EE 595

Part VI

Structural Modeling in VHDL
Introduction

- Structural VHDL constructs
  - Use of components
  - Component binding indications
  - Use of configuration declarations
  - GENERATE statements
Package body can be compiled any time after package declaration
# Order of Compilation

<table>
<thead>
<tr>
<th>Package</th>
<th>Entity</th>
<th>Architecture</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>Port</td>
<td>Structure</td>
<td>Hierarchy</td>
</tr>
<tr>
<td>Component</td>
<td>Generics</td>
<td>Dataflow</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Process</td>
<td>Process</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Define component - component declaration
2. Define component instance - component instantiation
3. Define component/design entity association-configuration.
-- Interface
entity MAJORITY is
-- Input/Output ports
  port
    (A_IN, B_IN, C_IN : in Bit;
    Z_OUT : out Bit);
end MAJORITy;
Majority Voter (cont’d)

-- Body
architecture STRUCTURE of MAJORITY is
-- Declare logic operators
  component AND2_OP
    port (A, B : in Bit; Z : out Bit);
  end component;
  component OR3_OP
    port (A, B, C : in Bit; Z : out Bit):
  end component;

-- Declare signals to interconnect logic operators
signal INT1, INT2, INT3 : Bit;
begin
  -- Connect logic operators to describe schematic
  A1: AND2_OP port map (A_IN, B_IN, INT1);
  A2: AND2_OP port map (A_IN, C_IN, INT2);
  A3: AND2_OP port map (B_IN, C_IN, INT3);
  O1: OR3_OP port map (INT1, INT2, INT3, Z_OUT);
end STRUCTURE;
Hierarchical Structure VHDL Model

A two-level structural model of two 3-way majority functions.
Hierarchical Structure VHDL Model (cont’d)

```vhdl
entity MAJORITY_2X3 is
  -- Input/output ports
  port
    (A1, B1, C1 : in Bit;
     A2, B2, C2 : in Bit;
     Z_OUT       : out Bit);
end MAJORITY_2X3;

architecture STRUCTURE of MAJORITY_2X3 is
  component AND2_OP
    port (A, B : in Bit; Z : out Bit);
  end component;
  component MAJORITY
    port (A_IN, B_IN, C_IN : in Bit;
          Z_OUT                  : out Bit);
  end component;
```
Hierarchical Structure VHDL Model (cont’d)

Continued-

```vhdl
-- Declare internal signals
signal INT1, INT2 : Bit;
begin
  -- Connect component instances to describe schematic
  M1: MAJORITY port map (A1, B1, C1, INT1);
  M2: MAJORITY port map (A2, B2, C2, INT2);
  A1: AND2_OP port map (INT1, INT2, Z_OUT);
end STRUCTURE;
```
A VHDL Model of Majority Function Using a Package

-- Package
package LOGIC_OPS is
  -- Declare logic operators
  component AND2_OP
    port (A, B: in BIT; Z: out Bit);
  end component;
  component OR3_OP
    port (A, B, C : in BIT; Z : out Bit);
  end component;
  component NOT_Op
    port (A : in BIT; A_BAR : out Bit);
  end component;
end LOGIC_OPS;

-- Interface
entity MAJORITY is
  -- Input/output is
  port
    (A_IN, B_IN, C_IN : in Bit;
     Z_OUT                  : out Bit);
end MAJORITY;
A VHDL Model of Majority Function Using a Package (cont’d)

```vhdl
-- Body
-- Use components in package LOGIC_OPS in library work
use WORK.MOGIC_OPS.all;
architecture STRUCTURE of MAJORITY is
-- Declare signals to interconnect logic operators
signal INT1, INT2, INT3 : Bit;
beg
-- Connect logic operators
A1: AND2_OP port map (A_IN, B_IN, INT1);
A2: AND2_OP port map (A_IN, C_IN, INT2);
A3: AND2_OP port map (B_IN, C_IN, INT3);
O1: OR3_OP port map (INT1, INT2, INT3, Z_OUT);
end STRUCTURE;
```
Half Adder

