Evaluating various branch-prediction schemes for biomedical-implant processors

Christos Strydis, Georgi N. Gaydadjiev
Computer Engineering Lab, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands
E-mail:christos@ce.et.tudelft.nl

***Abstract*— This paper evaluates various branch-prediction schemes under different cache configurations in terms of performance, power, energy and area on suitably selected biomedical workloads. The benchmark suite used consists of compression, encryption and data-integrity algorithms as well as real implant applications, all executed on realistic biomedical input datasets. Results are used to drive the (micro)architectural design of a novel microprocessor targeting microelectronic implants. Our profiling study has revealed that, under strict or relaxed area constraints and regardless of cache size, the ALWAYS TAKEN and ALWAYS NOT-TAKEN static prediction schemes are, in almost all cases, the most suitable choices for the envisioned implant processor. It is further shown that bimodal predictors with small Branch-Target-Buffer (BTB) tables are suboptimal yet also attractive solutions when processor I/D-cache sizes are up to 1024KB/512KB, respectively.**

I. Introduction

The field of biomedical microelectronic implants is relatively new (approximately 50 years). There are many widely known examples of such devices with the most famous being the implantable pacemaker that extended the lifetime of many patients in the years it has be used in the heart surgery practice. According to data from the American Heart Association [1] in USA alone approximately 180,000 pacemakers and 91,000 defibrillators have been implanted for the year 2005. In addition to those, biomedical implants are now being applied for an expanding range of medical applications, e.g. blood-glucose level detection, paralyzed-muscle restoration and deep-brain neuromodulation. This trend, combined with the observation that the life expectancy (and the average age of the world population) will keep growing, lead us to believe that more implant applications will emerge in the near future. Some will be mainly targeting improved quality of life and reduction of the clinical healthcare costs. A possible future scenario is where the hospital stay can be shortened by the application of tiny implants that are monitoring or assisting the patient body in various ways while people are moving around unhindered and performing their everyday tasks. Implants will monitor and log biological data in-vivo and, depending on the patient disease, will act on those readouts by regulating some physiological quantity in the body or inform the treating doctor in case of danger. A good example is releasing insulin to the blood stream when high blood-glucose levels are detected.

It has come to our attention that the biomedical implant market is mature enough to embrace implants relying on the technological innovations of late [2]. In recent years implant designers are slowly changing their approach from mostly analog, simple, full-custom design towards multifunctional systems relying on digital process and control units. The earlier custom-designed, application-specific (e.g. Finite-State-Machine-based) systems [3]–[5] are becoming more generic and software-based ($\mu P/\mu C$-based) ones [6]–[8]. This trend has been well-studied [2] and is depicted in Fig.1. What we can learn from this figure is that implant-processor design is becoming more structured than it used to be in the past and that in the near future implant functionality will heavily rely on software (written in some high-level, established language like C) rather than pure, hardwired circuitry.

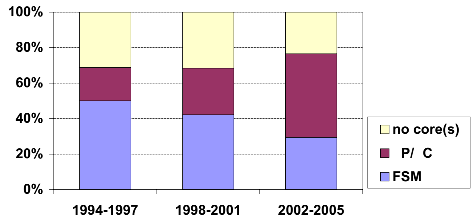

Fig. 1. Relative distribution of implant-core architecture types over the last 12 years (Source: [2]).

The anticipated rapid expansion of implant applications in the years to come, combined with the adoption of digital processing elements and software, calls for a formal, standardized way of designing future implant architectures. Our long-term goal is the design of a novel, minimalistic, highly reliable, low-power processor suitable for a large subset of well known as well as novel biomedical applications as the ones mentioned above. We are currently in the process of defining the architecture of our digital processor. In this paper, we investigate different branch-prediction alternatives under varying L1 I- and D-cache configurations. We perform tests against metrics of performance, power, energy and area. We, then, select the candidates with the best characteristics for the targeted application domain. We, thus, offer insights on the design and implementation of the branch-prediction subsystem of our targeted processor. Concisely, the contributions of this work are:

- Careful evaluation of various branch-prediction schemes under performance, power, energy, area constraints using different processor cache configurations and a collection of representative biomedical workloads;
- Precise analysis of the quantitative data for the evaluated branch-prediction schemes for current and future implant processors; and
- A sound methodology and toolset for selecting best-suited

branch-prediction mechanisms for different biomedical (or other) workloads.

The rest of the paper is organized as follows: section II gives an overview of related works in the field. Section III provides the details of our selected input datasets, application benchmarks as well as the profiling testbed used. Section IV contains, in detail, the findings of this evaluation work. Overall conclusions and future work are drawn in section V.

II. Related work

So far, extensive work has been put in identifying and profiling common applications to be executed on the targeted implant architecture. Algorithms for lossless data compression [9], symmetric-key encryption [10] and data integrity as well as representative real-world applications have been evaluated and suitable candidates have been isolated. Moreover, a carefully selected benchmark suite for microelectronic implants has been proposed [11], based on the profiled applications, to guide and assist future implant design. This benchmark suite has been shown to offer diverse program behaviors and, thus, be able to capture corners of our design space. We build on our previous work by using it in our following exploratory study on suitable cache organizations.

