Assessing the Effects of Low Voltage in Branch Prediction Units

Athanasios Chatzidimitriou, George Papadimitriou, Dimitris Gizopoulos
Dept. of Informatics and Telecommunications
University of Athens
Athens, Greece
{achatz, georgepap, dgizop}@di.uoa.gr

Shrikanth Ganapathy, John Kalamatianos
AMD Research
Advanced Micro Devices, Inc.
{Santa Clara CA, Boxborough MA}, USA
{shrikanth.ganapathy, john.kalamatianos}@amd.com

***Abstract*—Branch prediction units are key performance components in modern microprocessors as they are widely used to address control hazards and minimize misprediction stalls. The continuous urge of high performance has led designers to integrate highly sophisticated predictors with complex prediction algorithms and large storage requirements. As a result, BPUs in modern microprocessors consume large amounts of power. But when a system is under a limited power budget, critical decisions are required in order to achieve an equilibrium point between the BPU and the rest of the microprocessor.**

In this work, we present a comprehensive analysis of the effects of low voltage configuration Branch Prediction Units (BPU). We propose a design with separate voltage domain for the BPU, which exploits the speculative nature of the BPU (which is self-correcting) that allows reduction of power without affecting functional correctness. Our study explores how several branch predictor implementations behave when aggressively undervolted, the performance impact of BTB as well as in which cases it is more efficient to reduce the BP and BTB size instead of undervolting. We also show that protection of BPU SRAM arrays has limited potential to further increase the energy savings, showcasing a realistic protection implementation. Our results show that BPU undervolting can result in power savings up to 69%, while the microprocessor energy savings can be up to 12%, before the penalty of the performance degradation overcomes the benefits of low voltage. Neither smaller predictor sizes nor protection mechanisms can further improve energy consumption.

Keywords—Branch predictors, energy efficiency, gem5, microarchitectural simulation, power, voltage scaling.

I. INTRODUCTION

High-performance microprocessors require sophisticated branch predictors which facilitate high fetch bandwidth and better-IPC program executions. Accurate branch predictors employ large SRAM tables that are being accessed on every fetch cycle [1], leading to higher power consumption. The most power consuming components of a BPU are the Branch Target Buffer (BTB) and Branch direction prediction arrays [2], [3], [4], [5]. Recent work has estimated that the front-end of a superscalar microprocessor contributes up to 33% of the overall microprocessor power [4]. This property is undesirable in processors operating under limited power budgets such as those used in embedded systems.

Scaling down the supply voltage (V_{DD}) can achieve significant improvements in energy efficiency, given that voltage has a quadratic and linear relationship with dynamic power and frequency, respectively. However, as supply voltage is reduced, under the effects of manufacturing-induced parameter variations, many circuits begin to fail. The supply voltage can only be reduced (for a given clock frequency) to a minimum value (V_{min}) [9], such that the reliable operation of the chip is not compromised. Given that speculative components (including the BPU) cannot affect functional correctness, microprocessor reliability is not compromised by the existence of faults in SRAM arrays of these components. This provides an opportunity for energy savings without risking failures.

SRAM array operating voltage is optimized for a specific frequency and lowering the operating voltage would require to also lower their operating frequency in order to maintain reliable operation. However, in the case of branch predictors, this would lead to loss of fetch bandwidth and thus lower performance. Moreover, memory structures are optimized for area, and thus, low-voltage operation affects memory structures more than the logic elements. For these reasons, we assume that the operating frequency remains constant when the voltage is reduced and as a result, SRAM cells begin to malfunction [15] [14] [24]. The high failure rate of SRAM structures operate at low voltage conditions limits the extent of voltage scaling in the microprocessor. We use the experimentally derived SRAM fault rates collected from a 14nm FinFET technology x86-based microprocessor [19] to simulate faults in the predictor arrays, which are operating under high frequencies and sub-nominal voltage levels. Based on these fault rates, we evaluate the performance impact and energy savings of the BTB and 4 different state-of-the-art predictor algorithms: Bi-mode predictor [25], L-Tage predictor [1], Tournament predictor [26], and Perceptron predictor [27]. For our experiments, we use the Gem5 simulator.

More specifically, in this paper we make the following novel contributions:

i. We compare the tolerance of the four well-established branch prediction algorithms to the fault rate for each individual voltage step beyond the safe voltage. This analysis shows the trade-offs between the energy efficiency and the performance loss due to the high rate of faults occurring when the BPU operates at ultra-low voltage levels. The introduced fault rates were obtained

from real silicon measurements in a 14nm FinFET technology [19] and were simulated on top of an out-of-order highperformance microprocessor.

ii. We evaluate the performance impact of reduced size against the reduced yield caused by the low voltage. Our analysis shows that it is more efficient to partially disable predictor tables or reduce their size rather than lowering the supply voltage beyond a certain point.

iii. We showcase that protection schemes have a very small potential on further improving the energy savings. We validate this assumption by implementing a protection scheme oriented to protect speculative components. The benefits from the protection are observable only in low voltage levels, in which, however, voltage increment would deliver higher energy savings, consisting the protection pointless.

The rest of the paper is structured as follows: Section II presents a brief background and related work on this area, Section III describes the methodology and experimental setup, Section IV includes the reference BPU characterization, Section V explores energy saving opportunities, assesses their trade-offs and compares them, and Section VI concludes the paper.

II. BACKGROUND & RELATED WORK

Reduced Supply Voltage: There are many studies that focus on the effects and malfunctions of reduced supply voltage beyond nominal conditions in SRAM arrays of the microprocessor. Bacha *et al.* [6], [7] observe the manifested errors in caches of the Intel Itanium processor during the execution of benchmarks and use the error rates as a proxy to guide undervolting. Papadimitriou *et al.* in [8] presented an experimental study that aims to identify the voltage margins in two different commercial x86-64 microprocessors and present the available guardbands of these microprocessors, as well as the cache errors detected from hardware during undervolting. Studies on ARMv8-based microprocessors in [9], [10], [11], and [12] present several methods that can exploit the voltage margins and intra-die variation for energy efficiency. The authors in [13] propose micro-virus generation to stress all cache levels in the microprocessor and provide diagnosis of malfunctions beyond nominal voltage conditions. These studies propose techniques to improve energy efficiency via exploiting the pessimistic voltage margins of the CPU cores. In this paper, we investigate how different configurations of branch predictors can contribute to improve the total microprocessor's energy efficiency, when they operate in ultra-low voltage conditions.

Methods for Improved SRAM Reliability: Authors in [14], [15], and [16] propose several microarchitectural approaches to ensure the correct operation of caches in ultra-low voltage conditions. Abella *et al.* [17] proposed a disabling-and-remapping approach for cache arrays to deliver predictable performance at low voltages. Agarwal *et al.* [18] focused on yield improvements tolerating process variations. Ganapathy *et al.* [19] presented experimental results for near-threshold voltage operation of SRAM arrays using measurements taken from several wafers at 14nm FinFET process and proposed a method that estimates the operation of SRAM arrays in near-threshold voltage regions. Zimmer *et al.* [20] presented a study that uses sampling of dynamic failure metrics of SRAM arrays to quantify and analyze the effects of different assist techniques, array organization, and timing on V_{min} at design time.

