Circuit Design with VHDL

Chapter 6: Code structure and Composition

Instructor: Ali Jahanian

Outline

- I-CIRCUIT DESIGN
  - 2 Code Structure
    - 2.1 Fundamental VHDL Units
    - 2.2 LIBRARY Declarations
    - 2.3 ENTITY
    - 2.4 ARCHITECTURE
    - 2.5 Introductory Examples

Fundamental VHDL Units

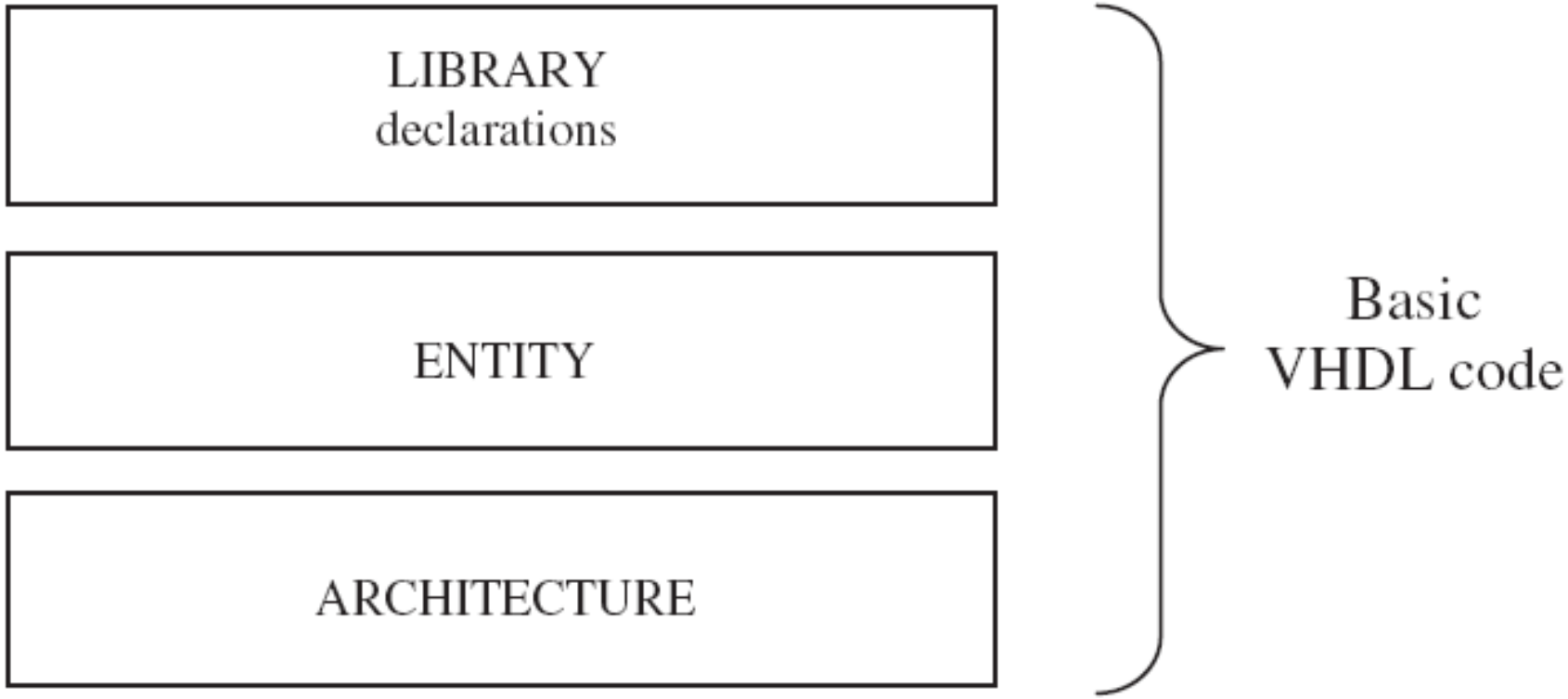

Figure 2.1
Fundamental sections of a basic VHDL code.

