CACTI 6.0: A Tool to Model Large Caches

Naveen Muralimanohar, Rajeev Balasubramonian, Norman P. Jouppi

Keyword(s):
No keywords available.

Abstract:
Future processors will likely have large on-chip caches with a possibility of dedicating an entire die for on-chip storage in a 3D stacked design. With the ever growing disparity between transistor and wire delay, the properties of such large caches will primarily depend on the characteristics of the interconnection networks that connect various sub-modules of a cache. CACTI 6.0 is a significantly enhanced version of the tool that primarily focuses on interconnect design for large caches. In addition to strengthening the existing analytical model of the tool for dominant cache components, CACTI 6.0 includes two major extensions over earlier versions: first, the ability to model Non-Uniform Cache Access (NUCA), and second, the ability to model different types of wires, such as RC based wires with different power, delay, and area characteristics and differential low-swing buses. This report details the analytical model assumed for the newly added modules along with their validation analysis.

CACTI 6.0: A Tool to Model Large Caches

Naveen Muralimanohar[†], Rajeev Balasubramonian[†], Norman P. Jouppi[‡]
[†] School of Computing, University of Utah
[‡] Hewlett-Packard Laboratories

Abstract

Future processors will likely have large on-chip caches with a possibility of dedicating an entire die for on-chip storage in a 3D stacked design. With the ever growing disparity between transistor and wire delay, the properties of such large caches will primarily depend on the characteristics of the interconnection networks that connect various sub-modules of a cache. CACTI 6.0 is a significantly enhanced version of the tool that primarily focuses on interconnect design for large caches. In addition to strengthening the existing analytical model of the tool for dominant cache components, CACTI 6.0 includes two major extensions over earlier versions: first, the ability to model Non-Uniform Cache Access (NUCA), and second, the ability to model different types of wires, such as RC based wires with different power, delay, and area characteristics and differential low-swing buses. This report details the analytical model assumed for the newly added modules along with their validation analysis.

Contents

1 Background

This section presents some basics on the CACTI cache access model. Figure 1(a) shows the basic logical structure of a uniform cache access (UCA) organization. The address request to the cache is first provided as input to the decoder, which then activates a wordline in the data array and tag array. The contents of an entire row 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 [14] 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. 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. A cache may be organized into a handful of banks. An example of a cache's physical structure is shown in Figure 1(b).

2 CACTI Terminologies

The following is a list of keywords introduced by various versions of CACTI.

- **Bank** - A memory structure that consists of a data and a tag array. A cache may be split into multiple banks and CACTI assumes enough bandwidth so that these banks can be accessed simultaneously. The network topology that interconnects these banks can vary depending on the cache model (UCA or NUCA).

- **Sub-arrays** - A data or tag array is divided into a number of sub-arrays to reduce the delay due to wordline and bitline. Unlike banks, at any given time, these sub-arrays support only one single access. The total number of sub-arrays in a cache is equal to the product of Ndwl and Ndbl.

- **Mat** - A group of four sub-arrays (2x2) that share a common central predecoder. CACTI's exhaustive search starts from a minimum of at least one mat.
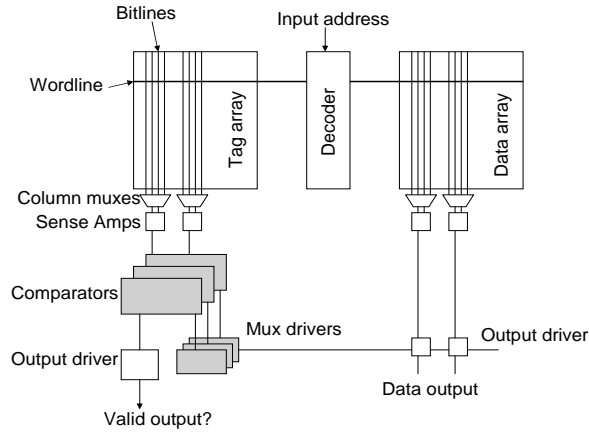
(a) Logical organization of a cache.
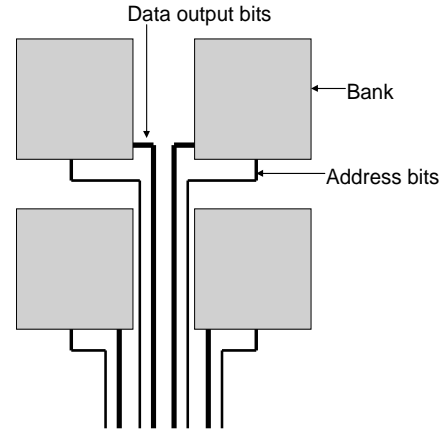
(b) Example physical organization of the data array.

Figure 1. Logical and physical organization of the cache (from CACTI 3.0 [13]).

- **Sub-bank** - In a typical cache, a cache block is scattered across multiple sub-arrays to improve the reliability of a cache. Irrespective of the cache organization, CACTI assumes that every cache block in a cache is distributed across an entire row of mats and the row number corresponding to a particular block is determined based on the block address. Each row (of mats) in an array is referred to as a sub-bank.

- **Ntwl/Ndwl** - Number of horizontal partitions in a tag or data array i.e., the number of segments that a single wordline is partitioned into.

- **Ntbl/Ndbl** - Number of vertical partitions in a tag or data array i.e., the number of segments that a single bitline is partitioned into.

- **Ntspd/Nspd** - Number of sets stored in each row of a sub-array. For a given Ndwl and Ndbl values, Nspd decides the aspect ratio of the sub-array.

