A Demonstration of Co-Design and Co-Verification in a Synchronous Language

Satnam Singh
Xilinx Inc.
2100 Logic Drive, San Jose, CA95124, USA.
Satnam.Singh@xilinx.com

This interactive demonstration illustrates how the synchronous programming language Esterel [3] can be used to design and verify both hardware and software. The interactive demonstration will use a laptop which will be used to drive an Esterel system description all the way to hardware and software which will be implemented on an FPGA using vendor tools. This laptop will also be used to interactively prove challenging properties of the generated system. Additionally an FPGA-based board will be used to execute the generated hardware and software.

Esterel has demonstrated very good results for the design of control based systems in either hardware or software. Esterel also has interesting static analysis capabilities for control based systems. This interactive demonstration illustrates that power of combining these two complimentary technologies for the design and verification of embedded systems. The demonstration will contain several case studies that use Xilinx's new Virtex™-II PRO FPGAs which contain a mix of configurable logic, IBM PowerPC405 processors and high speed serial links.

One problem with conventional system level design approaches is that it is very hard to reason about the composite behavior of a system that comprises hardware and software. The root of the problem is that the usual semantic models for software are virtually impossible to relate to the event-queue model semantics of hardware description languages. This makes it hard to analyze hardware/software systems and it also makes it hard to synthesise good hardware or software from the same description. Thus many conventional system level techniques show some promise for system level modelling but are not adequate for system level synthesis. Furthermore, many high level tools generate generic RTL code which leads to very sub-optimal implementations on any given architecture. This tutorial presents a methodology that avoids many of these problems.

This demonstration will illustrate what synchronous languages have to offer for the modelling, design, analysis and implementation of systems that comprise hardware and software. The requirement for a system notation that spans hardware and software and which also facilitates combined hardware/software static analyses has become especially acute now that vendors like Xilinx produce devices that contain configurable logic and embedded microprocessors e.g. Virtex-II PRO. This tutorial gives some insights into how we can contain the complexity of co-design and co-verification by leveraging on a simple but powerful semantic basis for system description with practical case studies based on Xilinx's Virtex-II PRO devices. Case studies will include examples of design space exploration by synthesizing hardware or software from the same description; static analysis of system level properties e.g. maximum latency on communication channels and formal verification of safety properties (e.g. bus protocol conformance). We shall relate our system level design and analysis flow to the conventional flows advocated with SystemC [7]. We shall also show how the output of this synchronous system level design methodology can be used in conjunction with other system level tools e.g. implementing communication via interface synthesis by exploiting the power of CoWare's N2C tool. We also show how we have used a synchronous approach to model IP blocks at a high level of abstraction. These models have proved to be useful for rapid design space exploration and verification at a system level without requiring detailed implementation and slow bit-level event-based simulation.

Finally we show how the Esterel technique fits into a conventional system level design flow based on Xilinx's Virtex-II PRO FPGAs and present several case studies and one actual demonstration of a complete system from concept to implementation. This will be illustrated on real hardware during the interactive demonstration.

For future work we have started the process of repeating these static property checks using Sugar (with IBM's FoCs system) and VERA with a view to writing a comparative study of the pros can cons of each approach. Sugar-based systems could be used indirectly in the Esterel flow by compiling synchronous observers into rules (which reside in a separate file from the VHDL design) and then using the generated VHDL with vendor software for performing static and dynamic analyses of Sugar-based assertions. This would allow Esterel safe charts to act as graphical front end for some types of Sugar assertions.

For the verification of CoreConnec™ [4] protocols we are developing Esterel models for the PLB (fast complex

1530-1591/04 $20.00 (c) 2004 IEEE

64-bit system bus), OPB (a peripheral bus used as a 32-bit bus in Xilinx IP and used as the main bus for Xilinx's soft processor) and DCR (a simpler device control register bus) components which can then be used to help write synchronous observers for IP blocks without replicating the functionality of the system arbiters in each observer. Previous work using the model checker Rulebase [1][2] for proving properties about CoreConnect arbiters suggests that this approach should be feasible.

