

Chapter 3

VHDL Constructs for Structure and Hierarchy Descriptions

VHDL Constructs for Structure and Hierarchy Descriptions

3.1 Basic Components

3.1.1 Basic Model

3.2 Component Instantiations

3.2.1 Direct Instantiation

3.2.2 Component Instantiation

3.3 Iterative Networks

3.3.1 Multi-bit Vectors

3.3.2 Multi-instance Generations

3.3.3 Simplified Generations

3.4 Binding Alternatives

3.5 Association Methods

VHDL Constructs for Structure and Hierarchy Descriptions

3.6 Generic Parameters

- 3.6.1 Using Generic Default Values
- 3.6.2 Generic Map Aspect
- 3.6.3 Generic Association List

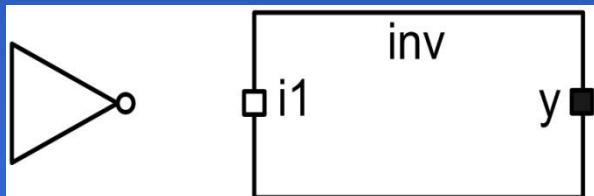
3.7 Design Configuration

- 3.7.1 Basic Configuration Declaration
- 3.7.2 Incremental Configuration
- 3.7.3 Configuring Nested Components
- 3.7.4 Indexing Block Configurations
- 3.7.5 Instantiating a Design Unit

3.8 Design Simulation

3.9 Summary

Basic Model



```
ENTITY inv IS
  PORT (i1 : IN BIT; y : OUT BIT);
END ENTITY;
--
ARCHITECTURE delay1 OF inv IS
BEGIN
  y <= NOT i1 AFTER 3 NS;
END ARCHITECTURE delay1;
```

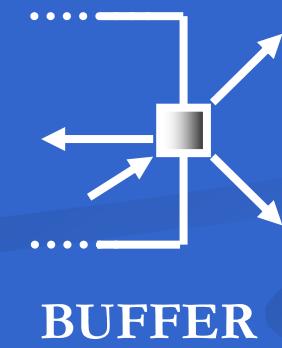
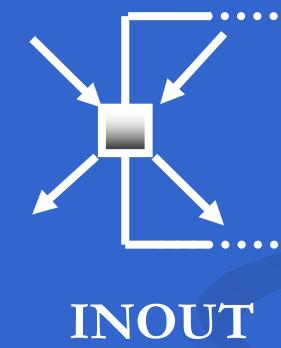
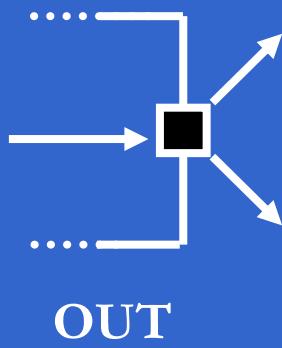
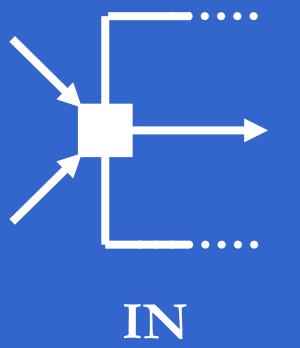
Basic Model

```
ENTITY inv IS
  PORT
    (
      i1 : IN BIT — interface_signal_declaration
    ;
      Y : OUT BIT — interface_signal_declaration
    )
    ;
END ENTITY;
```

The code illustrates the basic structure of an entity declaration in VHDL. It starts with the keyword `ENTITY`, followed by the entity name `inv`, then `IS`. This is followed by the `PORT` section, which begins with an opening parenthesis. Inside the `PORT` section, there are two signal declarations: `i1 : IN BIT` and `Y : OUT BIT`, separated by a semicolon. Both declarations include a comment `— interface_signal_declaration`. The entire `PORT` section is enclosed in a brace labeled `entity_declaration`. The entire entity declaration is enclosed in another brace labeled `port_clause`.

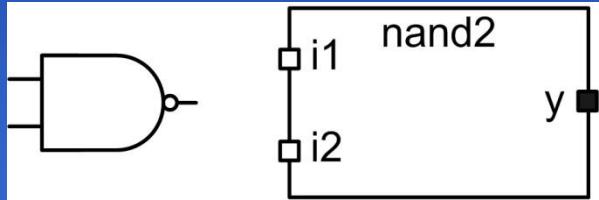
- Details of the Entity Declaration of the Inverter

Basic Model



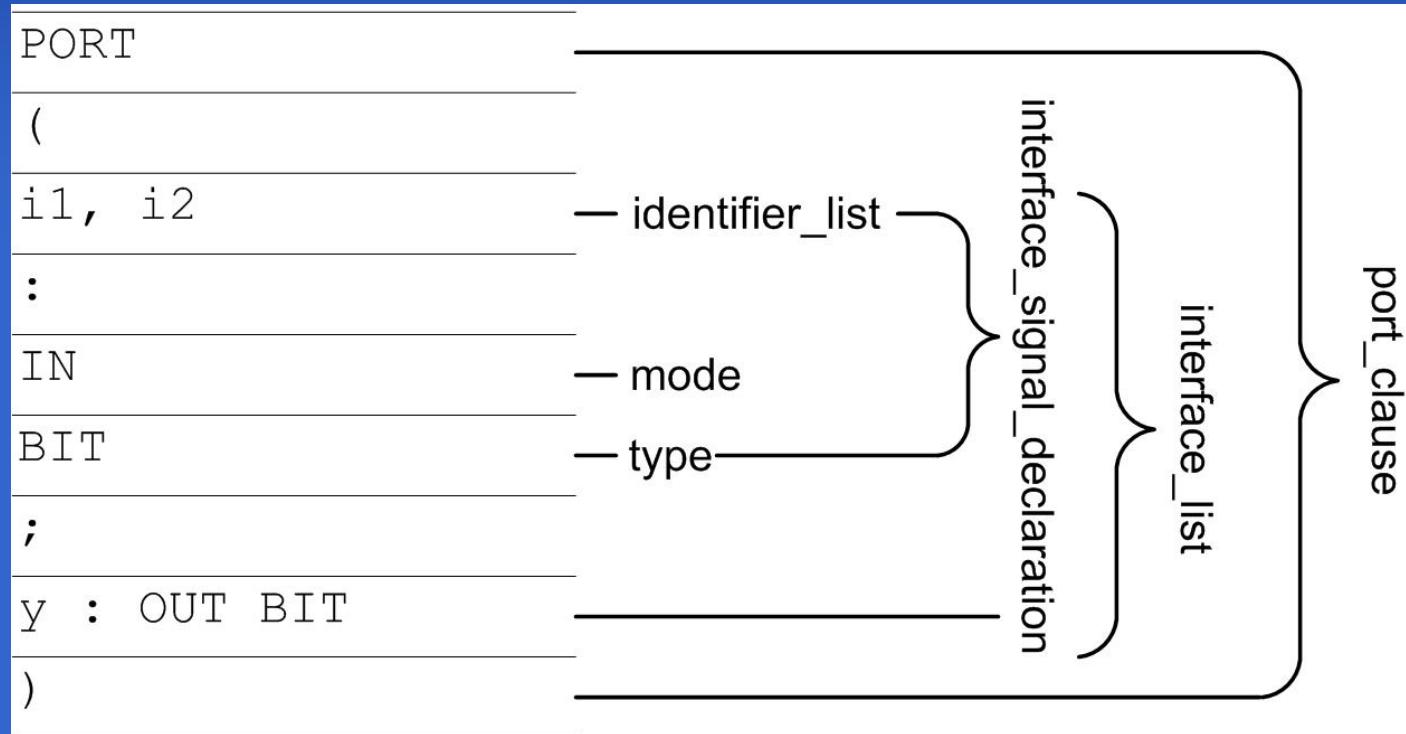
- Entity Parts, (a) Inputs, (b) Outputs, (c) Bidirectional, (d) Buffers

