Project 22703: SYRF

Deliverable 6.1.2

Translation to VHDL/Verilog

1 Approach

There are natural formal connections between synchronous reactive formalisms and models of synchronous hardware circuits (RTL netlists). These similarities have been recently exploited, both to apply optimisation techniques designed for circuits to synchronous processes, and conversely to give constructive causal meaning to circuits.

Now in the hardware community there already exist celebrated specification formalisms such as VHDL/VERILOG. Despite their numerous field-specific features (which synchronous formalisms currently lack), these languages suffer from informal and ambiguous semantics, specially for the cycle accurate description. So while very restricted subsets of Verilog/VHDL at the so-called "structural" level can be used for useful synthesis and analysis, the general-purpose "behavioural" level is most often only used for early stage simulation. Therefore a translation from synchronous formalisms (at DC level) to "synthesizable" VERILOG could be a mean to combine the advantage of high-level reactive programming and powerful analysis backends. Of course the aim is not so much to replace established languages that are successful in the development cycle with other formalisms, but rather to enjoy the benefits of hardware synthesis tools, even for software production.

2 INRIA Sophia-Antipolis contribution

We have defined and implemented a first prototype version of a translator from DC to VERILOG at the RTL level ("structural Verilog").

At this level (RTL netlists) VERILOG is often itself considered as an exchange format, used as input to a large variety of hardware synthesis tools. Unfortunately there is no large consensus on which kernel should be accepted as *synthesizable*, so that the allowed subset is generally quite limited. Of course the unformal state of semantics of most constructs in the language does not help much here. Therefore we are currently conducting experiments to effectively connect our programs to hardware CAD tools (mostly tools from the Cadence company), to validate the translation.

The issues in the translation itself mainly deal with the actual implementation of data. In the `SC` format produced from *Esterel*, only control is implemented as RTL netlists represented as boolean equation systems; the data values and types being externally described in tables. In common format `DC` the defining equations may freely merge variables from any declared types, with only the type declarations and function bodies actually deported elsewhere. Now in most *synthesizable* subsets of VERILOG, types are restricted to fixed width integers as boolean bitvectors. Which data types should be abstracted to such (concrete) types, and how, is often hard to determine automatically. Performances of analysis and verification techniques of course strongly depend on the faithfulness of abstraction, pointing at connections with tasks 2 and 3.3 inside SYRF.

Another translation issue deals with the fact the hardware description languages (HDL) often consider *wavefronts* (or signal value *changes* as the basic notion, while synchronous languages rather assume a unique time slicing and consider the mere signal level at discrete instants. This notion is also presnt in HDLs, but different constructs are allowed for signal wavefronts and levels.

We intend to provide documentation at the review on the DCVERILOG software tool.