The demonstration uses the Esterel Studio system's built-in model checker (based on Prover-SL from Prover Technology) which can be used to try and *prove* such properties. We use the latest V7 version of the Esterel language which allows us to reason about data as well as control which is an improvement from previous versions of the language. We configured the model check to see if the error signal corresponding to a bad state being entered is ever emitted i.e. might the circuit take longer than two clock ticks to acknowledge a transfer? It took Esterel Studio less than two seconds on a Pentium III 500MHz based PC to prove that this signal is never emitted.

Next we produced a deliberately broken version of the peripheral which did not acknowledge read requests. Within two seconds the software was able to prove that there is a case when the acknowledge signal is not asserted after a transaction and provided a counter-model and VCD file.

A conventional approach to catching such approach bugs involves either simulation (which has poor coverage) or the use of bus monitors which snoop the bus at execution time looking for protocol violations. A failure to acknowledge a transaction is one of the types of bugs that such systems can be configured to catch. However, it is far more desirable to catch such problems with a static analysis. We are currently trying to convert a list of around 20 such bug checks used in a commercial OPB bus monitor into a collection of Esterel synchronous observers to allow us to check peripheral protocol conformance with static analyses.

The approach of using Esterel to produce hardware and software seems to show some promise. Initial experiments show that serviceable hardware and software can be produced and implemented on real hardware and embedded processors.

There are some refinements that need to be made to the Esterel language to properly support hardware description. Most of these requirements are easily met without upsetting the core design of the language. Examples include a much more flexible way of converting between integers and bit-vectors and to allow arbitrary precision bit-vectors. Currently performing an integer-based address decode for a 64-bit bus is possible in Esterel but one has to process the bus in chunks not larger than 31 bits.

An appealing aspect of this flow is the ability to write assertions in the same language as the system specification. This means that engineers do not need to learn yet another language and logic. Furthermore, the formal nature of Esterel's semantics may help to make static analysis easier. Our initial experiments with using the integrated model checker are certainly encouraging. However, we need to design and verify more complex systems before we can come to a definitive conclusion about this promising technology for the design and verification of hardware and software from a single specification.

A very useful application of this technology would be to task-based dynamic reconfiguration. This method would avoid the need to duplicate implementation effort and it would also allow important properties of dynamic reconfiguration to be statically analysed to ensure that reconfiguration does not break working circuits.

To investigate how feasible it is to make hardware/software trade-offs using this flow we are developing implementations for network-on-chip protocols which are implemented in a combination of hardware and software. Using this flow can experiment with what portions need to be in hardware for performance and we may also be able to perform interesting static analyses of protocol behaviour, performance and correctness.

"Virtex-II" is a trademark of Xilinx Inc. "CoreConnect" is a trademark of IBM. We would like to thank the staff at Esterel Technologies for their generous assistance during this project.

References

[1] I. Beer, S. Ben-David, C. Eisner, D. Geist, L. Gluhovasky, T. Heyman, A. Landver, P. Paanah, Y. Rodeh, G. Ronin and Y. Wolfsthal. Rulebase: *Model Checking at IBM*. Conference on Computer Aided Verification. CAV'97. 1997.

[2] I. Beer, S. Ben-David, C. Eisner and A. Landver. Rulebase: *An Industry-Oriented Formal Verification Tool*. The 33rd Design Automation Conference. 1996.

[3] Gérard Berry. *The Foundations of Esterel*. Proof, Language and Interaction: Essays in Honour of Robin Milner, G. Plotkin, C. Stirling and M. Tofte, editors, MIT Press, 1998.

[4] IBM. *The CoreConnect™ Bus Architecture*. http://www.chips.ibm.com/product/coreconnect/docscrcon_wp.pdf, 1999.

[5] Gérard Berry and Ellen Sentovich. *Multiclock Esterel.* Correct Hardware Design and Verification Methods. CHARME 2001.

[6] G. Kahn. *Coroutines and networks of parallel processes*. Information Processing, North-Holland Publishing Company, 1977.

[7] SystemC. http://www.SystemC.org