Dependencies Evaluation in Superscalar Processors

Vladimir Lazarov, Maria Marinova

Abstract: *The superscalar processor is instruction-level parallel (ILP) machine, capable to issue and execute simultaneously several instructions in parallel. The main limitations for the superscalar performance advantage come from the dependencies between instructions. In this paper we evaluate the influence of the different kinds of dependencies between instructions over the performance of the superscalar processors, using SIMAN simulation environment. Description of the model and simulation results are reported. Conclusions and direction for possible future work are given.*

Keywords: *Superscalar Processor, Instruction dependencies, Simulation, Performance evaluation.*

INTRODUCTION

The term *superscalar* refers to a machine that is designed to improve the performance of the execution of scalar instructions. In most applications, the bulk of the operations are on scalar quantities and the superscalar approach represents an important step in the evolution of high-performance general-purpose processors.

The superscalar processor is instruction-level parallel (ILP) machine with multiple pipelined Execution Units (EUs), capable to issue and execute simultaneously several instructions in parallel. The number of EUs defines the degree of parallelism of the superscalar processor and determines how many instructions can be issued mostly in one cycle.

The main limitations for the superscalar performance advantage come from the dependencies between instructions. There are three possible kinds of dependency – data, control and resource dependencies. The success of the superscalar approach depends on the ability of the processor to discover and resolve these dependencies.

In this paper we evaluate the influence of the different kinds of dependencies between instructions over the performance of the superscalar processors with SIMAN simulation environment.

FORMULATION OF THE TASK

The idea is to accept some base structure for a superscalar processor with given degree of parallelism, number and kind of EUs, layout of the pipelines in EUs and to develop a SIMAN model of the processor. The model is studied through instruction load with given distribution of the weights of the different groups of instructions. In the first phase the model runs are repeated for different cases: without any instruction dependencies, with false data dependencies alone, with true data dependencies additionally included, with control dependencies additionally included. For simplicity we accept that resource dependencies are resolved by the use of enough EUs. The number of executed instructions for fixed period per case is measured and reported. In the second phase of the simulation we introduce consecutively specialized hardware for detecting and resolving the different dependencies. The model runs reflect the use of *shelving* for true data dependencies, of *register renaming* for false data dependencies and of *branch target address cash* (BTAC) with *two bits branch prediction* for control dependencies. The measured number of executed instructions for any case is compared with the results from the first phase and conclusions for the influence of the instruction dependencies over the superscalar performance are formulated.

DESCRIPTION OF THE MODEL

The model is created by the means of SIMAN simulation environment. Some part of the model block diagram, showing the Integer and Branch pipeline stages, is illustrated in Figure 1. Following the full block diagram, the two files MOD and EXP are created for subsequent model runs.

