Exploring Non-Volatile Main Memory Architectures for Handheld Devices

Sneha Ved Manu Awasthi
Indian Institute of Technology Gandhinagar
{sneha.ved,manua}@iitgn.ac.in

***Abstract*—As additional functionality is being added to contemporary handheld devices, the SoCs inside these devices are becoming increasingly complex. Similarly, the applications executing on these handhelds are beginning to exhibit an ever increasing memory footprint. To support these trends, main memory capacity of these SoCs has been increasing over time. Due to these developments, memory system's contribution to the overall system power has increased dramatically.**

Non-volatile memories have been used in server architectures to increase capacity as well as keep memory system's power consumption in check. However, in the handheld domain, where user experience and battery life are of paramount importance, the applicability of such technologies has not been widely studied. In this paper, we propose and evaluate a number of hybrid memory architectures using mobile DRAM and PCM. We show that intelligent memory architectures, cognizant of workload's memory access patterns can provide significant energy savings without compromising on user experience. Using proposed approach, we can devise architectures that exhibit significant energy savings with only a 2.8% performance loss.

I. Introduction

The smartphone and the handheld device market has seen tremendous improvements in the last decade. Since these devices are mostly used for information consumption [1], entertainment and communication, with an emphasis on mobility, the primary design metrics still revolve around response time and battery life [2]. In a sense, both these metrics are contradictory to each other - high performance usually comes at the cost of higher power dissipation (reduced battery life). For mobile devices, longer battery life has often taken precedence over performance. As a result, LPDDR variants are the popular choice for memory technology in mobile SoCs despite having performance capacity disadvantages over the more common DDRx variants [3]. However, LPDDR variants provide a bigger energy advantage, especially in standby power consumption [4].

As mobile applications become more complex, the memory capacity per SoC had to increase to keep up with their demands. High end smartphones are moving to 6-8 GB main memory capacity [5], and the number is projected to increase over time [6]. Furthermore, in addition to capacity, some applications, like games, have much higher bandwidth requirements, leading to SoCs incorporating two or more memory channels [7]. Clearly, the memory requirements of mobile SoCs have reached the inflection point where there is an immediate need to start looking at memory architectures that lead to increased memory capacity, reduction in overall power consumption, without compromising on performance.

Non volatile memories (NVMs) have been used in the server domain for the aforementioned goals [8]. NVM has the advantage of 2x-4x increased storage density and lower energy consumption than most DDR/LPDDR variants. However, it does suffer from increased access latencies, variable write times and limited write cycles. These properties, coupled with the fact that SoC design constraints, operating systems and workload characteristics for servers are vastly different as compared to handheld devices, hybrid memory architectures for servers cannot be ported over blindly for the handheld domain.

In this paper, we make a number of significant contributions. First, we carry out a thorough, memory-centric characterization of the Moby [9] benchmark suite to analyze the memory access patterns of mobile benchmarks. We identify the address regions of high and low memory activities and propose novel architectures that intelligently utilize this information to create hybrid memory architectures. Finally, we propose and evaluate new DRAM address mapping policies that further limit the performance degradation caused by the hybrid memory architectures created in the previous step. The combination of these three techniques results in memory architectures with lower energy consumption compared to baseline, LPDDRx variants and negligible effect on user experience.

II. Background and Motivation

A large number of specialized units including GPUs, modems and DRAM have been integrated to multiple CPU cores to form a SoC [10]. In order to keep power consumption in check, mobile SoCs are typically provisioned with low-power double data rate (LPDDR) DRAM. LPDDR eliminates energy inefficient circuits from commodity DDRx DRAM [11], making it an ideal fit for handheld devices.

