Dynamic Address Compression Schemes: A Performance, Energy, and Cost Study *

Jiangjiang Liu‡, Krishnan Sundaresan†, and Nihar R. Mahapatra†

‡*Dept. of Comp. Sci. & Eng.*
University at Buffalo, SUNY
Buffalo, NY 14260-2000, USA

†*Dept. of Elect. & Comp. Eng.*
Michigan State University
East Lansing, MI 48824-1226, USA

Email: ‡*jliu3@cse.buffalo.edu,* †{*sundare2,nrm*}*@egr.msu.edu*

Abstract

Dynamic address compression schemes that exploit address locality can help reduce both address bus energy and cost simultaneously with only a small performance penalty. In this work, we investigate two such schemes and determine their optimal parameters that result in the highest area/cost reductions and least performance penalty for various address buses (both on- and off-chip) in current systems. For addresses compressed with these schemes, we study energy reduction of buses in current and future nanometer technology nodes. Our study uses the cycle-accurate simulator for the Alpha 21264 processor called sim-alpha for performance estimation and accurate interconnect models considering inter-wire capacitances for bus energy estimation. Results show that using address compression will result in only small performance overheads (less than 1% for compressing a 38-bit bus to 14 bits) and reduce bus energy dissipation by as much as 13% when applied to on-chip buses in current technologies.

1 Introduction

Address compression, when applied to on-chip address buses in current microprocessors or systems-on-chip (SoCs) can reduce bus energy dissipation and costs by reducing the amount of bus hardware (wires and their associated driving and repeater gates). This can lead to reduction in wire density and indirectly facilitate better interconnect routing and floorplanning. Further, by using area no more than a bus of original width, a narrow bus can: (1) use greater spacing between bus lines, which will reduce inter-wire capacitance and hence delay, bus energy, and cross talk; and/or (2) use wider wires to reduce resistance and hence delay and potentially improve performance. Using an address compression scheme may itself entail some performance, area, and power consumption overheads due to extra logic. But these overheads will not be much compared to the savings potentially obtained by reducing the widths of long on-chip buses and off-chip buses. This is because the size, speed, and power consumption of logic (which will be used to do compression/decompression) scale better than those of interconnect (which will be used to communicate the information), and hence these overheads will continue to decrease over time.

1.1 Related work and our contributions

Address buses have been studied widely in previous work and schemes have been proposed to improve their performance, power consumption, and/or area/cost. Various bus encoding schemes have been proposed to reduce power consumption in address buses, many of which are surveyed in [3]. Compression, which is related to encoding, can also provide similar or greater energy benefits in addition to substantial cost reduction for almost all components in a processor-memory system as we found in our earlier work [8].

A specific scheme for address compression using a small *compression cache* (cache specially used for compression) at the sending end and a base register file at the receiving end of a bus was first proposed in [9, 6], and subsequently used for compressing instruction and data buses in [4]. But the above works did not consider the energy-efficiency benefits that can be potentially obtained with address compression; its application to on-chip address compression was also not studied. Only recently, the effectiveness of compression in reducing the switching activity of off-chip data buses was studied in [1]. However, results reported in the above work too do not reflect actual energy reductions for current technologies since: (i) only switching activities were considered and (ii) it does not provide an estimate of the effectiveness of address compression in reducing *self-energy* (bus energy dissipated due to transitions in the line self-capacitance) and *coupling energy* (bus energy dissipated due to transitions on the coupling capacitance between two adjacent lines). This is important because, in current and future technologies, coupling energy significantly dominates self-energy in on-chip buses.

Using a simulator that models a realistic processor, in this work, we present results on how address compression schemes perform when applied to on-chip or off-chip buses in modern superscalar processors. In particular, we explore the performance, energy, and cost benefits of address compression and the effect of technology scaling on energy-efficiency of various compressed address buses. We use two metrics in our study – *extra cycle penalty* and *energy ratio* – that help us quantify: (1) the actual perfor-

*This research was supported by a US National Science Foundation grant # 0102830.

mance penalty due to address compression including the effect of hardware latency and pipeline stalls on the system and (2) energy dissipation in buses including the effect of inter-wire capacitances for various metal routing layers in nanometer-scale technology nodes for on-chip buses. We consider buses carrying physical addresses—instruction and data addresses are carried on the same bus—between level-one (L1) and level-two (L2) caches, and report results for two cases: (i) an address bus connecting L1 and L2 caches that are both on-chip like in most modern processors; and (ii) an address bus that connects L1 cache to off-chip memory (L2 cache or DRAM) like in the case of many SoCs. Our work is perhaps the first to study address compression in detail, from the perspective of optimizing performance, energy, and cost, for these buses.

