A Computer-based Tool for Analyzing Information Flow in Pipelines

Mr. Ramander Singh [1], Mr. Vinod Kumar[2], Mr. Ajay Kumar[3]

[1,2,3]*Department of Computer Science and Engineering,*
Meerut Institute of Engineering and Technology, *Meerut, Uttar Pradesh, India*

Abstract- The Pipeline Analysis Package an interactive PC-based package for simulating data flow within an arithmetic pipeline, has been developed. It is designed for use by undergraduate and graduate students with background in computer ar-chitecture and parallel processing. Computer-based Tool for Analyzing Information Flow is an economical and effective tool in introducing arithmetic pipelines to students. It allows the student to design hardware and analyze its functionality without building the circuit. In addition, Pipelining shows detailed data flow which makes it easy for the student to envision exactly how the pipeline works. This paper presents the teacl1ing effects, as well as the performances of Computer-based Tool for Analyzing Information Flow.

Keywords: Arithmetic Pipelining, Secure Possessions

I. INTRODUCTION

High-performance computers are increasingly in demand today. Many challenges are impossible to conquer within an acceptable time period without high-performance computers. The main objective in the development of the Computer systems are to achieve the highest possible speed and throughput. Some speed up has been achieved through VLSI technology. However, increases in speed and density are limited, since VLSI technology has ultimate physical limits. A second method of increasing speed is to improve the Computer organization. This leads to the design of parallel architectures for computers. Because parallel processing can increase efficiency in computer systems, there is no question of the important role that parallel processing will play in the future of digital computer systems.

The ability to perform high-speed arithmetic calculations is a requirement for all parallel computers. Pipelining is an Economical method used to increase the throughput of arith-metic calculations. Arithmetic pipelines have been Incorporated into virtually all parallel computers for arithmetic calculations.

The Arithmetic Pipeline Analysis Package (APAP)[1] is designed for class work. That's why such computer simulation packages are being incorporated in to the curriculum at the New Jersey Institute of Technology is as follows:

Economical method: The costs associated with establishing a multiprocessor system design laboratory can be prohibitive, especially when a large number of stations is required. The use of computer-based simulation packages, which allow the student to design hardware and analyze its functionality without building the circuit, is an economical alternative. Even when circuits are to be built, analysis packages [2] allow the student to correct logic errors before implementation, leaving only timing and wiring errors to be resolved.

Effective tool: It takes about one semester to introduce the class to multiprocessors and parallel processing. The students do not have enough time to learn the course material and build and debug multiprocessor circuit. The computer simulation packages allow the students to design and simulate their circuits in a short time. The time saved in debugging circuits can be used to perform more analysis.
Easy to understand: Although the computer simulation packages cannot provide the students with more hand on experiences than actually building the circuit, they can show the detailed working conditions within the circuit which are impossible to see by looking at a physical circuit. For example, APAP can show detailed dataflow which enables the students to understand how the pipeline works.

By using APAP, the students can better understand the principles of arithmetic pipelines in class and practice designing real arithmetic pipelines. Computer-based simulation packages are also widely used in industry. Students who have used APAP in a laboratory environment will be better prepared to apply the design principles in an industrial setting.

II THE PRINCIPLES OF ARITHMETIC PIPELINES

To introduce the principles of arithmetic pipelines, the Meaning and significance of pipelining must first be given. Pipelining is similar in concept to an assembly line; the desired process is broken down into sub processes which can be performed sequentially. Although the process takes the same amount of time, different sub processes are performed on different data simultaneously, thus increasing the throughput. Pipelining provides a way to start a new task before a previous task has been finished. Therefore, the efficiency is not a function of the total processing time, but rather of the throughput of the pipeline.