- **Ntcm/Ndcm** - Degree of bitline multiplexing.

- **Ntsam/Ndsam** - Degree of sense-amplifier multiplexing.

3 New features in CACTI 6.0

CACTI 6.0 comes with a number of new features, most of which are targeted to improve the tool's ability to model large caches.

- Incorporation of many different wire models for the inter-bank network: local/intermediate/global wires, repeater sizing/spacing for optimal delay or power, low-swing differential wires.

- Incorporation of models for router components (buffers, crossbar, arbiter).

- Introduction of grid topologies for NUCA and a shared bus architecture for UCA with low-swing wires.
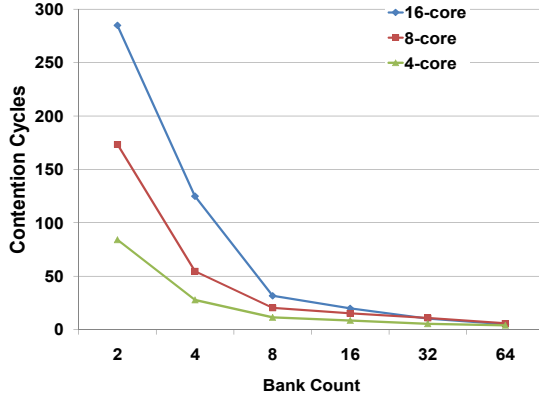
- An algorithm for design space exploration that models different grid layouts and estimates average bank and network latency. The design space exploration also considers different wire and router types.

- The introduction of empirical network contention models to estimate the impact of network configuration, bank cycle time, and workload on average cache access delay.

- An improved and more accurate wordline and bitline delay model.

- A validation analysis of all new circuit models: low-swing differential wires, distributed RC model for wordlines and bitlines within cache banks (router components have been validated elsewhere).

- An improved interface that enables trade-off analysis for latency, power, cycle time, and area.

4 NUCA Modeling

Earlier versions of CACTI assumed a Uniform Cache Access (UCA) model in which, the access time of a cache is determined by the delay to access the farthest sub-array. To enable pipelining, an H-tree network is employed to connect all the sub-arrays of a cache. For large caches, this uniform model can suffer from a very high hit latency. 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. [7] and has been the subject of many architectural evaluations. CACTI 6.0 builds upon this model and adopts the following algorithm to identify the optimal NUCA organization.

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 [15] 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 [4], and recently, speculative pipelines that take up three, two, and one pipeline stage have also been proposed [4, 8, 11]. 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. The user also has the flexibility to specify the operating frequency of the network (which defaults to 5 GHz). However, based on the process technology and the router model, the tool will calculate the maximum possible network frequency [11]. If the assumed frequency is greater than the maximum possible value, the tool will downgrade the network frequency to the maximum value.

In the above NUCA model, 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 [9]. While

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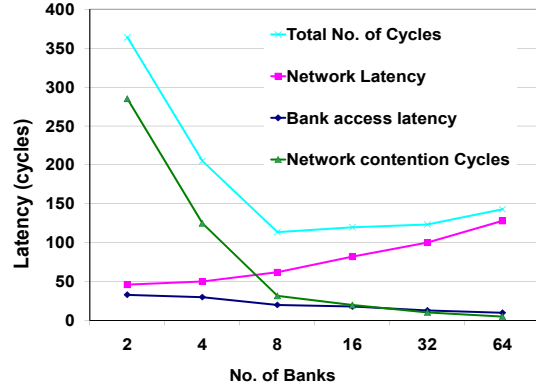
(b) Optimal NUCA organization

Figure 2. NUCA design space exploration.

the algorithm is guaranteed to find the cache structure with the lowest possible delay or power, the bandwidth of the cache might still not be sufficient enough for a multi core processor model. To address this problem, CACTI 6.0 further extends this algorithm by modeling contention in the network in much greater detail. This contention model 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.

To improve the search space of the NUCA model, CACTI 6.0 also explores different router types and wire types for the links between adjacent routers. The wires are modeled as low-swing differential wires as well as global wires with different repeater configurations to yield many points in the power/delay/area spectrum. 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.

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

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 1. Simplescalar simulator parameters.

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 2. Benchmark sets

suite. We divide the benchmark set into four categories, as described in Table 2. 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 1. 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.

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[1]. 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 many 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

---

[1]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.
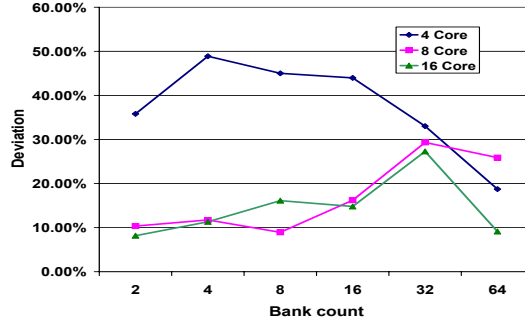
Figure 3. Contention deviation

transactional benchmarks with significant traffic from cache coherence.

Figure 2(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.

As mentioned earlier, contention values in Figure 2 correspond to the average values across different benchmark sets tabulated in Table 2. Depending upon the choice of benchmark set, the actual network contention can deviate from this mean value. Figure 3 shows the percentage deviation of contention values for different number of cores. Once again, the contention values are obtained by running 'n' different workloads from each benchmark set, where 'n' is equal to the number of cores in a CMP. For sixteen and eight core models, the deviation in contention values has negligible effect on optimal NUCA configuration. However, for some four core models, the optimal bank count can vary with the choice of benchmarks. Depending upon the bank cycle time, network contention typically accounts for around 25% of the NUCA access time. In the worst case, employing average network contention values for design space exploration can result in at most 10% error in NUCA access time.

