The **Microelectronics Training Center**

The MTC is an initiative within the INVOMEC division

Industrialization & Training in Microelectronics

Lab-exercise

Lab 4:

VHDL basics: microprocessor specifications

Cluster: Cluster1
Module: Module1a

Target group: Students

Version: 1.1
Date: 21/03/06
Author: Osman Allam
Modified by: Geert Vanwijnsberghe
History : 30/11/06 : testbench added

This material was developed with support of the European Social Fund.
ESF: Prevent and combat unemployment by promoting employability, entrepreneurship, adaptability and equal opportunities between women and men, and by investment in people.
http://www.esf-agentschap.be

Introduction

In this module, you will map some of the microprocessor Micro6 specifications in VHDL. Basically, it is setting the framework of your design.

Objectives

After completing this module, the student should be able to:

- Identify the steps of designing a complex system
- Build VHDL packages.

Knowledge background

- Knowledge of the different design units in VHDL

Classification

- Level: 1
- Duration: 20 minutes

Input

VHDL template of the package `micro_pk`.

The lab

For large designs, it is good practice to build a package in which to declare all the constants, types, subtypes, functions and procedures you will need in multiple units of your design.

Packages are defined in 2 parts.

1. Package declaration: defines the visible contents of the package. For example: constants, types, subtypes & subprogram declarations.
2. Package body: provides the hidden details. For example: constant specifications (for deferred constants), subprograms & subprogram bodies.

Using the template of the package `micro_pk (file: micro_pk.vhd)`, do the following.

1. Declare constants for the data width and the address width of the microprocessor.
2. Declare a subtype for the opcode of the microprocessor. Note, all jump and branch instructions are represented internally by a single opcode (`JYY`). For all other instructions, the same opcodes as those shown in the previous module are used. In total, there are only 28 different opcodes. The individual opcodes of each instruction are represented by constants of the subtype you declare for the opcode.
3. Declare a type to define an array of data words. You may use the declaration of the data width (from step 1).
4. Declare the function `is_pos` taking a `signed` as its argument and returning a `boolean`. `is_pos` returns true if the argument is positive and false otherwise. The negative numbers are represented by 2's complement. The MSB is the leftmost bit.

5. Write the body of the function `is_pos` in the package body.
6. Compile the micro_pack.vhd in library work
7. Compile the testbench tb_micro_pack.vhd in library work
8. Run a simulation for 1 ns
   If you code is correct you will get only the messages
   -- Check start --
   -- Check done --