Augsburg University, February 18th 2010

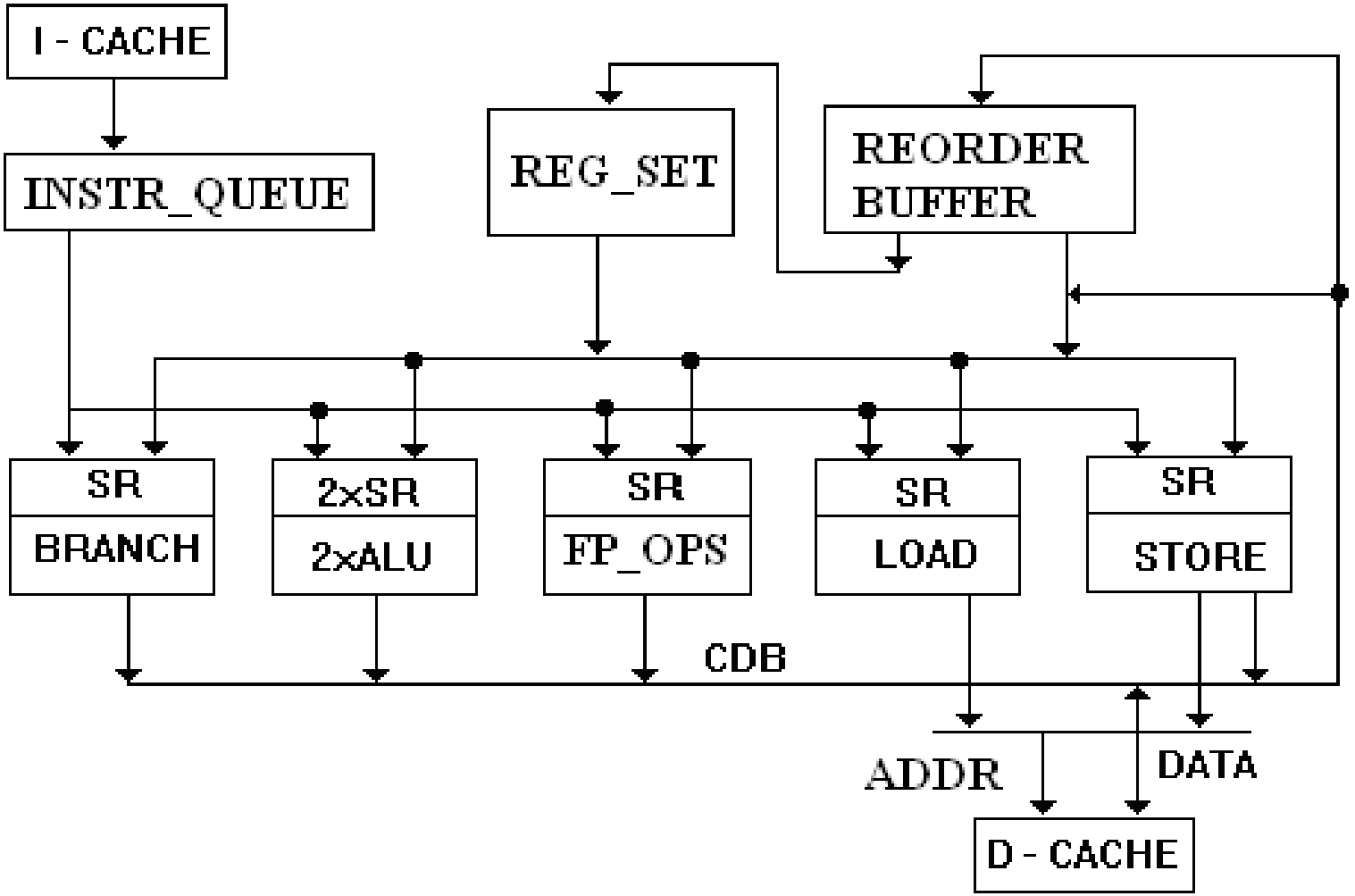
Anticipatory Techniques in Advanced Processor Architectures

Professor Lucian N. VINŢAN, PhD

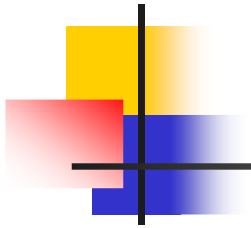
- "Lucian Blaga" University of Sibiu (RO), Computer Engineering Department,
Advanced Computer Architecture & Processing Systems Lab:
http://acaps.ulbsibiu.ro
- Academy of Technical Sciences from Romania: www.astr.ro

E-mail: lucian.vintan@ulbsibiu.ro

ILP Paradigm. A Typical Superscalar Microarchitecture



PHASES: IFetch→IDecode→Dispatch→Issue→ALU→Mem→Wr_Back→Commit



Instructions' Pipeline Processing. Branch Prediction

IFetch→IDecode→Dispatch→Issue→ALU→Mem→WB→Commit / **BR <cond>, Addr**

IFetch→IDecode→Dispatch→Issue→ALU→Mem→WBack→Commit

IFetch→IDecode→Dispatch→Issue→ALU→Mem→WBack→…

IFetch→IDecode→Dispatch→Issue→ALU→Mem→…

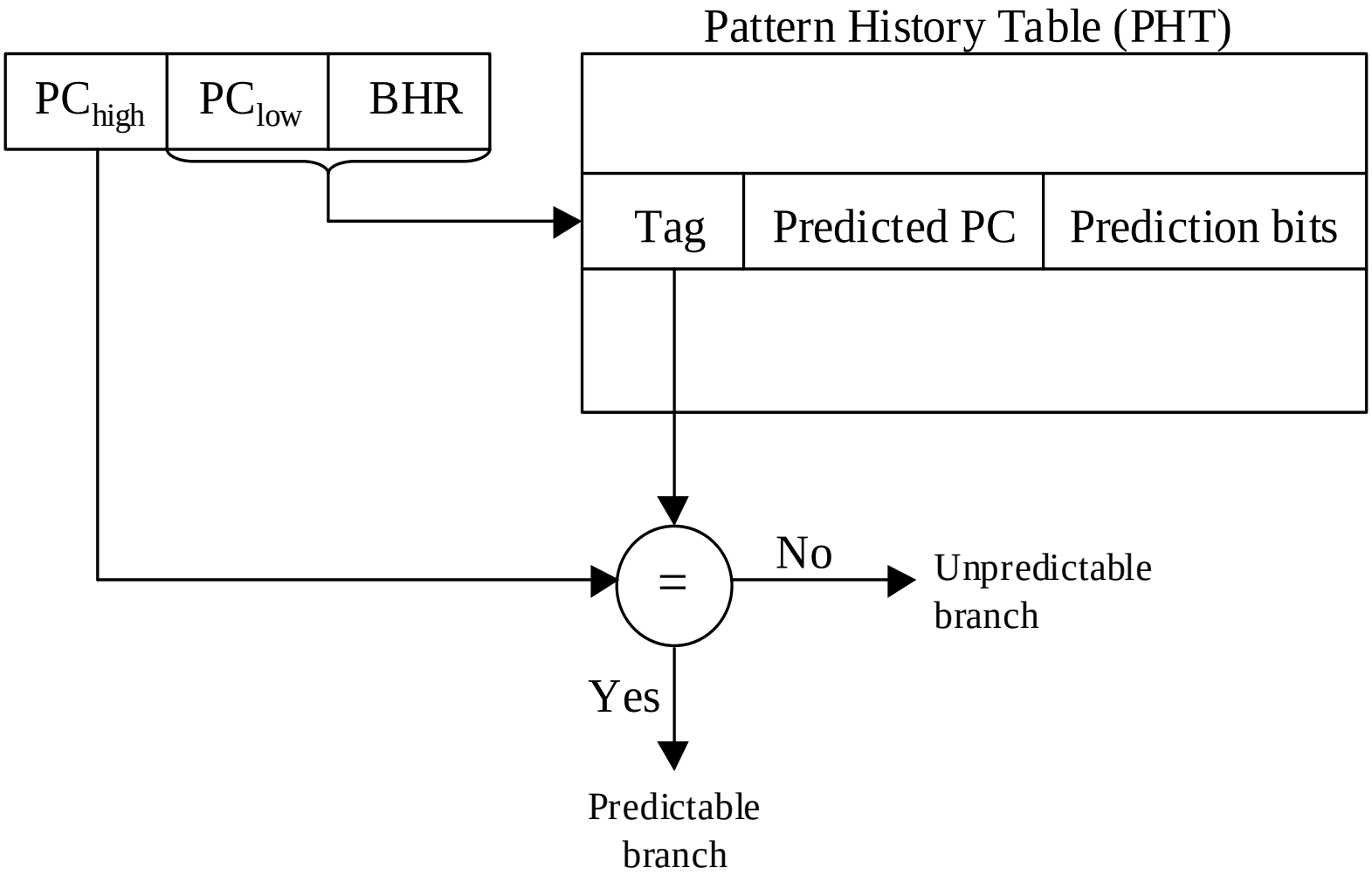
.
.
.

BRANCH PREDICTION NECESSITY INCREASES WITH:

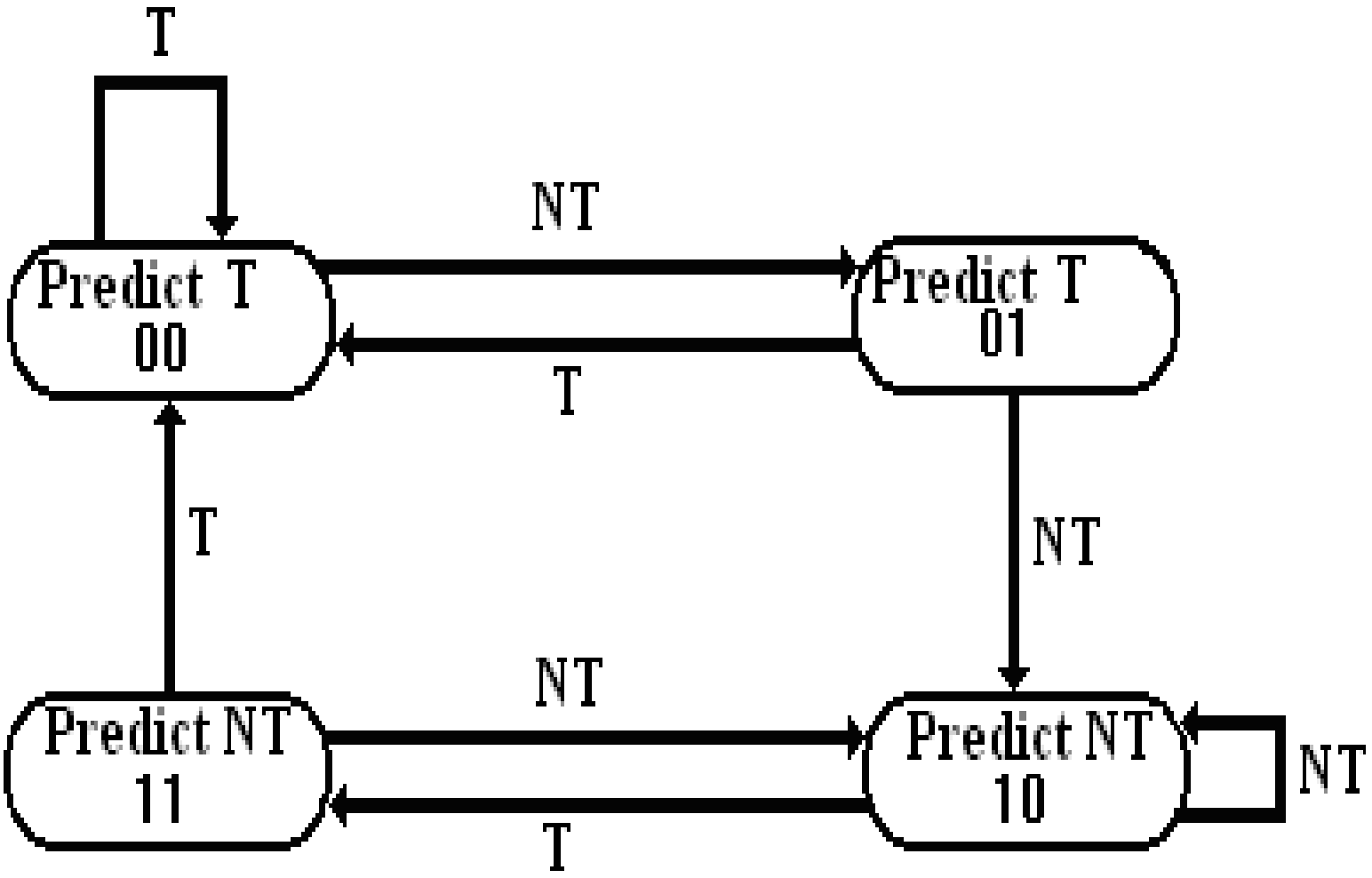
- Pipeline Depth
- Superscalar Factor (maximum achievable ILP)

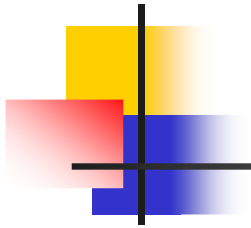
Branch Prediction significantly increases performance

A Dynamic Adaptive Branch Predictor



A typical per Branch FSM Predictor (2 Prediction Bits)





Fundamental Limits of ILP Paradigm. Solutions

FETCH BOTTLENECK

- **Fetch Rate is limited** by the **basic-blocks' dimension** (7-8 instructions in SPEC 2000);
- Fetch Bottleneck is due to the programs' intrinsic characteristics.

