# Combining Data Remapping and Voltage/Frequency Scaling of Second Level Memory for Energy Reduction in Embedded Systems

Sudarshan K. Srinivasan, Jun Cheol Park and Vincent J. Mooney III
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA-30332
{darshan, jcpark, mooney}@ece.gatech.edu

## Abstract

*In this paper we show that the energy reductions obtained from using two techniques, data remapping and voltage/frequency scaling of off-chip bus and memory, combine to provide interesting trade offs between energy, execution time and power. Both methods aim to reduce the energy consumed by the memory subsystem. Data remapping is a fully automatic compile time technique applicable to pointer-intensive dynamic applications. Voltage/frequency scaling of off-chip memory is a technique applied at the hardware level. When combined together, energy reductions can be as high as 46% .The improvements are verified in the context of two OLDEN pointer-centric benchmarks, namely Perimeter and Health.*

## 1 Introduction

In embedded systems, memory is a significant power/energy sink, often consuming as much as half of the total power/energy [2]. In this paper we focus on simultaneously applying a hardware technique and a compile time technique in order to obtain significant energy savings. Our target processor is an ARM-like processor. The two methods we apply are data remapping (compile time technique) and voltage/frequency scaling of the off-chip bus and memory.

An embedded system usually consists at least of a processor (including L1 cache), off-chip memory and off-chip bus. The off-chip components are highly capacitive. This causes them to consume close to half of the digital system power (where digital system power is the power consumed by the processor plus memory). Slowing down the off-chip memory by scaling the voltage and frequency[19, 20, 15] can be used to reduce the energy consumed by the off-chip memory. But slowing down the off-chip memory will also reduce performance.

Data remapping[11, 10] is a compile time technique. It is used to remapp the application's data layout so that data elements that are accessed contemporaneously are placed together in memory. Remapping improves spatial locality and thus reduces cache misses. Cache misses are expensive in terms of performance. Remapping leads to a reduction in the execution time and energy.

The main drawback of the voltage/frequency scaling technique is the reduction in performance. Combining the two techniques can increase the overall reduction in energy. Furthermore, where execution time allocated is fixed, the combination of faster execution due to data remapping can offset the slower execution time due to reduced clock frequency for off-chip memory resulting in the same original execution time at dramatically reduced power (and energy). Section 2 described related work. Section 3 describes the experimental infrastructure used for estimating the power and energy of the system for the before and after cases. Section 4 gives an overview of the data remapping algorithm. Section 5 describes the methodology used for voltage and frequency scaling, and Section 6 gives our design space exploration. Section 7 describe the results obtained after applying the two methods in terms of energy savings, and Section 8 concludes the paper.

## 2 Related Work

A framework similar to our infrastructure is Simplescalar ARM. It is a framework for power and performance analysis. Another is SimplePower[14] which implements a subset of the instructions supported by Simplescalar.

Unlike Simplescalar, our model (MARS, introduced later) is at the RTL (Register Transfer Level) level and thus is more accurate.

With respect to hardware techniques there is a lot of work going on to gate supply voltage to cache memories[6, 12], dynamically adjust the frequency or shutdown unused modules[4].

Related work in data reorganization[7] propose automated field re-ordering that assigns temporally related fields to adjacent memory locations. But they offer only partial solutions as they do not consider fields between different instances of a record.

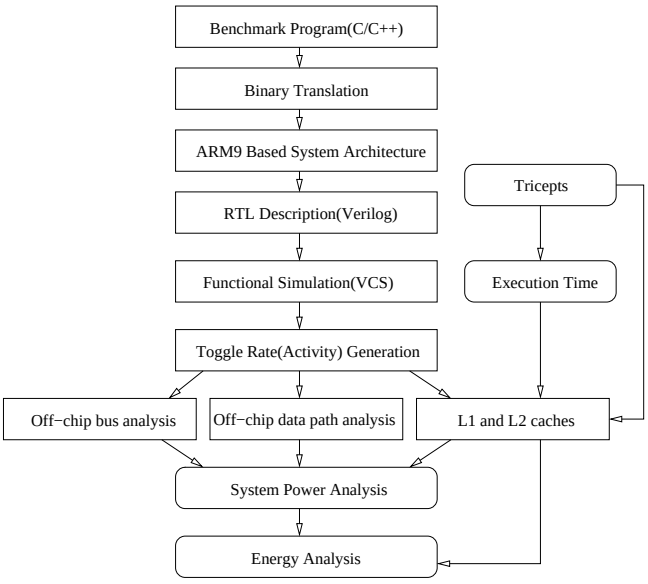


Figure 1: Experimental Infrastructure

# 3 Experimental Setup

In this section we describe the experimental setup used to simulate and evaluate the combined techniques of data remapping and voltage/frequency scaling.

