The **Microelectronics Training Center**

The MTC is an initiative within the INVOMEC division

Industrialization & Training in Microelectronics

Lab-exercise

Lab 4:

VHDL basics: registers

Cluster: Cluster1
Module: Module1b

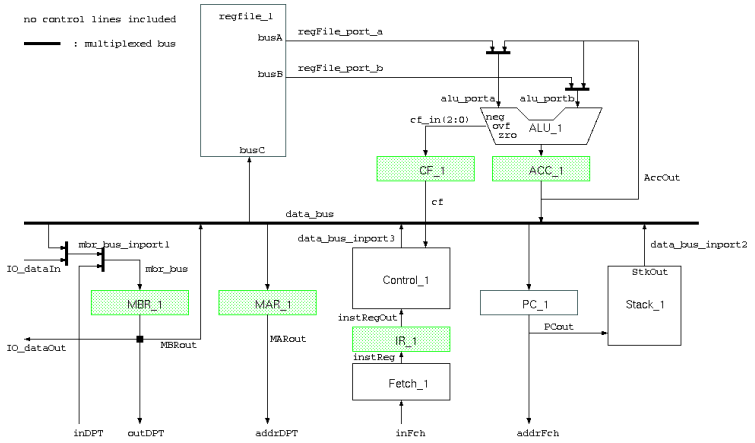
Target group: Students

Version: 1.0
Date: 21/03/06
Author: Osman Allam
Modified by: Geert Vanwijnsberghe
History : 1/12/06 : 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

The D-flip-flop is the elementary building block of all sequential circuits. Aggregations of D-flip-flops build registers. You will need several registers in your design to hold intermediate data. However, more registers will be inferred when synthesizing other sequential circuits which you will write in a higher level of abstraction.



Objectives

After completing this module, you will be able to:

- Design elementary sequential circuits

Knowledge background

- Basic VHDL knowledge

Classification

- Level: 1
- Duration: 15 minutes

Input

VHDL template

The lab

Sequential circuits can be modeled in VHDL as processes with sensitivity lists or *wait* expressions. Since many synthesis tools do not support *wait* expressions, we limit our discussion on processes with sensitivity lists.

The behavior and the synthesized circuit depend greatly on the signals that are in the sensitivity list. Careful selection of these signals is necessary. The behavior depends also on the order of sequential assignments within the process.

In your design you will need registers with one control input, the enable port. Resetting the registers with dedicated reset input is not necessary because some registers are reset by loading them with zeros during the reset sequence of the microprocessor after it is powered on. Other registers don't need to be reset because their contents are not used before they have been loaded with valid data.

There are two ways for writing a VHDL model for registers that can be re-used in different sizes:

1. Using generics: the register size is determined by the value of the generic which can be assigned at instantiation. You can give generics default values.

```
Entity register_en is
  Generic (size : positive := 8); – default size is 8
  Port (
    rst : in std_logic;
    clk : in std_logic;
    en  : in std_logic;
    d   : in std_logic_vector (size – 1 downto 0);
    q   : out std_logic_Vector (size – 1 downto 0));
End entity;
```

2. Using unconstrained ports: the register size is automatically determined at instantiation by the width of the input and output signals connected to it.

```
Entity register_en is
  Port (
    rst : in std_logic;
    clk : in std_logic;
    en  : in std_logic;
    d   : in std_logic_vector;  -- unconstrained vector
    q   : out std_logic_Vector);-- unconstrained vector
End entity;
```

Design a register with synchronous enable and asynchronous reset signal. The width of the register is unconstrained.
Use the entity declaration given in the file `register_en.vhd`

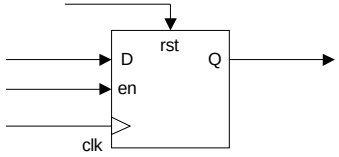


Figure 1: register

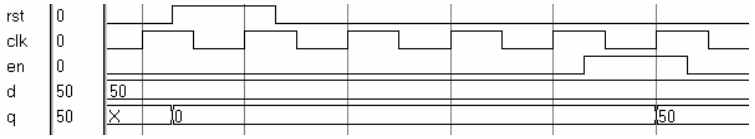


Figure 2: register wave form

A testbench file tb_register_en.vhd is given to allow you to verify your register_en entity/architecture.
Compile and simulate your code.