International Journal of Computer and Electrical Engineering, Vol. 2, No. 5, October, 2010
1793-8163

Performance Evaluation of Nonlinear Pipeline through UML

Dr. Vipin Saxena and Manish Shrivastava

Abstract—**Complex numerical computations often use repeated arithmetic operations. Pipelining is an architectural approach for speeding up the floating-point arithmetic operations. In pipelining, a task is partitioned into simpler computational steps that can be executed concurrently. In high performance computing, the multi-core technology has emerged as one of the major technologies that support such pipeline architecture. In this paper, the performance of Intel Dual Core processor has been evaluated for floating-point arithmetic operations. A common software program computing floating-point arithmetic has been developed in Visual C++ and Visual C# languages and the performance of these programs is measured on Intel Dual Core processor. To demonstrate the performance, a well known Unified Modeling Language (UML) is used for modeling of floating-point nonlinear pipeline. The UML class, activity and sequence diagrams are designed and results are presented in the form of tables and graphs.**

Index Terms— **Nonlinear Pipeline, Dual Core Processor, Floating-point, UML class diagram, UML sequence diagram**

I. Introduction

Pipelining is one of the techniques for improving processing performance of a processor. This architectural approach allows the concurrent execution of several instructions. For achieving this, a task is subdivided into a sequence of subtasks, each of which can be executed by a specialized hardware stage that operates concurrently with other stages in the pipeline. Many types of pipelines are used in a computing environment namely: Arithmetic Pipeline; Instruction Pipeline; Memory Access Pipeline; and Communication Pipeline. Arithmetic pipeline may be nonlinear. In nonlinear pipeline, instead of a steady progression through a fixed sequence of stages, a task may use more than one stage at a time, or may return to the same stage at several points in processing. This type of pipeline is extensively used to speed up processing of floating-point arithmetic computations. The architectural details of pipeline structure are best described in [1], [2], [3] and [4].

Modern Computer systems support Dual Core CPUs. Dual Core is an architecture that refers to a Central Processing Unit with two complete execution cores in a single processor. The two cores, their caches and cache controllers are all built in together in a single IC. They provide parallelism between instructions and operations. The Intel's Core-architecture can be found in [5], [6], [7] and [8].

The Unified Modeling Language (UML) is a very famous visual modeling language which is extensively used for software designs. It has been increasingly applied in computer architecture modeling also. The standards and recent developments of UML are available on [9]. The excellent descriptions of UML diagrams and notations are available in [10] and [11]. Gomma [12] has developed a UML based concurrent object modeling and architectural design method for designing real-time and distributed applications. Recently Saxena et al. [13] proposed the UML model for the Multiplex system for the processes which are executing in distributed environment. In another paper by Saxena and Raj [14], UML modeling has been used for the instruction pipeline design and computed the performance of design. Pustina Lukas et al. [15] presented a UML based modeling methodology of specifying processor details of ARM. In this paper, UML diagrams are used to model the system architecture and timing behavior.

There is not much work available on performance of floating-point arithmetic operations. Shirazi et al. [16] have done an examination of various floating-point arithmetic operations on FPGA based custom computing machine. Sosnowski and Bech [17] performed extensive testing of FPU units using some test programs. They developed deterministic and pseudorandom (PSR) test procedures for Intel Pentium FPU. Kaivola, R. and Narasimhan, N. [18] presented a formal verification case study of floating-point multiplier in the Intel IA-32 Pentium4 microprocessor. Boldo and Filliatre [19] presented a formal verification framework for floating-point programs. This paper introduced a methodology to perform formal verification of floating-point C programs. It extends an existing tool for the verification of C programs. Lopez [20] et al. used a test suite IeeeCC754, to evaluate IEEE754 floating-point conformity across a wide range of heterogeneous computers with different architectures, operating systems, precisions, and compilers. They performed their study on two processors i.e., AMD and Intel processors. Recently UML Modeling and performance evaluation of Static Arithmetic Pipeline Design is done by Saxena and Shrivastava [21].

In the present paper, nonlinear pipeline for floating-point computation is explained through UML modeling. The performance of floating-point arithmetic operations by the use of Visual C++ and Visual C# is evaluated through a case study of numerical computations of floating-point data.

