Homework 2

- Homework 2 is up on the web site, due 9/16.
- Start early...

VHDL Basics

Chapter 3
Entities, Architectures, and Processes

Review

- Entity and architecture
  - Entity: The interface
  - Architecture: The functionality
- Signals and variables
  - Signals: Time and value
  - Variables: Just a value
- Simulation cycle

Basic Code Structure

Entity Block

```
entity identifier is
  generic (generic_declarations);
  port (port_declarations);
end identifier;
```

Identifiers

- A sequence of one or more characters (a-z, A-Z, 0-9, _).
- The first character should be a letter. Cannot have two consecutive underscore characters
- Case insensitive

Example:

- DRIVE BUS
- SelectSignal
- RAM_Address
- en_b
- oe_b

NOTE: Variations of this will be presented in a later chapter
Reserved Words

abs access after alias all and architecture array assert attribute begin block body buffer bus case component configuration constant disconnect downto else elsif end entity exit file for function generate generic guarded if in inout is label library linkage loop map mod null new next not null of on open or others out package port procedure process range record register rem report return select severity signal subtype then to transport type units until use variable wait when while xor

Comments

-- Two adjacent dashes form a comment.
C <= B or A; -- Comments stop the parsing
-- until the end of the line.

A Note on Style

- VHDL is case insensitive
  - Use this to your advantage to improve readability
  - For example:
    - VHDL Keywords - Lower Case
    - User Defined Identifiers - Upper Case
- Spaces/tabs/<CR> usually have no meaning
  - Use to provide pleasing appearance.

Entity

entity identifier is
  generic (generic_declarations);
  port (port_declarations);
end identifier;

entity ONES_CTR is
  port (A : in Bit_Vector(2 downto 0);
       C : out Bit_Vector(1 downto 0));
end ONES_CTR;

Port Signals

identifier : direction signal_type ; next_signal

Built-in types:
- bit
- bit_vector
- integer
- others (defined later)
User-defined types:
- whatever

Architecture Block

An architecture body defines the functionality of a design entity. It specifies the relationship between the inputs and outputs of a design entity, and may be expressed in terms of structure, dataflow, behavior, or a combination of the three.

Format:
architecture architecture_name of entity_name is
  architecture_declarations
begin
  architecture_body
end architecture_name;
Sanity Check

Why separate ENTITY and ARCHITECTURE blocks?

ENTITY

ARCHITECTURE a

ARCHITECTURE b

ARCHITECTURE c

Did you know that ENTITY blocks and ARCHITECTURE blocks can reside in different files?

Architecture Block

architecture architecture_name of entity_name is

architecture_declarations

begin

architecture_body

end architecture_name ;

Architecture Declarations

Common things found in the ARCHITECTURE declaration

- Signal declarations
  - Local signals
    signal FUNNY : Bit;
  - Component declarations
    - Building blocks for a structural model
      component MYCOMPONENT
        port ( A : in Bit; B : out Bit );
      end component;

Signal Declaration

signal identifier_list : signal_type ;

signal identifier_list : signal_type := initial_value ;

EXAMPLE:
signal XCARRY_B, ZCARRY_B : Bit := '0';

Components

Hierachy within a model

component identifier
  generic (generic_declarations);
  port ( interface_port_signal_declarations);
end component;

EXAMPLE:
component NAND2
  generic (DELAY: Time);
  port ( IN1, IN2: in Bit; OUT1: out Bit );
end component;

for all: NAND2 use entity work.NAND2(BEHAV);signal D, E: Bit_Vector(1 downto 0) := "00";
begin
architecture_body
end STRUCTURAL;
**Component Preview**

(for all) NAND2 use entity work.NAND2(BEHAV);

*In English:*  
For all instances of component NAND2 found in the architecture body, use the entity called NAND2, with its corresponding architecture called BEHAV.

We’ll discuss configuration statement in more detail later in the semester.

