Energy-Aware Memory Allocation in Heterogeneous Non-Volatile Memory Systems

Hyung Gyu Lee and Naehyuck Chang*
School of Computer Science & Engineering Seoul National University, Korea
{hglee, naehyuck}@cselab.snu.ac.kr

ABSTRACT

Memory systems consume a significant portion of power in hand-held embedded systems. So far, low-power memory techniques have addressed the power consumption when the system is turned on. In this paper, we consider data retention energy during the power-off period. For this purpose, we first characterize the data retention energy and cycle-accurate active mode energy of the non-volatile memory systems. Next, we present energy-aware memory allocation for a given task set taking into account arrival rate, execution time, code size, user data size and the number of memory transactions by the use of trace-driven simulation. Experiments demonstrate that our optimal configuration can save up to 26% of the memory system energy compared with traditional allocation schemes.

Categories and Subject Descriptors

C.5.3 [**Computer System Implementation**]: Microcomputers—*Portable devices*; B.3.0 [**Hardware**]: Memory Structures—*General*

General Terms

Measurement, Design, Experimentation

Keywords

low-power memory, Non-volatile memory, Memory allocation

1. INTRODUCTION

As hand-held devices become equipped with high-performance large-capacity memory systems, battery capacity becomes one of their most significant limitations. So far, system-level low-power techniques for the memory system have focused on the reduction of power consumption while the system is powered on or is in data access mode. In reality, a non-volatile main memory system is one of the most distinct features of modern hand-held embedded systems. Thus memory systems consume significant amount of power when the system is in data retention mode. For example, low-power PDAs (Personal Data Assistants) are known to consume more than half of the energy in the battery for data retention. This paper addresses power reduction of memory systems, taking data retention energy into account, by the use of energy-aware memory allocation.

Typical embedded computing systems are equipped with three types of memory devices for boot-up storage, primary storage and secondary storage. The boot-up memory must be naturally non-volatile and random-accessible for the read operation. The primary storage must be high-performance both for read and write operations, and of large capacity. The secondary storage must be naturally non-volatile, with large capacity and of low cost.

Despite semiconductor progress, no single semiconductor non-volatile memory device at the mass-production stage satisfies all the above constraints. Thus, most hand-held systems are equipped with heterogeneous non-volatile memory devices combining Flash memories and battery-backed dynamic memories in self-refresh mode. A combination of different types of non-volatile memory devices works in a complementary manner.

As various types of non-volatile memory devices show different characteristics, we must allocate them carefully. But a traditional allocation scheme does not guarantee energy efficiency because it does not care about the characteristics of the memory systems and applications.

In this paper, we provide techniques for energy-saving memory allocation in the tasking environment as follows: Firstly, we explore the energy consumption of the three aforementioned non-volatile memory devices by cycle-accurate energy measurement and precise energy characterization. And then, we derive analytical energy models of various combinations of the non-volatile memory devices. Secondly, we perform trace-driven energy simulation so that we can propose energy-efficient memory allocation schemes in heterogenous non-volatile memory systems for given embedded applications and user profiles.

2. RELATED WORK

There is relatively little literature that deals with data retention energy. Palm Pilot opened a new era of PDA with their Hotsync technology. The active power consumption of a Palm Pilot Pro is reported by around 130mW to 150mW in the worst case, while the data retention power (in sleep mode) is 26mW [1]. Palm Pilots are equipped with asynchronous DRAM devices and retain data by battery back up. In case of a 1,200mAH battery, the manufacturer suggests up to 10 hour continuous usage is possible, and the power consumption in active mode justifies this data [1]. On the other hand, data sheets suggest 1mW data retention power for an asynchronous DRAM. The sleep mode power [1] does not seem to be the data retention power because we commonly experience a battery life of about a month if we do not turn on the device. Even though we use a Palm Pilot for weeks without battery charging, the actual run time is a few hours unless we play games for a long time; if so, we will run out of the battery shortly. This suggests that we usually use more than half of the battery energy for data retention.

Recently, WinCE PDAs have become more popular than the Palm Pilot series. They are equipped with a 32bit RISC processor running at hundreds of MHz. To provide at least 16MB (generally 64MB or higher) memory, there is no alternative to SDRAMs. Recent research reports the power consumption of four PDAs: the

*Corresponding author. The RIACT at Seoul National University provide research facilities for this study. This work was partly supported by the Brain Korea 21 Project.