Besides, a significant body of prior work has been published on branch-prediction behavior with respect to traditional metrics (e.g. accuracy, performance) as well as recent ones (e.g. power, energy, delay). Skadron et al. [12] have presented an exhaustive analysis of the interaction between branch prediction, instruction-window size and cache size. They have utilized as workload the SPECint95 benchmarks. Their main focus was in the interplay between the three structures in terms of processor performance. Youssif et al. [13] have compiled a comprehensive list of currently existing prediction schemes and have evaluated them in terms of performance. Tests have utilized the SPEC2000 benchmarks and a superscalar-processor simulation environment. Parikh et al. [14] have investigated power repercussions of three advanced branch predictors on a Alpha 21264 simulator under SPEC2000-selected benchmarks. They have, then, proposed three interesting techniques for reducing power consumption in the branch-prediction unit. Jimenez et al. [15] have approached the branch-prediction issue from the viewpoint of delay as well as, typically, of accuracy and area. To this end, they have proposed three techniques for accommodating delay since they indicate its increasingly dominating impact on performance in future processors with large prediction structures.

The work presented here is original, as compared to related works like the above, in that: i) it studies the whole processor when different prediction schemes are utilized and particularly their reaction to different I/D-cache sizes, ii) it involves 3 more metrics in the study apart from performance, and iii) it targets a different class of low-power devices with particular idiosyncrasies. To the best of our knowledge, no similar effort has been reported so far in explicitly studying branch-prediction techniques for an implant processor.

III. Experimental setup

A. Characteristics of implant applications

In order to correctly set up our experiments as well as to select suitable branch techniques we fi5rst elaborate on the particular idiosyncrasies of microelectronic implants. Such implants are highly resource-constrained devices. The (re)implantation frequency for battery replacement - a costly and risky undertaking - is directly related to the operational life of a device. In order to achieve long in-vivo operation times, we are aiming at a **tight power budget** (μW order of magnitude).

An **ultra-small form factor** is also required for such devices considering the space available for implantation inside the body. This means that available processor area is also limited. Besides there are further aspects benefitting from low transistor counts (but out of the scope of this paper) such as higher device yield, increased testability and higher coverage for fault-tolerant design.

There has been shown to exist [2] and we are targeting a significant category of biomedical applications displaying **moderate performance** requirements, e.g. a feedback loop periodically regulating the functionality of bioactuators based on readouts from biosensors. Even so, under tight power and area budgets, the implant still has to complete its real-time (repetitive) duties within specific time margins. To do so, it must maintain a minimal instruction rate under the worst-case scenario.

B. Input datasets

Typical biomedical readouts are often highly periodic signals (e.g. heart beat) or stable signals (e.g. blood temperature) which can, under specific circumstances, display gradual or abrupt changes in value (e.g. a sudden muscle contortion). We have collected and used various workloads, representative of such behaviors, capturing both stable as well as rapidly changing patterns. The original **datasets** have been provided from the BIOPAC (R) Student Lab PRO v3.7 Software. Paper-size limitations do not allow for an extensive description of the various workloads; for the work presented here, we have used a biological dataset containing 10 KB of ECG data and representative of all examined workloads. Reported literature and an extensive study [2] on implants have revealed that typical memory sizes inside the implants range from 1 KB to 10 KB; thus, the use of $10-KB$ ECG data. More particularly, they have revealed instruction-memory sizes in the $10-KB$ locus and data-memory sizes in the $1-KB$ locus.

C. Benchmark applications

Eight **benchmark applications** have been used to evaluate different cache configurations. They comprise the ImpBench benchmark suite [11] and consist of *lossless data compression* algorithms, *symmetric-key encryption* algorithms and *data-integrity* algorithms as well as representative code based on *real biomedical applications*. The benchmarks are reported in Table I for convenience; they represent common tasks in present and future implant applications and also exhibit

BENCHMARK TYPE	NAME	SIZE (KB)
Compression	miniLZO	16.30
	Finish	10.40
Encryption	MISTY1	18.80
	RC6	11.40
Data integrity	checksum	9.40
	CRC32	9.30
Real applications	motion	9.44
	DMU	19.50

TABLE I

IMPBENCH [11] BENCHMARKS.

BENCHMARK	#CC	#INSTR.	#BRANCHES	BR. RATIO
checksum	1,102,562	62,869	7,933	12.62%
CRC32	12,021,257	419,159	69,976	16.69%
DMU	483,432,846	36,808,268	4,393,796	11.94%
Finish	80,581,690	852,663	147,971	17.35%
MISTY1	41,006,520	1,268,465	63,086	4.97%
MiniLZO	32,482,046	199,163	34,008	17.08%
motion	25,891,030	859,371	130,773	15.22%
RC6	25,919,634	864,930	60,869	7.04%

TABLE II

TYPICAL, GENERAL BENCHMARK STATISTICS.

varied characteristics. Some statistics useful for this study are reported in Table II.

D. Simulation testbed

Our branch-predictor evaluation study has been based on the XTREM [16] **simulator**, a modified version of SimpleScalar [17]. The XTREM simulator is a cycle-accurate, microarchitectural, power- and performance- functional simulator for the Intel XScale core [18]. It models the effective switching node capacitance of various functional units inside the core, following a similar modeling methodology to the one found in Wattch [19]. XTREM has been selected for its straight-forward functionality but mostly for its accuracy in performance and power modeling. It exhibits an average performance error of 6.5% and an average power error of 4% compared to real hardware.

