www.ietdl.org

Published in IET Computers & Digital Techniques
Received on 13th July 2010
Revised on 10th November 2010
doi: 10.1049/iet-cdt.2010.0102

ISSN 1751-8601

Exploring branch target buffer access filtering for low-energy and high-performance microarchitectures

S. Wang[1] *J. Hu*[2] *S.G. Ziavras*[3]

[1]*National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiang Su 210093, People's Republic of China*
[2]*Intel Corporation, Portland, OR 97124, USA*
[3]*Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA*
E-mail: swang@nju.edu.cn

Abstract: Powerful branch predictors along with a large branch target buffer (BTB) are employed in superscalar and simultaneous multi-threading (SMT) processors for instruction-level parallelism and thread-level parallelism exploitation. However, the large BTB not only dominates the predictor energy consumption, but also becomes a major roadblock in achieving faster clock frequencies at deep sub-micron technologies. The authors propose here a filtering scheme to dramatically reduce the accesses to the BTB to achieve significantly reduced energy consumption in the BTB while maintaining the performance. For a simulated superscalar microprocessor, the experimental evaluation shows that the BTB access filtering (BAF) design achieves an 88.5% dynamic energy reduction with negligible performance loss. The authors also study the leakage behaviour and its control in the BAF design. The results show that by applying a drowsy strategy, very effective leakage control can be achieved. For the high-performance design, the BAF can also improve BTB's performance scalability at new technologies. For the simultaneous multi-threading environment, the authors evaluate the effectiveness of the BAF design and propose a banked BAF (BK-BAF) scheme to further reduce the energy consumption and performance overhead. The experimental results confirm that the BK-BAF scheme can be an energy/performance-effective design for next generation SMT processors.

1 Introduction

Modern high-performance superscalar processor design is mainly driven by techniques exploiting high instruction-level parallelism (ILP) and faster clock frequencies with continuously advancing complementary metal-oxide semiconductor (CMOS) technology. Besides out-of-order issue/execution and register renaming, speculative execution is another major form of ILP exploitation. With correct branch predictions, the processor not only eliminates pipeline stalls due to control hazards, but also enables the datapath front end to supply sufficient instructions for ILP exploitation at later pipeline stages. However, a mispredicted branch requires flushing from the datapath pipeline all instructions fetched along the speculated path and refilling the pipeline with new instructions from the resolved target address. Therefore the branch misprediction penalty is recognised as a major performance limiter in speculative superscalar processors. A highly accurate branch predictor is of critical importance in the design of ILP processors, which has been the focus of tremendous research efforts [2–5]. As the direction predictor is getting more and more sophisticated, a large BTB [6] is usually adopted to supply target addresses for predicted taken branches, leading to non-trivial energy consumption in branch predictors [7, 8]. Recent study [8] shows that the branch prediction unit contributes a non-trivial percentage, about 7–10% to the total processors energy consumption. Our experimental results show that a typical 2k-entry two-way set-associative BTB dissipates about 86% of the total branch prediction unit energy in a simulated Alpha 21364 processor. For SMT processors, which target at exploiting thread-level parallelism (TLP), the situation is even worse, since larger BTBs are needed in order to support the speculative execution for multiple threads [9–13]. Consequently, energy optimisation in the BTB is becoming an indispensable component in the design of energy-aware processors at deep sub-micron technologies.

As the logic depth of the pipeline stage keeps reducing [14] at deeply pipelined designs for higher clock frequencies, operations in many conventional large monolithic structures such as the issue queue and register file can no longer be completed within a single pipeline stage, eventually leading to reduced performance and significantly increased design complexity. Therefore plenty of research has been devoted to exploring complexity-effective issue queue and register file designs, for example, [15–20] among many others. On the other hand, research on branch prediction has traditionally focused on direction prediction [2–5]. There is limited work on the energy optimisation in branch predictors [7, 8, 21–23], and very few on the complexity and performance scalability of predictor designs. Moreover,

once major datapath components adopt such complexity-effective designs, the branch predictor, especially with a large BTB, may eventually become a performance hindrance owing to its monolithic structure, especially in the SMT environment [24]. It is then important to explore new scalable BTB designs for high-performance processors designed at new technology generations.

Based on a study of the distribution of the dynamic branch direction (taken/not-taken), we propose a filtering scheme controlled by the branch direction predictor to reduce the accesses to the BTB. In order to maintain the performance, we introduce an additional small BTB structure (filter buffer) into our filter, which further reduces the accesses to the BTB. For leakage control, the drowsy scheme becomes more effective in our BAF design, because access filtering makes the BTB inactive during most periods of time. Furthermore, our BAF design aims to provide a low-complexity scalable design supporting higher clock frequencies at new technology generations. Although a multi-level BTB was initially proposed and evaluated in [6] for performance improvement, it was not considered as a practical implementation option at that technology generation. In our BAF microarchitecture, the large BTB is rarely accessed because of the filtering effect. Therefore the performance degradation caused by the long access latency in the large BTB at new technology generations can be amortised. For the energy-efficient BTB design in the SMT processor, a BK-BAF design is proposed to further reduce the energy and performance overheads.

A preliminary version of this work [1] was presented at the IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2008). We have extended the work (i) by studying the energy-performance tradeoffs in branch target buffers (BTBs) for simultaneous multi-threading (SMT) processors, (ii) by proposing a banked BTB access filtering (BK-BAF) design to reduce the dynamic and leakage energy consumption as well as the performance overhead in BTBs for SMT processors and (iii) by evaluating the performance scalability of the BK-BAF design in the SMT environment.

The rest of the paper is organised as follows. Related work is discussed in Section 2. Section 3 presents the experimental evaluation framework for this work. We propose our BAF design in Section 4. A BK-BAF design is further proposed for the SMT environment in Section 5. The experimental results and analyses are presented in Section 6. Finally, Section 7 concludes this work.