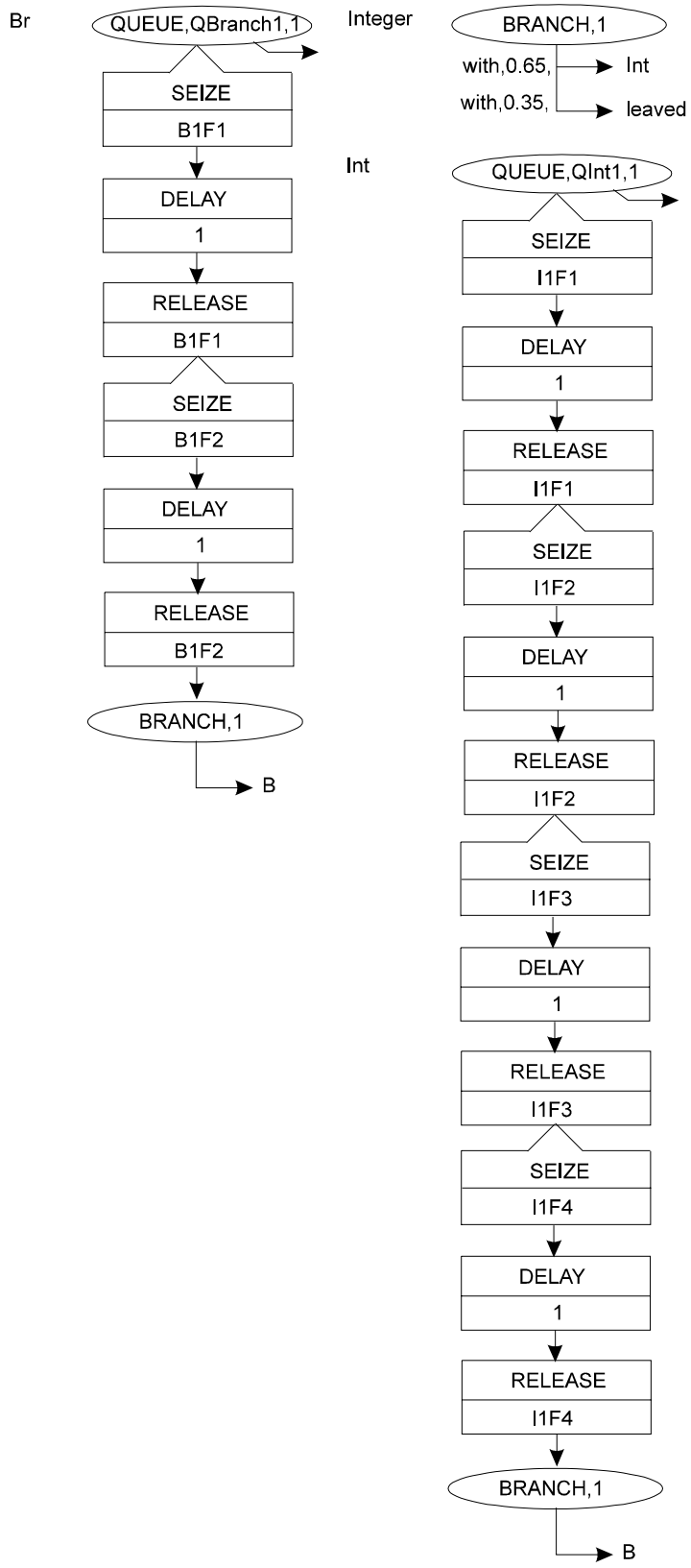

Fig.1 Simulation Model of Integer and Branch Pipelines

SIMULATION RESULTS

Some assumptions have been accepted for the simulation experiments.

The modeled superscalar structure is very close to the structure of the IBM PowerPC 601 with some improvements, part of which were introduced in the later model PowerPC 620. The degree of parallelism is 4 instead of 3; the Integer/Logical units are 2

instead of 1 with 4 pipeline stages; the Load/Store, Branch and Floating-Point units remain single with 5, 2 and 6 pipeline stages respectively; the Look-ahead Instruction Window can be increased up to 16 entries.

The accepted workload instruction distribution is: 45% - integer/logical instructions, 25% - load/store instructions, 25% - conditional branch instructions, 5% - floating-point instructions.

As mentioned before the simulated dependencies resolution techniques are: shelving for true data dependencies, register renaming for false data dependencies and branch target address cash (BTAC) with two bits branch prediction for control dependencies.

The model runs are simulated for 10 000 processor cycles and the number of the executed instructions is reported for each case. The results are shown in the following tables and diagrams.

Table 1. Performance degradation due to dependencies

	Scalar processor	Superscalar processor without dependencies	With false data dependencies	+ true data dependencies	+ control depend.
Instructions Executed per 10000 cycles	10 000	40 000	32 459	26 797	20 765

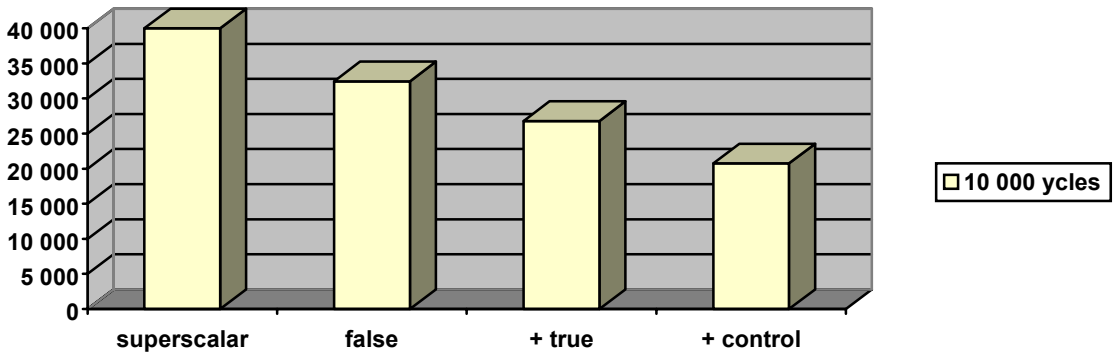

Fig 2. Superscalar performance degradation

The superscalar processor performs 4 times faster than the simple scalar processor and without instruction dependencies achieves full utilization of the EUs. However, the inclusion of the instruction dependencies impedes its processing and with all kinds of dependencies the degradation of the performance is almost twice (table 1).

