MALEC: A Multiple Access Low Energy Cache

Matthias Boettcher
University of Southampton
Southampton, UK
mb1e09@ecs.soton.ac.uk

Giacomo Gabrielli
ARM Ltd.
Cambridge, UK
giacomo.gabrielli@arm.com

Bashir M. Al-Hashimi
University of Southampton
Southampton, UK
bmah@ecs.soton.ac.uk

Danny Kershaw
NXP Semiconductors
Graz, Austria
daniel.kershaw@nxp.com

***Abstract*—This paper addresses the dynamic energy consumption in L1 data cache interfaces of out-of-order superscalar processors. The proposed Multiple Access Low Energy Cache (MALEC) is based on the observation that consecutive memory references tend to access the same page. It exhibits a performance level similar to state of the art caches, but consumes approximately 48% less energy. This is achieved by deliberately restricting accesses to only 1 page per cycle, allowing the utilization of single-ported TLBs and cache banks, and simplified lookup structures of Store and Merge Buffers. To mitigate performance penalties it shares memory address translation results between multiple memory references, and shares data among loads to the same cache line. In addition, it uses a Page-Based Way Determination scheme that holds way information of recently accessed cache lines in small storage structures called way tables that are closely coupled to TLB lookups and are able to simultaneously service all accesses to a particular page. Moreover, it removes the need for redundant tag-array accesses, usually required to confirm way predictions.**

For the analyzed workloads, MALEC achieves average energy savings of 48% in the L1 data memory subsystem over a high performance cache interface that supports up to 2 loads and 1 store in parallel. Comparing MALEC and the high performance interface against a low power configuration limited to only 1 load or 1 store per cycle reveals 14% and 15% performance gain requiring 22% less and 48% more energy, respectively. Furthermore, Page-Based Way Determination exhibits coverage of 94%, which is a 16% improvement over the originally proposed line-based way determination.

I. Introduction

Modern out-of-order superscalar processors rely on speculative execution, wide issue windows and multiple data paths to exploit instruction-level parallelism, and on memory hierarchies comprising fast on-chip SRAM caches to improve memory access latency and throughput. The high contribution of memory references to instruction streams as demonstrated in Sec. III, makes it vital for first-level data caches to support multiple accesses in parallel. However, finite energy sources and limitations in semiconductor technology scaling result in fixed energy budgets for mobile devices; in addition, cooling and electricity costs are becoming increasingly important in the desktop and server segments [4]. As a consequence, energy efficiency has become a determining factor for microprocessors and one of the main obstacles to performance improvements. As caches are one of the main contributors to the on-chip energy consumption [13], cache-level optimisations need to trade-off increased performance with degraded energy efficiency.

This paper addresses the problem of implementing multiple-access L1 data caches in an energy efficient manner. State of the art microarchitectures such as Intel's Sandy Bridge and AMD's Bulldozer allow up to two 128-bit loads and one 128-bit store per cycle [1] [2]. Both rely on physical multi-porting and cache banking (Sec. II). Next generation processors will require even more sophisticated caches to handle aggressive memory speculation and disambiguation techniques. The Multiple Access Low Energy Cache (MALEC) builds upon cache banking, but avoids the disadvantages of physical multi-porting in terms of latency, area and energy consumption. It is based on the observation that consecutive memory references tend to access the same page. Our key contributions are:

- Page-Based Memory Access Grouping:
  - restriction to access only one page per cycle; thus allowing single-ported TLBs and cache banks
  - simplified lookup structures for Store and Merge Buffers that share results of page comparisons
  - re-use of page address comparisons when merging loads that access the same cache line
- Page-Based Way Determination:
  - use of TLB lookups to index way information
  - service all accesses to a specific page simultaneously
  - use of validity information to allow the bypassing of tag-arrays and directly access desired data-arrays

Sec. II and III list related work and justify our assumptions on memory access patterns. Details on Page-Based Memory Access Grouping and Page-Based Way Determination are given in Sec. IV and V, respectively. Sec. VI then evaluates our proposals before this paper concludes in Sec. VII.

II. Related Work

Juan et al. [11] examined well known techniques to realize multiple-access caches; namely, physical multi-porting, banking, virtual multi-porting and redundant copies. Hybrids of the first two techniques achieve good performance-energy trade-offs and are common in modern superscalar processors [1] [2]. Sec. VI compares our proposal, which utilizes cache banking but does not rely on physical multi-porting, against such a hybrid baseline. Recent proposals regarding the reduction of leakage energy in caches, i.e. thanks to cache bank clock gating or cache decay, are considered orthogonal to MALEC and might be implemented in conjunction with it [7]. This is also the case for TLB specific optimizations such as Chan and Lan's banking and filtering schemes [5].

