Resource Tuning in a Multipath Superscalar Architecture

Tatiana G. S. dos Santos[1], Maurício L. Pilla[1], Rafael R. dos Santos[2], Eliseu Monteiro Chaves Filho[3], Sergio Bampi[1], Philippe O. A. Navaux[1], Mario D. Nemirovsky[4]

[1] Federal University of Rio Grande do Sul
Porto Alegre, RS, Brazil
{tatiana, pilla, bampi, navaux}@inf.ufrgs.br
[2] University of Santa Cruz do Sul
Santa Cruz do Sul, RS, Brazil
{rrsantos}@inf.ufrgs.br
[3] COPPE - Federal University of Rio de Janeiro
Rio de Janeiro, RJ, Brazil
{eliseu}@lpc.ufrj.br
[4] University of California at Santa Cruz
Santa Cruz, CA, USA
{mario}@xstreamlogic.com

Abstract—

This paper presents the resource tuning in a multipath superscalar architecture model. The main goal of this model is to provide a high bandwidth for superscalar architectures that need a high instruction throughput. This architecture has an aggressive fetch mechanism to increase the number of instructions available to the execution stage of the pipeline, allowing a high number of instructions to be executed per cycle. To achieve this, instructions from multiple paths are fetched in the same clock cycle. The model results in a reduction of the occurrence of breaks in the flow. A more detailed study into the resource distribution throughout the architecture is needed. Simply increasing the number of resources to support throughput is not recommended, given the prohibitive implementation costs. The resource tuning of this superscalar architecture is then an important focus of research, to balance resource requirements and to tune them.

Keywords— **Resource Tuning, Superscalar Architectures, Multipath Fetch Mechanism.**

I. INTRODUCTION

Currently, superscalar architectures represent the state of the art in architectures that exploit Instruction-Level Parallelism (ILP). Superscalar architectures have many independent functional units and are able to dispatch and execute more than one instruction in the same cycle.

In order to deliver high effective IPC (Instruction Per Cycle), superscalar processors require a large number of instructions available to feed the dispatch stage and the several functional units [ROT 96]. However, this approach has some problems regarding instruction cache misses, branch prediction accuracy and breaks in the input instruction flow. This last factor, caused by taken branches is very common: in average 20% of the instructions are conditional branches and 70% [HEN 96] of these branches are taken. Even with good branch prediction [YEH 91] mechanisms and efficient cache memory policies, the instruction flow is continually broken, emptying the instruction queue, decreasing the number of instructions ready to execute, and therefore reducing the effective processor performance.

The architecture model proposed in [SAN 98] intends to decrease the occurrence of the flow interruption and to provide larger fetch band width through the use of an aggressive fetch mechanism. In this way, the fetch scheme feeds the dispatch stage satisfactorily, increasing the global performance of the architecture.

Nevertheless, some issues deserve careful consideration. In general the improvement achievable in the throughput in superscalar architectures is limited by three basic factors: control dependencies, data dependencies and resource conflicts. Conditional branches that break the instruction flow and operand dependencies between instructions account for the former two. The last one is relative to the number and type of resources (e.g. ports, queue, buses, functional units) demanded at runtime and those actually available in the processor.

More advanced techniques have been used to reduce the time needed to compute the outcome of a branch and the time to fetch the instruction from the target address. Branch prediction and speculative execution reduce the branch penalty by reducing the number of mispredicted executed branches. Many schemes were suggested in the 80's but only now have been actually used in the commercial machines.

The data dependencies can be solved using schemes like the Tomasulo [TOM 67] algorithm and Scoreboard [THO 64], widely used in current microprocessors.

In practice the resource conflicts can not be resolved by merely increasing the number of resources. The cost and complexity of such endeavor makes it unfeasible. Further-

more, such an architecture may not have the highest performance achievable, as it is unbalanced. Therefore, a resource tuning is needed to extract the best performance from a limited set of resources. Thus, the goal of this work is to analyse the multipath architecture regarding the resource tuning in order to get the best configuration achieved.

Although the semiconductor technology still promises significant integration, it is advancing towards reaching physical and practical limits. Then, the question of how to use the integration in the most efficient way to obtain the maximum performance becomes even more paramount.

Undoubtedly the current trend to use increasing levels of chip density and complexity to explore instruction level parallelism will continue. Relevant questions for machine architects are: how to extract more parallelismo from the code, and how to design architectures with more machine parallelism? Most likely the trend will continue towards unsing multiple functional units, multiple instruction issue architectures, higher amount of cache memories integrated within the processor, while using even more aggressive techniques for dynamic instruction scheduling [CHA 94].