II. Background

A. Process Definition

A program in execution is called a process. A process can be defined as a block of instructions of a program that can be assigned to a processor for execution. For defining the process, first a processing unit needs to be defined. Using UML, a processing unit can be modeled using a stereotype. Stereotypes are used to define some specialized modeling

elements based on core UML base classes. Fig. 1a shows the UML stereotype definition of a processing unit and Fig. 1b shows the class diagram for representing a process.

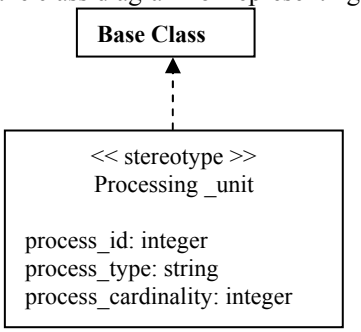

Fig. 1a: UML Stereotype of Process

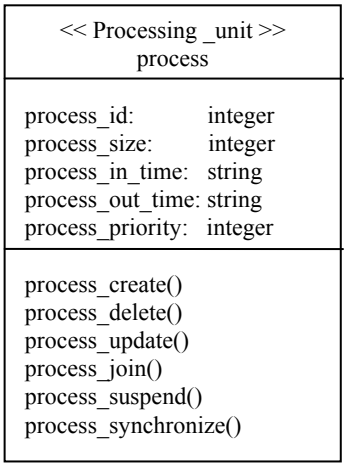

Fig. 1b: UML Class definition of Process

B. Nonlinear Pipeline

Large numerical and scientific applications normally use repeated sequence of arithmetic operations. The processors often provide pipelines to improve processing speed of such arithmetic sequences. This type of pipelining is called arithmetic pipelining. In Arithmetic pipeline the stages represents the key processing components such as adders, shifters, comparator etc. An important difference between the arithmetic pipelines and instruction pipeline is that an arithmetic pipeline may be nonlinear. In nonlinear pipeline, instead of a steady progression through a fixed sequence of stages, a task may use more than one stage at a time, or may return to the same stage at several points in processing. Nonlinear pipeline performs multiple functions and need dynamic scheduling. Floating-point computations (i.e. addition, multiplication etc.) are good examples of nonlinear arithmetic pipeline. IEEE has defined the structure and format of floating-point and is available in [22]. As an example, we have taken a multifunctional floating-point adder/multiplier pipeline. This pipeline structure performs partial shifts and partial products while doing addition and multiplication. The stages are defined as follows and are shown in Fig. 2. It is clear from the Fig. 2 that these two functions correspond to two different paths through the pipeline. The partial shifts and partial products are repeated for addition and multiplication respectively. This entire structure forms a nonlinear pipeline.

Stage1: Add/Subtract Exponents
Stage2: Partial shifts and Partial products
Stage3: Add Mantissa
Stage4: Normalise result by removing the leading 0's and adjusts the exponent

C. Intel Dual Core processor Architecture

The Dual Core processor is based on the Intel Core micro-architecture that uses CMP (i.e. core multi processors) technology. In this, two or more CPUs (known as Cores) put together on a single chip along with dual L2 caches. The Dual Core processors move hundreds of instructions into cache before executing them. The movement takes place in blocks of four or more at a time. This technique tries to execute even the most complex instructions in one clock tick. The Intel Core micro-architecture technology provides more efficient decoding stages and execution units for increasing processing capacity, reducing latency and achieving high performance. Each execution core consists of a separate pipelined integer unit and pipelined floating-point unit with dedicated adder, multiplier and divider. There are up to 12 integer and 4 floating-point registers. The architectural modeling of Intel Core micro-architecture is done by UML. The UML stereotypes for execution cores are defined. Fig. 3 shows the UML stereotype definition of Execution core. Fig. 4 shows the class diagram for representing a core and the Fig. 5 shows the single and multiple instances of core.

