CACTI 5.0

Shyamkumar Thoziyoor, Naveen Muralimanohar, and Norman P. Jouppi
Advanced Architecture Laboratory
HP Laboratories

HPL-2007-167
October 19, 2007*

cache, memory, area, power, access time

CACTI 5.0 is the latest major revision of the CACTI tool for modeling the dynamic power, access time, area, and leakage power of caches and other memories. CACTI 5.0 includes a number of major improvements over CACTI 4.0. First, as fabrication technologies enter the deep-submicron era, device and process parameter scaling has become non-linear. To better model this, the base technology modeling in CACTI 5.0 has been changed from simple linear scaling of the original CACTI 0.8 micron technology to models based on the ITRS roadmap. Second, embedded DRAM technology has become available from some vendors, and there is interest in 3D stacking of commodity DRAM with modern chip multiprocessors. As another major enhancement, CACTI 5.0 adds modeling support of DRAM memories. Third, to support the significant technology modeling changes above and to enable fair comparisons of SRAM and DRAM technology, the CACTI code base has been extensively rewritten to become more modular. At the same time, various circuit assumptions have been updated to be more relevant to modern design practice. Finally, numerous bug fixes and small feature additions have been made. For example, the cache organization assumed by CACTI is now output graphically to assist users in understanding the output generated by CACTI.

CACTI 5.0

Shyamkumar Thoziyoor, Naveen Muralimanohar, and Norman P. Jouppi
sthoziyo@cse.nd.edu, naveen@cs.utah.edu, norm.jouppi@hp.com

October 19, 2007

Abstract

CACTI 5.0 is the latest major revision of the CACTI tool for modeling the dynamic power, access time, area, and leakage power of caches and other memories. CACTI 5.0 includes a number of major improvements over CACTI 4.0. First, as fabrication technologies enter the deep-submicron era, device and process parameter scaling has become non-linear. To better model this, the base technology modeling in CACTI 5.0 has been changed from simple linear scaling of the original CACTI 0.8 micron technology to models based on the ITRS roadmap. Second, embedded DRAM technology has become available from some vendors, and there is interest in 3D stacking of commodity DRAM with modern chip multiprocessors. As another major enhancement, CACTI 5.0 adds modeling support of DRAM memories. Third, to support the significant technology modeling changes above and to enable fair comparisons of SRAM and DRAM technology, the CACTI code base has been extensively rewritten to become more modular. At the same time, various circuit assumptions have been updated to be more relevant to modern design practice. Finally, numerous bug fixes and small feature additions have been made. For example, the cache organization assumed by CACTI is now output graphically to assist users in understanding the output generated by CACTI.

Contents

1 Introduction **4**

2 Changes and Enhancements in Version 5.0 **4**

2.1 Organizational Changes . . . 4

2.2 Circuit and Sizing Changes . . . 4

2.3 Technology Changes . . . 5

2.4 DRAM Modeling . . . 6

2.5 Miscellaneous Changes . . . 7

2.5.1 Optimization Function Change . . . 7

2.5.2 New Gate Area Model . . . 7

2.5.3 Wire Model . . . 7

2.5.4 ECC and Redundancy . . . 7

2.5.5 Display Changes . . . 8

3 Data Array Organization **8**

3.1 Mat Organization . . . 8

3.2 Routing to Mats . . . 11

3.3 Organizational Parameters of a Data Array . . . 13

3.4 Comments about Organization of Data Array . . . 14

4 Circuit Models and Sizing **14**

4.1 Wire Modeling . . . 15

4.2 Sizing Philosophy . . . 16

4.3 Sizing of Mat Circuits . . . 16

4.3.1 Predecoder and Decoder . . . 16

4.3.2 Bitline Peripheral Circuitry . . . 18

4.4 Sense Amplifier Circuit Model . . . 21

4.5 Routing Networks . . . 23

4.5.1 Array Edge to Bank Edge H-tree . . . 23

4.5.2 Bank Edge to Mat H-tree . . . 23

5 Area Modeling **24**

5.1 Gate Area Model . . . 25

5.2 Area Model Equations . . . 26

6 Delay Modeling **30**

6.1 Access Time Equations . . . 30

6.2 Random Cycle Time Equations . . . 31

7 Power Modeling **32**

7.1 Calculation of Dynamic Energy . . . 32

7.1.1 Dynamic Energy Calculation Example for a CMOS Gate Stage . . . 32

7.1.2 Dynamic Energy Equations . . . 33

7.2 Calculation of Leakage Power . . . 34

7.2.1 Leakage Power Calculation for CMOS gates . . . 34

7.2.2 Leakage Power Equations . . . 35

8 Technology Modeling **37**

8.1 Devices . . . 37

8.2 Wires . . . 39

8.3 Technology Exploration . . . 39

9 Embedded DRAM Modeling **40**
9.1 Embedded DRAM Modeling Philosophy 40
9.1.1 Cell 40
9.1.2 Destructive Readout and Writeback 40
9.1.3 Sense amplifier Input Signal 40
9.1.4 Refresh 41
9.1.5 Wordline Boosting 41
9.2 DRAM Array Organization and Layout 41
9.2.1 Bitline Multiplexing 42
9.2.2 Reference Cells for VDD Precharge 42
9.3 DRAM Timing Model 42
9.3.1 Bitline Model 42
9.3.2 Multisubbank Interleave Cycle Time 43
9.3.3 Retention Time and Refresh Period 44
9.4 DRAM Power Model 44
9.4.1 Refresh Power 45
9.5 DRAM Area Model 45
9.5.1 Area of Reference Cells 45
9.5.2 Area of Refresh Circuitry 45
9.6 DRAM Technology Modeling 45
9.6.1 Cell Characteristics 45

10 Cache Modeling **47**
10.1 Organization 47
10.2 Delay Model 48
10.3 Area Model 49
10.4 Power Model 49

11 Quantitative Evaluation **49**
11.1 Evaluation of New CACTI 5.0 Features 49
11.1.1 Impact of New CACTI Solution Optimization 49
11.1.2 Impact of Device Technology 50
11.1.3 Impact of Interconnect Technology 51
11.1.4 Impact of RAM Cell Technology 51
11.2 Version 4.2 vs Version 5.0 Comparisons 52

12 Validation **64**
12.1 Sun SPARC 90 nm L2 cache 64
12.2 Intel Xeon 65 nm L3 cache 64

13 Future Work **71**

14 Conclusions **71**

A Additional CACTI Validation Results for 90 nm SPARC L2 **72**

B Additional CACTI Validation Results for 65 nm Xeon L3 **77**

1 Introduction

CACTI 5.0 is the latest major revision of the CACTI tool [1][2][3][4] for modeling the dynamic power, access time, area, and leakage power of caches and other memories. CACTI has become widely used by computer architects, both directly and indirectly through other tools such as Wattch.

CACTI 5.0 includes a number of major improvements over CACTI 4.0. First, as fabrication technogies enter the deep-submicron era, device and process parameter scaling has become non-linear. To better model this, the base technology modeling in CACTI 5.0 has been changed from simple linear scaling of the original 0.8 micron technology to models based on the ITRS roadmap. Second, embedded DRAM technology has become available from some vendors, and there is interest in 3D stacking of commodity DRAM with modern chip multiprocessors. As another major enhancement, CACTI 5.0 adds modeling support of DRAM memories. Third, to support the significant technology modeling changes above and to enable fair comparisons of SRAM and DRAM technology, the CACTI code base has been extensively rewritten to become more modular. At the same time, various circuit assumptions have been updated to be more relevant to modern design practice. Finally, numerous bug fixes and small feature improvements have been made. For example, the cache organization assumed by CACTI is now output graphically by the web-based server, to assist users in understanding the output generated by CACTI.

The following section gives an overview of these changes, after which they are discussed in detail in subsequent sections.

2 Changes and Enhancements in Version 5.0

2.1 Organizational Changes

Earlier versions of CACTI (up to version 3.2) made use of a single row predecoder at the center of a memory bank with the row predecoded signals being driven to the subarrays for decoding. In version 4.0, this centralized decoding logic was implicitly replaced with distributed decoding logic. Using H-tree distribution, the address bits were transmitted to the distributed sinks where the decoding took place. However, because of some inconsistencies in the modeling, it was not clear at what granularity the distributed decoding took place - whether there was one sink per subarray or 2 or 4 subarrays. There were some other problems with the CACTI code such as the following:

- The area model was not updated after version 3.2, so the impact on area of moving from centralized to distributed decoding was not captured. Also, the leakage model did not account for the multiple distributed sinks. The impact of cache access type (normal/serial/fast) [4] on area was also not captured;
- Number of address bits routed to the subarrays was being computed incorrectly;
- Gate load seen by NAND gate in the 3-8 decode block was being computed incorrectly; and
- There were problems with the logic computing the degree of muxing at the tristate subarray output drivers.

In version 5.0, we resolve these issues, redefine and clarify what the organizational assumptions of memory are and remove ambiguity from the modeling. Details about the organization of memory can be found in Section 3.

2.2 Circuit and Sizing Changes

Earlier versions of CACTI made use of row decoding logic with two stages - the first stage was composed of 3-8 predecode blocks (composed of NAND3 gates) followed by a NOR decode gate and wordline driver. The number of gates in the row decoding path was kept fixed and the gates were then sized using the method of logical effort for an effective fanout of 3 per stage. In version 5.0, in addition to the row decoding logic, we also model the bitline mux decoding logic and the sense-amplifier mux decoding logic. We use the same circuit structures to model all decoding logic and we base the modeling on the effort described in [5]. We use the sizing heuristic described in [5] that has been shown to be good from an energy-delay perspective. With the new circuit structures and modeling that we use, the limit on maximum number of signals that can be decoded is increased from 4096 (in version 4.2) to 262144 (in version 5.0). While we do not expect the number of signals that are decoded to be very high, extending the limit from 4096 helps with exploring area/delay/power tradeoffs in a more thorough manner for large memories, especially for large DRAMs. Details of the modeling of decoding logic are described in Section 4.

There are certain problems with the modeling of the H-tree distribution network in version 4.2. An inverter-driver is placed at branches of the address, datain and dataout H-tree. However, the dataout H-tree does not model tristate drivers. The output data bits may come from a few subarrays and so the address needs to be distributed to a few subarrays, however, dynamic power spent in transmitting address is computed as if all the data comes from a single subarray. The leakage in the drivers of the datain H-tree is not modeled.

In verson 5.0, we model the H-tree distribution network more rigorously. For the dataout H-tree we model tristate buffers at each branch. For the address and datain H-trees, instead of assuming inverters at the branches of the H-tree we assume the use of buffers that may be gated to allow or disallow the passage of signals and thereby control the dynamic power. We size these drivers based on the methodology described in [5] which takes the resistance and capacitance of intermediate wires into account during sizing. We also model the use of repeaters in the H-tree distribution network which are sized according to equations from [6].

2.3 Technology Changes

Earlier versions of CACTI relied on a complicated way of obtaining device data for the input technology-node. Computation of access/cycle time and dynamic power were based off device data of a 0.8-micron process that was scaled to the given technology-node using simple linear scaling principles. Leakage power calculation, however, made use of Ioff (subthreshold leakage current) values that were based off device data obtained through BSIM3 parameter extractions. In version 4.2, BSIM3 extraction was carried out for a few select technology nodes (130/100/70 nm); as a result leakage power estimation was available only for these select technology nodes.

There are several problems with the above approach of obtaining device data. Using two sets of parameters, one for computation of access/cycle time/dynamic power and another for leakage power, is a convoluted approach and is hard to maintain. Also, the approach of basing device parameter values off a 0.8-micron process is not a good one because of several reasons. Device scaling has become quite non-linear in the deep-submicron era. Device performance targets can no longer be achieved through simple linear scaling of device parameters. Moreover, it is well-known that physical gate-lengths (according to the ITRS, physical gate-length is the final, as-etched length of the bottom of the gate electrode) have scaled much more aggressively [7][8] than what would be projected by simple linear scaling from the 0.8 micron process.

In version 5.0, we adopt a simpler, more evolvable approach of obtaining device data. We use device data that the ITRS [7] uses to make its projections. The ITRS makes use of the MASTAR software tool (Model for Assessment of CMOS Technologies and Roadmaps) [9] for computation of device characteristics of current and future technology nodes. Using MASTAR, device parameters may be obtained for different technologies such as planar bulk, double gate and Silicon-On-Insulator. MASTAR includes device profile and result files of each year/technology-node for which the ITRS makes projections and we incorporate the data from these files into CACTI. These device profiles are based off published industry process data and industry-consensus targets set by historical trends and system drivers. While it is not necessary that these device numbers match or would match process numbers of various vendors in an exact manner, they do come within the same ball-park as can be seen by looking at the Ion-Ioff cloud graphic within the MASTAR software which shows a scatter plot of various published vendor Ion-Ioff numbers and corresponding ITRS projections. With this approach of using device data from the ITRS, it also becomes possible to incorporate device data corresponding to different device types that the ITRS defines such as high performance (HP), LSTP (Low Standby Power) and Low Operating Power (LOP). More details about the device data used in CACTI can be found in Section 8.

There are some problems with interconnect modeling of version 4.2 also. Version 4.2 utilizes 2 types of wires in the delay model, ‘local’ and ‘global’. The local type is used for wordlines and bitlines, while the global type is used for all other wires. The resistance per unit length and capacitance per unit length for these two wire types are also calculated in a convoluted manner. For a given technology, the resistance per unit length of the local wire is calculated by assuming ideal scaling in all dimensions and using base data of a 0.8-micron process. The base resistance per unit length for the 0.8-micron process is itself calculated by assuming copper wires in the base 0.8-micron process and readjusting the sheet resistance value of version 3.2 which assumed aluminium wires. As the resistivity of copper is about 2/3rd that of aluminium, the sheet resistance of copper was computed to be 2/3rd that of aluminium. However, this implies that the thickness of metal assumed in versions 3.2 and 4.2 are the same which turns out to be not true. When we compute sheet resistance for the 0.8-micron process with the thickness of local wire assumed in version 4.2 and assuming a resistivity of 2.2 μohm-cm for copper, the value comes out to be a factor of 3.4 smaller than that used in version 3.2. In version 4.2, resistance per unit length for the global wire type is calculated to be smaller than that of local wire type by a factor of 2.04. This factor of 2.04 is calculated based on RC delays and wire sizes of different wire types in the 2004 ITRS but the underlying assumptions are not known. Another problem is that even though the delay model makes use of two types of wires, local and global, the area model makes use of just the local wire type and the pitch calculation of all wires (local type and global type) are based off the assumed width and spacing for the local wire type; this results in an underestimation of pitch (and area) occupied by the global wires

Capacitance per unit length calculation of version 4.2 also suffers from certain problems. The capacitance per unit length values for local and global wire types are assumed to remain constant across technology nodes. The capacitance per unit length value for local wire type was calculated for a 65 nm process as (2.9/3.6)*230 = 185 fF/m where 230 is the published capacitance per unit length value for an Intel 130 nm process [10], 3.6 is the dielectric constant of the 130 nm process and 2.9 is the dielectric constant of an Intel 65 nm process [8]. Computing the value of capacitance per unit length in this manner for a 65 nm process ignores the fact that the fringing component of capacitance remains almost constant across technology-nodes and scales very slowly [6][11]. Also, assuming that the dielectric constant remains fixed at 2.9 for future technology nodes ignores the possibility of use of lower-k dielectrics. Capacitance per unit length of the global type wire of version 4.2 is calculated to be smaller than that of local type wires by a factor of 1.4. This factor of 1.4 is again calculated based on RC delays and wire sizes of different wire types in the 2004 ITRS but the underlying assumptions again are not known.

In version 5.0, we remove the ambiguity from the interconnect modeling. We use the interconnect projections made in [6][12] which are based off well-documented simple models of resistance and capacitance. Because of the difficulty in projecting the values of interconnect properties in an exact manner at future technology nodes the approach employed in [6][12] was to come up with two sets of projections based on aggressive and conservative assumptions. The aggressive projections assume aggressive use of low-k dielectrics, insignificant resistance degradation due to dishing and scattering, and tall wire aspect ratios. The conservative projections assume limited use of low-k dielectrics, significant resistance degradation due to dishing and scattering, and smaller wire aspect ratios. We incorporate both sets of projections into CACTI. We also model 2 types of wires inside CACTI - semi-global and global with properties identical to that described in [6][12]. More details of the interconnect modeling are described in Section 8.2. Comparison of area, delay and power of caches obtained using versions 4.2 and 5.0 are presented in Section 11.2.

2.4 DRAM Modeling

One of the major enhancements of version 5.0 is the incorporation of embedded DRAM models for a logic-based embedded DRAM fabrication process [13][14][15]. In the last few years, embedded DRAM has made its way into various applications. The IBM POWER4 made use of embedded DRAM in its L3 cache [16]. The main compute chip inside the Blue Gene/L supercomputer also makes use of embedded DRAM [17]. Embedded DRAM has also been used in the CPU used within Sony's Playstation 2 [18].

In our modeling of embedded DRAM, we leverage the similarity that exists in the global and peripheral circuitry of embedded SRAM and DRAM and model only their essential differences. We use the same array organization for embedded DRAM that we used for SRAM. By having a common framework that, in general, places embedded SRAM and DRAM on an equal footing and emphasizes only their essential differences, we are able to compare relative tradeoffs between embedded SRAM and DRAM. We describe the modeling of embedded DRAM in Section 9.

2.5 Miscellaneous Changes

2.5.1 Optimization Function Change

In version 5.0, we follow a different approach in finding the optimal solution with CACTI. Our new approach allows users to exercise more control on area, delay and power of the final solution. The optimization is carried out in the following steps: first, we find all solutions with area that is within a certain percentage (user-supplied value) of the area of the solution with best area efficiency. We refer to this area constraint as 'maxareaconstraint'. Next, from this reduced set of solutions that satisfy the maxareaconstraint, we find all solutions with access time that is within a certain percentage of the best access time solution (in the reduced set). We refer to this access time constraint as 'maxacctimeconstraint'. To the subset of solutions that results after the application of maxacctimeconstraint, we apply the following optimization function:

$$
\begin{aligned}
\text{optimization-func} \quad = \quad & \frac{\text{dynamic-energy}}{\text{min-dynamic-energy}}\text{flag-opt-for-dynamic-energy} + \\
& \frac{\text{dynamic-power}}{\text{min-dynamic-power}}\text{flag-opt-for-dynamic-power} + \\
& \frac{\text{leak-power}}{\text{min-leak-power}}\text{flag-opt-for-leak-power} + \\
& \frac{\text{rand-cycle-time}}{\text{min-rand-cycle-time}}\text{flag-opt-for-rand-cycle-time}
\end{aligned}
$$

where dynamic-energy, dynamic-power, leak-power and rand-cycle-time are the dynamic energy, dynamic power, leakage power and random cycle time of a solution respectively and min-dynamic-energy, min-dynamic-power, min-leak-power and min-rand-cycle-time are their minimum (best) values in the subset of solutions being considered. flag-opt-for-dynamic-energy, flag-opt-for-dynamic-power, flag-opt-for-leak-power and flag-opt-for-rand-cycle-time are user-specified boolean variables. The new optimization process allows exploration of the solution space in a controlled manner to arrive at a solution with user-desired characteristics.

2.5.2 New Gate Area Model

In version 5.0, we introduce a new analytical gate area model from [19]. With the new gate area model it becomes possible to make the areas of gates sensitive to transistor sizing so that when transistor sizing changes, the areas also change. With the new gate area model, transistors may get folded when they are subject to pitch-matching constraints and the area is calculated accordingly. This feature is useful in capturing differences in area caused due to different pitch-matching constraints that may have to be satisfied, particularly between SRAM and DRAM.

2.5.3 Wire Model

Version 4.2 models wires using the equivalent circuit model shown in Figure 1. The Elmore delay of this model is RC/2, however this model underestimates the wire-to-gate component (RwireCgate) of delay. In version 5.0, we replace this model with the Pi RC model, shown in Figure 2, which has been used in more recent SRAM modeling efforts [20].

2.5.4 ECC and Redundancy

In order to be able to check and correct soft errors, most memories of today have support for ECC (Error Correction Code). In version 5.0, we capture the impact of ECC by incorporating a model that captures the ECC overhead in memory cell and data bus (datain and dataout) area. We incorporate a variable that specifies the number of data bits per ECC bit. By default, we fix the value of this variable to 8.

In order to improve yield, many memories of today incorporate redundant entities even at the subarray level. For example, the data array of the 16 MB Intel Xeon L3 cache [21] which has 256 subarrays also incorporates 32 redundant subarrays. In version 5.0, we incorporate a variable that specifies the number of mats per redundant mat. By default, we fix the value of this variable to 8.

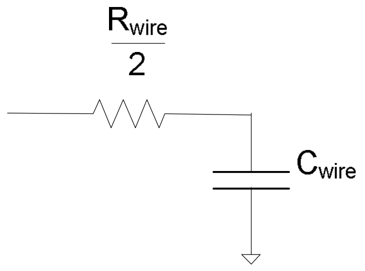


Figure 1: L-model of wire used in version 4.2.

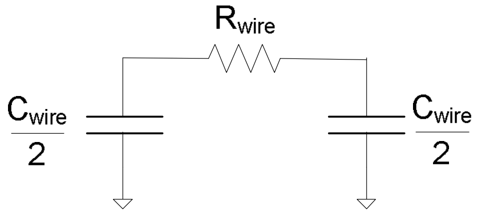


Figure 2: Pi RC model of wire used in version 5.0.

2.5.5 Display Changes

To facilitate better understanding of cache organization, version 5.0 can output data/tag array organization graphically. Figure 3 shows an example of the graphical display generated by version 5.0. The top part of the figure shows a generic mat organization assumed by CACTI. It is followed by the data and tag array organization plotted based on array dimensions calculated by CACTI.

3 Data Array Organization

At the highest level, a data array is composed of multiple identical banks (N_{banks}). Each bank can be concurrently accessed and has its own address and data bus. Each bank is composed of multiple identical subbanks (N_{subbanks}) with one subbank being activated per access. Each subbank is composed of multiple identical mats ($N_{\text{mats-in-subbank}}$). All mats in a subbank are activated during an access with each mat holding part of the accessed word in the bank. Each mat itself is a self-contained memory structure composed of 4 identical subarrays and associated predecoding logic. Each subarray is a 2D matrix of memory cells and associated peripheral circuitry. Figure 4 shows the layout of an array with 4 banks. In this example each bank is shown to have 4 subbanks and each subbbank is shown to have 4 mats. Not shown in Figure 4, address and data are assumed to be distributed to the mats on H-tree distribution networks.

The rest of this section further describes details of the array organization assumed in CACTI. Section 3.1 describes the organization of a mat. Section 3.2 describes the organization of the H-tree distribution networks. Section 3.3 presents the different organizational parameters associated with a data array.

3.1 Mat Organization

Figure 5 shows the high-level composition of all mats. A mat is always composed of 4 subarrays and associated predecoding/decoding logic which is located at the center of the mat. The predecoding/decoding logic is shared by all 4 subarrays. The bottom subarrays are mirror images of the top subarrays and the left hand side subarrays are mirror images of the right hand side ones. Not shown in this figure, by default, address/datain/dataout signals are assumed to enter the mat in the middle through its sides; alternatively, under user-control, it may also be specified to assume that they traverse over the memory cells.

Figure 6 shows the high-level composition of a subarray. The subarray consists of a 2D matrix of the memory cells and associated peripheral circuitry. Figure 7 shows the peripheral circuitry associated with bitlines of a subarray. After a wordline gets activated, memory cell data gets transferred to bitlines.

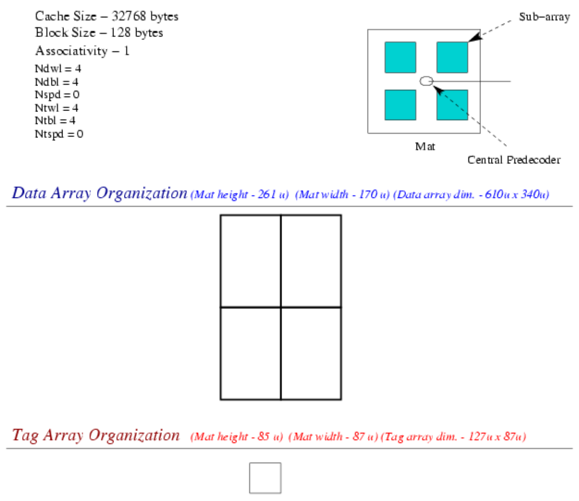

Cache Size – 32768 bytes
Block Size – 128 bytes
Associativity – 1
Ndwl = 4
Ndbl = 4
Nspd = 0
Ntwl = 4
Ntbl = 4
Ntspd = 0
Sub–array
Mat
Central Predecoder
Data Array Organization (Mat height - 261 u) (Mat width - 170 u) (Data array dim. - 610u x 340u)
Tag Array Organization (Mat height - 85 u) (Mat width - 87 u) (Tag array dim. - 127u x 87u)

Figure 3: Example of the graphical display generated by version 5.0.

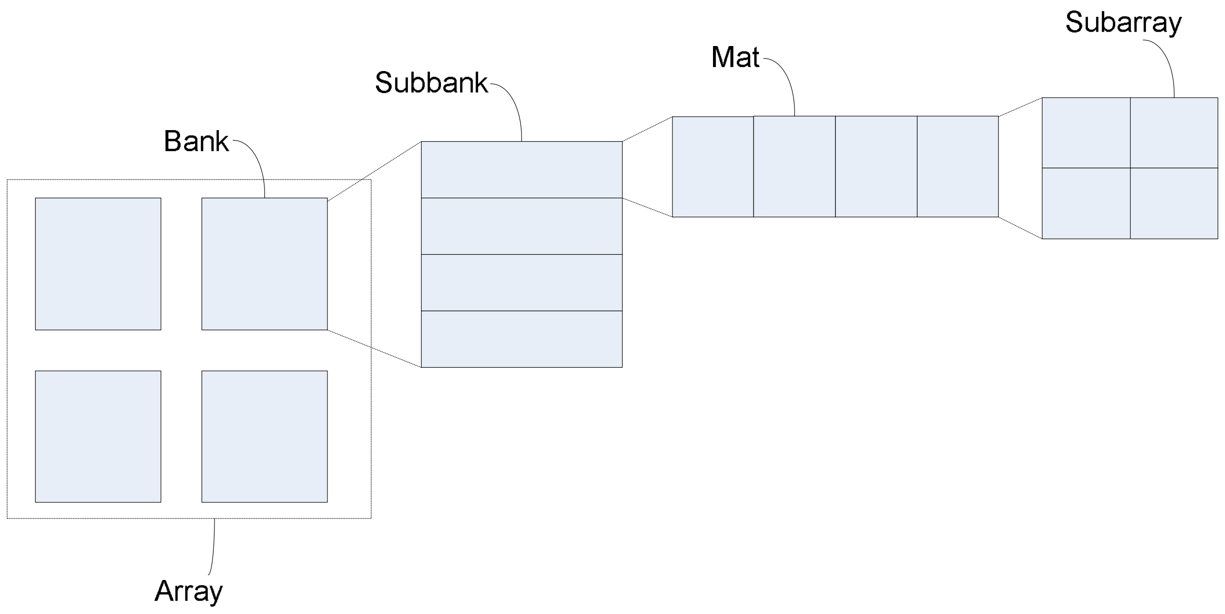


Figure 4: Layout of an example array with 4 banks. In this example each bank has 4 subbanks and each subbank has 4 mats.

The bitline data may go through a level of bitline multiplexing before it is sensed by the sense amplifiers. Depending on the degree of bitline multiplexing, a single sense amplifier may be shared by multiple bitlines. The data is sensed by the sense amplifiers and then passed to tristate output drivers which drive the dataout

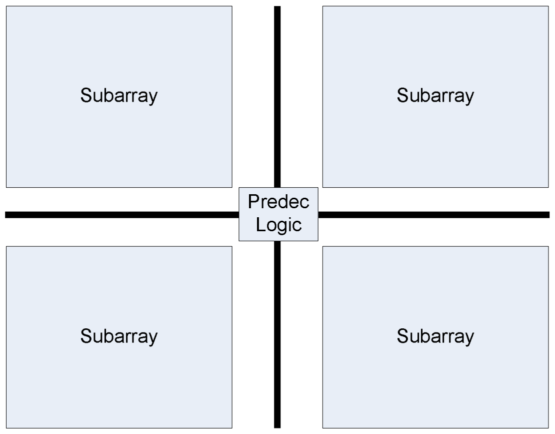


Figure 5: High-level composition of a mat.

vertical H-tree (described later in this section). An additional level of multiplexing may be required at the outputs of the sense amplifiers in organizations in which the bitline multiplexing is not sufficient to cull out the output data or in set-associative caches in which the output word from the correct way needs to be selected. The select signals that control the multiplexing of the bitline mux and the sense amp mux are generated by the bitline mux select signals decoder and the sense amp mux select signals decoder respectively. When the degree of multiplexing after the outputs of the sense amplifiers is simply equal to the associativity of the cache, the sense amp mux select signal decoder does not have to decode any address bits and instead simply buffers the input way-select signals that arrive from the tag array.

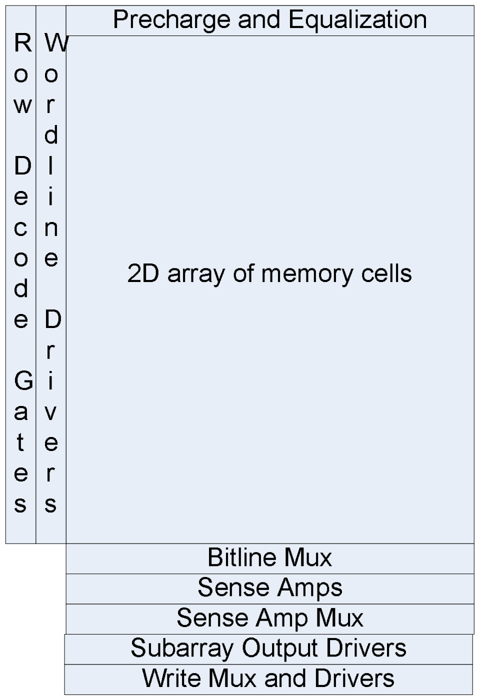


Figure 6: High-level composition of a subarray.

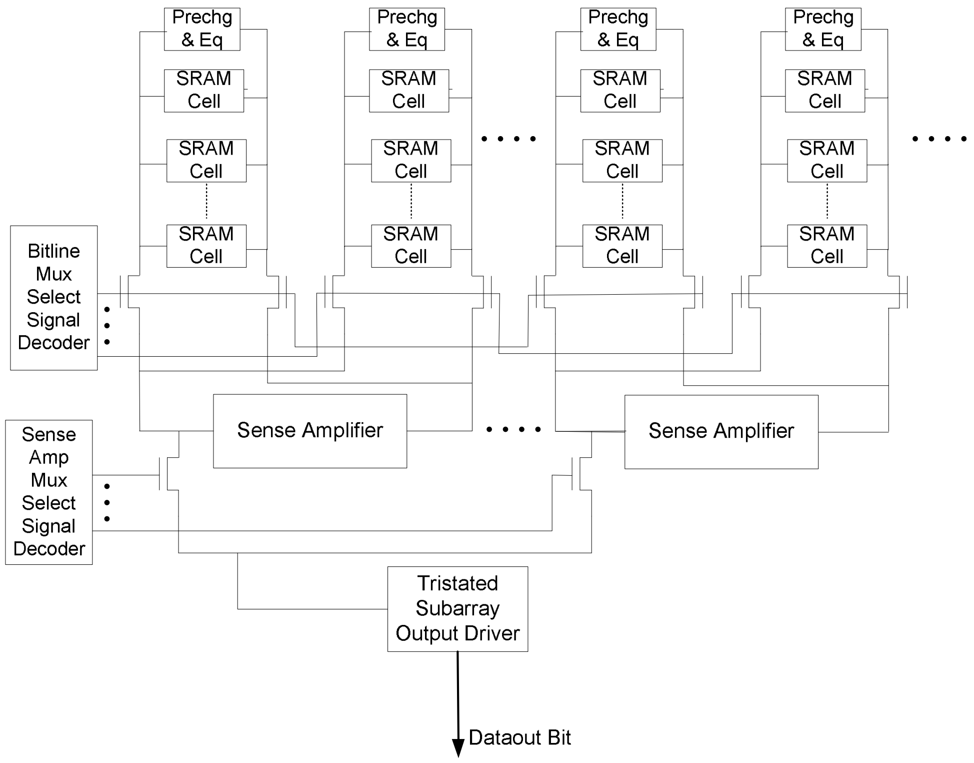

Prechg & Eq
SRAM Cell
Bitline Mux Select Signal Decoder
Sense Amplifier
Sense Amp Mux Select Signal Decoder
Tristated Subarray Output Driver
Dataout Bit

Figure 7: Peripheral circuitry associated with bitlines. Not shown in this figure, but the outputs of the muxes are assumed to be precharged high.

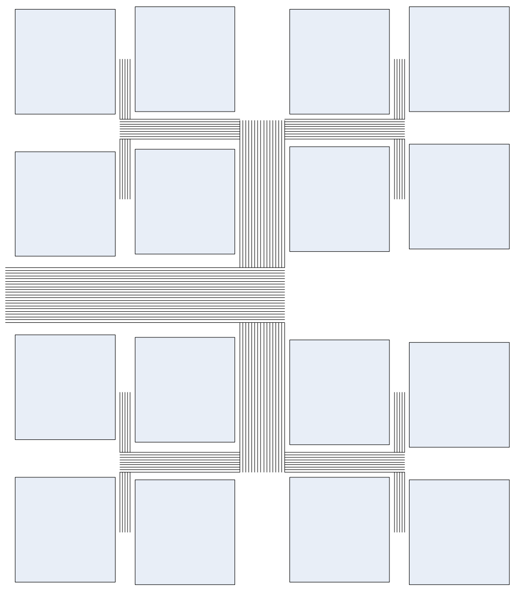


Figure 8: Layout of edge of array to banks H-tree network.

3.2 Routing to Mats

Address and data are routed to and from the mats on H-tree distribution networks. H-tree distribution networks are used to route address and data and provide uniform access to all the mats in a large memory.[1]

[1]Non-uniform cache architectures (NUCA) are currently beyond the scope of CACTI 5.0 but may be supported by future versions of CACTI.

