An Asymmetric Dual-Processor Architecture for Low Power Information Appliances

François Guimbretière, Shenwei Liu, Han Wang, Rajit Manohar, Cornell University

As users become increasingly conscious of their energy footprint—either to improve battery life or to respect the environment—improved energy efficiency of systems has gained in importance. This is especially important in the context of information appliances such as Ebook readers that are meant to replace books, since their energy efficiency impacts how long the appliance can be used on a single charge of the battery.

In this paper, we present a new software and hardware architecture for information appliances that provides significant advantages in terms of device lifetime. The architecture combines a low power micro-controller with a high-performance application processor, where the low power micro-controller is used to handle simple user interactions (e.g., turning pages, inking, entering text) without waking up the main application processor. We demonstrate how this architecture is easily adapted to the traditional way of building user interfaces using a user interface markup language. We report on our initial measurements using an E-ink-based prototype. When comparing our hybrid architecture to a simpler solution we found that we can increase the battery life by a factor of 1.72 for a reading task, and by a factor of 3.23 for a writing task. We conclude by presenting design guidelines aimed at optimizing the overall energy signature of information appliances.

Categories and Subject Descriptors: H.5.2 [**Information interfaces and presentation**]: User Interfaces

General Terms: C. Computer Systems Organization C.3 Special-purpose And Application-based Systems Subject: Real-time and embedded systems

Additional Key Words and Phrases: Human Computer Interface, Interface design, Energy consumption

ACM Reference Format:
Guimbretière, F., Liu, S., Wang, H., Manohar, R. 2011. An Asymmetric Dual-Processor Architecture for Low Power Information Appliances. ACM Trans. Embedd. Comput. Syst. XX, XX, Article XX (May 2011), 19 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Users are becoming increasingly conscious of the energy their information appliances consume, either because they would like to maximize the use of their battery power or because they aim to limit their environmental footprint. Yet, this area of research has seen limited progress compared to other performance metrics such as processor speed or available storage capacity. In fact, as most of us are aware, very few systems today fulfill Mark Weisers vision of “several days of continuous use” for the “Computer for the 21st Century” [Weiser 1991]. Battery life is, however, the corner-stone that allows people to enjoy the advantages of an environment containing “100 tabs, 10 or 20 pads

This work is supported by the National Science Foundation, under grant ISS-0916217. Additional support was provided by FXPAL.
Author’s addresses: François Guimbretière, Shenwei Liu, Han Wang, Information Science, Cornell University, Ithaca NY 14853, USA. Rajit Manohar, Computer Systems Lab, Cornell University, Ithaca, NY 14853. USA.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

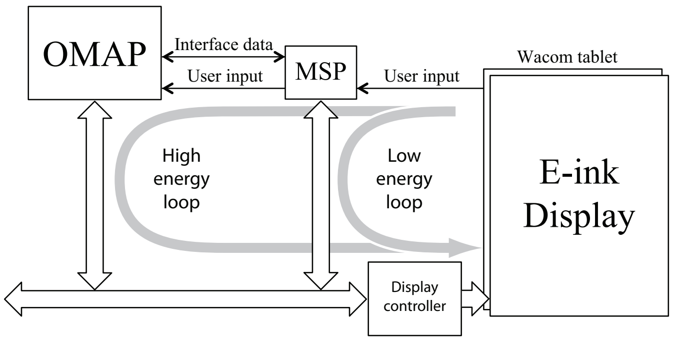


Fig. 1. Our Ebook reader architecture combines an OMAP and a MSP processor, both connected to the E-ink display controller. The OMAP is only used for high demand tasks, while the MSP can be used for simple interactions and improving system reactiveness when using low-power modes of the OMAP.

and one or two boards" [Weiser 1991]. Without sufficient battery life, the burden of maintaining a large number of devices far outweighs their advantages.

This problem is well recognized with regard to handheld devices, and a large body of work has been published on reducing their energy requirements [Chen et al. 2004; Rudenko et al. 1998; Wang and Li 2004; Li et al. 2001; Zhong 2005; Vallerio et al. 2006; Harter et al. 2004]. Hardware techniques such as voltage and frequency scaling have made it possible to significantly lower energy consumption of modern processors. From a software perspective, advanced scheduling algorithms have demonstrated the gains that can be made with respect to process and disk access scheduling. From a user interface perspective, recent work has shown that it is possible to redesign interfaces to make them more energy efficient [Vallerio et al. 2006; Harter et al. 2004; Zhong and Jha 2005]. These techniques are well adapted to the pattern of use typical to LCD-based PDAs in which users glance at the display for short times and then switch it off. In contrast, such techniques do not fare as well for other tasks, such as reading a book, in which devices have to display information for long periods of time.

Recently, book reading has taken on an increasingly large role for slate devices (such as the Apple iPad and Amazon Kindle). In part, this was driven by the availability of bi-stable displays such as the E-ink display used in the Kindle. Bi-stable displays can retain their image with either no power or very low levels of power. While the first generation of bi-stable display technology was characterized by a low refresh rate and a grey scale display, the new generation promises to provide full color and video capable displays [Liquavista 2008; Mirasol 2010]. Bi-stable displays offer the opportunity to create information appliances (e.g., Ebook readers, family calendars) with very small energy signatures. However, there has been very little research exploring how to maximize their potential by limiting energy consumption during active reading.

Because bi-stable displays only consume power when updated, the CPU dominates the energy signature of such systems. As a result, reducing the CPU power consumption is even more critical than for LCD-based devices. Many (if not all) modern high-performance CPUs support power management hardware and sleep modes. While these modes provide some benefit, they present drawbacks in the context of interactive appliances such as Ebook readers because the time taken to wake-up from sleep mode and restore system state is too high to support fluid user interactions. For example, re-starting the system when the pen is used to annotate text results in a lag between the location of the pen and where the ink is being drawn on the screen. This lag is severe enough to qualitatively change user behavior.

To address this problem, we start from the observation that many user interactions such as typing text, inking, or turning pages do not require the full power of a GigaHz application processor which consumes hundreds of mW in active state. Instead they can be executed by a micro-controller class device consuming only a couple of mW while active. Thus, we propose a dual processor architecture which seamlessly combines an application processor and a micro controller. Specifically, the micro-controller is used to handle simple user interactions and hide the resume latency of the application processor. The application processor, in turn, provides the computational resources required by more complex operations (Figure 1). In combination, this provides an always-on, highly reactive user interface with low power consumption while maintaining access to high computational resources on an as-needed basis.

To demonstrate the potential of this architecture for controlling bi-stable displays, we implemented a prototype of our architecture controlling an E-ink display and compared its performance to a standard implementation relying exclusively on an application processor. Our results show that our system can reduce the energy consumption of the system by up to 42% for a reading task and 69% for a writing task and the energy consumed by processors by 54% and 88% respectively. This translates to an improvement is battery life for the full system of 1.72× for reading, and 3.23× for writing. We conclude by suggesting design guidelines to adapt interfaces to maximally leverage the proposed dual processor architecture. We also discuss the potential impact of our approach in designing interfaces for bi-stable displays with high refresh rates (e.g, the upcoming Mirasol display).

2. RELATED WORK

As pointed out by Zhong [Zhong 2005], current portable device designers face a profound mismatch between the increasing speed of processors and a fixed human reaction time. This has detrimental results for energy consumption. While the user is processing information and deciding what to do, resource hungry components of the system, such as the display, are draining energy which results in reduced battery life. The obvious solution to this problem is to reduce the time the device stays on. Several techniques have been proposed to decrease the time it will take a user to perform a typical task. These include restructuring the interface by increasing the size of buttons and placing them closer to each other to leverage Fitts law [Fitts 1954; Vallerio et al. 2006], and offering auto-completion mechanisms to reduce keyboard text entry time [Vallerio et al. 2006]. Designers can also opt for auxiliary low power displays like that of a watch to display pieces of information [Zhong and Jha 2005], or consider alternative input techniques (e.g., voice recognition) when the computation requirements of these techniques are lower than maintaining the display on [Zhong and Jha 2005]. For display technologies like OLED, where consumption depends on the number of pixels lit, it is also possible to redesign the visual appearance of the interface to reduce energy consumption [Harter et al. 2004].