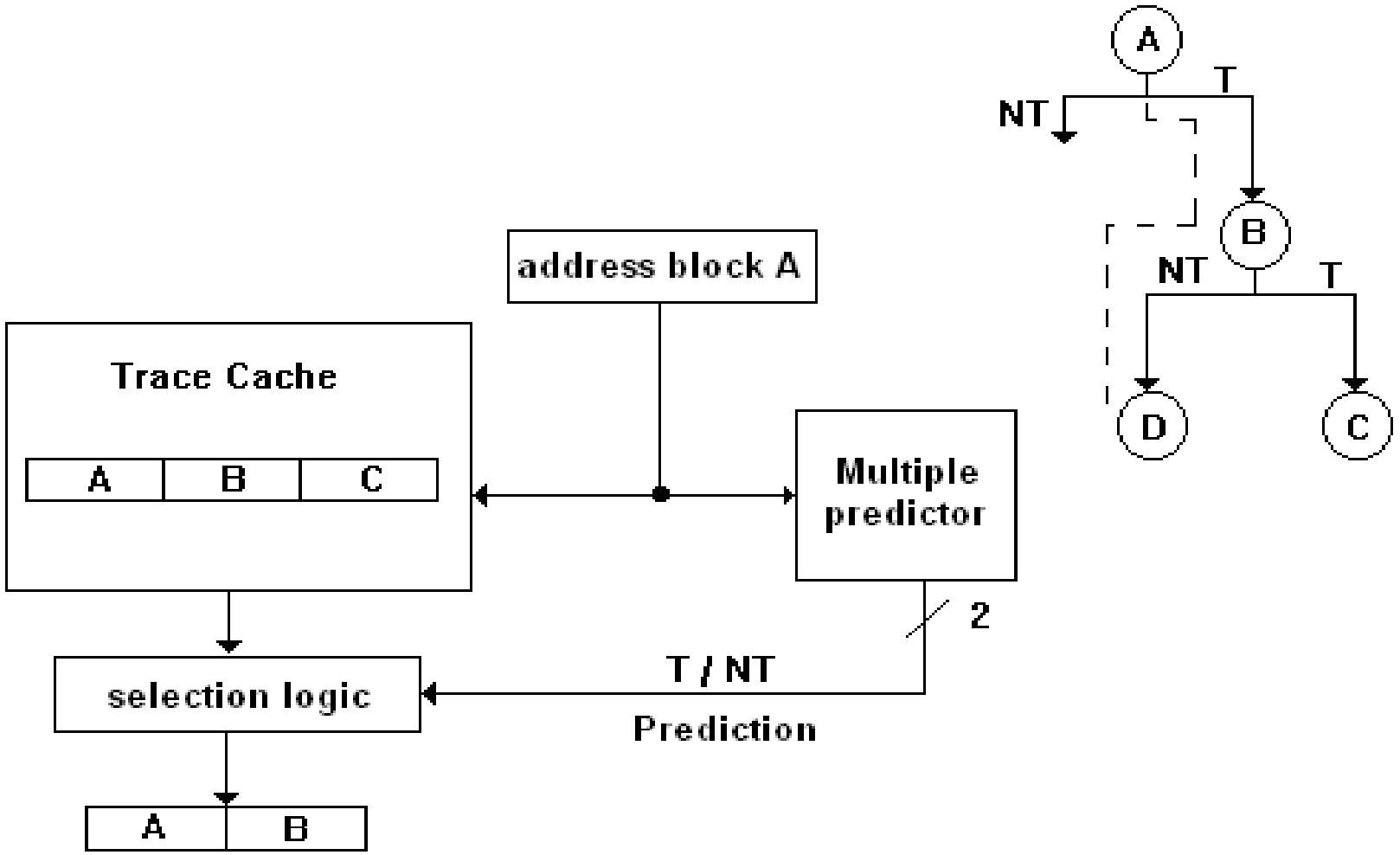
Solutions

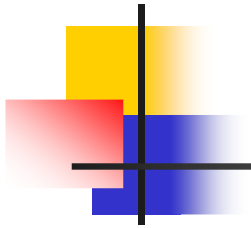
- **Trace-Cache & Multiple (M-1) Branch Predictors;**
- A TC entry contains **N instructions or M basic-blocks (N>M) written at the time they were executed**;
- Branch Prediction **increases ILP** by predicting branch **directions and targets** and speculatively **processing multiple basic-blocks** in parallel;
- As **instruction issue width** and the **pipeline depth** are getting higher, **accurate branch prediction** becomes more essential.

Some Challenges

- Identifying and solving some **Difficult-to-Predict Branches** (*unbiased branches*);
- Helping the computer architect to **better understand branches' predictability** and also if the predictor should be improved related to Difficult-to-Predict Branches.

Trace-Cache with a multiple-branch predictor





Fundamental Limits of ILP Paradigm. Solutions

ISSUE BOTTLENECK (DATA-FLOW)

- Conventional processing models are limited in their processing speed by the dynamic **program's critical path** (Amdahl)*;*
- It is due to the intrinsic sequentially of the programs.

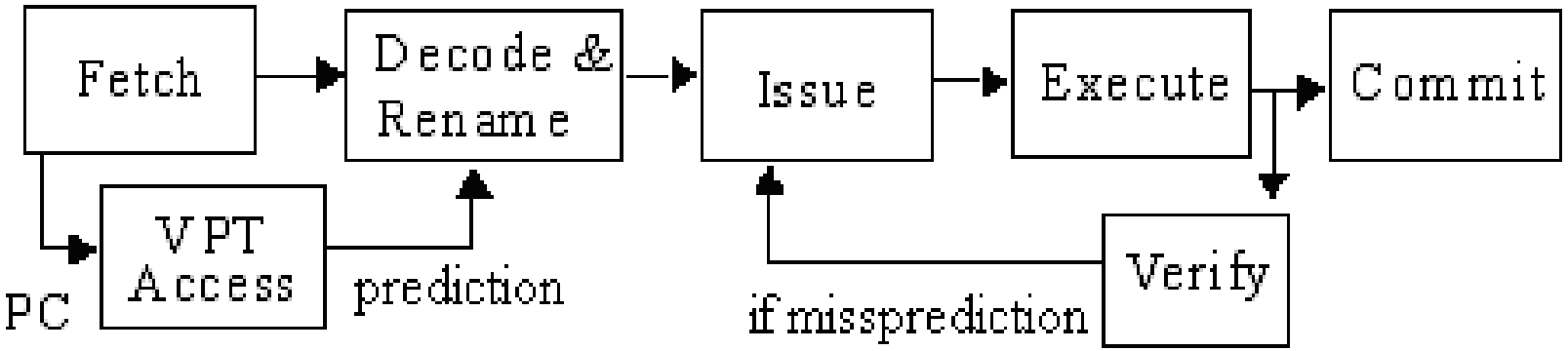
2 Solutions

- **Dynamic Instruction Reuse (DIR)** is a non-speculative technique. It comprises program critical path by **reusing** (dependent chains of) **instructions**;
- **Value Prediction (VP)** is a speculative technique. It comprises program critical path by **predicting the instructions results** during their fetch or decode pipeline stages, and unblocking dependent waiting instructions. **Value Locality.**

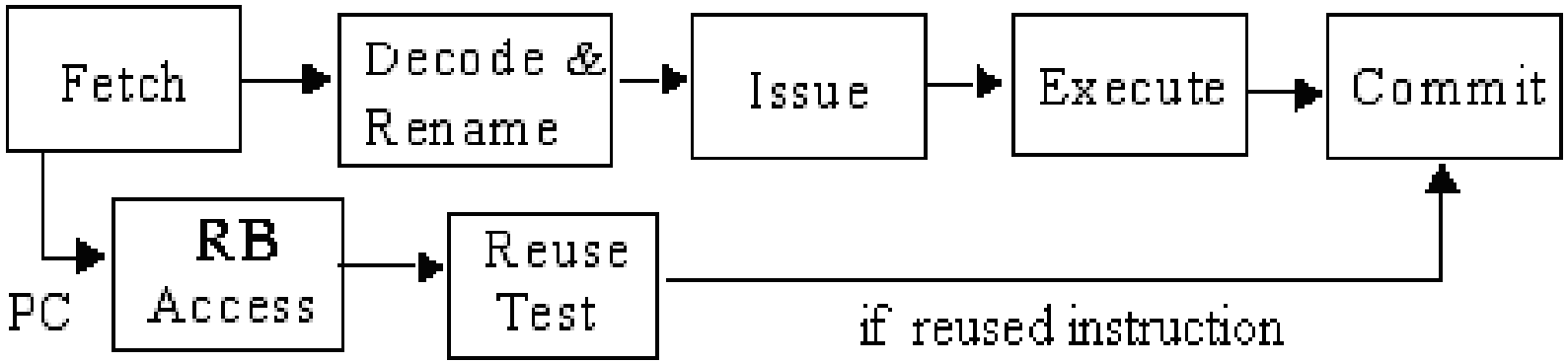
Challenge

- **Exploiting Selective Instruction Reuse and Value Prediction in a Superscalar / Simultaneous Multithreaded (SMT) Architecture to anticipate Long-Latency Instructions Results**
  - Selective Instruction Reuse (MUL & DIV)
  - Selective Load Value Prediction ("Critical Loads")

Dynamic Instruction Reuse vs. Value Prediction

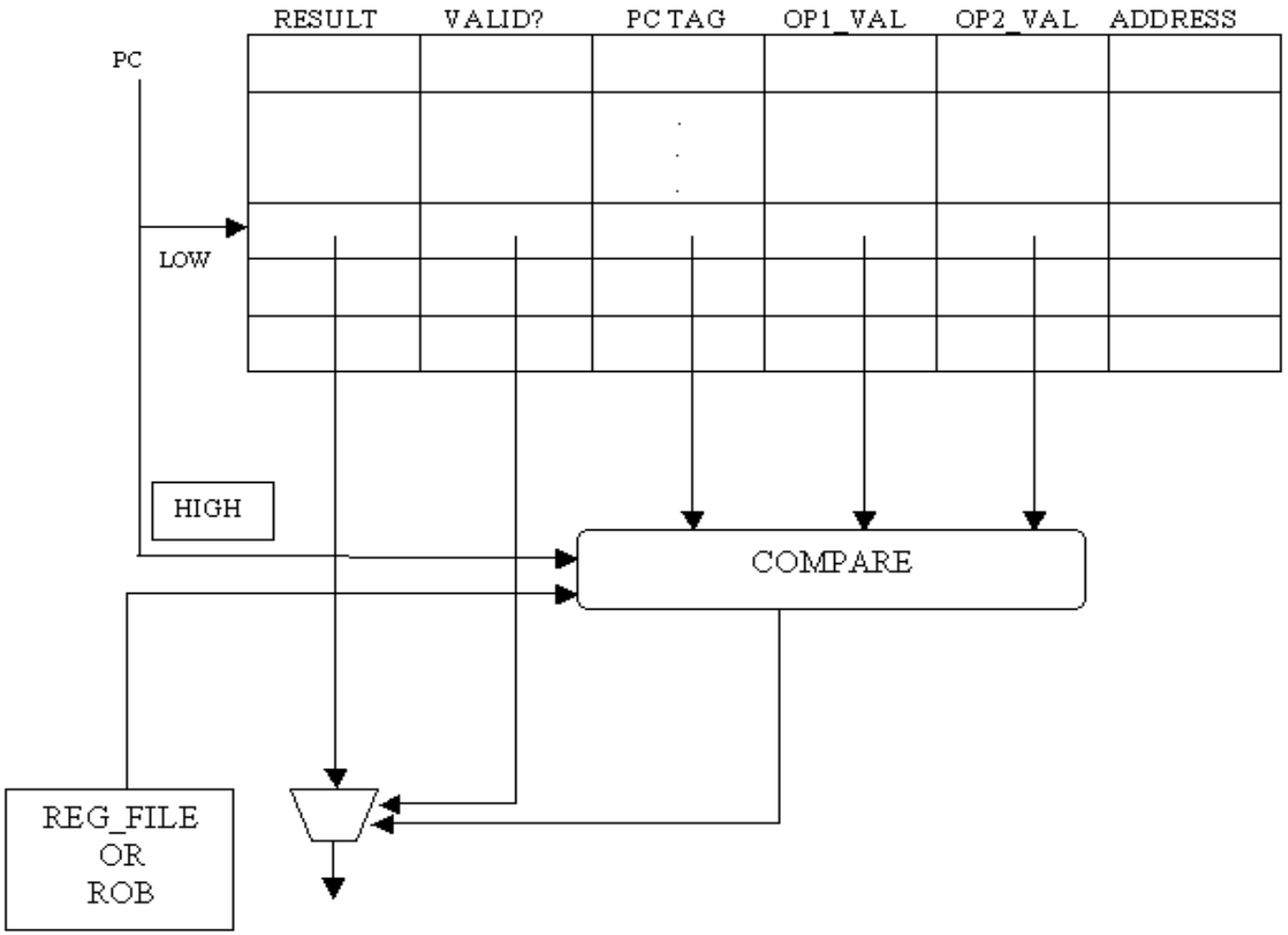


(a)

