Design of Advanced Computer Architectures, Based on PIM – Processors in Memory*

T. Tashev, S. Tashev, N. Tasheva

Key Words: *Processor in memory; cache coherent protocols; multiprocessing; performance evaluation*

Abstract. *The present article describes the design and simulation of computer architecture based on PIM (Processor-In-Memory). PIM models are analyzed according to the developed mathematical formulations used for simulation of the proposed architecture as a possibility to solve the bottleneck problem between the main processor and memory, providing high bandwidth and computational resources in the operating memory. The proposed architecture was analyzed for the performance measuring and time markers researches with Processor-In-Memory (PIM) computing.*

1. Introduction

It the first decade of the new millennium single processors reached big frequencies, pushing the hardware technologies to physical limits. These trends reached the end, because of the physical and architectural bounds narrowing the computing power of the classical single processor architecture. Alternative decisions to avoid the above mentioned limits are described in the present article, and a new model of computer architecture with computational resources in memory was designed and depicted. With its help some of the von Neumann limits are overcome, as well as the architectural way for computer performance was improved.

In the current article a computer architecture for single and multi processors configurations is proposed, with processor element insertion in the operating memory PIM. Distributed are the computational operations in the machine between the main processor and PIM, and alterations have been made in the main diagram of instruction execution. The analytical model described below was used for creation of the simulation model and design of the main elements from the PIM unit.

From the history analysis and the current status of the computer architectures it appears that the most critical place referring to the speed of the computational machine is the communication network, connecting the processor and the memory. This problem is also known as *von Neumann bottleneck* which rises permanently because of the big performance difference between the processors and the memory.

For the design of a single or multi processor system there are possible ways to find cheaper technical solutions to avoid the current limitations in the conventional systems. Such a hypothesis is the physical integration of a PIM module for additional computational resources in the operating memory and reduction of the information stream between the processor and the memory.

There are four basic steps, which almost all central processor units, based on von Neumann architecture use during the operations:

– *fetch* – IF
– *decode* – ID
– *execute* – EX
– *writeback* – WB

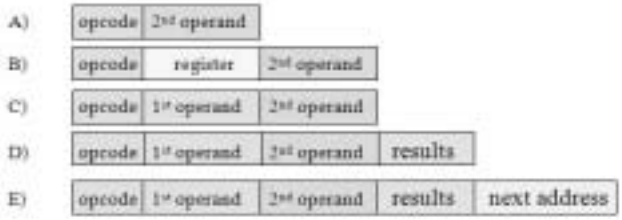

Figure 1.1. Instruction format

Figure 1.1. depicts the format of the instruction in different addressing methods.

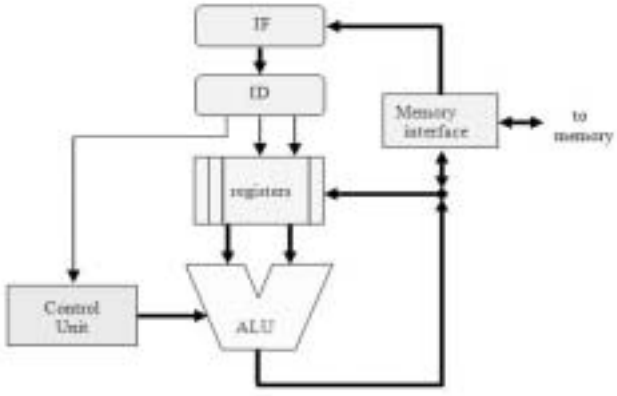

Figure 1.2. CPU block diagram

Figure 1.2. depicts the classical block diagram of a CPU.

The main trend for the high performance computer architectures during the last decade is the migration of the memory as a form of a high speed cache in the microprocessor as well as multiplying the cores inside of the CPU.