Studies for Speculative Components: Hsieh *et al.* in [21] evaluate the importance of identifying hard faults that lead to performance degradation for yield improvement. Foutris *et al.* in [22] and [23] evaluate the performance impact of permanent faults in several components, such as the branch predictor, BTB, return address stack and data prefetchers. Filippou *et al.* in [24] systematically investigate the behavior of BTBs with faulty memory cells, and how it affects the total performance of the microprocessor when disabling the faulty parts of BTBs. In this paper, we compare 4 different state-of-the-art prediction algorithms (Bi-mode [25], L-Tage [1], Tournament [26], and Perceptron [27]) evaluate their performance impact due to high fault rates when they operate at ultra-low voltage levels.

Low-Power Branch Predictors: Low power branch predictors have also been proposed in the past as a mean of power-saving scheme. Banasiadi *et al.* [28] introduce an energy efficient branch prediction scheme while Kim *et al.* [29] propose low-power predictors targeting the embedded domain, which can reduce about 40% the branch predictor power consumption, and 2.3% of the total processor's power consumption. Chang in [33] propose a low-power BTB design which can achieve on average 77% energy reduction of the BTB. Parikh *et al.* in [30] propose a scheme that uses pre-decode bits to eliminate unnecessary predictor and branch target buffer accesses, which reduces 45% the power of branch predictor and 5-6% of the total microprocessor's energy dissipation. Yang and Orailoglu in [31] propose a branch identification unit to achieve early identification of incoming branch addresses according to the static extracted program information regarding the control-flow structure and the branch distance. They compare two predictors: bimod and gshare and their proposed scheme can reduce the dynamic power by 76.6%, with a 1.8% and a 2.2% increase in misprediction rate for the bimod and gshare predictor, respectively. Chaver *et al.* in [32] propose a methodology, which disable components of the hybrid direction predictor and resize the branch target buffer for reducing the energy consumption. In this work we consider similar schemes in combination with undervolting. Chatzidimitriou *et al.* in [42] assess the impact of predictors in below-nominal voltage conditions, but only consider 2 predictors, without assessing the BTB nor considering alternative improvements.

A. SRAM Array Failures

SRAM cells can malfunction due to several reasons, varying from environmental conditions, aging, supply voltage noise to process variation, systematic variability, leakage, etc. Process variation can introduce variability among cells as, depending on which transistor is affected (weaker or stronger) can result in different types of failures and different voltage thresholds for the cell. This variation is mostly observed when the SRAM arrays operate at near-threshold voltage (NTV).

SRAM failure modes can be summarized as readability, writability, read stability and hold failures [20]. Readability failure occurs when a read discharge takes more time than the offset of the sense amplifier and at the end of the cycle, there is not enough voltage difference between the bitlines. As a result, the stored value cannot be obtained. Writability failure occurs if the internal node voltage does not reach the desired write value, which causes the storing of a value to fail, while read stability failures occur when a bit cell content is accidentally flipped during a read operation. Finally, Hold failures occur when the operating voltage is below the data retention voltage, which is the minimum required to maintain the stored value. Low supply voltage can cause all of these failures, introducing faulty cells inside an SRAM array.

III. EXPERIMENTAL SETUP

A. Microarchitecture-Level Modeling

To accurately model a high-performance microprocessor and quantify the performance impact of a BPU we use the widely adopted Gem5 cycle-accurate full system simulator [37]. Gem5 supports several Instruction Set Architectures (ISAs) and two highly detailed CPU core models that allow assessment of branch predictors: an in-order core and an out-of-order core; in this work we use the latter for x86 ISA. The branch prediction unit implementation is common among the two, and provides a detailed and accurate modeling for several branch predictors, and also for all of the commonly used internal units, such as Branch Target Buffer (BTB), Return Address Stack (RAS), etc. For all of these structures, Gem5 models all of the required storage elements (predictor tables) and functional logic. That being said, SRAM cell failures can be modeled as stuck-at faults on this abstraction level.

A fault-injection framework with support of multiple permanent faults was used to simulate the SRAM cell failures. The selected tool is GeFIN [34] as it provides all functionalities required for this study. GeFIN can be fed with a series of faults to be injected during simulation. Each fault is described by the location (at bit granularity), type (transient, permanent or intermittent), model (stuck-at 0, stuck-at 1 or bit-flip), time and duration (for transient and intermittent faults) [35]. For every voltage level, we have generated series of faults that match the actual SRAM failures and then GeFIN injects these faults and executes a full system simulation. Since the simulation is cycle-accurate, we are able to capture any performance deviations (compared to the fault-free simulation) caused by the malfunctioning BPU.

Every simulation generates a very detailed set of statistics including performance metrics, microarchitectural events, etc. Combined with McPAT [36], they can provide estimations on power consumption. We have modified McPAT to manually configure the supply voltage (V_{DD}) and applied the predictor tables for all of the different predictor implementations, since McPAT only provides a hard-copied built-in Tournament predictor. Using the modified McPAT and the GeFIN results, we can estimate the power consumption of ultra-low voltage BPUs, considering all affected parameters: V_{DD} reduction, performance penalties and area.

B. BPU Configuration

The BPU consists mainly of a Branch Predictor, a Branch Target Buffer, and a Return Address Stack. It can optionally have a separate indirect branch predictor. Gem5 provides several branch predictor implementations to be selected, including Bi-Mode [25], Tournament [26], and L-Tage [1]. In this study, we explore all of these three, and also a Perceptron branch predictor [27] that was implemented in the model of Gem5 for the purposes of this work; a total number of four predictors were studied. Each one of them makes different use of its available storage and may be differently affected by the introduction of faulty cells in its SRAM arrays. A brief description of the predictor designs follows to provide a reference point to the actual studied models.

Bi-mode predictor consists of 3 predictor tables; 2 direction predictors and 1 choice predictor, implemented based on the description of [25]. The direction tables are indexed using gshare, while the choice predictor is indexed using the branch address. The predictor selects the highest match among the two, and upon retire, it updates only the one that was used and the choice component.

Tournament predictor, which is the one used in Alpha 21264 microprocessor [26], is also a hybrid predictor with a choice component to determine which of the available predictions will be used. The other two predictors are a typical global history predictor, and a local history predictor which is accompanied with a local history table. The choice predictor selects the one that provided more accurate predictions in the last requests of an entry.

L-Tage [1] predictor is more sophisticated and complex. Unlike other predictors, it uses variable history lengths and can deliver different predictions for the same branch, depending on its current history. The predictor has tagged tables to match the branch and among the available predictions, it chooses the one that comes from the largest history. The implementation of this study matches the one described in [1].

The only neural-network based predictor in our tool chain is the Perceptron [27] and it is not originally part of the Gem5; the predictor was developed to serve the scope of this work and was integrated to Gem5 simulator. Perceptron predictor uses a single artificial neuron for every branch (address indexed), which consists of a series of weights to determine the branch outcome. Each weight corresponds to a history bit and depending on its value, it increases or decreases the final decision counter. The result is considered as *not taken* if the counter is calculated to a negative value or as *taken* on a positive value. To better address branch aliasing, a small local history table is used to append some additional local history bits to the global history. This history leads the decision for the branch based on the matching perceptron.

TABLE 1: SRAM LAYOUT OF EACH BPU STRUCTURE.