**Architecture Body**

architectures architecture_name of entity_name is

architecture_declarations

begin

architecture_body

concurrent_statements

end architecture_name;

**Signal Assignment Statements**

- Signal assignment operator: <=
- All concurrent statements happen in parallel
- VHDL is strongly typed:
  - Signal types and sizes on both sides of the <= must be the same.
- Example:
  - A <= B or C after 3 ns;
  - D <= A;
  - F <= ‘1’ after 10 ns;

**Operators**

VHDL provides several predefined operators that may be performed on objects to form an expression.

Logical operators: AND, OR, NAND, NOR, NOT, XOR

Relational operators: =, /=, <, <=, >, >=

Arithmetic operators: +, -, *, /, **, MOD, REM, ABS

Concatenation operator: &

**Concurrent Statements**

Used to describe the internal composition of the entity. All concurrent statements are executed as the input changes, and hence the order of appearance is insignificant.

Concurrent statements are:
- process-statement
- block-statement
- concurrent procedure call statement
- concurrent assertion statement
- concurrent signal assignment statement
- concurrent instantiation statement
- generate statement
Process Statements

Format:
process_label: process (sensitivity_list) constant_or_variable_declarations
begin
    sequential statements
end process process_label;

Execution of a Process Statement:
• Every process statement is executed once at the beginning of the simulation.
• The statements in a process are all executed sequentially at any simulation time when there is an event on any signal in the sensitivity list.

Sequential Statements

Used to describe the internal composition of the entity. All sequential statements are executed in the order in which they occur. Used to describe an algorithm as in sequential languages such as C. Sequential statements are:
- if statement
- case statement
- loop statement
- next statement
- exit statement
- variable assignment statement
- procedure call
- return
- null statement
- assertion statement

Component Instantiation

Defines interconnection of components.

Format:
component_label: component_name port map (association_list);

Example:
architecture MyStructure of ThreeNands is
begin
    component NAND2
        port (A, B: in BIT; Z: out BIT);
    end component;
    BEGIN
        -- Component Instantiation
        N1: NAND2 port map (S1, S2, S3); -- positional association
        N2: NAND2 port map (X, open, Z); -- the second input is open
        N3: NAND2 port map (A => C1, Z => C3, B => C2);
    end;
end

Putting it All Together

entity NAND2 is
    generic (TPHL, TPLH: Time);
    port (DA, DB: in Bit; DZ: out Bit);
end NAND2;

architecture BEHAVIORAL of NAND2 is
begin
    process (DA, DB)
        variable TEMPERATURE: Bit;
    begin
        TEMPERATURE := not (DA and DB);
        if (TEMPERATURE = '0')
            then DZ <= TEMPERATURE after TPHL;
        else DZ <= TEMPERATURE after TPLH;
    end if;
    end process;
end BEHAVIORAL;

entity CIRCUIT is
    port (A, B, C, D: in Bit; DZ: out Bit);
end CIRCUIT;

architecture STRUCTURAL of CIRCUIT is
begin
    signal T1, T2: Bit;
    component AND2
        port (A1, A2: in Bit; Z: out Bit);
    end component;
    component OR2
        port (A1, A2: in Bit; Z: out Bit);
    end component;
    for all: AND2 use entity work.AND2(BEH);
    for all: OR2 use entity work.OR2(BEH);
    begin

Putting it All Together

entity CIRCUIT is
    port (A, B, C, D: in Bit; DZ: out Bit);
end CIRCUIT;

architecture STRUCTURAL of CIRCUIT is
begin
    signal T1, T2: Bit;
    component AND2
        port (A1, A2: in Bit; Z: out Bit);
    end component;
    component OR2
        port (A1, A2: in Bit; Z: out Bit);
    end component;
    for all: AND2 use entity work.AND2(BEH);
    for all: OR2 use entity work.OR2(BEH);
    begin


Summary

- Entities, architectures
- Identifiers, reserved keywords
- Concurrent statements
- Components

- Next time: Testbenches
  - How do you test your VHDL model?
  - Create another VHDL model!
  - Reading: pp. 51-72