Optimizing NUCA Organizations and Wiring Alternatives for Large Caches With CACTI 6.0 *

Naveen Muralimanohar†, Rajeev Balasubramonian†, Norm Jouppi‡
† School of Computing, University of Utah
‡ Hewlett-Packard Laboratories

Abstract

A significant part of future microprocessor real estate will be dedicated to L2 or L3 caches. These on-chip caches will heavily impact processor performance, power dissipation, and thermal management strategies. There are a number of interconnect design considerations that influence power/performance/area characteristics of large caches, such as wire models (width/spacing/repeaters), signaling strategy (RC/differential/transmission), router design, etc. Yet, to date, there exists no analytical tool that takes all of these parameters into account to carry out a design space exploration for large caches and estimate an optimal organization. In this work, we implement two major extensions to the CACTI cache modeling tool that focus on interconnect design for a large cache. First, we add the ability to model different types of wires, such as RC-based wires with different power/delay characteristics and differential low-swing buses. Second, we add the ability to model Non-uniform Cache Access (NUCA). We not only adopt state-of-the-art design space exploration strategies for NUCA, we also enhance this exploration by considering on-chip network contention and a wider spectrum of wiring and routing choices. We present a validation analysis of the new tool (to be released as CACTI 6.0) and present a case study to showcase how the tool can improve architecture research methodologies.

Keywords: *cache models, non-uniform cache architectures (NUCA), memory hierarchies, on-chip interconnects.*

1. Introduction

Multi-core processors will incorporate large and complex cache hierarchies. The Intel Montecito employs two 12 MB private L3 caches, one for each core [25]. Intel is already prototyping an 80-core processor [30, 38] and there is speculation that entire dies in a 3D package may be employed for large SRAM caches or DRAM main memory [7, 23, 30]. Therefore, it is expected that future processors will have to intelligently manage many megabytes of on-chip cache. Future research will likely explore architectural mechanisms to (i) organize the L2 or L3 cache into shared/private domains, (ii) move data to improve locality and sharing, (iii) optimize the network parameters (topology, routing) for efficient communication between cores and cache banks. Examples of ongoing research in these veins include [5, 6, 10, 11, 12, 15, 19, 20, 21, 22, 33, 44].

Many cache evaluations employ the CACTI cache access modeling tool [43] to estimate delay, power, and area for a given cache size[1]. The CACTI estimates are invaluable in setting up baseline simulator parameters, computing temperatures of blocks neighboring the cache, evaluating the merits/overheads of novel cache organizations (banking, reconfiguration, additional bits of storage), etc. While CACTI 5.0 produces reliable delay/power/area estimates for moderately sized caches, it does not model the requirements of large caches in sufficient detail. Besides, the search space of the tool is limited and hence so is its application in power/performance trade-off studies. With much of future cache research focused on multi-megabyte cache hierarchy design, this is a serious short-coming. Hence, in this work, we extend the CACTI tool in many ways, with the primary goal of improving the fidelity of its large cache estimates. The tool can also aid in trade-off analysis: for example, with a comprehensive design space exploration, CACTI 6.0 can identify cache configurations that consume three times less power for about a 25% delay penalty.

The main enhancement provided in CACTI 6.0 is a very detailed modeling of the interconnect between cache banks. A large cache is typically partitioned into many smaller banks and an inter-bank network is responsible for communicating addresses and data between banks and the cache controller. Earlier versions of CACTI have employed a simple H-tree network with global wires and have assumed uniform access times for every bank (a uniform cache access architecture, referred to as UCA). Recently, non-uniform cache architectures (NUCA [21]) have also been proposed that employ a packet-switched network between banks and yield access times that are a function of where data blocks are found (not a function of the latency to the most distant bank). We add support for such an architecture within CACTI.

Whether we employ a packet-switched or H-tree network, the delay and power of the network components dominate the overall cache access delay and power as the

*We thank the anonymous reviewers for their helpful suggestions. This work was supported in parts by NSF grant CCF-0430063 and NSF CAREER award CCF-0545959.

[1] The first four versions of CACTI [31, 32, 35, 43] have been cited by more than 1000 papers and are also incorporated into other architectural simulators such as Wattch [8].

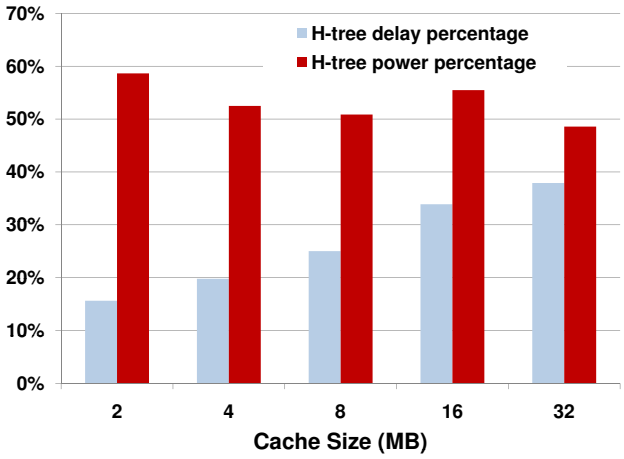


Figure 1. Contribution of H-tree network to overall cache delay and power.

size of the cache scales up. Figure 1 shows that the H-tree of the CACTI 5.0 model contributes an increasing percentage to the overall cache delay as the cache size is increased from 2 to 32 MB. Its contribution to total cache power is also sizeable- around 50% [2]. The inter-bank network itself is sensitive to many parameters, especially the wire signaling strategy, wire parameters, topology, router configuration, etc. The new version of the tool carries out a design space exploration over these parameters to estimate a cache organization that optimizes a combination of power/delay/area metrics for UCA and NUCA architectures. Network contention plays a non-trivial role in determining the performance of an on-chip network design. We also augment the design space exploration with empirical data on network contention.

Components of the tool are partially validated against detailed Spice simulations. We also present an example case study to demonstrate how the tool can facilitate architectural evaluations. The paper is organized as follows. Section 2 describes related work. Section 3 describes the baseline CACTI 5.0 model. Section 4 provides details on the new interconnect models and other enhancements integrated into CACTI 6.0. A case study using CACTI 6.0 is discussed in Section 5. We draw conclusions in Section 6.

2. Related Work

The CACTI tool was first released by Wilton and Jouppi in 1993 [43] and it has undergone four major revisions since then [31, 32, 35]. More details on the latest CACTI 5.0 version are provided in Section 3. The primary enhancements of CACTI 2.0 [31] were power models and multi-ported caches; CACTI 3.0 [32] added area models, independently addressed banks, and better sense-amp circuits; CACTI 4.0 [35] improved upon various SRAM circuit structures, moved from aluminium wiring to copper, and included leakage power models; CACTI 5.0 adds support for DRAM modeling. The CACTI 6.0 extensions described in this paper represent a major shift in focus and add support for new interconnect components that dominate cache delay and power. Unlike the prior revisions of CACTI that focused on bank and SRAM cell modeling, the current revision focuses on interconnect design. The results in later sections demonstrate that the estimates of CACTI 6.0 are a significant improvement over the estimates of CACTI 5.0.