Basic Model



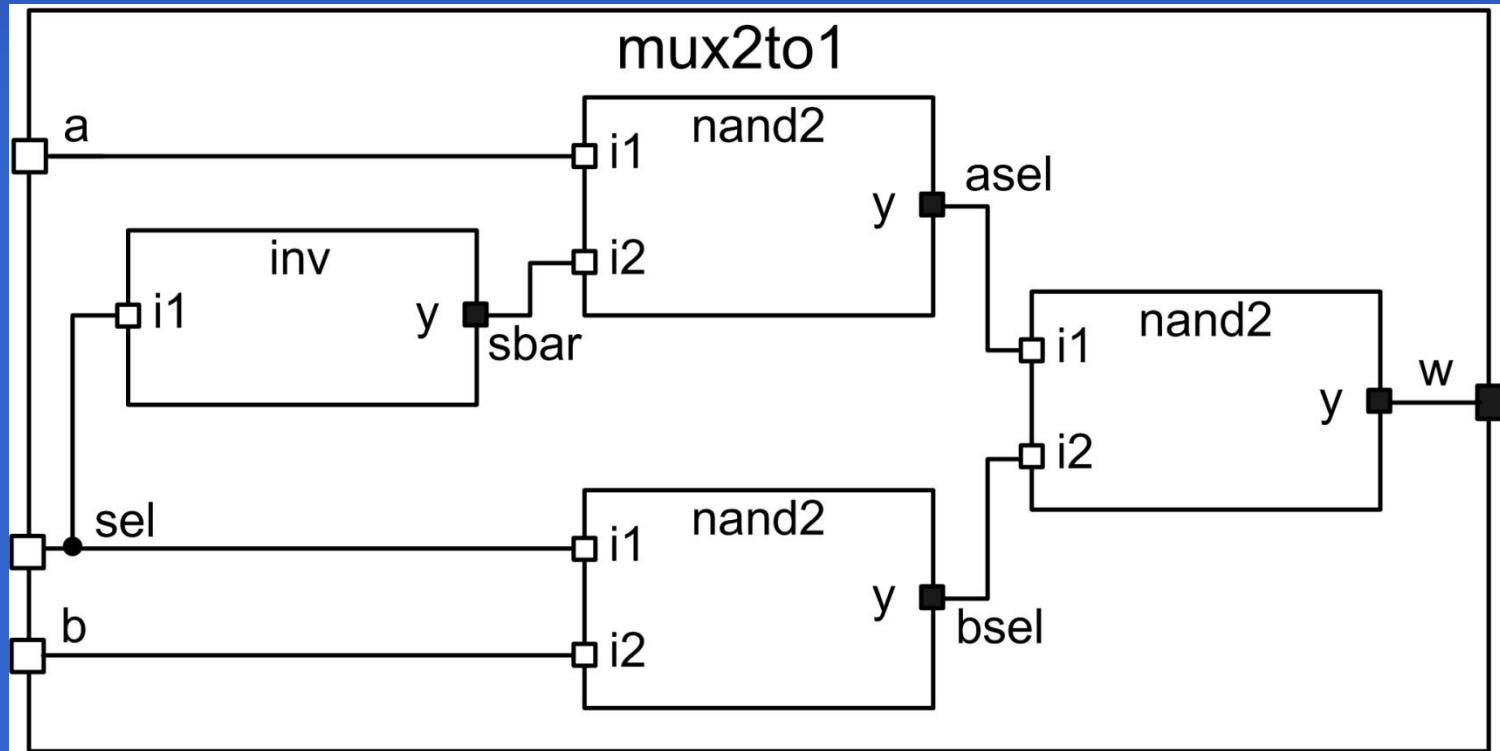
```
ENTITY nand2 IS
    PORT (i1, i2 : IN BIT; y : OUT BIT);
END ENTITY;
--
ARCHITECTURE delay1 OF nand2 IS
BEGIN
    y <= i1 NAND i2 AFTER 5 NS;
END ARCHITECTURE delay1;
```

