Destructive-Read in Embedded DRAM, Impact on Power Consumption

Haakon Dybdahl dybdahl@idi.ntnu.no
Per Gunnar Kjeldsberg per.gunnar.kjeldsberg@iet.ntnu.no
Marius Grannæs grannas@idi.ntnu.no
Lasse Natvig lasse@idi.ntnu.no
Norwegian University of Science and Technology, NO-7491 Trondheim, Norway

***Abstract*— This paper explores power consumption for destructive-read embedded DRAM. Destructive-read DRAM is based on conventional DRAM design, but with sense amplifiers optimized for lower latency. This speed increase is achieved by not conserving the content of the DRAM cell after a read operation. Random access time to DRAM was reduced from 6 ns to 3 ns in a prototype made by Hwang et. al. A write-back buffer was used to conserve data. We have proposed a new scheme for write-back using the usually smaller cache instead of a large additional write-back buffer. Write-back is performed whenever a cache line is replaced. This increases bus and DRAM bank activity compared to a conventional architecture which again increases power consumption. On the other hand computational performance is improved through faster DRAM accesses. Simulation of a CPU, DRAM and a 2 kbytes cache show that the power consumption increased by 3% while the performance increased by 14% for the applications in the SPEC2000 benchmark. With a 16 kbytes cache the power consumption increased by 0.5% while performance increased by 4.5%.**

Keywords: ***Embedded DRAM, power estimation, Simplescalar, destructive-read memory, processing in memory***

I. Introduction

Main memory and *central processing units* (CPUs) have both become increasingly powerful during the last 30 years, but their progress have taken different directions. CPUs have got faster clock cycles and more computations per clock cycle while main memory can store more data. Today there is a magnitude of difference in cycle time between main memory and CPUs, often referred to as the *processor memory performance gap.* Reducing the effect of this gap has been a main research objective for decades. This has resulted in various mechanisms such as caches, out-of-order scheduling, prefetchers and simultaneous multithreading. These mechanisms consume a substantial amount of power and are thus a challenge when designing battery driven equipment, but also for design of high performance CPUs. For systems with multiple cores on a single chip the overall power usage limits the performance and/or the number of cores that can be integrated.

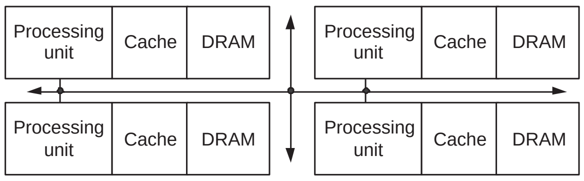


Fig. 1. Multiple cores and memory in a single chip.

The target architecture is shown in Figure 1. Each processor is relatively simple and has its own cache, DRAM banks and a communication channel to the other processors. Hwang et al. [10] made a prototype which enables lower latencies in DRAM by modifying the sense amplifiers to omit write-backs. Reading a memory cell thus destroys its content and the only copy now exists in a write-back buffer. To ensure that later write-backs of data to DRAM do not inflict a performance penalty, this buffer must at least be as large as the DRAM bank size. We have proposed a new scheme for write-back utilizing the usually smaller cache instead of this large additional write-back buffer [5]. In our scheme the size of the cache is not dependent on DRAM bank size. Simulations of a system with 2 kbytes cache show that speed is improved by 14% with our approach compared to conventional DRAM. Due to the increased speed of the DRAM, the cache size can be reduced by a factor four in a system with destructive-read DRAM compared to conventional DRAM without degrading performance. Our previous work has not considered energy consumption, and this is the topic for this work.

The concept of destructive-read DRAM is explained in Section II. The models used for power estimation is described in Section III. Section IV describes the simulations that are run and Section V describes the results. Section VI is discussion and Section VII conclusions followed by references.

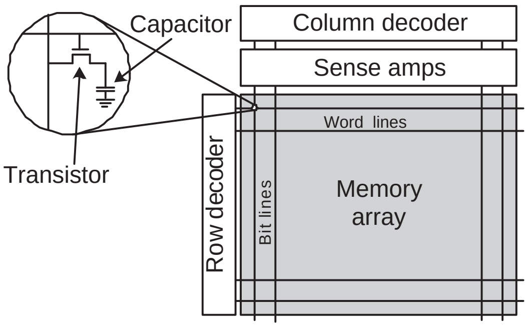


Fig. 2. Inside a DRAM bank.

Type	Access Time
PC133 SDRAM	71.4nS
DDR266 DRAM	58.8nS
RL DRAM	25.0nS
130nm embedded DRAM	12.0nS
90 nm embedded DRAM Dense	8.0nS
90 nm embedded DRAM Fast	4.5nS
DDR SRAM	3.3nS
Embedded SRAM	2.0nS

TABLE I

Random cycle time for various memories [11].

II. Embedded Destructive-Read DRAM

A. Embedded Memory

DRAM is cheap, dense and consumes little power compared to other technologies; therefore it is the main choice for main memory. DRAM is traditionally found on separate chips and connected to the CPU through off-chip buses. These off-chip buses introduce capacitance which again increases power consumption and latency. The number of pins available on the CPU packages introduces a practical limit for the bus width. By merging main memory and CPUs, this external bus is eliminated. Main memory has a much higher internal bandwidth than what is available through the off-chip buses. By utilizing embedded DRAM the internal bandwidth of main memory is available to the CPU. However, the process of merging main memory and central processing unit is not trivial as DRAM uses capacitors to store data (see Figure 2). DRAM chips are optimized for these analog circuits. Logic circuits on the other hand are optimized for speed and power distribution. As a consequence, embedded DRAM is not as dense (bits per area) as conventional DRAM chips. The actual density depends on the technology used [13]. The most sophisticated technology combines processes from DRAM manufacturing and CMOS logic chip manufacturing while the simplest generates the cells in pure CMOS. An additional advantage of embedded DRAM, compared to the normally faster embedded SRAM, is that the standby leakage power is much smaller. This factor becomes increasingly important as the technology continues to shrink below 180nm [12].

B. Related Work

Several projects have done research into merging processors and memory ([28], [4], [6], [7], [15], [17], [20], [22], [23], [30]). Most of these projects assume a conventional DRAM design. The C*RAM project [6] is an exception where small processing elements are integrated into the sense amplifiers, utilizing the parallelism available at that level. A scaled down prototype was made. It was a SIMD computer with single bit processors. This architecture is only suitable for problems with high data locality because of limited communication between the single bit processing elements.

