PARAMETERIZED SOC DESIGN FOR PORTABLE SYSTEMS.

Sumant Bhutoria and Chaitali Chakrabarti

Department of Electrical Engineering
Arizona State University
Tempe, Arizona 85287
sumantb@asu.edu , chaitali@asu.edu

ABSTRACT

Portable devices have ushered in the use of parameterizable SoC architectures that provide flexibility and applicability while remaining cost-effective. Platune provides a mechanism for designing such architectures by exploring a large configuration space to arrive at pareto-optimal configurations with respect to power. In this paper, we argue that the pareto-optimal configurations for portable devices have to be derived for battery performance and not for power or energy consumption. We illustrate through examples that the optimal configuration obtained by optimizing battery performance is quite different from those obtained by optimizing power or energy.

1. INTRODUCTION

The popularity of cheap portable computing and communication devices has made portability an important design consideration. Portability demands not only an autonomous supply of energy, but also an appreciation of the physical form factors. The capacity of the energy source or battery has improved very slowly (about 5-10% CAGR) [1]. In order to cope up with the soaring computational demands, optimising with respect to battery performance is very important.

To be competitive, portable systems must be flexible, cost-effective and have low development cycles. This has triggered the emergence of a new class of architectures called parameterizable system-on-a-chip (SoC) platforms. These architectures have a set of parameters that can be easily customized to meet the stringent power and performance constraints of a specific application.

Platune [2] provides a framework for performance and power-tuning of a parameterizable SoC. It enables an embedded system designer to search the entire configuration space and select the appropriate architectural parameter values for a given application. Platune provides an exploration tool [3] to optimize the parameters with respect to power. It does not include any battery model or address the issues related to optimization with respect to battery performance. In order to design parameterizable SoC for battery-powered system, battery models need to be integrated with the existing architectural models and the parameterizable SoC optimized with respect to battery performance, rather than power and energy. This is in line with previous attempts to optimize battery-powered systems [4].

This work was partially supported by the Center for Low Power Electronics, an NSF/State/Industry Co-operative Research Center and Connection One, an NSF/Industry Co-operative Research Center.

In this paper, we focus on a procedure to design parameterized SoC platforms for battery-powered systems. The procedure is based on integrating the battery model [5] with Platune [2] to help derive pareto-optimal curves that are optimized with respect to battery performance. We demonstrate through examples that the optimal configuration obtained by optimizing power, energy or battery performance are quite different from each other. In fact, the deviations observed in power and energy, while optimizing the residual battery charge can be quite significant. Battery-powered embedded systems are thus not necessarily optimized by making them energy-efficient or power-efficient.

The rest of the paper is organized as follows. In Section 2, we study the features of Platune and the battery model. In Section 3, we describe the integration of Platune with the battery model. In Section 4, we show the resulting pareto-optimal configurations for the JPEG, G3FAX, and UCBQSORT examples. In Section 5, we conclude the paper.

2. PRELIMINARIES

2.1. Platune

Platune [2, 3]is based on a parameterized SoC platform. It consists of a MIPS R3000 processor, instruction cache, and data cache that communicate over two processor local buses, the CPU-instruction cache and the CPU-data cache bus. The cache-memory bus connects the on-chip memory with the two caches. The peripherals are connected via the peripheral bus which bridges over to the CPU-data cache bus.

The MIPS can be set to run at thirty-two different voltage levels (and thus thirty-two different frequencies). Each of the two caches are five way set associative and have five line size settings and have ten total cache sizes. The interconnect buses are each in turn composed of a data and an address bus. Each bus can be set to four different widths and three different bus codings.

While Platune provides exploration tools to optimize a given parameterizable SoC with respect to power, it does not do so with respect to energy or battery performance. Clearly, for portable SoC design, a battery model has to be included in the exploration tool.

2.2. Battery Model

A battery is characterized by the open-circuit voltage, the cut-off voltage, and the battery capacity. The battery performance is sensitive to the discharge current profile due to two important effects: rate capacity effect and the recovery effect. Battery models are used to capture some of these non-linearities and predict the

behavior under various conditions of charge/discharge. A number of analytical, electrical circuit, stochastic and electro-chemical models are used to describe the behavior of a battery.