Example

X
Y

A
B
Q

A
B
Q

C
S
entity HA is
  port (X: in Bit;
       Y: in Bit;
       S: out Bit;
       C: out Bit);
end HA;

architecture STRUCT of HA is

--COMPONENT DECLARATION
  component XOR2
    port (A : in Bit;
          B : in Bit;
          Q : out Bit);
  end component;

EE 595  EDA / ASIC Design Lab
Half Adder
Example (cont’d)

```vhdl
entity HA is
  port(X: in Bit;
       Y: in Bit;
       S: out Bit;
       C: out Bit);
end HA;

architecture STRUCT of HA is

--COMPONENT DECLARATION
component XOR2
  port(A : in Bit;
       B : in Bit;
       Q : out Bit;
       end component;
```
Half Adder
Example (cont’d)

```vhdl
component AND2
    port (Q: out Bit;
         A: in Bit;
         B: in Bit);
end component;

--COMPONENT INSTANTIATION
begin
    X1: XOR2
        port map(Q => S,
                 A => X,
                 B => Y);
    A1: AND2
        port map(Q => C,
                 A => X,
                 B => Y);
end STRUCT;
```
Full Adder Example
Full Adder
Example (cont’d)

```vhdl
entity FA is
  port(X: in Bit;
       Y: in Bit;
       CIN: in Bit;
       S: out Bit;
       C: out Bit);
end FA;
architecture STRUCT of FA is
  signal IN1, IN2, IN3: Bit;

--COMPONENT DECLARATION
  component XOR2
    port(Q: out Bit;
         B: in Bit;
         A: in Bit;
    end component;
```
Full Adder
Example (cont’d)

```vhdl
component OR2
  port( Q : out Bit;
        A, B : in Bit;
  end component;

component AND2
  port ( Q: out Bit;
       A: in Bit;
       B: in Bit);
end component;

--COMPONENT INSTANTIATION
begin
  X1: XOR2
    port map(Q => IN1,
               A => X,
               B => Y);
end begin;
```
Full Adder

Example (cont’d)

O1: OR2

port map(Q => C,
    A => IN3,
    B => IN2);

X2: XOR2

port map(Q => S,
    A => IN1,
    B => CIN);

A1: AND2

port map(Q => IN2,
    A => X,
    B => Y);

a2: AND2

port map(Q => IN3,
    A => IN1,
    B => CIN);

end STRUCT;
Full Adder
Dataflow Modeling

entity FULL_ADDER is
  port (CI           : in Bit;
           A, B         : in Bit;
           SUM, CO : out Bit);
end FULL_ADDER;

architecture FULL_ADDER of FULL_ADDER is
begin
  SUM <= A XOR B XOR CI;
  CO   <= ((A OR B) AND CI) OR (A AND B);
end;

Note: This Design will be instantiated in Ripple Carry Adder
Ripple Carry Adder
Example
Ripple Carry Adder
Example (cont’d)

```vhdl
entity RIPADD_8 is
  port (Ci : in Bit;
        A7, A6, A5, A4, A3, A2, A1, A0 : in Bit;
        B7, B6, B5, B4, B3, B2, B1, B0 : in Bit;
        SUM7, SUM6, SUM5, SUM4, SUM3, SUM2, SUM1, SUM0 : out Bit;
        Co : out Bit);
end RIPADD_8;
```
Ripple Carry Adder
Example (cont’d)

architecture RIPPADD_8 of RIPPADD_8 is
  component FULL_ADD
    port (CI : in Bit;
          A,B : in Bit;
          SUM, CO : out Bit);
  end component;
  signal C1, C2, C3, C4, C5, C6, C7 : Bit;
begin
  U1 : FULL_ADDER port map (Ci, A0, B0, SUM, C1);
  U2 : FULL_ADDER port map (C1, A1, B1, SUM1, C2);
  U3 : FULL_ADDER port map (C2, A2, B2, SUM2, C3);
  U4 : FULL_ADDER port map (C3, A3, B3, SUM3, C4);
  U5 : FULL_ADDER port map (C4, A4, B4, SUM4, C5);
  U6 : FULL_ADDER port map (C5, A5, B5, SUM5, C6);
  U7 : FULL_ADDER port map (C6, A6, B6, SUM6, C7);
  U8 : FULL_ADDER port map (C7, A7, B7, SUM7, CO);
end;
Mechanism for Incorporating VHDL Design Object