Many other projects use SIMD architectures to utilize the extra bandwidth: e.g. the IRAM project [26], Yukon [17], Terasys [7] and Execube [19]. The Mitsubishi M32R/D chip [22] and Saulsbury et. al [28] use the bandwidth to increase the number of bits in the data bus between main memory and cache. FPGA and independent processors have also been proposed to utilize the bandwidth ([4], [20]). During the last few years, embedded DRAM has become more common, and chips are in mass production with this type of memory integrated with graphics or network processors such as Sony's Playstation 2, Xbox 360, EZchip's NP-1c [8] network processor and Nintendo's GameCube. Embedded DRAM is becoming commercially available at different speeds. Table I contains random access latency time for various memory technologies. By reducing the size of each memory bank, a smaller unit is activated during an access. As will be explained in Section III, the bank size has a large influence on the power consumption.

A comprehensive study of different aspects of memory and data intensive design can be found in [3] and [24].

C. Destructive-Read DRAM

Destructive-read DRAM [10] is a modified version of conventional DRAM. A memory bank with conventional DRAM is shown in Figure 2. A charged capacitor (normally) represents the logic high value, while an uncharged capacitor represents the logic low value. The row decoder is the first component activated in a read access. It enables one *word line* and causes all transistors in that row to be activated. These transistors connect the capacitors in the memory array to the *sense amplifiers* through *bit lines.* The bus out of a DRAM macro often has fewer lines than the number of bit lines inside the macro. The column decoder controls which subset of bit lines that are read or written. The sense amplifiers work in three phases as shown

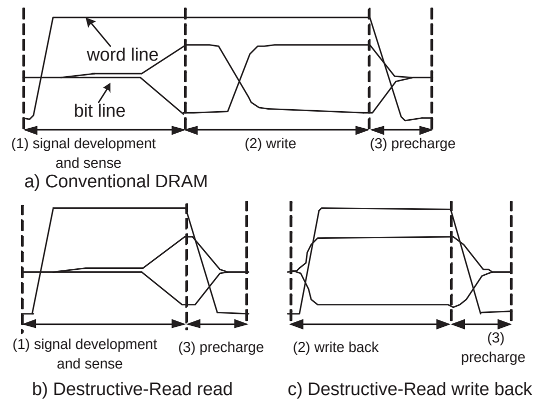


Fig. 3. Conceptual waveform diagrams of conventional DRAM architecture vs. destructive-read [10].

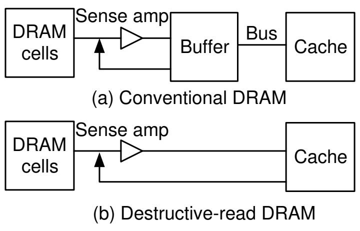


Fig. 4. Conceptual view of DRAM.

in Figure 3a. In the first phase the charge (or lack of charge) from the capacitor drives the sense amplifier into a logic high (or low), in both cases leaving the capacitor discharged. In the second phase the logic state is locked. In Figure 4a the locking works as a buffer. From this buffer the data is both sent to the processor and written back to memory. In the final phase the bit lines are precharged so they are ready for the next access. Destructive-read DRAM memory works differently. The read operation of conventional DRAM (see Figure 3a) is split into two cycles (see Figure 3b and c). The destructive-read DRAM does not lock the data after reading (as shown in Figure 4b). Instead the data are sent directly out of the chip, in this case to cache memory. Since data is not sent back to memory, the capacitor is left discharged and data is destroyed. Data is conserved by writing it back to DRAM later as shown in 3c. However, write-back is not performed immediately after the read, this in contrast to conventional DRAM where read and write-back are a single operation.

D. Write-backs

Hwang et al. made a prototype where random access time to DRAM was reduced from 6 ns (conventional read) to 3 ns (destructive-read). The prototype had four independent memory banks and a large write back buffer (WBB) that was the same size as one memory bank. The WBB was made out of SRAM. The purpose of the WBB was to hide write-backs, not to reduce latency. The WBB could write-back to several banks simultaneously and required significant chip area. Later a new scheme was made where the WBB was replaced with destructive-read DRAM [14]. The designs guaranteed that write-backs never conflicted with read operations. We have proposed new schemes that remove the large write-back buffer and increase performance by utilizing the cache [5]. The data is first read from DRAM and into the cache. At this time data are not stored in DRAM, only in the cache. Data are written back to DRAM when the cache line is replaced. In a conventional scheme data is only written back if data is modified, while in this scheme data is always written back. Therefore this scheme can be compared to a cache that always has dirty cache lines. Write-backs are partially hidden by using several memory banks so data can be read from one bank while writing to a different bank. However, in some cases the read operation and the write back access the same memory bank, causing a delay. Figure 5 shows the number of instructions per clock cycle (IPC) for a system with conventional DRAM and our destructive-read DRAM scheme, both with a 2 and 16 kbytes cache for the applications in SPEC2000 benchmark suite.

III. Model for Power Consumption

Power consumption in computers can be divided into *static* and *dynamic* power consumption. Dynamic power consumption for a CMOS chip is shown in Equation 1.

$$P_{dynamic} = C_{switched} * V_{dd}^2 * f_{clk} \qquad (1)$$

V_{dd} is supply voltage, f_{clk} is frequency and $C_{switched}$ is the total effective switched capacitance, i.e. is the average capacitance of the transistors and communication lines that are switch in each clock cycle. In synchronous designs such as a microprocessor the clock distribution adds a significant value to $C_{switched}$.

As chips get denser, leakage power increases, and transistors consume more power without switching [12]. This is called static power consumption. Caches have higher leakage currents per stored bit compared to DRAM because of higher transistor count, and will therefore consume more power in denser technologies compared to DRAM.