Many of the XScale architectural features have been integrated into XTREM. Thumb instructions and special memory-page attributes are not supported but they do not affect simulation results since they are not used by our benchmarked applications. XTREM allows monitoring of 14 different functional units of the Intel XScale core: Instruction Decoder (DEC), Branch-Target Buffer (BTB), Fill Buffer (FB), Write Buffer (WB), Pend Buffer (PB), Register File (REG), Instruction Cache (I$), Data Cache (D$), Arithmetic-Logic Unit (ALU), Shift Unit (SHF), Multiplier Accumulator (MAC), Internal Memory Bus (MEM), Memory Manager (MM) and Clock (CLK). However, to better match our application field and, also, to isolate cache behavior as much as possible, many of XTREM's architectural parameters have been cut down or disabled to better reflect the highly constrained implantable processors. The modified XTREM characteristics are summarized in Table III. Performance/power figures have been checked and scaled properly with the changes.

E. Branch-prediction schemes

A large range of branch-prediction techniques has already been proposed in the literature. As previously discussed, our envisioned biomedical-implant processor is being designed - among others - under constraints of ultra-low power consumption and miniature form factor at the calculated cost of limited performance. Accordingly, the processor pipeline will feature a small depth. This set of attributes has effectively narrowed our evaluation effort towards the less complex end of the branch-prediction spectrum.

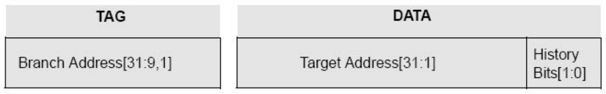

Fig. 2. Illustration of a BTB entry in the case of a bimodal predictor [20].

In the work at hand, we evaluate two static-prediction techniques, i.e. *ALWAYS TAKEN* and *ALWAYS NOT-TAKEN*; it is interesting to investigate how these low-sophistication (but also low-complexity) schemes perform in a implantable-device context. We further evaluate one dynamic-prediction technique, i.e. an *N-entry, direct-mapped BIMODAL (2-bit)* predictor which is coupled with a BTB structure used to drive branch penalties down. The BTB stores the history of branches that have executed along with their targets. Figure 2 shows an entry in the BTB, where the tag is the instruction address of a previously executed branch and the data contains the target address of the previously executed branch along with two bits of branch-history information (four states: strong-taken, weak-taken, weak-not-taken, strong-not-taken).

There clearly are more sophisticated techniques than a bimodal predictor to achieve higher prediction accuracy in the general case (e.g. skew predictor, gshare predictor) but their complexity is considerably higher, as well. Also, more general, n-bit predictors could be also studied but it has been sufficiently proven that 2-bit predictors score almost as high as infinite-bit predictors [21].

It is important at this point to mention that, for reasons of reliability as well as design complexity, our biomedical processor is meant to feature single-threaded execution, at least in the foreseeable future. Accordingly, all branch-prediction schemes are similarly evaluated on the XTREM simulator on single-executing, non-switched programs. Therefore, the accuracy and performance of the various branch-prediction schemes reported hereafter is pure and not not subject to deterioration due to context switching, as Pasricha and Veidenbaum have shown to occur [22].

IV. EVALUATION STUDY

In this study we have evaluated 10 different branch-predictor configurations, as shown in Table IV. The first two are the always-taken and always not-taken static predictors. The remaining eight entail the bimodal predictor, as discussed in the previous section, with an increasing number of entries for the BTB. The last one of those utilizes an unrealistically large BTB of 4K-entries - in effect, an infinite BTB - used as a reference for (almost) perfect predictions.

Besides, most implantable systems we have studied so far feature separate caches (or memories, in general). Cache organizations for our envisioned biomedical-implant processor have already been evaluated and (near-)optimal design points

feature	value	feature	value
ISA	32-bit ARMv5TE-compatible	BTB	1-entry fully-assoc. / direct-mapped
Pipeline depth / width	7/8-stage, super-pipelined / 32-bit	Branch Predictor	4cc mispred. lat. (32-entry ret. addr. stack)
RF size	16 registers	Mem. bus width	1B (1 mem. port)
Issue policy	in-order	INT ALUs	1
Instruction window	single-instruction	Clock frequency	2 MHz
I/D Cache L1 (separ)	var-size, 2-way assoc. (1-cc hit / 170-cc miss lat.)	Implem. technology	0.18 μm @ 1.5 Volt

TABLE III

XTREM (MODIFIED) ARCHITECTURE DETAILS.

bpred configuration	scheme	BTB #entries
bc01	TAKEN	n/a
bc02	NOTTAKEN	n/a
bc03	BIMOD	2
bc04	BIMOD	4
bc05	BIMOD	8
bc06	BIMOD	16
bc07	BIMOD	32
bc08	BIMOD	64
bc09	BIMOD	128
bc10 (perfect)	BIMOD	4K
L1-cache configuration	**I-cache, 2-way**	**D-cache, 2-way**
cc01 (min)	none	none
cc02	128B	64B
cc03	1KB	512B
cc04	8KB	4KB
cc05	32KB	16KB
cc06 (opt) [23]	64KB	32KB

TABLE IV

BRANCH-PREDICTION AND I/D-CACHE CONFIGURATIONS USED.