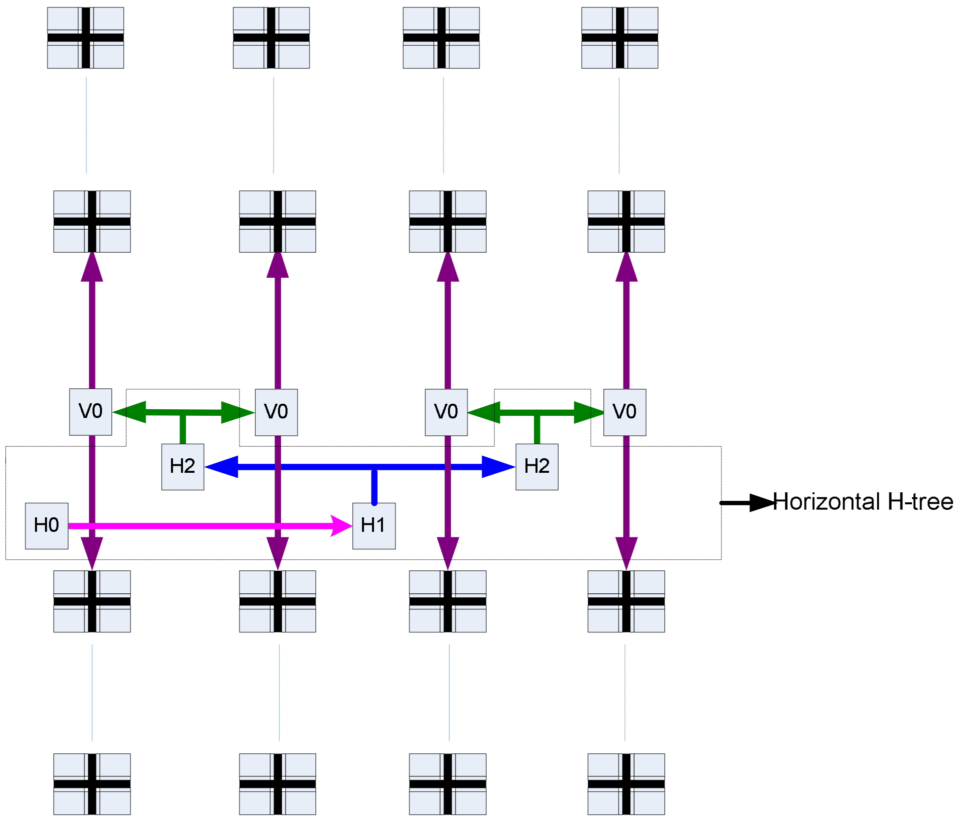


Figure 9: Layout of the horizontal H-tree within a bank.

Such a memory organization is interconnect-centric and is well-suited for coping with the trend of worsening wire delay with respect to device delay. Rather than shipping a bunch of predecoded address signals to the mats, it makes sense to ship the address bits and decode them at the sinks (mats) [22]. Contemporary divided wordline architectures which make use of broadcast of global signals suffer from increased wire delay as memory capacities get larger [20]. Details of a memory organization similar to what we have assumed may also be found in [23]. For ease of pipelining multiple accesses in the array, separate request and reply networks are assumed. The request network carries address and datain from the edge of the array to the mats while the reply network carries dataout from the mats to the edge of the array. The structure of the request and reply networks is similar; here we discuss the high-level organization of the request network.

The request H-tree network is divided into two networks:

1. The H-tree network from the edge of the array to the edge of a bank; and,

2. The H-tree network from the edge of the bank to the mats.

Figure 8 shows the layout of the request H-tree network between the array edge and the banks. Address and datain are routed to each bank on this H-tree network and enter each bank at the middle from one of its sides. The H-tree network from the edge of the bank to the mats is further divided into two 1-dimensional horizontal and vertical H-tree networks. Figure 9 shows the layout of the horizontal H-tree within a bank which is located at the middle of the bank while Figure 10 shows the layout of the vertical H-trees within a bank. The leaves of the horizontal H-tree act as the parent nodes (marked as V0) of the vertical H-trees. In order to understand the routing of signals on the H-tree networks within a bank, we use an illustrative example. Consider a bank with the following parameters: 1MB capacity, 256-bit output word, 4 subbanks, 4 mats in each subbank. Looked at together, Figures 9 and 10 can be considered to be the horizontal and vertical H-trees within such a bank. The number of address bits required to address a word in this bank is 15. As there are 8 subbanks and because each mat in a subbank is activated during an access, the number

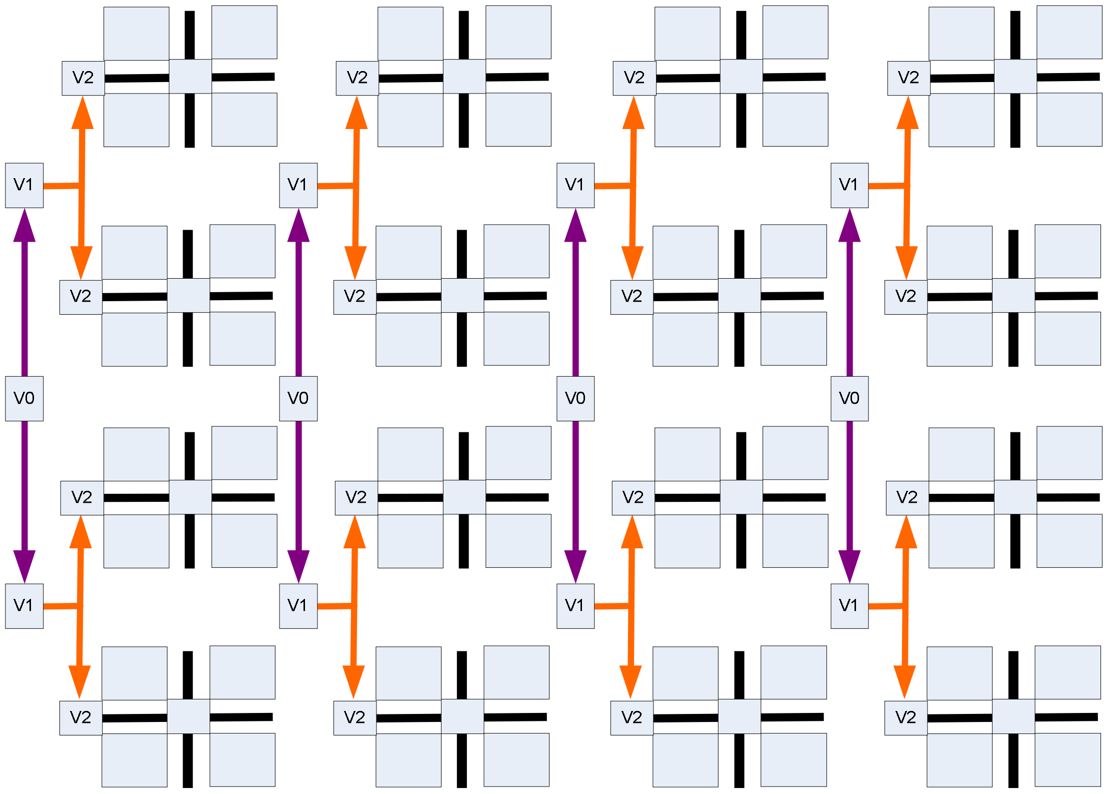


Figure 10: Layout of the vertical H-trees within a bank.

of address bits that need to be distributed to each mat is 12. Because each mat in a subbank produces 64 out of the 256 output bits, the number of datain signals that need to be distributed to each mat is 64. Thus 15 bits of address and 256 bits of datain enter the bank from the left side driven by the H0 node. At the H1 node, the 15 address signals are redriven such that each of the two nodes H1 receive the 15 address signals. The datain signals split at node H1 and 32 datain signals go to the left H2 node and the other 32 go to the right H2 node. At each H2 node, the address signals are again redriven such that all of the 4 V0 nodes end up receiving the 15 address bits. The datain signals again split at each H2 node so that each V0 node ends up receiving 64 datain bits. These 15 address bits and 64 datain bits then traverse to each mat along the 4 vertical H-trees. In the vertical H-trees, address and datain may either be assumed to be broadcast to all mats or alternatively, it may be assumed that these signals are appropriately gated so that they are routed to just the correct subbank that contains the data; by default, we assume the latter scenario.

The reply network H-trees are similar in principle to the request network H-trees. In case of the reply network vertical H-trees, dataout bits from each mat of a subbank travel on the vertical H-trees to the middle of the bank where they sink into the reply network horizontal H-tree, and are carried to the edge of the bank.

3.3 Organizational Parameters of a Data Array

In order to calculate the optimal organization based on a given objective function, like earlier versions of CACTI [1][2][3][4], each bank is associated with partitioning parameters N_{dwl}, N_{dbl} and N_{spd}, where N_{dwl} = Number of segments in a bank wordline, N_{dbl} = Number of segments in a bank bitline, and N_{spd} = Number of sets mapped to each bank wordline.

Unlike earlier versions of CACTI, in CACTI 5.0 N_{spd} can take on fractional values less than one. This is useful for small highly-associative caches with large line sizes. Without values of N_{spd} less than one, memory mats with huge aspect ratios with only a few word lines but hundreds of bits per word line would be created. For a pure scratchpad memory (not a cache), N_{spd} is used to vary the aspect ratio of the memory bank.

N_{subbanks} and $N_{\text{mats-in-subbank}}$ are related to N_{dwl} and N_{dbl} as follows:

$$N_{\text{subbanks}} = \frac{N_{\text{dbl}}}{2} \tag{1}$$

$$N_{\text{mats-in-subbank}} = \frac{N_{\text{dwl}}}{2} \tag{2}$$

Figure 11 shows different partitions of the same bank. The partitioning parameters are labeled alongside. Table 1 lists various organizational parameters associated with a data array.

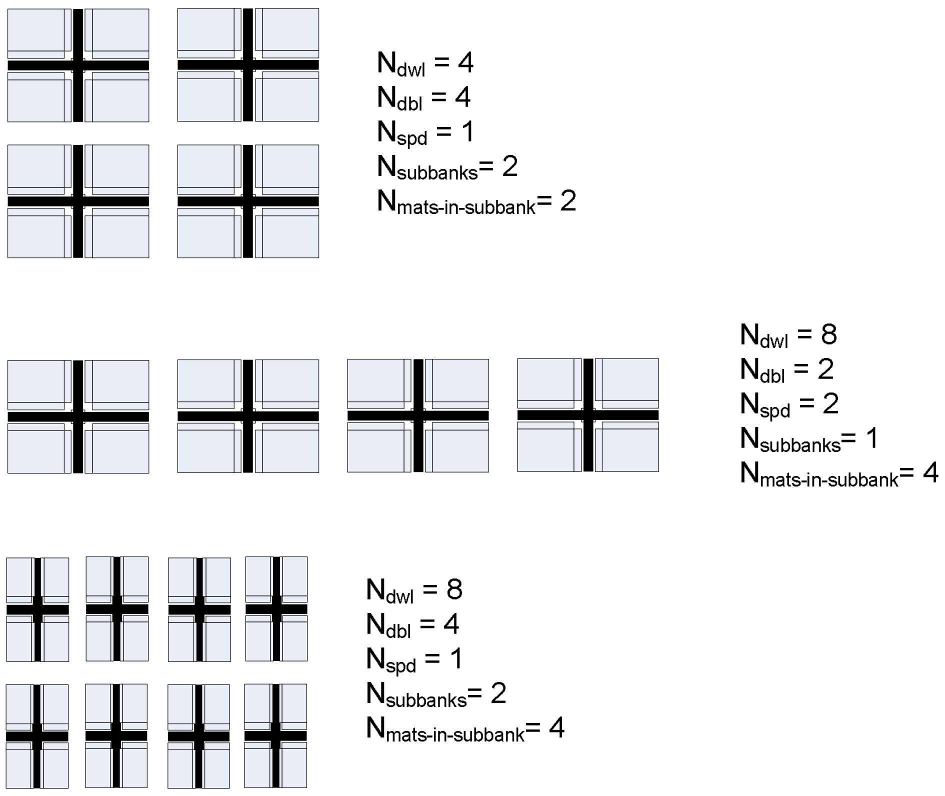


Figure 11: Different partitions of a bank.

3.4 Comments about Organization of Data Array

The cache organization chosen in the CACTI model is a compromise between many possible different cache organizations. For example, in some organizations all the data bits could be read out of a single mat. This could reduce dynamic power but increase routing requirements. On the other hand, organizations exist where all mats are activated on a request and each produces part of the bits required. This obviously burns a lot of dynamic power, but has the smallest routing requirements. CACTI chooses a middle ground, where all the bits for a read come from a single subbank, but multiple mats. Other more complicated organizations, in which predecoders are shared by two subarrays instead of four, or in which sense amplifiers are shared between top and bottom subarrays, are also possible, however we try to model a simple common case in CACTI.

4 Circuit Models and Sizing

In Section 3, the high-level organization of an array was described. In this section, we delve deeper into logic and circuit design of the different entities. We also present the techniques adopted for sizing different

Parameter Name	Meaning	Parameter Type
N_{banks}	Number of banks	User input
N_{dwl}	Number of divisions in a bank wordline	Degree of freedom
N_{dbl}	Number of divisions in a bank bitline	Degree of freedom
N_{spd}	Number of sets mapped to a bank wordline	Degree of freedom
$D_{\text{bitline-mux}}$	Degree of muxing at bitlines	Degree of freedom
N_{subbanks}	Number of subbanks	Calculated
$N_{\text{mats-in-subbank}}$	Number of mats in a subbank	Calculated
$N_{\text{subarr-rows}}$	Number of rows in a subarray	Calculated
$N_{\text{subarr-cols}}$	Number of columns in a subarray	Calculated
$D_{\text{out-driv-mux}}$	Degree of bitline multiplexing	Degree of freedom
$D_{\text{out-driv-mux}}$	Degree of sense amp multiplexing	Calculated
$N_{\text{subarr-sense-amps}}$	Number of sense amplifiers in a subarray	Calculated
$N_{\text{subarr-out-drivers}}$	Number of output drivers in a subarray	Calculated
$N_{\text{bank-addr-bits}}$	Number of address bits to a bank	Calculated
$N_{\text{bank-datain-bits}}$	Number of datain bits to a mat	Calculated
$N_{\text{bank-dataout-bits}}$	Number of dataout bits from a mat	Calculated
$N_{\text{mat-addr-bits}}$	Number of address bits to a mat	Calculated
$N_{\text{mat-datain-bits}}$	Number of datain bits to a mat	Calculated
$N_{\text{mat-dataout-bits}}$	Number of dataout bits from a mat	Calculated
$N_{\text{mat-way-select}}$	Number of way-select bits to a mat (for data array of cache)	Calculated

Table 1: Organizational parameters of a data array.

circuits. The rest of this section is organized as follows: First, in Section 4.1, we describe the circuit model that we have assumed for wires. Next in Section 4.2, we describe the general philosophy that we have adopted for sizing circuits. Next in Section 4.3, we describe the circuit models and sizing techniques for the different circuits within a mat, and in Section 4.5, we describe them for the circuits used in the different H-tree networks.

4.1 Wire Modeling

Wires are considered to belong to one of two types: ideal or non-ideal. Ideal wires are assumed to have zero resistance and capacitance. Non-ideal wires are assumed to have finite resistance and capacitance and are modeled using a one-section Pi RC model shown in Figure 12. In this figure, R_{wire} and C_{wire} for a wire of length L_{wire} are given by the following equations:

$$R_{\text{wire}} = L_{\text{wire}} R_{\text{unit-length-wire}} \quad (3)$$
$$C_{\text{wire}} = L_{\text{wire}} C_{\text{unit-length-wire}} \quad (4)$$

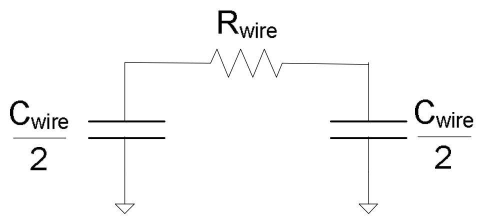


Figure 12: One-section Pi RC model that we have assumed for non-ideal wires.

For computation of $R_{\text{unit-length-wire}}$ and $C_{\text{unit-length-wire}}$ wires, we use the equations presented in [6][12] which are reproduced below. Figure 13 shows the accompanying picture for the capacitance model from [6].

$$R_{\text{unit-length-wire}} = \alpha_{\text{scatter}} \frac{\rho}{(thickness - barrier - dishing)(width - 2 * barrier)} \quad (5)$$

$$C_{\text{unit-length-wire}} = \epsilon_0 (2M\epsilon_{\text{horiz}} \frac{thickness}{spacing} + 2\epsilon_{\text{vert}} \frac{width}{ILD_{\text{thick}}}) + fringe(\epsilon_{\text{horiz}}, \epsilon_{\text{vert}}) \quad (6)$$

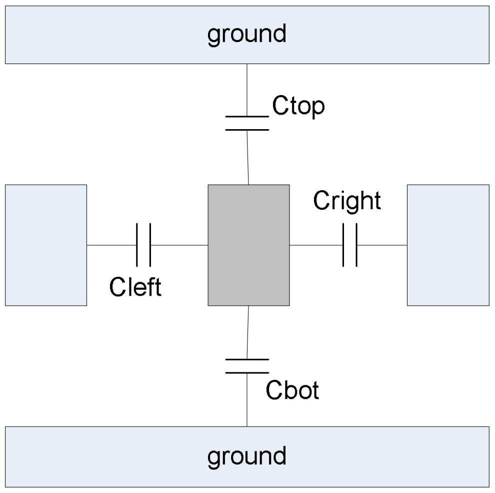


Figure 13: Capacitance model from [6].

4.2 Sizing Philosophy

In general the sizing of circuits depends on various optimization goals: circuits may be sized for minimum delay, minimum energy-delay product, etc. CACTI's goal is to model simple representative circuit sizing applicable to a broad range of common applications. As in earlier SRAM modeling efforts [5][20][24], we have made extensive use of the method of logical effort in sizing different circuit blocks. Explanation of the method of logical effort may be found in [25].

4.3 Sizing of Mat Circuits

As described earlier in Section 3.1, a mat is composed of entities such as the predecoding/decoding logic, memory cell array and bitline peripheral circuitry. We present circuits, models and sizing techniques for these entities.

4.3.1 Predecoder and Decoder

As discussed in Section 2, new circuit structures have been adopted for the decoding logic. The same decoding logic circuit structures are utilized for producing the row-decode signals and the select signals of the bitline and sense amplifier muxes. In the discussion here, we focus on the row-decoding logic. In order to describe the circuit structures assumed within the different entities of the row-decoding logic, we use an illustrative example. Figure 14 shows the structure of the row-decoding logic for a subarray with 1024 rows. The row-decoding logic is composed of two row-predecode blocks and the row-decode gates and drivers. The row-predecode blocks are responsible for predecoding the address bits and generating predecoded signals. The row-decode gates and drivers are responsible for decoding the predecoded outputs and driving the wordline load. Each row-predecode block can predecode a maximum of 9 bits and has a 2-level logic structure. With 1024 rows, the number of address bits required for row-decoding is 10. Figure 15 shows the structure of each row predecode block for a subarray with 1024 rows. Each row predecode block is responsible for predecoding 5 address bits and each of them generates 32 predecoded output bits. Each predecode block has two levels.

The first level is composed of one 2-4 decode unit and one 3-8 decode unit. At the second level, the 4 outputs from the 2-4 decode unit and the 8 outputs from the 3-8 decode unit are combined together using 32 NAND2 gates in order to produce the 32 predecoded outputs. The 32 predecoded outputs from each predecode block are combined together using the 1024 NAND2 gates to generate the row decode signals.

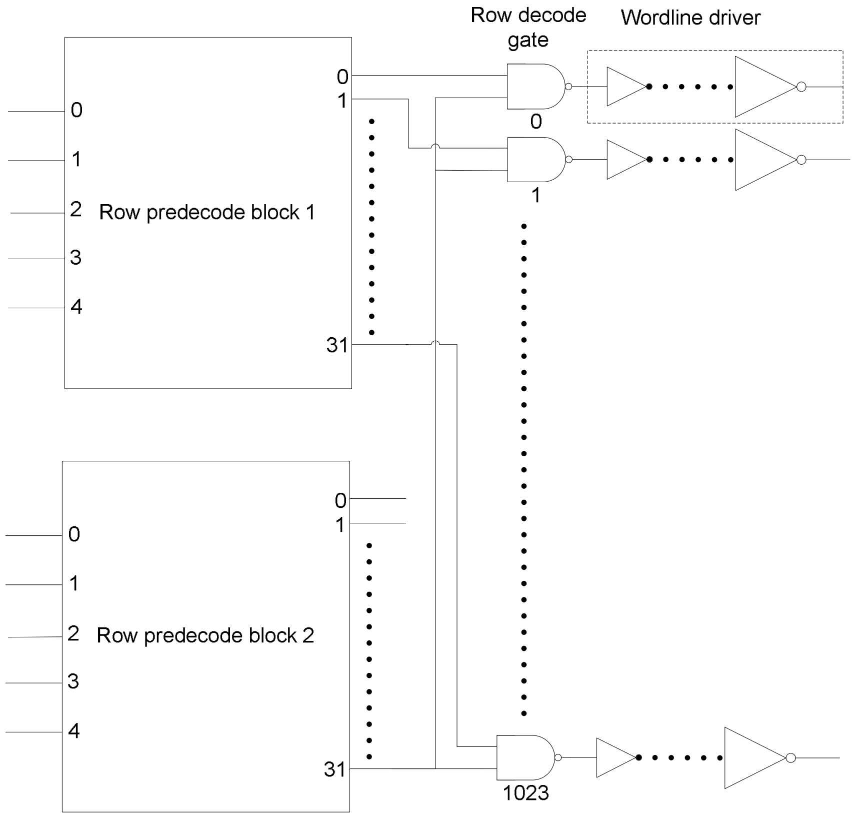


Figure 14: Structure of the row decoding logic for a subarray with 1024 rows.

Figure 17 shows the circuit paths in the decoding logic for the subarray with 1024 rows. One of the paths contains the NAND2 of the 2-4 decode unit and the other contains the NAND3 gate of the 3-8 decode unit. Each path has 3 stages in its path. The branching efforts at the outputs of the first two stages are also shown in the figure. The predecode output wire is treated as a non-ideal wire with its $R_{\text{predec-out-wire}}$ and $C_{\text{predec-out-wire}}$ computed using the following equations:

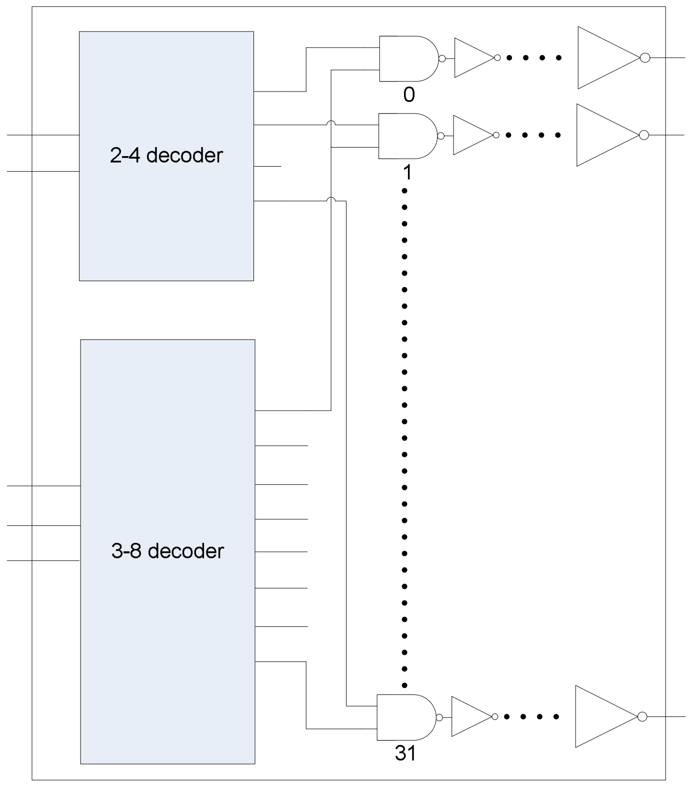
$$R_{\text{predec-output-wire}} = L_{\text{predec-output-wire}} R_{\text{unit-length-wire}} \tag{7}$$

$$C_{\text{predec-output-wire}} = L_{\text{predec-output-wire}} C_{\text{unit-length-wire}} \tag{8}$$

where $L_{\text{predec-output-wire}}$ is the maximum length amongst lengths of predecode output wires.

The sizing of gates in each circuit path is calculated using the method of logical effort. In each of the 3 stages of each circuit path, minimum-size transistors are assumed at the input of the stage and each stage is sized independent of each other using the method of logical effort. While this is not optimal from a delay point of view, it is simpler to model and has been found to be a good sizing heuristic from an energy-delay point of view [5].

In this example that we considered for decoding logic of a subarray with 1024 rows, there were two different circuit paths, one involving the NAND2 gate and another involving the NAND3 gate. In the general case, when each predecode block decodes different number of address bits, a maximum of four circuit paths may exist. When the degree of decoding is low, some of the circuit blocks shown in Figure 14 may not be required. For example, Figure 16 shows the decoding logic for a subarray with 8 rows. In this case, the decoding logic simply involves a 3-8 decode unit as shown.



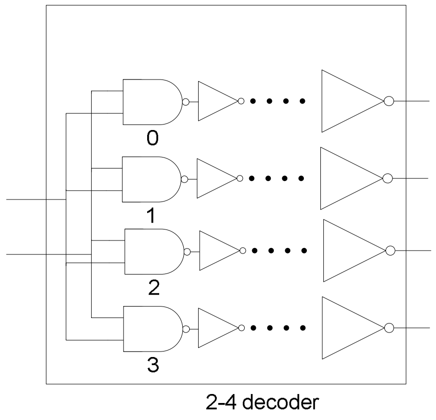


Figure 15: Structure of the row predecode block for a subarray with 1024 rows.

As mentioned before, the same circuit structures used within the row-decoding logic are also used for generating the select signals of the bitline and sense amplifier muxes. However, unlike the row-decoding logic in which the NAND2 decode gates and drivers are assumed to be placed on the side of subarray, the NAND2 decode gates and drivers are assumed to be placed at the center of the mat near their corresponding predecode blocks. Also, the resistance/capacitance of the wires between the predecode blocks and the decode gates are not modeled and are assumed to be zero.

4.3.2 Bitline Peripheral Circuitry

Memory Cell Figure 18 shows the circuit assumed for a 1-ported SRAM cell. The transistors of the SRAM cell are sized based on the widths specified in [17] and are presented in Section 8.

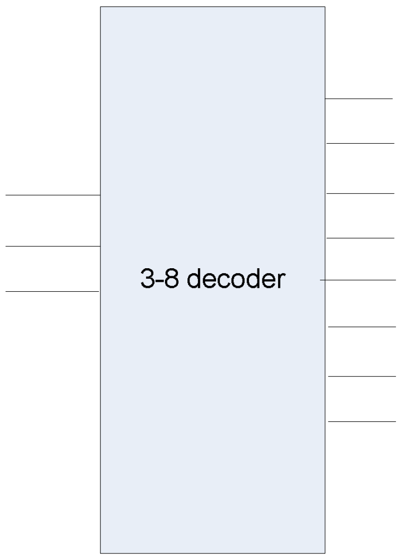
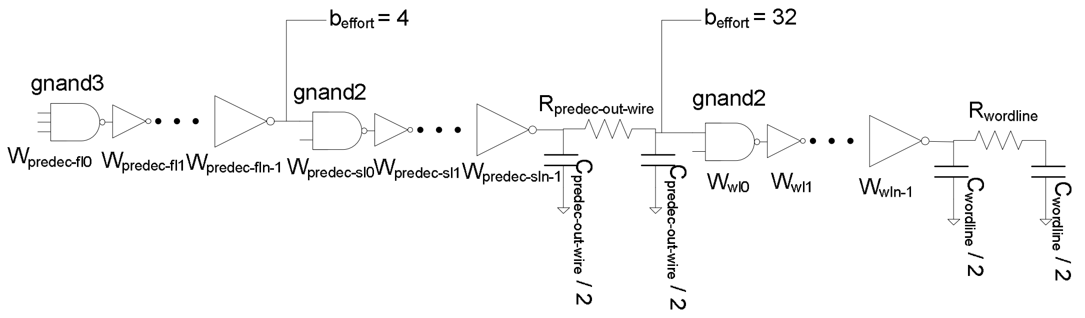


Figure 16: Structure of the row-decoding logic for a subarray with 8 rows. The row-decoding logic is simply composed of 8 decode gates and drivers.



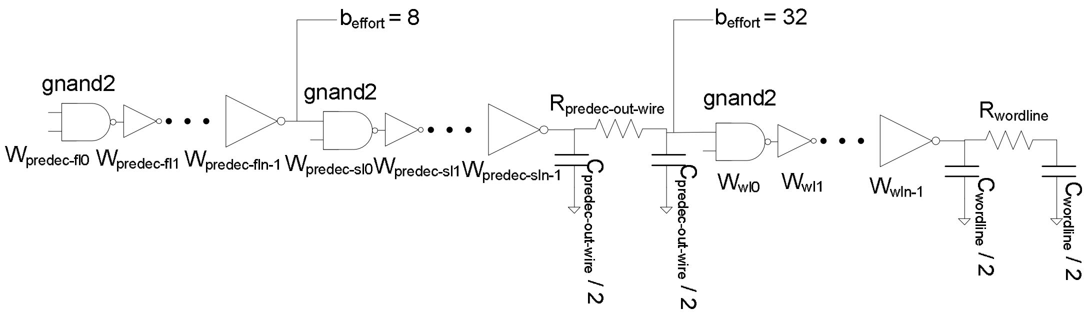


Figure 17: Row decoding logic circuit paths for a subarray with 1024 rows. One of the circuit paths contains the NAND2 gate of the 2-4 decode unit while the other contains the NAND3 gate of the 3-8 decode unit.

Sense Amplifier Figure 19 shows the circuit assumed for a sense amplifier - it's a clocked latch-based sense amplifier. When the ENABLE signal is not active, there is no flow of current through the transistors of the latch. The small-signal circuit model and analysis of this latch-based sense amplifier is presented in Section 4.4.

Bitline and Sense Amplifier Muxes Figure 20 shows the circuit assumed for the bitline and sense amplifier muxes. We assume that the mux is implemented using NMOS pass transistors. The use of NMOS

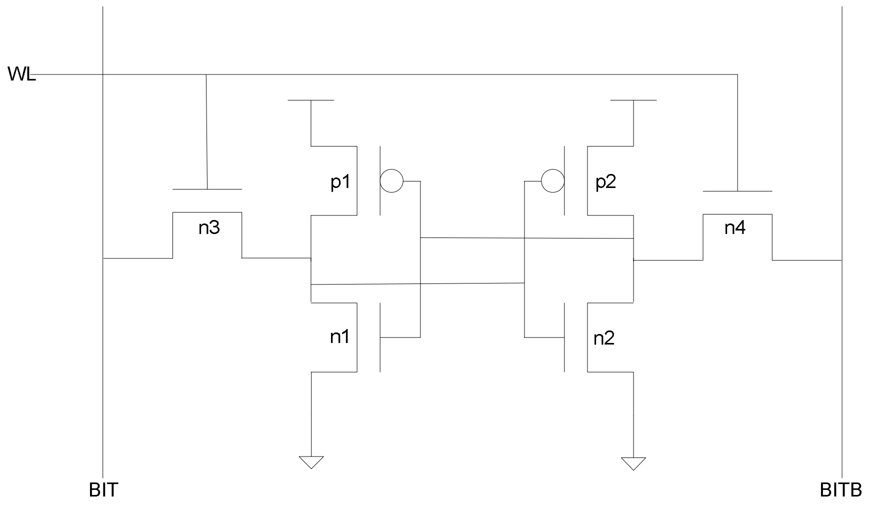


Figure 18: 1-ported 6T SRAM cell

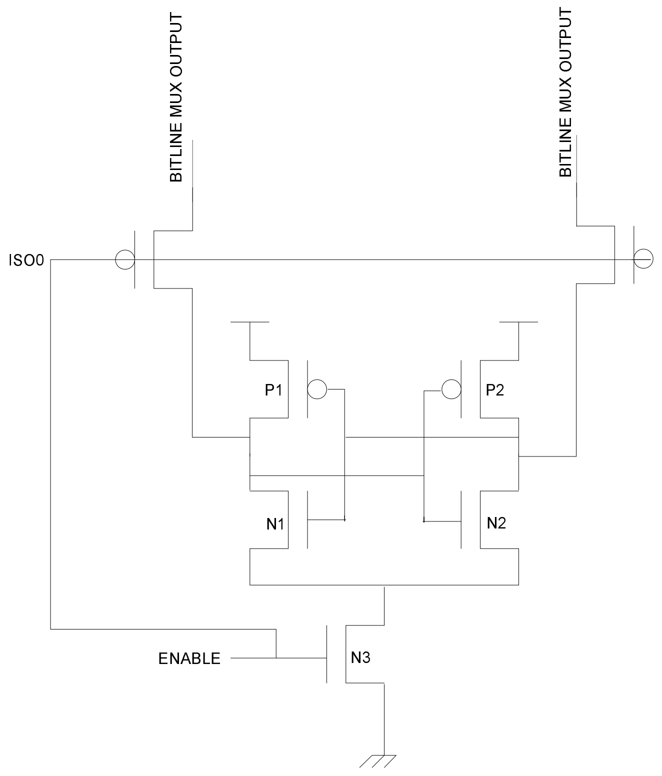


Figure 19: Clocked latch-based sense amplifier

transistors implies that the output of the mux needs to be precharged high in order to avoid degraded ones. We do not attempt to size the transistors in the muxes and instead assume (as in [20]) fixed widths for the NMOS transistors across all partitions of the array.

Precharge and Equalization Circuitry Figure 21 shows the circuit assumed for precharging and equalizing the bitlines. The bitlines are assumed to be precharged to VDD through the PMOS transistors. Just like the transistors in the bitline and sense amp muxes, we do not attempt to size the precharge and equalization transistors and instead assume fixed-width transistors across different partitions of the array.

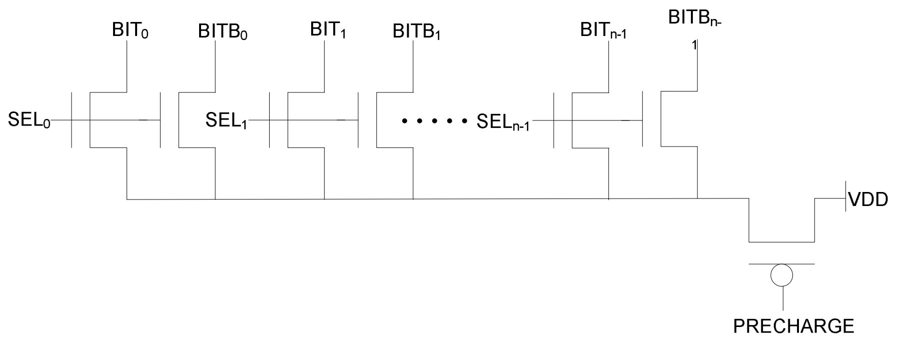


Figure 20: NMOS-based mux. The output is assumed to be precharged high.