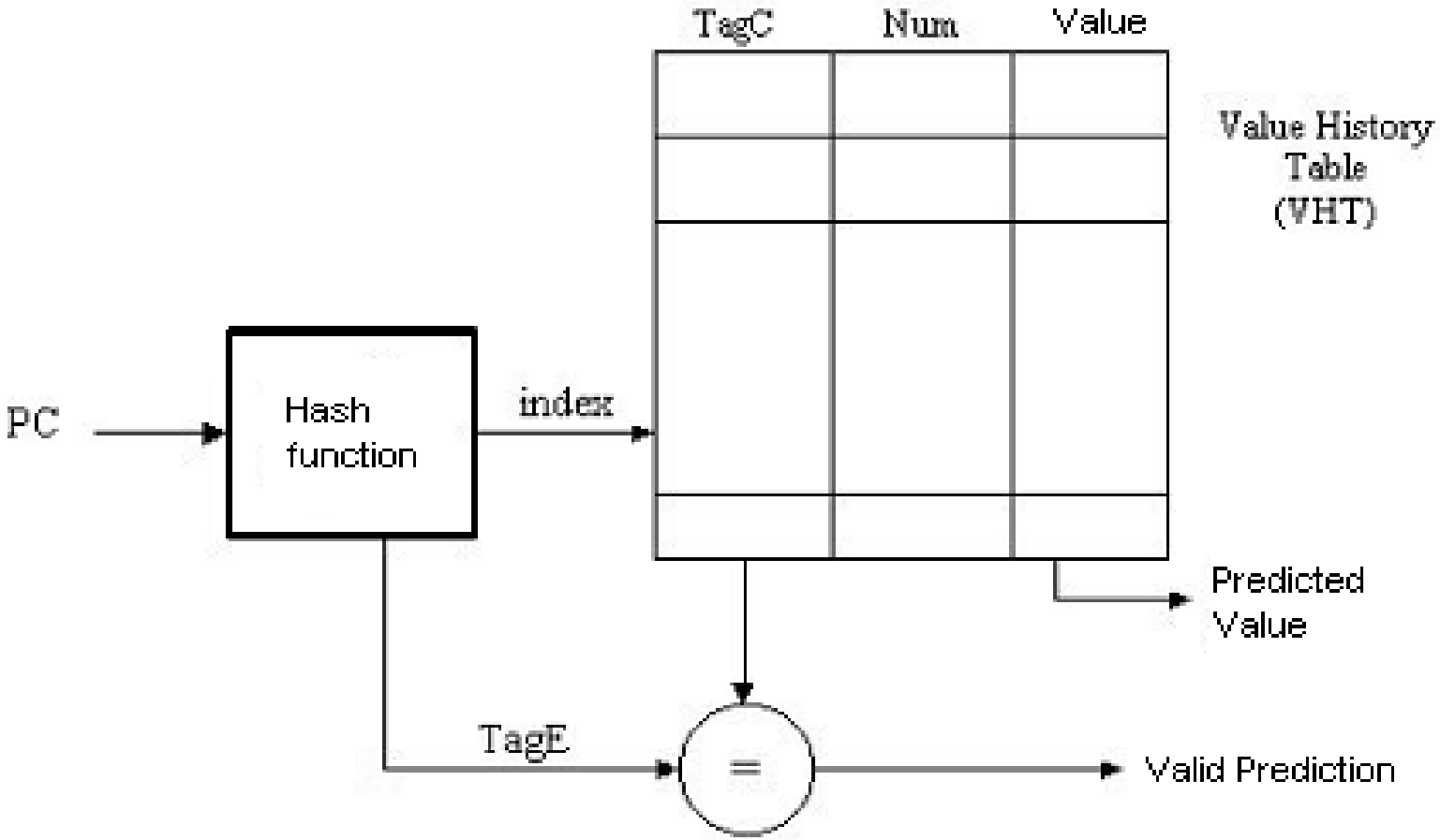


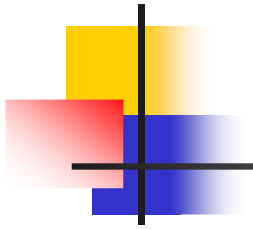
(b)

A Dynamic Instruction Reuse Scheme



A Last Value Prediction Scheme





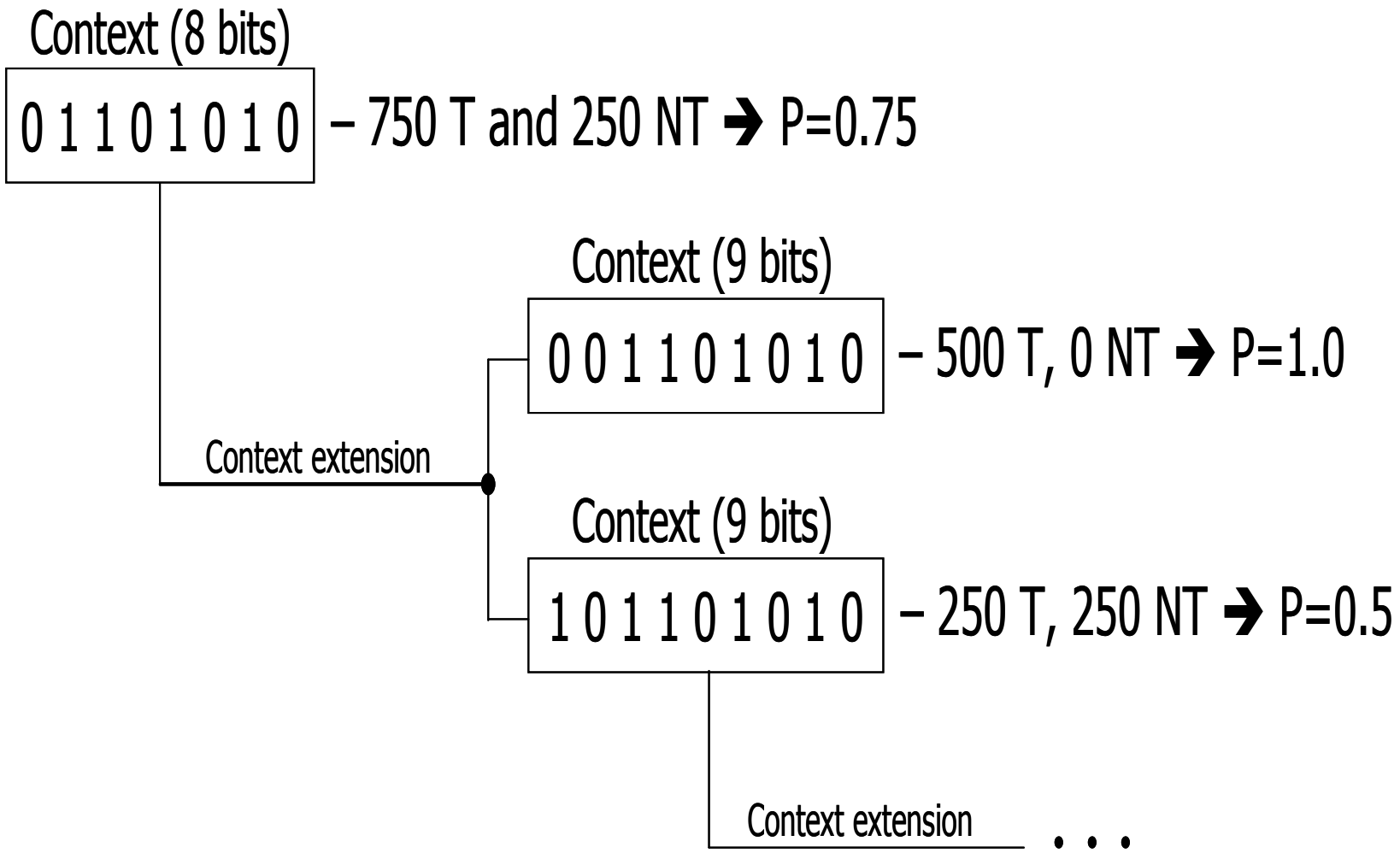
Identifying some Difficult-to-Predict Branches

Our Scientific Hypothesis was:

A **branch** in a certain **dynamic context (GHR, LHRs, etc.)** is **difficult-to-predict** if:

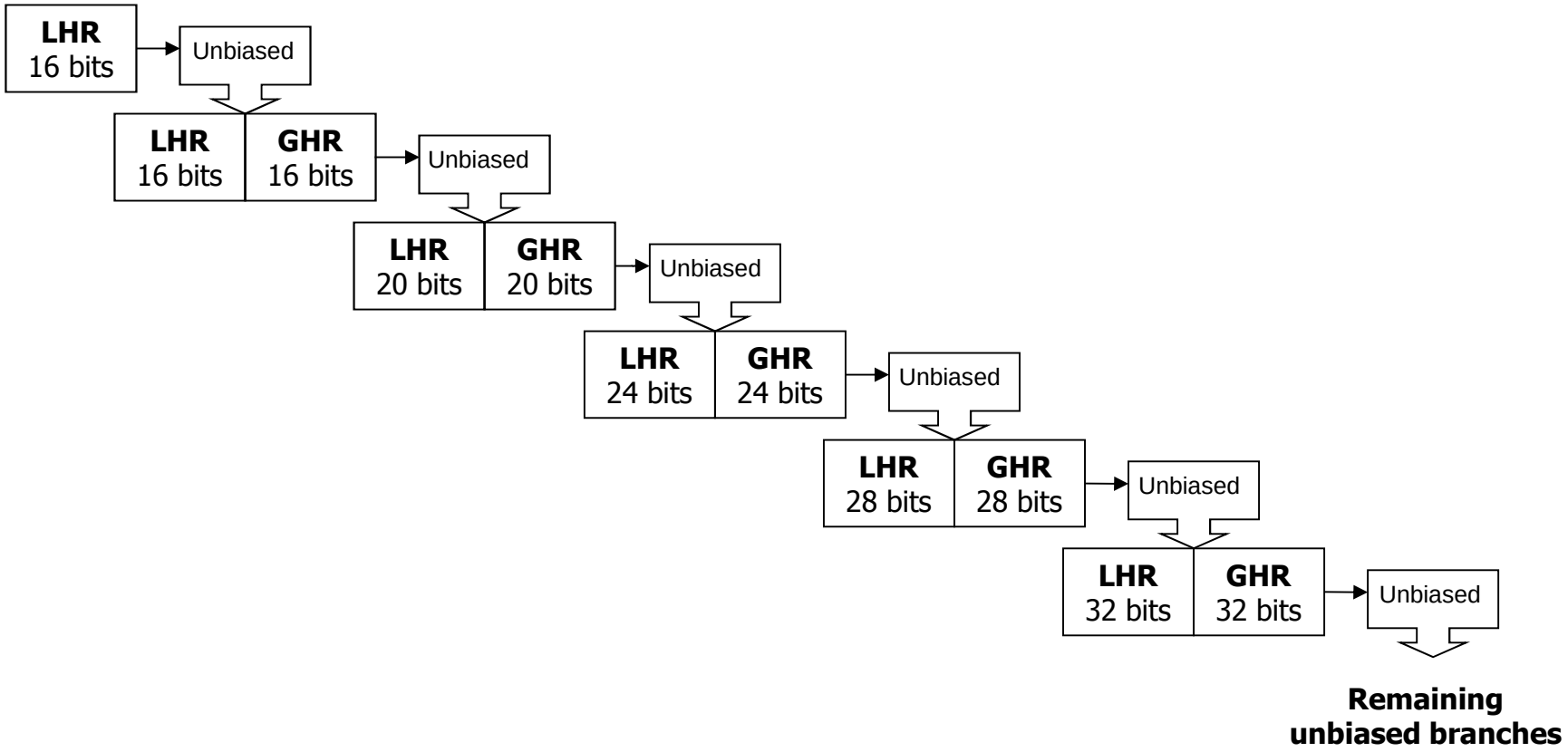
- It is *unbiased* – the branch behavior (taken/not taken) **is not sufficiently polarized** for that context;
- The taken/not taken **outcomes are „highly shuffled".**

An Unbiased Branch. Context Extension

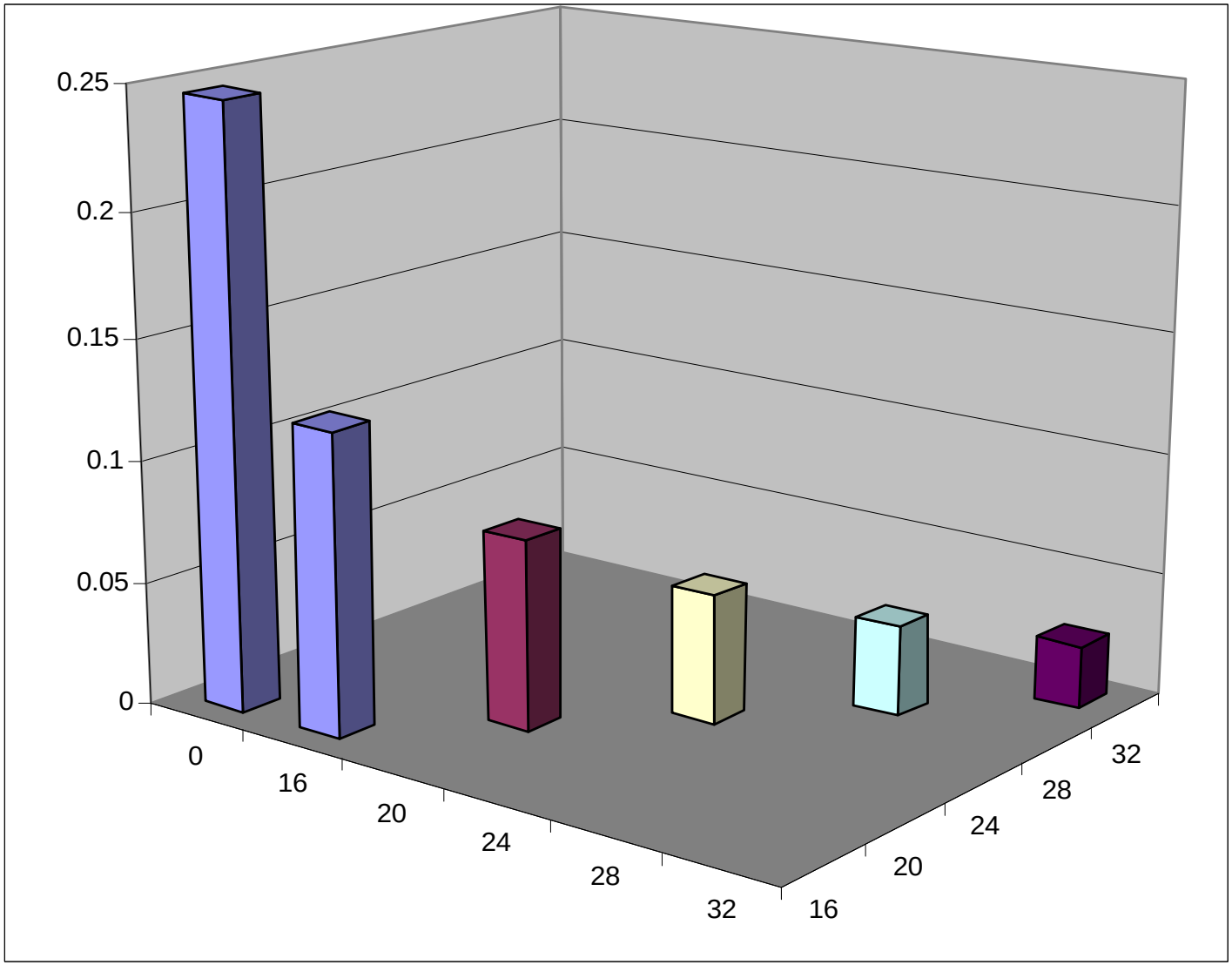


Identifying Difficult-to-Predict Branches (SPEC)

