CPE 487: Digital System Design

Spring 2018

Lecture 11

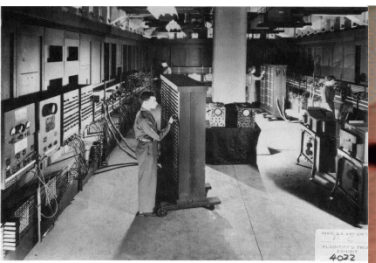
Packages and Libraries

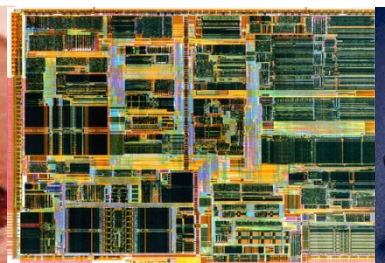
Bryan Ackland

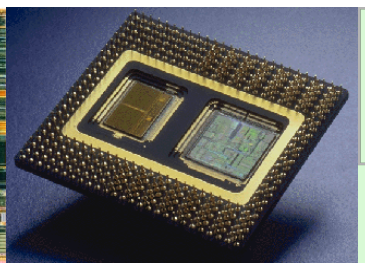
Department of Electrical and Computer Engineering

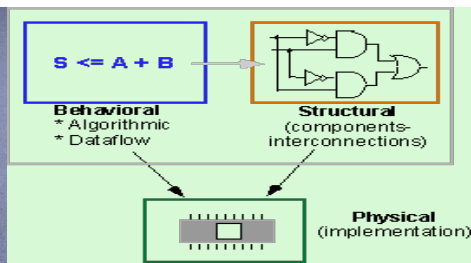
Stevens Institute of Technology

Hoboken, NJ 07030









Packages

- A package provides a convenient mechanism to store and share declarations that are common across many design units.
- Packages may contain:
  - type and sub-type declarations
  - subprogram (function & procedure) declarations
  - constant, signal & variable declarations;
  - component declarations
- Packages consist of
  - A package declaration and, optionally,
  - A package body

Package Organization

- Package Declaration
  - Externally visible portion of package
  - Declaration of the functions, procedures, and types that are available in the package
  - Serves as a package interface

- Items declared in a package declaration can be accessed by other design units using the *library* and *use* clauses.

- Package Body
  - Implementation of the functions and procedures declared in the package header
  - Instantiation of constants provided in the package header
  - Can be hidden from user of package

Package Declaration Example: std_logic_1164

```
package std_logic_1164 is
    type std_ulogic is ( 'U',    --Unitialized
                         'X',    -- Forcing Unknown
                         '0',    -- Forcing 0
                         '1',    -- Forcing 1
                         'Z',    -- High Impedance
                         'W',    -- Weak Unknown
                         'L',    -- Weak 0
                         'H',    -- Weak 1
                         '-'     -- Don't care
                         );
    type std_ulogic_vector is array (natural range <>) of std_ulogic;
    function resolved (s : std_ulogic_vector) return std_ulogic;
    subtype std_logic is resolved std_ulogic;
    type std_logic_vector is array (natural range <>) of std_logic;
    function "and" (l, r : std_logic_vector) return std_logic_vector;
    -- <other function declarations>
end package std_logic_1164;
```

Package Body

```
package body my_package is
--
-- type definitions, functions, and procedures
-- constant values
--
end my_package;
```

- Package body contains the behavior of the subprograms and the values of deferred constants declared in a package declaration.
- The package name must be the same as the name of its corresponding package declaration.
- Body of std_logic_1164 is not visible to us

Package Example

- VHDL defines an arithmetic right shift operator SRA for bit_vectors, e.g.:

```
result <= data sra n;
```

result and data are of type bit_vector, n is an integer

- Make a package that extends and overloads the sra operator to allow arithmetic right shifts on objects of type std_logic_vector
- Package declaration:

```
package std_shift is

function "sra" (word: in std_logic_vector; nshift: in integer)
        return std_logic_vector;

end std_shift;
```