2. System Description of PIM

The system model based on PIM architecture has a main superscalar processor (MP) and a PIM node, depicted on *figure 2.1.* MP as a standard processor contains memory of two levels L1 and L2. The communication network (CN) is critical

**This work is supported and performed in the frame of Project No. DO 02-115 of the Bulgarian National Science Fund (NSF).*

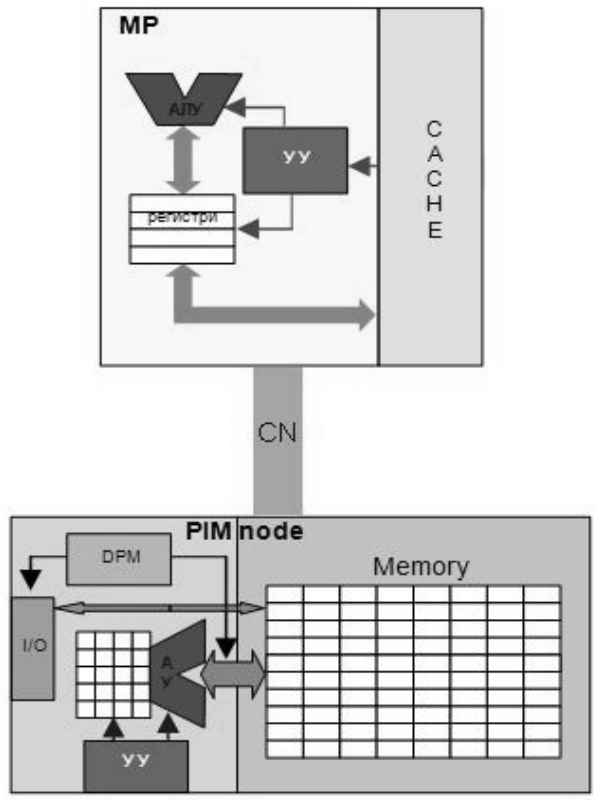

Figure 2.1. Block diagram of PIM architecture with 1 PIM node

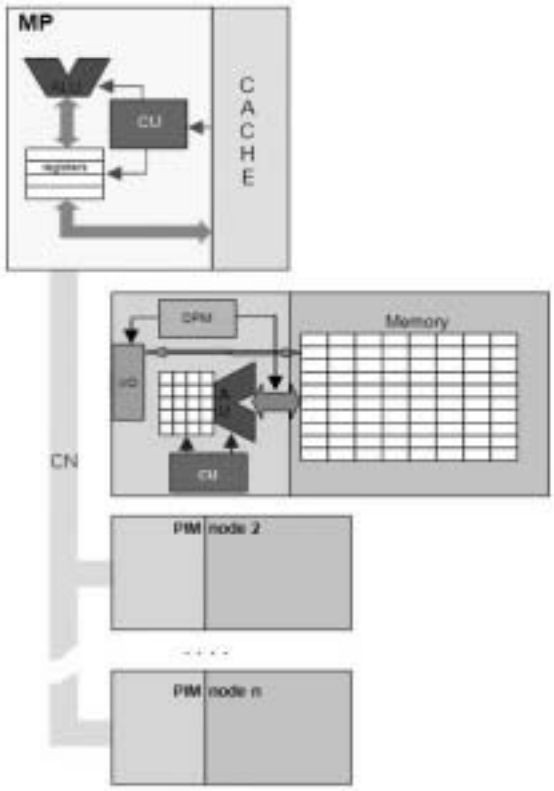

Figure 2.2. Block diagram of n-PIMs architecture

with respect to the speed, also known as von Neumann bottleneck. To reduce the frequent MP – memory access a PIM node is used. Part of the computations are done on memory level (PIM node), but the main computations are on level MP.

Although the view of the processors looks the same, both processors MP and PIM have considerable differences described below.

To be able to increase the performance of the system model, we are using the technique to multiply the memory nodes. Studying the research done by the University of Notre Dame, which proves that the performance of PIM is increasing with a segmentation model, adding index bits directly to the row buffers of the DRAM, it is possible to use the PIM architecture depicted on *figure 2.2*.