The organization of the rest of this paper is as follows. In Sec. 2, we discuss two dynamic address compression schemes and discuss ways to optimize system performance and area/cost when using these schemes practically. Next, in Sec. 3, we describe our simulation environment and methodology. Then, in Sec. 4, we describe our experiments and discuss results. Finally, we conclude in Sec. 5.

2 Dynamic Address Compression

Our study focuses on two schemes—dynamic base register caching (DBRC) and bus expander (BE)—that were originally proposed for processor-memory address compression [9, 4]. These are described briefly below. In DBRC, the original address is split into a higher order and a lower order component (Fig. 1(a)) and the former is stored in a compressor, which is a cache of base registers, on the processor side. Upon a cache hit, the index and entry number to the base-register cache (BRC) are transmitted on the bus with the uncompressed lower order part of the original address in a single cycle. A miss in the processor BRC is indicated by sending a reserved bit pattern on the bus in the first cycle followed by the missed address in subsequent cycles (Fig. 1(b)). The memory side consists of a register file that is loaded with this missed address. The BE scheme is similar to DBRC except for the following. As shown in Fig. 1(b), a miss in the sender cache is not explicitly indicated with a reserved bit pattern as is done in the DBRC scheme. Rather, the BE logic at the sending end begins to transmit the entire address immediately starting from the first cycle and a separate control signal line is used to indicate hit or miss in the sender cache.

Previous work has not clearly specified any format for transmission of compressed addresses since they did not consider the energy-efficiency of address compression. We explored different bit-field placements and found the format shown in Fig. 1(c) to be a good choice. We discuss next how address buses can be optimized for performance and cost by choosing cache sizes and bus widths appropriately. Previous work has not considered such optimizations.

When a cache-based scheme like DBRC or BE is used to compress addresses, the compressed bus width w can be determined using the relation: $w = u + \log_2(e)$ (plus 1 control bit for BE), where u is the portion of the address (lower order part) that is left uncompressed and e is the number of entries in the compression cache. Note that if W is the original address width (also the original bus width), then $t = W - (\log_2(e/a) + u)$ is the width of the tag that needs to be stored in a compression cache of associativity a. Decreasing the address bus width (w), while maintaining the width of the uncompressed portion (u) the same, will reduce the number of entries (e) in the compression cache. This may lead to higher miss penalties (the number of extra cycles needed to transmit an address if it misses in the compression cache) and therefore degrade system performance due to the following reasons: (1) fewer entries in the compression cache cause more misses and (2) wider tag (t) to find a match for reduces the chances of a hit. Thus, there exists a range of values for w and e for which system performance will be affected the least when address compression is used. We are interested in finding these ranges of values. Note that $e \times t$ (along with a) corresponds to the compression cache hardware cost and $(W - w)$ reflects bus hardware savings.

3 Simulation Methodology

We used *sim-alpha*, the validated Alpha 21264 simulator, as the platform for our experiments [5]. The benchmarks and simulator configuration we used are summarized in Table 1. In this simulator, we implemented DBRC and BE to compress addresses transmitted on the L1→L2 address bus. We assume that the compression hardware is placed after the L1-cache but before the buffer chains that drive the L1→L2 address bus. Thus, the L1 miss address file (MAF) stores uncompressed addresses that missed in L1-cache and is drained when the compression hardware is free. Also, in our setup, instruction and data addresses are compressed using the same hardware. Further, in the default case, we assume that the compression and decompression of addresses take negligible time and that buses are not pipelined. The former assumption is justified even for the largest size of our compression cache, which is on the order of a few kilobits, because current technologies have made it possible to design L1 caches—which are at least 10 times larger than the compression caches we use—with a single cycle latency for gigahertz processors.