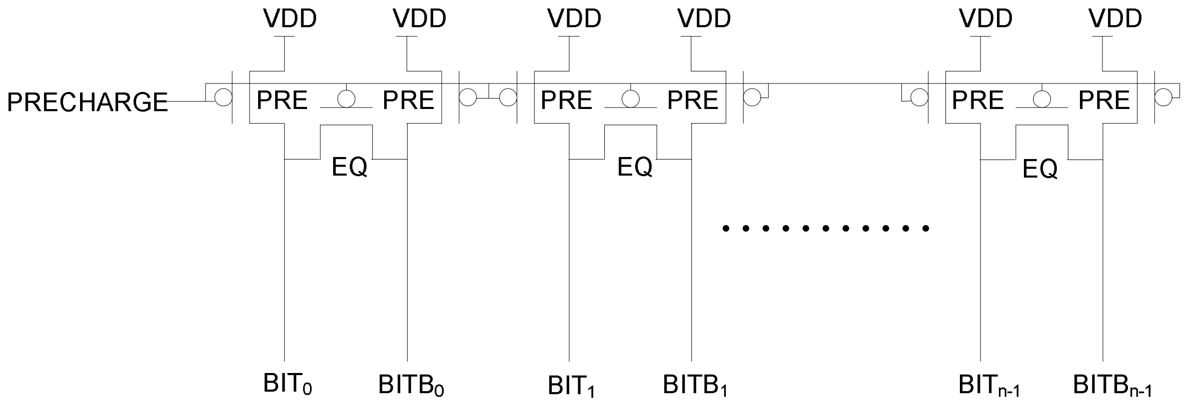


Figure 21: Bitlines precharge and equalization circuitry.

Bitlines Read Path Circuit Model Figure 22 shows the circuit model for the bitline read path between the memory cell and the sense amplifier mux.

4.4 Sense Amplifier Circuit Model

Figure 19 showed the clocked latch-based sense amplifier that we have assumed. [26] presents analysis of this circuit and equations for sensing delay under different assumptions. Figure 23 shows one of the small-signal models presented in [26]. Use of this small-signal model is based on two assumptions:

1. Current has been flowing in the circuit for a sufficiently long time; and
2. The equilibriating device can be modeled as an ideal switch.

For the small-signal model of Figure 23, it has been shown that the delay of the sensing operation is given by the following equation:

$$T_{\mathrm{sense}} = \frac{C_{\mathrm{sense}}}{G_{\mathrm{m}}} \ln(\frac{\mathrm{VDD}}{V_{\mathrm{sense}}}) \tag{9}$$

Use of this equation for calculation of sense amplifier delay requires that the value of G_{mn} and G_{mp} for the circuit be known. We assume that the transistors in the sense amplifier latch exhibit short-channel effects. For a transistor that exhibits short-channel effect, we use the following typical current equation [27] for computation of saturation current:

$$I_{\mathrm{dsat}} = \frac{\mu_{\mathrm{eff}}}{2} C_{\mathrm{ox}} \frac{W}{L}(V_{\mathrm{GS}} - V_{\mathrm{TH}})V_{\mathrm{dsat}} \tag{10}$$

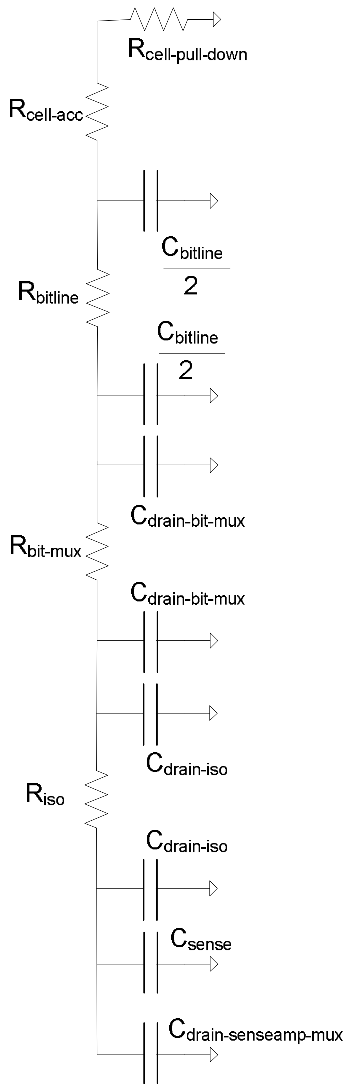


Figure 22: Circuit model of the bitline read path between the SRAM cell and the sense amplifier input.

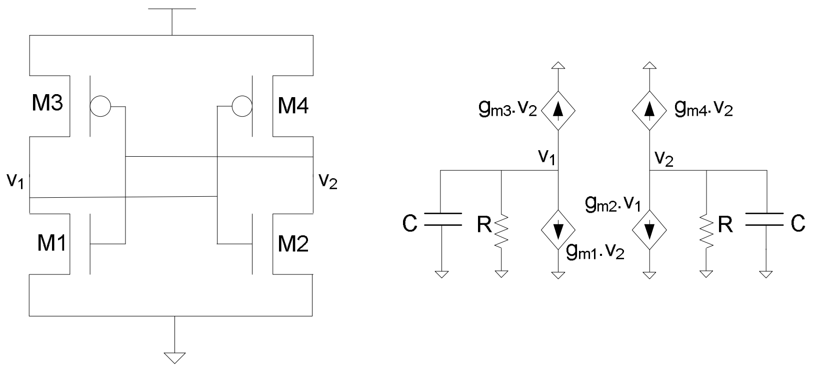


Figure 23: Small-signal model of the latch-based sense amplifier (from [26]).

Differentiating the above equation with respect to V_{GS} gives the equation for G_{m} of the transistor. It can be seen that because of short-channel effect, G_{m} comes out to be independent of V_{GS}.

$$G_{\mathrm{m}} = \frac{\mu_{\mathrm{eff}}}{2} C_{\mathrm{ox}} \frac{W}{L} V_{\mathrm{dsat}} \tag{11}$$

4.5 Routing Networks

As described earlier in Section 3.2, address and data are routed to and from the mats on H-tree distribution networks. First address/data are routed on an H-tree from array edge to bank edge and then on another H-tree from bank edge to the mats.

4.5.1 Array Edge to Bank Edge H-tree

Figure 8 showed the layout of H-tree distribution of address and data between the array edge and the banks. This H-tree network is assumed to be composed of inverter-based repeaters. The sizing of the repeaters and the separation distance between them is determined based on the formulae given in [6]. In order to allow for energy-delay tradeoffs in the repeater design, we introduce an user-controlled variable "maximum percentage of delay away from best repeater solution" or 'maxrepeaterdelayconstraint' in short. A maxrepeaterdelayconstraint of zero results in the best delay repeater solution. For a maxrepeaterdelayconstraint of 10%, the delay of the path is allowed to get worse by a maximum of 10% with respect to the best delay repeater solution by reducing the sizing and increasing the separation distance. Thus, with the maxrepeaterdelayconstraint, limited energy savings are possible at the expense of delay.

4.5.2 Bank Edge to Mat H-tree

Figures 9 and 10 showed layout examples of horizontal and vertical H-trees within a bank, each with 3 nodes. We assume that drivers are placed at each of the nodes of these H-trees. Figure 24 shows the circuit path and driver circuit structure of the address/datain H-trees, and Figure 25 shows the circuit path and driver circuit structure of the vertical dataout H-tree. In order to allow for signal-gating in the address/datain H-trees we consider multi-stage buffers with a 2-input NAND gate as the input stage. The sizing and number of gates at each node of the H-trees is computed using the methodology described in [5] which takes into account the resistance and capacitance of the intermediate wires in the H-tree.

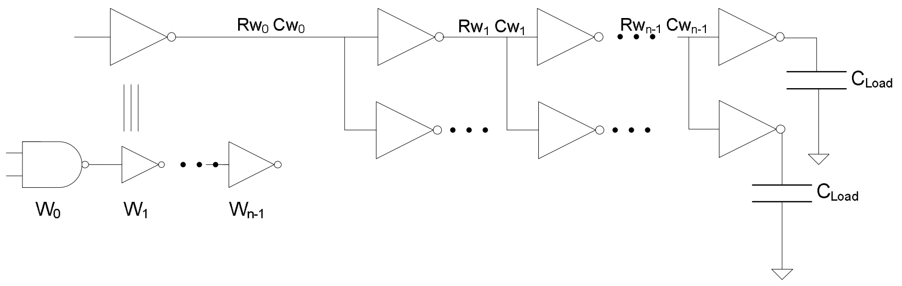


Figure 24: Circuit path of address/datain H-trees within a bank.

One problem with the circuit paths of Figures 24 and 25 is that they start experiencing increased wire delays as the wire lengths between the drivers start to get long. This also limits the maximum random cycle time that can be achieved for the array. So, as an alternative to modeling drivers only at H-tree branching nodes, we also consider an alternative model in which the H-tree circuit paths within a bank are composed of buffers at regular intervals (i.e. repeaters). With repeaters, the delay through the H-tree paths within a bank can be reduced at the expense of increased power consumption. Figure 26 shows the different types of buffer circuits that have been modeled in the H-tree path. At the branches of the H-tree, we again assume buffers with a NAND gate in the input stage in order to allow for signal-gating whereas in the H-tree segments between two nodes, we model inverter-based buffers. We again size these buffers according to the buffer sizing formulae given in [6]. The maxrepeaterdelayconstraint that was described in Section 4.5.1 is also used here to decide the sizing of the buffers and their separation distance so that delay in these H-trees also may be traded off for potential energy savings.

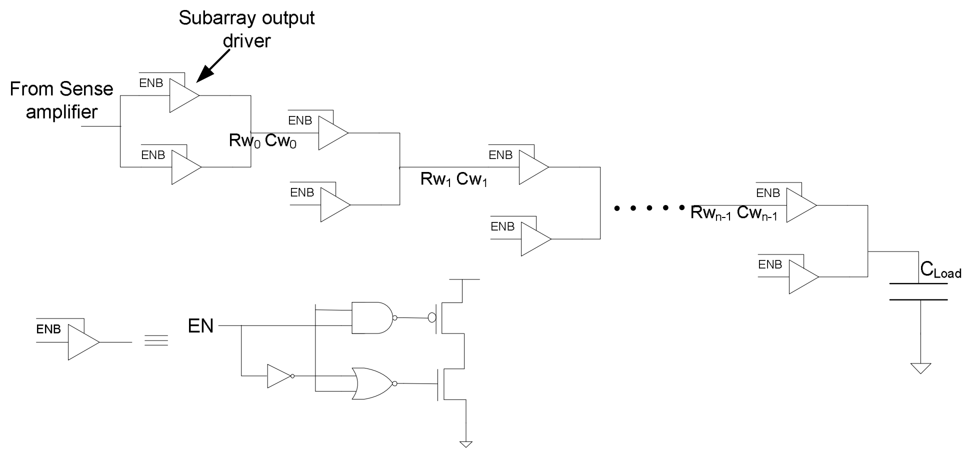


Figure 25: Circuit path of vertical dataout H-trees.

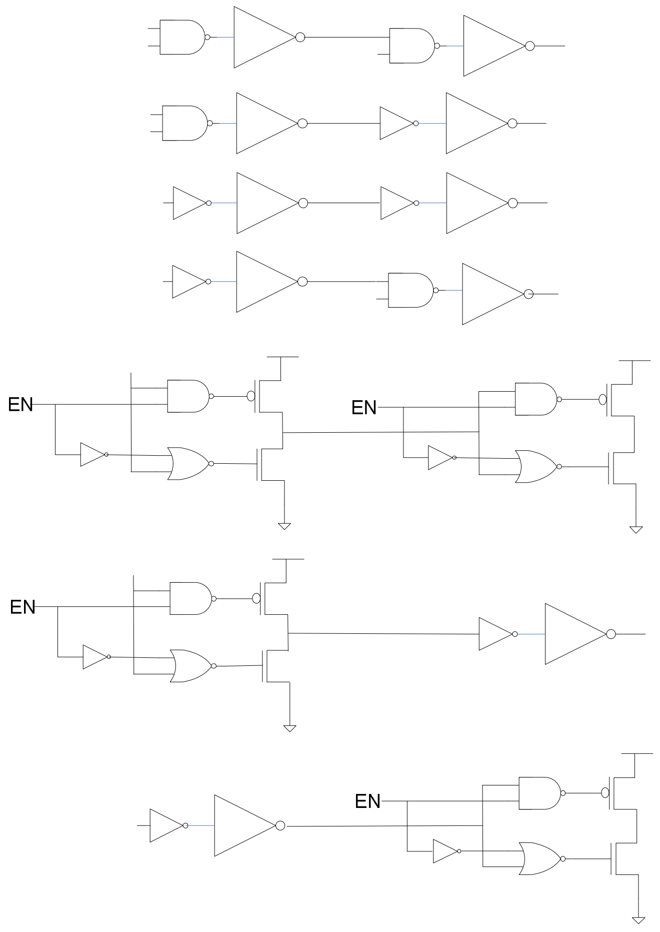


Figure 26: Different types of buffer circuit stages that have been modeled in the H-trees within a bank.

5 Area Modeling

In this section, we describe the area model of a data array. In Section 5.1, we describe the area model that we have used to find the areas of simple gates. We then present the equations of the area model in Section 5.2.

5.1 Gate Area Model

A new area model has been used to estimate the areas of transistors and gates such as inverter, NAND and NOR gates. This area model is based off a layout model from [19] which describes a fast technique to estimate standard cell characteristics before the cells are actually laid out. Figure 27 illustrates the layout model that has been used in [19]. Table 2 shows the process/technology input parameters required by this gate area model. For a thorough description of the technique, please refer to [19]. Gates with stacked transistors are assumed to have a layout similar to that described in [1]. When a transistor width exceeds a certain maximum value ($H_{\text{n-diff}}$ for NMOS and $H_{\text{p-diff}}$ for PMOS in Table 2), the transistor is assumed to be folded. This maximum value can either be process-specific or context-specific. An example of when a context-specific width would be used is in case of memory sense amplifiers which typically have to be laid out at a certain pitch.

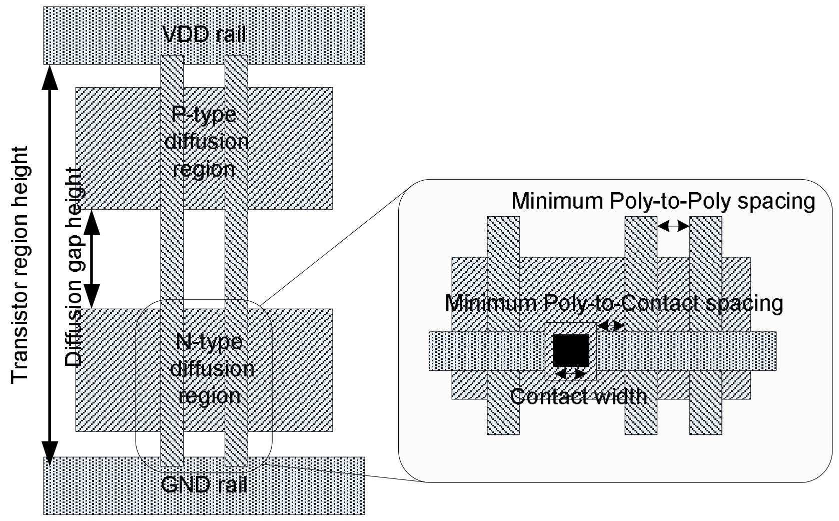


Figure 27: Layout model assumed for gates (from [19]).

Parameter name	Meaning
$H_{\text{n-diff}}$	Maximum height of n diffusion of a transistor
$H_{\text{p-diff}}$	Maximum height of p diffusion for a transistor
$H_{\text{gap-bet-same-diffs}}$	Minimum gap between diffusions of the same type
$H_{\text{gap-bet-opp-diffs}}$	Minimum gap between n and p diffusions
$H_{\text{power-rail}}$	Height of VDD (GND) power rail
W_p	Minimum width of poly (poly half-pitch or process feature size)
$S_{\text{p-p}}$	Minimum poly-to-poly spacing
W_{c}	Contact width
$S_{\text{p-c}}$	Minimum poly-to-contact spacing

Table 2: Process/technology input parameters required by the gate area model.

Given the width of an NMOS transistor, $W_{\text{before-folding}}$, the number of folded transistors may be calculated as follows:

$$N_{\text{folded-transistors}} = \lceil \frac{W_{\text{before-folding}}}{H_{\text{n-diff}}} \rceil \tag{12}$$

The equation for total diffusion width of N_{stacked} transistors when they are not folded is given by the following equation:

$$\text{total-diff-width} = 2(W_c + 2S_{\text{p-c}}) + N_{\text{stacked}}W_p + (N_{\text{stacked}} - 1)S_{\text{p-p}} \tag{13}$$

The equation for total diffusion width of N_{stacked} transistors when they are folded is given by the following equation:

$$\begin{aligned}\text{total-diff-width} &= 2N_{\text{folded-transistors}}(W_c + 2S_{\text{p-c}}) + N_{\text{folded-transistors}}N_{\text{stacked}}W_p + \\ &\quad N_{\text{folded-transistors}}(N_{\text{stacked}} - 1)S_{\text{p-p}}\end{aligned} \tag{14}$$

Note that Equation 14 is a generalized form of the equations used for calculating diffusion width (for computation of drain capacitance) in the original CACTI report [1]. Earlier versions of CACTI assumed at most two folded transistors; in version 5.0, we allow the degree of folding to be greater than 2 and make the associated layout and area models more general. Note that drain capacitance calculation in version 5.0 makes use of equations similar to 13 and 14 for computation of diffusion width.

The height of a gate is calculated using the following equation:

$$H_{\text{gate}} = H_{\text{n-diff}} + H_{\text{p-diff}} + H_{\text{gap-bet-opp-diffs}} + 2H_{\text{power-rail}} \tag{15}$$

5.2 Area Model Equations

The area of the data array is estimated based on the area occupied by a single bank and the area spent in routing address and data to the banks. It is assumed that the area spent in routing address and data to the bank is decided by the pitch of the routed wires. Figures 28 and 29 show two example arrays with 8 and 16 banks respectively; we present equations for the calculation of the areas of these arrays.

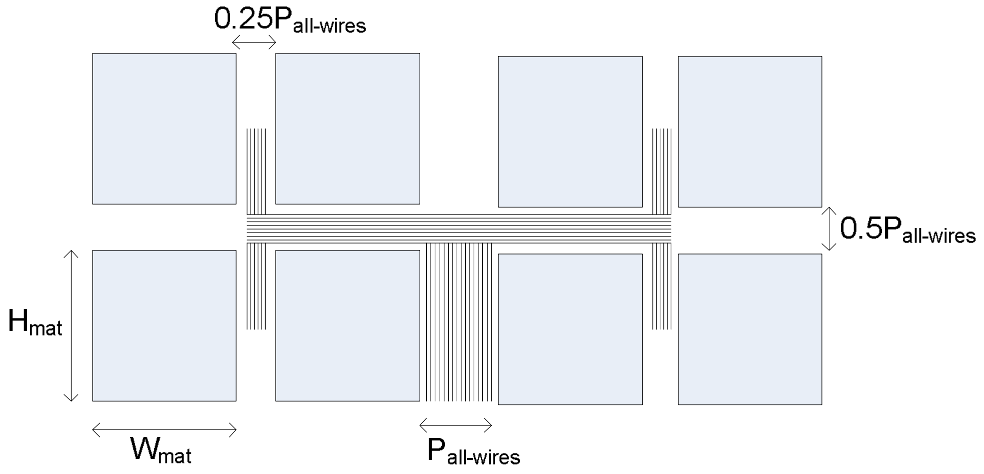


Figure 28: Supporting figure for example area calculation of array with 8 banks.

$$A_{\text{data-arr}} = H_{\text{data-arr}}W_{\text{data-arr}} \tag{16}$$

The pitch of wires routed to the banks is given by the following equation:

$$P_{\text{all-wires}} = P_{\text{wire}}N_{\text{wires-routed-to-banks}} \tag{17}$$

For the data array of Figure 28 with 8 banks, the relevant equations are as follows:

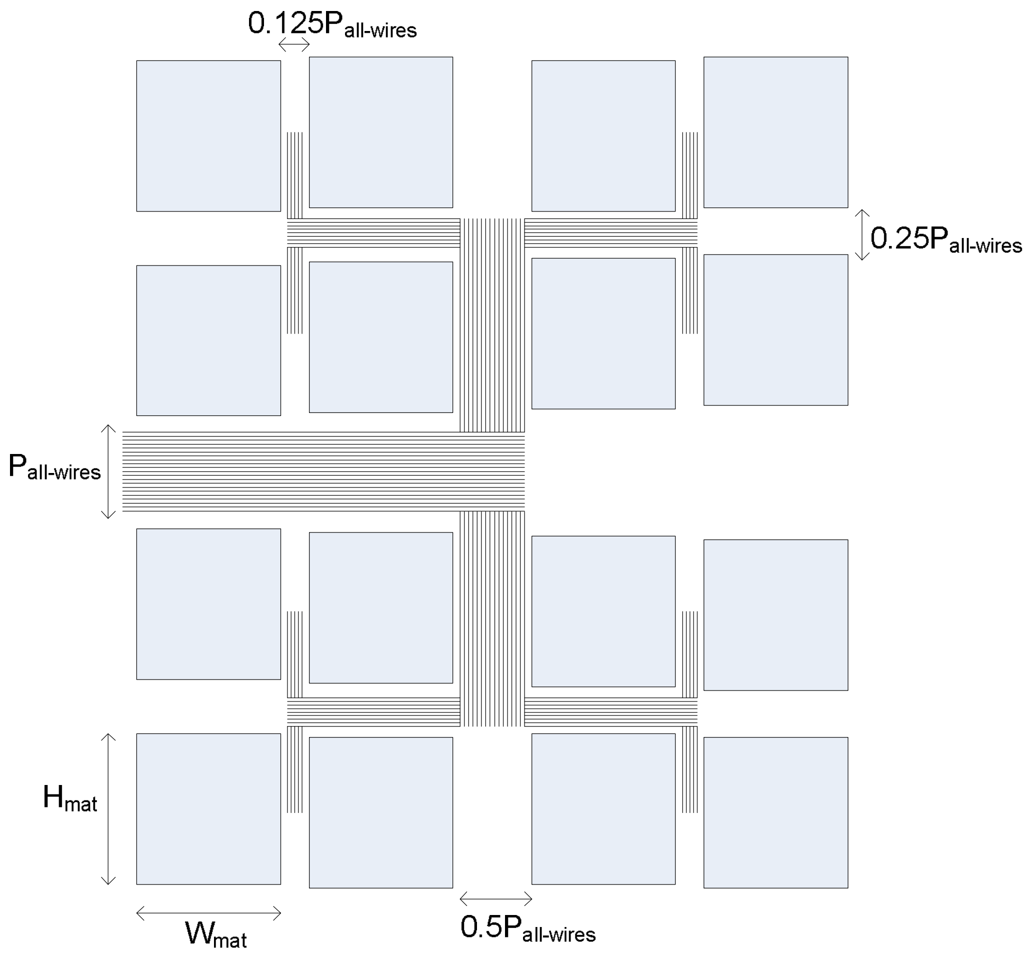


Figure 29: Supporting figure for example area calculation of array with 16 banks.

$$
\begin{aligned}
W_{\text{data-arr}} &= 4W_{\text{bank}} + P_{\text{all-wires}} + 2\frac{P_{\text{all-wires}}}{4} && (18)\\
H_{\text{data-arr}} &= 2H_{\text{bank}} + \frac{P_{\text{all-wires}}}{2} && (19)\\
N_{\text{wires-routed-to-banks}} &= 8(N_{\text{bank-addr-bits}} + N_{\text{bank-datain-bits}} + N_{\text{bank-dataout-bits}} + \\
&\quad N_{\text{way-select-signals}}) && (20)
\end{aligned}
$$

For the data array of Figure 29 with 16 banks, the relevant equations are as follows:

$$
\begin{aligned}
H_{\text{data-arr}} &= 4H_{\text{bank}} + P_{\text{all-wires}} + 2\frac{P_{\text{all-wires}}}{4} && (21)\\
W_{\text{data-arr}} &= 4W_{\text{bank}} + \frac{P_{\text{all-wires}}}{2} + 2\frac{P_{\text{all-wires}}}{8} && (22)\\
N_{\text{wires-routed-to-banks}} &= 16(N_{\text{bank-addr-bits}} + N_{\text{bank-datain-bits}} + N_{\text{bank-dataout-bits}} + \\
&\quad N_{\text{way-select-signals}}) && (23)
\end{aligned}
$$

The banks in a data array are assumed to be placed in such a way that the number of banks in the horizontal direction is always either equal to or twice the number of banks in the vertical direction. The height and width of a bank is calculated by computing the area occupied by the mats and the area occupied by the routing resources of the horizontal and vertical H-tree networks within a bank. We again use an example to illustrate the calculations. Figures 9 and 10 showed the layouts of horizontal and vertical H-trees within a bank. The horizontal and vertical H-trees were each shown to have three branching nodes (H0, H1 and H2; V0, V1 and V2). Combined together, these horizontal and vertical H-trees may be considered as H-trees within a bank with 4 subbanks and 4 mats in each subbank. We present area model equations for such a bank.

$$A_{\text{bank}} = H_{\text{bank}} W_{\text{bank}} \tag{24}$$

In version 5.0, as described in Section 4.5, for the H-trees within a bank we assume that drivers are placed either only at the branching nodes of the H-trees or that there are buffers at regular intervals in the H-tree segments. When drivers are present only at the branching nodes of the vertical H-trees within a bank, we consider two alternative models in accounting for area overhead of the vertical H-trees. In the first model, we consider that wires of the vertical H-trees may traverse over memory cell area; in this case, the area overhead caused by the vertical H-trees is in terms of area occupied by drivers which are placed between the mats. In the second model, we do not assume that the wires traverse over the memory cell area and instead assume that they occupy area besides the mats. The second model is also applicable when there are buffers at regular intervals in the H-tree segments. The equations that we present next for area calculation of a bank assume the second model i.e. the wires of the vertical H-trees are assumed to not pass over the memory cell area. The equations for area calculation under the assumption that the vertical H-tree wires go over the memory cell area are quite similar. For our example bank with 4 subbanks and 4 mats in each subbank, the height of the bank is calculated to be equal to the sum of heights of all subbanks plus the height of the routing resources of the horizontal H-tree.

$$H_{\text{bank}} = 4H_{\text{mat}} + H_{\text{hor-htree}} \tag{25}$$

The width of the bank is calculated to be equal to the sum of widths of all mats in a subbank plus the width of the routing resources of the vertical H-trees.

$$W_{\text{bank}} = 4(W_{\text{mat}} + W_{\text{ver-htree}}) \tag{26}$$

The height of the horizontal H-tree is calculated as the height of the area occupied by the wires in the H-tree. These wires include the address, way-select, datain, and dataout signals. Figure 30 illustrates the layout that we assume for the wires of the horizontal H-tree. We assume that the wires are laid out using a single layer of metal. The height of the area occupied by the wires can be calculated simply by finding the total pitch of all wires in the horizontal H-tree. Figure 31 illustrates the layout style assumed for the vertical H-tree wires, and is similar to that assumed for the horizontal H-tree wires. Again the width of the area occupied by a vertical H-tree can be calculated by finding the total pitch of all wires in the vertical H-tree.

$$H_{\text{hor-htree}} = P_{\text{hor-htree-wires}} \tag{27}$$
$$W_{\text{ver-htree}} = P_{\text{ver-htree-wires}} \tag{28}$$

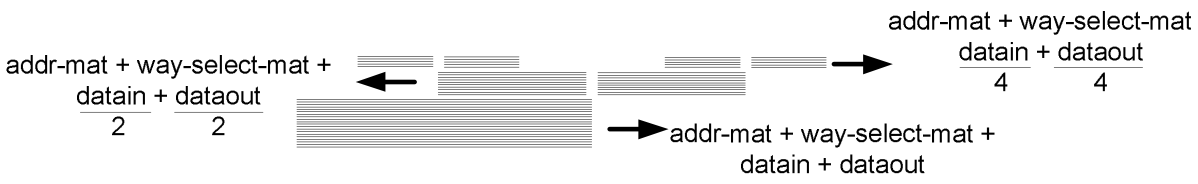

addr-mat + way-select-mat + $\frac{\text{datain}}{2}$ + $\frac{\text{dataout}}{2}$
addr-mat + way-select-mat $\frac{\text{datain}}{4}$ + $\frac{\text{dataout}}{4}$
addr-mat + way-select-mat + datain + dataout

Figure 30: Layout assumed for wires of the horizontal H-tree within a bank.

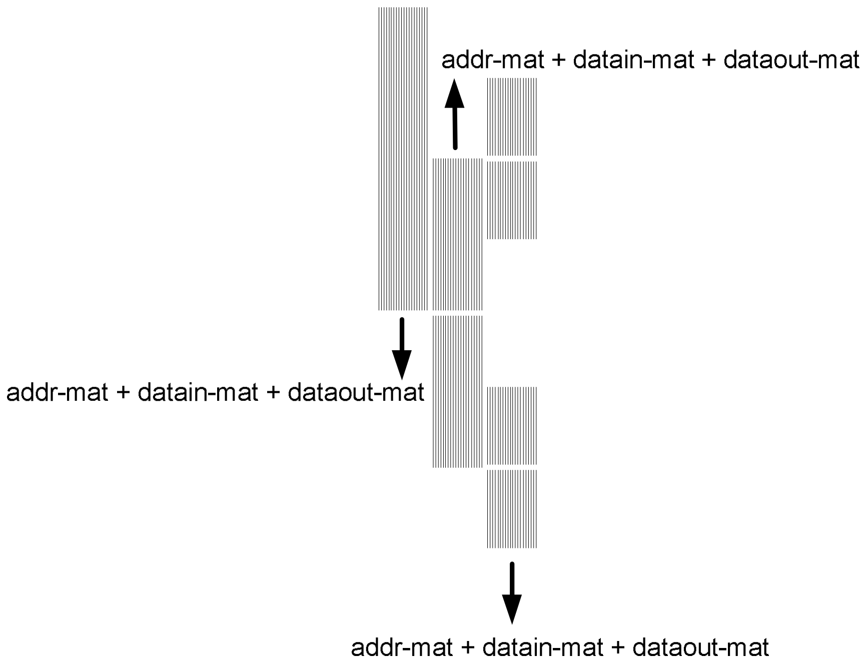

addr-mat + datain-mat + dataout-mat
addr-mat + datain-mat + dataout-mat
addr-mat + datain-mat + dataout-mat

Figure 31: Layout assumed for wires of the vertical H-tree within a bank.

The height and width of a mat are estimated using the following equations. Figure 32 shows the layout of a mat and illustrates the assumptions made in the following equations. We assume that half of the address, way-select, datain and dataout signals enter the mat from its left and the other half enter from the right.

$$W_{\text{mat}} = \frac{H_{\text{mat}} W_{\text{mat-initial}} + A_{\text{mat-center-circuitry}}}{W_{\text{initial-mat}}} \tag{29}$$

$$H_{\text{mat}} = 2H_{\text{subarr-mem-cell-area}} + H_{\text{mat-non-cell-area}} \tag{30}$$

$$W_{\text{initial-mat}} = 2W_{\text{subarr-mem-cell-area}} + W_{\text{mat-non-cell-area}} \tag{31}$$

$$\begin{aligned} A_{\text{mat-center-circuitry}} = {} & A_{\text{row-predec-block-1}} + A_{\text{row-predec-block-2}} \\ & + A_{\text{bit-mux-predec-block-1}} + A_{\text{bit-mux-predec-block-2}} \\ & + A_{\text{senseamp-mux-predec-block-1}} + A_{\text{senseamp-mux-predec-block-2}} + \\ & A_{\text{bit-mux-dec-drivers}} + A_{\text{senseamp-mux-dec-drivers}} \end{aligned} \tag{32}$$

$$H_{subarr-mem-cell-area} = N_{\text{subarr-rows}} H_{\text{mem-cell}} \tag{33}$$

$$\begin{aligned} W_{subarr-mem-cell-area} = {} & N_{\text{subarr-cols}} W_{\text{mem-cell}} + \lfloor \frac{N_{\text{subarr-cols}}}{N_{\text{mem-cells-per-wordline-stitch}}} \rfloor W_{\text{wordline-stitch}} + \\ & \lceil \frac{N_{\text{subarr-cols}}}{N_{\text{bits-per-ecc-bit}}} \rceil W_{\text{mem-cell}} \end{aligned} \tag{34}$$

$$H_{\text{mat-non-cell-area}} = 2H_{\text{subarr-bitline-peri-circ}} + H_{\text{hor-wires-within-mat}} \tag{35}$$

$$\begin{aligned} H_{\text{hor-wires-within-mat}} = {} & H_{\text{bit-mux-sel-wires}} + H_{\text{senseamp-mux-sel-wires}} + H_{\text{write-mux-sel-wires}} + \\ & \frac{H_{\text{number-mat-addr-bits}}}{2} + \frac{H_{\text{number-way-select-signals}}}{2} + \\ & \frac{H_{\text{number-mat-datain-bits}}}{2} + \frac{H_{\text{number-mat-dataout-bits}}}{2} \end{aligned} \tag{36}$$

$$W_{\text{mat-non-cell-area}} = \max(2W_{\text{subarr-row-decoder}}, W_{\text{row-predec-out-wires}}) \tag{37}$$

$$H_{\text{subarr-bitline-peri-cir}} = H_{\text{bit-mux}} + H_{\text{senseamp-mux}} + H_{\text{bitline-pre-eq}} + H_{\text{write-driver}} + H_{\text{write-mux}} \tag{38}$$

Note that the width of the mat is computed as in Equation 30 because we optimistically assume that the

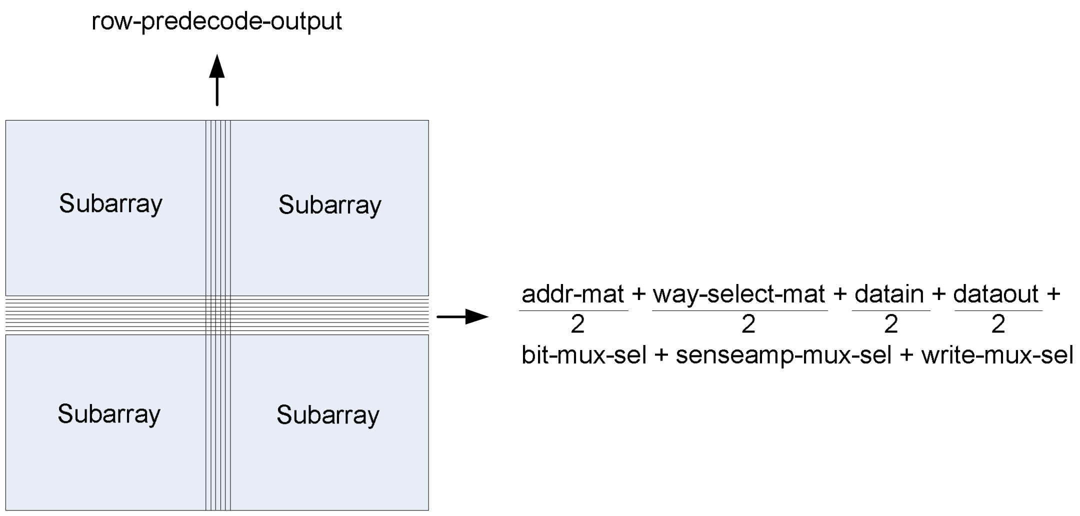

row-predecode-output
Subarray
Subarray
Subarray
Subarray
addr-mat/2 + way-select-mat/2 + datain/2 + dataout/2 +
bit-mux-sel + senseamp-mux-sel + write-mux-sel

Figure 32: Layout of a mat.

circuitry laid out at the center of the mat does not lead to white space in the mat. The areas of lower-level circuit blocks such as the bitline and sense amplifier muxes and write drivers are calculated using the area model that was described in Section 5.1 while taking into account pitch-matching constraints.