Consider Figure 1, which shows a process being per- formed stage-by-stage over a period of time. The total time required

to finish this process is N time units, assuming that each stage takes onetime unit to complete its sub process. In this figure, each stage performs a single step, and the label in each stage gives the number of the step being executed.
To implement the same job using the principle of pipelining, consider Figure 2, which shows a continuous stream of jobs going through the N sequential steps of the process. In this figure, each horizontal row of boxes represents the time sequence of one sub process. Each vertical column of boxes represents the activity at a specific time. Note that up to N different jobs may be performed at any time in this example, because there are N independent stations to perform the sequence of steps in the process. The total time to perform one process is the same in Fig 1 and Figure 2, but the completion rate of jobs in Figure 2 is one per cycle as op- posed to one job every N cycle in Figure 1. Pipelining can tremendously improve the efficiency of the system.

-[]-{}{]-...-&

Figure 1: An N-step sequential process

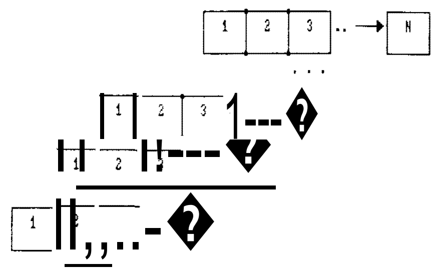

Figure2:Pipelined execution of a N-step process

Pipelining is a method used to increase the throughput of arithmetic calculations in multiprocessor systems and has led to the tremendous improvement of system throughput in modern digital computers. A basic linear arithmetic pipeline is shown in Figure 3. It consists of several different function stages. The stages' input and output data are held by latches which are under the control of the system clock. Because data streams flow through the pipes sequentially, this is called linear data flow. The other type of pipeline which has feedback connections is called an on linear arithmetic pipeline (see Figure 4). The circle in Figure 4 refers to a multiplexer which selects one desired data from several inputs and transits that data to the next stage. APAP allows nonlinear connections in order to implement multiple functions, but the timing of the feedback inputs is critical. Improper use of nonlinear data! Low may reduce or even destroy its functions. The latches and clock lines in Figures 3 and 1 are not shown in APAP.
In defining the data flow of an arithmetic pipeline, APAP provides a two-dimensional chart known as the l's reservation table. This reservation table was originally suggested by Davidson[3] and sub sequent refined by Sharand Davidson[4], Patel and Davidson[.5], and Kogge[6]. A reservation table for a linear or a static pipeline can be generated easily because data flow follows a linear stream as static pipeline performs a specific operation. But in case of dynamic pipeline or non-linear pipeline [3] a non-linear pattern is followed so multiple reservation tables can be generated for different functions. The number of columns in a reservation table specifies the evaluation time of a given function. The reservation table controls the working of the whole pipeline and maximizes the pipeline efficiency. In addition, it also prevents the pipeline operations from colliding at the same stage. A nonlinear arithmetic pipeline can have multiple functions simultaneously and their individual reservation tables can be overlaid in the same reservation chart.

The arithmetic pipeline is constructed in order to implement many arithmetic calculations and speed up the generation of results, especially in repetitious and complex cal-culations. By using arithmetic pipelines, large amounts of data can be processed within a relatively short time period.

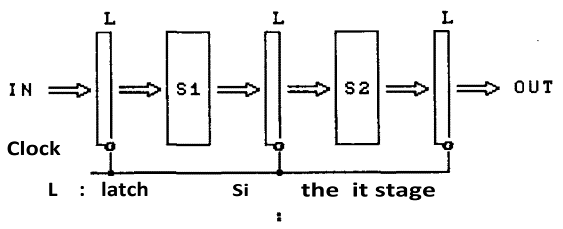

Figure 3: Llinear Arithmetic Pipeline

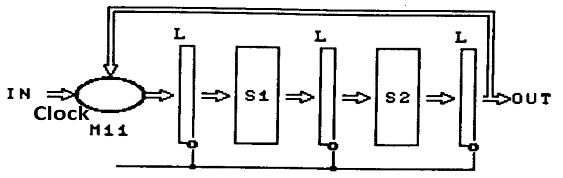

Figure 4: Nonlinear Arithmetic Pipeline

Clock drives the CPU. Each clock pulse not do the same thing rather, logic in the CPU directs successive pulses to different places to perform a useful sequence.

There are many reasons that the entire execution of a machine instruction cannot happen at once. In pipelining, effects that cannot happen at the same time are made into dependent steps of the instruction.