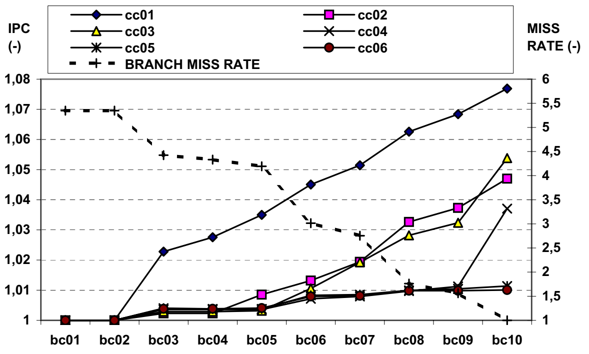

Fig. 3. Normalized to minimum, averaged, average IPCs (left y-axis values) and normalized overall branch miss rate (right y-axis values) for various bpred/cache configurations.

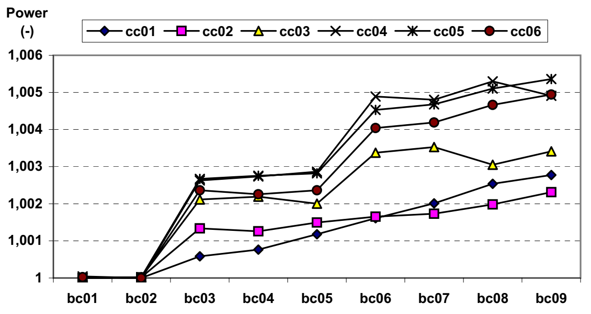

Fig. 4. Normalized to minimum, averaged, total power consumption for various bpred/cache configurations.

have been identified for both the I-cache and the D-cache structures [23]. To make the study more involved and identify subtler interactions among the various processor components, we have also chosen, along with the different predictor configurations, to co-vary also the L1 I/D-cache structures. Based on previous findings, we have selected two extreme cache configurations (one with no L1 caches and one with optimally-sized L1 caches) as well as four intermediate configuration nodes, as shown also in Table IV. The combination of the branch-predictor and cache configurations brings the total number of processor-simulator configurations up to 60, that is, for each cache configuration, all predictor configurations have been evaluated. The 8 benchmark applications presented above, have been run for each possible predictor/cache combination and their results have been averaged over all possible configuration combinations.

A. Performance analysis

We first evaluate the performance of the processor with improving the branch-predictor scheme utilized. In Fig.3, normalized[1] IPCs for all six cache configurations are plotted. Overall, we can see that for our simulated simple and slow ($2-MHz$ clock frequency) processor, IPC gains with improving predictor schemes are diminishing (up to about 8% w.r.t. the baseline) with increasing cache sizes, although the total branch miss rate drops considerably.

Relatively speaking, cache configurations cc01, cc02 and cc03 benefit the most from improved branch prediction. That is, a processor with larger caches hides the branch misprediction penalties better than one with smaller caches by capturing more instruction fetch requests from main memory. Configuration cc01 (no cache), in particular, displays the most impressive IPC gain compared to the other cache configurations. Inversely, this means that processors with smaller caches ought to benefit the most from an efficient branch prediction (*bpred*) scheme. Last, we can observe that both static schemes (predictor configurations bc01 and bc02) impact IPC minimally (they form the baseline) and in a similar fashion for all cache sizes while significant speedup is observed from configurations bc05 or bc06 and up for most cache configurations.

B. Power analysis

The next metric we investigate is the average power consumption. In Fig. 4, total power figures are plotted for all configuration combinations. The bc10 configuration results in excessive power consumption in the BTB component of the

[1]Normalized values to the *per-category* minimum value are used in the following discussion to make the differences between the various presented schemes more clear.

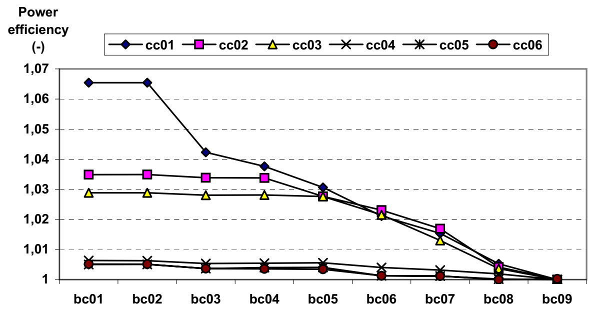

Fig. 5. Normalized to minimum, averaged, total power consumption per instruction per cycle (i.e. instruction efficiency w.r.t. power) for various bpred/cache configurations.