The core of the simulation environment is MARS (Michigan Arm Simulator, obtained from University of Michigan)[16], capable of running ARM instructions. The power of the core processor can be estimated using Synopsys Power Compiler. The switching activity of the various nets is collected via simulation. The MARS model was synthesized using TSMC .25u library[18]. Using Synopsys Power Compiler[17], power models of the synthesized MARS model were created. The processor power was estimated using the power models and the switching activity of the nets at 2.75 volts.

The remapping benchmarks were implemented using TRIMARAN, a compiler framework (which includes the TRICEPS[8] ARM code generator and smart memory and cache hierarchy simulator (SMACHS)[21]. The execution statistics from TRIMARAN are used to estimate the power of the memory subsystem.

The model used has an L1 on-chip cache and L2 off-chip cache. To estimate the energy of the primary and secondary cache we assume an SRAM model. The Kamble and Ghose approach is used for energy estimation[9]. Execution statistics such as cache requests, read hits and misses, write hits and misses, execution cycles for both L1 and L2 cache are required. These execution statistics were obtained from simulations using SMACHS. Also, information about the cache configuration such as cache size, block size and tag size are required. One of the main disadvantages of the Kamble and Ghose method is that it does not model the I/O pads. Another disadvantage is that the model only accounts for dynamic power dissipation. This approximation (not including static/leakage power) is valid with respect to 0.25u technology but may not be valid for smaller(e.g., 0.09u) technologies.

The off-chip bus power is estimated using Spectre simulation. The driver component is modeled by a series of inverters. The model is designed using 0.25u TSMC library. The bus line capacitance values are obtained from actual measurements of a PCB with an Intel StrongARM1110 processor.

The total system power is estimated using the following approach. The energy for the L1 and L2 cache is obtained directly using the Kamble and Ghose model[15, 9]. The processor power is obtained from the Synopsys Power Compiler. Processor power is multiplied with the execution time to obtain the processor energy. The bus power obtained from the Spectre simulations is also multiplied with the execution time to obtain the energy of the off-chip bus. The energy from the bus, processor, L1 and L2 cache are summed up to get the total system power. The resulting measurements for our examples are shown in Section 7.

# 4 Data Remapping

Data Remapping is a compile time technique. It is an efficient remapping of an application's data layout in memory such that data elements that are accessed contemporaneously are placed together in memory. If a reference stream does not exhibit address adjacency, valuable resources are wasted as data is unnecessarily fetched and cached. The remapping technique remaps elements into new sets such that data items that are more likely to be used together are grouped together into the same cache block.

The applications to which data remapping can be usefully applied are record data type-heavy and pointer-heavy applications. Consider an example where in a file of records, a particular field of all records has to be searched or modified. The original mapping of the data in the memory will be such that fields belonging to a particular record will be placed together. If a cache line is fetched then all the data other than that particular field is wasted. Also the search for the next field will lead to a cache miss. Instead, if all fields were placed together in the above example then cache misses will be reduced. Also, energy is not wasted in fetching data that is not useful. The remapping algorithm is a combination of field reordering and customized placement to exhibit better spatial locality.

The remapping optimization consists of three phases – gathering phase, remapping of global data objects and remapping of dynamic data objects. In the gathering phase, an analysis of application memory access patterns along program hotspots[1] is performed. The remapping strategy cannot be arbitrarily applied to all data objects in the program. It is applied

based on the analysis obtained from the gathering phase. In the second phase global data objects are remapped. Once the candidate records have been identified, global program variables are filtered to isolate the arrays of records which are remapped. The third phase remaps dynamic data objects (i.e., pointer variables). The third phase is crucial as applications increasingly rely on dynamically allocated objects[3, 5].