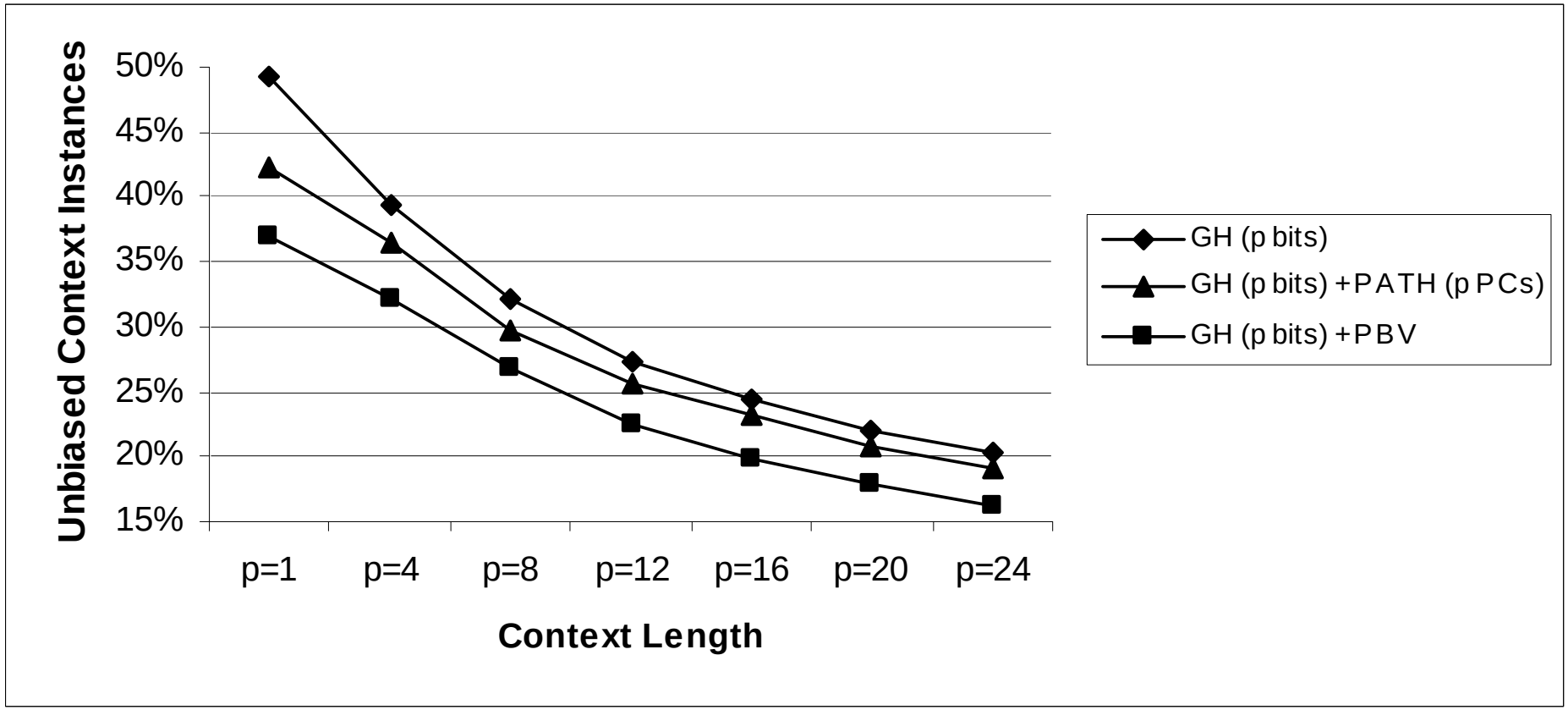
Identification Methodology



Decreasing the average percentage of unbiased branches by **extending the contexts** (*GHR, LHRs*)

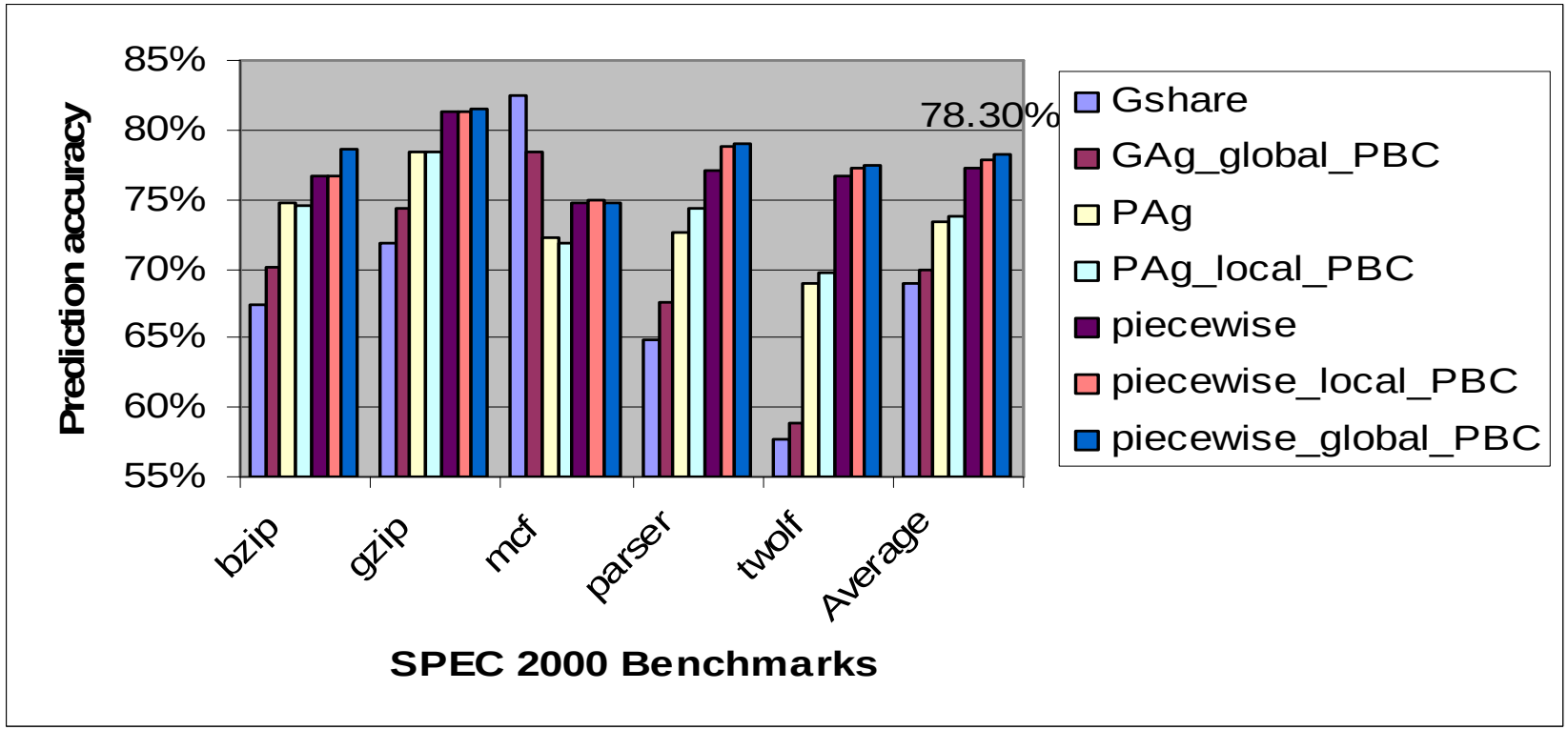


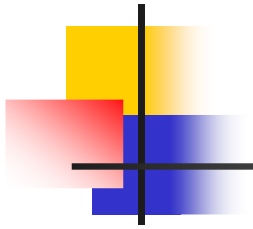
Decreasing the average percentage of unbiased branches by **adding new information** (*PATH*, *PBV*)



Predicting Unbiased Branches

- Even ***state of the art*** **branch predictors are unable to accurately predict** unbiased branches;
- The problem consists in **finding new relevant information** that could reduce their entropy instead of developing new predictors;
- **Challenge**: **adequately representing unbiased branches in the feature space!**
- **Accurately Predicting Unbiased Branches is still an Open Problem!**





Random Degrees of Unbiased Branches

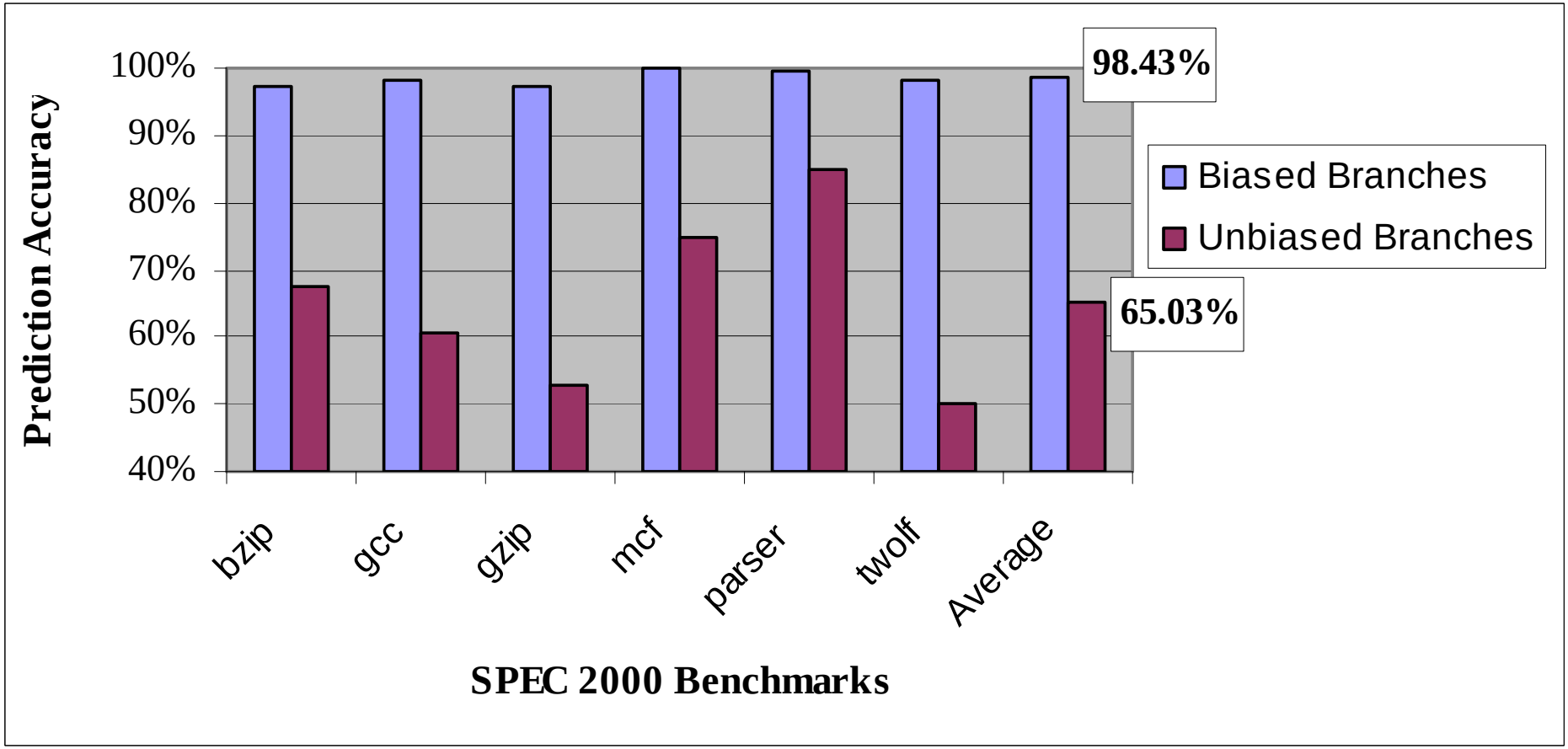
Random Degree Metrics

Based on:

- **Hidden Markov Model (HMM)** – a strong method to evaluate the predictability of the sequences generated by unbiased branches;
- **Discrete entropy** of the sequences generated by unbiased branches;
- **Compression rate** (*Gzip*, *Huffman*) of the sequences generated by unbiased branches.