978-3-9815370-0-0/DATE13/©2013 EDAA

Although a particular datum can only be located in one way of an n-way set-associative cache, a conventional lookup requires n tag- and data-array accesses. Techniques that attempt to avoid redundant accesses may be categorized as "way prediction", "way estimation" and "way determination". The first category makes predictions based on MRU statistics [9] [12]. While it is simple to implement, false predictions require a second cache access to find the desired datum. Powell et al. [16] mitigate this problem by increasing prediction accuracy based on a combination of selective direct-mapping and way-prediction. Way estimation techniques deliver multiple instead of a single way [6] [8]. If the desired data resides within the cache, it is guaranteed to be found within the given set of ways. Consequently, no additional cycles are introduces, but energy might be consumed for several redundant tag comparisons. Nicolaescu et al.'s Way Determination Unit (WDU) stores way information of recently accessed cache lines in a small buffer [15]. Each line is associated with exactly one way and guaranteed to hit there or miss the whole cache. Based on this scheme, we propose Page-Based Way Determination and compare it against an adapted WDU implementation in Sec. V and VI-C, respectively. As our scheme operates on page rather than line granularity, it can simultaneously service all accesses to the same page and re-use TLB lookups to index its way information. Furthermore, it uses validity information to eliminate the need for tag-array accesses to confirm predictions.

III. Motivation

To estimate potential benefits from multiple-access data caches, we analysed the most representative execution phase (1 billion instructions identified by Simpoint v3.0) of SPEC CPU2000 and MediaBench2 benchmarks. We observed an average contribution of memory references to the overall instruction count of 40% and a load/store ratio of 2/1. Fig. 1 illustrates the number of consecutive read accesses to the same page, allowing n intermediate accesses to a different page. The bar colours represent - from dark to light grey - groups of 1, 2, 3 to 4, 5 to 8, and more than 8 consecutive accesses. Hence, a primarily light coloured bar indicates a very high page locality. On average, 70% of all loads are directly followed by one or more loads to the same page. Allowing one, two or three intermediate access to a different page increases this ratio to 85%, 90% or 92%, respectively. Consequently, the majority of loads are suitable for Page-Based Memory Access Grouping and Page-Based Way Determination introduced in Sec. IV and V. Stores show an even higher spatial locality. However, as this property is already exploited by Merge Buffers, it is not further discussed here. Focusing on line rather than page granularity reveals that 46% of loads are directly followed by one or more loads to the same line. MALEC exploits this observation by allowing multiple loads to share data read from the L1 cache.

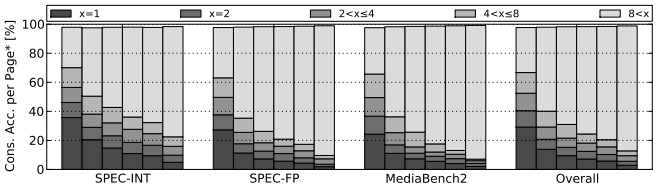

Fig. 1. Number of Consecutive Accesses to the same Page, allowing - left to right - 0, 1, 2, 3, 4 and 8 intermediate Access to a different Page

IV. Page-Based Access Grouping

Fig. 2a and 2b give a high level description of MALEC and its operation, emphasizing novel and modified elements in black and dark grey, respectively. To demonstrate its scalability, Fig. 2a is parameterized to service up to four loads and two stores in parallel. Multiple parallel L1 accesses are achieved by distributing accesses to independent cache banks and merging loads to the same line (Arbitration Unit; MB to merge stores). Energy savings result from the limitation to access only one page per cycle (Input Buffer), allowing the utilisation of single-ported components (i.e. uTLB, TLB and L1 banks) and simplified Store (SB) and Merge Buffer (MB) lookup structures. To reduce the performance impact of Page-Based Memory Access Grouping, accesses are prioritized and potentially re-ordered (Input Buffer) and newly introduced latencies overlapped with address translations.