Structure	Array	Dimension	Size (bits)
Bi-Mode	Taken counters	32,768 × 3	96 K
	Not-Taken counters	32,768 × 3	96 K
	Choice counters	32,768 × 2	64 K
	Total		**256 K**
Tournament	Local history table	2,048 × 15	30 K
	Local counters	32,768 × 2	64 K
	Global counters	65,536 × 2	128 K
	Choice counters	16,384 × 2	32 K
	Total		**256 K**
L-Tage	Loop table	256 × 52	13 K
	Bimodal pred	16,384 ×1	16 K
	Bimodal hysteresis	4,096 × 1	4 K
	Tage 1 & 2	1,024 × 12	12 K
	Tage 3 & 4	2,048 × 13	26 K
	Tage 5	2,048 × 14	28 K
	Tage 6	2,048 × 15	30 K
	Tage 7	1,024 × 16	16 K
	Tage 8 & 9	1,024 × 17	17 K
	Tage 10	1,024 × 18	18 K
	Tage 11	512 × 19	9.5 K
	Tage 12	512 × 20	10 K
	Total		**254.5 K**
Perceptron	Perceptrons	587 × 440	252.2 K
	Bimodal	64 × 64	4 K
	Total		**256.2 K**
BTB	Branch target buffer	4096 × 75	300 K
	Total		**300 K**

For our experimentation and for a fair comparison among these prediction algorithms, all predictors are configured to utilize an equal amount of area, which is set to approximately 256K bits. TABLE 1 summarizes the SRAM arrays that are present on each one of the 4 predictors and BTB.

C. Low-Voltage SRAMs

Throughout this work we assume that the frequency of the SRAMs remains constant (tied to the processor at 1.0 GHz) and only the voltage is regulated. As a result, we encounter massive SRAM cell failures as the voltage drops. In order to calculate the number and location of the failing SRAM cells for every voltage level, we employ the methodology proposed in [19], to generate the Fault Maps (*fmaps)*. Each map contains a voltage threshold for every cell in the array, below which the cell stops operating and behaves as a stuck-at 0 or 1. This reduces the number of functional cells of the array as the voltage is reduced.

TABLE 2: LIST OF EXECUTED WORKLOADS.

SPEC® CPU2006 Benchmarks			
473.astar	403.gcc	456.hmmer	458.sjeng
401.bzip2	459.gemsFDTD	470.lbm	482.sphinx3
436.cactusADM	445.gobmk	462.libquantum	465.tonto
454.calculix	435.gromacs	444.namd	483.xalancbmk
416.gamess	464.h264ref	471.omnetpp	434.zeusmp

To better capture the process variation, we generated 10,000 fmaps, out of which, we selected 50 representatives for our experiments.

For every fmap, we locate the faulty cells, and for each one of them we generate a permanent fault for the GeFIN fault injector. This results to 5 fault lists for every fmap, corresponding to 5 different V_{DD} settings, which range from 0.6 of the nominal Voltage (V_{nom}) down to 0.5 Vnom, in a step of 0.025 Vnom. TABLE 3 shows the average number of faults that were generated for each one of the predictors and the BTB, along with the remaining functional yield of the structure.

D. Benchmarks

The workloads of our study come from SPEC® CPU2006 benchmark suite. 20 benchmarks were used in total, which are listed in TABLE 2. Due to the long simulation time of the detailed CPU cores and the large number of simulations required (more than 80,000 simulations), SimPoint [39] was used to reduce the simulation duration. For each of the workloads, we used SimPoint 3.0 to profile simpoints of 50 million instructions. Out of the generated simpoints, for each benchmark we selected the one with the largest weight to perform our study. A warming-up period of 10 million instructions was used after which we start measuring the simulation statistics. As a result, for every benchmark, 40M instructions of the largest weight simpoint were eventually used to deliver the results of this study.

IV. ULTRA-LOW VOLTAGE EFFECTS ON BPUS

The presence of faulty cells negatively affects the operation of branch predictors. It reduces the ability to train according to the executed workload as it cannot update all of the data related to a prediction. The fact, however, that most of the studied predictors are hybrid allows some level of tolerance, as predictions are calculated in multiple arrays and a fault in one of them can be covered by a prediction from alternative source. If, for example, a stuck-at fault on a counter constantly gives a "not-taken" prediction, which may be incorrect, the choice component will identify the poor performance and fallback to the next

TABLE 3: NUMBER OF FAULTY CELLS AND PERCENTAGE OF FUNCTIONAL CELLS (AVG CELLS) FOR THE FIVE STRUCTURES ON EACH VOLTAGE STEP.

Voltage level	Bi-Mode		Tournament		L-Tage		Perceptron		BTB	
	# Faults	Av. Cells	# Faults	Av. Cells	# Faults	Av. Cells	# Faults	Av. Cells	# Faults	Av. Cells
0.600 V_{nom}	1.1	100%	1.2	100%	1.5	100%	4.3	100%	1.9	100%
0.575 V_{nom}	120	99.95%	113	99.96%	113	99.96%	131	99.95%	138	99.96%
0.550 V_{nom}	5,913	97.74%	5,826	97.78%	5,775	97.78%	5,865	97.76%	6,943	97.74%
0.525 V_{nom}	82,403	68.57%	81,495	68.91%	80,865	68.97%	81,417	68.97%	96,670	68.53%
0.500 V_{nom}	166,138	36.62%	159,382	39.20%	169,094	35.12%	189,103	27.92%	197,465	35.72%

available predictor. On the other hand, if a choice component entry is stuck-at and forces the choice of one particular predictor at all time, it will reduce the accuracy for that entry to the accuracy of the used predictor. These scenarios will both have marginal impact, as the predictions will fall back to the secondary available predictor and the performance loss will be limited to their difference for that particular entry.

In order to have a noticeable impact on the performance, a combination of faults must occur. Following the previous example, if the choice component forces the use of a predictor that also happens to suffer from a fault, then this will certainly lead to multiple mispredictions. As the arrays are quite large in size, these cases only happen in lower voltage configurations, where the SRAMs start to fail massively. On the other hand, cell failures in the BTB start impacting the performance even for single faults, as the tag or the target becomes unavailable, which will certainly cause mispredictions. To better assess the impact of each particular component in the BPU, we have run simulations with faults only in the BP part and separate simulations in all BPU (BP and BTB), as the usage of BTB is highly dependent on the BP. This allows us to estimate which portion of the slowdown comes from the BP and which from the BTB.

A. *Branch Predictor Accuracy Loss*

Because of the different design, each of the four predictors is expected to have different level of performance degradation when undervolted. Fig. 1 presents how the accuracy is affected in the four predictors measured in mispredictions per kilo instructions (MPKI). Among the four predictors, perceptron stands out as it appears to be the more tolerant to faults and follows a trend of its own. Its neural network nature makes it less sensitive to the presence of faults. Even thought that it starts reporting accuracy degradation earlier than the rest of the predictors, it manages to keep all voltage levels close to each other in a narrower range, compared to the other predictors, where low voltage settings lead to massive accuracy degradation. Perceptron reports an average accuracy loss of 1.3x at 0.55 V_{nom} while interestingly, the accuracy loss in both 0.525 V_{nom} and 0.5 V_{nom} is approx. 2.6x. The other predictors suffer from smaller losses at 0.55 V_{nom}, but report average slowdown greater than 5x and 13x at 0.525 and 0.5 V_{nom} respectively.

Bi-Mode and Tournament predictors report very similar behavior, with approximately 9x accuracy loss at 0.525 V_{nom} and 13x at 0.5 V_{nom}. L-Tage on the other hand performs better in 0.525 V_{nom} with only 5x accuracy loss, but is massively affected in 0.5 V_{nom} with 25x accuracy loss. The results are aligned to the observation that applications with neural network structures are less affected from faults and should be considered as an option on a system that is designed to operate under these conditions.

