Logic Filter Cache for Wide-VDD-Range Processors

Alen Bardizbanyan[†], Oskar Andersson[‡], Joachim Rodrigues[‡], and Per Larsson-Edefors[†]
[†]Chalmers University of Technology, Gothenburg, Sweden
[‡]Lund University, Lund, Sweden

***Abstract*—Wide-VDD-range processors offer high energy efficiency for varying embedded workloads. But reducing the VDD of the cache as aggressively as the VDD of the CPU logic is not straightforward, since standard 6T SRAMs cease to operate at lower VDDs. We implement a data and instruction filter cache, using logic cells located in the CPU VDD domain, to permit the level-1 (L1) cache to be reliably powered at a higher SRAM VDD. On top of eliminating many energy-wasting L1 cache accesses, the filter cache reduces the total number of executed cycles. Furthermore, the filter cache can be reconfigured as CPU VDD is reduced, to filter out an increasing proportion of cache accesses. We evaluate our approach using a 65-nm 1.2-V low-leakage CMOS process, with a minimal CPU and SRAM VDD of 0.4 and 0.95 V, respectively. Assuming 16kB+16kB L1 caches and 256B+256B filter caches, introducing the filter cache reduces the total cache access energy by 71% at 1.2 V and 87% at 0.4 V at an area overhead which is 13% of the L1 cache area.**

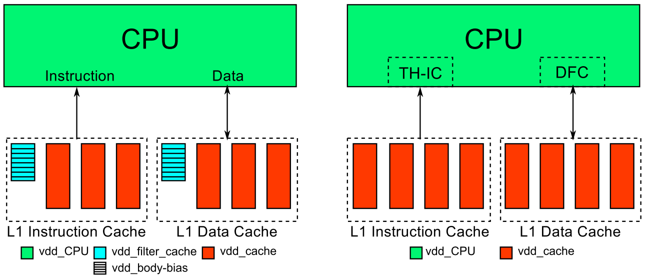
I. Introduction

Energy efficiency has become a key requirement for processors in general. The varying performance requirement presents designers with an opportunity to throttle performance and save power by way of supply voltage (VDD) regulation; when the desired performance level is lower, VDD is accordingly reduced, saving switching power. For high-performance processors, the dynamic voltage and frequency scaling (DVFS) scheme has long been an efficient way to regulating performance and power dissipation. However, due to robustness and performance reasons, DVFS schemes in commercial processors scale VDD quite conservatively (e.g., to 70% of nominal VDD [1]). For embedded processors that can have a very wide range of performance targets, there is a potential to scale down VDD much more than to 70% of nominal VDD. As long as the exponentially deteriorating switching speeds are acceptable, reducing VDD to the region near the transistor threshold voltage is favorable since this can, especially for performance-oriented process technologies, minimize the energy per operation [2].

Intel showcased a wide-VDD-range processor that can operate across a range of 280 mV to 1.2 V [3]. Thanks to the dual-VDD design, the logic VDD is not limited by the lowest voltage of the SRAMs but it can continue to scale down to 280 mV enjoying the power dissipation benefit of voltage scaling. Jacquet et al. demonstrated a very wide-VDD-range ARM Cortex-A9 processor that uses several circuit-level techniques of which forward body biasing is the most prominent one [4]. Here, the FD-SOI process technology employed gives an improvement over other technologies in terms of body-biasing capabilities.

While reducing VDD for the CPU logic to near-threshold levels is associated with significantly lower switching speed, SRAMs cease to operate in a reliable way altogether. The SRAM's robustness to variations at lower VDDs can be improved, e.g., by using write-assist circuitry but that comes with a significant energy overhead [5]. For processors tailored for low-VDD operation, more robust SRAMs may offer an acceptable performance-power tradeoff. But for wide-VDD-range processors, such SRAMs are undesirable since they degrade the processor's efficiency in the nominal-VDD mode.

Dreslinski et al. proposed an SRAM-based level-1 (L1) cache architecture that supports filter cache functionality [6]: As shown in Fig. 1a, one cache way is designed with body-biased near-threshold-tolerant SRAMs, while the other three ways use conventional, area-efficient SRAMs. When in near-threshold mode, the near-threshold way is used as a filter cache and to ensure timing constraints are met, body biasing is used.