LIBRARY

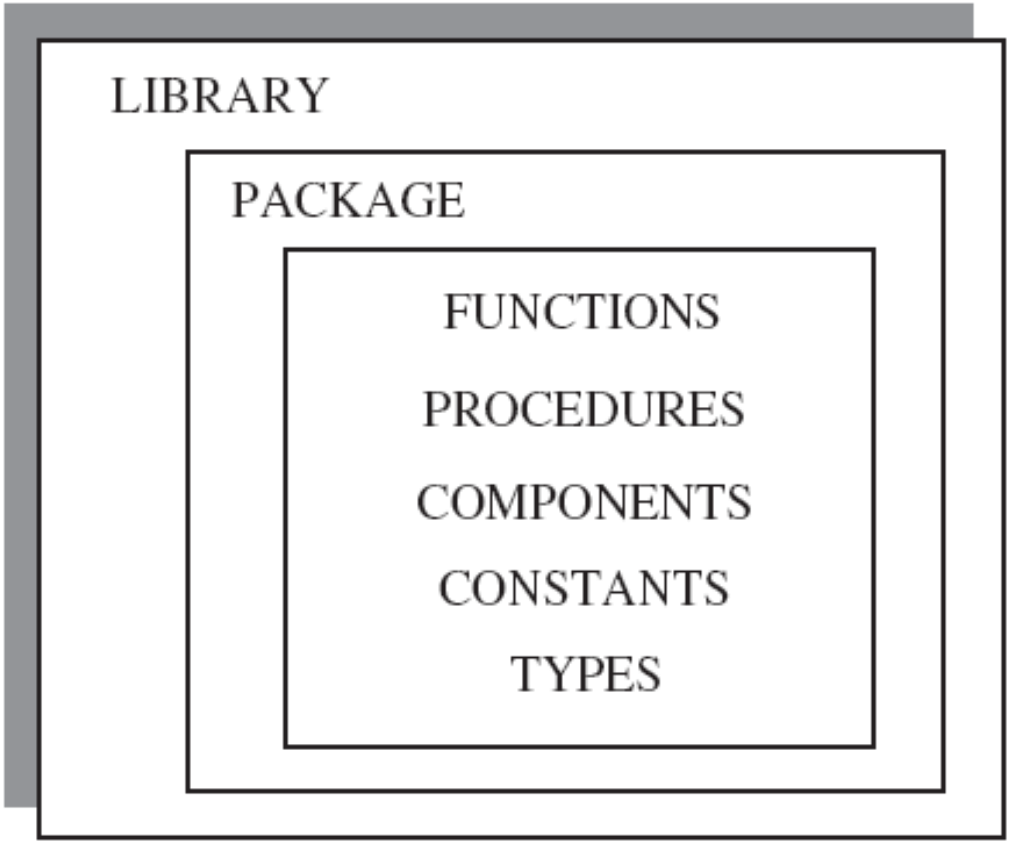

Figure 2.2
Fundamental parts of a LIBRARY.

LIBRARY Declarations

- **To declare a LIBRARY**

```
LIBRARY library_name;
USE library_name.package_name.package_parts;
```

At least three packages, from three different libraries, are usually needed in a design:

- *ieee.std_logic_1164* (from the *ieee* library),
- *standard* (from the *std* library), and
- *work* (*work* library).

LIBRARY Declarations …

```
LIBRARY ieee;                          -- A semi-colon (;) indicates
USE ieee.std_logic_1164.all;           -- the end of a statement or

LIBRARY std;                           -- declaration, while a double
USE std.standard.all;                  -- dash (--) indicates a comment.

LIBRARY work;
USE work.all;
```

specifies a multi-level logic system

***std* is a resource library (data types)**

where we save our design

only the *ieee* library must be explicitly written

ieee library

- *std_logic_1164*
  - STD_LOGIC (8 levels)
  - STD_ULOGIC (9 levels)
- *std_logic_arith*
  - the SIGNED and UNSIGNED data types
  - arithmetic and comparison operations
  - data conversion functions
- *std_logic_signed*
- *std_logic_unsigned*

ENTITY

- **An ENTITY is a list with specifications of all input and output pins (PORTS) of the circuit.**

```
ENTITY entity_name IS
   PORT (
      port_name : signal_mode signal_type;
      port_name : signal_mode signal_type;
      ...);
END entity_name;
```

BIT, STD_LOGIC, INTEGER, etc.