A. Voltage and Frequency

The following relationship between frequency and power has been described by Khellah and Elmasry [16]:

$$T_d \approx \frac{C_L * V_{dd}}{\kappa * (V_{dd} - V_{th})^{\bar{\alpha}}} \qquad (2)$$

T_d is the delay of a CMOS gate, where C_L is the load capacitance, κ is a factor that depends on the process

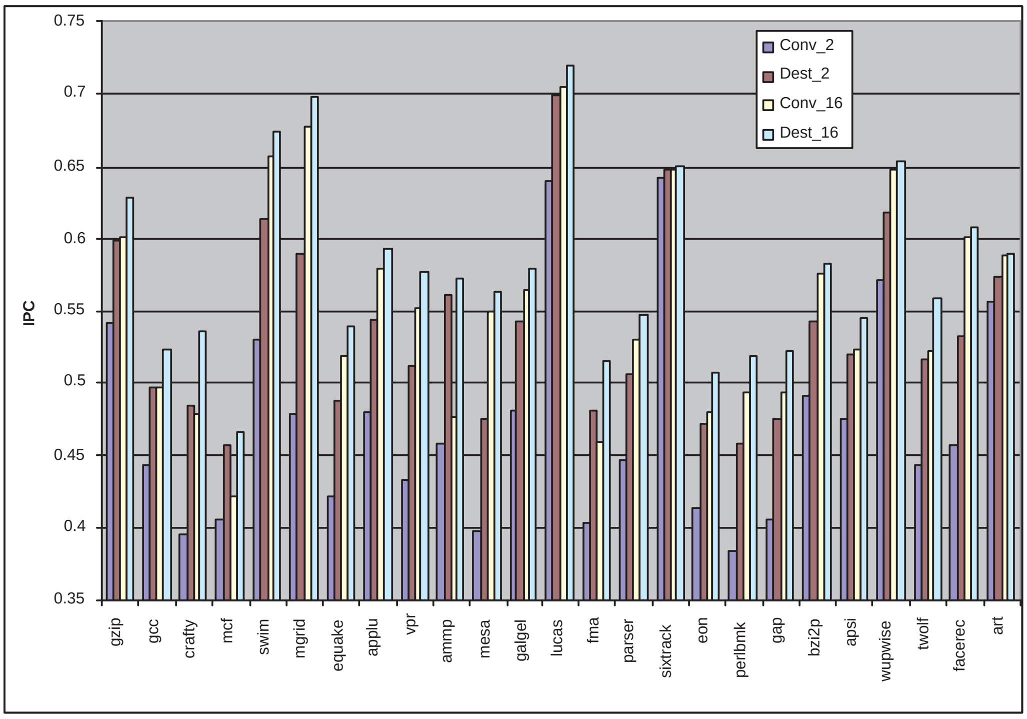


Fig. 5. IPC for the applications in the SPEC2000 benchmark suite for conventional (conv) and destructive-read (dest) DRAM with 2 and 16 kbytes cache configuration. The average IPC is 13.7% higher for *dest_2* compared to *conv_2*, and 4.5% higher for *dest_16* compared to *conv_16*.

and gate size, $\bar{\alpha}$ takes a value between one and two, V_{th} is the threshold voltage and V_{dd} is the supply voltage. Even though the formula is not complicated, a lot of complexity is hidden in the $\bar{\alpha}$, V_{th}, C_L and κ factors. However, what can be read from this formula is that for a given technology and circuit, the maximum operating frequency is a function of supply voltage. This fact is used for power saving in commercial computer systems in a technique called dynamic voltage-frequency scaling. The technique reduces both frequency and voltage for the system when maximum computational performance is not needed.

B. Power model of DRAM with bus

The data flow of accessing DRAM from cache is shown in Figure 6. The address and control signals are sent to the DRAM bank. Data are read out of the bank and written back along the route shown as a thick gray line. Equation 3 shows a model for the energy consumption (E_{MEM}) for these components for the execution of a complete application, e.g. one of the benchmarks from Figure 5.

$$E_{MEM} = N_{ACC} * (E_{DRAM} + E_{SWITCH} + E_{BUS}) \quad (3)$$

N_{ACC} is the number of DRAM memory accesses, and for a single access: E_{DRAM} is energy consumed in the DRAM banks, E_{BUS} is energy consumed by the bus

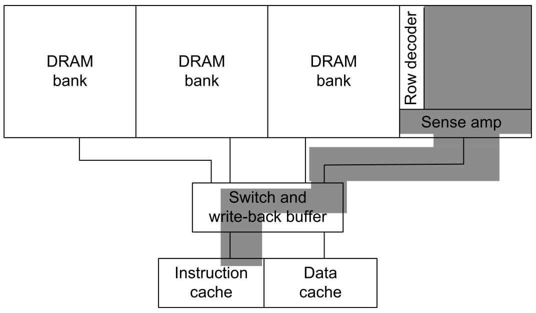


Fig. 6. Data communication when accessing a memory bank.

and E_{SWITCH} is energy consumed by the switch and the write-back buffer.

Yong-Ha Park et. al. [25] have made a model for the dynamic power consumption of embedded DRAM at an abstraction level that matches the simulation model that we are using for the processor. We have based Equation 4 on this model.

$$E_{DRAM} = N_{ROW} * E_{ROW} + N_{COL} * E_{COL} \quad (4)$$

N_{ROW} is the number of row activations, N_{COL} is the number of column activations, E_{ROW} is energy consumed by activation of a row in the embedded DRAM and E_{COL} is energy consumed by activation of a column in the embedded DRAM. All these variables are for a single memory access. Burst transfer is not used in this work because communication is on-chip and the bus width is increased instead. In this way the column decoder is not needed as one word line of the memory array is read or written simultaneously. Refresh of DRAM is not different for the two DRAM models and is therefore omitted for simplicity. The result is that N_{COL} is one and N_{ROW} is the number of data bits which is 512 in our model. The consequence is that E_{DRAM} is equal for all memory accesses because each access activates one row and 512 columns. In order to quantify E_{DRAM} we use a DRAM macro made by Morishita et.al [21]. The power consumption is 0.260 W at 250 MHz (in 130 nm technology) for accessing a 16 Mb bank. This is approx. 1 nJ for one access with 128 bits, and we conservatively multiply this with 4 for 512 bits width resulting in $E_{DRAM} = 4nJ$ for our DRAM bank. Power consumption depends on DRAM macro size as this impacts wire lengths, the number of cells activated in parallel, etc. Smaller macros consume less power while larger macros consume more power per access. Research on interconnection is a large topic and several techniques exists (see for example [27]). Ron Ho et.al. [9] have made efforts to quantify energy consumption for buses, and found that a wire of length 10 mm in 180 nm with 1.8 volt require $<$ 1 pJ per bit for techniques with voltage swing reduction and $<$ 10 pJ for a simple bus wire. We conservatively use the simple wire and 10 mm bus (10 pJ). This result in $E_{BUS} = 5.4pJ$ for 544 bus lines (512 data + 32 address).

The power consumption for the write-back buffer and switch was modeled as a 2 kB cache with four banks and 64 bytes block size in CACTI [29] (in 180 nm technology). It consumes 0.31 nJ per bank per access. We conservatively use $E_{SWITCH} = 1nJ$.