4.1 Interconnect Model

With shrinking process technologies, interconnect plays an increasingly important role in deciding the power and performance of large structures. In the deep sub-micron era, the properties of a large cache are heavily impacted by the choice of the interconnect model [9, 10]. Another major enhancement to the tool that significantly improves the search space is the inclusion of different wire models with varying power and delay characteristics. The properties of wires depend on a number of factors like dimensions, signaling, operating voltage, operating
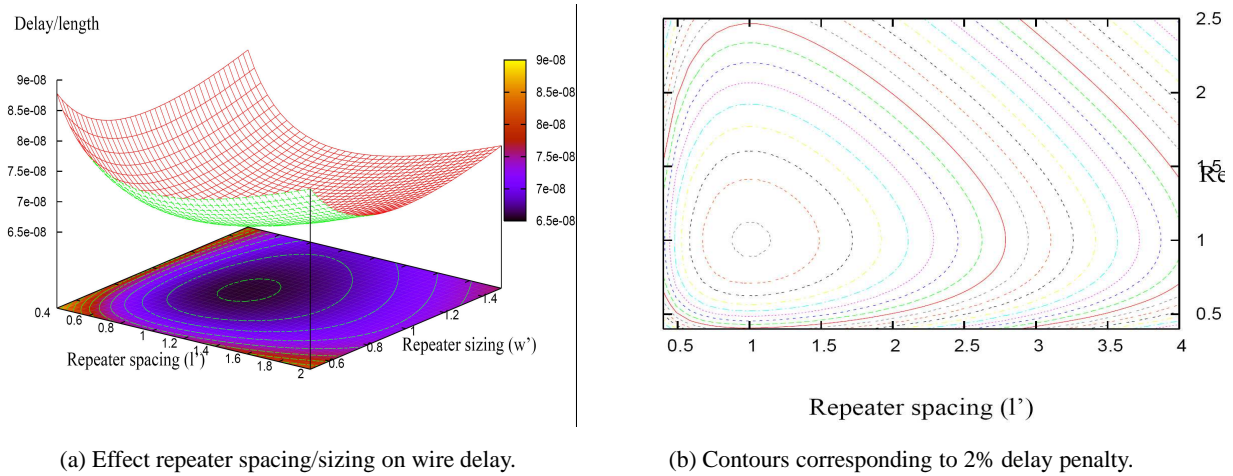
8

(a) Effect repeater spacing/sizing on wire delay.

(b) Contours corresponding to 2% delay penalty.

**Figure 4. Repeater overhead vs wire delay**

frequency, etc. Based on the signaling strategy, RC wires can be classified into two broad categories [2]: 1. Traditional full-swing wires, 2. Differential, low-swing, low power wires.

The delay of an RC wire increases quadratically with its length. To avoid this quadratic relationship, a long wire is typically interleaved with repeaters at regular intervals. This makes delay a linear function of wire length. However, the use of repeaters at regular intervals requires that voltage levels of these wires swing across the full range (0-Vdd) for proper operation. Given the quadratic dependence between voltage and power, these full swing wires are accompanied by very high power overhead. Figure 5 shows the delay and power values of global wires for different process technologies.

With power emerging as a major bottleneck, focusing singularly on performance is not possible. Alternatively, we can improve the power characteristics of these wires by incurring a delay penalty. In a typical, long, full swing wire, repeaters are one of the major contributors of interconnect power. Figure 4(a) shows the impact of repeater sizing and spacing on wire delay. Figure 4(b), shows the contours corresponding to the 2% delay penalty increments for different repeater configurations. As we can see, by tolerating a delay penalty, significant reduction in repeater overhead is possible. Figure 5 shows the power values of different wires that take 10%, 20%, and 30% delay penalty for different process technologies.

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. However, 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

---

[2]Many recent proposals advocate designing wires with very low resistance and/or high operating frequency so that wires behave like a transmission line. While transmission lines incur very low delay, they are accompanied by high area overheads and suffer from signal integrity issues. For these reasons, we limit our discussion in this report to RC wires.
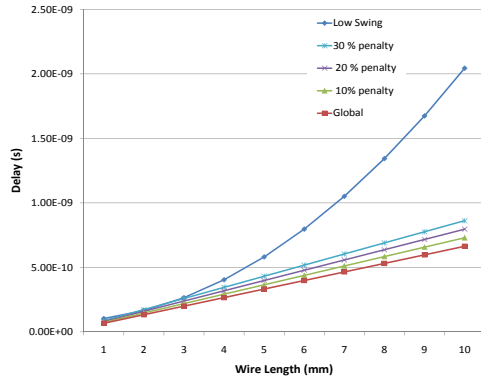
(a) Delay characteristics of different wires at 32nm process technology.
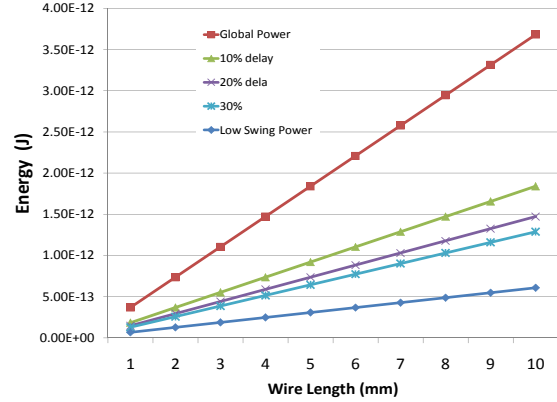
(b) Energy characteristics of different wires at 32nm process technology.

