Region-Based Caching: 
An Energy-Delay Efficient Memory Architecture for Embedded Processors

Hsien-Hsin S. Lee  
Gary S. Tyson

Advanced Computer Architecture Lab  
Electrical Engineering and Computer Science Department  
The University of Michigan, Ann Arbor  
{linear, tyson}@eecs.umich.edu

Abstract
Power consumption has been a major concern in designing microprocessors for portable systems such as notebook computers, hand-held computing and personal telecommunication devices. As these devices increase in popularity and are used in a wider range of applications, a low power design becomes more critical. In this paper, we propose a new microarchitectural data cache design called region-based caching that can reduce power consumption. Power savings is achieved by re-organizing the the first level cache to more efficiently exploit memory reference characteristics produced by programming language semantics. These characteristics enable the cache to be partitioned by memory region (stack, global, heap), reducing power consumption, while retaining comparable performance to a conventional cache design. Applications from the MediaBench benchmark suite indicate that a design with two additional small region-based caches results in 66% reduction in average in energy-delay product.

1. INTRODUCTION
As process technology continues to make progress following Moore’s Law, manufacturing cost per transistor is decreasing dramatically. This enables more sophisticated microarchitectural features to be integrated into future generation high performance processors to improve performance. Power density (measured in capacitance per unit die area) is therefore increasing, making it necessary to supply large amperage and making it more difficult to dissipate waste heat from the chip [27]. Reducing power requirements has not been the highest priority goal in developing microprocessors targeted at desktop or high-end server market. However, as notebook computers, hand-held computing, mobile and personal telecommunication devices are getting more popular, power is no longer a secondary goal in the process of microprocessor design. Furthermore, as embedded processors gain overall market share, processor designers are targeting more resources to meet high performance requirements while simultaneously reducing power consumption. Researchers from different disciplines including devices, circuits, logic, architectures and even operating systems and compilers, are investigating new low-power technologies.

Power dissipation in the memory subsystem constitutes a major portion of the overall power dissipation in these embedded processors [3][13][20][26]. Advances in instruction compression algorithms [25] or compressed instruction coding such as Thumb instruction set extensions [31] in the ARM architecture have reduced the average power consumption per instruction in the instruction cache, but these techniques cannot reduce the power requirements in the data cache(s).

Several techniques have been proposed to reduce power consumption in data caches [1][4][22][32][33]. Generally, these techniques achieve power reduction by partitioning the data cache into smaller, low-power components. This partitioning can reduce the power required to perform a data access, but the same partitioning often increases average access latency, leading to longer execution time.

Most high-performance processors already employ a split first-level cache structure to partition code and data into distinct caches. In this research, we proposed a further partitioning of the data cache into stack, global and heap regions. Region-based caching can effectively reduce power by re-directing the stack and global data accesses into smaller separate cache structures. Region-based caching can also achieve this power reduction without increasing average memory latency and execution time. This is due to the high temporal and spatial locality exhibited by stack and global data references; smaller cache structures can reduce power dissipation per access while retaining high cache hit rates for stack and global references since their working sets are small. In this paper, we examine several region-based cache designs and quantify their performance and power efficiency. With a 2KB stack cache and a 2KB global cache, our design results in a 66% reduction in energy-delay product for the data cache compared with a conventional cache design.

This paper is organized as follows: Section 2 characterizes reference behavior for each individual memory region. Section 3 describes the region-based caching mechanism. Section 4 describes our simulation infrastructure, power model and evaluation metric used in the data analysis. Section 5 describes the MediaBench applications used to evaluate our cache design. Results are presented in Section 6. Section 7 reviews related work. We conclude our work in Section 8.
2. MEMORY REFERENCE REGIONS

2.1 Memory Reference Distribution

Defined by programming language semantics, run-time memory accesses can be categorized by the region of memory they access and the index method used [1]. Figure 1 shows the virtual memory partitioning used by the MIPS Architecture [21]. A system-defined amount of space is allocated to the stack, which grows from high memory addresses down as automatic variables are created (e.g., stack activation records are allocated during function calls). The top of stack dynamically maintains the size of the stack, which forms a bound on address references to the stack. The bottom address range, allocated during compilation, includes read-only data (e.g., literal pool), the instruction code region and the global data region. Memory is dynamically allocated at run-time by the program from the heap, which grows upwards from the middle address range.