ISLPED'03, August 25–27, 2003, Seoul, Korea.

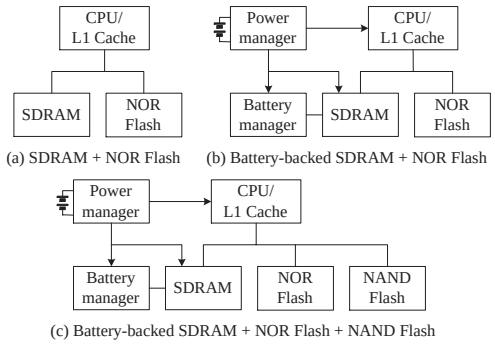

Figure 1: Typical configurations of the heterogeneous non-volatile memory systems.

Compaq Aero (MIPS R4000 @70MHz, 16MB RAM), the Compaq iPAQ (StrongArm @206MHz, 32MB RAM), the Casio Cassiopeia, (MIPS VR4121 @131MHz, 32MB RAM) and the HP Jornada (SH3 @133MHz, 32MB RAM) [2]. The power consumption varies depending on the application, but generally ranges from 1W to 3W, and thus results in a 3-hour to 5-hour continuous run. These devices are usually equipped with a Lithium-ion battery of around 1,500mAH capacity. Data sheets describe the data retention energy of two SDRAM devices as roughly 10mW, and thus data retention is guaranteed for around two weeks.

Theoretical research has introduced power consumption models and system-wide estimation of power requirement for a hand-held embedded system [3]. It includes power analysis of a DRAM memory system, but no analysis of data retention power. Finding an optimal configuration for a memory system also reduces power consumption [4]. Proper clustering can gather busy blocks into the same banks. This helps to maximize the chance of the others being powered down [5]. All of these techniques focus on power consumption when the system is turned on even though they manage the device power in the idle mode, (i.e., there is no consideration of data retention energy).

3. NON-VOLATILE MEMORY SYSTEMS

As mentioned in the previous section, the ultimate memory device does not really exist. A practical approach is to provide a heterogeneous non-volatile memory system.

Every system must have a naturally random-accessible non-volatile memory device for boot up. The NOR Flash memory is suitable for this purpose. Fig. 1 (a) shows a non-volatile memory systems consisting of NOR Flash memory and SDRAM. The NOR Flash memory is used to support only asynchronous interface, which does not provide enough transfer bandwidth for microprocessors operated at a high clock frequency. Thus, they are typically employed shadow memory configuration, which means copying the whole contents of the NOR Flash memory to the SDRAM before use. Recent Intel StrataFlash memory supports a 33MHz page-mode read operation and thus brought the NOR Flash memory into use for XIP (eXecution In-Place).

As commercial products cannot justify the cost of large capacity NOR Flash memory, non-volatile mass storage with battery back up of the SDRAM is provided, as shown in Fig.1 (b). The SDRAM is already used as primary storage; adding a battery back up simply achieves large amount of non-volatile storage. But we have to provide the data retention energy. The combination of NOR Flash memory and battery-backed SDRAM is a basic setup of modern PDAs.

NAND Flash memory devices can be used only for the secondary storage in cooperation with the primary storage. Many products are equipped with optional NAND Flash memory in the form of a removable card such as CompactFlash (Fig. 1(c)). Alternatively, the NAND Flash memory is sometimes permanently mounted on the circuit board. This configuration may not back up the SDRAM while saving and restoring the SDRAM contents to and from the NAND Flash memory.

Table 1: Summary of the energy and timing associated coefficients (@66MHz with 32Bit address and data buses).