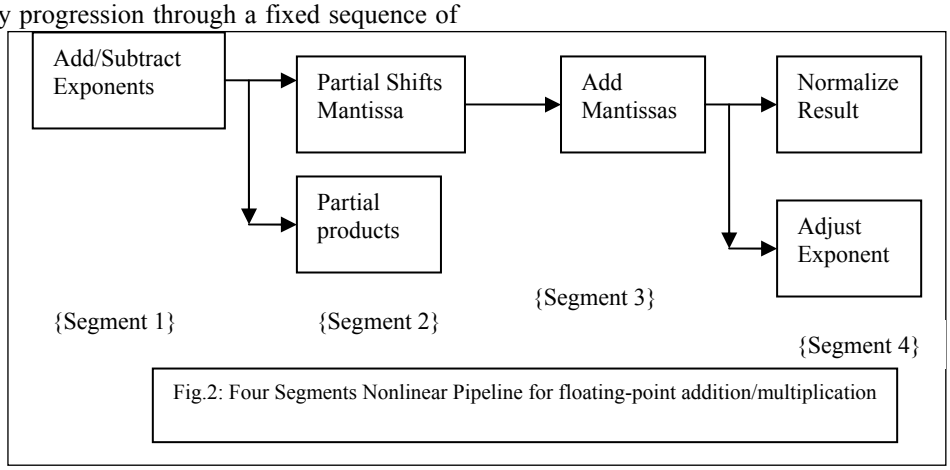

Fig.2: Four Segments Nonlinear Pipeline for floating-point addition/multiplication

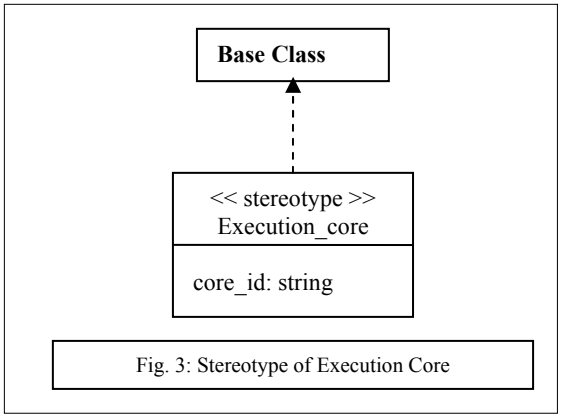

Fig. 3: Stereotype of Execution Core

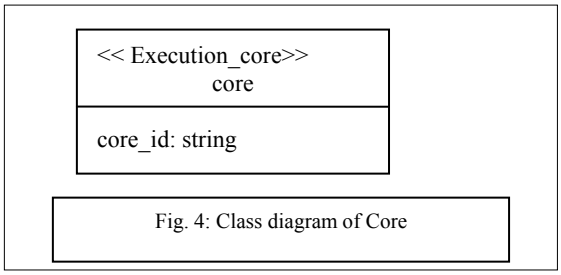

Fig. 4: Class diagram of Core

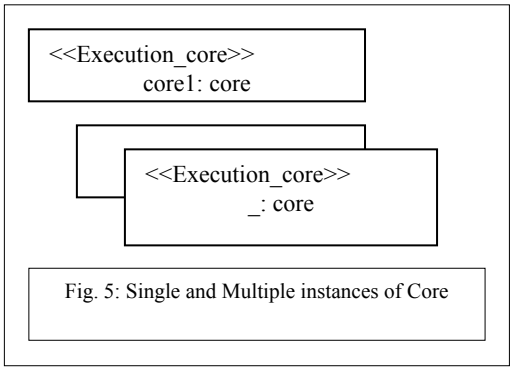

Fig. 5: Single and Multiple instances of Core

III. UML Modeling of Non Linear Pipeline Design

A. UML Class Diagram

The Fig. 6 shows the complete architectural model of Dual Core processor architecture. The class Process is directly interacting with the class Process_Execution_Controller (PEC), which is fully responsible for the execution of the assigned task.

The PEC is controlling the processes by message exchanging between the classes Processor and Memory.

The Processor class contains two cores, i.e. Core1 and Core2 and each core contains many components responsible for process execution as shown in the figure. The class diagram of the entire memory unit is also shown in the figure. Here class L2_Cache is shared between two cores and caches instructions through the class I_ Cache whereas the class D_Cache is responsible for caching the data, which is a sub class of L1_Cache. The class ALU is responsible for integer arithmetic and logical operations where as the class FPU is responsible for floating-point operations as represented in figure. This class contains three classes namely Adder, Multiply and Divider for performing multiplication, addition and division.