- VHDL mechanisms to incorporate design objects
  - Using direct instantiation (not available prior to VHDL-93)
  - Using *component* declarations and instantiations
    - Create idealized local *components* (i.e. declarations) and connect them to local signals (i.e. instantiations)
    - *Component* instantiations are then bound to VHDL design objects either:
      - Locally -- within the architecture declaring the component
      - At higher levels of design hierarchy, via *configurations*
D Flip Flop with Enable as Running Example

First, need to find the building block(s)

```vhdl
use WORK.RESOURCES.all;

entity DFF is
    generic(
        TPROP : DELAY := 8 NS;
        TSU   : DELAY := 2 NS);
    port(
        D            : in Bit;
        CLK        : in Bit;
        ENABLE   : in Bit;
        Q            : out Bit;
        QN         : out Bit);
end DFF;
```
D Flip Flop with Enable as Running Example (cont’d)

```vhdl
architecture BEHAV of DFF is
begin
  ONE : process (CLK)
  begin
    -- rising clock edge
    if ((CLK = '1' AND Clk'LAST_VALUE = '0')
        AND ENABLE = '1') then
      -- ff enabled
      -- check setup
      if (D'STABLE(TSU)) then
        -- check valid input data
        if (D = '0') then
          Q <= '0' after TPROP;
          QN <= '1' after TPROP;
      end if;
  end if;
end process ONE;
```

EE 595  EDA / ASIC Design Lab

California State University
Northridge
D Flip Flop with Enable as Running Example (cont’d)

```vhdl
elsif (D = '1') then
    Q <= '1' after TPROP;
    QN <= '0' after TPROP;
else
    Q <= 'X';
    QN <= 'X';
end if;
else
    Q <= 'X';
    Qn <= 'X';
end if;
end if;
end process ONE;
end BEHAV;
```
General Steps to Incorporate VHDL Design Objects

- A VHDL design object to be incorporated into an architecture must *generally* be:
  - declared -- where a local interface is defined
  - instantiated -- where local signals are connected to the local interface
    - Regular structures can be created easily using `GENERATE` statements in component instantiations
  - bound -- where an Entity/Architecture object which implements it is selected for the instantiated object
Using Component Declarations and Local Bindings

- Component declaration defines interface for idealized local object
  - Component declarations may be placed in architecture declarations or in package declarations

- Component instantiation connects local signals to component interface signals
Component Declarations and Local Binding

Example

```vhdl
use WORK.RESOURCES.all;

architecture STRUCT_2 of REG4 is
  component REG1 is
    port (D, CLK : in Bit;
          Q : out Bit);
  end component REG1;
  constant ENABLED : Bit := '1';
  for all : REG1 use WORK.DFF(BEHAV)
    port map(D=>D,CLK=>CLK,ENABLE=>ENABLED,Q=>Q,QN=>OPEN);
begin
  R0 : REG1 port map (D=>D0,CLK=>CLK,Q=>Q0);
  R1 : REG1 port map (D=>D1,CLK=>CLK,Q=>Q1);
  R2 : REG1 port map (D=>D2,CLK=>CLK,Q=>Q2);
  R3 : REG1 port map (D=>D3,CLK=>CLK,Q=>Q3);
end STRUCT_2;
```
Using Component Declarations and Configurations

```vhdl
use WORK.RESOURCES.all;

architecture STRUCT_3 of REG4 is
  component REG1 is
    port (D, CLK : in Bit;
          Q : out Bit);
  end component REG1;
  constant ENABLED : Bit := '1';
begin
  R0 : REG1 port map (D<=D0,CLK<=CLK,Q<=Q0);
  R1 : REG1 port map (D<=D1,CLK<=CLK,Q<=Q1);
  R2 : REG1 port map (D<=D2,CLK<=CLK,Q<=Q2);
  R3 : REG1 port map (D<=D3,CLK<=CLK,Q<=Q3);
end STRUCT_3;
```
Using Component Declarations and Configurations

```vhdl
use WORK.RESOURCES.all;

configuration REG4_CONF_1 of REG4 is
  constant ENABLED : Bit := '1';
  for STRUCT_3
    for all : REG1 use WORK.DFF(BEHAV)
      port map(D=>D,CLK=>CLK,ENABLE=>ENABLED,Q=>Q,QN=>OPEN);
    end for;
  end for;
end REG4_CONF_1;

-- Architecture in which a COMPONENT for reg4 is declared
...
for all : REG4_COMP use configuration
  WORK.REG4_CONF_1;
```
Power Of Configuration Declarations