The next table (table 2) shows the results in the second phase when the mentioned techniques for dependencies resolution are introduced and simulated.

Table 2. Performance improvement with dependencies resolution

	Register renaming	Branch prediction	Shelving
Instruct. Window - 8	21 885	24 500	28 958
Instruct. Window -16	22 736	25 723	30 393

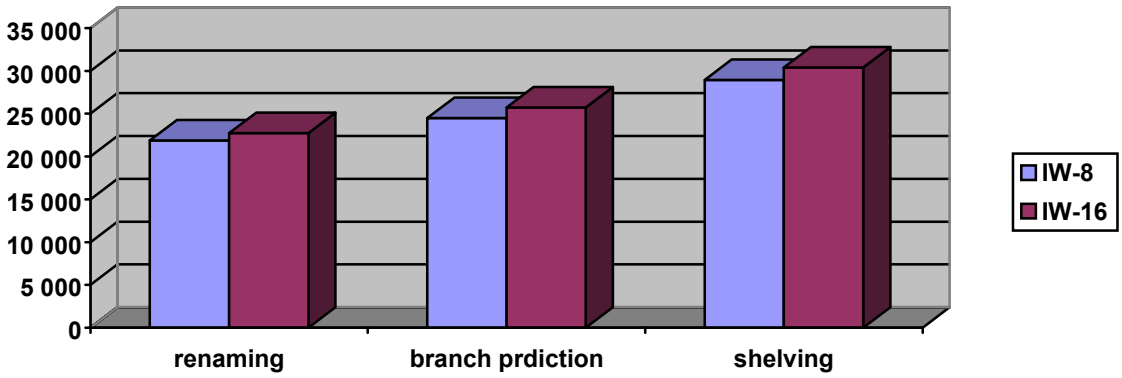

Fig 3. Resolution of dependencies

The renaming eliminates completely the false data dependencies and improves the performance with about 25%. It leads to enlargement of the pipelines with one stage. The accepted scheme for branch prediction provides 90% accuracy and improves the performance with about 28%. The shelving provides full speed of the processor despite of the presence of true data dependencies and improves its performance with about 25%. With the utilization of all three techniques the superscalar performance is about 3/4th of the theoretical limit. Further improvement can be achieved only by the means of additional static parallel code optimisation.

CONCLUSIONS AND FUTURE WORK

The simulation experiments prove the essential influence of the instruction dependencies over the performance of the superscalar processor. Their presence invalidates the effect of the use of the multiple EUs if no special techniques are introduced for detecting and resolving them. With extra hardware for coping with these ILP limitations good results can be achieved, not far from the theoretical ones, supposing enough instruction parallelism is available in the program code. This latter parameter can be improved through static parallel code optimization.

This work can be developed further by studying the superscalar processors with different structures, with different instruction distributions and with different techniques for dependencies resolution. Also, the scope of ILP processors can be enlarged. Possible fields of interest with similar approach are the performance issues in the fine-grained multithreaded architectures, i.e. simultaneous multithreaded processors (SMT), chip multiprocessors (CMP) and symmetric multiprocessors (SMP).

REFERENCES

[1] Pegden D., Shannon R., Sadowski R. Introduction to Simulation Using SIMAN, McGraw-Hill, NY, 1990.

[2] Sima Dezso, Terence Fountain, Peter Kacsuk. Advanced Computer Architectures, A Design Space Approach. Addison Wesley Longman, 1997.

[3] Stallings William. Computer Organization and Architecture, Designing for Performance. Prentice-Hall International Inc., 1996.

ABOUT THE AUTHORS

Prof. PhD. Vladimir Lazarov, Institute for Parallel Processing, Bulgarian Academy of Sciences, Phone: +359 2 736 156, e-mail: lazarov@bas.bg

Assist. Prof. Maria Marinova, Department of Computer Systems, TU Sofia - Branch Plovdiv, Phone: +359 32 659 705, e-mail: mroidova@hotmail.com