These approaches have proven effective in reducing the energy requirements for portable devices. Unfortunately, they also have several drawbacks. First, they work best for interactions that only last for short periods of time (e.g., making quick computations on a calculator or checking an appointment on ones PDA). However, PDAs and Smartphones are increasingly used to access digital content such as web pages and digital books. These tasks require the display to be on for long periods of time and significantly limit the applicability of the previously described techniques.

At the same time, new developments in hardware technology might fundamentally change how information appliances (Ebook readers, PDAs, family calendars, etc.) consume energy. The most notable development is the availability of bi-stable displays, which only consume power when the display is changed. Several such technologies are commercially available or in the final development stages, including Nemoptics BiNem technology [Nemoptic 2006], E-inks electronic ink technology [E-Ink 2006] (used extensively in Ebook readers [Amazon 2007; Irex Technologies 2006; Sony 2006] because of its high readability), and Philips electrowetting displays [Liquavista 2008]. The current generation of these displays still suffer from a slow refresh rate which limits their utility in highly interactive systems. It is expected, however, that steady technological advances will move the frame rate beyond 10fps in the next couple of years—a key threshold for interactive systems [Card et al. 1983; Mackinlay et al. 1990; Robertson et al. 1989]. Using bi-stable displays will fundamentally relax design constraints for information appliances, since the time the display is on has no impact on the energy consumption of the device (assuming no refresh).

2.1. Heterogeneous architectures

There has been a large body of work in the architecture community for creating a range of heterogeneous cores in the context of modern multi-core architectures. The work that is closest to ours is the proposal to integrate two different generations of cores in a single architecture family [Kumar et al. 2004]. The approach maps different applications to the appropriate core based on application resource requirements [Kumar et al. 2004]. Our work differs from this approach, because Kumar's work maps different applications to different cores depending on the needs of each application. In our work, we take one application (in our case, an e-book reader) and restructure it so that it can be mapped to two different processors—a high-performance processor for complex operations and a low-power processor for simple user-interface management. The approach that we propose is general, and can be adapted by any application where different user interaction modes have different computational requirements. Thus, in our approach, a single application benefits by exploiting the characteristics of both processors.

Our approach is also related to work on sensor networks [Dutta and Culler 2005; Hohlt et al. 2004], which use different radio systems depending of the amount of data they need to transmit, and the Little Rock system [Priyantha et al. 2010]. In Little Rock, a low power processor is used for data gathering while the high power processor remains asleep. The high power processor is only awakened from time to time to process the data gathered. Our approach is also related to the Somniloquy system [Agarwal et al. 2009], where a lower power processor is used to support certain network functions while the personal computer is sleeping. In Somniloquy, stub code executes on the lower power processor to handle certain network functions on behalf of the operating system/application while the main CPU is sleeping. As opposed to Somniloquy, our approach is targeted toward an interactive embedded system that has responsiveness constraints for human interactions. As in these systems, our goal is to offer a better match between resource demands and the hardware used to execute them.

2.2. Clock Frequency and Voltage Scaling

There has been a significant body of work investigating the impact of dynamic clock frequency and voltage scaling (DVFS) on the power consumption of microprocessors [Pering and Brodersen 1998; Burd et al. 2000; Ishihara and Yasuura 1998; Govil et al. 1995; Saha 2011; Simunic et al. 2001; Grunwand et al. 2000]. These studies evaluate scheduling and operating system techniques for DVFS in the presence of time-varying workloads using varying degrees of information about application deadlines, start times, and computation needs. DVFS techniques can be very effective in certain cases—for example, in reducing the power requirements for multimedia and predictable workloads. This can be extended to a larger class of applications by using a rate-matching based approach, where the application and hardware attempt to equal-

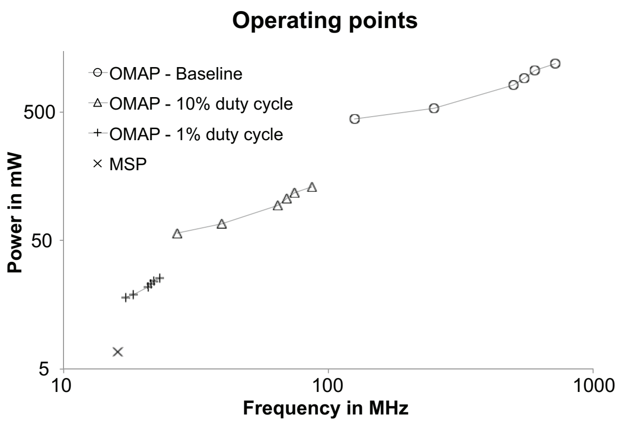


Fig. 2. We show the range of operating points possible for the baseline OMAP processor [Texas Instruments 2013a], the MSP processor [Texas Instruments 2013b], as well as the operating points enabled by our hybrid OMAP/MSP system obtained at different duty cycles of the OMAP processor while keeping the MSP always active and ready to respond to user input.

ize their execution rates to pick an optimal voltage and frequency setting [Simunic et al. 2001; Biermann et al. 2004]. These approaches are effective when one processor supports all the necessary power/performance settings.

A pure DVFS approach has limitations because the only properties changed within the processor are its voltage and operating frequency. Instructions are still executed in the same way as before, which fundamentally means that the complexity of instruction execution has not been reduced even though DVFS can result in significant energy savings. In our work, we create an architecture that is capable of going beyond the limitations of DVFS. We augment the system with a separate, simpler processor that has a fundamentally lower energy per instruction compared to the main high-performance processor. Thus, the operating points available to the application are extended to settings which are unavailable in the base DVFS system. In our experimental work, we used a TI OMAP 3530 which operates at 600 MHz using 1W of power (with a low power DVFS operating point of 125 MHz using ≈ 440mW and sleep mode power of ≈ 10mW) augmented with a TI MSP 430 which operates at 16 MHz using ≈ 7mW of power when fully active and has a sleep mode that consumes only 1μW. Figure 2 shows the list of (frequency, power) operating points attainable using only the baseline OMAP processor, as well as new operating points enabled through our hybrid MSP430-OMAP system design. The work in this paper shows that those new points enable the lifetime of the system to be extended by a significant amount for e-book reader applications.

As more sophisticated processors are manufactured and become available, the attainable (frequency, power) operating points will improve. Even if a single processor can increase the range of points attainable, our hybrid architecture can provide additional benefits. For example, modern microprocessors use different manufacturing technologies for low power versus high-performance design points. Our approach enables us to exploit different processors fabricated in different manufacturing technologies, each optimized for their range of (frequency, power) operating points.

3. ASYMMETRIC DUAL-PROCESSOR ARCHITECTURE

Current techniques to manage power consumption have been quite successful. They combine hardware and software support for modulating the system clock to reflect the current load, managing the voltage to the minimum value necessary for a given clock setting, disconnecting the clock from any modules which have not been used for a short time, and disconnecting power from modules that will not be used for a longer period of time. Looking at a typical application processor like the TI OMAP 3530, this technique can modulate the power consumption of the system from about 1 W when fully active, to about 10 mW when suspended to memory and less than 1 mW when suspended to disk. As a general rule of thumb, the transition time from and to fully active state increases as more energy is saved. These parameters have led to the current interaction paradigm of battery powered devices: upon a timeout, the devices transition to low power mode and transition back upon explicit wake-up by the user. This model is well adapted to the current generation of screens, which can only be read when powered, but breaks down in the context of bi-stable displays.

