Using Asymmetric Cores to Reduce Power Consumption for Interactive Devices with Bi-stable Displays

Jaeyeon Kihm **François Guimbretière** **Julia Karl** **Rajit Manohar**
Cornell University
Ithaca, NY 14850
jk2443@cornell.edu francois@cs.cornell.edu julia@csl.cornell.edu rajit@csl.cornell.edu

ABSTRACT

Low power "helper" cores have been increasingly included on application processors to accomplish low intensity tasks such as music playing and motion sensing with minimum energy consumption. Recently, Guimbretière et al. [1] demonstrated that such helper cores could also be used to execute simple user interface tasks. We revisit this approach by implementing a similar system on an off-the-shelf application processor (TI OMAP4). Our study shows that in the case of high event rate interactions (pen inking and virtual keyboard), significant battery life gains (×1.7 and ×2.3 respectively) can be achieved with the helper core executing the interface. Having the helper core only dispatch input events incurs a 18% penalty relative to the maximum savings rate, but allows for simplified deployment since it merely requires a change in toolkit infrastructure.

Author Keywords
Energy efficiency, User interface system, Asymmetric architecture, Pen interaction, Bi-stable display

ACM Classification Keywords
H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

INTRODUCTION

The power efficiency of everyday information appliances such as general-purpose slates or specialized ebook readers has made striking progress in recent years. Most of this progress can be traced back to improvements in the underlying hardware such as the use of dynamic voltage and frequency scaling to optimize power management, the use of solid state disks, and in some cases, the use of bi-stable displays which consume power only while refreshing pixels. Redesigning the interface can further reduce power consumption [3, 5, 6]. Recently, Guimbretière et al. [1] introduced a new solution for devices using bi-stable displays which does not require visual changes to the interface. They observed that common reading tasks, such as turning a page or annotating, can be performed using a slow, yet highly efficient micro-controller without having the main processors involved (akin to playing music in the background). Using a custom design board combining an application processor (TI OMAP3) and a microcontroller driving a bi-stable e-ink display, they demonstrated a possible increase of battery life by a factor of 1.7 for reading and 3.2 for writing.

To further explore the potential of using asymmetric dual-core to run the user interface, we re-implemented Guimbretiere et al. approach [1] on an off-the-shelf application processor, the TI OMAP 4460, featuring two high-performance ARM Cortex-A9 (A9) cores and two low power ARM Cortex-M3 (M3) cores. Since the TI OMAP 4460 has the shared memory system for all cores, we streamlined the inter-processor communications and assigned a shared memory for the whole interface description tree in order for all cores to access it directly.

Using our prototype, we conducted an experiment comparing four main settings: 1) the traditional single application processor, 2) the single application processor but switching the display driver on only when it is needed; 3) the M3 only dispatches user inputs and an application processor executes callbacks, and 4) the M3 dispatches inputs and executes callbacks. All four settings offer similar end-users' experiences, and our results confirm the benefits of using low power cores for high event rate user interactions such as inking (×1.7 in battery life) or a virtual keyboard (×2.3). In low event rate interactions such as reading, the benefit was much smaller and driven by display driver management savings. For high event rate interactions, the M3 dispatch only setting was 18% less efficient than the M3 dispatch and execute setting, but had the advantage of being easier to deploy requiring no recompiling of current applications for new architecture.

RELATED WORK

Several previous systems have studied overall interface design for optimized power consumption. Vallerio [5] demonstrated that by reformatting the interface layout to optimize pointing performance or by introducing auto-completion mechanisms, one can increase user speed and thus reduce energy requirements for short interactions. Zhong and Jha [6] utilized a secondary display (i.e. the display of a watch) and voice recognition to reduce the overall energy consumption of the primary display for interaction. These solutions are not well suited for interactions such as

CHI 2014, April 26 - May 01 2014, Toronto, ON, Canada
Copyright 2014 ACM 978-1-4503-2473-1/14/04ⓢ$15.00.
http://dx.doi.org/10.1145/2556288.2557135