This results in $E_{MEM} = 10.5nJ * N_{ACC}$. The same energy model is applied to both destructive-read DRAM and conventional DRAM, and used for both read and write operation. This is acceptable as our main goal is to compare the two techniques, not necessarily to have exact estimates. There are fewer operations performed in the destructive-read DRAM since no write-back is performed during each read. Static power consumption is not included in this power model as it is not different for conventional and destructive-read DRAM. Using destructive-read DRAM results in shorter execution time and the static power consumption will be slightly lower. If in error, the power consumption penalty of using the destructive-read DRAM will therefore be overestimated.

IV. Simulations

The purpose of our simulations is to study the energy consumption and performance of the destructive-read DRAM and compare this to conventional DRAM.

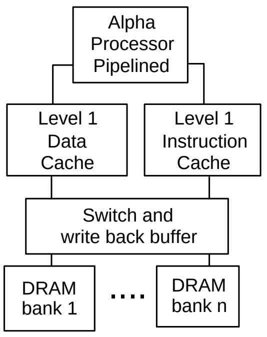

Alpha Processor Pipelined
Level 1 Data Cache
Level 1 Instruction Cache
Switch and write back buffer
DRAM bank 1
....
DRAM bank n

Fig. 7. The simulated single chip computer.

For simplicity, only one node of the parallel architecture is simulated. The simulator is based on SimpleScalar version 3 [1]. It is extended with Wattch [2] and HotLeakage [31]. We have further extended it to simulate a configurable number of DRAM banks with congestion and a configurable stand alone write-back buffer. A logical sketch of the simulated computer is shown in Figure 7.

Wattch with *HotLeakage* contain parameters for power consumption for different technologies and the simulation model is fully configurable. However, they do not contain values for DRAM and memory buses to DRAM. For this, the estimate of 10.5 nJ per access from the previous section is used. The configurations for the baseline of the simulations are cycle-true simulation with a five stage Alpha processor at 180 nm technology. Processor properties are single issue, static branch prediction, no translation look-aside buffer, in-order execution, single decode, single commit and single ALU. There are two independent caches, one instruction cache and one data cache each with 64 bytes cache lines, two ways set associative, and cache size of one kbytes each. Latency is one clock cycle. Eight independent memory banks are used for simulation of congestion. Memory bus width is the same as cache line width (64 bytes). Latency of conventional DRAM is six clock cycles for a read operation. For destructive-read, this is three clock cycles for reading and three clock cycles for writing. DRAM refresh is not simulated as it is presumed to have little and similar effects on the result for both configurations. Total access time for a cache miss includes one clock cycle for cache plus access time for DRAM. A write-back buffer is implemented for each memory bank capable of storing one cache line.

SPEC2000 applications were used as benchmark with *lgred* (large reduced input dataset) [18] as the data set. One of the 26 applications found in the *SPEC2000* did not work with the simulator (*vortex* application).

V. Results

The energy consumption for the various SPEC2000 benchmarks is shown in Figure 8. The total energy consumption level shows little difference between the two models. However, for the destructive-read DRAM model, a larger part of the energy is consumed in the DRAM. On average the energy consumption is increased with 0.46% for destructive-read DRAM compared to conventional DRAM. As will be shown later, this is compensated by a much larger increase in performance.

Details of the energy consumption components for *gcc* is shown in Figure 9. Since the destructive-read DRAM model results in less computational time, less energy is spent on clock distribution and other active waiting components. On the other hand more energy is spent on DRAM components.

The average number of DRAM accesses per clock cycle is shown in Figure 10. The number of accesses is more than doubled for some applications. This is because more instructions are executed per clock cycle due to lower memory latency. Other applications show little increase, as more data is modified by the processor and has to be written back in both schemes.

Three of the applications from the SPEC2000 benchmark were selected for further study: *art, ammp* and *twolf*. *art* was chosen because of the small energy consumption in DRAM, *ammp* was chosen due to high amount of DRAM energy usage and *twolf* was chosen as an average application.

Performance and energy consumption for *twolf* as function of cache size is shown in Figure 11c). Optimizing for low execution time gives larger cache sizes and optimizing for low energy consumption results in cache size of 4 kbytes. The destructive-read architecture consumes more energy than the conventional model for this application. The difference is largest for small caches. This is due to reduced DRAM traffic for larger caches caused by lower miss-ratio in the cache. Performance is better for destructive-read DRAM, especially for small caches. The product of execution time and energy consumption is shown in Figure 12c). A cache of size 16 kbytes is the optimal for minimizing (linearly) both energy consumption and execution time.

For the *art* application the energy consumption and execution time are shown in Figure 11b). For small caches there is a difference while for caches larger than 4 kbytes there is little difference. The product of execution time and energy consumption is shown in Figure 12b). Caches of 4 kbytes is the optimal size for minimizing (linearly) both energy consumption and execution time.

A study of performance and energy consumption for *ammp* application as a function of cache size is shown in Figure 11a). For larger caches the destructive-read model requires less energy than the conventional model. This is due to shorter execution time. The product of execution time and energy consumption is shown in Figure 12a). Caches of 2 kbytes is the optimal size for minimizing (linearly) both energy consumption and execution time.