IN, OUT, INOUT, or BUFFER

Signal modes

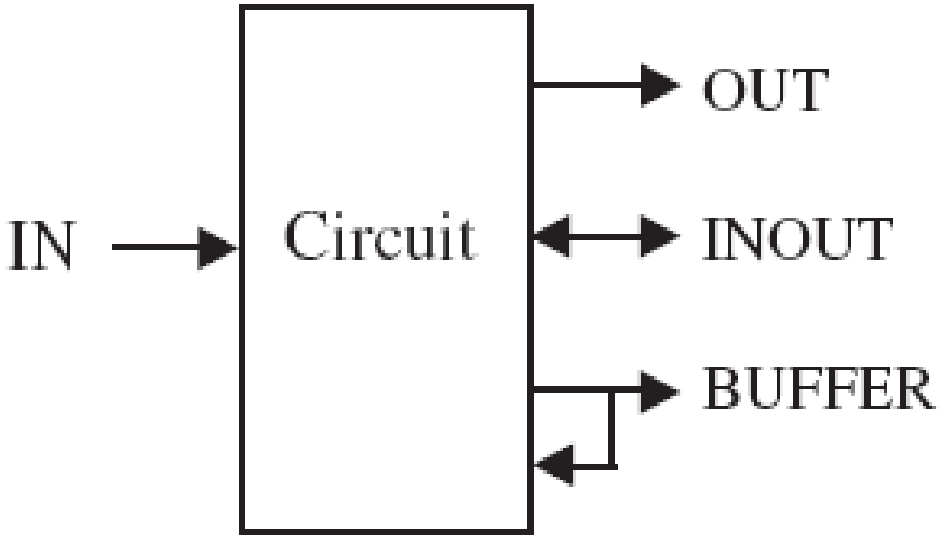

Figure 2.3
Signal *modes*.

Example (For ENTITY)

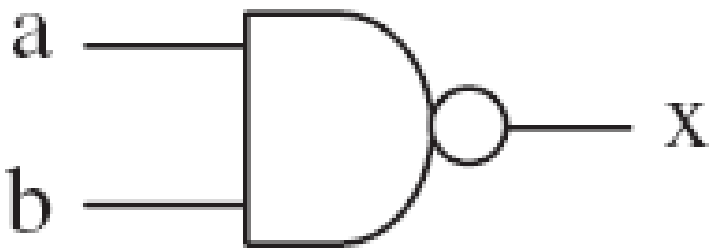

Figure 2.4
NAND gate.

```
ENTITY nand_gate IS
        PORT (a, b : IN BIT;
                     x : OUT BIT);
END nand_gate;
```

ARCHITECTURE

- **An architecture has two parts:**
  - **declarative part (optional),**
  - **code part**

```
ARCHITECTURE architecture_name OF entity_name IS
    [declarations]
BEGIN
    (code)
END architecture_name;
```

[©Hwan08]

Example (For ARCHITECTURE)

Figure 2.4
NAND gate.

```
ENTITY nand_gate IS
        PORT (a, b : IN BIT;
              x : OUT BIT);
END nand_gate;
```

```
ARCHITECTURE myarch OF nand_gate IS
BEGIN
        x <= a NAND b;
END myarch;
```

Introductory Examples

1- DFF with Asynchronous Reset

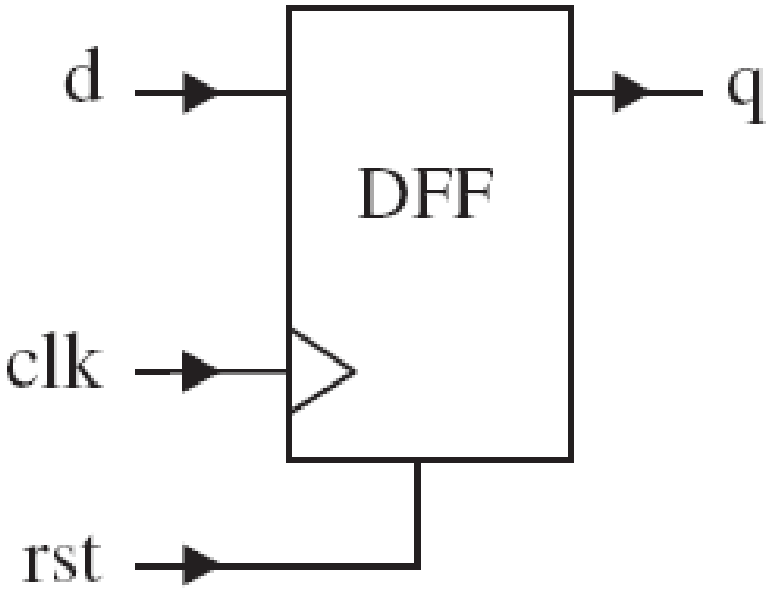

Figure 2.5
DFF with asynchronous reset.