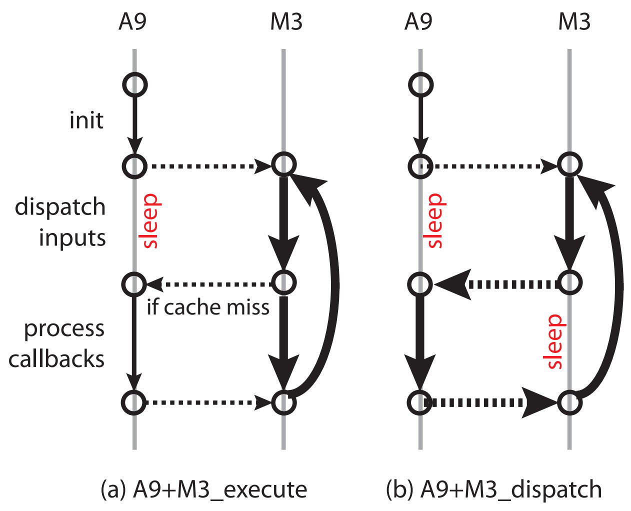

Figure 1. Asymmetric dual-core system. (a) The M3 for dispatch and callbacks, and (b) for dispatch only. The stroke width represents the execution frequency.

reading or note-taking since the display needs to stay on for prolonged periods of time. Harter et al. [3], in turn, studied how users react to the visual redesign of an interface to leverage the fact that in an OLED display, "off" pixels have a low energy signature.

More recently, Guimbretiere et al. [1] studied the possible impact of bi-stable displays on the energy signature of interfaces. Bi-stable displays only consume power while being updated, but then maintain an image with no or little power. While the first generation of bi-stable displays offered a low refresh rate, the most recent systems, such as the Mirasol display by Qualcomm, are video capable. Noticing that many interactions such as page turning and annotations require little computational power, Guimbretiere et al. proposed an architecture combining an application processor with a micro-controller. During high demand tasks, the application processor runs as usual, but when the demand is reduced, the application processor shuts off and the micro-controller handles user interactions. Using custom hardware, they demonstrated improvement varying from 1.7 to 3.2 in energy demand, while this approach does not require a significant redesign of the interface.

In this paper, we revisit Guimbretiere et al.'s approach by demonstrating its implementation on commodity asymmetric multi-core hardware (OMPA4) and comparing two possible policies for the use of the power efficient core. We empirically evaluate energy savings for different tasks related to reading and note-taking. As asymmetric multi-core designs (e.g., the ARM big.LITTLE architecture) are becoming the norm, it is important to systematically explore their energy-savings potential.

LOW POWER USER INTERFACE FRAMEWORK

The system design proposed by Guimbretiere et al. [1] introduced a trade-off between power management and communication bandwidth to combine two asymmetric processors. Their custom board design allowed for a complete shut-off of the Linux processor including memory and peripherals to maximize energy savings. At the same time, the design offered only limited bandwidth between the two processors making it difficult to share the interface state between them.

We re-implemented this system on a commodity chip, the TI OMAP4. The OMAP4 is a typical application processor including two high performance ARM Cortex-A9 cores (running at up to 1.2 Ghz) and two energy efficient ARM Cortex-M3 helper cores (running at up to 384 Mhz) as well as interfaces modules to manage memory, displays, cameras, networks and USB to name a few. The A9 core and the M3 core are running at a very different efficiency points. Thus, preventing the the A9 from processing small tasks such as dispatching user events will result in energy savings. Since both cores share memory access, this configuration offers a greater bandwidth between core. This might come at the cost of a higher power signature than the configuration used by Guimbretiere et al. [1] because some subsystems always stay on.

Role of the low core processor

Considering the OMAP4 architecture, the logical extension of Guimbretiere et al.'s design would utilize the low power M3 not only to execute event dispatches but also the corresponding callbacks, thus minimizing the use of the high power A9. In that setting, the main system first initializes an application by loading a user interface tree and caching image data on the shared memory region. Then it sends a message to the M3 to initiate user interactions. The A9 can now safely fall asleep and the M3 will execute interface events upon user inputs until the use of the A9 is required. When this happens, the M3 will issue a remote-procedure call to perform the required computation such as updating a page cache (Figure 1a).