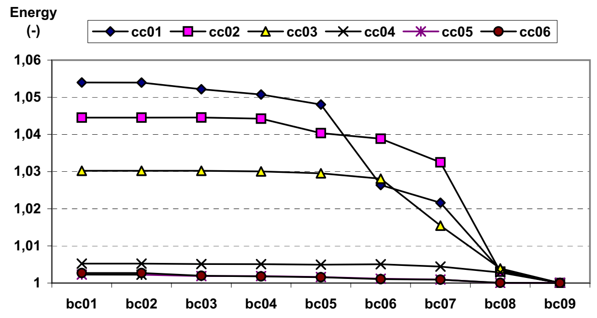

Fig. 6. Normalized to minimum, averaged, total energy expenditure for various bpred/cache configurations.

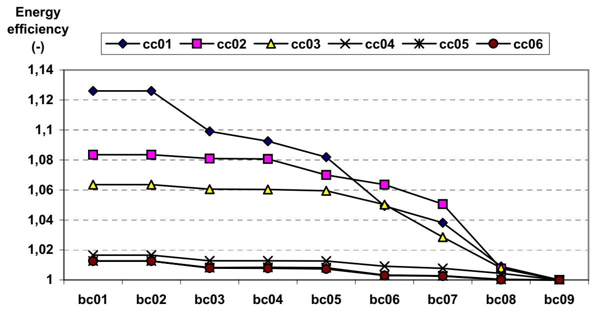

Fig. 7. Normalized to minimum, averaged, total energy expenditure per instruction per cycle (i.e. instruction efficiency w.r.t. energy) for various bpred/cache configurations.

processor ranging from 338 mW[2] (or 376% normalized w.r.t. the baseline) for cc01 to 563 mW (or 962% normalized w.r.t. the baseline) for cc06. This is due to its excessive size and has been omitted from the current and following plots to maintain a good resolution for the other nine bpred configurations.

The main observation in this figure is that smaller-cache processor configurations increase their power consumption at a slower pace with improving bpred schemes. When combined with Fig.3, this implies that small-cache processors achieve relatively higher IPC gains when improving their bpred scheme while incurring relatively smaller power increases. That is, more useful (cf. overhead) computations take place and the power consumed per executed instruction is reduced. This is clearly visible in Fig.5 which shows the normalized figures for the power consumed (in mW) per instruction per cycle, i.e. the normalized instruction power efficiency.

Last, the minimal impact the two static predictors (bc01 and bc02) have on the IPC is revealed in Fig.4, as well. Since they promote instruction-level parallelism (ILP) the least, they also don't stress the core much compared to the other bpred schemes, resulting in the lowest power profiles across all evaluated schemes. The IPC boost observed in Fig.3 starting in the locus of bc05 to bc06 is followed by a related increase in power consumption across the majority of cache configurations.

C. Energy analysis

Apart from average power consumption, for embedded systems with a very constrained energy budget such as implants are, it is also important to examine the overall energy spent by the processor for executing all assigned tasks. Energy, by definition, depends heavily on program execution time and, thus, energy plots are not necessarily identical to average-power plots.

In Fig. 6 overall energy budgets for different bpred/cache configurations are plotted. Opposite to the case made on power before, the metric at hand (i.e. energy) achieves a minimum value when moving to more complex bpred configurations, showing a dramatic improvement from configuration nodes bc07 and bc08 and onwards. When seen from the cache perspective, the cache configurations that lead to the highest energy savings are the larger-cache ones (cc04, cc05 and cc06), opposite to the power results. Choosing between a power-efficient and an energy-efficient configuration depends on the priorities imposed on the design specifications of the processor.

In the locus of bc05 or bc06 where IPC shows a non-trivial speedup, energy expenditure does not visibly drop for almost all cache configurations. This hints toward the fact that, at that point, the power costs to sustain the higher IPC outperform the benefits on speedup. To illustrate further, Fig.7 plots the instruction energy efficiency of the various configurations. With the exception of cc01, the lines do not drop as radically around the locus of bc5 and bc06.

D. Area analysis

In this part of our analysis, it also makes sense to consider the area cost of the various bpred schemes when moving to more advanced techniques. We have, therefore, properly configured and run CACTI v3.0 to collect area-utilization figures for various predictor circuits. Area for the static predictors as well as for the first two bimodal configurations have been based on estimations. CACTI v3.0 (instead of any newer versions) has been used since it is suitable for modeling simpler (older) cache-like structures (such as the BTB) and at an implementation technology identical to the one of the simulator ($0.180\mu m$). CACTI results are illustrated in Fig. 8. Please notice the logarithmic scale of the plot.

[2]The XTREM simulator has not been initially designed for modeling a μW processor. Thus, the relative differences between (not the absolute values of) the power figures reported here should be considered.

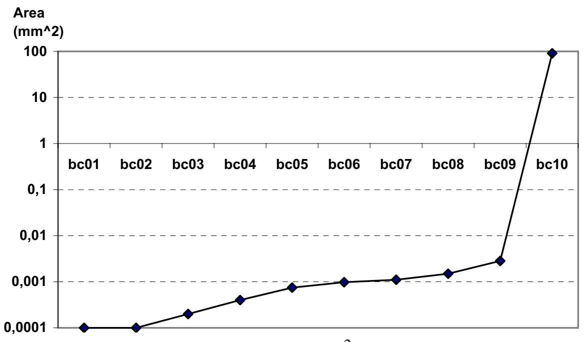

Fig. 8. Total bpred-scheme area (in mm^2) for all bpred configurations.

