Language Issues in Designing Future Systems

Stephen A. Edwards, Columbia

Sean Gibb, Calgary

Erwan Fabiani, UBO

William Jiang, Berkeley

New vs. Existing Languages

Many reasons to both stick with existing languages and adopt new ones.

Hypothesis: A compelling algorithm warrants a new language.

Why Stay with an Existing Language?

Momentum. Many people know an existing language.

Extensive infrastructure: tools, books.

Compatibility: Who in the room knows Esparanto?

Why learn something new when my next employer expects the old stuff?

Why Create a New Language?

Blank slate: can create the world in your image.

Chance to get the semantics “right,” to support new models of computation.

Chance to support higher levels of abstraction (e.g., communication).

Chance to get away from historical implementation details (e.g., word width in C).

Ways to Evolve a Language

- Libraries (Java, C++ STL)
  - Least change, easiest for users
  - Does not support new semantics
- Helper language (RPC, CORBA/COM)
  - More user difficulty, but usually less impact
  - Can only “add” semantics, not modify them.
- Restriction w/new semantics (RTL Verilog/VHDL, SystemC)
  - Even more user difficulty
  - Add new semantics, but can cause confusion
- New syntax and semantics (DE Verilog/VHDL)
  - Most user difficulty
  - Maximum flexibility

When Languages Evolve

Java: Automatic garbage collection required object references instead of pointers.

C: Efficient Translation of expressions to PDP-8 assembly

C++: Efficient use of existing C compilers

Perl: Common text-processing tasks easy, good library support

DE Verilog: Created as input for a good discrete event simulator.

RTL Verilog: Coopted for use as synthesis specification

SystemC: Uses a thread package to simulate concurrency, classes for grouping of local signals.

Where to Look for New Languages

Places where new algorithms or methodologies arise:

- Language with support for IP-based design
- Language for behavioral synthesis
- Language for what's beyond behavioral synthesis
- For your nifty new algorithm