The **Microelectronics Training Center**

The MTC is an initiative within the INVOMEC division

Industrialization & Training in Microelectronics

Lab-exercise

Lab 4:

Design of the CPU

Cluster: Cluster1
Module: Module5a

Target group: Students

Version: 1.0
Date: 8/1/07
Author: Osman Allam
Modified by: Geert Vanwijnsberghe

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 combine the units you have developed in the previous modules to build the CPU.

Objectives

After completing this module, you should be able to:

- Describe structural models in VHDL

Knowledge background

- Basic VHDL knowledge
- Understanding of the CPU architecture of Micro6 (presented in cluster 0)

Classification

- Level: 2
- Duration: 30 minutes

Input

VHDL template

The lab

This module is an extension to module1e. Structural modeling is presented in greater detail. Reviewing the above module is strongly recommended. Out of the several binding techniques explained before, we are going to utilize only default component binding in this module.

Default component binding (review)

The prerequisites for default binding are:

- Components must be declared before they are used.
- The component interface (name and port list) must match that of the entity.
- The entity must have been compiled in a visible library.
- The last compiled architecture is associated with the instantiated entity.

Top-level

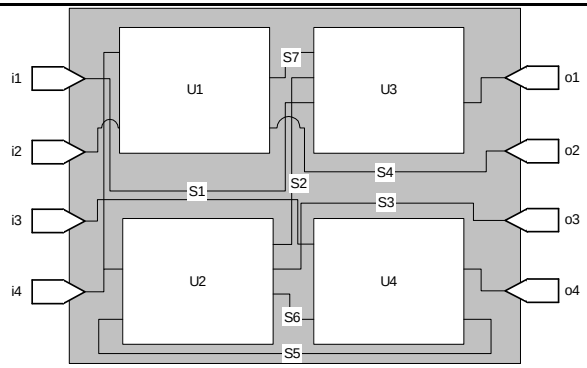

Figure 1: Tol level Model

The top-level of a structural model is composed of ports, components and signals connecting different components. Writing structural models in VHDL is rather straightforward.

```
entity top_level is
  port (
    -- declare ports: i1, i2,.. , o1, o2,..
end entity;
architecture struct of top_level is
  -- declare component: U1, U2, ..
  -- declare signals: S1, S2,..
begin
  -- component instantiations
end architecture;
```

It is advised to build a package for all components in your design. Each component is declared once and can be used in several structural entities. A change in a component declaration requires changing it in a single place.

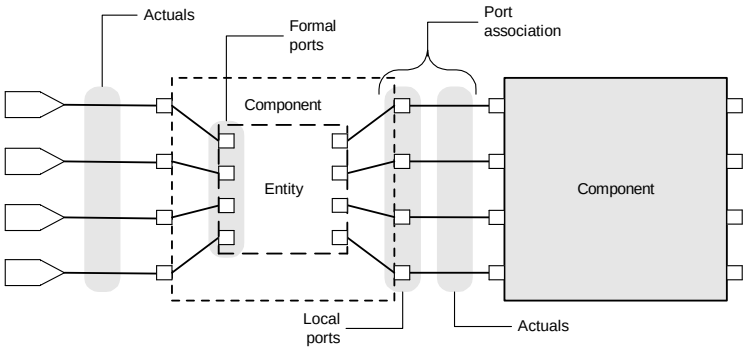

Figure 2

There are two types of ports in structural descriptions:

1. Formal ports: ports that belong to instantiated entities/components.
2. Local ports: porst that belong to components.

Note that in this case there is no need to associate local ports of a component and formal ports of its corresponding entity because both use the same port names. Remember that this is true only with default component instantiation. In both configuration specification and configuration declaration methods, you need to provide information about how local ports are associated with formal ports. This module doesn't cover these techniques.

However, default binding doesn't provide the HDL simulator with any information about local ports' association with *actual* signals in the design. You have to provide this port association information explicitly in your code. Port association can be described in VHDL in one of two ways: positional or named.

1. Positional association
   In the port map clause, actual signals are arranged in the same order as **local ports** in the component declaration.

> ⓘ For an entity-component pair, the order of formal ports doesn't necessarily have to match the order of local ports.

Example:

```
Entity reg4 is
  Port (
    rst : in std_logic;
    clk : in std_logic;
    d : in std_logic_vector (3 downto 0);
    q : out std_logic_vector (3 downto 0));
end entity;
..
component reg4 is -- note: local names are not in the same order
  -- as formal ports!
  port (
    clk : in std_logic;
    q : out std_logic_vector (3 downto 0);
    rst : in std_logic;
    d : in std_logic_vector (3 downto 0));
end component;
..
-- in the top-level's architecture declarative part
  signal clk1, rst1 : std_logic;
  signal d1, q1 : std_logic_vector (3 downto 0);
-- in the top-level's architecture statement part
u1: reg4 port map (clk1, q1, rst1, d1);
```

2. Named association (recommended):
   In the port map clause, actual signals are associated with explicitly *named* local ports.
   The above example can be re-written as:

```
-- all the same except the top-level architecture statement part
u1: reg4 port map (
  clk => clk1,
  d => d1,
  q => q1,
  rst => rst1);
```

Note that the order of individual port associations doesn't matter.

Shared buses

Shared buses, in Micro6, are implemented as multiplexers. All the units that can transmit data on the bus are connected to the multiplexer input port. The multiplexer output port is connected to all the units receiving data from the bus. The control unit manages the shared buses. This means that the multiplexer's selection lines are connected to the control unit.

Example:

```
signal shared_bus : std_logic_vector (7 downto 0);
..
shared_bus1: mux port map ( -- Multiplexer
  in1 => unit1_out; -- output port of unit1
  in2 => unit2_out; -- output port of unit2
  ..
  output => shared_bus);
..
unit3: <component_name> port map (
  ..
  in_port => shared_bus;
  ..);
unit4: <component_name> port map (
  ..
  in_port => shared_bus;
  ..);
```

Where unit1, unit2,... are sharing the `shared_bus` bus and unit3,unit4 are using this shared_bus as input.

Exercise: CPU architecture

Micro6 CPU is composed of 11 units and 5 shared buses as shown in the diagram below.

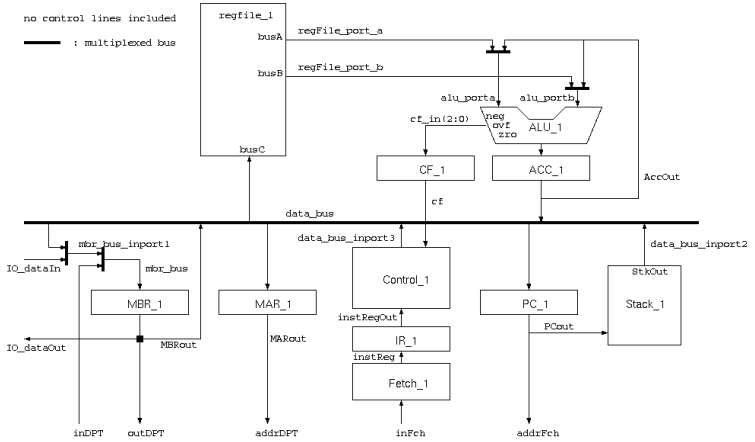

Figure 3: CPU architecture

Complete the VHDL description of the CPU architecture provided in the template in the file `cpu_vhd`. All the control lines are already included only the data signals have to be added. No testbench is given since it is easier to verify the complete system at a higher level. In this module of the lab you only have to complete the code for the cpu and compile it.