3.1 Estimation of extra cycle penalty

The extra cycles that a benchmark program running on the modified target system (system with address compression) takes compared to its running time on the default target system (system with no compression) is reported as the performance overhead due to address compression. Since we use an execution-driven simulator, our calculation of performance overhead includes any latencies due to pipeline stalls also, and not just the extra latencies due to compressed address transmission. We report

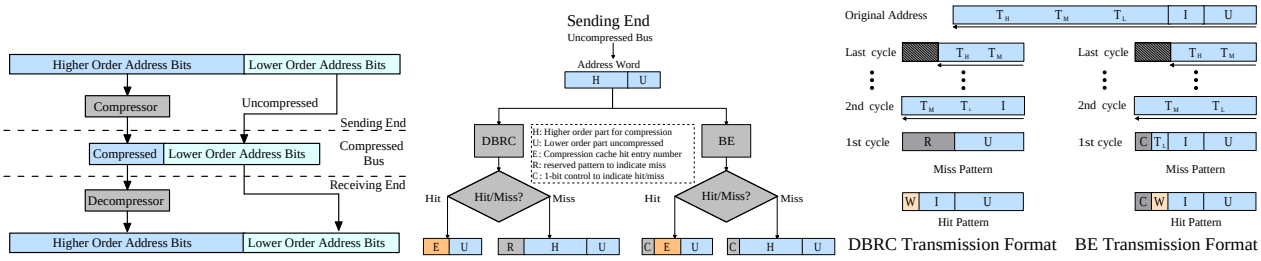


Figure 1: **Dynamic Address Compression Schemes:** (a) General schematic of a dynamic address compression scheme. (b) Schematic depicting how DBRC and BE form a compressed address word differently before sending it on the compressed bus. (c) Our default transmission formats for DBRC and BE.

Processor Core	
Clock rate	600MHz
Issue width	6 (4 integer and 2 floating point)
LSQ	32 entries each
Memory System	
P↔L1 bus	Non-pipelined; 64-bit data, 128-bit instruction, and 44-bit address lines
L1 D-cache	Virtually-indexed physically-tagged (VIPT), 64KB, 2-way set assoc., 64B block size, LRU policy, 3 cycle hit latency, write-back
L1 I-cache	Virtually-indexed virtually-tagged (VIVT), 64KB, 2-way set assoc., 64B block size, LRU policy, 1 cycle hit latency
L1 MAF	8 entries
L1↔L2 bus	Non-pipelined; 128-bit data/instruction lines and 38-bit address lines (21 bits for block index and 17 bits for tag)
L2 cache	Physically-indexed physically-tagged (PIPT), 2MB, direct-mapped, 64B block size, LRU policy, 12 CPU cycles hit latency, write-back policy, operating at 2x CPU clock cycle
L2↔M bus	Non-pipelined; 64-bit data/instruction lines and 38-bit address lines
DRAM	256MB, operating at 2x CPU clock cycle, 96 CPU cycles hit latency
Benchmarks	
CINT2000	gcc, gzip, parser, vpr, twolf, mcf, crafty
CFP2000	applu, swim, wupwise, lucas, art, ammp, equake
Sample & warmup	50 million committed instructions after skipping 2 billion committed instructions initially.

Table 1: **Target System and Benchmarks:** Default configurations for our target system, benchmarks, and sample sizes used in our simulations. LSQ= load/store queue, MAF= miss address file. This target system is broadly based on the Alpha 21264 processor.

the performance overhead as a percentage *average extra cycle penalty* (ECP) which is averaged over the 14 benchmarks that we used.

3.2 Bus energy model

The self-energy dissipated in a bus can be computed using the following expression: $E_{self} \propto N_{s,edge} \cdot C_{s,edge} + N_{s,middle} \cdot C_{s,middle}$, where $N_{s,edge}$ corresponds to the total number of self-transitions occurring in the two edge wires of a bus, $C_{s,edge}$ is the self-capacitance of an edge wire, $N_{s,middle}$ is the total number of self-transitions occurring in all the non-edge wires, and $C_{s,middle}$ is the self-capacitance of a non-edge wire. Note that $C_{s,edge} > C_{s,middle}$ in current technologies due to the extra *fringing effect* of the isolated side wall in each edge wire. The effect of fringing fields are non-negligible in current technologies because wire-height, and hence sidewall area, is more than wire-width for global and intermediate metal layers where most long buses are routed.

