Prediction and Speculation in ILP Processors

by

Pece J. Mitrevski, M.S.E.E.

An extended abstract
of the dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
(Computer Science)

at the

Ss. Cyril and Methodius University – Skopje
Faculty of Natural Sciences and Mathematics
Institute of Informatics

2001

DOCTORAL COMMITTEE:

DR. MARJAN GUŠEV

ASSOCIATE PROFESSOR

INSTITUTE OF INFORMATICS
FACULTY OF NATURAL SCIENCES AND MATHEMATICS
SKOPJE, MACEDONIA

DR. DAVID J. EVANS

PROFESSOR

NOTTINGHAM TRENT UNIVERSITY
FACULTY OF ENGINEERING AND COMPUTING
NOTTINGHAM, UNITED KINGDOM

DR. MILE STOJČEV

PROFESSOR

FACULTY OF ELECTRONIC ENGINEERING
NIŠ, YUGOSLAVIA

DR. GJORGJI JOVANČEVSKI

ASSOCIATE PROFESSOR

INSTITUTE OF INFORMATICS
FACULTY OF NATURAL SCIENCES AND MATHEMATICS
SKOPJE, MACEDONIA

DR. DANILO GLIGOROSKI

ASSISTANT PROFESSOR

INSTITUTE OF INFORMATICS
FACULTY OF NATURAL SCIENCES AND MATHEMATICS
SKOPJE, MACEDONIA

Modeling based analysis is applied to assess the efficiency of predictions and speculative execution in boosting the performance of ILP processors that fetch, issue, execute and commit a large number of instructions per cycle. The primary objective of the analysis is to better understand branch and value prediction techniques and their performance potential under different scenarios, with varying parameters of both the microarchitecture, as well as the operational environment. So far, the estimations of the influence that these techniques have on ILP processor performance, more or less, rely upon the use of microarchitectural simulators with trace-driven or execution-driven simulation – only a few simple deterministic models are known. As opposed to this common trait, a stochastic model of the dynamic behavior of an ILP processor that aims to minimize the impact of control and true data dependences and employs speculative execution based on branch and value prediction, is introduced for the first time in this dissertation. At the same time, the characteristics of the operational environment are formalized and their influence on ILP processor performance is considered, as well. The proposed dynamic behavior model is built using Fluid Stochastic Petri Nets (FSPN) and the stochastic process underlying the Petri Net is described by a system of first-order hyperbolic partial differential equations with appropriately chosen initial and boundary conditions. The attempt to capture the dynamic behavior of an ILP processor is a rare example that demonstrates the usage of this recently introduced formalism in modeling actual systems. Moreover, it is the first example to take into consideration numerical transient analysis and present numerical solution of a Fluid Stochastic Petri Net with more than three fluid places. Both the application of finite-difference approximations for the partial derivatives, as well as the discrete-event simulation of the proposed FSPN model, allow for the evaluation of a number of performance measures and lead to numerous conclusions regarding the performance impact of predictions and speculative execution with varying parameters of both the microarchitecture and the operational environment. The numerical solution makes possible the probabilistic analysis of the dynamic behavior, whereas the advantage of the discrete-event simulation is the much faster generation of performance evaluation results, even when compared to the broadly utilized microarchitectural simulators. The proposed novel approach is essentially implementation-independent and reveals considerable potential for further research in this area using numerical transient analysis and/or discrete-event simulation of FSPN models.

1 Motivation and objective

In its brief lifetime of just about 30 years, the microprocessor has achieved a total performance growth of about 9,000X due to both *technology improvements* and *microarchitecture innovations*. Both transistor count and clock frequency have increased by an order of magnitude in each of the last three decades. During the 1980's *the average number of Instructions Per Cycle* (IPC) also increased by almost an order of magnitude, which is not the case in the last decade (Table 1) [60, 61]. While the contribution to performance by technology improvements seems to be accelerating, the contribution from microarchitecture innovations seems to be diminishing. It is clear that the anticipated technology can support the implementation of much wider and deeper machines. The key question is *how to achieve additional IPC proportional to the increase of machine resources*.

Table 1. Evolution of Microprocessors

	1970-1979	1980-1989	1990-1999	2000+
Transistor Count	10k-100k	100k-1M	1M-30M	100M
Clock Frequency	0.2-2MHz	2-20MHz	20-500MHz	1GHz
Instructions/Cycle	<<0.1	0.1-0.9	0.9-2.5	10 (?)
MIPS/MFLOPS	<<1	1-20	20-1,000	10,000

Current microprocessor architectures assume *sequential programs* as an input and a *parallel execution model*. Their efficiency is highly dependent on the *Instruction-Level Parallelism* (ILP) the programs exhibit, as *a measure of the potential number of instructions that can be executed simultaneously*. Basically, there are two fundamental approaches to executing multiple instructions in parallel: the *superscalar* processor and the *VLIW* (Very Long Instruction Word) processor. The superscalar approach extracts the parallelism from a sequential program at run time (dynamically), by employing sophisticated hardware mechanisms. On the other hand, the performance of VLIW architectures is dependent on the capability of the compiler to detect and exploit instruction-level parallelism during instruction scheduling using advanced (*static*) techniques.

Regardless of the approach used, instructions cannot always be eligible for parallel execution due to three classes of constraints: *control dependences*, *true data dependences* and *name (false) dependences* [45]. *Control dependences* appear when the execution of instructions depends on the outcome of a branch (either conditional or unconditional). The resolution of the outcome involves deciding whether the branch is taken or not (in the case of conditional branch) and computing its target address. As long as control dependences are not manipulated, control dependent instructions cannot even begin their execution because the branch outcome should determine the next sequence of instructions to be executed. *True data dependences* occur when an instruction consumes a value that is generated by a preceding instruction, and therefore these instructions cannot be executed in parallel. *Name dependences* occur when instructions use the same register or memory location (*name*), but there is no flow of data between them as in true data dependences. One can distinguish between two kinds of name dependences: *output dependences* – when instructions attempt to write simultaneously to the same name, and *anti-dependences* – when a later instruction attempts to write to a name before it is read by an earlier instruction. As long as name dependences are not handled, they can avoid instructions from being executed in parallel since the program correctness would be violated.

Several techniques eliminate the effect of dependences or reduce their consequences. For example, *data forwarding* and implementation of *shelving* reduce the effect of true data dependences, while *renaming* eliminates the name dependences. However, the potential of upcoming processors with hundreds of processing units requires more parallelism than the one obtained by these techniques. One alternative is the introduction of advanced *elimination* techniques [7, 75], and the other is the introduction of various *speculation* techniques based on *prediction*.

Many ILP processors speculatively execute control dependent instructions before resolving the branch outcome. They rely upon *branch prediction* in order to tolerate the effect of control dependences. A branch predictor uses the current fetch address to predict whether a branch will be fetched in the current cycle, whether that branch will be taken or not, and what the target address of the branch is. The predictor uses this information to decide where to fetch from in the next cycle. Since the branch execution penalty is only seen if the branch was mispredicted, a highly accurate branch predictor is a very important mechanism for reducing the branch penalty in a high performance ILP processor.

Given that a majority of static instructions exhibit very little variations in values that they produce/consume during the course of a program's execution, data dependences can be eliminated at run-time by predicting the outcome values of instructions (*value prediction*) and by executing the true data dependent instructions. In general, the outcome value of an instruction can be assigned to registers, memory locations, condition codes, etc. The execution is speculative, as it is not assured that consumer-instructions were fed with correct

input values. Since the correctness of the execution must be maintained, speculatively executed instructions retire only if the predictions they rely upon were proven to be correct – otherwise, they are discarded and reissued with the correct values.

Vast majority of the studies on speculative execution based on predictions have been conducted in two mutually orthogonal directions:

- *providing new refined prediction models* [17, 20, 25, 55, 64, 65, 93, 98, 106, 110, 114] and
- *improving implementation of existing prediction models* [12, 13, 21, 50, 51, 53, 54, 65, 66, 67, 68, 77, 78, 81, 84, 87, 88, 91, 95, 101, 111, 112].

They all have a common objective – *increasing the number of correct predictions*, based on the assumption that each particular successful elimination of a control or true data dependence improves the performance of an ILP processor. These previous works focus on evaluating the impact of different prediction models on a particular processor microarchitecture.

The aim of a plethora of works [10, 28, 29, 30, 31, 34, 52, 79, 90, 100] has been *studying the limits of ILP*, i.e. *the influence that predictions and speculative execution have on ILP processor performance under different scenarios (microarchitectural features)*: instruction fetch bandwidth, prediction accuracy, available resources, instruction window size, issue width, etc. More or less, they all rely upon the use of microarchitectural simulators with either:

- *trace-driven simulation*, or
- *execution-driven simulation*.