Package Body Example

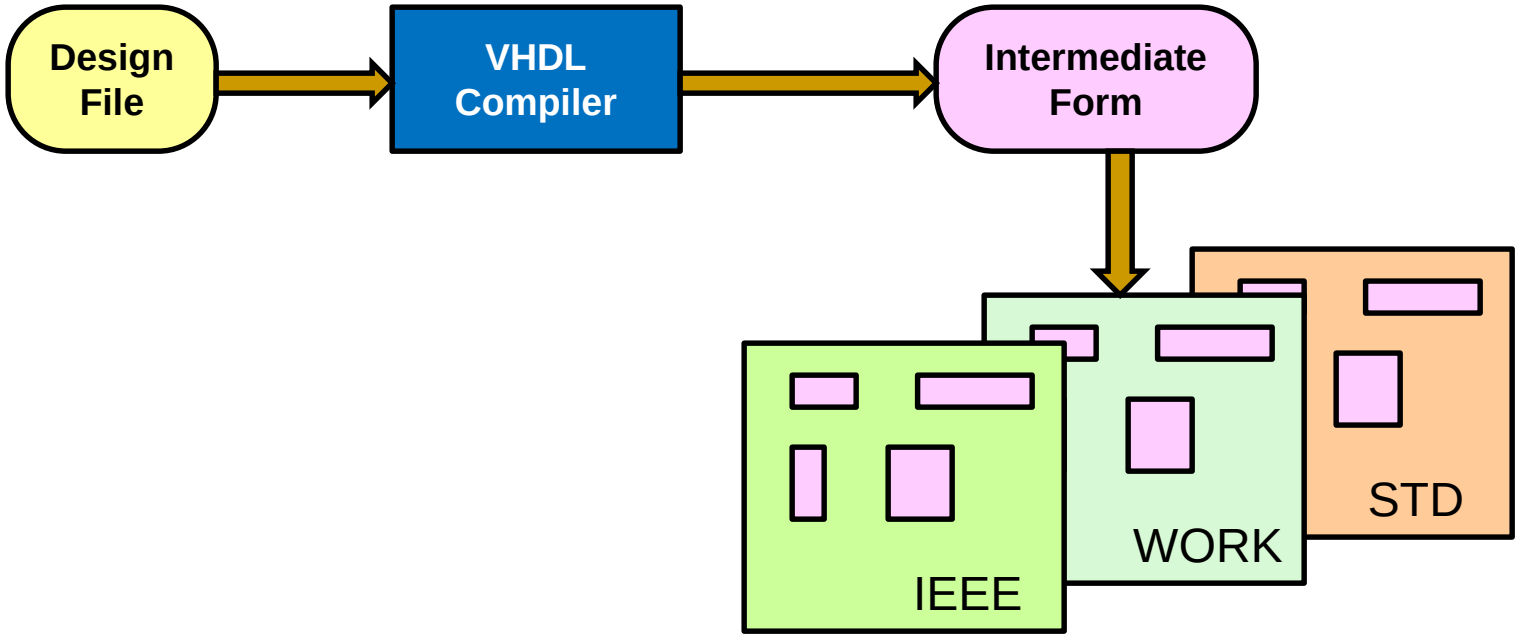
```
package body std_shift is
    function "sra" (word: in std_logic_vector; nshift: in integer)
                return std_logic_vector is
    variable v: std_logic_vector (word'range);
    begin
        v:=word;
        if v'ascending then
            for i in 1 to nshift loop
                v:= v(v'left) & v(v'left to (v'right-1));
            end loop;
        else
            for i in 1 to nshift loop
                v:= v(v'left) & v(v'left downto (v'right+1));
            end loop;
        end if;
        return v;
    end "sra";
end std_shift;
```