The step sequence on memory level is accomplished by a PIM node, who does the following:

– superscalar instruction pre decoding;
– administration and operand fetch;
– policy control (write-back and write-through);
– replacement algorithms (LRU, LFU and FIFO);
– data dependencies.

On a level MP we have the following functions:

– instruction address calculation;
– data processing;
– policy control (write-back and write-through);
– register renaming;
– branch predictions;
– rearrangements.

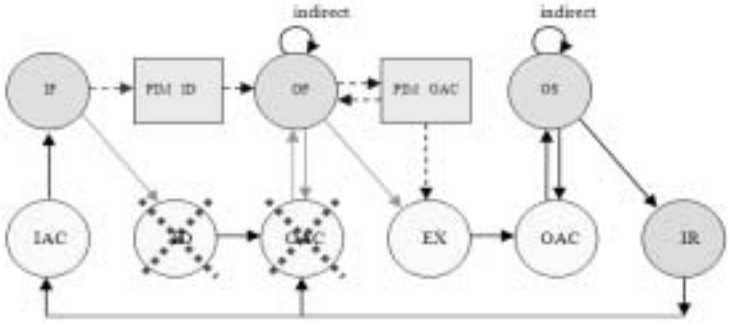

Figure 2.3. Instruction cycle diagram

Figure 2.3. depicts the instruction cycle diagram for PIM architecture.

3. Analytical Model of PIM

This analytical model is analyzed through simulations and the results were used for the hardware implementation of the proposed architecture.

$$T = 1 - \%Whit \times \left\{ 1 - \frac{1}{N} \times \left[\frac{\tau_{pim} + LS \times (T_{dpim} - \tau_{pim})}{1 + LS \times (T_{Cmp} - 1 + P \times T_{Mop})} \right] \right\}$$

if we assume $$M = \left[\frac{\tau_{pim} + LS \times (T_{dpim} - \tau_{pim})}{1 + LS \times (T_{Cmp} - 1 + P_{MP} \times T_{Mop})} \right]$$

for T we receive $$T = 1 - \%Whit \times \left\{ 1 - \frac{M}{N} \right\}$$

where:

% W_{PIM} = work performed by PIM;

τ_{PIM} = PIM tact;

T_{Mpim} = PIM memory access;

LS = average time for Load/Store;

T_{Cmp} = MP time for local cache access;

P = misses in cache;

N = number of PIM nodes;

Parameters $\%W_{PIM}$, M and N are independent.

The conveyer operations of course will influence the main performance of the PIM node. *Figure 3.1.* depicts time diagram of PIM conveyer performance.

The speed up factor for the instruction conveyer in PIM architecture compared to architecture without conveyer performance is defined as:

$$S_k = T_1 / T_k = \frac{n k \tau}{[k + (n - 1)]\tau} = \frac{n k}{k + (n - 1)}$$

Figure 3.2. depicts the speed up factor as a function of the number of instructions executed without branch and *figure 3.3.* depicts the speed up factor as a function of number of stages.

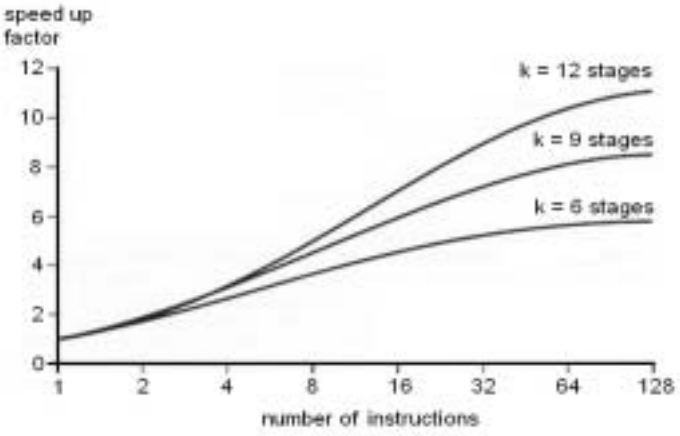

Figure 3.2. Speed up factor f (n-instructions)