(a) Prior work (b) Proposed approach

Fig. 1. Two approaches to cache access filtering for wide-VDD-range processors. In both approaches, vdd_CPU is aggressively scaled, while vdd_cache is a higher SRAM-compatible VDD. In (a), vdd_filter_cache is an intermediate voltage that varies with operating mode. In contrast, the proposed approach (b) with TH-IC and DFC filter caches uses only two voltage domains.

As an alternative to approaches that require customized cache SRAMs to be developed, forcing designers to work outside the standard design flows and adding to project cost, we propose an approach to implementing filter caches for wide-VDD-range processors using only standard logic gates. As shown in Fig. 1b, the entire filtering mechanism, denoted TH-IC (Sec. II-A) and DFC (Sec. II-B), is embedded inside the CPU VDD domain, which drastically reduces the number of accesses to the higher voltage of the SRAM VDD domain. Our filtering approach offers several advantages: 1) Since all L1 cache SRAMs remain at the higher VDD, the effective capacity of the L1 caches remains constant and does not reduce when VDD is scaled down as in previous approaches [6], [7], 2) no SRAM customization is required, 3) in contrast to conventional data filter caches which degrade performance [8], our implementation *decreases* the number of executed cycles, and 4) the power overhead of the filter cache scales down with CPU VDD. Since our filter cache is based on standard cells, body biasing to further tune the performance/energy operating point becomes optional and complementary.

II. Filter Cache Based on Logic Gates

The proposed filter cache is implemented inside a 5-stage pipeline to create a context in which it can be evaluated. Fig. 2 shows how the constituent parts, i.e., the TH-IC and DFC units, are integrated between the pipeline and the SRAM-based L1 data cache (L1 DC) and instruction cache (L1 IC).

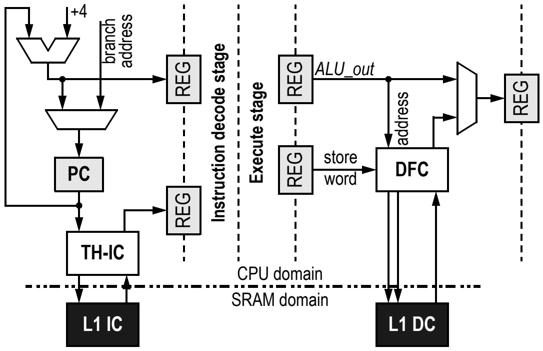


Fig. 2. Logic filter cache integrated in a 5-stage pipeline. TH-IC handles instruction access filtering to L1 IC, while DFC handles data access filtering to L1 DC.

A. Filtering of L1 IC Accesses

The tagless hit instruction cache (TH-IC) [9] was proposed as an improvement over a conventional instruction filter cache; since it guarantees hits, the TH-IC can completely eliminate the miss penalty. Furthermore, the TH-IC also eliminates the tag checks in the instruction filter cache structure for the guaranteed hits, which can potentially eliminate accesses to the instruction translation lookaside buffer.

To integrate the TH-IC with conventional L1 ICs, we implement direct-mapped TH-ICs that fetch a single instruction word per cycle during misses. Since the words are fetched on demand, energy efficiency is improved. This is in contrast to fetching an entire L1 IC line on a miss, where some words are fetched unnecessarily [6]. We add a Next Fetched (NF) bit to prohibit the TH-IC from fetching an entire line during a miss, by indicating if the next instruction is already fetched from the L1 IC. On a TH-IC miss, the requested instruction is fetched by conventionally accessing all L1 IC tag and data arrays, after which the way information is saved. Thus, the next L1 IC access does not require any tag check and only accesses a single data way.