VI. Discussion

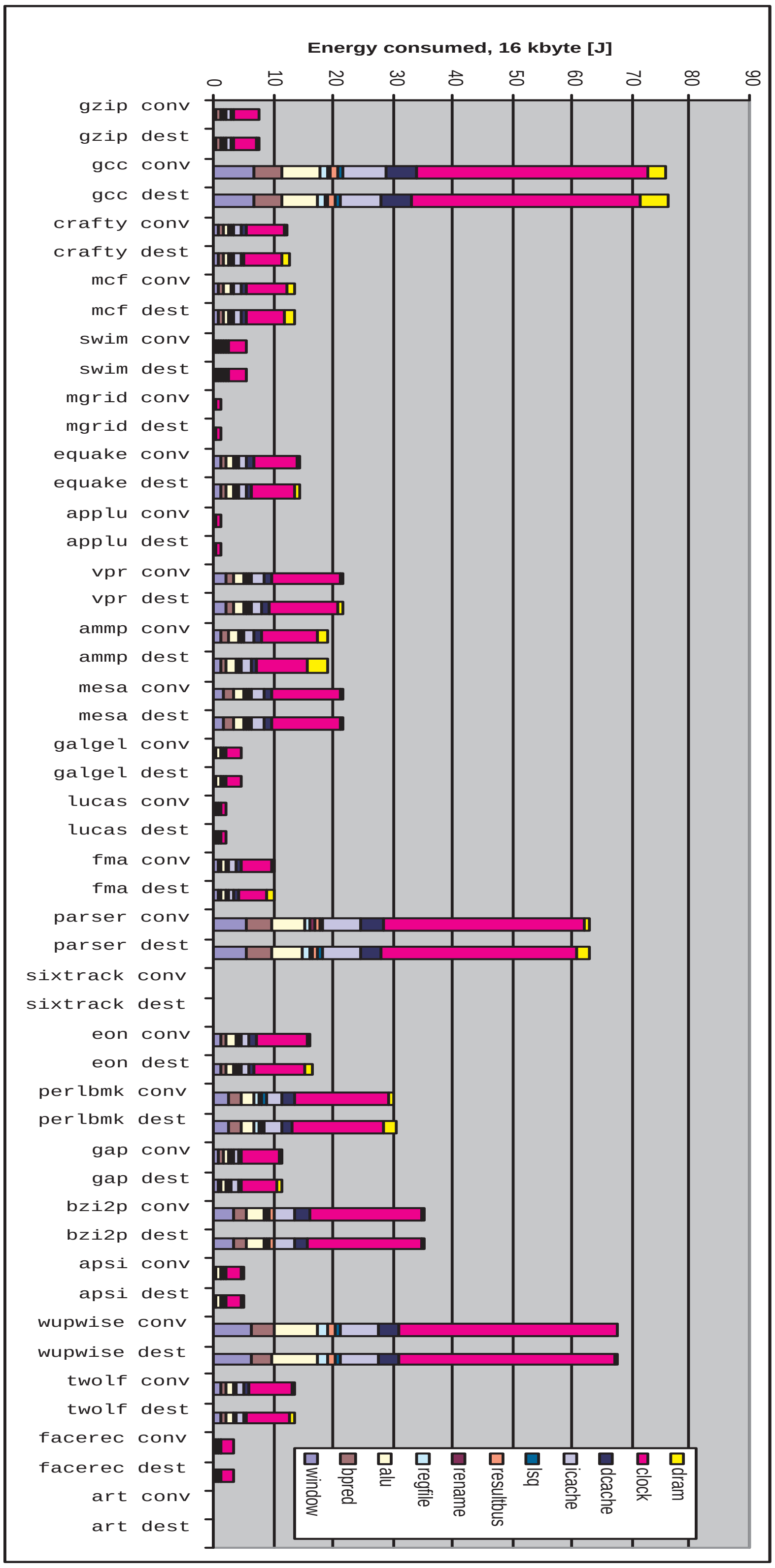
It is assumed that $E_{MEM} = 10.5nJ$ is consumed for each access to the DRAM subsystem. This assumption influences the power consumption and not the computational speed (IPC). The impact of this value depends on factors such as cache miss ratio and the power consumption of the processor. We have assumed that energy consumption is proportional with the number of DRAM accesses plus the energy consumption in processor and cache. For the *twolf* application with 16 kbytes cache the power consumption is shown in Equations 5 and 6. The values are taken from simulations. By looking at the components in Equation 5 and comparing this to Equation 6 it can be seen that a system with conventional DRAM uses more energy in the processor and less in DRAM memory compared to destructive-read DRAM, and vice versa. The number of DRAM accesses is increased from 58 million to 97 million with the destructive-read DRAM configuration, an increase of 70%. The processor itself consumes 2% less energy since it executes the task in less time. With higher leakage currents in denser technologies this difference will be even more significant.

$$E_{conv} = 13.2J + 58 * 10^6 * E_{MEM} \tag{5}$$

$$E_{dest} = 12.9J + 97 * 10^6 * E_{MEM} \tag{6}$$

For the simulated architecture, technology, and applications, destructive-read DRAM has slightly higher power consumption. Smaller DRAM memories will result in a smaller E_{MEM}, and can result in *less* energy consumption for DRAM due to shorter computation time. The result is also dependent on processor architecture: More complex processor will require more static power. Also, denser technology will have more leakage current and computation time will be more important. These estimates should be conservative since static power is still not significant at 180 nm technology.

There are several benefits by using destructive-read DRAM. (1) Cache size can be reduced by a factor four, decreasing chip area correspondingly [5]. This has a positive impact on power because each access to the smaller cache consumes less energy. In dense technologies with high leakage currents this will be even more the case, since caches have many transistors. (2) Power consumption in CPU is also reduced since computation time is reduced. However, since the number of accesses to the DRAM is increased the total power consumption is also increased. In our simulation models power is increased by 0.5% and 3% for 16 and 2 kbytes cache respectively. For the same execution time, the frequency and voltage can be scaled down when



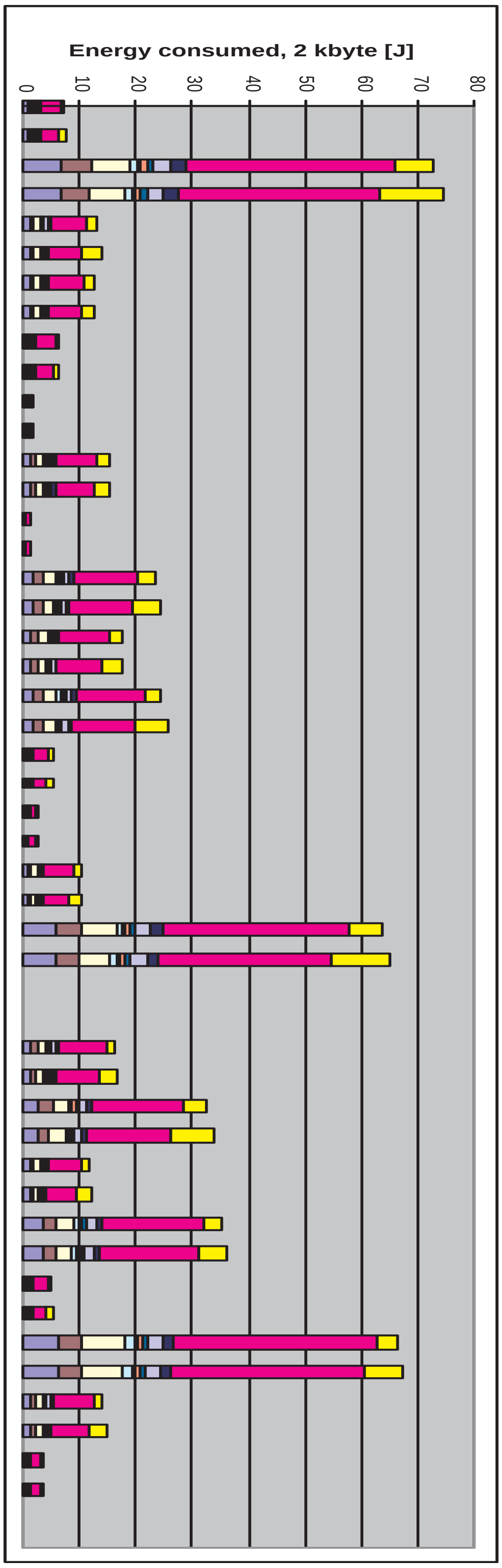


Fig. 8. Energy consumption for various applications in the SPEC2000 suite for conventional DRAM (conv) and destructive-read DRAM (dest). Total cache size is 16 kbytes to the left and 2 kbytes to the right. Average increase in power consumption from conventional to destructive are 0.5% and 3% for 16 kbytes and 2 kbytes caches respectively. The graphs of *sixtrack* and *art* are invisible due to short execution time.

