Project 22703: SYRF

Deliverable 6.2.2

Mixed controller/datapath design

1 Approach

We have explored the generation of VHDL-RTL code from SIGNAL source specification. VHDL-RTL is the Register Transfer Level of VHDL, i.e., a low level code in the VHDL design suite. This requires that SIGNAL itself is used to actually specify an architecture at RTL level. This is achieved by using the notion of a SIGNAL process as a direct counterpart of a hardware component. From this RTL architecture specification in SIGNAL, the generation of VHDL-RTL is immediate, by the way of performing a direct mapping from the SIGNAL-RTL architecture down to the VHDL-RTL architecture. Simulations, verifications, optimizations, and exploration of the design space, are thus performed within the SIGNAL environment itself, before VHDL code is generated. For the previous delivery, we have indicated how SIGNAL clocks are encoded in a way suitable to VHDL-RTL generation, from which the translation principles follow.

This year, two main routes have been explored, resulting in significant changes in VHDL code generation.

1. Pushing the work of previous year further, we have explored the possibility of using the SIGNAL source specification for deducing the possibility of resource sharing for the different items constituting the architecture, namely

   - variables and registers
   - basic blocks
   - path segments and busses

   and the like, in a uniform way. To this end, we systematically exploit the hierarchical clock-and-dependency structure which the compiler computes as a result from SIGNAL source codeanalysis. The automatic discovery of mutual exclusion, in a hierarchical way, is the basis for such type of optimisation. This delivery focusses on this part of the work.

2. Using the same SIGNAL based architecture description to generate a model, also written in SIGNAL , foremulating the real-time consumption of the considered hardware architecture. This is a profilingservice. This model could then be submitted to the formal timing analysers studied in task 4/5.3.

2 Achieved Results

1. A pre-prototype has been developed implementing the automatic detection of resource sharing andresulting rearchitecturing of the considered circuit, see the deliverable.

2. A prototype has been developed for SIGNAL program profiling. This prototype is currently under integration in the framework of the overall SIGNAL/DC+-compiler. This should probably be demonstrated at the review.

3 On using hierarchical dependency graphs for mutual exclusiveness identification

Here we give additional details on the first topic, see the references below for more information.

3.1 Motivation

Identifying operation mutual exclusiveness is important in hardware design, in order to reduce the required number of control steps and/or hardware resources during behavioural synthesis, by the way resource sharing. Candidate resources for possible sharing include variables and registers, basic blocks, and path segments and busses.

Consequently, mutual exclusiveness identification must be performed at the RTL description level, in which architectural elements are visible. On the other hand, recent techniques for mutual exclusiveness identification include both accurate dataflow analysis, and reasoning on the control of the program. Combining both analyses allows one to perform conditional resource sharing, i.e., the sharing of resources *depending on the current state of the control.*

Performing such type of identification generally involves difficult program analysis, in particular for designs specified using VHDL. Our idea is to perform such an analysis prior to VHDL code generation, by the way of starting from a SIGNAL description of the hardware architecture at the RTL level. The technique consists in the following steps:

1. start from a SIGNAL description of the hardware architecture at the RTL level;

2. perform mutual exclusiveness identification;

3. propose conditional resource sharing;

4. map the resulting modified architecture into VHDL;

5. use VHDL tools for evaluation of the design (surface, power consumption, trhoughput, and the like).

We now describe how step 2 above is performed.

3.2 Mutual exclusiveness identification

Fortunately enough, the process of SIGNAL program compilations requires a thorough dataflow and control analysis. This is performed as follows:

1. *Clocks* are attached to flows, indicating when each considered flow is "present" or available. SIGNAL primitive statements result in relations between the clocks of the involved flows.

2. On the other hand, *conditional dependencies* or *scheduling specifications* can be expressed using the SIGNAL primitive statement

   `flowX` $\rightarrow$ `flowY` **when** `Condition`

   which expresses that `flowY` cannot be evaluated prior to `flowX` when `Condition` holds true. Such a property for the triple {`flowX`,`flowY`,`Condition`} typically results from a causality analysis of the considered SIGNAL program.

3. Thus a direct analysis of the SIGNAL source code results in a bench of clock equations and conditional dependencies. A major step in the compilation of SIGNAL programs consists in restructuring these items in such a way that

   - The number of clock equations which are not clock definitions of the form $h = clkexpr(c_1, c_2, \ldots)$ is minimized, the ideal case being that this number is brought to zero. In the latter case, all clocks are functionally determined using clocks and boolean flows, in a non circular way. The resulting clock hierarchy is generally a family of trees interconnected by clock relations, and is a single tree in the most favourable case.
   - Dependencies which hold at a particular clock are grouped together and attached to the considered clock in the above clock hierarchy.

4. At this stage, mutual exclusiveness is directly visible in the resulting structure.

Then, resource sharing is expressed by the way of rewriting the SIGNAL program so that desired multiplexing is made explicit. Then VHDL generation is performed, and can be given to VHDL analysis tools for performance evaluation, as said before.

References

[1] A.A. KOUNTOURIS, PhD Thesis, Univ. Rennes I, 1998.

[2] A. Kountouris, C. Wolinski, Hierarchical Conditional Dependency Graphs for Mutual Exclusiveness Identification, to appear in the proceedings of VLSI'99, Goa, India, Jan. 7-10, 1999.

[3] A. Kountouris, C. Wolinski, Combining Speculative Execution and Conditional Resource Sharing toEfficiently Schedule Conditional Behaviors, to appear in the proceedings of ASP-DAC'99,Hong-Kong, Jan. 18-21, 1999. 45K)

[4] A. Kountouris, C. Wolinski, False Path Analysis based on a Hierarchical Control Representation, toappear in the proceedings of ISSS'98, Hsinchu, Taiwan, R.O.C., December 2-4, 1998.

[5] A. Kountouris, C. Wolinski, Hierarchical Conditional Dependency Graphs for Conditional ResourceSharing, to appear in the proceedings of Euromicro 98, 24th-27th, August 1998, Vasteras, Sweden,IEEE Computer Society Press.

[6] A. Kountouris, C. Wolinski, Efficient Mutual Exclusiveness Identification in Hierarchical ConditionalDependency Graphs, IRISA Research Report, under review, to appear. (compressed postscript 72K)

Bibliographical items are available at