2 Related work

Filter cache [25, 26] was proposed for the low power on-chip cache design. Different from the use of a small cache structure as the filter in [25, 26], our filtering scheme is controlled by the output of the branch direction predictor. The small filter buffer is introduced to further improve the performance and the filtering effect. In [8], the prediction probe detector (PPD) detects the non-branch instructions in each cache line to avoid unnecessary accesses to the branch predictor and BTB. However, for complete energy savings, the PPD needs to be accessed before the branch predictor, which may increase the pipeline latency. Moreover, since each PPD entry is corresponding to a single cache line in the instruction cache, it will increase the energy consumption and design complexity especially with a set-associative instruction cache. Different from PPD, our BAF targets at the removal of unnecessary accesses (for all predicted not-taken branches) to the BTB for energy savings. Further, the BAF design only incurs a negligible performance loss and slight changes in the hardware implementation. An adaptive scheme to resize the BTB according to its on-demand utilisation was proposed in [22] to reduce energy consumption. This adaptive scheme requires a profiling phase to collect utilisation information and relies on software to apply resizing. In contrast, our BAF is a pure hardware implementation that requires no changes to the compiler or the application program, which makes BAF transparent to the software layer. In the next generation processor architecture, both Intel Nehalem [27] and IBM System z10 [28] use the two-level BTB design to reduce the performance penalty of mispredicted branches. Different from the conventional hierarchical memory structure designs, where a miss in the upper level of the memory hierarchy always results in an access to the lower level, the output of the branch direction predictor in our BAF design acts as a filter and significantly reduces the unnecessary accesses to the BTB.

3 Experimental setup

We derive our simulators from SimpleScalar V3.0 [29] to model a contemporary high-performance microprocessor similar to Alpha 21364 [30]. In the new simulator, the original RUU structure is replaced by separated integer issue queue, floating-point issue queue, integer register file, floating-point register file and the active list (a.k.a. the re-order buffer). A MIPS R10000 style register renaming scheme is adopted in our implementation. Table 1 gives the detailed configuration of the simulated microprocessor. To evaluate the energy efficiency and performance scalability of our BAF design, a modified version of the Wattch power model [31] is used for power profiling (at 70 nm technology) during the simulation.

For the BAF study in the SMT environment, we develop an SMT simulator based on the SimpleScalar to model a modern microprocessor similar to Alpha 21464 [32]. The fetch architecture uses the ICOUNT.1.8 [33] fetch policy (up to eight instructions from one thread each cycle). Table 2 shows the configuration of the simulated SMT processor core.

Table 1 Parameters of the simulated superscalar processor

Processor core	
datapath width	4 inst. per cycle
int issue queue	20 entries
FP issue queue	15 entries
load/store queue	64 entries
active list (ACL)	80 entries
int register file	80 registers
FP register file	72 registers
function units	4 IALU, 2 IMULT/IDIV
	2 FALU, 1 FMULT/FDIV/FSQRT
	2 MemPorts
Branch predictor	
branch predictor	Alpha 21264 tournament predictor [34]
	32-entry RAS
BTB	2048-entry 2-way, 1 port
Memory hierarchy	
L1 I/DCache	64 KB, 2 ways, 64B blocks, 2 cycles
L2 UCache	4 MB, 8 ways, 128B blocks, 12 cycles
memory	225 cycles first chunk, 12 cycles rest
TLB	Fully-assoc., 128 entries

Table 2 Parameters of the simulated SMT processor

Processor core	
fetch policy	ICOUNT.1.8
RUU	128 entries
load/store queue	128 entries
function units	6 IALU, 2 IMULT/IDIV 4 FALU, 2 FMULT/FDIV/FSQRT 4 MemPorts
Branch predictor	
branch predictor	Alpha 21264 tournament predictor [34] 32-entry RAS
BTB	4096-entry 4-way, 2 ports
Memory hierarchy	
L1 I/DCache	64 KB, 2 ways, 64B blocks, 2 cycles
L2 UCache	4 MB, 8 ways, 128B blocks, 12 cycles
memory	225 cycles first chunk, 12 cycles rest
TLB	Fully-assoc., 128 entries

Table 3 SMT workloads

Workloads	Benchmarks	Categories
1	wupwise, swim, gzip, vpr,	2 fp + 2 int
2	mesa, galgel, crafty, parser	2 fp + 2 int
3	art, equake, eon, perlbmk	2 fp + 2 int
4	apsi, wupwise, swim, mgrid	4 fp
5	gap, vortex, bzip2, twolf	4 int
6	gzip, vpr, gcc, mcf	4 int

For experimental evaluation, we use the SPEC CPU2000 benchmark suite compiled for the Alpha instruction set architecture using the '-arch ev6-non-shared' option with 'peak' tuning. For the simulated superscalar processor, we use the reference input sets for this study. Each benchmark is first fast-forwarded to its early single simulation point (gap and ammp use the standard single simulation point instead of the very large early single simulation point) specified by SimPoint [35]. We use the last 100 million instructions during the fast-forwarding phase to warm up the caches if the number of skipped instructions is more than 100 million. Then, we simulate the next 100 million instructions in detail. For the simulated SMT processor, six workloads with four threads are used. We randomly choose the combinations from all integer and floating-point benchmarks, which are shown in Table 3. 200 million instructions are used for fast-forwarding and warm-up. Then, the next 400 million instructions are simulated in detail.

4 BAF for superscalar processors

4.1 Motivation

Since a large BTB dominates the area and energy consumption of a branch predictor [7, 8], we focus on low-energy and high-performance BTB designs. In conventional BTB designs, in order to make the branch target address available immediately after the branch is predicted as taken, the large BTB is accessed along with the branch direction predictors simultaneously in every cycle. Even if the branch is finally predicted as not-taken, the large BTB has been accessed for the purpose of improving the performance. However, this conventional design implies many unnecessary BTB accesses for predicted not-taken branches, leading to extra energy consumption in the BTB. Our experimental results show that on an average about 38.1% of dynamic branches are predicted as not-taken for which the BTB access can be avoided.

To avoid the unnecessary BTB accesses for predicted not-taken branches, the branch direction predictor needs to be accessed one cycle before the BTB in order to use the direction prediction as the BTB access filter. However, such a sequential filter design will put the branch direction prediction and BTB access into two consecutive pipeline stages, which will hurt the processor performance (IPC) due to the increased branch resolution loop [36]. Our experimental results show that simply delaying the BTB access one cycle after the branch direction prediction incurs a 2.9% performance loss on the average, which is not desirable in a high-performance processor design.

To address the performance issue in the above simple filter scheme, we introduce a small filter buffer (FB), which itself is a very small BTB structure, into our filtering scheme. In this enhanced BAF scheme, the FB is accessed in parallel with the branch direction predictor, while the BTB is accessed only when a predicted taken branch misses in the FB.