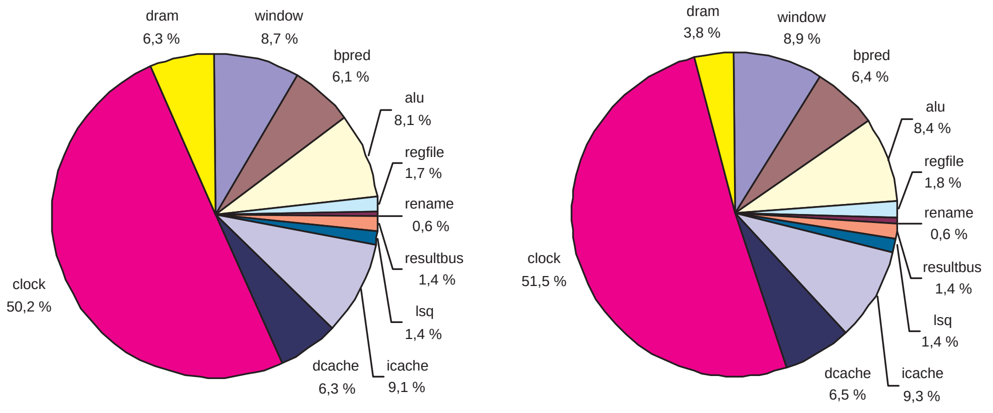


Fig. 9. Energy consumption for *gcc* for destructive-read DRAM to the left and conventional DRAM to the right. Cache size is 16 kbytes.

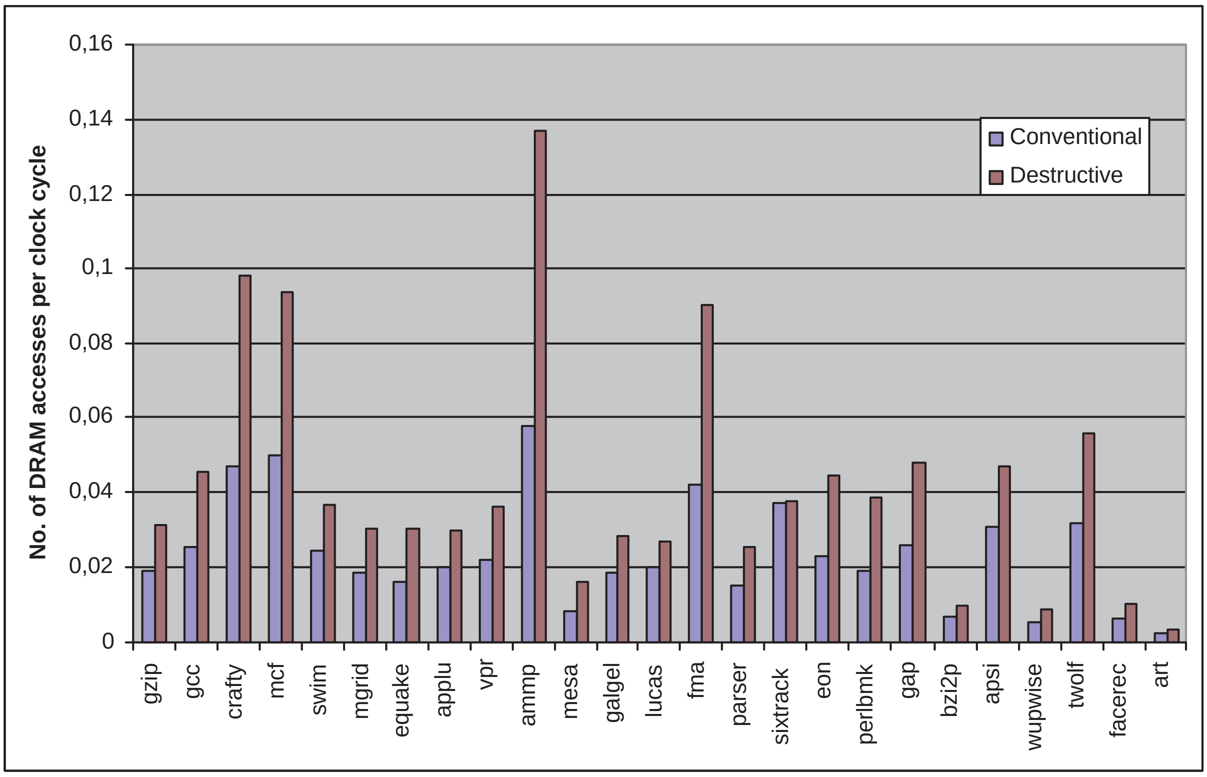


Fig. 10. Number of DRAM accesses per clock cycle for 16 kbytes cache. Due to increased IPC the number of DRAM accesses can be more than doubled for each clock cycle.

utilizing destructive-read DRAM since the instructions per clock cycle (IPC) is higher. This reduces power consumption, but is not analyzed in this paper. (3) The performance is increased, in our simulation this is 5% with 16 kbytes caches and 14% with 2 kbytes caches.

In systems with multiple processors and shared memory with cache coherence protocol, the cache line has to be marked dirty when read so that data is conserved. For multiple processors with message passing the data has to be read through the corresponding cache, i.e. the local processor should process the messages in order to conserve the data.