A few other extensions of CACTI can also be found in the literature, including multiple different versions of *e-CACTI* (enhanced CACTI). eCACTI from the University of California-Irvine models leakage parameters and gate capacitances within a bank in more detail [24] (some of this is now part of CACTI 4.0 [35]). A prior version of eCACTI [1] has been incorporated into CACTI 3.0 [32]. 3DCACTI is a tool that implements a cache across multiple stacked dies and considers the effects of various inter-die connections [37].

A number of tools [3, 9, 14, 16, 34, 39, 41] exist in the literature to model network-on-chip (NoC). The Orion toolkit from Princeton does a thorough analytical quantification of dynamic and leakage power within router elements [41] and some of this is included in CACTI 6.0. However, Orion does not consider interconnect options, nor carry out a design space exploration (since it is oblivious of the properties of the components that the network is connecting). NOCIC [39] is another model that is based on Spice simulations of various signaling strategies. Given a tile size, it identifies the delay and area needs of each signaling strategy.

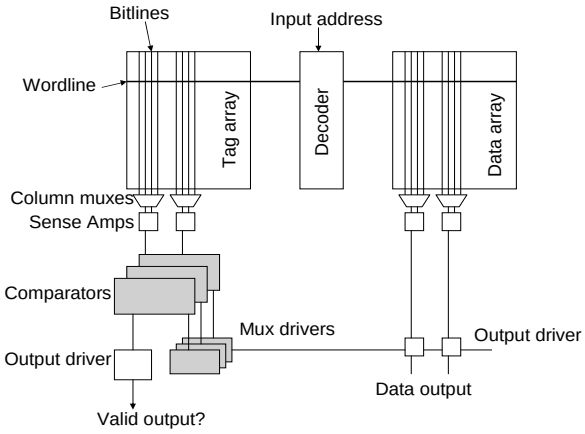
A recent paper by Muralimanohar and Balasubramonian [27] describes a methodology to extend CACTI's design space exploration to estimate an optimal NUCA cache organization. That work also proposed techniques to exploit specific wires and topologies for the address network. This work adopts a similar initial strategy when considering NUCA organizations. In addition to the creation of a tool for public distribution, we include a number of features that are not part of prior work:

- Extend the design space exploration to different wire and router types.
- Consider the use of low-swing differential signaling in addition to traditional global wires.
- Incorporate the effect of network contention during the design space exploration.
- Take bank cycle time into account in estimating the cache bandwidth.
- Validate a subset of the newly incorporated models.
- Improve upon the tool API, including the ability to specify novel metrics involving power, delay, area, and bandwidth.
- Provide insight on the tool's design space exploration and trade-off analysis, as well as example case studies.

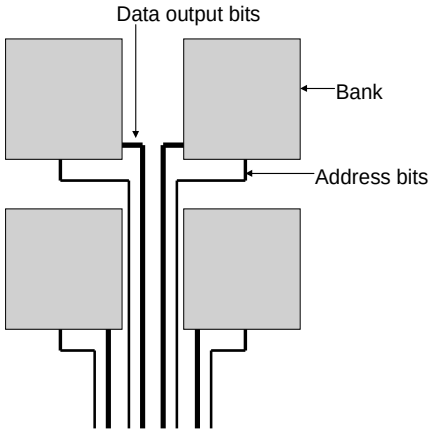
3. Background

This section presents some basics on the CACTI 5.0 cache access model. Figure 2(a) shows the basic logical

[2] As the cache size increases, the bitline power component also grows. Hence, the contribution of H-tree power as a percentage remains roughly constant.



(a) Logical organization of a cache.



(b) Example physical organization of the data array.

Figure 2. Logical and physical organization of the cache (from CACTI 3.0 [32]).

structure of a uniform cache access (UCA) organization. The address is provided as input to the decoder, which then activates a wordline in the data array and tag array. The contents of an entire row (referred to as a *set*) are placed on the bitlines, which are then sensed. The multiple tags thus read out of the tag array are compared against the input address to detect if one of the ways of the set does contain the requested data. This comparator logic drives the multiplexor that finally forwards at most one of the ways read out of the data array back to the requesting processor.

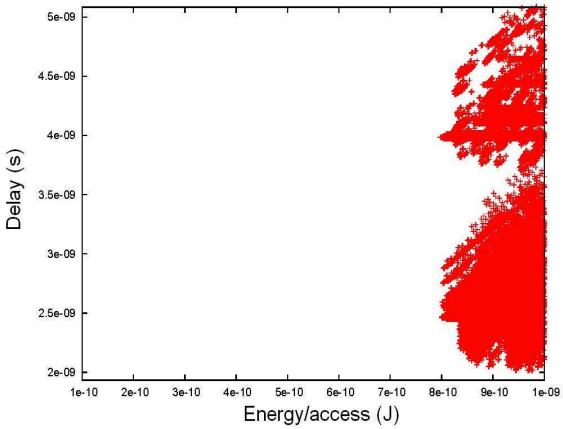
The CACTI cache access model [35] takes in the following major parameters as input: cache capacity, cache block size (also known as cache line size), cache associativity, technology generation, number of ports, and number of independent banks (not sharing address and data lines). As output, it produces the cache configuration that minimizes delay (with a few exceptions), along with its power and area characteristics. CACTI models the delay/power/area of eight major cache components: decoder, wordline, bitline, senseamp, comparator, multiplexor, output driver, and inter-bank wires. The wordline and bitline delays are two of the most significant components of the access time. The wordline and bitline delays are quadratic functions of the width and height of each array, respectively. In practice, the tag and data arrays are large enough that it is inefficient to implement them as single large structures. Hence, CACTI partitions each storage array (in the horizontal and vertical dimensions) to produce smaller *sub-arrays* and reduce wordline and bitline delays. The bitline is partitioned into *Ndbl* different segments, the wordline is partitioned into *Ndwl* segments, and so on. Each sub-array has its own decoder and some central pre-decoding is now required to route the request to the correct sub-array. The most recent version of CACTI employs a model for semi-global (intermediate) wires and an H-tree network to compute the delay between the predecode circuit and the furthest cache sub-array. CACTI carries out an exhaustive search across different sub-array counts (different values of Ndbl, Ndwl, etc.) and sub-array aspect ratios to compute the cache organization with optimal total delay. Typically, the cache is organized into a handful of sub-arrays. An example of the cache's physical structure is shown in Figure 2(b).

4. CACTI 6.0 Enhancements

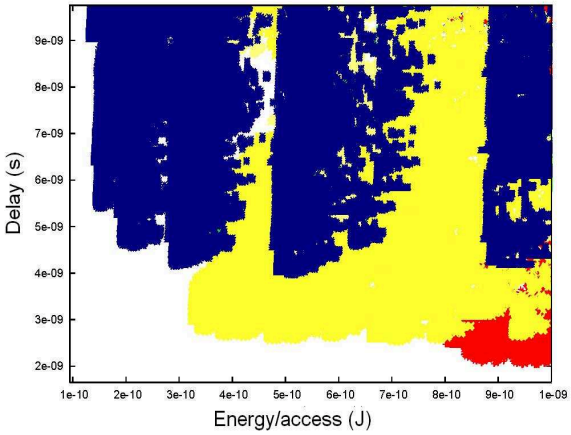
4.1. Interconnect Modeling for UCA Caches

As already shown in Figure 1, as cache size increases, the interconnect (within the H-tree network) plays an increasingly greater role in terms of access time and power. The interconnect overhead is impacted by (i) the number of sub-arrays, (ii) the signaling strategy, and (iii) the wire parameters. While prior versions of CACTI iterate over the number of sub-arrays by exploring different values of Ndbl, Ndwl, Nspd, etc., the model is restricted to a single signaling strategy and wire type (global wire). Thus, the design space exploration sees only a modest amount of variation in the component that dominates the overall cache delay and power. We therefore extend the design space exploration to also include a low-swing differential signaling strategy as well as the use of local and *fat* wires.

