

Hybrid Reconfigurable Processors — The Road to Low-Power Consumption

Invited Address

Jan M. Rabaey

Department of EECS, University of California at Berkeley

ABSTRACT

Energy considerations are at the heart of important paradigm shifts in next-generation designs, especially in systems-on-a-chip era. With multimedia and communication functions becoming more and more prominent, coming up with low-power solutions for these signal-processing applications is a clear must. Harvard-style architectures, as used in traditional signal processors, incur a significant overhead in power dissipation. It is therefore worthwhile to explore novel and different architectures and to quantify their impact on energy efficiency. Recently, reconfigurable programmable engines have received a lot of attention. In this paper, the opportunity for substantial power reduction by using hybrid reconfigurable processors will be explored. With the aid of a number of small benchmarks, it will be demonstrated that power reductions of orders of magnitude are attainable.

1. INTRODUCTION

Systems-on-a-chip are becoming a reality even today, combining a wide range of complex functionalities on a single die.^{1,2} Integrated circuits that merge core processors, DSPs, embedded memory, and custom modules have been reported by a number of companies. It is by no means a wild projection to assume that a future generation design will combine all the functionality of a mobile multimedia terminal, including the traditional computational functions and operating system, the extensions for full multimedia support including graphics, video and high quality audio, and wired and wireless communication support. In short, such a design will mix a wide variety of architecture and circuit styles, ranging from RF and analog to high-performance and custom digital.

Such an integration complexity may seem daunting to a designer and might make all our nightmares regarding performance, timing and power come true. On the other hand, the high level of integration combined with its myriad of design choices might be a blessing as well and can effectively help us to address some of the most compelling energy or power-dissipation problems facing us today. Even more, it might enable the introduction of novel power-reducing circuit techniques that are harder to exploit in traditional architectures.

The rest of the text is structured as follows. First of all, the opportunities for power (energy) reduction at the architectural level are discussed. Next we discuss how reconfigurable architectures can exploit some of these opportunities and we quantify the resulting power (energy) reductions for a number of benchmark examples.

2. OPPORTUNITIES FOR ENERGY MINIMIZATION

While most of the literature of the last decade has focused on power dissipation, it is really minimization of the energy dissipation in the presence of performance constraints that we are interested in. For real-time fixed-rate applications such as DSP, energy and power metrics are freely interchangeable as the rate is a fixed design constraint. In multi-task computation, on the other hand, both energy and energy-delay metrics are meaningful, depending upon the prime constraints of the intended design. In the remainder of the text, we will focus mainly on the energy metric, although energy-delay minimization is often considered as well.

Assuming that energy consumption will be dominated by capacitive switching for some time to come, the parameters the architecture designer can manipulate to reduce the energy budget include switching capacitance, and supply and signal voltages.

2.1. Voltage as a Design Variable

While traditionally the latter were fixed over a complete design, it is fair to state that in future systems-on-a-chip voltage can be considered as a parameter that can vary depending upon the location on the die and dynamically over time. This has been explored by a number of researchers over recent years and the potential benefits of varying supply voltages are too large to be ignored. It has been demonstrated that scaling the supply voltage in concert with the required workload (i.e. performance) produces a cubic reduction in dissipation — the result of a quadratic reduction in voltage and a linear reduction in switching capacitance! In contrast, powering down a module after a task is completed yields only a linear reduction.

Matching the desired supply voltage to a task can be accomplished in different ways. For a hardware module with a fixed functionality and performance requirement, the preferred voltage can be set statically (e.g. by choosing from a number of discrete voltages available on the die). Computational resources that are subject to varying computational requirements have to be enclosed in a dynamic voltage loop that regulates the voltage (and the clock) based on the dialed performance level.³ This concept is called *adaptive voltage scaling*.

Based on the above discussion, we may assume that an energy minimization flow treats supply voltage as an open variable, to be fixed in concert with architectural optimizations for switching capacitance.

2.2. Eliminating Architectural Waste

Reducing switching capacitance typically comes down to a single, relatively obvious task **avoid waste**. A perfunctory investigation of current integrated systems demonstrates that this is not that obvious or trivial. Energy is being wasted at all levels of the design hierarchy, typically as a result of the chosen implementation platform and implementation strategy:

- Circuit level — devices are often oversized.
- Logic level — the prevalent use of the standard cell methodology leads to large waste in energy (both logic and interconnect) for regular structures such as data paths.
- Architecture level — load-store architectures bring with them a huge overhead in terms of instruction fetching, decoding, data communication, and memory accesses.
- Application level — a large gap exists between the application designers and the underlying implementation platforms. As a result, applications specifications and algorithms are often selected without any clear insight in the impact on performance and energy.

