

Reconfigurable Processors for High-Performance, Embedded Digital Signal Processing*

Paul Graham and Brent Nelson

459 Clyde Building, Brigham Young University, Provo UT 84602, USA
grahamp@ee.byu.edu, nelson@ee.byu.edu

Abstract. For high-performance, embedded digital signal processing, digital signal processors (DSPs) are very important. Further, they have many features which make their integration with on-chip reconfigurable logic (RL) resources feasible and beneficial. In this paper, we discuss how this integration might be done and the potential area costs and performance benefits of incorporating RL onto a DSP chip. For our proposed architecture, a reconfigurable coprocessor can provide speed-ups ranging from 2-32x with an area cost of about a second DSP core for a set of signal processing applications and kernels.

1 Introduction

For high-performance, embedded digital signal processing, digital signal processors (DSPs) are very important, but, in some cases, DSPs alone cannot provide adequate amounts of computational power. As shown in [1, 2], reconfigurable logic (RL), specifically, FPGAs, can profitably be used for signal processing applications which require large amounts of computation and outperform existing high-performance DSPs despite the weaknesses of current commercial FPGAs in performing arithmetic. Unfortunately, FPGAs cannot effectively handle applications requiring high-precision arithmetic or complex control. As a compromise, we have been exploring DSP-RL hybrid architectures which enjoy the flexibility and precision of DSPs while experiencing performance improvements due to RL.

In this paper, we introduce a hybrid DSP-RL processor which tightly couples the DSP core to reconfigurable logic for greater performance and flexibility with digital signal processing applications without altering the DSP's embeddable nature. First, we will discuss the benefits and drawbacks of such an architecture. Following this discussion, we will describe many of the possible hybrid architectures and provide our performance and silicon-area estimates for one specific architecture. The conclusion to the paper provides some summary comments on this hybrid and a few words regarding on-going and future work in this area.

* Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-97-1-0222. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

2 Why a DSP Hybrid?

A large number of projects, including [3, 4, 5, 6, 7, 8], have explored the use of processor-RL hybrids for “general-purpose” computing applications. These projects have generally involved coupling a RISC, often a MIPS, processor core with reconfigurable logic and, by doing so, have shown promising speed-ups for a range of applications, including DES encryption, the SPECint92 benchmarks, and image processing. The combination of the processor core with either reconfigurable function units or coprocessors enables the host processor to communicate through high-bandwidth, on-chip interconnection to the RL resources rather than the relatively slow interconnection made through an interface with an external system bus. With reconfigurable resources, these processors were able to compute with greater efficiency than software alone, having less software overhead due to address generation, branching, and function calls and exploiting more parallelism than is possible with the processor’s normal data path.

One hybrid processor-RL system called Pleiades [9, 10] has specifically targeted ultra-low power, embedded digital signal processing. The approach fuses an ARM core with a combination of reconfigurable logic, reconfigurable data path, and/or reconfigurable data-flow resources. In this case, due to power and performance constraints, the RISC core is used mainly for administrative tasks such as programming the reconfigurable resources and not for computation. According to the results in the above-cited papers, the architecture has proven its power-performance advantage over a number of common technologies such as DSPs and FPGAs for a few digital signal processing applications.

Unfortunately, RISC processors, despite their computation power, are not well suited for many embedded high-performance applications. These processors are often power hungry, require a number of support ICs, and exhibit behavior which is hard to deterministically characterize for real-time applications. These qualities often make DSPs better choices for embedded applications. Further, a handful of DSPs offer glueless multiprocessing and provide a computing density per board advantage over multiple embedded RISC processors—a metric very important to embedded system designers.

Though DSPs may be good candidates for embedded processors, they also have some disadvantages resulting from their design as efficient embeddable processors. For instance, programming DSPs is laborious relative to programming RISC processors since the related tools are less sophisticated and less efficient. Much of this follows from the emphasis on making DSPs both power and memory efficient as opposed to being compiler friendly. Further, DSPs’ explicit parallelism must be managed efficiently, which is not always an easy task. Also, due to the emphasis on low-power and low-latency design, DSPs tend to operate at lower clock frequencies than RISC processors.