B. *Performance Impact*

Predictor accuracy is sufficient to present how the component is affected, but in order to assess these results to power and energy (which is the reason we perform the undervolting), we need to capture the system-level effects. In an out-of-order CPU model, the BPU guides instruction fetching, and possible mispredictions cause pipeline flushes and fetch redirection, which can be further stalled by the instruction cache. Some level of the accuracy loss can be potentially covered by the complexity of the pipeline, as not all mispredictions cause the same penalty. Fig. 2 presents how each predictor affects the overall system performance, in terms of Instructions per Cycle (IPC). The graph shows how the accuracy loss is degraded when considering the system level.

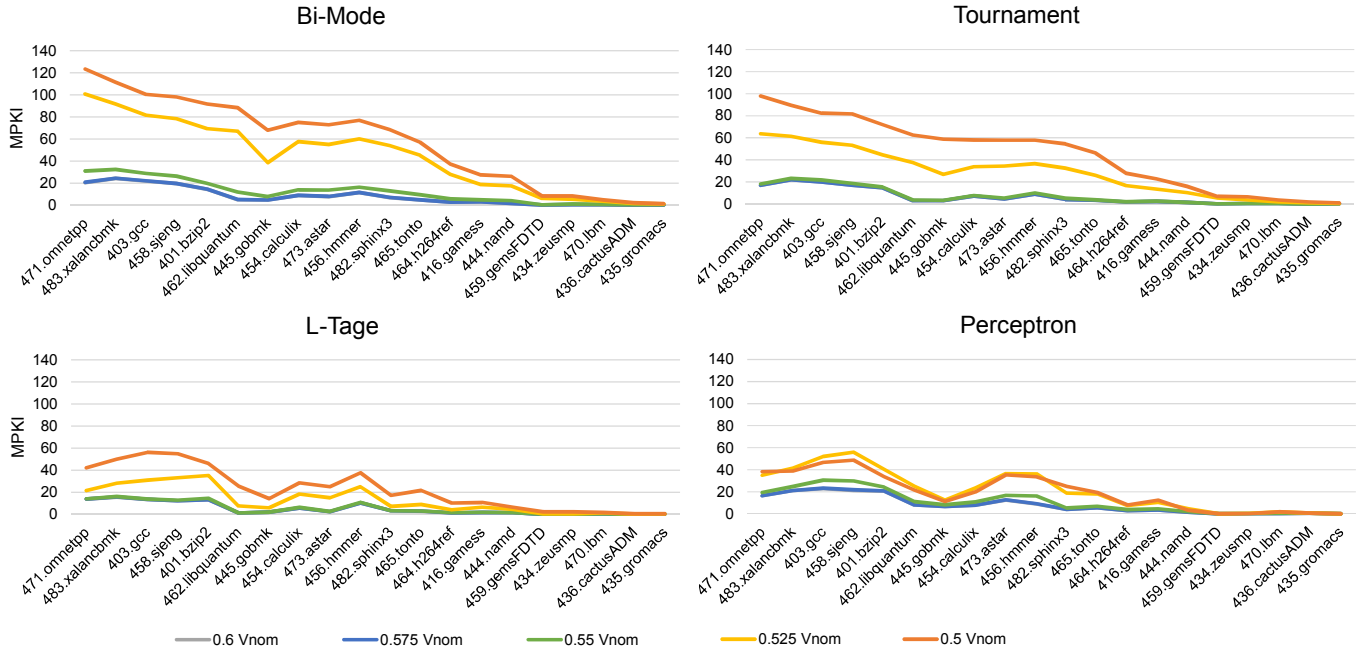

Fig. 1: Predictor accuracy measured in Mispredictions per Kilo Instructions, for all predictors and 5 voltage levels.

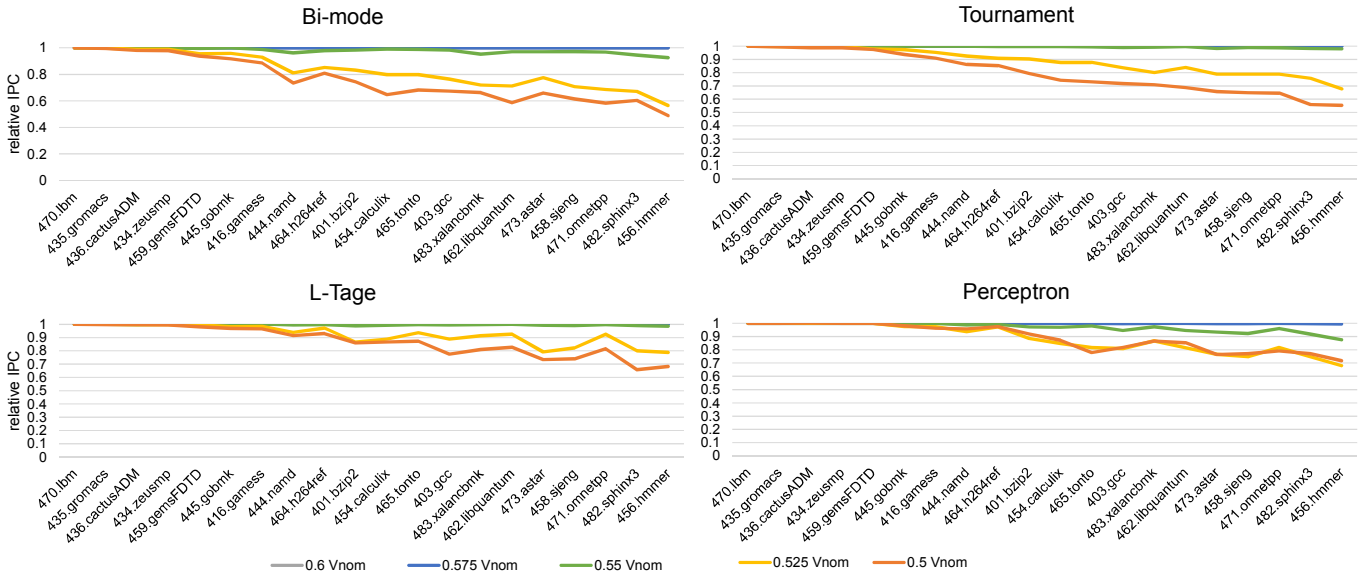

Fig. 2: Instructions per Cycle, normalized to nominal voltage IPC, expresses the system slowdown.

Interestingly, only Bi-Mode and Perceptron predictors report noticeable deviations in 0.5 V_{nom}, as L-Tage and Tournament graph lines are unified with 0.6 and 0.575 V_{nom} lines. Unlike the accuracy loss, in the performance we observe a maximum slowdown of 2x and even L-Tage, which reported average 25x accuracy loss, reports a maximum IPC slowdown less than 2x. The smallest average slowdown at 0.5 V_{nom} is achieved by Perceptron predictor, while at 0.525 V_{nom} by L-Tage. At 0.5 V_{nom} both Tournament and L-Tage report an average slowdown of less than 1%, while Perceptron and Bi-Mode report a performance loss >3%.

C. BTB Impact

Unlike branch predictors, the BTB stores data that are used as a whole to guide the fetching stage. Any change in these data directly leads to a false address or a misprediction (due to tag mismatch and therefore, target unavailability). Fig. 3 shows the average IPC slowdown when the BTB operates in ultra-low voltage conditions along with the BP, for all four predictors. We can notice that the performance impact caused by the BTB is so severe that there is seemingly no difference among the four configurations. The average maximum difference observed between the four implementations is <0.5% at 0.6 to 0.55 V_{nom}, while the difference remains below 2.2% in 0.525 and 0.5 V_{nom}.