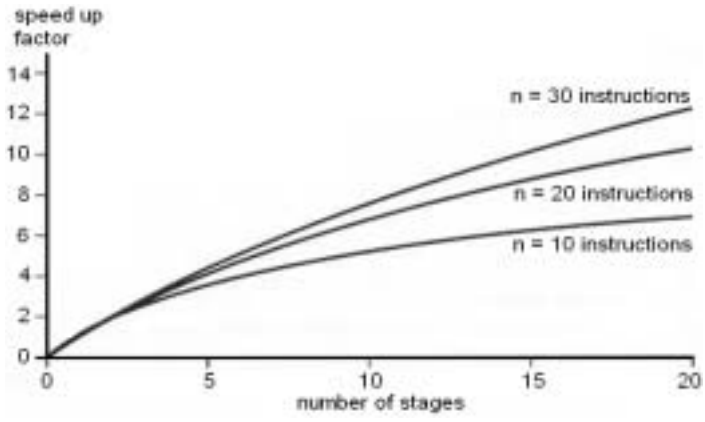

Figure 3.3. Speed up factor f(n-stages)

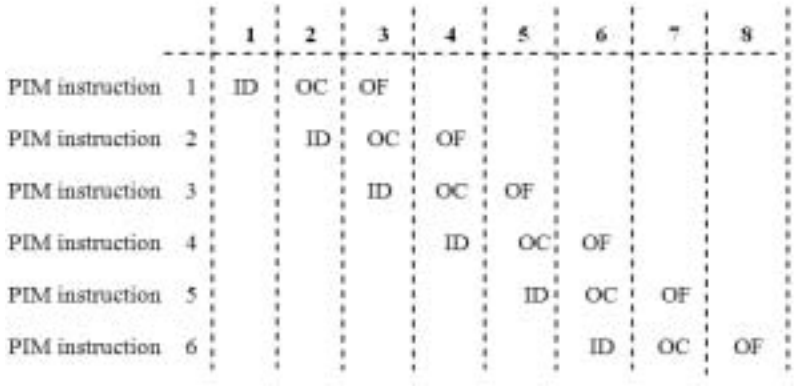

Figure 3.1. Time diagram of PIM conveyer performance

4. Simulation Results

To simulate the above proposed structural model we used the simulation technique SUNSiman. To get the performance of the above proposed PIM architecture we compared its results to a superscalar architecture PowerPC620. The simulation is based on a queue modelling and uses probability distributions. The conditions for the simulation of both architectures are as follows:

– instruction window: 8 instructions;
– 30% – workloads;
– 40% – branch instructions;
– 5% – floating point instructions and
– 25% – Load / Store instructions;
– simulation duration – 2000 tacts;
– superscalar processor stages – 4;
– conveyer depth FPU(6), LSU(5), BU(2), IU(4).

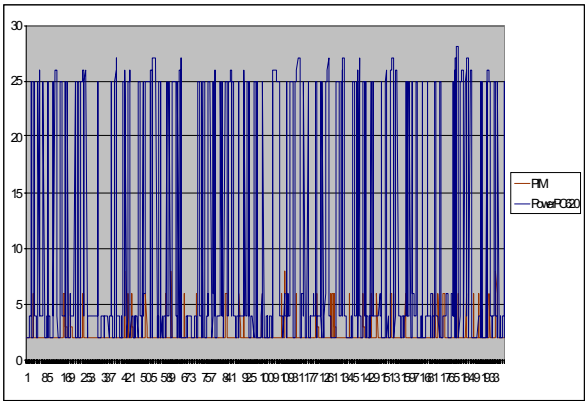

Figure 4.1. PIM vs. PowerPC620

A graphical presentation of the results could be seen on *figure 4.1.* The number of the conducted iterations was 2000, and from the graphics it is clear that the performance of PowerPC620 architecture is lower then the performance reached with architecture with built in PIM module.