The amount of DRAM capacity in mobile SoCs has been consistently increasing over time. The earliest versions of smartphones (circa 2007) were provisioned with 256 MB of main memory [12]. Over the past decade, the capacity has consistently been increasing. Contemporary, high-end smartphones are provisioned with 1 GB to 4 GB, and 8 GB capacity is projected to be the new norm in 2017 [6]. As a testament to increasing working set sizes for smartphone applications, [13] reported that almost 15% of the applications running on smartphones has to be relaunched because of a lack of available DRAM capacity. Increased capacity has also led to increase in the power consumption of the DRAM subsystem. In order to verify these claims, we analyzed the effects of increasing DRAM part capacity on the power consumption of the part. Micron's power calculator for LPDDR2 devices [14] was used for this analysis. Doubling DRAM capacity from 1 Gb to 2 Gb under same usage conditions leads to a power consumption increase of 12.9% for highest row-buffer hit

This work was supported in part by SERB grant ECR/2017/000887

978-3-9819263-0-9/DATE18/©2018 EDAA

rate (85%). The increase is higher in cases of low locality – for 15% hit rates, the corresponding increase in power is 26.7%. This increase in the power consumption of the DRAM component also leads to a substantial increase in the memory subsystem's contribution to the overall energy consumption. [15] projects that with increasing capacity, DRAM could constitute upto 20% of the overall system power consumption. This suggests that in the near future, blindly adding more capacity using higher capacity parts will become infeasible from an energy perspective.

III. Related Work

A number of techniques have been proposed to use NVMs in the context of handheld devices. The use of NVM has broadly been classified into three major classes :

a) NVM + DRAM - As Hybrid memory: Zhao et al. [16] propose a DRAM-PCM hybrid memory hierarchy for handheld devices. They make the observation that system software techniques are required to partition the data effectively between DRAM and NVM. They propose an energy cost model for each data for handling data placement.

b) NVM + DRAM - As swap space: A number of recent research has proposed the use of NVMs as swap spaces. Kim et al. [17] present CAUSE, a paging mechanism that manages data between DRAM and NVM swap at an OS page granularity. Kawata et al. [18] turn the tables around on most proposals and advocate using NVM as the main memory, and DRAM as the "swap" space.

Zhong et al. [19] propose DR-Swap, an in-memory paging design to reduce main memory energy consumption in smartphones. They first propose IMP (in-memory paging) which treats NVM as a swap space, which lets the kernel swap pages between the DRAM and NVM regions without having to go through the storage stack. The Direct Read (DR) optimization to IMP lets the applications read data directly from the NVM swap area, without having to bring the pages into DRAM. In a similar approach, Zhong et al. [20] propose `NVM-Swap`, a swap mechanism that implements 1) copy-on-write to reduce the movement of data between the two memory portions, and 2)`Heap-Wear` which addresses the wear leveling issues of NVM, while being cognizant of the space requirements.

c) NVM + FLASH - As low latency Flash: [21] proposes *vFlash*, a cross-layer scheme to manage hybrid storage architecture comprising of NVM and Flash by adding virtualized Flash (vFlash) software layer to the Linux kernel.

To the best of our knowledge, the work done Duan el al. [22] to carry out coarse grained analysis of a number of popular mobile apps to characterize the available performance wiggle room to estimate the maximum performance overhead that can be tolerated without effecting user experience, comes closest to the work done in this paper. This paper differs from prior work in several ways. First, we carry out a detailed memory activity characterization of various smartphone applications to gauge memory regions with little or no activity, that can be retrofitted with low performance, low energy memory technologies like PCM. This is explained in greater detail in the next section.

IV. Workload Analysis

To assess suitability of heterogeneous memory architectures for handheld devices, we analyzed memory addresses and regions accessed by various workloads in the Moby suite [9]. First, to quantify the effects of memory hierarchy on the user experience, we ran experiments where the memory configuration was changed from LPDDR3 to PCM.We observe that in many cases, while the user experience is adversely effected, there were significant savings in memory energy consumption.

