System-Level Description Languages

Andrew Mihal
EE249 Fall 1999
Project Presentation
4 December 1999
Outline

- SLDLs and their features
- SLDL classification
  - C++ based SLDLs
  - HDL based SLDLs
- Related work
- Conclusions
What is an SLDL?

- A design language for describing systems
  - Design Entry
  - Simulation
  - Synthesis
  - Verification
- High-level embedded systems design tool
  - Increasing system complexity
  - Designer productivity gap
SLDL Features

- Composeable / Hierarchical
- Multilevel Abstraction / Multilevel Refinement
- Requirements Specification and Assignment
- Constraints Propagation
- Concurrency and Communication
- Computation
- Implementation Independence
Designing an SLDL

- Start from scratch
  - Language design is an art form
  - Nobody wants to learn new languages
  - Can’t use existing tools
- Adapt an existing language
  - Strong user base
  - Lots of tools
  - Existing languages are lacking in features
SLDL Classes

- **C/C++ Based**
  - **PROS**
    - Hierarchical
    - Good Data Modeling
    - Software Components
    - Lots of tools / debuggers
  - **CONS**
    - No Concurrency
    - Not Reactive
    - No bit-level data
    - HW synthesis

- **HDL Based**
  - **PROS**
    - Concurrent
    - Reactive
    - Lots of IP Available
    - HW Synthesis
  - **CONS**
    - Low-level data modeling
    - Not Dynamic
    - Low-level timing
    - No Constraints modeling
Scenic (SystemC)

- Developed at UC Irvine and Synopsys
- Add features to C++ using base classes, without changing the syntax or requiring new compilers
- Base class for concurrent processes
- Lambda Functions for reactivity
  - `wait_until` and `watching`
  - `wait_until(start == '1');`
- Uses C++ exceptions
- Templates for bit-level data types and math
Scenic (SystemC)

- No constraints modeling
- Manual partitioning
- Behavioral synthesis is syntax-specific
  - Line-by-line literal translation
- Limits caused by C++ exceptions
  - What happens when two events occur very close?
Suave

- P. Ashenden (Univ. of Adelaide), P. Wilsey, D. Martin (Univ. of Cincinnati)
- Add Ada-95 object-oriented features to VHDL
- Functional superset of VHDL
- Abstract data modeling
  - Abstract types and inheritance
  - Data encapsulation
  - Does not interfere with packages or entities
Suave

- No constraints modeling
- Abstract data types as signals
  - Sharing access values between processes
- Levels of abstraction for Entities are still limited
  - Behavioral and Structural
  - How to refine an architecture?
- Vista OO-VHDL language applies object-oriented features to entity/architecture interface
Constraints Modeling - VSpec

- Allow designers to model additional "facets" of a system
  - Regular VHDL: Behavioral and Structural
  - Functional Requirements, Performance Constraints
- Add clauses to the entity declaration
  - sensitive to
  - requires
  - ensures
  - constrained by

```vhdl
entity foo is ...
    constrained by
    area <= (3um x 5um)
    and power <= 10mW
    and clock_freq <= 50MHz;
end foo;
```
Constraints Modeling - Rosetta

• Developed for the System-Level Design Language Initiative
• Independent constraints specification language
• Declare "facets" as objects that model abstractions of a system
• Algebra for the composition and verification of constraints
• One unified language is not appropriate for system-level design
Related Work

- **POLIS**
  - Based on Esterel, a synchronous reactive language
  - Esterel-C extends Esterel with C’s data modeling features
  - Object-oriented data modeling features for Esterel?
- **SDL**
  - ITU Standard, uses concurrent processes and non-blocking communication
  - Used in rapid prototyping frameworks such as REAR
  - Object-oriented data features under consideration
Conclusions

- Extend an existing domain-specific language
  - Add features required for an SLDL
  - Try not to change the syntax or semantics
- Constraints and requirements are usually left out
- New features interfere with underlying semantics
  - Scenic: Reactivity and C++ exceptions
  - SUAVE: ADTs as signals and concurrency
- Quality of design tools is deciding factor