CS/EE 3710: Computer Design Laboratory

Lecture 5: Modeling Structure in VHDL

(Chapter 5: Yalamanchili)

Fall Semester 1999

Chris Myers

---

Structural Models

- Digital system is interconnected set of components.
- So far, we know how to describe components using CSA and processes (behavioral models).
- This lecture describes how to interconnect these components in VHDL (structural model).

---

Example: Full Adder Circuit

Figure 5.1

---

Keys to Describe Structure

- A list of components.
- Definitions of signals to interconnect components.
- Unique labels to distinguish multiple copies of the same component.
Example: Structural Model of a Full Adder

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity full_adder is
  port(In1, In2, c_in : in std_logic;
       sum, c_out : out std_logic);
end full_adder;

architecture structural of full_adder is
component half_adder
  port(a,b : in std_logic;
       sum, carry : out std_logic);
end component;

signal s1, s2, s3 : std_logic;
begin
  H1: half_adder port map(a=>In1, b=>In2, sum=>s1, carry=>s3);
  H2: half_adder port map(a=>s1, b=>c_in, sum=>sum, carry=>s2);
  O1: or_2 port map(a=> s2, b=>s3, c=>c_out);
end structural;
```

Example: Structural Model of the Half Adder

```vhdl
architecture structural of half_adder is
component xor2
  port(a, b : in std_logic;
       c : out std_logic);
end component;
component and2
  port(a, b : in std_logic;
       c : out std_logic);
end component;
begin
  EX1: xor2 port map(a=>a, b=>b, c=>sum);
  AND1: and2 port map(a=>a, b=>b, c=>carry);
end structural;
```

Example: Logic Gates

```vhdl
architecture behavior of or2 is
begin
  c <= a or b after 5 ns;
end behavior;
architecture behavior of xor2 is
begin
  c <= a xor b after 5 ns;
end behavior;
architecture behavior of and2 is
begin
  c <= a and b after 5 ns;
end behavior;
```
Hierarchy of Models in Full Adder

```
(full_adder.vhd)

<table>
<thead>
<tr>
<th>or2.vhd</th>
<th>half_adder.vhd</th>
</tr>
</thead>
<tbody>
<tr>
<td>and2.vhd</td>
<td>xor2.vhd</td>
</tr>
</tbody>
</table>
```

Hierarchy, Abstraction, and Accuracy

- Each level of hierarchy is entity/architecture pair.
- Lowest level of hierarchy is behavioral.
- Each level corresponds to different levels of detail, or abstraction.
- For simulation, each structural level is replaced with underlying behavioral model (flattened).
- Complexity of lowest level components can be used to trade simulation accuracy with time.

Example: 4-bit Adder

Example: 4-bit Adder (cont)
### Generics

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity xor2 is
    generic(gatedelay : Time:=2 ns);
    port(a, b : in std_logic;
         c : out std_logic);
end xor2;

architecture behavior of xor2 is
begin
    c <= a xor b after gatedelay;
end behavior;
```

### Generics (cont)

```vhdl
architecture gen_delay of half_adder is
component xor2
    generic(gatedelay : Time); -- HERE TOO
    port(a,b,in std_logic; c:out std_logic);
end component;
component and2
    generic(gatedelay : Time);
    port(a,b,in std_logic; c:out std_logic);
end component;
architecture behavior of xor2 is
begin
    EX1: xor2 generic map(gatedelay=>6 ns)
        port map(a=>a, b=>b, c=>sum);
    AND1: and2 generic map(gatedelay=>3 ns)
        port map(a=>a, b=>b, c=>carry);
end gen_delay;
```

### Passing Generics Through Multiple Levels

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
entity half_adder is
    generic(gatedelay : Time:=3 ns);
    port(a,b : in std_logic;
         sum, carry : out std_logic);
end half_adder;
architecture gen_delay2 of half_adder is
    . . .
begin
    EX1: xor2 generic map(gatedelay=>gatedelay)
        port map(a=>a, b=>b, c=>sum);
    AND1: and2 generic map(gatedelay=>gatedelay)
        port map(a=>a, b=>b, c=>carry);
end gen_delay2;
```

### Example: N-Input OR Gate

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
entity generic_or is
    generic(n : positive:=2);
    port(in1:in std_logic_vector(n-1 downto 0);
         z : out std_logic);
end generic_or;
architecture behavioral of generic_or is
begin
process(in1)
    variable sum:std_logic:=’0’;
    begin
        sum:=’0’;
        for i in 0 to (n-1) loop
            sum:=sum or in1(i);
        end loop;
        z<=>sum;
    end process;
end behavioral;
```

---

The image contains a page from a document discussing VHDL generics. The text is divided into sections that explain the use of generics in VHDL, including how to pass generics through multiple levels of architecture, and an example of an N-input OR gate. Each section is clearly marked, and the code snippets are presented in a readable format.
Configurations

* For each entity, there may be many different architectures.
* Configurations are used to associate (or bind) an architecture with each component in model.
* W/o configuration, default binding rules apply.

Default Binding Rules

* Search working directory.
* If entity name is the same as component name, then this entity is bound to the component.
* If no match is found, binding is deferred.
* If multiple architectures exist, then most recently compiled architecture is used.

Configuration Specification

library IEEE;
library POWER;
use IEEE.std_logic_1164.all;
entity full_adder is . . .
architecture structural of full_adder is
component half_adder . . .
for H1:half_adder
use entity WORK.half_adder(behavioral)
for H2:half_adder
use entity WORK.half_adder(structural)
for O1:or_2
use entity POWER.lpo2(behavioral)
generic map(gatedelay=>gatedelay)
port map(I1=>a, I2=>b, Z=>c);
Configuration Declaration

configuration Config_A of full_adder is
for structural
    for H1:half_adder
        use entity WORK.half_adder(behavioral)
    end for;
    for H2:half_adder
        use entity WORK.half_adder(structural)
    end for;
    for O1:or_2
        use entity POWER.lpo2(behavioral)
generic map(gatedelay=>gatedelay)
    port map(I1=>a, I2=>b, Z=>c);
end for;
end for;
end Config_A;

Common Programming Errors

* If you modify component, you must reanalyze it.
* Be aware that generic parameters may be declared in three places (within the model, in component declaration, and in instantiation).
* When using default bindings, the name, type, and mode of each signal in a component declaration must match the entity.

What We Have Learned So Far

* Structural models:
  * Component declaration and instantiation
* Tradeoffs between accuracy and speed.
* Generics for values and parameterized models.
* Configurations:
  * Default binding rules,
  * Configuration specification
  * Configuration declaration
* NEXT TIME: Subprograms, packages, libraries.