The **Microelectronics Training Center**

*The MTC is an initiative within the INVOMEC division*

Industrialization & Training in Microelectronics

Lab-exercise

# *Lab 4:* Design of the fetch unit

Cluster: Cluster1
Module: Module4c

Target group: Students

Version: 1.0
Date: 03/01/07
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

The Fetch Unit reads instructions from the memory and passes them to the control unit for decoding and executing.

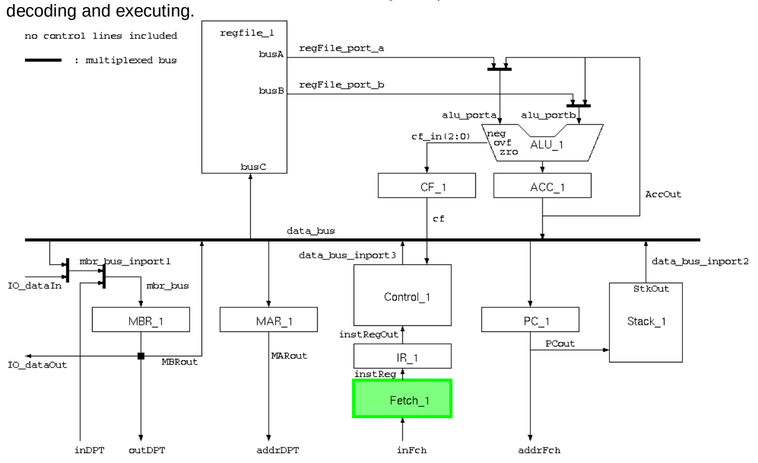

**Figure 1: CPU architecture**

## Objectives

After completing this module, you should be able to:

- Translate FSM state transition tables into VHDL descriptions.

## Knowledge background

- Basic VHDL knowledge
- Understanding Moore FSMs

## Classification

Level: 2
Duration: 2 hours

## Input

- fetch.vhd :VHDL template
- fetch_tb.vhd : testbench for fetch
- compile.do : modelsim script to compile all needed entities and packages
- wave.do : modelsim script to view the most important signals

## The lab

The fetch unit is the first of two stages of the instruction pipeline inside Micro6. The second stage consists of a separate IR register that is located between the fetch unit and the control unit (see figure 1).The pipeline makes 2 instructions ready for decoding and executing. This means that, in best case, while the decode and execute units are busy with instruction *n*, the fetch unit is ready to read instruction *(n + 2)* from memory. There are 2 reasons why the pipeline is so short:

1. Executing branch and jump instructions becomes less efficient as the length of the pipeline increases.
2. Memory access takes roughly the same number of clock cycles needed for executing instructions. Hence it is sufficient to read only one instruction in advance.

The fetch unit is composed of a finite state machine and a register to hold one instruction. The basic functionallity is as follows.

- wait until the control unit gives a signal to read the next instruction (readInstr)
- transfer the instruction from the instruction register inside the fetch unit to the IR register (figure 1 & 2)
- read the next instruction from memory by asserting memRd and waiting until memReady becomes true.
- increment the program counter while storing memData in the first pipeline stage (instruction register inside fetch)

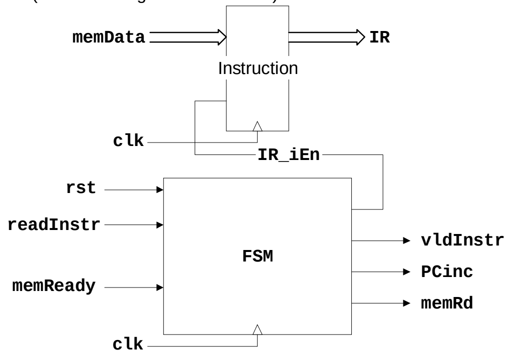

**Figure 2: Fetch unit**

The vldInstr is always asserted except when the fetch unit is busy reading from the memory.

The next 2 state diagrams show how the finite state machine inside the control unit interacts with the finite state machine inside the fetch unit.