In contrast to the undervolted predictors alone, the BTB starts to have noticeable slowdown at 0.575 V_{nom}. Interestingly, the level of slowdown on each voltage setting matches the one of the next voltage setting of the BP only, while the voltage of 0.525 V_{nom} delivers identical slowdown with 0.5 V_{nom}.

D. Process Variability

Each benchmark was simulated with ultra-low voltage in all 50 fmaps, which allows us to observe how the results are different among these variants, when all other parameters (workload, design, and voltage) remain constant. Fig. 4 shows the level of variability on two corner-case (in terms of variability) workloads, 459.gemsFDTD and 458.sjeng. 459.gemsFDTD reports very small performance differences in average, however, as the variability graphs in Fig. 4 present, we can see that Bi-Mode and Tournament predictors report significant differences among the 50 fmaps. L-Tage also reports large differences (wide curve) only for the 0.5 V_{nom} case. The rest of the voltage levels report almost no variability. Perceptron predictor proves to deliver the lowest level of variation, as its deviation U curves are very narrow in all benchmarks. This means that all of 50 chips report similar slowdown, which allows easier characterization. The 458.sjeng benchmark delivers similar results in all predictors, minimizing the effects on process variation. An interesting observation is that there are workloads, such as 458.sjeng, that hide process variability.

V. Power, Area & Performance Trade-offs

A. Power Consumption

Using a modified version of McPAT, which allows us to use custom V_{DD} and BP arrays, we can estimate the power consumption of the BPU. For both BPs and BTB, we measure the

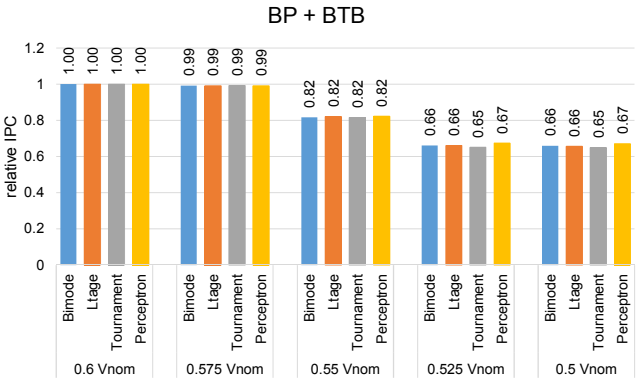

Fig. 3: Relative IPC for BPs and BTB under low-voltage conditions.

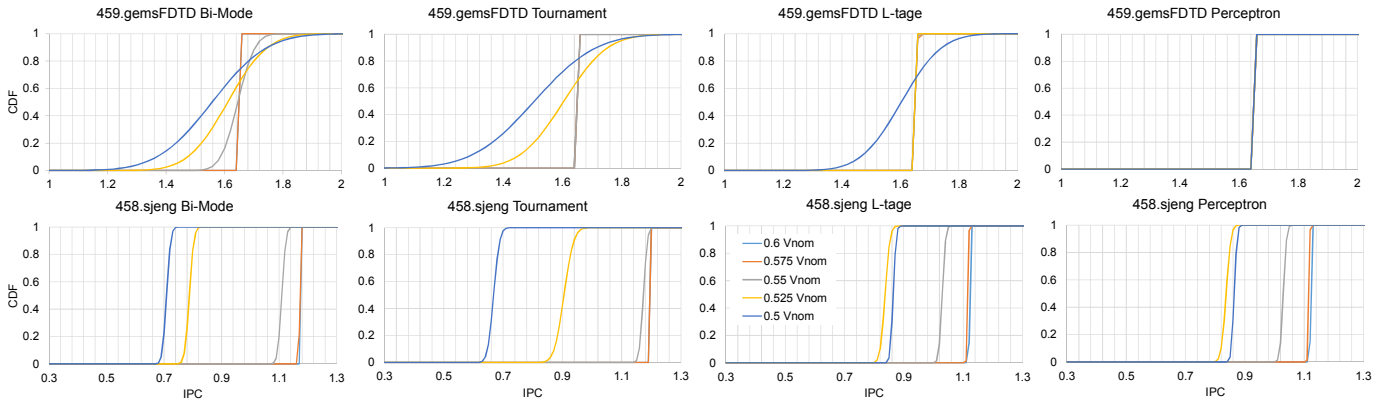

Fig. 4: IPC Cumulative distribution graph for 459.gemsFDTF and 458.sjeng benchmarks for the four branch predictors.

power consumption difference in all 5 voltage levels, compared to the power consumption of the nominal voltage setting. Fig. 5 shows how the power consumption is calculated among the 5 voltage settings. As expected, lower voltage delivers lower power consumption. However, there are a few other attributes that also affect the power consumption, relative to the misprediction rate (which tends to decrease as the voltage decreases). The increased misprediction rate comes along with additional power consumption, which is the reason we observe cases where 0.525 V_{nom} consumes more power than 0.5 V_{nom}. An overall 60.2% BPU power reduction is achieved when voltage is set to 0.6 V_{nom} while it can reach 69.7% at 0.5 V_{nom}. Although the difference between sequential steps is not constant for the 4 predictors, it can be quantified as an average 7% power reduction for each 0.025 V_{nom} voltage step.

B. *Undersizing versus Undervolting*

Considering the number of faulty cells in each voltage level, as presented in TABLE 3, we can see that the first voltage step from 0.6 V_{nom} to 0.575 V_{nom} only reduces the number of functional cells about 0.05%, while the reduction in last step is as high as 32%. Yet, each of these steps deliver approximately the same amount of static power savings, which is 7%. In terms of area, an SRAM with 10% smaller size than the reference 256K budget would consume approximately 7% less power. If the size is reduced by 10% instead of reducing the voltage from 0.55 V_{nom} to 0.525 V_{nom} then power savings would be close but we would have 19% more functional cells compared to the undervolting. Fig. 6 shows the SRAM cells available for every power step, for both undervolting and undersizing.

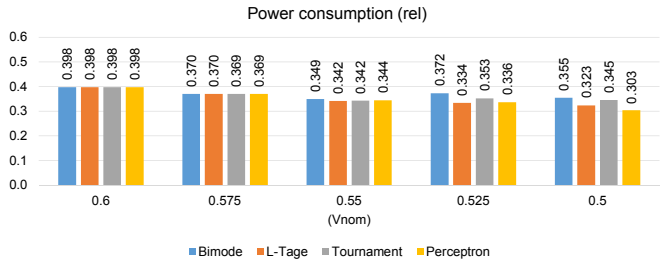

Fig. 5: Relative power consumption of the different BPU configurations.

Unlike undervolting, size reduction cannot be done during system operation. In addition, some of the predictors also require the number of entries to be power of two. To enable undersizing of the BPU during system operation, we disable SRAM components that correspond to 20% of the predictor area size. For example, in the case of L-Tage this is achieved by completely disabling 2 TAGE components. However, this was not possible in all cases. Some predictors had no candidate SRAMS of size and we thus, had to split one or more predictor arrays in two parts, such as the new smaller chunks will be equal to 20% of the total predictor size budget. Disabling of these SRAMs allowed the system to operate with a 20% size reduction in the BPU. The performance degradation reported on average is: 0.15% for Bi-Mode, 0.1% for Tournament, 0.5 % for L-Tage and 0.05% for Perceptron. Interestingly, we can see that the reduction of functional cells delivers lower performance degradation compared to undervolting, as the reduced size results in less data, compared to wrong data providing by the undervolting. We consider the undersizing to be part of a mixed solution along with 0.575 V_{nom} in the BPU. Similar approach has been presented in [32].

