

A MEDIUM-GRAIN RECONFIGURABLE CELL ARRAY FOR DSP

José G. Delgado-Frias, Mitchell J. Myjak, Fredrick L. Anderson, and Daniel R. Blum

School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752 USA
Email: {jdelgado, mmyjak, fanderso, dblum}@eecs.wsu.edu

ABSTRACT

Digital signal processing (DSP) is an essential component of many applications, including multimedia and communications systems. The recent surge in wireless and mobile computing underscores the need for high-performance low power DSP hardware. Reconfigurable hardware balances these requirements with development costs by providing system designers a viable alternative to custom integrated circuits.

This paper describes a novel reconfigurable architecture for DSP applications. The device contains an array of medium-grain cells that can perform arithmetic, memory, and control operations. The main features of the architecture are as follows: flexible structures, variable word length, pipeline latches, and error correction. A prototype of the cell is being fabricated in 0.5- μm technology. Circuit simulations indicate that the array achieves a clock frequency of 100 MHz even with this modest technology and a performance comparable to the highest-performance DSP processors today.

KEY WORDS

Reconfigurable Cell Array, Medium-Grain, VLSI Circuits and Systems, Digital Signal Processing.

1. INTRODUCTION

Many digital systems—ranging from computers to cellular phones, and DVD players to satellites—rely on digital signal processing (DSP) to achieve their functionality. DSP underlies the human-computer interface of digital multimedia applications, including sound cards, video cards, and speech recognition systems. Communications front-ends, such as DSL modems and CDMA receivers, use DSP algorithms extensively for decoding and compression. In recent years, the application space of DSP has shifted to include wireless and mobile computing. As a result, the most critical metrics for DSP implementations today are performance and power consumption. This evolution requires novel hardware architectures to meet the new demands and challenges.

Conventional DSP implementations do not integrate performance and low power with high flexibility. Application-specific integrated circuits (ASIC) achieve the best performance and power consumption, but can only support one DSP algorithm. General-purpose processors can execute a wide variety of software programs, but offer relatively poor performance compared to an ASIC. Balancing these two extremes is the goal of reconfigurable hardware. Reconfigurability allows a device to adapt to new applications or changes to the current system, even after deployment. Traditional reconfigurable devices such as field programmable gate arrays (FPGA) have good fine-grain flexibility. However, FPGA implementations of regular DSP structures are very inefficient.

This paper describes a novel reconfigurable architecture for DSP applications. In this approach, the device contains an array of cells, each of which handles a small portion of the overall operation. The structure of a cell incorporates the functionality, performance, and power requirements necessary for DSP applications today. In addition, the organization of the device provides exceptional flexibility. Users can tailor the device to the processing task at hand by controlling the word length, number of parallel functional units, and functional unit connectivity. More generally, exposing these hardware resources to software management allows for more efficient parallelism via the tradeoff of temporal and spatial utilization of the device. The architecture also contains internal error correction schemes necessary for mission-critical applications. After an error is detected, users can re-map functionality around damaged cells and continue using the device normally.

Section 2 of this paper describes the relationship between the reconfigurable cell array and other DSP implementations. Section 3 provides an overview of the design, while Section 4 discusses some of the major features that increase performance, flexibility, or reliability. Section 5 presents several performance estimates gathered through circuit-level and system-level simulations, and describes the prototype currently being fabricated to demonstrate the architecture. The paper ends with some concluding remarks.

2. DSP IMPLEMENTATION SPECTRUM

Currently, digital systems may use a variety of components to perform DSP, ranging from application-specific integrated circuits to general-purpose microprocessors. Table 1 provides a comparison of these approaches in terms of performance, power consumption, and flexibility. Note that reconfigurable hardware is very flexible because its functionality and internal structure of the device can be customized after fabrication.

Table 1. Comparison of DSP implementations [1]

Device	Performance	Power	Flexibility
General-purpose processor	Low	Medium	High
DSP processor	Medium	Medium	Medium
Configurable processor	Medium	Med/Low	Medium
Reconfigurable hardware	Med/High	Med/Low	High/Med
ASIC	High	Low	Low

General-purpose processors can execute a wide variety of programs, including DSP algorithms. However, their performance may not meet the application requirements [1,2]. DSP processors include some instructions tailored for DSP computations. They generally achieve better performance than general-purpose processors, but their architecture may not be optimized for the different requirements that DSP applications may have, such as speed, power, and word length.