In section 2 previous related research work in this field is reviewed. Section 3 presents the main goal and features of the architecture model, while section 4 deals with the specific problem approached in this paper. Section 5 discusses the methodology used to simulate and determine the best tuning for the architecture. Section 6 details the simulation results, followed by a section with conclusions and remarks.

II. Related Works

A complete review on the multipath fetch mechanism can be found in [SAN 97]. Aspects relative to control dependencies are widely explored. In that study an efficient mechanism to provide high bandwidth was proposed and the fetch stage of the MULFLUX [MUL95, MUL99] microarchitecture was firstly designed, but this architecture was not balanced. Many simulations were done with several configurations. All results were significant but an ideal tuning of the architecture has to be developed to achieve a performance improvement, which is the goal for the next step in the development.

The trace cache [SMI 97] is a possible trend to be followed in the next microprocessors generation. The study which addresses this trend assumes unlimited hardware resources to allow the fetch mechanism to feed the functional units. This ideal-scenario simulation was used just to prove the researcher's ideas to implement the scheme. In real processors, however, a study involving the balance of the system should be done.

In [WAL 93] a superscalar microprocessor was studied under many aspects. Cache, dynamic scheduling, bypassing, branch prediction and fetch efficiency were issues discussed in this work. The architectural organization was presented including mechanisms which are widely used in current architectures and the model was simulated using 17 benchmarks from 5 different suites. The simulation, however, was not used to find out the best way to distribute the resources.

The research in [CHA 95] studied the impact on the performance of superscalar processors caused by the dispatch width, the number of functional units and the processing capability of the functional units. In such case, attention has been paid to the correct use of the resources of the superscalar architecture model. Many configurations were simulated and the impact of those three aspects were observed. Nevertheless, the main goal was to identify the limitations of the superscalar processors performance and not tuning the architecture at all.Most important outcomes of the study were many relevant conclusions about the balance among the three parameters and the effects of control dependencies in the speculative execution.

It is possible to verify that the most part of the research works concentrate on the simulations to prove the viability of the proposed mechanisms. Usually, the number of hardware resources used in the simulation experiments are assumed infinite. Thus, the point addressed in this work is extremely important to implement the models in real architectures.

III. The Architecture Model

In a regular superscalar model the great bottleneck in execution throughput is the instruction dispatch low rate. Figure 1 shows the three main causes for the occurence of a no-dispatch in a regular superscalar processor, as a function of the architecture fetch width [SAN 97]. It is possible to observe that in an architecture with an 8-instruction fetch width, the percentage of no-dispatch cycles caused by empty queue represents more than 20% of the total number of the execution cycles. In the case of a 2-instruction fetch width architecture this percentage reaches almost 30%. Furthermore, the occurrence of empty queue is usually produced by the high number of breaks in the instruction flow. These happen mainly due to the ocurrence of conditional branches. The functional units process the readily available instructions faster than the fetch unit feeds the dispatch stage with the target instruction.

A solution found to this no-dispatch problem is to fetch instructions in the two possible paths of the branches. In other words, in the multipath superscalar architecture proposed by [SAN 97], the fetch stage was modified to enable the fetch of both paths of a branch instruction.

Figure 2 depicts the buffer structure in the fetch stage of the multipath pipeline. The number of flow buffers defines the fetch depth.

Each flow has four independent elements:

- Program Counter

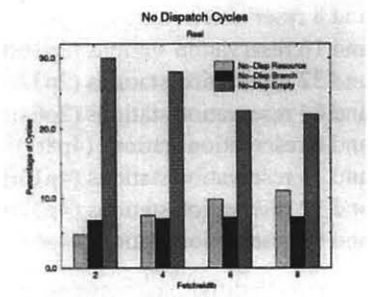

Fig. 1. No Dispatch in a real architecture

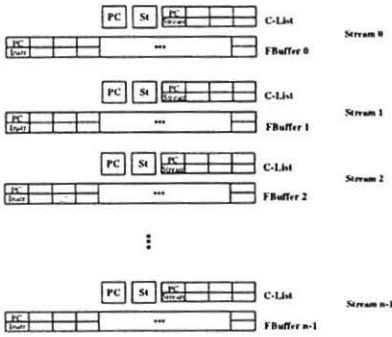

Fig. 2. Fetch Buffer Structure (e.g. fetch depth equal to 4 flows)

- Status Bit
- Children List
- Fetch Buffer

Fetched instructions are stored in the *Fetch Buffer*. The *PC* indicates the next instruction in the flow. The *status bit* indicates whether the flow structure is busy and the *Children-List* stores the identification of the children flows.