Substituting LPDDR3 hierarchy with a PCM one has a drastic impact on both performance and energy consumption. The move results in a maximum performance degradation of 50% (Kingsoftoffice), and an average slowdown of 19%. Although memory energy consumption is reduced by 75% across the suite, the tradeoff however, is unacceptable. In order to reduce the decrease in performance, we wanted to identify the regions of memory that aren't as frequently accessed and could better tolerate the increased memory access latencies incurred due to inclusion of an NVM in the hierarchy. We found that memory accesses, for almost all programs in the Moby suite, are clustered around a few, select address ranges as shown in Figure 1. The Y-axis represents the physical addresses, while the X-axis depicts the time during program execution a particular address was accessed. The more an address is accessed, the darker it appears on the map. We observe that over 75% of all distinct addresses accessed were concentrated within 25% of the total address space. These experiments were done with an LPDDR3 memory system with two channels and default DRAM address mapping scheme. These observations are in line with how Android allocates memory [23]. The observations from these experiments form the basis of the hybrid memory architecture devised in this paper : *if frequently accessed data can be concentrated to the fastest regions of a hybrid memory architecture, memory system's energy consumption can be reduced significantly, without any loss in performance*.

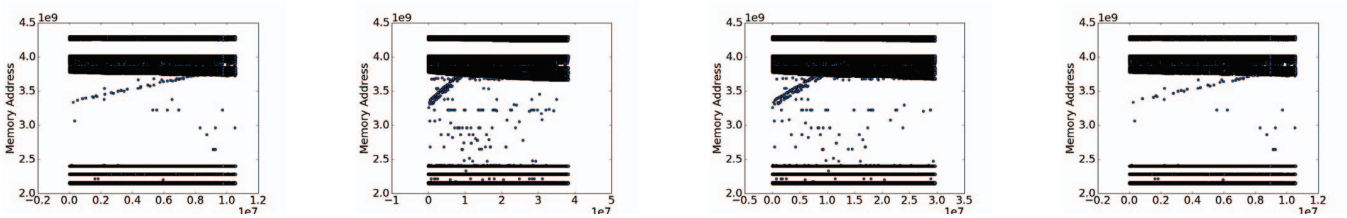
V. Proposal

Based on the observations in Section IV, we devise hybrid memory architectures using PCM and LPDDR3. LPDDR3 is an obvious choice for mobile DRAM. We start with a baseline, homogeneous memory architecture where the entire physical address space comprises of either LPDDR3 or PCM connected via a memory controller and two channels.

To devise heterogeneous architectures from this baseline, we assume that the entire memory address space can be divided into equal sized sub-address spaces (SAS). Each SAS can be mapped to a particular memory technology, without making any changes to the underlying system software. We also assume that each SAS can be controlled via a combination of independent 1) memory controller and, 2) memory channel(s). Most of these assumptions have been shown to be implementable in prior, independent studies [24]. As a result, a memory address space can potentially be divided into N SAS, with N memory controllers and N channels.

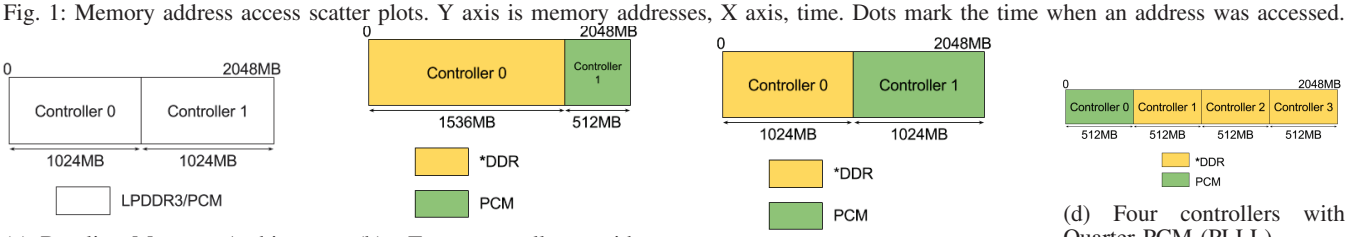
In order to keep the complexity of the SoC in check, contiguous regions of the SAS that map to the same memory technology to *fuse* their controllers and share their memory channels. This optimization can potentially reduce the amount of available parallelism and hence increase the queuing times at the controller.