Reasons to use *configuration declarations*:
- Large design may span multiple levels of hierarchy
- When the architecture is developed, only the component interface may be available
- Mechanism to put the pieces of the design together

Configurations can be used to customize the use VHDL design objects interfaces as needed:
- Entity name can be different than the component name
- Entity of incorporated design object may have more ports than the component declaration
- Ports on the entity declaration of the incorporated design object may have different names than the component declaration
The instantiation statement connects a declared component to signals in the architecture.

The instantiation has 3 key parts:
- **Name** -- to identify unique *instance* of component
- **Component type** -- to select one of the declared components
- **Port map** -- to connect to signals in architecture
  - Along with optional Generic Map presented on next slide

```vhdl
R0 : REG1 port map (D=>D0,CLK=>CLK,Q=>Q0);
```
Generic Map

- Generics allow the component to be customized upon instantiation
  - Entity declaration of design object being incorporated provides default values
- The GENERIC MAP is similar to the PORT MAP in that it maps specific values to the generics of the component

```vhdl
use WORK.MY_STUFF.all;
architecture TEST of TEST_ENTITY is
  signal S1, S2, S3 : Bit;
begin
  GATE1 : MY_STUFF.AND_GATE -- component found in package
  generic map (TPLH=>2 NS, TPHL=>3 NS)
  port map (S1, S2, S3);
end TEST;
```
Component Binding Specifications

- A component binding specification provides binding information for instantiated components
  - Single component
    ```
    for A1 : AND_GATE use BINDING_INDICATION;
    ```
  - Multiple components
    ```
    for A1, A2 : AND_GATE use BINDING_INDICATION;
    ```
  - All components
    ```
    for all : AND_GATE use BINDING_INDICATION;
    ```
    -- All components of this type are effected
  - Other components
    ```
    for others : AND_GATE use BINDING_INDICATION;
    ```
    -- i.e. for components that are not otherwise specified
The binding indication identifies the design object to be used for the component.

- Two mechanisms available:
  - VHDL entity/architecture design object
    - for all : REG1 use WORK.DFF(BEHAV);
  - VHDL configuration
    - for REG4_INST : REG4_COMP use configuration WORK.REG4_CONF_1;

- Binding indication may also include a PORT MAP and/or GENERIC MAP to customize the component(s)
Using Direct Instantiation

- Provides one-step mechanism for plugging in previously defined VHDL design objects
- Only available one level up in hierarchy from level of incorporated building block(s)

```vhd
use WORK.RESOURCES.all;

architecture STRUCT_1 of REG4 is
constant ENABLED : Bit := '1';
beg"
Rules for Actuals and Locals

- An *actual* is either signal declared within the architecture or a port in the entity declaration.
  - A port on a *component* is known as a *local* and must be matched with a compatible *actual*.
- VHDL has two main restrictions on the association of *locals* with *actuals*:
  - Local and actual must be of same data type.
  - Local and actual must be of compatible modes:
    - Locally declared signals do not have an associated mode and can connect to a local port of any mode.