In systems using bi-stable displays, there is no longer a clear mapping between the state of the display and the overall power consumption. This leads to two problems: First, it becomes more difficult to hide the latency between the high and low power state because there is no clear transition from the users' point of view. One solution, adopted by the Kindle, is to introduce an artificial transition by displaying a special page (similar to a closed book) when the system goes to sleep. This skeuomorphism breaks the metaphor of the Kindle being a substitute for paper. Second, in bi-stable display systems, the processor is the main source of power consumption—even in light usage like drawing annotations or turning pages. Thus, it is important to explore ways to significantly reduce power consumption in active state while preserving the sense that the system is reactive to user input.

To solve both problems, we leverage the fact that one does not need a high power processor to serve simple interactions such as inking or page-turning. Further, we note that more computationally intensive user interactions (e.g., searching for a word, or jumping to a given page) often require several simpler interactions to be set up. To issue a search, for example, users need to type the target word. Similarly, to jump to a given page, users need to enter the page number. Each of these setup stages provides a window of several seconds before higher computational resources are needed. Using a limited, low power processor to serve these setup states will not only save a considerable amount of power but also provide a way to mask the time required to wake-up the larger processor. Thus, we propose a highly asymmetric dual processor architecture in which a very low power processor (consuming only a few mW when active, but only capable of operating at tens of MHz) is coupled with a more traditional application processor. Another way to view our solution is that it is a generalization of the current trend to use dedicated subsystems (like audio and video codec) to reduce power consumption by limiting the time a full, general processor is active; in our approach, the "dedicated subsystem" is a low power (and low performance) programmable general purpose micro-controller.

3.1. Hardware Considerations

Commercially available processors are not designed to span the entire range of power/performance that is of interest for our application. Our prototype uses a TI OMAP 3530 system microprocessor, and the ARM core it contains spans a frequency that ranges from 600 MHz down to 125 MHz. It supports operating voltages from 0.985V to 1.2V. The following first-order analysis shows that this is not sufficient to meet our low power needs. The active power in a digital system is proportional to CV^2f, where C is the average effective capacitance that is switching state, V is the power supply voltage, and f is the frequency of operation [Weste and Harris 2010]. Hence, shifting from the OMAP 3530's high voltage/high frequency operating point to its low voltage/low frequency operating point would reduce power by about a factor of 7.1. However, for the OMAP 3530, this still corresponds to a $\approx$98.6 mW operating point. Thus it is difficult to meet the high performance operating point and low power operating point with one processor, forcing us to adopt a highly asymmetric dual processor architecture.

Combining two different processors with different properties on a single board presents a number of challenges. At the hardware level, the processors have to interact with each other in the same way that they would interact with I/O devices, since micro-controllers typically do not support sophisticated memory models. This limits the amount of communication bandwidth between the two processors. Hence we must minimize the information that must be exchanged between the two processors to reduce the overhead of switching between high-performance mode and low power mode.

The two processors also do not share their memory hierarchy, so any state changes that occur on one processor must be communicated to the other one via the low bandwidth I/O interface. We can address this challenge because the low-power processor can only handle a limited number of tasks; hence, only a limited amount of state needs to be communicated between the two processors through the I/O interface.

3.2. Software Considerations

When we developed our prototype, we could not find two processors with the same core unified instruction set architecture (ISA) that provided the specifications required to operate in high-performance mode and low-power mode. Hence, our implementation is a hybrid between an ARM ISA and an MSP430 ISA. This presents challenges when attempting to provide a unified software view of the system during development.

We addressed this challenge by using an abstract representation of the interface state (detailed below) so that the same state representation could be used on both architectures. However, in future work we plan to limit ourselves to two processors that share the same core ISA, as lower power processors become commercially available.

4. IMPLEMENTATION

Based on the considerations outlined in the previous section, we developed a prototype Ebook reader using our own custom hardware board, along with the necessary software infrastructure to demonstrate our ideas.

4.1. Hardware Architecture

We built a prototype system combining an Overo Computer (on Modules based on the TI OMAP 3530) with a commonly used low-power micro-controller (the TI MSP 430). The diagram of the system is shown in Figure 1. In our somewhat extreme setting, the OMAP 3530 application processor is running at 600MHz with 256MB of RAM and consumes up to 1W while running a Linux workload. The MSP 430 is running at 16MHz using 16KB of RAM, 256KB of Flash memory, and consumes about 10 mW when fully active. Both systems share a common bus to the BroadSheet E-Ink display controller, which contains its own memory to store graphic data shared by both the OMAP and the MSP. To track pen input, our system uses a Wacom digitizer, placed directly below the E-ink screen. The Wacom subsystem is connected directly to the MSP. This layout assures that the MSP can process inputs even when the OMAP is asleep. When the OMAP is active, the MSP simply forwards data to the OMAP. Interprocessor communication is handled through an SPI link controlled by the OMAP and two interrupt lines from the MSP to the OMAP. Raising the first interrupt line causes the OMAP to wake up from sleep. Raising the second interrupt line causes the OMAP to initiate a transaction on the SPI bus. The corresponding board implementation is shown in Figure 3. Importantly, this configuration was selected because of its was simple to build, but other configurations are possible (see Section 6).

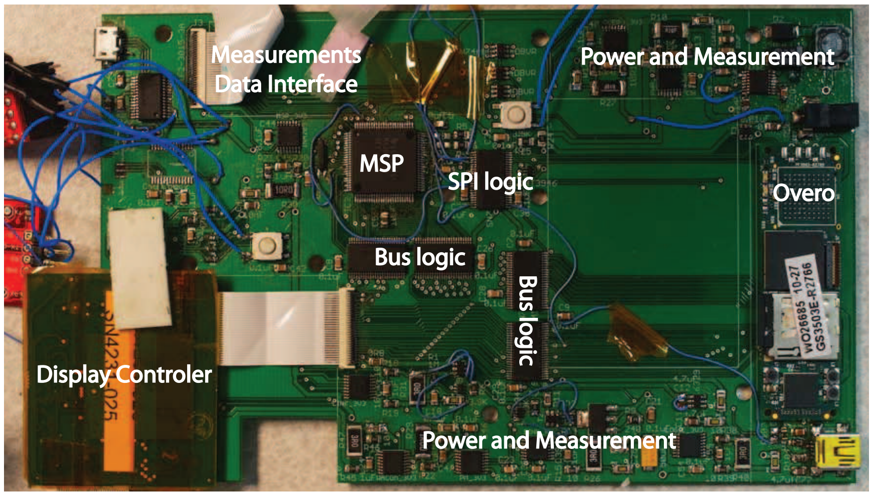


Fig. 3. Our implementation showing key elements of our system. The Overo is a computer module based on the OMAP processor.

4.2. Software Architecture

Our general approach to designing the software architecture was to consider the MSP as an interface co-processor capable of executing a simplified version of the full interface. Like the standard version, the simplified version is composed of a windows tree. However, instead of describing how to render the interface (as is appropriate for the OMAP), each element of the tree includes a pointer to the display controller memory location that stores the pre-rendered version of the interface. The tree also includes active areas associated with a script to be triggered by user interactions (e.g., pen down, pen up) in that area. In our current implementation the scripts are described as three-address code with simple flow control relying heavily on high level functions (e.g., bringing a window to the top, changing the background of a window, inking, text input). This format is very compact and can be easily executed on the MSP. Yet, as

