Language Concepts

Introductory VHDL Methodology

Objectives

After completing this module, you will be able to…

- State the VHDL design units
- Write VHDL 'entity' & 'architecture' description
- Build hierarchical units using instantiation
- State the four stages of compilation
- Instantiate library macro using component declaration
- Insert comments in VHDL code
- Define a VHDL process
- Differentiate concurrent and sequential statements

Inference Vs. Instantiation

- **Given that synthesis and implementation are implicitly tool and technology dependent, occasionally the need to trade-off maximum code portability against design optimal implementation within a target technology must be considered**

Z <= A + B ;

The "+" operator infers that a generic Adder be built, without consideration of device level optimization

Code portability is maintained

component
ADSU8
port (…
…end component

This is in effect a direct call to the Xilinx library macro ADSU8. It is an 8-bit add/subtract library macro instantiation

Device optimization is achieved

VHDL Design Units

Design Units

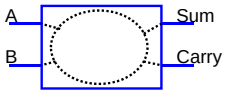
- **Entity**
- **Architecture**
- **Process**
- **Configuration**
- **Package**
- **Library**

VHDL is composed of design units. These consist of primary, and any dependent secondary units

Entity

The entity describes the external interface to the design unit, along with attributes relating to the interface

```
entity  Half_Add is
    port  ( A, B : in  std_logic ;
    Carry, Sum : out  std_logic) ;
end  Half_Add ;
```



Architecture

- **The architecture describes the internal operation of its associated entity. Multiple architectures can exist for each entity, each describing one possible implementation**

```
architecture My_Arch of  Half_Add is
begin
    Sum <= A xor B ;
    Carry <= A and B ;
end  architecture  My_Arch ;
```

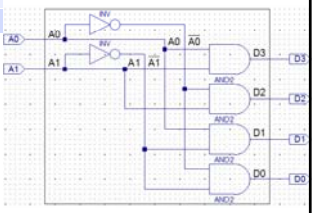
Note: VHDL'93

Allows for the optional reserved word *entity* or *architecture* after the reserved word *end* in their respective declarations

Example: 2 to 4 Decoder

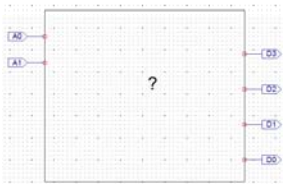
Input		Output			
A1	A0	D3	D2	D1	D0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$$D0 = \overline{A1} \bullet \overline{A0}$$
$$D1 = \overline{A1} \bullet A0$$
$$D2 = A1 \bullet \overline{A0}$$
$$D3 = A1 \bullet A0$$



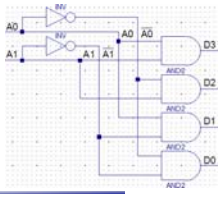
Example: 2 to 4 Decoder (cont.)

```
entity  Decoder2to4 is
    port  ( A0, A1 : in  bit;
            D0, D1, D2, D3 : out  bit);
end  Decoder2to4;
```



Example: 2 to 4 Decoder (cont.)

```
architecture Decoder2to4 is
   begin D0 <= (not A1) and (not A0);
         D1 <= (not A1) and A0;
         D2 <= A1 and (not A0);
         D3 <= A1 and A0;
end  Decoder2to4;
```



Process

- **The process contains sequential statements --that is, actions to be executed in sequence**

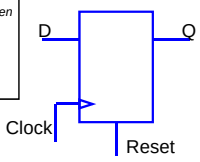
  Each process has a means of being triggered, either by changes on signals into the process, or specific conditions implied in the *wait* statement

```
architecture RTL  of  DFF is
begin
. . .
process (Clock, Reset)
begin
  if  ( Reset = '1' )  then
    Q <= '0';
   elsif  (Clock'event and Clock = '1' )
then
    Q <= D ;
  end if ;
end process ;
. . .
end RTL;
```


Example: DFF

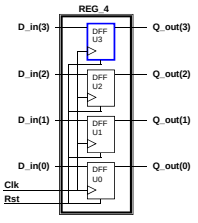
```
architecture RTL  of  DFF is
begin
   process (Clock, Reset)
   begin
     If  (Reset = '1' ) then
     Q <= '0' ;
    elsif ( Clock'event  and  Clock = '1' )  then
    Q <= D ;
    end if ;
  end process  ;
end  architecture   RTL ;
```

```
entity  DFF is
   port ( D, Clock : in  std_logic ;
          Reset : in std_logic ;
          Q : out  std_logic) ;
end  entity  DFF ;
```