B. UML Activity Diagram

For proper analysis and clear understanding of multifunction nonlinear pipeline operations, a high level formalism and abstract representations are needed. Therefore, a formal visualization technique is an important approach for describing the operation of a pipeline. A UML activity diagram is designed for multifunction pipeline (Adder/Multiplication) and shown below in Fig. 7. This activity diagram clearly shows the various activities which are performed concurrently. The first activity is the addition and subtraction of exponents. The addition of exponent is performed in multiplication and the subtraction is performed for comparing the exponents in addition of floating point numbers. Since shift stage is more costly in floating-point adder operation, therefore it is required to restrict this stage to perform partial shifts and repeat this stage for a number of times to complete the addition operation. In multiplication operation, mantissas are divided in to high and lower halves, multiplying them and accumulating the partial products. Finally, the result is normalized by adjusting the exponent of the final result.

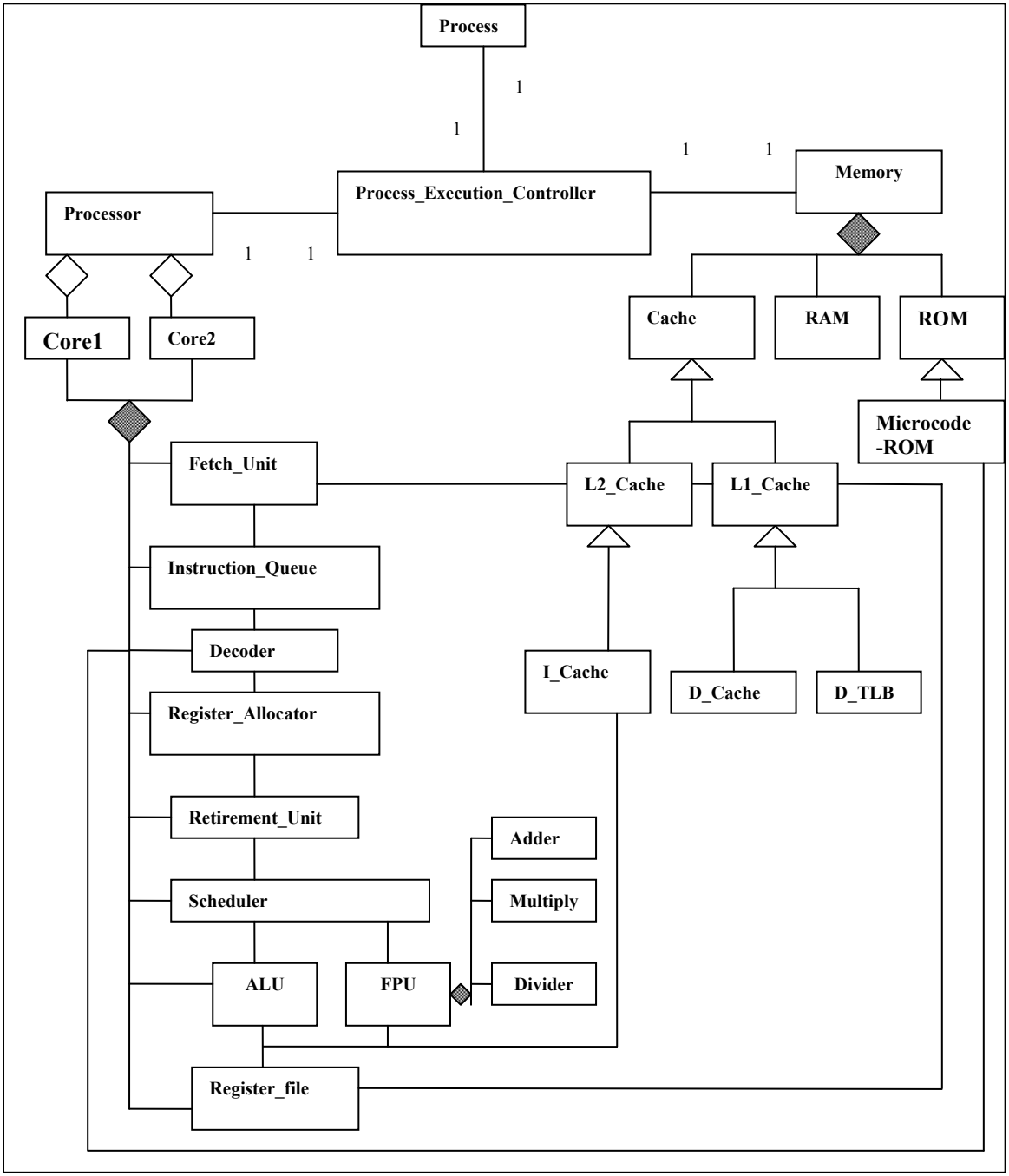

Fig. 6: UML Class Diagram for Processor Architecture

C. UML Sequence Diagram for Floating-point Operation

The Sequence diagram is used to model the dynamic aspects of the system. Through this diagram one can find how the actors interact with objects at a very high level by showing a series of sequential steps. The UML sequence diagram for floating point pipeline unit for Intel's Core micro-architecture is shown in Fig. 8. The Pentium processor executes individual instructions faster through execution pipelining, which allows multiple floating-point instructions to be executed at the same time. The scheduler receives instructions and dispatches them to the appropriate floating-point execution units. An interlocking mechanism was implemented to prevent the starting of a new instruction before the previous one is finished. The renaming logic (i.e. Register_Allocator) maps virtual registers specified by the instruction into physical registers, which access the actual Register_file (12 integer and 8 floating-point read ports) in the next stage. Instructions then initiates a floating-point add (or subtract) and a floating-point multiply operation simultaneously. The Retirement_Unit writes final results into the Register_ file (10 integer and 10 floating-point write ports).

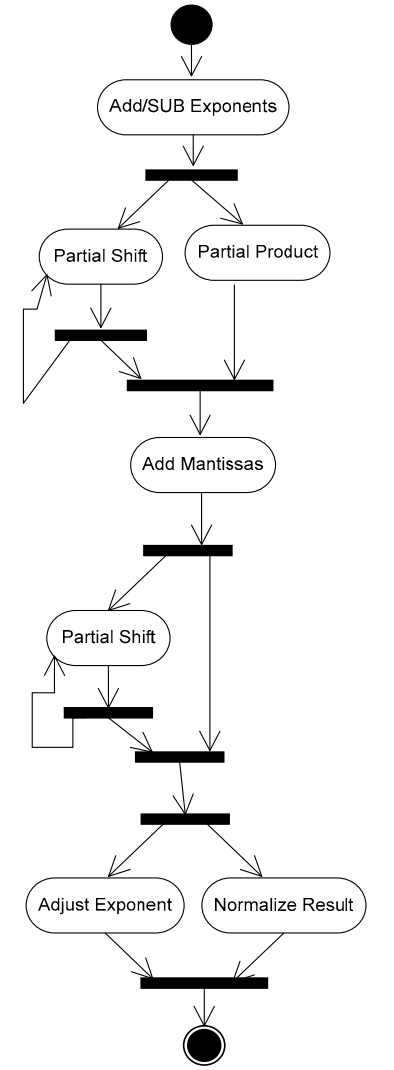

Fig. 7: UML Activity Diagram

IV. Experimental Results and Discussions

The floating-point computations are available in all major programming languages. A numeric example of summation of complex multiplications and divisions is taken as a case study. A common program is developed in programming languages namely Visual C++ and Visual C#. The computation is performed inside a loop by varying the number of repetition of terms for getting their sum. The programs are executed on Intel's Dual core processor. The specifications of the system are given in table 1. The experimental results are obtained by executing a common code written in each programming language. The Visual C++ and Visual C# programs are developed as windows applications and executed under Visual studio 2008 on Microsoft.Net framework v3.5. The execution time in each case is measured for five different runs. All the experimental results are averaged from 5 different runs and their average execution time is taken for comparison. Table 2 shows the execution time computed in milliseconds on Dual Core processor for which the experimental study is performed.

Fig. 9 clearly displays above results in the form of graph as a performance comparison in terms of execution time of number of 100 and 1000 floating-point computations, where as Fig. 10 displays the same for 10000 and 100000 computations. Based on the experimental results, it was found that Visual C++ performs better in comparison to Visual C#. It is clear from these tables that the execution time is lesser in case of Visual C++ in comparison to Visual C#.