Figure 2b presents a case where a quarter of the total address space (SAS 1) is mapped to LPDDR3, and the rest is mapped to PCM (SAS 0). This design only calls for two controllers,



(a) FrozenBubble (b) SinaWeibo (c) Netease (d) Baidu

Fig. 1: Memory address access scatter plots. Y axis is memory addresses, X axis, time. Dots mark the time when an address was accessed.



(a) Baseline Memory Architecture (LL/PP)

(b) Two controllers with Quarter PCM and fused controllers for LPDDR

(c) Two controllers with Half PCM (LP)

(d) Four controllers with Quarter PCM (PLLL)

Fig. 2: Hybrid Memory Architectures for Multiple Controllers

and two channels. The variant presented in Figure 2d is a memory architecture where the top quarter SAS is PCM, while the bottom three-fourths is equally divided into quarter chunks, with each chunk connected using its own controller and an independent channel. To simplify heterogeneous designs, we disallow interleaving of memory requests across channels, even in the baseline. This has additional benefits in real life implementations as reads and writes for a particular cacheline are not split across multiple memory technologies, which might result in access times being dominated by the access times of the slowest memory technology in play. Due to limitations of Gem5 with the ARM ISA, we experimented with 2 GB of available memory capacity. In the rest of the paper, a two controller homogeneous system with only LPDDR3 memory will be referred to as **LL**, and a heterogeneous system, with LPDDR3 in the top half and PCM in the second half will be referred to as **LP**. Similarly, for the case with four controllers, in an hybrid architecture with LPDDR3 and PCM, with the third quarter assigned to PCM and the rest to LPDDR3, will be referred to as **LLPL**.

VI. Methodology and Results

We evaluated the proposed techniques using MofySim [25]. Table I lists the simulation parameters used in this study. The timing parameters for PCM were derived from [26], while the energy calculations were based on NVMain [27]. All the proposals were evaluated using Moby [9] benchmark suite.

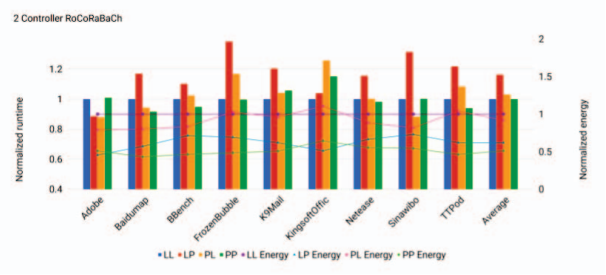
For the case with two controllers, we evaluate all possible combinations possible with LPDDR3 and PCM. Figure 3a presents the execution time and energy consumed (normalized to **LL**, lower is better for both) for each benchmark. We observe that a naive shift from LL to PP results in unacceptable performance losses - 20% on average across the entire suite at a significant energy savings of 49%. Intelligent choice of SAS for LPDDR and PCM technologies is able to contain the performance loss for many benchmarks while still resulting in significant energy savings. For example, Sinaweibo and Netease have significant number of memory accesses in the middle chunk of two heavily accessed SAS. The **LP** configuration for these workloads results in mapping almost all of these to the slower, PCM technology. As a result, for these configurations, the **PL** variants perform better than **LP** ones.

Furthermore, some of the slowdown exhibited in moving from an **LP** to **PL** variants could be due to the use of the (default) DRAM address mapping scheme that distributes consecutive addresses across multiple SAS. Figure 3b presents the results of 2 controller configurations with ChRoRaBaCo[1] mapping scheme, which tries to restrict the accesses to consecutive addresses within a SAS. This helps us further reduce the performance gap between LL and PL variants to only 1%. The energy savings for the PL variant with the new address mapping scheme are still very significant at 23%.