Fig. 2a includes several components common in state of the art memory interfaces; e.g. a Translation Lookaside Buffer (TLB) and a micro TLB (uTLB) to facilitate address translations. The latter is not strictly required by MALEC, but increases the efficiency of Page-Based Way Determination (Sec. V). Store and Merge Buffers allow the speculative execution of stores and reduce the number of L1 cache accesses by merging data from multiple stores to the same address region, respectively. Four independent 4-way set-associative single-ported cache banks, each holding data corresponding to a specific address region, allow up to four parallel cache accesses with moderate miss rates. The utilized physically indexed, physically tagged (PIPT) L1 data cache and the serialized address translation and data access are common in energy sensitive systems. Sec. VI evaluates the effects of alternative setups.

Input Buffer: Stores finishing address computation are sent to the SB and remain there until they commit into the MB (Fig. 2b). Evicted MB entries (MBEs) and loads are forwarded to the Input Buffer, which prioritizes them and identifies groups accessing the same page. From high to low priority, it includes up to: three loads from previous cycles, four loads finishing address computation, and one evicted MBE (not time critical, as corresponding stores already committed). Each cycle, the virtual page ID (vPageID) of the highest priority entry is passed to the uTLB for address translation and simultaneously compared against all remaining, currently valid entries. Matching entries are then passed to the Arbitration Unit. Unmatched loads and those rejected by the Arbitration Unit are held until the next cycle. Should the Input Buffer's storage elements be insufficient, one or more address computation units are stalled.

Arbitration Unit: Based on the current selection of Input Buffer elements, the Arbitration Unit selects up to four loads

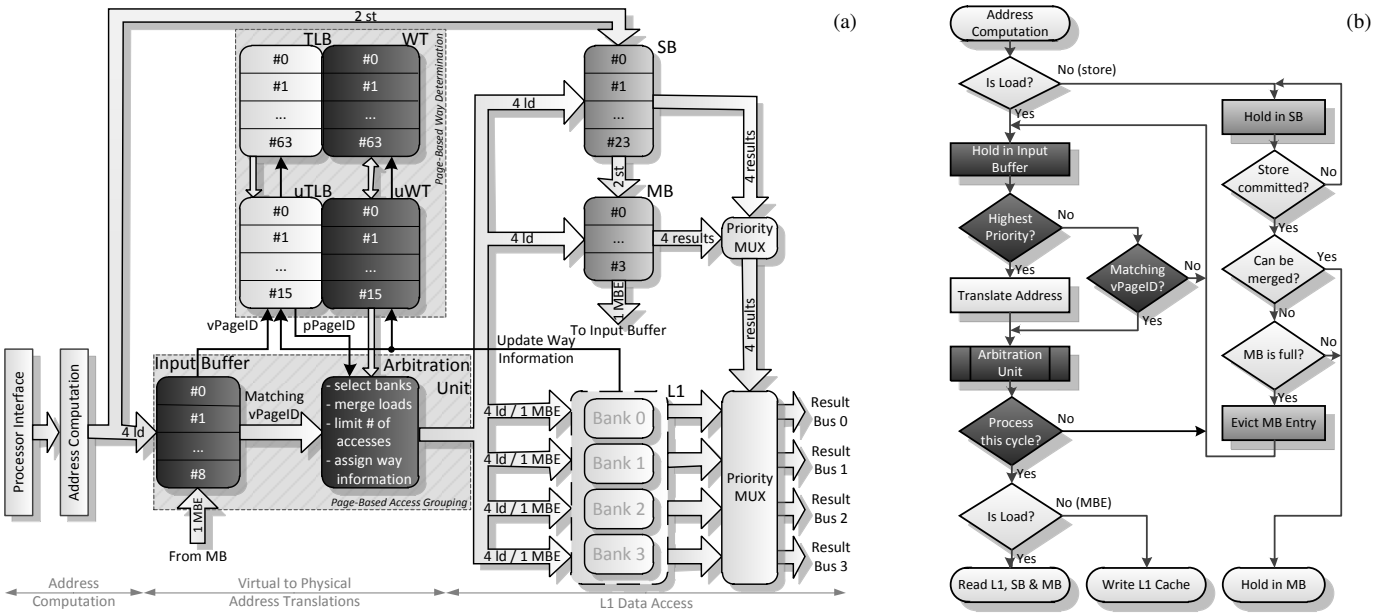

Fig. 2. MALEC Configuration servicing up to four Loads and two Stores in parallel(a); High Level Descriptions of Load and Store Handling (b)

