Reconfigurable Processing:
The Solution to Low-Power Programmable DSP

Jan M. Rabaey

Department of EECS, University of California at Berkeley

Abstract

One of the most compelling issues in the design of wireless communication components is to keep power dissipation between bounds. While low-power solutions are readily achieved in an application-specific approach, doing so in a programmable environment is a substantially harder problem. This paper presents an approach to low-power programmable DSP that is based on the dynamic reconfiguration of hardware modules. This technique has shown to yield at least an order of magnitude of power reduction compared to traditional instruction-based engines for problems in the area of wireless communication.

1. Introduction

Keeping the power dissipation within bounds is rapidly becoming one of the main challenges in contemporary digital design. This is especially the case in the domain of wireless communications. For compelling business reasons, extending the time between battery recharges has long been one of the highest priorities in the design of the mobile terminals. Energy-efficient implementations combining dedicated hardware with specialized processing elements have been proposed so that in-use and standby times of terminals are now measured in the range of hours and hundreds of hours, respectively.

There has been much ado recently about the need for "multimodal" of "software" radios, where most of the functionality of the radio is relegated to software. This concept has the advantage that a single terminal can support multiple standards (e.g. DECT and GSM) or can adapt dynamically to the requested services (voice, low-rate or high-rate data). The implied programmable approach annihilates many of the power-reduction techniques used in traditional terminals.

Minimizing the power dissipation has also become an issue for the basestation, where a large array of channels are processed simultaneously. It is desirable for a single basestation module to support multiple standards (e.g. GSM, IS95, DOCOMO), which by necessity leads to a programmable implementation platform. The higher energy dissipation associated with the programmable solution combined with the high computational requirements and the inaccessible and distributed locations of the basestations provide for a strong argument to keep power dissipation minimal there as well.

All these arguments point to the need of powerful programmable low-power compute engines. Numerous approaches towards low-power processing for wireless applications have been proposed: general-purpose (GP) processors with extended instruction sets; GP + co-processor structures; application-specific processors, and processors with power-scalable performance. This paper proposes an alternative approach to power-minimization based on a dynamic matching between architecture and computation. It will be demonstrated that reconfigurable architectures are an excellent vehicle for accomplishing this goal assuming that the granularity of reconfiguration is chosen in accordance with the model of computation of the targeted task.

2. Granularity of Programming Model

The term "programmable" typically has a strong association with the "stored-program" concept as originated from the computer world. A program is typically a set of instructions that dynamically modify the behavior of an otherwise statically connected modules such as memories, registers, and (a) datapath(s). In the last few decades, the concept of programming has gradually been extended to include the dynamic reconfiguration of interconnect networks and logic functionality as well. A first step in this direction was to connect multiple traditional processors in a dynamically adjustable configuration (multi-processor architectures). More recently, "adaptive computing systems (ACS)" have received increasing attention [1]. Typically, adaptive computing systems are constructed from arrays of field-programmable devices (FPGAs) and claim orders-of-magnitude in performance improvement over traditional computers by providing programmability (reconfigurability) at the gate level. While such performance improvements are possible for specific computational kernels, they may come at the price of an increase in area, power dissipation, and programming ease.

Programmability can actually come at multiple levels of granularity, and each has its own preferred and optimal application domain. Some of these levels are illustrated in Fig. 1. The two extremes are the already mentioned stored-program (a) and the gate (or transistor)-level (d) reconfigurable modules. Other options for reconfiguration at the functional-module (b) and the datapath-operator (c) levels.

The difference between the various computational models lies in the granularity of the composing modules, the distribution of the program storage, and the interconnect structure. Especially the latter requires some extra explanation: Stored-instruction engines rely on shared buses for the transfer of data (and instructions); the reconfigurable dataflow architecture uses either complete or reduced crossbar networks of busses; the interconnect network of a reconfigurable datapath exploits the bit-sliced structure and the predominantly one-dimensional flow of data by using an asymmetrical network: reconfigurable busses in one direction and a bit-level