In this paper, we concentrate on the potential optimizations at the architectural level. As stated, only a small fraction of the energy is typically spent on the real purpose of the design, i.e. computation. The rest is wasted in “overhead” functions such as clock distribution, instruction fetching and decoding, busing, caching, etc. Energy-efficient design should strive to make this overhead as small as possible, which can be accomplished by sticking to a number of basic guidelines (the *low-energy roadmap*):

Match architecture and computation to a maximum extent

- Preserve locality and regularity inherent in the algorithm
- Exploit signal statistics and data correlations
- Energy (and performance) should only be delivered on demand, i.e. an unused hardware module should consume no energy whatsoever.

This is most easily achievable in ASIC implementations, and it hence comes as no surprise that dedicated custom implementations yield the best solutions in terms of the traditional cost functions such as power, delay, and area (PDA). Indeed, it is hard to beat a solution that is optimized to perform solely a single well-defined task. However, it is also realized that time-to-market and flexibility are important assets as well. Many applications tend to require multi-functionality (for example, the multi-modal radio), or adaptivity. Very often, the specification of application tends to evolve during the design time and even after the part has been shipped for fabrication. This forms the lure for the so-called programmable solutions, which trade-off PDA for flexibility and (re)programmability.

It is our point of contention that adhering strictly to the *low-energy roadmap* can also lead to programmable architectures that consume dramatically less power than the traditional programmable engines. Reconfigurable architectures that program by restructuring the interconnections between modules are especially attractive in that respect, especially because they allow for obtaining an adequate match between computational and architectural granularity.

3. PROGRAMMABLE ARCHITECTURES — AN OVERVIEW

For a long time, programmable architectures have been narrowly defined to be of the *load-store* style processors, either in stand-alone format, or in clusters of parallel operating units (SIMD, MIMD). The latter have traditionally been of the homogeneous type, i.e. all processing units are of the same type and operate on the same type of data. In recent years, it has been observed at numerous sites that this model is too confining and that other programmable or *configurable* architectures should be considered as well. This was inspired by the success of programmable logic (FPGA) to implement a number of computationally intensive tasks at performance levels or costs that were substantially better than what could be achieved with traditional processors.^{4.)} While intrinsically not very efficient, FPGAs have the advantage that a computational problem can be directly mapped to the underlying gate structure, hence avoiding the inherent overhead of fixed-word length, fixed-instruction-set processors. Configurable logic represents an alternative architecture model, where programming is performed at a lower level of granularity.

Trading off between those architectures requires an in-depth understanding of the basic parameters and constraints of the architecture, their relationship to the application space, and the PDA (power-delay-area) cost functions. While most studies with this respect have been either qualitative or empirical, a quantitative approach in the style advocated by Hennessy and Patterson^{6.)} for traditional processor architectures is desirable. Only limited results in that respect have been reported. The most in depth analysis on the efficiency and application space of FPGAs for computational tasks was reported by Andre Dehon^{5.)}. A number of authors have considered various combinations of the above parameters. For instance, Dehon advocated the introduction of multiple contexts in an FPGA architecture. The PADDI architectures, introduced at UC Berkeley^{7.)-8.)} and targeting the rapid prototyping of high performance applications, increased *w*, *d*, and *c*. For instance, these parameters were set to 16, 6, and 8, respectively, in the PADDI-2 architecture.

Most of these studies ignore the impact of changing the flexibility index, which can have an enormous impact on the PDA cost function. This is illustrated by the example of a correlator for a CDMA radio. When comparing the energy needed to perform the operation on a variety of PEs, dramatic differences can be observed as shown in Table 1.

Table 1. Impact of architectural choice on energy dissipation for two important DSP functions.