# 5 Voltage and Frequency Scaling of Off-Chip Memory and Buses

The power consumed is proportional to the square of the voltage. Thus, reducing the voltage will lead to a quadratic reduction in power. But when the voltage of a component is lowered it leads to increase in delay which affects performance. The off-chip memory and buses are highly capacitive and thus consume close to half of the system power. To reduce the overall system power significantly we scale the voltage of the off-chip memory and buses. In our system the off-chip memory is an L2 cache.

Figure 2 shows the slowing down of L2 memory. The original system runs at 100MHz with the processor at 2.75 volts and off-chip components (including bus and memory) at 3.3 volts. The voltage of the off-chip bus and memory was scaled from 3.3 volts to 2 volts. This causes the delay of the off-chip memory and bus to double. To take into consideration the increase in delay, the frequency of the off-chip components was scaled from 100MHz to 50MHz.

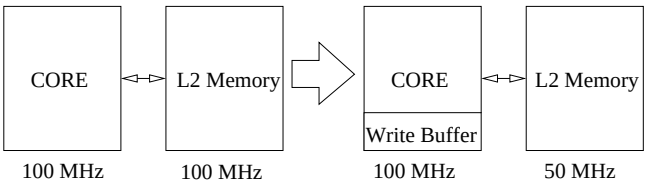


Figure 2: Slowing down L2 Memory

Frequency scaling is achieved by simulating the off-chip components at 50MHz instead of the original 100MHz. The D-Cache and I-Cache controllers were modified such that they fetch data from the memory at 50MHz instead of the previous 100MHz. This is done by doubling the latency of the memory (in our case the L2 cache) from 7 to 14 cycles. The voltage at which power is estimated is reduced from 3.3 volts to 2 volts to simulate voltage scaling in case of the off-chip bus and memory.

# 6 Design Space Exploration

The original system consists of the processor and off-chip components running at 100MHz. We simulate the system using two benchmarks health and perimeter before remapping. We call the original system the before case. The after case is where the processor is simulated at 100MHz and the off-chip components are simulated at 50MHz. The health and perimeter benchmarks are remapped and simulated with 50MHz L2 memory so that effect of combining both the techniques can be determined. Switching activity files are collected from the simulations using the MARS model and are used to determine the processor and bus power in both cases. The execution statistics from the Trimaran ARM simulator is used to determine the power for the L1 cache and off-chip L2 cache.

The data remapping allows the program to achieve the same overall execution time with half the cache resources. Since the cache is expensive in terms of both power and cost, halving the cache size would lead to roughly half the cost and power requirements. Results have been obtained by using half the L1 and L2 cache size.

The power calculations do not include the overhead of multiple supply voltages as we assume that multiple supply voltages are already present in the board. Also it is assumed that voltage scaling (i.e., changing the frequency of off-chip components from 3.3 volts to 2 volts) is done statically.

# 7 Results

The energy savings from combining the two techniques has been shown for two Olden benchmarks[13], namely perimeter and health. The benchmarks selected are such that they are suitable for remapping. The perimeter allocates quad trees at the program startup and do not modify them. The primary data structure used in health is a link list to which elements are added and removed.

Table 1 shows the results for the health benchmark. The L1 is a 32KB cache with 16 bytes line size. The L2 is a 1MB cache with 32 bytes line size. We find that for the health benchmark there is a large reduction in the execution cycles, but for perimeter the reduction in execution cycles is not as much (see Table 2. Data remapping will cause an increase in performance but much of this performance gain is lost due to slowing down the off-chip memory. This is clearly seen in the case of the Health benchmark. Also we find that for both benchmarks there is a large decrease in the energy of the L2 cache. Even though the processor power is almost constant, the decrease in processor energy is due to gains in performance due to remapping. We are able to achieve a maximum of 46% energy gains in the Health benchmark. From our experiments, we observed that there is no simple linear relationship among data remapping, frequency/voltage scaling of second level memory, energy reduction and energy delay reduction.