TOLERANCE LEVELS						
n	**cc01**	**cc02**	**cc03**	**cc04**	**cc05**	**cc06**
IPC	**0.985**	**0.990**	**0.985**	**0.995**	**1.000**	**1.000**
power	**0.990**	**0.990**	**0.990**	**1.000**	**1.000**	**1.000**
energy	**0.990**	**0.990**	**0.990**	**1.000**	**1.000**	**1.000**
area	**0.980**	**0.985**	**0.980**	**0.995**	**0.995**	**0.995**

TABLE V

TOLERANCE LEVELS FOR IPC, POWER, ENERGY AND AREA IN OBJECTIVE FUNCTION (1) FOR ALL CACHE CONFIGURATIONS WHEN AREA IS CONSIDERED.

E. Optimal-branch/cache configuration selection

For selecting the best bpred configuration for different I/D-cache geometries, we have based our evaluation on performance, power consumption, energy expenditure and area. As a performance metric, we have chosen the IPC instead of the branch miss rate since we do not wish to study the bpred techniques in an isolated environment but, rather, we wish to capture overall system performance as a function of predictor type. That is why we have used a processor (rather than branch-predictor) simulator as our testbed. For the very same reason we have also used total average power consumption and total energy budget as our second and third metric, respectively.

To find optimal bpred schemes for each cache configuration, we have used the following formula as our **objective function** for *minimization*:

$$F^n(x) = IPC^n_{PD}(x) + P^n_{PD}(x) + E^n_{PD}(x) + A^n_{PD}(x), \quad (1)$$

where x represents a single bpred node and n a single cache node. Each term $VAR_{PD}(x)$ represents the *percentage difference* between the VAR value at node x and the best VAR value across all bpred nodes (maximum value for IPC, minimum value for power, energy and area). This percentage difference is given by the formula:

$$VAR_{PD}(x) = \frac{|VAR(x) - VAR_{OPT}|}{(1/2) * (VAR(x) + VAR_{OPT})} * 100, \quad (2)$$

where $VAR_{OPT} = max(VAR(x))$ or $min(VAR(x))$, with x in the range [bc01,bc10]. We have chosen to use percentage differences in our objective function (1) so as to normalize all involved variables by calculating their "relative" deviation from the per-case optimal value.

We have sought a bpred configuration for each cache node that optimizes all four imposed metrics. Table V shows, per cache node, the tolerance levels we have used for all metrics. The closer to '1.000' a metric's tolerance level is, the more strictly constrained the objective function (1) becomes and the more close to optimal is the solution. In the table above, tolerance levels have been iteratively decreased until a single solution (i.e. the optimal) to each objective function was found.

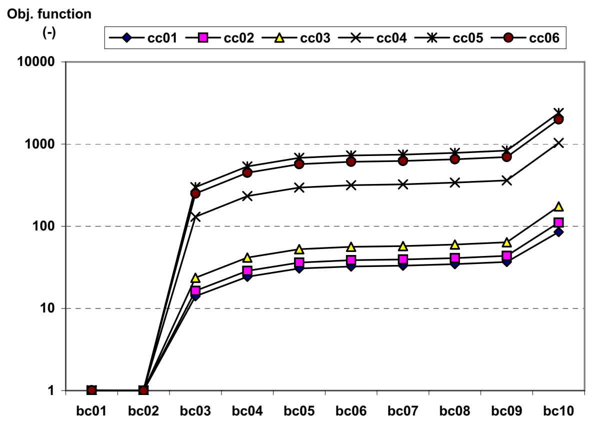

Fig. 9. Normalized results to the minimum value for various cache configurations of objective function (1) when area is included.

In the absence of accurate processor design constraints[3] we have imposed an intuitive ordering to the metrics used in the objective function. In order of decreasing importance, the metrics have been ranked: power and energy first, then IPC and, last, area. It is with this order that we have adjusted the tolerance levels in search of the best solution for all six objective functions.

We can readily see from Table V that as we move from configuration cc01 to cc06 (i.e. to higher cache sizes), the tolerance levels become increasingly higher, that is, stricter. This indicates that with increasing cache sizes, optimal values across the four metrics are less scattered; thus, the solution is more straightforward.

The results of the optimization process are graphically depicted in Fig.9. With deviations of 1% or less between them, the static predictors ALWAYS TAKEN and ALWAYS NOT-TAKEN minimize the objective function across all cache configurations. The remaining configurations follow with significantly worse ranking, although we can see that smaller-cache configurations would benefit more from the bimodal predictors of any size, compared with larger-cache configurations.

Some commenting on this result is needed here. In the objective functions above we have included the area metric. However, area calculation has been done through CACTI, outside the XTREM simulator. As a result, the area utilization for specific bpred configurations as well as the modeling of the BTB with CACTI has been to a certain degree based on speculations and assumptions.

[3]Previous, related work does not provide solid or intuitive data for properly adjusting the metric weights, i.e. to contribution of the different metrics, in formula (1). To avoid unfair biasing of the results, we have assumed here a policy of equal weights.