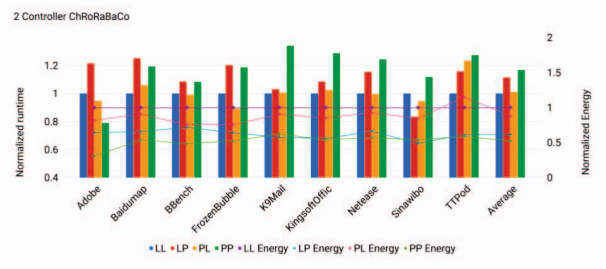
Figure 4 presents the results of another set of experiments where the address space consists of 4 SAS (LLLL, LLLP, LLPL, LPLL, PLLL and PPPP). The trends in this case remain similar. Ability to create smaller SAS results in multiple benefits. First, if chosen carefully, the area of memory with little or no memory activity when replaced with PCM, results in almost negligible performance loss and some energy benefits. As a testament to this, the best performing architectures across the entire benchmark suite (with the default address mapping scheme, Figure 3a) are LLPL and LPLL. The former exhibits an average slowdown of 7.3% over LLLL configuration, while the latter bridges the gap exhibiting an average slowdown of 0.9%. However, without address mapping optimizations, the energy savings are small as well - 2% for LPLL.

As shown in Figure 4b, the use of new DRAM address mapping mechanism bridges this gap. The LLLP configuration results in a 72% energy savings with only a 3% performance penalty. Similarly, other configurations provide a multitude of design choices to architects for various energy - performance tradeoff points. For example, the PPPP configuration will result in 95% energy savings with 13% performance loss,

[1]Channel:Row:Rank:Bank:Column. We are not able to present the results of other mapping schemes due to space restrictions.

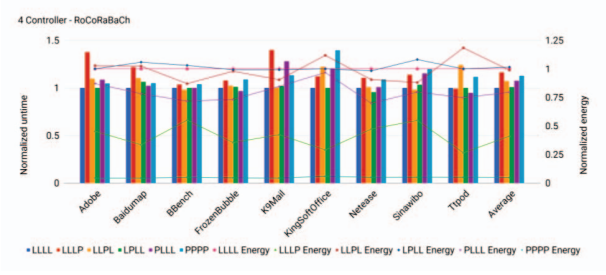


(a) Default Address Mapping

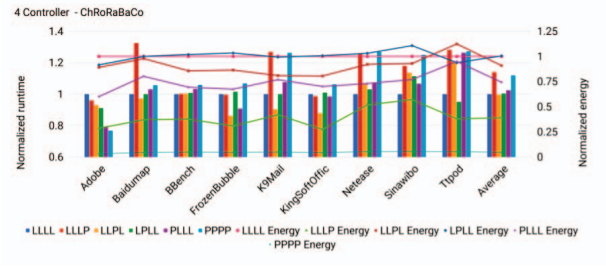


(b) New/Proposed address mapping (ChRoRaBaCo)

Fig. 3: Results for 2 controller experiments. Primary Y axis - runtime relative to LL, secondary Y axis - relative energy consumption.



(a) Default Address Mapping, 4 Ctrl



(b) New/Proposed address mapping (ChRoRaBaCo), 4 Ctrl

Fig. 4: 4 controller experiments. Primary Y axis - runtime relative to LLLL, secondary Y axis relative energy consumption.

while the LLPL configuration helps provide 10% energy savings at 0.5% performance loss.

ISA,CMP size,Core Freq.	ARM, 4-core, 1 GHz
L1 cache	16KB/4-way, private, 6-cycle
L2 cache	2MB/8-way, private, 44-cycle
Baseline DRAM Configuration	2x 32-bit Channels, 1/Memory Controller 1 DIMM/Channel 1 Ranks/DIMM,1 device/Rank, 8 Banks/Rank
DRAM Frequency/Addr Map	800MHz /RoCoRaBaCh
PCM Timings	tRCD:44ns,tCL:7.5ns,tRAS:25.5ns,tWR:300ns
LPDDR3 Timings	tRCD:18ns,tCL:15ns,tRAS:42ns,tWR:15ns
PCM Energy (per word)	Read:0.08nJ, Write:1.68nJ
LPDDR3 Energy (per word)	Read:3.40nJ, Write:1.02nJ, Refresh:38.55nJ
OS	Android 4.2.2

TABLE I: Simulator parameters, energy [27] and latency [26] values

VII. Conclusions and Future Work

In this paper, we present a first of it's kind study in the handheld domain to devise memory architectures with non volatile memories like PCM. These hybrid architectures contain performance losses while maximizing the energy gains. Unlike previous studies, which incorporate NVMs in the hierarchy as a swap devices, we propose architectures and DRAM address mapping mechanisms to make PCM a first class component of the memory hierarchy. Our approach provides the SoC architects a wide variety of options to choose from the performance vs. energy savings curve. Using a complete set of mobile benchmarks, we show that architectures that exhibit up to 61% energy savings in the memory subsystem with only a 2.8% performance loss can be devised.