Despite these drawbacks, DSP cores have several existing features which make them good candidates for supporting reconfigurable resources. First, many DSPs have either multiple on-chip memory banks or otherwise provide multiple memory accesses per cycle. Thus, the memory architecture of the processor does not have to be drastically modified to provide many independent memory ports

to the reconfigurable resources. Second, the parallel execution of the DSP core and the RL resources can often be expressed as simple extensions of DSP instruction sets, which already directly express some level of concurrency. Third, the on-chip DMA controllers which many DSPs already have may prove useful for configuring the RL without burdening the processor with the task. Moreover, since the execution of many DSP applications are quite deterministic, configuration pre-fetching scheduled at compile time [11] should be a useful technique. The expense of configuration management hardware does not appear to be justified for a DSP-RL hybrid processor if only a few kernels are deterministically loaded during application execution.

DSPs can benefit in several ways from using reconfigurable logic for on-chip processing. First, DSPs are often asked to perform tasks which require a large amount of bit-level data manipulation such as error control coding—functions for which they are ill-suited but can perform. Second, the parallelism which is inherent in many DSP applications often cannot be exploited well by a processor which can, at most, perform a few arithmetic operations per cycle. Some recently announced DSPs such as the Analog Devices’ Hammerhead and Tiger SHARCs use SIMD techniques to boost the performance of the processors for some applications, but, unfortunately, these techniques can only be used for a portion of all DSP applications. The parallelism that the on-chip reconfigurable resources can exploit is not limited to SIMD techniques. Another interesting, but minor benefit that DSPs can experience by using RL resources on-chip is the ability to generate application-specific address streams with RL. In [1], we discussed an application, delay-sum sonar beamforming, which required address generation for which the usual DSP hardware address generators were inefficient. As we show in Sect. 4, RL used just for address generation can provide a three-times speed-up for this application’s kernel at a very small silicon area cost. Lastly, RL can lead to more efficient use of memory bandwidth by performing computations on data as they are streamed to and from memory.

The DSP core also provides the hybrid processor with capabilities which a strictly reconfigurable architecture cannot offer. First, due to the fast reconfigurability of the DSP core from instruction to instruction and the size of its instruction store, the processor is better suited for complex, control-heavy portions of applications. Second, depending on the processor core used, the core can provide higher precision arithmetic than may be practical in the RL resources, meaning that the processor core and RL can play complementary roles in applications. The processor core can deal with operations which require high precision while the RL can provide speed-ups for functions which require large amounts of computation on lower-precision data objects.

3 Hybrid DSP Architectures

For our architectural studies, we have decided to use the Analog Devices’ SHARC DSP family as a model because of its features and performance in DSP applica-

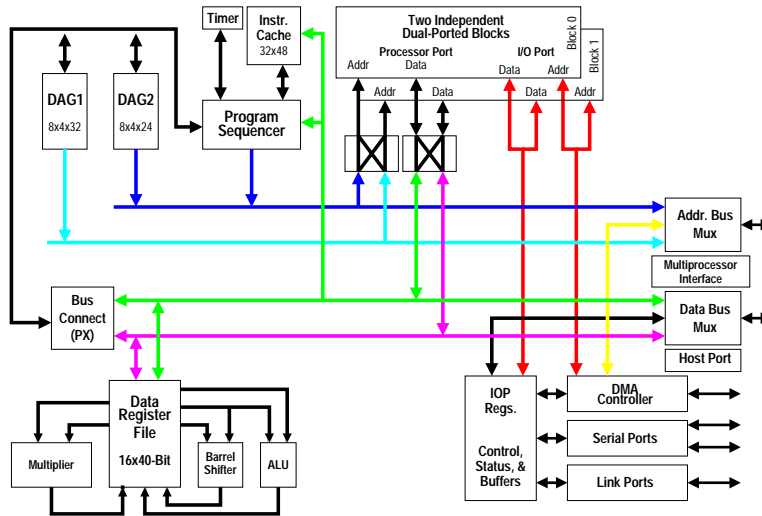


Fig. 1. SHARC Architecture

tions [12]. A block diagram of the architecture is provided in Fig. 1. Among the features which influenced our choice of the SHARC are:

1. its large on-chip memory (512 KB) organized as two banks of dual-ported memory, allowing for concurrent off-chip I/O and processing;
2. a data path capable of performing up to 3 arithmetic operations per cycle (a multiply with subtract and add);
3. several on-chip I/O processor peripherals including DMA controllers;
4. and, the support of glueless multiprocessing.