The *Children-List* also stores the branch address and the identifier of the children flows, thus allowing the predict stage to start the transfer of instructions of the new flow when a branch is predicted, without any delay.

The predict stage transfers instructions from the fetch buffer to the instruction queue (as in conventional superscalar architectures) looking for branch instructions. When it finds a branch instruction, it also makes a prediction. However, when the prediction is to be taken, this stage just concatenates the instructions which are in the children flow of this branch, discarding the closest instructions.

When the prediction is not taken, the children flow is discarded and the neighboring instructions continue to be transferred to the instruction queue. When a flow is discarded, all children flows originated by this flow are also discarded, through a recursive operation.

A simulator was developed to allow the study of the multipath superscalar model. In this simulator, the Tomasulo algorithm as well as speculative execution and two-level branch

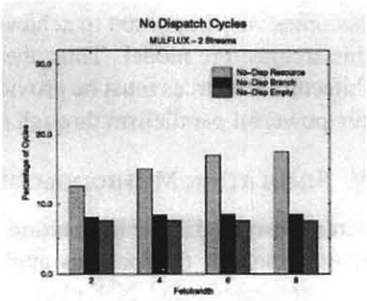

Fig. 3. No Dispatch with 2 paths

prediction were implemented. Despite the number of paths fetched, only the flow that was predicted is executed. Furthermore, it is possible to test several different configurations, varying, for example, the number of functional units, number of paths, fetch width, dispatch width, instruction queue size, besides the cache size and its policies.

IV. The Problem Approached

In the preliminary simulations [SAN 97] the multipath architecture provided satisfactory results regarding the occurence of empty instruction queue, because the architecture provides a sufficient number of instructions ready to be executed. The multipath model is evaluated according to the reasons of no-dispatch in Figure 3. As the chart shows, the percentage of lost cycles due to the occurence of empty queue decreased satisfactorily by way of fetching of two paths.

The divergence between the components is important in the real machine. The occurence of an empty queue is the most important component that may contribute to the existence of no-dispatch cycles. This is not the case for the multipath machine. It is predictable that when there are more instructions ready to dispatch, the number of functional units becomes the main cause of no-dispatch cycles. This is true in the multipath model, as Figure 3 shows.

The occurence of empty queue in the real machine decreases from 30.05%, with fetch width equal to 2 to 21.79%, with fetch width equal to 8 instructions per cycle. In the multipath model, the results obtained are 7.73% and 7.80% for the same configurations, due to the multiple path fetching.

As Figure 3 shows, now the main problem is not the empty queue occurence, but the resource conflicts causing no-dispatch. Thus, it is necessary to perform simulations to find out the best way to make use of the resources available in the architecture.

With the introduction of the multiple paths mechanism in the superscalar model, tunings found for previous architectures cannot be applied to the resulting architecture because the throughput of instructions increase, and then entire system behavior becomes different. Investigation of this par-

ticular point becomes very important to achieve high performance with this architecture model. Thus, the ideal balancing of the architecture resources must be provided to support a new and more powerful parallelism through the pipeline.

V. Simulation Methodology

The architecture described in the last section was validated by a trace-driven simulator that accepts application traces from benchmarks generated on a UNIX platform. A subset of the SPEC benchmarks (*espresso, go, cc1, m88ksim, ijpeg*) is used in the preliminary simulations as it is commonly done in other studies in this field. The *dhrystone* benchmark is used as well.

The methodology to verify the performance and to get the best tuning of the architecture is herein described.

The simulation assumed that the architecture has a perfect cache, without misses. The Branch History Table and the Pattern History Table used in the simulations have typical configurations taken from the current superscalar microprocessors.

In a preliminary work some parameter values have been identified as factors that do not limit the number of instructions executed per cycle. Hence, these parameters were fixed to:

- 9 Functional Units were used. 1 for branch and 8 for general purpose units
- The fetch width used was equal to 8
- The dispatch width used was set equal to 8
- The size of the instruction queue was equal to 64
- 30 million of instructions were simulated for each application code

The other architecture parameters needed were modified for each simulation. These parameters were varied because, according to the preliminary simulations, their values affected directly the performance of the architecture. Thus, the number of flows fetched was varied between 2 to 4 and the number of reservation stations varied as to have 8, 16, 32, 64 stations per functional unit.