4.2 Microarchitecture of BAF

Fig. 1 gives the microarchitecture of our BAF design. At the fetch stage, the instruction cache, the branch direction predictor and the FB are accessed simultaneously. If the branch is predicted as not-taken, or the branch is predicted as taken and the FB hits, the BAF will not involve the BTB in the next cycle. However, if the branch is predicted as taken and the FB lookup results in a miss, the BTB needs to be accessed in the next cycle. In this case, the fetch logic either stalls or inserts NOP instructions (by supplying an address of 0) till the result of the BTB access becomes available. Notice that the access to the BTB is not only controlled by the outcome of the branch direction prediction, but also by the result of the FB access (i.e. hit or miss). This specific characteristic of our BAF differs significantly from the conventional BTB design, where the BTB is accessed in parallel with the branch direction predictor. In our BAF, only the FB is accessed in parallel with the branch direction predictor every cycle and the BTB is rarely accessed.

The BAF is updated as follows. If there is a FB miss but a BTB hit for a committed taken branch, we only update the FB. Otherwise, if misses occur in both the FB and BTB for a committed taken branch, we update them both. Since the access miss, especially in the BTB, is very infrequent, the energy consumption due to updates is relatively small.

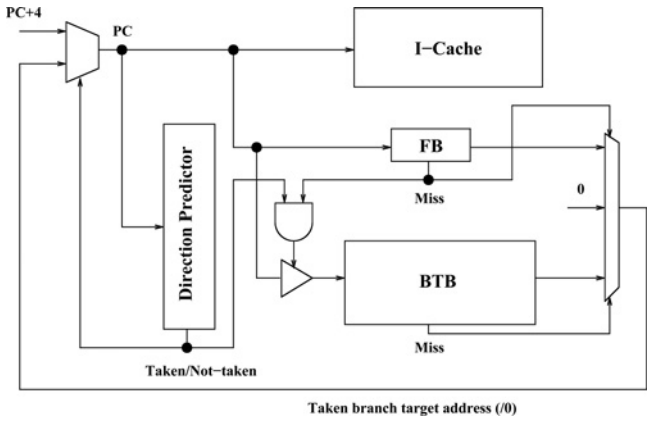

Fig. 1 *Schematic of the BAF for the superscalar processor*

doi: 10.1049/iet-cdt.2010.0102

Notice that the major hardware cost of our BAF design is the FB which is very small compared to the original BTB.

4.3 Energy efficiency of BAF

4.3.1 Dynamic energy: In our BAF design, FB is the one of small size and simple organisation, for example, a 128-entry direct-mapped structure. Therefore the average dynamic energy consumption per access to the FB is very small compared to a conventional large BTB. The BTB in our BAF design can be as large as the conventional ones. Since most BTB accesses will be served by the FB or filtered by the non-taken direction prediction before going to the large BTB, accesses to the large BTB can be dramatically reduced. Consequently, the dynamic energy consumption in our BAF can be significantly reduced.

4.3.2 Leakage energy: With the advancing technology, the leakage energy is becoming an important contributor to the energy consumption. For leakage savings in our BAF design, the large BTB can be either designed with high V_T low-leakage transistors, or drowsy [37], or decay [38] control logic. Owing to its very infrequent activities in the BTB, our BAF scheme provides additional opportunities to further optimise the leakage energy in the BTB. In most cases, only the FB is accessed and the BTB is in the idle state, which makes the drowsy scheme very effective in our BAF design. We can put the idle entries into drowsy mode. When the drowsy entries need to be accessed, there will be additional cycles for waking up these entries. However, the performance will not degrade too much because of the rare occurrence of the wake-ups. This scheme will put most of the BTB entries into the drowsy mode, which can reduce the leakage energy of the BTB significantly. In this work, we explore how aggressively the BTB entries can be put into the drowsy mode for leakage control. To reduce the drowsy logic overhead, multiple BTB entries can also share single drowsy control logic.

4.4 Performance scalability of BAF

Besides its energy efficiency, the design of BAF also aims to provide a performance and complexity-oriented scalable solution to BTB designs at new technology generations. Since the wire delay does not scale very well with advancing CMOS technology, large monolithic datapath components, for example, the conventional large BTB, dominated by the long wire delay are becoming relatively slower, that is, requiring multiple cycles for access. However, a simple small BTB to fit in the single-cycle access latency will also introduce noticeable performance loss due to dramatically increased BTB misses. The proposed BAF scheme solves this problem by filtering out most accesses to the large slow BTB. In our BAF design, owing to its small size, the FB scales much better with an advanced technology and normally can be accessed in one cycle. On the other hand, the large BTB can tolerate longer access latency because of the filtering effect. Since our BAF design has a relatively high filtering rate, in most cases the long BTB access latency will not be incurred. If there is a FB miss and the branch is predicted as taken, the large BTB is then accessed, which may cause additional two or three cycles delay. Notice that the FB and the BTB can be accessed simultaneously to overlap one cycle latency for the large BTB, which can further improve the performance. However, this design is less energy efficient because it introduces a huge number of accesses to the large BTB. Therefore in order to fully exploit the energy efficiency, we prefer accessing them separately in different cycles.

5 BAF design for SMT processors

5.1 Energy-performance tradeoffs

The SMT architecture [39, 40] was proposed to better exploit the TLP for high performance microprocessor designs. The pressure on the BTB is further aggravated in the SMT environment due to the increased instructions/branches from multiple threads. Moreover, large BTBs with multiple ports are also needed in order to support multiple predictions per cycle. Therefore large and multi-ported BTBs in the SMT processors will consume even more energy compared to that in the superscalar processor [41, 42].

One simple solution is to reduce the BTB size, which is similar to what we have discussed for the superscalar processors. However, simply reducing the BTB size will cause significant performance loss and this situation will become even worse in the SMT environment due to the high pressure on the BTB. To apply our BAF design in the SMT environment, we first conduct the profiling on the directions of the dynamic branch predictions. Our experimental results show that 31.5% of dynamic branches are predicted as not-taken, which is similar to that (38.1%) in the superscalar processor. If we avoid the unnecessary accesses of these not-taken branches by delaying the BTB access, it will incur 19.1% performance loss, which is much worse than that (2.9%) in the superscalar processor.

5.2 BK-BAF for SMT

In the superscalar environment, the BTB miss can be categorised by two different types: the cold start miss that occurs during the first time a branch is encountered and the capacity miss that occurs when branches conflict with each other due to the limited size of the BTB. The SMT environment will introduce a new type of BTB miss called thread competition miss that is caused by branches of different threads contending for the same BTB entry. In order to alleviate the thread competition miss in the FB of our BAF design, we propose a BK-BAF that uses the FB with the banked design instead of a single monolithic one.