We chose this DSP despite the fact that it was a floating-point DSP and the applications and kernels in the benchmarks, described in Sect. 4, are implemented in fixed-point. Our rationale for this choice was that no fixed-point DSP currently has features equivalent to those listed above, all of which greatly improve the processor's performance for large applications. Second, considering that the DSP core is only 12% of the total die area, we do not believe the area comparisons would be greatly affected if the core was a fixed-point processor.

The reconfigurable portion of a DSP-RL architecture can be interfaced with the DSP core in several ways. First, the reconfigurable logic can act much like a function unit in the data path of the processor, having access only to the processor's register file. This approach is problematic since the memory bandwidth to the reconfigurable logic is effectively restricted to the one or two memory ports which service the register file. As was shown in [1, 13] and other work, restricting memory ports to only one or two greatly limits the performance of the architecture and causes the memory subsystem to be the bottleneck.

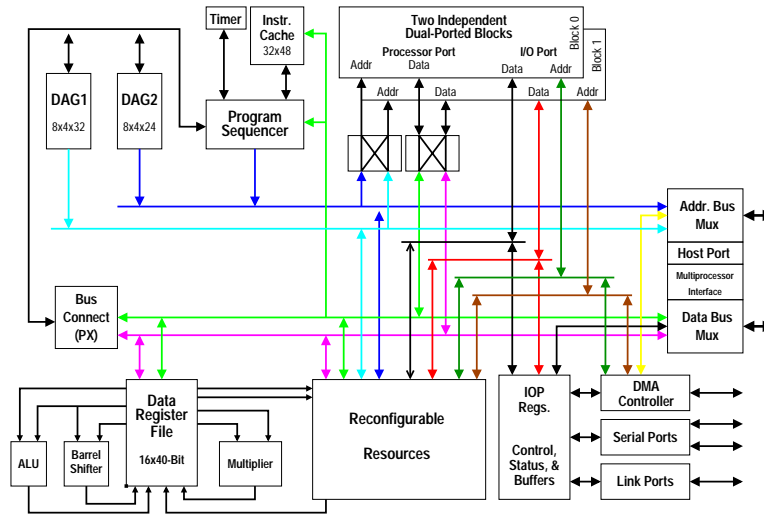


Fig. 2. DSP with a Reconfigurable FU/Coprocessor Combination

Another alternative is to treat the reconfigurable portion of the processor as a coprocessor. If the reconfigurable logic is treated as another peripheral such as the I/O processor of the SHARC, it again has only two memory ports and must compete with the I/O processor for memory, but the RL can operate concurrently with the processor core without disturbing the core's operation. Of course, this does not alleviate the lack of memory ports.

Another variation of this architecture is to modify the memory interfaces to provide a total of 4 memory ports for the reconfigurable processor; in the SHARC architecture, this means sharing both the DSP core's memory ports and the I/O processors' memory ports, i.e., having access to all of the memory ports of the two dual-ported, on-chip memories. Though the DSP cannot run concurrently with the reconfigurable coprocessor when all of the memory ports are being used by the coprocessor, the accessibility of the independent memory ports can enable the reconfigurable coprocessor to speed up applications beyond what the DSP core itself can provide. With the proper, careful assignment of memory ports, concurrent operation should still be possible for the DSP and reconfigurable coprocessor with this structure.

Among the other possibilities, another variation is to provide the reconfigurable resources both access to the processor register file and the four on-chip memory ports; this is illustrated in Fig. 2. This makes the reconfigurable resources a cross between a function unit and a coprocessor. This approach provides flexibility and reduces the requirement of using memory as the means of communication between the two data paths, further reducing the memory bandwidth burden and allowing tighter cooperation between the DSP core and the RL.