Device	Energy		Timing	
	Coeff.	Value	Coeff.	Value
CPU	E_C	100 (mJ/sec)	NA	
SDRAM	E_{RSD}	70.2 (nJ/4words)	T_{RSD}	120 (ns/4words)
	E_{WSD}	51.6 (nJ/4words)	T_{WSD}	120 (ns/4words)
	E_{ISD}	45.1 (mJ/sec)	NA	
	E_{FSD}	99.2 (nJ/refresh)	NA	
	E_{TSD}	1.8 (mJ/sec)	NA	
NOR Flash	E_{RNR}	33.6 (nJ/4words)	T_{RNR}	210 (ns/4words)
	E_{WNR}	107 (μJ/16words)	T_{WNR}	464 (μs/16words)
NAND Flash	E_{RND}	1.18 (μJ/512bytes)	T_{RND}	35.8 (μs/512bytes)
	E_{WND}	9.51 (μJ/512bytes)	T_{WND}	226 (μs/512bytes)
$\{E\|T\}_{ab}$ a (state): R (Read), W (Write), I (Idle), F (reFresh), T (reTention) b (device): SD (SDRAM), NR (NOR Flash), ND (NAND Flash)				

4. ENERGY CHARACTERIZATION

4.1 System-wide energy consumption

The energy consumption of an embedded system for $[t_0, t)$ is denoted by

$$E_T(t) = \sum_{i=1}^{N} E_{Xi}(t) + E_M(t). \quad (1)$$

We divide the total system energy into energy for task execution and common mode energy, denoted by $E_{Xi}(t)$ and $E_M(t)$, respectively. Let us assume that $t_0 = 0$, for convenience and without loss of generality. The goal of this paper is to find a memory allocation having the minimum $E_T(t)$ for a given task set, $T = (\tau_1, ..., \tau_N)$.

We consider a task model with independent exponential arrival rates. A task τ_i has a 5-tuple, $(\lambda_i, \varepsilon_i, C_i, D_i, \rho_i)$, where λ_i is the arrival rate, ε_i is the average execution time, C_i is the code size, D_i is the data size, and ρ_i is 0 if data is read-only, otherwise ρ_i is 1.

At each invocation j, the task τ_i activates with $\varepsilon_i(j)$. The average execution time of task τ_i is given by

$$\varepsilon_i = \frac{1}{k} \sum_{j=1}^{k} \varepsilon_i(j). \quad (2)$$

The energy for task execution is given by

$$E_{Xi}(t) = \lambda_i t(E_C(\varepsilon_i + t_{Li} + t_{Si}) + (Em_i + El_i + Es_i)). \quad (3)$$

E_C is the energy consumption of the CPU for unit time; t_{Li} is the elapsed time for loading the code and the data if necessary; t_{Si} is the elapsed time for data saving if necessary; Em_i is the energy consumption of the memory system during task execution; El_i is the energy consumption of the memory system for loading the code and the data; Es_i is the energy consumption of the memory system during data saving. We derive the average active-mode energy values of E_C from the data sheets and/or measurement.

The common mode energy consumption, $E_M(t)$, is given by

$$E_M(t) = E_K(t) + E_R(t) + E_F(t), \quad (4)$$

where $E_K(t)$, $E_R(t)$ and $E_F(t)$ are leakage energy, data retention energy and refresh energy, respectively.

4.2 Device energy characteristics

This paper covers three different types of non-volatile memory devices; battery-backed SDRAM, NOR Flash memory, and NAND Flash memory. We derive the energy consumption of non-volatile devices for data access by the use of a state-machine-based energy characterization [6] so that we can model the active and idle energy correctly.

Table 1 summarizes the energy and timing coefficients of the above equations. Although the atomic access sizes are different,

Table 2: Memory allocation methods for the heterogenous non-volatile memory systems (C: code, D: data).

Allocation No.	1	2	3	4	5	6	7	8	9
SDRAM	C, D			C	D	C	D		
NOR		C, D		D	C			C	D
NAND			C, D			D	C	D	C

if we convert the values so as to compare the energy characteristic of each device for the same amount of data, we find out that NOR Flash memory has best energy density in Read operation and SDRAM has best in write operation. A static analysis can be used to derive t_{Li}, t_{Si}, El_i and Es_i in Eq. 3. On the other hand, we have to profile the memory transactions in order to obtain Mc_i, Mr_i and Mw_i, which are the number of instruction cache misses, the number of missed reads of the data cache and the number of missed writs or flushed of the data cache, respectively. We can also perform cycle-accurate simulation of the bus function and derive average values of Mc_i, Mr_i and Mw_i for unit time.

4.3 Memory system energy consumption

Table 2 shows the possible cooperative memory allocation schemes of the heterogeneous non-volatile memory systems. The following three cases explain their energy consumption.