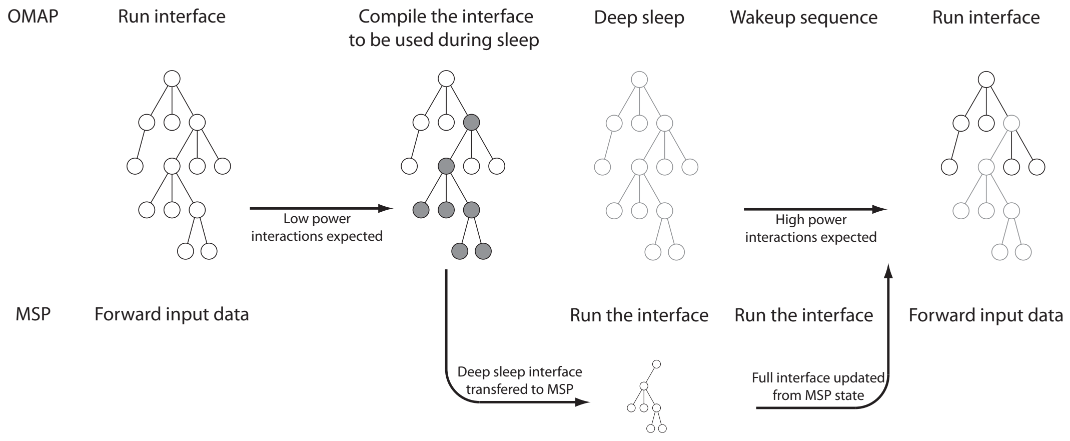

OMAP
Run interface
Compile the interface to be used during sleep
Deep sleep
Wakeup sequence
Run interface
Low power interactions expected
High power interactions expected
MSP
Forward input data
Run the interface
Run the interface
Forward input data
Deep sleep interface transfered to MSP
Full interface updated from MSP state

Fig. 4. Software architecture. When low power interactions are expected, the OMAP transfers the sub-tree corresponding to the continuation of the interface to the MSP. The MSP executes the tree until the OMAP is required. At this point, the updated tree is transferred back to the OMAP.

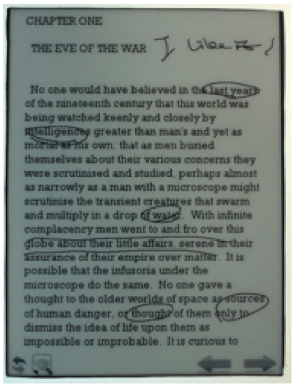
```
{
  name = "btn_search";
  parent = "toolbar";
  size_x = 50;
  size_y = 750;
  size_w = 50;
  size_h = 50;
  svisible = 1;
  on_up = "
    push 'search_dialogue'
    funccall 'tree_node_show'
    push 'keyboard'
    funccall 'tree_node_show'
    funccall 'tree_refresh'";
}
```

```
{
  name = "btn_next";
  parent = "toolbar";
  size_x = 500;
  size_y = 750;
  size_w = 100;
  size_h = 50;
  svisible = 1;
  on_up = "
    load r0, 'cur_doc'
    load r1, 'cur_page'
    addi r1, 1
    i2s s3, cur_doc
    i2s s4, cur_page
    assign s1, 'page'
    concat s1, s3
    concat s1, s4
    concat s1, 'addr'
    load r5, s1
    compi r5, 0
    jne L2
    jmp L3
L2:
    push r5
    funccall
      'show_page'
L3:
    funccall
      'sync_overo'
    funccall
      'tree_node_show'";
}
```

Fig. 5. Script examples. (a) the description of the search button; and (b) the script for the turn page button.

demonstrated by programs like Hypercard, combining visual and active regions can be a powerful way to simulate an interface. The MSP is therefore never responsible for rendering an interface; instead, it only needs to: (i) display pre-rendered objects already stored in the display controller memory, and (ii) handle user input.

In a typical run of our Ebook application, the OMAP will first load a description of the interface and build the corresponding tree in memory, caching pre-computed visual information in the display controller as it does so. When the initialization is done, the OMAP transmits the tree to the MSP, which starts executing the interface. The OMAP can now safely go to sleep. The MSP will execute the interface (including turning pages, inking, and entering text in a search dialog box) until it runs out of pages to display, it runs out of stroke data memory, or a search is initiated. In those cases, it wakes up the OMAP, transmits the current version of the interface, and forwards all Wacom events to the OMAP until the OMAP decides to fall asleep again. This process is illustrated in Figure 4. Note that in some cases it is possible to wake up the OMAP preemptively to mask latency. For example, the OMAP might be activated as soon as a character is entered in the text box of the search dialog box since a processing intensive search is anticipated.



CHAPTER ONE

THE EVE OF THE WAR I like it!

No one would have believed in the last years of the nineteenth century that this world was being watched keenly and closely by intelligences greater than man's and yet as mortal as his own; that as men busied themselves about their various concerns they were scrutinised and studied, perhaps almost as narrowly as a man with a microscope might scrutinise the transient creatures that swarm and multiply in a drop of water. With infinite complacency men went to and fro over this globe about their little affairs, serene in their assurance of their empire over matter. It is possible that the infusoria under the microscope do the same. No one gave a thought to the older worlds of space as sources of human danger, or thought of them only to dismiss the idea of life upon them as impossible or improbable. It is curious to

Fig. 6. Screen shot of our software implementation on our custom Ebook reader.

We show samples of scripts in Figure 5. In Figure 5a, we show the description of the "Search" button. Upon clicking on the button, the corresponding dialog box is raised, so that the text of the search can be entered. In Figure 5b, the script associated with the active area over the "Next" button simply updates the current background of the background window. This can be done easily by sending a small command to the display controller initiating a copy from the cache to the frame-buffer. Note that when there is a cache miss, the system simply wakes up the OMAP which then takes control of the application.

Using this approach, we implemented a simple Ebook reader and a calculator. Using the Ebook reader, one can flip through a book and annotate its pages (Figure 6). It is also possible to search a word in the book using a simple search box and a on-screen keyboard Annotations and page turning can be handled by the MSP while the search is handled by the OMAP. The calculator lets user perform simple calculations and can run on the MSP without waking the OMAP.

5. EVALUATION

To quantify the potential of the proposed approach we observe its impact on the overall energy use of an information appliance. Specifically, we consider the following sequence of active reading interactions on an Ebook reader: After scanning a couple of pages, the user finds a passage of interest, highlights it, and continues reading. For the same scenario, we measure the energy consumed by a number of different configurations of our platform. By computing the ratio of energy consumed in one configuration to another, we can determine how various configurations impact the battery lifetime of the system. A concrete illustration of this procedure is shown in Figure 7. The three highlighted sections of the figure correspond to when the user is turning pages ("Page Turning $\times$ 3"), writing on the Ebook to highlight a passage ("Writing"), and where the system caches new pages in display memory in anticipation of another page turn.

Figure 7 shows the energy profile of this series of interactions for an OMAP-only system running the application with 3s timeout (OMAP PM), and for an OMAP augmented with an MSP processor. As shown in Figure 7, in the OMAP PM configuration, the OMAP is active 53.9% of the time, driving the overall energy consumption of the system to 35.1J. In contrast, in the MSP-augmented configuration, the OMAP is active 3.7% of the time, resulting in an overall energy consumption of the system of 13.3J—translating to an improvement in battery life by a factor of $35.1/13.3 = 2.64$.

The overarching goal of the evaluation was to compare the total energy signature of the proposed dual-processor approach (OMAP+MSP) against an OMAP based implementation with aggressive power management (OMAP PM). Both of these were contrasted with a standard OMAP configuration. This illustrates the differences in both energy consumption and responsiveness across the two energy-saving configurations.

5.1. Hardware and Software Settings

The hardware platform shown in Figure 3 was used to implement all configurations. The Wacom digitizer was set to the standard configuration with respect to power saving. The Broadsheet controller was set to low power state when not in use (using the corresponding software interface with a 1.5s timeout). The MSP was programmed to sleep when not processing events. In all configuration the OMAP was running the Power Management (PM) branch of the 2.6.35 Linux kernel.