Basic Model



- Port Clause Details for *nand2*

Component Instantiations



Direct Instantiation

```
ENTITY multiplexer IS
    PORT (a, b, s : IN BIT;
          w : OUT BIT);
END ENTITY;

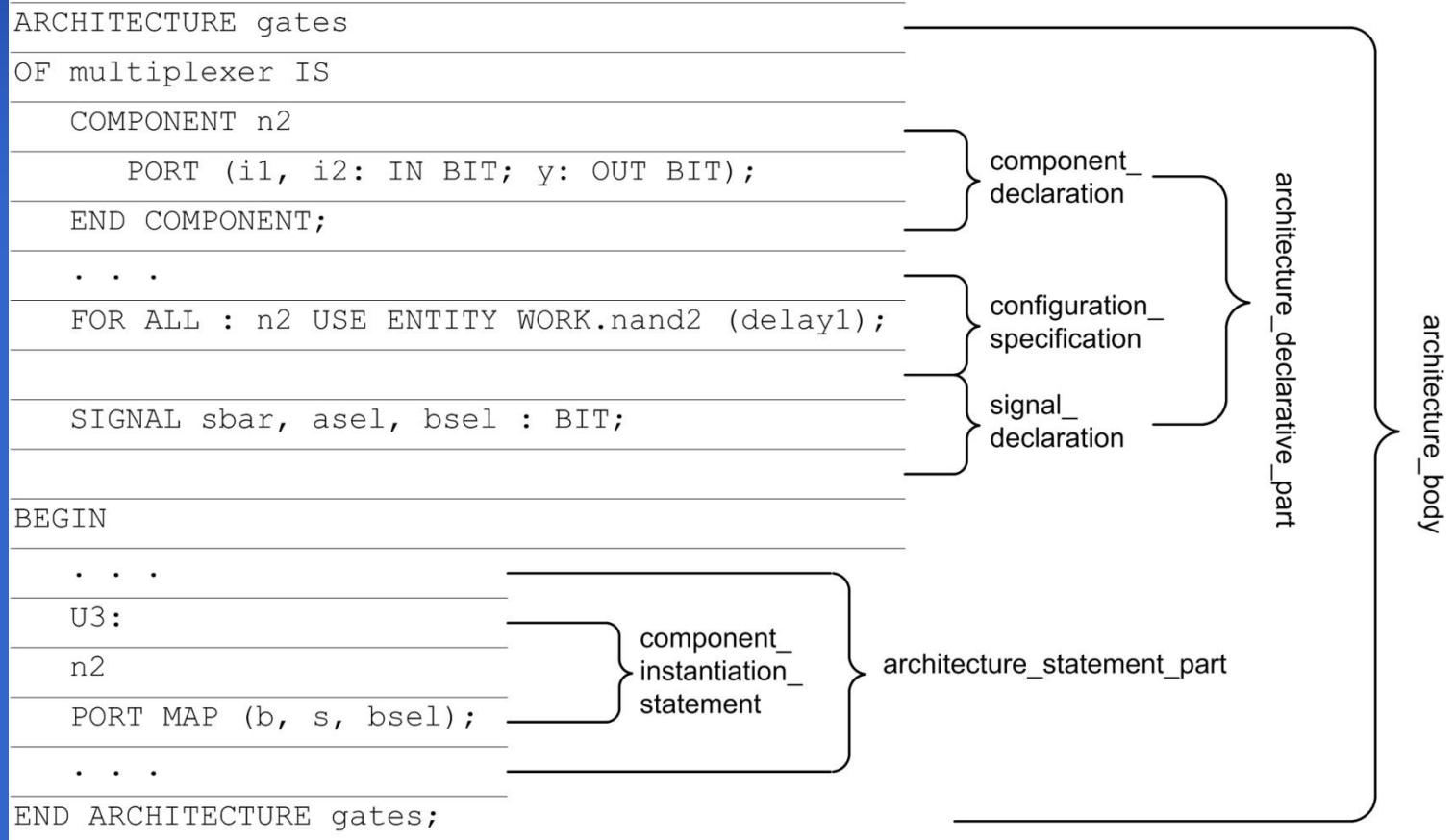
--
ARCHITECTURE direct OF multiplexer IS
    SIGNAL sbar, asel, bsel : BIT;
BEGIN
    U1: ENTITY WORK.inv (delay1) PORT MAP (s, sbar);
    U2: ENTITY WORK.nand2 (delay1) PORT MAP (a, sbar, asel);
    U3: ENTITY WORK.nand2 (delay1) PORT MAP (b, s, bsel);
    U4: ENTITY WORK.nand2 (delay1) PORT MAP (asel, bsel, w);
END ARCHITECTURE direct;
```

- Using Direct Instantiations

Component Instantiation

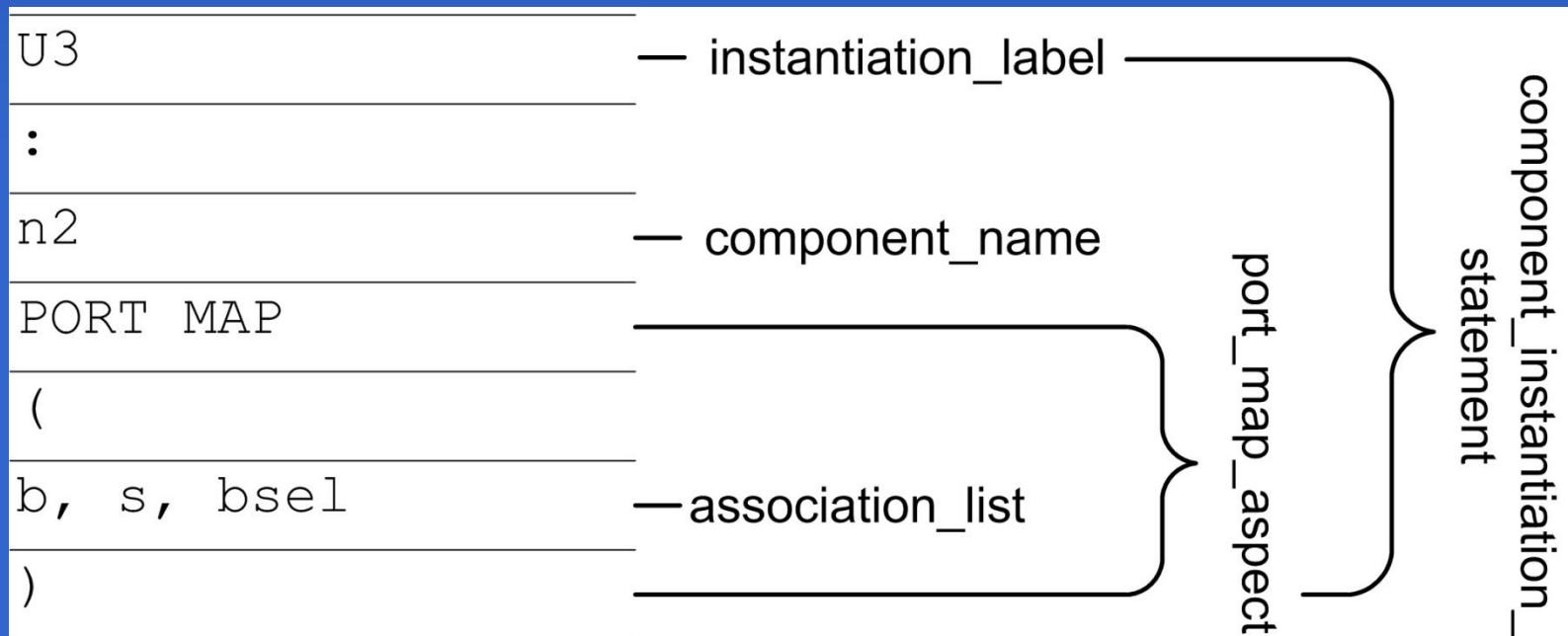
```
ARCHITECTURE gates OF multiplexer IS
COMPONENT n1
    PORT (i1: IN BIT; y: OUT BIT);
END COMPONENT;
COMPONENT n2
    PORT (i1, i2: IN BIT; y: OUT BIT);
END COMPONENT;
FOR ALL : n1 USE ENTITY WORK.inv (delay1);
FOR ALL : n2 USE ENTITY WORK.nand2 (delay1);
SIGNAL sbar, asel, bsel : BIT;
BEGIN
    U1: n1 PORT MAP (s, sbar);
    U2: n2 PORT MAP (a, sbar, asel);
    U3: n2 PORT MAP (b, s, bsel);
    U4: n2 PORT MAP (asel, bsel, w);
END ARCHITECTURE gates;
```