Having set the range of paramenters to simulate, each configuration was simulated for every benchmark chosen. Analyzing the results makes it possible to find out the best way to balance the architecture. The results of the simulations as well as the data analyzed are shown in the next section.

VI. Results

A. Setting Up the Number of Paths

In the first simulations it was observed that the variation of the number of paths was not the bottleneck of the system. The plots in Figures 4 to 9 depict the averages, in number of instructions, of the dispatch, issue and IPC reached by the following configurations:

- 2 paths and 8 reservation stations (2p8rs)
- 2 paths and 16 reservation stations (2p16rs)
- 2 paths and 32 reservation stations (2p32rs)
- 2 paths and 64 reservation stations (2p64rs)
- 4 paths and 8 reservation stations (4p8rs)
- 4 paths and 16 reservation stations (4p16rs)
- 4 paths and 32 reservation stations (4p32rs)
- 4 paths and 64 reservation stations (4p64rs)

There are not enough difference between the performance using 2 and 4 paths that justify the cost of a four paths architecture implementation. This is caused by the existence of true data dependencies identified in the issue stage and, in this case, fetching four paths does not affect neither the pipeline execution nor the global performance. As may be viewed in the graphics below, the instructions executed per cycle (IPC) did not exceed 3 instructions, even with the reduction of the empty queue occurrence produced by the multipath utilization. Moreover, only two of the six benchmarks exceeded 2 instruction per cycle (see Figures 6 and 7) .

The main reason for this are the dependencies discussed as well as the misprediction and speculation penalties.

The misprediction penalty is the number of cycles between the misprediction detection and the dispatch of the target instruction produced by the mispredicted branch. During these cycles, the dispatch stage stalls.

Figure 10 depicts this metric in percentage of execution cycles. Each bar means a configuration and they follow the same order described above. This penalty is still high, even using a good branch predictor, and this limits the performance. As indicated in the simulation of the *dhrystone* benchmark, the penalty reaches almost 25% of the number of cycles, i. e., a quarter of execution time is lost because of misprediction. Being a synthetic benchmark, *dhrystone* is more irregular in terms of its branch behaviour.

The speculation penalty is the number of the lost cycles in the execution of invalids instructions caused by fetch/decode/dispatch of instructions that were in a misprediction path. This ratio is presented in Figure 11 and as the misprediction, the penalties caused by wrong speculations are high and have the same reasons. In the *ijpeg* simulation, 55% of the execution time is lost in cycles caused by speculation penalty.

In fact, both speculation and misprediction penalties are consequences of the branch preditor's accuracy. Thus, even with the accuracy of the branch prediction reaching 90%, those penalties are a major concern for the microprocessors designers.

All the results presented in the following sections show the two paths architecture simulation.