In this paper, we use an analytical high-level battery model for use in portable electronic systems developed by Rakhmatov and Vrudhula [5]. This model holds for both constant and variable loads. For the case of variable loads, the load current is approximated in the interval [0,T] by a N-step function comprising of a current component I_k from time t_k to t_{k+1}. The charge consumed is given by:

$$\sigma \approx \sum_{k=0}^{N-1} I_k F(T, t_k, t_{k+1}, \beta)$$

where,

$$F(T, t_k, t_{k+1}, \beta) = \sum_{m=1}^{10} \frac{e^{-\beta^2 m^2 (T - t_{k+1})} - e^{-\beta^2 m^2 (T - t_k)}}{\beta^2 m^2} + t_{k+1} - t_k$$

The battery performance is described in terms of residual charge defined by $Q = \alpha - \sigma$, where α is the initial charge. Both α and β are quantities to be estimated from experimental data. This model has been verified against the DUALFOIL simulator as well as the Compaq's ITSY battery data, and the mean error for the model never exceeds 3% [5].

3. INTEGRATION OF THE BATTERY MODELS WITH PLATUNE

Realizing the need for battery life optimization, we have incorporated the battery in [5] into the existing Platune frame-work. The input to the battery model is a current profile. We generate the current profile from the power values generated every 450,000 cycles by the simulation engine of Platune.

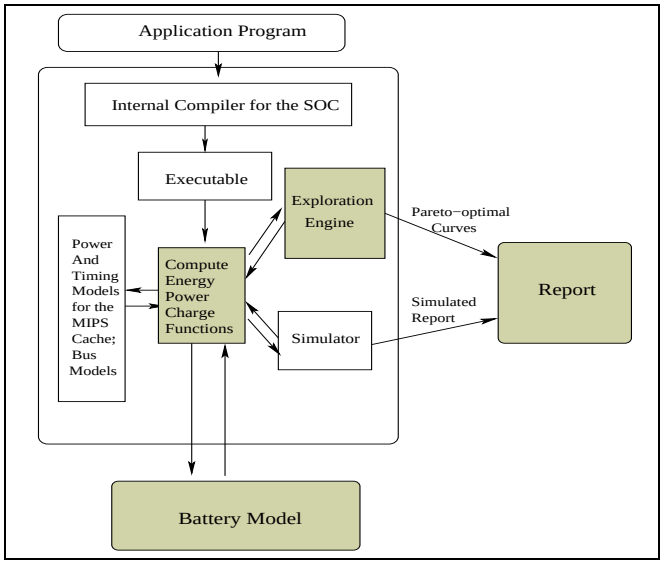

Fig. 1. Control Flow of Platune and Daler's model

In Figure 1, we describe the control flow. The application program is loaded onto Platune, which compiles it into its executable. Platune can, then, either explore the parameters for all pareto-optimal configurations, or else simulate for a given set of parameters. For power, energy and charge evaluation, the *Compute Power* and *Compute Charge* Functions profile the executable using the timing and power models in Platune, along with the battery-model that was added. These models help generate the power, energy and charge data that is then sent to the user as a report.

In the next section, we will see that the pareto-optimal configurations achieved with respect to power, energy, and battery life-time are distinctly separate. Optimization of one does not guarantee the optimization of the other two parameters.

4. RESULTS

The results presented in this section consist of:
1. The role played by each of the parameters (voltage, I$, D$ cache, and each of the data buses) on power, energy and battery performance.
2. The pareto-optimal configurations with respect to power, energy and battery performance. We focus our attention on the regions where significant deviations are noticed.

The optimizations and simulations were performed for the JPEG code taken from Motorola's Powerstone benchmark suite. We have also performed simulations on G3FAX and UCBQSORT. The G3FAX is a Group Three Fax Decode program containing sample fax data. The UCBQORT is the Berkeley Quick-sort, a sorting algorithm for a sample of 1000 numbers.

Additionally, since the initial 10% of the running time of an application involves a lot of variable initialisations and is not representative of the flow of the program, it is not included in our analysis.