and one MBE to be serviced. For each cache bank, it identifies the access with the highest priority given by its order within the Input Buffer. In case of a load, it performs partial address comparisons to identify other loads to the same cache line. To reduce the energy impact of those comparisons, only the three loads consecutive to the initial Input Buffer entry are evaluated. The performance degradation due to this limitation is less than 0.5% (analysis based on Sec. VI). Considering that all loads/MBEs are known to share one page ID, the actual comparators are very narrow and therefore fast and energy efficient ($comparator_{bit}$ = $address\_space_{bit}$ - $pageID_{bit}$ - $line\_offset_{bit}$). Finally, to account for the limited number of available result busses, only the four highest priority loads are selected. An alternative, but more complex and potentially less efficient, approach might determine the combination of loads accessing the least number of cache banks.

SB, MB and L1: Memory accesses selected by the Arbitration Unit are sent to the L1 cache and in case of loads to the SB and MB, too. The cache itself is unmodified to allow the re-use of existing, highly optimized designs. A special case are sub-blocked caches that split data-arrays in several independent segments (e.g. 128 bit wide) to save dynamic power. MALEC expects those to return data from two adjacent sub-blocks on every read, instead of only for loads that exceed one sub-block. This requires additional comparator bits within the Arbitration Unit, but doubles the probability for loads to be merged. To reduce the energy impact of additional SB and MB ports required to service up to four parallel loads, their lookup structures are split into a shared segment for page IDs and four separate segments for the remaining address bits. As both segments can be looked up simultaneously, they do not impose additional latency.

WT and uWT: Way Tables are not essential for MALEC, but provide further energy savings as explained in Sec. V.

V. Page-Based Way Determination

Fig. 3 depicts the components involved in Page-Based Way Determination, i.e. TLBs, WTs and the L1 cache. WTs are closely coupled to their respective address translation component; e.g. TLB hits return corresponding WT entries in addition to address translation results. Consequently, the energy per page-wide address lookup (20 bit for 4 KByte pages and a 32 bit address space) is split over both structures. The number of entries per WT, and therefore the number of pages covered by it, equals the number of entries within the corresponding TLB. Each entry can service all memory references to a particular page simultaneously. Based on validity information, the scheme allows the majority of cache accesses (94%, Sec. VI-C) to bypass the cache's tag-array. This distinguishes it from other prediction schemes that need to verify their results with at least one tag comparison (Sec. II). Hence, MALEC supports two different cache access modes:

- Conventional cache access (way unknown):
  - Parallel access to all tag- and all data-arrays
  - Select data associated with matching tag
- Reduced cache access (way known and valid):
  - Tag–arrays bypassed
  - Access to one specific data-array only

The format used for WT entries (Fig. 3) combines validity and way information within 2 bits per cache line, reducing

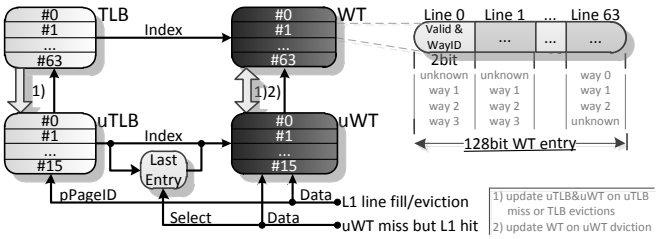

Fig. 3. Overview of Page-Based Way Determination

area and leakage power by $1/3$ compared to the naive format that uses separate bit fields; i.e. 128bit instead of 192bit for 64 lines per page (4 KByte pages, 64 Byte cache lines). A cache consisting of four banks may allocate lines 0..3 to separate banks and lines 0, 4, 8, .., 60 to the same bank. By deeming way 0 and 1 as "way unknown" for lines 0..3 and lines 4..7, respectively, we limit the number of available ways for a particular cache line to 3, allowing 2 bits to encode way and validity information. Note, while a single line is limited to three cache ways, working sets may still utilize all four ways. In fact, simulations based on Sec. VI show no measurable increase of the L1 miss rate due to this limitation.