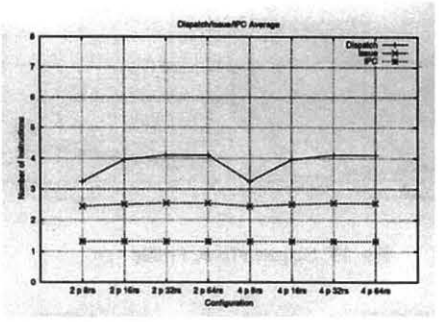

Fig. 4. Dispatch/Issue and IPC - gcc

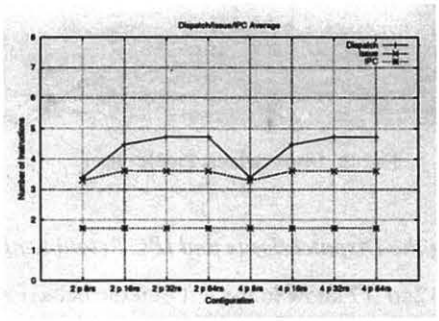

Fig. 5. Dispatch/Issue and IPC - dhry

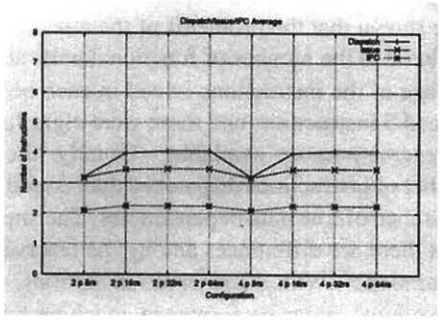

Fig. 6. Dispatch/Issue and IPC - espresso

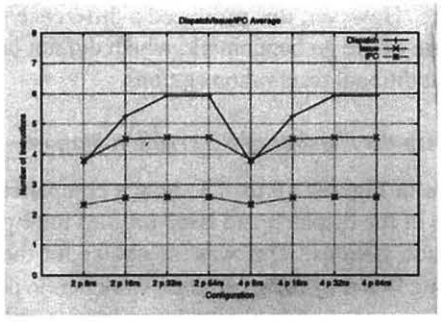

Fig. 7. Dispatch/Issue and IPC - go

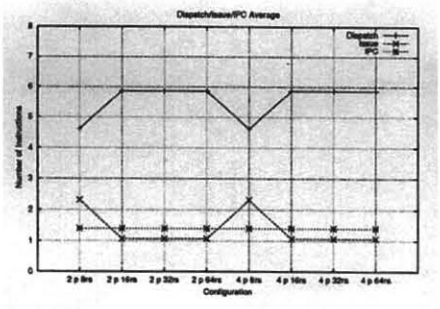

Fig. 8. Dispatch/Issue and IPC - ijpeg

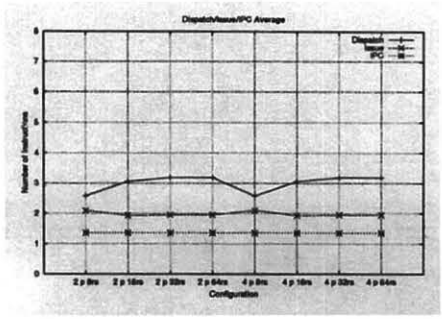

Fig. 9. Dispatch/Issue and IPC - m88ksim

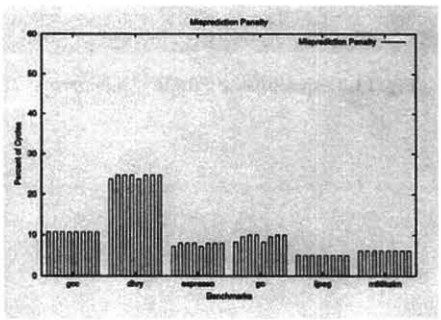

Fig. 10. Misprediction penalties

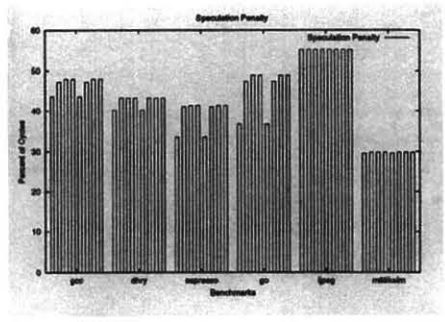

Fig. 11. Speculation penalties

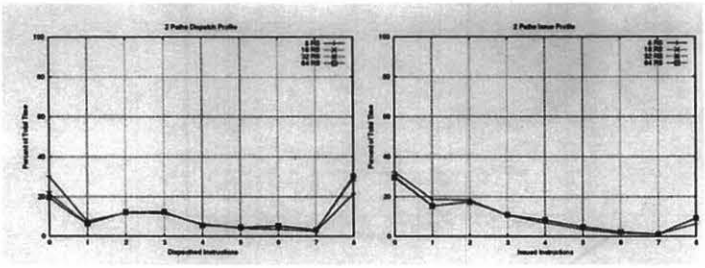

Fig. 12. Dispatch/Issue Profile - gcc

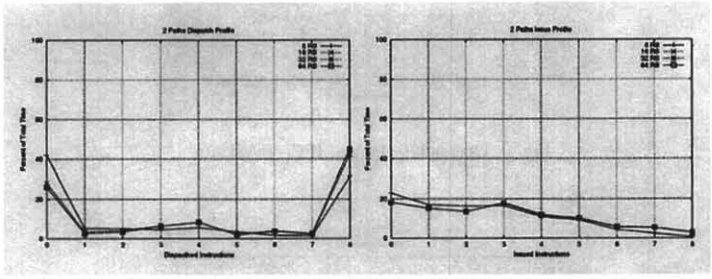

Fig. 13. Dispatch/Issue Profile - Dhrystone

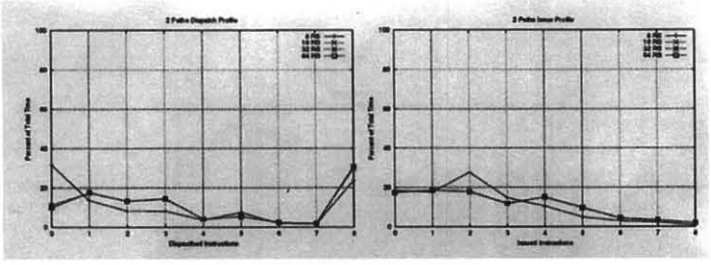

Fig. 14. Dispatch/Issue Profile - espresso

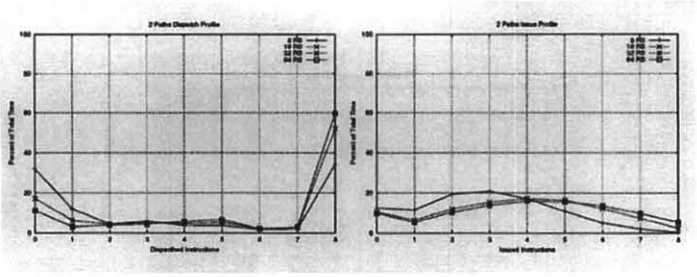

Fig. 15. Dispatch/Issue Profile - go

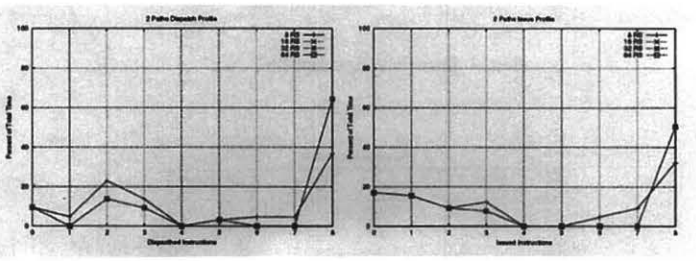

Fig. 16. Dispatch/Issue Profile - ijpeg

B. *Profiling the Dispatch/Issue and IPC Relationship*

Figures 12 to 17 show in the left plot the behaviour of the dispatch and in the right plot the behaviour of the issue for 8, 16, 32 and 64 reservation stations considering only 2 paths. It is possible to observe the variation of the instruction percentage dispatched/issued in a benchmark. The simulation results have shown that the problems of the resource conflits were not related to the number of functional units at all.

The average of the instructions issued in each benchmark did not exceed 5 instructions and there were eight functional units for general purpose available. Usually, there are a higher number of instructions dispatched than issued because of the occurrence of true data dependencies. The simulations verified that there are differences among the reservation station configurations. With 32 reservation stations, dispatch and issue exhibited better performance, reaching 64.47% of the execution time dispatching eight instructions and 50.54% of the execution time issuing eight instructions in the *ijpeg* benchmark. In few cases, 64 reservation stations got a better performance. However, this produced a difference so small, as in the case of the *go* benchmark, which did not justify the cost of 32 additional reservation stations.

C. *Setting Up the Number of Reservation Stations*

The plots in Figures 18 to 23 show a comparison in the same graph of the dispatch and issue profiles for 2 paths and 32 reservation stations. This was the choice for the best resource tuning found by the simulation work herein presented. Even in the presence of true data dependencies, this configuration held the best performance evaluating cost and bene-