Figure 5. Power-delay properties of different wires

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 in a later section. In general, low-swing wires have superior power characteristics but incur high area and delay overheads. Figure 5 compares power delay characteristics of low-swing wires with global wires.

5   Analytical Models

The following sections discuss the analytical delay and power models for different wires. All the process specific parameters required for calculating the transistor and wire parasitics are obtained from ITRS [1].

5.1   Wire Parasitics

The resistance and capacitance per unit length of a wire is given by the following equations [5]:

$$R_{wire} = \frac{\rho}{d * (thickness \ - \ barrier)(width \ - \ 2\,barrier)} \tag{1}$$

where, $d\,(<1)$ is the loss in cross-sectional area due to dishing effect [1] and ρ is the resistivity of the metal.

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

In the above equation for the capacitance, the first term corresponds to the side wall capacitance, the second term models the capacitance due to wires in adjacent layers, and the last term corresponds to the fringing capacitance between the sidewall and the substrate.
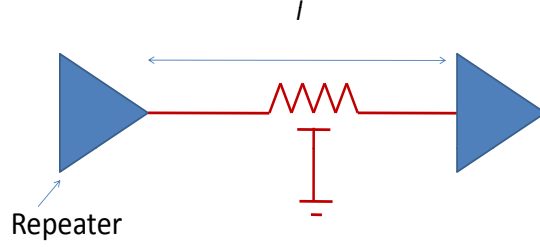
Figure 6. Interconnect segment

5.2 Global Wires

For a long repeated wire, the single pole time constant model for the interconnect fragment shown in Figure 6 is given by,

$$\tau = (\frac{1}{l}r_s(c_0 + c_p) + \frac{r_s}{s}C_{wire} + R_{wire}sc_0 + 0.5R_{wire}C_{wire}l) \tag{3}$$

In the above equation, c_0 is the capacitance of the minimum sized repeater, c_p is its output parasitic capacitance, r_s is its output resistance, l is the length of the interconnect segment between repeaters and s is the size of the repeater normalized to the minimum value. The values of c_0, c_p, and r_s are constant for a given process technology. Wire parasitics R_{wire} and C_{wire} represent resistance and capacitance per unit length. The optimal repeater sizing and spacing values can be calculated by differentiating equation 3 with respect to s and l and equating it to zero.

$$L_{optimal} = \sqrt{\frac{2r_s(c_0 + c_p)}{R_{wire}C_{wire}}} \tag{4}$$

$$S_{optimal} = \sqrt{\frac{r_sC_{wire}}{R_{wire}c_0}} \tag{5}$$

The delay value calculated using the above $L_{optimal}$ and $S_{optimal}$ is guaranteed to have minimum value. The total power dissipated is the sum of three main components (equation 6) [3].

$$P_{total} = P_{switching} + P_{short-circuit} + P_{leakage} \tag{6}$$

The dynamic and leakage components of the interconnect are computed using equations 7 and 8.

$$
\begin{aligned}
P_{dynamic} &= \alpha V_{DD}^2 f_{clock}(\frac{S_{optimal}}{L_{optimal}}(c_p + c_0) + c) \\
&+ (\alpha V_{DD}W_{min}I_{SC}f_{clock}\log_e 3)S_{optimal}\frac{\tau}{L_{optimal}}
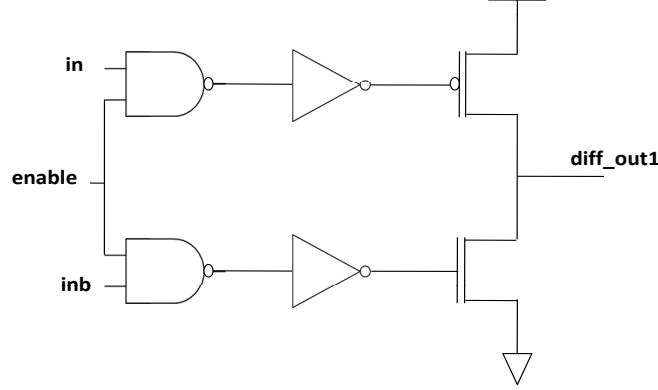\end{aligned}
$$

11

Figure 7. Low-swing transmitter (actual transmitter has two such circuits to feed the differential wires)

f_{clock} is the operating frequency, W_{min} is the minimum width of the transistor, I_{SC} is the short-circuit current, and the value $(\tau/L)_{optimal}$ can be calculated from equation 7.

$$(\frac{\tau}{L})_{optimal} = 2\sqrt{r_s c_0 rc} \left(1 + \sqrt{0.5 * \left(1 + \frac{c_p}{c_0} \right)} \right) \qquad (7)$$

$$P_{leakage} = \frac{3}{2} V_{DD} I_{leak} W_n S_{optimal} \qquad (8)$$

I_{leak} is the leakage current and W_n is the minimum width of the nMOS transistor.

With the above equations, we can compute the delay and power for global and semi-global wires. Wires faster than global wires can be obtained by increasing the wire width and spacing between the wires. Wires whose repeater spacing and sizing are different from equation 4 and 5 will incur a delay penalty. For a given delay penalty, the power optimal repeater size and spacing can be obtained from the contour shown in Figure 4(b). The actual calculation involves solving a set of differential equations [3].