(a) Stored-instruction processor   (b) Reconfigurable dataflow

(d) Reconfigurable datapath   (d) Reconfigurable logic
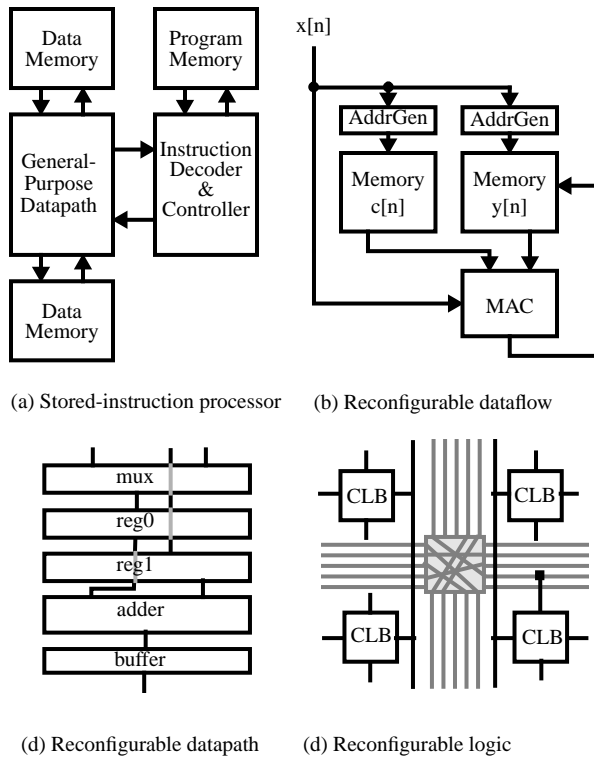
Fig. 1. Granularity of reconfiguration

mesh in the other; reconfigurable logic typically relies on bit-level mesh networks.

While it is theoretically conceivable to map virtually any computational function onto any of the reconfiguration structures, doing so inevitably leads to either area, power or performance penalties. In this paper, we demonstrate that combining the various models into a single architecture can lead to dramatic power reductions in wireless applications (but also for other domains that rely heavily on arithmetic and bit-level computations such as multimedia).

3. Granularity and Energy Consumption

The intense research in low-power electronics of the last five years has clearly identified the two key approaches to achieve substantial energy reduction:

- **Operate at the lowest possible voltage and frequency.** The loss in performance due to the lowering of the supply voltage is off-set by exploiting concurrency in the form of pipelining and parallelism. Observe that most wireless and multimedia applications have ample opportunity to do so as the algorithms in question exhibit an extensive amount of implicit concurrency. It was also observed [2] that the overexploitation of concurrency to reduce the supply voltage can lead to energy penalties: at a certain level of parallelism the overhead circuitry starts to dominate any gains from voltage reduction and the energy/computation starts to increase. This results in an optimum supply voltage and a optimum level of concurrency (Fig. 2.).
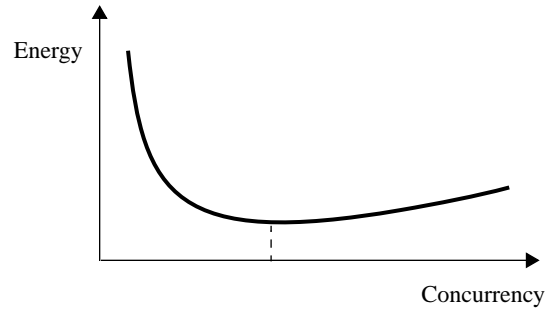

Fig. 2. Energy versus Concurrency

Observe that the position of that optimum is a strong function of the computation at hand. For example, it is perfectly conceivable to implement a complex computational algorithm on an PGA, thus exploiting the bit-level concurrency to the maximum. This approach will most often lead to an energy-inefficient solution as the PGA architecture implicitly carries an extensive energy overhead. On the other hand, for algorithms that require extensive bit-level operations (such as for instance encryption) the PGA architecture provides the most energy-efficient solution.

- **Reduce the energy waste**. While the former approach has received the most attention due to its quadratic nature, it is ultimately by keeping the energy overhead to a minimum that the most impressive power savings will be made (the reduction of supply voltage is lower-bounded by the implementation technology and the allowable standby power dissipation). Application-specific solutions present the most effective way of reducing energy waste and have been shown to lead to huge power savings [3]. As stated in the introduction, however, programmability is often a desirable and necessary feature. Making an architecture programmable (or reconfigurable) inherently carries with it a large energy overhead which most often dominates the energy dissipated for the intended computation. For instance, less than 15% of the energy dissipated in microprocessors goes to computation (it is actually far less if one accounts for the fact that this number also includes global communication) [4]. A energy/gate switch in an FPGA is approximately 7 to 10 times larger than in an ASIC implementation of the same gate.