This approach promises to be very energy efficient, but, as in Guimbretiere et al.'s solution, it requires developers to review the application code to identify which part of the code should be run on the M3 and which part must be run on the A9. Developers must then re-structure the code to manage transitions between the two architectures. Depending on the compiler support, this could be done either by inserting pragma, using specific system libraries embedding the remote procedure call, or introducing explicit remote procedure calls. Most likely the latter approach (and the most labor intensive) will provide better results, but in all cases the executable must be recompiled anyway.

The dispatch only approach

To address this problem, we considered a new setting in which the M3 role is limited to dispatching events along the interface tree to identify a proper callback. This callback is then executed by the A9 (Figure 1b). The dispatch process can be very inefficient on the A9 when the full OS is active to process simple callbacks because many events do not lead to a callback. For example, in a pen-based interaction, the system will generate hover events, but often an application does not assign callbacks for them. Although one could dynamically filter the input stream, the use of a low power dispatch offers more control to the current program. In this configuration, developers do not need to know about the use of the M3. Thus, legacy applications can be run with little or no modifications depending on the linking strategy.

Implementation

We use a Pandaboard ES embedding a TI OMAP 4460. A Wacom sensor or a keyboard is connected through a serial port accessible to both cores. Ubuntu Linux with Kernel 3.7 (with power management) runs on Cortex-A9 with all the unnecessary system modules disabled. SYS/BIOS 6.32 runs on Cortex-M3. RPMsg, an inter-processor communication driver, is used to call remote procedures.

EVALUATION

The main focus of our evaluation was to better understand overall energy efficiency of power management policies for information appliances using bi-stable display. We considered four reference configurations:

A9_single: The reference single-core system. The OMAP4 display subsystem is always on, but the display is activated only when it should be refreshed. This represents the simplest implementation.

A9+display_control: The same configuration as A9_single, but we control the OMAP4 display subsystem to leverage the bi-stable display.

A9+M3_execute: The M3 dispatches events and executes callbacks. The A9 sleeps except when it is strictly required. The display sub-system runs on demand.

A9+M3_dispatch: The M3 dispatches events, and the A9 executes the corresponding callbacks using remote procedure calls. The display sub-system runs on demand.

Acknowledging that results may be task dependent, we considered two task domains - reading and note-taking - as important aspects of reading activities. With respect to note-taking, we considered 3 different approaches spanning a range of computing requirements: inking, a virtual keyboard using a pen, and a standard keyboard.

Energy Measurements

The main dependent variable for our experiment is the energy consumed to perform a given task in each setting. We measure energy consumption on each by replacing filter beads with 1% resistors of known values. Wacom power is measured by a separate shunt to the Wacom controller. These real-time measurements are sampled by a National Instruments USB-6251 controlled by a custom design LabView application. One of the key features of bi-stable displays is that the application does not have to drive the display when it is not in use, thus further increasing power savings. Since appropriate bi-stable displays are not yet available, we cannot switch off the display controller while users interact with the system. To obtain accurate data, we therefore captured accurate interaction logs and then replayed them in a same system configuration with display controls. While all data are sampled in real-time, bi-stable display power consumption is estimated by referring

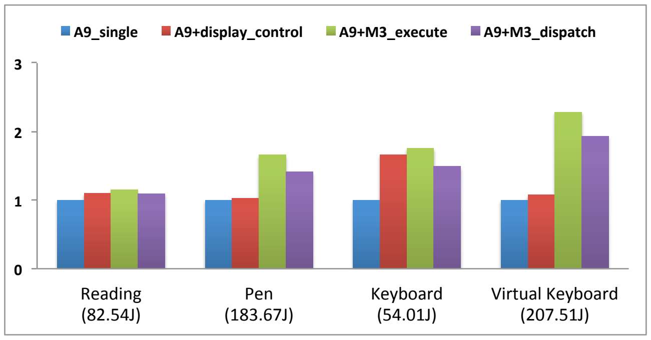

Figure 2. Total battery extension of tasks in configurations. The number in the bracket next the task label is the energy consumption of A9_single in each task.