Component Instantiation



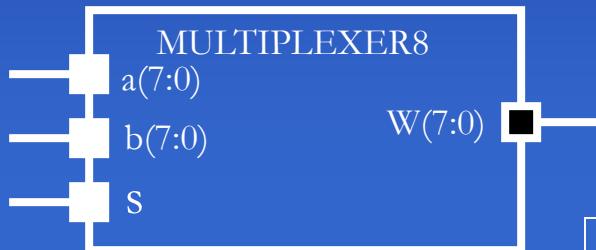
- Syntax Details of the Architecture Body of *multiplexer(gates)*

Component Instantiation



- Component Instantiation Statement Syntax Details

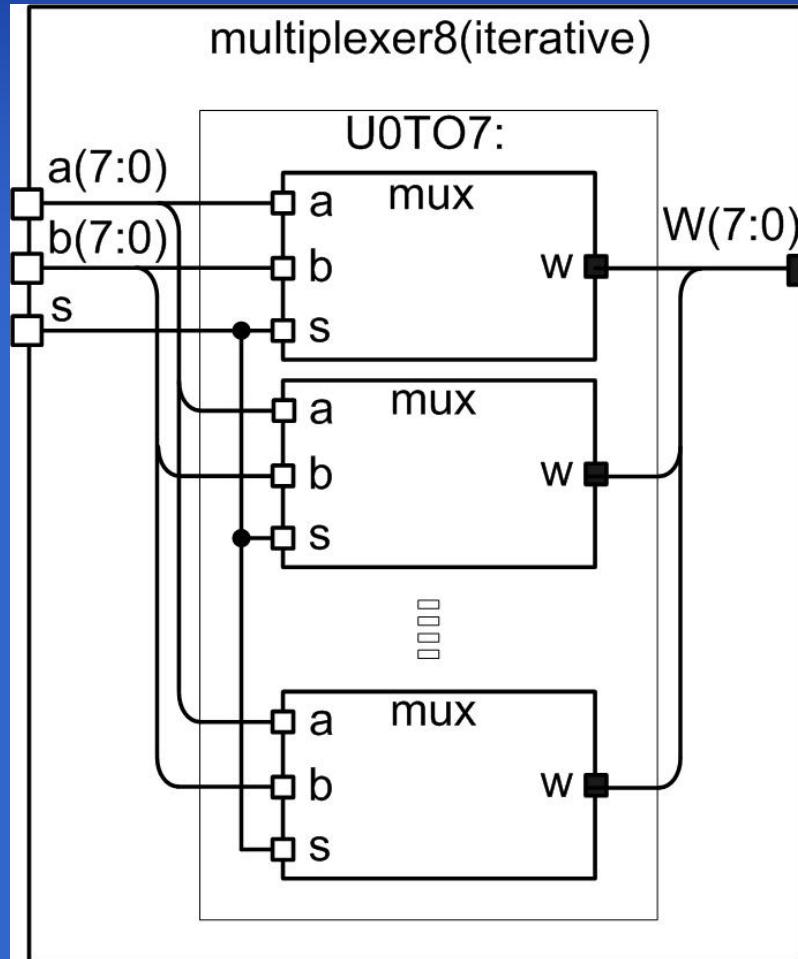
Multi-bit Vectors



```
ENTITY multiplexer8 IS
  PORT (
    a, b : IN BIT_VECTOR (7 DOWNTO 0);
    s : IN BIT;
    w : OUT BIT_VECTOR (7 DOWNTO 0) );
END ENTITY;
```

- Interface Description *multiplexer8*

Multi-instance Generations



- *multiplexer8* Hierarchical Structure

Multi-instance Generations

```
ARCHITECTURE iterative OF multiplexer8 IS
COMPONENT mux PORT (a, b, s : IN BIT; w : OUT BIT);
END COMPONENT;
BEGIN
U0TO7: FOR i IN 0 TO 7 GENERATE
    FOR ALL : mux USE ENTITY WORK.multiplexer (gates);
BEGIN
    Ui: mux PORT MAP (a(i), b(i), s, w(i));
END GENERATE;
END ARCHITECTURE iterative;
```

- Iterative Architecture of *multiplexer8*

Multi-instance Generations

```
U0TO7
:
FOR i IN 0 TO 7
GENERATE
    FOR ALL : mux
        USE ENTITY WORK.multiplexer (gates);
BEGIN
    Ui
    :
    mux
        PORT MAP (a(i), b(i), s, w(i));
END GENERATE;
```

The diagram illustrates the syntax of a VHDL generate statement. It shows a code snippet with annotations:

- A large curly brace on the right side groups the entire "GENERATE" block as a `generate_statement`.
- Inside this group, another curly brace groups the "USE" clause and the "PORT MAP" statement as `block_declarative_item`.
- Below that, another curly brace groups the "Ui" and "mux" lines as `concurrent_statement`.
- Annotations include:
 - `generation_label`: points to the label "U0TO7".
 - `generation_scheme`: points to the "FOR i IN 0 TO 7" and "GENERATE" lines.