The total coupling energy dissipated in a bus can be computed using the following expression: $E_{coupling} \propto (N_{charge} + N_{discharge} + 4 \cdot N_{toggle}) \cdot C_c$, where N_{charge} is the total number of *charging coupling transitions* ($00 \rightarrow 01$, $00 \rightarrow 10$, $11 \rightarrow 01$, and $11 \rightarrow 10$), $N_{discharge}$ is the total number of *discharging coupling transitions* ($01 \rightarrow 00$, $01 \rightarrow 11$, $10 \rightarrow 00$, and $10 \rightarrow 11$), N_{toggle} is the total number of *toggle* transitions ($01 \rightarrow 10$ and $10 \rightarrow 01$), and C_c is the coupling capacitance between two adjacent lines of the bus. Note that $C_{s,edge}$, $C_{s,middle}$, and C_c are values that depend on technology and the layer of metal in which the bus is routed. We evaluated these quantities using TSMC 0.18μ global wire dimensions and applying formulas used in the Berkeley predictive technology model (BPTM) for interconnects [2].

For off-chip buses, only self-transitions need to be considered because inter-wire spacings are large. Fringing effects are also negligible since wire widths are substantially larger than wire heights. In our results, we report *average on-chip energy ratio*, $E_{on-chip}$, and *average off-chip energy ratio*, $E_{off-chip}$, instead of absolute energies. These are obtained by summing the compressed bus energies for 14 benchmarks and dividing by the sum of original bus energies for the same set of benchmarks.

4 Simulations and Results

4.1 Performance, energy, and cost tradeoffs

In this experiment, we examine three-way tradeoffs between performance, energy, and cost when using DBRC and BE schemes for L1→L2 addresses. Table 2 reports values for five quantities: ECP, cache size, on- and off-chip energy ratios, and miss rates for various bus widths. These bus widths were chosen so that all trends for variations in the above-mentioned quantities can be captured with minimum number of bus widths.

As explained earlier, an optimal number of bits that should be allotted to the index field resulting in the minimum extra cycle penalty for a given bus width can be found. These values, which we determined experimentally, are reported on the bottom lines of each row in Table 2. For example, the ECP

of a 16-bit bus for the optimal index width is reported as: [5, 0.18%], i.e., if an index width of 5 bits is used, the ECP will be only 0.18% more compared to address transmission on an uncompressed bus. The corresponding cache size is given on the bottom line of the next row and is 1575 bits. Similarly, the energy ratios are 1.16 (16% energy overhead) and 0.94 (6% energy reduction) for the off-chip and on-chip cases, respectively, when the optimal index width is used for this bus.

We also found that, by tolerating a slightly higher ECP, it may be possible to reduce the compression cache size (hardware cost) substantially. Results for this configuration are indicated on the top lines of each row. In this study, we limited the ECP to 0.3% higher values than those for the optimal index and experimentally determined the compression cache size, energy ratios, and miss rates. For the 16-bit example explained above, we found that the minimum index that can be used is only 3 bits. Thus the cache size can be reduced from 1575 bits (as in the optimal case) to only 375 bits—a 76.2% reduction. However, this may result in slightly worse energy ratios as observed from Table 2. For address compression using the BE scheme (Table 3), the corresponding cache size reduction for a 16-bit bus is from 3328 bits when using the optimal index size (6 bits) to 208 bits when using the minimum index of 2 bits—a 93.75% reduction. Other values of ECP can also be used to derive corresponding minimal index widths and similar trends as reported below will be observed.

Comparing across the two schemes for same bus widths, the following observations can be made. First, both schemes result in negligible performance penalty when address bus widths are reduced up to 20 bits from the original 38 bits. This results in immediate savings of 18 out of 38 (47.4%) bus lines and their associated buffers, repeaters, and receiving circuitry. For these savings, the compression cache size needed is also small (maximum of 63 bits). Thus, there will be net savings in area/cost even if the size of address compression hardware is taken into account. Reduction of bus width (beyond 20 bits) increases the ECP for both DBRC and BE. Also, the energy ratios show a broadly decreasing trend as we move towards narrower bus widths. For these buses, BE results in greater energy reduction than DBRC for most of the bus widths we considered. Finally, compression cache miss rates for both schemes are roughly similar for all bus widths—they vary between almost 0% (very few misses) for larger bus widths to about 39% for narrow bus widths.