Design Unit

- A design file is an ASCII file containing a VHDL source.
- A design file can contain one or more design units, where a design unit is one of:
  - Entity declaration *(primary)*
  - Architecture body *(secondary)*
  - Configuration declaration *(primary)*
  - Package declaration *(primary)*
  - Package body *(secondary)*
- Each design unit may be placed in a separate file or there may be multiple units in each file

Compilation Process

- After checking for syntactic and semantic correctness, each design unit present in a file is compiled into an intermediate form
- Intermediate forms are stored in a working library – normally called WORK



Libraries

- Libraries are generally implemented as directories (folders)
- Always two libraries that are implicitly declared (i.e. contents are visible and available for use):
  - STD (standard packages *standard.vhd* and *textio.vhd* provided with VHDL distribution)
  - WORK (library into which your code is compiled)
- Other libraries (e.g. IEEE) and their packages (e.g. std_logic_1164) may be declared (i.e. contents made visible) using the library and use clauses

Implicit Visibility

- All items declared in WORK and STD libraries are visible to your code
- Secondary design units implicitly inherit all declarations of their associated primary unit
  - An architecture body implicitly inherits all declarations in the entity since it is tied to that entity by virtue of the statement:

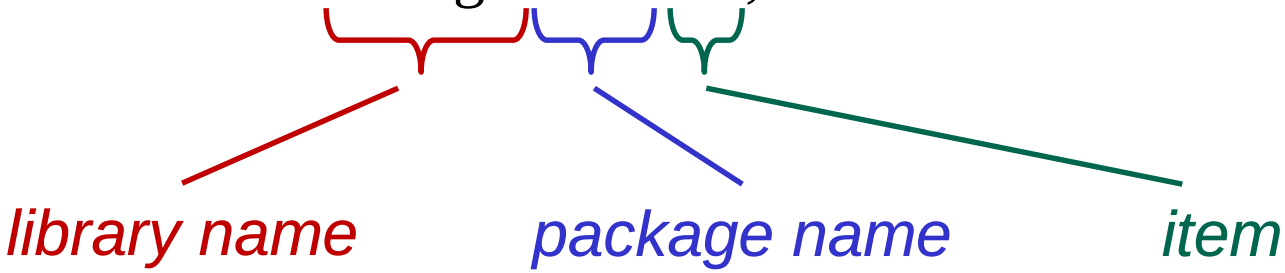
    architecture architecture-name **of** entity-name **is**…
  - A package body implicitly inherits all items declared in the package declaration by virtue of its package name:

    package body package-name **is** ….

Explicit Visibility

- Other items (stored in other libraries) are made explicitly visible using library and use clauses:

```
library abclogic;
use abclogic.TTL.all;
```



- or:

```
library abclogic;
use abclogic.nand2;
```

if nand2 is declared at top level in library

(i.e. not within a package)

More examples

```
library CMOS;
use CMOS.NOR2

library IEEE;
use IEEE.STD_LOGIC_1164.STD_LOGIC;
-- std_logic is a type declared in the std_logic_1164 package.
-- the package std_logic_1164 is part of the design library IEEE.

entity NAND2 is
      port (A, B: in STD_LOGIC;…) ….

use IEEE.STD_LOGIC_1164.all;
-- make all items declared in package STD_LOGIC_1164 in design
library IEEE visible

use IEEE.all;
-- make all items declared in library IEEE visible
```

Visibility Rules

```
library IEEE;
use IEEE.std_logic_1164.all;
entity design_1 is
….

library IEEE;
use IEEE.std_logic_1164.std_logic;
entity design_2 is
….
```

- When multiple design units are in the same file, visibility of libraries and packages must be established for each ***primary*** design unit (entity, package declaration, configuration) separately!
  - Secondary design units (architecture body and package body) derive library information from associated primary design unit
- The **use** clause may selectively establish visibility, e.g., only the type *std_logic* is visible within entity design-2