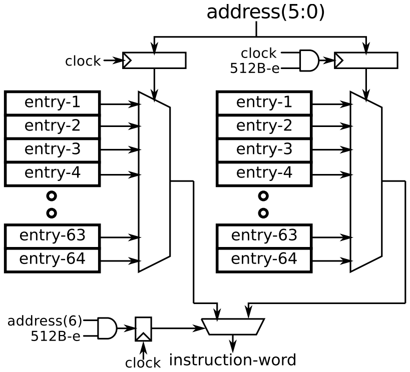


Fig. 3. TH-IC unit reconfigurable between 256B and 512B.

To identify the optimal cache line size, we evaluate TH-ICs mainly with respect to miss rates. Assuming a constant total storage capacity, TH-ICs that use 8-word cache lines turn out to be more efficient than configurations based on 16 words. Assuming 32b (4B) words, the following TH-ICs are considered here: 4Lx8W (128B), 8Lx8W (256B), and 16Lx8W (512B). Note, however, that the TC-IC sizes need not be constant: One important advantage of using logic-based filtering circuits is that it becomes easy to introduce reconfigurability. Fig. 3 shows a block schematic of a TH-IC that is reconfigurable between 256 and 512B.

B. Filtering of L1 DC Accesses

It is desirable that a filter cache does not degrade CPU performance, especially when the processor is in the nominal-VDD mode. We recently proposed a data filter cache (DFC) unit [10] that eliminates the performance degradation of conventional data filter caches [8]: By accessing our DFC speculatively in an early stage of the pipeline, the miss penalty is eliminated. This way, our DFC boosts performance in terms of cycle count, since data dependencies can be resolved earlier. The improvement in executed cycles for a 128B DFC, a 256B DFC, and a 512B DFC is 2.8%, 3.6%, and 4.3%, respectively, regardless of VDD.

The fully associative DFC configuration was shown to reduce the miss rate significantly compared to the direct-mapped one [10]. Hence, we here consistently use fully associative DFC implementations, with a write-allocate policy, for three sizes: 128B, 256B, and 512B. Additionally, the L1 DC SRAM bitwidth is a single word for area efficiency and, hence, the line fetch scheme uses a single word per cycle.

C. Circuit Implementation Aspects

Our implementation is based on a 65-nm 1.2-V low-leakage (LL) CMOS process technology. Although near-threshold operation is favorable from energy-per-operation point of view, Fig. 4 shows that this process exhibits no clear minimum in the near-threshold region, which is in contrast to performance-oriented process technologies.

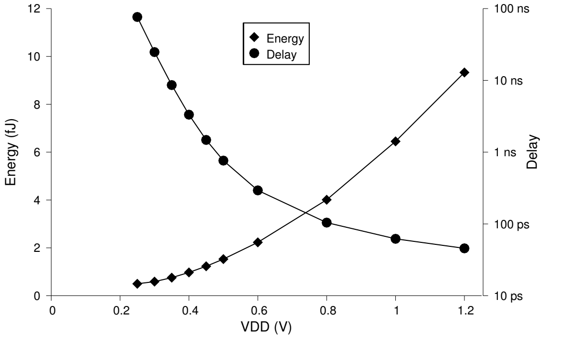


Fig. 4. Energy per operation and delay for a 65-nm LL inverter chain.

Our processor implementation (Fig. 2) comprises two voltage domains: One L1 cache SRAM VDD and one standard-cell VDD for the CPU logic and the filter cache. Since the data fetch schemes retrieve a single word per cycle, the number of level shifters between the two voltage domains can be kept low: 64 signals are used between TH-IC and L1 IC, whereas

96 signals are used between DFC and L1 DC. Since they are few, we do not consider the energy of the level shifters in the evaluation in Sec. III. Furthermore, we assume that the processor is integrated in a larger system-on-chip solution designed for a wide VDD range and, thus, that the supply voltages are provided by the system.

The downscaling of CPU VDD will be largely dependent on circuits with intrinsic contention, such as flip-flops and interfacing level shifters. While our approach requires neither customized SRAMs nor VDD-optimized cell libraries, it may benefit from circuit techniques that further reduce the minimal CPU VDD or the minimal SRAM VDD.