Table 1. Benchmark Applications and Kernels

Function	Description
BYU Applications	
Delay-Sum Beamforming	Beamforming in which the phase shifts are performed by time delays
Frequency-Domain Beamforming	Beamforming in which the phase shifts are performed in the frequency domain
Matched Field Beamforming	Beamforming in which a multi-ray model is used to estimate angle and distance of the target.
Benchmarks Inspired by BDTmark benchmark	
Real Block FIR	Finite impulse response filter that operates on a block of real (not complex) data.
Complex Block FIR	FIR filter that operates on a block of complex data.
Real Single-Sample FIR	FIR filter that operates on a single sample of real data.
LMS Adaptive FIR	Least-mean-square adaptive filter; operates on a single sample of real data.
IIR	Infinite impulse response filter that operates on a single sample of real data.
Vector Dot Product	Sum of the point-wise multiplication of two vectors.
Vector Add	Point-wise addition of two vectors, producing a third vector.
Vector Maximum	Find the value and location of the maximum value in a vector.
Convolutional Encoder	Apply convolutional forward error correction code to a block of bits.
Finite State Machine	A contrived series of control (test, branch, push, pop) and bit manipulation operations.
256-Point, Radix-2, In-Place FFT	Fast Fourier transform converts a normal time-domain signal to the frequency domain.

4 Area and Performance Results

As a starting point for our work, we have been evaluating the architecture of Fig. 2 for performance potential and silicon area considerations. As a way of benchmarking the performance, we have chosen a collection of applications and kernels which represent typical digital signal processing computations; these are listed in Table 1. In the mix of applications and kernels, we have included a few sonar beamforming applications [1, 14] with which we are very familiar. The other kernels are modeled after the collection of kernels in the BDTmark benchmark defined by Berkeley Design Technology Inc., who have identified these kernels as important to many DSP applications[15].

Our initial studies assume that the reconfigurable logic looks much like the Xilinx 4000 family of FPGAs. Though we expect coarser-grain RL architectures

such as CHESS [16], CFPA [17], and the Garp RL array [6] would be better for the hybrid, we use this RL architecture because it is well known and understood. Also, it provides a nice upper bound for the area-performance trade-offs in the hybrid architecture since the Xilinx 4000 architecture will generally be more costly for implementing data-path operations than the coarse-grain field programmable architectures mentioned above. In fact, we expect these coarse-grain architectures to require only about 40% to 55% of the silicon area of the Xilinx 4000 architecture for the same data-path functionality. To its credit, though, the Xilinx 4000 architecture will often be more flexible for control hardware and bit-level operations due to its finer granularity.

From [12], we learn that the SHARC, using a .6 micron, two-layer metal process, requires about $3.32 \times 10^9 \lambda^2$ in area. Due to the large on-chip memories of the architecture, about $1.86 \times 10^9 \lambda^2$, or 56% of the die, is devoted to SRAM, while about 12% of the die area, or $3.99 \times 10^8 \lambda^2$, is the DSP core and data path. The remaining die area is devoted to on-chip peripherals, the I/O pads, and interfacing logic. With this information and area estimates for Xilinx 4000 CLBs with interconnect drawn from [18], we can estimate the relative area increases due to the reconfigurable logic for each of the applications in the benchmark suite. For instance, from [18], we learn that a CLB has a cost of approximately $1.26 \times 10^6 \lambda^2$, so a design requiring 100 CLBs would require about $1.26 \times 10^8 \lambda^2$ in silicon area, which is only a 3.79% increase in total die area.

Table 2 describes the performance per area increase for the hybrid architecture for several of these benchmarks. The entries are sorted in decreasing speed-up per silicon area change. As an explanatory note, we should point out that the speed-up numbers do not take into account reconfiguration time for the kernel or application because our analysis assumes that reconfiguration is infrequent. Also, the RL implementations account for bit growth due to arithmetic operations, though simple scaling is essential in some cases. You should further note that only the reconfigurable logic portion of the hybrid is executing the benchmarks, except in a few cases that are noted in the table where the RL and the DSP core are operating cooperatively. In this study, we also assume that the reconfigurable coprocessor operates at the same clock frequency as the DSP. Though, in general, we would not expect this from the Xilinx 4000 architecture for higher-frequency DSPs (100–200 MHz), we expect that coarser-grain, data-path-oriented field-programmable architectures such as CHESS, CFPA, or the Garp RL array would be able to support these frequencies. For instance, the Garp RL array has been estimated to execute at a minimum of 133 MHz [6]. Lastly, as mentioned before, the reconfigurable implementations of the benchmarks are fixed-point implementations, not floating point.