References

[1] Q. Xu *et al.*, "Identifying diverse usage behaviors of smartphone apps," in *ACM SIGCOMM IMC*, 2011.
[2] S. Hosio *et al.*, "Monetary assessment of battery life on smartphones," in *CHI*, 2016.
[3] K. K. Yee, "Transitions: A Roadmap to Low-Power Memory," 2014.
[4] "Mobile Low-Power DDR SDRAM," 2011.
[5] "One Plus 5." [Online]. Available: https://oneplusstore.in/5
[6] N. Shah, "Smartphone Trends: More Memory, More Market Share," 2017.
[7] A. Frumusanu, "Early Exynos 8890 Impressions And Full Specifications," 2016.
[8] M. K. Qureshi *et al.*, "Scalable High Performance Main Memory System Using Phase-change Memory Technology," in *ISCA*, 2009.
[9] Y. Huang *et al.*, "Moby: A Mobile Benchmark Suite for Architectural Simulators," in *ISPASS*, 2014.
[10] L. Torres *et al.*, *An Introduction to Multi-Core System on Chip – Trends and Challenges*, 2011.
[11] K. T. Malladi *et al.*, "Towards Energy-proportional Datacenter Memory with Mobile DRAM," in *ISCA*, 2012.
[12] N. Patel, "iPhone 3G S processor specs: 600MHz CPU, 256MB of RAM," 2009.
[13] W. Song *et al.*, "Personalized Optimization for Android Smartphones," *ACM TECS*, vol. 13, no. 2s, pp. 60:1–60:25, Jan. 2014.
[14] Micron, "LPDDR2 System Power Calculator," 2016.
[15] A. Carroll *et al.*, "The Systems Hacker's Guide to the Galaxy Energy Usage in a Modern Smartphone," in *APSYS*, 2013.
[16] Z. Shao *et al.*, "Utilizing PCM for Energy Optimization in Embedded Systems," in *ISVLSI*, 2012.
[17] Y. Kim *et al.*, "CAUSE: Critical Application Usage-Aware Memory System Using Non-volatile Memory for Mobile Devices," in *ICCAD*, 2015.
[18] H. Kawata *et al.*, "Experimental design of high performance non volatile main memory swapping using DRAM," in *SNPD*, 2015.
[19] K. Zhong *et al.*, "Energy-Efficient In-Memory Paging for Smartphones," *IEEE TCAD*, vol. 35, no. 10, 2016.
[20] ——, "Building High-performance Smartphones via Non-volatile Memory: The Swap Approach," in *EMSOFT*, 2014.
[21] R. Chen *et al.*, "Unified non-volatile memory and nand flash memory architecture in smartphones," in *ASP-DAC*, 2015.
[22] R. Duan *et al.*, "Exploring memory energy optimizations in smartphones," in *IGCC*, 2011.
[23] C. Rinaldi, "Android vs IOS - How different is their RAM Management," 2017.
[24] G. Dhiman *et al.*, "PDRAM: A hybrid PRAM and DRAM main memory system," in *DAC*, 2009.
[25] M. Ju *et al.*, "MofySim: A mobile full-system simulation framework for energy consumption and performance analysis," in *ISPASS*, 2016.
[26] C. Xu *et al.*, "Overcoming the challenges of crossbar resistive memory architectures," in *HPCA*, 2015.
[27] M. Poremba *et al.*, "Nvmain 2.0: A user-friendly memory simulator to model (non-)volatile memory systems," *IEEE CAL*, vol. 14, no. 2, pp. 140–143, July 2015.