```
Locally_Declared_Sig_a ─── in1 ─── out1 ─── Locally_Declared_Sig_b
     |                                  |
     |                                  |
Input_Port_a ─── in2 ─── out2 ─── Output_Port_a
```
Summary of Concepts of Structural VHDL

- Various levels of abstraction supported in description of VHDL structural models
  - Direct instantiation requires detailed knowledge of building blocks when they are incorporated
  - Use of components allows definition and use of idealized local building blocks
    - Can define local interface for component to be connected to local signals
    - Declared components bound to VHDL design objects (i.e. entity/architecture descriptions)
      - Binding done either locally or deferred to higher levels in design hierarchy via use of configurations
  - Actuals and locals must be of compatible types and modes
Generate Statement

- VHDL provides the GENERATE statement to create well patterned structures easily
  - Some structures in digital hardware are repetitive in nature (e.g. RAMs, adders)
- Any VHDL concurrent statement may be included in a GENERATE statement, including another GENERATE statement
  - Specifically, component instantiations may be made within GENERATE bodies
Generate Statement
FOR-scheme

- All objects created are similar
- The GENERATE parameter must be discrete and is undefined outside the GENERATE statement
- Loop cannot be terminated early

NAME : for n in 1 TO 8 generate
current-statements
end generate NAME;
-- this uses the and gate component from before
architecture TEST_GENERATE of TEST_ENTITY is
begin
    G1 : for N in 7 DOWNTO 0 generate
        AND_ARRAY : AND_GATE
        generic MAP (2 ns, 3 ns)
        port map (S1(n), S2(n), S3(n));
    end generate G1;
end TEST_GENERATE;
Generate Statement
IF-scheme

- Allows for conditional creation of components
- Can not use ELSE or ELSIF clauses with the IF-scheme

NAME : if (BOOLEAN EXPRESSION) generate
CONCURRENT-STATEMENTS
end generate NAME;
IF- scheme
Example

architecture TEST_GENERATE of TEST_ENTITY
signal S1, S2, S3: Bit_Vector(7 DOWNTO 0);
begin
  G1 : for n in 7 DOWNTO 0 generate

    G2 : if (n = 7) generate
        OR1 : OR_GATE
        generic map (3 ns, 3 ns)
        port map (S1(n), S2(n), S3(n));
    end generate G2;

    G3 : if (n < 7) generate
        AND_ARRAY : AND_GATE
        generic map (2 ns, 3 ns)
        port map (S1(n), S2(n), S3(n));
    end generate G3;
  end generate G1;
end TEST_GENERATE;
Examples

- Build higher level modules from the library of basic gates
  - AND-OR-Invert
  - 8 Bit Register using DFFs
  - 8 Bit Shift Register using Multiplexors and DFFs
- Use these modules to construct a datapath for unsigned 8 bit multiplication
library GATE_LIB;
use GATE_LIB.RESOURCES.all;

entity AOI2_STR is
  generic(
    TRISE : DELAY := 12 ns;
    TFALL : DELAY := 9 ns);
  port(
    A : in Bit;
    B : in Bit;
    C : in Bit;
    D : out Bit);
end AOI2_STR;
Structural And-Or-Invert Gate Architecture
Example

architecture STRUCTURAL of AOI2_STR is

-- COMPONENT DECLARATIONS
component AND2
generic(TRISE : DELAY;
TFALL : DELAY);
port(A : in Bit;
B : in Bit;
C : out Bit);
end component;

component OR2
generic(TRISE : DELAY;
TFALL : DELAY);
port(A : in Bit;
B : in Bit;
C : out Bit);
end component;
Structural And-Or-Invert Gate Architecture
Example (cont’d)

component INV
  generic(TRISE : DELAY;
           TFALL : DELAY);
  port(A : in Bit;
       B : out Bit);
end component;

-- BINDING INDICATIONS
for all : AND2 use entity GATE_LIB.AND2(BEHAV);
for all : OR2 use entity GATE_LIB.OR2(BEHAV);
for all : INV use entity GATE_LIB.INV(BEHAV);

signal AND_OUT : Bit; -- signal for output of AND gate
signal OR_OUT : Bit; -- signal for output of OR gate
begin
  -- COMPONENT INSTANTIATIONS
  AND_1 : AND2 generic map(TRISE => TRISE,
                         TFALL => TFALL)
    port map(A => A, B => B,
             C => AND_OUT);
  OR_1  : OR2  generic map/TRISE => TRISE,
             TFALL => TFALL)
    port map(A => AND_OUT, B => C,
             C => OR_OUT);
  INV_1 : INV  generic map/TRISE => TRISE,
             TFALL => TFALL)
    port map(A => OR_OUT, B => D);
end STRUCTURAL;
Structural 8 Bit Register
Example
library GATE_LIB;
use GATE_LIB.RESOURCES.all;

entity REG8_STR is

  generic(TPROP : DELAY := 8 ns;
          TSU   : DELAY := 2 ns);

  port(D            : in  Bit_Vector(7 DOWNTO 0);
       CLK        : in  Bit;
       ENABLE     : in  Bit;
       Q           : out Bit_Vector(7 DOWNTO 0);
      QN          : out Level_Vector(7 DOWNTO 0));

end REG8_STR;
architecture STRUCTURAL of REG8_STR is

-- COMPONENT DECLARATION
component DFF
generic(TPROP : DELAY;
        TSU   : DELAY);
port(D : in Bit;
    CLK : in Bit;
    ENABLE : in Bit;
    Q : out Bit;
    QN : out Bit);
end component;
Structural 8 Bit Register Architecture
Example (cont’d)

-- BINDING INDICATIONS
for all : DFF use entity GATE_LIB.DFF(BEHAV);

begin

-- COMPONENT INSTANTIATION (GENERATE)
R1:for i in 1 to 8 generate
 I1: DFF generic map(TPROP => TPROP,
           TSU => TSU)
     port map(D => D(i-1), CLK => CLK,
       ENABLE => ENABLE,
       Q => Q(i-1), QN => QN(i-1));
end generate R1;

end STRUCTURAL;
Structural 8 Bit Register Entity

Example
library GATE_LIB;
use GATE_LIB.RESOURCES.all;

entity SHIFT_REG8_STR is

  generic(TPROP : DELAY := 15 ns;
          TSU   : DELAY := 2 ns);

  port(D                 : in  Bit_Vector(7 DOWNTO 0);
       CLK             : in  Bit;
       ENABLE      : in  Bit;
       SCAN_IN     : in  Bit;
       SHIFT          : in  Bit;
       SCAN_OUT     : out Bit;
       Q                  : out Level_Vector(7 DOWNTO 0));

end SHIFT_REG8_STR;
architecture STRUCTURAL of SHIFT_REG8_STR is

-- COMPONENT DECLARATION
component MUX2
generic(TPROP : DELAY);
port(A : in Bit;
     B : in Bit;
     SEL : in Bit;
     C  : out Bit);
end component;

component DFF
generic(TPROP : DELAY;
        TSU  : DELAY);

Structural 8 Bit Shift Register Architecture
Generate with IF Scheme
Example (cont’d)