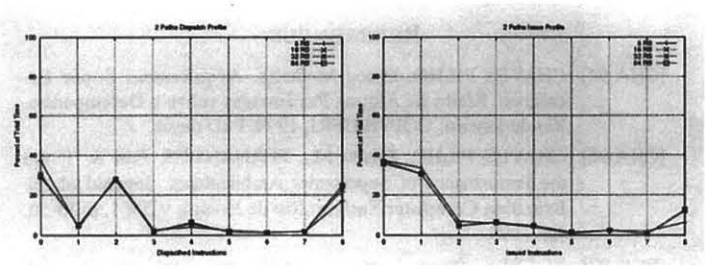

Fig. 17. Dispatch/Issue Profile - m88ksim

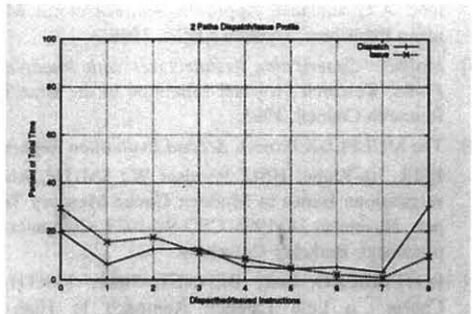

Fig. 18. Dispatch/Issue - gcc

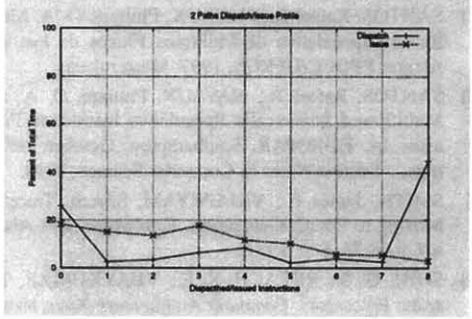

Fig. 19. Dispatch/Issue - dhry

fit produced by each one of the configurations studied. As shown in the Figure 22, in case of the *ijpeg* benchmark the 2 paths and 32 reservation stations configuration got more than 60% of the time dispatching eight instructions and also more than 50% of the time issuing eight instructions to be executed. The dispatch rate was also considerable for this tuned configuration in the other benchmark cases, being above 40% for the *go* and *dhrystone* benchmarks.

VII. Conclusions and Remarks

Many solutions for state-of-the-srt problems involving performance of computers are proposed every day. Some of these are intented for implementation in the new generation of microprocessors. In the near future one has to deal with the alternatives and intriguing possibilities of using close to one billion of transistors in a microprocessor. The alternatives on where to put these large resources to

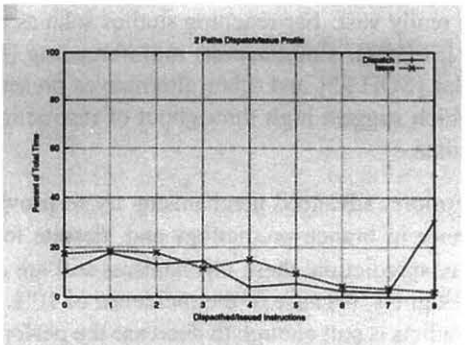

Fig. 20. Dispatch/Issue - espresso

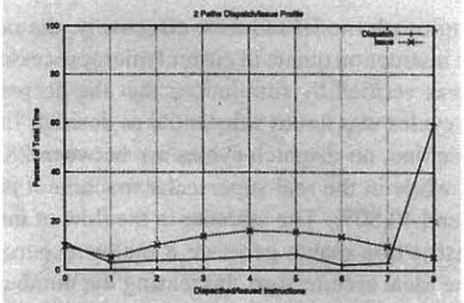

Fig. 21. Dispatch/Issue - go

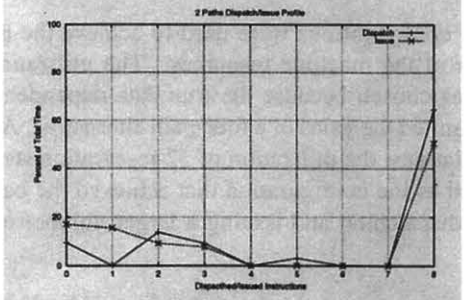

Fig. 22. Dispatch/Issue - ijpeg

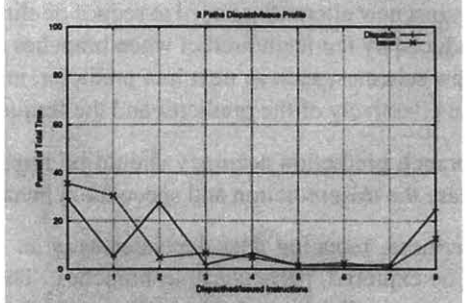

Fig. 23. Dispatch/Issue - m88ksim

work are really vast. Far-reaching studies such as trace processors [SMI 97], simultaneous multithreading [EGG 97], multiscalar [SOH 95] and other alternatives present futures trends which suggest high throughput of instruction execution and data.

Furthermore, advanced mechanisms try to provide better performance in branch predictions and, despite low probability of misprediction, these mechanisms still are a concern for the designers, because of the incidence of 10% or less of wrong predicts is still enough to decrease the performance of the processor. In the same way, new caches design [PEI 98] such as multiport, pipeline and prefetch caches intent to provide a large number of instructions with a low miss rate.

The multipath model reduces, effectively, the occurrence of empty instruction queue in current microprocessors. However, it was verified by simulations that the decrease in no-dispatch cycles was not as substantial as desired. In the multipath machine, no-dispatch cycles are between 28.99% and 33.11%, while in the real superscalar machine it is between 39.30% and 40.50%. The increase in the flow of instructions in the instruction queue generate a major resource conflict like in the ideal architecture. Increasing the number of functional units while keeping the speculation depth did not result in better figures in the experiment. Due to this finding, a new investigation about the tuning that results in the best performance of the multipath model was done.