OMAP configuration. In this configuration, the role of the MSP was limited to forwarding data received from the Wacom. The OMAP was running the complete interface, receiving events from the MSP and dispatching them through the interface tree. On each node, event handlers were implemented using the scripting language

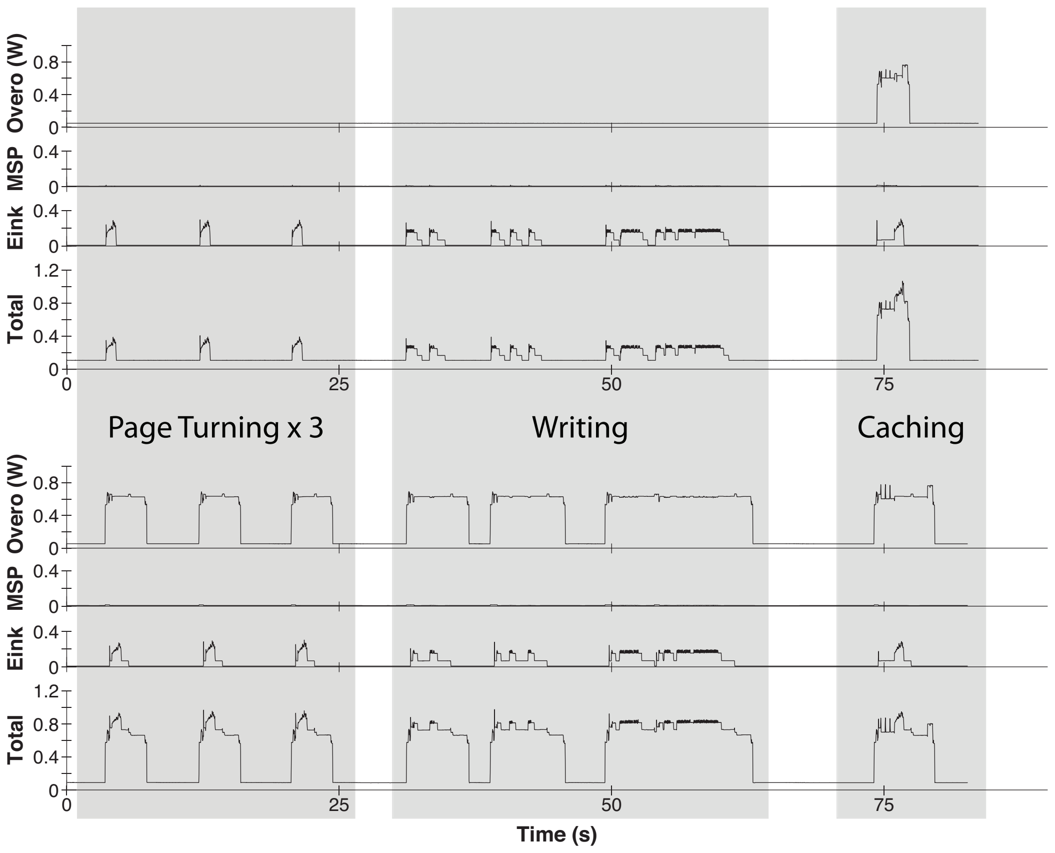


Fig. 7. Example of system performance. Top: OMAP+MSP; Bottom OMAP PM. During writing tasks, the timeout keeps the OMAP on during most of the tasks in the OMAP PM configuration. The y-axis is a measure of the instantaneous power for different parts of the system.

described above. In this configuration, we enabled all power management features that were built into the operating system, including sleep modes and dynamic voltage and frequency scaling.

OMAP with power management (OMAP PM) configuration. This configuration was identical to the OMAP configuration, but upon a period of 3 seconds of inactivity, the application directed the operating system to put the OMAP to sleep, waiting to be awakened by the MSP upon pen input (pen down event). This timeout value was selected to ensure interface reactivity during writing. In this configuration, the MSP cached the data received from the Wacom during the wakeup transition. This approach guarantees that no information from the pen input is lost during the transition. We gave priority to power efficiency over latency. For example we woke up the OMAP upon pen down to avoid having the OMAP running when the pen was in the hover state. The system started with the OMAP sleeping. This configuration is more aggressive (i.e. lower power) than a platform where the application directs the OMAP to sleep without the support of the MSP (since I/O information would be lost due to wake-up latency—an unacceptable outcome) or one where the OMAP was in a low power active mode without an MSP (since the OMAP low power active mode uses significantly more power than the OMAP in sleep mode augmented with an MSP in active mode).

OMAP+MSP configuration. In this configuration, the system was designed to execute as much of the interface on the MSP as possible. Upon expecting low power interactions (such as those involved in our active reading tasks), the OMAP transferred control to the MSP (see Figure 4) and went to sleep. From then on, the MSP executed the interface and only transferred control to the OMAP when high computing resources were necessary (e.g., when loading the cache). Any cached pen data were transferred to the OMAP before the end of a given task. This last step was implemented for testing purposes to ensure that all of the tasks started and ended in a state in which the OMAP had access to an updated version of the interface. As in the OMAP PM configuration, the system started with the OMAP sleeping.

5.2. Tasks

The tasks were implemented with the goal of covering a representative set of interactions typical for active reading. During active reading, users alternate among reading itself, annotating the document, navigating back and forth inside the document, and consulting alternative sources of information (e.g., a dictionary or references made inside the document) [Sellen and Harper 2001]. From this set, we picked three tasks: reading, writing notes, and annotating, as they represent interesting behavior from an energy point of view.

Reading task. Users were asked to read a 5-page segment from War of the Worlds by H. G. Wells. This task demonstrated the power signature of the system in a very low power mode setting. The system was set such that no cache was missed. Note that the energy needed for serving a miss will be the same in all conditions. Thus, it is easy to derive the energy signature with a miss from our data.

Writing task. Users were asked to transcribe a paragraph of text onto the screen. This task illustrated the power signature of our system in a task requiring constant low level user feedback.

Annotation/Marking task. Users were asked to scan a document and highlight non-auxiliary verbs. This task was included to demonstrate the potential impact of having the OMAP alternating between run and sleep mode. Many verbs were far enough apart to put the OMAP to sleep in the OMAP PM configuration. Thus, we could observe user perceptions of wake-up delay.

5.3. Measures

Like Raghunathan et al. [Raghunathan et al. 2004] we monitored the power consumption of our system with shunt resistors placed on the power supply lines of each of the sub-systems. The measurements were performed by TI INA209 chips, connected to a custom made capturing board via I2C buses. This setting was selected to optimize measurements in realistic situations. Each measuring chip reports the average of 32 measurements every 17ms (about 58Hz). Our system reports data from the MSP, the Overo board (including the OMAP, the OMAP companion power chip, and the memory present on the board), the Wacom, the logic, and the screen (including display controller and the 7.68V high voltage supply). Total task duration was recorded as well.

5.4. Protocols and Participants

Since participants interaction style might have an impact on power consumption, we opted for a within-subject design in which each participant tested each condition in turn. To limit learning effects, we varied the presentation order of conditions according to a latin square. Further, we used 3 sets of documents and alternated the mapping between documents and condition.

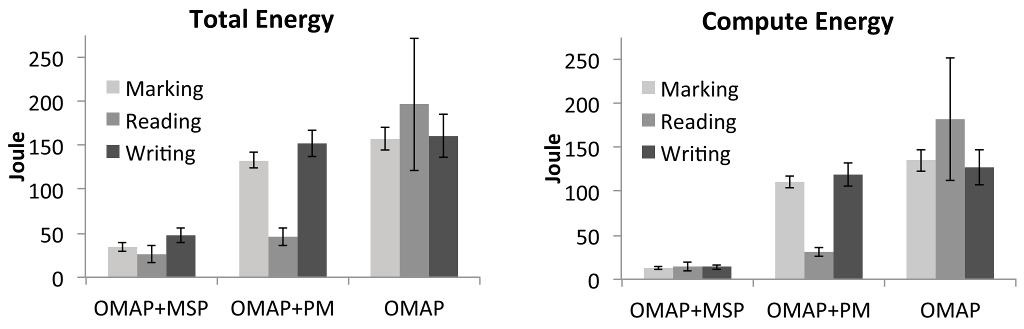


Fig. 8. Measured energy consumption results. Error bars represent 95% confidence intervals.

Before each task, we explained the requirements to participants and asked them to demonstrate the use of the interface controls needed to complete the task. Then the participant executed the task in the three conditions in turn. After each condition, we asked them if they had any comments about that condition regarding speed latency or stroke quality. After each task, participants completed a questionnaire asking them to rate each of the conditions for that task. The total protocol lasted about an hour. Six participants were recruited from the local campus and received $10 for their participation.