Fig. 2 shows the microarchitecture of the BK-BAF design with two FB banks. Compared to the original BAF design, the FB is split into two banks and the thread id is used to

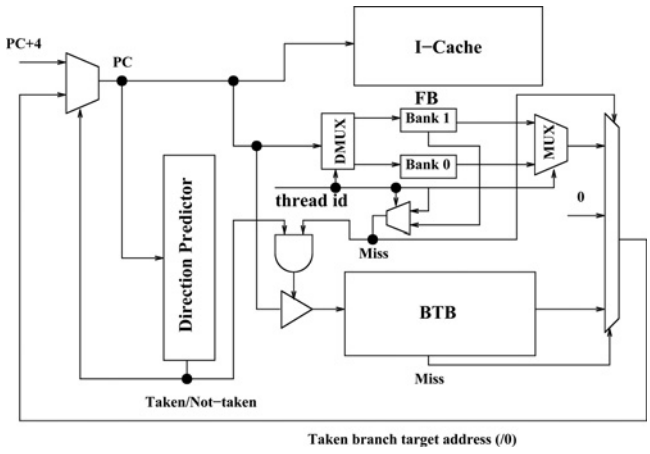

Fig. 2 *Schematic of the BK-BAF for the SMT processor*

determine which bank should be accessed. For example, in our simulated four-thread SMT, only thread 0 and thread 1 will use the bank 0, and thread 2 and thread 3 will access the bank 1. Therefore the overall competition among different threads in the FB will be reduced since only two threads will contend for one bank. The thread competition miss can be further reduced by splitting the FB into four banks. However, the capacity miss might be significantly increased if certain thread (application) demands a large number of branch predictions, since the size of one bank is halved or quartered compared to the original FB. Notice that the performance loss can be further mitigated by the large BTB accessed in the next cycle.

In the SMT environment, if a single-ported BTB is adopted, it will hurt the performance due to the high pressure on the BTB. However, a multi-ported BTB will dramatically increase the dynamic and leakage energy consumption, as well as the access latency. In our BK-BAF design, the large BTB is single-ported because the number of accesses to the BTB is much less than that in the original design because of the high filtering effect of our BK-BAF design. Therefore the energy consumption will be further reduced due to the reduced port number. Notice that each FB bank in our BK-BAF design is still multi-ported, that is, two ports in our simulated SMT processor, in order to support multiple branch predictions per cycle.

5.3 Dynamic and leakage energy savings

For the dynamic energy consumption, our banked FB (e.g. 128-entry, direct-mapped for each bank) consumes much less dynamic energy owing to its small size compared to the original BTB (e.g. 4K-entry, four-way set-associative). Moreover, it also consumes less dynamic energy compared to a non-banked FB (e.g. 256-entry, direct-mapped), since the dynamic energy consumption per access is also reduced due to the reduced bank size.

For the leakage energy control, our single-ported BTB in the BK-BAF reduces the leakage energy consumption significantly over the original multi-ported BTB design. Furthermore, the similar drowsy strategy can be applied as discussed for the superscalar processor.

5.4 Performance scalability

In the deeply-pipelined design at new technology generations, the performance scalability issue of the BTB in the SMT environment will be exacerbated because of its large size and the increased port number. Although the banked FB in our BK-BAF is still multi-ported, its access latency can be maintained at one cycle because of its small size. Further, high access latency of the BTB in BK-BAF design can also be reduced due to the single-ported design.

6 Experimental results and analyses

6.1 BAF for superscalar processors

6.1.1 Dynamic energy savings of BAF: To fully exploit the ILP and the speculative execution, the simulated Alpha 21364 microprocessor in this work uses a large BTB which is two-way set-associative with 2K entries. The energy consumption of the large BTB is a big issue in low energy design. If we reduce the size and organisation of the conventional BTB, for example, to a 128-entry direct-mapped BTB, as shown in Fig. 3*a*, the dynamic energy consumption of the BTB is reduced by 92.7% on the average for all benchmarks. However, the energy savings come at an average performance loss of 1.5% as shown in Fig. 3*b*. For some integer benchmarks, such as gap and perlbmk, the performance losses are 10.8 and 7.6%, respectively. Therefore simply reducing the size of the BTB is not always an effective solution for the low-energy design due to the noticeable performance degradation.

Our BAF design intends to minimise the performance loss while significantly reducing the BTB energy consumption. In this work, we choose a 128-entry direct-mapped FB in order to maintain a high filtering rate at low energy consumption. Fig. 4 shows the filtering effectiveness of our BAF. On an average, only 4% of the dynamic branches will eventually access the BTB in our BAF design. Fig. 3*a* shows that the BAF reduces the dynamic energy by 88.5% over a conventional (2K-2W) BTB at a negligible 0.1% performance loss as shown in Fig. 3*b*, an average across all benchmarks. Notice that the additional energy consumption by the FB is included in the BAF.

6.1.2 Effectiveness of leakage control in BAF: For leakage control, we adopt the drowsy strategy [37] for our BAF. BTB entries are periodically put into the drowsy mode and an additional 1-cycle activation delay overhead is assumed when accessing a drowsy BTB entry. We use leakage power numbers provided in [37] for this study. The

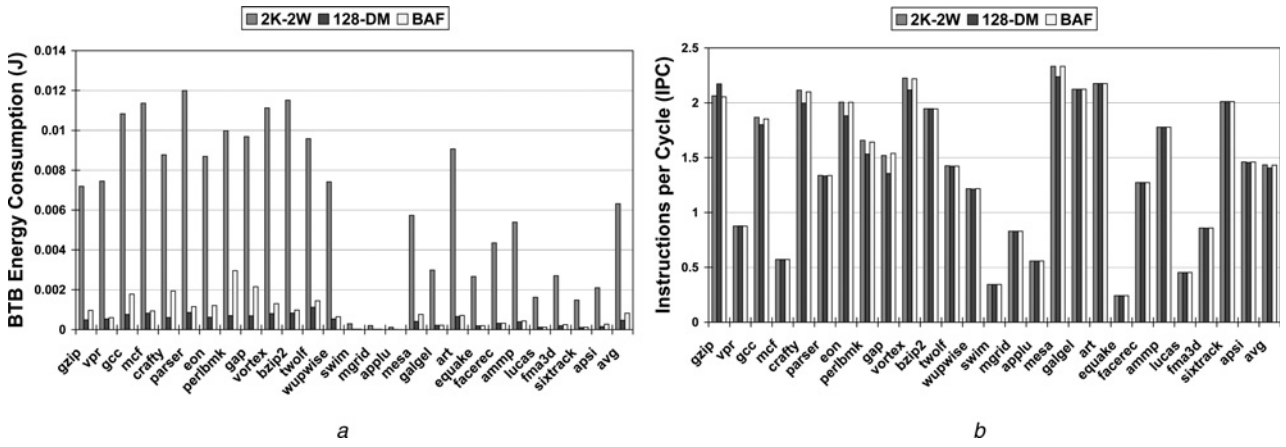

Fig. 3 *Comparison of the dynamic energy consumption and the performance among different BTB designs in the superscalar processor*

