CHARACTERIZING AND IMPROVING ENERGY-DELAY TRADEOFFS IN HETEROGENEOUS COMMUNICATION SYSTEMS*

Nan Jiang, Jayaprakash Pisharath, Alok Choudhary

Department of Electrical & Computer Engineering
Northwestern University
Evanston IL - 60208 USA
{jiangjf, jay, choudhar}@ece.northwestern.edu

ABSTRACT

Communication systems with multiple computing resources are gaining popularity. These devices consume a considerable amount of energy, and require that the system resources be fully utilized at all times. In this paper, we aim to reduce the energy consumption and also improve the system utilization by incorporating performance-enhancement and power-optimization techniques into a multi-resource heterogeneous communication system. We study the performance and energy improvements contributed by our techniques, using a traditional evaluation framework. However, this framework fails to clearly model the performance-energy tradeoffs in the system. Hence, we propose a more relevant framework with a new metric named *Energy-resource efficiency (ERE)* to study these systems. ERE defines a link between the performance and energy variations in a system to clearly highlight the various performance-energy tradeoffs. Our experimental results show that, when ERE is used to calibrate the performance-energy tradeoffs, one can achieve up to 80% performance and energy gains in a power-aware heterogeneous communication system.

1. INTRODUCTION

The advancement in technology scaling has enabled designers to build complex systems on a single chip. But the growing complexity of today's chips are causing designers to think about smaller, more partitioned designs, and this is one driver of simpler embedded computing systems. Embedded devices like set-top boxes and mobile phones are a result of the merger of technologies like broadband communications or 3G wireless networks with interactive multimedia. Such embedded devices support numerous functions like multimedia (MP3, MPEG2 media playback), wireless communication (GSM, digital radio, Bluetooth) and some mandatory functions including user interfaces and file management, to be carried out at the same time. Even further, newer standards such as MPEG4, WMV and JVT (H.26L) require these devices to have a high performance in order to handle the data flow in real time. As a result, system designers are opting to embed more than one processor or processor core into these devices in order to satisfy both multitasking and performance needs [4][9]. The more the number of processors in a device, the higher is the power dissipation of that device. Techniques to curtail the energy consumption of such heterogeneous embedded systems are already in place [2][5][11]. Researchers have also proposed mechanisms to study performance-energy tradeoffs for various systems [3][12][10]. In reality, achieving a good tradeoff for heterogeneous and parallel systems is a tedious task since it requires a thorough analysis of the system, which might be time consuming.

In this paper, we first design an application-driven scheme that aims to improve the performance and concurrently reduce the energy consumption of a communication system. The scheme targets a multi-resource heterogeneous communication system consisting of low-power embedded processors that serve as computing resources and a general-purpose conventional processor that acts as the master controller. We analyze the benefits of using our scheme by using a traditional evaluation framework with *power, execution time, energy* and *energy-delay product* as the metric. Then, we highlight the drawbacks of using a traditional framework and define a more relevant framework for evaluating multi-resource heterogeneous systems. In this framework, a new metric called *Energy-Resource Efficiency (ERE)*, illustratively portrays the performance-energy tradeoffs by directly linking the performance and energy variations in a system. The ERE can also be used as a guide to determine the amount of resources needed to set the system to a predefined performance-energy configuration.

In the next section, we discuss our experimental setup. Section 3 elaborates our experimental results based on a traditional analysis framework. Subsequently in Section 4, we define the new framework with ERE, and detail its implications on performance and energy consumption. Section 5 concludes the paper by summarizing our results.

2. EXPERIMENTAL SETUP

A heterogeneous communication system with DSPs and PowerPC forms the backbone for all our experiments. Our setup involves a board with one Motorola MPC7410 PowerPC chip [8] and a farm of twelve Motorola 16-bit DSP processors (MSC8101 [6]). The block diagram for our setup is shown in Fig. 1. The MPC7410 PowerPC (also referred to as PPC) is used only as a master controller. The MSC8101 DSPs are the actual computing resources.