4.1. Role of the parameters

In this section, we look at the role that each of the SoC parameters play on power, energy and battery performance and present the results for JPEG.

Parameter	Power (in W)	Energy (in Joule)	Charge (in mA-min)
Voltage	0.2931	0.0157	0.0693
I$	0.1976	0.0165	0.1002
D$	0.0706	0.0102	0.0676
CPU-I$ Bus	0.0493	0.0013	0.0149
CPU-D$ Bus	0.0282	0.0007	0.0065
I$/D$-Mem Bus	0.0026	0.0001	0.0004

Table 1. The role played by the different parameters on power, energy and charge for JPEG.

We identify independent clusters of parameters, and determine the impact that each of these clusters might have on power, energy and battery performance, while keeping the other parameters constant. These clusters are: Voltage, I$ and D$ cache (consisting of the total size, Line size, and associativity), CPU-I$, CPU-D$, and I$/D$-Memory address and data bus (consisting of bus width and code). We simulated for all the parameter values within a cluster, while taking the default value for the remaining parameters. Table 1 summarizes our results. For instance, the variation in power values due to different I$ cache configurations (everything else remaining constant) is 0.1976 Watts. We notice that voltage has a significant impact on power, while the I$

Time (in sec)	Power optimized Configuration		Energy optimized Configuration		Charge optimized Configuration	
	% Dev. in Energy	% Dev. in Charge	% Dev. in Power	% Dev. in Charge	% Dev. in Power	% Dev. in Energy
1.0	9.49	3.22	0.48	0.48	1.01	5.37
3.0	0.00	7.67	0.00	2.72	3.89	3.89
5.0	5.23	0.00	0.86	0.86	0.00	2.55
7.0	39.23	0.00	11.37	11.37	0.00	19.60
9.0	91.74	0.00	3.79	3.79	0.00	32.27

Table 2. Percentage deviations from the optimal for JPEG optimizations.

parameters play an important role in energy and charge. The memory bus parameters play an insignificant role in all three optimizations.

4.2. Pareto-optimal Configurations for JPEG

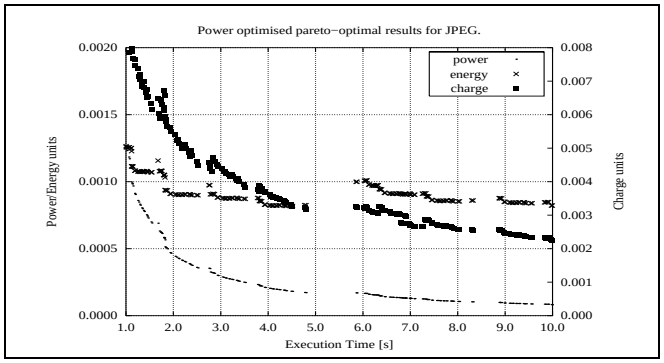

Fig. 2. Pareto-optimal curves for power for JPEG

In this section, we describe the pareto-optimal curves for power, energy and battery performance. The simulations were performed for 17 levels of voltage, I$, and D$ cache sizes of 16K, 1K, 512 and 256, line widths of 64, 16 and 4, associativity of 1 and 16 and bus widths of 4 and 32.

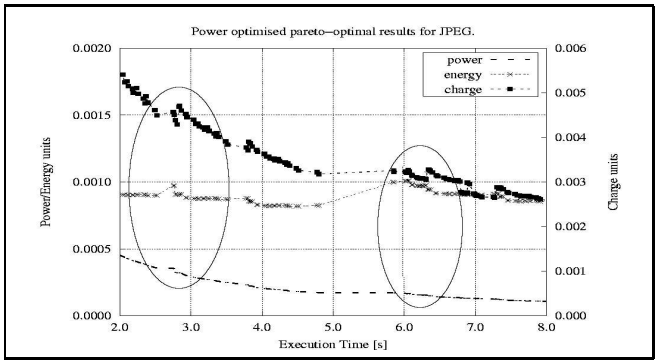

Fig. 3. Energy and residual charge increase, while power decreases.