When redundancy in mats is also considered, the following area contribution due to redundant mats is added to the area of the data array computed in Equation 16.

$$A_{\text{redundant-mats}} = N_{\text{redundant-mats}} A_{\text{mat}} \tag{39}$$

$$N_{\text{redundant-mats}} = \lfloor \frac{N_{\text{banks}}}{N_{\text{mats}}} N_{\text{mats-per-redundant-mat}} \rfloor \tag{40}$$

where $N_{\text{mats-per-redundant-mat}}$ is the number of mats per redundant mat that and is set to 8 by default. The final height of the data array is readjusted under the optimistic assumption that the redundant mats do not cause any white space in the data array.

$$H_{\text{data-arr}} = \frac{A_{\text{data-arr}}}{W_{\text{data-arr}}} \tag{41}$$

6 Delay Modeling

In this section we present equations used in CACTI to calculate access time and random cycle time of a memory array.

6.1 Access Time Equations

$$T_{\text{access}} = T_{\text{request-network}} + T_{\text{mat}} + T_{\text{reply-network}} \tag{42}$$

$$T_{\text{request-network}} = T_{\text{arr-edge-to-bank-edge-htree}} + T_{\text{bank-addr-din-hor-htree}} + T_{\text{bank-addr-din-ver-htree}} \tag{43}$$

$$T_{\text{mat}} = \max(T_{\text{row-decoder-path}}, T_{\text{bit-mux-decoder-path}}, T_{\text{sense-amp-decoder-path}}) \tag{44}$$

$$T_{\text{reply-network}} = T_{\text{bank-dout-ver-htree}} + T_{\text{bank-dout-hor-htree}} + T_{\text{bank-edge-to-arr-edge}} \tag{45}$$

The critical path in the mat usually involves the wordline and bitline access. However, Equation 44 also must include a max with the delays of the bitline mux decoder and sense amp mux decoder paths as these circuits operate in parallel with the row decoding logic, and in general may act as the critical path for certain partitions of the data array. Usually when that happens, the number of rows in the subarray would be too few and the partitions would not get selected.

$$\begin{aligned}
T_{\text{row-decoder-path}} &= T_{\text{row-predec}} + T_{\text{row-dec-driver}} + T_{\text{bitline}} + T_{\text{sense-amp}} && (46)\\
T_{\text{bit-mux-decoder-path}} &= T_{\text{bit-mux-predec}} + T_{\text{bit-mux-dec-driver}} + T_{\text{sense-amp}} && (47)\\
T_{\text{senseamp-mux-decoder-path}} &= T_{\text{senseamp-mux-predec}} + T_{\text{senseamp-mux-dec-driver}} && (48)\\
T_{\text{row-predec}} &= \max(T_{\text{row-predec-blk-1-nand2-path}}, T_{\text{row-predec-blk-1-nand3-path}},\\
&\quad T_{\text{row-predec-blk-2-nand2-path}}, T_{\text{row-predec-blk-2-nand3-path}}) && (49)\\
T_{\text{bit-mux-sel-predec}} &= \max(T_{\text{bit-mux-sel-predec-blk-1-nand2-path}}, T_{\text{bit-mux-sel-predec-blk-1-nand3-path}},\\
&\quad T_{\text{bit-mux-sel-predec-blk-2-nand2-path}}, T_{\text{bit-mux-sel-predec-blk-2-nand3-path}}) && (50)\\
T_{\text{senseamp-mux-sel-predec}} &= \max(T_{\text{senseamp-mux-sel-predec-blk-1-nand2-path}}, T_{\text{senseamp-mux-sel-predec-blk-1-nand3-path}},\\
&\quad T_{\text{senseamp-mux-sel-predec-blk-2-nand2-path}}, T_{\text{senseamp-mux-sel-predec-blk-2-nand3-path}}) && (51)
\end{aligned}$$

The calculation for bitline delay is based on the model described in [28]. The model considers the effect of the wordline rise time.

$$T_{\text{bitline}} = \begin{cases} \sqrt{2T_{\text{step}}\frac{VDD-V_{\text{TH}}}{m}} & if\ T_{\text{step}} <= 0.5\frac{VDD-V_{\text{TH}}}{m} \\ T_{\text{step}} + \frac{VDD-V_{\text{TH}}}{2m} & if\ T_{\text{step}} > 0.5\frac{VDD-V_{\text{TH}}}{m} \end{cases} \quad (52)$$

$$\begin{aligned}
T_{\text{step}} &= (R_{\text{cell-pull-down}} + R_{\text{cell-acc}})(C_{\text{bitline}} + 2C_{\text{drain-bit-mux}} + C_{\text{iso}} + C_{\text{sense}} + C_{\text{drain-senseamp-mux}}) +\\
&\quad R_{\text{bitline}}(\frac{C_{\text{bitline}}}{2} + 2C_{\text{drain-bit-mux}} + C_{\text{iso}} + C_{\text{sense}} + C_{\text{drain-senseamp-mux}}) +\\
&\quad R_{\text{bit-mux}}(C_{\text{drain-bit-mux}} + C_{\text{iso}} + C_{\text{sense}} + C_{\text{drain-senseamp-mux}}) + R_{\text{iso}}(C_{\text{iso}} + C_{\text{sense}} +\\
&\quad C_{\text{drain-senseamp-mux}}) && (53)\\
m &= \frac{VDD - V_{\text{TH}}}{2T_{\text{step}}} && (54)
\end{aligned}$$

The calculation of sense amplifier delay makes use of the model described in [26].

$$\begin{aligned}
T_{\text{sense}} &= \tau ln(\frac{VDD}{V_{\text{sense}}}) && (55)\\
\tau &= \frac{C_{\text{sense}}}{G_{\text{m}}} && (56)
\end{aligned}$$

6.2 Random Cycle Time Equations

Typically, the random cycle time of an SRAM would be limited by wordline and bitline delays. In order to come up with an equation for lower bound on random cycle time, we consider that the SRAM is potentially pipelineable with placement of latches at appropriate locations.

$$\begin{aligned}
T_{\text{random-cycle}} &= \max(T_{\text{row-dec-driver}} + T_{\text{bitline}} + T_{\text{sense-amp}} + T_{\text{wordline-reset}} +\\
&\quad \max(T_{\text{bitline-precharge}}, T_{\text{bit-mux-out-precharge}}, T_{\text{senseamp-mux-out-precharge}}),\\
&\quad T_{\text{between-buffers-bank-hor-htree}}, T_{\text{between-buffers-bank-ver-dataout-htree}}, T_{\text{row-predec-blk}},\\
&\quad T_{\text{bit-mux-predec-blk}} + T_{\text{bit-mux-dec-driver}},\\
&\quad T_{\text{senseamp-mux-predec-blk}} + T_{\text{senseamp-mux-dec-driver}}) && (57)
\end{aligned}$$

We come up with an estimate for the wordline reset delay by assuming that the wordline discharges through the NMOS transistor of the final inverter in the wordline driver.

$$
\begin{aligned}
T_{\text{wordline-reset}} &= \ln(\frac{\text{VDD} - 0.1\text{VDD}}{\text{VDD}})(R_{\text{final-inv-wordline-driver}}C_{\text{wordline}} + \\
&\quad \frac{R_{\text{final-inv-wordline-driver}}C_{\text{wordline}}}{2}) \\
T_{\text{bitline-precharge}} &= \ln(\frac{\text{VDD} - 0.1V_{\text{bitline-swing}}}{\text{VDD} - V_{\text{bitline-swing}}})(R_{\text{bit-pre}}C_{\text{bitline}} + \frac{R_{\text{bitline}}C_{\text{bitline}}}{2}) && (58) \\
T_{\text{bit-mux-out-precharge}} &= \ln(\frac{\text{VDD} - 0.1V_{\text{bitline-swing}}}{\text{VDD} - V_{\text{bitline-swing}}})(R_{\text{bit-mux-pre}}C_{\text{bit-mux-out}} + \\
&\quad \frac{R_{\text{bit-mux-out}}C_{\text{bit-mux-out}}}{2}) && (59) \\
T_{\text{senseamp-mux-out-precharge}} &= \ln(\frac{\text{VDD} - 0.1V_{\text{bitline-swing}}}{\text{VDD} - V_{\text{bitline-swing}}})(R_{\text{senseamp-mux-pre}}C_{\text{senseamp-mux-out}} + \\
&\quad \frac{R_{\text{senseamp-mux-out}}C_{\text{senseamp-mux-out}}}{2}) && (60)
\end{aligned}
$$

7 Power Modeling

In this section, we present the equations used in CACTI to calculate dynamic power and leakage power of a data array. Here we present equations for dynamic read power; the equations for dynamic write power are similar.

$$P_{\text{read}} = \frac{E_{\text{dyn-read}}}{T_{\text{random-cycle}}} + P_{\text{leak}} \tag{61}$$

where $E_{\text{dyn-read}}$ is the dynamic read energy per access of the array, $T_{\text{random-cycle}}$ is the random cycle time of the array and P_{leak} is the leakage power in the array.

7.1 Calculation of Dynamic Energy

7.1.1 Dynamic Energy Calculation Example for a CMOS Gate Stage

We present a representative example to illustrate how we calculate the dynamic energy for a CMOS gate stage. Figure 33 shows a CMOS gate stage composed of a NAND2 gate followed by an inverter which drives the load. The energy consumption of this circuit is given by:

$$
\begin{aligned}
E_{\text{dyn}} &= E_{\text{dyn-nand2}} + E_{\text{dyn-inv}} && (62) \\
E_{\text{dyn-nand2}} &= 0.5(C_{\text{intrinsic-nand2}} + C_{\text{gate-inv}})\text{VDD}^2 && (63) \\
E_{\text{dyn-inv}} &= 0.5(C_{\text{intrinsic-inv}} + C_{\text{gate-load-next-stage}} + C_{\text{wire-load}})\text{VDD}^2 && (64) \\
C_{\text{instrinsic-nand2}} &= \text{draincap}(nand2, W_{\text{nand-pmos}}, W_{\text{nand-nmos}}) && (65) \\
C_{\text{gate-inv}} &= \text{gatecap}(inv, W_{\text{inv-pmos}}, W_{\text{inv-nmos}}) && (66) \\
C_{\text{drain-inv}} &= \text{draincap}(inv, W_{\text{inv-pmos}}, W_{\text{inv-nmos}}) && (67)
\end{aligned}
$$

The multiplicative factor of 0.5 in the equations of $E_{\text{dyn-nand2}}$ and $E_{\text{dyn-inv}}$ assumes consecutive charging and discharging cycles for each gate. Energy is consumed only during the charging cycle of a gate when its output goes from low to high.

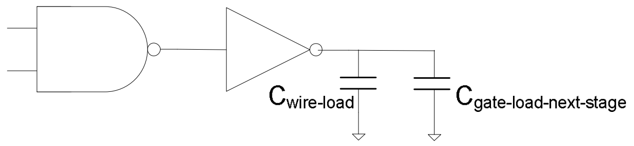


Figure 33: A simple CMOS gate stage composed of a NAND2 followed by an inverter which is driving a load.

7.1.2 Dynamic Energy Equations

The dynamic energy per read access consumed in the data array is the sum of the dynamic energy consumed in the mats and that consumed in the request and reply networks during a read access.

$$\begin{aligned}
E_{\text{dyn-read}} &= E_{\text{dyn-read-request-network}} + E_{\text{dyn-read-mats}} + E_{\text{dyn-read-reply-network}} && (68)\\
E_{\text{dyn-read-mats}} &= (E_{\text{dyn-predec-blks}} + E_{\text{dyn-decoder-drivers}} + E_{\text{dyn-read-bitlines}} + \\
&\quad E_{\text{senseamps}})N_{\text{banks}}N_{\text{subbanks}}N_{\text{mats-in-subbank}} && (69)\\
E_{\text{dyn-predec-blks}} &= E_{\text{dyn-row-predec-blks}} + E_{\text{dyn-bit-mux-predec-blks}} + \\
&\quad E_{\text{dyn-senseamp-mux-predec-blks}} && (70)\\
E_{\text{dyn-row-predec-blks}} &= E_{\text{dyn-row-predec-blk-1-nand2-path}} + E_{\text{dyn-row-predec-blk-1-nand3-path}} + \\
&\quad E_{\text{dyn-row-predec-blk-2-nand2-path}} + E_{\text{dyn-row-predec-blk-2-nand3-path}} && (71)\\
E_{\text{dyn-bit-mux-predec-blks}} &= E_{\text{dyn-bit-mux-predec-blk-1-nand2-path}} + E_{\text{dyn-bit-mux-predec-blk-1-nand3-path}} + \\
&\quad E_{\text{dyn-bit-mux-predec-blk-2-nand2-path}} + E_{\text{dyn-bit-mux-predec-blk-2-nand3-path}} && (72)\\
E_{\text{dyn-senseamp-mux-predec-blks}} &= E_{\text{dyn-senseampmux-predec-blk-1-nand2-path}} + \\
&\quad E_{\text{dyn-senseamp-mux-predec-blk-1-nand3-path}} + \\
&\quad E_{\text{dyn-senseamp-mux-predec-blk-2-nand2-path}} + \\
&\quad E_{\text{dyn-senseamp-mux-predec-blk-2-nand3-path}} && (73)\\
E_{\text{dyn-decoder-drivers}} &= E_{\text{dyn-row-decoder-drivers}} + E_{\text{dyn-bitmux-decoder-driver}} + \\
&\quad E_{\text{dyn-senseampmux-decoder-driver}} && (74)\\
E_{\text{dyn-row-decoder-drivers}} &= 4E_{\text{dyn-mat-row-decoder-driver}} && (75)\\
E_{\text{dyn-read-bitlines}} &= N_{\text{subarr-cols}}E_{\text{dyn-read-bitline}} && (76)\\
E_{\text{dyn-read-bitline}} &= C_{\text{bitline}}V_{\text{bitline-swing}}VDD && (77)\\
V_{\text{bitline-swing}} &= 2V_{\text{sense}} && (78)\\
E_{\text{dyn-read-request-network}} &= E_{\text{dyn-read-arr-edge-to-bank-edge-request-htree}} + E_{\text{dyn-read-bank-hor-request-htree}} + \\
&\quad E_{\text{dyn-read-bank-ver-request-htree}} && (79)\\
E_{\text{dyn-read-reply-network}} &= E_{\text{dyn-read-bank-ver-reply-htree}} + E_{\text{dyn-read-bank-hor-reply-htree}} + \\
&\quad E_{\text{dyn-read-bank-edge-to-arr-edge-reply-htree}} && (80)
\end{aligned}$$

Equation 79 assumes that the swing in the bitlines rises up to twice the signal that can be detected by the sense amplifier [20]. $E_{\text{dyn-read-request-network}}$ and $E_{\text{dyn-read-reply-network}}$ are calculated by determining the energy consumed in the wires/drivers/repeaters of the H-trees. The energy consumption in the horizontal and vertical H-trees of the request network within a bank for the example 1MB bank discussed in Section 4.5 with 4 subbanks and 4 mats in each subbank may be written as follows (referring to Figures 9 and 10 in Section 3.2):

$$\begin{aligned}
E_{\text{dyn-read-bank-hor-request-htree}} &= E_{\text{dyn-read-req-network-H0-H1}} + E_{\text{dyn-read-req-network-H1-H2}} + \\
&\quad E_{\text{dyn-read-req-network-read-H2-V0}} && (81)\\
E_{\text{dyn-read-bank-ver-request-htree}} &= E_{\text{dyn-read-req-network-V0-V1}} + E_{\text{dyn-read-req-network-V1-V2}} && (82)
\end{aligned}$$

The energy consumed in the H-tree segments depends on the location of the segment in the H-tree and the number of signals that are transmitted in each segment. In the request network, during a read access, between nodes H0 and H1, a total of 15 (address) signals are transmitted; between node H1 and both H2 nodes, a total of 30 (address) signals are transmitted; between all H2 and V0 nodes, a total of 60 (address) signals are transmitted. In the vertical H-tree, we assume signal-gating so that the address bits are transmitted to the mats of a single subbank only; thus, between all V0 and V1 nodes, a total of 56 (address) signals are transmitted; between all V1 and V2 nodes, a total of 52 (address) signals are transmitted.

$$E_{\text{dyn-read-req-network-H0-H1}} = (15)E_{\text{H0-H1-1-bit}} \tag{83}$$
$$E_{\text{dyn-read-req-network-H1-H2}} = (30)E_{\text{H1-H2-1-bit}} \tag{84}$$
$$E_{\text{dyn-read-req-network-H2-V0}} = (60)E_{\text{H2-V0-1-bit}} \tag{85}$$
$$E_{\text{dyn-read-req-network-V0-V1}} = (56)E_{\text{V0-V1-1-bit}} \tag{86}$$
$$E_{\text{dyn-read-req-network-V1-V2}} = (52)E_{\text{V1-V2-1-bit}} \tag{87}$$

The equations for energy consumed in the H-trees of the reply network are similar in form to the above equations. Also, the equations for dynamic energy per write access are similar to the ones that have been presented here for read access. In case of write access, the datain bits are written into the memory cells at full swing of the bitlines.

7.2 Calculation of Leakage Power

We estimate the standby leakage power consumed in the array. Our leakage power estimation does not consider the use of any leakage control mechanism in the array. We make use of the methodology presented in [29][24] to simply provide an estimate of the drain-to-source subthreshold leakage current for all transistors that are off with VDD applied across their drain and source.

7.2.1 Leakage Power Calculation for CMOS gates

We illustrate our methodology of calculation of leakage power for the CMOS gates that are used in our modeling. Figure 34 illustrates the leakage power calculation for an inverter. When the input is low and the output is high, there is subthreshold leakage through the NMOS transistor whereas when the input is high and the output is low, there is subthreshold leakage current through the PMOS transistor. In order to simplify our modeling, we come up with a single average leakage power number for each gate. Thus for the inverter, we calculate leakage as follows:

$$P_{\text{leak-inv}} = \frac{W_{\text{inv-pmos}}I_{\text{off-pmos}} + W_{\text{inv-nmos}}I_{\text{off-nmos}}}{2} \tag{88}$$

where $I_{\text{off-pmos}}$ is the subthreshold current per unit width for the PMOS transistor and $I_{\text{off-nmos}}$ is the subthreshold current per unit width for the NMOS transistor.

Figure 35 illustrates the leakage power calculation for a NAND2 gate. When both inputs are high, the output is low and for this condition there is leakage through the PMOS transistors as shown. When either of the inputs is low, the output is high and there is leakage through the NMOS transistors. Because of the stacked NMOS transistors [29][24], this leakage depends on which input(s) is low. The leakage is least when both inputs are low. Under standby operating conditions, for NAND2 and NAND3 gates in the decoding logic within the mats, we assume that the output of each NAND is high (deactivated) with both of its inputs low. Thus we attribute a leakage number to the NAND gate based on the leakage through its stacked NMOS transistors when both inputs are low. We consider the reduction in leakage due to the effect of stacked transistors and calculate leakage for the NAND2 gate as follows:

$$P_{\text{leak-nand2}} = W_{\text{inv-nmos}}I_{\text{off-nmos}}SF_{\text{nand2}} \tag{89}$$

where SF_{nand2} is the stacking fraction for reduction in leakage due to stacking.

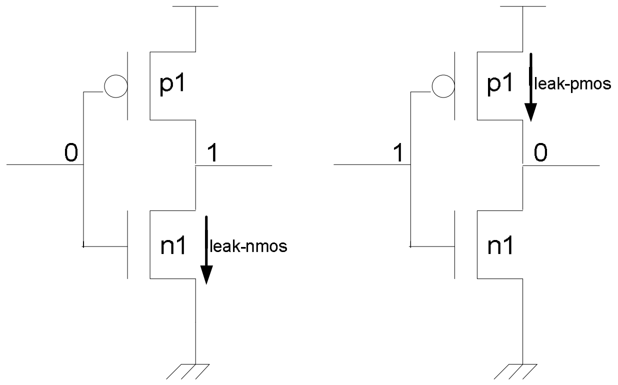


Figure 34: Leakage in an inverter.

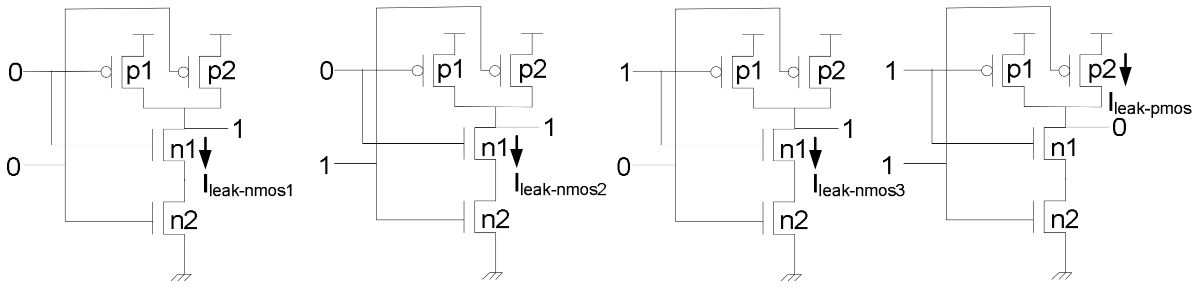


Figure 35: Leakage in a NAND2 gate.

7.2.2 Leakage Power Equations

Most of the leakage power equations are similar to the dynamic energy equations in form.

$$
\begin{aligned}
P_{\text{leak}} &= P_{\text{leak-request-network}} + P_{\text{leak-mats}} + P_{\text{leak-reply-network}} && (90)\\
P_{\text{leak-mats}} &= (P_{\text{leak-mem-cells}} + P_{\text{leak-predec-blks}} + P_{\text{leak-decoder-drivers}} + \\
&\quad P_{\text{leak-senseamps}})N_{\text{banks}}N_{\text{subbanks}}N_{\text{mats-in-subbank}} && (91)\\
P_{\text{leak-mem-cells}} &= N_{\text{subarr-rows}}N_{\text{subarr-cols}}P_{\text{mem-cell}} && (92)\\
P_{\text{leak-decoder-drivers}} &= P_{\text{leak-row-decoder-drivers}} + P_{\text{leak-bitmux-decoder-driver}} + \\
&\quad P_{\text{leak-senseampmux-decoder-driver}} && (93)\\
P_{\text{leak-row-decoder-drivers}} &= 4N_{\text{subarr-rows}}P_{\text{leak-row-decoder-driver}} && (94)\\
P_{\text{leak-request-network}} &= P_{\text{leak-arr-edge-to-bank-edge-request-htree}} + P_{\text{leak-bank-hor-request-htree}} + \\
&\quad P_{\text{leak-bank-ver-request-htree}} && (95)\\
P_{\text{leak-reply-network}} &= P_{\text{dyn-ver-reply-htree}} + P_{\text{dyn-hor-reply-htree}} + P_{\text{dyn-bank-edge-to-arr-edge-reply-htree}} && (96)
\end{aligned}
$$

Figure 36 shows the subthreshold leakage paths in an SRAM cell when it is in idle/standby state [29][24]. The leakage power contributed by a single memory cell may be given by:

$$
\begin{aligned}
P_{\text{mem-cell}} &= \text{VDD}I_{\text{mem-cell}} && (97)\\
I_{\text{mem-cell}} &= I_{\text{p1}} + I_{\text{n2}} + I_{\text{n3}} && (98)\\
I_{\text{p1}} &= W_{\text{p1}}I_{\text{off-pmos}} && (99)\\
I_{\text{n2}} &= W_{\text{n2}}I_{\text{off-nmos}} && (100)\\
I_{\text{n3}} &= W_{\text{n2}}I_{\text{off-nmos}} && (101)
\end{aligned}
$$

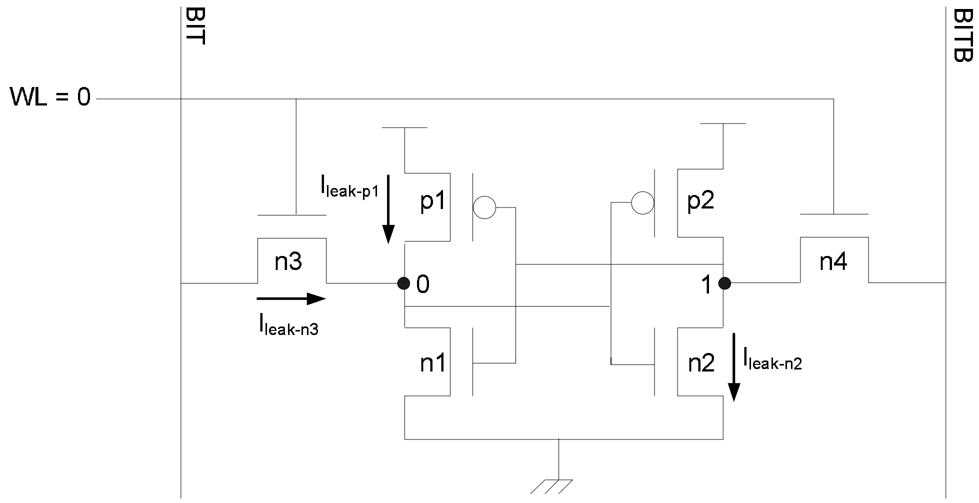


Figure 36: Leakage paths in a memory cell in idle state. BIT and BITB are precharged to VDD.

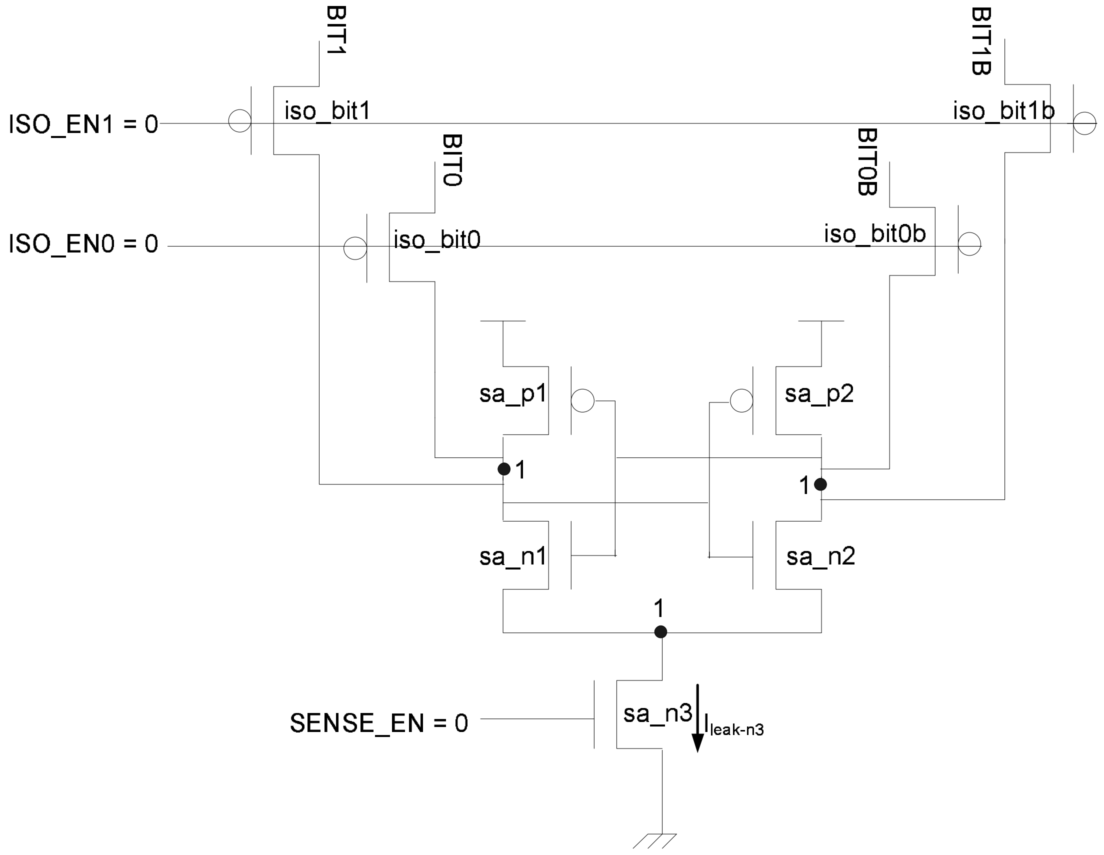


Figure 37: Leakage paths in a sense amplifier in idle state.

Figure 37 shows the subthreshold leakage paths in a sense amplifier during an idle/standby cycle [29][24].

8 Technology Modeling

Version 5.0 makes use of technology projections from the ITRS [7] for device data and projections from [6][12] for wire data. Currently we look at four ITRS technology nodes (we use MPU/ASIC metal 1 half-pitch to define the technology node) – 90, 65, 45 and 32 nm – which cover years 2004 to 2013 in the ITRS. Section 8.1 gives more details about the device data and modeling and Section 8.2 gives more details about the wire data and modeling.

8.1 Devices

Table 3 shows the characteristics of transistors modeled by the ITRS that are incorporated within CACTI. We include data for the three device types that the ITRS defines - High Performance (HP), Low Standby Power (LSTP) and Low Operating Power (LOP). The HP transistors are state-of-the-art fast transistors with short gate lengths, thin gate oxides, low V_{th} and low VDD whose CV/I is targeted to improve by 17% every year. As a consequence of their high on-currents, these transistors tend to be very leaky. The LSTP transistors on the other hand are transistors with longer gate lengths, thicker gate oxides, higher V_{th} and higher VDD. The gate-lengths of the LSTP transistors lag the HP transistors by 4 years. The LSTP transistors trade off high on-currents for maintenance of an almost constant low leakage of 10 pA across the technology nodes. The LOP transistors have performance that lie in between the HP and LSTP transistors. They use the lowest VDD to control the operating power and their gate-lengths lag those of HP transistors by 2 years. The CV/I of the LSTP and LOP transistors improves by about 14% every year.

Parameter	Meaning	Units
VDD	Voltage applied between drain and source, gate and source	V
L_{gate}	physical length of the gate	micron
V_{th}	Saturation threshold voltage	V
M_{eff}	Effective mobility	cm^2/Vs
V_{dsat}	Drain saturation voltage	V
$C_{\text{ox-elec}}$	Capacitance of gate-oxide in inversion	F/μ^2
$C_{\text{gd-overlap}}$	Gate to drain overlap capacitance	F/μ
$C_{\text{gd-fringe}}$	Gate to drain fringing capacitance	F/μ
$C_{\text{j-bottom}}$	Bottom Junction capacitance	F/μ^2
$C_{\text{j-sidewall}}$	Sidewall junction capacitance	F/μ^2
I_{on}	On-current (saturation)	A/μ
I_{off}	Channel leakage current (for $V_{\text{gate}} = 0$ and $V_{\text{drain}} = \text{VDD}$)	A/μ

Table 3: Technology characteristics of transistors used in the model.

Technology-node	90 nm	65 nm	45 nm	32 nm
L_{gate} (nm)	37/75/53	25/45/32	18/28/22	13/20/16
EOT (Equivalent oxide thickness) (nm)	1.2/2.2/1.5	1.1/1.9/1.2	0.65/1.4/0/9	0.5/1.1/0.8
VDD (V)	1.2/1.2/0.9	1.1/1.2/0.8	1/1.1/0.7	0.9/1/0.7
V_{th} (mV)	237/525/318	195/554/315	181/532/256	137/513/242
I_{on} ($\mu A/\mu$)	1077/465/550	1197/519/573	2047/666/749	2496/684/890
I_{off} (nA/μ)	32.4/8E-3/2.0	196/9E-3/4.9	280/10E-3/4.0	1390/21E-3/65
$C_{\text{ox-elec}}$ (fF/μ^2)	17.9/12.2/16.0	18.8/13.6/18.7	37.7/20.1/28.2	45.8/22.9/31.2
τ (Intrinsic switching delay) (ps)	1.01/2.98/1.78	0.64/1.97/1.17	0.4/1.33/0.79	0.25/0.9/0.53
FO1 delay (ps)	7.3/25.1/19.9	4.8/18.1/10.0	2.75/11.5/6.2	1.63/7.13/3.51

Table 4: Values of key technology metrics of HP, LSTP and LOP NMOS transistors for four technology-nodes from the 2005 ITRS[7].

Table 4 shows values of key technology metrics of the HP, LSTP and LOP NMOS transistors for four technology nodes. The data is obtained from MASTAR [9] files. According to the 2003 ITRS[2], the years 2004, 2007, 2010 and 2013 correspond to 90, 65, 45 and 32 nm technology-nodes. Because the 2005 ITRS does not include device data for the 90 nm technology-node (year 2004), we obtain this data using MASTAR and targeting the appropriate CV/I. Note that all values shown are for planar bulk devices. The ITRS actually makes the assumption that planar high-performance bulk devices reach their limits of practical scaling in 2012 and therefore includes multiple parallel paths of scaling for SOI and multiple-gate MOS transistors such as FinFETs starting from the year 2008 which run in parallel with conventional bulk CMOS scaling. We however use MASTAR to meet the target CV/I of the 32 nm node with planar bulk devices. For all technology nodes, the overlap capacitance value has been assumed to be 20% of ideal (no overlap) gate capacitance. The bottom junction capacitance value for the planar bulk CMOS transistors has been assumed to be 1fF/μ^2, which is the value that MASTAR assumes. As MASTAR does not model sidewall capacitance, we compute values for sidewall capacitance in the following manner: we use process data provided at the MOSIS website [30] for TSMC and IBM 130/180/250 nm processes and compute average of the ratios of sidewall-to-bottom junction capacitances for these processes. We observe that average error in using this average value for projecting sidewall capacitance given bottom junction capacitance is less than 10%. We use this average value in projecting sidewall capacitances for the ITRS processes.

