Lecture 7: VHDL (1)

Computer Aided Digital Design
EE 3109
Gopi K. Manne
Fall 2007

VHDL?

VHDL (Very High Speed Integrated Circuit) Hardware Description Language
VHDL is a “Description Language”, not a programming language like C, Pascal, Java.
Originally developed for Department of Defense, released first in 1985.
Not technology specific but predominantly used for digital system design.

ASIC Chips

- Mostly logic design in Verilog and VHDL
- Little transistor-level work
- Synthesis/Place/Route cells from logic library - tools
- Library includes
  - standard cells (i.e. ANDs, ORs, Flip Flops)
  - memories
  - I/O blocks (input and output of chip)
  - Phase-Lock Loops
  - probably developed in another company (i.e. Artisan)
- Fabricate in a foundry - Fabless Design House
- 20 to 30 engineers in one year to design chip

VHDL Capabilities

- Exchange medium between chip vendors and CAD tool users.
- Supports both synchronous and asynchronous timing models.
- Various delay constraints can be described.
- Allows defining new data types.
- IEEE and ANSI standard, therefore very portable.
- Supports a wide range of abstraction levels, design, modeling styles.

VHDL Primary Design Constructs

- Five primary design constructs, also known as “Design Units” used to describe logic
  - Entity – The interface of a logic circuit.
  - Architecture – Particular implementation of an entity
  - Configuration – Binds entities, architectures, and component declarations
  - Package declaration – Convenient way to define and group functions, procedures, types, components etc.
  - Package body – Implementation of the functionality exposed by package declaration.

Full Custom CMOS design

- Only done when performance is king
- Unit cost is medium - $50 a chip
- Performance is best – 3 GHz processor
  - Development cost (one time cost) is very high
    - 1000 engineers working for 2 years and costed at $200k/year
    - $400 million dollar development cost – wow!
    - turn around and sell 100 million chips for $150 each
    - $15 billion income – wow! development cost insignificant
      - moral of the story: get a lot of engineers to hand tweak design for max performance
Design Unit (1) : ENTITY

- Defines the interface of the hardware module to the outside environment in which it is used
- Entity declaration syntax

```
entity entity_name is
  generic specifications
  port specifications
  entity statements -- passive statements (generally not used)
end entity_name;
```

Port – pass declared signals to and from external design
Generic – pass parameters from higher level design to the module

Example

```
generic (delay : time:=25ns);
port (a, b, cin: in bit; sum, cout: out bit);
```

Design Unit (2) : ARCHITECTURE

- Functionality description of system declared by entity
- Can be behavioral or structural in terms of simpler components
- Architecture/Entity pair defines a circuit
- Multiple architectures can be defined for an entity
- Architecture declaration syntax

```
architecture architecture_name of entity_name is
  signal declarations
  component declarations
  description statements
end architecture_name;
```

Signal – internal signals generated inside the module
Component – declaring any external entities used in this module
Description – behavioral or structural

Example

```
architecture behavioral of full_adder is
  signal temp : bit;
begin
  P1: temp<=a xor b;
P2: sum<=cin xor temp;
P3: cout<=( and b) or (a and cin) or (b and cin);
end behavioral;
```

Topics

- Identifiers
- Data Types
- Objects
- Operators
- Modeling Styles

Identifiers

- Signals, Variables, constant names
- Names of entities, architectures, & Packages
  - Letters, numerals, underscores
  - Case insensitive
  - No two consecutive underscores
  - Must begin with a letter
  - No VHDL keyword
Identifiers (contd..)

- mySignal_23 -- normal identifier
- rdy, RDY, Rdy -- identical identifiers
- vector & vector -- X: special character
- last of Zout -- X: white spaces
- idle_state -- X: consecutive underscores
- 24th_signal -- X: begins with a numeral
- case, register -- X: VHDL keywords
- http://www.peakfpga.com/vhdlref/

Scalar Data Types

- **Integer**
  - Minimum range for any implementation as defined by standard: -2,147,483,647 to 2,147,483,647
  - Example assignment to a variable of type integer:

```vhdl
ARCHITECTURE test_int OF test IS
BEGIN
  PROCESS (X)
  VARIABLE a: INTEGER;
  BEGIN
    a := 1;  -- OK
    a := -1;  -- OK
    a := 1.5;  -- illegal
    END PROCESS;
  END test_int;
END ARCHITECTURE;
```

Scalar Data Types (Cont.)