C. *Protection Mechanism*

Although the BPU operation is completely speculative and cannot harm the correctness of the microprocessor, it can still benefit from a protection mechanism which can limit the performance degradation caused by the faulty SRAM cells in low voltage conditions. Previous studies [14], [15] have shown that

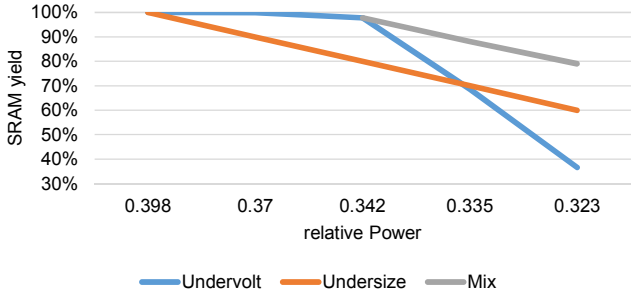

Fig. 6: Percentage of available cells per power step for undervolting, undersizing and combination

traditional ECC protection schemes are not particularly effective when referring to near-threshold SRAM arrays. Since the BPU can tolerate faults, and in order to keep the protection overhead low (as the goal is a better power/performance tradeoff), we implement a protection scheme that uses protection caches that exploit the property of locality [38]. This means that a part of the SRAMs will remain unprotected and there will be no way of detecting faults. However, we can use existing processor infrastructure to provide a more efficient detection scheme, which will allow detection of faults in unprotected SRAM rows.

Upon update (at retire pipeline stage), the BPU is required to generate the updated data for the branch prediction entries that were affected as well as for all of the entries that belong to the same SRAM row, which is called "metadata". This metadata is held in a separate table known as Branch Repair Table (BRT), which is also used to facilitate control flow in a case of a misprediction. We can exploit the availability of BRT to locate faulty cells in the SRAM, as a cheap fault detection solution. More precisely, the correct metadata can be compared against what was eventually stored in the SRAM row and count the number of visible faults. If this number is below (or equal) to what the mechanism can correct, then the row is placed in the cache and will be corrected on its next access. At this point there may be faults that were masked due to the current data in the SRAM row (stuck-at 0 faults with 0 value or stuck-at 1 with 1 as value). As long as these faults do not require correction, they can be disregarded.

If, however, the number of detectable faults is beyond the correction capability of the ECC, the row is marked as invalid. These rows are not checked for faults again, as the mechanism does not keep track of the faulty cells, and since they are uncorrectable, they may falsely invoke usable entries in the ECC cache upon their next use.

D. Protection Capability & Overhead

To select the appropriate level of protection we must look at the number of faults that are typically detected in the workloads. Fig. 7 presents the cumulative distribution of faults that were observable through our mechanism for all different SRAM arrays. For the majority of the predictors, a protection with 4-bit correction would be sufficient to repair more than half of the detections, even in the lowest voltage levels. Perceptron is the predictor with the larger fault detections, which can be attributed to the fact that its SRAM array consists of a very large number of columns, and thus multiple faulty cells can coexist in the same SRAM row, even at 0.5 V_{nom}; the same applies for BTB. Unlike the predictor tables, BTB stores tags and addresses, where the existence of a single fault consists the row unusable. Any mechanism smaller than duplication would fail to cover the requirements of BTB.

All BPU configurations use the same BTB, which is 307,200 bits large. The SRAM layout is 4096 x 75. To calculate the additional area required by the protection mechanism, we need to consider the ECC cache size as well as the SRAM row

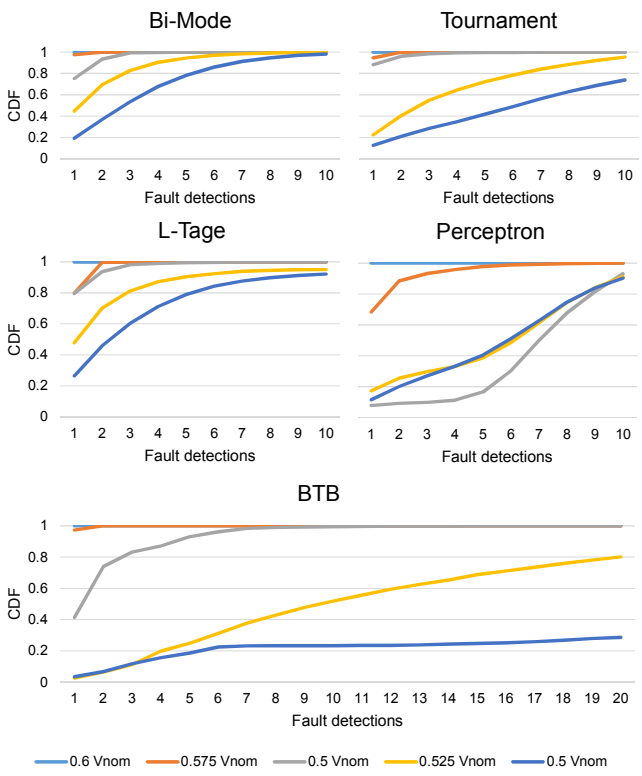

Fig. 7: Cumulative distribution number of faults detected by proposed mechanism per voltage level.

valid bits. Since the cache will use duplication for protection, each entry will require 75 bits and an additional 5 bits for the tag, which sum up to 80 bits. For a total of 64 entries, the additional space required by the ECC cache is 5,120 bits. Added to 4,096 invalid bit flags, the total overhead for the BTB is 9,216 bits, which equals to 3% of the BTB area.

TABLE 4 summarizes the area overhead for each of the predictors. The protection varies in size among the different BPU components as it highly relies on their layout. L-Tage is affected the most, as the area overhead for the per-Tage SRAM protection sums up to 7.7%. Bi-Mode and Perceptron have the smallest overheads because of their single SRAM implementations. Fig. 8 illustrates the improved IPC due to the added protection. We can see how voltage levels of 0.55 to 0.5 V_{nom} have significantly improved up to 12% of nominal IPC (see Fig. 3).

E. Energy Savings

The BPU power consumption can is estimated as 20% of the whole microprocessor power when operated in nominal

TABLE 4: PROTECTION OVERHEAD.