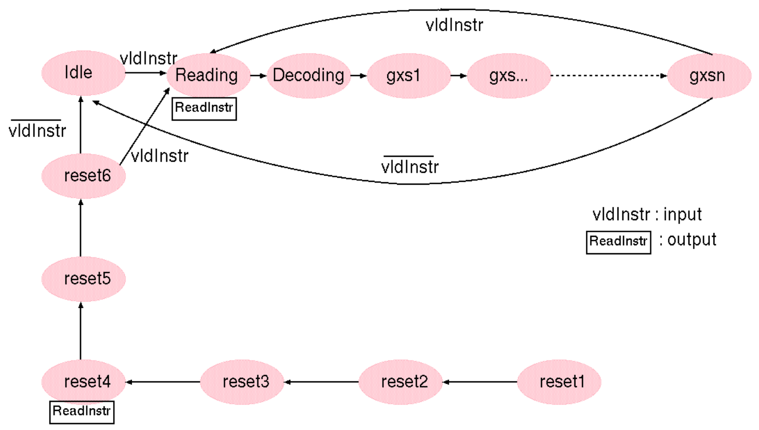

**Figure 3: fsm inside control**

In the Reading state a ReadInstr pulse is passed to the fetch unit in order to load the next instruction. At the same time the available instruction in the fetch unit is passed to the IR register (figure 1).

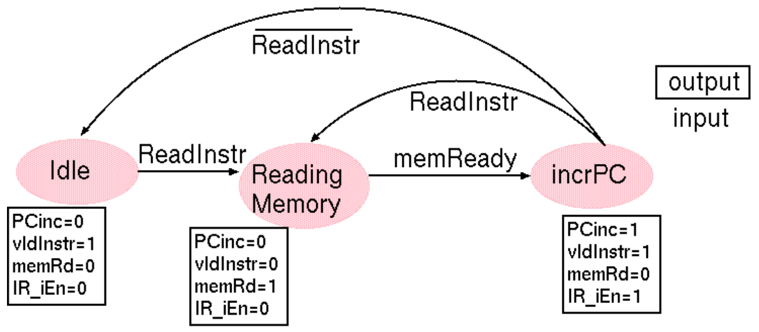

**Figure 4 : fsm inside fetch**

The state machine of the fetch unit is very simple. It waits in state Idle till it receives a ReadInstr signal. Then it waits in state ReadingMemory till the instrcution is received from memory while the vldInstr signal is kept low.

Both fsm's will work correctly together when the control fsm never creates a ReadInstr signal while vldInstr is low.

Note : eg. executing a jump instruction by the control unit

1. The program counter is loaded with the jump address
2. When vldInstr is asserted a ReadInstr signal is given to the fetch unit that will load the new instruction into its instruction register. When this is done the vldInstr signal is asserted and the control unit will go to its reading state. Then a new ReadInstr signal will

be given and at the same time the fetched instruction will be passed to the IR register (figure1).

## Exercise

Implement the state machine of the fetch unit as a Moore machine. Use the template provided in the file `fetch.vhd`.
Use the testbench tb_fetch.vhd to verify the fetch unit and the interface between control and fetch unit. Use the modelsim scripts compile.do and wave.do.

```
VSIM> do compile.do
VSIM> vsim work.tb_fetch
VSIM> do wave.do
VSIM> run –all
```

Some extra debug code has been added to the control.vhd in order to view the mnemonics of the instructions executed by the control unit (opcode_debug).

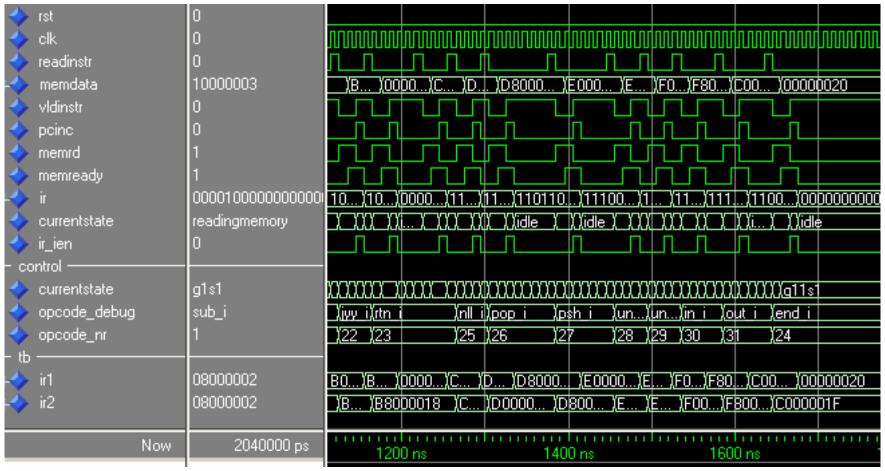

**Figure 5 : sim result**

You should have a waveform as shown above.