Random Degrees of Unbiased Branches

Prediction Accuracies using our best evaluated HMM (2 hidden states)



Random Degrees of Unbiased Branches

Random Degree Metric Based on Discrete Entropy

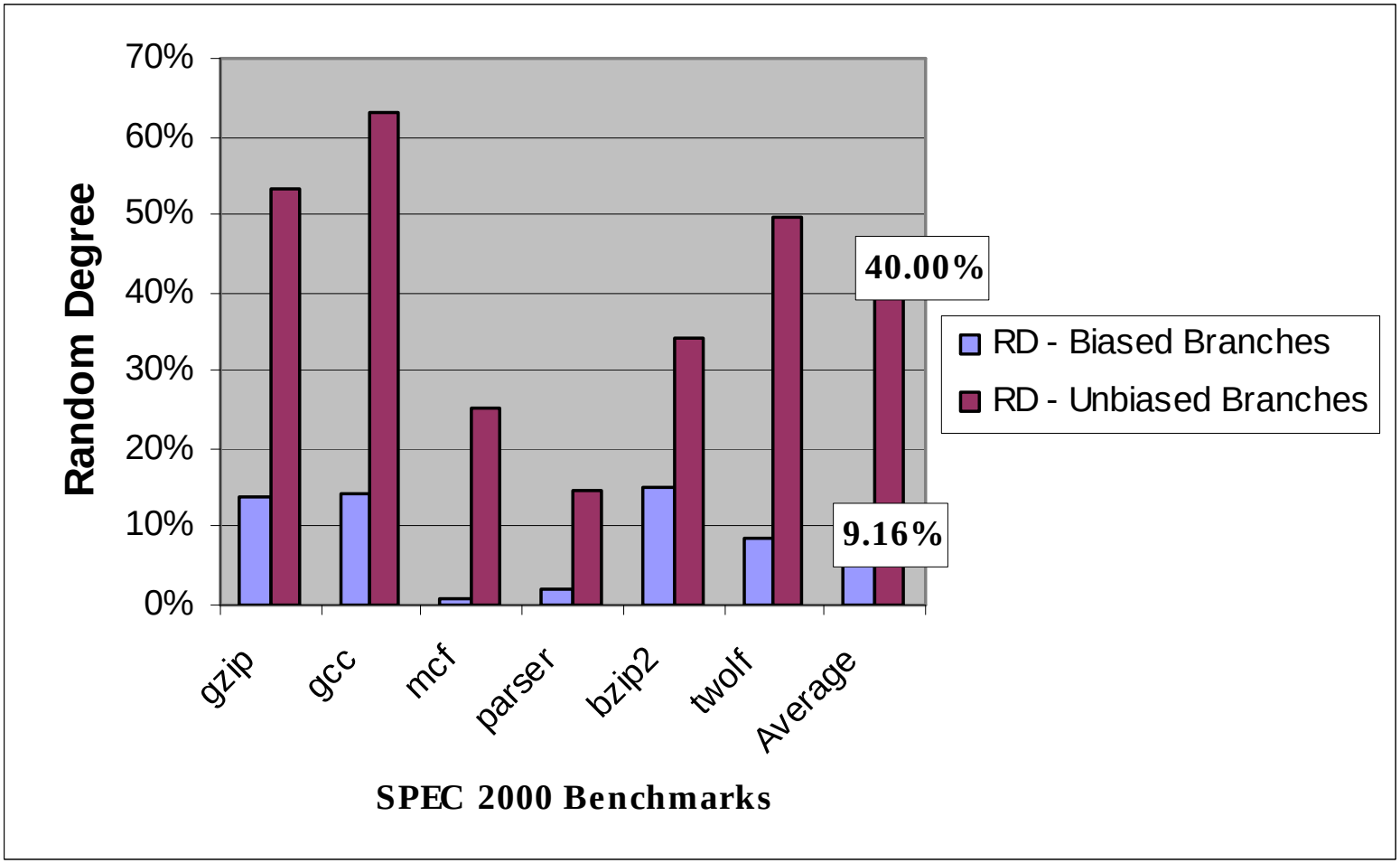
- $$E(S) = -\sum_{i=1}^{k} P(Xi)\log_2 P(Xi) \geq 0$$

- $$D(S_i) = \begin{cases} 0, & n_t = 0 \\ \dfrac{n_t}{2 \cdot \min(NT, T)}, & n_t > 0 \end{cases}$$

- $$RD(S) = D(S) \cdot E(S) \in [0, \log_2 k]$$

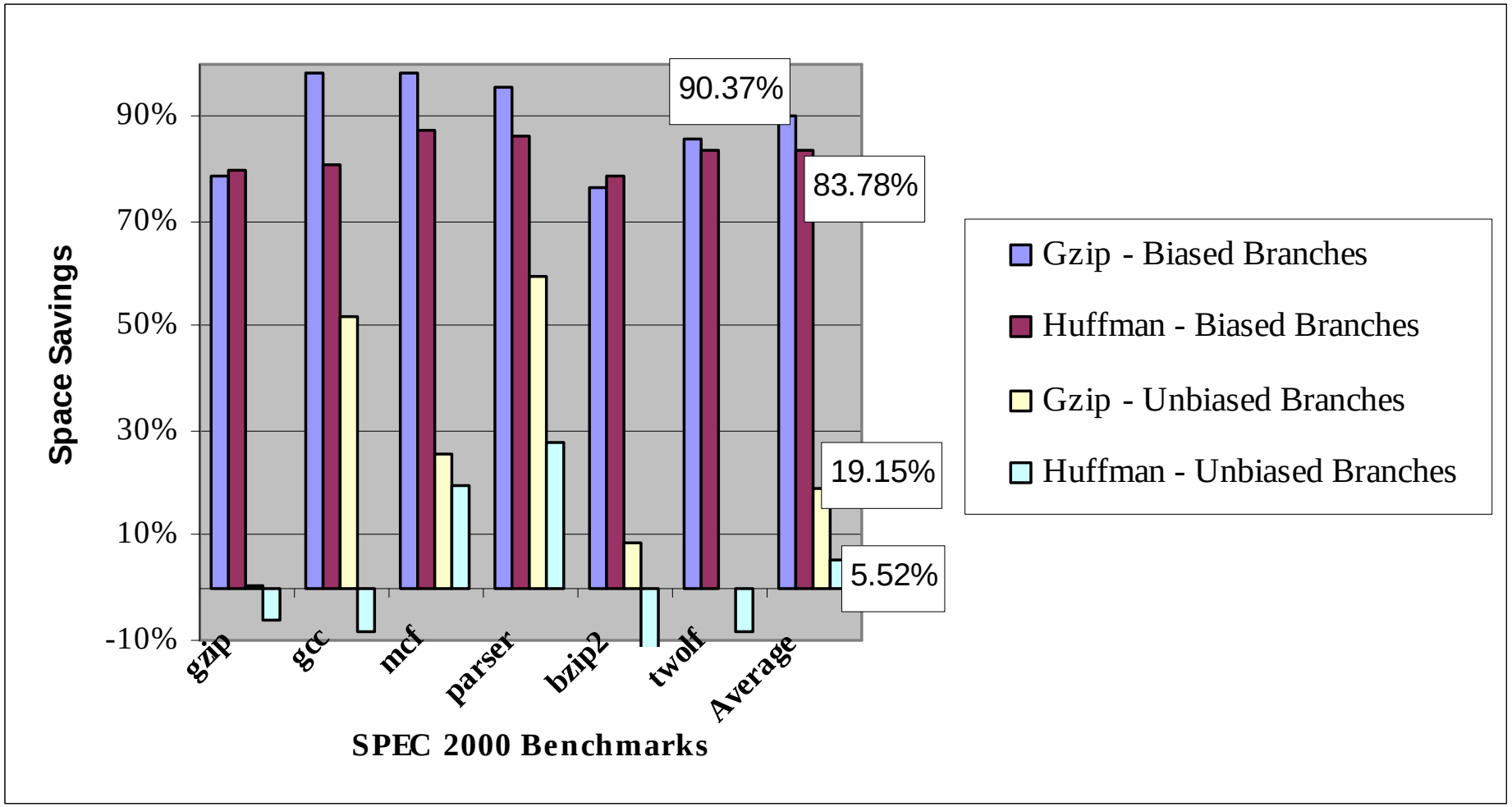
Random Degrees of Unbiased Branches

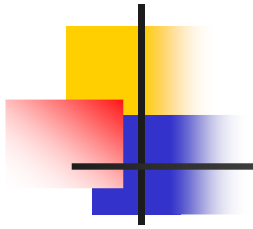
Random Degree Metric Based on Discrete Entropy. Results



Random Degrees of Unbiased Branches

"Space Savings" using *Gzip* and *Huffman* Algorithms





Exploiting Selective Instruction Reuse and Value Prediction in a Superscalar Architecture

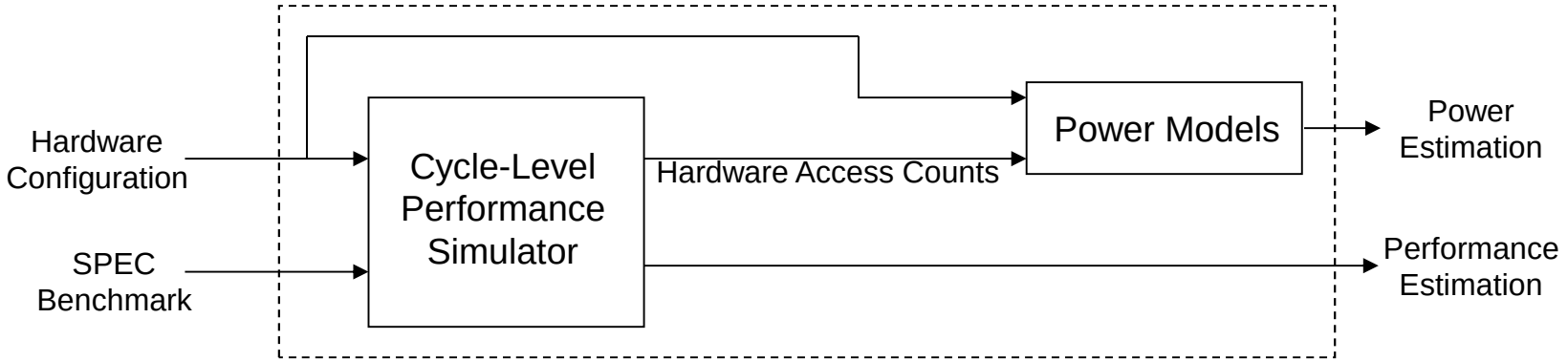
- **Long-latency instructions** represent another source of **ILP limitation**;
- This limitation is accentuated by the fact that about 28% of branches (5.61% being unbiased) subsequently **depend on critical Loads**;
- 21% of branches (3.76% being unbiased) subsequently **depend on Mul/Div**;
- In such cases **misprediction penalty is much higher** because the long-latency instruction must be solved first;
- Therefore we **speed-up the execution of long-latency instruction by anticipating their results**;
- We **predict critical Loads** and **reuse Mul/Div** instructions (results).

Parameters of the simulated **superscalar/(SMT)** architecture (M-Sim)

	Execution unit	Number of units	Operation latency
Execution latencies	intALU	4	1
	intMULT/intDIV	1	3/20
	fpALU	4	2
	fpMULT/fpDIV	1	4/12
Superscalarity	*Fetch*/*Decode*/*Issue*/*Commit* width = 4		
Branch predictor	Bimodal predictor with 2048 entries		
Caches and memory	Memory unit		Access latency
	4-way associative L1 data cache, 32 KB		1 cycle
	8-way associative unified L2 data cache, 512 KB		6 cycles
	Memory		100 cycles
Resources	Register file: 32 INT/32 FP		
	Reorder buffer (ROB): 128 entries		
	Load/store queue (LSQ): 48 entries		

Exploiting Selective Instruction Reuse and Value Prediction in a **Superscalar** Architecture

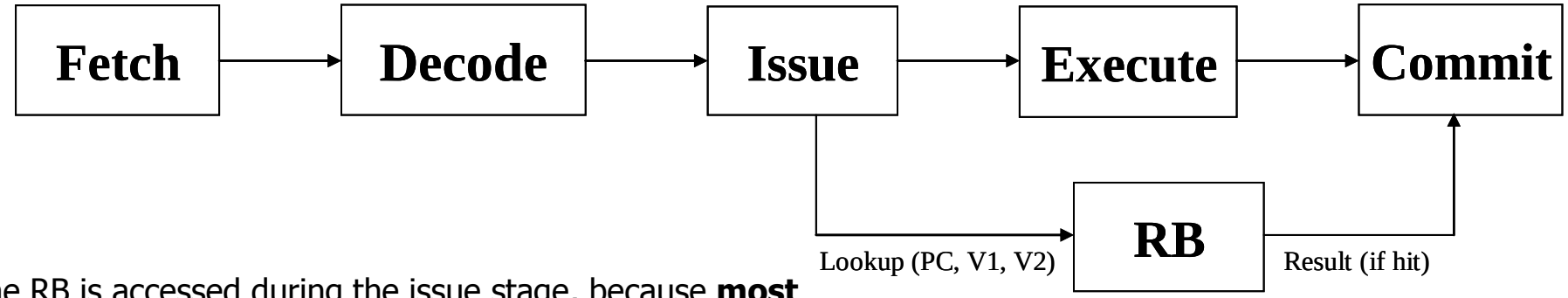
The M-SIM Simulator



- $$IPC\ Speedup = \frac{IPC_{improved} - IPC_{base}}{IPC_{base}} \cdot 100\%$$
- $$EDP = \frac{Total\ Power}{IPC^2}$$
- $$EDP\ Gain = \frac{EDP_{base} - EDP_{improved}}{EDP_{base}} \cdot 100\%$$

Exploiting Selective Instruction Reuse and Value Prediction in a **Superscalar** Architecture

Selective Instruction Reuse (MUL & DIV)



The RB is accessed during the issue stage, because **most of the MUL/DIV instructions** found in the RB during the dispatch stage **do not have their operands ready**.