```vhdl
port( D : in Bit;
     CLK : in Bit;
     ENABLE : in Bit;
     Q : out Bit;
     Qn : out Bit);
end component;

-- BINDING INDICATIONS
for all : MUX2 use entity
  GATE_LIB.MUX2(BEHAV);
for all : DFF use entity
  GATE_LIB.DFF(BEHAV);

signal MUX_OUT : Bit_Vector(7 DOWNTO 0);
signal DFF_OUT : Bit_Vector(7 DOWNTO 0);
```
Structural 8 Bit Shift Register Architecture
Generate with IF Scheme
Example (cont’d)

-- COMPONENT INSTANTIATION (GENERATE W/ IF)
G1: for i in 0 to 7 generate

    G2 : if (i = 0) generate
    MUX1 : MUX2 generic map(TPROP => TPROP/2)
    port map(A => SCAN_IN,
            B => D(i),
            SEL => SHIFT,
            C => MUX_OUT(i));
    DFF1 : DFF generic map(TPROP => TPROP/2,
                            TSU => TSU)
    port map(D => MUX_OUT(i),
            CLK => CLK,
            ENABLE => ENABLE,
            Q => DFF_OUT(i);
            Q(i) <= DFF_OUT(i);
    end generate G2;

    -- continue

    G3 : IF (i > 0) GENERATE
    MUX1 : mux2 GENERIC MAP(tprop => tprop/2)
    PORT MAP(a => dff_out(i-1),
              b => d(i),
              sel => shift,
              c => mux_out(i));
    DFF1 : dff GENERIC MAP(tprop => tprop/2,
                             tsu => tsu)
    PORT MAP(d => mux_out(i),
              clk => clk,
              enable => enable,
              q => dff_out(i);
              q(i) <= dff_out(i);
    END GENERATE G3;
    END GENERATE G1;
    scan_out <= dff_out(7);

END structural;
Unsigned 8 Bit Multiplier Entity

Example