- **Real**
  - Minimum range for any implementation as defined by standard: -1.0E38 to 1.0E38
  - Example assignment to a variable of type real:

```vhdl
ARCHITECTURE test_real OF test IS
BEGIN
  PROCESS (X)
  VARIABLE a: REAL;
  BEGIN
    a := 1.3;  -- OK
    a := -7.5;  -- OK
    a := 1;  -- illegal
    a := 1.7E13;  -- OK
    a := 5.3 ns;  -- illegal
    END PROCESS;
  END test_real;
END ARCHITECTURE;
```

Scalar Data Types (Cont.)

- **Enumerated**
  - User specifies list of possible values
  - Example declaration and usage of enumerated data type:

```vhdl
TYPE binary IS (ON, OFF);
... some statements ...
ARCHITECTURE test_enum OF test IS
BEGIN
  PROCESS (X)
  VARIABLE a: binary;
  BEGIN
    a := ON;  -- OK
    a := OFF;  -- OK
    ... more statements ...
    END PROCESS;
  END test_enum;
END ARCHITECTURE;
```

Scalar Data Types (Cont.)

- **Physical**
  - Require associated units
  - Range must be specified
  - Example of physical type declaration:

```vhdl
TYPE resistance IS RANGE 0 TO 10000000 UNITS ohm;
  KOhm = 1000 ohm;  -- i.e. 1 kΩ
  Mohm = 1000 kOhm;  -- i.e. 1 MΩ
  END UNITS;
```

- Time is the only physical type predefined in VHDL standard
Composite Data Types

- **Array**
  - Used to group elements of the same type into a single VHDL object
  - Range may be unconstrained in declaration
  - Range would then be constrained when array is used
  - Example declaration for one-dimensional array (vector):
    ```vhdl
    TYPE data_bus IS ARRAY(0 TO 31) OF BIT;
    VARIABLE X : data_bus;
    VARIABLE Y : BIT;
    Y := X(12);  -- Y gets value of element at index 12
    ```

  - Range would then be constrained when array is used
    - Example declaration for one-dimensional array (vector):
      ```vhdl
      TYPE reg_type IS ARRAY(15 DOWNTO 0) OF BIT;
      VARIABLE X : reg_type;
      VARIABLE Y : BIT;
      Y := X(4);  -- Y gets value of element at index 4
      ```

- **Records**
  - Used to group elements of possibly different types into a single VHDL object
  - Elements are indexed via field names
  - Examples of record declaration and usage:
    ```vhdl
    TYPE Binary IS (ON, OFF);
    TYPE switch_info IS
      RECORD
        status : Binary;
        IDnumber : INTEGER;
      END RECORD;
    VARIABLE switch : switch_info;
    switch.status := ON;  -- status of the switch
    switch.IDnumber := 30;  -- e.g. number of the switch
    ```

Access Data Type

- **Access**
  - Analogous to pointers in other languages
  - Allows for dynamic allocation of storage
  - Useful for implementing queues, fifos, etc.

Subtypes

- **Subtype**
  - Allows for user defined constraints on a data type
    - e.g. a subtype based on an unconstrained VHDL type
  - May include entire range of base type
  - Assignments that are out of the subtype range are illegal
    - Range violation detected at run time rather than compile time because only base type is checked at compile time
  - Subtype declaration syntax:
    ```vhdl
    SUBTYPE name IS base_type RANGE <user range>;
    ```
  - Subtype example:
    ```vhdl
    SUBTYPE first_ten IS INTEGER RANGE 0 TO 9;
    ```

Lab 6

- **4 to 1 Multiplexer**
  - INPUTS
    - D0, D1, D2, D3 and Enable
    - S1, S0 as your control signals
  - OUTPUTS
    - Z is the output
    ```plaintext
    | s0 | s1 | Z |
    |----|----|---|
    | 0  | 0  | D0|
    | 0  | 1  | D1|
    | 1  | 0  | D2|
    | 1  | 1  | D3|
    ```