Table 1: Architectural details of Pentium Dual Core System

Specification	Intel® Pentium® Dual Core CPU
Number of cores	02
Family	Intel Pentium Dual Core
Model number	E2200
Clock speed	2.20GHZ
Bus speed	800 MHZ
Level 1 cache size	2 x 32 KB instruction caches 2 x 32 KB data caches
Level 2 cache size	shared 1 MB
Instruction sets	MMX instruction set, SSE, SSE2, SSE3, EM64T
Memory size	1.0 GB
Operating System	Windows XP Professional, 2002, SP 2

V. Concluding Remarks

It is concluded that UML is a powerful modeling language for formal specifications of hardware systems and various scientific problems. Most of the critical applications often use a lot of floating-point computations. Nonlinear arithmetic pipelining is an architectural approach for speeding up these computations. Most of the major microprocessor architectures support multi-core technology and also support floating-point arithmetic operations. In the present paper, nonlinear pipeline for floating-point arithmetic computations is explained through UML modeling. The UML class, sequence and activity diagrams are designed. The performance of floating-point computations is evaluated on Intel Dual Core processor by executing a common code having varied number of complex floating-point computations. From the results, it is observed that Visual C++ takes less execution time as compared to Visual C# over large number of floating-point computations. Therefore Visual C++ is recommended for long floating-point computations.

Acknowledgements

The authors are very thankful to Prof. B. Hanumaiah, Vice-Chancellor, Babasaheb Bhimrao Ambedkar University (A Central University), Vidya Vihar, Rae Bareilly Road, Lucknow, India, for providing excellent computation facilities in the University campus. Thanks are also due to the University Grant Commission, India, for providing financial assistance to the Central University for research work.

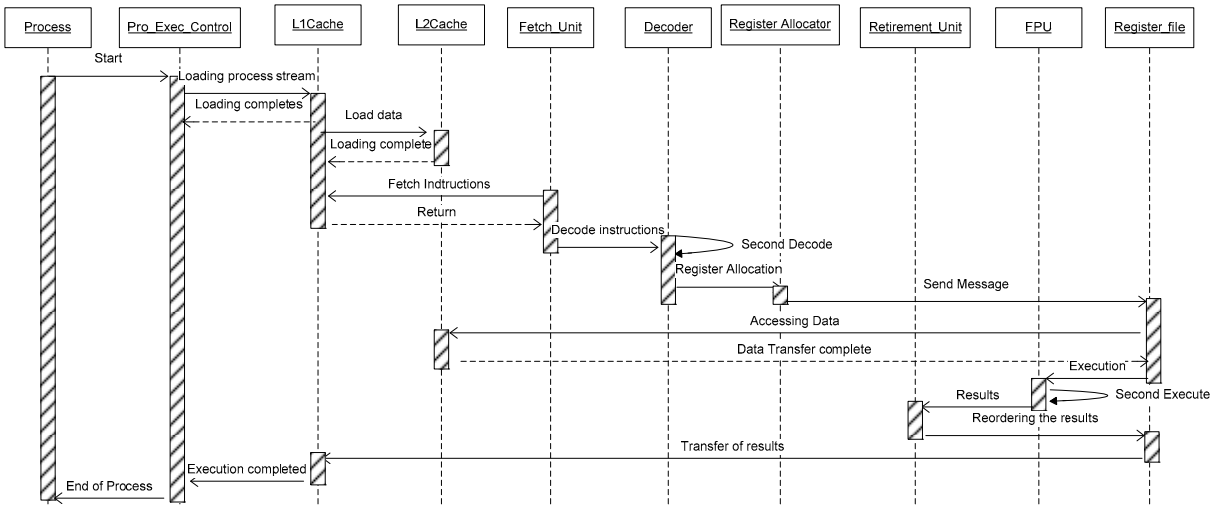

Fig. 8: UML Sequence Diagram for Floating-point Execution

TABLE 2: EXECUTION TIMES ON INTEL DUAL CORE CPU

	VC++				VC#			
Lines of Code	10^2	10^3	10^4	10^5	10^2	10^3	10^4	10^5
Execution Time in Milli Seconds	15	93	890	8953	15	109	921	9234
	15	110	890	8968	15	109	937	9171
	15	109	875	8781	15	109	921	9265
	15	94	891	8797	15	109	921	9250
	15	109	906	8813	15	109	937	9281
Average Execution Time (in ms)	15	103	890.4	8862.4	15	109	927.4	9240.2