We designed a centralized power-management module to handle the idle states of our system. This centralized module executing on the PPC, manages the power modes of both PPC and DSPs. By existing as an arbiter between the hardware and software layers, this module enables an application to set the entire system to various low-power modes. Our centralized power-management module puts the master PPC to *doze* mode whenever there is no

*This work was supported by DARPA under the contract F33615-01-C-1631.

0-7803-7979-9/03/$17.00 ©2003 IEEE

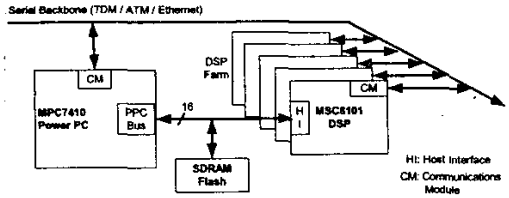


Figure 1: Heterogeneous architecture for our experiments

Table 1: Software setup

	PPC	DSP
Chip Name	MPC7410	MSC8101
OS	VxWorks 5.4	-
Development Environment	Tornado 2.0 (IDE)	GreenHills Multi 2000 (IDE)
Compiler	Cygnus 2.7.2 (gcc)	Optimized C Compiler

communication with the DSPs, or if the PPC is not controlling any task [7]. Even after assigning a task to the DSPs, the PPC remains in a low-power "receive" state waiting for communication from DSPs. If the DSPs are idle at any point of time, the module instantly transfers them to the *wait* state [6].

The PowerPC operates at 1.8V and each of the MSC8101 DSP processors at 1.5V. A HP34401A multimeter [1] connected across a small resistance (0.972 Ohms), is used for measuring the voltage, current and hence, the power consumption of our board. The default power mode for the MPC7410 PPC, is full-power mode and for the MSC8101 DSP, it is full-power mode (of DSP). Our board boots up and stabilizes to consume 15 watts of power (no low-power are enabled). Table 1 describes our development environment.

3. PERFORMANCE AND ENERGY ANALYSIS

3.1. Methodology

Modern communication-related systems (like set-top boxes, routers, switches) offer a lot of computing resources. In this section, we introduce a strategy that fully utilizes these resources to improve both performance and energy consumption. This scheme aims improvements by first studying an application, and then applying power optimizations along with some parallel techniques at the required code segments. This scheme can be implemented at any level of a system. A hardware-profiler, compiler, OS, or a generic scheduler can study the application (or its instructions) before execution and apply such schemes at their respective levels. We demonstrate our technique through a generic scheduler, running centrally on PPC. We execute a set of embedded benchmark applications and evaluate the setup using a traditional framework with *execution time (delay)*, *power*, *energy* and *energy-delay product (EDP)* as the metric.

Table 2 shows the benchmarks used in our experiments. The *art* and *bzip2* benchmarks are from the SPEC CPU2000 suite, whereas *g721*, *jpeg* and *pegwit* are from the MediaBench suite. Each benchmark is partitioned in a way that allows the core segments to be run in parallel, to ensure maximum utilization of DSPs. We integrate both high-performance and power-aware techniques into the same partitioning algorithm. The following part elaborates the partitioning strategy used for implementing each benchmark in our system.

1. The master controller (PPC) downloads the application code to the DSPs that are involved in execution.
2. After reading the input data, the PPC splits the raw data into blocks of static size. Each block is assigned a "pending" status. The PPC assigns one of the "pending" blocks to each participating DSP.
3. The DSPs then work on their respective blocks in parallel. Once the computation starts, the PPC switches itself to *doze* mode if idle.
4. When a DSP finishes working on its block, it replies to the PPC with its output. The PPC converts the status of the received block from "pending" to "completed". If there is more chunk of work to be done (i.e., any more blocks with "pending" status), the returning DSP grabs another block to work on.
5. If there are no more pending jobs to take, the returning DSP switches to *wait* mode.
6. The PPC finally assembles the output data when all involving DSPs complete their execution.