We calculate the drive resistance of a transistor during switching as follows:

$$R_{\text{on}} = \frac{\text{VDD}}{I_{\text{eff}}} \tag{102}$$

The effective drive current is calculated using the following formula described in [31][32]:

$$I_{\text{eff}} = \frac{I_{\text{H}} + I_{\text{L}}}{2} \tag{103}$$

where $I_{\text{H}} = I_{\text{DS}}$ (V_{GS} = VDD, $V_{\text{DS}} = \frac{\text{VDD}}{2}$) and $I_{\text{L}} = I_{\text{DS}}$ ($V_{\text{GS}} = \frac{\text{VDD}}{2}$, V_{DS} = VDD).

For PMOS transistors, we find the width of the transistor that produces the same I_{off} as a unit-width NMOS transistor. Using this width, we compute the PMOS effective drive current ($I_{\text{eff-pmos}}$) and the PMOS-to-NMOS sizing ratio that is used during the application of the method of logical effort:

$$S_{\text{pmos-to-nmos-logical-effort}} = \frac{I_{\text{eff-nmos}}}{I_{\text{eff-pmos}}} \tag{104}$$

Table 5 shows technology data that we have assumed for an SRAM cell.

Parameter	Value	Reference
$A_{\text{sram-cell}}$ (Area of an SRAM cell) (μ^2)	$146F^2$	[17]
$W_{\text{sram-cell-acc}}$ (Width of SRAM cell access transistor) (μ)	$1.31F$	[17]
$W_{\text{sram-cell-pd}}$ (Width of SRAM cell pull-down transistor) (μ)	$1.23F$	[17]
$W_{\text{sram-cell-pu}}$ (Width of SRAM cell pull-up transistor) (μ)	2.08	[17]
$AR_{\text{sram-cell}}$ (Aspect ratio of the cell)	1.46	[17]

Table 5: Technology data assumed for an SRAM cell.

It may be useful to know that while currently we provide device data for just the three ITRS device types, it is not difficult to incorporate device data from other sources into CACTI. Thus, published data of various industrial fabrication processes or data from sources such as [33] may also be utilized. Also, by making use of MASTAR, it is possible to obtain device data for scaling models and assumptions that are different from those of the ITRS. As an example, while the ITRS device data for its High Performance device type is based on an improvement in device CV/I of 17 % every year, one may obtain alternative device data by targeting a different CV/I improvement and/or I_{off}. Another example is to start off with the ITRS High Performance device type and use MASTAR to come up with higher Vt or longer channel variations of the base device.

[2]Because of ambiguity associated with the "technology-node" term, the 2005 ITRS has discontinued the practice of using the term, however, for the sake of convenience, we continue to use it in CACTI.

8.2 Wires

Wire characteristics in CACTI are based on the projections made in [6][12]. The approach followed in [6][12] is to consider both aggressive (optimistic) and conservative (pessimistic) assumptions regarding interconnect technology. The aggressive projections assume aggressive use of low-k dielectrics, insignificant resistance degradation due to dishing and scattering, and tall wire aspect ratios. The conservative projections assume limited use of low-k dielectrics, significant resistance degradation due to dishing and scattering, and smaller wire aspect ratios. For these assumptions, [6][12] looks at two types of wires, semi-global and global. Wires of semi-global type have a pitch of 4F (F = Feature size) whereas wires of global type have a pitch of 8F. We incorporate the properties of both these wire types into CACTI. The values of the semi-global and global wire characteristics under aggressive and conservative assumptions are presented in Table 6 for 90/65/45/32 technology nodes. The resistance per unit length and capacitance per unit length values are calculated based off Equations 5 and 6 respectively. For the capacitance per unit micron calculation, we assume a Miller factor of 1.5 as a "realistic worst-case" value [11]. For material strength, we assume that low-k dielectrics are not utilized between wire layers as suggested in [11].

Technology-node	90 nm	65 nm	45 nm	32 nm
Common wire characteristics (aggressive/conservative)				
$\rho (m\Omega.\mu)$	0.022/0.022	0.018/0.022	0.018/0.022	0.018/0.022
$\epsilon_{r} for C_{c}$	2.709/3.038	2.303/2.734	1.958/2.46	1.664/2.214
Semi-global wire properties (aggressive/conservative)				
Pitch(nm)	360	280	180	128
Aspect ratio	2.4/2.0	2.7/2.0	3.0/2.0	3.0/2.0
Thickness (nm)	432/400	351/280	270/200	192/140
ILD (nm)	480/480	405/405	315/315	210/210
Miller factor	1.5/1.5	1.5/1.5	1.5/1.5	1.5/1.5
Barrier (nm)	10/8	0/6	0/4	0/3
Dishing (%)	0/0	0/0	0/0	0/0
$\alpha_{scatter}$	1/1	1/1	1/1	1/1
Resistance per unit length (Ω/μ)	0.33/0.38	0.34/0.73	0.74/1.52	1.46/3.03
Capacitance per unit length (fF/μ)	0.314/0.302	0.302/0.282	0.291/0.265	0.269/0.254
Global wire properties (aggressive/conservative)				
Pitch(nm)	800	560	400	280
Aspect ratio	2.7/2.2	2.8/2.2	3.0/2.2	3.0/2.2
Thickness (nm)	1080/880	784/616	600/440	420/308
ILD (nm)	960/1100	810/770	630/550	420/385
Miller factor	1.5	1.5	1.5	1.5
Barrier (nm)	10/8	0/6	0/4	0/3
Dishing (%)	0/10	0/10	0/10	0/10
$\alpha_{scatter}$	1/1	1/1	1/1	1/1
Resistance per unit length (Ω/μ)	0.067/0.09	0.095/0.17	0.19/0.36	0.37/0.72
Capacitance per unit length (fF/μ)	0.335/0.315	0.308/0.298	0.291/0.281	0.269/0.267

Table 6: Aggressive and conservation wire projections from [6].

8.3 Technology Exploration

As an additional feature in version 5.0, we allow the user to map different device and wire types to different parts of the array. We divide the devices in the array into two parts: one, devices used in the memory cells and wordline drivers, and two, the rest of the peripheral and global circuitry. Different device types such as the ITRS HP, LSTP, LOP or other user-added device types may be mapped to the devices in the two parts of the array. [3] We divide the wires in the array also into two parts, wires inside mats and wires outside mats. Different wire types such as the semi-global or global wire types or other user-defined wire types may be mapped to the wires inside and outside mats.

9 Embedded DRAM Modeling

In this section, we describe our modeling of embedded DRAM.

9.1 Embedded DRAM Modeling Philosophy

We model embedded DRAM and assume a logic-based embedded DRAM fabrication process [13][14][15]. A logic-based embedded DRAM process typically means that DRAM has been embedded into the logic process without affecting the characteristics of the original process much [37]. In our modeling of embedded DRAM, we leverage the similarity that exists in the global and peripheral circuitry of embedded SRAM and DRAM and model only their essential differences. We also use the same array organization for embedded DRAM that we used for SRAM. By having a common framework that, in general, places embedded SRAM and DRAM on an equal footing and emphasizes only their essential differences, we would be able to compare relative tradeoffs involving embedded SRAM and DRAM.

We capture the following essential differences between embedded DRAM and SRAM in our area, delay and power models:

9.1.1 Cell

The most essential difference between SRAM and DRAM is in their storage cell. While SRAM typically makes use of a 6T cell and the principle of positive feedback to store data, DRAM typically makes use of a 1T-1C cell and relies on the charge-storing capability of a capacitor. Because it makes use of only one transistor, a DRAM cell is usually laid out in a much smaller area compared to an SRAM cell. For instance the embedded DRAM cells presented in [38] for four different technology nodes – 180/130/90/65 nm have areas in the range of 19–26F^2 where F is the feature size of the process. In contrast, a typical SRAM cell would have an area of about 120–150F^2.

9.1.2 Destructive Readout and Writeback

When data is read out from a DRAM cell, the charge stored in the cell gets destroyed because of charge redistribution between the cell and its capacitive bitline. Because of the destructive readout, there is a need for data to be written back into the cell after every read access. This writeback takes time and increases the random cycle time of a DRAM array. In an SRAM there is no need for writeback because the data is not destroyed during a read.

9.1.3 Sense amplifier Input Signal

In a DRAM, the maximum differential signal that is developed on the bitlines is limited by the amount of charge transferred between the DRAM cell and the bitline which in turn depends on the capacitance of the DRAM cell and the bitline. The lower the differential signal, the greater the sense amplifier delay. In an SRAM, there is no charge-based limit on the differential signal developed on the bitlines. In any case, in modern technologies the sense amplifiers of SRAMs or DRAMs are operating at signal level inputs of more or less the same amplitude [37], so the delay of the sense amplifier in either SRAM or DRAM can come out to have similar values.

[3]It is important to note that in reality, SRAM cell functionality and design does depend on device type [34][35][36], however, we do not model different SRAM cell designs for the different device types.

9.1.4 Refresh

In a DRAM cell, charge cannot be stored for an infinite time in the capacitor and the charge leaks out because of various leakage components. If charge from a DRAM cell is allowed to leak out for a sufficient period of time, the differential voltage developed on the bitline pair becomes so small that the data stored in the cell can no longer be detected by the sense amplifier. Thus there is an upper bound on the time for which data may be retained in a DRAM cell without it being refreshed, and this time is known as the retention time. Because of a finite retention time, the DRAM cell needs to be refreshed periodically.

9.1.5 Wordline Boosting

In a DRAM cell, because the access takes place through an NMOS pass transistor, there is a V_{TH} drop during the write/writeback of a 1 into the cell. In order to prevent this V_{TH} drop, DRAM wordlines are usually boosted to a voltage, VPP = VDD + V_{th}. In commodity DRAMs, V_{th} is relatively high in order to maintain the high refresh period (64 ms) that requires extremely low leakage. This means that VPP is also high and forces the use of high voltage (thicker gate-oxide) slower transistors in the wordline driver. For the embedded DRAMs that we have modeled, however, V_{TH} is not very high, consequently VPP is also not very high.

9.2 DRAM Array Organization and Layout

For DRAM, we assume a folded array architecture [39] in the subarray, shown in Figure 38. In the folded array architecture, the bitline that is being read (true bitline) and its complement are laid out next to each other, similar to the dual bitlines of an SRAM cell. The difference here is that the true and complement bitlines connect to alternate rows of the array and not to the same row as in SRAM. This has an impact on bitline capacitance calculation. Assuming drain contacts are shared, the bitline capacitance for DRAM may be given by the following equation:

$$C_{\text{bitline}} = \frac{N_{\text{subarr-rows}}}{2} C_{\text{drain-cap-acc-transistor}} + N_{\text{subarr-rows}} C_{\text{bit-metal}} \tag{105}$$

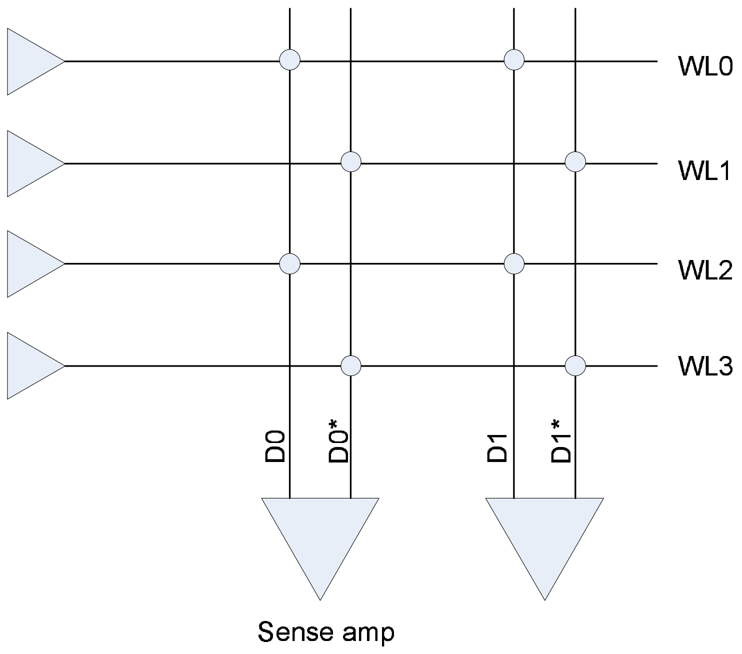


Figure 38: Folded array architecture from [39].

9.2.1 Bitline Multiplexing

In DRAM, the read access is destructive. This means that during a read access after data is read from a DRAM cell, it needs to be written back into the cell. This writeback is typically accomplished by using the sense amplifier which detects the data stored in the cell during a read. During a read access, because each cell that is connected to a wordline is read out through its associated bitline, this means that there needs to be a sense-amplifier associated with each cell that is connected to a wordline. Hence bitline multiplexing, which is common is SRAMs to connect multiple bitlines to a single sense amplifier, is not feasible in DRAMs. Thus in DRAMs, there needs to be a sense amplifier associated with every bitline that can carry out the writeback. With respect to the bitline peripheral circuitry shown in Figure 7 this means that DRAM arrays do not have a bitline mux between the bitlines and sense amplifiers.

9.2.2 Reference Cells for VDD Precharge

We assume that the bitlines are precharged to VDD (GND) just like the DRAM described in [40][15]. As in [40], we assume the use of reference cells that store VDD/2 and connect to the complement bitline during a read. Figure 39 shows the bitline peripheral circuitry with the reference cells. For each subarray, we assume an extra two rows of reference cells that store VDD/2. One of the rows with reference cells is activated during read of even-numbered rows in the subarray and the other row is activated during read of odd-numbered rows in the subarray.

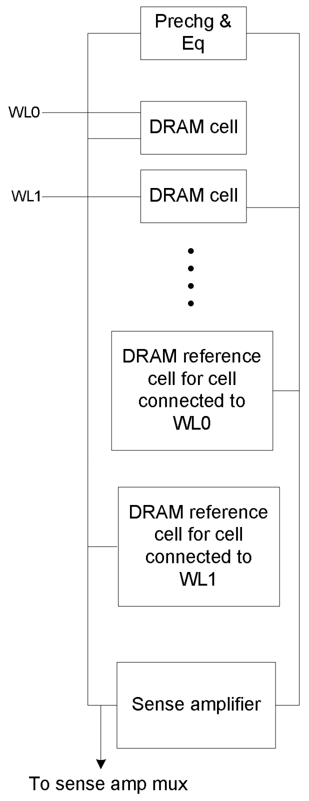

Prechg & Eq
WL0
DRAM cell
WL1
DRAM cell
DRAM reference cell for cell connected to WL0
DRAM reference cell for cell connected to WL1
Sense amplifier
To sense amp mux

Figure 39: DRAM bitline circuitry showing reference cells for VDD precharge.

9.3 DRAM Timing Model

9.3.1 Bitline Model

In DRAM the differential voltage swing developed on a bitline pair that acts as input to the sense amplifier is limited by the ratio of charge transferred between the bitline and DRAM cell, and given by the following equation:

$$V_{\text{sense-max}} = \frac{\text{VDD}}{2} \frac{C_{\text{dram}}}{C_{\text{dram}} + C_{\text{bitline}}} \tag{106}$$

The delay for the above differential signal to develop may be given by the following equation [41] (ignoring the effect of wordline rise time):

$$T_{\text{step}} = 2.3 R_{\text{dev}} \frac{C_{\text{dram}} C_{\text{bitline}}}{C_{\text{dram}} + C_{\text{bitline}}} \tag{107}$$

where R_{dev} is the resistance in series with the storage capacitor of the DRAM cell and may be given by the following equation:

$$R_{\text{dev}} = \frac{\text{VDD}}{I_{\text{cell-on}}} \tag{108}$$

It is important to note that use of Equations 107 and 108 assumes that the impact of bitline resistance on signal development time is negligible. This approximation works well for contemporary logic-based embedded DRAM processes. When bitline resistance becomes significant, as in the case of commodity DRAM processes that do not make use of copper bitlines, more sophisticated models need to be used.

Equation 107 assumes that 90% of the data stored in the cell is read out and corresponds to the development of approximately $V_{\text{sense-max}}$ (given by Equation 106) on the bitline pair. In order to improve the random cycle time of a DRAM macro further, nowadays less than 90% of the data stored in a cell is read out [42], just enough to generate the required input signal of the sense amplifier ($V_{\text{senseamp-input}}$). To accomodate this case, Equation 107 may be generalized as follows:

$$T_{\text{step-generalized}} = 2.3 R_{\text{dev}} \frac{C_{\text{dram}} C_{\text{bitline}}}{C_{\text{dram}} + C_{\text{bitline}}} \frac{V_{\text{senseamp-input}}}{V_{\text{sense-max}}} \tag{109}$$

When $V_{\text{senseamp-input}}$ is equal to $V_{\text{sense-max}}$, Equation 109 reduces to Equation 107. In CACTI, we assume a certain value for $V_{\text{senseamp-input}}$ (such as 80 mV) and use Equation 109 to compute the signal development delay.

When rise time of the wordline is also considered, the bitline delay (T_{bitline}) of DRAM may be calculated using the same methodology that was used for SRAM (Equation 52 in Section 6).

The time taken to write data back into a DRAM cell after a read depends on the time taken for the charge transfer to take place between the bitline and the DRAM and thus may be given by the following equation:

$$T_{\text{writeback}} = T_{\text{step}} \tag{110}$$

9.3.2 Multisubbank Interleave Cycle Time

For a DRAM array, we consider three timing characteristics: random access time, random cycle time and multibank interleave cycle time.

Calculation of random access time makes use of the same equations that were used for calculation of random access time of an SRAM array (in Section 6).

For a DRAM array, typically there are two kinds of cycle time: random cycle time and multibank interleave cycle time. Random cycle time has the same meaning as the random cycle time of an SRAM viz. it is the time interval between two successive random accesses. This time interval is typically limited by the time it takes to activate a wordline, sense the data, write back the data and then precharge the bitlines. Random cycle time can thus be calculated using the following equation:

$$T_{\text{random-cycle}} = T_{\text{row-dec-driver}} + T_{\text{bitline}} + T_{\text{sense-amp}} + T_{\text{writeback}} + T_{\text{wordline-reset}} + \max(T_{\text{bitline-precharge}}, T_{\text{bit-mux-out-precharge}}, T_{\text{senseamp-mux-out-precharge}}) \quad (111)$$

In order to improve the rate at which a DRAM array is accessed so that it is not limited by the random cycle time of the array, DRAM arrays usually employ the concept of multibank interleaving. Multibank interleaving takes advantage of the fact that while random access to a particular bank is limited by the random cycle time, accesses to other banks need not be. With multibank interleaving, accesses to multiple DRAM banks that are on the same address/data bus are interleaved at a rate defined by the *multibank interleave cycle time*. In our terminology, each *bank* in an array has its own address and data bus and may be concurrently accessed. For our array organization, the concept of multibank interleaved mode is relevant to subbank access and not bank access, so in the rest of this discussion we use the terminology of multisubbank interleave mode and multisubbank interleave cycle. Thus, the multisubbank interleave cycle time is the rate at which accesses may be interleaved between different subbanks of a bank. The multisubbank interleave cycle time depends on the degree of pipelining employed in the request and reply networks of a subbank, and is limited by the pipelining overhead. We assume minimum pipeline overhead and use the following simple equation to calculate multisubbank-interleave cycle time:

$$T_{\text{multisubbank-interleave}} = \max(T_{\text{request-network}} + T_{\text{row-predec}}, T_{\text{reply-network}}) \quad (112)$$

9.3.3 Retention Time and Refresh Period

An equation for the retention time of a DRAM array may be written as follows [43]:

$$T_{\text{retention}} = \frac{C_{\text{dram-cell}} \Delta V_{\text{cell-worst}}}{I_{\text{worst-leak}}} \quad (113)$$

where $\Delta V_{\text{cell-worst}}$ is the worst-case change in the voltage stored in a DRAM cell which leads to a read failure, and $I_{\text{cell-worst-leak}}$ is the worst-case leakage in a DRAM cell.

We assume that $\Delta V_{\text{cell-worst}}$ is limited by $V_{\text{min-sense}}$, the minimum input signal that may be detected by the bitline sense amplifier. Thus, for a given array organization, $\Delta V_{\text{cell-worst}}$ may be calculated by solving the following equation for $\Delta V_{\text{cell-worst}}$:

$$V_{\text{min-sense}} = \frac{C_{\text{dram-cell}}}{C_{\text{dram-cell}} + C_{\text{bitline}}} \left(\frac{VDD}{2} - \Delta V_{\text{cell-worst}}\right) \quad (114)$$

If we assume that the differential voltage detected by the sense amplifier is independent of array organization, then this means that different array partitions would have different retention times depending on the charge transfer ratio between the DRAM cell and the bitlines. For each array organization, it's thus possible to calculate the value for $\Delta V_{\text{cell-worst}}$ using Equation 114, which may then be plugged into Equation 113 to find the retention time for that array organization.

The upper bound on the refresh period of a DRAM cell would be equal to its retention time. We assume that a safety margin of 10% with respect to the retention time is built into the refresh period and thus calculate the refresh period using the following equation:

$$T_{\text{refresh}} = 0.9 T_{\text{retention}} \quad (115)$$

9.4 DRAM Power Model

During the read of a 0 from a DRAM cell, the true bitline is pulled down to GND during the writeback. Energy is then consumed in restoring the bitline to VDD during the precharge operation. During the read of a 1 from a DRAM cell, because of our assumption of VDD-precharge, the voltage of the true bitline does not change but the voltage of the complementary bitline gets pulled down to GND and needs to be restored to VDD. So for DRAM, the power consumed in a bitline during a read may be approximated by the following equation:

$$E_{\text{dyn-read-bitline}} = C_{\text{bitline}} VDD^2 \tag{116}$$

9.4.1 Refresh Power

Refreshing the data in each cell of the array consumes power. In order to carry out refresh, every cell in the array needs to be accessed, its data read out, and then written back.

$$\begin{aligned}
P_{\text{refresh}} &= \frac{E_{\text{refresh}}}{T_{\text{refresh}}} && (117)\\
E_{\text{refresh}} &= E_{\text{refresh-predec-blks}} + E_{\text{refresh-row-dec-drivers}} + E_{\text{refresh-bitlines}} && (118)\\
E_{\text{refresh-predec-blks}} &= N_{\text{banks}} N_{\text{subbanks}} N_{\text{mats-in-subbank}} E_{\text{dyn-mat-predec-blks}} && (119)\\
E_{\text{refresh-row-dec-drivers}} &= 4 N_{\text{banks}} N_{\text{subbanks}} N_{\text{mats-in-subbank}} E_{\text{dyn-mat-row-dec-drivers}} && (120)\\
E_{\text{refresh-bitlines}} &= 4 N_{\text{banks}} N_{\text{subbanks}} N_{\text{mats-in-subbank}} N_{\text{subarr-cols}} E_{\text{dyn-read-bitline}} && (121)
\end{aligned}$$

9.5 DRAM Area Model

9.5.1 Area of Reference Cells

As mentioned earlier in Section 9.2.2, the use of VDD-precharge leads to the use of reference cells in the array [40]. For our array organization, this means that there are two additional wordlines per subarray.

9.5.2 Area of Refresh Circuitry

In order to enable continued scaling of a logic-based embedded DRAM cell in terms of performance and cell area, [38] describes a new scalable embedded DRAM cell that makes use of an access transistor with an intermediate gate-oxide of moderate thickness (2.2 nm for 90/65 nm). This transistor is a standard offering in the logic process which incorporates the embedded DRAM. Conventional cells [17] in earlier technologies made use of access transistors with much thicker gate-oxides. An effect of the scalable embedded DRAM cell described in [38] is that it results in the cell having a lower retention time and a lower refresh period (because of higher worst-case leakage - 10s of pAs compared to 1 fA for commodity DRAM). The macro discussed in [44] that makes use of the cell described in [38] has a refresh period of 64 μs compared to conventional macros which have refresh period of 64 ms. This low refresh period required innovation at the circuit level through the development of a concurrent refresh scheme described in [44] in order to guarantee high availability of the DRAM macro. This concurrent refresh scheme adds an extra bank select port to each bank (subbank in our terminology) thereby allowing for concurrent memory access and bank refresh operations in different banks. Each bank is equipped with a row address counter that contains the address of the row to be refreshed. A concurrent refresh scheduler composed of an up-shift-register and a down-shift-register is required in order to generate the bank select signals.

Because we loosely base the parameters of our logic-based embedded DRAM technology on information presented in [38][40][44], we model the overhead of the concurrent refresh scheme on area. For our organization in which each subbank has multiple mats, we assume that each mat incurs overhead of a row address counter placed at the center of the mat. Because of the requirements of the design of the concurrent refresh scheme, for our organization, we assume $N_{\text{subbanks-in-mat}}$ number of concurrent refresh schedulers per bank.

9.6 DRAM Technology Modeling

9.6.1 Cell Characteristics

Similar to the SRAM technology assumptions, we assume two types of transistors in the DRAM array. One transistor type is used in the DRAM cell and wordline driver, while the other is used in the rest of the peripheral and global circuitry. Table 3 showed a list of transistor characteristics that are used in CACTI. Table 7 shows characteristics of the DRAM cell and wordline driver that we consider in our model.

Parameter	Meaning	Unit
C_{dram}	Storage capacitance of a DRAM cell	F
$A_{\text{dram-cell}}$	Area occupied by the DRAM cell	mm^2
$AR_{\text{dram-cell}}$	Aspect ratio of the DRAM cell	
$VDD_{\text{dram-cell}}$	Voltage representing a 1 in a DRAM cell	V
$V_{\text{th-dram-acc-transistor}}$	Threshold voltage of DRAM cell access transistor	mV
$L_{\text{dram-acc-transistor}}$	Length of DRAM cell access/wordline transistor	nm
$W_{\text{dram-acc-transistor}}$	Width of DRAM cell access transistor	nm
$EOT_{\text{dram-acc-transistor}}$	Equivalent oxide thickness of DRAM access transistors	nm
$I_{\text{on-dram-cell}}$	DRAM cell on-current under nominal conditions	μA
$I_{\text{off-dram-cell}}$	DRAM cell off-current under nominal conditions	pA
$I_{\text{worst-off-dram-cell}}$	DRAM cell off-current under worst-case conditions	A/μ
VPP	Boosted wordline voltage applied to gate of access transistor	V
$I_{\text{on-dram-wordline-transistor}}$	On-current of wordline transistor	$\mu A/\mu$

Table 7: Characteristics of the DRAM cell and wordline driver.

Parameter	90 nm	65 nm
C_{dram} (F)	20	20
$A_{\text{dram-cell}}$ (F - Feature size)	$20.7F^2$	$25.6F^2$
$VDD_{\text{dram-cell}}$	1.2	1.2
$V_{\text{th-dram-acc-transistor}}$	350	350
$L_{\text{dram-acc-transistor}}$ (nm)	120	120
$W_{\text{dram-acc-transistor}}$	140	90
$I_{\text{on-dram-cell}}$ (μA)	45	36
$I_{\text{off-dram-cell}}$ (pA)	2	2
VPP	1.5	1.5

Table 8: DRAM technology data for 90 nm and 65 nm from [38][44].

We obtain embedded DRAM technology data for four technology nodes – 90, 65, 45 and 32 nm – by using an approach that makes use of published data, transistor characterization using MASTAR and our own scaling projections. For 90 nm and 65 nm, we use technology data from [38][44]; Table 8 shows this data. We obtain the transistor data by using MASTAR with input data from Table 8. In order to obtain technology data for the 45 nm and 32 nm technology nodes, we make the following scaling assumptions:

1. Capacitance of the DRAM cell is assumed to remain fixed at 20 fF;

2. The nominal off-current is assumed to remain fixed at 2 pA for the cell;

3. Gate oxide thickness is scaled slowly in order to keep gate leakage low and subthreshold current as the dominant leakage current. It has a value of 2.1 nm for 45 nm and 2 nm for 32 nm;

4. $VDD_{\text{dram-cell}}$ is scaled such that the electric field in the dielectric of the DRAM ($VPP/EOT_{\text{dram-acc-transistor}}$) access transistor remains almost constant;

5. There is excellent correlation in the 180–130 nm (for conventional thick-oxide device) and 90–65 nm (for the intermediate-oxide device) scaling-factors for width and length of the DRAM cell access transistor. We assume that there would be good correlation in the 130–90 nm and 65–45 nm scaling-factors as well. For 32 nm, we assume that the width and length are scaled in the same proportion as feature size;

6. We calculate area of the DRAM cell using the equation $A_{\text{dram-cell}} = 10W_{\text{dram-acc-transistor}}L_{\text{dram-acc-transistor}}$. This equation has good correlation with the actual cell area of the 90 and 65 nm cells that made use of the intermediate-oxide based devices; and

7. We simply assume that nominal on-current of the cell can be maintained at the 65 nm value. This would require aggressive scaling of the series parasitic resistance of the transistor.

The transistor data that is obtained with these scaling assumptions is input to MASTAR and transistor data is obtained It is assumed that the resulting channel doping concentrations that are calculated by MASTAR would be feasible. Table 9 shows the characteristics of the transistor used in the DRAM cell and wordline driver that we have used for the four technology nodes.

Parameter	90 nm	65 nm	45 nm	32 nm
C_{dram} (F)	20	20	20	20
$A_{\text{dram-cell}}$ (F - Feature size)	$20.7F^2$	$25.6F^2$	$30.4F^2$	$30.6F^2$
$VDD_{\text{dram-cell}}$ (V)	1.2	1.2	1.1	1.1
$V_{\text{th-dram-acc-transistor}}$ (mV)	455	438	446	445
$L_{\text{dram-acc-transistor}}$ (nm)	120	120	78	56
$W_{\text{dram-acc-transistor}}$ (nm)	140	90	79	56
$I_{\text{on-dram-cell}}$ (μA)	45	36	36	36
$I_{\text{off-dram-cell}}$ (pA)	2	2	2	2
$I_{\text{worst-off-dram-cell}}$ (pA)	21.1	19.6	19.5	18.9
VPP (V)	1.6	1.6	1.5	1.5
$I_{\text{on-dram-wordline-transistor}}$ ($\mu A/\mu$)	45	36	36	36

Table 9: Values of DRAM cell and wordline driver characteristics for the four technology nodes.

10 Cache Modeling

In this section we describe how a cache has been modeled in version 5.0. The modeling methodology is almost identical to earlier versions of CACTI with a few changes that simplify the code.

10.1 Organization

As described in [1], a cache has a tag array in addition to a data array. In earlier versions of CACTI the data and tag arrays were modeled separately with separate code functions even though the data and tag arrays are structurally very similar. The essential difference between the tag array and the data array is that the tag array includes comparators that compare the input tag bits with the stored tags and produce the tag match output bits. Apart from the comparators, the rest of the peripheral/global circuitry and memory cells are identical for data and tag arrays. In version 5.0, we leverage this similarity between the data and tag arrays and use the same set of functions for their modeling. For the tag array, we reuse the comparator area, delay and power models.

Figure 40 illustrates the organization of a set-associative tag array. Each mat includes comparators at the outputs of the sense amplifiers. These comparators compare the stored tag bits with the input tag bits and produce the tag match output bits. These tag match output signals are the way-select signals that serve as inputs to the data array. The way-select signals traverse over the vertical and horizontal H-trees of the tag array to get to the edge of the tag array from where they are shipped to the data array. For a cache of normal access type, these way-select signals then enter the data array where, like the address and datain signals, they travel along the horizontal and vertical H-tree networks to get to mats in the accessed subbank. At the mats, these way-select signals are 'anded' with sense amplifier mux decode signals (if any) and the resultant signals serve as select signals for the sense amplifier mux which generates the output word from the mat.

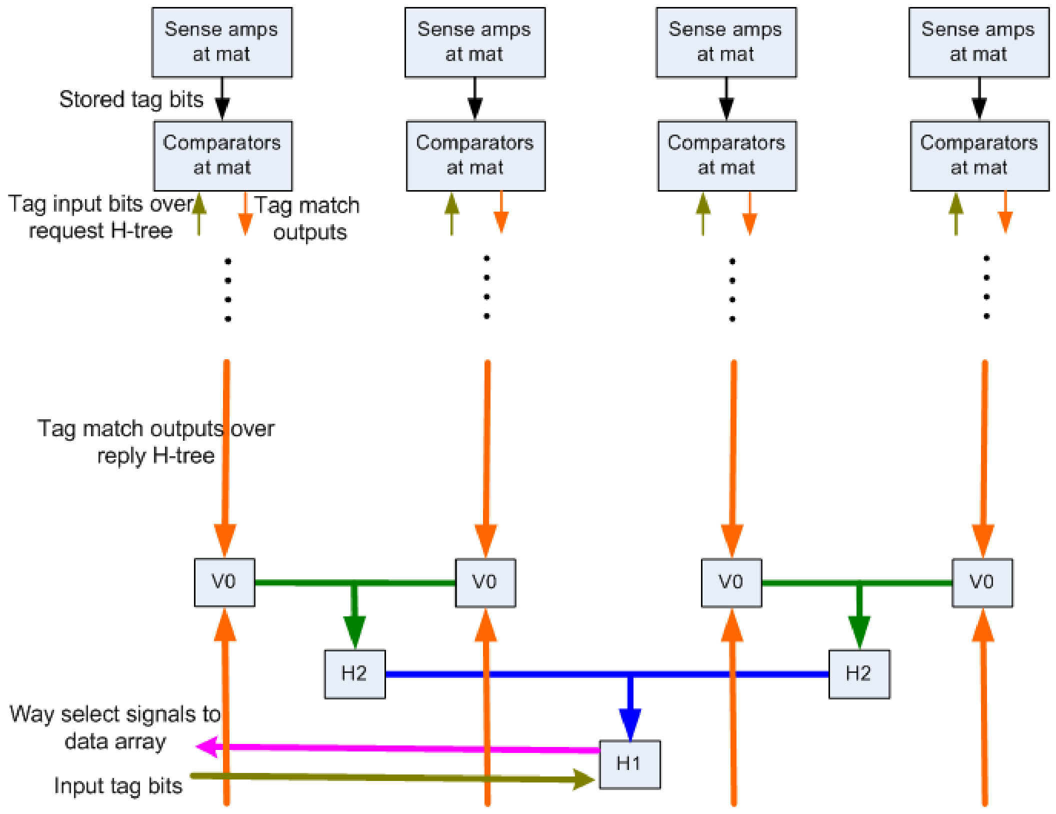


Figure 40: Organization of a set-associative tag array.

