Ports and Their Usage

- **Port Modes**
  - *in* reads a signal
  - *out* writes a signal
  - *inout* reads or writes a signal
    - Can be connected to multiple signals
  - *buffer* reads or writes a signal
    - Can be connected to only one signal
  - *linkage* special use
Illegal Port Usage

- Signal Defined As “Out” Cannot Be Used As Signal Connected to Internal Device.
- Need to Create Internal Signal.

ENTITY (W, X, Y: IN bit; A, B: OUT bit);
Constant-Valued Signals

May Want to Hard-Wire an Input Signal to a Particular Value

- Control signal in a model of real device, *e.g.*, preset or clear on a register
- Inputs to MUX to implement a particular combinational logic circuit
- Not use all inputs to a NAND/NOR gate while minimizing loading output of driving gate
Constant-Valued Signals

Three Methods for Generating a Constant

- Define local signal with default equal to desired value
  - e.g., `signal Zero_Input : BIT := '0' ;`
- Directly define input as ‘0’ or ‘1’ (VHDL-93)
Constant-Valued Signals

- Set default inputs on entity and use keyword OPEN

```
ENTITY NAND_2 IS
    PORT ( A, B : IN BIT := '1';
           C : OUT BIT ) ;
END NAND ;

... X1: NAND_2 ( OPEN, W, Z ) ;
...```
Regular VHDL Structures

- Iterative Circuits Are Composed of Many Identical Circuits
  - Ripple-carry adder
  - RAM
  - Counters
  - Comparators
Generate Statement

- Use Generate Statement to Reduce Coding Effort
- Can Include Any Concurrent Statement Including Another Generate Statement
- Does Not Execute Directly, But Expands into Code Which Does Execute
Generate Statement

- Automatically Generates Multiple Component Instantiations
- Two Kinds of Statements
  - Iteration
    - FOR . . . GENERATE
  - Conditional
    - IF . . . GENERATE
Iteration

- Instantiates Identical Components
- **FOR** Syntax

```plaintext
identifier : FOR N IN 1 TO 8
            GENERATE concurrent-statements
            END GENERATE name ;

- $N$ is a constant and cannot be changed
- “name” is required
```
Conditional

- Takes Care of Boundary Conditions
- **IF** Syntax

```plaintext
identifier : IF (boolean expression)
  GENERATE
  concurrent-statements
  END GENERATE name ;
```

- Cannot use “else” or “ifelse” clauses
Generate e.g., R-C Adder

```vhdl
ENTITY RCAAdder_16 IS
  PORT
  ( A, B : IN Bit_Vector (15 downto 0);
    Cforce : IN Bit;
    Sum : OUT Bit_Vector(15 downto 0);
    Cout : OUT Bit );
END RCAAdder_16;
```
Generate e.g., R-C Adder

ARCHITECTURE Generate_S OF RCAdder_16 IS

COMPONENT Full_Adder
--defined elsewhere

PORT ( A, B, Cin : IN bit ;
      S, Cout : OUT bit );

END COMPONENT Full_Adder;

SIGNAL Int_C : BIT_VECTOR (15 DOWNTO 0);
Generate e.g., R-C Adder

```
BEGIN    --RC Adder
All_Bits:
  FOR I IN 15 DOWNTO 0 GENERATE
  LSB :
    IF (I = 0) GENERATE
      BEGIN
        S0: Full_Adder
        PORT MAP ( A(I), B(I), Cforce, 
                   Sum(I), Int_C(I) );
      END GENERATE S0 ;
```

Generate e.g., R-C Adder

Middle_bits:

\[
\text{IF } ( I < 15 \ \text{AND} \ I > 0 ) \ \text{GENERATE} \\
\text{BEGIN} \\
\text{SI: Full_Adder} \\
\quad \text{PORT MAP} \ ( A(I), B(I), \text{Int}_C(I-1), \\
\quad \quad \text{Sum}(I), \text{Int}_C(I) ); \\
\quad \text{END GENERATE SI;} \\
\]
Generate \textit{e.g.}, R-C Adder

\begin{verbatim}
MSB:
\textbf{IF ( I = 15 ) GENERATE BEGIN}
  S15: Full_Adder
    \textbf{PORT MAP ( A(I), B(I), Int_C(I-1),}
    \hspace{1cm} \textbf{Sum(I), Cout );}
  \textbf{END GENERATE MSB;}
\textbf{END GENERATE All_Bits}
END Generate_S;
\end{verbatim}
Unconstrained Ports

- Entity Declarations Can Have Ports Defined Using Arrays Without Explicitly Including the Size of the Array
- Leads to General Specification of Iterative Circuit
- Uses Predefined Array Attribute `LENGTH`
Generate *e.g.*, R-C Adder