Figure 2 shows the pareto-optimal configurations of JPEG optimized with respect to power. The energy and charge consumed for the optimal power configuration is shown in the same figure. Due to the large number of points, we have taken only those configurations which run from 1 second to 10 seconds. We note that while the power is a non-increasing function with time, both energy and battery life-time have a more random disposition. In fact, there are cases when they show significant increases with time, compared to power which decreases with time as further illuminated in figure 3.

Figure 4 shows the pareto-optimal configurations of JPEG optimized with respect to energy. We note that after time t=4.5s, there are no configurations that result in lower energy.

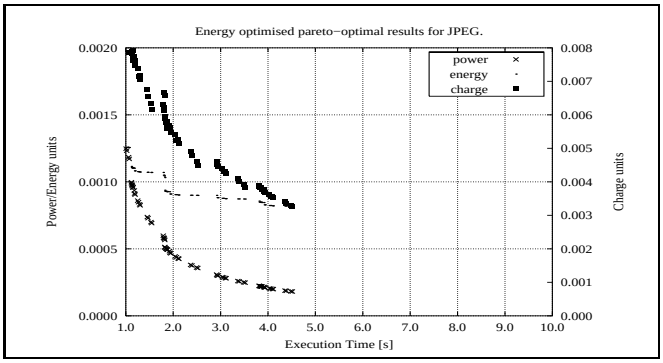

Fig. 4. Pareto-optimal curves for energy for JPEG

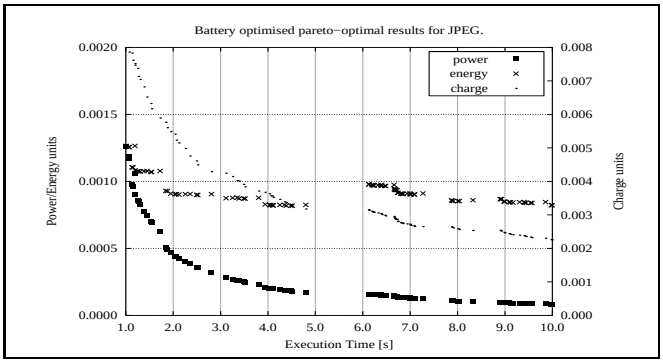

Fig. 5. Pareto-optimal curves for battery residual charge for JPEG

Figure 5 shows the pareto-optimal configurations with respect to residual battery charge. Once again, we notice that power and energy are not a strictly decreasing function with respect to time. From these sets of plots, it is clear that a SoC design optimized for power may not necessarily be a design that is optimized for energy or battery performance. As an example, if a designer had to design a configuration to run the JPEG for a time constraint of 6.0

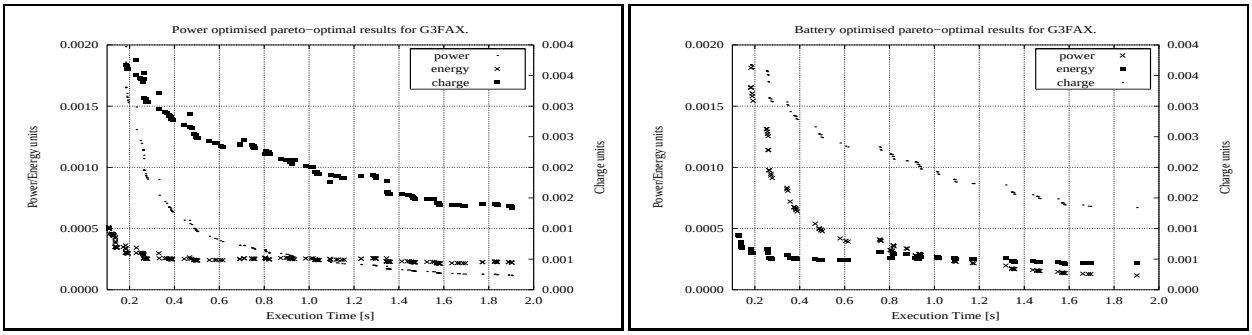

Fig. 6. Power and battery optimal curves for G3FAX.