- **Generate Statement Syntax Details**

Simplified Generations

```
ARCHITECTURE direct OF multiplexer8 IS
BEGIN
    U0TO7: FOR i IN 0 TO 7 GENERATE
        Ui: ENTITY WORK.multiplexer
            PORT MAP (a(i), b(i), s, w(i));
    END GENERATE;
END ARCHITECTURE direct;
```

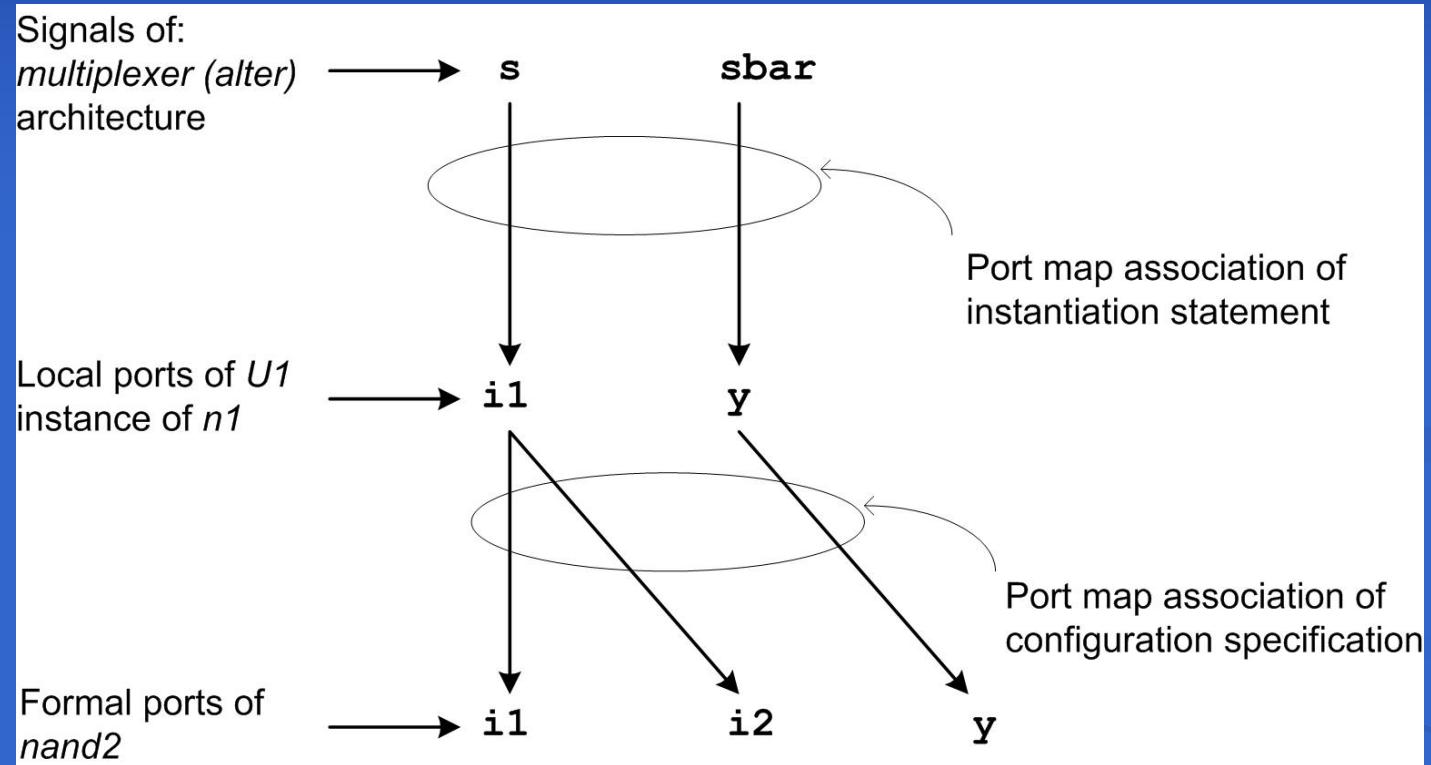
- Generating Direct Instantiation

Binding Alternatives

```
ARCHITECTURE alter OF multiplexer IS
COMPONENT n1
    PORT (i1: IN BIT; y: OUT BIT);
END COMPONENT;
COMPONENT n2
    PORT (i1, i2: IN BIT; y: OUT BIT);
END COMPONENT;
FOR U1 : n1                                -- Line 8
    USE ENTITY WORK.nand2 (delay1) PORT MAP (i1, i1, y);
FOR ALL : n2 USE ENTITY WORK.nand2 (delay1);
SIGNAL sbar, asel, bsel : BIT;
BEGIN
    U1: n1 PORT MAP (s, sbar);
    U2: n2 PORT MAP (a, sbar, asel);
    U3: n2 PORT MAP (b, s, bsel);
    U4: n2 PORT MAP (asel, bsel, w);
END ARCHITECTURE alter;
```

- Configuration Specification Port Map

Binding Alternatives



- Two-step Associations

Generic Parameters

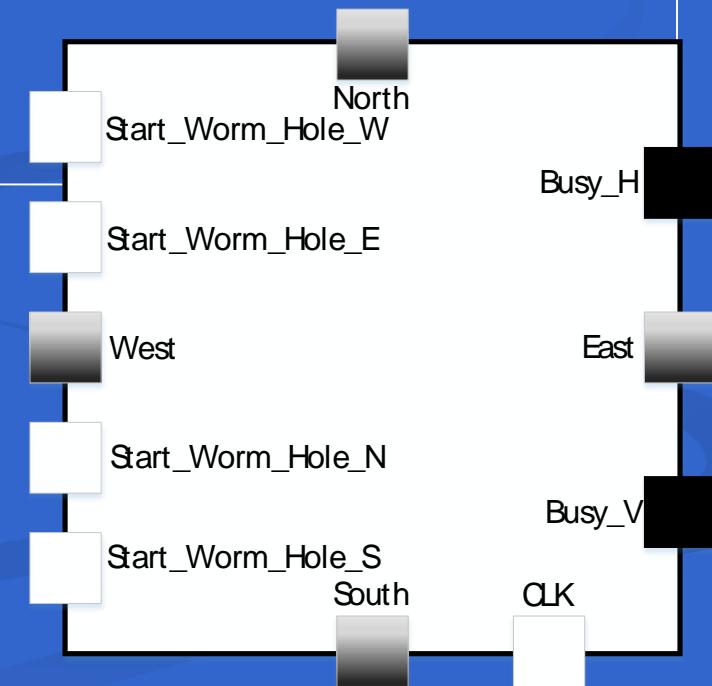
```
ENTITY inv_t IS
  GENERIC (tplh : TIME := 5 NS; tphl : TIME := 3 NS);
  PORT (i1 : IN BIT; y : OUT BIT);
END ENTITY;
--
ARCHITECTURE delay2 OF inv_t IS
BEGIN
  y <= '1' AFTER tplh WHEN (NOT i1) = '1' ELSE
    '0' AFTER tphl;
END ARCHITECTURE delay2;
-- --
```