```vhdl
ENTITY RCAdder_N IS
  PORT ( A, B : IN Bit_Vector;
         Cforce : IN Bit;
         Sum : OUT Bit_Vector;
         Cout : OUT Bit );
END RCAdder_N ;
```
Generate e.g., R-C Adder

ARCHITECTURE Generate_S OF RCAAdder_N IS

COMPONENT Full_Adder --defined elsewhere

PORT ( A, B, Cin : IN bit ;
      S, Cout : OUT bit ) ;

END COMPONENT Full_Adder ;

SIGNAL Int_C : BIT_VECTOR

   ( (A’LENGTH - 1) DOWNTO 0) ;
BEGIN --RC Adder

All_Bits:

FOR I IN (A'LENGTH -1) DOWNTO 0 GENERATE LSB:

IF (I = 0) GENERATE BEGIN

S0: Full_Adder

PORT MAP ( A(I), B(I), Cforce, Sum(I), Int_C(I) );

END GENERATE S0 ;
Generate \textit{e.g.,} R-C Adder

Middle\_bits:

\begin{verbatim}
IF ( I < ( A'LENGTH - 1 ) AND I > 0 ) 
GENERATE
BEGIN
SI: Full_Adder
    PORT MAP ( A(I), B(I), C(I-1),
                Sum(I), Int_C(I) );
END GENERATE SI ;
\end{verbatim}
Generate e.g., R-C Adder

MSB:

\[ \text{IF ( I = A'\text{LENGTH} - 1 ) GENERATE} \]
\[ \text{BEGIN} \]

\[ \text{SN: Full_Adder} \]

\[ \text{PORT MAP ( A(I), B(I), INT_C(I-1),} \]
\[ \text{Sum(I), Cout );} \]

\[ \text{END GENERATE MSB;} \]

\[ \text{END GENERATE All_Bits} \]

\[ \text{END Generate_S ;} \]
Arithmetic Operators

- Four Classes of Operators in VHDL
  - Logic
  - Relational
  - Shift
  - Arithmetic
    - +, -, *, /, **
    - mod, rem, abs, abs
    - (concatenation)
Number Types

Integer Literals

MAX_INT = Maximum Positive Value of Integer
Positive Numbers = \{ 1, 2, 3, \ldots , MAX_INT \}
Counting Numbers = \{ Positive Numbers \} \cup \{ 0 \}
Integers = \pm \text{Counting Numbers}
Number Types

- Floating Point Literals
  - Number containing a radix point
  - Only one type, REAL
  - Range of values is implementation dependent
Type Conversion

- There is NO implicit type conversion
  - Mixed type operations are not allowed

- Explicit type conversion IS allowed
  - e.g., INTEGER (4.8), REAL (5)
Arithmetic Operator Limitations

- REM and MOD Operators Only Defined for Integers
- ** (Exponentiation) Accepts Both Integer and Real Arguments With Restrictions
  - REAL can only be raised to integer power
  - INTEGER can only be raised to positive power
Writing Literals

- Underscore Can Be Inserted Anywhere for Readability
- Exponential Notation Allowed for Both Integers and Reals
- Any Radix From 2 to 16 Is Allowed
  - radix # number_in_radix #
Literal Examples

- **Decimal Integer, e.g.**,
  - 43800, 43_800, 438e2, 438E+2
- **Other Radix Integer, e.g.**,
  - 2#1101011#, 3#21201#, 16#faff0#
- **Real, e.g.**,  
  - 0.0, 0.0_26, 3.8e-4, 9.8E+4
- **Other Radix Real, e.g.**,  
  - 2#11.011#, 3#22.1#, 7#46.31#e-1
End of Lecture

- Ports
- Constants
- Regular Structures
- Arithmetic Operators
- Literals