Einsterel: A Dynamically Scheduled Compiled Event-Driven Simulator for Esterel

Michael Halas
Columbia University
International Business Machines

Vimal Kapadia
Columbia University
International Business Machines

Abstract

The performance of compiled Esterel code is suboptimal. Prior work has been done to improve the performance of compiled Esterel, but more work needs to be done. This paper presents a high-performance compiled event-driven simulator for the Esterel language that builds on prior work in this area.

ESUIF is used as the front-end to parse the Esterel, The intermediate output of ESUIF is taken as input to the event driven simulator. The events are scheduled at run-time, as opposed to other Esterel compilers, which scheduled at compile-time.

Fill results in here.

Introduction

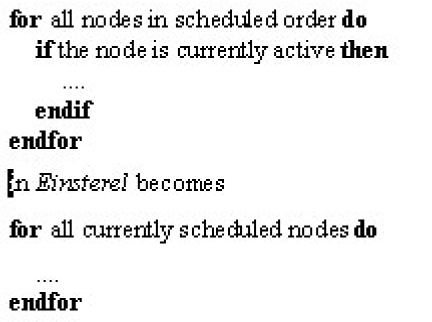
We propose the first compiled event-driven simulator that schedules events at run-time for Esterel code (are we the first?), Einsterel. Some aspects of EVCF (Event-Driven Conditional-Free) simulation, introduced by Maurer [4], are used for performance.

EVCF offers improved performance over other event-driven simulation techniques, by avoiding loops and conditionals. Essentially, there are 2 types of statements: assignments and computed gotos. A virtual function table of function addresses is used instead of a type code, to distinguish the node type. Function addresses are branched to directly, avoiding decoding of the type code [4].

Edwards [2] wrote ESUIF, an open Esterel compiler built on the SUIF 2 system. The ESUIF front-end builds a very high-level abstract syntax-tree-like representation of the source Esterel program [2]. Because the front-end is separate, it allows for different compilation paths, making it a good front-end for Einsterel, as well.

In Esterel, statements can be executed more than once in a cycle. Statements also can be executed in a different order, based on the inputs. This is a problem with Esterel that not all Esterel compilers can overcome. Einsterel can handle this because it dynamically schedules the events as opposed to scheduling them at compile time such as Bertin et al's [1] work . Being event-driven also enables it to skip work on threads that are waiting for a signal, since they do not have to be scheduled until that signal changes. The work by Bertin, et al [1] and any statically scheduled compilers suffer from many of the limitations stated above.

However, aspects of static scheduling can be incorporated into a dynamically scheduled event driven simulator. Continuous assignment optimizations [French 3] can be done to treat separate events as one event. By combining the events, we can reduce the number of events needed to be scheduled.



```
for all nodes in scheduled order do
   if the node is currently active then
      ....
   endif
endfor
```

In *Einsterel* becomes

```
for all currently scheduled nodes do

   ....
endfor
```

Fig 1. Comparison of basic loop structure in SAXO-RT Esterel Compiler vs Einsterel

References

1. BERTIN, V., POIZE, M., AND PULOU, J. 1999. Une nouvelle m´ethode de compilation pour le language ESTEREL [A new method for compiling the Esterel language]. In Proceedings of GRAISyHM-AAA. (Lille, France, March 1999).
2. Stephen A. Edwards, "ESUIF: An Open Esterel Compiler in Proceedings of Synchronous Languages, Applications, and Programming (SLAP)," *Electronic Notes in Theoretical Computer Science (ENTCS)* 65(5), April 2002
3. FRENCH, R. S., LAM, M. S., LEVITT, J. R., AND OLUKOTUN, K. 1995. A general method for compiling event-driven simulations. In *Proceedings of the 32nd Design Automation Conference* (San Francisco, California, June 1995). pp. 151–156.
4. Peter M. Maurer: Event Driven Simulation Without Loops or Conditionals. ICCAD 2000: 23-26