5.5. Results

In the following results, the Greenhous-Geiser correction was used when sphericity could not be assumed and Bonferroni correction was used for pairwise comparisons.

Due to limitations in our measurement setup when conducting the user study, our raw data measured from the platform did not include the energy consumed to transmit the interface tree to the MSP (and update it at the end of the task). The energy used for this process (a fixed overhead in all OMAP+MSP experiments) was 0.412J, and this correction was added to all OMAP+MSP raw experimental measures before performing the analyses below. This energy value represents the energy needed for transmitting the tree, having the OMAP go to sleep, waking up the OMAP, and transmitting the update back to the OMAP. All energy consumption results are shown in Figure 8.

To study the total energy consumption in the different conditions we performed a 2 way (condition $\times$ task) repeated measures analysis of variance (ANOVA), a standard analysis tool to compare more than two matched samples. We found a main effect of condition (F(1.08, 5.38) = 76.6, p $<$ .001, η_p^2 = .939[1]) as well as a weak effect of task (F(2, 10) = 3.65, p = .065, η_p^2 = .422). These results were qualified by an interaction (F(1.19, 5.96) = 12.2, p = .011, η_p^2 = .709). As a result, main effects are to be interpreted with caution, so we proceed by a task by task analysis.

Reading task. A typical power consumption pattern of this task is shown in Figure 9. The pattern is quasi periodic, as a user takes about the same amount of time to read one page of text. The spikes correspond to when the user decides to turn the page and a new page is rendered. The OMAP+MSP configuration can keep the Overo asleep, as

[1]η_p^2 (partial η^2) is a measure of power for ANOVA. Cohen [Cohen 1992] suggests .10, .30, .50 as level for small, medium and large effect respectively. $\eta^2 > 0.5$ indicates a statistically significant effect even for small sample sizes.

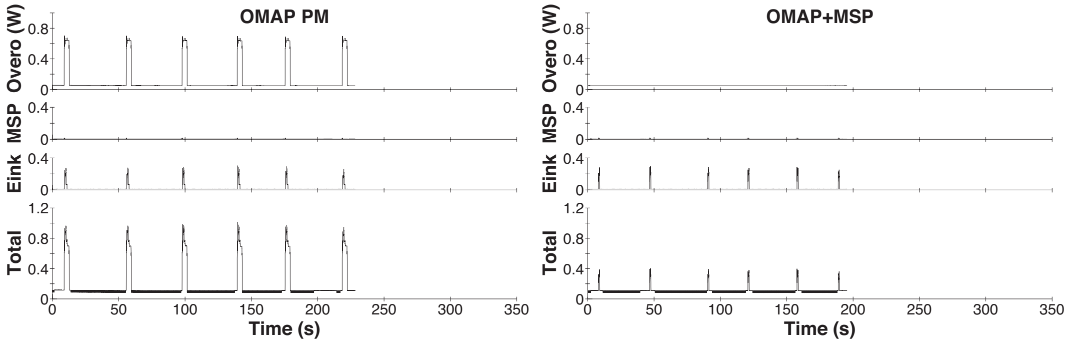


Fig. 9. Measured energy during reading task. The left hand panel shows the OMAP PM configuration, and the right hand panel shows the OMAP+MSP configuration. The noise is created by the Wacom system switching to a low sampling rate.

the MSP can handle all the activity needed for this task. A one-way repeated measures ANOVA reveals an effect of condition on total energy consumption (F(1.00, 5.02) = 23.2, p = .005, η_p^2 = .823). A pair wise comparison reveals that the OMAP+MSP condition has the lowest energy consumption (M = 26.4, SE = 4.92) followed by the OMAP PM condition (M = 45.5, SE = 5.14), followed by the OMAP condition (M = 197, SD = 38.4, all differences significant $p < .02$). For this task OMAP+MSP represents a 42% saving (1.72$\times$ in battery time) over the more traditional OMAP PM. The pattern is even more accentuated if one focuses on the energy consumed by the processors. Here the OMAP+MSP (M = 14.4, SD = 2.63) achieves a 54% saving over OMAP PM (M = 31.2, SD = 2.74). Note that the confidence interval for this task appears large in the OMAP condition due to P6 being a slow reader. The pattern of results is identical with P6 excluded.

Writing task. A typical power consumption pattern of this task is shown in Figure 10. Unlike the OMAP PM configuration where the Overo is mostly active, the OMAP+MSP configuration keeps the Overo in sleep mode while the MSP handles all user interactions (seen in the EInk power consumption signature). For this task, a one-way repeated measures ANOVA reveals an effect of condition on total energy consumption (F(2, 10) = 82.3, $p < .001$, η_p^2= .959). A pair wise comparison reveals that the OMAP+MSP condition has the lowest energy consumption (M = 47.5, SE= 4.09),

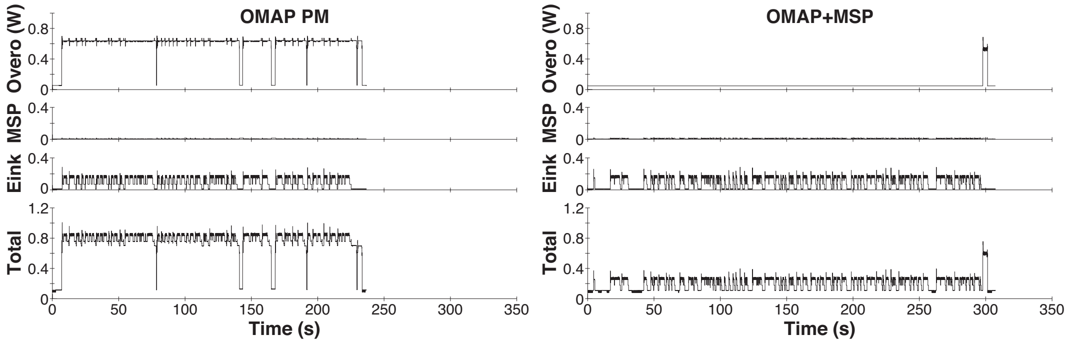


Fig. 10. Measured energy during writing task. The left hand panel shows the OMAP PM configuration, and the right hand panel shows the OMAP+MSP configuration. In the OMAP PM configuration, the timeout required to keep the system responsive to user input forces the OMAP to be on almost all the time.

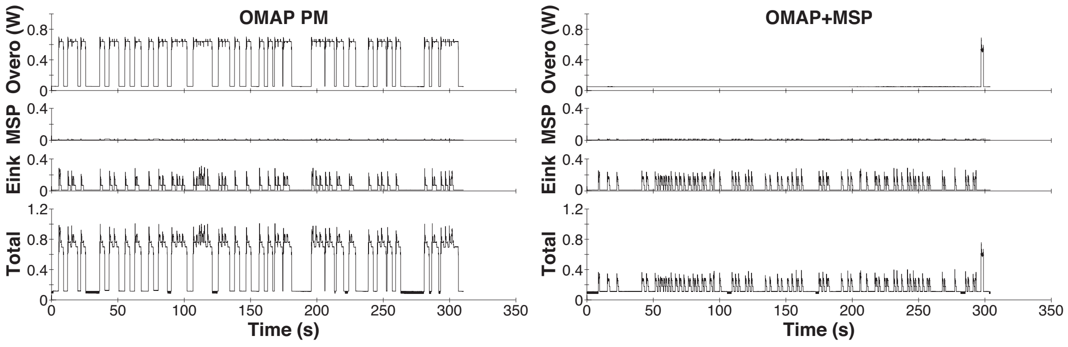


Fig. 11. Measured energy during annotation task. The left hand panel shows the OMAP PM configuration, and the right hand panel shows the OMAP+MSP configuration. In the OMAP PM configuration, the OMAP can be put into sleep mode but only at a cost in latency and system responsiveness.

followed by OMAP PM condition (M = 152; SE = 7.47), followed by the OMAP condition (M = 161, SE = 12.5), although only the OMAP+MSP is significantly lower than the other two ($p < .001$). This can be explained by the fact that the timeout was maintaining the OMAP active in the OMAP PM condition. For this task OMAP+MSP represent a 69% saving (3.23$\times$ in battery time) over the more traditional OMAP PM. The pattern is even more accentuated if one focuses on the energy consumed by the processors. In that case, the OMAP+MSP (M = 14.0, SE = 1.35) achieves a 88% saving over OMAP PM (M = 119, SE = 6.52).