Simulators [3, 8, 18, 19, 40] produce measurements of the dynamic use of machine resources, the throughput at various pipeline stages, and ultimately the performance of the machine, measured in IPC, given a particular processor microarchitecture. As processor complexity continues to increase at a rapid rate and microarchitectures continue to become more speculative, it is not clear whether the implementation and use of performance simulators can continue to effectively predict actual machine performance due to several crucial issues [4]:

- *simulator retargetability* – reuse for the next generation design;
- *simulator validation* – in the early design phase the hardware model is unavailable;
- *simulation speed* – lengthening of the traces used and simulation time growth;
- *simulation accuracy* – due to fundamental limitations of this paradigm.

In such circumstances, one alternative is the analytical modeling approach – a set of formulas or equations describe the system, and manipulating or solving the equations leads to results that describe the system behavior. In simpler cases, equations can be solved to get a closed-form answer, but more often a numerical solution needs to be carried out. Analytical models are generally more of an abstraction of the system than the microarchitecture models used in simulators. Nevertheless, the models published so far do not even distantly capture the dynamic behavior of an ILP processor with speculative execution based on predictions. Only a few *deterministic* models are known [29, 67, 90] – they deal with average parameter values (or a parameter takes only one value), there is no randomness and the result is based on known functions of inputs with no dependence on chance. Some authors point out that *these models provide some insight by isolating important parameters, but they are still too simple to capture the behavior of a real system.*

Therefore, the primary objective of this work is *to assess the efficiency of predictions and speculative execution in boosting the performance of ILP processors, using stochastic modeling of the dynamic behavior of an ILP processor where a large number of instructions are fetched, issued, executed and committed per cycle. The analysis is needed to better understand branch and value prediction techniques and their performance potential under different scenarios, with varying parameters of both the microarchitecture, as well as the operational environment.*

2 Main contributions

As opposed to the broadly utilized microarchitectural simulators and the deterministic models known to date, a stochastic model of the dynamic behavior of an ILP processor that aims to minimize the impact of both control and true data dependences and employs speculative execution based on branch and value prediction, is introduced for the first time in this dissertation. It relies on the use of random variables and their distribution functions. Among the rest, assumed is that:

- The distribution of the time between two consecutive occurrences of branch instructions in the instruction stream is exponential with rate λ. The rate depends on the instruction fetch bandwidth, as well as the program's average basic block size;
- The distribution of the time between two consecutive occurrences of consumer-instructions (instructions that consume values generated by simultaneously initiated producer instructions) in the instruction stream is exponential with rate μ. The rate depends on the number of instructions that simultaneously initiate execution at a functional unit, as well as the program's average dynamic instruction distance.

The model is able to deal with:

- finite processor resources;
- realistic (non-perfect) branch prediction and realistic value prediction, at the same time;
- arbitrary prediction accuracy;
- random number of instructions per branch;
- random branch misprediction position;
- random dynamic instruction distance between producer- and consumer-instructions.

Moreover, given that different programs (instruction streams) have wildly different behavior during their execution across many different metrics (available parallelism, branch predictability, value predictability, occupancy of resources, etc.), the characteristics of the operational environment are formalized and their influence on ILP processor performance is considered, as well. Programs that exhibit as nearly as possible homogenous predictability behavior represent a single program class. The input space is partitioned in order to reduce the vast number of possible input states, and probabilities that do not depend on time characterize the occurrence of programs from the different classes. This is an adaptation of an already

known concept for modeling the operational environment of N-version fault tolerant software [35, 70].

The proposed dynamic behavior model is built using Fluid Stochastic Petri Nets [36, 38, 39, 47, 103, 108]. FSPNs contain two types of places: discrete places containing a non-negative integer number of tokens, and continuous places containing fluid (non-negative real quantity). Transition firings are determined by both discrete and continuous places, and fluid flow is permitted either with deterministic fluid rates through the enabled timed transitions, or in the form of fluid jumps (transportation of fluid in zero time) through enabled immediate transitions in the Petri Net. The main motivation of this approach emerges from the observation that in a machine that employs multiple execution units capable to execute large number of instructions in parallel, the service and storage requirements of each individual instruction are small compared to the total volume of the instruction stream. Individual instructions may then be regarded as atoms of a fluid, and large buffer levels approximated by continuous fluid levels. As a result, state-space complexity is decreased. From the FSPN point of view, despite the microarchitectural complexity, fairly simple concept lies beneath the dynamic behavior model: instructions flow and pass through separate pipeline stages connected by buffers. Control dependences stall the inflow of useful instructions (fluid) into the pipeline, whereas true data dependences decrease the aperture of the pipeline and the outflow rate. The buffer levels always vary and affect both the inflow and outflow rates. The speculative execution based on branch prediction tends to eliminate stalls in the inflow, while the speculative execution based on value prediction helps keeping the outflow rate as high as possible.

Based on the assumption that the pipeline is organized in, more or less, four stages – Fetch, Decode/Issue, Execute and Commit, fluid places $P_{IC,}$ $P_{IB,}$ $P_{RS/LSQ,}$ $P_{ROB,}$ $P_{RR,}$ P_{EX} and P_{REG}, depicted by means of two concentric circles (Fig. 1), represent buffers between pipeline stages – instruction cache, instruction buffer, reservation stations & load/store queue, reorder buffer, rename registers, instructions that have completed execution and architectural registers. The fluid place P_{TIME} has the function of an hourglass: it is constantly filled at rate 1 up to the level 1 and then flushed out, which corresponds to the machine clock cycle. Fluid arcs are drawn as double arrows to suggest a pipe. Flow rates are piecewise constant, i.e. take different values at the beginning of each cycle and are limited by the fetch/issue width of the machine (W). Rates depend on fluid levels and change when T_{CLOCK} fires and the fluid in P_{TIME} is flushed out. The flush-out arc is drawn as thick single arrow. A high-bandwidth instruction fetch mechanism fetches up to W instructions per cycle with rate r_{FETCH} and places them in the instruction buffer. In the case of a branch misprediction, the fetch unit is effectively stalled and no useful instructions are added to the buffer. Instruction cache misses are ignored. Instruction issue tries to send W instructions to the appropriate reservation stations or the load/store queue on every clock cycle. The actual issue rate is r_{ISSUE}. Rename registers are allocated to hold the results of the instructions and reorder buffer entries are allocated to ensure in-order completion. Among the instructions that initiate execution in the same cycle, speculatively executed consumer-instructions are forced to retain their reservation stations. Up to W instructions are *in execution* at the same time. With the assumption that functional units are always available and out-of-order execution is allowed, the instructions *initiate* and *complete* execution with rate $r_{INITIATE} = r_{COMPLETE}$. During the execute stage, the instructions first check to see if their source operands are available (predicted or computed). For simplicity, the execution latency of each instruction is a single cycle. Instructions execute and forward their own results back to subsequent instructions that might be waiting for them (no result forwarding delay). Every reference to memory is present in the first-level cache –

the effect of the memory hierarchy is eliminated. The instructions that have completed execution are ready to move to the last stage. Up to *W* instructions may commit per cycle. The results in the rename registers are written into the register file and the rename registers and reorder buffer entries freed with rate r_{COMMIT}. In order to capture the relative occurrence frequencies of different program classes, a set of weighted immediate transitions is introduced in the Petri Net. Each program class is assigned an immediate transition T_{CLASS_i}. The operational profile is a set of weights w_{CLASS_i}, and the probability of firing the immediate transition T_{CLASS_i} represents the probability of occurrence of a class *i* program.