Configurable processors have a customizable instruction set, datapath, and memory organization. Devices of this type are configured for a particular application prior to fabrication [3]. However, each configuration requires a new compiler to generate optimal code. In addition, the use of such a processor may be limited to a specific application.

Reconfigurable hardware allows designers to change the configuration of the hardware at any time. As other researchers have recognized, this approach provides an excellent alternative for performance, power, flexibility, and fault tolerance [1,4,5]. Users may select between different trade-offs, such as performance versus fault tolerance, depending on the application at hand.

ASICs are optimized for a particular DSP algorithm. These devices can achieve maximum performance and minimum power consumption, but incur high development costs. Due to the cost and limited applicability of an ASIC, this approach may only be feasible for high-volume designs.

The field of reconfigurable devices today includes fine-grain and coarse-grain architectures. Traditional fine-

grain devices such as FPGAs achieve good flexibility [1,6]. However, implementing basic DSP operations, such as multiplication, on an FPGA results in tremendous interconnection overhead and inefficiencies [2]. Recently, researchers have proposed coarse-grain devices that exploit the symmetry and regularity of DSP algorithms. Each cell may contain adders, multipliers, lookup tables, and other functional units [2]. One drawback to these architectures is reduced utilization. A DSP algorithm that requires many multipliers but few lookup tables, for example, would result in many unused functional units. In addition, the fixed number of functional units limits flexibility.

The architecture presented here bridges the gap between fine-grain and coarse-grain reconfigurable devices. In this approach, each cell in the device contains a 4×4 matrix of reconfigurable elements. Each element can act as a lookup table or a small memory. As discussed in the next section, the matrix of elements allows the cell to implement numerous DSP operations.

3. DESIGN DESCRIPTION

The architecture consists of an array of reconfigurable cells, as shown in Figure 1. Each cell performs operations on 4-bit words. Sixteen 4-bit busses connect each cell to its eight neighbors.

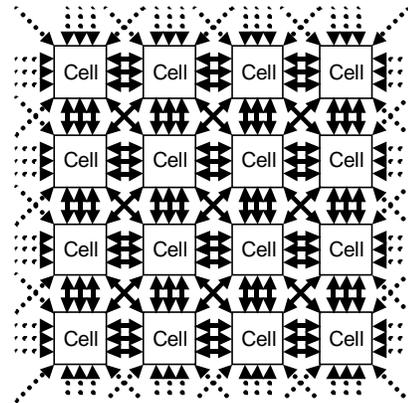


Figure 1. Portion of reconfigurable cell array

Figure 2 depicts the organization of a cell, which contains four main components. The processing core performs the 4-bit operations necessary for DSP computations. The switch routes data between the cell and its neighbors. The interface contains buffers and pipeline latches to improve performance. Finally, the control circuitry buffers the global clock signal and manages the reconfiguration process.

The following subsections present a description of the processing core and the switch in detail.

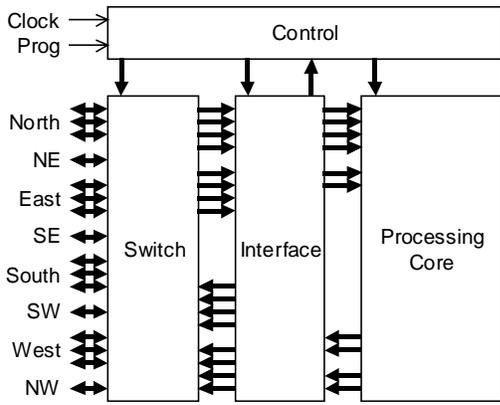


Figure 2. Components of cell

Processing Core

The processing core consists of a 4×4 matrix of reconfigurable elements. Each element is organized as a 16×2-bit random-access memory. The processing core can be configured into two structures: one optimized for memory operations, the other optimized for mathematical functions. Both structures perform one operation per clock cycle.

In memory mode, shown in Figure 3, the matrix of elements implements a 64×8-bit random-access memory. Inputs A, B, C, and D select a 4-bit address in each column. Additional inputs enable one of the columns for reading or writing. For a read operation, the column outputs the selected entry on QR. For a write operation, the column overwrites the selected entry with IJ.

