Reuse Through Genericity in SUAVE

Peter J. Ashenden
University of Adelaide

Philip A. Wilsey, Dale E. Martin
University of Cincinnati

This work was partially supported by Wright Laboratory
under USAF contract F33615-95-C-1638.

SUAVE

SAVANT and
University of
Adelaide
VHDL
Extensions

Outline

- Design Objectives
- Overview of Extensions
  - genericity

Design Objectives

- Support re-use and incremental development
  - polymorphism, dynamic binding, type genericity
- Preserve capability for synthesis & other analysis
- Support hw/sw codesign
  - improved integration with programming languages
- Preserve correctness of existing models
- Design principles from VHDL-93
  - preserve “conceptual integrity”

Existing Generics

- Generic clause on entity, component, block
  - specifies generic constants
- Used for
  - operational parameters (eg, timing)
  - bounds for array ports

```
entity mux is
        generic ( Tpd : time;
                  width : positive;
                  trace : boolean := false);
        port ( d0, d1 : in bit_vector(0 to width - 1);
               . . . );
end entity mux;
```


Extended Genericity

- Object-orientation is not a panacea
  - OO extension meets many, but not all, objectives
  - Doesn't include type genericity needed for reuse
- Adopt generics from Ada-95
  - modified to integrate with VHDL generics
  - formal types, subprograms, packages
  - allow generic clause in subprograms and packages
- Instantiation done at elaboration-time

Formal Types in Entities

```
entity generic_multiplexer is
      generic ( type data_type is private );
      port ( control : in bit;  in0, in1 : in data_type;
                  data_out : out data_type );
end entity generic_multiplexer;

architecture data_flow
```