The remapping technique allows a program to run with the same execution time but with far less the memory. To explore the design space, we also considered reducing the L1 and L2

Table 1: Energy Delay with Frequency/Voltage Scaling of Memory (FVM) and Data Remapping(DR) for Health Benchmark

| | Before DR, FVM | After DR | After FVM | After DR+FVM | After DR+FVM 1/2size L1 | After DR+FVM 1/2size L2 | After DR+FVM 1/2size L1,L2 |
|---|---|---|---|---|---|---|---|
| Execution Cycles | 803645821 | 479612138 | 892552982 | 578046486 | 603275469 | 711151104 | 736311686 |
| Delay(Execution Time)(s) | 8.036 | 4.796 | 8.926 | 5.780 | 6.033 | 7.112 | 7.363 |
| Energy(J) | 17.076 | 9.274 | 14.316 | 9.274 | 9.468 | 11.158 | 10.134 |
| Energy*Delay | 137.231 | 44.479 | 127.778 | 53.608 | 57.118 | 79.350 | 74.618 |
| % Energy Reduction | 0.00 | 39.33 | 16.16 | 45.69 | 44.55 | 34.66 | 40.65 |
| % Energy*Delay Reduction | 0.00 | 67.59 | 6.89 | 60.94 | 58.38 | 42.18 | 45.63 |

Table 2: Energy Delay with Frequency/Voltage Scaling of Memory (FVM) and Data Remapping(DR) for Perimeter Benchmark

| | Before DR, FVM | After DR | After FVM | After DR+FVM | After DR+FVM 1/2size L1 | After DR+FVM 1/2size L2 | After DR+FVM 1/2size L1,L2 |
|---|---|---|---|---|---|---|---|
| Execution Cycles | 1065497740 | 967549770 | 1073983968 | 999065267 | 999305168 | 999095525 | 999339410 |
| Delay(Execution Time)(s) | 10.655 | 9.675 | 10.740 | 9.991 | 9.993 | 9.991 | 9.993 |
| Energy(J) | 23.361 | 21.648 | 17.860 | 16.897 | 16.414 | 14.221 | 13.828 |
| Energy*Delay | 248.911 | 209.455 | 191.814 | 168.812 | 164.026 | 142.081 | 138.189 |
| % Energy Reduction | 0.00 | 7.33 | 23.55 | 27.67 | 29.74 | 39.13 | 40.81 |
| % Energy*Delay Reduction | 0.00 | 15.85 | 22.94 | 32.18 | 34.10 | 42.92 | 44.48 |

Table 3: Energy results after remapping and Voltage Scaling(L1=32KB, L2=1MB) for Health Benchmark

| Execution Cycles | Processor Energy(J) | Off-chip Bus Energy(J) | L1 Cache Energy(J) | L2 Cache Energy(J) | Total Energy(J) | % reductions Total Energy |
|---|---|---|---|---|---|---|
| 578046486 | 6.415 | 0.433 | 0.3155 | 2.104 | 9.274 | 45.69 |

cache sizes to half their original sizes. The last three columns in Tables 1 and 2 show the energy results after halving L1 cache, L2 cache and both L1 and L2 cache respectively. We find that as expected the energy requirements of the cache also reduced by half. In case of the Perimeter benchmark the execution time remains the same and thus the energy saving in the memory subsystem is reflected in the overall energy gains. A maximum of 40.81% energy reduction is achieved in case of Perimeter benchmark when both caches are reduced to half their size. But in case of the Health benchmark, reduction in cache size leads to increase in the execution time. Even though the energy requirement of the memory subsystem is reduced, this is not reflected in the overall energy gains due to the increase in execution cycles. Thus, for the Health benchmark, the maximum energy reduction of 45.69% is found with both caches at their original sizes (L1=32KB, L2=1MB).

# 8 Conclusion

There are many techniques at both the hardware and compiler level aimed at saving energy and power. In this work we have demonstrated a combination of two techniques, one at the hardware level and one at the compiler level. The main drawback of hardware techniques is that they tradeoff power with performance. In our work by combining the two techniques, we are able to obtain energy gains without leading to a performance loss. For future work, we are looking at additional architecture level techniques aimed at the memory subsytem (specifically at the cache) and processor where compiler and hardware techniques interact to reduce energy.

# 9 Acknowledgements

This research was funded bt DARPA under contract number F30602-00-2-0564. We acnowledge help received from Rodric Rabbah in running the Trimaran simulations of the Health and Perimeter benchmarks. We also acknowledge donations received from Cadence, Hewlett-Packard, LEDA Systems, Mentor Graphics, Sun and Synopsys.