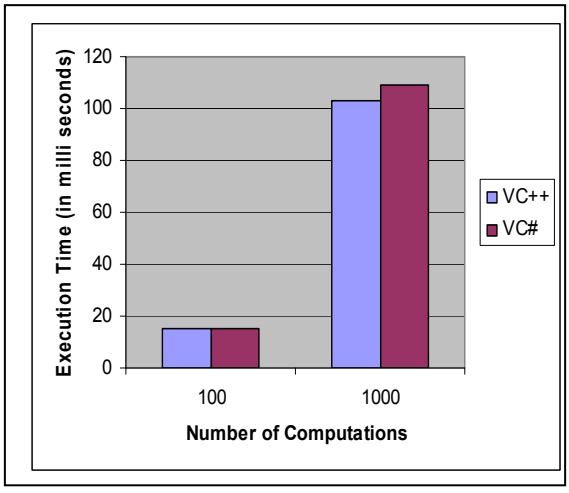

Fig. 9: Comparison of Execution Times for 100 and 1000 computations

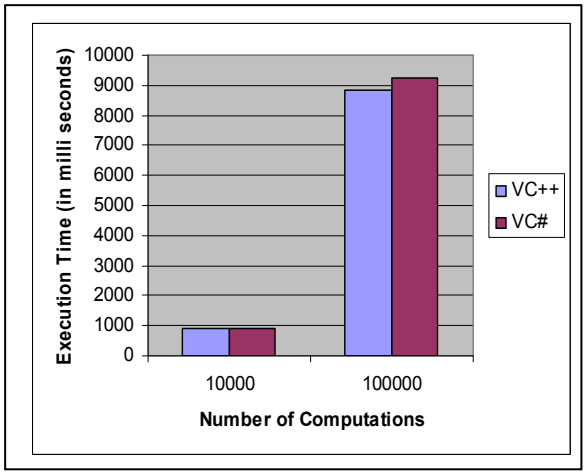

Fig. 10: Comparison of Execution Times for 10000 and 100000 computations

REFERENCES

[1] Hwang, Kai, "Advanced Computer Architecture: Parallelism, Scalability, Programmability", Fourteenth reprint, Tata McGraw-Hill Edition, ISBN-0-07-053070-X, 2007.

[2] Mano Morris M., "Computer System Architecture" Third edition, Prentice-Hall of India Pvt. Ltd. ISBN-978-81-203-0855-8, 2007.

[3] Patterson, David A., Hennessy, John L., "Computer Organization and Design: The hardware/Software Interface" Morgan Kaufmann Publishers. Elsevier Inc., 2005.

[4] Cragon, Harvey G., "Memory Systems and Pipelined Processors", Narosa Publishing House, New Delhi, 1998.

[5] Saini, Avtar (1993), Design of the Intel Pentium TM Processor, Intel Corporation, IEEE, Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00393370.