```vhdl
library WORK;
library GATE_LIB;
use GATE_LIB.RESOURCES.all;

entity MULT_DATAPATH is
    port(
        MULTIPLICAND : in Bit_Vector (7 DOWNTO 0);
        MULTIPLIER : in Bit_Vector(7 DOWNTO 0);
        A_ENABLE   : in Bit;  -- clock enable for A register
        A_RESET           : in Bit;  -- Reset control for A register
        A_MODE             : in Bit;  -- Shift or load mode for A
        C_ENABLE    : in Bit;  -- clock enable for C register
        M_ENABLE     : in Bit;  -- clock enable for M register
        Q_ENABLE     : in Bit;  -- clock enable for Q register
        Q_MODE       : in Bit;  -- Shift or load mode for Q
        CLK        : in Bit;
        PRODUCT     : out Bit_Vector(15 DOWNTO 0));
    end MULT_DATAPATH;
```
Unsigned 8 Bit Multiplier

Data Path

Multiplicand

M_{n-1}  \bullet  \bullet  \bullet  M_0

n-Bit Adder

A_{n-1}  A_0

Multiplier

Q_{n-1}  Q_0

Product

Control Unit

EE 595  EDA / ASIC Design Lab
architecture STRUCTURAL of MULT_DATAPATH is

component DFF
generic (TPROP : DELAY;
    TSU   : DELAY);
port(D : in Bit;
    CLK  : in Bit;
    ENABLE: in Bit;
    Q     : out Bit;
   QN    : out Bit);
end component;

component REG8_STR
generic (TPROP : DELAY;
    TSU   : DELAY);

EE 595  EDA / ASIC Design Lab
Unsigned 8 Bit Multiplier Architecture
Example (cont’d)

```vhdl
port (D : in Bit_Vector(0 TO 7);
     CLK : in Bit;
     ENABLE : in Bit;
     Q : out Bit_Vector(0 TO 7);
     QN : out Bit_Vector(0 TO 7));
end component;
component SHIFT_REG8_STR
generic(TPROP : DELAY;
       TSU     : DELAY);
port (D : in Bit_Vector(0 TO 7);
     CLK : in Bit;
     ENABLE : in Bit;
     SCAN_IN : in Bit;
     SHIFT : in Bit;
     SCAN_OUT : out Bit;
     Q : out Bit_Vector(0 TO 7));
end component;
```
unsigned 8 bit multiplier architecture
example (cont’d)

```
component ALU_STR
  generic(TPROP : DELAY);
  port(A       : in Bit_Vector(7 DOWNTO 0);
       B       : in Bit_Vector(7 DOWNTO 0);
       MODE   : in Bit;
       CIN     : in Bit;
       SUM     : out Bit_Vector(7 DOWNTO 0);
       COUT    : out Bit);
end component;
for all : AND2    use entity GATE_LIB.AND2(BEHAV);
for all : DFF     use entity GATE_LIB.DFF(BEHAV);
for all : REG8_STR use entity WORK.REG8_STR(STRUCTURAL);
for all : SHIFT_REG8_STR use entity WORK.SHIFT_REG8_STR(STRUCTURAL);
for all : ALU_STR  use entity WORK.ALU_STR(STRUCTURAL);
```
Unsigned 8 Bit Multiplier Architecture
Example (cont’d)

```vhdl
signal GND : LEVEL := '0';
signal C_OUT, A_SCAN_OUT, CARRY_OUT : Bit;
signal A_OUT, ALU_OUT, M_OUT : Bit_Vector(7 DOWNTO 0);

begin
  -- A, C, M, and Q registers
  A1 : SHIFT_REG8_STR generic map(6 ns, 1 ns)
  port map(D(0)=>ALU_OUT(0),D(1)=>ALU_OUT(1),
            D(2)=>ALU_OUT(2),D(3)=>ALU_OUT(3),
            D(4)=>ALU_OUT(4),D(5)=>ALU_OUT(5),
            D(6)=>ALU_OUT(6),D(7)=>ALU_OUT(7),
            CLK => CLK, ENABLE => A_ENABLE,
            SCAN_IN => C_OUT, SHIFT => A_MODE,
            ...
```

California State University
Northridge
Unsigned 8 Bit Multiplier Architecture
Example (cont’d)