The delay of a wire is a function of its RC time constant which increases quadratically with the length of the wire. The insertion of optimally sized repeaters at regular intervals makes the delay a linear function of wire length. For long latency wires, the wire throughput can be increased by inserting pipelined latches at regular intervals. However, the use of repeaters and pipeline latches at regular intervals requires that the voltage levels on these wires swing across the full range $(0 - V_{dd})$ for proper operation. Given the quadratic dependence between voltage and power, these full-swing wires dissipate a large amount of power. Also, the silicon area requirement imposed by repeaters and latches precludes the possibility of routing these wires on top of other modules. In essence, traditional global wires, of the kind employed in CACTI 5.0, are fast but entail high power and routing complexity. Further, the delay, power, and bandwidth properties of these wires can be varied by changing the following parameters: (i) wire width, (ii) wire spacing, (iii) repeater size, and (iv) repeater spacing. By considering these choices, a segment on the H-tree network can be made to have optimal delay while having much higher power and area requirements. Alternatively, the segment can be made to have low power requirements while incurring a delay or area penalty. The considered wire types and their properties are summarized later in this section. By examining these choices and carrying out a more comprehensive design space explo-



(a) Design space exploration with global wires



(b) Design space exploration with full-swing global wires (red, bottom region), wires with 30% delay penalty (yellow, middle region), and differential low-swing wires (blue, top region)

Figure 3. Power/Delay trade-off in a 16MB UCA cache

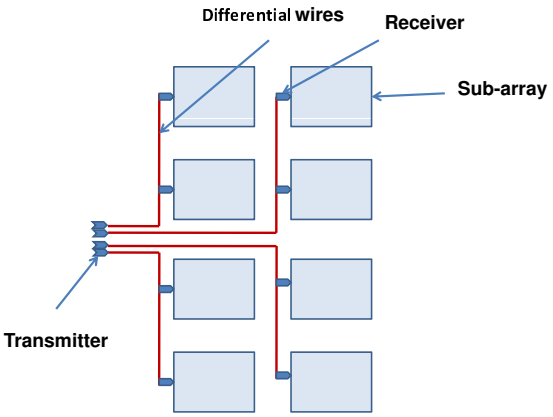


Figure 4. 8-bank data array with a differential low-swing broadcast bus.

Memory intensive benchmarks	applu, fma3d, swim, lucas equake, gap, vpr, art
L2/L3 latency sensitive benchmarks	ammp, apsi, art, bzip2, crafty, eon, equake, gcc
Half latency sensitive & half non-latency sensitive benchmarks	ammp, applu, lucas, bzip2 crafy, mgrid, mesa, gcc
Random benchmark set	Entire SPEC suite

Table 1. Benchmark sets

ration, CACTI 6.0 is able to identify cache organizations that better meet the user-specified metrics. Figure 3(a) shows a power-delay curve where each point represents one of the many hundreds of cache organizations considered by CACTI 5.0. The red points represent the cache organizations that would have been considered by CACTI 5.0 with its limited design space exploration with a single wire type (global wire with delay-optimal repeaters). The yellow points (middle region) in Figure 3(b) represent cache organizations with different wire types that are considered by CACTI 6.0. Clearly, by considering the trade-offs made possible with wires and expanding the search space, CACTI 6.0 is able to identify cache organizations with very relevant delay and power values.

One of the primary reasons for the high power dissipation of global wires is the full swing requirement imposed by the repeaters. While we are able to somewhat reduce the power requirement by reducing repeater size and increasing repeater spacing, the requirement is still relatively high. Low voltage swing alternatives represent another mechanism to vary the wire power/delay/area trade-off. Reducing the voltage swing on global wires can result in a linear reduction in power. In addition, assuming a separate voltage source for low-swing drivers will result in a quadratic savings in power. But, these lucrative power savings are accompanied by many caveats. Since we can no longer use repeaters or latches, the delay of a low-swing wire increases quadratically with length. Since such a wire cannot be pipelined, they also suffer from lower throughput. A low-swing wire requires special transmitter and receiver circuits for signal generation and amplification. This not only increases the area requirement per bit, but also assigns a fixed cost in terms of both delay and power for each bit traversal. In spite of these issues, the power savings possible through low-swing signalling makes it an attractive design choice. The detailed methodology for the design of low-swing wires and their overhead is described later in this section. In general, low-swing wires have superior power characteristics but incur high area and delay overheads.

The choice of an H-tree network for CACTI 5.0 (and earlier versions of CACTI) was made for the following reason: it enables uniform access times for each bank, which in turn, simplifies the pipelining of requests across the network. Since low-swing wires cannot be pipelined and since they better amortize the transmitter/receiver overhead over long transfers, we adopt a different network style when using low-swing wires. Instead of the H-tree network, we adopt a collection of simple broadcast buses that span across all the banks (each bus is shared by half the banks in a column – an example with eight banks is shown in Figure 4). The banks continue to have uniform access times, as determined by the worst-case delay. Since the bus is not pipelined, the wire delay limits the throughput as well and decreases the operating frequency of the

Fetch queue size	64	Branch predictor	comb. of bimodal and 2-level
Bimodal predictor size	16K	Level 1 predictor	16K entries, history 12
Level 2 predictor	16K entries	BTB size	16K sets, 2-way
Branch mispredict penalty	at least 12 cycles	Fetch width	8 (across up to 2 basic blocks)
Dispatch and commit width	8	Issue queue size	60 (int and fp, each)
Register file size	100 (int and fp, each)	Re-order Buffer size	80
L1 I-cache	32KB 2-way	L1 D-cache	32KB 2-way set-associative,
L2 cache	32MB 8-way SNUCA		3 cycles, 4-way word-interleaved
L2 Block size	64B		
I and D TLB	128 entries, 8KB page size	Memory latency	300 cycles for the first chunk
Network topology	Grid	Flow control mechanism	Virtual channel
No. of virtual channels	4 /physical channel	Back pressure handling	Credit based flow control

Table 2. Simplescalar simulator parameters.

cache. The cycle time of a cache is equal to the maximum delay of a segment that cannot be pipelined. Typically, the sum of bitline and sense amplifier delay decides the cycle time of a cache. In a low-swing model, the cycle time is determined by the maximum delay of the low-swing bus. We also consider low-swing wires with varying width and spacing that further play into the delay/power/area trade-offs.

With low-swing wires included in the CACTI design space exploration, the tool is able to identify many more points that yield low power at a performance and area cost. The blue points (top region) in Figure 3(b) represent the cache organizations considered with low-swing wires. Thus, by leveraging different wire properties, it is possible to generate a broad range of cache models with different power/delay characteristics.

4.2. NUCA Modeling

The UCA cache model discussed so far has an access time that is limited by the delay of the slowest sub-bank. A more scalable approach for future large caches is to replace the H-tree bus with a packet-switched on-chip grid network. The latency for a bank is determined by the delay to route the request and response between the bank that contains the data and the cache controller. Such a NUCA model was first proposed by Kim et al. [21] and has been the subject of many architectural evaluations. We therefore extend CACTI to support such NUCA organizations as well.

