# VHDL DESIGN OF EDUCATIONAL, MODERN AND OPEN-ARCHITECTURE CPU

**Martin Straka**
Doctoral Degree Programme (1), FIT BUT
E-mail: strakam@fit.vutbr.cz

Supervised by: Zdeněk Kotásek
E-mail: kotasek@fit.vutbr.cz

## ABSTRACT

The paper deals with design of a modern, open-architecture CPU utilizable for educational purposes. It is expected that use of the CPU in the educational process will greatly contribute to deeper understanding of key-topics taught in the area of modern architectures. Our CPU is based on the Von-Neumann architecture, equipped with a five-stage pipeline, cache memory unit and simple branch prediction unit. The architecture is designed in VHDL including set of 16 instructions. Rich variety of educative tasks can be performed by means of the CPU. It has been both successfully simulated in ModelSim and synthesized in Precision RTL Synthesis in order to be implemented in FPGA and utilized in practice as a real working CPU.

## 1. INTRODUCTION

It may seem to be nearly unbelievable from today's perspective but the computer market wasn't flowing with dozens of general-purpose CPUs ever since. Systematic development enabled the advent of microcontrollers and control units in a form of widespread components. However, these were still dedicated to a given task and the inherently rigid interface, in fact, put significant constraints on other application scenarios. This situation was gradually changing with the introduction of modern CPUs and programmable circuits like FPGA, which have taken a dominant position in computation systems. The universal nature of such electronic components offers a convenient platform for mostly any kind of possible tasks.

The core of our interest is focused on more advanced processor architectures, especially the CPU with features including 5-stage pipeline, fast instruction and data cache, and simple branch prediction unit. The resulting design of our CPU reflects Von Neumann conception [3] and employs RISC instruction set with 16 basic instructions. Built-in register file is optimized for efficient data and instructions manipulation at corresponding locations in both cache and RAM memory. Execution of instructions takes place in a 5-stage pipeline with parallel overlay of its slices.

The overall design has been implemented in VHDL language [1, 2], which represents very convenient way of digital systems design. The implementation and description in VHDL

was evaluated through ModelSim environment. Final step included synthesis with Precision RTL Synthesis for target FPGA circuit. Achieved results can be found in [4].

## 2. DESIGN OF CPU

The initial phase of our effort to develop a processor with modern features was closely tied with simple and, in the same time, sequential instruction processing method. This architecture has been completely reworked and modified towards scalar CPU. Our processor is 16bit device with both data and address bus at width of 16bit. Length of every instruction word is again 16bit.

### 2.1. 5-STAGE PIPELINE

The conception of pipeline, or chained processing, can be described as the implementation technique which partitions execution of a given operation into number of subsequent steps, as far as possible of the same duration. Furthermore, each section assigned to a particular step can exploit standalone technical resources. Execution of single instruction requires 5 clock cycles but this overhead can be effectively hidden by parallel operation. Instruction processing may go through the following stages:

- IF – instruction fetch
- ID – instruction decode
- EX – execution
- MA – memory access
- WB – write back

Overlay processing of several consecutive instructions shows Table 1:

| | | | | CPU clock | | | | |
|---|---|---|---|---|---|---|---|---|
| instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| i | IF | ID | EX | MA | WB | | | |
| i+1 | | IF | ID | EX | MA | WB | | |
| i+2 | | | IF | ID | EX | MA | WB | |

**Table 1:** Instruction processing flow.

Synchronous pipeline is composed of 5 stages where each of them is dedicated to different stage of instruction processing. Individual stages are separated by embedding additional registers. Interconnection scheme for 5-stage pipeline and respective building blocks are shown on Figure 1.

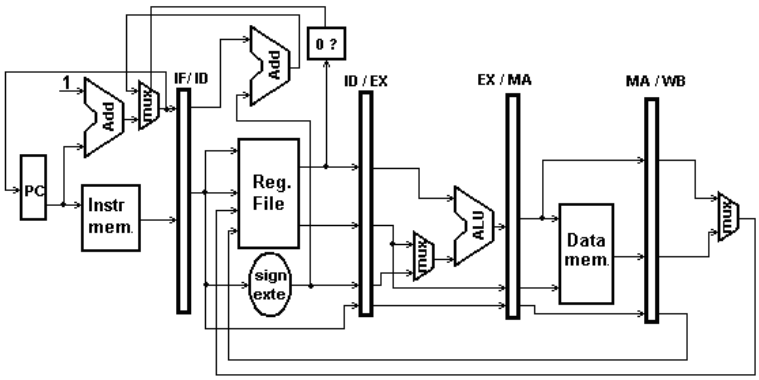


**Figure 1:** Structure of implementation for 5-stage pipeline.

Pipelined or chained instruction processing may impose three types of conflicts which have been addressed in our design by appropriate circuitry structures. Summary of these conflicts and recommended solutions are shown in Table 2:

| Conflict type | Unit eliminating the conflict |
|---|---|
| Data(RAW,WAR,WAW) | Stalling & forwarding unit |
| Control | Branch & prediction unit |
| Structural | Stalling unit |

**Table 2:** Hazards in pipeline.

Every functional block of CPU is characterized by behavioural description. Then, mutual relationship between these elements is set up on structural level, where the physical interconnections are defined. General block scheme of our CPU, together with its interface, is depicted on Figure 2:

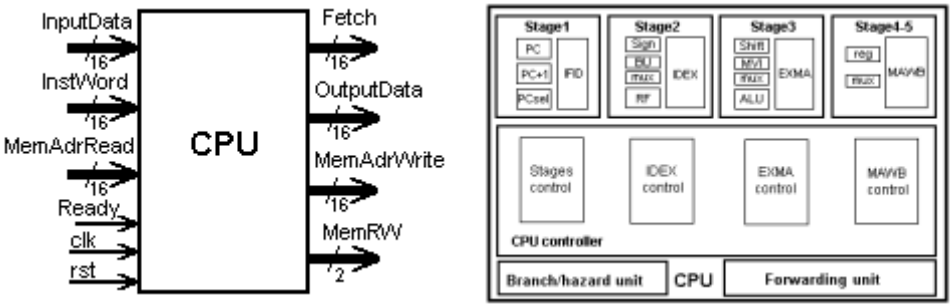


**Figure 2:** Entity of CPU & CPU block

## 2.2. INSTRUCTIONS AND DATA CACHE

The other part of implementation is dealing with L2 cache memory. The proposal of data and instruction cache is based on 2-way associative architecture. These two blocks of memory work in an asynchronous way and communication with CPU by means of ac-

knowledgement signals. Implemented cache allows for both methods of data update with LRU algorithm – write-back and write-through. Their interface is outlined on Figure 3:

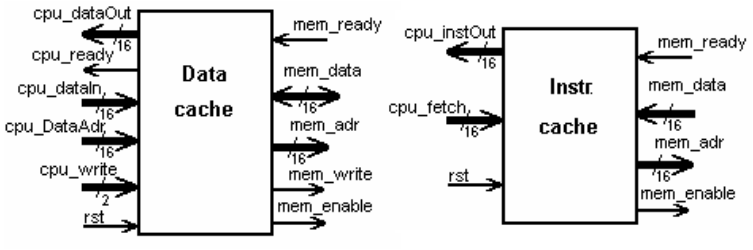


**Figure 3:** Entity of Data and Instruction cache

### 2.3. UNIT FOR PREDICTION BRANCH

Characteristic feature of modern processors is, among many others, the built-in branch prediction unit. This functional block keeps a track of jump instructions that have been already used. Such information are used to estimate in advance, already during the decode phase, whether the current jump under evaluation will really happen together with target address determination. Thus, our CPU will use this prediction to choose next instruction according to its probability occurrence ratio. Interface of 1-bit predictor, which was implemented, is shown on Figure 4.

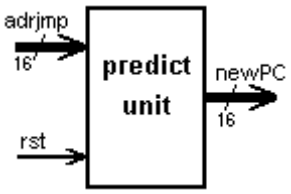


**Figure 4:** Entity of Prediction unit

## 3. RESULTS

It can be easily concluded that characteristics of both processors (sequential and pipelined) are significantly different in all relevant parameters. From the architectural point of view, implementation of pipeline is much more complex than simple sequential processor. Let's compare execution speed with the following code snippet, where z=100, x=2 and y=2 are values stored in RAM (see Table 3).

```
FOR (i = 1, i <= z, i++){
      x = x + y;
      y = x - y;}
```

```
        LOD   R1, adr(1)
        LOD   R2, adr(2)
        LOD   R3, adr(3)
loop :  ADD   R2, R2,R3
        SUB   R3, R2, R3
        DEC   R1, R1
        BNZ   R1, loop
        STO   R3, adr(4)
```

**Table 3:** Instruction code for CPU

Simulation results are recapitulated in Table 4:

| CPU Functions unit | Pipeline with cache | CPU [cycle] without cache | Sequential CPU [cycle] without cache |
|---|---|---|---|
| Sequential | 2020 | 2040 | **2040** |
| pipeline | 1114 | 1134 | - |
| prediction | 1017 | 1037 | - |
| forwarding | 913 | 933 | - |
| **Prediction+forwarding** | **828** | 848 | - |
| 50% successful in cache | 838 | - | - |

**Table 4:** Results of sequential CPU and pipelined procesor with cache

## 4. CONCLUSION

Future directions of our work will be connected with the implementation of more sophisticated branch prediction and dynamic instruction scheduling. Last but not least, the eventuality of pipeline modification towards higher concurrency by means of additional functional units or even the whole pipeline replication is going to be thoroughly evaluated. However, the introduction of higher scale parallelism brings new issues on the stage, which are related to improved pipeline control and conflicts elimination. With extension of instruction set, floating point unit merged with the rest of CPU and some other changes our CPU could be probably refined to superscalar processor.

## REFERENCES


[1] Cohen, B. VHDL Coding Styles and Methodologies, Kluwer, Dordrecht, 453 p., 2001, ISBN 0-7923-8474-1.

[2] Douša, J. VHDL Language, Czech Technical University Publishing House, Praha, 76 p., 2003, ISBN 80-01-02670-1.

[3] Hennessy, J.L., Patterson, D.A. Computer Architecture – A Quantitative Approach, Third Edition, Morgan Kaufmann Publishers, New York, 1000 p., 2003, ISBN 1-55860-596-7.

[4] Straka, M. CPU Design in VHDL, Brno University of Technology, Brno, Diploma work, 90 p., 2006.

[5] Dvořák, V., Drábek , V. Architektura procesorů, VUTIUM, Brno, 293 p., 1999, ISBN 80-214-1458-8