The majority of data memory references fall into the stack, global data and heap regions. To understand the memory reference behavior by regions, the distributions of run-time data memory accesses are profiled using the MediaBench applications [23] compiled with the GCC compiler in PISA format. The data shown in Figure 2 are normalized to the total number of memory instructions. Unlike the SPEC CPU2000 benchmark [24] that shows 56% of instructions access memory, the MediaBench benchmarks contain an average of 24% of instructions accessing memory. Stack references average 40% of all memory references, while global data references and heap references average about 30% each of the total memory references. The remaining data references (less than 1%) access read-only data memory (e.g., string literals), identified as rdata in the figure. Djpeg, mpeg2encode, epic and smpc show these averages with an extraordinarily large portion of heap accesses.

2.2 Locality of Data Cache Regions

To understand the access locality of a cache line brought into the L1 cache, we calculated the number of cache line hits prior to a line eviction. We refer to the total number of access hits prior to cache line eviction as the life span of a cache line. Figure 3 illustrates the average life span of cache line in each data region. In this experiment, all the data regions compete in a single L1 data cache. Simulations were performed for cache sizes from 256B to 64KB with fully associative cache (represented by FA with dashed lines) and direct-mapped (represented by DM with solid lines) cache. The y-axis plots the cache line life span on a log scale. For most cache configurations and applications, the stack cache lines show the greatest life span, the heap cache lines has the shortest lifespan, and the global cache lines fall between. For example, for a fully associative 4KB L1 cache, stack cache lines has an average life span of 166 — i.e., each line was re-accessed an average of 165 times prior to eviction. In contrast, heap cache lines show an average life span of only 9.5.

Figure 4 shows the miss ratios for a spectrum of cache sizes, again, from 256B to 64KB when a dedicated cache is allocated for each individual memory region. These data show that the stack data consistently demonstrate the best cache locality for a given cache size. Furthermore, the hit rate approaches 99% for a very small (2KB) stack-cache. The heap data show the worst locality with a hit rate increasing linearly as the cache size doubles, reaching 95% at a 64KB heap-cache. As expected, the hit rate of global ref-
REFERENCES falls between the stack and heap approaching a 99% hit rate at a relatively small (4KB) global-cache configuration. These experiments show that by partitioning the cache structure into three components — a small stack-cache, a small global-cache and a larger cache for heap and others — a majority of memory references access small cache structures (40% stack-cache, 30% global-cache) while retaining a high hit rate since the caches are small, they consume less power; since the hit rates are high, they provide good performance with low access latency.

3. REGION-BASED CACHING

Recent energy reduction techniques proposed in architectural level cache designs can be classified into two primary schemes, vertical partitioning and horizontal partitioning. The basic idea of these partitioning techniques is to reduce power dissipation by referencing a smaller storage structure. For the vertical partitioning, i.e. employing a multi-level cache hierarchy, an extra level of caching is added nearest to the processor (e.g., a line buffer [14][32] or a filter cache [22]). These extended structures capture short-term locality and consume much less power when the requested data are found in the small buffer or cache. However, according to the prior studies, the hit rate for these small structures is relatively low and each miss requires an L1 access after this miss is determined; this increases the effective latency of an L1 access since the L1 access request is delayed.

An alternative to vertical partitioning is to perform a horizontal cache partitioning. Horizontal partitioning involves slicing each cache line into smaller segments (e.g., cache sub-banking [14][33]). The processor accesses (and powers) only the line segment that is referenced (requiring additional early address decode circuitry), saving power by not driving data paths in the cache that are not referenced. This approach is orthogonal to vertical partitioning.

Region-based Caching, is another horizontal partitioning design method that can reduce power dissipation of data caches more effectively by exploiting the nature of memory allocation conventions. As discussed earlier, the basic idea of this approach is to partition data references based on semantically defined memory regions into distinct caches. Data exhibiting high degree of utilization and locality, e.g. stack data or global data as discussed in Section 2, can be filtered out from the regular cache. Figure 5 sketches one implementation of a region-based caching design in block diagram. In this example, two horizontally partitioned region-based caches are added to the regular L1 cache — one for stack data and one for global data. All heap and other memory references are sent to the L1 cache as normal. All cache miss fill requests and evictions are directed to the next-level caches or DRAM memory. The region cache is activated (drawing power) only when a memory reference is made to its respective memory region. Note that the stack and global region caches, based on the requirements of target applications in the embedded processor, can be built much smaller than the L1 cache.