In steps 3 and 5 of the above scheme, the low-power modes (*doze* for PPC, *wait* for DSP) are enabled by invoking the centralized power-management controller of Section 2. To recall, the controller already has schemes defined for remotely enabling low-power modes for the entire system.

3.2. Observations & Shortcomings of Traditional Metric

art, ***g721*** and ***bzip2*** benchmarks are scalable parallel algorithms that have the potential to reduce energy consumption too, besides improving performance [Fig.2(a),Fig.2(b), Fig.2(c)]. For an algorithm that is hard to parallelize or that which is not scalable (like ***pegwit, jpeg***), it is better to do the computation without much communication, i.e., with fewer DSPs [Fig.2(e),Fig.2(d)]. Our results show that by combining low-power optimizations with existing parallel techniques, one can achieve sizable improvements in both energy and performance of multi-resource heterogeneous systems.

A metric that highlights performance-energy tradeoffs is essential in an analysis framework. An existing metric to study performance-energy tradeoffs is *energy-delay product (EDP)*, which

Table 2: Benchmarks used in our study

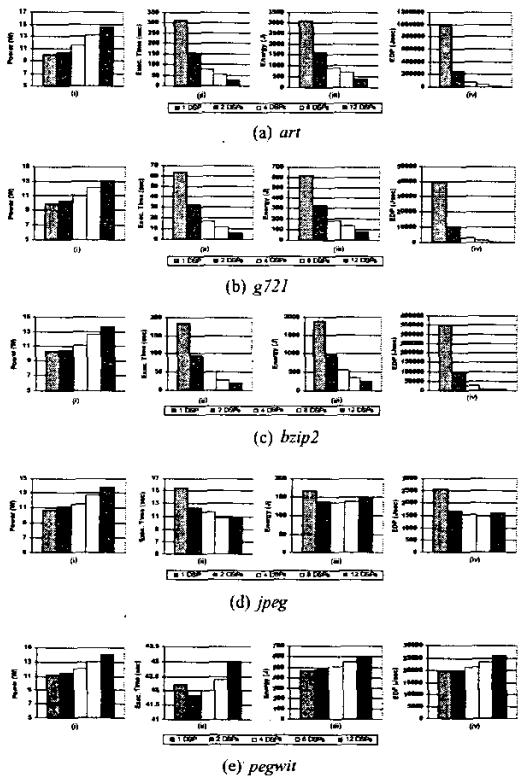
Benchmark	Explanation	DSP Code Size	Input Size
art	Neural network based pattern recognition algorithm	17KB	10KB image, 600KB weight
bzip2	Data compression	19KB	4MB data
g721	Voice compression	12KB	289KB voice
jpeg	Image compression	10KB	10MB image
pegwit	Public key encryption	29KB	220KB data

takes into account the delay and the energy consumption of a system. In our graphs, the EDP follows the trend of delay (execution time). Decreasing trends in EDP indicate good savings in both performance and energy. EDP is good at capturing overall trends. It is difficult to clearly say from EDP the mutual impact of performance and energy variations on one another. Moreover, in a multi-resource environment similar to our setup, the number of resources that are used to achieve any savings, also need to be considered during evaluation. This factor is not included in EDP. These shortcomings and special requirements motivate us to introduce a new framework of analysis in the following section.

4. RESOURCE-AWARE FRAMEWORK

The primary entity that drives our framework is a new metric called *Energy-Resource Efficiency (ERE)*. ERE illustrates the trade-off that occurs between the energy consumed by an application and the amount of resources needed to achieve any reduction in energy consumption.

Analytically, the *Energy-Resource Efficiency* is defined as



(a) *art*

(b) *g721*

(c) *bzip2*

(d) *jpeg*

(e) *pegwit*

Figure 2: The power consumption(i), execution time(ii), energy(iii) and energy-delay product(iv) for all benchmark applications

$$ERE = \Delta \times \eta \tag{1}$$

where, Δ is the fraction of the energy saved by using a particular configuration, and η is the efficiency of parallelization.

$$\Delta = \frac{E_S - E_N}{E_S} \quad \text{and} \quad \eta = \frac{S_p}{N} \tag{2}$$

where, E_S is the energy consumed by the non-parallelized (serial) version of the application and E_N is the energy consumed by the parallel version running on N processing elements. S_p is the speedup in execution time achieved through execution on N processors.