5.3   Low-swing Wires

A low-swing interconnect system consists of three main components: (1) a transmitter that generates and drives the low-swing signal, (2) twisted differential wires, and (3) a receiver amplifier.

5.3.1   Transmitter

For an RC tree with a time constant τ, the delay of the circuit for an input with finite rise time is given by equation 9

$$delay_r = t_f \sqrt{[\log \frac{v_{th}}{V_{dd}}]^2 + 2t_{rise}b(1 - \frac{v_{th}}{Vdd})/t_f} \qquad (9)$$

where, t_f is the time constant of the tree, v_{th} is the threshold voltage of the transistor, t_{rise} is the rise time of the input signal, and b is the fraction of the input swing in which the output changes (we assume b to be 0.5).

For falling input, the equation changes to

$$delay_f = t_f \sqrt{[\log(1 - \frac{v_{th}}{V_{dd}})]^2 + \frac{2t_{fall}bv_{th}}{t_f V_{dd}}} \qquad (10)$$

where, t_{fall} is the fall time of the input. For the falling input, we use a value of 0.4 for b [18].

To get a reasonable estimate of the initial input signal rise/fall time, we consider two inverters connected in series. Let d be the delay of the second inverter. The t_{fall} and t_{rise} values for the initial input can be approximated to

$$t_{fall} = \frac{d}{1 - v_{th}}$$

$$t_{rise} = \frac{d}{v_{th}}$$

For the transmitter circuit shown in Figure 7, we employ the model proposed by Ho et al. [6].

The total delay of the transmitter is given by,

$$t_{delay} = nand_{delay} + inverter_{delay} + driver_{delay} \qquad (11)$$

Each gate in the above equation ($nand$, $inverter$, and $driver$) can be reduced to a simple RC tree. Later a Horowitz approximation is applied to calculate the delay of each gate. The power consumed in different gates can be derived from the input and output parasitics of the transistors.

NAND gate:

The equivalent resistance and capacitance values of a NAND gate is given by,

$$R_{eq} = 2 * R_{nmos}$$

$$C_{eq} = 2 * C_{Pdrain} + 1.5 * C_{Ndrain} + C_L$$

where C_L is the load capacitance of the NAND gate and is equal to the input capacitance of the next gate. The value of C_L is equal to $INV_{size} * (C_{Pgate} + C_{Ngate})$ where INV_{size} is the size of the inverter whose calculation is discussed later in this section.

NOTE: The drain capacitance of a transistor is highly non-linear. In the above equation for C_{eq}, the effective drain capacitance of two nMOS transistors connected in series is approximated to 1.5 times the drain capacitance of a single nMOS transistor.

$$\tau_{nand} = R_{eq} * C_{eq}$$

13

Using the τ_{nand} and t_{rise} values in equation 10, $nand_{delay}$ can be calculated. Power consumed by the NAND gate is given by

$$P_{nand} = C_{eq} * V_{dd}^2$$

The fall time (t_{fall}) of the input signal to the next stage (NOT gate) is given by

$$t_{fall} = nand_{delay}(\frac{1}{1 - v_{th}})$$

Driver:

To increase the energy savings in low-swing model, we assume a separate low voltage source for driving low-swing differential wires. The size of these drivers depends on its load capacitance which in turn depends on the length of the wire. To calculate the size of the driver, we first calculate the drive resistance of the nMOS transistors for a fixed desired rise time of eight F04.

$$R_{drive} = \frac{-Risetime}{C_L * ln(0.5)}$$

$$W_{dr} = \frac{R_m}{R_{drive}} * W_{min}$$

In the above equation, C_L is the sum of capacitance of the wire and input capacitance of the sense amplifier. R_m is the drive resistance of a minimum sized nMOS transistor and W_{min} is the width of the minimum sized transistor.

From the R_{drive} value, the actual width of the pMOS transistor can be calculated [3].

NOTE: The driver resistance R_{drive} calculated above is valid only if the supply voltage is set to full V_{dd}. Since low-swing drivers employ a separate low voltage source, the actual drive resistance of these transistors will be greater than the pMOS transistor of the same size driven by the full V_{dd}. Hence, the R_{drive} value is multiplied with an adjustment factor RES_ADJ to account for the poor driving capability of the pMOS transistor. Based on the SPICE simulation, RES_ADJ value is calculated to be 8.6.

NOT gate:

The size of the NOT gate is calculated by applying the method of logical effort. Consider the NAND gate connected to the NOT gate that drives a load of C_L, where, C_L is equal to the input capacitance of the driver. Let p_ef and s_ef represent path effort and stage effort respectively.

$$p\_ef = \frac{C_L}{C_{Ngate} + C_{Pgate}}$$

The delay will be minimum when the effort in each stage is same.

$$s\_ef = \sqrt{(4/3) * p\_ef}$$

$$C_{NOT\_in} = \frac{(4/3) * C_L}{s\_ef}$$

---

[3]In our model, we limit the transistor width to 100 times the minimum size.

$$INV_{size} = \frac{C_{NOT_in}}{C_{Ngate} + C_{Pgate}}$$

Using the above inverter size, the equivalent resistance and the capacitance of the gate can be calculated.

$$R_{eq} = R_{pmos}$$

$$C_{eq} = C_{Pdrain} + C_{Ndrain} + C_L$$

where C_L for the inverter is equal to (C_{Ngate}).

$$\tau_{not} = R_{eq} * C_{eq}$$

Using the above τ_{not} and t_{fall} values, not_{delay} can be calculated. Energy consumed by this NOT gate is given by

$$E_{not} = C_{eq} * V_{dd}^2$$

The rise time for the next stage is given by

$$t_{rise} = \frac{not_{delay}}{v_{th}}$$

5.3.2 Differential Wires

To alleviate the high delay overhead of the un-repeated low-swing wires, similar to differential bitlines, we employ pre-emphasis and pre-equalization optimizations. In pre-emphasis, the drive voltage of the driver is maintained at higher voltage than low-swing voltage. By overdriving these wires, it takes only a fraction of time constant to develop the differential voltage. In pre-equalization, after a bit traversal, the differential wires are shorted to recycle the charge. Developing a differential voltage on a pre-equalized wires takes less time compared to the wires with opposite polarity.