For example, if one clock pulse latches a value into a register or begins a calculation, it will take some time for the value to be stable at the outputs of the register or for the calculation to complete. As another example, reading an instruction out of a memory unit cannot be done at the same time that an instruction writes a result to the same memory unit.

The number of dependent steps varies with the machine architecture.

Example:-

1. The IBM Stretch project proposed the terms Fetch, Decode, and Execute that have become common.

2. The classic RISC pipeline comprises
   a. Instruction fetch
   b. Instruction decodes and registers fetch
   c. Execute
   d. Memory access
   e. Register writes back
3. The Atmel AVR and the PIC microcontroller each have a two-stage pipeline.
4. Many designs include pipelines as long as 7, 10 and even 20 stages (as in the Intel Pentium 4.
5. The later "Prescott" and "Cedar Mill" Net burst cores from Intel, used in the latest Pentium 4 models and their Pentium D and Xeon derivatives, have a long 31-stage pipeline.
6. The Xelerated X10q Network Processor has a pipeline more than a thousand stages long.

As the pipeline is made "deeper" (with a greater number of dependent steps), a given step can be implemented with simpler circuitry, which may let the processor clock run faster. Such pipelines may be called super pipelines.
A processor is said to be fully pipelined if it can fetch an instruction on every cycle. If some instructions or conditions require delays that inhibit fetching new instructions, the processor is not fully pipelined.

III. DESIGN GOALS OF APAP:

To build a multiprocessor system lab, the cost would be prohibitively high. Even if the laboratory was built, the chance for every student to use it might not be sufficient to attain the desired teaching effect.
The design goal of APAP[1] is to provide students with a good design and research environment.

Easy to use: APAP is an interactive IBM PC-based package. The PC is already used widely in school labs, for personal use at home, and in industry. The Students can use APAP conveniently which provides more chances to practice. In addition, APAP is also very easy to use. Because all commands in APAP are menu driven, it is easier for the students to operate and understand.

Great attraction: Most computer software should possess active graphics to attract the user. APAP applies the principles of computer graphics to make the results of analysis as active as possible.

Feedback from its use Produce highly readable analysis results: APAP pro has been satisfactory in this regard. Vides three choices to show data flow: tabular, simple graphic and animated graphic. Each one of these representations is helpful in its own way in understanding how the pipeline works.

Test the degree of comprehension: APAP not only allows the students to design and analyze pipelines from the screen, but it also can print the results of the analysis to the printer. APAP can act as a teaching and testing tool in class.

IV. COURSE STRUCTURE:

APAP is provided for students taking the advanced computer architecture course. In order for the students to better understand the design principles of arithmetic pipelines and APAP, they should have the following background:

1. A course in digital circuits design.
2. A course in basic computer architecture.
3. Knowledge of the basic principles of arithmetic pipelines and general knowledge related to parallel processing.
4. Elementary knowledge in operating the PC.

With the above background, the students are ready to use APAP. The basic equipment needed to run APAP[1] is an IBM PC compatible with graphics capabilities. Therefore why APAP was designed to run on an IBM PC is that all under- graduate students are issued PC/XT-compatible computers upon enrollment at NJIT [4]. Also, several PC laboratories are available on campus for student use. In addition, a printer is also helpful, since APAP can print the results of analysis directly to the printer. The APAP software occupies 96 Kbyte total on a disk this includes a main file and four overlay files. The PC should have at least 256 Kbyte of RAM in order to run APAP.

The main simulation process follows major Steps:

Construct the arithmetic pipeline: The first step to simulate the arithmetic pipeline in APAP is to construct an arithmetic pipeline. The students can design their pipelines using the Create/Modify mode in APAP. Stages and multiplexers can be added to or deleted from the pipeline easily [5]. The pipeline can be modified any time during the design process. The pipelines can also be stored and accessed later via disk.

Specify pipeline connections : This step is to define the pipeline interconnection. APAP allows the student to access a stored pipeline as well as manually enter a new one. The student defines the number of inputs and outputs for each stage, and also specifies the output of each segment as a function of the inputs. In addition, the inputs for each multiplexer must also be defined.