Introductory Examples …

2 - DFF plus NAND Gate

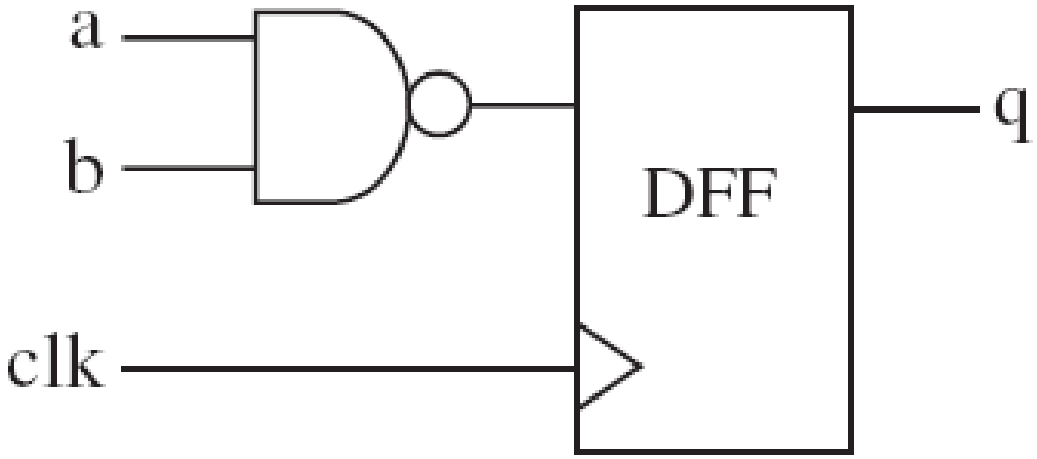

Figure 2.7
DFF plus NAND gate.

DFF with Asynchronous Reset

```
-------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------
ENTITY dff IS
   PORT ( d, clk, rst: IN STD_LOGIC;
          q: OUT STD_LOGIC);
END dff;
-------------------------------------
```

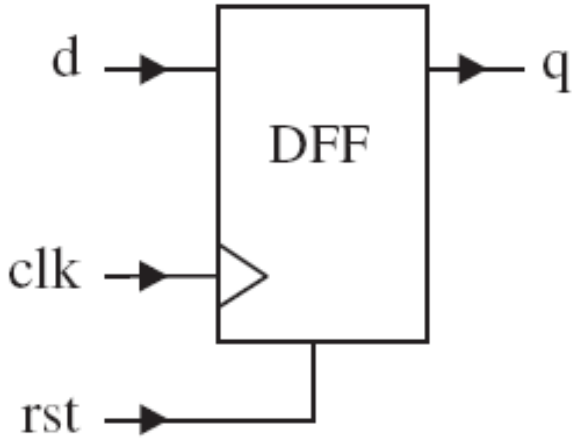

Figure 2.5
DFF with asynchronous reset.

DFF with Asynchronous Reset …

```
-------------------------------------
ARCHITECTURE behavior OF dff IS
BEGIN
    PROCESS (rst, clk)
    BEGIN
        IF (rst='1') THEN
            q <= '0';
        ELSIF (clk'EVENT AND clk='1') THEN
            q <= d;
        END IF;
    END PROCESS;
END behavior;
-------------------------------------
```

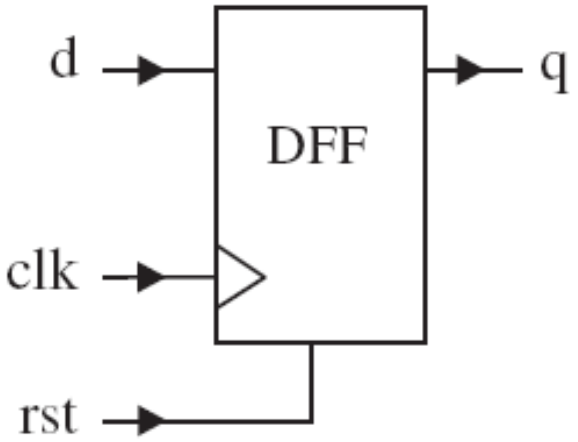

Figure 2.5
DFF with asynchronous reset.

Note: VHDL is not case sensitive.