# References

[1] T. Ball and J. Larus, ”Efficient path profiling,” *Proceedings of the 29th Annual International Symposium on Microarchitecture*, December 1996.

[2] P. Panda, N. Dutt and A. Nicolau, “Memory Issues In Embedded Systems-On-Chip, Optimizations and Exploration, ” Kluwer Academic Publishers, 1999.

[3] B. Calder, C. Krintz, S. John and T. Austin, ” Cache-conscious data placement,” *Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems,* pp. 139-149, October 1998. Symposium on Theory of Computing, 1978.

[4] A. Acquaviva, L. Benini and B. Ricco, “An Adaptive Algorithm for Low-Power Streaming Multimedia Processing,” *Proceedings of Design Automation and Test in Europe*, pp. 273-279, March 2001.

[5] T. Chilimbi, M. Hill and J. Larus, ”Cache-conscious structure layout,” *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementatio,* pp. 1-12, May 1999.

[6] L. Benini, A. Macii, E. Macii and M. Poncino, "Synthesis of Application-Specific Memories for Power Optimization in Embedded Systems," *Proceedings of 38th Design Automation Conference*, pp. 300-303, June 2000.

[7] T. Chilimbi, B. Davidson and J. Larus, “Cache-conscious structure definition,” *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation,* pp. 13-24, May 1999.

[8] L. Chakrapani, K. Palem and W. Wong, “Enhancing the TRIMARAN compiler infrastructure for StrongARM code generation,” Technical Report CREST-TR-01-001, Georgia Institute of Technology, May 2001

[9] M. Kamble and K. Ghose ”Analytical energy dissipation models for low power caches,” *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 143-148, Aug. 1997.

[10] K. Palem, R. Rabbah, P. Korkmaz, V. Mooney and K. Puttaswamy, "Design Space Optimization of Embedded Memory Systems via Data Remapping," *Proceedings of the Languages, Compilers, and Tools for Embedded Systems (LCTES’02)*, pp. 28-37, June 2002.

[11] K. Palem and R. Rabbah. “Data remapping for design space optimization of embedded cache systems.” Technical Report GIT-CC-02-10, Georgia Institute of Technology, March 2002.

[12] T. Ishihara and K. Asada, "A System Level Memory Power Optimization Technique using Multiple Supply and Threshold Voltages," *Proceedings of 38th Design Automation Conference*, pp. 456-461, June 2001.

[13] OLDEN benchmark suite. *http://www.cs.princeton.edu/mcc/olden.html*.

[14] W. Ye, N. Vijaykrishnan, M. Kandemir and M. J. Irwin, "The Design and Use of Simplepower: A Cycle-Accurate Energy Estimation Tool," *Proceedings of 38th Design Automation Conference*, pp. 340-345, June 2000.

[15] P. Korkmaz, K. Puttaswamy and V. Mooney, “Energy modeling of a Processor core using Synopsys and of the Memory Hierarchy using the Kamble and Ghose Model,” Technical Report, CREST-TR-02-002, Georgia Institute of Technology, February 2002.

[16] The SimpleScalar-Arm Power Modeling Project, *http://www.eecs.umich.edu/~jringenb/power/*

[17] Synopsys, Inc., *http://www.synopsys.com*

[18] LEDA Systems, Inc., *http://www.ledasys.com*

[19] K. Puttaswamy, K. Choi, J. C. Park, V. J. Mooney III, A. Chatterjee and P. Ellervee, “System Level Power-Performance Trade-Offs in Embedded Systems Using Voltage and Frequency Scaling of Off-Chip Buses and Memory,” *Proceedings of International Symposium on System Synthesis*, to appear, October, 2002, Kyoto, Japan.

[20] K. Puttaswamy, L. N. Chakrapani, K. W. Choi, Y. S. Dhillon, U. Diril, P. Korkmaz, K. K. Lee, J. C. Park, A. Chatterjee, P. Ellervee, V. Mooney, K. Palem and W. F. Wong, "Power-Performance Trade-Offs in second level memory used by an ARM-Like RISC Architecture," in the book *Power Aware Computing*, Rami Melhem and Robert Graybill, Eds., Kluwer Academic/Plenum Publishers, 2002.

[21] Trimaran, *http://www.trimaran.org*