a Dynamic energy
b Performance

doi: 10.1049/iet-cdt.2010.0102

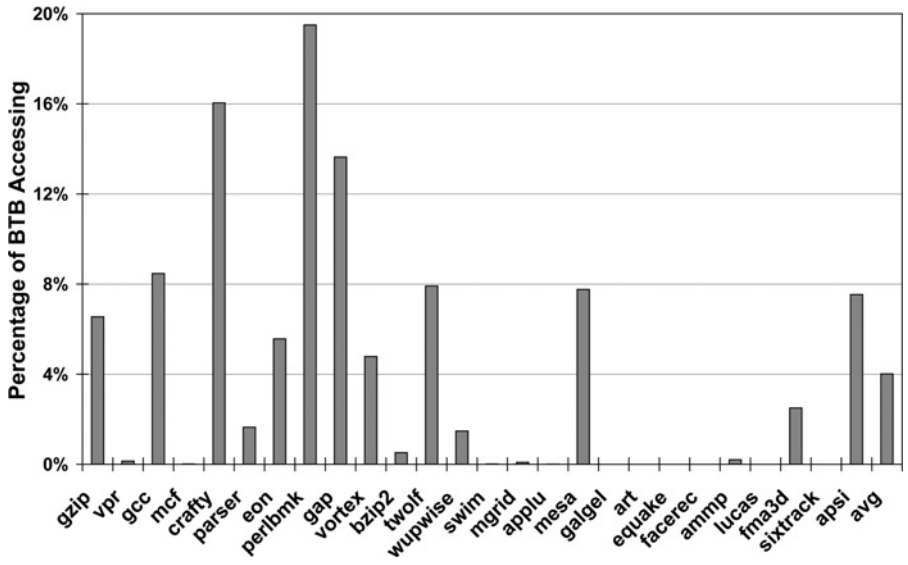

Fig. 4 *Filtering effect of the BAF in the superscalar processor*

leakage energy model of the BTB is given by (1). L_{normal} and L_{drowsy} are the leakage energy consumptions of the BTB entries in normal and drowsy mode, respectively. L_{wakeup} is the additional energy consumed by waking up drowsy entries into the normal mode when they need to be accessed.

$$L_{\text{BTB}} = L_{\text{normal}} + L_{\text{drowsy}} + L_{\text{wakeup}} \tag{1}$$

Fig. 5*a* shows the leakage reduction rates when the drowsy interval increases from 32 to 8K cycles. On an average, integer (floating-point) benchmarks achieve a maximum leakage reduction of 81.8% (83.6%) with the drowsy interval of 128 (256) cycles, at the cost of a minor performance loss of 0.6% (0.03%) as shown in Fig. 5*b*. Our experimental results have demonstrated the superior energy-performance efficiency of our BAF design.

6.1.3 Evaluation of the performance-scalable BAF design: To evaluate the performance scalability of the BAF design at new technology generations, we performed additional experiments with access latencies of 2 and 3 cycles for the large conventional BTB. Similarly, in the BAF, the large BTB (two-way set-associative with 2K-entry) will need 2 or 3 cycles to be accessed at new technology generations and the small FB (direct-mapped with 128-entry) will maintain its 1-cycle access latency.

Our results presented in Fig. 6 show that (i) 2-cycle (2K-2W-2) and 3-cycle (2K-2W-3) conventional BTBs incur significant performance losses of 4.4 and 12% for integer benchmarks compared to an ideal 1-cycle BTB (2K-2W-1); (ii) BAF with a 2-cycle (BAF-2) or 3-cycle BTB (BAF-3) only loses 0.8 or 1.6% performance compared to the ideal BTB, an average for integer benchmarks; and (iii) for floating-point benchmarks, BAF introduces an even more negligible performance loss of 0.04% (0.1%), compared to the 1.6% (3.6%) performance loss in the conventional BTBs when the access latency is increased to 2 (3) cycles. These results further confirm that BAF can be a performance-effective/complexity-effective design for next generation processors.

6.2 BK-BAF for SMT processors

6.2.1 Dynamic energy savings: In our simulated SMT processor, a large 4K-entry four-way set-associative BTB

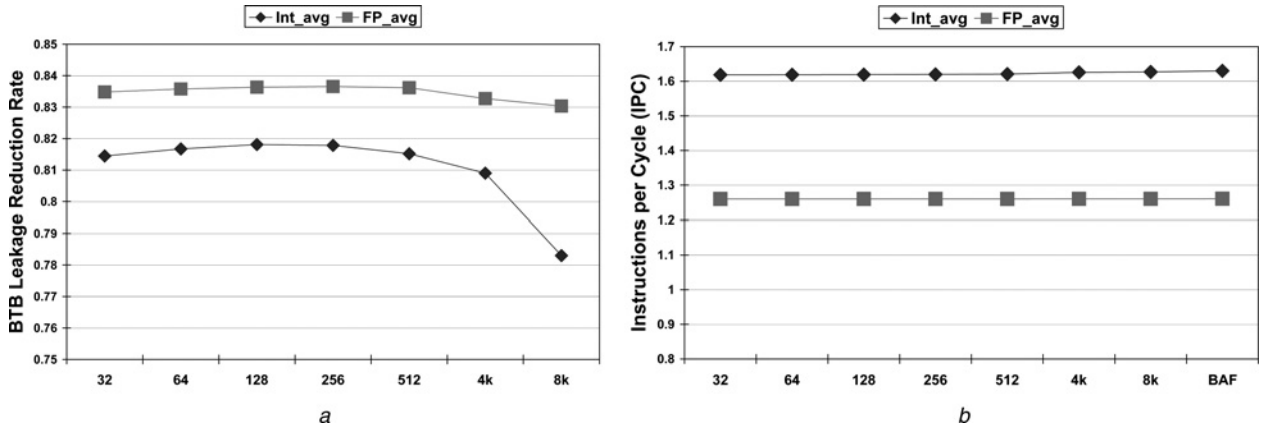

Fig. 5 *BTB leakage energy reduction rates and performance comparison at different drowsy intervals in the superscalar processor*