S_v Reuse Buffer (RB)

PC of MUL / DIV →

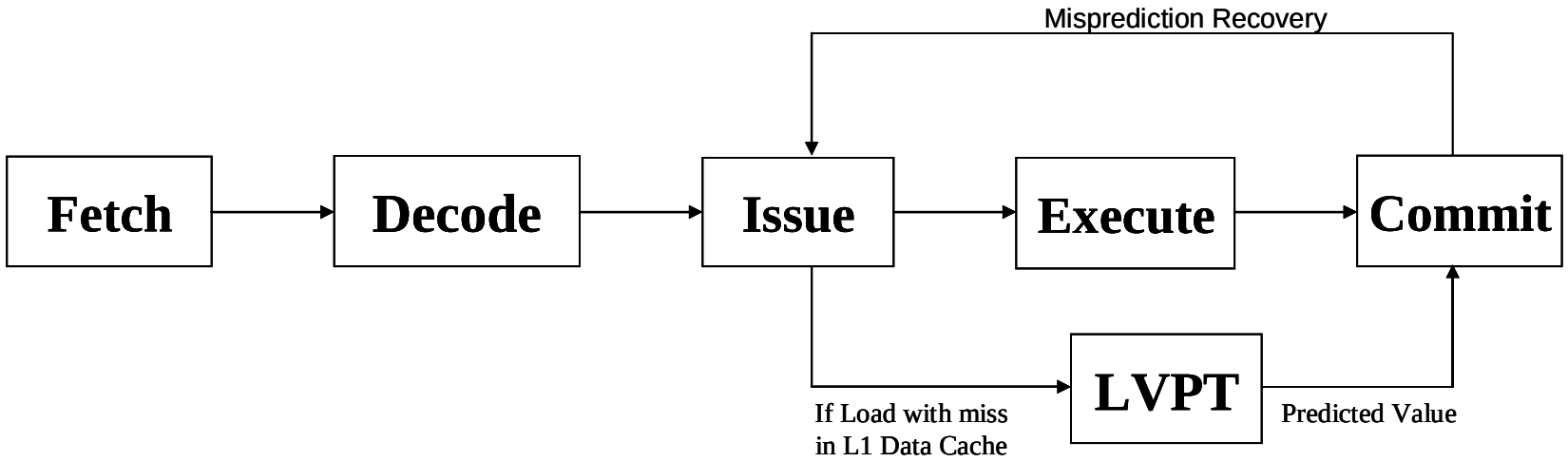
Tag	SV1	SV2	Result

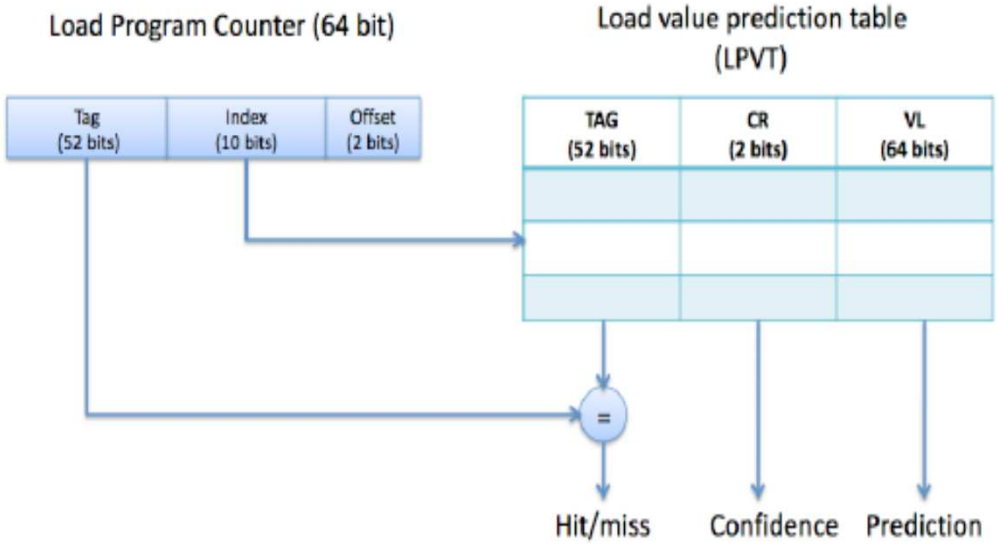
Trivial MUL/DIV Detection:

- V*0, V*1;
- 0/V, V/1, V/V.

Exploiting Selective Instruction Reuse and Value Prediction in a **Superscalar** Architecture

Selective Load Value Prediction (Critical Loads)

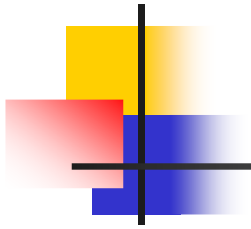




LVP Recovery Principle

- ***1: ld.w 0($r2) → $r3;*** **miss in D-Cache!**
- ***2: add $r4, $r3 → $r5***
- ***3: add $r3, $r6 → $r8***

- Unless $r3 is known, both 2 and 3 should be **serialized**;
- A LVP allows predicting $r3 before instruction 1 has been completed to start **both 2 and 3 earlier** (and **in parallel**);
- Whenever the prediction is verified to be wrong, a **recovery mechanism** is activated;
- In the previous case, this consists of **squashing** both instructions 2 and 3 from the ROB and **re-execute** them with the $r3 correct value (**selective re-issue**).

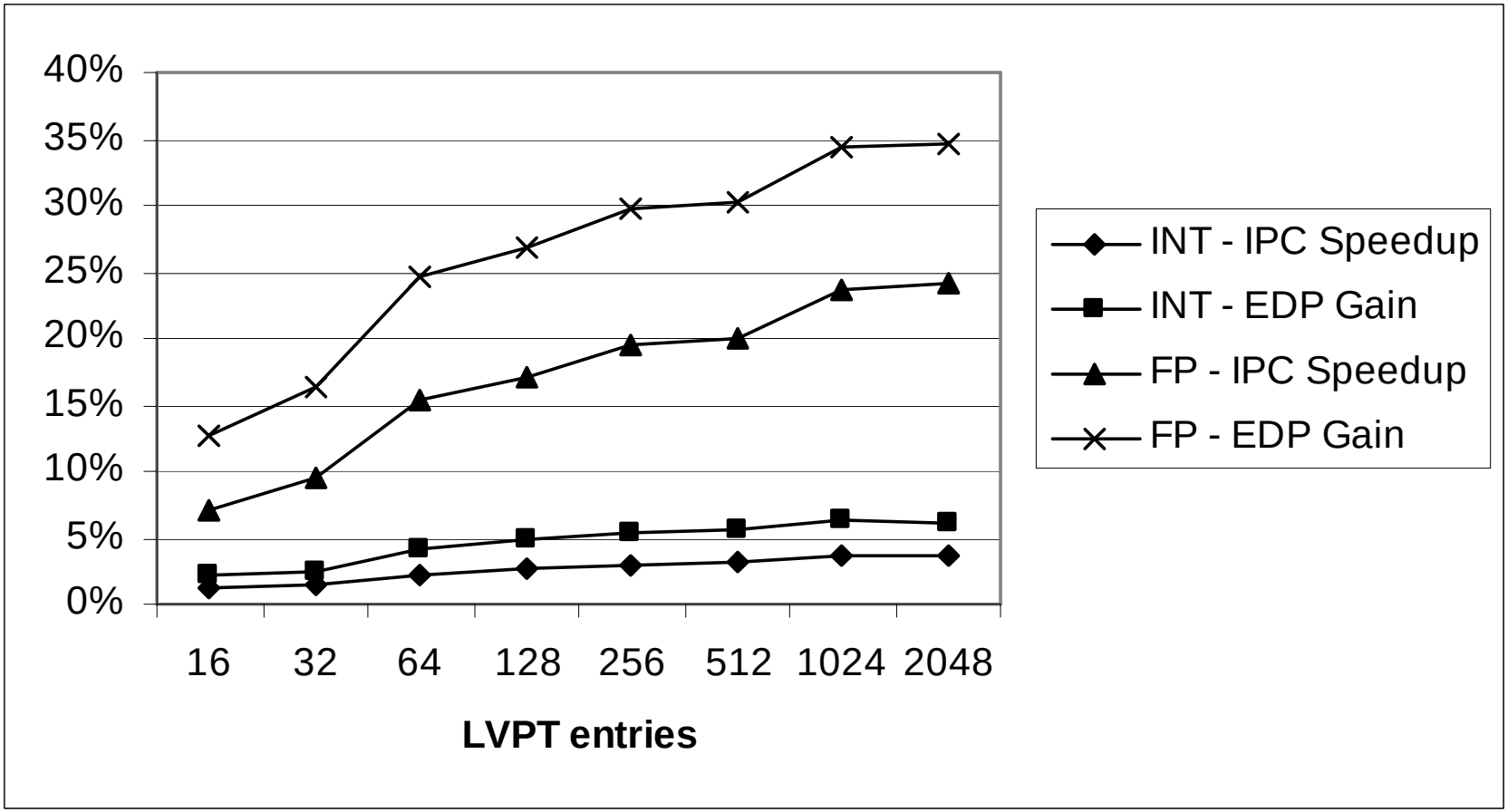


Benchmarking Methodology

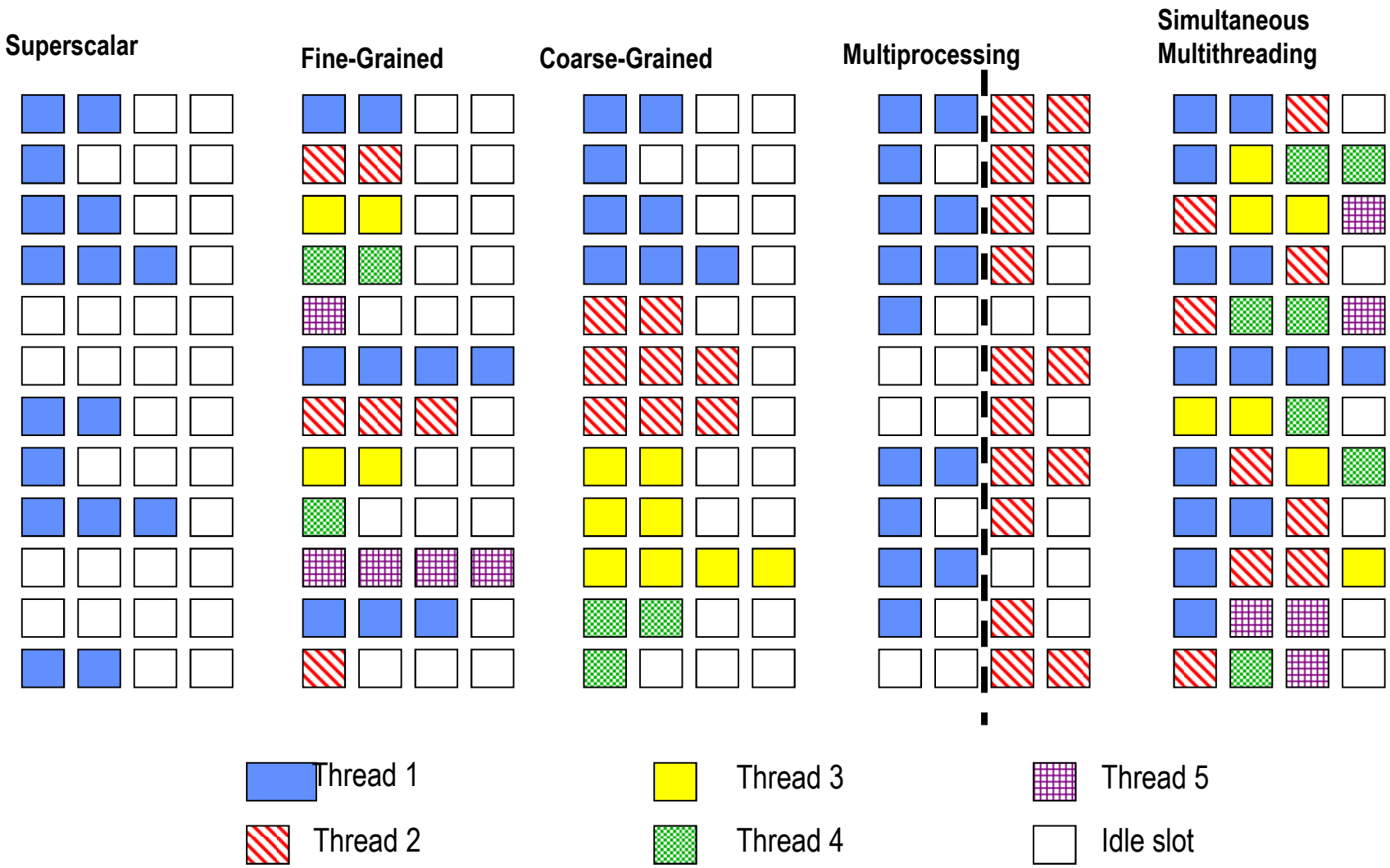
- **7 integer SPEC 2000 benchmarks computation intensive** (*bzip, gcc, gzip*) and **memory intensive** (*mcf, parser,twolf, vpr*);
- **6 floating-point SPEC 2000 benchmarks** (*applu, equake, galgel, lucas, mesa,mgrid*);
- Results reported on **1 billion dynamic instructions**, skipping the first 300 million instructions.