TOLERANCE LEVELS						
n	cc01	cc02	cc03	cc04	cc05	cc06
IPC	0.980	0.980	0.990	0.995	0.999	0.999
power	0.980	0.990	0.990	0.999	0.999	0.999
energy	0.980	0.990	0.990	0.999	0.999	0.999
area	0.000	0.000	0.000	0.000	0.000	0.000

TABLE VI

TOLERANCE LEVELS FOR IPC, POWER, ENERGY AND AREA IN OBJECTIVE FUNCTION (1) FOR ALL CACHE CONFIGURATIONS WHEN AREA IS NOT CONSIDERED.

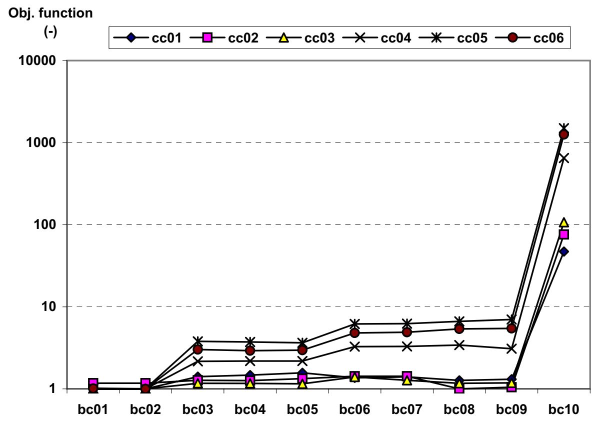

Fig. 10. Normalized results to the minimum value for various cache configurations of objective function (1) when area is not included.

For fear of skewing the optimization results and for purposes of completeness we have also optimized the objective function (1) once more after disregarding the area metric $A_{PD}(x)$. In so doing, the tolerance levels of the remaining metrics have also been readjusted, as illustrated in Table VI. The new tolerance levels are somewhat lower than before to compensate for the area variable, yet they exhibit the same trend as in the previous case; that is, they can be raised higher with increasing cache sizes.

Graphical depictions of the objective-function results in this case are shown in Fig.10. Most suitable bpred configurations for almost all cache sizes are, as before, the ALWAYS TAKEN and ALWAYS NOT-TAKEN static schemes. However, the general objective-function trends are different in this case. We can observe that for smaller-cache configurations the objective function is not monotonously increasing but, rather, features more than one (local) minimum. As a result, when the area metric is factored out of the equation, more solutions than the TAKEN/NOT-TAKEN schemes can be found, especially for smaller-cache configurations. To exemplify, for cache configuration cc03 the optimal bpred configuration is in fact bc08, i.e. a bimodal predictor with a 64-entry BTB, followed by the typical TAKEN/NOT-TAKEN schemes. These suboptimal configuration nodes could become the nodes of choice in a design where e.g. a lower threshold on IPC is imposed; thus, in cases of constrained optimization.

Overall, and to combine the results of both Fig.9 and Fig.10, it becomes clear that - even by factoring the area component out of the optimization function - the cost of bimodal predictors of any BTB size is too high to justify the overheads incurred to the processor, across all cache sizes. Practically speaking, this means that the reduction in branch mispredictions offered does not improve performance to the point that introduced power penalties of a dynamic predictor can be justified.

V. CONCLUSIONS

In this paper we have provided a detailed investigation of various branch-prediction configurations in conjunction with different I/D-cache configurations, tested on a specially modified, low-power, cycle-accurate processor simulator. We have fed the machine with benchmarks suitable for profiling biomedical-implant applications and have focused on performance, power, energy and area results. We have, then, run iterative optimization (minimization) functions on the specified design space and have identified best branch-predictor candidates per I/D-cache node for our end goal which is the design of a novel implant processor. Findings indicate that, under (relaxed) area constraints, the optimal selections for branch prediction interchangeably are the static schemes ALWAYS TAKEN and ALWAYS NOT-TAKEN, regardless of processor cache size. This means that for slow-performing, ultra-low power processors and the given biomedical workloads, dynamic-prediction schemes are too expensive to implement, their drawbacks outperforming their benefits.

A second interesting result, however, is that processor configurations with smaller caches (i.e. for I/D-cache sizes up to $1024KB/512KB$, respectively) benefit from efficient bpred schemes more. Especially under relaxed area constraints, more suboptimal configurations but close to the optimal one exist. In lack of more accurate data from the literature, in this work we have imposed loose design constraints on the evaluated metrics. Yet this last observation can be particularly useful under highly constrained processor design.

As a final contribution, the paper offers a sound methodology and, at the same time, defines a suitable starting point for the design-space exploration work for our envisioned processor. Besides, the methodology we have used, supported by a cycle-accurate power/performance simulator and our developed toolflow, can be used to find optimal branch-predictor/cache configurations for different application scenarios. Updating objective functions with more variables (i.e. design parameters) is straight-forward and adjusting their contribution (weight) and tolerance levels to the optimization problem can be modified just as easily. Last, this work to our best knowledge is the first attempt to study branch-predictor behavior under different cache geometries for the application field of biomedical implants. Based on our previous profiling study and the current work, our future work entails the full (micro)architectural specification and prototyping of our targeted biomedical-implant processor.

VI. ACKNOWLEDGEMENTS

This work has been partially supported by the ICT Delft Research Centre (DRC-ICT) of the Delft University of Technology.