III. Evaluation Method

While it is likely there are opportunities to fine tune cells for a wide VDD range, our evaluation uses a foundry library that is optimized for nominal VDD in a standard implementation flow. We re-characterized an existing 65-nm low-leakage cell library at a number of desired operating voltages, providing cell models across the processor's VDD operating range. Simulations were done on SPICE-level netlists of all library cells, using extracted parasitics at different VDDs for all cell input vector combinations. During characterization, the conditions for input slew rates and fanout were varied for each cell and the characterization results were compiled into a single .lib (liberty format) file for each condition.

We developed a VHDL description for a complete 5-stage MIPS-like pipeline, including the data access circuits of DFC and TH-IC, and the SRAM-based L1 cache memories. As shown in Fig. 2, the TH-IC and the DFC were integrated tightly with the pipeline to enable timing-driven implementation for performance and energy estimations. The design was synthesized using Synopsys Design Compiler under slow-slow corner and 125°C conditions and the postsynthesis netlist was verified via logic simulations using EEMBC benchmarks [11]. After place and route in Cadence Encounter, RC-extracted netlists were backannotated to Synopsys PrimeTime PX for power estimation. We estimated the energy per operation for different units for a wide range of VDDs for the typical-typical corner and 25°C. For these conditions, the CPU VDD range of 0.4–1.2 V yields a top clock rate of 7.6–660 MHz. For the SRAM blocks, we used the foundry library corners.

We compiled 20 different MiBench benchmarks [12], across six different categories with the large dataset option, using mips-gcc. We ran the benchmarks using SimpleScalar [13], assuming a 2-level BPB, a 256-entry BTB, a branch penalty of 7 cycles, one ALU, one integer multiplier and a memory latency of 120 cycles. The L1 DC and L1 IC were each 16kB 4-way caches with a 32B line size and a 2-cycle hit. All memory cycle latencies are constant regardless of VDD.

In the final step, the energy per operation in different blocks obtained from PrimeTime PX was backannotated to SimpleScalar, where this was merged with information on events, such as TH-IC hits, to generate the final energy values.

IV. Results

Fig. 5 presents the total instruction access energy at different supply voltages, when normalized to the energy of a conventional L1 IC access at 1.2 V. Due to the regular instruction access pattern, the TH-IC reduces the instruction access energy significantly. The 128B TH-IC saves 80% of the access energy at 1.2 V. Even though 256B and 512B TH-ICs eliminate more accesses to L1 IC, the energy savings are less due to the increased energy of accessing the TH-IC unit.

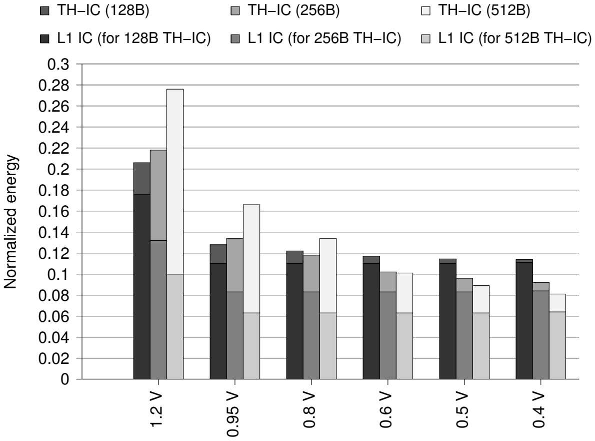


Fig. 5. Instruction access energy (L1 IC + TH-IC) at different supply voltages normalized to the energy of a conventional L1 IC access at 1.2 V. While CPU VDD scales down to 0.4 V, SRAM VDD only scales to 0.95 V.

As Fig. 5 shows, the energy trends are different for the three TH-IC configurations. This difference is mainly caused by the limitations of the conventional SRAM's supply voltage. While the energy of the standard cells keeps on reducing when the CPU VDD is reduced below 0.95 V, the minimum SRAM VDD is defined as 0.95 V by the foundry. Thus, as CPU VDD is lowered, the energy of the TH-IC which is based on logic gates is reducing significantly relative to the SRAM access energy. When the SRAM energy starts to dominate the overall instruction access energy, more complex TH-IC configurations that have more entries can save more instruction energy since they can filter out more accesses to the SRAMs.