The base case for our experiments is the configuration that has the master controller and just one DSP to do the task. The parallelized version involves more DSPs doing the same work in parallel. The graphs in Fig. 3 present our values for η, Δ and ERE framework. A system designer has various choices to make depending on what is desired. For instance, if the designer is looking for improvements in performance, a quick look at the speedups [Fig. 3(i)] would be sufficient. Even further, by calculating the efficiency of parallelization η [Fig. 3(ii)], one can compare the improvements in speedups with the amount of parallelization. For improvements in energy consumption, the designer can look at the energy-savings [Δ graph - Fig. 3(iii)]. Alternatively, a system designer can deploy the corresponding number of DSPs (N) for parallelization depending on the desired savings (by defining the desired S_p, Δ or η).

4.1. Tradeoff Analysis using η, Δ and ERE

The ERE graph for our benchmark applications is shown in Fig. 3(iv). One can study performance-energy tradeoffs from the ERE graph. In Fig. 3(iv), *art* and *g721* have the same performance-energy behavior. That is, their trade-off points are the same. For a designer looking at both performance and energy, the 4 processor case proves to the best point with an ERE of 0.70. Beyond 4 processors, the ERE drops to 0.59 (for 8 DSPs) and then increases back to 0.80 (for 12 DSPs). There is only a minimal increase in ERE from 4 to 12 processor case, which makes the 4 DSP case a better choice. The reason is because the efficiency of parallelization (η) dips from 100% in the 4 processor case to almost 88% in the 12 processor case. Hence it makes more sense to use lesser number of DSPs to achieve a comparable ERE. ERE graph suggests that *art* is slightly better than *g721*, which is actually the case when comparing their energy and performance values independently too.

In the case of *bzip2*, the ERE values follow *g721* and *art* until the 4 DSP processor case. Up to this point, the algorithm utilizes the increasing number of DSPs very well to achieve improvements in both efficiency and energy-savings. Beyond this point, the ERE value of *bzip2* saturates to 0.69. This implies that beyond 4 processors, the improvements in performance or energy does not influence the other. Hence, depending on desired performance or energy values, a lesser or larger number of DSPs can be chosen.

In case of *pegwit*, the negative and zero values of ERE indicate that the parallel implementation proves to be futile toward both energy and performance improvements.

For *jpeg*, the tradeoff is clearly visible. There is definitely an improvement in both the performance and energy consumption,

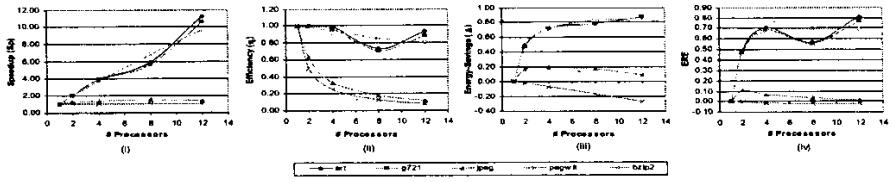


Figure 3: Speedup(i), efficiency(ii), fraction of savings in energy(iii) and Energy-Resource Efficiency (iv) for all benchmark applications.

which is indicated by the positive values of ERE. The ERE decreases beyond 2 DSPs because the improvements in energy get negated by deteriorating efficiencies (it should be noted that there still is a speedup that is achieved). A system designer would be interested in this graph to identify the number of DSPs to deploy, based on the desired trade-off. For instance, in the case of *jpeg*, if only the savings in energy consumption is important, the designer can implement 4 DSPs into the system (from the Δ graph). If performance is crucial, 12 DSPs can be deployed (from S_p graph). A lesser number of DSPs should be used if efficiency of parallelization is to be considered (looking at η graph). If both performance and energy-savings are crucial, 2 DSPs would be a good number as seen from the ERE graph [Fig. 3(iv)].