4.2 Influence of compression cache size

In the previous experiment, we set limits on how much ECP can increase to determine the minimum index widths or compression cache sizes that can be used for various bus widths. In this experiment, we assume that the designer is allowed to use a compression cache in a given range of sizes. For four different compressed bus widths, 12, 14, 16, and 24 bits that represent different area/cost reductions of the address bus, we estimate the extra cycle penalty and energy ratios that will result. The compression cache sizes we consider are in the range 4–2048 entries.

4.2.1 Performance penalty

In Fig. 2, for narrower buses (e.g., a 12 bit bus), we observe that the ECP first decreases as the number of entries is increased from 4 to 64 and then increases dramatically as we increase the number of entries to 2048. The reason for this is the following. A larger number of entries means more bits of the compressed bus need to be used for transmitting the index during a hit. Thus, a lesser number of bits of the bus can be used for the uncompressed low-order portion of the address word. Reducing the uncompressed portion, in turn, means increasing the compressed portion, which lowers the compression cache hit rate to some extent, and so is the reason for the degradation in performance. For slightly wider buses (e.g., 14 and 16-bit buses), the number of entries in compression cache becomes less critical to performance than for narrower buses as our results show. This is because, for the same compressed portion of an address word, the increased bus width allows more entries in the compression cache, which can reduce the miss rate. Also, even in the case of miss, the wider bus facilitates transfer of the missed address in fewer cycles and hence the miss penalty becomes smaller. For even wider buses (24-bits or more), the uncompressed part is relatively large across different compression cache sizes, so varying compression cache size does not have any discernible impact on performance.

Another observation from Fig. 2 is that DBRC performs better than BE when smaller number of entries (notably four and eight entries) are used and this trend is true for all bus widths. The reason for this is the following. As mentioned in Sec. 2, the miss penalty for BE will be less than or equal to that for DBRC for the same bus width since the former uses a single bit and the latter uses a longer reserved bit-pattern to indicate a miss. However, for a given bus width and also fixed number of entries in the compression cache, the compressed portion for BE is one bit wider than that for DBRC due to the control bit, so the ECP in the case of BE can become higher due to increased miss rate leading to worse performance than DBRC.

4.2.2 Bus energy dissipation

Off-chip bus energy ratios are reported in Fig. 3(a). From this plot, we observe that BE consumes less energy on the average than DBRC for most bus widths—average results are shown in the table on the top-left corner of each plot. BE is more energy-efficient because it has lesser miss penalty than DBRC for the same bus width. But, for a bus width of 24 bits or greater, not much off-chip energy savings can be obtained with either address compres-

	Bus Width								
	8	10	12	14	16	20	24	28	32
Ext. Cyc. Penalty	[1, 5.42%]	[3, 2.95%]	[3, 1.55%] [6, 1.49%]	[2, 0.92%] [3, 0.64%]	[3, 0.32%] [5, 0.18%]	[1, 0.06%] [8, 0.01%]	[1, 0.00%] [5, 0.00%]	[1, 0.00%] [9, 0.00%]	[1, 0.00%] [3, 0.00%]
Size in Bits	[1, 99]	[3, 465]	[3, 435] [6, 3683]	[2, 189] [3, 405]	[3, 375] [5, 1575]	[1, 63] [8, 10731]	[1, 51] [5, 1071]	[1, 39] [9, 13299]	[1, 27] [3, 135]
Off-Chip Energy	[1, 1.32]	[3, 1.20]	[3, 1.23] [6, 1.27]	[2, 1.29] [3, 1.23]	[3, 1.21] [5, 1.16]	[1, 1.08] [8, 0.99]	[1, 1.00] [5, 1.01]	[1, 1.00] [9, 1.00]	[1, 1.00] [3, 1.00]
On-Chip Energy	[1, 0.91]	[3, 0.81]	[3, 0.85] [6, 0.85]	[2, 0.93] [3, 0.90]	[3, 0.97] [5, 0.94]	[1, 1.01] [8, 0.99]	[1, 1.00] [5, 1.01]	[1, 1.00] [9, 1.00]	[1, 1.00] [3, 1.00]
Miss Rate	[1, 0.42]	[3, 0.25]	[3, 0.20%] [6, 0.20]	[2, 0.19%] [3, 0.15]	[3, 0.10%] [5, 0.08]	[1, 0.042%] [8, 0.00]	[1, 0.00] [5, 0.00]	[1, 0.00] [9, 0.00]	[1, 0.00] [3, 0.00]
Comp. Ratio	[1, 0.56]	[3, 0.48]	[3, 0.48] [6, 0.49]	[2, 0.50] [3, 0.48]	[3, 0.49] [5, 0.48]	[1, 0.55] [8, 0.53]	[1, 0.63] [5, 0.63]	[1, 0.74] [9, 0.74]	[1, 0.84] [3, 0.84]