DFF with Asynchronous Reset …

- Simulation result

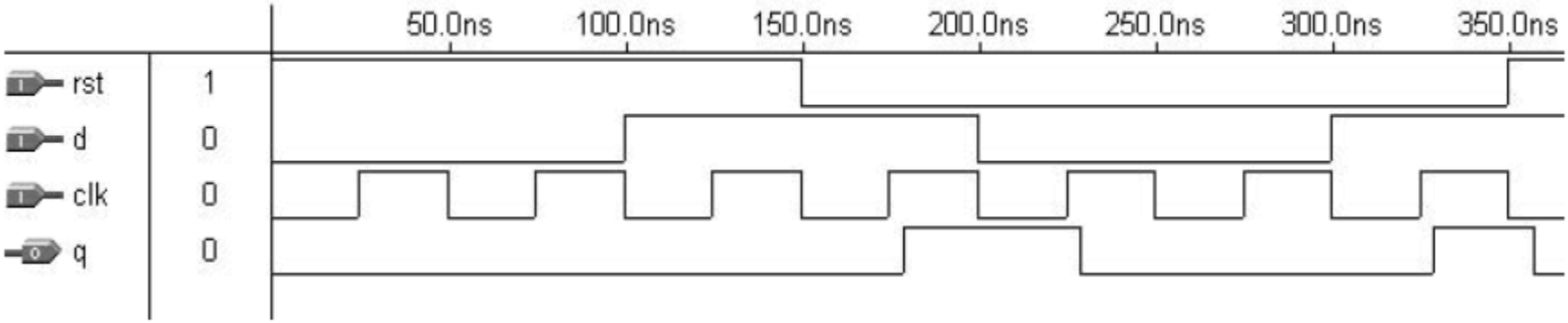

Figure 2.6
Simulation results of example 2.1.

DFF plus NAND Gate

```
-------------------------------------
ENTITY example IS
   PORT ( a, b, clk: IN BIT;
          q: OUT BIT);
END example;
-------------------------------------
ARCHITECTURE example OF example IS
   SIGNAL temp : BIT;
BEGIN
   temp <= a NAND b;
   PROCESS (clk)
   BEGIN
      IF (clk'EVENT AND clk='1') THEN q<=temp;
      END IF;
   END PROCESS;
END example;
-------------------------------------
```

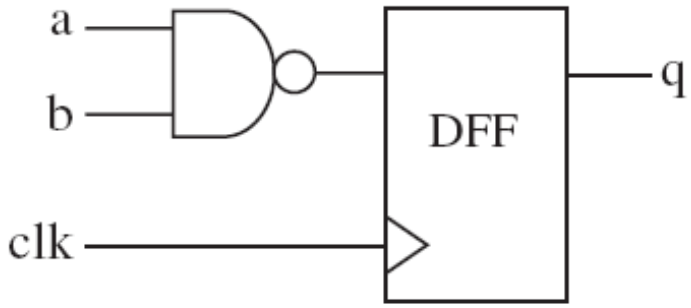

Figure 2.7
DFF plus NAND gate.

DFF plus NAND Gate …

- Simulation result

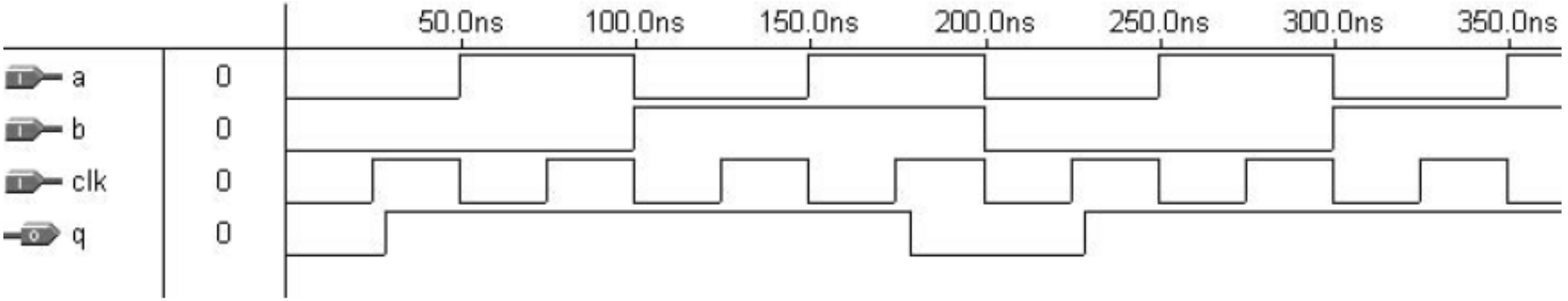

Figure 2.8
Simulation results of example 2.2.

Next session

3 Data Types

- 3.1 Pre-Defined Data Types
- 3.2 User-Defined Data Types
- 3.3 Subtypes
- 3.4 Arrays
- 3.5 Port Array
- 3.6 Records
- 3.7 Signed and Unsigned Data Types
- 3.8 Data Conversion
- 3.9 Summary
- 3.10 Additional Examples