a Leakage reduction
b Performance

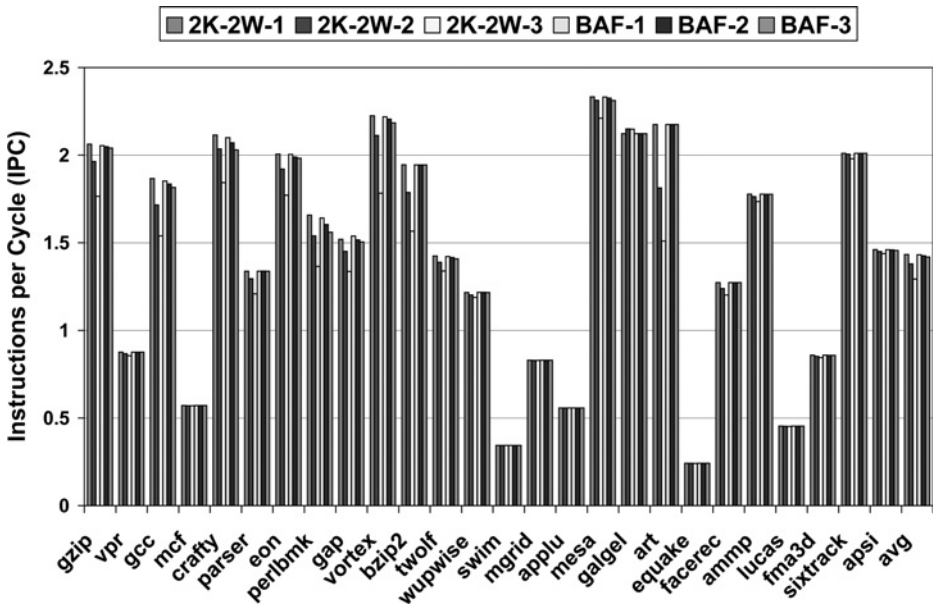

Fig. 6 *Performance scalability of the BAF in the superscalar processor*

with two ports is used. If we reduce the size and organisation of the BTB to a 256-entry direct-mapped BTB with two ports, the dynamic energy consumption will be reduced by 84.6% and the performance will degrade by 7.8% on average, as shown in Fig. 7.

If the BAF design for superscalar processors is adopted for the SMT environment, results in Fig. 7*a* show that a single 256-entry direct-mapped FB with a double-ported BTB (BAF-2P) achieves a 75.8% reduction in dynamic energy consumption and a single FB with an single-ported BTB (BAF-1P) reduces the dynamic energy consumption by 80.1%. Fig. 7*b* shows that the performance degradation for the BAF-2P and BAF-1P is nearly the same, which is about 1.2%. Our BK-BAF design with two 128-entry direct-mapped FB banks and a single-ported BTB (BK-BAF-2B) increases the dynamic energy reduction to 88.1% and reduces the performance loss to 0.86%. If we further divide the FB into four banks (BK-BAF-4B), the dynamic energy reduction slightly increases to 89.1% while the performance loss also increases to 1.1% because of its small bank size (64-entry). Therefore the BK-BAF with two FB banks will be the best choice in our SMT BTB design and will be used as the default configuration in our following study.

6.2.2 Leakage energy savings: The single-ported BTB in our BK-BAF design reduces the leakage energy consumption by 33.5% compared to the original double-ported BTB design. We further conduct the drowsy strategy study on the leakage energy reduction in our BK-BAF design. Our simulation results show that a 4K-cycle drowsy interval achieves a maximum 86.4% leakage energy reduction with only a 0.75% performance loss, which are shown in Fig. 8.

6.2.3 Performance scalability: To evaluate the performance scalability of our BK-BAF design, we conduct similar experiments with access latencies of 2 and 3 cycles for the large BTB in SMT. The small FB banks still maintain 1 cycle access latency. Our simulation results in Fig. 9 show that (i) 2-cycle (4K-4W-2) and 3-cycle (4K-4W-3) conventional BTBs cause dramatic performance degradations of 19.1 and 33.5% compared to an ideal

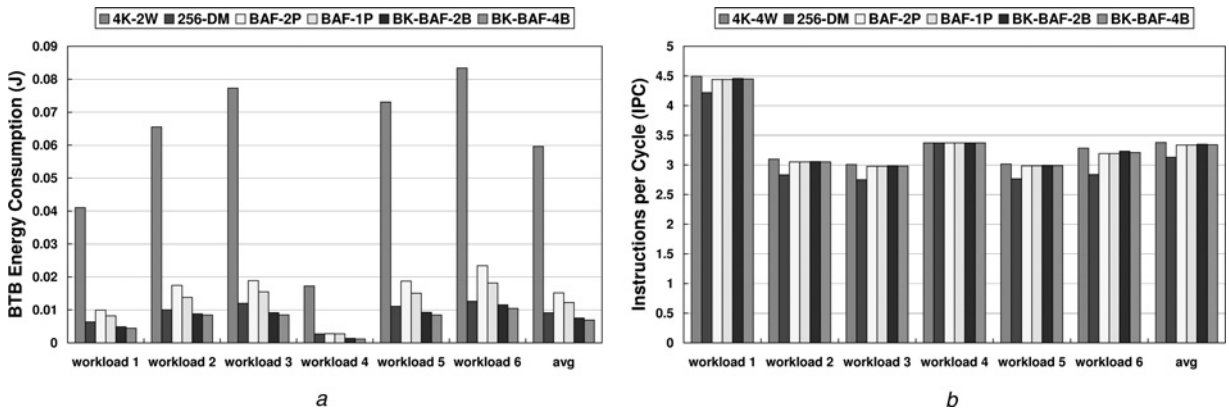

Fig. 7 *Comparison of the dynamic energy consumption and the performance among different BTB designs in the SMT processor*