The following equations present the time constant and capacitance values of the segment that consist of low-swing drivers and wires.

$$t_{driver} = (R_{driver} * (C_{wire} + 2 * C_{drain}) + R_{wire}C_{wire}/2 + (R_{driver} + R_{wire}) * C_{sense\_amp}) \qquad (12)$$

The C_{wire} and R_{wire} in the above equation represents resistance and capacitance parasitics of the low-swing wire. R_{driver} and C_{drain} are resistance and drain capacitance of the driver transistors. The pre-equalization and pre-emphasis optimizations bring down this time constant to 35% of the above value.

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

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

The dynamic energy due to charging and discharging of differential wires is given by,

$$C_{load} * V_{overDrive} * V_{lowswing}$$

For our evaluations we assume an overdrive voltage of 400mV and a low swing voltage of 100mV.
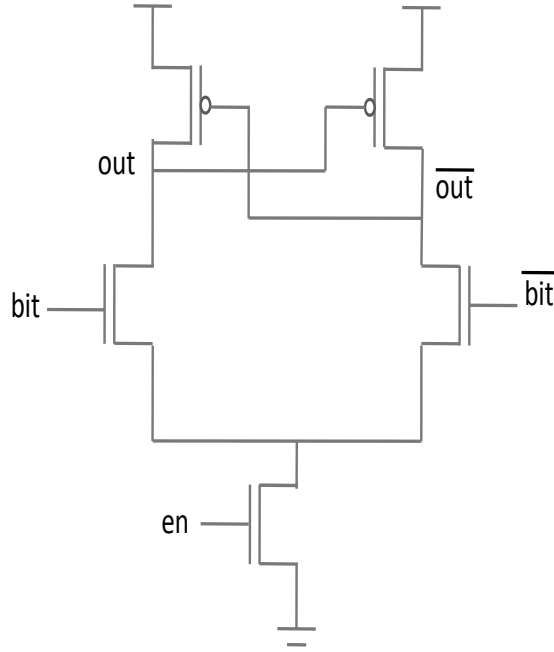
Figure 8. Sense-amplifier circuit

5.3.3 Sense Amplifier

Figure 8 shows the cross-coupled inverter sense amplifier circuit used at the receiver. The delay and power values of the sense amplifier were directly calculated from SPICE simulation. The simulation methodology and the actual delay and power values of the sense-amplifier for different process technologies are discussed in the validation section 8.

5.4 Router Models

There have been a number of router proposals in the literature with different levels of speculation and pipeline stages [4, 8, 11]. The number of pipeline stages for routers in CACTI 6.0 is left as a user-specified input, defaulting to 3 cycles. Buffers, crossbars, and arbiters are the major contributors to the router power. CACTI 6.0's analytical power models for crossbars and arbiters is similar to the model employed in Orion toolkit [17]. Buffer power is modeled using CACTI's inbuilt RAM model.
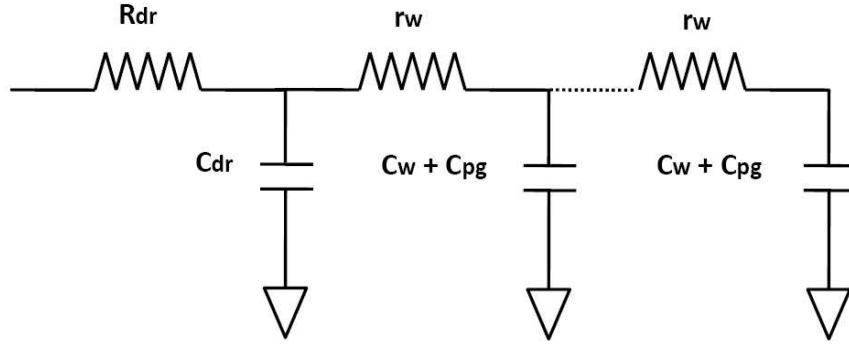
Figure 9. RC model of a wordline

5.5 Distributed Wordline Model

Figure 9 shows the wordline circuit and its equivalent RC model. Earlier versions of CACTI modeled the wordline wire as a single lumped RC tree. In process technologies where wire parasitics dominate, a distributed RC model of the type shown in the figure will significantly improve the accuracy of the model.

Let c_w and r_w be the resistance and capacitance values of the wire of length l where, l is the width of the memory cell. The time constant governing the above RC tree is given by

$$\tau = R_{dr} * C_{dr} + n * R_{dr} * (c_w + C_{pg}) + \frac{r_w * (c_w + C_{pg}) * n * (n+1)}{2}$$

where,

R_{dr} - Resistance of the pMOS transistor in the driver.

C_{dr} - Sum of the drain capacitance of the pMOS and nMOS transistors in the driver.