Table 2: **Extra Cycle Penalty, Optimal Index Widths, Compression Cache Sizes, Bus Energy Ratios, Miss Rates, and Compression Ratios for Address Compression Using DBRC Scheme.** For a given bus width (column) and metric (rows), the notation [A1, A2] means that A1 is the index width (minimum or optimal) and A2 is the value for the metric for that index width. For columns corresponding to bus widths 8 and 10, the minimum and optimal values are the same. Hence only one is reported.

	Bus Width								
	8	10	12	14	16	20	24	28	32
Ext. Cyc. Penalty	[1, 4.98%] [2, 4.91%]	[2, 2.34%] [3, 2.34%]	[3, 1.62%] [4, 1.61%]	[2, 0.83%] [5, 0.63%]	[2, 0.46%] [6, 0.27%]	[1, 0.07%] [6, 0.01%]	[1, 0.00%] [10, 0.00%]	[1,0.00%] [8, 0.00%]	[1, 0.00%] [4, 0.00%]
Size in Bits	[1, 136] [2, 272]	[2, 256] [3, 512]	[3, 480] [4, 960]	[2, 224] [5, 1792]	[2, 208] [6, 3328]	[1, 88] [6, 2816]	[1, 72] [10, 36864]	[1, 56] [8, 7168]	[1, 40] [4, 320]
Off-Chip Energy	[1, 1.19] [2, 1.18]	[2, 1.07] [3, 1.08]	[3, 1.06] [4, 1.06]	[2, 1.11] [5, 1.06]	[2, 1.11] [6, 1.07]	[1, 1.10] [6, 1.00]	[1, 1.00] [10, 1.00]	[1, 1.00] [8, 1.00]	[1, 1.00] [4, 1.00]
On-Chip Energy	[1, 0.88] [2, 0.87]	[2, 0.82] [3, 0.82]	[3, 0.84] [4, 0.84]	[2, 0.88] [5, 0.87]	[2, 0.94] [6, 0.92]	[1, 1.01] [6, 1.00]	[1, 1.00] [10, 1.00]	[1, 1.00] [8, 1.00]	[1, 1.00] [1, 1.00]
Miss Rate	[1, 0.40] [2, 0.39]	[2, 0.28] [3, 0.28]	[3, 0.21] [4, 0.21]	[2, 0.18] [5, 0.15]	[2, 0.12] [6, 0.10]	[1, 0.06] [6, 0.00]	[1, 0.00] [10, 0.00]	[1, 0.00] [8, 0.00]	[1, 0.00] [4, 0.00]
Comp. Ratio	[1, 0.54] [2, 0.53]	[2, 0.48] [3, 0.48]	[3, 0.47] [4, 0.47]	[2, 0.48] [5, 0.47]	[2, 0.50] [6, 0.48]	[1, 0.56] [6, 0.53]	[1, 1.63] [10, 0.63]	[1, 0.74] [8, 0.74]	[1, 0.84] [4, 0.84]

Table 3: **Extra Cycle Penalty, Optimal Index Widths, Compression Cache Sizes, Bus Energy Ratios, Miss Rates, and Compression Ratios for Address Compression Using BE Scheme.** For a given bus width (column) and metric (rows), the notation [A1, A2] means that A1 is the index width (minimum or optimal) and A2 is the value for the metric for that index width.

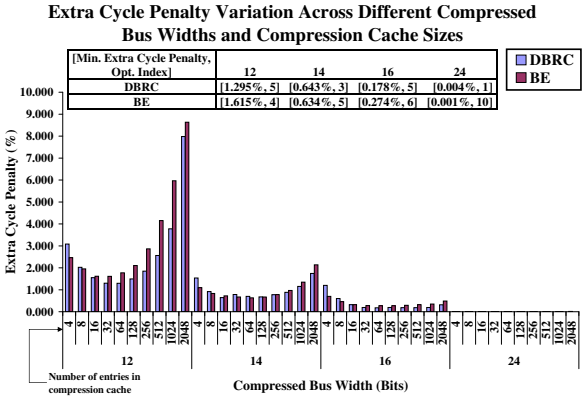


Figure 2: **Extra Cycle Penalty for DBRC and BE for Different Compression Cache Sizes.**

