Suppose we want to add hardware support for separation to a stock embedded processor design, allowing safe interleaving of processes handling classified and unclassified data.

Where does the new “separation module” go?

Problem: Original design is modular in a structural sense, but not in a semantic sense.

Example at Scale: PicoBlaze from Xilinx

We have developed an MMS-style specification for the PicoBlaze soft microcontroller from Xilinx. The MMS semantics corresponds nicely to the informal documentation.

The core technique here is modular monadic semantics (MMS).

Challenge: Compilation

We need an expressive functional language to support MMS. We choose Haskell.

When it comes to synthesis, however, Haskell has many features that are hard to translate directly to gates.

- General recursion, recursive data types, higher-order functions...

Solution: Partial Evaluation

We have implemented a prototype compiler called ReWire that translates MMS specifications written in Haskell into VHDL suitable for use on FPGAs.

The core technique here is partial evaluation, a program transformation technique that works by performing as much evaluation at compile time as possible.

Partial evaluation is effective at eliminating language constructs that cannot be directly translated to hardware, producing a normal-form program that can easily be translated into a finite state machine.

Case Study

Paper discusses the synthesis of a very simple processor design in MMS style, with a tiny instruction set (four instructions), two general-purpose registers, an external program memory, and an instruction decoder.

As compiled by ReWire, this processor design utilizes 115 logic slices on a Spartan-3E series FPGA. Detailed usage statistics for a Spartan-3E XC3S500E FPGA, speed grade 5, are as follows. Maximum clock rate on this particular chip is around 133MHz.

<table>
<thead>
<tr>
<th></th>
<th>Used</th>
<th>Hardware</th>
<th>Slice %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slices</td>
<td>115</td>
<td>906</td>
<td>4.47%</td>
</tr>
<tr>
<td>Slice Flip Flips</td>
<td>68</td>
<td>9132</td>
<td>0.52%</td>
</tr>
<tr>
<td>Input Shifts</td>
<td>251</td>
<td>3132</td>
<td>2.20%</td>
</tr>
</tbody>
</table>

Ongoing Work

Apart from the benefits in extensibility, monadic semantics offers a powerful basis for formal verification, namely equational reasoning.

Ongoing work involves adapting existing techniques by several of the authors, previously used to verify monadic security kernels implemented in software, to prove separation properties of hardware circuits.

Further reading:


This research was supported by the U.S. Department of Education under GAANN grant number P200A100053.