4.3.1 Battery-backed SDRAM

Energy consumption for a burst-mode read access, E_{RSD}, occurs when there is a cache miss in the instruction cache and a read miss in the data cache. Energy consumption for a burst-mode write access is denoted by E_{WSD}. The SDRAM consumes leakage energy, E_{ISD}, at all times except in the self-refresh mode.

The memory system energy consists of energy for instruction fetch, data read and data write, and is given by

$$Em_i = \varepsilon_i(Mc_iE_{RSD} + Mr_iE_{RSD} + Mw_iE_{WSD}). \quad (5)$$

The data retention energy does not vary with the amount of data in conventional SDRAM. Since we assume a single microprocessor, the total execution time of the task set T is denoted by

$$t_A = \sum_{i=1}^{N} \lambda_i t(\varepsilon_i + t_{Li} + t_{Si}). \quad (6)$$

The leakage energy, the data retention energy, and the energy for refresh are as follows:

$$E_K(t) = t_A E_{ISD} \quad (7)$$

$$E_R(t) = E_{TSD}(t - t_A) \quad (8)$$

$$E_F(t) = E_{FSD}\lceil \frac{t_A}{15.6us} \rceil. \quad (9)$$

4.3.2 Battery-backed SDRAM and NOR Flash memory

As mentioned in the previous section, NOR Flash memory such as Intel StrataFlash supports XIP without noticeable performance degradation. Thus cooperative operation of the SDRAM and the NOR Flash memory means that the code and the data are stored in the NOR and the SDRAM, respectively. The memory system energy is given by

$$Em_i = \varepsilon_i(Mc_iE_{RNR} + Mr_iE_{RSD} + Mw_iE_{WSD}). \quad (10)$$

Another alternative is to store not only the code but the read-only user data, such as MP3 music files or JPEG images, in the NOR Flash memory. The memory system energy is given by

$$Em_i = \varepsilon_i(Mc_iE_{RNR} + (1-\eta)Mr_iE_{RSD} + \eta Mr_iE_{RNR} + Mw_iE_{WSD}), \quad (11)$$

where η is the proportion of NOR Flash memory accesses among the data cache read miss transactions. When a read transaction occurs in the NOR Flash instead of in the SDRAM, it requires an additional access time, $T_{RNR} - T_{RSD}$, and thus the energy consumption of the microprocessor increases proportional to $T_{RNR} - T_{RSD}$.

Table 3: Examples of the user profile (minutes).

State	t'	MP3 $(\lambda_i t', \varepsilon_i)$	MPEG4 $(\lambda_i t', \varepsilon_i)$	CJPEG $(\lambda_i t', \varepsilon_i)$	DJPEG $(\lambda_i t', \varepsilon_i)$
s_1	120	1, 20	1, 5	0, 0	2, 5
s_2	180	2, 10	1, 10	1, 10	2, 5
⋮	⋮	⋮	⋮	⋮	⋮
s_n	180	1, 10	2, 20	0, 0	0, 0
D_i (MB)		20	8	10	9

4.3.3 Battery-backed SDRAM and NAND Flash memory

As the NAND Flash memory is the secondary storage, it must cooperate with the SDRAM primary storage. There are two choices: either the code and the data are both stored in the NAND Flash memory or the code is stored in the SDRAM and the data is stored in the NAND Flash memory. The former involves additional time and energy for loading both code and data, and the latter involves time and energy for the data only. Once the data has changed after execution, the data is again transferred to the NAND Flash memory. We assume that all the data is written back with a part of changed content since there is usually a file system on the NAND Flash memory. This rather overestimates the migration overhead in later sections, and thus underestimates the power reduction.