Generic Parameters

```
ENTITY nand2_t IS
  GENERIC (tplh: TIME := 6 NS; tphl : TIME := 4 NS);
  PORT (i1, i2 : IN BIT;
        y : OUT BIT);
END ENTITY;
--
ARCHITECTURE delay2 OF nand2_t IS
BEGIN
  y <= '1' AFTER tplh WHEN (i1 NAND i2) = '1' ELSE
    '0' AFTER tphl;
END ARCHITECTURE delay2;
```

Generic Parameters

```
ENTITY SWITCH IS
  GENERIC (Psize : INTEGER := 16);
  PORT (CLK : IN STD_LOGIC;
         Start_Worm_Hole_E, Start_Worm_Hole_W,
         Start_Worm_Hole_N, Start_Worm_Hole_S : IN STD_LOGIC;
         East, West, North, South : inout STD_LOGIC_VECTOR(Psize-1 downto 0);
         Busy_H, Busy_V : OUT STD_LOGIC);
END ENTITY;
```



Generic Parameters

```
ENTITY inv_t IS
  GENERIC
    (
      tplh : TIME := 5 NS;
      tphl : TIME := 3 NS
    );
  PORT
    (i1 : IN BIT;
     y : OUT BIT);
END ENTITY;
```

The diagram illustrates the structure of the VHDL code for an inverter entity. It shows the following components and their relationships:

- entity declaration:** The outermost structure, containing the `ENTITY`, `END ENTITY;`, and the `entity header`.
- entity header:** Contains the `GENERIC` clause and the `PORT` clause.
- (formal) generic clause:** Contains the generic interface list, which includes the `tplh` and `tphl` declarations.
- interface constant declaration:** A bracket groups the `TIME` assignments for `tplh` and `tphl`.
- (generic) interface list:** A bracket groups the `tplh` and `tphl` declarations.
- (formal) port clause:** A bracket groups the `i1` and `y` declarations.

- Details of the Entity Declaration of the Inverter with Generics

Using Generic Default Values

```
ARCHITECTURE default OF multiplexer IS
  COMPONENT n2
    PORT (i1, i2: IN BIT; y: OUT BIT);
  END COMPONENT;
  FOR ALL : n2 USE ENTITY WORK.nand2_t (delay2);
  SIGNAL sbar, asel, bsel : BIT;
BEGIN
  U1: n2 PORT MAP (s, s, sbar);
  U2: n2 PORT MAP (a, sbar, asel);
  U3: n2 PORT MAP (b, s, bsel);
  U4: n2 PORT MAP (asel, bsel, w);
END ARCHITECTURE default;
```