10.2 Delay Model

We present equations for access and cycle times of a cache. The access time of a cache depends on the type of cache access (normal, sequential or fast [4]).

The equation for access time of a normal cache which is set-associative is as follows:

$$\begin{aligned} T_{\text{access-normal-set-associative}} &= \max(T_{\text{tag-arr-access}} + T_{\text{data-arr-request-network}} + T_{\text{data-arr-sense-amp-mux-decode}}, \\ &\quad T_{\text{data-arr-request-network}} + T_{\text{data-arr-mat}}) + T_{\text{data-arr-reply-network}} && (122) \\ T_{\text{tag-arr-access}} &= T_{\text{tag-arr-request-network}} + T_{\text{tag-arr-mat}} + T_{\text{tag-arr-reply-network}} && (123) \end{aligned}$$

In the above equation, $T_{\text{tag-arr-access}}$, the access time of the tag array, is calculated using the following equation.

$$T_{\text{tag-arr-access}} = T_{\text{tag-arr-request-network}} + T_{\text{tag-arr-mat}} + T_{\text{tag-arr-reply-network}} + T_{\text{comparators}} \quad (124)$$

The equation for access time of a normal cache which is direct-mapped is as follows:

$$T_{\text{access-normal-direct-mapped}} = \max(T_{\text{tag-arr-access}}, T_{\text{data-arr-access}}) \quad (125)$$

The equation for access time of a sequentially accessed (tag array is accessed first before data array access begins) cache is as follows:

$$T_{\text{access-sequential}} = T_{\text{tag-arr-access}} + T_{\text{data-arr-access}} \quad (126)$$

The equation for access time of a 'fast' cache (late way-select multiplexing) cache is as follows:

$$T_{\text{access-fast}} = \max(T_{\text{tag-arr-access}}, T_{\text{data-arr-access}}) + T_{\text{way-select-mux}} \quad (127)$$

where $T_{\text{way-select-mux}}$ is the delay through the way-select mux. The way-select mux is assumed to be placed at the edge of the data array and selects the appropriate output word corresponding to the correct way (which is selected based on the way-select signals from the tag array).

10.3 Area Model

Total area of the cache is calculated by simply adding the areas occupied by the tag and data arrays.

$$A_{\text{cache}} = A_{\text{data-array}} + A_{\text{tag-array}} \quad (128)$$

$A_{\text{tag-array}}$ is calculated using the equations presented in Section 5 with the area of the comparators also added.

10.4 Power Model

The dynamic energy consumed in the cache and its standby leakage power are calculated by simply adding their values for the data and tags arrays. For the tag array, the leakage in the comparators is also considered.

$$E_{\text{dyn-energy-cache}} = E_{\text{dyn-energy-data-array}} + E_{\text{dyn-energy-tag-array}} \quad (129)$$

$$P_{\text{leak-cache}} = P_{\text{leak-data-array}} + P_{\text{leak-tag-array}} \quad (130)$$

11 Quantitative Evaluation

In this section we evaluate the impact of new CACTI 5.0 features. We also compare results from CACTI 5.0 with version 4.2 in order to give users an idea of what to expect when upgrading to CACTI 5.0.

11.1 Evaluation of New CACTI 5.0 Features

Table 10 shows the default parameters that we have used while carrying out the evaluation of new CACTI 5.0 features. For this evaluation we use plain RAMs instead of caches. In the next section on validation of CACTI, we show results for caches. For each study, we present charts that show the following metrics: access time, random cycle time, area, dynamic energy per read access and standby leakage power.

11.1.1 Impact of New CACTI Solution Optimization

Figure 41 shows the impact of varying maxareaconstraint (that was described in Section 2.5.1) for a 16MB SRAM. As maxareaconstraint is increased, the number of subarrays in the SRAM is allowed to grow, and so the area grows steadily. As the number of subarrays increases, the components of delay within a mat decreases and the access time falls up to a point after which it starts to increase again. The random cycle time keeps decreasing as the number of subarrays increases because the wordline and bitline delays keep getting smaller. The trend for dynamic read energy per access shows some up-and-down variation. For our assumed CACTI cache organization, an increase in Ndwl typically increases the dynamic energy consumption because more wordlines are activated per read access, while an increase in Ndbl typically decreases the dynamic energy consumption because of reduction in bitline power. The standby leakage power keeps growing as the area of the RAM increases.

Figure 42 shows the impact of varying 'maxacctimeconstraint'. For the assumed set of SRAM parameters, it can be seen that the solution with best access time (corresponding to maxacctimeconstraint of 0) also has

Parameter	Value
Capacity (MB)	16
Output width (bits)	512
Number of banks	1
Number of read/write ports	1
Number of exclusive read ports	0
Number of exclusive write ports	0
Technology-node (nm)	65
DRAM	No
maxareaconstraint	40
maxacctimeconstraint	10
maxrepeaterdelayconstraint	10
Optimize for dynamic energy	No
Optimize for dynamic power	No
Optimize for leakage power	No
Optimize for cycle time	Yes
Temperature (K)	360
SRAM cell/wordline technology flavor	ITRS HP
Peripheral/Global circuitry technology flavor	ITRS HP
Interconnect projection type	Conservative
Wire type inside mat	Semi-global
Wire type outside mat	Semi-global

Table 10: CACTI input parameters

the best dynamic read energy per access. So further relaxation of the maxacctimeconstraint does not lead to any energy reduction benefits in this case.

Figure 43 shows the impact of varying maxrepeaterdelayconstraint. The maxrepeaterdelayconstraint changes the separation distance and sizing of repeaters/buffers in the H-tree networks and is useful for trading off delay for energy benefits. It can be seen here that varying maxrepeaterdelayconstraint does not lead to energy savings much unless the access time is allowed to degrade heavily. Initially, as maxrepeaterdelayconstraint is increased from 0 to 20%, it can be seen that the access time does not change and there are no energy savings. This is because of the maximum limit that we have imposed on transistor size in CACTI. For values of maxrepeaterdelayconstraint between 0 and 20%, the sizing of the repeater comes out to be larger than the maximum allowed transistor size and is therefore being fixed at the maximum allowed transistor size (the maximum allowed transistor size was fixed at 100F (F = Feature size) for NMOS transistors). For a maxrepeaterdelayconstraint of 400% there is significant energy savings but with a disproportionate degradation of access time.

Figure 44 shows the impact of optimizing the solution generated by CACTI for a 16MB SRAM in different ways. Table 11 shows the different optimization scenarios targeting metrics of random cycle time, dynamic read energy per access, dynamic power and standby leakage power. The percentage variation between the worst and best values for each metric shown in Figure 44 is as follows: access time (5%), random cycle time (296%), area (27%), dynamic read energy per access (21%), and standby leakage power (36%). These variations illustrate the dependence of RAM and cache performance estimation on the kind of optimization that is applied.

11.1.2 Impact of Device Technology

Figure 45 illustrates the tradeoffs associated with assuming different types of devices in the memory cells/-wordline drivers and the rest of the peripheral/global circuitry. Three scenarios are considered:

1. ITRS HP only;
2. ITRS LSTP (memory cells/wordline drivers) + ITRS HP (peripheral/global circuitry);

Optimization Scenario	Optimize for random cycle time	Optimize for dynamic energy	Optimize for dynamic power	Optimize for leakage power
A	Yes	No	No	No
B	Yes	Yes	No	No
C	No	Yes	No	No
D	No	No	Yes	No
E	No	No	No	Yes
F	Yes	Yes	Yes	Yes

Table 11: Different solution optimization scenarios targeting metrics of random cycle time, dynamic read energy per access, dynamic power and standby leakage power.

3. ITRS LSTP only;

It can be seen that the areas of RAMs for the 3 considered scenarios remain more or less the same. With respect to "ITRS HP only", on average over the considered capacities, "ITRS LSTP + ITRS HP" exhibits an improvement of 86% in the standby leakage power. This improvement comes at the cost of 9% worse access time and 72% worse random cycle time. "ITRS LSTP only" shows an improvement of almost 100% in standby leakage power with respect to "ITRS HP only", and this improvement comes at a cost of 166% worse access time and 253% worse random cycle time.

11.1.3 Impact of Interconnect Technology

Figure 46 illustrates the dependence of RAM/cache performance on interconnect technology assumptions. As described in Section 8 on "Technology Modeling", instead of assuming a single set of scaling assumptions for interconnect technology, we consider aggressive and conservative scaling projections as in [12][6]. From Figure 46, it can be seen that for the SRAM capacities and technology (65 nm) considered, the cache performance is not very different under either conservative or aggressive interconnect technology assumptions. The lower resistance per unit length of the aggressive projections leads to lowering of the access time by about 10% on an average. For smaller technologies, the impact of interconnect technology assumptions would be more.

Figure 47 shows the impact of wire type on cache performance. As described in Section 8 on "Technology Modeling", wires outside a mat can be of either 'semi-global' or 'global' type. With respect to semi-global type, global type wires outside mats lead to an improvement in access time of 30% on an average for more or less the same investment in area. The global type wires take up greater area than the semi-global type wires, so the number of mats in the bank with global wire type is fewer than that with semi-global wire type. This leads to increase in the random cycle time because of greater wordline and bitline delays.

11.1.4 Impact of RAM Cell Technology

Figure 48 illustrates the dependence of cache performance on the type of RAM cell technology – SRAM or logic-based embedded DRAM. For each capacity, the properties of SRAMs and DRAMs shown in Figure 48 are for CACTI solutions with the best access time amongst all solutions. It can be seen from this figure that up to about 1 MB capacity, the access time of SRAM is lower after which the access time of DRAM becomes lower. This is because of the decreased wire delay experienced in DRAMs which occupy a much smaller area compared to the SRAMs. For the larger-capacity RAMs ($\geq$ 1MB), on average over all capacities, the area of the DRAMs is about a factor of 2.3 smaller than that of the SRAMs. Because of the destructive read out from a capacitor and the subsequent writeback, it can be seen that the random cycle time of the DRAM is much higher than that of the SRAM. On an average overall capacities it is higher by a factor of about 1.8. However, it can be seen that the multisubbank interleave cycle time of the DRAMs can come close to the random cycle time of SRAMs up to about 1MB. With further pipelining in the request and reply networks, the multisubbank interleave cycle time can be improved further.

The standby leakage power of the DRAM is much lower than that of the SRAMs because of the use of the low-leakage 1T cell. For the larger-capacity RAMs ($\geq$ 1MB), on average the standby leakage power of DRAMs is lower than that of SRAMs by a factor of about 6. For the larger-capacity RAMs, it can also be seen that the dynamic read energy per access of the DRAMs is more or less the same as that of the SRAMs. It is important to note that the comparisons presented here were for SRAMs and DRAMs with the best access time amongst all solutions.

11.2 Version 4.2 vs Version 5.0 Comparisons

We first present the differences in the technology metrics of versions 4.2 and 5.0. Figure 49 shows FO4 delays for versions 4.2 and 5.0. For version 5.0, the FO4 delay has been shown for the ITRS HP device type. It can be seen that, surprisingly, there is good agreement between the FO4 delays of versions 4.2 and 5.0, particularly at the 65 nm node.

Figures 50, 51 and 52 shows the trends for resistance per unit micron, capacitance per unit micron and unbuffered delay for a wire of length 1-micron for versions 4.2 and 5.0. For version 4.2, trends are shown for both 'local' and 'global' wire types which were discussed in Section 2.3. For version 5.0, trends are shown for both 'semi-global' and 'global' wire types and for both aggressive and conservative assumptions of interconnect technology as discussed in Section 8.2. It can be seen that the unbuffered wire delay of the local wire type of version 4.2 is greater than that of the semi-global wire type of version 5.0 by a factor of about 3.5.

Figure 53 compares results generated by CACTI version 4.2 and version 5.0. For version 5.0, we show three solutions – solutions with best access time, best area and best dynamic read energy per access. We also show results from a modified version of version 4.2. In the modified version, we increase the limits defined for Ndwl, Ndbl, Nspd, number of subarrays and number of segments allowed in the H-trees. Increasing these set limits allows a richer exploration of the search space of array partitions, particularly for large SRAMs. We call this version, version 4.2-limits-removed.

Because of significant modeling differences between versions 4.2 and 5.0, it is not easy to compare and analyze the behavior of the two versions, but we make general observations and present high-level plausible reasons for the trends. Firstly, regarding area, we observe that the areas of solutions produced by version 5.0 are much greater than that of the version 4.2 solution. Version 5.0 has made a major update to the area model and has introduced new features such as inclusion of ECC and redundancy overheads which increase area overhead. Also version 5.0 makes use of a bigger SRAM cell at $146F^2$ compared to the $120F^2$ cell used in version 4.2. For the 32 MB SRAM, the solution generated by version 4.2-limits-removed has an area of 343 mm^2 which is greater than the areas of all version 5.0 solutions.

Regarding access time, it can be seen that the access times for the best access time and best dynamic energy solutions of version 5.0 are much lower and scale in a much better way compared to the version 4.2 solution. The access time of the 32 MB SRAM gets better with version 4.2-limits-removed, however, it's still much worse than the access time of the version 5.0 best access time and best dynamic energy solutions. The main reason for the larger access times of version 4.2 is because of the high resistance per unit length of the local wires in version 4.2.

Regarding dynamic energy, it can be seen that the dynamic energy per read access of the version 5.0 best access time and best dynamic energy solutions are greater than that of version 4.2 by a factor of about 5 on an average. This is mainly because of the organization that has been assumed in version 5.0 in which wordlines and bitlines in multiple mats are activated per access. Also, as described in Section 2.2, bugs in version 4.2 with respect to dynamic energy of routing in the H-trees causes the dynamic energy per read access to be underestimated.

Leakage power is heavily dependent on the underlying technology data. The standby leakage of version 4.2 is greater than that of the best access time and best dynamic energy solutions of version 5.0 by a factor of about 3 on an average.

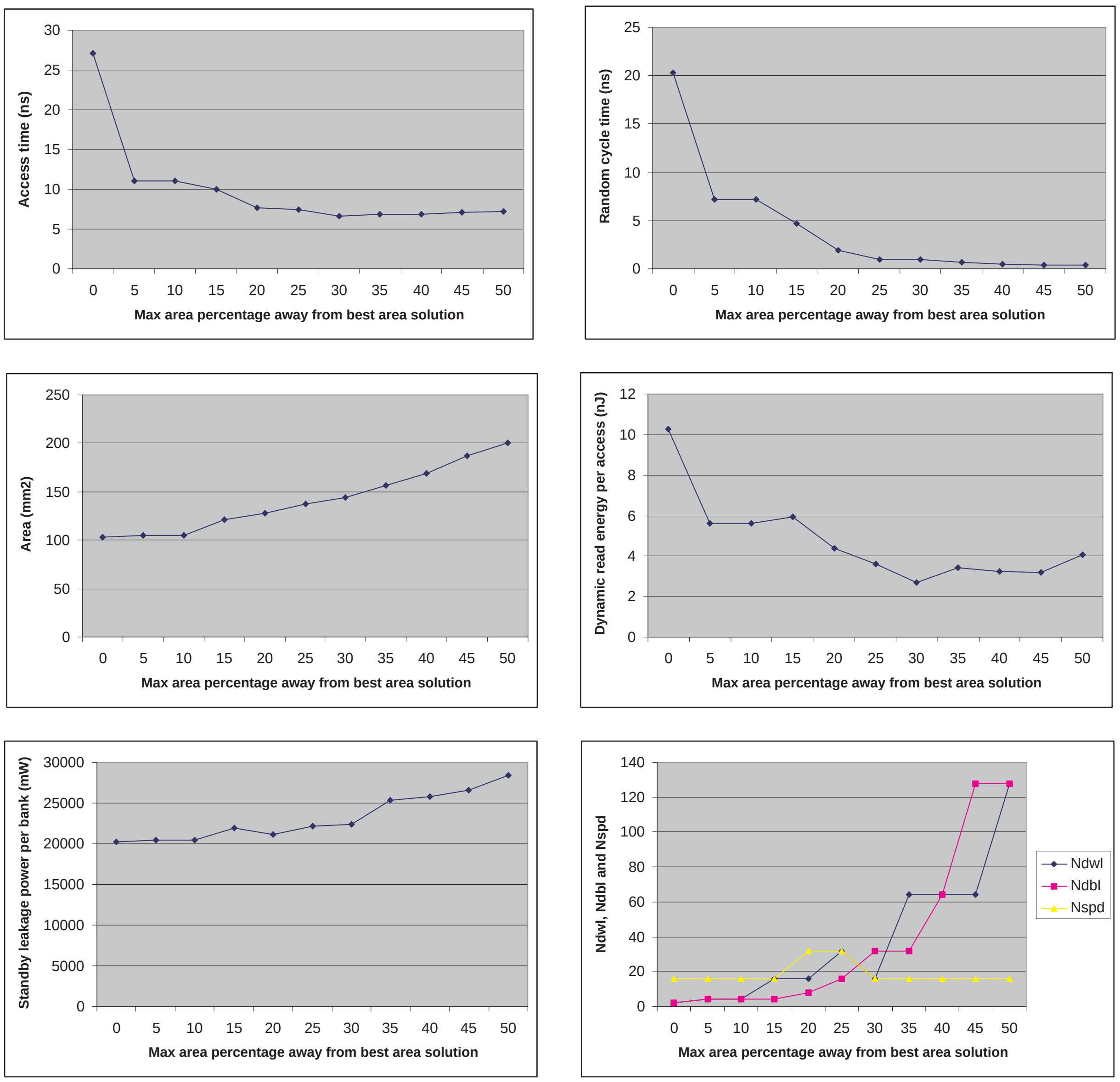


Figure 41: Access time, random cycle time, area, dynamic energy and leakage power of a 16MB SRAM as maxareaconstraint is varied.

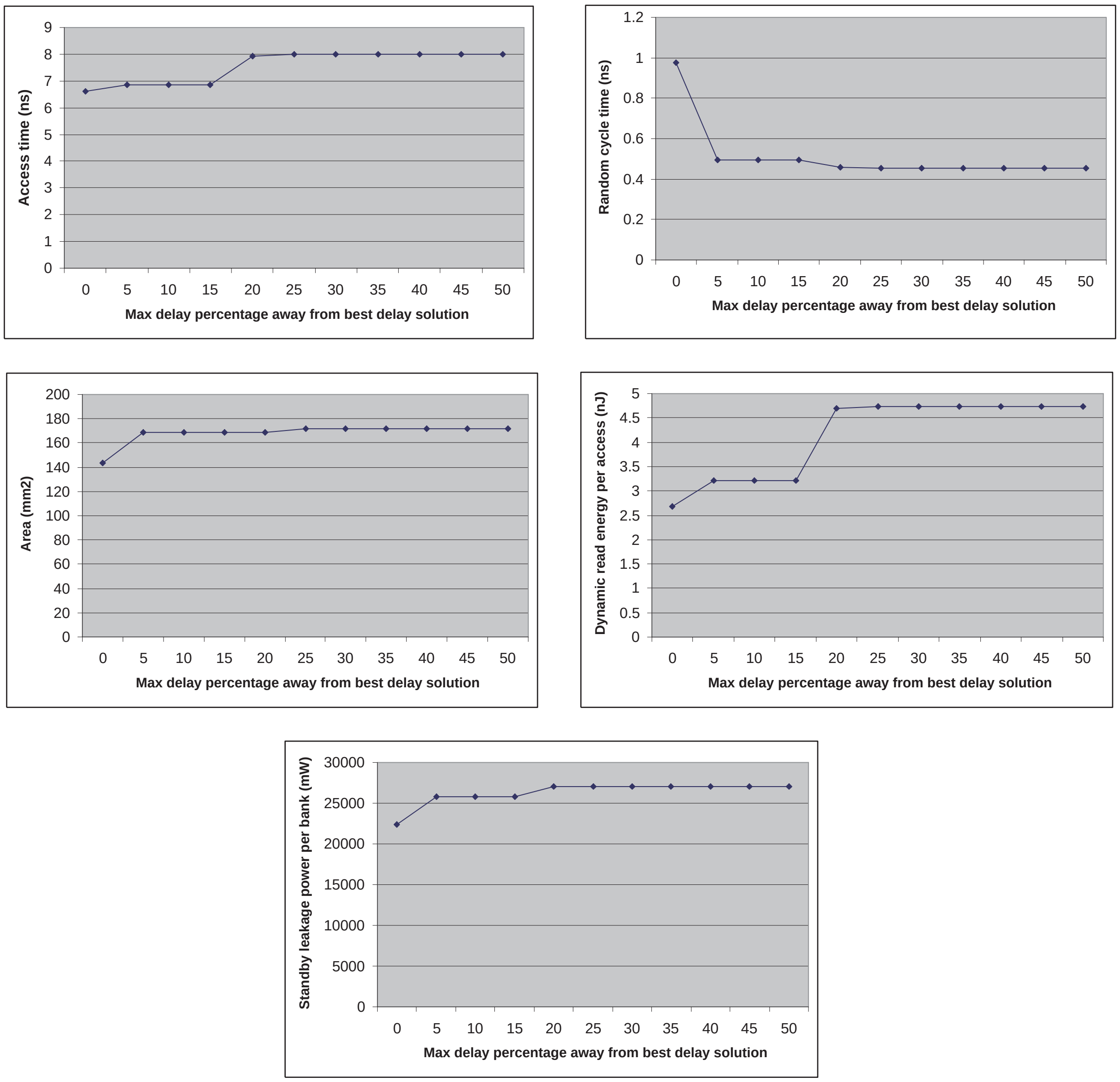


Figure 42: Access time, random cycle time, area, dynamic energy and leakage power of a 16MB SRAM as maxacctimeconstraint is varied.

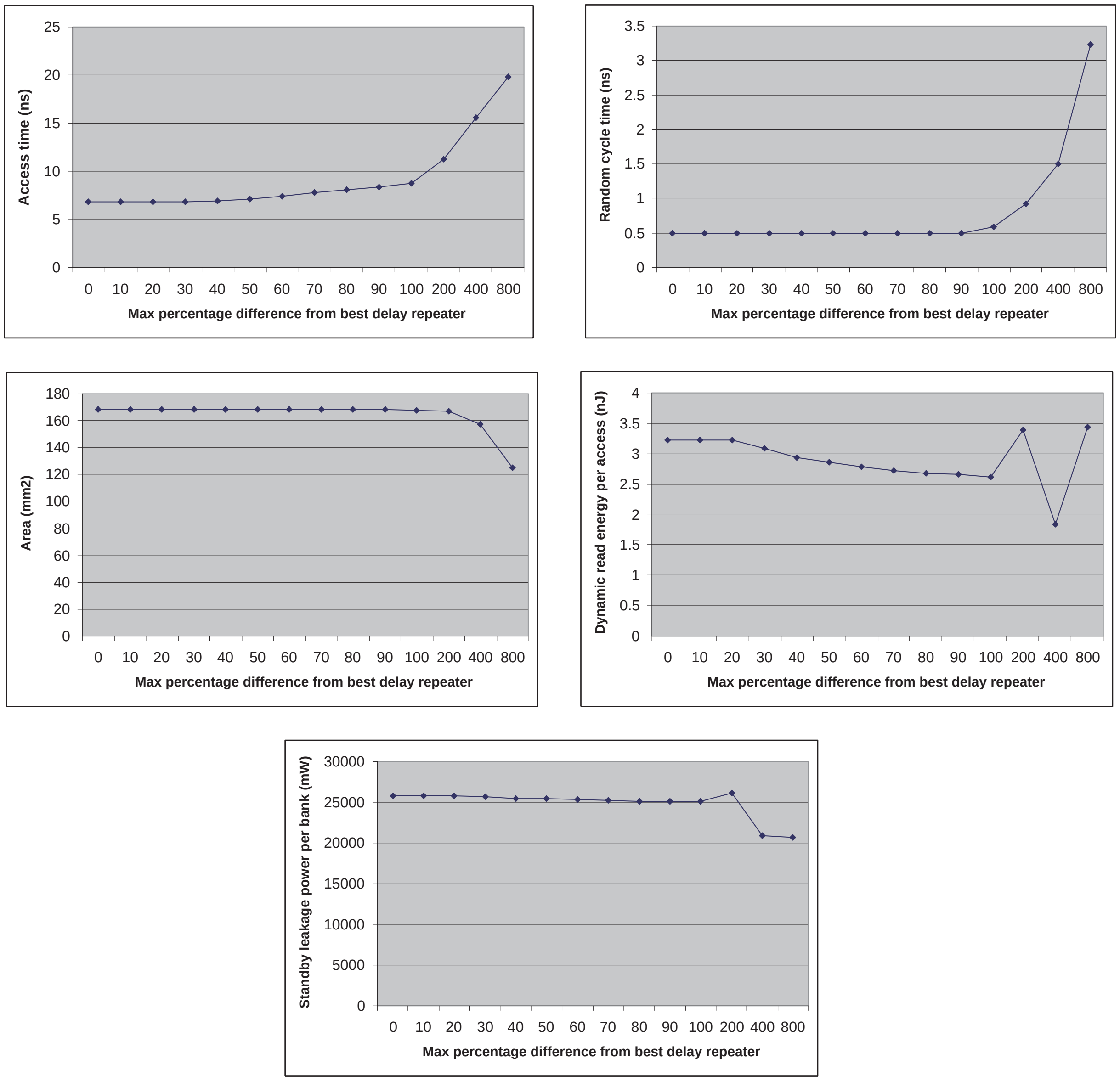


Figure 43: Access time, random cycle time, area, dynamic energy and leakage power of a 16MB SRAM as maxrepeaterdelayconstraint is varied.

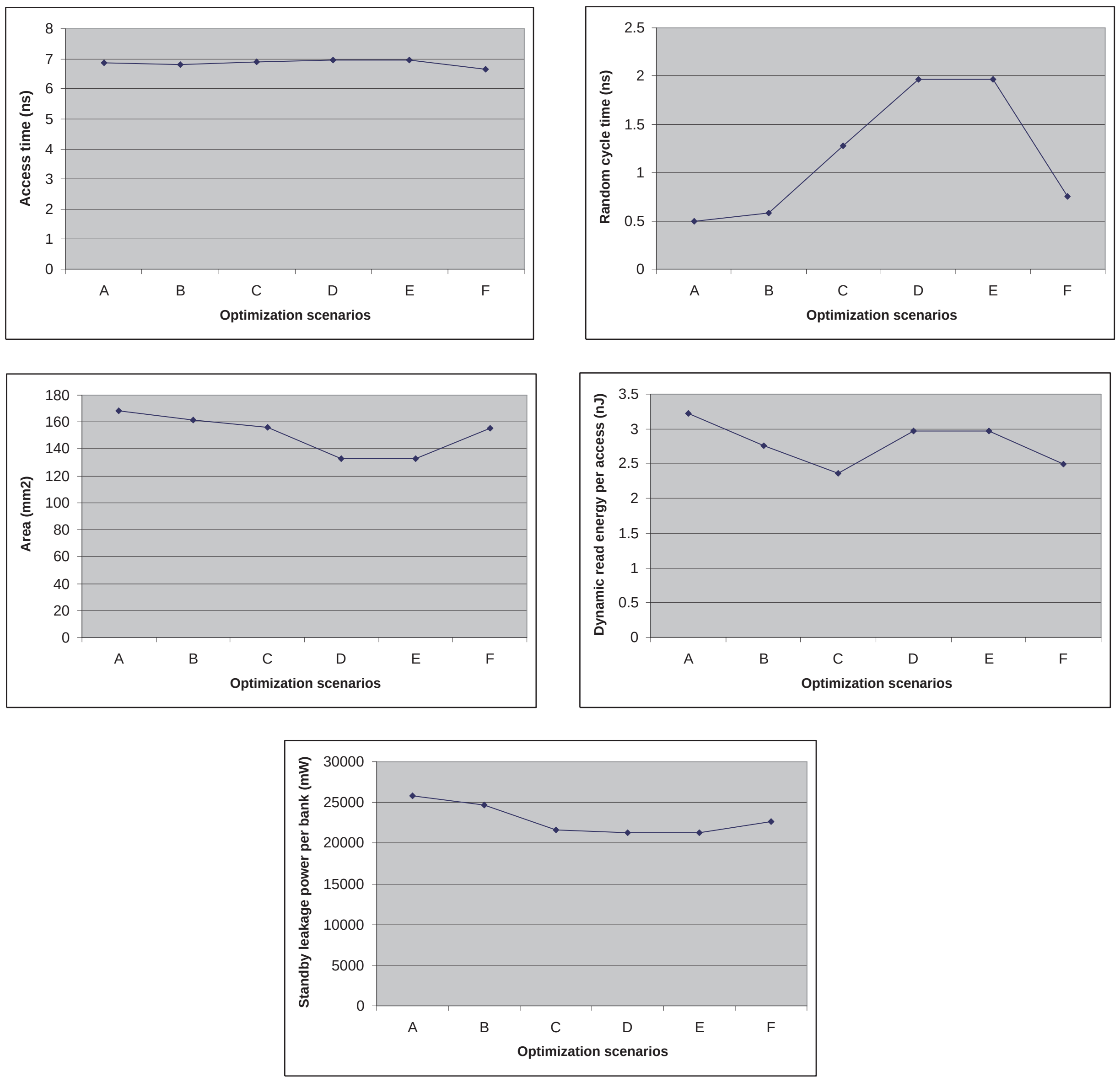


Figure 44: Access time, random cycle time, area, dynamic energy and leakage power of a 16MB SRAM under different optimization function scenarios.

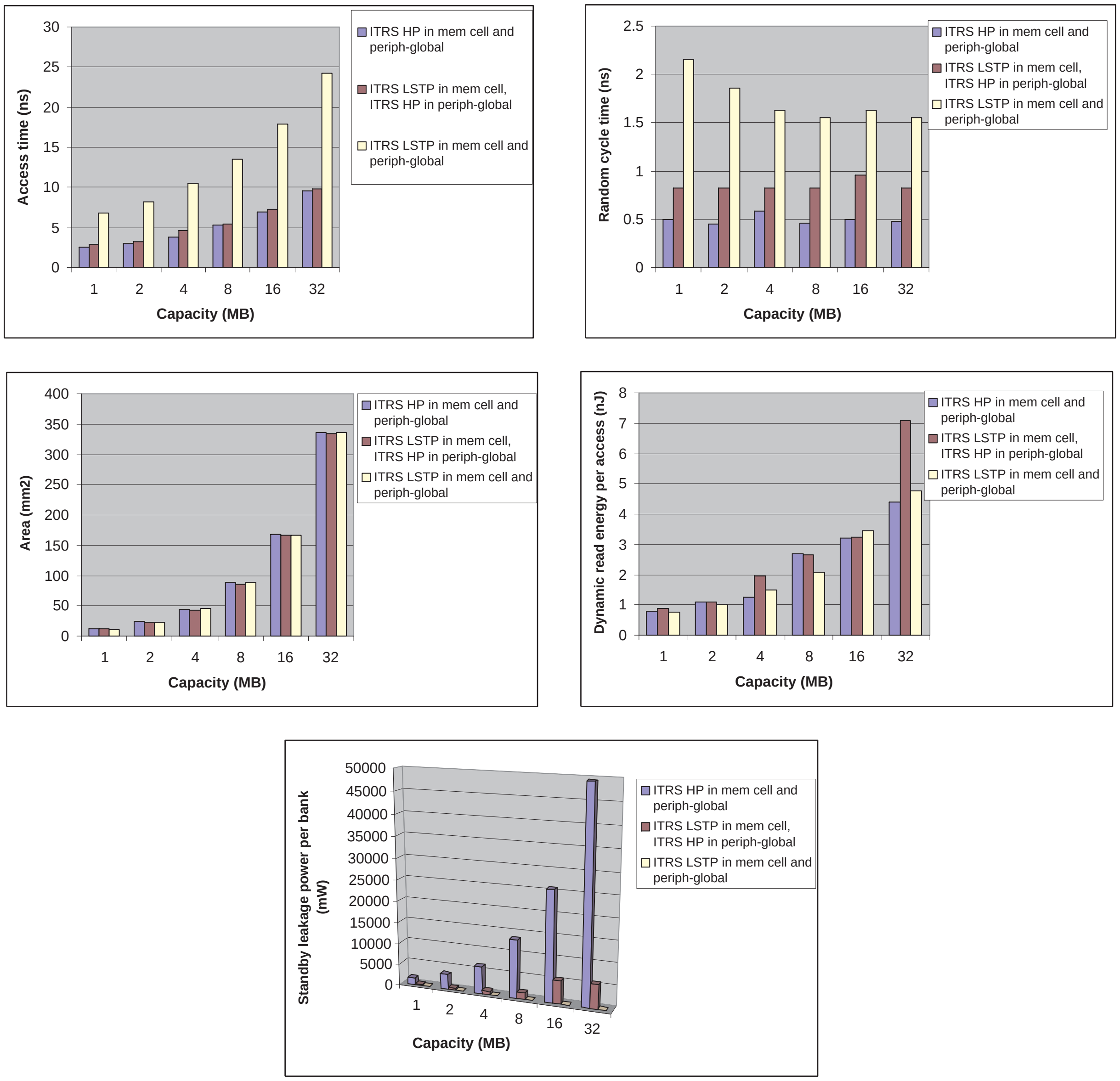


Figure 45: Access time, random cycle time, area, dynamic energy and leakage power of SRAMs for different 65 nm device technology assumptions.