a Dynamic energy
b Performance

doi: 10.1049/iet-cdt.2010.0102

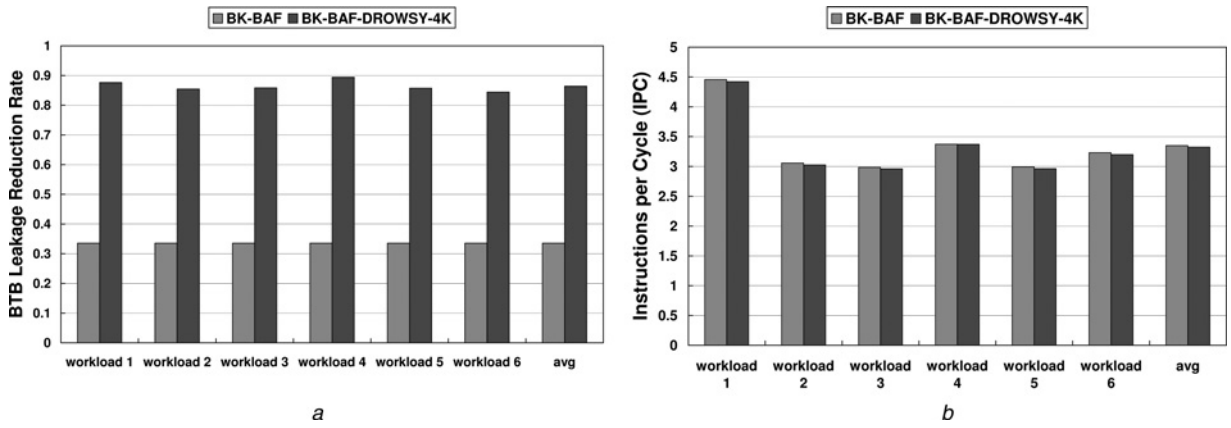

Fig. 8 *Leakage energy reduction rates and the performance impact of the drowsy scheme in the BK-BAF design (single-ported BTB) in the SMT processor*

a Leakage reduction
b Performance

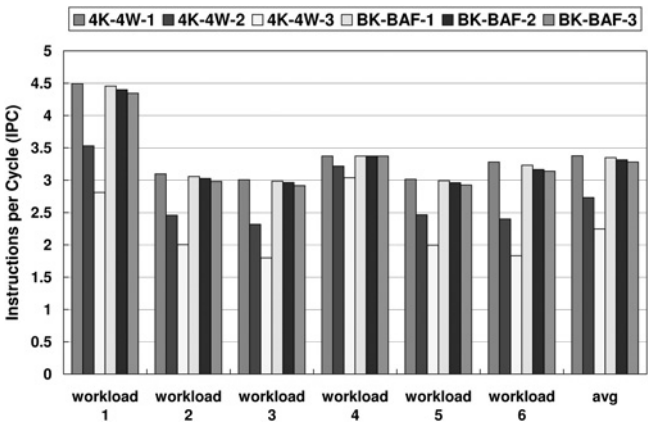

Fig. 9 *Performance scalability of the BK-BAF design in the SMT processor*

1-cycle BTB (4K-4W-1); (ii) BK-BAF with a 2-cycle (BK-BAF-2) or 3-cycle BTB (BK-BAF-3) only incurs 1.8 or 2.9% performance loss compared to the ideal BTB.

7 Conclusions

The exponentially increasing on-chip power density and wire delay at new technology generations demand new efforts to deliver high performance and low energy processor designs. In this work, we focus on the BTB design that dominates the energy consumption of the branch prediction unit. First, we performed a detailed study on the performance and energy tradeoffs of conventional BTBs in superscalar processors. Based on the observed filtering effect of the branch direction predictor, we proposed a BAF design to significantly reduce the dynamic energy consumption in the BTB at minimum performance overhead. The experimental results show that our BAF can achieve an 88.5% dynamic energy reduction with only a 0.1% performance loss. BAF also provides new opportunities for leakage control in the BTB. By employing the drowsy scheme, the BTB leakage energy is reduced by 83% at a negligible 0.3% performance loss. Our BAF design also provides a performance-scalable design at deep sub-micron technologies. In deeply pipelined designs, our BAF can achieve nearly the ideal one cycle access latency. Furthermore, we evaluate our BAF design in the SMT environment and propose a BK-BAF design to improve the energy efficiency and performance. The results show that our BK-BAF with a two-bank FB design reduces the dynamic energy consumption by 88.1% and leakage energy consumption in BTB by 86.4% with drowsy scheme at minimum performance overhead. Our BK-BAF also demonstrates good performance scalability at new technologies. These results confirm us that our proposed BAF can be a practical low energy and high performance BTB design for new generation microprocessors.

8 References

1 Wang, S., Hu, J., Ziavras, S.G.: 'BTB access filtering: a low energy and high performance design'. Proc. IEEE Computer Society Annual Symp. on VLSI, April 2008, pp. 81–86
2 Smith, J.E.: 'A study of branch prediction strategies'. Proc. Eighth Annual Symp. on Computer Architecture, ISCA'81, 1981, pp. 135–148
3 Yeh, T.-Y., Patt, Y.N.: 'Alternative implementations of two-level adaptive branch predictions'. 19th Annual Int. Symp. Computer Architecture, Gold Coast, Australia, May 1992, pp. 124–134
4 McFarling, S.: 'Combining branch predictors'. WRL Technical Note TN-36. Technical report, 1993
5 Jimnez, D.A., Lin, C.: 'Dynamic branch prediction with perceptrons'. Proc. Seventh Int. Symp. on High-Performance Computer Architecture, HPCA'01, 2001, pp. 197–206
6 Perleberg, C., Smith, A.: 'Branch target buffer design and optimization', *IEEE Trans. Comput.*, 1993, **42**, (4), pp. 396–412
7 Chang, Y.-J.: 'Lazy BTB: reduce BTB energy consumption using dynamic profiling'. Proc. 2006 Conf. Asia South Pacific Design Automation, ASP-DAC'06, 2006, pp. 917–922
8 Parikh, D., Skadron, K., Zhang, Y., Stan, M.: 'Power-aware branch prediction: characterization and design', *IEEE Trans. Comput.*, 2004, **53**, (2), pp. 168–186
9 Hily, S., Seznec, A.: 'Branch prediction and simultaneous multithreading'. Proc. Int. Conf. on Parallel Architectures and Compilation Techniques, October 1996, pp. 169–173
10 Raasch, S., Reinhardt, S.: 'The impact of resource partitioning on SMT processors'. Proc. Int. Conf. on Parallel Architectures and Compilation Techniques, 2003, pp. 15–25
11 Ramsay, M., Feucht, C., Lipasti, M.H.: 'Exploring efficient SMT branch predictor design'. Proc. Workshop on Complexity-Effective Design, June 2003
12 Pizzol, G.D., Navaux, P.O.A.: 'Branch prediction topologies for SMT architectures'. Proc. 17th Int. Symp. on Computer Architecture and High Performance Computing, 2005, pp. 118–125
13 Falcon, A., Santana, O.J., Ramirez, A., Valero, M.: 'A latency-conscious SMT branch prediction architecture', *Int. J. High Perform. Comput. Netw.*, 2004, **2**, (1), pp. 11–21
14 Hrishikesh, M.S., Burger, D., Keckler, S.W., Shivakumar, P., Jouppi, N.P., Farkas, K.I.: 'The optimal logic depth per pipeline stage is 6 to 8