Exploiting Selective Instruction Reuse and Value Prediction in a **Superscalar** Architecture

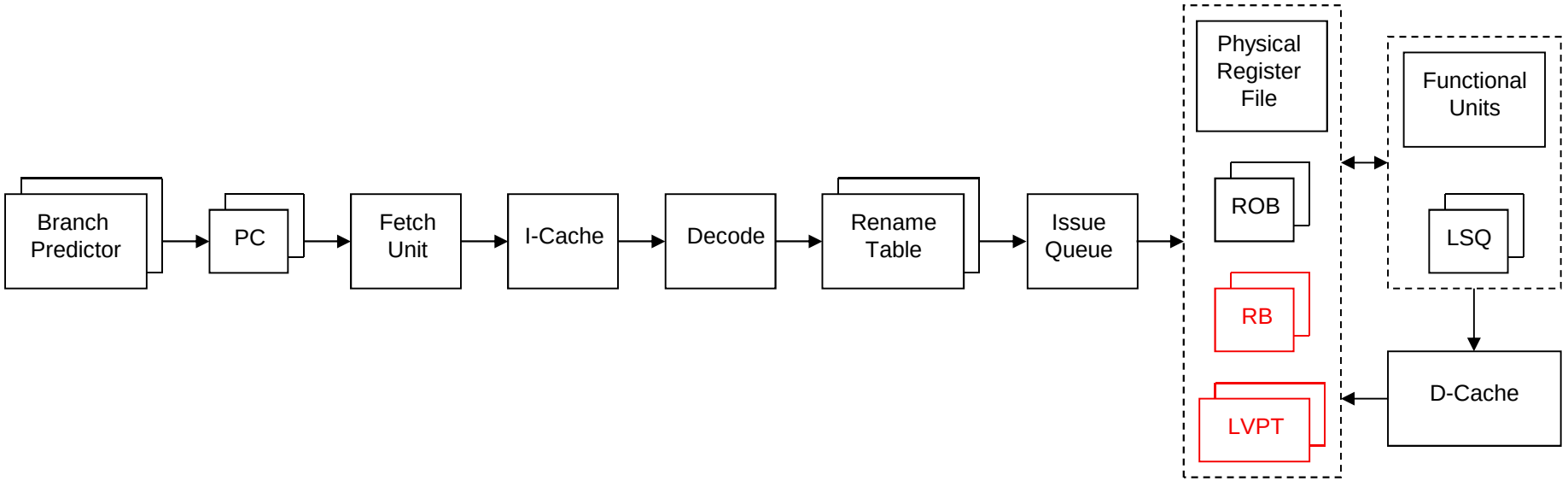
Relative IPC speedup and relative energy-delay product gain with a Reuse Buffer of 1024 entries, the Trivial Operation Detector, and the Load Value Predictor



Multithreaded Processing Metaphors

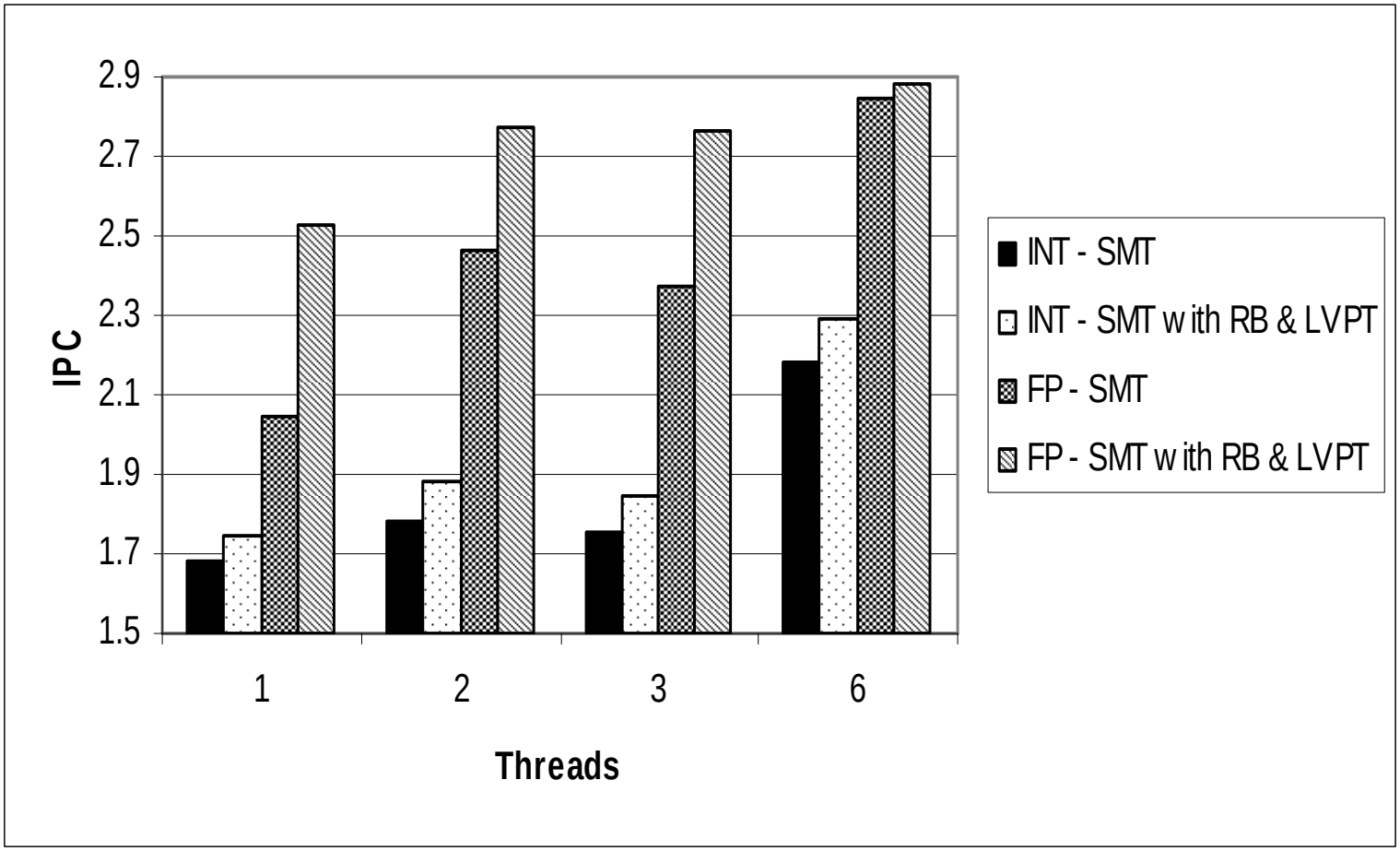


SMT Architecture (M-Sim) enhanced with per Thread RB and LVPT Structures



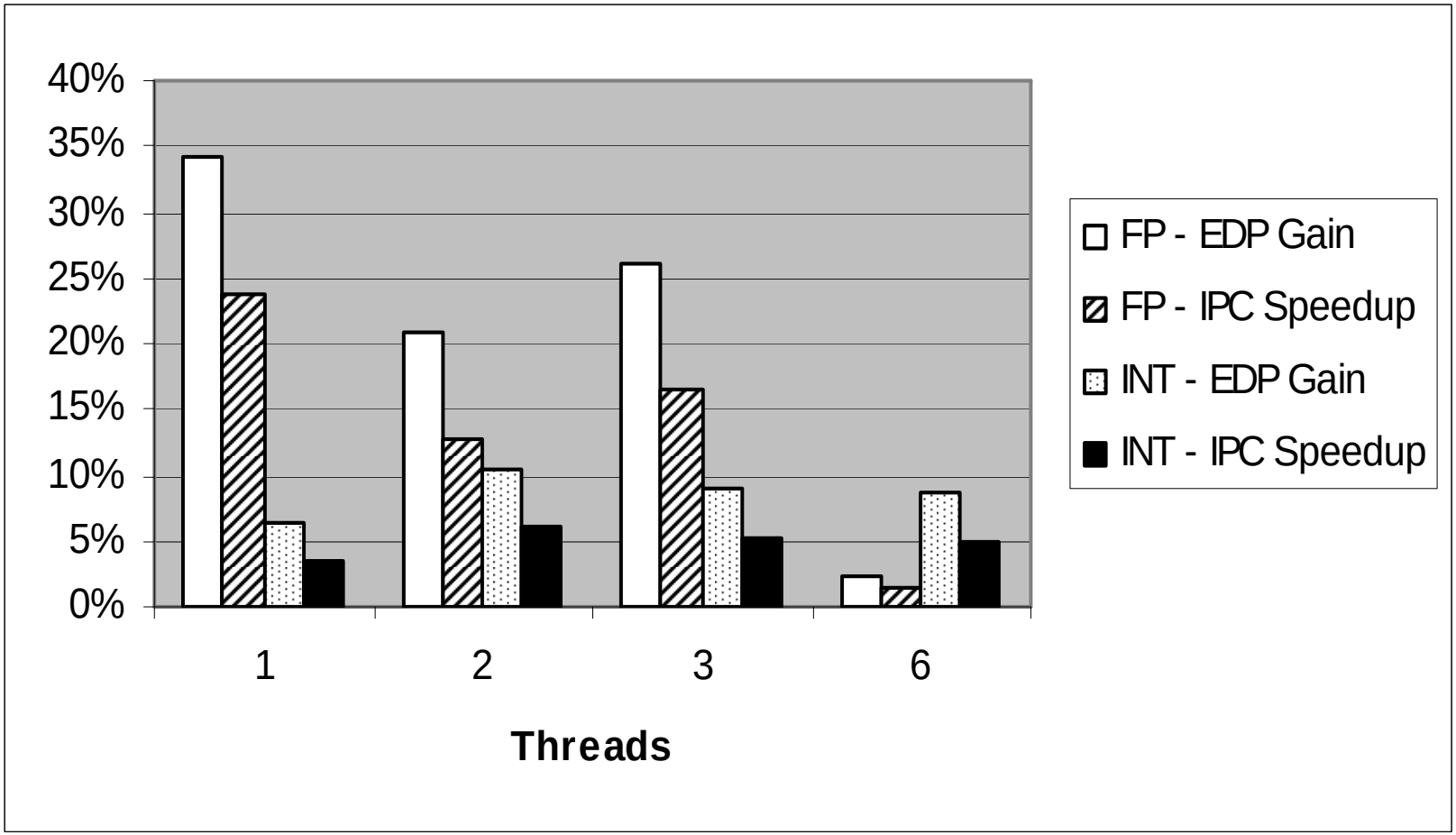
Selective Instruction Reuse and Value Prediction in **Simultaneous Multithreaded** Architectures

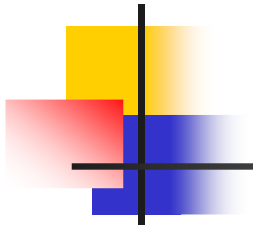
IPCs obtained using the SMT architecture with/without 1024 entries RB & LVPT



Selective Instruction Reuse and Value Prediction in **Simultaneous Multithreaded** Architectures

Relative IPC speedup and EDP gain (enhanced SMT vs. classical SMT)