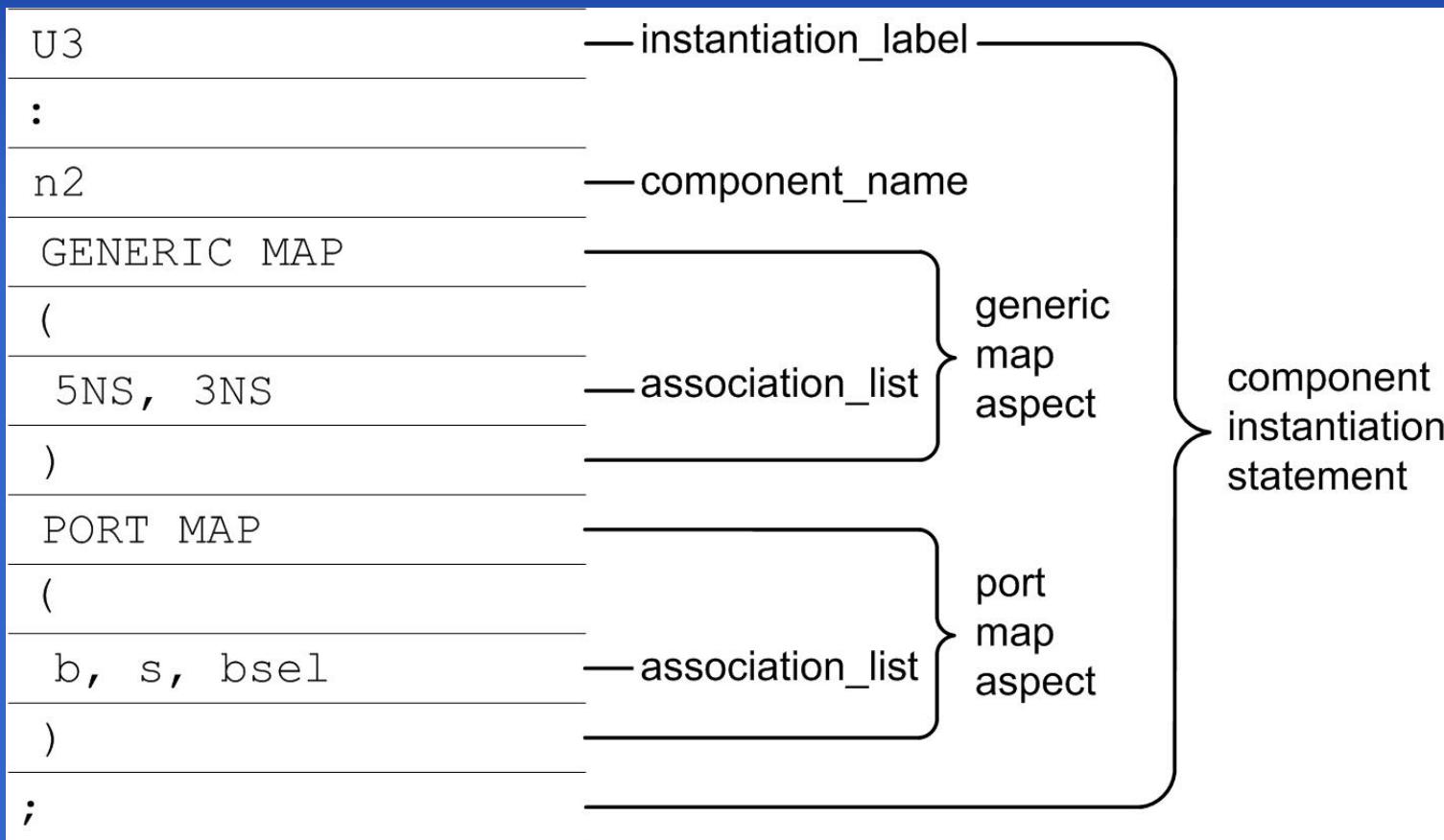
- Default Values for Generic Parameters

Generic Map Aspect

```
ARCHITECTURE fixed OF multiplexer IS
COMPONENT n2
    GENERIC (tplh, tphl : TIME);
    PORT (i1, i2: IN BIT; y: OUT BIT);
END COMPONENT;
FOR ALL : n2 USE ENTITY WORK.nand2_t (delay2);
SIGNAL sbar, asel, bsel : BIT;
BEGIN
    U1:n2 GENERIC MAP (5 NS, 3 NS) PORT MAP (s, s, sbar);
    U2:n2 GENERIC MAP (5 NS, 3 NS) PORT MAP (a, sbar, asel);
    U3:n2 GENERIC MAP (5 NS, 3 NS) PORT MAP (b, s, bsel);
    U4:n2 GENERIC MAP (5 NS, 3 NS) PORT MAP (asel, bsel, w);
END ARCHITECTURE fixed;
```

- Using Generic Map Aspect

Generic Map Aspect



- Component Instantiation Statement with a Generic Map Aspect

Basic Configuration Declaration

```
ARCHITECTURE configurable OF multiplexer IS
COMPONENT n2
    GENERIC (tplh, tphl : TIME := 4 NS);
    PORT (i1, i2: IN BIT; y: OUT BIT);
END COMPONENT;
SIGNAL sbar, asel, bsel : BIT;

BEGIN
    U1: n2 PORT MAP (s, s, sbar);
    U2: n2 PORT MAP (a, sbar, asel);
    U3: n2 PORT MAP (b, s, bsel);
    U4: n2 PORT MAP (asel, bsel, w);
END ARCHITECTURE configurable;
```

Basic Configuration Declaration

```
{  
  CONFIGURATION configured OF multiplexer IS  
    FOR configurable  
      {  
        FOR ALL : n2  
          USE ENTITY WORK.nand2_t (delay2)  
          GENERIC MAP (5 NS, 3 NS);  
        END FOR;  
      }  
    END FOR;  
  END CONFIGURATION configured;
```

- Basic Configuration Declaration

Basic Configuration Declaration

```
CONFIGURATION another_configured OF multiplexer IS
  FOR configurable
    FOR ALL : n2
      USE ENTITY WORK.nand2_t (delay2);
    END FOR;
  END FOR;
END CONFIGURATION another_configured;
```

- Another Configuration Declaration for Multiplexer

Basic Configuration Declaration

```
CONFIGURATION
  configured
    OF
      multiplexer
    IS
      FOR configurable
        FOR ALL : n2
          USE
            ENTITY
              WORK.nand2_t (delay2)
            GENERIC
              MAP (5 NS, 3 NS)
            ;
          END FOR;
        END FOR;
      END CONFIGURATION configured;
```

The diagram illustrates the structure of a VHDL configuration declaration. It shows the following components:

- identifier**: Points to the word "configured".
- entity_name**: Points to the word "multiplexer".
- component_declaration**: A bracketed group covering the entire configuration block.
- binding_indication**: A bracketed group covering the "ENTITY" and "GENERIC" sections.
- blob_configuration**: A bracketed group covering the "USE" section and the "MAP" generic map.

Incremental Configuration

```
ARCHITECTURE reconfigurable OF multiplexer IS
COMPONENT n2
    GENERIC (tplh, tphl : TIME := 4 NS);
    PORT (i1, i2: IN BIT; y: OUT BIT);
END COMPONENT;
--Primary Binding:
FOR ALL : n2 USE ENTITY WORK.nand2_t (delay2);
SIGNAL sbar, asel, bsel : BIT;
BEGIN
    U1: n2 PORT MAP (s, s, sbar);
    U2: n2 PORT MAP (a, sbar, asel);
    U3: n2 PORT MAP (b, s, bsel);
    U4: n2 PORT MAP (asel, bsel, w);
END ARCHITECTURE reconfigurable;
```

```
CONFIGURATION reconfigured OF multiplexer IS
FOR reconfigurable
FOR ALL : n2
    GENERIC MAP (5 NS, 3 NS);
END FOR;
END FOR;
END CONFIGURATION reconfigured;
```

- Incremental Binding

Configuring Nested Components

```
ARCHITECTURE multiconfig OF multiplexer8 IS
  COMPONENT mux
    PORT (a, b, s : IN BIT; w : OUT BIT);
  END COMPONENT;
BEGIN
  U0TO7: FOR i IN 0 TO 7 GENERATE
    Ui: mux PORT MAP (a(i), b(i), s, w(i));
  END GENERATE;
END ARCHITECTURE multiconfig;
```

- Unspecified Components and Sub-components

Configuring Nested Components

```
01: CONFIGURATION multiconfiged OF multiplexer8 IS
02:   FOR multiconfig
03:     FOR U0TO7
04:       FOR ALL : mux
05:         USE ENTITY WORK.multiplexer (configurable);
06:           FOR configurable
07:             FOR ALL : n2
08:               USE ENTITY WORK.nand2_t (delay2)
09:                 GENERIC MAP (5 NS, 3 NS);
10:               END FOR;
11:             END FOR;
12:           END FOR;
13:         END FOR;
14:   END CONFIGURATION multiconfiged;
```

- Multi-level Configuration Declaration

Configuring Nested Components

Starts and Ends in Line	Language Structure	Visibility to:	Binding to:
1-14	Configuration Declaration	-	-
2-13	Block Configuration	<i>multiplexer8</i> (<i>multiconfig</i>) Architecture	-
3-12	Block Configuration	<i>U0TO7:</i> Generate	-
4-11	Component Configuration	-	<i>multiplexer</i> (<i>configurable</i>)
5-10	Block Configuration	<i>multiplexer8</i> (<i>multiconfig</i>) Architecture	
6-9	Component Configuration	-	<i>nand2_t</i> (<i>delay2</i>)