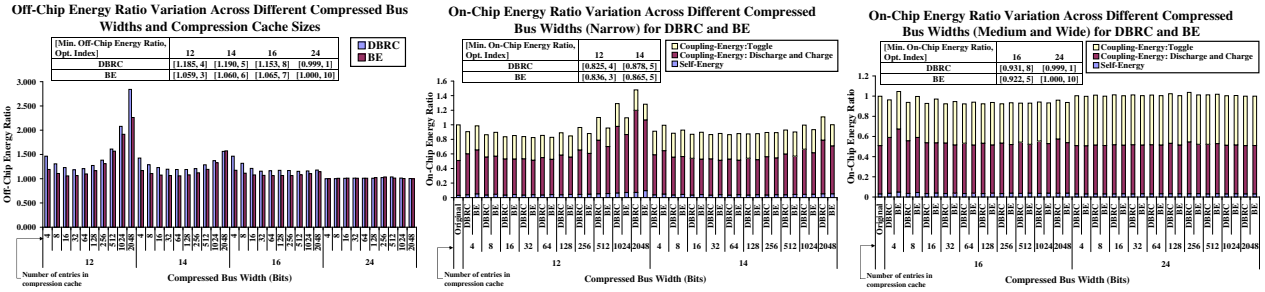
sion design. This is because, for wider buses, the uncompressed part has more bits and the compressed part is small, which leads to low miss rate (smaller than 0.002%) for both schemes. So most bus lines are used for the uncompressed portion and hence the bit pattern of the compressed address word is similar to the bit pattern of uncompressed original address word.

Figs. 3(b) and (c) show on-chip energy ratios, including the contribution of self-energy and components of coupling energy across different number of entries in compression cache, for four bus widths. These are shown across two plots: bus widths 12 and 14 in Fig. 3(b) and 16 and 24 in Fig. 3(c). From these figures, it can be observed that energy savings obtained with address compression in on-chip buses is more than savings for off-chip buses. It can also be seen that, across bus widths, most of the energy saving is due to reduction in toggle energy, and across different compression cache sizes for the same bus width, the savings—which are better when smaller cache sizes are used—are due to reductions in coupling charge and discharge energies. Also, similar to what was observed in Sec. 4.2.1 for performance, BE resulted in a worse energy ratio compared to DBRC when smaller compression caches (number of entries of 4 and 8) are used. The reason for this is also the same as that described in Sec. 4.2.1. Finally, we also observe that, similar to trends in off-chip energy dissipation, the wider the compressed bus width is, the smaller the energy that can be saved with dynamic address compression.

4.3 Influence of other factors

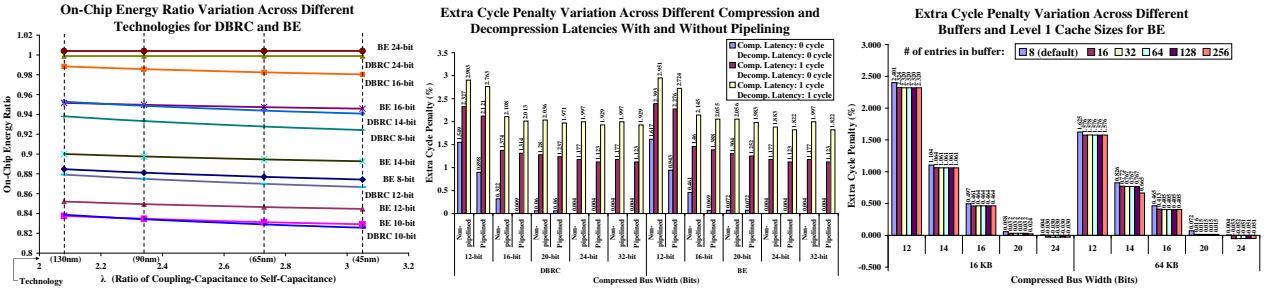
The energy ratios of compressed address buses as technology scales down is shown in Fig. 4(a). Here, the parameter λ (ratio of coupling capacitance to the self-capacitance of a wire), takes values of approximately the following: 2.08 for 130-nm, 2.34 for 90-nm, 2.73 for 65-nm, and 3.05 for 45-nm, for topmost layer interconnects in current and future nanometer technologies [7]. From this plot, we observe that address compression improves energy efficiency of most compressed address buses at the same rate even as technology scales down.

