Outline

- VHDL Fundamentals
- Design Entity
- Architecture
- Library
- STD_LOGIC
- Modeling Wires and Buses
- Types of VHDL Description
- Examples
Reading

- **Required**
  - P. Chu, *FPGA Prototyping by VHDL Examples*
    - *Chapter 1, Gate-level combinational circuit*

- **Recommended**
  - S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with VHDL Design*
    - *Chapter 2.10, Introduction to VHDL*
Review of Previous Classes - 1

- VHDL is a language for describing digital hardware used by industry worldwide

  - **VHDL** is an acronym for **V**ery **H**igh **S**peed **I**ntegrated **C**ircuit **H**ardware **D**escription **L**anguage
Review of Previous Classes - 2

- **Verilog**
  - Slightly better at gate/transistor level
  - Language style close to C/C++
  - Pre-defined data type, easy to use

- **VHDL**
  - Slightly better at system level
  - Language style close to Pascal
  - User-defined data type, more flexible

- Equally effective, personal preference
Review of Previous Classes - 3

**ASICS**

- High performance
- Low power
- Low cost (but only in high volumes)

**FPGAs**

- Off-the-shelf
- Low development costs
- Short time to the market
- Reconfigurability
VHDL Fundamentals - 1

- VHDL is case insensitive
VHDL Fundamentals - 2

General rules of thumb (according to VHDL-87)
Extended Identifiers
Allowed only in VHDL-93 and higher:
  • Enclosed in backslashes
  • May contain spaces and consecutive underscores
  • May contain punctuation and reserved characters within a name (!, ?, ., &, +, -, etc.)
  • VHDL keywords allowed
  • Case sensitive
Free Format

• VHDL is a “free format” language
  No formatting conventions, such as spacing or indentation imposed by VHDL compilers. Space and carriage return treated the same way.
Comments

- Comments in VHDL are indicated with a “double dash”, i.e., “--”
  - Comment indicator can be placed anywhere in the line
  - Any text that follows in the same line is treated as a comment
Review of Basic Gates - AND, OR, NOT
Review of Basic Gates NAND, NOR
DeMorgan’s Theorem and other symbols for NAND, NOR
Basic Gates – XOR
Basic Gates – XNOR
Design Entity

- Example: NAND Gate
Example VHDL Code
Design Entity
Entity Declaration

- Entity Declaration describes an interface of the component, i.e. input and output ports.

```
ENTITY nand_gate IS
  PORT ( a : IN STD_LOGIC;
         b : IN STD_LOGIC;
         z : OUT STD_LOGIC
       );
END nand_gate;
```
Entity declaration - simplified syntax
Architecture (Architecture body)
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY nand_gate IS
    PORT(
        a : IN STD_LOGIC;
        b : IN STD_LOGIC;
        z : OUT STD_LOGIC);
END nand_gate;

ARCHITECTURE dataflow OF nand_gate IS
BEGIN
    z <= a NAND b;
END dataflow;
Libraries
Library declarations - syntax & blocks
STD_LOGIC

What is **STD_LOGIC** you ask?

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY nand_gate IS
  PORT(
    a : IN STD_LOGIC;
    b : IN STD_LOGIC;
    z : OUT STD_LOGIC);
END nand_gate;

ARCHITECTURE dataflow OF nand_gate IS
BEGIN
  z <= a NAND b;
END dataflow;
```
BIT versus STD_LOGIC
STD_LOGIC type demystified
STD_LOGIC Rules

• In ECE 443, use `std_logic` or `std_logic_vector` for all *entity input or output ports*
  ▫ Do not use integer, unsigned, signed, bit for ports
  ▫ You can use them inside of architectures if desired
  ▫ You can use them in generics
• Instead use `std_logic_vector` and a conversion function inside of your architecture
  [Talk with your GTA to get more detailed information]
Modeling Wires and Buses

SIGNAL a : STD_LOGIC;

SIGNAL b : STD_LOGIC_VECTOR(7 DOWNTO 0);
Standard Logic Vectors

SIGNAL a: STD_LOGIC;
SIGNAL b: STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL c: STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL d: STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL e: STD_LOGIC_VECTOR(8 DOWNTO 0);

Examples:
Vectors and Concatenation

SIGNAL a: STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL b: STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL c, d, e: STD_LOGIC_VECTOR(7 DOWNTO 0);

a <= "0000";
b <= "1111";

Examples:
Types of VHDL Description: Convention used in this class

VHDL Descriptions

- dataflow
- structural
- behavioral
Structural Description
Behavioral Description
Example: XOR 3
Entity xor3_gate
Dataflow Architecture (xor3_gate)
ARCHITECTURE structural OF xor3_gate IS
SIGNAL U1_OUT: STD_LOGIC;
COMPONENT xor2
  PORT(
    I1 : IN STD_LOGIC;
    I2 : IN STD_LOGIC;
    Y : OUT STD_LOGIC
  );
END COMPONENT;
BEGIN
  U1: xor2 PORT MAP (I1 => A,
                      I2 => B,
                      Y => U1_OUT);
  U2: xor2 PORT MAP (I1 => U1_OUT,
                      I2 => C,
                      Y => Result);
END structural;
Data-flow VHDL: Example
Data-flow VHDL: Example
Another examples could be found in
-Textbook Ch. 1: 2-bit Comparator
-Course Website: 2:1 MUX
Library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity RC5_core is
  port(
    clock, reset, encr_decr: in std_logic;
    data_input: in std_logic_vector(31 downto 0);
    data_output: out std_logic_vector(31 downto 0);
    out_full: in std_logic;
    key_input: in std_logic_vector(31 downto 0);
    key_read: out std_logic;
  );
end AES_core;
Next Class

- Operators and Data Types
- Priority of logic and relational operators
- when-else and with-select-when
- RT level Combinational Circuits

**Required**

- P. Chu, *FPGA Prototyping by VHDL Examples*
  
  Chapter 3, RT-level combinational circuit Sections 3.1, 3.2, 3.3, 3.6, 3.7.1, 3.7.3.

**Recommended**

- S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with VHDL Design*
  
  Chapter 6, Combinational-Circuit Building Blocks

  Chapter 5.5, Design of Arithmetic Circuits Using CAD Tools