How can a high level Haskell system model finally reach an implementation in HW and/or SW?

1. Participants

KTH: Ingo Sander
LiTH: Zebo Peng
Saab: Stefan Kvist, Torbjörn Månefjord, Ingemar Söderquist
FMV: Gunnar Ericson
Spirea: Fredrik Westman

2. Moderators

Stefan Kvist, Ingo Sander

3. Purpose

The task was to come up with a clear view of the SAVE design flow, including interface to tools for HW and SW implementation. Abstraction levels, skeletons, transformation steps, languages, formats, ...?

In earlier discussions a number of technically important issues have been brought up, e.g. handling of design constraints (buffer sizes, power, etc.), IP and interaction with the environment (”test bench”). Such issues should be identified and defined. Also, actions to resolve these issues should be proposed.

4. Discussion

4.1 Design flow

System design in the ForSyDe-methodology starts with an initial system model, based on a synchronous system model and using skeletons. This model is ideal in the sense, that it uses infinite data types and that it abstracts from implementation details. The next phase in the design flow is the design transformation phase, where the system model is transformed by applying formally defined tranformations to the system model. These design transformations can either be semantic preserving or they introduce design decisions, e.g. constraining the buffer size of an ideal (infinite) buffer. The transformation process is supported by estimation results. The result of the design transformation phase is the implementation model, which is the starting point for the synthesis phase. Each skeleton has a structural hardware and software interpretation, which means, that also the implementation model has a combined hardware and software interpretation. Using this interpretation the implementation model is synthesized into hardware (VHDL) and software (C++) parts.

4.2 Use of IP in the SAVE design flow

Currently, microprocessors can be handled. Other IP:s could be modeled in Haskell or possibly wrapped in some Haskell construct. When an IP written in an other language than Haskell, is included by wrapping it is no longer possible to verify the model as with Haskell only.

Today it is possible to link C-code to Haskell but not VHDL.

4.3 GUI for Haskell models

To make the Haskell language easier to use and read and to make it possible for non Haskell programmers to view the model a graphical representation may help. The graphical interface should produce Haskell code as output and should also be able to read Haskell code and present it graphicly.

Ingo Sander and Ingemar Söderqvist defines an ex-job for creating this graphical interface.

4.4 Issues

- To achieve efficient simulations of systems with both hardware and software the difference in speed between hardware and software has to be handled in some way.

- To synthesize asynchronous implementations from initially synchronous system models is not covered by todays synthesis technique.