The tool first iterates over a number of bank organizations: the cache is partitioned into 2^N banks (where N varies from 1 to 12); for each N, the banks are organized in a grid with 2^M rows (where M varies from 0 to N). For each bank organization, CACTI 5.0 is employed to determine the optimal sub-array partitioning for the cache within each bank. Each bank is associated with a router. The average delay for a cache access is computed by estimating the number of network hops to each bank, the wire delay encountered on each hop, and the cache access delay within each bank. We further assume that each traversal through a router takes up R cycles, where R is a user-specified input. Router pipelines can be designed in many ways: a four-stage pipeline is commonly advocated [13], and recently, speculative pipelines that take up three, two, and one pipeline stage have also been proposed [13, 26, 28]. While we give the user the option to pick an aggressive or conservative router, the tool defaults to employing a moderately aggressive router pipeline with three stages.

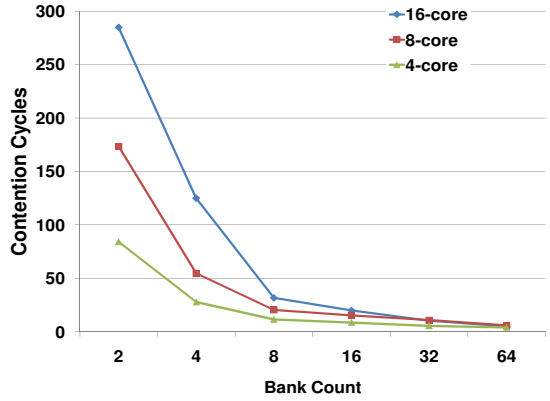
More partitions lead to smaller delays (and power) within each bank, but greater delays (and power) on the network (because of the constant overheads associated with each router and decoder). Hence, the above design space exploration is required to estimate the cache partition that yields optimal delay or power. The above algorithm was recently proposed by Muralimanohar and Balasubramonian [27]. We further extend this algorithm in the following ways.

First, we explore different wire types for the links between adjacent routers. These wires are modeled as low-swing differential wires as well as local/global/fat wires to yield many points in the power/delay/area spectrum.

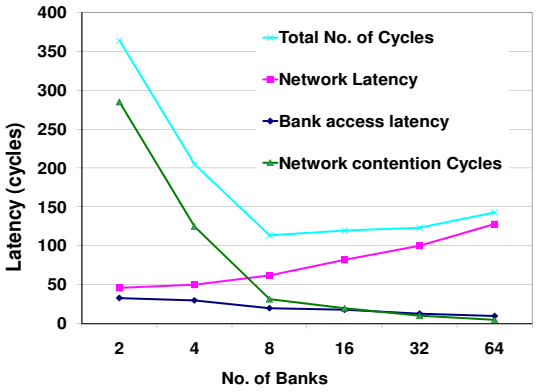
Second, we model different types of routers. The sizes of buffers and virtual channels within a router have a major influence on router power consumption as well as router contention under heavy load. By varying the number of virtual channels per physical channel and the number of buffers per virtual channel, we are able to achieve different points on the router power-delay trade-off curve.

Third, we model contention in the network in much greater detail. This itself has two major components. If the cache is partitioned into many banks, there are more routers/links on the network and the probability of two packets conflicting at a router decrease. Thus, a many-banked cache is more capable of meeting the bandwidth demands of a many-core system. Further, certain aspects of the cache access within a bank cannot be easily pipelined. The longest such delay within the cache access (typically the bitline and sense-amp delays) represents the cycle time of the bank – it is the minimum delay between successive accesses to that bank. A many-banked cache has relatively small banks and a relatively low cycle time, allowing it to support a higher throughput and lower wait-times once a request is delivered to the bank. Both of these two components (lower contention at routers and lower contention at banks) tend to favor a many-banked system. This aspect is also included in estimating the average access time for a given cache configuration.

The contention values for each considered NUCA cache organization are empirically estimated for typical workloads and incorporated into CACTI 6.0 as look-up tables. For each of the grid topologies considered (for different values of N and M), we simulated L2 requests originating from single-core, two-core, four-core, eight-core, and sixteen-core processors. Each core executes a mix of programs from the SPEC benchmark suite. We divide the benchmark set into four categories, as described



(a) Total network contention value/access for CMPs with different NUCA organizations



(b) Optimal NUCA organization

Figure 5. NUCA design space exploration.

in Table 1. For every CMP organization, we run four sets of simulations, corresponding to each benchmark set tabulated. The generated cache traffic is then modeled on a detailed network simulator with support for virtual channel flow control. Details of the architectural and network simulator are listed in Table 2. The contention value (averaged across the various workloads) at routers and banks is estimated for each network topology and bank cycle time. Based on the user-specified inputs, the appropriate contention values in the look-up table are taken into account during the design space exploration. Some of this empirical data is represented in Figure 5(a). We observe that for many-core systems, the contention in the network can be as high as 30 cycles per access (for a two banked model) and cannot be ignored during the design space exploration.

For a network with completely pipelined links and routers, these contention values are only a function of the router topology and bank cycle time and will not be affected by process technology or L2 cache size[3]. If CACTI is being employed to compute an optimal L3 cache organization, the contention values will likely be much less because the L2 cache filters out most requests. To handle this case, we also computed the average contention values assuming a large 2 MB L1 cache and this is incorporated into the model as well. In summary, the network contention values are impacted by the following parameters: M, N, bank cycle time, number of cores, router configuration (VCs, buffers), size of preceding cache. We plan to continue augmenting the tool with empirical contention values for other relevant sets of workloads such as commercial, multi-threaded, and transactional benchmarks with significant traffic from cache coherence.

Figure 5(b) shows an example design space exploration for a 32 MB NUCA L2 cache while attempting to minimize latency. The X-axis shows the number of banks that the cache is partitioned into. For each point on the X-axis, many different bank organizations are considered and the organization with optimal delay (averaged across all banks) is finally represented on the graph. The Y-axis represents this optimal delay and it is further broken down to represent the contributing components: bank access time, link and router delay, router and bank contention. We observe that the optimal delay is experienced when the cache is organized as a 2×4 grid of 8 banks.

[3]We assume here that the cache is organized as static-NUCA (SNUCA), where the address index bits determine the unique bank where the address can be found and the access distribution does not vary greatly as a function of the cache size. CACTI is designed to be more generic than specific. The contention values are provided as a guideline to most users. If a user is interested in a more specific NUCA policy, there is no substitute to generating the corresponding contention values and incorporating them in the tool. As a case study in Section 5, we examine a different NUCA policy.

4.3. Wire Models

This section details the analytical model for delay and power calculation of different wires. We begin with a description of the delay and power model for global wires, then describe how wires with different power-delay characteristics can be modeled. Finally, we discuss the methodology for calculating delay and power for low-swing wires.