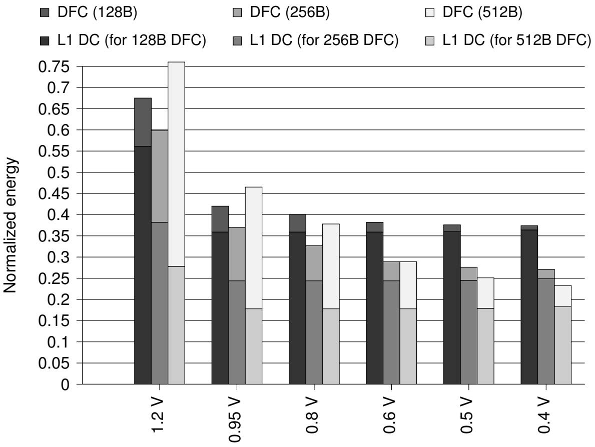


Fig. 6. Data access energy (L1 DC + DFC) at different supply voltages normalized to the energy of a conventional L1 DC access at 1.2 V. While CPU VDD scales down to 0.4 V, SRAM VDD only scales to 0.95 V.

Fig. 6 shows the total data access energy at different supply voltages, when normalized to the energy of a conventional L1 DC access at 1.2 V. The energy savings are lower compared to the instruction access energy savings. This is due to the higher miss rates in the DFC unit; data access patterns do not show as much locality as instruction accesses do.

Since accesses to the L1 DC are more infrequent compared to L1 IC accesses, the impact of the SRAM leakage energy may become an issue for data accesses at very low supply voltages. However, since our evaluation is using a low-leakage process technology, the overall impact of leakage is very low.

Fig. 7 presents the total cache access energy at different supply voltages. Note that the pipeline portions of the processor are likely to enjoy bigger improvements at very low CPU VDDs, but the purpose of this work is to investigate the possible gains in the cache access energy.

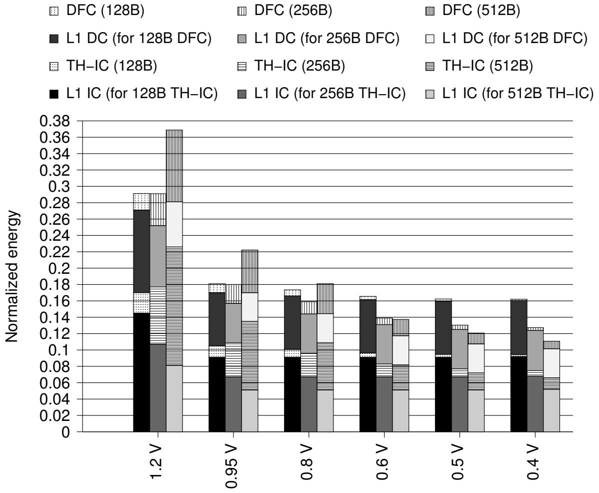


Fig. 7. Overall cache access energy at different supply voltages normalized to the total access energy of L1 IC and L1 DC at 1.2 V. While CPU VDD scales down to 0.4 V, SRAM VDD only scales to 0.95 V.

Three different configurations are given in Fig. 7; 128B, 256B, and 512B. As expected from the previous evaluations, the 512B configuration has a considerable energy overhead down to a CPU VDD of 0.6 V. At very low CPU VDDs, the overall energy gains for the 128B configuration are limited. Rather the energy starts to increase, due to leakage, for the 128B configuration when going from 0.5 to 0.4 V. A reconfigurable 256B/512B TH-IC, which is in 512B mode for a CPU VDD less than or equal to 0.6 V and in 256B mode otherwise, can provide an improvement of 11% in instruction access energy over a fixed-size 256B TH-IC.

While the configurations of size 256B are slightly less energy efficient than the ones of size 512B, area should also be considered: The 256B and 512B filter caches occupy around 13% and 27%, respectively, of the area of the 16kB L1 cache.