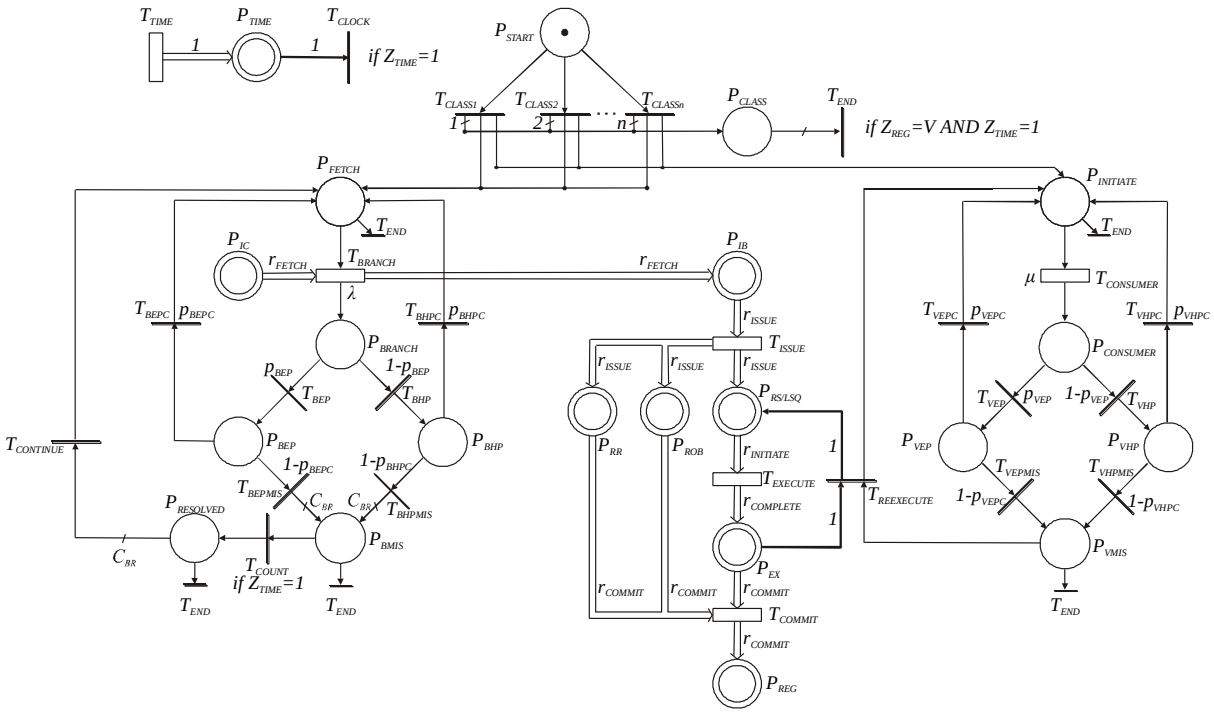

Figure 1. The Fluid Stochastic Petri Net model

A token in P_{START} denotes that a new execution is about to begin. The process of firing an immediate transition randomly chooses a program from one of the classes. The firing of transition T_{CLASS_i} puts *i* tokens in place P_{CLASS}, which identify the class. At the same time instant, tokens occur in places P_{FETCH} and $P_{INITIATE}$, while the fluid place P_{IC} is filled with fluid with volume V_i equivalent to the total number of useful instructions (program volume). Firing of exponential transition T_{BRANCH} corresponds to a branch instruction occurrence. The parameter λ changes at the beginning of each clock cycle and formally depends on both the number of tokens in P_{CLASS} and the fetch rate: $\lambda = \lambda_i \cdot r_{FETCH} / W$, where λ_i is its upper limit for a given program class *i* at maximum fetch rate (r_{FETCH}=*W*). The branch is classified as easy-to-predict with probability p_{BEP}, or hard-to-predict with probability $1\text{-}p_{BEP}$. In either case, it is correctly predicted with probability p_{BEPC} (p_{BHPC}), or mispredicted with probability $1\text{-}p_{BEPC}$ ($1\text{-}p_{BHPC}$). These probabilities are included in the FSPN model as weights assigned to immediate transitions T_{BEP}, T_{BHP}, T_{BEPC}, T_{BHPC}, T_{BEPMIS} and T_{BHPMIS}, respectively. This is known as synthetic branch prediction. Branch mispredictions stall the fluid inflow for as many cycles as necessary to resolve the branch (C_{BR} tokens in place P_{BMIS}). Usually, a branch is not resolved until its execution stage (C_{BR}=3). With several consecutive firings of T_{CLOCK},

these tokens are consumed one at a time and moved to $P_{RESOLVED}$. As soon as the branch is resolved, transition $T_{CONTINUE}$ fires, a token appears in place P_{FETCH} and the inflow resumes. Similarly, firing of exponential transition $T_{CONSUMER}$ corresponds to the occurrence of a consumer-instruction among the instructions that initiated execution. The parameter μ changes at the beginning of each clock cycle and formally depends on both the number of tokens in P_{CLASS} and the initiation rate: $\mu = \mu_i \cdot r_{INITIATE} / W$, where μ_i is its upper limit for a given program class i when maximum possible number of instructions simultaneously initiate execution ($r_{INITIATE}=W$). The consumed value is classified as easy-to-predict with probability p_{VEP}, or hard-to-predict with probability $1\text{-}p_{VEP}$. In either case, it is correctly predicted with probability p_{VEPC} (p_{VHPC}), or mispredicted with probability $1\text{-}p_{VEPC}$ ($1\text{-}p_{VHPC}$). These probabilities are included in the FSPN model as weights assigned to immediate transitions T_{VEP}, T_{VHP}, T_{VEPC}, T_{VHPC}, T_{VEPMIS} and T_{VHPMIS}, respectively. Whenever a misprediction occurs (token in place P_{VMIS}), the consumer-instruction has to be rescheduled for execution. The firing of immediate transition $T_{REEXECUTE}$ causes transportation of fluid in zero time. Fluid jumps have deterministic height of 1 (one instruction). Jumps that would go beyond the boundaries cannot be carried out. The arcs connecting fluid places and immediate transitions are drawn as thick single arrows. The fluid flow terminates at the end of the cycle when all the fluid places except P_{REG} are empty and T_{END} fires.

As far as the execution of a "class i" program is concerned, the nodes m_i of the reachability graph (Fig. 2) consist of all the tangible discrete markings, as well as those in which the enabling of immediate transitions depends on fluid levels and cannot be eliminated, since they are of mixed tangible/vanishing type (Table 1). It is important to note that the number of discrete markings does not depend on the machine width in any way.

Table 1. Discrete markings (C_{BR}=3)

Number of tokens / Marking (m_i)	#P_{FETCH}	#P_{BMIS}	#$P_{INITIATE}$	#P_{VMIS}
1_i	1	0	1	0
2_i	0	3	1	0
3_i	0	2	1	0
4_i	0	1	1	0
5_i	1	0	0	1
6_i	0	3	0	1
7_i	0	2	0	1
8_i	0	1	0	1
9_i	0	0	0	0

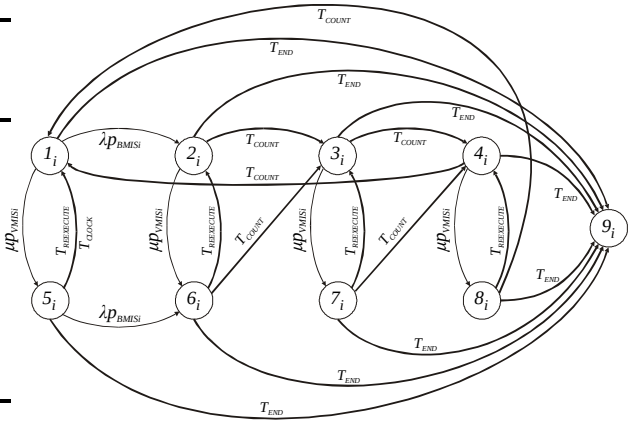

Figure 2. Reachability graph

A vector of fluid levels supplements discrete markings. It gives rise to a stochastic process in continuous time with continuous state space. Fluid levels $Z_{IB}(t)$, $Z_{RS/LSQ}(t)$, $Z_{EX}(t)$ and $Z_{REG}(t)$ are identified as four supplementary variables (components of the fluid vector $\mathbf{Z}(t)$), which provide a full description of each state. Let π_{m_i} be an abbreviation for the *volume density* $\pi_{m_i}(t, z_{IB}, z_{RS/LSQ}, z_{EX}, z_{REG})$, which is the transient probability of being in discrete marking m_i at time t, with fluid levels in an infinitesimal environment around $\mathbf{z} = [z_{IB} \quad z_{RS/LSQ} \quad z_{EX} \quad z_{REG}]$. If $\mathbf{R}_{IB}$, $\mathbf{R}_{RS/LSQ}$, $\mathbf{R}_{EX}$, $\mathbf{R}_{REG}$ are diagonal matrices of the

instantaneous rates at which fluid builds in each fluid place, $\mathbf{Q}_i$ is the matrix of transition rates of exponential transitions causing the state changes, and $\boldsymbol{\pi}_i = [\pi_{1_i} \quad \pi_{2_i} \quad ... \quad \pi_{9_i}]$, according to [103, 108], the evolution of the stochastic process underlying the Petri Net is described by a coupled system of partial differential equations in four continuous dimensions plus time:

$$\frac{\partial \boldsymbol{\pi}_i}{\partial t} + \frac{\partial(\boldsymbol{\pi}_i \cdot \mathbf{R}_{IB})}{\partial z_{IB}} + \frac{\partial(\boldsymbol{\pi}_i \cdot \mathbf{R}_{RS/LSQ})}{\partial z_{RS/LSQ}} + \frac{\partial(\boldsymbol{\pi}_i \cdot \mathbf{R}_{EX})}{\partial z_{EX}} + \frac{\partial(\boldsymbol{\pi}_i \cdot \mathbf{R}_{REG})}{\partial z_{REG}} = \boldsymbol{\pi}_i \cdot \mathbf{Q}_i$$

with appropriately chosen initial and boundary conditions – the firing of transitions T_{CLOCK}, T_{COUNT} and T_{END} at time t_0 causes switching from one discrete marking to another, while the firing of transition $T_{REEXECUTE}$ at time t can be seen as a jump to another location in the four-dimensional hypercube defined by the components of the fluid vector (fluid jumps shift probability mass along the continuous axes, in addition to discrete state change). The probabilities of the discrete markings are obtained by integrating volume densities, and the fluid levels at the beginning of each clock cycle are computed via marginal densities. The probability mass conservation law is used as a normalization condition: since no particle can pass beyond barriers, the sum of integrals of the volume densities over the definition range evaluates to one. A more detailed explanation can be found in [43, 72, 73, 74].

