchronous reactive component below:



The output o_1 must count the number of times both inputs i_1 and i_2 are simultaneously *true*.

1. Write down an imperative code description of the component. Your description should also include an appropriate initial value for the state variable x.

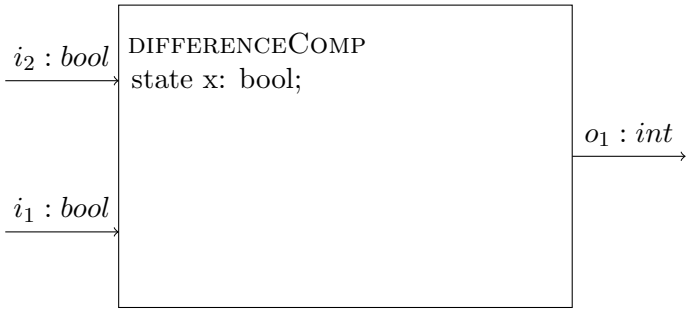
2. Write down an extended state machine description of the component.

Consider a small part of the overall state machine corresponding to the component. The states are shown below but no edges are shown.

3. Indicate the initial state and draw suitable arrows between states. Label edges by their inputs and outputs.

4. Design a combinational component Subtractor that inputs two integers and outputs their difference at each round.

We wish to design a component differenceComp that computes the difference between (a) the number of times the inputs i_1, i_2 are both true and and (b) the number of times the inputs i_1, i_2 are both false.



5. Design differenceComp by using subcomponents SimultaneouslyTrue (described above), SimultaneouslyFalse that counts the number of times the inputs are both false, and the Subtractor (described above).

6. Write down the precedences between the subcomponents and all possible valid execution orders for the subcomponents.

P2 In this assignment, you will write down the extended state machine description of a component CountdownTimer shown below.

Draw an *extended state machine* to model this kitchen timer using the specification below: The inputs to the component include:

1. The input event `setMinute` of type integer sets the number of minutes left in the timer.
2. The input event `setSecond` of type integer sets the number of seconds left in the timer.
3. At each second interval, an input event `tick` is delivered to the component.
4. An input event `start` starts the countdown.
5. An input event `stop/reset` stops the clock and resets it to 0.

The outputs include:

1. `minsLeft` shows how many minutes are left in the countdown.
2. `secsLeft` shows how many seconds are left in the countdown.
3. `alarm` is an output event signifying the sounding of an alarm tone.

The component has three main states in its extended state machine (you can add more states, if needed to implement it):

1. *idle state*: the default state when it is not counting down.
2. *countdown state*: the state where it is counting down.
3. *paused state*: where the countdown has been paused.

The desired behavior is as follows:

1. At the beginning, the clock is in an *idle state*.
2. If a `setMinute` event is received from the *idle state*, then the number of minutes left is updated to the set value, while staying in *idle state*.

3. If a `setSecond` event is received from the *idle state*, then the number of seconds left is updated to the set value while staying in *idle state*.
4. If from the *idle state*, the `start` event occurs and a nonzero time remains, then the *countdown state* is entered.
5. In the *countdown* state, a tick causes the time remaining to decrease.
6. During the *countdown*, the events `setMinute`, `setSecond`, and `start` are "ignored".
7. During the *countdown* the event `stop/reset` results in the *paused* state entered.
8. If the count is *paused* and `start` is received, it resumes once again going back to *countdown* state with the time remaining.
9. If the count is *paused* and `stop/reset` is received, it goes back to *idle state* with time remaining set to 0.
10. During the countdown, if the clock reaches 0, then it outputs the `alarm` event and automatically goes back to *idle state*.

A. Draw the extended state machine clearly on a clean sheet of paper. Label the edges appropriately with the guard and actions.
B. (**Deferred Assignment**). Design the component in Stateflow.