Validity bits are set/reset on cache line fills/evictions. Although the WT includes all uWT entries, it is only updated if no corresponding uWT entry was found. The synchronization of uWT and WT is based on full entries. To reduce the number of those transfers, we chose the second chance algorithm as the uTLB replacement policy (random replacement for the TLB). Because the cache performs line fills and evictions based on physical tags, the uTLB and TLB need to be modified to allow lookups based on physical, in addition to virtual, PageIDs. Furthermore, the finite number of TLB entries (64, Fig. 3) might require the eviction of a page that still has corresponding lines within the cache. If the page is re-accessed later on, a new WT entry is allocated and all way information invalidated. To compensate for the loss of information, the uWT is updated if it returns way unknown but a subsequent conventional cache access hits. The last entry register (Fig. 3) allows the feedback of those accesses to update the corresponding uWT entry without a prior uTLB lookup. In case of a multi cycle delay between way prediction and L1 access, the register may be implemented as a FIFO (the uTLB's second chance replacement policy thereby prevents the eviction of relevant entries). Simulations based on Sec. VI show that this update mechanism increases the coverage of Page-Based Way Determination from 75% to 94%.

The Arbitration Unit (Sec. IV) assigns way information to groups of memory references accessing the same cache line, and forwards it to the corresponding cache banks. As the maximum number of ways required equals the number of available cache banks, the energy consumed to evaluate WT entries is independent of the number of memory references to be serviced in parallel; hence, the scheme is highly scalable.

VI. Evaluation

A. Methodology

The gem5 Simulator System was extended to allow the performance evaluation of MALEC with cycle-level accuracy. The resulting access statistics were then combined with energy estimates obtained from CACTI v.6.5 [14] to determine the dynamic and static energy consumption of the following structures: L1 data cache (tag&data-arrays and control logic), uTLB+uWT and TLB+WT. To account for reverse lookups (i.e. based on physical addresses), uTLB and TLB are treated as two separate fully associative tag-arrays for their uWT/WT data-array. While conventional address translations access both tag-arrays, uWT/WT updates following cache line fills and evictions only access the physical tag-array. Furthermore, as the energy consumption of LQ, SB, and MB is very similar for all analyzed configurations it is not taken into account. This is also the case for lower memory levels, as MALEC alters the timing of L2 accesses, but does not significantly impact their number or miss rate. The Input Buffer utilized in this section (five 20 bit comparators and storage for up to two loads) is smaller than the uWT, which contributes to only 0.3% and 2.1% of the overall leakage and dynamic energy consumption, respectively. Hence, the energy contribution of the Input Buffer and the even smaller Arbitration Unit are considered negligible.

TABLE I
Basic Configurations

	Addr. Comp. per Cycle	uTLB/TLB ports	Cache Ports
Base1ldst	1 ld/st	1 rd/wt	1 rd/wt
Base2ld1st	2 ld + 1 st	1 rd/wt + 2 rd	1 rd/wt + 1 rd
MALEC	1 ld + 2 ld/st	1 rd/wt	1 rd/wt

Tab. I characterizes the configurations analyzed in this section. Although both baselines resemble state of the art caches, *Base1ldst* relies on single-ported components to ensure high energy efficiency, while the performance oriented *Base2ld1st* utilizes physical multi-porting in addition to cache banking (Sec. II). The analyzed MALEC configuration (Tab. I) is parameterized to fit the underlying processor-cache interface, which was optimized for *Base2ld1st* (Tab. II). Moreover, to allow fair comparisons, it operates on the same number of LQ, SB and MB ports as well as address computation units as *Base2ld1st*. While the latencies introduced by our proposal, do not actually induce additional CPU cycles, $\text{MALEC}_{3cycleL1}$ and $\textit{Base2ld1st}_{1cycleL1}$ widen the scope of this evaluation by illustrating the impact of variations in the L1 access latency (2 ± 1 cycle). The energy values given for $\textit{Base2ld1st}_{1cycleL1}$ represent a best case scenario, because they are based on the same (slow, but low energy) transistors used for all other configurations and do not account for additional circuitry required for parallel TLB and L1 lookups. The analyzed benchmark suites and execution phases are equivalent to those used in Sec. III.