Dot-vector product	ARM 6	Superscalar DSP	Reconfigurable Arithm.
	17 MIPS/W	266 MIPS/W	6 GIPS/W
CDMA correlator (per symbol)	ARM 6	Xylinx 4003	ASIC
	2765 nJ	394 nJ	1.2 nJ

The FPGA solution is an order of magnitude more efficient in energy than the general purpose processor (both architectures have a large degree of flexibility). This difference can mostly be attributed to the mismatch between granularity of application and architecture. On the other hand, optimizing the processing ele-

ment for a single function (the ASIC approach) results in another improvement by more than two orders of magnitude! This dramatic reduction in energy argues that reducing the flexibility is an option that should not be ignored when delineating the architectural space for the future systems-on-a-chip. This brings us to the next level of architectural modeling, the composition.

4. THE BERKELEY PLEIADES PROJECT

4.1. Concept

While it is theoretically conceivable to map virtually any computational function onto any of the possible reconfigurable structures (FPGA, reconfigurable data paths or arithmetic or instruction-set processor), doing so inevitably leads to either area, power or performance penalties. It is essential match computational and architectural granularity. Performing a multiplication on a PGA is bound to carry a huge amount of waste. So does executing a large dot-vector product on a microprocessor. The choice of the correct programming model can help to enable a wide range of power-reduction techniques, such as running at the minimum supply voltage and frequency, exploitation of locality of reference, and temporal and spatial signal correlations.

4.2. Architecture

The Pleiades architecture, under development at UC Berkeley^{9,11}, attempts to integrate a wider variety of reconfigurable components into a single structure. The architecture, shown in Figure 1, presents a reusable template that can be used to implement a *domain-specific processor instance*, that can then be programmed to implement a variety of algorithms within a given domain of interest. All instances of the architecture template share a common set of control and communication primitives. The type and the number of processing elements may vary; they depend upon the properties and the computational requirements of the particular domain of interest.

The architecture is centered around a reconfigurable communication network. Communication and computation activities are coordinated via a distributed data-driven control mechanism. Connected to the network are an array of heterogeneous, autonomous processing elements, called *satellite processors*. These could fall into any of the reconfigurable classes: a general microprocessor core (most of the time only one of these is sufficient), a

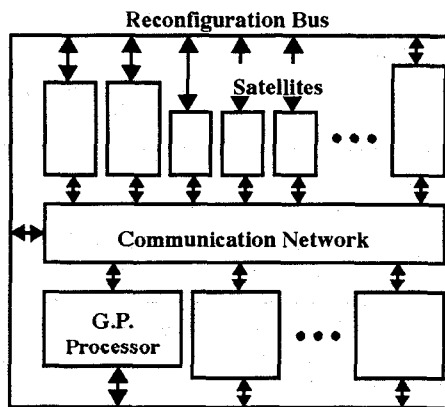


Fig. 1. Multi-granularity architecture template.

dedicated functional module such as a multiply-accumulator or a DCT unit, an embedded memory, a reconfigurable data path, or an embedded PGA. Observe that each of the satellite processors has its own autonomous controller, although the instruction set of most of these modules is very shallow (i.e. weakly programmable). The dedicated nature of the satellite processors makes it possible to execute multimedia operations with minimal overhead, hence achieving low energy per operation. The controller overhead is minimal as the instruction set of a given satellite processor is typically small, and very often is nothing more than a single control register. High performance can be achieved at low voltage level through the use of concurrency, both within a processor or by dividing a task over multiple processors. Finally, most of the data transfers and memory references within a processor access only local resources and are hence energy-efficient.

The microprocessor core plays a special role. Besides performing a number of miscellaneous non-compute intensive or control-dominated tasks, it configures the satellite processors and the communication network over the reconfiguration bus. It also manages the overall control flow of the application, either in a static compiled order, or through a dynamic real-time kernel.

The application is partitioned over the various computational resources, based on granularity and recurrence of the computational sub-problem. For instance, a convolution is mapped onto a combination of address generator, memory, and multiply-accumulate processors (Figure 2). The connection between these modules is set up by the control processor and remains static during the course of the computation. The same modules can in another phase of application be used in a different configuration to compute, for instance, an FFT.

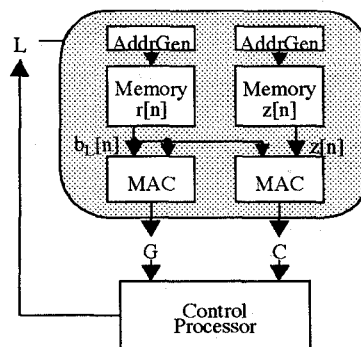


Fig. 2. Configured grouping of satellite processors to perform dot-vector product. Computationally non-intensive code, such as the final tests and the looping structures are implemented on the core processor.