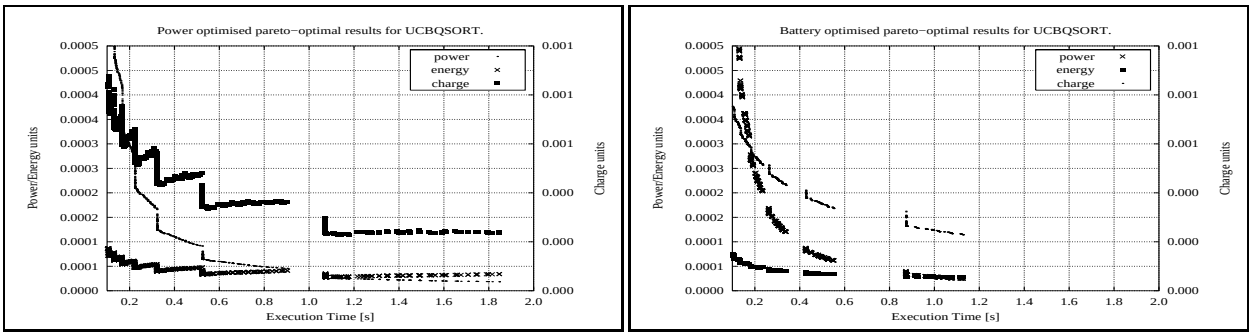

Fig. 7. Power and battery optimal curves for UCBQSort.

seconds, a power optimal configuration would have the processor running at 1.2V, an instruction cache size of 1024, line size of 4 and associativity of 1, a data cache size of 512, line size of 4 and associativity of 1, a CPU-I$ bus size of 4, CPU-D$ bus size of 32 and a I$/D$-Memory bus size of 4. However, the battery optimal configuration is not identical and would have the processor runing at 1.1V, a data cache size of 1024, a CPU-I$ bus size of 32, CPU-D$ bus size of 4; all other parameters are the same.

For a given time-bound, a power-optimized configuration may have energy (battery performance) that is quite different from an energy-optimized (battery-optimized) configuration. Table 2 describes (1) the deviations in energy and charge for a power-optimized configuration, (2) the deviations in power and charge for a energy-optimized configuration and (3) the deviations in power and energy for a charge-optimized configuration for five different execution time bounds.

Next, we try to understand the reason for the deviations. The main reasons are as follows:

1. If two current profiles (due to competing configurations) have the same $\sum(i * t)$, then the one with decreasing current has better battery performance

2. The current profile with lower $\sum(i * t)$ has better battery performance.

4.3. Additional Examples

The pareto-optimal configurations with respect to power, and battery have been shown for G3FAX and UCBQSORT in figures 6 and 7. We see that here too optimizing for power does not necessarily mean optimizing for energy or battery performance.

5. SUMMARY

In this paper, we show how portable SoC can be designed by integrating Platune with a battery model and deriving pareto-optimal configurations with respect to battery performance. We show through examples that optimizing with respect to power or energy does not guarantee maximal battery performance. Our current work is in designing a more efficient exploration engine.

6. REFERENCES

[1] Kanishka Lahiri, Anand Raghunathan, Sujit Dey, and Debashis Panigrahi, "Battery-driven system design: A new frontier in low power design," *International Conference on VLSI Design/ASP-DAC*, pp. 261–267, January 2002.

[2] Tony Givargis and Frank Vahid, "Platune: A tuning framework for systems-on-a-chip platforms," *IEEE Transactions on Computer Aided Design (TCAD)*, vol. 21, no. 11, November 2002.

[3] Tony Givargis, Frank Vahid, and Jorg Honkel, "System-level exploration for pareto-optimal configurations in parameterized system-on-a-chip," *International Conference on Computer-Aided Design*, November 2001.

[4] Tajana Simunic, Luca Benini, and Giovanni De Micheli, "Energy-efficient design of battery-powered embedded systems," *IEEE Transactions on VLSI systems*, vol. 9, no. 1, February 2001.

[5] Daler N. Rakhmatov and Sarma B.K. Vrudhula, "An analytical high-level battery model for use in energy management of portable electronic systems," *IEEE International Conference on Computer Aided Design*, pp. 488–493, November 2001.