A number of performance measures can be evaluated:

- *The distribution of the execution time* of a program with volume V_i is:

  $$F_{t_{EX_i}}(t) = \pi_{9_i}(t,0,0,0,V_i)$$

- with *mean execution time*

  $$t_{EX_i} = \int_0^\infty \left(1 - F_{t_{EX_i}}(t)\right) dt$$

- Consequently, the *sustained number of instructions per cycle* (*IPC*) is given by:

  $$IPC_i = V_i / t_{EX_i}$$

- When the input space is partitioned, *IPC* is the ratio between the average volume and the average execution time of all the programs of different classes, as indicated by the operational profile:

  $$IPC = \frac{\sum_{k=1}^{n} V_k \widehat{w}_{T_{CLASS_k}}}{\sum_{k=1}^{n} t_{EX_k} \widehat{w}_{T_{CLASS_k}}}$$

- The sum of probabilities of the discrete markings that do not carry a token in place P_{FETCH} gives the probability of a stall in the instruction fetch unit at time t:

  $$P_{STALL_i}(t) = \sum_{\substack{m \neq 1 \\ m \neq 5}} \int_0^{Z_{IB_{MAX}}} \int_0^{Z_{RS/LSQ_{MAX}}} \int_0^{Z_{EX_{MAX}}} \int_0^{V_i} \pi_{m_i}\left(t, z_{IB}, z_{RS/LSQ}, z_{EX}, z_{REG}\right) dz_{IB}\, dz_{RS/LSQ}\, dz_{EX}\, dz_{REG}$$

- Because of the discrete nature of pipelining, additional attention should be given to the probability that no useful instructions will be added to the instruction buffer in the cycle beginning at time t_0 (complete stall in the instruction fetch unit that can

lead to an effectively empty instruction buffer) due to branch misprediction. It can be obtained by summing up the probabilities of the discrete markings that still carry one or more tokens in place P_{BMIS} immediately after firing of T_{CLOCK}:

$$P_{NO\_FETCH_i}(t_0) = \sum_{m=3}^{4} \int_0^{Z_{IB_{MAX}}} \int_0^{Z_{RS/LSQ_{MAX}}} \int_0^{Z_{EX_{MAX}}} \int_0^{V_i} \pi_{m_i}(t_0, z_{IB}, z_{RS/LSQ}, z_{EX}, z_{REG}) dz_{IB}\, dz_{RS/LSQ}\, dz_{EX}\, dz_{REG}$$

- In addition, the *execution efficiency* is introduced, taken as a ratio between the number of useful instructions and the total number of instructions executed during the course of a program's execution:

$$\eta_{EX_i} = \frac{V_i}{V_i + \begin{pmatrix} \text{instructions reexecuted} \\ \text{due to value misprediction} \end{pmatrix}} \approx \frac{V_i}{V_i + \bar{\mu} \cdot p_{VMIS_i} \cdot t_{EX_i}} = \frac{1}{1 + \dfrac{\mu_i \cdot \bar{r}_{INITIATE_i} \cdot p_{VMIS_i}}{W \cdot IPC_i}}$$

where $\bar{r}_{INITIATE}$ is the *average initiation rate*.

The attempt to capture the dynamic behavior of an ILP processor is a rare example that demonstrates the usage of the recently introduced FSPN formalism in modeling actual systems [6]. *Moreover, it is the first example to take into consideration numerical transient analysis and present numerical solution of a Fluid Stochastic Petri Net with more than three fluid places.* The one alleviating circumstance is the fact that fluid rates, although dependent on fluid levels, are piecewise constant. Finite difference approximations [23, 46] are used to replace the derivatives that appear in the PDEs: forward difference approximation for the time derivative and first-order upwind differencing for the space derivatives, in order to improve the stability of the method. The discretization is carried out on a hypercube of size $Z_{IB_{\max}} \times Z_{RS/LSQ_{\max}} \times Z_{EX_{\max}} \times V_i$ with step size Δz in direction of z_{IB}, z_{RS}, z_{EX} and z_{REG}, and step size Δt in time. The computational complexity for the solution is

$$O\left(8 \cdot \frac{t}{\Delta t} \cdot \frac{Z_{IB_{\max}} \cdot Z_{RS/LSQ_{\max}} \cdot Z_{EX_{\max}} \cdot V_i}{\Delta z^4}\right) \text{ floating-point operations,}$$

since for each of $t/\Delta t$ time steps, each solution value is incremented in the four-dimensional grid for eight of the nine discrete markings. The storage requirements of the algorithm are at least

$$8 \cdot \frac{Z_{IB_{\max}} \cdot Z_{RS/LSQ_{\max}} \cdot Z_{EX_{\max}} \cdot V}{\Delta z^4} \cdot 4 \text{ bytes,}$$

since for eight of nine discrete markings, a four-dimensional grid of floating-point numbers must be stored (solutions at successive time steps are overwritten). Because of the hardware limitations emerging from the computational complexity and the storage requirements of the numerical solution, as well as the current state of the scientific thought in dealing with this kind of problems [48, 102, 109], the correctness of the discretization method is verified by comparing the numerical transient analysis results with the results obtained by *discrete-event simulation*. In addition, discrete-event simulation alone is used to obtain performance evaluation results for wider machines with much more aggressive instruction issue (W>>1). *The numerical solution makes possible the probabilistic analysis of the dynamic behavior, whereas the advantage of the discrete-event simulation is the much faster generation of performance evaluation results, even when compared to the broadly utilized*

microarchitectural simulators. In this dissertation, the discrete-event simulation is specifically implemented for the proposed model of the dynamic behavior of an ILP processor, and not for a general FSPN. The types of events that need to be scheduled in the event queue are either transition firings or the hitting of a threshold dependent on fluid levels. Unif[0,1] pseudo-random number generator is used to generate samples from the respective cumulative distribution functions and determine transition firing times via inversion of the *cdf* ("Golden Rule for Sampling"). Despite the large number of fluid places, the numerical transient analysis and the discrete-event simulation results are satisfactorily close to each other – the higher the prediction accuracy, the better they match. Yet, many questions are still open for further research in the field of development of strategies for reducing the amount of memory needed to represent the volume densities, as well as the development of efficient discretization schemes for numerical transient analysis of general FSPNs.

Being the first of its kind, the proposed model leads to numerous conclusions regarding the performance impact of predictions and speculative execution in ILP processors where a large number of instructions are fetched, issued, executed and committed per cycle. The confirmation of previously published results [34, 67, 79, 90] represents validation of the model. Analyzing the performance potential of predictions and speculative execution under different scenarios, with varying parameters of both the microarchitecture as well as the operational environment, the following conclusions are drawn:

- Both the branch (Fig. 3) and value prediction accuracy improvements (Fig. 4) reduce the mean execution time of a program;

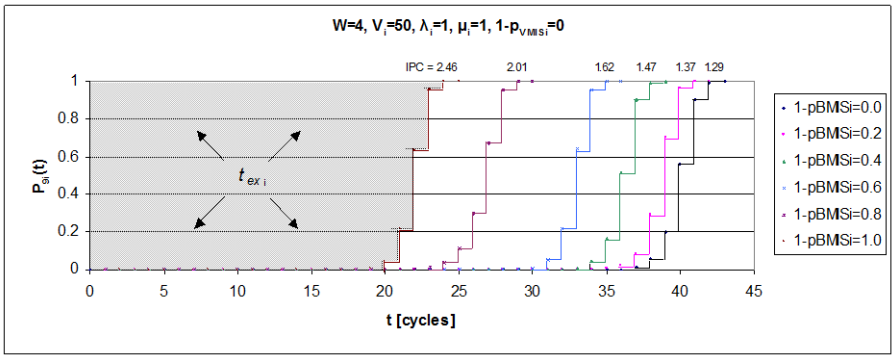

Figure 3.

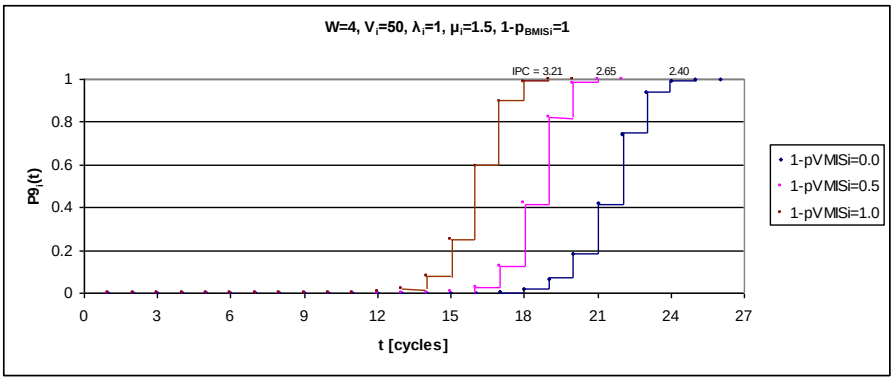

Figure 4.