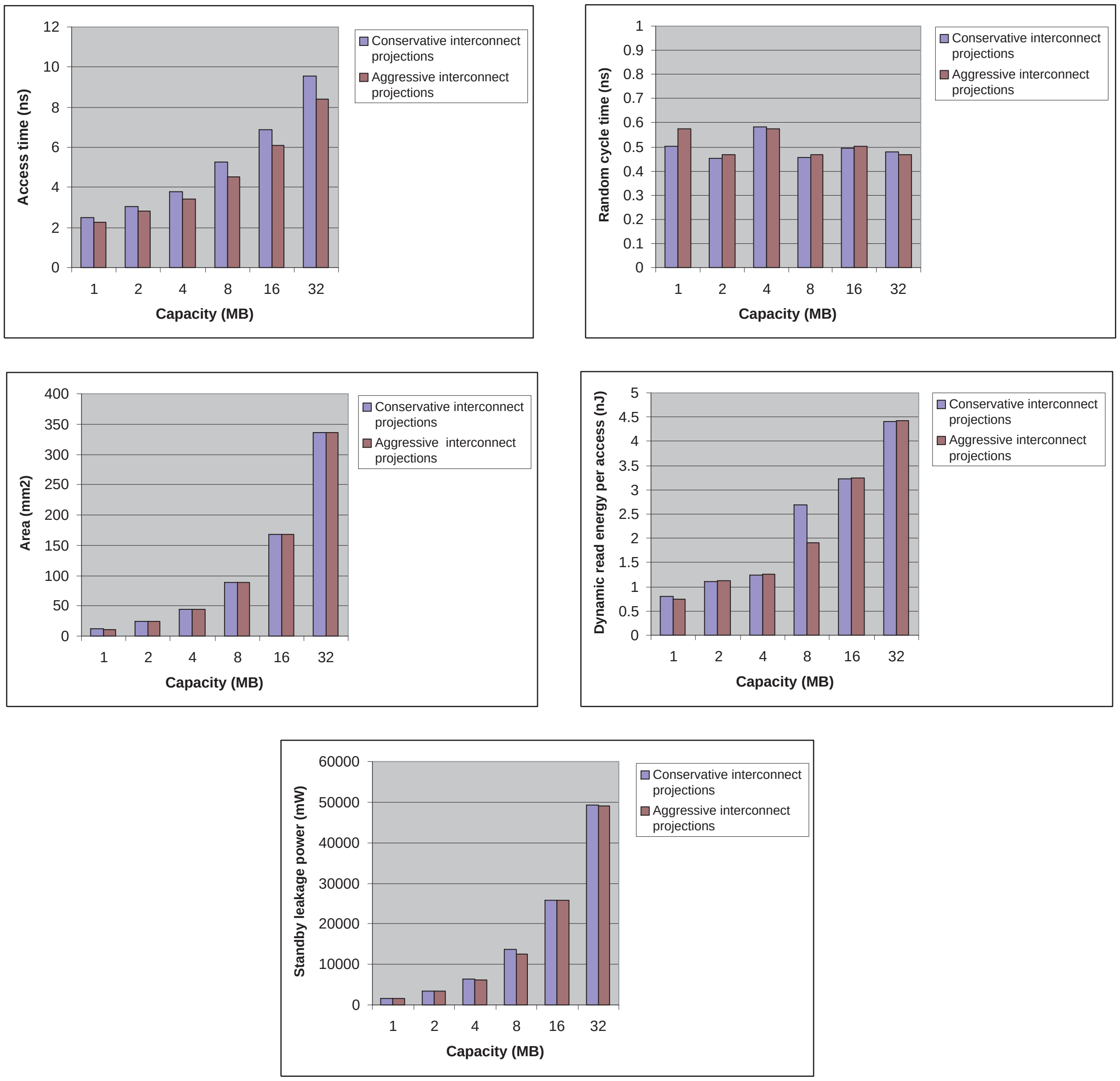


Figure 46: Access time, random cycle time, area, dynamic energy and leakage power of SRAMs under aggressive and conservative interconnect technology assumptions.

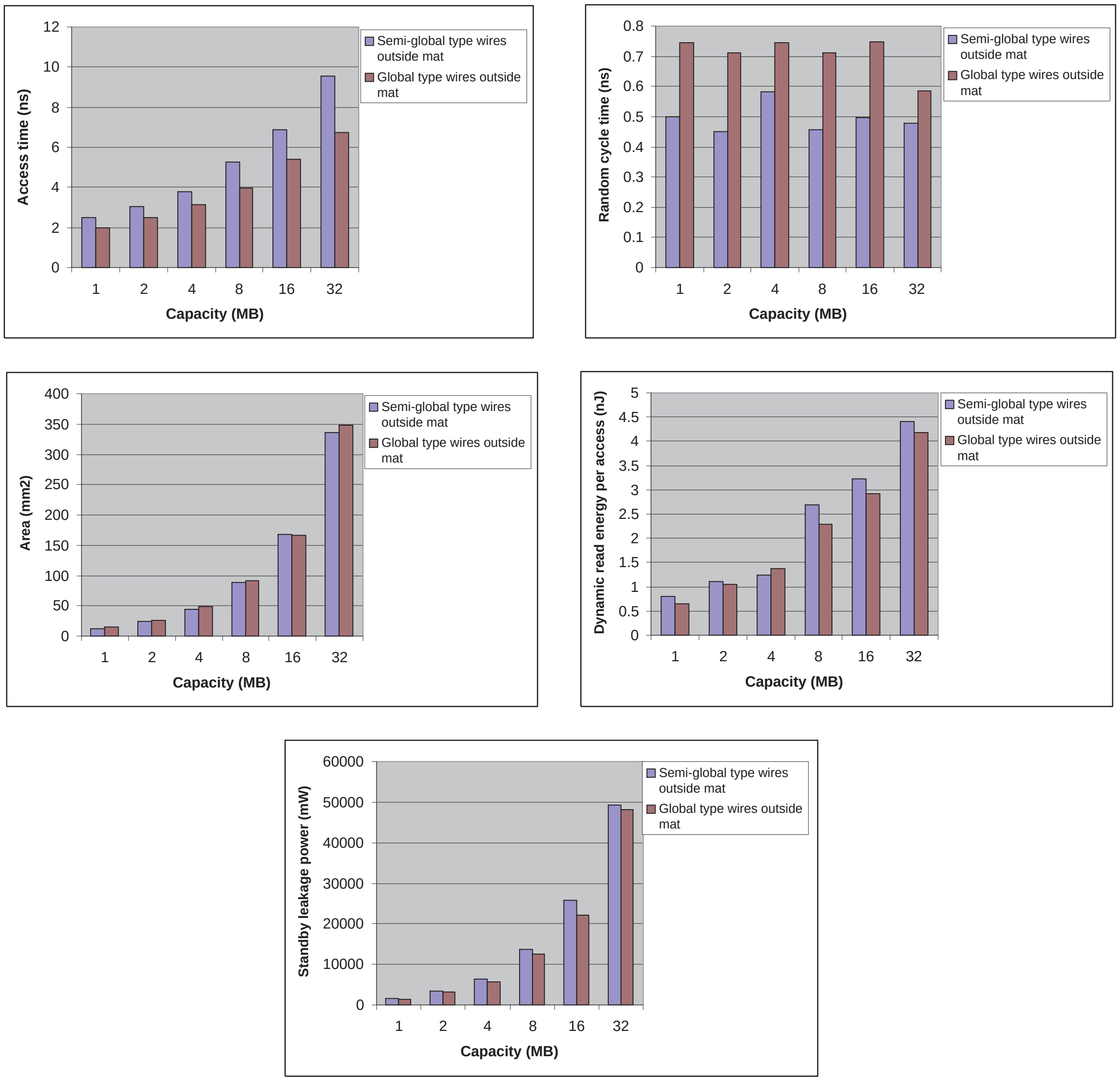


Figure 47: Access time, random cycle time, area, dynamic energy and leakage power of SRAMs with "global" interconnect type used for wires outside the mat.

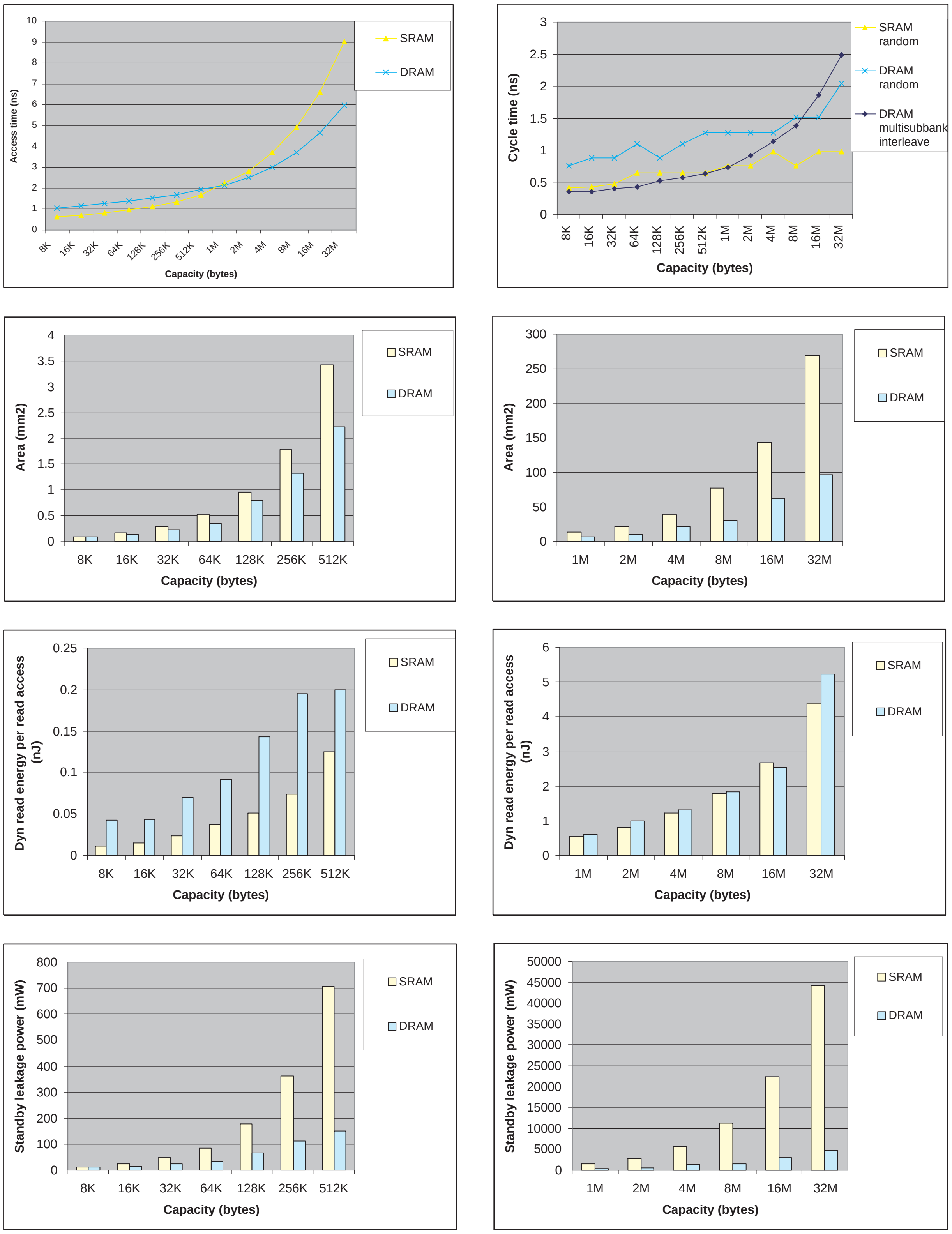


Figure 48: Access time, cycle time, area, dynamic read energy per access and standby leakage power of SRAM and logic-based embedded DRAM for 65 nm technology. The area, dynamic read energy per access and standby leakage power trends are split up into two charts based on capacities.

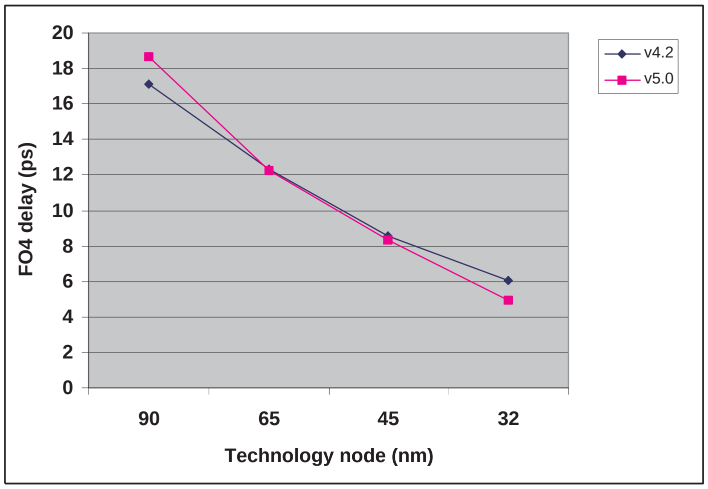


Figure 49: FO4 delays for various technology nodes in versions 4.2 and 5.0. The version 5.0 FO4 delays are for ITRS high-performance device type.

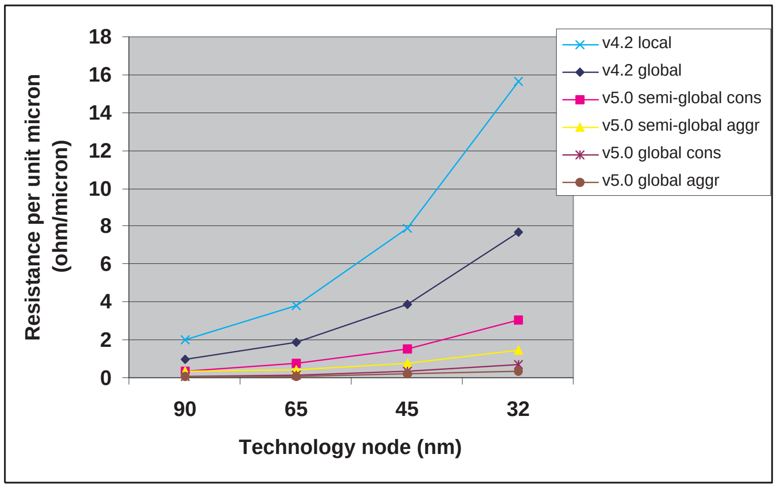


Figure 50: Resistance per unit micron of wire at various technology nodes for versions 4.2 and 5.0.

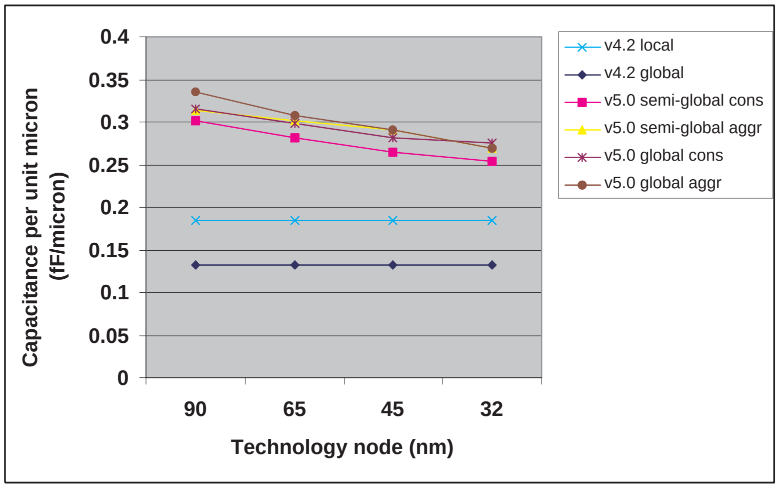


Figure 51: Capacitance per unit micron of wire at various technology nodes for versions 4.2 and 5.0.

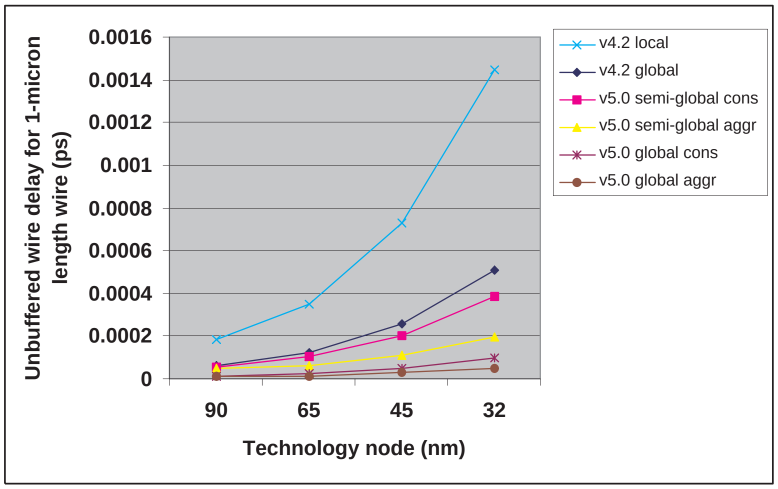


Figure 52: Unbuffered delays through a wire of length 1-micron for various technology nodes in versions 4.2 and 5.0. The wire delays for version 5.0 are for both aggressive and conservative projections.

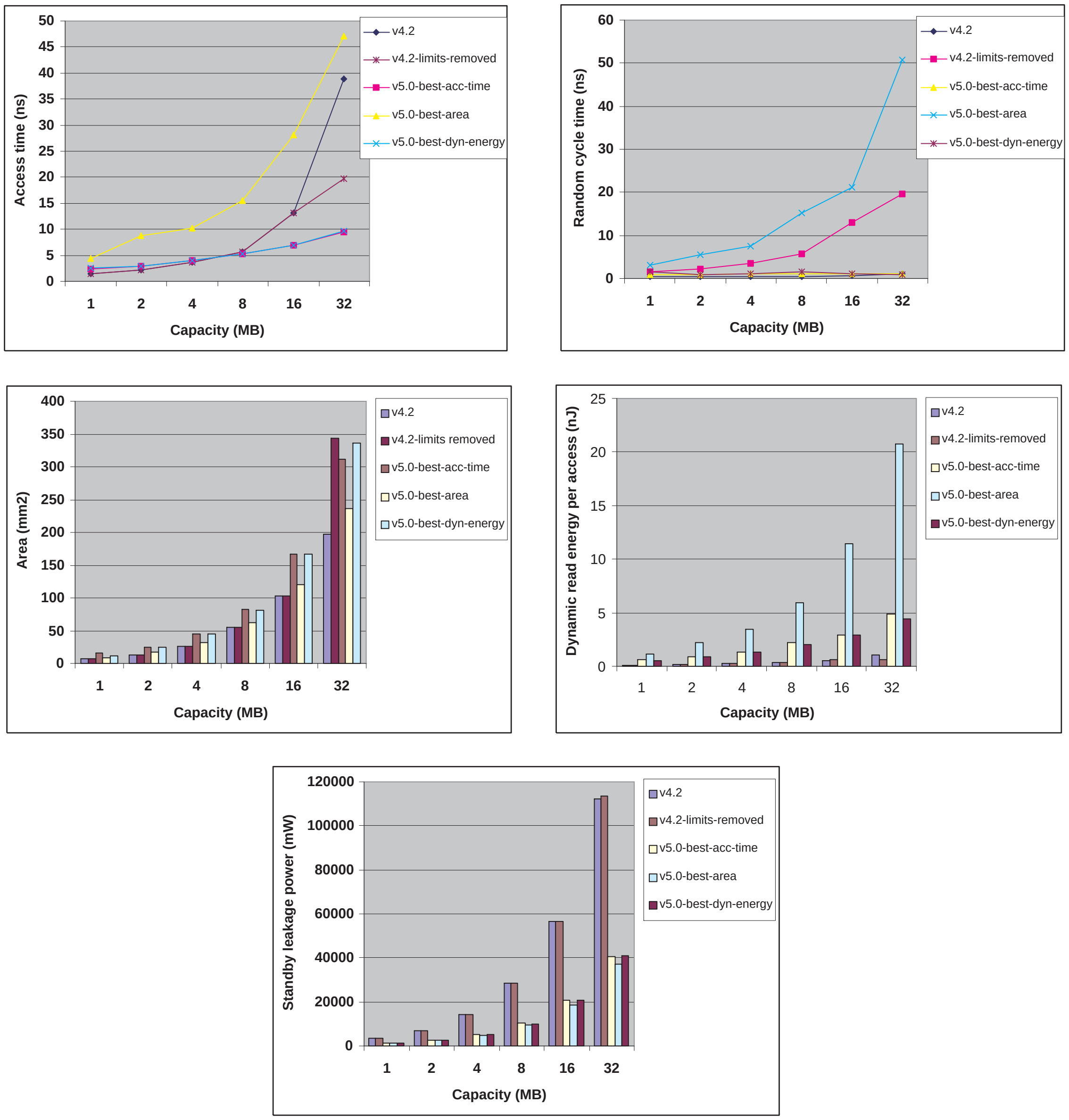


Figure 53: Access time, random cycle time, area, dynamic energy and leakage power obtained from versions 4.2 and 5.0 for SRAMs in 70 nm technology.

12 Validation

In this section, we compare the published values of area, delay and power of real cache designs with the projections of area, delay and power produced by CACTI. The area, delay and power of a real cache or RAM design are influenced by various factors. The design process inherently makes certain area, delay and power tradeoffs based on budgets and requirements. Area, delay and power are also influenced by design methodology, human bias and other practical considerations such as availability of IP from past designs etc. CACTI is based on generic assumptions of cache organization, circuits, design methodology, layout, design rules and technology, whereas a real cache design is based on specific choices of all these. With CACTI 5.0, however, as was shown in the previous section, we provide a number of knobs that can be turned in order to try to emulate a real cache design in a better way. So it is interesting to see how the projections produced by CACTI would compare with real designs.

We use information from real cache specifications to fix as many of the input parameters required by CACTI as possible, such as capacity, associativity, line size, technology-node, etc. In order to understand and explore area/delay/power tradeoffs, we vary parameters such as "Maximum percentage away from best area solution" and "Maximum percentage away from best access time solution" within reasonable bounds.

12.1 Sun SPARC 90 nm L2 cache

[45] describes the implementation of a 90 nm SPARC 4MB L2 cache. Table 12 shows the area, access time, random cycle time and power of the SPARC L2 cache. The clock frequency of the CPU core itself is 1.6 GHz but the L2 cache has a throughput of two clock cycles, so we fix the random cycle time of the L2 cache as 800 MHz.

Table 13 presents the input parameters used with CACTI to model this cache. From the description of the cache, we could not be sure whether the cache access mode is 'normal' or 'fast', so we try out both scenarios. In order to explore a range of area/delay/power tradeoffs, we vary 'maxareaconstraint' between 0 and 50 and 'maxdelayconstraint' between 0 and 30. In order to meet the aggressive random cycle time of the cache, we optimize the solution for random cycle time only. Because we do not have information about the interconnect properties of the fabrication process, we consider both aggressive and conservative projections for interconnect. Also, for 'wire type outside mat', we try out both 'semi-global' and 'global' wire types.

Figure 54 shows bubble charts showing access time, area and power of the SPARC L2 cache and the various solutions generated by CACTI. The charts shown in Figure 54 are for CACTI solutions with 'fast' access mode, 'conservative' interconnect projections and 'semi-global' wire type outside mat. We believe that these values for the parameters are likely to be closest to the actual cache design. Results of the validation exercise with other configuration values are presented in the appendix. As we do not know the operating conditions corresponding to the published value for power of the SPARC L2, we compute dynamic power for the CACTI solutions for three activity factors – 0.1, 0.5 and 1.0. Also, we assume that the ratio of read to write accesses is 3. Note that the solutions shown in Figure 54 are the ones that can meet the random cycle time of the SPARC L2. It can be seen that many solutions have access time, area and power that are quite similar to that of the L2 cache. Table 14 shows error percentages of prominent CACTI solutions with respect to the SPARC L2.

Parameter	Value
Area (mm^2)	128
Access time (ns)	5
Clock frequency (MHz)	800
Total power (W)	8

Table 12: Characteristics of Sun's SPARC 90 nm L2 cache.

12.2 Intel Xeon 65 nm L3 cache

Table 15 shows the area, access time, dynamic power and leakage power of an Intel Xeon L3 cache in a 65 nm process. [46] mentions that the access time of the cache is less than 9 ns; we assume the access

Parameter	Value
Capacity (MB)	4
Line size (bytes)	32
Associativity	4
Number of read/write ports	1
Number of exclusive ports	0
Number of banks	1
Technology-node (nm)	90
Output width (bits)	256
Specific tag	Yes
Tag width	34
Access mode	Normal/Fast
Pure RAM	No (cache)
DRAM	No
Repeaters in bank H-trees	Yes
maxareaconstraint	0 - 70
maxacctimeconstraint	0 - 30
maxrepeaterdelayconstraint	10
Optimize for dynamic energy	No
Optimize for dynamic power	No
Optimize for leakage power	No
Optimize for cycle time	Yes
Temperature (K)	360
SRAM cell/wordline technology flavor	ITRS HP
Peripheral/Global circuitry technology flavor	ITRS HP
Interconnect projection type	Conservative
Wire type inside mat	Semi-global
Wire type outside mat	Semi-global/Global

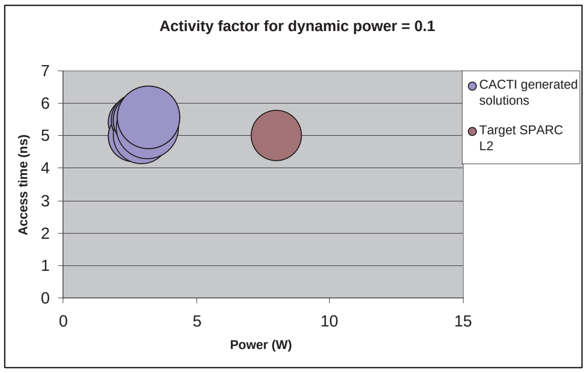
Table 13: CACTI input parameters used for modeling 90 nm SPARC L2 cache

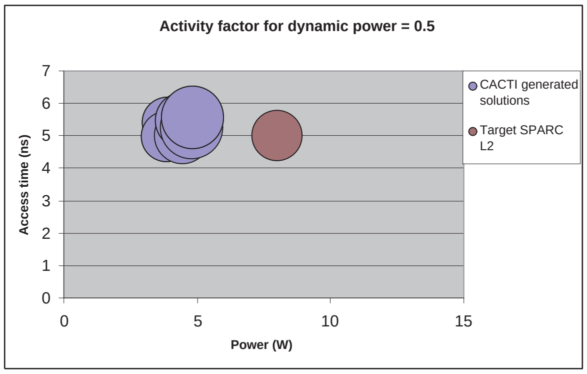
time to be 9 ns. The clock frequency of the core itself is given to be 3.4 GHz and the clock frequency of the L3 is given to be half that of the core [47]. Also, because the output bus width of the L3 is 256 bits, it would require two cycles to transmit a 64-byte line, so we fix the random cycle frequency of the L3 to be one-fourth that of the CPU, i.e. 850 MHz. The dynamic power of the cache comes out to be 5.4W based on information from [47][48], however [21] mentions that the cache consumes about 1.7W for "average applications". We speculate that these differences in dynamic power numbers that have been quoted are because of different activity factors in the cache caused due to measurements or simulations of applications with different characteristics. While carrying out comparisons of the area, delay and power of the Intel cache with those of the solutions generated by CACTI, we use both values of power.

The 65 nm process offers transistors with 35 nm gate-lengths. The cache itself, however, makes use of longer-channel devices with lengths that are about 10% longer than the nominal. The longer-channel devices have on-currents that are about 10% less than the nominal devices but have leakage that is lower by a factor of 3. The cache operates in a voltage domain different from that of the cores. The cores can operate at 1.25V while the cache operates at 1.1V.

In order to control leakage in the cache, the Intel cache implements n and p sleep transistors at the level of 'blocks' within subarrays (Each subarray within the Intel cache is composed of multiple 'blocks' with one block within a subarray activated per access. The 'subarray' of the Intel cache is not the same as that of CACTI). The impact of these sleep transistors is that leakage power in all blocks that are not activated during an access is cut down by half.

Table 16 shows the input parameters used with CACTI to model the Intel L3 cache. In order to compare the power numbers produced by CACTI with those of the Intel cache in a fair manner, we assume the use





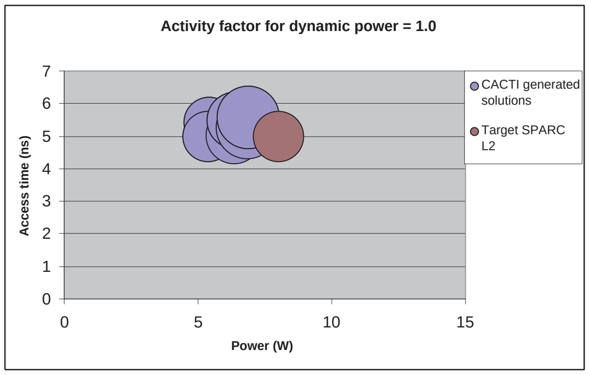


Figure 54: Access time, area and power of the 90 nm SPARC L2 cache and of solutions generated by CACTI. The CACTI solutions assume 'fast' access mode, 'conservative' interconnect projections and 'semi-global' wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-generated solutions.

of leakage control mechanisms within CACTI similar to that used in the Intel cache. To model the longer-channel devices that have been used in the Intel cache which reduce leakage by a factor of 3, we also reduce the leakage of the CACTI 65 nm high-performance transistors by a factor of 3. Also, we assume the use of sleep transistors that cut down the leakage of all mats that are not activated during an access by half.

Figure 55 shows bubble charts of access time, area, and power of the Intel cache and the various solutions generated by CACTI. The charts shown in Figure 55 are for CACTI solutions with 'conservative' interconnect projections and 'semi-global' wire type outside mat as we believe that these values for the parameters are likely to be closest to the actual cache design. Results of the validation exercise with other configuration values are again presented in the appendix. Again, as we do not know the operating conditions corresponding to the published value for power, we compute dynamic power for the CACTI solutions for three activity factors – 0.1, 0.5 and 1.0. Again, we assume that the ratio of read to write accesses is 3. There are two targets for the Intel cache corresponding to the two values of dynamic power shown in Table 15. It can be seen from Figure 55 that many CACTI solutions have area, access time and power that are quite similar to that of the Xeon L3. Tables 17 and 18 show error percentages of prominent CACTI solutions with respect to the Xeon L3.

Solution	% error in acc time	% error in area	% error in power	Avg of acc time, area and power % errors
Best % error in acc time	0/0/0	-3/-3/-3	-67/-52/-33	23/18/12
Best % error in area	0/0/0	-3/-3/-3	-67/-52/-33	23/18/12
Best % error in power	0/0/0	-3/-3/-3	-67/-52/-33	23/18/12
Best average of area, acc time and power % errors	0/0/0	-3/-3/-3	-67/-52/-33	23/18/12
Best average of area and acc time % errors	0/0/0	-3/-3/-3	-67/-52/-33	23/18/12
Best average of acc time and power % errors	0/0/5	20/20/49	-63/-44/-14	28/22/23
Best acc time	0/0/0	-3/-3/-3	-67/-52/-33	23/18/12
Best area	0/0/0	-3/-3/-3	-67/-52/-33	23/18/12
Best power	0/0/0	-3/-3/-3	-67/-52/-33	23/18/12

Table 14: Error percentages of some prominent solutions generated by CACTI with respect to a 90 nm SPARC L2 cache. The CACTI solutions assume 'fast' access mode, 'conservative' interconnect projections and 'semi-global' wire type outside mat. We have used 3 activity factors of 0.1, 0.5 and 1, and so each entry in the table has 3 values.

Area (mm^2)	200	Measured from die photo [49]
Access time (ns)	9 ns	[46]
Clock frequency (GHz)	850 MHz	[47]
Dynamic power (W)	1.7/5.4	[47][48][21]
Leakage power (W)	6.6	[47][48]

Table 15: Characteristics of Intel Xeon's 65 nm L3 cache.

Parameter	Value
Capacity (MB)	16
Line size (bytes)	64
Associativity	16
Number of read/write ports	1
Number of exclusive ports	0
Number of banks	2
Technology-node (nm)	65
Output width (bits)	512
Specific tag	No
Access mode	Serial
Pure RAM	No (cache)
DRAM	No
maxareaconstraint	0 - 50
maxdelayconstraint	0 - 30
maxrepeaterdelayconstraint	10
Optimize for dynamic energy	No
Optimize for dynamic power	No
Optimize for leakage power	No
Optimize for cycle time	Yes
Temperature (K)	360
SRAM cell/wordline technology flavor	ITRS HP
Peripheral/Global circuitry technology flavor	ITRS HP
Interconnect projection type	Conservative
Wire type inside mat	Semi-global
Wire type outside mat	Semi-global/Global

Table 16: CACTI input parameters used for modeling 65 nm Intel Xeon L3 cache.

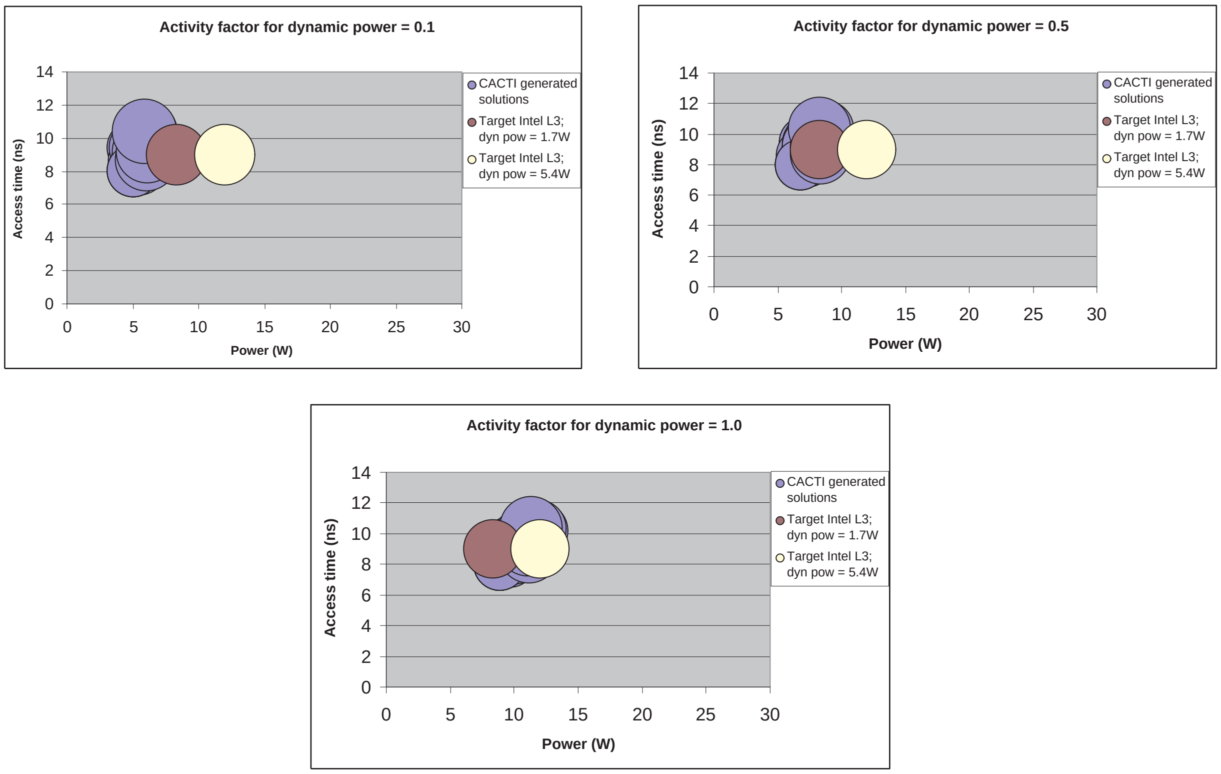


Figure 55: Access time, area and power of the 65 nm Xeon L3 cache and of solutions generated by CACTI. The CACTI solutions assume 'conservative' interconnect projections and 'semi-global' wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-generated solutions.