The region-based cache design provides several benefits. First, line conflicts are eliminated between regions since different regions are routed to different structures. This makes it more feasible to implement each cache with lower associativity; particularly the stack-cache since the active region of the stack is generally a single contiguous section at the top of stack. A direct mapped stack-cache generally has no more conflict misses than a fully associative stack cache [24]. Lower associativity can reduce the cache design complexity and lead to a faster cache access logic. Second, building smaller separate caches provides more flexibility in increasing overall data storage than enlarging a single cache. For instance, to enlarge a 32KB direct-mapped cache, one needs to either double the cache size to 64KB or opt for a possibly higher latency multi-way cache, e.g. a 5-way 40KB cache. Finally, as mentioned earlier, a smaller cache dissipates less power when accessed. Since about 70% of the references hit in the stack and global data region, the overall data cache power consumption can be significantly reduced when the sizes of those caches are made small (but large enough to retain a high hit rate). We will quantify the performance impact in our analysis in Section 6.

4. SIMULATION MODEL

4.1 Machine Models and Simulators

The infrastructure of our experiments is based on Watch toolset [7] developed at Princeton University. Watch, an extension of the Simplescalar tool suite [8], generates both performance data and power estimation using execution-driven simulation. We use Watch to evaluate relative performance and power dissipation for different processor design configurations by integrating the region-based caching mechanism.
4.2 Power Models

Total power consumption of CMOS circuits primarily consists of the following three components: static leakage dissipation, dynamic short-circuit dissipation, and dynamic switching dissipation [9] as summarized in the equations below.

\[ P = P_{\text{leakage}} + P_{\text{sc}} + P_{\text{switching}} \]

\[ P = \sum_{i=1}^{n} I_{\text{leak},i} \cdot V_{dd} + I_{sc} \cdot V_{dd} + \alpha_{0 \rightarrow 1} C_L V_{dd}^2 f_{clk} \]

The leakage current is due to the reverse-biased leakage between the substrate and the diffusions of a CMOS gate. Short-circuit power dissipation occurs in the brief period when both n-transistor and p-transistor are simultaneously active, generating a current pulse from \( V_{dd} \) to \( V_{ss} \). The switching power is required to change capacitor state — charging or discharging capacitors when state changes from logical 0 to 1 or vice versa.

We assume that 0.35um process technology parameters are used in this study. Under this assumption the leakage current power can be ignored [27]. The \( P_{sc} \) component is typically small and there exists design and fabrication technologies [9] to eliminate the short-circuit current, \( I_{sc} \). The dominant component of the total power dissipation is \( P_{\text{switching}} \), i.e., transitions that charge or precharge the load capacitance [18][28]. This is also the major power dissipation component that has been focused on in the past for reducing power of CMOS circuits. In this component, \( \alpha_{0 \rightarrow 1} \) is defined as the average number of times in each clock cycle a node with capacitance \( C_L \) will make a transition. \( f_{clk} \) is the clock frequency. Power generally can be reduced by reducing the supply voltage, load capacitance or switching frequency.

The Watch tool estimates power at the architectural level by storing the event occurrences of each functional unit during simulations. We assume that a simple clock gating [13] technique is applied to each cache module; therefore, each cache is activated only when an access is requested — zero power dissipation otherwise. The device capacitances used in Watch are similar to those published in [34]. The power consumption models of each cache consider typical components of a cache array structure including tag arrays, address decoder, wordline drive, bitline drive, and sense amplifiers. More details are documented in [7].

4.3 Energy-Delay Product Metric

In [15], Gonzales and Horowitz argue that the widely used metric, energy, measured in Watt/MIPS or Watt/SPEC, is not an ideal metric for evaluating the efficiency of a machine design. By simply reducing supply voltage or load capacitance, energy can be reduced at the expense of increasing circuit delay. For such a design, a lower energy processor would also have lower performance. Instead of using the energy metric, they propose to use the energy-delay (ED) product in Watt/SPEC as the metric for an energy efficient design. The ED product considers both performance and energy simultaneously in a design. For an energy efficient design without compromising performance, a design should attempt to minimize the ED product. If a processor trades off performance for energy, then its ED product will be unlikely to decrease.