[6] Cameron McNairy, Rohit Bhatia, "Montecito: A Dual-Core, Dual-Thread Itanium Processor"IEEE Micro, vol. 25, No. 2, pp. 10-20, IEEE Computer Society, IEEE.
[7] Ofri Wechsler, "Inside Intel® Core™ Microarchitecture: Setting New Standards for Energy-Efficient Performance" Technology@Intel Magazine, March 2006.
[8] Simcha Gochman, Avi Mendelson, Alon Navh and Efraim Rotem , "Introducttion to Intel Core TM DUO Processor Architecture", Intel technology Journal, vol. 10, no. 2, May, 15, 2006.
[9] OMG, UML Superstructure Specification, v2.0, Available:http://www.omg.org/cgi-bindoc?formal/05-07-04.
[10] Booch, G., Rumbaugh, J., Jacobson, I., "The Unified Modeling Language User Guide", Twelfth Indian Reprint. Pearson Education, 2004.
[11] [11] Roff, T.,"UML: A Beginner's Guide" Tata McGraw-Hill Edition. Fifth Reprint, 2006.
[12] [12] Gomaa, H., "Designing Concurrent, Distributed, and Real-Time Applications with UML", Proceedings of the 23rd International Conference on Software Engineering (ICSE'01), IEEE Computer Society, 2001.
[13] Saxena, V., Arora D. and Ahmad S. "Object Oriented Distributed Architecture System through UML" IEEE International Conference on Advanced in Computer Vision and Information Technology (ACVIT-07), Nov. 28-30, ISBN 978-81-89866-74-7, pp. 305-310.
[14] Saxena, V. and Raj D., "UML Modeling for Instruction pipeline", World Conference on Science, Engineering and Technology (WCSET 2008), www.waset.org/pwaset.
[15] Pustina, Lukas, Schwarzer, Simon, Martini, Peter, Muurinen, Jari, Salomaki, Ari,, "A Methodology for Performance Predictions of Future ARM Systems Modelled in UML", SysCon 2008 - IEEE International Systems Conference, Montreal, Canada, April 7–10, 2008.
[16] Shirazi, N., Walters, A., and Athanas, P., "Quantitative analysis of floating point arithmetic on FPGA based custom computing machines" IEEE Symposium on FPGA's for Custom Computing Machines (FCCM '95), IEEE. Available: http://www2.computer.org/portal/web/csdl/doi/10.1109/FPGA.1995.477421
[17] Sosnowski, Janusz and Bech, Tomasz, "Extensive Testing of Floating Point Unit", Proceedings of the 26th EUROMICRO Conference (EUROMICRO'00), IEEE 2000.
[18] Kaivola, R., and Narasimhan, N., "Formal Verification of the Pentium4 Floating-Point Multiplier. Design", Automation and Test in Europe Conference and Exhibition (DATE'02), IEEE. Available: http://www2.computer.org/portal/ web/ csdl/abs/ proceedings/date/2002/1471/00/14710020abs.htm
[19] Boldo, Sylvie and Filliatre, Jean-Christophe, "Formal Verification of Floating-Point Programs", 8th IEEE Symposium on Computer Arithmetic (ARITH '07), pp.187-194. Available: http://www2.computer.org/ portal/web/csdl/doi/10.1109/ARITH.2007.20
[20] Lopez, G., Taufer, M. and Teller, P.J., "Evaluation of IEEE 754 Floating-Point Arithmetic Compliance Across a Wide Range of Heterogeneous Computers", Proceedings of the 2007 Richard Tapia Celebration of Diversity in Computing Conference, October 2007, Orlando, Florida, USA. Available: http:// gcl.cis.udel.edu/publications/ conferences/007tapia_mlopez.pdf
[21] Saxena, Vipin and Shrivastava, Manish, "UML Modeling of Static Arithmetic Pipeline Design", The ICFAI University Journal of Systems Management, vol.7, no. 1, pp. 22-31, ICFAI University press, February 2009.
[22] IEEE Standard for Floating-Point Arithmetic. IEEE Computer Society. Sponsored by the Microprocessor Standards Committee, 29 August 2008, IEEE Std 754™-2008 (Revision of IEEE Std 754-1985), Available:http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4610935&isnumber=4610934

Dr. Vipin Saxena: He is a Reader, Founder and Ex-Head, Dept. of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow, India. He got his M.Phil. Degree in Computer Application in 1992 & Ph.D. Degree work on Scientific Computing from University of Roorkee (renamed as Indian Institute of Technology, India) in 1997. He has about 14 years of teaching experience and 17 years research experience in the field of Scientific Computing & Software Engineering. He has published more than 75 International and National publications. Phone: +91-9452372550, Fax: +91-522-2440821, E-mail: vsax1@rediffmail.com

Manish Shrivastava: He is a Research Scholar, Dept. of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow, India. He got his M.Phil. Degree in Computer Applications in 1992. He has more than 12 years of teaching experience. Currently he is actively engaged in the research work on the Unified Modeling Language. He has produced several outstanding research publications. Phone: +919610753805 E-mail: mshrivastava@yahoo.com