A number of techniques can be identified to reduce energy overhead in programmable architectures:

- 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.

- preserve the locality inherent in the algorithm. Distributing the storage, interconnect, programming, and computation resources leads to a reduced cost per operation. For instance, executing a single instruction on a processor requires the fetching of the instruction from a large memory, decoding the instruction, and routing the control signals to the datapath.

- energy-on-demand. no unit should ever consume energy if not in use.

■ exploit signal statistics. avoid extensive multiplexing of communication and computation resources if data signals exhibit strong temporal correlations.

The choice of the correct programming model can help to enable these power-reduction techniques. In the next paragraph, we introduce a domain-specific architectural template that combines the four models of computation.

4. Multi-granularity Architecture

Fig. 3. 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 [5]. 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.
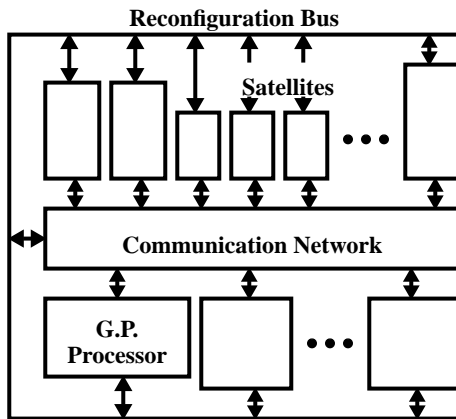

Fig. 3. Multi-granularity architecture template.

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 dedicated functional module such as a multiply-accumulator or a DCT unit, an embedded memory, a reconfigurable datapath, 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. For instance, the programmability of a multiply-accumulate processor is typically restricted to the number of samples to be accumulated, and numeric parameters such as the overflow, rounding and scaling characteristics.

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. 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.

The architectural template meets all the requirements for low-energy computation, as stated higher: it supports concurrency at multiple levels of granularity; it preserves locality; it employs application-specific modules if needed; it minimizes control overhead; its data-driven synchronization mechanism ensures that modules will only consume energy when needed; and the static configuration approach of the interconnect network preserves signal correlations.

For more information about the architecture and its components, please refer to [5].

5. Examples

The effectiveness of the multi-granularity architecture in reducing energy dissipation will be demonstrated using two examples.

5.1 A voice-coder processor

Low-power voice coders are an essential component of virtually all cellular terminal and basestation modules. Analysis of the computational requirements of a full-rate VSELP voice coder shows that 92% of the computational energy can be attributed to the inner loops of four functions, while vector-dot products account for 65% of the total power. Obviously, doing well on these operations brings us a long way towards a low-power solution. demonstrates how the dot-vector products are mapped to the reconfigurable architecture. The resulting structure consumes only 175 pJoule per MAC operation (including the memory accesses) (at 1.5V in a 0.6 mm CMOS technology). At a maximum clock rate of 30 MHz, this translates in a truly astounding 5.7 GOPS/Watt (with each OP being a full dot-vector operation). The total predicted power-dissipation for a full VSELP processor ranges between 2 and 5 mW.

5.2 CDMA baseband processing

Analysis of the baseband processing in a direct-conversion CDMA receiver (Fig. 5.) reveals that most of the computation resides in the execution of the 9 high-speed correlators [6], each of which multiplies a sequence of 4-bit numbers with a serial bit-stream. The high-performance requirements for the correlator units (Fig. 6.) make it impossible to construct the processor from functional modules, as was done for the dot-vector processor in the previous chapter. While it is conceivable to build a dedicated correlator processor, this unit would only be useful for the implementation of CDMA-based receivers. More appropriate here is the use of a reconfigurable datapath processor, that can also be
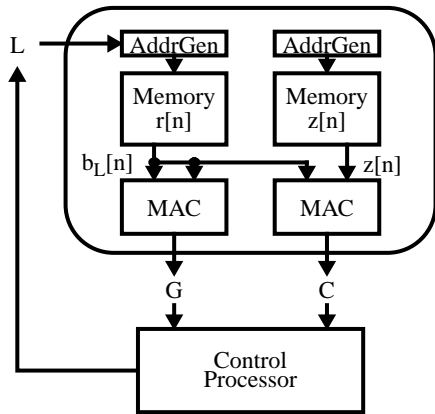
(a) Computational structure for dot-vector product.
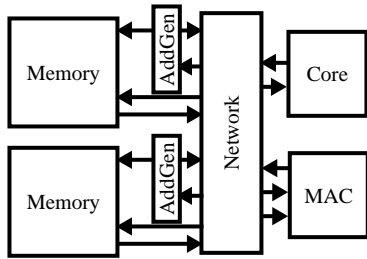
(b) Mapping of dot-vector product onto processor.