Mirasol power data [2]. Except data for the A9_single, the analysis below reflects the data captured during replays.

Protocol

The experimental procedure consisted of demonstrating a task, having users try out the interface, and finally having users complete the designated tasks. All participants completed four tasks in all configurations excluding A9+display _control which was simulated using A9_single data. We assigned different datasets to each technique using a Latin Square to address order effects, and different documents in each condition to limit learning effects. For the reading task, participants are asked to read four pages (188 ± 8 characters on each page), and for the note-taking task, type or write down a short article (61 ± 6 characters) at their own pace. We recruited 9 participants with prior experience on mobile devices. They received $10 for a 1.5-hour session.

RESULTS

We use the Greenhouse-Geisser correction to address deviations from sphericity and the Bonferroni correction for all pairwise comparisons. Reading and annotation tasks are analyzed separately. For the **reading task**, a one-way repeated measure ANOVA on total energy consumption shows a main effect for technique, ($F(3,24) = 27.29$, $p < .001$, partial $\eta^2 = .773$). Pairwise comparisons show a significant difference between A9_single and all other conditions ($p < .005$), but no other significant differences. This suggests that the modest savings (×1.10 - 1.15) are primarily driven by the direct control of the display.

With respect to the **note-taking tasks**, a two-way repeated measures (task×condition) ANOVA on total energy consumption shows main effects of task ($F(2, 16) = 105.3$, $p < .001$, partial $\eta^2 = .929$) and condition ($F(1.065, 8.517) = 195.8$, $p < .001$, partial $\eta^2 = .961$) qualified by a strong interaction ($F(1.590, 12.724) = 98.514$, $p < .001$, partial $\eta^2 = .925$). We therefore report a task-by-task analysis.

The **keyboard entry task** shows a pattern similar to the reading task. A one-way repeated measures ANOVA shows a main effect of technique ($F(1.207,9.658) = 109.7$, $p < .001$, partial $\eta^2 = .932$). Pairwise comparisons show a significant difference between A9_single and all other conditions ($p < .001$). Further, A9+M3_execute offers a signifi-

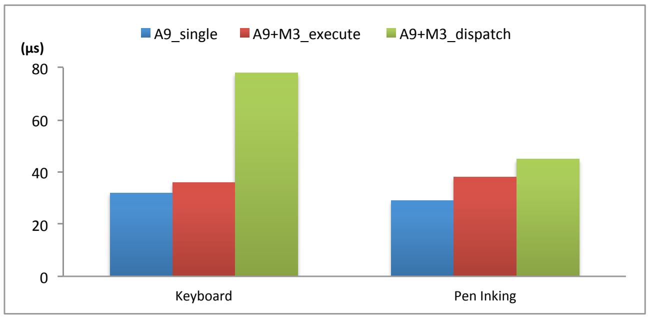

Figure 3. Average display latency

cant increase in battery life (×1.76) over display-control (×1.66) and the simpler A9+M3_dispatch (×1.48). The overhead caused by the remote procedure calls in the A9+M3_dispatch might explain these results.

With respect to the **virtual keyboard**, a one-way repeated measures ANOVA shows a main effect for technique (F(1.013, 8.107) = 143.5, $p < .001$, partial $\eta^2 = .947$). Pairwise comparisons show that all pairs are significantly different ($p < .001$), with A9+M3_execute offering the best battery life improvement (×2.28), followed by A9+M3_dispatch (×1.93) and display control (×1.08). The **inking task** follows a very similar pattern (F(1.243, 9.943) = 187.3, $p < .001$, partial $\eta^2 = .959$). All pairwise comparisons are significantly different ($p < .001$), with A9+M3_execute offering the best battery life improvement (×1.66), followed by A9+M3_dispatch (×1.41), and display-control (×1.03).

