4 Discussion on Project 1

Scribe: Ghid Maatouk 23. March 2007

1 Defining a failure model

While presenting the current state of work, it appeared generally that all groups have refined their model of computation. The underlying assumptions have been made clear. Once the model of computation is well described, the next step is to define a precise failure model.

Principle A definition itself is neither true nor false. However, for a complex notion (such as the correctness of an implementation), you need to convince your reader that your definition is valid, as you may not have both the same thing in mind. In particular, you should prove that what you define is meaningful in some interesting setting.

To define correctness of an implementation, you must clearly state the following:

1. Assumptions:
   - What is your model of computation? Specifying exactly how the outputs are computed from the input both in a component (component semantics) and in a component mapped to a platform (implementation semantics).
   - What is your failure model? Does it assume discrete time (processor is either up or down during a round/computation/cycle/$\cdots$) or continuous time?
2. The restrictions you impose on $s, \Phi, \ldots$ For example, you may choose to disallow cycles in the component program.

After specifying these, you can proceed with the definition keeping in mind any interesting property of the defined notion that may help to convince the reader that your definition is "reasonable".