Hardware and Software Synthesis, Optimization, and Verification from Esterel Programs

Gérard Berry

Ecole des Mines de Paris and INRIA
2004 Route des Lucioles
06565 Sophia-Antipolis, France
berry@cma.inria.fr

Abstract. The Esterel synchronous programming language is dedicated to hardware or software reactive systems. The constructive semantics determines the reaction of a program to an input event. Esterel programs can be implemented in several ways that yield different time / space tradeoffs. For all the implementations, the code is composed of a control finite-state machine that drives data-handling actions. The FSM can be explicit or implicit. Implicit machines are Boolean circuits that may contain cycles. The cycles are analyzed and removed using BDD-based technique. Optimization techniques consist in register removal and logic optimization techniques; they can be tailored to both hardware and software targets. Verification of Esterel programs is based either on synchronous observers and symbolic reachability techniques or on explicit or implicit bisimulation reduction.

1 The Esterel Language

Esterel is a deterministic concurrent imperative language dedicated to software or hardware reactive systems. It is being used in process control, robotics, supervision, embedded systems, communication protocols, hardware glue logic, MMI drivers, and more generally for control-dominated reactive applications. The language is described in [2, 3, 4] and in the Esterel v5 system documentation. The main primitives are signal broadcasting, sequencing, concurrency, and preemption constructs.

An Esterel programs reacts to a sequence of inputs by producing an output for each input. A reaction is considered to be instantaneous, which means that the bookkeeping necessary to perform a reaction is viewed as not consuming input time. The mathematical semantics is based on a deterministic zero-delay model similar to that of digital circuits. Unlike most other circuit or synchronous formalisms, Esterel makes it possible for programs to contain static combinational (zero-delay) cycles provided that these cycles are dynamically sound. The constructive semantics exactly characterizes sound programs. The mathematical theory of Esterel is presented in [1].

2 The Execution Structure

The Esterel v5 compiler separates the control and data part of an Esterel program. The data part is implemented either by a data path in hardware or by standard data-handling procedures in software. The control part is a finite-state machine that can be implemented explicitly or implicitly. Explicit automata are very fast but they are limited to small or medium-size examples since they can be exponential in the size of the source programs. Implicit automata are sequential Boolean circuits composed of combinational operators and registers.

An acyclic control circuit is handled as usual by topological sorting. For a cyclic circuit, the Esterel interpretor uses a linear-time algorithm to compute whether the circuit is sound (constructive) for a given input. The compiler uses a BDD-based algorithms to symbolically computes whether the circuit is constructive for all input sequences [8, 6]. In this case, the cyclic circuit can be replaced by an equivalent acyclic one.

3 Optimization

We have developed original optimization algorithms for efficient hardware or software generation from Esterel circuit format. The first step consists in removing redundant registers, using BDD-based reachability analysis techniques [9]. This step must be carefully controlled because removing too many redundant registers can make the combinational logic explode. It is implemented using the TiGeR BDD package developed by O. Coudert, J-C. Madre, and H. Touati and property of Digital Equipment Corp. The second step consists in performing combinational logic optimization, with speed criteria for hardware and area criteria for software. It is implemented as a script for Berkeley's SIS system [7]. Optimization can be conducted in a hierarchical way for large programs.

4 Verification

There are two main methods to verify properties of Esterel programs. The first method is based on the notion of a synchronous observer introduced in [5]. A synchronous observer is an Esterel program placed in parallel with the program of interest and listening to its inputs and outputs. The observer emits a particular output `BUG` if an anomaly shows up. The verification consists in showing that `BUG` can never be emitted and in producing a counter-example otherwise. We use very efficient reachability analysis techniques tailored for control-dominated programs and implemented in TiGeR. The second method is the classical bisimulation reduction method that consists in computing a reduced image of the finite state machine, hiding what is not relevant to the property of interest. Both explicit and implicit bisimulation are available in the Esterel verification system Xeve.

5 Tool Distribution

The Esterel system, the optimizers, and the verification tools are available at `http://www.inria.fr/meije/meije-eng.html`.

References

1. G. Berry. *The Constructive Semantics of Esterel.* available on the Web at address http://www.inria.fr/meije/esterel/Documentation/main-papers.html, 1996.
2. G. Berry and G. Gonthier. The Esterel synchronous programming language: Design, semantics, implementation. *Science Of Computer Programming*, 19(2):87–152, 1992.
3. F. Boussinot and R. de Simone. The Esterel language. *Another Look at Real Time Programming, Proceedings of the IEEE*, 79:1293–1304, 1991.
4. N. Halbwachs. *Synchronous Programming of Reactive Systems.* Kluwer, 1993.
5. N. Halbwachs, F. Lagnier, and C. Ratel. Programming and verifying critical systems by means of the synchronous data-flow programming language Lustre. *IEEE Transactions on Software Engineering*, Special Issue on the Specification and Analysis of Real-Time Systems, September 1992.
6. S. Malik. Analysis of cyclic combinational circuits. *IEEE Trans. Computer–Aided Design*, 13(7):950–956, 1994.
7. E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, R.K. Brayton, and A.L. Sangiovanni-Vincentelli. Sis: A system for sequential circuit synthesis. Technical report, University of California at Berkeley, May 1992.
8. T. Shiple and G. Berry. Constructive analysis of cyclic circuits. In *Proc. International Design and Test Conference ITDC 96, Paris, Franc e*, 1996.
9. H. Toma, E. Sentovich, and G. Berry. Latch optimization in circuits generated from high-level descriptions. In *Proc. International Conf. on Computer-Aided Design ICCAD'96*, 1996.