C_{pg} - Input gate capacitance of the pass transistor.

n - Length of the wordline in terms of number of memory cells.

5.6 Distributed Bitline Model

Figure 10 shows the RC model of the bitline read path. The time constant of the RC tree is given by,

$$\begin{aligned}
\tau = &(R_{pass} + R_{pd}) * C_{pass} + \\
&(R_{pd} + R_{pass} + r * n + R_{bmux}) * C_{bmux} + \\
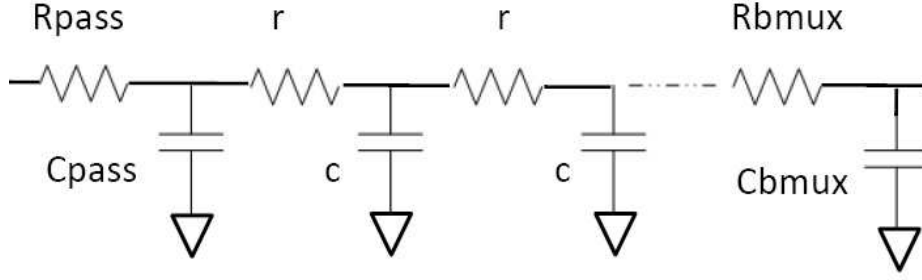&(R_{pd} + R_{pass}) * c * n + n * (n+1) * r * c/2
\end{aligned}$$

17

Figure 10. RC model of a bitline

where,

R_{pass} - Resistance of the pass transistor

C_{pass} - Drain capacitance of the pass transistor

R_{bmux} - Resistance of the transistor in the bitline multiplexer

C_{bmux} - Drain capacitance of the transistor in the bitline multiplexer

n - Length of the bitline in terms of number of memory cells

c - Capacitance of the bitline segment between two memory cells that include wire capacitance and the drain capacitance of the pass transistor

r - Resistance of the wire connecting two pass transistors

We follow a methodology similar to the one proposed in the original version of CACTI [18] to take into account the effect of finite rise time of wordline signal on the bitline delay.

6 Extensions to UCA Model

A traditional UCA model employs an H-tree network for address and data communication for the following reason: it enables uniform access times for each bank, which in turn, simplifies the pipelining of requests across the network. In addition to supporting traditional full-swing, repeated wires, CACTI 6.0 also has an option to employ low-swing wires for address and data transfers in UCA caches. 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 multiple banks. Each bus is shared by half the banks in a column - an example with eight banks is shown in Figure 11. The banks continue to have uniform access times, as determined by the worst-case delay.

7 Trade-off Analysis

The new version of the tool adopts the following default cost function to evaluate a cache organization (taking into account delay, leakage power, dynamic power, cycle time, and area):
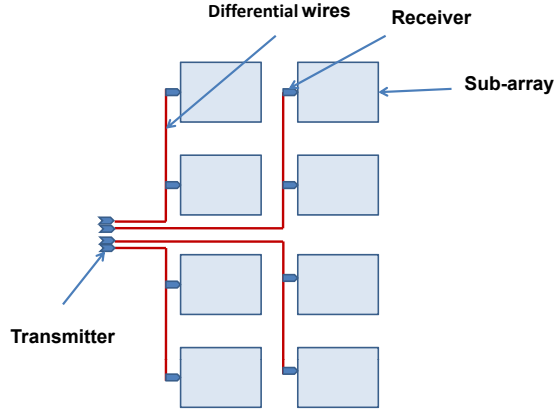
Figure 11. 8-bank data array with a differential low-swing broadcast bus.

$$cost = W_{acc\_time}\frac{acc\_time}{min\_acc\_time} + W_{dyn\_power}\frac{dyn\_power}{min\_dyn\_power} + W_{leak\_power}\frac{leak\_power}{min\_leak\_power} +$$

$$W_{cycle\_time}\frac{cycle\_time}{min\_cycle\_time} + W_{area}\frac{area}{min\_area}$$

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
```

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
```

CACTI 6.0 also takes an ED or ED^2 value as input to its cost function to determine a cache organization that has the best energy-delay or energy-delay square product.

Technology	Delay (ps)	Energy (fJ)
90nm	279	14.7
68nm	200	5.7
45nm	38	2.7
32nm	30	2.16

Table 3. Sense-amplifier delay and energy values for different process technologies.

8   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 [15] 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 [18].

Figure 12 (a) & (b) show the comparison of delay and power values of the differential, low-swing analytical models against SPICE values. As mentioned earlier, a low-swing wire model can be broken into three components: transmitters (that generate the low-swing signal), differential wires[5], and sense amplifiers. The modeling details of each of these components are discussed in section 5.3. 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

---

[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.