4.3.1 Full-Swing Repeated Wires

For full-swing wires, the delay of a wire is governed by its RC time constant (R is resistance, C is capacitance). The resistance and capacitance per unit length are governed by the following equations [17]:

$$R_{wire} = \frac{\rho}{(thickness \; - \; barrier)(width \; - \; 2\,barrier)} \quad (1)$$

$$C_{wire} = \epsilon_0(2K\epsilon_{horiz}\frac{thickness}{spacing} + 2\epsilon_{vert}\frac{width}{layerspacing}) + fringe(\epsilon_{horiz}, \epsilon_{vert})$$

$Thickness$ and $width$ represent the geometrical dimensions of the wire cross-section, $barrier$ represents the thin barrier layer around the wire to prevent copper from diffusing into surrounding oxide, and ρ is the material resistivity. The potentially different relative dielectrics for the vertical and horizontal capacitors are represented by

ϵ_{horiz} and ϵ_{vert}, K accounts for Miller-effect coupling capacitances, $spacing$ represents the gap between adjacent wires on the same metal layer, and $layerspacing$ represents the gap between adjacent metal layers.

A change in the width and spacing of wires yields different delay and power characteristics. For the CACTI design space exploration, we restrict ourselves to semi-global, global, and fat wires that have width and spacing in the ratio 1:2:16. As a result, the latencies for these wires are in the ratio 8:4:1, and their power consumption is in the ratio 2:1.8:1.

It has also been derived that a wire yields optimal delay if repeaters have the following spacing ($L_{optimal}$) and size ($S_{optimal}$) [4]:

$$L_{optimal} = \sqrt{\frac{2r_s(c_0 + c_p)}{R_{wire}C_{wire}}} \quad (2)$$

$$S_{optimal} = \sqrt{\frac{r_s C_{wire}}{R_{wire}c_0}} \quad (3)$$

In the above equations, c_0 is the capacitance of the minimum sized repeater, c_p is its output parasitic capacitance, and r_s is its output resistance. Banerjee et al. [4] describe a methodology to compute a repeater configuration that minimizes power while giving up performance. We adopt a similar methodology to compute the power-delay trade-off for various repeater configurations. Figure 6 shows the relative power and delay as a function of wire length for a delay-optimal global wire and wires that trade-off delay for lower power.

4.3.2 Differential Low-swing Wires

A low-swing interconnect system consists of three main components: (1) a transmitter that generates the low-swing signal, (2) twisted differential wires, and (3) a receiver amplifier. For the transmitter circuit, we employ the model proposed by Ho et al. [18]. To improve delay characteristics (equations not reproduced here), the transmitter circuit uses pre-emphasis and pre-equalization optimization techniques. Pre-emphasis reduces the wire charging/discharging time by using a drive voltage significantly higher than the minimum signal required by the receiver. Pre-equalizing the wires enables signal transfer with half the voltage swing.

The total capacitance of the low-swing segment is given by

$$C_{load} = C_{wire} + 2 * C_{drain} + C_{sense\_amp}$$

C_{drain} is the drain capacitance of the driver transistor. The dynamic energy is expressed as $C_{load}.V_{overDrive}.V_{lowswing}$. For our evaluations, we assume an overdrive voltage of 200mV and a low swing voltage of 100mV. At the receiver, we employ the same sense-amplifier circuit used by CACTI for its bitline sensing [35]. The power and delay characteristics of low-swing wires are also represented in Figure 6.

4.4. Router Models

As discussed earlier, various routers have been proposed with differing levels of speculation and pipeline stages [13, 26, 28]. The number of stages for each router is left as a user-specified input, defaulting to 3 cycles. For router power, we employ the analytical power models for crossbars and arbiters employed in the Orion toolkit [41]. CACTI's RAM model is employed for router buffer power. These represent the primary contributors to network power (in addition to link power, that was discussed in the previous sub-section). We restrict ourselves to a grid topology where each router has 5 inputs and 5 outputs, and consider three points on the power-performance trade-off curve. Each point provides a different number of buffers per virtual channel and a different number of virtual channels per physical channel. Accordingly, we see a significant variation in buffer capacity (and power) and contention cycles at routers. As before, the contention cycles are computed with detailed network simulations. Table 3 specifies the three types of routers and their corresponding buffer, crossbar, and arbiter energy values.

4.5. Improvement in Trade-Off Analysis

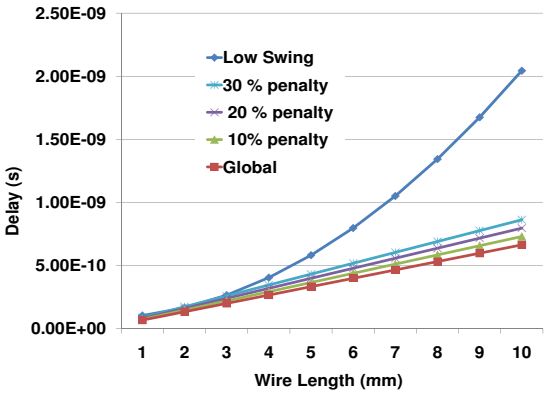
For architectural studies, especially those related to memory hierarchy design, an early estimate of cache access time and power for a given input configuration is crucial in making a sound evaluation. As described in Section 3, CACTI 5.0 carries out a design space exploration over various sub-array partitions; it then eliminates organizations that have an area that is 50% higher than the optimal area; it further eliminates those organizations that have an access time value more than 10% the minimum value; and finally selects an organization using a cost function that minimizes power and cycle time.

Modern processor design is not singularly focused on performance and many designers are willing to compromise some performance for improved power. Many future studies will likely carry out trade-off analyses involving performance, power, and area. To facilitate such analyses, the new version of the tool adopts the following cost function to evaluate a cache organization (taking into account delay, leakage power, dynamic power, cycle time, and area):

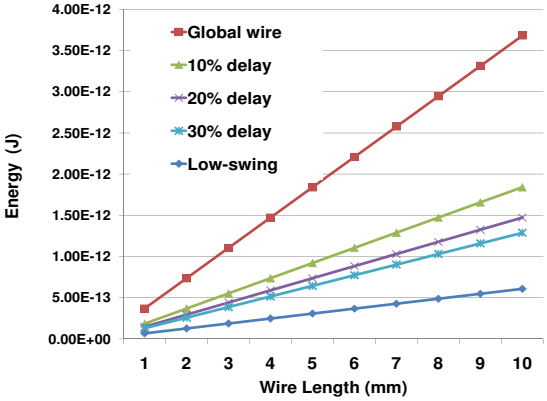
$$\begin{aligned} cost = \; & W_{acc\_time}\frac{acc\_time}{min\_acc\_time} + \\ & W_{dyn\_power}\frac{dyn\_power}{min\_dyn\_power} + \\ & W_{leak\_power}\frac{leak\_power}{min\_leak\_power} + \\ & W_{cycle\_time}\frac{cycle\_time}{min\_cycle\_time} + \\ & W_{area}\frac{area}{min\_area} \end{aligned}$$

The weights for each term ($W_{acc\_time}, W_{dyn\_power}, W_{leak\_power}, W_{cycle\_time}, W_{area}$) indicate the relative importance of each term and these are specified by the user as input parameters in the configuration file:

```
-weight 100 20 20 10 10
```



(a) Delay characteristics of different wires



(b) Energy characteristics of different wires

Figure 6. Energy/Delay values for different wires

Component	Configuration 1 4 VCs/PC; 16 buffers/VC	Configuration 2 2VCs/PC; 8 buffers/VC	Configuration 3 2 VCs/PC; 2 buffers/VC
Arbiter	0.33e-12	0.27e-12	0.27e-12
Crossbar (avg)	0.99e-11	0.99e-11	0.99e-11
Buffer read operation/VC	0.11e-11	0.76e-12	0.50e-12
Write buffer operation/VC	0.14e-11	0.10e-11	0.82e-12

Table 3. Energy consumed (in J) by arbiters, buffers and crossbars for various router configurations at 32nm technology (flit size of 128 bits).