- The probability of a complete stall in the instruction fetch unit, which can lead to an empty instruction buffer in the subsequent cycle, decreases with branch prediction accuracy improvement (Fig. 5);

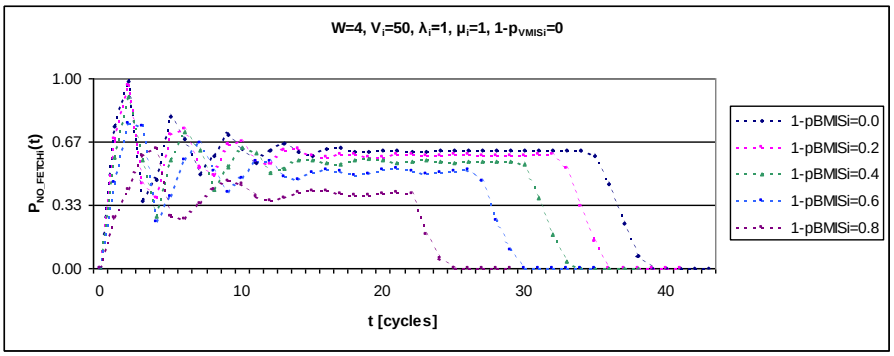

Figure 5.

- The curves that represent the speedup as a function of branch prediction accuracy have an exponential shape (Fig. 6);
- The benefits of branch prediction are higher in programs with higher branch rate, and lower consumer-instruction rate;

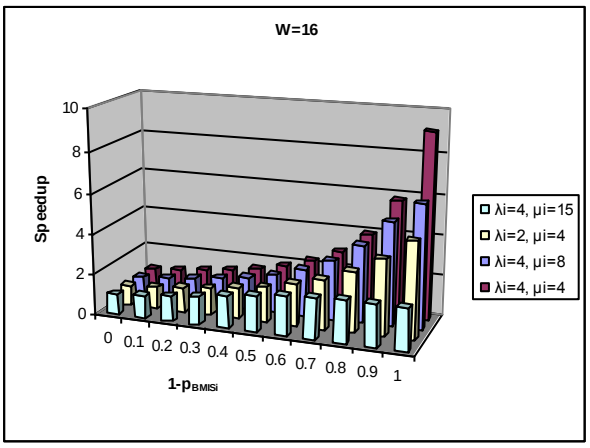

Figure 6.

- With perfect branch prediction, the speedup can be approximated by a linear function of the value prediction accuracy (Fig. 7);
- The benefits of value prediction are higher in programs with higher consumer-instruction rate and lower branch misprediction rate;

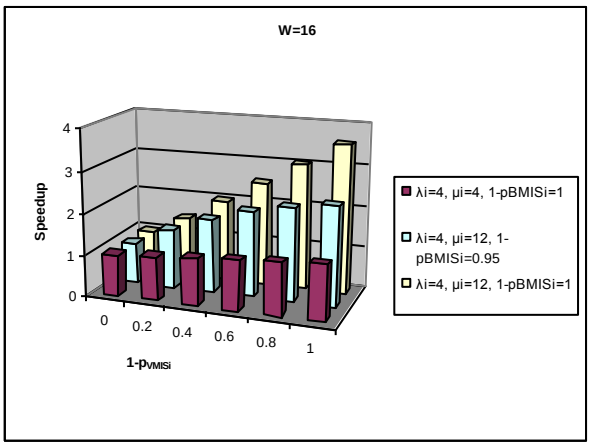

Figure 7.

- The lower the consumer-instruction rate, the higher is the execution efficiency. It increases linearly with value prediction accuracy improvements (Fig. 8);

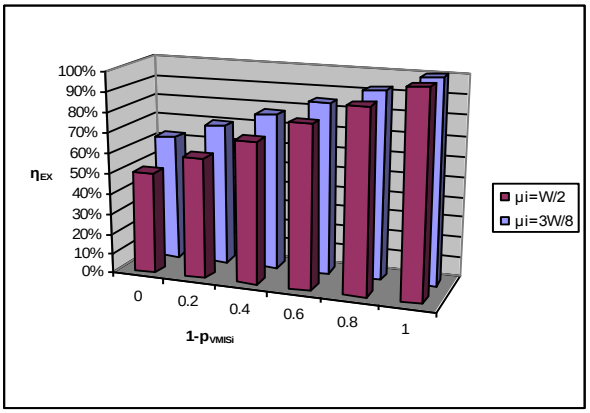

Figure 8.

- With realistic branch prediction, there is a threshold effect on the machine width: below the threshold the speedup increases with the machine width, whereas above the threshold the speedup is close to a limit (Fig. 9);
- The threshold value decreases with increasing the branch misprediction rate;

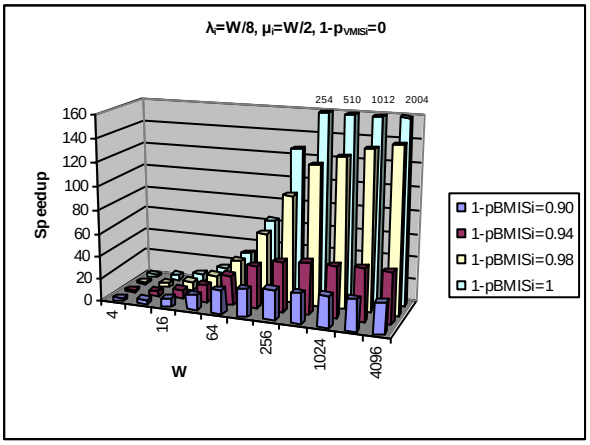

Figure 9.

- With realistic branch prediction, the maximum additional speedup that value prediction can provide diminishes when the machine width is above a threshold value (Fig. 10);
- The threshold value decreases with decreasing the consumer-instruction rate, and/or increasing the branch misprediction rate;

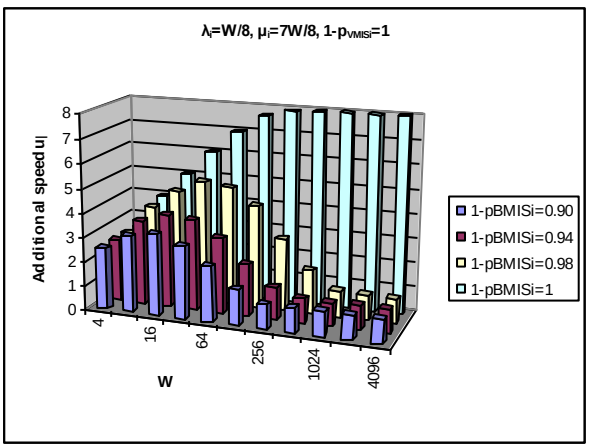

Figure 10.

- Given the width, there is a threshold effect on the instruction window size: below the threshold the speedup that value prediction can provide increases with the instruction window size, whereas above the threshold the speedup is close to a limit (Fig. 11);
- In wider machines with realistic branch prediction, value prediction might enable higher levels of parallelism without the need to increase the instruction window size;

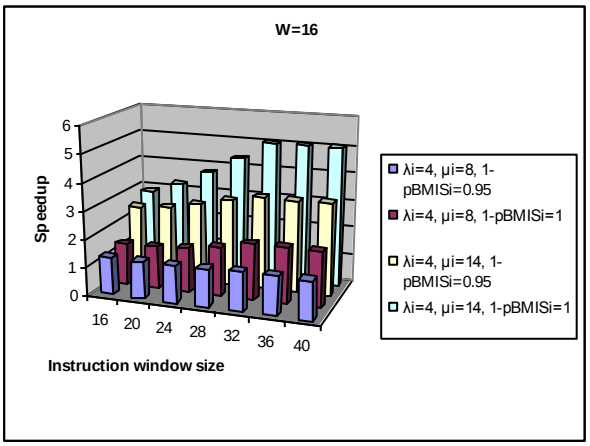

Figure 11.

- The overall machine performance decrease is more pronounced when programs with poor characteristics are executed more frequently (Fig. 12);
- The performance decrease is largest when the operational profile includes execution of programs with larger number of branches, especially hard-to-predict ones;
- The sporadic appearance of programs with either larger number of consumer-instructions, or larger number of instructions that consume hard-to-predict values, has a lower influence on overall performance.

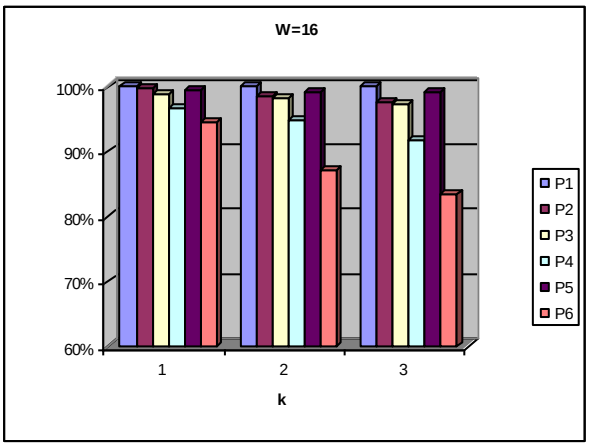

Figure 12.