```vhdl
SCAN_OUT => A_SCAN_OUT,
Q(0)=A_OUT(0), Q(1)=A_OUT(1),
Q(2)=A_OUT(2), Q(3)=A_OUT(3),
Q(4)=A_OUT(4), Q(5)=A_OUT(5),
Q(6)=A_OUT(6), Q(7)=A_OUT(7));

C1 : DFF generic map(5 ns, 1 ns)
  port map( D => CARRY_OUT, CLK => CLK,
            ENABLE => C_ENABLE, Q => C_OUT);

M1 : REG8_STR generic map(4 ns, 1 ns)
  port map( D => MULTIPLICAND, CLK => CLK,
            ENABLE => M_ENABLE, Q => M_OUT);

Q1 : SHIFT_REG8_STR generic map(6 ns, 1 ns)
  port map( D(0) => MULTIPLIER(0),
            D(1) => MULTIPLIER(1),
            D(2) => MULTIPLIER(2),
```
Unsigned 8 Bit Multiplier Architecture
Example (cont’d)

D(3) => MULTIPLIER(3),
    D(4) => MULTIPLIER(4),
    D(5) => MULTIPLIER(5),
    D(6) => MULTIPLIER(6),

D(7) => MULTIPLIER(7),
    CLK => CLK,
    ENABLE => Q_ENABLE,
    SCAN_IN => A_SCAN_OUT,
    SHIFT => Q_MODE,

Q(0) => PRODUCT(0),
    Q(1) => PRODUCT(1),
    Q(2) => PRODUCT(2),
    Q(3) => PRODUCT(3),
Unsigned 8 Bit Multiplier Architecture

Example (cont’d)

Q(4) => PRODUCT(4),
Q(5) => PRODUCT(5),
Q(6) => PRODUCT(6),
Q(7) => PRODUCT(7));

-- ALU

ALU1 : ALU_STR generic map(8 ns)
    port map(A => M_OUT, B => A_OUT,
         MODE => A_RESET,
         CIN => GND,
         SUM => ALU_OUT,
         COUT => CARRY_OUT);

-- CONNECT A REGISTER OUTPUT TO PRODUCT
PRODUCT(15 DOWNTO 8)<=A_OUT(7 DOWNTO 0);
end STRUCTURAL;
Configuration

- Primary Design Unit
- Binds Component Instances to Entities
- Allows Specification of instance specific Generic values
- Creates a simulatable object
- Facilates port re-mapping
Explicit Configuration

entity XR2 is
  port (........);
end XR2;

architecture FAST of XR2 is
  ...
end FAST;

architecture SLOW of XR2 is
  ...
end SLOW;

Configuration ONE of XR2 is for FAST end for;
end ONE;

Configuration TWO of XR2 is for SLOW end for;
end TWO;

Minimal configuration associates a name with entity-architecture pair
Explicit Configuration

configuration FIRST_ONE of MUX is
  for DATAFLOW
  end for;
end FIRST_ONE;

- A configuration statement selects a particular architecture of an entity (for simulation)
- When no explicit configuration exists, the latest compiled architecture is used (null configuration)
Choose a Configuration to Simulate

vhdldbx FIRST_ONE

config FIRST_ONE

arch DATAFLOW

entity MUX

config SEC_TIME

arch STRUCT

vhdlsim SEC_TIME
Components

- Component declaration is:
  - Primary design unit
  - Defines the binding of some or all of the components in your design description to corresponding lower-level entities and architectures.
  - Can form a simple part list for your design or
  - Can be written to contain detail information about how each component is “wired into” the rest of the design.
Example of Configuration

configuration THIS_BUILD of BOARD is
  for STRUCTURE
    for U1: Chip
      use entity WORK.CHIP(A1);
      port map(RESET => GND, CIN => 1....);
    end for;
  end for;
end THIS_BUILD;