The above default weights used by the tool reflect the priority of these metrics in a typical modern design. In addition, the following default line in the input parameters specifies the user's willingness to deviate from the optimal set of metrics:

```
-deviate  1000 1000 1000 1000 1000
```

The above line dictates that we are willing to consider a cache organization where each metric, say the access time, deviates from the lowest possible access time by 1000%. Hence, this default set of input parameters specifies a largely unconstrained search space. The following input lines restrict the tool to identify a cache organization that yields least power while giving up at most 10% performance:

```
-weight   0  100 100  0  0
-deviate 10  1000 1000 1000 1000
```

4.6. Validation

In this work, we mainly focus on validating the new modules added to the framework. This includes low-swing wires, router components, and improved bitline and wordline models. Since SPICE results depend on the model files for transistors, we first discuss the technology modeling changes made to the recent version of CACTI (version 5) and later detail our methodology for validating the newly added components to CACTI 6.0.

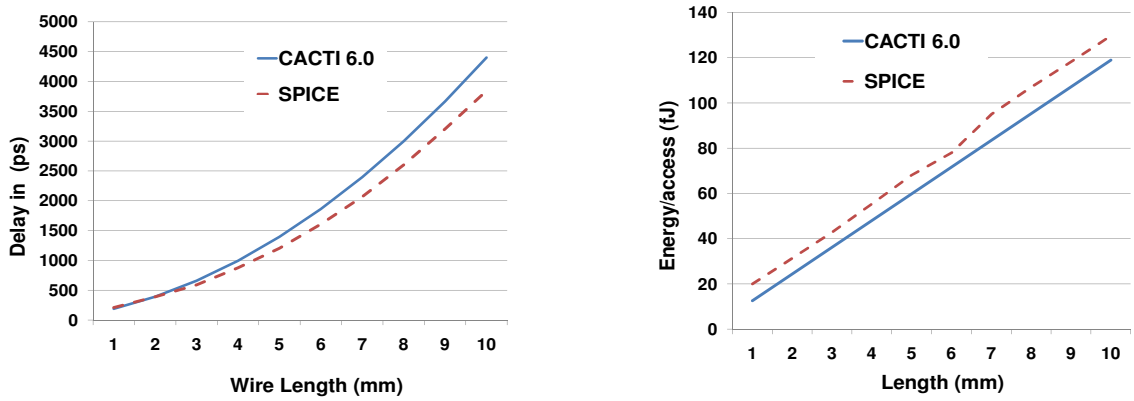
Earlier versions of CACTI (version one through four) assumed linear technology scaling for calculating cache parameters. All the power, delay, and area values are first calculated for 800nm technology and the results are linearly scaled to the user specified process value. While this approach is reasonably accurate for old process technologies, it can introduce non-trivial error for deep sub-micron technologies (less than 90nm). This problem is fixed in CACTI 5 [36] by adopting ITRS parameters for all calculations. The current version of CACTI supports four different process technologies (90nm, 65nm, 45nm, and 32nm) with process specific values obtained from ITRS. Though ITRS projections are invaluable for quick analytical estimates, SPICE validation requires technology model files with greater detail and ITRS values cannot be directly plugged in for SPICE verification. The only non-commercial data available publicly for this purpose for recent process technologies is the Predictive Technology Model (PTM) [2]. For our validation, we employ the HSPICE tool along with the PTM 65 nm model file for validating the newly added components. The simulated values obtained from HSPICE are compared against CACTI 6.0 analytical models that take PTM parameters as input [4]. The analytical delay and power calculations performed by the tool primarily depend on the resistance and capacitance parasitics of transistors. For our validation, the capacitance values of source, drain, and gate of n and p transistors are derived from the PTM technology model file. The threshold voltage and the on-resistance of the transistors are calculated using SPICE simulations. In addition to modeling the gate delay and wire delay of different components, our analytical model also considers the delay penalty incurred due to the finite rise time and fall time of an input signal [42].

Figure 7 (a) & (b) show the comparison of delay and power values of the differential, low-swing analyti-

[4]The PTM parameters employed for verification can be directly used for CACTI simulations. Since most architectural and circuit studies rely on ITRS parameters, CACTI by default assumes ITRS values to maintain consistency.

Transmitter	Delay, SPICE - 84ps, CACTI - 92ps	Power, SPICE - 7.2fJ , CACTI - 7.5fJ
Differential Wires	Delay, SPICE - 1.204ns, CACTI - 1.395ns	Power, SPICE - 29.9fJ, CACTI - 34fJ
Sense amplifier	Delay, SPICE - 200ps	Power, SPICE - 5.7fJ

Table 4. Delay and energy values of different components for a 5mm low-swing wire.



(a) Delay verification

(b) Energy verification

Figure 7. Low-swing model verification

cal models against SPICE values. As mentioned earlier, a low-swing wire model can be broken into three components: transmitter (that generates the low-swing signal), differential wires[5], and sense amplifiers. The modeling details of each of these components are discussed in section 4.3.2. Table 4 shows the delay and power values of each of these compenents for a 5mm low-swing wire. Though the analytical model employed in CACTI 6.0 dynamically calculates the driver size appropriate for a given wire length, for the wire length of our interest, it ends up using the maximum driver size (which is set to 100 times the minimum transistor size) to incur minimum delay overhead. Earlier versions of CACTI also had the problem of over estimating the delay and power values of the sense-amplifier. CACTI 6.0 eliminates this problem by directly using the SPICE generated values for sense-amp power and delay. On an average, the low-swing wire models are verified to be within 12% of the SPICE values.

The lumped RC model used in prior versions of CACTI for bitlines and wordlines are replaced with a more accurate distributed RC model in CACTI 6.0. Based on a detailed spice modeling of the bitline segment along with the memory cells, we found the difference between the old and new model to be around 11% at 130 nm technology. This difference can go up to 50% with shrinking process technologies as wire parasitics become the dominant factor compared to transistor capacitance [29]. Figure 8 (a) & (b) compare the distributed wordline and bitline delay values and the SPICE values. The length of the wordlines or bitlines (specified in terms of memory array size) are carefully picked to represent a wide range of cache sizes. On an average, the new analytical models for the distributed wordlines and bitlines are verified to be within 13% and 12% of SPICE generated values.

Buffers, crossbars, and arbiters are the primary components in a router. CACTI 6.0 uses its scratch RAM model to calculate read/write power for router buffers. We employ Orion's arbiter and crossbar model for calculating router power and these models have been validated by Wang et al. [40].