In summary:

- *The benefits of branch and value prediction are higher when control and true data dependences have a much higher influence on the total execution time of a program;*

- *Value prediction is an effective approach that might enable higher levels of parallelism without the need to increase machine width and/or instruction window size;*
- *There is a correlation between the value prediction efficiency and the branch prediction efficiency;*
- *The wider the machine, the more significant performance limitation the branch mispredictions become;*
- *ILP processor's performance is strongly influenced by the characteristics of the operational environment.*

3 Thesis organization

The thesis is divided in six chapters. Some aspects of the microarchitecture, or hardware organization, of a typical ILP processor are presented in the second chapter. Then, through the review of existing prediction models and their variations in the context of speculative execution of control and true data dependent instructions, the fundamentals of branch prediction (with an emphasis on multiple branch prediction and high-bandwidth instruction fetch mechanisms), as well as value prediction, are presented.

The third chapter reviews related work – studies that assess the efficiency of predictions and speculative execution in boosting the performance of ILP processors under different scenarios (microarchitectural features), most of which rely upon the use of microarchitectural simulators, while only a few bring into play analytical modeling. Looking at them from a critical point of view, the motives to introduce a new approach are identified.

The main contributions of the dissertation are given in the fourth and the fifth chapter. At the start, a stochastic model of the dynamic behavior of an ILP processor – first of its kind, built using Fluid Stochastic Petri Nets, is introduced. Then (i) the stochastic process underlying the FSPN is described, (ii) state equations and a number of performance measures are derived, and (iii) limitations of the numerical transient analysis and possibilities of overcoming them, including discrete-event simulation of the model, are discussed.

Numerical transient analysis and/or discrete-event simulation results are presented in the fifth chapter. The analysis leads to numerous conclusions regarding the performance impact of predictions and speculative execution with varying parameters of both the microarchitecture and the operational environment.

At the very end, in the sixth chapter, the thesis concludes with a summary and a discussion of future work.

4 References

[1] Austin, T.M., Sohi, G.S., "Dynamic Dependency Analysis of Ordinary Programs", *Proc. of the 19th International Symposium on Computer Architecture*, pp. 342-351, Queensland, Australia, 1992

[2] Baiocchi, A., Blefari-Melazzi, N., "An Error-Controlled Approximate Analysis of a Stochastic Fluid Flow Model Applied to an ATM Mutiplexer with Heterogenous On-Off Sources", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 6, pp. 628-637, 1993

[3] Bechem, C., Combs, J., Utamaphethai, N., Black, B., Blanton, S., Shen, J.P., "An Integrated Functional Performance Simulator", *IEEE Micro*, pp. 26-34, May-June 1999

[4] Black, B., Huang, A., Lipasti, M., Shen, J.P., "Can Trace-Driven Simulators Accurately Predict Superscalar Performance?", *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 478-485, San Antonio, USA, 1996

[5] Black, B., Rychlik, B., Shen, J.P., "The Block-based Trace Cache", *Proceedings of the 26th International Symposium on Computer Architecture*, pp. 196-207, Atlanta, USA, 1999

[6] Bobbio, A., Garg, S., Gribaudo, M., Horvath, A., Sereno, M., Telek, M., "Modeling Software Systems with Rejuvenation, Restoration and Checkpointing through Fluid Stochastic Petri Nets", *Proc. of the 8th International Workshop on Petri Nets and Performance Models (PNPM'99)*, pp. 82-91, Zaragoza, Spain, 1999

[7] Bodik, R., Gupta, R., Soffa, M.L., "Interprocedural Conditional Branch Elimination", *Proc. of the ACM SIGPLAN'97 Conference on Programming Language Design and Implementation*, pp. 146-158, Las Vegas, USA, 1997

[8] Burger, D., Austin, T.M., *"The SimpleScalar Tool Set, Version 2.0"*, University of Wisconsin – Madison, Computer Sciences Department, Technical Report #1342, 1997

[9] Burtscher, M., "*Improving Context-Based Load Value Prediction"*, PhD Thesis, University of Colorado, 2000

[10] Butler, M., Yeh, T.Y., Patt, Y., Alsup, M., Scales, H., Shebanow, M., "Single Instruction Stream Parallelism is Greater than Two", *Proc. of the 18th International Symposium on Computer Architecture*, pp. 276-286, Toronto, Canada, 1991

[11] Calder, B., Reinman, G., Tullsen, D., "Selective Value Prediction", *Proc. of the 26th Annual International Symposium on Computer Architecture*, pp. 64-73, Atlanta, USA, 1999

[12] Chang, P.Y., Evers, M., Patt, Y., "Improving Branch Prediction Accuracy by Reducing Pattern History Table Interference", *International Journal of Parallel Programming*, Vol. 25, No. 5, pp. 339-362, 1997

[13] Chang, P.Y., Hao, E., Patt, Y., "Alternative Implementations of Hybrid Branch Predictors", *Proc. of the 28th Annual International Symposium on Microarchitecture*, pp. 252-263, Ann Arbor, USA, 1995

[14] Chang, P.Y., Hao, E., Yeh, T.Y., Patt, Y., "Branch Classificaion: a New Mechanism for Improving Branch Predictor Performance", *Proc. of the 27th Annual International Symposium on Microarchitecture*, pp. 22-31, San Jose, USA, 1994

[15] Ciardo, G., German, R., Lindemann, C., "A Characterisation of the Stochastic Process Underlying a Stochastic Petri Net", *IEEE Transactions on Software Engineering*, Vol. 20, No. 7, pp. 506-515, 1994

[16] Ciardo, G., Nicol, D., Trivedi, K., "Discrete-Event Simulation of Fluid Stochastic Petri Nets", *Proc. of the 7th international Workshop on Petri Nets and performance Models (PNPM'97)*, pp. 217-225, Saint Malo, France, 1997

[17] Conte, T., Menezes, T., Mills, P., Patel, B., "Optimization of Instruction Fetch Mechanisms for High Issue Rates", *Proc. of the 22nd International Symposium on Computer Architecture*, pp. 333-344, Margherita Ligure, Italy, 1995

[18] Diep, T.A., Shen, J.P., Phillip, M., "EXPLORER: A Retargetable and Visualisation-Based Trace-Driven Simulator for Superscalar Processors", *Proc. of the 26th International Symposium on Microarchitecture*, pp. 225-235, Austin, USA, 1993

[19] Diep, T.A., Shen, J.P., "VMW: Visualisation Based Microarchitecture Workbench", *IEEE Computer*, pp. 57-64, December 1995

[20] Dutta, S., Franklin, M., "Control Flow Prediction with Tree-Like Subgraphs for Superscalar Processors", *Proc. of the 28th International Symposium on Microarchitecture*, pp. 258-263, Ann Arbor, USA, 1995

[21] Eden, A.N., Mudge, T., "The YAGS Branch Prediction Scheme", *Proc. of the 31st Annual International on Microarchitecture,* pp. 69-77, Dallas, USA, 1998

[22] Evers, M., "Improving Branch Prediction by Understanding Branch Behavior", PhD Thesis, University of Michigan, USA, 2000

[23] Ferziger, J.H., Perić, M., *"Computational Methods for Fluid Dynamics"*, Springer-Verlag, 1997

[24] Friendly, D.H., Patel, S.J., Patt, Y., "Alternative Fetch and Issue Policies for the Trace Cache Fetch Mechanism", *Proc. of the 30th Annual International Symposium on Microarchitecture*, pp. 24-33, Research Triangle Pk, USA, 1997

[25] Gabbay, F., *"Speculative Execution based on Value Prediction"*, EE Department Technical Report #1080, Technion – Israel Institute of Technology, Haifa, Israel, 1996

[26] Gabbay, F., Mendelson, A., "Can Program Profiling Support Value Prediction", *Proc. of the 30th Annual International Symposium on Microarchitecture*, pp. 270-280, Research Triangle Pk, USA, 1997

[27] Gabbay, F., Mendelson, A., "Improving Achievable ILP through Value Prediction and Program Profiling", *Microprocessors and Microsystems*, Vol. 22, No. 6, pp. 315-332, Elsevier Science, 1998

[28] Gabbay, F., Mendelson, A., "The Effect of Instruction Fetch Bandwidth on Value Prediction", *Proceedings of the 25th International Symposium on Computer Architecture*, pp. 272-281, Barcelona, Spain, 1998

[29] Gabbay, F., Mendelson, A., "Using Value Prediction to Increase the Power of Speculative Execution Hardware", *ACM Transactions on Computer Systems*, Vol. 16, No. 3, pp. 234-270, 1998

[30] Gonzalez, J., Gonzalez, A., "Data Value Speculation in Superscalar Processors", *Microprocessors and Microsystems*, Vol. 22, No. 6, pp. 293-301, Elsevier Science, 1998