The time for loading and saving the code and the data are given by

$$t_{Li} = \lceil \frac{C_i + D_i}{512\text{bytes}} \rceil T_{RND} + \lceil \frac{C_i + D_i}{4\text{words}} \rceil T_{WSD} \quad (12)$$

and

$$t_{Si} = \lceil \frac{\rho_i D_i}{4\text{words}} \rceil T_{RSD} + \lceil \frac{\rho_i D_i}{512\text{bytes}} \rceil T_{WND}. \quad (13)$$

In the same way, the energy for loading and saving are denoted by

$$El_i = \lceil \frac{C_i + D_i}{512\text{bytes}} \rceil E_{RND} + \lceil \frac{C_i + D_i}{4\text{words}} \rceil E_{WSD} \quad (14)$$

and

$$Es_i = \lceil \frac{\rho_i D_i}{4\text{words}} \rceil E_{RSD} + \lceil \frac{\rho_i D_i}{512\text{bytes}} \rceil E_{WND}. \quad (15)$$

As observed in the battery-backed SDRAM-only memory system, the SDRAM requires the same amount of leakage energy and refresh energy during task execution (i.e., $E_K(t)$ and $E_F(t)$ are equal to Eqs. 7 and 9, respectively). If both code and data are stored in the NAND Flash memory, we do not have to retain the data in the SDRAM, and thus $E_R(t) = 0$, otherwise $E_R(t)$ is the same as in Eq. 8.

5. ENERGY-AWARE MEMORY ALLOCATION

5.1 Target memory system architecture

The target system is equipped with a heterogeneous non-volatile memory system. A 64MB mobile SDRAM memory system is composed of two SDRAM devices, K4S56163LC: 4banks × 4M × 16bit. A 16MB NOR Flash memory system is composed of two NOR Flash memory devices, Intel StrataFlash 28F640J3A: 4M × 16bit. A 64MB NAND Flash memory system is composed of a NAND Flash memory device, K9K1208U: 64M × 8bit.

5.2 Application programs and user profiles

We considered four embedded applications: an MP3 decoder, an MPEG4 player, a JPEG compressor and a JPEG decompressor.

Table 4: Experimental results for static and dynamic allocation (J).

Profile No.	Typical allocation			Static allocation					Dynamic allocation				
	Total	Memory	Allocation	Total	%	Memory	%	Allocation	Total	%	Memory	%	Allocation
1	579	316	6-6-6-6	517	89.3	241	76.2	8-8-8-9	515	88.9	239	75.6	Table 5
2	634	332	6-6-6-6	601	94.9	283	85.2	8-8-8-9	563	88.8	246	74.1	
3	571	290	6-1-6-7	555	97.2	261	90.0	5-5-8-2	553	96.8	260	89.6	
4	735	400	6-1-6-6	713	97.0	365	91.3	8-8-5-8	702	95.5	354	88.5	

Table 5: Dynamic allocation and migration costs (J).

Profile No.		State					
		s1	s2	s3	s4	s5	s6
1	Alloc.	8-8-3-9	8-8-8-9	8-8-3-9	8-8-8-9	8-3-8-9	8-8-8-9
	Cost	-	0.28	0.012	0.28	0.044	1.11
2	Alloc.	5-2-5-8	8-2-8-8	8-2-8-8	5-2-5-8	8-2-8-8	8-2-8-8
	Cost	-	4.61	0	0.72	4.61	0
3	Alloc.	5-8-5-2	8-5-5-2	8-5-5-2	5-8-5-2	-	-
	Cost	-	2.91	0	2.43	-	-
4	Alloc.	5-8-8-8	8-5-8-8	8-8-5-8	8-8-8-5	-	-
	Cost	-	2.91	2.35	2.33	-	-

We chose four profiles with 25% to 30% average system utilization (i.e., the duty ratio of power-on and power-off states). Profile 1 consists of sub-states whose system utilizations are similar to each other. Each task is also assumed to have similar execution times, and the number of executions is assumed to be small. We further assume that the size of the user data is small enough to avoid allocation to the SDRAM if we do not want to lose the data retention energy. In Profile 2, on the other hand, each sub-state is assumed to have a different system utilization, with large variations between them. Other conditions are the same as in Profile 1. Profile 3 is similar to Profile 1, but each task require a lot of data so that allocation to the SDRAM is inevitable, even though we want to avoid expending data retention energy. Profile 4 is similar to Profile 3, but each task has a different execution time, and the number of executions at each sub-state is different as well. Table 3 presents examples of the user profiles.

5.3 Static allocation

Static allocation fixes the memory allocation in advance and this initial allocation does not change. With given n tasks and three types of non-volatile memory system, we have 9^n possible memory allocation schemes. Exhaustive search requires 9^n trials which implies a non-polynomial time complexity. However, since we can determine $E_{Xi}(t)$ and $E_M(t)$ for all i in advance, allocation is equivalent to the well-known Knapsack problem.