B. Performance

Fig. 4a depicts execution times normalized to *Base1ldst*. The analyzed MALEC configuration achieves a performance improvement of 14% over *Base1ldst*. This is only 1% less than *Base2ld1st*, which relies on a physically multi-ported

TABLE II
Relevant Simulation Parameters

Component	Parameter
Processor	single-core, out-of-order, 1 GHz clock, 168 ROB entries, 6 element fetch&dispatch-width, 8 element issue-width
L1 interface	64 TLB entries, 16 uTLB entries, 40 LQ entries, 24 SB entries, 4 MB entries, 32 bit addr. space, 4 KByte pages
L1 D-cache	32 KByte, 2 cycle latency, 64 byte lines, 4-way set-assoc., 4 independent banks, PIPT, 128 bit sub-blocks per line
L2 cache	1 MByte, 12 cycle latency, 16-way set-assoc.
DRAM	256 MByte, 54 cycle latency
CACTI	32nm node, design objective low dyn. power, cell types: low standby power data&tag-arrays, high perform. peripherals

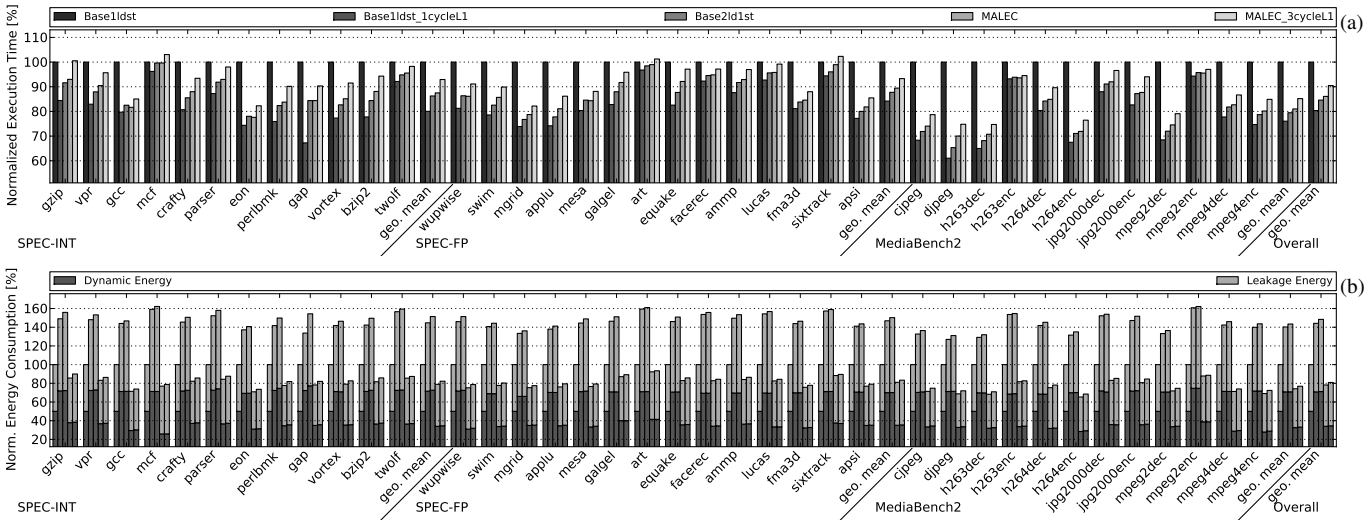

Fig. 4. Normalized Execution Times and Energy Consumption of - left to right - *Base1ldst*, *Base2ld1st*$_{1cycleL1}$, *Base2ld1st*, MALEC and MALEC$_{3cycleL1}$

uTLB, TLB and cache. The additional cycle introduced for MALEC$_{3cycleL1}$ reduces this average benefit from 14% to 10%, and even results in a performance degradation of up to 4% for mcf. Contrarily, the single-cycle variant of *Base2ld1st* increases performance benefits from 15% to 20%. The exceptionally high improvements (≈17%) for gap are based on a high proportion of loads (≈37%) on the overall instruction count combined with the frequent execution of instruction sequences that exhibit dependencies that prevent re-ordering.

Comparing SPEC-Int, SPEC-FP and MB2 reveals performance improvements of 14%, 12% and 21%, respectively, and a relatively high sensitivity to L1 access latency variations for SPEC-Int. One reason for the increased benefits of SPEC-Int over SPEC-FP is the higher contribution of memory accesses within this suite; i.e. 45% instead of 40%. Although this ratio is actually lower for MB2 (37%), its media kernels rely on frequent, highly structured memory accesses that benefit more from cache banking and load merging. Exceptionally low improvements over *Base1ldst* are shown by mcf and art. Reasons for this are large working sets combined with low memory access locality leading to high miss rates and a reduced influence of faster L1 accesses. Contrarily, djpeg and h263dec exhibit excellent access locality and tend to execute numerous memory accesses in parallel, resulting in speedups of ≈30% for MALEC.