Solution	% error in acc time	% error in area	% error in power	Avg of acc time, area and power % errors
Best % error in acc time	-11/-11/-11	-23/-23/-23	-40/-19/7	24/18/14
Best % error in area	-11/-11/-11	-23/-23/-23	-40/-19/7	24/18/14
Best % error in power	-11/-11/-11	-23/-23/-23	-40/-19/7	24/18/14
Best average of area, acc time and power % errors	2/2/2	0/6/0	-31/0/21	11/3/8
Best average of area and acc time % errors	2/2/2	0/0/0	-31/-8/21	11/4/8
Best average of acc time and power % errors	2/2/-4	6/6/-14	-27/0/10	12/3/9
Best acc time	-11/-11/-11	-23/-23/-23	-40/-19/7	24/18/14
Best area	-11/-11/-11	-23/-23/-23	-40/-19/7	24/18/14
Best power	-11/-11/-11	-23/-23/-23	-40/-19/7	24/18/14

Table 17: Error percentages of some prominent solutions generated by CACTI with respect to a 65 nm Intel Xeon L3 cache when we assume that the dynamic power consumed by the cache is 1.7W. The CACTI solutions assume 'conservative' interconnect projections and 'semi-global' wire type outside mat. We have used 3 activity factors of 0.1, 0.5 and 1, and so each entry in the table has 3 values.

Solution	% error in acc time	% error in area	% error in power	Avg of acc time, area and power % errors
Best % error in acc time	-11/-11/-11	-23/-23/-23	-58/-44/-26	31/26/20
Best % error in area	-11/-11/-11	-23/-23/-23	-58/-44/-26	31/26/20
Best % error in power	-11/-11/-11	-23/-23/-23	-58/-44/-26	31/26/20
Best average of area, acc time and power % errors	2/2/2	0/6/6	-52/-31/-7	18/13/5
Best average of area and acc time % errors	2/2/2	0/0/0	-52/-37/-17	18/13/6
Best average of acc time and power % errors	2/2/2	6/6/6	-49/-31/-7	19/13/5
Best acc time	-11/-11/-11	-23/-23/-23	-58/-44/-26	31/26/20
Best area	-11/-11/-11	-23/-23/-23	-58/-44/-26	31/26/20
Best power	-11/-11/-11	-23/-23/-23	-58/-44/-26	31/26/20

Table 18: Error percentages of some prominent solutions generated by CACTI with respect to a 65 nm Intel Xeon L3 cache when we assume that the dynamic power consumed by the cache is 5.4W. The CACTI solutions assume 'conservative' interconnect projections and 'semi-global' wire type outside mat. We have used 3 activity factors of 0.1, 0.5 and 1, and so each entry in the table has 3 values.

13 Future Work

Non Uniform Cache Access (NUCA) is an interesting architecture that CACTI could support in the future. Incorporation of models for low-swing interconnect into CACTI could also be an interesting enhancement.

14 Conclusions

In this technical report, we have described the various enhancements carried out in CACTI 5.0 while also providing a comprehensive overview of the CACTI area, access time and power modeling. CACTI 5.0 includes a number of major improvements over CACTI 4.0. The base technology modeling has been changed from simple linear scaling of the original 0.8 micron technology to models based on the ITRS roadmap. Data for different ITRS device types has been incorporated into CACTI. Interconnect technology data has also been updated so that it is now based off well-documented models and data. CACTI 5.0 has also added support for modeling embedded DRAM in such a way that it now becomes possible to compare tradeoffs involving the use of embedded SRAM or DRAM for identical input specifications. This has been achieved by an extensive rewrite of the CACTI code base. Various organizational and circuit assumptions have been clarified and updated. The modeling has also been restructured in such a way that it is now more modular and easier to extend and evolve.

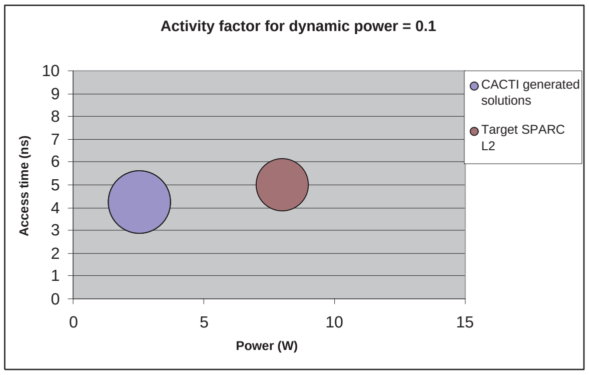
In the studies shown in this report, the impact of technology assumptions on cache performance has been pointed out and emphasized. The importance of solution optimization techniques on memory and cache performance has also been highlighted. Area, delay and power results obtained from version 5.0 have been compared against published data available for two prominent caches. Taking into account the extremely generic nature of CACTI, it was found that there is reasonable agreement between the results produced by CACTI and the published data.

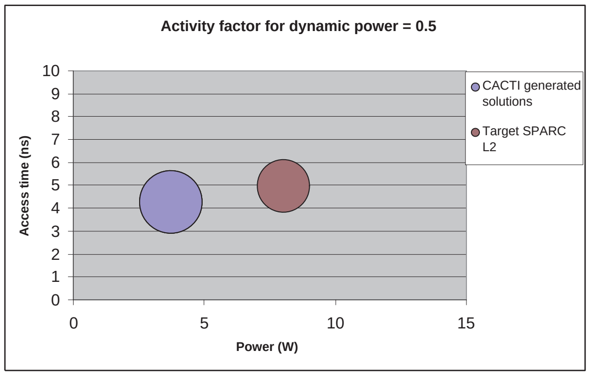
Finally, as in the original CACTI report, we would like to caution users against making too many conclusions based on results shown in this report. It is important to know that CACTI is a simplified model for memories and caches with various limitations at the various levels of modeling, so appropriate caution and judgement needs to be exercised with its use. In general, it is best to use CACTI for studies that involve relative optimization.

Acknowledgments

We would like to thank the following people for many helpful discussions and comments: Jung Ho Ahn and Al Davis from HP Labs, and Jay Brockman and Christian Poellabauer from the University of Notre Dame.

A Additional CACTI Validation Results for 90 nm SPARC L2





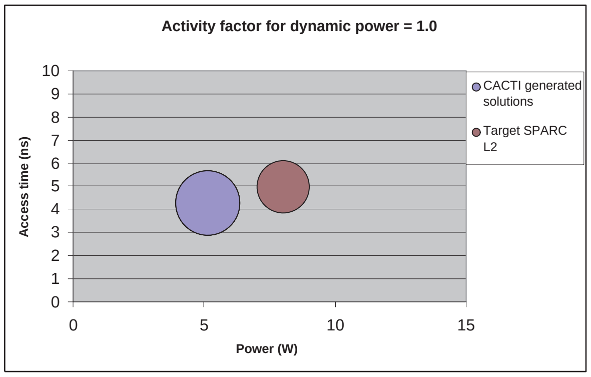


Figure 56: Access time, area and power of the 90 nm SPARC L2 cache and of solutions generated by CACTI. The CACTI solutions are for assumptions of 'fast' access mode, 'conservative' interconnect projections and 'global' wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-generated solutions.

Solution	% error in acc time	% error in area	% error in power	Avg of acc time, area and power % errors
Best % error in acc time	-15/-15/-15	45/45/45	-68/-54/-36	43/38/32
Best % error in area	-15/-15/-15	45/45/45	-68/-54/-36	43/38/32
Best % error in power	-15/-15/-15	45/45/45	-68/-54/-36	43/38/32
Best average of area, acc time and power % errors	-15/-15/-15	45/45/45	-68/-54/-36	43/38/32
Best average of area and acc time % errors	-15/-15/-15	45/45/45	-68/-54/-36	43/38/32
Best average of acc time and power % errors	-15/-15/-15	45/45/45	-68/-54/-36	43/38/32
Best acc time	-15/-15/-15	45/45/45	-68/-54/-36	43/38/32
Best area	-15/-15/-15	45/45/45	-68/-54/-36	43/38/32
Best power	-15/-15/-15	45/45/45	-68/-54/-36	43/38/32

Table 19: Error percentages of some prominent solutions generated by CACTI with respect to a 90 nm SPARC L2 cache under assumptions of ‘fast’ access mode, ‘conservative’ interconnect projections and ‘global’ wire type outside mat. We have used 3 activity factors of 0.1, 0.5 and 1, and so each entry in the table has 3 values.

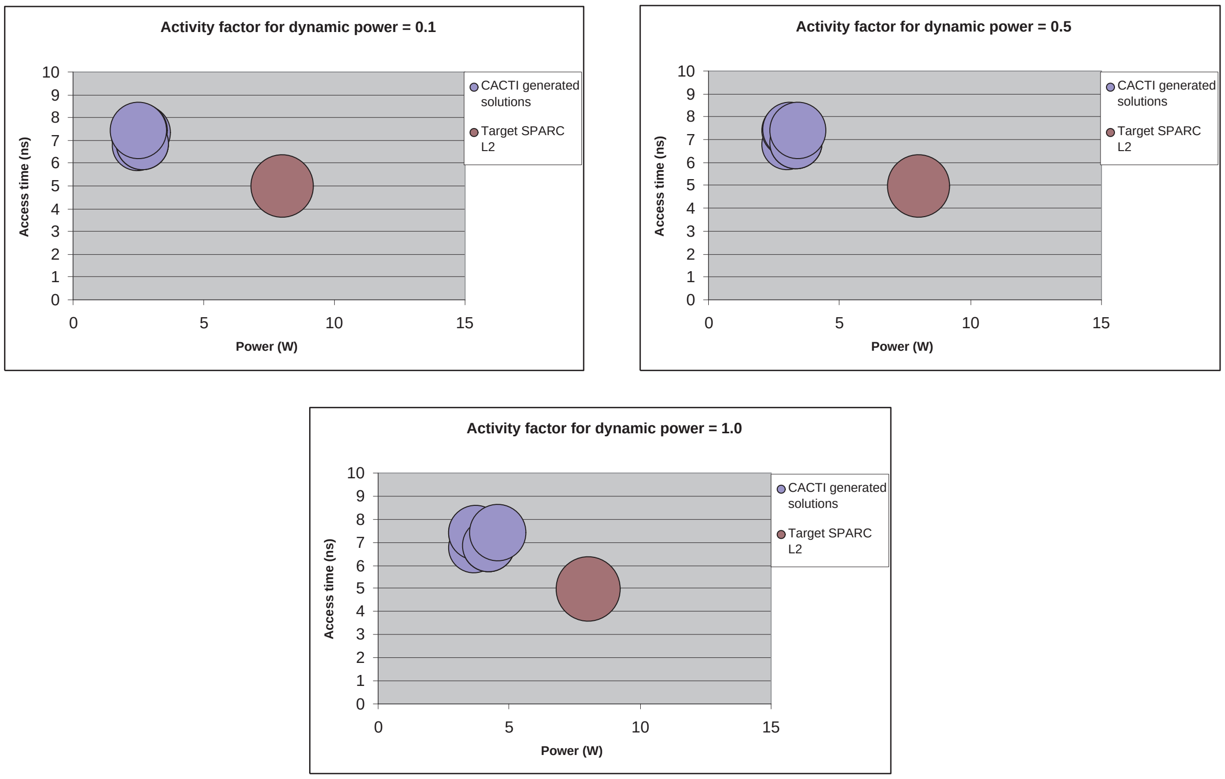


Figure 57: Access time, area and power of the 90 nm SPARC L2 cache and of solutions generated by CACTI. The CACTI solutions are for assumptions of ‘normal’ access mode, ‘conservative’ interconnect projections and ‘semi-global’ wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-generated solutions.

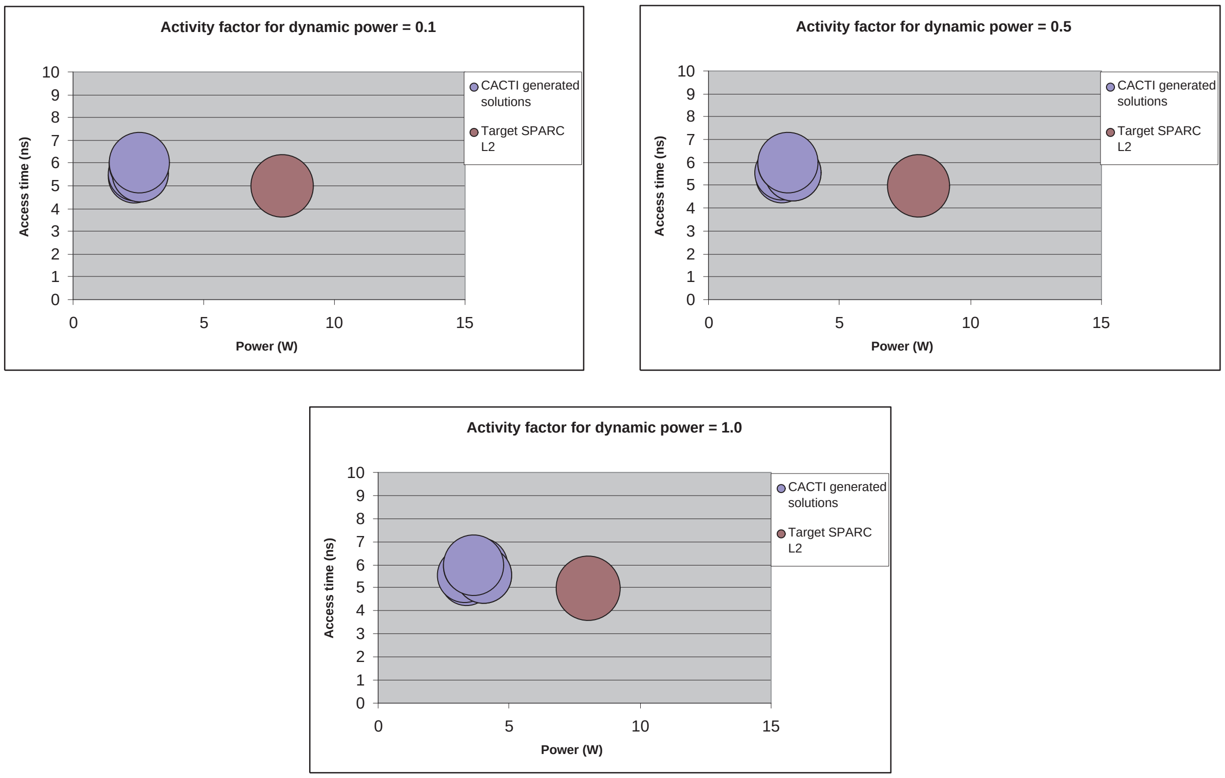


Figure 58: Access time, area and power of the 90 nm SPARC L2 cache and of solutions generated by CACTI. The CACTI solutions are for assumptions of 'normal' access mode, 'conservative' interconnect projections and 'global' wire type outside mat. The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-generated solutions.

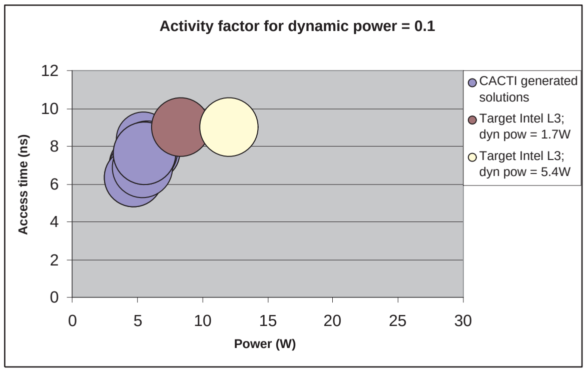
Solution	% error in acc time	% error in area	% error in power	Avg of acc time, area and power % errors
Best % error in acc time	35/35/35	-38/-38/-38	-69/-63/-54	47/45/42
Best % error in area	44/44/44	-41/-41/-41	-70/-62/-52	52/49/46
Best % error in power	44/35/35	-41/-38/-38	-70/-63/-54	52/45/42
Best average of area, acc time and power % errors	48/48/48	-19/-19/-19	-69/-57/-43	45/42/37
Best average of area and acc time % errors	48/48/48	-19/-19/-19	-69/-57/-43	45/42/37
Best average of acc time and power % errors	37/37/37	-32/-32/-32	-67/-58/-47	46/43/39
Best acc time	35/35/35	-38/-38/-38	-69/-63/-54	47/45/42
Best area	44/44/44	-41/-41/-41	-70/-62/-52	52/49/46
Best power	44/35/35	-41/-38/-38	-70/-63/-54	52/45/42

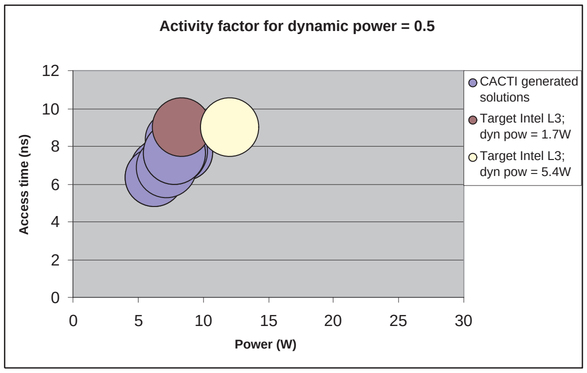
Table 20: Error percentages of some prominent solutions generated by CACTI with respect to a 90 nm SPARC L2 cache for a normal cache under assumptions of 'normal' access mode, 'conservative' interconnect projections and 'semi-global' wire type outside mat. The CACTI solutions and errors depends on the activity factor. We have used 3 activity factors of 0.1, 0.5 and 1, and so each entry in the table has 3 values.

Solution	% error in acc time	% error in area	% error in power	Avg of acc time, area and power % errors
Best % error in acc time	8/8/8	-31/-31/-31	-71/-65/-58	37/35/32
Best % error in area	16/16/16	-35/-35/-35	-70/-63/-53	40/38/34
Best % error in power	8/11/11	-31/-24/-24	-71/-65/-59	37/33/31
Best average of area, acc time and power % errors	20/20/11	-8/-8/-20	-68/-62/-49	32/30/27
Best average of area and acc time % errors	20/20/20	-8/-8/-8	-68/-62/-54	32/30/27
Best average of acc time and power % errors	8/11/11	-31/-20/-20	-71/-60/-49	37/30/27
Best acc time	8/8/8	-31/-31/-31	-71/-65/-58	37/35/32
Best area	16/16/16	-35/-35/-35	-70/-63/-53	40/38/34
Best power	8/11/11	-31/-24/-24	-71/-65/-59	37/33/31

Table 21: Error percentages of some prominent solutions generated by CACTI with respect to a 90 nm SPARC L2 cache for a normal cache under assumptions of 'normal' access mode, 'conservative' interconnect projections and 'global' wire type outside mat. We have used 3 activity factors of 0.1, 0.5 and 1, and so each entry in the table has 3 values.

B Additional CACTI Validation Results for 65 nm Xeon L3





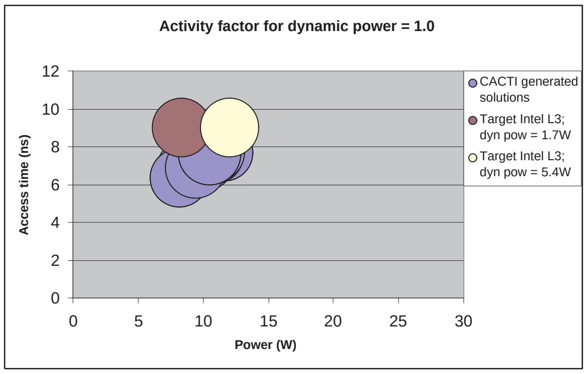


Figure 59: Access time, area and power of the 65 nm Xeon L3 cache and of solutions generated by CACTI. The CACTI solutions are for assumptions of 'conservative' interconnect projections and 'global' wire type outside mat.The 3 plots correspond to 3 activity factors assumed while computing dynamic power for the CACTI-generated solutions.

Solution	% error in acc time	% error in area	% error in power	Avg of acc time, area and power % errors
Best % error in acc time	-29/-29/-29	2/2/2	-43/-25/-2	25/19/11
Best % error in area	-17/-17/-17	-11/-11/-11	-34/-9/24	21/12/17
Best % error in power	-29/-29/-29	2/2/2	-43/-25/-2	25/19/11
Best average of area, acc time and power % errors	-15/-15/-29	-1/-1/2	-26/3/-2	14/7/11
Best average of area and acc time % errors	-15/-15/-15	-1/-1/-1	-26/3/41	14/7/19
Best average of acc time and power % errors	-15/-7/-25	-1/-10/7	-26/-8/5	14/8/12
Best acc time	-29/-29/-29	2/2/2	-43/-25/-2	25/19/11
Best area	-17/-17/-17	-11/-11/-11	-34/-9/24	21/12/17
Best power	-29/-29/-29	2/2/2	-43/-25/-2	25/19/11

Table 22: Error percentages of some prominent solutions generated by CACTI with respect to a 65 nm Intel Xeon L3 cache when we assume that the dynamic power consumed by the cache is 1.7W. The CACTI solutions are for assumptions of "conservative' interconnect projections and 'global' wire type outside mat. We have used 3 activity factors of 0.1, 0.5 and 1, and so each entry in the table has 3 values.

Solution	% error in acc time	% error in area	% error in power	Avg of acc time, area and power % errors
Best % error in acc time	-29/-29/-29	2/2/2	-61/-48/-32	31/27/21
Best % error in area	-17/-17/-17	-11/-11/-11	-55/-37/-14	27/21/14
Best % error in power	-29/-29/-29	2/2/2	-61/-48/-32	31/27/21
Best average of area, acc time and power % errors	-15/-15/-15	-1/-1/-1	-49/-28/-3	22/15/65
Best average of area and acc time % errors	-15/-15/-15	-1/-1/-1	-49/-28/-3	22/15/6
Best average of acc time and power % errors	-7/-15/-15	-10/-1/-1	-55/-28/-3	24/15/6
Best acc time	-29/-29/-29	2/2/2	-61/-48/-32	31/27/21
Best area	-17/-17/-17	-11/-11/-11	-55/-37/-14	27/21/14
Best power	-29/-29/-29	2/2/2	-61/-48/-32	31/27/21

Table 23: Error percentages of some prominent solutions generated by CACTI with respect to a 65 nm Intel Xeon L3 cache when we assume that the dynamic power consumed by the cache is 5.4W. The CACTI solutions are for assumptions of 'conservative' interconnect projections and 'global' wire type outside mat. We have used 3 activity factors of 0.1, 0.5 and 1, and so each entry in the table has 3 values.

References

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

[2] G. Reinman and N. P. Jouppi. CACTI 2.0: An Integrated Cache Timing and Power Model. Technical Report technical report number 2000/7, DEC WRL, 2000.

[3] P. Shivakumar and N. P. Jouppi. CACTI 3.0: An Integrated Cache Timing, Power, and Area Model. Technical Report WRL-2001-2, Hewlett Packard Laboratories, 2001.

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

[5] B. S. Amrutur and M. A. Horowitz. Fast low power decoders for RAMs. *IEEE Journal of Solid-State Circuits*, 36(10):1506–1515, Oct 2001.

[6] R. Ho. *On-chip Wires: Scaling and Efficiency*. PhD thesis, Stanford University, 2003.

[7] Semiconductor Industries Association. International Technology Roadmap for Semiconductors. http://www.itrs.net/, 2005.

[8] P. Bai, C. Auth, S. Balakrishnan, M. Bost, R. Brain, V. Chikarmane, R. Heussner, M. Hussein, J. Hwang, D. Ingerly, R. James, J. Jeong, C. Kenyon, E. Lee, S. Lee, N. Lindert, M. Liu, Z. Ma, T. Marieb, A. Murthy, R. Nagisetty, S. Natarajan, J. Neirynck, A. Ott, C. Parker, J. Sebastian, R. Shaheed, S. Sivakumar, J. Steigerwald, S. Tyagi, C. Weber, B. Woolery, A. Yeoh, K. Zhang, and M. Bohr. A 65nm logic technology featuring 35nm gate lengths, enhanced channel strain, 8 Cu interconnect layers, low-k ILD and 0.57 micron2 SRAM cell. In *IEDM*, Dec 2004.

[9] Semiconductor Industries Association. Model for Assessment of CMOS Technologies and Roadmaps (MASTAR), 2005. http://www.itrs.net/models.html.

[10] S. Thompson, M. Alavi, M. Hussein, P. Jacaob, C. Kenyon, P. Moon, M. Prince, S. Sivakumar, S. Tyagi, and M. Bohr. 130nm Logic Technology Featuring 60nm Transistors, Low-K Dielectrics, and Cu Interconnects. *Intel Technical Journal*, 6(2), May 2002.

[11] Ron Ho. Tutorial: Dealing with issues in VLSI interconnect scaling. In *ISSCC*, 2007.

[12] M. Horowitz, R. Ho, and K. Mai. The future of wires. Technical report, In Invited Workshop Paper for SRC Conference., 1999. Available at http://velox.stanford.edu/.

[13] V. Klee, J. Norum, R. Weaver, S. Iyer, C. Kothandaraman, J. Chiou, M. Chen, N. Kusaba, S. Lasserre, C. Liang, J. Liu, A. Lu, P. Parries, B. Park, J. Rice, N. Robson, D. Shum, B. Khan, Y. Liu, A. Sierkowski, C. Waskiewiscz, P. Wensley, T. Wu, J. Yan, and S. Iyer. A 0.13-micron logic-based embedded DRAM technology with electrical fuses, Cu interconnect in SiLK, sub-7ns random access time and its extension to the 0.10-micron generation. In *IEDM*, 2001.

[14] Y. Matsubara, M. Habu, S. Matsuda, K. Honda, E. Morifuji, T. Yoshida, K. Kokubun, K. Yasumoto, T. Sakurai, T. Suzuki, J. Yoshikawa, E. Takahashi, K. Hiyama, M. Kanda, R. Ishizuka, M. Moriuchi, H. Koga, Y. Fukuzaki, Y. Sogo, H. Takahashi, N. Nagashima, Y. Okamoto, S. Yamada, and T. Noguchi. Fully integration of high density embedded DRAM with 65 nm CMOS technology (CMOS5). In *IEDM*, 2003.

[15] K. Noh, Y. Choi, J. Joo, M. Kim, J. Jung, J. Lim, C. Lee, G. im, and M. Kim. A 130nm 1.1V 143 MHz SRAM-like embedded DRAM compiler with dual asymmetric bit line sensing scheme and quiet unselected IO scheme. In *Symposium on VLSI Circuits*, 2004.

[16] J. M. Tendler, J. S. Dodson, Jr. J. S. Fields, H. Le, and B. Sinharoy. POWER4 system microarchitecture. *IBM Journal of Research and Development*, 46(1):5–26, January 2002.

[17] S. S. Iyer, Jr. J. E. Barth, P. C. Parries, J. P. Norum, J. P. Rice, L. R. Logan, and D. Hoyniak. Embedded DRAM: Technology platform for the Blue Gene/L chip. *IBM Journal of Research and Development*, 49(2/3):333–350, Mar/May 2005.

[18] Masaaki Oka and Masakazu Suzuoki. Designing and Programming the Emotion Engine. *IEEE Micro*, 19(6):20–28, 1999.

[19] Hiroaki Yoshida, Kaushik De, and Vamsi Boppana. Accurate pre-layout estimation of standard cell characteristics. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 208–211, New York, NY, USA, 2004. ACM Press.

[20] B. S. Amrutur and M. A. Horowitz. Speed and power scaling of SRAM's. *IEEE Journal of Solid-State Circuits*, Feb 2000.

[21] J. Chang, M. Huang, J. Shoemaker, J. Benoit, S. Chen, W. Chen, S. Chiu, R. Ganesan, G. Leong, V. Lukka, S. Rusu, and D. Srivastava. The 65-nm 16-MB Shared On-Die L3 Cache for the Dual-Core Intel Xeon Processor 7100 Series. *IEEE Journal of Solid-State Circuits*, 42(4), Apr 2007.

[22] P. Saxena, N. Menezes, P. Cocchini, and D. Kirkpatrick. The scaling challenge: can correct-by-construction help? In *International Symposium on Physical Design*, Apr 2003.

[23] S. Alexenian. Multi-level semiconductor memory architecture and method of forming the same. *US Patent 20050041513*, Feb 2005.

[24] M Mamidipaka and N Dutt. eCACTI: An Enhanced Power Estimation Model for On-chip Caches. Technical Report TR-04-28, Center for Embedded Computer Systems (CECS), 2004.

[25] I. E. Sutherland, R. F. Sproull, and D. Harris. *Logical Effort:Designing Fast CMOS Circuits*. Morgan Kaufmann, San Mateo, CA, 1st edition, 1999.

[26] A. Hajimiri and R. Heald. Design issues in cross-coupled sense amplifier. In *International Symposium on Circuits and Systems*, 1998.

[27] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits*. Prentice Hall, 2nd edition, 2003.

[28] S. Wilton and N. Jouppi. CACTI: An enhanced cache access and cycle time model. *IEEE Journal of Solid-State Circuits*, 31(5):677–688, May 1996.

[29] M. Mamidipaka, K. Khouri, N. Dutt, and M. Abadir. Analytical models for leakage power estimation of memory array structures. In *CODES+ISSS*, 2004.

[30] The MOSIS Service. http://www.mosis.org.

[31] M. Na, E. Nowak, W. Haensch, and J. Cai. The effective drive current of CMOS inverters. In *IEDM*, Dec 2002.

[32] P. Yeh, D. Nayak, and D. Gitlin. Improved CV/I methodology to accurately predict CMOS technology performane. *IEEE Transactions on Electron Devices*, 54(7), Jul 2007.

[33] W. Zhao and Y. Cao. New generation of predictive technology model for sub-45nm design exploration. In *ISQED*, 2006.

[34] R. Mann, W. Abadeer, M. Breitwisch, O. Bula, J. Brown, B. Colwill, P. Cottrell, Jr. W. Crocco, S. Furkay, M. Hauser, T. Hook, D. Hoyniak, J. Johnson, C. Lam, R. Mih, J. Rivard, A. Moriwaki, E. Phipps, C. Putnam, B. Rainey, J. Toomey, and M. Younus. Ultralow-power SRAM technology. *IBM Journal of Research and Development*, 47(5/6), Sep/Nov 2003.

[35] T. Hook, M. Breitwisch, J. Brown, P. Cottrell, D. Hoyniak, C. Lam, and R. Mann. Noise margin and leakage in ultra-low leakage SRAM cell design. *IEEE Transactions on Electron Devices*, 49(8), Aug 2002.

[36] L. Chang, D. Fried, J. Hergenrother, J. Sleight, R. Dennard, R. Montoye, L. Sekaric, S. McNab, A. Topol, C. Adams, K. Guarini, and W. Haensch. Stable SRAM cell design for the 32 nm node and beyond. In *Symposium on VLSI*, 2005.

[37] R Matick and S. Schuster. Logic-based eDRAM: Origins and rationale for use. *IBM Journal of Research and Development*, 49(1), Jan 2005.

[38] G. Wang, P. Parries, B. Khan, J. Liu, Y. Otani, J. Norum, N. Robson, T. Kirihata, and S. Iyer. A 0.168/0.11 micron2 highly scalable high performance embedded DRAM cell for 90/65 nm logic applications. In *Symposium on VLSI Circuits*, 2004.

[39] B. Keeth and R. Baker. *DRAM Circuit Design: A Tutorial.* IEEE Press, 2000.

[40] J. Barth, J. Dreibelbis, E. Nelson, D. Anand, G. Pomichter, P. Jakobsem, M. Nelms, J. Leach, and G. Belansek. Embedded DRAM design and architecture for the IBM 0.11-micron ASIC offering. *IBM Journal of Research and Development*, 46(6), Nov 2002.

[41] J. Barth, W. Reohr, P. Parries, G. Fredeman, J. Golz, S. Schuster3, R. Matick, H. Hunter, C. Tanner III, J. Harig, H. Kim, B. Khan, J. Griesemer, R. Havreluk3, K. Yanagisawa, T. Kirihata, and S. Iyer. A 500MHz random cycle 1.5ns-latency SOI embedded DRAM macro featuring a 3T micro sense amplifier. In *International Solid-State Circuits Conference*, San Francisco, CA, Feb 2007.

[42] Jr. J. Barth, D. Anand, S. Burns, J. Dreibelbis, J. Fifield, K. Gorman, M. Nelms, E. Nelson, A. Paparelli, G. Pomichter, D. Pontius, and S. Sliva. A 500-MHz multi-banked compilable DRAM macro with direct write and programmable pipelining. *IEEE Journal of Solid-State Circuits*, 40(1), Jan 2005.

[43] S. Jin, J. Yi, J. Choi, D. Kang, Y. Park, and H. Min. Prediction of data retention time distribution of DRAM by physics-based statistical simulation. *IEEE Transactions on Electron Devices*, 52(11), Nov 2005.

[44] T. Kirihata, P. Parries, D. Hanson, H. Kim, J. Golz, G. Fredeman, R. Rajeevakumar, J. Griesemer, N. Robson, A. Cestero, B. Khan, G. Wang, M. Wordeman, and S. Iyer. An 800-MHz embedded DRAM with a concurrent refresh mode. *IEEE Journal of Solid-State Circuits*, 40(6), Jun 2005.

[45] H. McIntyre, D. Wendell, K. Lin, P. Kaushik, S. Seshadri, A Wang, V. Sundararaman, P. Wang, S. Kim, W. Hsu, H. Park, G. Levinsky, J. Lu, M. Chirania, R. Heald, P. Lazar, and S. Dharmasena. A 4-MB On-Chip L2 Cache for a 90-nm 1.6-GHz 64-bit Microprocessor. *IEEE Journal of Solid-State Circuits*, 40(1), Jan 2005.

[46] D. Gunadi J. Gilbert, S. Hunt and G. Srinivasa. TULSA, A Dual P4 Core Large Shared Cache Intel Xeon Processor for the MP Server Market Segment. In *Hot Chips*, Aug 2006.

[47] S. Rusu, S. Tam, H. Muljono, D. Ayers, J. Chang, B. Cherkauer, J. Stinson, J. Benoit, R. Varada, J. Leung, R. Limaye, and S. Vora. A 65-nm Dual-Core Multithreaded Xeon Processor With 16-MB L3 Cache. *IEEE Journal of Solid-State Circuits*, 42(1), Jan 2007.

[48] R. Varada, M. Sriram, K. Chou, and J. Guzzo. Design and Integration Methods for a Multi-threaded Dual Core 65nm Xeon Processor. In *ICCAD '06*, 2006.

[49] S. Rusu, S. Tam, H. Muljono, D. Ayers, and J. Chang. A Dual-Core Multi-Threaded Xeon Processor with 16MB L3 Cache. In *ISSCC*, San Francisco, CA, February 2006.