Annotation task. A typical power consumption pattern of this task is shown in Figure 11. This task differs from writing in that the user has intermittent activity (seen by the power signature of the Overo in the OMAP PM configuration), and hence the Overo has to switch from active to idle mode repeatedly in the OMAP PM configuration. For this task, a one-way repeated measure ANOVA reveals an effect of condition on total energy consumption (F(2, 10) = 323, $p < .001$, $\eta_p^2 = .985$). A pair wise comparison reveals that the OMAP+MSP condition has the lowest energy consumption (M = 34.0, SE = 2.57), followed by OMAP PM condition (M = 133, SE = 4.42), followed by the OMAP condition (M = 157, SE = 6.43, all differences significant $p < .026$). For this task, OMAP+MSP represent a 74% saving (3.85$\times$ in battery time) over the more traditional OMAP PM. The pattern is even more accentuated if one focuses on the energy consumed by the processors. In that case, the OMAP+MSP (M = 13.0, SE = 1.12) achieve a 88% saving over OMAP PM (M = 110, SE = 3.40).

In this condition a one-way ANOVA showed an effect of condition on both perceived speed and reactiveness (F(2, 10) = 6.38, $p < .016$, $\eta_p^2 = .561$ and F(2, 10) = 7.00, $p < .013$, $\eta_p^2 = .583$ respectively). Looking at pairwise differences, only the reactiveness rating of the OMAP+MSP (M = 4.3, SE = 2.11) was significantly higher than the OMAP PM condition (M = 2.83, SE = .167). This was expected, given the time it takes for the OMAP to wake up.

6. DISCUSSION

Our results clearly illustrate the potential of the proposed system architecture. However, the resulting energy savings are very sensitive to specific design parameters. We now discuss some of the most relevant design considerations.

Cache management. In the present implementation, the graphical data are cached in the memory of the display controller, while the inking data are stored in the memory of the MSP. Thus, the base cost of cache management (i.e., loading new pages in the display memory or flushing strokes from the MSP to the OMAP) is very high because a full wake-up cycle is about 500ms using the current version of our kernel. Thus it would be beneficial to use large caches that are updated as seldom as possible. Of course, bigger cache memory would require more power to maintain, but the use of static RAM would mean that we could significantly improve the performance of our stroke caching.

The issue of cache size also applies to other areas. For example it is important for the MSP to store as much stroke information as possible to delay the full processing of the stroke until the next time the OMAP needs to be awakened. Similarly, during searches, it would be more energy efficient to transfer a list of possible targets to the MSP than to transmit one target at a time.

Finally during our implementation, it became evident that the RAM of the MSP posed a limitation for large interfaces. In part, this is due to storing the tree in RAM instead of Flash. This could be easily addressed, but a larger RAM will still allow us to manage more script variables.

Board design. The effective benefit of the proposed approach is highly dependent on the baseline consumption of the system including the power draw by the OMAP module while asleep, the draw by the MSP while asleep, the draw by the Wacom subsystem, and parasite draw by the rest of the board. Using Figure 7 as an example, we consider a case in which we improve our software and hardware such that the sleep power draw by the OMAP is reduced from 50mW to 10mW. In that case, the total energy consumed by the OMAP and MSP combined would be reduced from 6.1J to 3.1J and the overall performance of our approach would increase from a five-fold energy reduction (83% reduction) to a 10 fold energy reduction (90% reduction). Such improvements are difficult to achieve using off-the-shelf components because they are not always optimized for power consumption. However, we believe that they are well within reach for a fully custom built system.

Another substantial power draw stems from the Wacom system. We believe that this can be alleviated by including supplemental sensors to infer the current context of use [Hinckley et al. 2005]. For example, a capacitive strip around the display can be used to detect hand proximity and switch on the Wacom. Also, light sensors can detect situations in which the ambient light level is too low for reading and power down the system in response. In combination, such techniques may achieve further power reduction.

Performance could also be optimized using a slightly different architecture to simplify data exchange between the OMAP and MSP. When developing our prototype, we observed that the SPI link between the OMAP and the MSP was a limiting factor because of the inability of the MSP to handle high bandwidth transfers. As a result, the MSP had to drop Wacom data samples from time to time. Part of the problem in our prototype was due to software restrictions (in particular, we did not use DMA). The best solution would be for both processors to share some amount of memory to simplify the transfer of both the interface description and stroke data. It will also be interesting to consider a design in which both processors share the same instruction set architecture or are even on the same die. In fact such a designs are already available or have been announced recently: for example the OMAP 4 includes 2 ARM cortex-A9 cores and two ARM Cortex M3 cores. The next generation of the OMAP processor (OMAP 5) will include two high performance ARM cortes-A15 MPCore and two low power cores (ARM Cortex-M4).

Finally it is important to consider possible impact on cost. An alternate way to extend system lifetime would be to increase the size of the battery. If we take a reasonably sized baseline battery to be the one used by a commercial device such as the Amazon Kindle, it has a current list price of $20. The list price for an MSP430 is under $4. Since the introduction of an MSP430 extends the battery life by over a factor of three, adding the MSP430 (˜$4) is cost-effective compared to tripling the size of the battery (˜$60). The benefit might even be greater if one consider a design in which the low power-core is included in the high power-core die as described above.

Display characteristics. At the time we developed this system, the E-ink system was the only bi-stable display readily available. The E-ink technology is a truly bi-stable system cable of maintaining an image at zero power. However, its refresh speed is quite slow (≈0.5Hz). Using the equivalent of pipelining, the Broadsheet controller is able to maintain responsive ink, but it is difficult to update large areas of the screen interactively. Overall, we believe that these display characteristics had a relatively minimal impact on our design since fast-paced interactions will have to be driven by the OMAP anyhow. Thus, we believe that our approach will work equally well with the new generation of interactive bi-stable displays such as Mirasol [Mirasol 2010] and AquaVista [Liquavista 2008]. Nevertheless the inherent latency of the E-ink display makes it very difficult to evaluate the impact of potential delays introduced by the MSP, so more empirical studies will be needed.

Connected devices. So far, we have only considered the case of an unconnected device. Many of the slate devices on the market have network connectivity. Following the work done by the sensor network community [Dutta and Culler 2005; Hohlt et al. 2004], we envision a system relying on two radios. One will be a very low power, low bandwidth system making it possible to transmit small amounts of information and implement push strategies. The other one will be a high bandwidth, high power radio to be powered only when the corresponding bandwidth is required. This is analogous to our dual processor approach, except for communication rather than computation. Offloading network processing is also a possibility, as has been previously shown by other work [Agarwal et al. 2009].

Toward complex interfaces. The interface implemented in the context of the present prototype is not a full-fledged application on the scale of a commercial product. It is therefore important to note that more complex interfaces would only require adapting the user interface markup language environment for the target architecture. This modification would include: 1) generating renderings of interface elements to be transferred to the display controller; 2) generating the interaction tree; 3) generating the scripts to be run by the MSP; 4) creating a new tag to label sections of the interface safe to run on the MSP. For the third step, one could either use a simple bytecodes implementation (as in the present case), or generate native MSP code to improve performance.