VII. Conclusions

Destructive-read DRAM looks promising both from an energy and a speed perspective. More energy is spent on the DRAM memory and bus, but the execution time is reduced and energy is saved in the processor. Denser technologies with higher leakage currents will benefit even more as more energy is wasted when

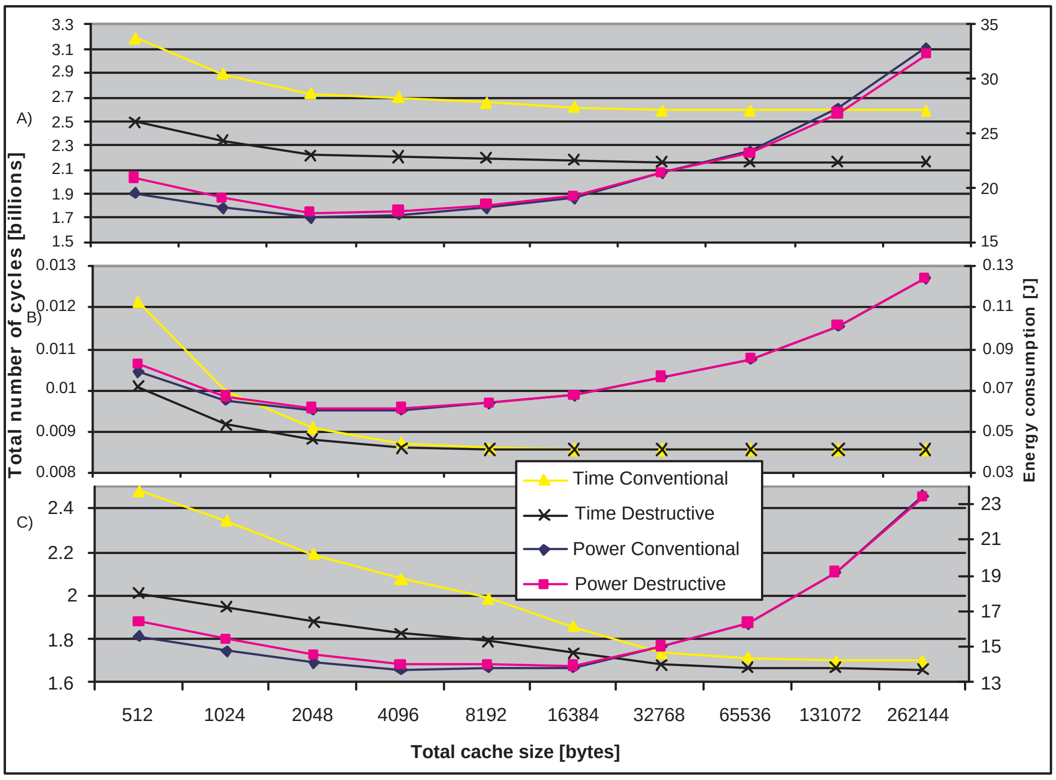


Fig. 11. Number of clock cycles and energy consumption for for a) *Ammp.*, b) *Art* and c) *Twolf*.

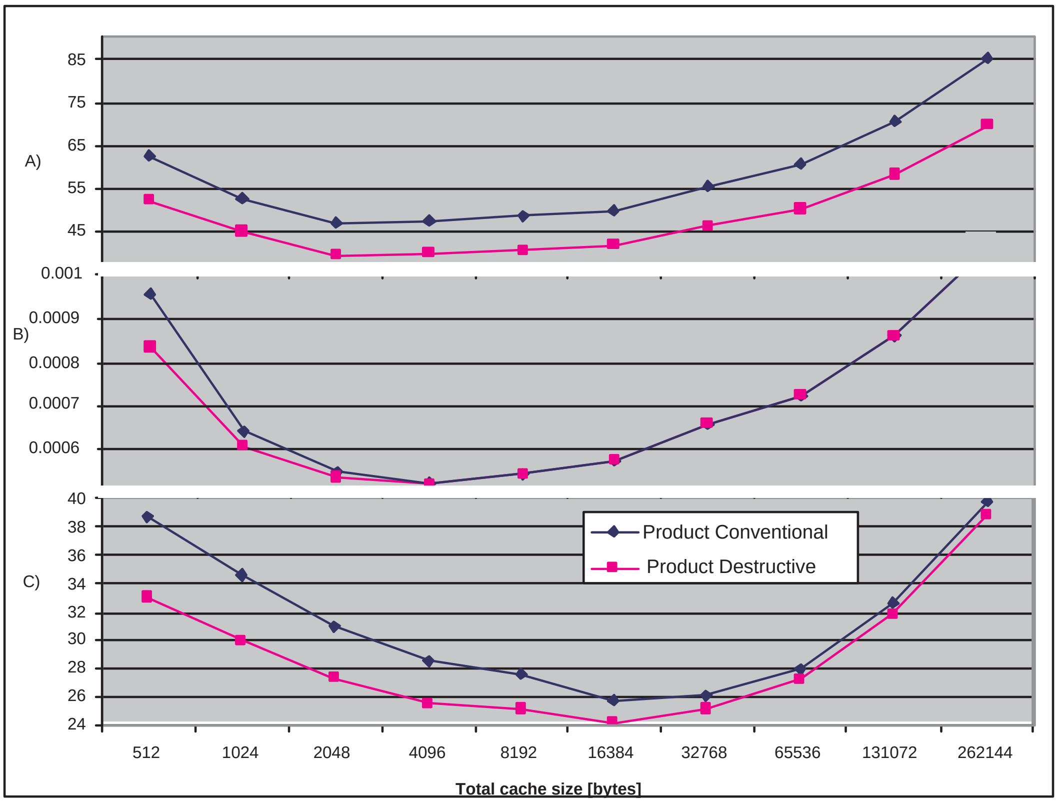


Fig. 12. Product of execution time and energy consumption for a) *Ammp.*, b) *Art* and c) *Twolf*

the processor is stalled waiting for memory access to finish. For the simulated architecture the average power consumption were increased with 0.5% and 3% while the performance were increased with 4.9% and 14% for the applications in SPEC2000 benchmark for 16 kbytes and 2 kbytes caches respectively. Benefits from using destructive-read DRAM are increased performance and a reduction of chip area (by reducing cache size). Lower energy consumption might be possible through voltage-frequency scaling, but this also depends on the configuration and technology used. With higher leakage currents in denser technologies the destructive-read DRAM will be even more beneficial as the leakage currents (static power consumption) is higher.

References

[1] T. Austin, E. Larson, and D. Ernst. SimpleScalar: an infrastructure for computer system modeling. IEEE Computer, Volume 35, Issue2, February 2002.

[2] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *ISCA '00: Proceedings of the 27th annual International Symposium on Computer Architecture*, pages 83–94, New York, NY, USA, 2000. ACM Press.

[3] F. Catthoor, K. Danckaert, C. Kulkarni, E. Brockmeyer, P. G.Kjeldsberg, T. Van Achteren, and T. Omnes. *Data Access and Storage Management for Embedded Programmable Processors.* Kluwer Acad. Publ., Boston, USA, 2002. ISBN 0-7923-7689-7.