[31] Gonzalez, J., Gonzalez, A., "Limits of Instruciton Level Parallelism with Data Value Speculation", *Proc. of VECPAR'98*, pp. 452-465, Porto, Portugal, 1998

[32] Gonzalez, J., Gonzalez, A., "Memory Address Prediction for Data Speculation", *Proc. of EUROPAR '97*, pp. 1084-1091, Passau, Germany, 1997

[33] Gonzalez, J., Gonzalez, A., "Speculative Execution via Address Prediction and Data Prefetching", *Proc. of the ACM International Conference on Supercomputing*, pp.196-203, Vienna, Austria, 1997

[34] Gonzalez,J., Gonzalez,A., "The Potential of Data Value Speculation to Boost ILP", *Proc. of the 12th ACM International Conference on Supercomputing*, pp. 21-28, Melbourne, Australia, 1998

[35] Goševa-Popstojanova, K., Grnarov, A., "Performability Modeling of N-Version Programming Technique", *Proc. of the 6th International Symposium on Software Reliability Engineering*, pp. 209-218, Toulouse, France, 1995

[36] Gribaudo, M., Horvath, A., *"Fluid Stochastic Petri Nets Augmented with Flush-out Arcs: A Transient Analysis Technique"*, Tehnical Report, Dipartimento di Informatica, Universita di Torino, 2001

[37] Gribaudo, M., Sereno, M., "Simulation of Fluid Stochastic Petri Nets", Proc. of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 231-239, San Francisco, USA, 2000

[38] Gribaudo, M., Sereno, M., Bobbio, A., "Fluid Stochastic Petri Nets: An extended Formalism to Include non-Markovian Models", *Proc. of the 8th International Workshop on Petri Nets and Performance Models*, pp. 74-81b, Zaragoza, Spain, 1999

[39] Gribaudo, M., Sereno, M., Horvath, A., Bobbio, A., "Fluid Stochastic Petri Nets Augmented with Flush-out Arcs: Modeling and Analysis", *Discrete Event Dynamic Systems*, 11(1/2), pp. 97-117, Kluwer Academic Publishers, 2001

[40] Gušev, M., Popovski, G., Mišev, A., "Simulation of Superscalar Processor", *Proc. of the 20th International Conference on Information Technology Interfaces*, pp. 169-174, Pula, Croatia, 1998

[41] Gušev, M, Mišev. A, Popovski. G., "Memory Address Dependencies", *Proc. of the 21st International Conference on Information Technology Interfaces*, pp.191-196, Pula, Croatia, 1999

[42] **Gušev, M., Mišev, A., Popovski., G., Mitrevski, P., "Reducing the Number of Instructions", *Proc. of the 22nd International Conference on Information Technology Interfaces*, pp. 55-60, Pula, Croatia, 2000**

[43] **Gušev, M., Mitrevski, P., "Modeling and Performance Evaluation of Branch and Value Prediction in ILP Processors", to appear in *International Journal of Computer Mathematics*, Volume 79, Gordon and Breach Publishers, 2002**

[44] Haungs, M., Salee, P., Farrens, M., "Branch Transition Rate: A New Metric for Improved Branch Classification Analysis", *Proceedings of the 6th International Symposium on High-Performance Computer Architecture*, pp. 241-250, Toulouse, France, 2000

[45] Hennessy, J.L., Patterson, D.A., *"Computer Architecture: A Quantitative Approach"*, Second Edition, Morgan Kaufmann Publishers, San Francisco, USA, 1996

[46] Hoffmann, K.A., Chiang, S.T., *"Computational Fluid Dynamics for Engineers"*, Volume I & II, Engineering Education System, 1993

[47] Horton, G., Kulkarni, V., Nicol, D., Trivedi, K., "Fluid Stochastic Petri Nets: Theory, Applications, and Solution", *European Journal of Operations Research*, 105(1), pp. 184-201, 1998

[48] Horvath, A. (University of Torino), *personal communication*, August 24, 2000

[49] Huang, J., Choi, Y., Lilja, D., *"Improving Value Prediction by Exploiting Both Operand and Output Value Locality"*, Technical Report ARCTiC-99-06, University of Minnesota, Minneapolis, USA, 1999

[50] Jacobson, Q., Rotenberg, E., Smith, J.E., "Path-Based Next Trace Prediction", *Proc. of the 30th Annual International Symposium on Microarchitecture*, pp. 14-23, Research Triangle Pk, USA, 1997

[51] Juan, T., Sanjeevan, S., Navarro, J., "Dynamic History-Length Fitting: A third level of adaptivity for branch prediction", *Proc. of the 25th Annual International Symposium on Computer Architecture*, pp. 155-166, Barcelona, Spain, 1998

[52] Lam, M.S., Wilson, R.P., "Limits of Control Flow on Parallelism", *Proc. of the 19th International Symposium on Computer Architecture*, pp. 46-57, Queensland, Australia, 1992

[53] Lee, C-C., Chen, I-C.K., Mudge, T., "The Bi-mode Branch Predictor", *Proc. of the 30th Annual International Symposium on Microarchitecture*, pp. 4-13, Research Triangle Pk, USA, 1997

[54] Lee, J., Moon, S., Sung, W., "An enhanced two-level adaptive multiple branch prediction for superscalar processors", *Journal of Systems Architecture*, 45, pp. 591-602, Elsevier Science, 1999

[55] Lee, J.K.F., Smith, A.J., "Branch Prediction Strategies and Branch Target Buffer Design", *IEEE Computer*, pp. 6-22, January 1984

[56] Lee, S., Yew, P., "On Some Implementation Issues for Value prediction on Wide-Issue ILP Processors", *Proc. of the International Conference on Parallel Architectures and Compilation Techniques*, pp.145-156, Philadelphia, USA, 2000

[57] Lee, S., Yew, P., "On Table Bandwidth and Its Update Delay for Value Prediction on Wide-Issue ILP Processors", *IEEE Transactions on Computers*, Vol. 50, No. 8, pp. 847-852, August, 2001

[58] Lepak, K., Lipasti, M., "On the Value Locality of Store Instructions", *Proc. of the 27th Annual International Suymposium on Computer Architecture*, pp. 182-191, Vancouver, Canada, 2000

[59] Lipasti, M., *"Value Locality and Speculative Execution"*, PhD Thesis, Technical Report CMU-CSC-97-4, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, USA, 1997

[60] Lipasti, M., Shen, J.P., *"Approaching 10 IPC via Superspeculation"*, Technical Report CMU-CSC-97-1, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, USA, 1997

[61] Lipasti, M., Shen, J.P., "Superspeculative Microarchitecture for Beyond AD 2000", *IEEE Computer*, Vol. 30, No. 9, pp. 59-66, 1997

[62] Lipasti, M., Shen, J.P., "The Performance Potential of Value and Dependence Prediction", *Proceedings of EUROPAR-97*, pp. 1043-1052, Passau, Germany, 1997

[63] Lipasti,M., Shen,J.P., "Exceeding the dataflow limit via value prediction", *Proc. of the 29th Annual International Symposium on Microarchitecture*, pp. 226-237, Paris, France, 1996

[64] Lipasti,M., Wilkerson,C., Shen,J.P., "Value Locality and Load Value Prediction", *Proc. of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 138-147, Cambridge, USA, 1996

[65] McFarling, S., *"Combining Branch Predictors"*, Technical Report TN-36, Digital Equipment Corporation, Western Research Lab, 1993

[66] Menezes, K.N., Sathaye, S.W., Conte, T.M., "Path Prediction for High Issue-Rate Processors", *Proc. of the 3rd International Conference on Parallel Architectures and Compilation Techniques*, pp. 178-188, San Francisco, USA, 1997

[67] Michaud, P., Seznec, A., Jourdan, S., "Exploring Instruction-Fetch Bandwidth Requirement in Wide-Issue Superscalar Processors", *Proc. of the International Conference on Parallel Architectures and Compilation Techniques*, pp. 2-10, Newport Beach, USA, 1999

[68] Michaud, M., Seznec, A., Uhlig, R., "Trading Conflict and Capacity Aliasing in Conditional Branch Predictors", *Proc. of the 24th International Symposium on Computer Architecture*, pp. 292-303, Denver, USA, 1997

[69] Milton, J.S., Arnold, J.C., "Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences", Second Edition, McGraw-Hill, 1990

[70] **Mitrevski, P., Goševa-Popstojanova, K., Grnarov, A., "Reliability and Performability Modeling of N-Version Fault-Tolerant Software in Real-Time Environment using Markov Regenerative Stochastic Petri Nets", *Proc. of the 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2000)* , Vol. VII, pp. 463-468, Orlando, USA, 2000**

[71] **Mitrevski, P., Gušev, M., Mišev, A., "Prediction and Speculation Techniques in ILP", *Proc. of the 22nd International Conference on Information Technology Interfaces (ITI 2000)*, pp. 67-72, Pula, Croatia, 2000**