For the results presented in Section 6, we show the ED product of a given machine relative to that of baseline machines for the comparison metric (in addition to performance and power). The following equations describe how we compare the ED products of two machines. For a target machine \( A \), a better design will reduce its ED product ratio with respect to that of a base machine. In other words, the goal of an energy-delay efficient system design should minimize the ED Product Ratio, i.e. \( \pi_{ED} \).

\[ E = \frac{\text{Watt}}{\text{MIPS}} = W \cdot \text{Delay} \]

\[ \text{ED Product} = E \cdot D = W \cdot (\text{Delay})^2 \]

\[ \frac{\text{EDP}_A}{\text{EDP}_B} = \frac{W_A \cdot (\text{Delay}_A)^2}{W_B \cdot (\text{Delay}_B)^2} = \frac{W_A^{\text{EDP}_A}}{W_B^{\text{EDP}_B}} \cdot \frac{1}{(\text{Speedup}_A)^2} \]

5. BENCHMARK

We use the MediaBench benchmark suite [23] in this study. The programs from MediaBench represent the workloads for a variety of emerging multimedia and communication applications. These applications are commonly seen in personal telecommunication and PDA devices. Table 1 describes the algorithm for each application. The binaries were compiled using SimpleScalar GCC compiler that generates code in the portable ISA (PISA) format. The PISA encoding and addressing modes are almost identical to the MIPS ISA format. All the simulations were run to completion except for mpegdecode and gs that exit after 600 million instructions to reduce simulation time.

6. SIMULATION RESULTS AND ANALYSIS

We present our simulation results and analyze them in this section. First, we evaluate one region-based configuration, comparing that configuration with alternative conventional...
## 6.1 Comparisons with Baseline Design

In this set of experiments, we will compare the power dissipation and performance of region-based cache with a 4KB direct-mapped stack-cache, a 4KB direct-mapped global-cache and a direct-mapped 32KB conventional L1 cache. Each cache has a single cycle access latency. This design is compared to three machine designs in these experiments. The first cache uses a 32KB direct-mapped L1 cache with single cycle access latency. The second cache has a 4-way 32KB L1 cache. The third cache expands the cache size to 40KB by increasing the associativity to five ways. Both multi-way caches have a two-cycle latency. As mentioned earlier, we used the timing information gathered from CACTI 2.0 [29] to determine cache access time for each cache configuration. Both 4-way and 5-way 32KB caches had access timing exceeding the 7ns target necessary to achieve single cycle access on our target architecture (they were 11ns and 12ns respectively). The purpose of using a 40KB cache is to match up the cache capacity of our region-based caches. As mentioned in Section 4.1, we add a 512KB level-two cache for all configurations as the backing storage in order to perform a fair comparison.

Figure 6 shows the performance comparison of our region-based caching design with regular cache designs. For the MediaBench applications, the region-based caches design performs almost on par or slightly faster than the regular cache designs. It reduces performance between 4% to 7% in mesa and rasta when compared to the 4-way and 5-way cache designs. For the same L1 size, it is simply because stack and global data increase much locality moving from the 4KB cache to the 32KB cache. Performance increases relative to all baseline cache design for cjpeg, dpjpeg and epic. For the 32KB configurations this can be due to the increased overall cache size. There is a relative performance improvement of about 3% for these applications with respect to the 40KB cache as well. This speed-up primarily comes from reducing the access time to one cycle and secondarily from reducing set conflicts between different data regions.

Figure 7 demonstrates the power dissipation of data references in the caches. The average relative power dissipation of the region based cache is significantly reduced to 96%, 45%, and 23% for mesa, rasta, and epic respectively.
and 37% of the 32KB DM, 32KB 4-way and 40KB 5-way designs respectively. The major power reduction occurs for stack and global references that were re-routed to the smaller stack and global caches. Power savings are significantly lower for unepic, epic, mpeg2encode and djpeg. This is due to the unusually high occurrences of heap accesses shown earlier in Figure 2.

Combining the results in Figure 6 and Figure 7, the Energy-Delay Product Ratios of the region-based caching design versus the baseline machines are plotted in Figure 8. This plot is normalized to the ED products of the baseline designs; Lower ED product ratio occurs when the region-based cache is the better cache design — the lower the ED product, the better the design. The average ED product ratio of the region-based cache is 0.54 compared to a 32KB direct mapped baseline cache, 0.45 compared to a 32KB 4-way baseline cache, and 0.37 compared to the alternate 40KB 5-way cache design.