Pre-rendered pages stored in display memory may require more sophisticated caching techniques if content on those pages are dynamically generated. This would require cache invalidation approaches, where the system would have to detect when a previously pre-rendered page is to be invalidated. Alternatively, some situations may warrant an implementation where multiple versions of a page are pre-rendered, with run-time information used to select the appropriate version of the page. More empirical studies are needed to determine how effective our proposed architecture would be in the context of dynamic content.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a new system architecture for information appliances which uses a small microcontroller to offload simple interactive computations from the main application processor. We demonstrated that implementing our system requires only limited changes to the way applications are programmed. Preliminary measurements suggest that the proposed architecture can reduce the overall energy consumption of an information appliance by up to 69% (3.23× in battery time) and the compute energy by up to 88%. These savings are highly dependent on the standby power of the system and could be further improved with a more advanced hardware design. We believe that when combined with the new generation of video-capable bi-stable displays, our system will offer an efficient platform for a wide range of activities from reading to video watching.

Our system has the potential to be efficient over a large spectrum of energy signatures from simple page turning to video playing. Unfortunately, since our current prototype was based on E-ink a grayscale display with low refresh rate we were not able to explore the full potential of the proposed architecture. Accordingly, we plan to test the performance of our system using one of the high-speed displays currently under development (e.g., Mirasol). A high-speed prototype would also allow us to explore how more interactive techniques (e.g., marking menu, page-flipping applications) could be implemented with our architecture. In the long run, we plan to gather more realistic interaction data in the wild to quantify actual energy savings resulting from the proposed architecture.

REFERENCES

AGARWAL, Y., HODGES, S., CHANDRA, R., SCOTT, J., BAHL, P., AND GUPTA, R. 2009. Somniloquy: Augmenting network interfaces to reduce pc energy usage. In *USENIX Symposium on Networked Systems Design and Implementation*.

AMAZON. 2007. *Amazon Kindle, http://www.amazon.com/kindle/*.

BIERMANN, D., SIRER, E. G., AND MANOHAR, R. 2004. A rate matching-based approach to dynamic voltage scaling. In *Proc. First Watson Conference on the Interaction between Architecture, Circuits, and Compilers*.

BURD, T. D., PERING, T. A., STRATAKOS, A. J., AND BRODERSEN, R. 2000. Dynamic voltage scaled microprocessor system. *IEEE Journal of Solid-State Circuits 35*, 1571–1580.

CARD, S. K., MORAN, T. P., AND NEWELL, A. 1983. *The psychology of human-computer interaction*. Erlbaum Associates.

CHEN, G., KANG, B.-T., KANDEMIR, M., VIJAYKRISHNAN, N., IRWIN, M., AND CHANDRAMOULI, R. 2004. Studying energy trade offs in offloading computation/compilation in java-enabled mobile devices. *IEEE Transaction on Parallel and Distributed systems 15,* 9, 795 – 809.

COHEN, J. 1992. A power primer. *Psychological Bulletin 112,* 1, 155–159.

DUTTA, P. K. AND CULLER, D. E. 2005. System software techniques for low-power operation in wireless sensor networks. 925 – 932.

E-INK. 2006. http://www.eink.com/.

FITTS, P. M. 1954. The infomation capacity of the human motor system in controlling amplitude of movement. *Journal of Experimental Psychology 47*, 381 – 391.

GOVIL, K., CHAN, E., AND WASSERMANN, H. 1995. Comparing algorithms for dynamic speed-setting of a low-power cpu. In *Proceedings of the 1st Conference on Mobile Computing and Networking*.

GRUNWAND, D., LEVIS, P., FARKAS, K., III, C. B. M., AND NEUFELD, M. 2000. Policies for dynamic clock scheduling. In *Proc. Operating Systems Design and Implementation*.

HARTER, T., VROEGINDEWEIJ, S., GEELHOED, E., MANAHAN, M., AND RANGANATHAN, P. 2004. Energy-aware user interfaces: an evaluation of user acceptance. ACM Press New York, NY, USA, 199 – 206.

HINCKLEY, K., PIERCE, J., HORVITZ, E., AND SINCLAIR, M. 2005. Foreground and background interaction with sensor-enhanced mobile devices. *ACM Trans. Comput.-Hum. Interact. 12,* 1, 31–52.

HOHLT, B., DOHERTY, L., AND BREWER, E. 2004. Flexible power scheduling for sensor networks. 205 – 214.

IREX TECHNOLOGIES. 2006. *iLiad User Manual (V2.7)*. Irex Technologies.

ISHIHARA, T. AND YASUURA, H. 1998. Voltage scheduling problem for dynamically variable voltage processors. In *International Symposium on Low Power Electronics and Design*.

KUMAR, R., TULLSEN, D. M., RANGANATHAN, P., JOUPPI, N. P., , AND FARKAS, K. I. 2004. Single-isa heterogeneous multi-core architectures for multithreaded workload performance. In *International Symposium on Computer Architecture*. 64–75.

LI, Z., WANG, C., AND XU, R. 2001. Computation offloading to save energy on handheld devices: a partition scheme.

LIQUAVISTA. 2008. http://www.liquavista.com.

MACKINLAY, J. D., CARD, S. K., AND ROBERTSON, G. G. 1990. Rapid controlled movement through a virtual 3d workspace. ACM Press New York, NY, USA, 171 – 176.

MIRASOL. 2010. www.mirasoldisplays.com.

NEMOPTIC. 2006. http://www.nemoptic.com/.

PERING, T. AND BRODERSEN, R. 1998. Energy efficient voltage scheduling for real-time operating systems. In *4th IEEE Real-Time Technology and Applications Symposium, Work in Progress Session*.

PRIYANTHA, B., LYMBEROPOULOS, D., AND LIU, J. 2010. Littlerock: Enabing energy efficient continuous sensing on moble phones. Tech. rep., Microsoft Research.

RAGHUNATHAN, V., PERING, T., WANT, R., NGUYEN, A., AND JENSEN, P. 2004. Experience with a low power wireless mobile computing platform. 363 – 368.

ROBERTSON, G. G., CARD, S. K., AND MACKINLAY, J. D. 1989. *The cognitive coprocessor architecture for interactive user interfaces*. ACM, 10–18.

RUDENKO, A., REIHER, P., POPEK, G. J., AND KUENNING, G. H. 1998. Saving portable computer battery power through remote process execution. *ACM SIGMOBILE Mobile Computing and Communications Review 2,* 1, 19 – 26.

SAHA, S. 2011. An experimental evaluation of real-time dvfs scheduling algorithms. M.S. thesis, Virginia Polytechnique Institute and State University.

SELLEN, A. J. AND HARPER, R. H. R. 2001. *The Myth of the Paperless Office* 1st Ed. MIT press.

SIMUNIC, T., BENINI, L., ACQUAVIVA, A., GLYNN, P., AND MICHELI, G. D. 2001. Dynamic voltage scaling and power management for portable systems. In *Proc. 38th Design Automation Conference*.

SONY. 2006. *Operation Guide, PRS-500, portable Reader System*. Sony Inc.

TEXAS INSTRUMENTS. 2013a. http://processors.wiki.ti.com/index.php/omap3530_power_estimation_spread sheet.

TEXAS INSTRUMENTS. 2013b. http://www.ti.com/general/docs/lit/getliterature.tsp?literaturenumber= sprab98&filetype=zip.

VALLERIO, K. S., ZHONG, L., AND JHA, N. K. 2006. Energy-efficient graphical user interface design. *IEEE Transaction on Mobile Computing 5,* 7, 846–859.

WANG, C. AND LI, Z. 2004. Parametric analysis for adaptive computation offloading.

WEISER, M. 1991. The computer for the 21st century. *Scientific American (International Edition) 265,* 3, 66–75.

WESTE, N. AND HARRIS, D. 2010. *CMOS VLSI Design: A Circuits and Systems Perspective*.

ZHONG, L. 2005. Energy-efficient mobile system design: The user's perspective. M.S. thesis, Princeton University.

ZHONG, L. AND JHA, N. K. 2005. Energy efficiency of handheld computer interfaces: Limits, characterization, and practice. 247 – 260.