[72] **Mitrevski, P., Gušev, M., "Analytical Modeling Approach for the Performance Evaluation of Prediction Techniques in ILP Processors", *Proc. of the 36th International Scientific Conference on Energy and Information Systems and Technologies (EIST 2001)*, Vol. III, pp. 672-679, Bitola, Macedonia, 2001**

[73] **Mitrevski, P., Gušev, M., "Modeling the Dynamic Behavior of an ILP Processor", *Proc. of the 23rd International Conference on Information Technology Interfaces (ITI 2001)*, pp. 69-74, Pula, Croatia, 2001**

[74] **Mitrevski, P., Gušev, M., "Performance Evaluation of Branch and Value Prediction using Discrete-Event Simulation of FSPNs", to appear in *Proc. of the 2nd CiiT Conference of Informatics and Information Technology*, Molika-Bitola, Macedonia, 2001**

[75] Mueller, F., Whalley, D.B., "Avoiding Conditional Branches by Code Replication", *Proc. of the ACM SIGPLAN '95 Conference on Programming Language Design and Implementation*, pp. 56-66, 1995

[76] Musa, J.D., "Operational Profiles in Software Reliability Engineering", *IEEE Software*, pp. 14-32, March 1993

[77] Nair, R., "Dynamic Path-Based Branch Correlation", *Proc. of the 28th Annual International Symposium on Microarchitecture*, pp. 15-23, Ann Arbor, USA, 1995

[78] Nakra, T., Gupta, R., Soffa, M.L., "Global Context-Based Value Prediction", *Proc. of the 5th International Symposium on High Performance Computer Architecture*, pp. 4-12, Orlando, USA, 1999

[79] Neefs, H., De Bosschere, K., Van Kampenhout, J., "Exploitable levels of ILP in future processors", *Journal of Systems Architecture*, 45, pp. 687-708, Elsevier Science, 1999

[80] Noonburg, D.B., Shen, J.P., "A Framework for Statistical Modeling of Superscalar Processor Performance", *Proc. of the 3rd International Symposium on High Performance Computer Architecture*, pp. 298-309, San Antonio, USA, 1997

[81] Perleberg, C.H., Smith, A.J., "Branch Target Buffer Design and Optimisation", *IEEE Transactions on Computers*, 42(4), pp. 396-412, 1993

[82] Peterson, J., "Petri Net Theory and the Modeling of Systems", Prentice-Hall, 1981

[83] Rajan, R., *"General Fluid Models for Queuing Networks"*, PhD Thesis, University of Wisconsin – Madison, USA, 1995

[84] Rakvic, R., Black, B., Shen, J.P., "Completion Time Multiple Branch Prediction for Enhancing Trace Cache Performance", *Proc. of the 27th Annual International Suymposium on Computer Architecture*, pp. 47-58, Vancouver, Canada, 2000

[85] Reinman, G., Calder, B., "Predictive Techniques for Aggressive Load Speculation", *Proc. of the 31st Annual International Symposium on Microarchitecture*, pp. 127-137, Dallas, USA, 1998

[86] Rotenberg, E., Bennett, S., Smith, J., "Trace Cache: a Low Latency Approach to High Bandwidt Instruction Fetching", *Proc. of the 29th Annual International Symposium on Microarchitecture*, pp. 24-35, Paris, France, 1996

[87] Rychlik, B., Faistl, J., Krug, B., Kurland, A., Sung, J., Velev, M., Shen, J.P., *"Efficient and Accurate Value Prediciton Using Dynamic Classification"*, Technical Report CMµART-1998-01, Carnegie Mellon University, Pittsburgh, USA, 1998

[88] Rychlik, B., Faistl, J., Krug, B., Shen, J.P., "Efficacy and Performance Impact of Value Prediciton", *Proc. of the International Confrernce on Parallel Architectures and Compilation Techniques*, pp. 148-154, Paris, France, 1998

[89] Rychlik, B., Lipasti, M., Shen, J.P., *"Experimental Characterization of Value Locality"*, Technical Report CMµART-1997-04, Carnegie Mellon University, Pittsburgh, USA, 1997

[90] Sazeides, Y., "An Analysis of Value Predictability and its Application to a Superscalar Processor", PhD Thesis, University of Wisconsin-Madison, 1999

[91] Sazeides, Y., Smith, J.E., *"Implementations of Context Based Value Predictors"*, Technical Report ECE-97-8, University of Wisconsin – Madison, USA, 1997

[92] Sazeides, Y., Smith, J.E., "Limits of Data Value Predictability", *International Journal of Parallel Programming*, 27(4), pp. 229-256, Kluwer Academic Publishers, 1999

[93] Sazeides, Y., Smith, J.E., "The Predictability of Data Values", *Proc. of the 30th Annual International Symposium on Microarchitecture*, pp. 248-258, Research Triangle Pk, USA, 1997

[94] Sazeides, Y., Vassiliadis, S., Smith, J.E., "The Performance Potential of Data Dependence Speculation & Collapsing", *Proc. of the 29th Annual International Symposium on Microarchitecture*, pp. 238-247, Paris, France, 1996

[95] Seznec, A., Jourdan, S., Sainrat, P., Michaud, P., "Multiple-Block Ahead Branch Predictors" *Proc. of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 116-127, Cambridge, USA, 1996

[96] Sherwood, T., Calder, B., *"Time Varying Behavior of Programs"*, UC San Diego Technical Report UCSD-CS99-630, August 1999

[97] Sherwood, T., Perelman, E., Calder, B., "Basic Block Distribution Analysis to Find Periodic Behavior and Simulation Points in Applications", *Proc. of the International Conference on Parallel Architectures and Compilation Techniques*, pp. 3-14, Barcelona, Spain, 2001

[98] Smith, J., "A Study of Branch Prediciton Strategies", *Proc. of the 8th Annual International Symposium on Computer Architecture*, pp. 135-148, 1981

[99] Smith, J.E., Sohi, G.S., "The Microarchitecture of Superscalar Processors", *Proc. of the IEEE*, Vol. 83, No. 12, pp. 1609-1624, 1995

[100] Smith, M.D., Johnson, M., Horowitz, M.A., "Limits on Multiple Instruction Issue", *Proc. of the 3rd International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 290-302, Boston, USA, 1989

[101] Sprangle, E., Chappell, R.S., Alsup, M., Patt, Y.N., "The Agree Predictor: a Mechanism for Reducing Negative Branch History Interference", *Proc. of the 24th International Symposium on Computer Architecture*, pp. 284-291, Denver, USA, 1997

[102] Telek, M. (TU Budapest), *personal communication*, August 18, 2000

[103] Trivedi, K., Kulkarni, V., "FSPNs: Fluid Stochastic Petri Nets", Lecture Notes in Computer Science, Vol. 691, M.Ajmone Marsan (ed.), *Proc. of the 14th International Conference on Applications and Theory of Petri Nets*, pp. 24-31, Heidelberg, Germany, 1993

[104] Wall, D.W., "Limits of Instruction-Level Parallelism", Proc. of the 4th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 176-188, Santa Clara, USA, 1991

[105] Wallace, S., Bagherzadeh, N., "Multiple Branch and Block Prediction", *Proc. of the 3rd International Symposium on High Performance Computer Architecture*, pp. 94-105, San Antonio, USA, 1997

[106] Wang, K., Franklin, M., "Highly Accurate Data Value Prediction using Hybrid Predictors", *Proc. of the 30th Annual International Symposium on Microarchitecture*, pp. 281-290, Research Triangle Pk, USA, 1997

[107] Wolter, K., "Performance and Dependability Modelling with Second Order Fluid Stochastic Petri Nets", PhD Thesis, TU Berlin, Germany, 1999

[108] Wolter, K., Horton, G., German, R., *"Non-Markovian Fluid Stochastic Petri Nets"*, Technical Report 1996-13, TU Berlin, Germany, 1996

[109] Wolter, K. (TU Berlin), *personal communication*, October 25, 2000

[110] Yeh, T.Y, Patt, Y.N., "Two-Level Adaptive Branch Prediction", *Proc. of the 24th Annual International Symposium on Microarchitecture*, pp. 51-61, Albuquerque, USA, 1991

[111] Yeh, T.Y, Patt, Y.N., "Alternative Implementations of Two-Level Adaptive Branch Prediction", *Proc. of the 19th Annual International Symposium on Computer Architecture*, pp. 124-134, Gold Coast, Australia, 1992

[112] Yeh, T.Y, Patt, Y.N., "A Comparison of Dynamic Branch Predictors that Use Two Levels of Branch History", *Proc. of the 20th Annual International Symposium on Computer Architecture*, pp 257-266, San Diego, USA, 1993

[113] Yeh, T.Y., "Two-Level Adaptive Branch Prediction and Instruction Fetch Mechanisms for High Performance Superscalar Processors", PhD Thesis, University of Michigan, USA, 1993

[114] Yeh, T.Y., Marr, D., Patt, Y., "Increasing the Instruction Fetch Rate via Multiple Branch Prediction and a Branch Address Cache", *Proc. of the International Conference on Supercomputing*, pp. 67-76, Tokyo, Japan, 1993