To summarize, system designers can use the *Energy-Resource Efficiency (ERE)* metric to

- identify the advantages and disadvantages of a particular configuration (by looking at positive and negative ERE values),
- detect the break-even point where the desired performance-energy tradeoff is achieved (by looking at the trends of ERE graph along with η and Δ),
- estimate the number of DSPs (resources) that are needed to achieve the desired performance-energy tradeoff (using the corresponding ERE values).

Also, it is evident that the ERE graphs capture trends that are not captured by other graphs and other equivalent metric. For instance, the ERE graph of *jpeg* in Fig. 3(iv) is different from its EDP graph [Fig. 2(d)] since it considers efficiency during evaluation, which is necessary when evaluating a multi-resource environment. The performance-energy tradeoffs are more clearly visible when the η, Δ, ERE framework is used during evaluation. ERE can easily be integrated into existing evaluation frameworks since it uses straight-forward mechanisms and formulas to study the tradeoffs. A design will be both performance- and power-aware when ERE is used.

5. CONCLUSIONS

Multi-resource heterogeneous communication systems have the potential to curtail energy consumption substantially, while simultaneously speeding up the application. In this work, we verified this by deploying a heterogeneous system with general purpose PowerPC as the master controller and a set of low-power DSPs as processing elements. By incorporating some existing parallelization techniques into a power-aware scheduler, we proved that one can achieve up to 80% improvement in power and performance of such systems.

From our experimental results, it is clear that energy consumption needs to be considered besides performance while assessing a system. ERE clearly portrays performance-energy tradeoffs in a system, in a way not done by any existing metric. Hence, by using our ERE and a guided performance-energy tradeoff, an application-aware communication system with multiple computing resources turns out to be power-aware with substantial performance improvements.

6. REFERENCES

[1] Agilent Technologies. *34401A Digital Multimeter Datasheet*, 2001.

[2] L. Benini and G. Micheli. System-level power optimization: techniques and tools. *ACM Transactions on Design Automation of Electronic Systems.*, 5(2):115–192, January 2000.

[3] D. Brooks, M. Martonosi, J.Wellman, and P. Bose. Power-performance modeling and tradeoff analysis for a high end microprocessor. *Lecture Notes in Computer Science*, 2008:126, 2001.

[4] Hitachi Ltd. *Hitachi SuperH Mobile Application Processor Specifications*, 2003. http://global.hitachi.com/New/cnews/E/2002/0415/.

[5] M. Lee, V. Tiwari, S. Malik, and M. Fujita. Power analysis and minimization techniques for embedded dsp software. *IEEE Transactions on VLSI Systems*, 5(1):123–135, March 1997.

[6] Motorola Inc. *MSC8101 16-Bit Digital Signal Processor Reference Manual, Document No. MSC8101RM/D*, 2001.

[7] Motorola Inc. *MPC7410 RISC Microprocessor Hardware Specifications, Document No. MPC7410EC/D*, 2002.

[8] Motorola Inc. *MPC7410 RISC Microprocessor Users Manual, Document No. MPC7410 UM/D*, 2002.

[9] Motorola Inc. *Motorola DCT5200 Digital Set-top Terminal Specifications*, 2003. http//broadband.motorola.com/.

[10] V. Narayanan, M. Kandemir, M.J. Irwin, H.S. Kim, and W. Ye. Energy-driven integrated hardware-software optimizations using simplepower. In *ISCA*, pages 95–106, 2000.

[11] M. Wan, Y. Ichikawa, D. Lidsky, and J. Rabaey. An energy conscious methodology for early design exploration of heterogeneous dsps. In *Custom Integrated Circuit Conference*, 1998.

[12] H. Yang, R. Govindarajan, G.R.Gao, and K.B.Theobald. Power-performance trade-offs for energy-efficient architectures: A quantative study. In *International Conference on Computer Design*, 2002.