The performance benefits provided by MALEC originate from two mechanisms: merging of loads and accessing multiple cache banks in parallel. Both mechanisms require multiple address translations per cycle, and therefore take advantage of the ability to share translation results among accesses to the same page. On average, merged loads contributes approximately 21% to MALEC's overall performance improvement. The benchmarks gap and equake achieve significantly higher percentages of 56% and 66%, due to particularly suitable memory access patterns. Contrarily, mgrid exhibits a value of less than 2%, implying loads with a relatively low spatial (intra cache line) or temporal locality. In summary, the low performance penalty of MALEC over *Base2ld1st* confirms Sec. III's assumption that it is sufficient to handle only instructions accessing the same page within one cycle.

C. Energy Consumption

The dynamic and overall energy consumption of the configurations analyzed in the previous section are illustrated in Fig. 4b. *Base2ld1st* exhibits an increase in dynamic energy of 42%. The primary cause of this are the additional physical ports of its uTLB, TLB and L1 cache (Table I), required to achieve its high performance operation. Contrarily, MALEC saves 33% of dynamic energy compared to *Base1ldst*. The unusually high savings for mcf originate in the exceptionally high miss rate of the benchmark (≈ $7x$ the average). As MALEC attempts to share L1 data among loads addressing the same cache line, the effective number of loads accessing and missing the cache is reduced. Without this ability, MALEC would actually consume 5% more instead of 51% less dynamic energy for mcf. Considering both, dynamic and leakage energy, *Base2ld1st*'s average energy consumption actually lies 48% above *Base1ldst* (Fig. 4b). Reason for this is the leakage introduced by additional uTLB, TLB and L1 ports, which outweighs savings due to reduced computation times; e.g. the additional rd port increases L1 leakage by ≈80%, but the average computation time is only reduced by 15%. A similar effect can be observed for MALEC. Although it has the same number of uTLB, TLB and L1 ports as *Base1ldst*, it's uWT and WT induce additional leakage, and thereby reduce its overall energy saving to 22% (48% relative to *Base2ld1st*).

The dynamic energy consumption of the 1- and 3-cycle latency adaptations is almost equivalent to their corresponding baseline configurations. An exception is the improvement of 5% for gap thanks to *Base2ld1st*$_{1cycleL1}$, which indicates a reduced number of memory accesses. One explanation for this is the faster computation of branch outcomes and targets, due to shorter load latencies reducing the number of speculatively executed instructions. In particular, while the number of committed loads is unaffected by *Base2ld1st*$_{1cycleL1}$, the number of executed loads is ≈ 9% lower than for *Base2ld1st*. Note, this

effect is only partially mitigated by a slightly increased L1D miss-rate (+12% relative to *Base2ld1st*).

Page-Based Way Determination is based on Nicolaescu et al.'s Way Determination Unit (WDU, Sec. II). To allow fair comparisons between both schemes, we extended the WDU with validity bits to allow reduced cache accesses (Sec. V). Substituting WTs with 8, 16 and 32 entry-WDUs reveals 4%, 5% and 8% higher energy consumption. There are two reasons for this. First, contrarily to the single-ported, lookup free WTs, WDUs require four fully associative, tag-sized lookup ports to support this specific MALEC configuration. Second, the analyzed WDUs exhibit average coverage of only 68%, 76% and 78% for 8, 16 and 32 entries, respectively. In comparison to 94% for MALEC with WTs, this implies a significantly lower number of reduced cache accesses. Generally speaking, thanks to their small storage requirements, WDUs are suitable for processor configurations similar to *Base1ldst* that are designed for single-access caches. Contrarily, as the energy consumption of page-based way determination is independent of the number of memory references to be serviced in parallel (Sec. V), its energy efficiency increases rather than decreases on high performance processors.