Define the reservation table : During the analysis process, the reservation table performs a very important role. It controls all data inputs, feedback and out puts for each segment of the pipeline. APAP supports linear pipelines, as well as those with feedback. In addition, APAP can define up to five different functions in the same pipeline. The reservation table can be accessed via disk and modified if necessary. It can also be printed.
Define input data: Input data is important for processing. APAP will show or print the results of calculations, which are dependent upon the input data. Input data can be implemented by manual entry or accessed via disk. In addition, the user can also easily modify data for other calculations.

Calculate results and show data flow: After the above processes have been specified, analysis can be implemented.

APAP will simulate the processing exactly as it would occur hardware. It responds sever cycle by activating the speaker and showing the calculation results for each specific function [8]. This makes it easy for the student to see what is going on inside the pipeline. Finally APAP will show there acquired cycles to implement such a pipeline for performance comparison. In order to provide the student with a detailed data flow to further understand the entire operating condition, APAP allows the student to choose one of the following data formats for analysis of results:

Flow diagram: It will show the interconnection diagrams of executed stages and multiplexers and their input and output data at any cycle.

Flow analysis: In order for the student to understand the data flow along the selected data paths in a pipeline, APAP can show the traced path along the entire reservation table. Active paths at each cycle are shown by dashed lines.

Flow table: It will show input and output data for each stage and multiplexer at every cycle in tabular format.

V. Conclusions

APAP has been developed and attained the aforementioned goals. Feedback from its use in graduate classes at the New Jersey Institute of Technology (N.JIT) is being used to de-fine optimal applications for students in both the graduate and undergraduate curricula. It has been shown that APAP is not only a powerful and active software tool in analyzing arithmetic pipelines but also is easy to use and understand. The students' comments indicate that APAP is very helpful to them in understanding the concepts mentioned in the lectures and some modifications involving the human factors might be helpful. APA P will be upgraded continuously for more advanced applications in the future.

In addition to APAP, the following related simulation packages are also being developed at NJIT.

1. Data Flow Dependency Generator.
2. Arithmetic Pipeline Scheduler.
3. Interconnection Network.
4. Analysis Packages.
5. Mesh Connected Computer Simulator.

The ultimate goal is to provide an economical, effective tool to the student for advanced design and research in the area of parallel processing and we believe that this goal is being attained day by day.

References

[1] Hwang, Kai and Faye A. Briggs. Computer Architecture and Parallel Processing, New York, McGraw-Hill, 1984

[2] Stone, Harold S. High-Performance Computer Architecture, Boston, Addison-Wesley, 1987

[3] Davidson, E. S. "The Design and Control of Pipelined Function Generators", Proceedings of the 1971 International Conference on Systems, Networks, and Computers, Oaxtepec, Mexico, January 1971, pp.19-21.

[4] Shar, L.E., and E.S. Davidson. "A Multi mini processor System Implemented through Pipelining", Computer, vol. 7, no. 2, February 1974, pp. 42-51.

[5] Patel, J.H., and E.S. Davidson. "Improving the Through- put of a Pipeline by Insertion of Delays", Proceedings of the Third Annual Computer Architecture Symposium, IEEE No.76 CHO 143-5C, 1976, pp. 159-163.

[6] Kogge, P.M. The Architecture of Pipelined Computers, New York, McGraw-Hill, 1981.

[7] Lin, Chang-Chieh. A PAP: The Arithmetic Pipeline Analysis Package, M.S.E.E. Thesis, New Jersey Institute of Technology, Newark, NJ, May, 1988.

[8] B. Nelson, B.I.P. Rubinstein, L. Huang, A.D. Joseph, and J.D. Tygar, “Classifier Evasion: Models and Open Problems,” Proc. Int’l ECML/PKDD Conf. Privacy and Security Issues in Data Mining and Machine Learning (PSDML ’10), pp. 92-98, 2011.

[9] R. Sommer and V. Paxson, “Outside the Closed World: On Using Machine Learning for Network Intrusion Detection,” Proc. IEEE Symp. Security and Privacy, pp. 305-316, 2010.