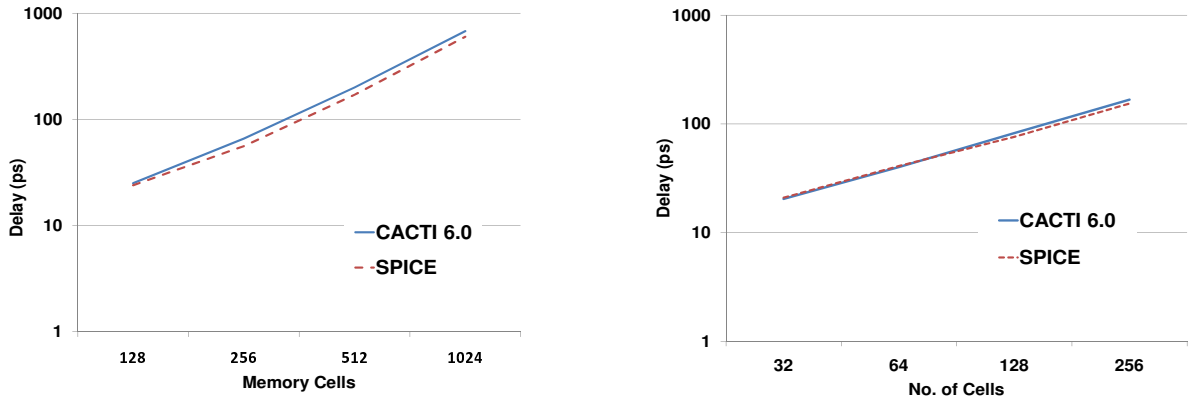
[5]Delay and power values of low-swing driver is also reported as part of differential wires.

5. Case Study

We expect that CACTI 6.0 will continue to be used in architectural evaluations in many traditional ways: it is often used to estimate cache parameters while setting up architectural simulators. The new API makes it easier for users to make power/delay/area trade-offs and we expect this feature to be heavily used for architectural evaluations that focus on power-efficiency or are attempting to allocate power/area budgets to cache or processing. With many recent research proposals focused on NUCA organizations, we also expect the tool to be heavily used in that context. Since it is difficult to generalize NUCA implementations, we expect that users modeling NUCA designs may need to modify the model's parameters and details to accurately reflect their NUCA implementation. Hence, as a case study of the tool's operation, we present an example NUCA evaluation and its inter-play with CACTI 6.0.

Many recent NUCA papers have attempted to improve average cache access time by moving heavily accessed data to banks that are in close proximity to the core [6, 12, 19, 20, 21]. This is commonly referred to as dynamic-NUCA or D-NUCA becasue a block is no longer mapped to a unique bank and can move between banks during its L2 lifetime. We first postulate a novel idea and then show how CACTI 6.0 can be employed to evaluate that idea. Evaluating and justifying such an idea could constitute an entire paper – we are simply focusing here on a high-level evaluation that highlights the changes required to CACTI 6.0.

The Proposal: For a D-NUCA organization, most requests will be serviced by banks that are close to the cache controller. Further, with D-NUCA, it is possible that initial banks will have to be searched first and the request forwarded on if the data is not found. All of this implies that initial banks see much higher activity than distant banks. To reduce the power consumption of the NUCA cache, we



(a) Wordline (b) Bitline

Figure 8. Distributed wordline and bitline model verification

propose that heterogeneous banks be employed: the initial banks can employ smaller power-efficient banks while the distant banks can employ larger banks.

For our case study evaluation, we will focus on a grid-based NUCA cache adjacent to a single core. The ways of a set are distributed across the banks, so a given address may reside in one of many possible banks depending on the way it is assigned to. Similar to D-NUCA proposals in prior work [21], when a block is brought into the cache, it is placed in the most distant way and it is gradually migrated close to the cache controller with a swap between adjacent ways on every access. While looking for data, each candidate bank is sequentially looked up until the data is found or a miss is signaled.

Recall that CACTI 6.0 assumes an S-NUCA organization where sets are distributed among banks and each address maps to a unique bank. When estimating average access time during the design space exploration, it is assumed that each bank is accessed with an equal probability. The network and bank contention values are also estimated for an S-NUCA organization. Thus, two changes have to be made to the tool to reflect the proposed implementation:

- The design space exploration must partition the cache space into two: the bank sizes for each partition are estimated independently, allowing the initial banks to have one size and the other banks to have a different size.
- Architectural evaluations have to be performed to estimate the access frequencies for each bank and contention values so that average access time can be accurately computed.

With our simulation infrastructure, we considered a 32 MB 16-way set-associative L2 cache and modeled the migration of blocks across ways as in the above D-NUCA policy. Based on this, the access frequency as shown in Figure 9 was computed, with many more accesses to initial banks (unlike the S-NUCA case where the accesses per bank are uniform). With this data integrated into CACTI 6.0, the design space exploration loop of CACTI 6.0 was wrapped around with the following loop structure:

```
for i = 0 to 100
```

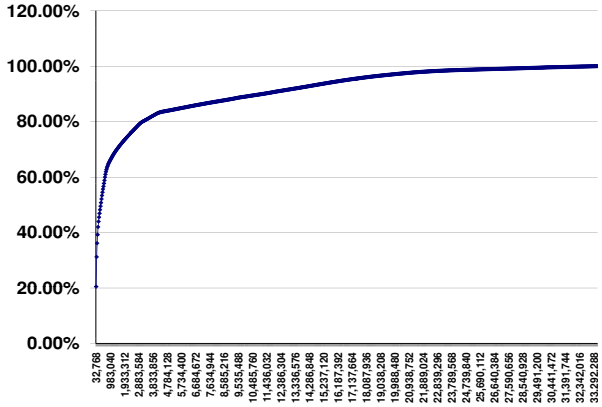


Figure 9. Access frequency for a 32MB cache. The y-coordinate of a point in the curve corresponds to the percentage of accesses that can be satisfied with x KB of cache.

```
# Assume i% of the cache has one
# bank size and the remaining
# (100-i)% has a different bank size
for the first i% of cache,
    perform CACTI 6.0 exploration
    (with new access frequencies
     and contention)
for the remaining (100-i)% of cache,
    perform CACTI 6.0 exploration
    (with new access frequencies
    and contention)
```

As an input, we provide the `-weight` and `-deviate` parameters to specify that we are looking for an organization that minimizes power while yielding performance within 10% of optimal. The output from this modified CACTI 6.0 indicates that the optimal organization employs a bank size of 4MB for the first 16MB of the cache and a bank size of 8MB for the remaining 16MB. The average power consumption for this organization is 20% lower than the average power per access for the S-NUCA organization yielded by unmodified CACTI 6.0.

6. Conclusions

This paper describes major revisions to the CACTI cache modeling tool. Interconnect plays a major role in the delay and power of large caches and we extended CACTI's design space exploration to carefully consider many different implementation choices for the interconnect components, including different wire types, routers, signaling strategy, and modeling contention. We also added modeling support for a wide range of NUCA caches. CACTI 6.0 identifies a number of relevant design choices on the power-delay-area curves. The estimates of CACTI 6.0 can differ from the estimates of CACTI 5.0 significantly, especially when more fully exploring the power-delay trade-off space. CACTI 6.0 is able to identify cache configurations that can reduce power by a factor of three, while incurring a 25% delay penalty. We validated components of the tool against Spice simulations and showed good agreement between analytical and transistor-level models. Finally, we present an example case study of heterogeneous NUCA banks that demonstrates how the tool can benefit architectural evaluations.

References

[1] V. Agarwal, M. Hrishikesh, S. Keckler, and D. Burger. Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures. In *Proceedings of ISCA-27*, pages 248–259, June 2000.

[2] Arizona State University. Predictive Technology Model. http://www.eas.asu.edu/˜ptm.

[3] H. Bakoglu and J. Meindl. A System-Level Circuit Model for Multi- and Single-Chip CPUs. In *Proceedings of ISSCC*, 1987.