With a reconfigurable logic budget of about 1000 CLBs, a good portion of the kernels and applications can be accelerated without frequent reconfiguration of the RL. This accounts for about a 40% increase in chip area. From our estimates and the published work on other field-programmable architectures, we believe that the cost of this amount of reconfigurable logic can be as little as 16–20% with computational fabrics such as CHESS or CFPA—an area just a

Table 2. Performance and Area of Hybrid for Benchmark Circuits

Application	Implementation	Pipe- lining	Speed- Up	Area (CLBs)	Total Area	Speed- Up / Δ Area
Convolutional Encoder	For V.32 Modem, with differential encoding	Full	34	5	1.0019	17929
Vector Add	Cooperative w/ DSP	N/A	2	40	1.0152	131
Vector Max.	16b data	Full	2	48	1.0182	110
Delay Sum BF	Coop. w/ DSP (Addr. Gen.)	N/A	3	100	1.0379	79.1
Delay Sum	Full PE	Full	3	246	1.0933	32.2
IIR Biquad	4 8b \times , 4 20b +	1/2	4	360	1.1365	29.3
FIR	32 tap, 16b KCM, 40b +	1/2	32	2976	2.1287	28.4
FIR	16 tap, 16b KCM, 40b +	1/2	16	1488	1.5643	28.4
FIR	2 tap, 16b KCM, 40b +	1/2	2	186	1.0705	28.4
FFT	CORDIC-based butterfly	1/2	4	380	1.1441	27.8
Matched Field BF	Sub-voxel, memory sensitive	1/2	8	928	1.3520	22.7
Matched Field BF	Sub-voxel, memory sensitive	Full	8	1073	1.4070	19.7
Vector Dot	Coop. w/ DSP, 16b \times , 40b +	1/2	2	308	1.1168	17.1
Matched Field BF	Sub-voxel, memory intensive	1/2	6	928	1.3520	17.0
Matched Field BF	Sub-voxel, memory intensive	Full	6	1073	1.4070	14.7
Freq. BF	Reported in [1]	1/2	4	856	1.3247	12.3
FIR	2 tap, 16b \times , 40b +	1/2	2	552	1.2094	9.55
IIR Biquad	4 16b \times , 4 24b +	1/2	4	1200	1.4551	8.79
IIR Biquad	5 16b \times , 4 24b +	1/2	4	1488	1.5643	7.09

little larger than the size of the DSP core. In addition, since the computation density per board is often one of the most important characteristics for embedded system designers, silicon area increases of 16%–40% are acceptable if the total computational density per board is increased by a factor of 2 or more.

5 Conclusions and Future Work

We have demonstrated that a reconfigurable architecture which incorporates a reconfigurable coprocessor into a DSP can have performance benefits for a reasonable increase in chip area. In addition, DSPs have many architectural features which make a combination with reconfigurable logic feasible and beneficial. Despite the raw clock rate disadvantage of DSPs when compared with general-purpose microprocessors, DSPs serve an important role in high-performance, embedded computing; a reconfigurable coprocessor on-chip can help DSPs exploit more of the parallelism found in digital signal processing applications, thus improving the processor's overall performance.

A large amount of work still remains to be performed and many outstanding questions exist. For instance, the amount of concurrent DSP-RL execution which

can be expected for DSP algorithms should be determined—a process which may require the use of automated programming tools and the mapping of many DSP algorithms to the architecture. If our current experience is representative, there may not be many situations in which concurrent operation is beneficial or possible, indicating that the connection of the RL array to the DSP's register file may not be needed.

Another unanswered question is whether the DSP and RL can truly serve complementary roles in applications—the DSP performing the control-heavy and high-precision arithmetic portions of the algorithm while the RL performs highly parallel operations on lower-precision data items. Again, a large amount of application mapping to the architecture may be required to answer this question, though adaptive signal processing algorithms may provide a fruitful set of applications for this study.

Other on-going and future work include a refinement of the programming methodology for the DSP hybrid processor as well as the many issues of interfacing the RL with the DSP. We expect that programming methodologies for the architecture will use a library-based design approach for the reconfigurable logic in which the designer simply describes the interconnection of the library modules using an assembly-like language. The intention is to make the programming task resemble more of a software creation problem than a hardware design exercise, a requirement crucial in making the architecture usable by DSP programmers and not just hardware designers. Future work will also quantify the effects of frequent reconfiguration on application performance.