Task time and latency

Our results show that there is no statistically different between task times. The reading task took 224s for A9_single, 221s for A9+M3_execute and 225s for A9+M3_dispatch. The inking task took 148s, 144s, and 148s respectively, the virtual keyboard took 222s, 220s, and 223s respectively, and the keyboard task took 83s in all three configurations. With respect to the **latency**, we measured the time from user inputs to display update for keyboard and writing interactions by replaying user traces. We show the results in Figure 3. While the differences between setting are statistically significant (p<.001) for both keyboard and pen writing tasks, the differences between A9_single and A9+M3_execute are small. The A9+M3_dispatch configuration is slower probably due to the message passing mechanism. In all cases, the values are very small, and therefore, they should not impact user experience.

DISCUSSION

Our results suggest that extending Guimbretiere et al.'s [1] approach to off-the-shelf commodity hardware can realize significant energy savings although savings rates are more modest than those seen for custom hardware. This may be explained by the fact that the present system uses a Linux kernel, which is already implementing an effective power management scheme. For **low event rate tasks** (reading and keyboard text entry), battery life improvement is primarily driven by more efficient control of the display and whereas full use of the M3 offers small additional improvements (about 5%), the M3 dispatch-only is counterproductive in such tasks because of the extended use of remote procedures. For **high event rate tasks** (writing and virtual keyboards), in contrast, efficient display control only offers small benefits (×1.05), but the full use of the M3 brings significant improvements in battery life (×1.6 - 2.3). If the M3 is only used for dispatch, energy savings in high event rate tasks are about 18% lower but still substantial.

Our findings imply that by simply using the low power core to dispatch pen events in interactive frameworks, designers can realize significant system wide power savings. This might be even more important in the near future as designers can leverage high refresh displays to reduce interaction latency [4]. Our results suggest that document navigation such as Space-filling Thumbnails can take advantage of this approach, and ultimately making it possible to create a highly efficient digital documents review system. We believe that interactions such as Marking Menu, which require limited feedback from expert users will also benefit from this design.

CONCLUSION AND FUTURE WORK

In this paper, we extended Guimbretière et al.'s [1] approach to a commodity processor (TI OMAP4) and demonstrated that, in high event rate applications, it provides significant energy savings (up to ×2.3). Somewhat more modest savings (about 18% less) could be achieved by having the small core (M3) only dispatch the input events, but this solution is simpler to deploy. We plan to build a toolkit to take full advantage of our system on current mobile architectures before conducting more extensive evaluations.

ACKNOWLEDGMENTS

This work was supported by a grant from NSF (IIS - 0916217). We thank Ohad Ben-Cohen, Omar Ramirez, and Robert Nelson for their contributions on development.

REFERENCES

1. Guimbretiére, F., Liu, S., Wang, H., and Manohar, R. An Asymmetric Dual-Processor Architecture for Low Power E-ink Display. *ACM Transactions on Embedded Computing Systems* (in press).
2. Gusev, E. Mems Growing Mainstream; Mirasol Display Technology. 2011. http://asdn.net/ngc2011/presentations/gusev.pdf.
3. Harter, T. and Vroegindeweij, S. Energy-aware user interfaces: an evaluation of user acceptance. *In proc. of SIGCHI 2004*, (2004), 199–206.
4. Jota, R., Ng, A., Dietz, P., and Wigdor, D. How fast is fast enough?: a study of the effects of latency in direct-touch pointing tasks. *In proc. of the SIGCHI 2013*, (2013), 2291–2300.
5. Vallerio, K.S., Zhong, L., and Jha, N.K. Energy-efficient graphical user interface design. *Mobile Computing, IEEE Transactions on 5*, 7 (2006), 846–859.
6. Zhong, L. and Jha, N.K. Energy efficiency of handheld computer interfaces: limits, characterization and practice

. *Proceedings of MobiSys 2005*, ACM (2005), 247–260.