Figure 4.2. show the conveyer work load results for both architectures. The data for the superscalar processor PowerPC620 in standard von Neumann architecture are basic and are used for comparison and analysis of the received results for the PIM architecture.

The researched conveyers for both models are Branch Unit, Instruction Unit, Floating Point Unit и Load/Store Unit.

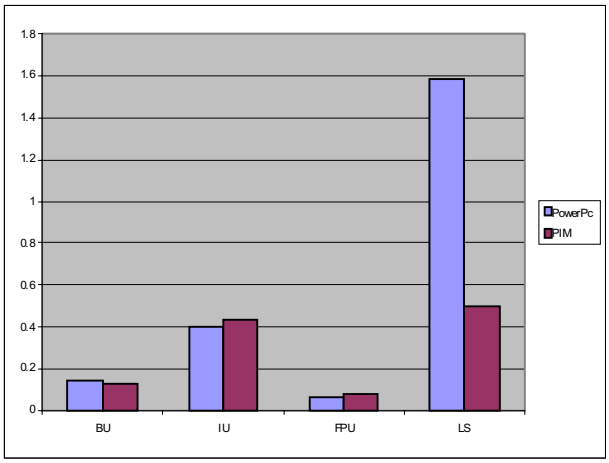

Figure 4.2. Conveyer work load in PowerPC620 and PIM

5. Future Work

The future work includes hardware implementation of the proposed architecture using advanced FPGA technology. The main goal is to design the different PIM elements and to compare the received simulation results with the real performance.

Another direction of the research is to analyze multiprocessor configurations and the cache coherence problems.

6. References

1. Tashev, T. Computer Architecture with Computing Resources in the Operating Memory. National Conference: European Integration, Innovation and Information Society, Sofia, May 2007.

2. Tashev, T., J. Zidarova. Distribution Calculations between the Main PEs and PIMs. International Simposium Hissar Info Days BIS 21++, Hissar, 26 - 28 March 2007, 157-167.

3. Tashev, T. PIM (Processor-In-Memory) in Parallel Architectures, Based on FPGA. Open IPP PhD Workshop, Sofia, 5 Oct 2005, 25-28.

4. http://www.cse.nd.edu/~pim

5. http://iram.cs.berkeley.edu

6. Kang, Jung-Yup et al. An Efficient PIM (Processor-In-Memory) Architecture for Motion Estimation. In proceedings of the 14th IEEE International Conference on Application-Specific Systems, Architectures, and Processors, 2003,.282-292.

7. http://www.ai.mit.edu/projects/aries/

8. Gilgamesh: A Multithreaded Processor-In-Memory Architecture for Peta°ops Computing, NASA Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, U.S.A., 2002.

Maniscript received on 19.10.2009

Svetoslav Tashev *(born 1977) received the MA degree in Computer Science from Technical University – Sofia, Bulgaria in 2001. PhD student in Computer Systems, Complexes and Networking in Bulgarian Academy of Sciences. He has been teaching, consulting and working on projects, in the field of his thesis.*

Contacts:
Bulgarian Academy of SciencesInstitute for Information and Communication Technologies
e-mail: svetlio@bas.bg

Todor Tashev *(born 1975) received the MA degree in Automatics from Technical University – Sofia, Bulgaria in 1998 and MA degree in Computer Science from Technical University – Sofia, Bulgaria in 1999. PhD in Computer Systems, Complexes and Networking. He has experience in the field of Computer Architecture. He has been teaching, consulting and working on projects, in relation to e-learning and Grid computing.*

Contacts:
Bulgarian Academy of SciencesInstitute for Information and Communication Technologies
e-mail: teo@bas.bg

Nina Tasheva *(born 1955) received the MA degree in Technical University – Sofia, Bulgaria in 1981. Since 1997 she is M. Sc. I degree in IPP – Bulgarian Academy of Sciences. She has been teaching, consulting and working on projects in the field of Computer Organization, Computer Architecture and e-learning.*

Contacts:
Bulgarian Academy of SciencesInstitute for Information and Communication Technologies
e-mail: tasheva@bas.bg