[4] K. Banerjee and A. Mehrotra. A Power-optimal Repeater Insertion Methodology for Global Interconnects in Nanometer Designs. *IEEE Transactions on Electron Devices*, 49(11):2001–2007, November 2002.

[5] B. Beckmann, M. Marty, and D. Wood. ASR: Adaptive Selective Replication for CMP Caches. In *Proceedings of MICRO-39*, December 2006.

[6] B. Beckmann and D. Wood. Managing Wire Delay in Large Chip-Multiprocessor Caches. In *Proceedings of MICRO-37*, December 2004.

[7] B. Black, M. Annavaram, E. Brekelbaum, J. DeVale, L. Jiang, G. Loh, D. McCauley, P. Morrow, D. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb. Die Stacking (3D) Microarchitecture. In *Proceedings of MICRO-39*, December 2006.

[8] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proceedings of ISCA-27*, pages 83–94, June 2000.

[9] A. Caldwell, Y. Cao, A. Kahng, F. Koushanfar, H. Lu, I. Markov, M. Oliver, D. Stroobandt, and D. Sylvester. GTX: The MARCO GSRC Technology Extrapolation System. In *Proceedings of DAC*, 2000.

[10] J. Chang and G. Sohi. Co-Operative Caching for Chip Multiprocessors. In *Proceedings of ISCA-33*, June 2006.

[11] Z. Chishti, M. Powell, and T. Vijaykumar. Distance Associativity for High-Performance Energy-Efficient Non-Uniform Cache Architectures. In *Proceedings of MICRO-36*, December 2003.

[12] Z. Chishti, M. Powell, and T. Vijaykumar. Optimizing Replication, Communication, and Capacity Allocation in CMPs. In *Proceedings of ISCA-32*, June 2005.

[13] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 1st edition, 2003.

[14] J. Eble. A Generic System Simulator (Genesys) for ASIC Technology and Architecture Beyond 2001. In *Proceedings of 9th IEEE International ASIC Conference*, 1996.

[15] N. Eisley, L.-S. Peh, and L. Shang. In-Network Cache Coherence. In *Proceedings of MICRO-39*, December 2006.

[16] B. M. Geuskins. *Modeling the Influence of Multilevel Interconnect on Chip Performance*. PhD thesis, Rensselaer Polytechnic Institute, Troy, New York, 1997.

[17] R. Ho, K. Mai, and M. Horowitz. The Future of Wires. *Proceedings of the IEEE*, Vol.89, No.4, April 2001.

[18] R. Ho, K. Mai, and M. Horowitz. Managing Wire Scaling: A Circuit Prespective. *Interconnect Technology Conference*, pages 177–179, June 2003.

[19] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. Keckler. A NUCA Substrate for Flexible CMP Cache Sharing. In *Proceedings of ICS-19*, June 2005.

[20] Y. Jin, E. J. Kim, and K. H. Yum. A Domain-Specific On-Chip Network Design for Large Scale Cache Systems. In *Proceedings of HPCA-13*, February 2007.

[21] C. Kim, D. Burger, and S. Keckler. An Adaptive, Non-Uniform Cache Structure for Wire-Dominated On-Chip Caches. In *Proceedings of ASPLOS-X*, October 2002.

[22] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, N. Vijaykrishnan, and M. Kandemir. Design and Management of 3D Chip Multiprocessors Using Network-in-Memory. In *Proceedings of ISCA-33*, June 2006.

[23] G. Loi, B. Agrawal, N. Srivastava, S. Lin, T. Sherwood, and K. Banerjee. A Thermally-Aware Performance Analysis of Vertically Integrated (3-D) Processor-Memory Hierarchy. In *Proceedings of DAC-43*, June 2006.

[24] M. Mamidipaka and N. Dutt. eCACTI: An Enhanced Power Estimation Model for On-Chip Caches. Technical Report CECS Technical Report 04-28, University of California, Irvine, September 2004.

[25] C. McNairy and R. Bhatia. Montecito: A Dual-Core, Dual-Thread Itanium Processor. *IEEE Micro*, 25(2), March/April 2005.

[26] R. Mullins, A. West, and S. Moore. Low-Latency Virtual-Channel Routers for On-Chip Networks. In *Proceedings of ISCA-31*, May 2004.

[27] N. Muralimanohar and R. Balasubramonian. Interconnect Design Considerations for Large NUCA Caches. In *Proceedings of the 34th International Symposium on Computer Architecture (ISCA-34)*, June 2007.

[28] L.-S. Peh and W. Dally. A Delay Model and Speculative Architecture for Pipelined Routers. In *Proceedings of HPCA-7*, 2001.

[29] J. M. Rabaey. *Digital Integrated Circuits*.

[30] J. Rattner. Predicting the future, 2005. Keynote at Intel Developer Forum, http://www.anandtech.com/tradeshows /showdoc.aspx?i=2367&p=3.

[31] G. Reinman and N. Jouppi. CACTI 2.0: An Integrated Cache Timing and Power Model. Technical Report 2000/7, WRL, 2000.

[32] P. Shivakumar and N. P. Jouppi. CACTI 3.0: An Integrated Cache Timing, Power, and Area Model. Technical Report TN-2001/2, Compaq Western Research Laboratory, August 2001.

[33] E. Speight, H. Shafi, L. Zhang, and R. Rajamony. Adaptive Mechanisms and Policies for Managing Cache Hierarchies in Chip Multiprocessors. In *Proceedings of ISCA-32*, June 2005.

[34] D. Sylvester and K. Keutzer. System-Level Performance Modeling with BACPAC - Berkeley Advanced Chip Performance Calculator. In *Proceedings of 1st International Workshop on System-Level Interconnect Prediction*, 1999.

[35] D. Tarjan, S. Thoziyoor, and N. Jouppi. CACTI 4.0. Technical Report HPL-2006-86, HP Laboratories, 2006.

[36] S. Thoziyoor, N. Muralimanohar, and N. P. Jouppi. CACTI 5.0: An Integrated Cache Timing, Power, and Area Model. Technical report, HP Laboratories Palo Alto, 2007.

[37] Y.-F. Tsai, Y. Xie, N. Vijaykrishnan, and M. Irwin. Three-Dimensional Cache Design Using 3DCacti. In *Proceedings of ICCD*, October 2005.

[38] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar. An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS. In *Proceedings of ISSCC*, February 2007.

[39] V. Venkatraman, A. Laffely, J. Jang, H. Kukkamalla, Z. Zhu, and W. Burleson. NOCIC: A Spice-Based Interconnect Planning Tool Emphasizing Aggressive On-Chip Interconnect Circuit Methods. In *Proceedings of International Workshop on System Level Interconnect Prediction*, February 2004.

[40] H.-S. Wang, L.-S. Peh, and S. Malik. A Power Model for Routers: Modeling Alpha 21364 and InfiniBand Routers. volume 23, January/February 2003.

[41] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: A Power-Performance Simulator for Interconnection Networks. In *Proceedings of MICRO-35*, November 2002.

[42] S. Wilton and N. Jouppi. An Enhanced Cache Access and Cycle Time Model. *IEEE Journal of Solid-State Circuits*, May 1996.

[43] S. J. E. Wilton and N. Jouppi. An Enhanced Access and Cycle Time Model for On-Chip Caches. Technical Report 93/5, WRL, 1994.

[44] M. Zhang and K. Asanovic. Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors. In *Proceedings of ISCA-32*, June 2005.