References

[1] H. Ector and P. Vardas, "Heart disease and stroke statistics - 2008 Update (At-a-Glance Version)," *American Heart Association*, 2008.

[2] C. Strydis *et al.*, "Implantable microelectronic devices: A comprehensive review," Computer Engineering, TU Delft, CE-TR-2006-01, Dec. 2006.

[3] M. Ghovanloo and K. Najafi, "A modular 32-site wireless neural stimulation microsystem," in *IEEE Journal of Solid-State Circuits*, vol. 39, December 2004, pp. 2457–2466.

[4] P. Mohseni and K. Najafi, "Wireless multichannel biopotential recording using an integrated FM telemetry circuit," in *26th Annual International Conference of the IEEE in Engineering in Medicine and Biology Society (EMBS)*, San Francisco, CA, USA, 1-5 September 2004, pp. 4083–4086.

[5] H. Park, H. Nam, B. Song, and J. Cho, "Design of miniaturized telemetry module for bi-directional wireless endoscopy," *IEICE Transactions on Fundamentals on Electronics, Communications and Computer Sciences*, vol. 6, pp. 1487–1491, June 2003.

[6] C. Liang, J. Chen, C. Chung, C. Cheng, and C. Wang, "An implantable bi-directional wireless transmission system for transcutaneous biological signal recording," *Physiological Measurement*, vol. 26, pp. 83–97, February 2005.

[7] P. Cross, R. Kunnemeyer, C. Bunt, D. Carnegie, and M. Rathbone, "Control, communication and monitoring of intravaginal drug delivery in dairy cows," in *International Journal of Pharmaceuticals*, vol. 282, 10 September 2004, pp. 35–44.

[8] H. Lanmuller, E. Unger, M. Reichel, Z. Ashley, W. Mayr, and A. Tschakert, "Implantable stimulator for the conditioning of denervated muscles in rabbit," in *8th Vienna International Workshop on Functional Electrical Stimulation*, Vienna, Austria, 10-13 September 2004.

[9] C. Strydis and G. Gaydadjiev, "Lossless data compression in ultra-low-power embedded systems: An analysis," in *Submitted to International Conference on Hardware-Software Codesign and System Synthesis (CODES'08)*, Atlanda, Georgia, 19-24 October 2008.

[10] C. Strydis, D. Zhu, and G. Gaydadjiev, "Profiling of symmetric encryption algorithms for a novel biomedical-implant architecture," in *ACM International Conference on Computing Frontiers (CF'08)*, Ischia, Italy, 5-7 May 2008, pp. 231–240.

[11] C. Strydis, C. Kachris, and G. Gaydadjiev, "ImpBench - A novel benchmark suite for biomedical, microelectronic implants," in *To appear in International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS'08)*, Samos, Greece, 21-24 July 2008.

[12] K. Skadron, P. Ahuja, M. Martonosi, and D. Clark, "Branch prediction, instruction-window size, and cache size: performance trade-offs and simulation techniques," *Computers, IEEE Transactions on*, vol. 48, no. 11, pp. 1260–1281, Nov 1999.

[13] A. Youssif, N. Ismail, and F. Torkey, "Comparison of branch prediction schemes for superscalar processors iceec 2004," *Electrical, Electronic and Computer Engineering, 2004. ICEEC '04. 2004 International Conference on*, pp. 257–260, Sept. 2004.

[14] D. Parikh, K. Skadron, Y. Zhang, and M. Stan, "Power-aware branch prediction: Characterization and design," *IEEE Transactions on Computers*, vol. 53, no. 2, pp. 168–186, 2004.

[15] D. Jimenez, S. Keckler, and C. Lin, "The impact of delay on the design of branch predictors," *Microarchitecture, 2000. MICRO-33. Proceedings. 33rd Annual IEEE/ACM International Symposium on*, pp. 67–76, 2000.

[16] G. Contreras *et al.*, "XTREM: A Power Simulator for the Intel XScale Core," in *LCTES'04*, 2004, pp. 115–125.

[17] T. Austin *et al.*, "SimpleScalar: an infrastructure for computer system modeling," *IEEE Computer*, vol. 35, no. 2, pp. 59–67, Feb. 2002.

[18] *Intel XScale Microarchitecture for the PXA255 Processor: User's Manual*, Intel Corp., March 2003.

[19] D. Brooks *et al.*, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," in *ISCA'00*, 2000, pp. 83–94.

[20] *Intel XScale Core Developer's Manual*, Intel, December 2000.

[21] J. Hennessy and D. Patterson, *Computer Architecture - A Quantitative Approach*, 4th ed., D. Penrose, Ed. Morgan Kaufmann, 2003.

[22] S. Pasricha and A. Veidenbaum, "Improving branch prediction accuracy in embedded processors in the presence of context switches," *Computer Design, 2003. Proceedings. 21st International Conference on*, pp. 526–531, Oct. 2003.

[23] C. Strydis and G. Gaydadjiev, "Suitable cache organizations for a novel biomedical-implant architecture," in *International Conference of Computer Design (ICCD'08)*, Lake Tahoe, California, USA, 12-15 October 2008, pp. 591–598.