D. Sensitivity Analysis

This section briefly discusses MALEC's sensitivity to system parameters. First, the efficiency of way prediction schemes strongly depends on access locality; i.e. for streaming applications like mcf, Page-Based Way Determination exhibits negative energy benefits. This dependency has been investigated by prior work in the context of cache pollution; hence, schemes like run-time cache bypassing might be applied [10]. These would also reduce the number of uTLB/TLB conflicts due to frequent uWT/WT updates required by workloads with very high miss rates. Second, although MALEC can mask most of its latency behind address translations, it introduces variability in load latency, by potentially holding Input Buffer elements for several cycles. For the analysis above, we utilized a conservative LQ implementation that does not issue instructions that depend on loads until they are selected to be serviced by the Arbitration Unit. More aggressive systems may issue such instructions speculatively, i.e. assume a certain latency, hold results if they arrive early, and trigger replays similarly to load misses if necessary. In this context, the speculative state held by MALEC is limited to the Input Buffer, the Arbitration Unit and the SB; hence, exception handling mechanisms do not need to consider way tables or the MB. Third, while the previous section focuses on a single-cores system, we consider multithreading and potential synchronization or memory sharing issues as an orthogonal aspect to our research. Finally, Page-Based Memory Access Grouping and Page-Based Way Determination scale well with most cache parameters, e.g. capacity, line size, associativity, number of banks, and available address space. In fact, MALEC's performance is primarily limited to the number of memory references issued per cycle and the number of available result busses. A special case are wider pages, which increase the number of lines per WT entry. Hence, we suggest to quantize TLB entries into 4 KByte segments and potentially apply techniques like [3] that favour small pages. Alternatively, the WT itself might be segmented. By allocating and replacing WT chunks in a FIFO or LRU manner, their number could be smaller than required to represent full pages.

VII. Conclusions

This paper presented an energy efficient L1 data cache interface designed for out-of-order superscalar processors. Based on the observation that consecutive memory references tend to access the same page, it shares memory address translation results between multiple loads and stores, simplifies Store and Merge Buffer lookup structures and shares data among loads accessing the same cache line. Page-Based Way Determination simultaneously provides way information for all cache lines mapping to a single page. Using an energy-oriented cache interface (one load or store per cycle) as baseline, MALEC achieves 14% speedup. On the contrary, a performance-oriented interface (up to two loads and one store per cycle) achieves 15% speedup, but consumed 48% more instead of 22% less energy than the baseline. To better utilize the performance capabilities offered by MALEC, we are currently developing a processor configuration capable of executing advanced SIMD operations and a set of highly vectorized benchmarks.

References

[1] "Intel 64 and IA-32 Architectures Optimization Reference Manual," Intel Corporation, Tech. Rep. June, 2011.

[2] "Software Optimization Guide for AMD Family 15h Processors," Advanced Micro Devices, Tech. Rep. 47414, 2011.

[3] T. Barr, A. Cox, and S. Rixner, "SpecTLB: a mechanism for speculative address translation," in *ISCA'11*, 2011.

[4] S. Borkar and A. A. Chien, "The future of microprocessors," *Communications of the ACM*, vol. 54, no. 5, 2005.

[5] Y. Chang and M. Lan, "Two new techniques integrated for energy-efficient TLB design," *Trans. on VLSI Systems*, vol. 15, no. 1, 2007.

[6] Y. Chang and S. Ruan, "Sentry tag: an efficient filter scheme for low power cache," in *Proc. ACSAC'2002*, 2002.

[7] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: simple techniques for reducing leakage power," in *ISCA'02*.

[8] M. Ghosh, E. Ozer, S. Ford, S. Biles, and H. Lee, "Way guard: a segmented counting bloom filter approach to reducing energy for set-associative caches," in *ISLPED'09*, 2009.

[9] K. Inoue, T. Ishihara, and K. Murakami, "Way-predicting set-associative cache for high performance and low energy consumption," in *ISLPED'99*, Aug. 1999.

[10] T. Johnson, D. Connors, M. Merten, and W.-M. Hwu, "Run-time cache bypassing," *Computer-IEEE*, vol. 48, no. 12, 1999.

[11] T. Juan, J. J. Navarro, and O. Temam, "Data caches for superscalar processors," *ICS '97*, 1997.

[12] G. Keramidas, P. Xekalakis, and S. Kaxiras, "Applying decay to reduce dynamic power in set-associative caches," *HiPEAC'07*, 2007.

[13] S. Li, J. Ahn, R. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO'42*, no. c, 2009.

[14] N. Muralimanohar and R. Balasubramonian, "CACTI 6.0: A tool to model large caches," *HP Laboratories*, 2009.

[15] D. Nicolaescu, A. Veidenbaum, and A. Nicolau, "Reducing Power Consumption for High-Associativity Data Caches in Embedded Processors," in *DATE'03*, Mar. 2003.

[16] M. Powell, A. Agarwal, T. Vijaykumar, B. Falsafi, and K. Roy, "Reducing set-associative cache energy via way-prediction and selective direct-mapping," *MICRO-34*, 2001.