Structure	Size	Protection size	Overhead
Bi-Mode	256 K	5,376	2.0%
Tournament	256 K	7,936	3.0%
L-Tage	254.5 K	20,224	7.7%
Perceptron	256.2 K	2,133	0.8%
BTB	300 K	9,216	3.0%

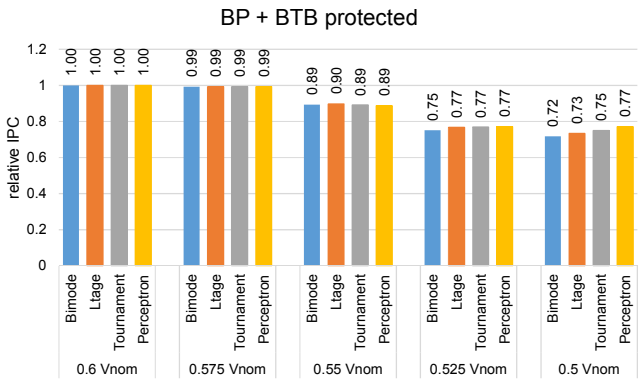

Fig. 8: Relative average IPC with protection.

voltage [5], [32], [40], [41]. Strictly based on their sizes, this is reduced approximately to 9.5% for the BP and 10.5% for the BTB. To quantify the energy consumption, we need to take into consideration both the reduced power and performance degradation. Since the largest part of the system's power remains unchanged, the energy consumption is only reduced as long as the penalty of the overall slowdown does not cover the savings of the power difference. Fig. 9 presents the processor energy consumption difference compared to nominal operation, for all configurations that were assessed in this work. The results can be grouped in: undervolted predictors (BP names only), undervolted including BTB (BP names +BTB), undervolting plus undersizing (BP name +SZ), protected (BP name +PROT). For each voltage configuration, different colors where used to easily show how voltage reduction can affect energy savings.

Due to the large size of the BTB, the highest energy savings are observed when both BP and BTB are undervolted. Peak energy saving is 12.04% for all predictors at 0.6 V_{nom} for both BP and BTB. The performance degradation of the BTB in the remaining voltage settings consists any level of undervolting below 0.6 V_{nom} less efficient. When only the BPs are undervolted, we can see that the lowest energy is consumed at 0.575 V_{nom}, but still fall behind the undervolted BPU that includes BTB. The dotted line marks the energy in nominal voltage. All configurations below that consume less energy. Interestingly we can see that we did not find a single case where 0.525 V_{nom} reduced the energy consumption, while we can see that protected arrays can still save energy at 0.5 V_{nom}. However, all protected cases fall behind configurations without protection in higher voltage, which shows that a protection mechanism cannot improve the energy savings. Additionally, we can also see that undersizing does not provide energy improvements, as the decreased power falls behind the processor slowdown, which is marginal but still larger than the power improvement.

VI. CONCLUSION

We presented an extensive experimental study of how different configurations of branch predictors (Bi-Mode, L-Tage, Tournament, and Perceptron) can contribute to improve the total microprocessor's energy consumption, when they operate in ultra-low voltage conditions. We show that the trade-offs between power reduction and the performance loss can affect the processor's energy consumption, proving that undervolting is not always beneficial. The introduced fault rates were obtained from real silicon measurements in a 14nm FinFET technology and were simulated on an x86 model of the Gem5 cycle-accurate simulator. We also compared against reduced size and protected arrays to additionally reduce energy consumption and we have shown that none of these techniques can push the savings any further than what is achieved when both BP and BTB is undervolted in 0.6 V_{nom}, where a peak 12% energy saving can be achieved.

ACKNOWLEDGMENT

This work is funded by the H2020 Framework Program of the European Union through the UniServer Project (Grant Agreement 688540) – http://www.uniserver2020.eu. AMD, the AMD Arrow logo, Radeon and combinations thereof are trademarks of Advanced Micro Devices, Inc. SPEC CPU® is a registered trademark of the Standard Performance Evaluation Corporation. See www.spec.org for more information.

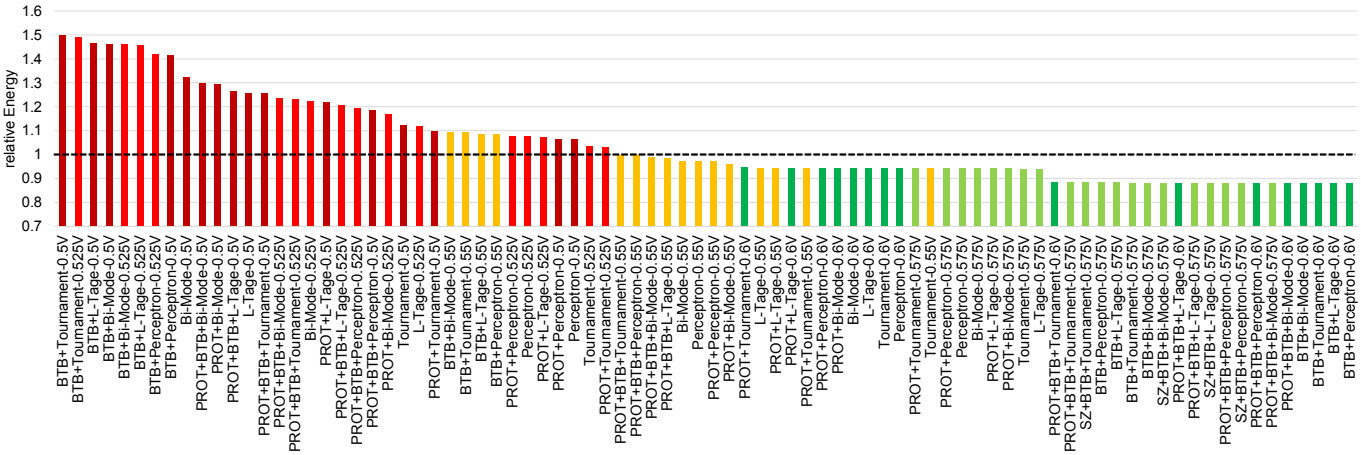

Fig. 9: Energy consumption (relative) of the different configurations.

VII. REFERENCES

[1] A. Seznec, "A 256 Kbits L-TAGE branch predictor," Journal of Instruction-Level Parallelism (JILP) Special Issue: The Second Championship Branch Prediction Competition (CBP-2), vol. 9, 2007.

[2] A. Baniasadi and A. Moshovos, "SEPAS: A highly accurate energy-efficient branch predictor," Proc. Intl. Symp. on Low Power Electronics and Design (ISLPED), 2004.

[3] S. Kim, E. Jo, and H. Kim, "Low Power Branch Predictor for Embedded Processors," Proc. IEEE Intl. Conference on Computer and Information Technology (ICCIT), 2010.

[4] J. Haj-Yihia, A. Yasin, Y. B. Asher, and A. Mendelson, "Fine-Grain Power Breakdown of Modern Out-of-Order Cores and Its Implications on Skylake-Based Systems," in ACM Transactions on Architecture and Code Optimization (TACO) , Volume 13 Issue 4, 2016.

[5] D. Parikh, K. Skadron, Y. Zhang, M. Barcella, and M. Stan, "Power Issues Related to Branch Prediction," Intl. Symp. on High-Performance Computer Architecture (HPCA), 2002.

[6] A. Bacha and R. Teodorescu, "Dynamic reduction of voltage margins by leveraging on-chip ECC in Itanium II processors," Proc. Intl. Symp. on Computer Architecture (ISCA), 2013.

[7] A. Bacha and R. Teodorescu. "Using ECC Feedback to Guide Voltage Speculation in Low-Voltage Processors," Proc. IEEE/ACM Intl. Symp. on Microarchitecture (MICRO), 2014.

[8] G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, C. Magdalinos, and D. Gizopoulos, "Voltage Margins Identification on Commercial x86-64 Multicore Microprocessors," Proc. 2017 IEEE Intl. Symp. on On-Line Testing and Robust System Design (IOLTS), 2017.

[9] G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, D. Gizopoulos, P. Lawthers, and S. Das, "Harnessing Voltage Margins for Energy Efficiency in Multicore CPUs," Proc. IEEE/ACM Intl. Symp. on Microarchitecture (MICRO), 2017.