Hierarchical Representation

```
architecture  Structural  of  REG_4  is

component  DFF
   port ( D, Clock : in  std_logic ;
          Reset : in  std_logic;
          Q : out  std_logic  ) ;
end component  ;

begin
U3 : DFF port map ( D_in(3), Clk, Rst, Q_out(3));
U2 : DFF port map      ( D_in(2), Clk, Rst, Q_out(2));
U1 : DFF port map      ( D_in(1), Clk, Rst, Q_out(1));
U0 : DFF port map      ( D_in(0), Clk, Rst, Q_out(0));
end  Structural;
```




```
entity  REG_4  is
   port ( D_in : in  std_logic_vector (3 downto 0);
          Clk, Rst : in  std_logic;
          Q_out : out  std_logic_vector (3 downto 0)):
end   REG_4;
```

Signal Association

- There are two methods of associating signals with their respective ports
- **Positional Association: Signals are listed in strict order of port listing in declaration**
  - *U1: DFF port map (D_in, Clk, Rst, Q_out) ;*

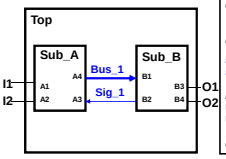
```
component DFF
  port ( D, Clock : in std_logic ;
  Reset : in std_logic ;
  Q : out std_logic ) ;
end component ;
```

- **Named Association: Ports and signals are listed explicitly, order independent *(This is recommended)***
  - *U1: DFF port map (D =>D_in(1), Clock =>Clk, Reset =>Rst, Q =>Q_out(1)) ;*

Signal Declaration

- **If internal signals are required in the structural description, they must be explicitly declared**

```
entity Top is
  port ( I1, I2 : in std_logic;
         O1, O2 : out std_logic) ;
end Top ;
```



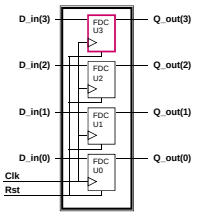

```
architecture Structural of Top is

component Sub_A
  port ( A1,A2,A3 : in std_logic ;
         A4 : out std_logic_vector (3 downto 0 ) ;
end component ;

component Sub_B
  port ( B1: in std_logic_vector (3 downto 0 ) ;
         B2,B3,B4 : out std_logic ;
end component ;

signal Bus_1 : std_logic_vector (3 downto 0 ) ;
signal Sig_1:  std_logic ;

begin
U0 : Sub_A port map ( I1, I2, Sig_1, Bus_1 ) ;
U1 : Sub_B port map ( Bus_1, Sig_1, O1, O2 ) ;

end Structural ;
```


Macro Instantiation

```
entity REG_4 is
  port ( D_in : in std_logic_vector (3 downto 0) ;
  Clk, Rst : in std_logic ;
  Q_out : out std_logic_vector (3 downto 0)) :
end REG_4 ;
```

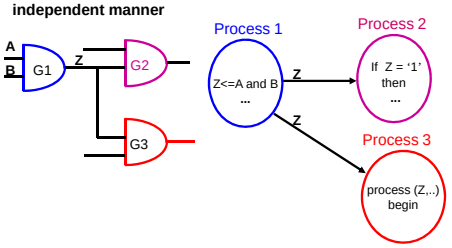
```
architecture Xilinx_Struct of REG_4 is
component FDC
  port ( D : in std_logic ;
  Clock, Reset : in std_logic ;
  Q : out std_logic ) ;
end component ;
begin
U3 : FDC port map ( D=>D_in(3), Clock=>Clk,
Reset=>Rst, Q=> Q_out(3)) ;
U2 : FDC port map ( D=>D_in(2), Clock=>Clk,
Reset=>Rst, Q=> Q_out(2)) ;
U1 : FDC port map ( D=>D_in(1), Clock=>Clk,
Reset=>Rst, Q=> Q_out(1)) ;
U0 : FDC port map ( D=>D_in(0), Clock=>Clk,
Reset=>Rst, Q=> Q_out(0)) ;
end Xilinx_Struct ;
```