These experiments indicate that a region-based cache consisting of a 4KB stack cache, a 4KB global cache and a 32KB L1 cache will achieve the same execution performance as a 40KB, 5-way cache while achieving a much more energy efficient implementation.

6.2 Exploiting Design Space of Region-based Caching

In this section, a spectrum of region-based caching design choices is investigated. In all comparisons, we use the 40KB, 5-way cache presented in Section 6.1 as the baseline for comparison. We examine seven different region-based cache configurations in Figure 9, varying the sizes of the stack and global regions. Each cache configuration uses a 32KB, direct-mapped L1 cache (represented as dm), except for the leftmost bar which uses a 32KB, 4-way conventional L1 cache (represented as 4w in the symbol). We use the following naming conventions in the figure. The SmGn symbols show the size of region caches: a mKB Stack cache and an nKB Global cache; when G is absent, there is no global-region cache and global data are stored in the L1 cache. For example, the rightmost configuration $S32G32-dmL1$ consists of a 2KB stack-cache and a 2KB global-cache and a 32KB, direct mapped L1 cache.

Figure 9 shows the average performance speedup, power reduction and energy-delay product ratio for MediaBench benchmark. Table 2 describes the energy-delay product ratio for each application in the benchmark (used to calculate the average). The 2KB stack cache and 2KB global cache ($S2G2-dmL1$) demonstrates the best design in ED product ratio. It consumes only one third of the power in a 5-way 40KB counterpart while achieving 99% of the execution performance. All but three of the applications have ED ratios above 0.50 while 7 of the 19 applications have ED ratios of less than 0.20. This shows that the overall performance (power and execution) of a region-based cache is significantly better than the design alternatives studied. Region-base caching reduces the power dissipation by routing data references to small, special purpose cache structures. High hit rates are maintained because the routing algorithm exploits known characteristics of high-level language programs. These hit rates translate into high performance execution, while retaining the power dissipation advantage.

7. RELATED WORK

Low-power IC design techniques can be classified into several levels of design space from system level, architecture, logic, to transistor level. Frenkil in [13] presented an overview of research activities at each level. We will briefly overview the techniques proposed in architecture domain for low-power cache design.

Power dissipation is generally proportional to the size of the SRAM array structure. Researchers and embedded processor architects have been studying designs employing smaller structures for the majority of the cache accesses to reduce power dissipation. Line buffers (or block buffering) and sub-banking [14][32] have been proposed to reduce power. To exploit spatial locality and reduce power, line buffers hold most recently accessed cache lines for potential hits by subsequent accesses. The cache is not exercised when a cache access hits in the line buffers. Kin et al. in [22] described a similar technique by inserting a very small filter cache as the first-level (L0) cache to the CPU. The filter cache design approach sacrifices cache performance in exchange of power-saving as the filter cache has poorer data locality. In
not to reduce power dissipation.

In the paper, we propose a novel region-avoiding cache algorithm that significantly improves the performance of many applications running on cache-based processor architectures. The approach is based on a comprehensive study of cache behavior and the identification of patterns that lead to power dissipation. The results show that the proposed algorithm achieves a 20% improvement in overall system performance compared to conventional cache designs. The cost of the proposed algorithm is significantly lower than that of other approaches, making it a viable solution for power-constrained systems.


table 1: Energy-Delay Product for Various Region-based Cache Designs

<table>
<thead>
<tr>
<th>Region</th>
<th>Energy-Delay Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>1000.0</td>
</tr>
<tr>
<td>Small</td>
<td>950.0</td>
</tr>
<tr>
<td>Medium</td>
<td>900.0</td>
</tr>
<tr>
<td>Large</td>
<td>850.0</td>
</tr>
<tr>
<td>Extra</td>
<td>800.0</td>
</tr>
</tbody>
</table>

By applying the proposed region-avoiding cache algorithm, a 20% reduction in energy dissipation can be achieved without significant performance degradation.
9. ACKNOWLEDGMENT

This research is sponsored by the NSF grant CCR-9812415, NSF Career grant MP-9734023 and generous gifts from IBM and Intel corporations.

10. REFERENCES