[10] M. Kaliorakis, A. Chatzidimitriou, G. Papadimitriou, and D. Gizopoulos, "Statistical Analysis of Multicore CPUs Operation in Scaled Voltage Conditions," IEEE Computer Architecture Letters (CAL), 2018.

[11] G. Karakonstantis, *et al.*, "An Energy-Efficient and Error-Resilient Server Ecosystem Exceeding Conservative Scaling Limits", ACM/IEEE Design, Automation, and Test in Europe (DATE), 2018.

[12] K. Tovletoglou, *et al.*, "Measuring and Exploiting Guardbands of Server-Grade ARMv8 CPU Cores and DRAMs", IEEE/IFIP Intl. Conference on Dependable Systems and Networks (DSN), 2018.

[13] G. Papadimitriou, A. Chatzidimitriou M. Kaliorakis, Y. Vastakis, and D. Gizopoulos, "Micro-Viruses for Fast System-Level Voltage Margins Characterization in Multicore CPUs," Proc. IEEE Intl. Symp. on Performance Analysis of Systems and Software (ISPASS), 2018.

[14] C. Wilkerson, H. Gao, A. R. Alameldeen, Z. Chishti, M. Khellah, and S. L. Lu, "Trading off Cache Capacity for Reliability to Enable Low Voltage Operation," Proc. Intl. Symp. on Computer Architecture (ISCA), 2008.

[15] Z. Chishti, A. R. Alameldeen, C. Wilkerson, W. Wu, and S.-L. Lu, "Improving cache lifetime reliability at ultra-low voltages," Proc. IEEE/ACM Intl. Symp. on Microarchitecture (MICRO), 2009.

[16] H. Duwe, X. Jian, D. Petrisko, and R. Kumar, "Rescuing uncorrectable fault patterns in on-chip memories through error pattern transformation," Proc. Intl. Symp. on Computer Architecture (ISCA), 2016.

[17] J. Abella, J. Carretero, P. Chaparro, X. Vera, and A. González, "Low Vccmin fault-tolerant cache with highly predictable performance," Proc. EEE/ACM Intl. Symp. on Microarchitecture (MICRO), 2009.

[18] A. Agarwal, B. C. Paul, H. Mahmoodi, A. Datta, and K. Roy, "A process-tolerant cache architecture for improved yield in nanoscale technologies," in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, Vol.: 13, No.: 1, 2005.

[19] S. Ganapathy, J. Kalamatianos, K. Kasprak, and S. Raasch, "On Characterizing Near-Threshold SRAM Failures in FinFET Technology," Proc. Design Automation Conference (DAC), 2017.

[20] B. Zimmer, *et al.*, "SRAM Assist Techniques for Operation in a Wide Voltage Range in 28-nm CMOS," in IEEE Transactions on Circuits and Systems, Vol.: 59, No.: 12, 2012

[21] T. Y. Hsieh, *et al.*, "Tolerance of Performance Degrading Faults for Effective Yield Improvement," Proc. Intl. Test Conference (ITC), 2009.

[22] N. Foutris, D. Gizopoulos, J. Kalamatianos, and V. Shridharan, "Assessing the Impact of Hard Faults in Performance Components of Modern Microprocessors," Proc. Intl. Conference on Computer Design (ICCD), 2013.

[23] N. Foutris, A. Chatzidimitriou, D. Gizopoulos, J. Kalamatianos, and V. Sridharan, "Faults in data prefetchers: Performance degradation and variability," Proc. VLSI Test Symp. (VTS), 2016.

[24] F. Filippou, G. Keramidas, M. Mavropoulos, and D. Nikolos, "Recovery of Performance Degradation in Defective Branch Target Buffers," Proc. IEEE Intl. Symp. on On-Line Testing and Robust System Design (IOLTS), 2016.

[25] C. C. Lee, I. K. Chen, and T. N. Mudge, "The bi-mode branch predictor", Proc. ACM/IEEE Intl. Symp. on Microarchitecture (MICRO), 1997.

[26] R. E. Kessler, "The Alpha 21264 Microprocessor," IEEE Micro, vol. 19, no. 2, 1999.

[27] D. A. Jiménez and C. Lin, "Dynamic Branch Prediction with Perceptrons", Proc. Intl. Symp. on High-Performance Computer Architecture (HPCA), 2001.

[28] A. Baniasadi and A. Moshovos, "SEPAS: A highly accurate energy-efficient branch predictor," Proc. Intl. Symp. on Low Power Electronics and Design (ISLPED), 2004.

[29] S. Kim, E. Jo, and H. Kim, "Low Power Branch Predictor for Embedded Processors", Proc. IEEE Intl. Conference on Computer and Information Technology, 2010.

[30] D. Parikh, K. Skadron, Y. Zhang, M. Barcella, and M. R. Stan, "Power Issues Related to Branch Prediction", Proc. Intl. Symp. on High-Performance Computer Architecture (HPCA), 2002.

[31] C. Yang and A. Orailoglu, "Power Efficient Branch Prediction through Early Identification of Branch Addresses", Proc. Intl. Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES), 2006.

[32] D. Chaver, L. Pinuel, M. Prieto, F. Tirado, and M. C. Huang, "Branch prediction on demand: an energy-efficient solution", Proc. Intl. Symp. on Low Power Electronics and Design (ISLPED), 2003.

[33] Y. J. Chang, "Lazy BTB: Reduce BTB Energy Consumption using Dynamic Profiling", Proc. Design Automation Asia and South Pacific Conference (ASP-DAC), 2006.

[34] M. Kaliorakis, S. Tselonis, A. Chatzidimitriou, and D. Gizopoulos, "Differential fault injection on microarchitectural simulators," Proc. IEEE Intl. Symp. on Workload Characterization (IISWC), 2015.

[35] A. Chatzidimitriou and D. Gizopoulos, "Anatomy of microarchitecture-level reliability assessment: Throughput and accuracy," Proc. IEEE Intl. Symp. on Performance Analysis of Systems and Software (ISPASS), 2016.

[36] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Joupp, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," Proc. IEEE/ACM Intl. Symp. on Microarchitecture (MICRO), 2009.

[37] N. Binkert, *et al.* , "The Gem5 simulator," in ACM SIGARCH Computer Architecture News, vol. 39, no. 2, May, 2011.

[38] N. Axelos, K. Pekmestzi, and D. Gizopoulos, "Efficient Memory Repair Using Cache-Based Redundancy", in IEEE Transactions on VLSI Systems, Vol. 20, December, 2012.

[39] E. Perelman, G. Hamerly, M. V. Biesbrouck, T. Sherwood, and B. Calder, "Using SimPoint for accurate and efficient simulation", Proc. ACM SIGMETRICS Intl. conference on Measurement and modeling of computer systems, June, 2003.

[40] M. Co, D. Weikle, and K. Skadron, "A Break-Even Formulation for Evaluating Branch Predictor Energy," Proc. Workshop on Complexity-Effective Design (WCED), 2005.

[41] M. C. Huang, D. Chaver, L. Pinuel, M. Prieto, and F. Tirado, "Customizing the Branch Predictor to reduce Complexity and Energy Consumption," in IEEE Micro Sept/Oct 2003.

[42] A. Chatzidimitriou, G. Papadimitriou, D. Gizopoulos, S. Ganapathy, and J. Kalamatianos "Analysis and Characterization of Ultra Low Power Branch Predictors", Proc. IEEE Intl. Conference on Computer Design (ICCD), 2018.