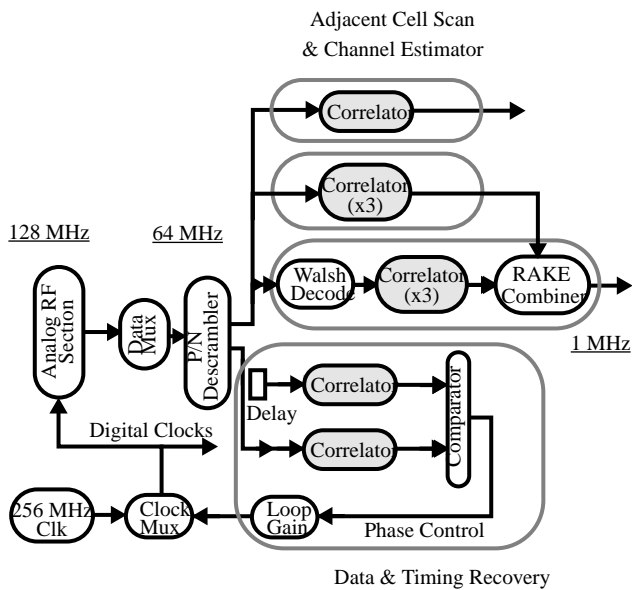Fig. 4. Energy-efficient implementation of dot-vector product on multi-granularity architecture.

Fig. 5. Digital baseband processing for 100 Mbits/sec spread-spectrum CDMA receiver.

used to implement other computational structures such as filters or modulators.

6. Summary

Contrary to the common belief, reconfigurable computing presents a very attractive opportunity high-performance programmable computing with high energy efficiency. Central to the approach is the matching between the granularity of computation in application and architecture. This by necessity leads to heterogeneous domain-specific architectures. This approach lends itself perfectly to the design of multi-modal, soft-programmable wireless terminals and basestations.

7. Acknowledgments

References

[1] *Adaptive Computing Systems*, http://www.ito.darpa.mil/ResearchAreas/Adaptive_Computing_Systems/ACSintro.html.

[2] A. Chandrakasan, *Low Power Digital CMOS Design*, Chapter 4, Kluwer Academic Publishers, 1995.

[3] J. Rabaey, *Low Power Digital Design*, in Circuits and Systems Tutorials, pp. 373- 386, LTP Electronics LTD, 1994.

[4] R. Gonzalez and M. Horowitz, "Energy Dissipation in General Purpose Processors," Digest of Technical Papers, 1995 IEEE Symposium on Low Power Electronics, pp 12-13, San Jose, 1995.

[5] A. Abnous and J. Rabaey, "Ultra-Low Power Domain-Specific Multimedia Processors", in *VLSI Signal Processing* IX, Ed. W. Burleson et al., IEEE Press, pp. 459-468, November 1996.

[6] S. Sheng, L. Lynn, J. Peroulas, K. Stone, R.W. Brodersen, "A Low-Power CMOS Chipset for Spread-Spectrum Communications," 1996 *International Solid-State Circuits Conference*, Feb 8-10, 1996, San Francisco, CA. Oct. 1995.
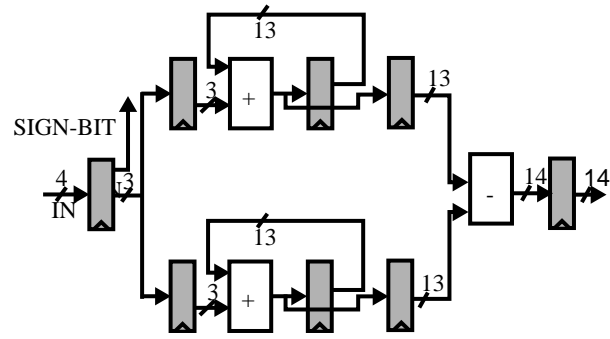
Fig. 6. Structure of correlator.