V. Conclusion

Voltage scaling has been key to reducing power and energy dissipation of processors. A wide-VDD-range processor reaps the energy benefit of voltage scaling by reducing performance for lighter workloads, while delivering high performance at nominal VDD. However, the challenge is to design a processor that is efficient in both extremes of the wide voltage range. The approach to wide-VDD-range processors that we take in this paper is that we use logic gates—different types of filter caches—to locate as much data and instruction accesses inside the CPU logic domain, whose supply voltage can be more aggressively scaled down than that of the SRAM caches. An important benefit of our approach is that it becomes practical to implement a wide-VDD-range processor, since a standard implementation flow can be used.

We have evaluated our approach for a 3X-range processor whose CPU logic can operate across 0.4–1.2 V. During implementation we use a standard flow down to place and route, with a cell library that we characterize for a number of VDDs below the nominal one from the foundry. Since the L1 caches are made up of standard, area-efficient 6T SRAM blocks, the foundry's minimal SRAM VDD is 0.95 V.

Thanks to the filtering that takes place in the CPU logic domain, a high energy efficiency can be maintained as the CPU VDD is reduced: We demonstrate in a 65-nm low-leakage CMOS process technology that when changing operation mode, from 1.2 to 0.4 V, the total access energy reduction changes from 71% to 89%. Thanks to the combined effect of voltage scaling and logic-based filtering of cache accesses, our approach makes the 3X-range processor data and instruction access energy scale down by the 9X that would be expected if the caches were driven by the CPU VDD.

Adding the filter cache incurs an area overhead: The 256B+256B filter cache is 13% of the 16kB+16kB L1 cache area. On the other hand and in contrast to conventional data filter caches which are known to degrade performance, our implementation boosts performance: The 256B filter cache decreases the number of executed cycles by 3.6%.

References

[1] R. G. Dreslinski *et al.*, "Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits," *Proc. IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010.

[2] B. Zhai *et al.*, "Theoretical and practical limits of dynamic voltage scaling," in *Design Automation Conf.*, Jul. 2004, pp. 868–873.

[3] S. Jain *et al.*, "A 280mV-to-1.2V wide-operating-range IA-32 processor in 32nm CMOS," in *Int. Solid-State Circuits Conf.*, Feb. 2012, pp. 66–68.

[4] D. Jacquet *et al.*, "A 3 GHz dual core processor ARM Cortex$^{\text{TM}}$-A9 in 28 nm UTBB FD-SOI CMOS with ultra-wide voltage range and energy efficiency optimization," *IEEE J. Solid-State Circuits*, vol. 49, no. 4, pp. 812–826, Apr. 2014.

[5] M. E. Sinangil *et al.*, "A 28 nm 2 Mbit 6 T SRAM with highly configurable low-voltage write-ability assist implementation and capacitor-based sense-amplifier input offset compensation," *IEEE J. Solid-State Circuits*, vol. 51, no. 2, pp. 557–567, Feb. 2016.

[6] R. G. Dreslinski *et al.*, "Reconfigurable energy efficient near threshold cache architectures," in *Int. Symp. on Microarchitecture*, Nov. 2008, pp. 459–470.

[7] D. Bortolotti *et al.*, "Hybrid memory architecture for voltage scaling in ultra-low power multi-core biomedical processors," in *Design, Automation and Test in Europe Conf.*, Mar. 2014.

[8] J. Kin *et al.*, "Filtering memory references to increase energy efficiency," *IEEE Trans. Comput.*, vol. 49, no. 1, pp. 1–15, Jan. 2000.

[9] S. Hines *et al.*, "Guaranteeing hits to improve the efficiency of a small instruction cache," in *Int. Symp. on Microarchitecture*, Dec. 2007, pp. 433–444.

[10] A. Bardizbanyan *et al.*, "Designing a practical data filter cache to improve both energy efficiency and performance," *ACM Trans. Archit. Code Optim.*, no. 4, pp. 54:1–54:25, 2013.

[11] Embedded Microprocessor Benchmark Consortium. [Online]. Available: http://www.eembc.org

[12] M. R. Guthaus *et al.*, "MiBench: A free, commercially representative embedded benchmark suite," in *Int. Workshop on Workload Characterization*, Dec. 2001, pp. 3–14.

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