Macro/component instantiation from target library may be helpful for chip level optimization!
i.e. Xilinx XC4000XL



Processes are Concurrent

- **In hardware modeling, the concept of concurrency is essential, i.e. a logic change on signal Z, the output of the gate G1 is seen at the input of G2 and G3 in a concurrent and independent manner**



Package

- **The VHDL design unit *package* is used to store data that will be accessed by multiple modules. This usually includes constants, data_types, sub-types, sub-program declarations, etc**

```
package My_Pack is
constant. . .
. . .
function. . .
. . .
component . . .
. . .
subtype. . .

end package My_pack ;
```

```
library IEEE;
use IEEE.std_logic_1164.all ;
. . .
use work.My_Pack.all ;

entity . . .
```


Package Body

- **The *package body* is a dependent unit of the package. It is used to store details of the objects declared in the package. This includes sub-programs, algorithms, etc**

```
package My_Pack is
constant. . .
. . .
function (bv_to_integer..
. . .
component . . .
. . .
subtype. . .

end My_Pack ;
```

declaration

details

```
package body My_Pack is
function. . bv_to_integer (BV: bit_v..
            return integer is
variable …
  begin
    for index in BV'range loop
      . . . .
. . .
end My_Pack ;
```

Library

- **Usually referred to as the *work* library. This is an actual sub-directory or physical location used to store all compiled design units**
- **Each simulator or synthesis tool will create such a structure**

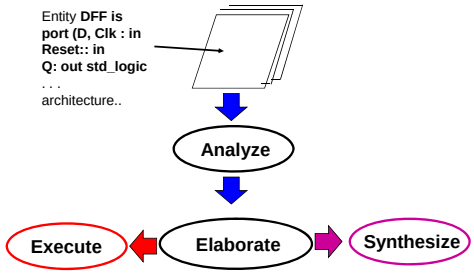
Design Unit	Identifier
entity	**HALF_ADD**
entity	DFF
entity	REG4
Configuration	CFG_REG4
. . .	
architecture	**XILINX**
architecture	STRUCTURAL

Example ***work*** library contents

Compilation

- **There are four levels of processing that may take place for a VHDL hardware model**
  - *Analyzation*
    - The design unit is checked for syntactic errors. Once finalized, it is stored in the work directory
  - *Elaboration*
    - The design hierarchy is fleshed out (flattened) starting from the top. A unique copy of each sub-module instance is created
  - *Execution*
    - The model is simulated in discrete time steps. This is driven primarily by events on signals that trigger processes
  - *Synthesis*
    - A netlist description of the logic is generated in either an industry standard or vendor specific format

Compilation Path



Compilation Order

- **Because of the primary and secondary design unit relationship and the ability to instantiate lower level modules, the compilation process is governed by a strict dependency order**
- **Entities must be analyzed before corresponding architectures**
- **Packages must be analyzed before dependent *package bodies***
- **Any module before it is referenced by another (Bottom -> Up)…!**

Comments

- **Comments can greatly enhance the readability of the code, provide useful documentation and make the user's intent absolutely clear**

```
- - Comments begin with two dash characters
- - They continue only till the end of the line < c r >

- - A comment on multiple lines will require the double
- - dash on each line as shown here

A_OUT <= '1' ; - - Comments can begin anywhere on a line
```


Review Questions

- **Where are sequential statements placed?**
- **What must be done to make the contents of a package visible within a given declaration?**
- **What are the necessary levels of processing for a VHDL model for simulation? For synthesis?**
- **How would you enter a multi-line comment?**

Answers

- **Where are sequential statements placed?**
  - *Within the process*
- **What must be done to make the contents of a package visible within a given declaration?**
  - *The 'use' clause*
- **What are the necessary levels of processing for a VHDL model for simulation? For synthesis?**
  - *Analyzation, Elaboration, Execution…..Synthesis*
- **How would you enter a multi-line comment?**
  - *Double dash '--' on each line*

Summary

- **VHDL is composed of primary and secondary design units**
- **There is a strict dependency order for compilation**
- **VHDL contains concurrent and sequential statements**
- **All analyzed design units are stored in the design or *work* library**