Many configurations were used to achieve the better distribution of the machine resources. The utilization of two paths was chosen because the true data dependence occurrences limited the gains of a four-path alternative. Also based on simulations, the utilization of 32 reservation stations was identified as the configuration that achieved the best performance, dispatching and issuing a larger number of instructions.

However, true data dependencies and misprediction penalty represent still a hard limiting problem. This study led to the conclusion that these are the main topics to deserve more investigation in the future. Also, this work will be compared against new alternatives used to reduce or eliminate the hole produced by the fetch/predict when branches are taken. These new schemes, such as next line predictor, may reduce either the effectively of the predictor and the frequency.

The branch prediction accuracy should be improved and so decrease the misprediction and speculation penalties.

Furthermore, reducing data dependecies is an important point to be explored, particularly in branches. This kind of instruction normally depends on previous instructions. So, a mechanism to predict data should be develop to meet the processor future requirements.

REFERENCES

[CHA 94] CHAVES FILHO, Eliseu Monteiro. **Arquiteturas Super Escalares: Efeito de Alguns Parâmetros sobre o Desempenho**. Rio de Janeiro: COPPE/UFRJ, 1994. PhD thesis.

[CHA 95] CHAVES FILHO, Eliseu M.; FERNANDES, Edil S. T. On the Performance of Superscalar Architectures. **Journal of the Brazilian Computer Society**. Rio de Janeiro, v.2, n.1, p. 38-50, Jul. 1995.

[EGG 97] EGGERS, Susan J. et al. Simultaneous Multithreading: A Plataform for Next-Generation Processors.

[HEN 96] HENNESSY, John; PATTERSON David. Computer Architecture: A Quantitative Approach. San Francisco: Morgan Kaufmann Publishers, segunda edição, 1996.

[MUL95] *Mulflux: Superscalar Architectures with Multiple Instruction Flows*, Research Proposal submitted to the Brazilian National Research Council, 1995.

[MUL99] The MULFLUX Project, *Second Evaluation Report*, May 1999.

[PEI 98] PEIR, Jih-Know; HSU, Windsor W.; SMITH, Alan J. **Implementations Issues in Modern Cache Memory** Technical Report. November 18, 1998. CSD-98-1023: Computer Science Departament, Berkeley University.

[ROT 96] ROTEMBERG, Eric; BENNET, Steve; SMITH, Jim Trace Cache: a Low Latency Approach to High Bandwidth Instruction Fetching. Madison: University of Wisconsin-Madison/Computer Science Departament, 1996. Technical Report.

[SAN 97] SANTOS, Rafael R.; NAVAUX, Philippe O. A. Mecanismo de Busca Especulativa de Múltiplos Fluxos de Instruções. Porto Alegre: PPGCC/UFRGS, 1997. Master thesis.

[SAN 98] SANTOS, Rafael R.; NAVAUX, Philippe O. A. Analysing a Multiflowed Superscalar Speculative Instruction Fetch Mechanism. In: EUROPAR. Southampton, October 1998. **Proceedings....** Lecture Notes in Computer Science, 1998.

[SMI 97] SMITH, James E.; VAJAPEYAM, Sriram. Trace Processors: Moving to Fourth-Generation. **Computer**, Los Alamitos, v.30, n.9, p.68-74, Sep. 1997.

[SOH 95] SOHI, G. S.; BREACH, S. E.; VIJAYKUMAR, T. N. Multiscalar Processors. *Computer Architetcure News*, New York, v.23, n.2, p. 414-425, 1995.

[TOM 67] TOMASULO, R. M. An Efficient Algorithm for Exploiting Multiple Arithmetic Units. **IBM Journal of Research and Development**, Armonk, v.11, n.1, p. 25-33, Jan. 1967.

[THO 64] THORNTON, James E. **Parallel Operation in the Control Data 6600**. In: AFIPS Fall Joint Computer Conference, 1964, v.26, pt. 2, pp. 30-40.

[WAL 93] WALLACE, Steven D. **Performance Analisys of a Superscalar Architecture**, Irvine: University of California, 1993. PhD thesis

[YEH 91] YEH, Tse-Yu; PATT, Yale N. Two-Level Adaptive Training Branch Prediction. In: ANNUAL INTERNATIONAL SYMPOSIUM ON MICROARCHITECTURE, 24., 1991. **Proceedings...** New York: ACM, 1991. p. 51-61.