VHDL-II
Structural Modeling

ECE-331, Digital Design
Prof. Hintz
Electrical and Computer Engineering
Variables

- **Exist Only Within an Architecture**
  - Values of variables cannot be passed to other entities except through signals

- **Variables Change Value When They Are Evaluated.**
  - Signals change at a “later” time
Signals

- Entities are Interconnected by Signals
  - Process executions result in new values being assigned to signals which are then accessible to other processes
  - A signal may be accessed by a process in another architecture by connecting the signal to ports in the entities associated with the two architectures
Signals

- Signals Can Be Declared **Internal to an Architecture** to Connect Internal Entities
- Variables Are Not Appropriate Since They Do Not Have the **Temporal Characteristics** of Hardware
- Signals Declared Within an Entity **Are Not Available to Other Entities Unless Specified in the Port Clause** of the Entity Declaration.
Entity Syntax

```text
entity identifier is

[ port ( port_interface_list ); ]

{ entity_declarative_item }

end [ entity ] [ identifier ] ;
```
Entity Syntax

\[
\text{port\_interface\_list} \leq ...
\]

\[
\begin{align*}
( \text{identifier} \{ , \ldots \} : \\
[ \text{mode} ] \text{ subtype\_indication} \\
[ := \text{expression} ] ) \{ ; \ldots \}
\end{align*}
\]

\[
\text{mode} \leq \text{in} | \text{out} | \text{inout}
\]
entity NiCadCharger is

port ( Voltage, Current : in real := 0.0 ; AC : in bit := ‘1’ ; Charged, Recharge : out bit );

end entity NiCadCharger ;
Architecture Syntax

architecture identifier of
entity_name is

{ block_declarative_item }

begin
{ concurrent_statement }
end [architecture][ identifier ];
Structural Model

A Representation of a System in Terms of the Interconnections of a Set of Defined Components.

- Components can be described either structurally or behaviorally
- Smallest components are behavioral entities
- Components usually stored in libraries
Structural Models

Components Can Be **Instantiated** As **Concurrent Statements** in Architectures

- If architecture not specified in statement
  - Must be specified later, or
  - Most recently analyzed architecture used
- **Ports** can be specified **two ways**
  - **Positional** association
  - **Named** association
Entity Ports Which are Declared within an Architecture Body Are Local Signals
- These signals are not available outside the architecture unless connected to one of the architecture’s ports
Odd Parity Generator Example
entity Odd_Parity is

port(
    IN_1, IN_2, IN_3 : in bit;
    Out_1 : out bit);
end entity Odd_Parity ;
Odd Parity Behavior Architecture

architecture Odd_Parity_B of

    Odd_Parity is

        $f_{odd}(A,B,C) = AB\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C}$

    begin

        Out_1 <= ( IN_1 and not IN_2 and IN_3 )
        OR ( not IN_1 and not IN_2 and not IN_3 )
        OR ( not IN_1 and IN_2 and IN_3 )
        OR ( IN_1 and IN_2 and not IN_3 )

    end architecture Odd_Parity_B ;
INV Entity/Architecture

```vhdl
entity INV is
  port(
    In_1    : in  bit ;
    In_1_Bar : out bit );
end entity INV ;

architecture INV_B of INV is
begin
  In_1_Bar <= not IN_1 ;
end architecture INV_B ;
```
entity AND_3 is
  port(
    IN_1, IN_2, IN_3 : in bit ;
    Out_1 : out bit );
end entity AND_3 ;
architecture AND_3_B of AND_3 is
begin
  Out_1 <= IN_1 and IN_2 and IN_3 ;
end architecture AND_3_B ;
entity OR_4 is
  port(
    IN_1, IN_2, IN_3, IN_4 : in bit ;
    Out_1 : out bit );
end entity OR_4 ;
architecture OR_4_B of OR_4 is
begin
  Out_1 <= IN_1 or IN_2 or IN_3 or IN_4 ;
end architecture OR_4_B ;
Odd Parity Structural Arch.

```vhdl
architecture Odd_Parity_S of 
  Odd_Parity is 
  --block_declarative_items 
  --components 
  component INV is 
    port( 
      In_1 : in bit ; 
      In_1_Bar : out bit ); 
  end component INV ; 
```
Odd Parity Structural Arch

component AND_3 is
  port( IN_1, IN_2, IN_3 : in bit ;
       Out_1 : out bit );
end component AND_3 ;

component OR_4 is
  port( IN_1, IN_2, IN_3, IN_4 : in bit ;
        Out_1 : out bit );
end component OR_4 ;
Structural Mapping

\[ f( A, B, C ) \]
Odd Parity Structural Arch

--block_declarative_items
--internal signals
signal MT_0, MT_3, MT_5, MT_6 : bit;
signal INV_1, INV_2, INV_3 : bit;
begnin --parity structural architecture
--connect gates
G1: INV port map ( In_1, INV_1 );
G2: INV port map ( In_2, INV_2 );
G3: INV port map ( In_3, INV_3 );
Odd Parity Structural Arch

G4: AND_3 port map
    ( IN_1, INV_2, IN_3, MT_5 );
G5: AND_3 port map
    ( INV_1, INV_2, INV_3, MT_0 );
G6: AND_3 port map
    ( INV_1, IN_2, IN_3, MT_3 );
G7: AND_3 port map
    ( IN_1, IN_2, INV_3, MT_6 );
Odd Parity Structural Arch.

G8: OR_4 port map
    ( MT_0, MT_3, MT_5, MT_6, Out_1 );
end architecture Odd_Parity_S ;
Packages

- Method for Grouping Related Declarations Which Serve a Common Purpose
  - Set of subprograms to operate on particular data type
  - Set of declarations for particular model
  - Allows declaration of “global” signals, e.g., clocks.
Packages

- Design Unit Similar to Entity Declarations and Architecture Bodies
  - Can be put in library and made accessible to other units
  - Access to items declared in the package is through using its Selected Name
    - library name . package name . item name
  - Aliases can be used to allow shorter names for accessing declared items
Packages

- Two Components to Packages
  - Package declaration
  - Package body
    - Not necessary if package declaration does not declare subprograms
Package Declaration

- Declares
  - Subprograms using header, implementation is hidden
  - Constants, do not need to be initialized in declaration
  - Types, must be completely specified
    - Can have variable size arrays
  - Signals must be completely specified
Package Declaration Syntax

```plaintext
package identifier is

  { package_declarative_item }

end [ package ] [ identifier ] ;
```
package dp32_types is

    constant unit_delay : Time := 1 ns;

    type bool_to_bit_table is array (boolean) of bit;

end dp32_types ;
Package Body

- Declared Subprograms Must Include the Full Declaration As Used in Package Declaration
  - Numeric literals can be written differently if same value
  - Simple name may be replaced by a selected name provided it refers to same item
Package Body

- May Contain Additional Declarations Which Are Local to the Package Body
  - Cannot declare signals in body
package body identifier is

{ package_ body_declarative_item }

end [ package body ] [ identifier ] ;
End of Lecture

- Entities
- Architectures
- Packages