FO4 inverter delays'. Proc. 29th Annual Int. Symp. on Computer Architecture, May 2002, pp. 14–24
15 Palacharla, S., Jouppi, N.P., Smith, J.: 'Complexity-effective superscalar processors'. Proc. 24th Annual Int. Symp. on Computer Architecture, June 1997, pp. 206–218
16 Ernst, D., Hamel, A., Austin, T.: 'Cyclone: a broadcast-free dynamic instruction scheduler selective replay'. Proc. 30th Annual Int. Symp. Computer Architecture, June 2003, pp. 235–262
17 Canal, R., Gonzalez, A.: 'Reducing the complexity of the issue logic'. Proc. 2001 Int. Conf. on Supercomputing, June 2001, pp. 312–320
18 Wallace, S., Bagherzadeh, N.: 'A scalable register file architecture for dynamically scheduled processors'. Proc. 1996 Conf. on Parallel Architectures and Compilation Techniques, 1996, pp. 179–184
19 Park, I., Powell, M., Vijaykumar, T.: 'Reducing register ports for higher speed and lower energy'. Proc. Int. Symp. on Microarchitecture, December 2002, pp. 171–182
20 Tseng, J., Asanovic, K.: 'Banked multiported register files for high-frequency superscalar microprocessors'. 30th Int. Symp. on Computer Architecture (ISCA-30), San Diego, CA, June 2003, pp. 62–71
21 Hu, Z., Juang, P., Skadron, K., Clark, D., Martonosi, M.: 'Applying decay strategies to branch predictors for leakage energy savings'. Proc. 2002 Int. Conf. Computer Design, September 2002, pp. 442–445
22 Huang, M.C., Chaver, D., Pinuel, L., Prieto, M., Tirado, F.: 'Customizing the branch predictor to reduce complexity and energy consumption', *IEEE Micro*, 2003, **23**, (5), pp. 12–25
23 Petrov, P., Orailoglu, A.: 'Low-power branch target buffer for application-specific embedded processors'. Proc. Euromicro Symp. on Digital Systems Design, DSD'03, 2003, pp. 158–165
24 Seznec, A., Felix, S., Krishnan, V., Sazeides, Y.: 'Design tradeoffs for the alpha EV8 conditional branch predictor', *ACM SIGARCH Comput. Architect. News*, 2002, **30**, (2), pp. 295–306
25 Kin, J., Gupta, M., Mangione-Smith, W.H.: 'The filter cache: an energy efficient memory structure'. Proc. Annual ACM/IEEE Int. Symp. on Microarchitecture, 1997, pp. 184–193
26 Kin, J., Gupta, M., Mangione-Smith, W.H.: 'Filtering memory references to increase energy efficiency', *IEEE Trans. Comput.*, 2000, **49**, (1), pp. 1–15
27 Casazza, J.: 'First the tick, now the tock: Intel microarchitecture (Nehalem)' (Intel White Paper, 2008)
28 Webb, C.F.: 'Ibm z10: the next-generation mainframe microprocessor', *IEEE Micro*, 2008, **28**, (2), pp. 12–19
29 Burger, D., Austin, T.M.: 'The SimpleScalar tool set, version 2.0′'. Technical report 1342, Computer Sciences Department, University of Wisconsin, 1997
30 Bannon, P.: 'Alpha 21364: a scalable single-chip SMP'. Microprocessor Forum, 1998
31 Brooks, D., Tiwari, V., Martonosi, M.: 'Wattch: a framework for architectural-level power analysis and optimizations'. Proc. Int. Symp. on High-Performance Computer Architecture, 2000, pp. 83–94
32 Preston, R.P., Badeau, R.W., Bailey, D.W., *et al.*: 'Design of an 8-issue superscalar RISC microprocessor with simultaneous multithreading'. Proc. IEEE Int. Solid-State Circuits Conf., 2002
33 Tullsen, D., Eggers, S., Emer, J., Levy, H., Lo, J., Stamm, R.: 'Exploiting choice: instruction fetch and issue on an implementable simultaneous multithreading processor'. Proc. 22nd Annual Int. Symp. on Computer Architecture, May 1996, pp. 191–202
34 Kessler, R.E.: 'The alpha 21264 microprocessor', *IEEE Micro*, 1999, **19**, (2), pp. 24–36
35 Sherwood, T., Perelman, E., Hamerly, G., *et al.*: 'Automatically characterizing large scale program behavior'. Proc. ASPLOS X, October 2002, pp. 45–57
36 Borch, E., Tune, E., Manne, S., Emer, J.: 'Loose loops sink chips'. Proc. HPCA-8, February 2002, pp. 270–281
37 Flautner, K., Kim, N., Martin, S., Blaauw, D., Mudge, T.: 'Drowsy caches: simple techniques for reducing leakage power'. Proc. 29th Int. Symp. on Computer Architecture, Anchorage, AK, May 2002, pp. 148–157
38 Kaxiras, S., Hu, Z., Martonosi, M.: 'Cache decay: exploiting generational behavior to reduce cache leakage power'. Proc. Int. Symp. on Computer Architecture, 2001, pp. 240–251
39 Tullsen, D., Eggers, S., Levy, H.: 'Simultaneous multithreading: maximizing on-chip parallelism'. Proc. 22nd Annual Int. Symp. Computer Architecture, June 1995, pp. 392–403
40 Eggers, S., Emer, J., Levy, H., Lo, J., Stamm, R., Tullsen, D.: 'Simultaneous multithreading: a platform for next-generation processors', *IEEE Micro*, 1997, **17**, (5), pp. 12–19
41 Weglarz, E., Saluja, K., Lipasti, M.: 'Minimizing energy consumption for high-performance processing'. Proc. Asia and South Pacific Design Automation Conf., 2002, pp. 199–204
42 Falcon, A., Santana, O.J., Ramirez, A., Valero, M.: 'Tolerating branch predictor latency on SMT', *Lect. Notes Comput. Sci.*, 2003, **2585**, pp. 86–98

doi: 10.1049/iet-cdt.2010.0102