[5]Delay and power values of low-swing drivers are also reported as part of differential wires.
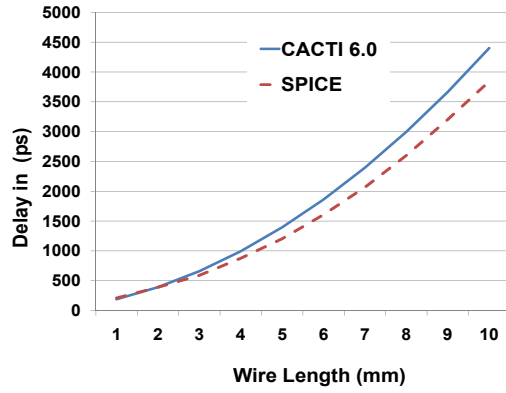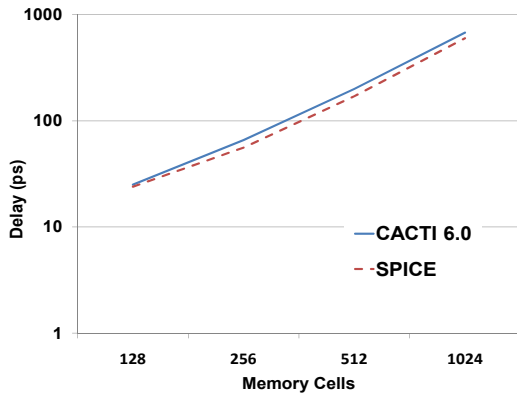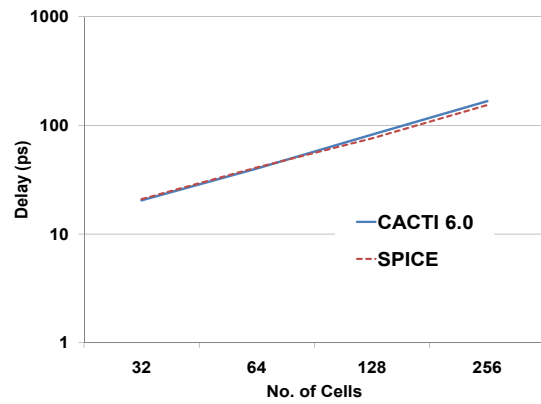
(a) Delay verification
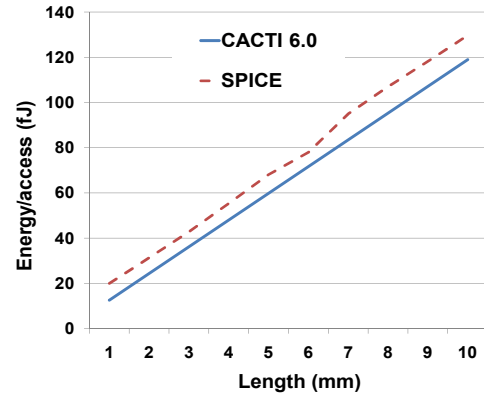
(b) Energy verification

Figure 12. Low-swing model verification

(a) Wordline

(b) Bitline

Figure 13. Distributed wordline and bitline model verification

21

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 SPICE generated values for sense-amp power and delay. Table 3 shows the delay and power values of the sense-amplifier for different process technologies. To calculate these values, the sense amplifier load was set to twice the input capacitance of the minimum sized inverter. 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 [12]. Figure 13 (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. [16].

9 Usage

Prior versions of CACTI take cache parameters such as cache size, block size, associativity, and technology as command line arguments. In addition to supporting the command line input, CACTI 6.0 also employs a configuration file (cache.cfg) to enable user to describe the cache parameters in much greater detail. The following are the valid command line arguments in CACTI 6.0:

```
C B A Tech NoBanks
  and / or
-weight <delay> <dynamic> <leakage> <cycle> <area>
  and / or
-deviate <delay> <dynamic> <leakage> <cycle> <area>


C - Cache size in bytes
B - Block size in bytes
A - Associativity
Tech - Process technology in microns or nano-meter
NoBanks - No. of UCA banks
```

Command line arguments are optional in CACTI 6.0 and a more comprehensive description is possible using the configuration file. Other non-standard parameters that can be specified in the cache.cfg file include,

- No. of read ports, write ports, read-write ports in a cache

- H-tree bus width

- Operating temperature (which is used for calculating the cache leakage value),

- Custom tag size (that can be used to model special structures like branch target buffer, cache directory, etc.)

- Cache access mode (fast - low access time but power hungry; sequential - high access time but low power; Normal - less aggressive in terms of both power and delay)

- Cache type (DRAM, SRAM or a simple scratch RAM such as register files that does not need the tag array)

- NUCA bank count (By default CACTI calculates the optimal bank count value. However, the user can force the tool to use a particular NUCA bank count value)

- Number of cores

- Cache level - L2 or L3 (Core count and cache level are used to calculate the contention values for a NUCA model)

- Design objective (weight and deviate parameters for NUCA and UCA)

More details on each of these parameters is provided in the default cache.cfg file that is provided with the distribution.

10   Conclusions

This report details major revisions to the CACTI cache modeling tool along with a detailed description of the analytical model for newly added components. Interconnect plays a major role in deciding the delay and power values 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 contention modeling. 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.

References

[1] Semiconductor Industry Association, International Technology Roadmap for Semiconductors 2005. http://public.itrs.net/Links/2005ITRS/Home2005.htm.

[2] Arizona State University. Predictive technology model.

[3] 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.

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

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

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

[7] 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.

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

[9] 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.

[10] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi. Optimizing NUCA Organizations and Wiring Alternatives for Large Caches With CACTI 6.0. In *Proceedings of MICRO-40*, 2007.

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

[12] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits - A Design Perspective*. Prentice-Hall, 2nd edition, 2002.

[13] 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.

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

[15] S. Thoziyoor, N. Muralimanohar, and N. Jouppi. CACTI 5.0. Technical Report HPL-2007-167, HP Laboratories, 2007.

[16] H.-S. Wang, L.-S. Peh, and S. Malik. Power-Driven Design of Router Microarchitectures in On-Chip Networks. In *Proceedings of MICRO-36*, December 2003.

[17] 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.

[18] S. Wilton and N. Jouppi. An Enhanced Access and Cycle Time Model for On-Chip Caches. Technical Report TN-93/5, Compaq Western Research Lab, 1993.