4.3. Benchmark Results

A simple benchmark (IIR filter) helps to illustrate the impact of these choices on the PDA cost functions, in this case energy and energy-delay (Figure 3). The satellite processors selected for the Pleiades instance used in this analysis include address generators, SRAM memories, and multiply-accumulators. Savings of more than two order of magnitude can be observed with respect to a traditional processor (ARM). Even when comparing to DSP processors, which are optimized for these tasks, savings between 5 and 30 in energy-delay product — which compares the effi-

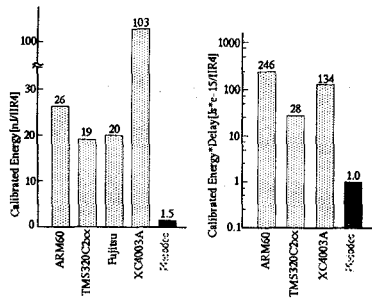


Fig. 3. Energy and Energy-Delay numbers for IIR filter, implemented on a variety of architectures. Numbers are normalized with respect to voltage and technology.

ciency of an implementation in the energy-performance domain — can be observed.

Current estimates indicate that compared to current general-purpose programmable signal processors, the proposed architecture can reduce the power dissipation of CELP-based speech coding algorithms by up to an order of magnitude, while maintaining a high level of programmability that can be used to implement various types of speech coding algorithms. We project that the VSELP coder can be implemented on our architecture in under 5 mW in a conservative 0.6 μm CMOS technology.

5. SUMMARY

Agile computing architectures, consisting of a heterogeneous collection of computational engines ranging from microprocessors to embedded programmable logic, will play a dominant role in the system-on-a-chip era. They combine the advantages of programmability, hence leveraging off the design cost of a complex part over a number of designs and providing at the same time adaptivity and flexibility, with the energy efficiency of more dedicated architectures. Selecting the correct model of computation for a given application (or application kernel) is the single most important prescription of the low-energy roadmap, presented in this chapter.

Research into the composition of these heterogeneous reconfigurable structures has just started and has some long way to go. Most importantly, quantifiable models of the impact of architectural parameters such as flexibility and granularity on the PDA costs and their relationship to the application parameters have to be developed. These will translate into a number of estimation tools that form the underpinning of a co-design methodology for heterogeneous architectures. The lack of an established implementation methodology is by far the most important impediment to the success of agile architectures.

6. REFERENCES

1. J. Borel, "Technologies for Multimedia Systems on a Chip", *Proc. IEEE ISSCC Conference 1997*, pp. 18-21, San Francisco, February 1997.
2. H. Sasaki, "Multimedia Complex on a Chip," *Proc. IEEE ISSCC Conference 1996*, pp. 16-19, San Francisco, February 1996.
3. A. Chandrakasan, "Data-driven signal processing: An approach for Energy Efficient Computation", *Proc. ISLPED 96*, pp. 347-352, Monterey.
4. J. Villasenor and W. Mangione-Smith, "Configurable Computing", *Scientific American*, pp. 66-73, June 1997.

5. A. DeHon, "Reconfigurable Architectures for General Purpose Computing," Technical Report 1586, MIT Artificial Intelligence Laboratory, September 1996.
6. Hennessy and D. Patterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufman Publishers, San Mateo, 1990.
7. D. Chen and J. Rabaey, "A Reconfigurable Multiprocessor IC for Rapid Prototyping of Real Time Data Paths", *Proc. IEEE ISSCC Conference 1992*, pp. 56-57, San Francisco, February 1992.
8. A. Yeung and J. Rabaey, "A 2.4 GOPS Data-Driven Reconfigurable Multiprocessor IC for DSP", *Proc. IEEE ISSCC Conference 1995*, pp. 108-109, San Francisco, 1995.
9. *The Pleiades Project Home Page*, <http://infopad.eecs.berkeley.edu/research/reconfigurable>
10. A. Abnous and J. Rabaey, "Ultra-Low-Power Domain-Specific Multimedia Processors," *Proceedings 1996 VLSI Signal Processing Workshop*, pp. 459-468, San Francisco, October 1996.
11. J. Rabaey, "Reconfigurable Computing — The Road to Low Power DSP", *Proceedings IEEE ICASSP Conference*, Munich, April 1997.