[4] Jeff Draper, Chang Woo Kang, Ihn Kim, Gokhan Daglikoca, Jacqueline Chame, Mary Hall, Craig Steele, Tim Barrett, Jeff LaCoss, John Granacki, Jaewook Shin, and Chun Chen. The architecture of the DIVA processing-in-memory chip. Proc. 16th ACM Int'l Conf. Supercomputing, pages:14-25, June 2002.

[5] H. Dybdahl, M. Grannæs, and L. Natvig. Cache write-back schemes for embedded destructive-read DRAM. Submitted to ARCS 2006, 2006.

[6] D. G. Elliott, W. Martin Snelgrove, and Michael Stumm. Computational RAM: A memory-SIMD hybrid and its application to DSP. In Custom Integrated Circuits Conference, Boston, MA, pages:30.6.1-30.6.4, May 1992.

[7] M. Gokhale, B. Holmes, and K. Iobst. Processing in memory; the Terasys massively parallel PIM array. IEEE Computer, pages:23-31, April 1995.

[8] Linley Gwennap. Embedded DRAM use rises. Nikkei Electronics Asia, June, June 2003.

[9] R. Ho, K. Mai, and M. Horowitz. Efficient on-chip global interconnects. IEEE Symposium on VLSI Circuits, 2003.

[10] Chomg-Lii Hwang, T. Kirihata, M. Wordernan, J. Fifield, D.Storaska, D. Pontius, G. Fredernanand B. Ji, S. Tomashot, and S. Dhong. A 2.9ns random access cycle embedded DRAM with a destructive-read. VLSI Circuits Digest of Technical Papers, Symposium on, pages:174-175, June 2002.

[11] IBM microelectronics presentation: Embedded DRAM comparison charts. IBM Microelectronics, 2003.

[12] International technology roadmap for semiconductors, 2003. `http://public.itrs.net/`.

[13] 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 J. Res & Dev. vol 49 no. 2/3 March/may, 2005.

[14] B.L. Ji, S. Munetoh, C-L. Hwang, M. Wordeman, and T. Kirihata. Destructive-read random access memory system buffered with destructive-read memory cache for SoC applications. VLSI Circuits, Digest of Technical Papers. Symposium on, pages:85 - 88, June 2003.

[15] Y. Kang, W. Huang, S.-M. Yoo, D. Keen, Z. Ge, V. Lam, P. Patnaik, and J. Torellas. FlexRAM: Towards an advanced intelligent memory system. International Conference on Computer Design, October 1999.

[16] M.M. Khellah and M.I. Elmasry. Power minimization of high-performance submicron CMOS circuits using a dual-vdd dual-vth (DVDV) approach. ACM Int'l Symp. Low-Power Electronics and Design, pages:106-108, 1998.

[17] G. Kirsch. Active memory: Micron's Yukon. Parallel and Distributed Processing Symposium, Proceedings. International, pages:11, April 2003.

[18] AJ KleinOsowski and David J. Lilja. MinneSPEC: A new SPEC benchmark workload for simulation-based computer architecture research. *Computer Architecture Letters*, 1, June 2002.

[19] P.M. Kogge, T. Sunaga, H. Miyataka, K. Kitamura, and E. Retter. Combined DRAM and logic chip for massively parallel systems. IEEE, Advanced Research in VLSI, Proceedings. Sixteenth Conference on, pages:4-16, March 1995.

[20] K. Mai, T. Paaske, N. Jayasena, W R. Ho, Dally, and M. Horowitz. Smart Memories: A modular reconfigurable architecture. ISCA, June 2000.

[21] F. Morishita, I. Hayashi, H. Matsuoka, K. Takahashi, K. Shigeta, T. Gyohten, M. Niiro, H. Noda, M. Okamoto, A. Hachisuka, A. Amo, H. Shinkawata, T. Kasaoka, K. Dosaka, K. Arimoto, K. Fujishima, K. Anami, and T. Yoshihara. A 312-MHz 16-Mb random-cycle embedded DRAM macro with a power-down data retention mode for mobile applications. Solid-State Circuits, IEEE Journal of, Vol.40, Iss.1, Pages: 204- 212, 2005.

[22] Y. Nunomura, T. Shimizu, and O. Tomisawa. M32R/D-integrating DRAM and microprocessor. Micro, IEEE, Volume: 17, Issue: 6, pages:40-48, November 1997.

[23] M. Oskin, F.T. Chong, and T. Sherwood. Active Pages: A model of computation for intelligent memory. International Symposium on Computer Architecture, Barcelona, Spain, 1998.

[24] P.R. Panda, F. Catthoor, N. D. Dutt, K. Danckaert, A. Vandercappelle E. Brockmeyer, C. Kulkarni, and P.G. Kjeldsberg. Data and memory optimization techniques for embedded systems. ACM Trans. Design Automation of Electronic Systems, 6(2):149–206, April 2001.

[25] Yong-Ha Park, Hoi-Jun Yoo, and Jeonghoon Kook. Embedded DRAM (eDRAM) power-energy estimation for system-on-a-chip (SoC) applications. Proceedings of the 15th International Conference on VLSI Design (VLSID), p. 625, ASP-DAC/VLSI, 2002.

[26] D. Patterson, T. Anderson, and K. Yelick. A case for intelligent DRAM: IRAM. Presented at Hot Chips VIII, Palo Alto CA, pages:18-20, August 1996.

[27] Vijay Raghunathan, Mani B. Srivastava, and Rajesh K. Gupta. A survey of techniques for energy efficient on-chip communication. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 900–905, New York, NY, USA, 2003. ACM Press.

[28] A. Saulsbury, F. Pong, and A. Nowatzyk. Missing the memory wall: the case for processor/memory integration. In *ISCA '96: Proc. of the 23rd annual int. symp. on Computer architecture*, pages 90–101, New York, NY, USA, 1996. ACM Press.

[29] Premkishore Shivakumar and Norman P. Jouppi. Cacti 3.0: An integrated cache timing, power and area model. Western Research Lab, Research Report 2001/2, 2001.

[30] L.V. Yerosheva, S.K. Kuntz, J.B. Brockman, and P.M. Kogge. A microserver view of HTMT. Parallel and Distributed Processing Symposium, Proceedings 15th International, pages:10, April 2001.

[31] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. R. Stan. HotLeakage: An architectural, temperature-aware model of subthreshold and gate leakage. University of Virginia Dept. of Computer Science, Tech. Report CS-2003-05, March 2003.