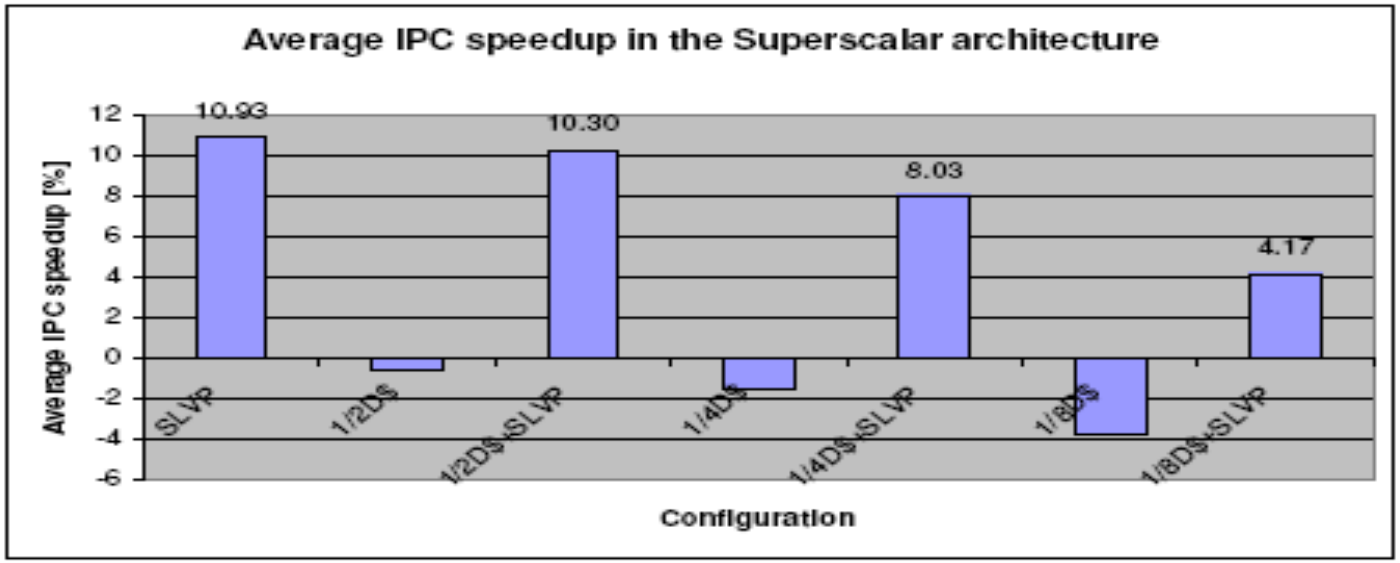


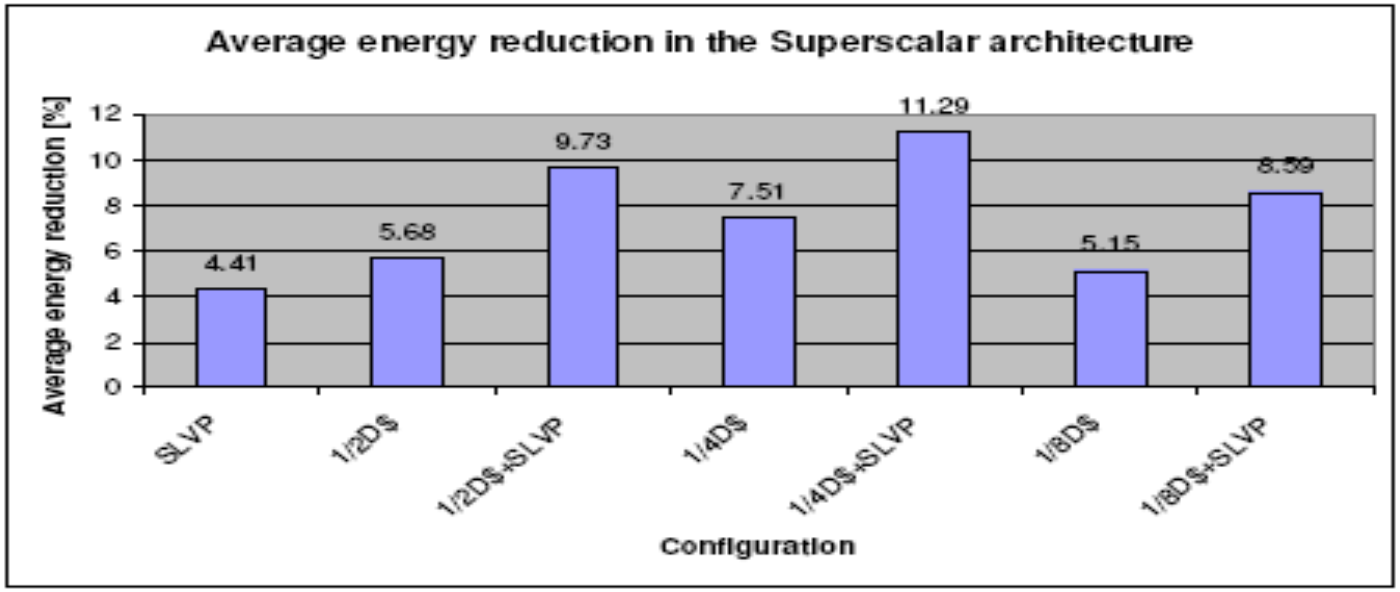


Superscalar/Simultaneous Multithreaded Architectures with **only SLVP**

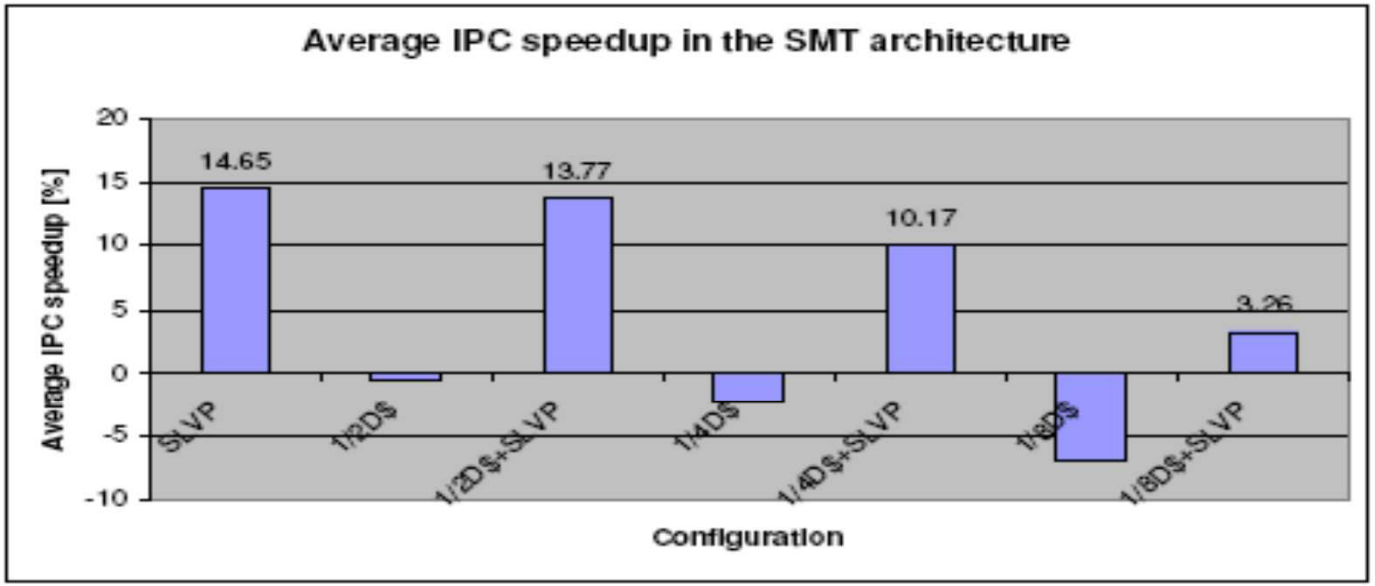
Processor	
Frequency	1.2 GHz @ 80nm
Branch predictor	Bimodal predictor 2K entries
Fetch / Decode / Issue / Commit width	4 / 4 / 4 / 4
Register File size	128 INT + 128 FP
Fetch queue size	32 entries
ROB size	128 entries
Load/Store queue size	48 entries
Integer ALU #units / latency	4 / 1 cycle
Integer MUL-DIV #units / latency	1 / 3(MUL) - 20(DIV) cycles
Floating Point ALU #units / latency	4 / 2 cycles
Floating Point MUL-DIV #units / latency	1 / 4(MUL) - 12(DIV) cycles
Caches and Memory	
L1 I$ size/associativity/block	64KB / 2-way / 64B
L1 D$ size/associativity/block	64KB / 2-way/ 64B
L2$ size/associativity/block	4MB / 8-way / 64B
L1 I$ / L1 D$ / L2$ access latency	2 / 2 / 14 cycles
Memory access latency	270 cycles
Load Value Prediction Table	
SLVP size	1024 entries, direct mapped
SLVP access latency	1 cycle

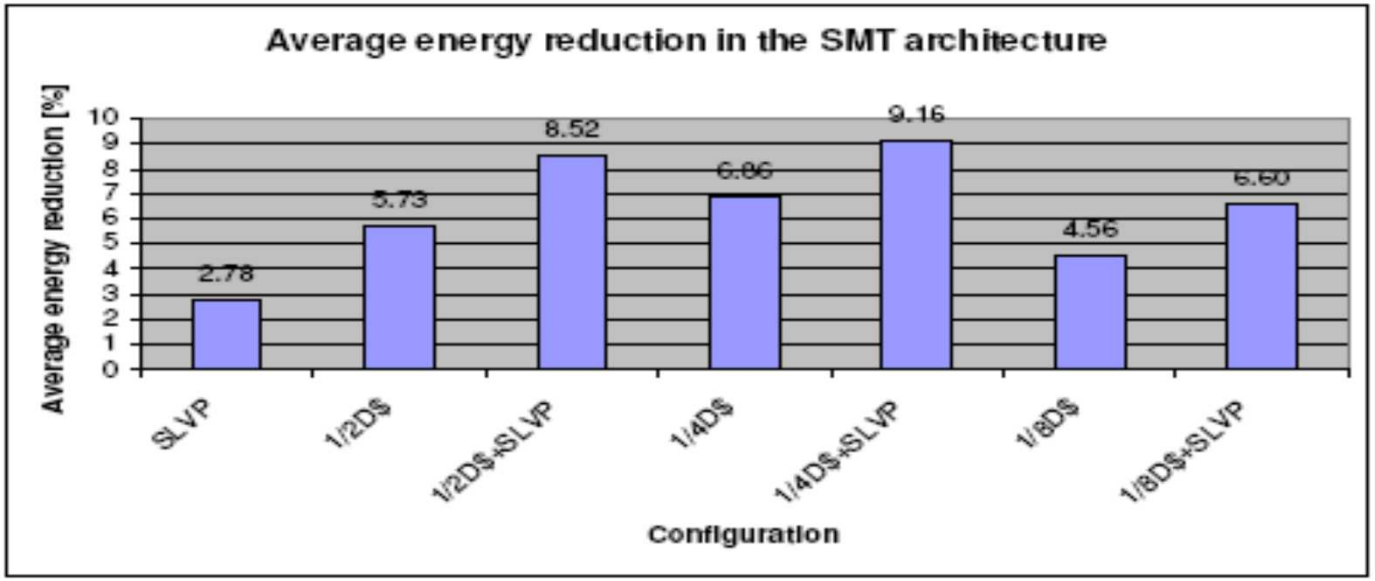
Design Space Exploration in the **Superscalar & SLVP Architecture** (1/4D+SLVP)

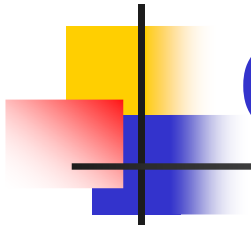




Design Space Exploration in the **SMT & SLVP Architecture** (1/4D+SLVP)



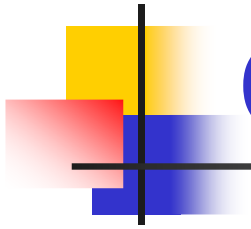




Conclusions and Further Work

Conclusions (I)

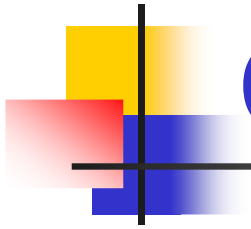
- We developed some **random degree metrics** to characterize the randomness of sequences produced by unbiased branches. All these metrics are showing that **sequences generated by unbiased branches are characterized by high "random degrees"**. They might help the computer architect;
- We **improved superscalar architectures** by **selectively anticipating long-latency instructions**. **IPC Speedup**: 3.5% on SPECint2000, 23.6% on SPECfp2000. **EDP Gain**: 6.2% on SPECint2000, 34.5% on SPECfp2000;
- We analyzed the efficiency of these **selective anticipatory methods in SMT architectures**. **They improve the IPC** on all evaluated architectural configurations.



Conclusions and Further Work

Conclusions (II)

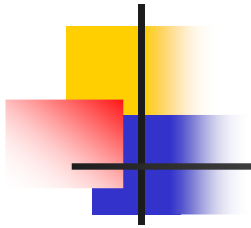
- A SLVP **reduces the energy consumption** of the on-chip memory comparing it with a Non-Selective LVP scheme;
- It creates room for a **reduction of the D-Cache size by preserving performance**, thus enabling **a reduction of the system cost;**
- **1024 entries SLVP + ¼ D-Cache** (16 KBytes/2-way/64B) seem to be a good trade-off in both superscalar and SMT cases.



Conclusions and Further Work

Further Work

- **Indexing the SLVP table with the memory address** instead of the instruction address (PC);
- **Exploiting an N-value locality** instead of 1-value locality;
- Generating the **thermal maps** for the optimal superscalar and SMT configurations (and, if necessary, developing a run-time thermal manager);
- Understanding and exploiting **instruction reuse and value prediction** benefits in a **multicore architecture**.

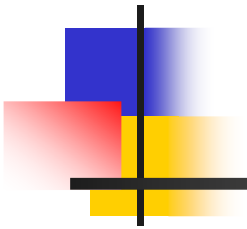


Anticipatory multicore architectures

- Anticipatory multicores would significantly **reduce the pressure on the interconnection network→ performance/energy**;
- Predicting an instruction value and, later verifying the prediction might be not sufficient. There could appear **data consistency errors** (e.g. the CPU correctly predict a value representing a D-memory address but it subsequently could read an incorrect value from that speculative memory address!) → **consistency violation detection and recovery;**
- **The inconsistency cause: VP might execute *out of order* some dependent instructions;**
- Between **value prediction, multithreading and the cache coherence/consistence mechanisms** there are subtle, not well-understood relationships;
- Nobody analyzed Dynamic Instruction Reuse in a multicore system. It will supplementary add the **Reuse Buffers coherence problems**. The already developed cache coherence mechanisms would help solving Reuse Buffers coherency.

Some Refererences

- **L. VINTAN, A. GELLERT, A. FLOREA, M. OANCEA, C. EGAN – *Understanding Prediction Limits through Unbiased Branches*, Eleventh Asia-Pacific Computer Systems Architecture Conference, Shanghai 6-8th, September, 2006 - http://webspace.ulbsibiu.ro/lucian.vintan/html/LNCS.pdf**
- **A. GELLERT, A. FLOREA, M. VINTAN, C. EGAN, L. VINTAN - *Unbiased Branches: An Open Problem*, The Twelfth Asia-Pacific Computer Systems Architecture Conference (ACSAC 2007), Seoul, Korea, August 23-25th, 2007 - http://webspace.ulbsibiu.ro/lucian.vintan/html/acsac2007.pdf**
- **VINTAN L. N., FLOREA A., GELLERT A. – *Random Degrees of Unbiased Branches,* Proceedings of The Romanian Academy, Series A: Mathematics, Physics, Technical Sciences, Information Science, Volume 9, Number 3, pp. 259 - 268, Bucharest, 2008 - http://www.academiaromana.ro/sectii2002/proceedings/doc2008-3/13-Vintan.pdf**
- **A. GELLERT, A. FLOREA, L. VINTAN. - *Exploiting Selective Instruction Reuse and Value Prediction in a Superscalar Architecture*, Journal of Systems Architecture, vol. 55, issues 3, pp. 188-195, ISSN 1383-7621, Elsevier, 2009 - http://webspace.ulbsibiu.ro/lucian.vintan/html/jsa2009.pdf**
- **GELLERT A., PALERMO G., ZACCARIA V., FLOREA A., VINTAN L., SILVANO C. - *Energy-Performance Design Space Exploration in SMT Architectures Exploiting Selective Load Value Predictions*, Design, Automation & Test in Europe International Conference (DATE 2010), March 8-12, 2010, Dresden, Germany - http://webspace.ulbsibiu.ro/lucian.vintan/html/Date_2010.pdf**



THANK YOU!