The influence of extra compression/decompression latencies on ECP is shown in Fig. 4(b). We observe that when address compression and decompression both have a one cycle latency, the ECP does not drop off as rapidly with



(a) (b) (c)

Figure 3: **Influence of Compression Cache Size on Extra Cycle Penalty and Bus Energy Dissipation.** (a) Off-chip bus energy dissipation ratio for DBRC and BE for different compressed bus widths. (b)–(c) On-chip bus energy dissipation ratios for DBRC and BE for different compressed bus widths.



(a) (b) (c)

Figure 4: **Other Factors:** (a) Influence of technology scaling on energy-efficiency. (b) Influence of compression/decompression latency on performance, with and without address bus pipelining. (c) Influence of varying L1 cache and buffer sizes on performance.

bus width as in the default case (both compression and decompression take zero cycles). This is because the hit-rate of the compression cache with wider buses becomes better, which necessitates one extra cycle for decompression each time a hit occurs. Note that a miss is assumed to cause no extra decompression latency because there is no need for a register file access to decompress the address. We also observe that pipelining the address bus helps reduce the ECP to some extent, but the effect is not much probably because L1→L2 address references are more or less sparsely distributed over time.

We also experimented with six different miss address file (MAF) sizes, 8, 16, 32, 64, 128, and 256, and three L1 cache sizes to ascertain their impact on ECP in a system with address compression. From Fig. 4(c) we observe that, for all bus widths and L1 cache sizes, increasing the MAF size from 8 to 16 entries reduces the extra cycle penalty by a small amount, but further increase in the MAF does not result in much benefit for most buses.

5 Conclusions

In this paper, we showed that dynamic cache-based address compression schemes applied to L1→L2 address bus result in substantial energy and cost benefits for on- and off-chip buses. With simulations using a cycle-accurate simulator for 14 SPEC CPU2000 benchmarks, we reported the optimal compression cache sizes that result in minimum extra cycle penalty and the corresponding energy savings, compression cache miss rates, and address compression ratios for a wide range of compressed bus widths. We showed that aggressive bus-width reduction (as much as 63%, for example) will result in only an extra cycle penalty of about 1% or less and that energy dissipation in address buses will reduce appreciably (up to 13%) with compression for current technologies. These savings were found to increase for future nanometer technology nodes.

References

[1] K. Basu, A. Choudhary, J. Pisharath, and M. Kandemir. Power Protocol: Reducing Power Dissipation on Off-Chip Data Buses. In *Proceedings of the Annual ACM/IEEE International Symposium on Microarchitecture*, pages 345–355, 2002.

[2] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu. New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 201–204, June 2000.

[3] W.-C. Cheng and M. Pedram. Memory Bus Encoding for Low-power: A Tutorial. In *Proceedings of International Symposium on Quality of Electronics Design*, pages 199–204, March 2001.

[4] D. Citron and L. Rudolph. Creating a Wider Bus using Caching Techniques. In *Proceedings of International Symposium on High Performance Computer Architecture*, pages 90–99, January 1995.

[5] R. Desikan, D.C Burger, S.W. Keckler, and T.M. Austin. Sim-alpha: A Validated, Execution-Driven Alpha 21264 Simulator. Technical Report TR-01-23, The University of Texas at Austin, Department of Computer Sciences, 2001.

[6] M. Farrens and A. Park. Dynamic Base Register Caching: A Technique for Reducing Address Bus Width. In *Proceedings of the Annual International Symposium on Computer Architecture*, pages 128–137, June 1991.

[7] ITRS. International Technology Roadmap for Semiconductors, 2001 edition, 2001.

[8] N.R. Mahapatra, J. Liu, K. Sundaresan, S. Dangeti, and B.V. Venkatrao. The Potential of Compression to Improve Memory System Performance, Power Consumption, and Cost. In *Proceedings of IEEE Performance, Computing and Communications Conference, Phoenix, AZ, USA*, April 2003.

[9] A. Park and M. Farrens. Address Compression through Base Register Caching. In *Proceedings of the Annual ACM/IEEE International Symposium on Microarchitecture*, pages 193–199, November 1990.