Allocation is now straightforward. First, we derive $E_{Xi}(t)$ and $E_M(t)$ for the nine feasible allocations. The complexity of this equation is only $9N$, and we do not have to simulate the energy consumption $9N$ times because Mc_i, Mr_i and Mw_i are independent of the configuration. We derive the optimal allocations for time t using a heuristic approach to the Knapsack problem. Here, t is the time to the next battery recharge.

Table 4 shows the energy reduction achieved by the energy-aware memory allocation for the heterogeneous non-volatile memory systems. The total energy consumption includes energy for the CPU and the memory systems. The four digits in the allocation column mean the allocation number of MP3, MPEG4, CJPEG and DJPEG (Table 2) in the order of appearance. We compare the energy consumption of our scheme with a typical allocation scheme for commercial PDAs, which allocates the codes and small amount of user data to the SDRAM, extensive user data to the NAND Flash memory, and default applications to the NOR Flash memory. The energy gain is 10% to 24% depending on the characteristics of the task and user profiles. This gain is significant but variable to user profiles and applications.

5.4 Dynamic allocation

Static allocation sometimes fails to find an allocation that is distinctly superior to others. Dynamic allocation changes the initial allocation as the user profile changes. Table 3 shows that user behavior can be divided into several sub-states. Each sub-state shows different task invocations for time t' and thus different optimal memory allocation. The proposed dynamic allocation migrates the allocation when the sub-state changes. The dynamic allocation involves a migration overhead as shown in Table 5.

Table 4 also shows the energy gain achieved by dynamic allocation. If each memory device has enough capacity to store all the applications in one device, allocation is a trade-off between the data retention energy and the migration overhead. Profile 1 shows similar energy reduction for the static and the dynamic allocations, while Profile 2 shows significant further energy reduction after the dynamic allocation. If each memory does not have enough capacity to avoid allocation to the SDRAM, we must expend the data retention energy. In this case, we can save the energy during active mode by allocating the tasks to the SDRAM as much as possible, considering task behavior and energy consumption for active mode. Profile 3 shows the reduction of active energy for static and dynamic allocation. Because task behavior is similar in each sub-state, static and dynamic allocations show similar energy reduction ratios, while Profile 4 shows that dynamic allocation achieves more energy saving than static allocation.

6. CONCLUSIONS

We propose an energy-aware memory allocation of code and data in heterogeneous non-volatile memory systems, considering not only active mode energy but also data retention energy. For this purpose, we derive analytical energy consumption models of various heterogeneous non-volatile memory systems based on the high-fidelity energy characterization of memory devices.

Energy-optimal memory allocation is achieved by taking into account the active-mode energy, the idle-mode energy and the data retention energy, which are affected by the device characteristics as well as the code, the data and the user profile. We present a static allocation and a dynamic migration scheme, and analyze energy savings under diverse user profiles and task models reflecting practical patterns of use. Trace-driven simulation results demonstrate up to 26% of memory system energy reduction over traditional allocations.

7. REFERENCES

[1] M. Newman and J. Hong, "A look at power consumption and performance on the 3Com Palm Pilot," *UC Berkeley CS252*, Spring 1998.

[2] R. Lee and R. Nathuji, "Power and performance analysis of PDA architectures," *Advanced VLSI Computer Architecture*, Fall 2000.

[3] T. Simunic, L. Benini, and G. D. Micheli, "Energy-efficient design of battery-powered embedded systems," in *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 212–217, August 1999.

[4] S. L. Coumeri and D. E. Thomas, "An environment for exploring low power memory configurations in system level design," in *Proceedings of ICCD*, pp. 348–353, September 1999.

[5] A. R. Lebeck, X. Fan, H. Zeng, and C. S. Ellis, "Power aware page allocation," in *Architectural Support for Programming Languages and Operating Systems*, pp. 105–116, 2000.

[6] Y. Joo, Y. S. Choi, H. Shim, H. G. Lee, K. Kim, and N. Chang, "Energy exploration and reduction of SDRAM memory systems," in *Proceedings of DAC 2002*, pp. 892–897, June 2002.