References

- [1] P. Graham and B. Nelson. FPGA-based sonar processing. In *Proceedings of the Sixth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '98)*, pages 201–208. ACM/SIGDA, ACM, 1998.
- [2] P. Graham and B. Nelson. Frequency-domain sonar processing in FPGAs and DSPs. In *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines (FCCM '98)*. IEEE Computer Society, IEEE Computer Society Press, 1998.
- [3] Rahul Razdan. *Programmable Reduced Instruction Set Computers*. PhD thesis, Harvard University, Cambridge, MA, May 1994.
- [4] R. D. Wittig and P. Chow. OneChip: An FPGA processor with reconfigurable logic. In J. Arnold and K. L. Pocek, editors, *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines*, pages 126–135, Napa, CA, April 1996.
- [5] Scott Hauck, Thomas W. Fry, Matthew M. Hosler, and Jeffery P. Kao. The Chimaera reconfigurable functional unit. In Kenneth L. Pocek and Jeffery Arnold, editors, *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, pages 87–96. IEEE Computer Society, IEEE Computer Society Press, April 1997.
- [6] John R. Hauser and John Wawrzynek. GARP: A MIPS processor with a reconfigurable coprocessor. In J. Arnold and K. L. Pocek, editors, *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines*, pages 12–21, Napa, CA, April 1997.

- [7] Charle R. Rupp, Mark Landguth, Tim Garverick, Edson Gomersall, Harry Holt, Jeffrey M. Arnold, and Maya Gokhale. The NAPA adaptive processing architecture. In Kenneth L. Pocek and Jeffrey M. Arnold, editors, *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines (FCCM '98)*, pages 28–37. IEEE Computer Society, IEEE Computer Society Press, April 1998.
- [8] Jeffery A. Jacob and Paul Chow. Memory interfacing and instruction specification for reconfigurable processors. In *Proceedings of the Seventh ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '99)*, pages 145–154. ACM/SIGDA, ACM, 1999.
- [9] Arthur Abnous and Jan Rabaey. Ultra-low-power domain-specific multimedia processors. In *Proceedings of the IEEE VLSI Signal Processing Workshop*. IEEE, IEEE, October 1996.
- [10] Arthur Abnous, Katsunori Seno, Yuji Ichikawa, Marlene Wan, and Jan Rabaey. Evaluation of a low-power reconfigurable DSP architecture. In *Proceedings of the Reconfigurable Architectures Workshop*, March 1998.
- [11] Scott Hauck. Configuration prefetch for single context reconfigurable coprocessors. In *Proceedings of the Sixth ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '98)*, pages 65–74. ACM/SIGDA, ACM, 1999.
- [12] Joe E. Brewer, L. Gray Miller, Ira H. Gilbert, Joseph F. Melia, and Doug Garde. A single-chip digital signal processing subsystem. In R. Mike Lea and Stuart Tewksbury, editors, *Proceedings of the Sixth Annual IEEE International Conference on Wafer Scale Integration*, pages 265–272, Piscataway, NJ, 1994. IEEE, IEEE.
- [13] Csaba Andras Moritz, Donald Yeung, and Anant Agarwal. Exploring optimal cost-performance designs for Raw microprocessors. In Kenneth L. Pocek and Jeffrey M. Arnold, editors, *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines (FCCM '98)*, pages 12–27. IEEE Computer Society, IEEE Computer Society Press, 1998.
- [14] Paul Graham and Brent Nelson. FPGAs and DSPs for sonar processing—inner loop computations. Technical Report CCL-1998-GN-1, Configurable Computing Laboratory, Electrical and Computer Engineering Department, Brigham Young University, 1998.
- [15] Garrick Blalock. The BDTImark: A measure of DSP execution speed. Technical report, Berkeley Design Technology, Inc., 1997. Available at <http://www.bdti.com/articles/wtpaper.htm>.
- [16] Alan Marshall, Tony Stansfield, Igor Kostarnov, Jean Vuillemin, and Brad Hutchings. A reconfigurable arithmetic array for multimedia applications. In *Proceedings of the Seventh ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '99)*, pages 135–143. ACM/SIGDA, ACM, 1999.
- [17] Alireza Kaviani, Daniel Vranesic, and Stephen Brown. Computational field programmable architecture. In *Proceedings for the IEEE Custom Integrated Circuits Conference (CICC '98)*, pages 12.2.1–12.2.4. IEEE, IEEE, 1998.
- [18] André DeHon. *Reconfigurable Architectures for General-Purpose Computing*. PhD thesis, Massachusetts Institute of Technology, September 1996.