Indexing Block Configurations

```
CONFIGURATION differentlyconfiged OF multiplexer8 IS
  FOR multiconfig
    FOR U0TO7 (0)
      FOR Ui : mux
        USE ENTITY WORK.multiplexer (configurable);
        FOR configurable
          FOR U3 : n2
            USE ENTITY WORK.nand2_t (delay2)
            GENERIC MAP (3 NS, 2 NS);
          END FOR;
          FOR OTHERS : n2
            USE ENTITY WORK.nand2_t (delay2)
            GENERIC MAP (4 NS, 6 NS);
          END FOR;
        END FOR;
      END FOR;
    END FOR;
  END FOR;
```

```
FOR U0TO7 (1 TO 7)
  FOR Ui : mux
    USE ENTITY WORK.multiplexer (configurable);
    FOR configurable
      FOR ALL : n2
        USE ENTITY WORK.nand2_t (delay2)
        GENERIC MAP (8 NS, 5 NS);
      END FOR;
    END FOR;
    END FOR;
  END FOR;
END CONFIGURATION differentlyconfiged;
```

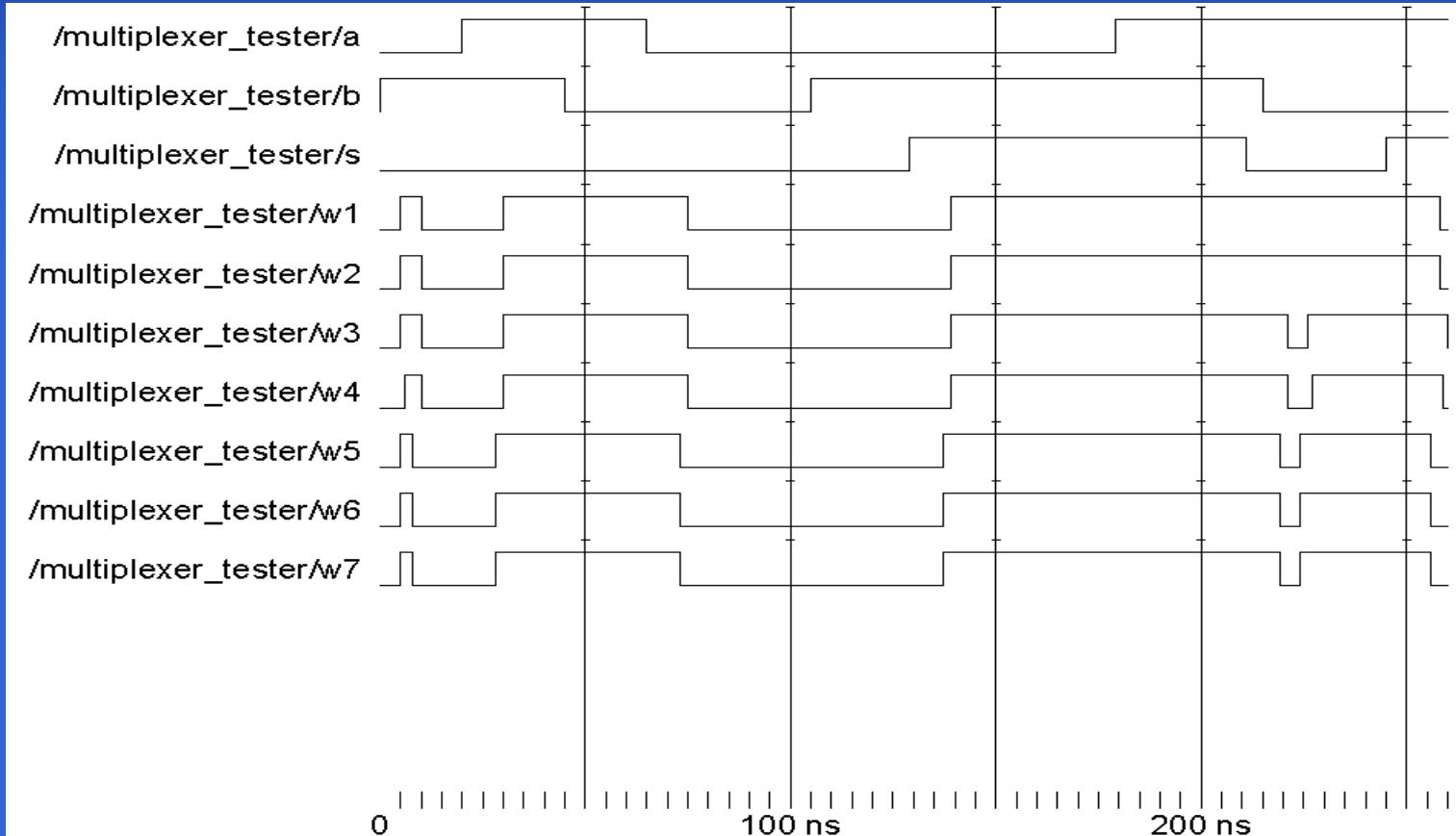
- Indexing Configurations for Generate Statements

Instantiating a Design Unit

```
ENTITY multiplexer_tester IS
END ENTITY;
--
ARCHITECTURE timed OF multiplexer_tester IS
  SIGNAL a, b, s, w1, w2, w3, w4, w5, w6, w7 : BIT;
BEGIN
  UUT1:ENTITY WORK.multiplexer(direct) PORT MAP (a,b,s,w1);
  UUT2:ENTITY WORK.multiplexer(gates) PORT MAP (a,b,s,w2);
  UUT3:ENTITY WORK.multiplexer(alter) PORT MAP (a,b,s,w3);
  UUT4:ENTITY WORK.multiplexer(default) PORT MAP (a,b,s,w4);
  UUT5:ENTITY WORK.multiplexer(fixed) PORT MAP (a,b,s,w5);
  UUT6:CONFIGURATION WORK.configured PORT MAP (a,b,s,w6);
  UUT7:CONFIGURATION WORK.reconfigured PORT MAP (a,b,s,w7);
  a <= '0', '1' AFTER 020 NS,'0' AFTER 065 NS, '1' AFTER 179 NS;
  b <= '1', '0' AFTER 045 NS, '1' AFTER 105 NS, '0' AFTER 215 NS;
  s <= '0', '1' AFTER 129 NS, '0' AFTER 211 NS, '1' AFTER 245 NS;
END ARCHITECTURE timed;
```

- Instantiating Design Units

Design Simulation



- Multiplexer Simulation Report

Summary

- This chapter presented:
 - Formation and configuration of upper level structures based on lower level design units
 - Illustrating various forms of component specifications, configurations, and delay parameter specifications
 - Simple examples to avoid discussion of more complex language constructs and focus only on the structural descriptions
 - Using lower level design example (**NAND** gate), and forming a multiplexer based on that.

Acknowledgment

Slides developed by:

Nadereh Hatami

First revision 2019 by

Saba Yousefzadeh