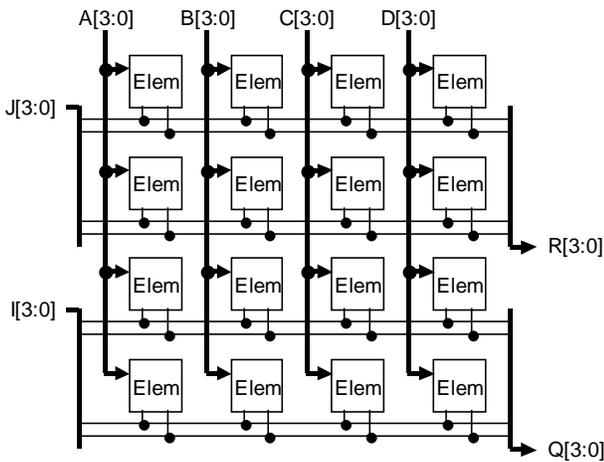


Figure 3. Core in Memory Mode

In mathematics mode, shown in Figure 4, the matrix of elements assumes a structure resembling a carry-save multiplier. However, the elements can perform many functions besides multiplication. The 16×2-bit memory in each element now implements a lookup table for the desired function. Examples of functions are

$$YZ = (A \times B) + C + D$$

$$YZ = A + B - C - D$$

$$Y = (A \text{ and } B) \text{ or } (C \text{ and } D)$$

where Y and Z are 4-bit outputs.

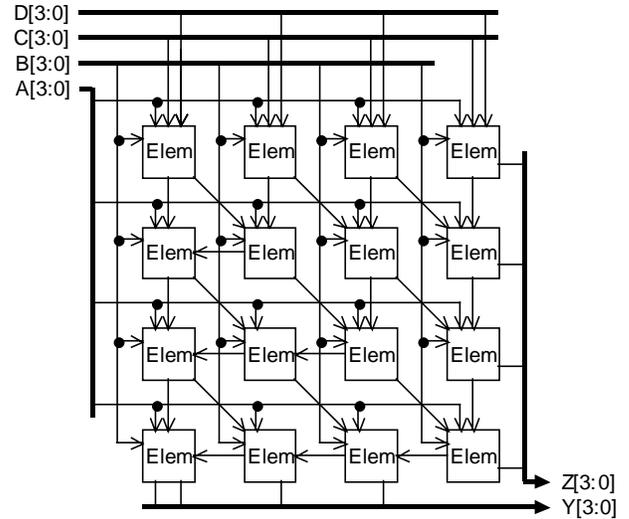


Figure 4. Core in Mathematics Mode

In practice, DSP algorithms manipulate data longer than 4 bits. For this reason, cells can be used as 4-bit building blocks to create larger structures. For example, Figure 5 illustrates an 8-bit multiply-accumulate (MAC) unit. Inputs V1, V2, V3, and V4 are 8 bits wide, while M contains 16 bits. Together, the four cells perform the operation $M = (V1 \times V2) + V3 + V4$.

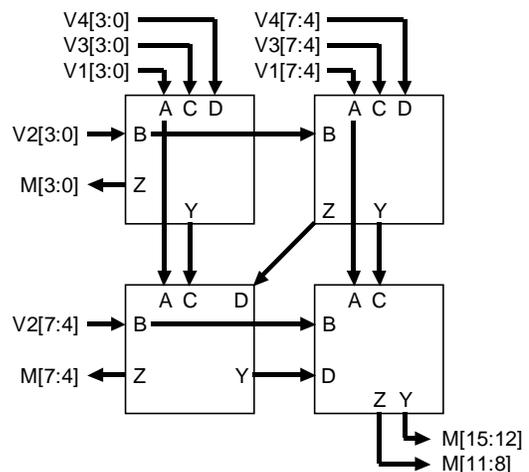


Figure 5. 8-bit Multiply-Accumulate Unit

Switch

Routing data between cells is the purpose of the switch. The switch allows each cell to transfer data to and from its eight neighbors in 4-bit units. There are twelve bi-directional busses connecting cells in the horizontal and

vertical directions, and four additional busses connecting cells diagonally, bringing the total to sixteen.

As shown in Figure 6, the switch is the 16×16, 4-bit crossbar. The component selects a subset of the sixteen data busses to transfer to the interface module. The interface module buffers the data before sending it to the processing core or back through the switch. The output of the processing core travels through the pipeline registers in the interface module before reaching the switch. The switch then routes the data to the appropriate busses.

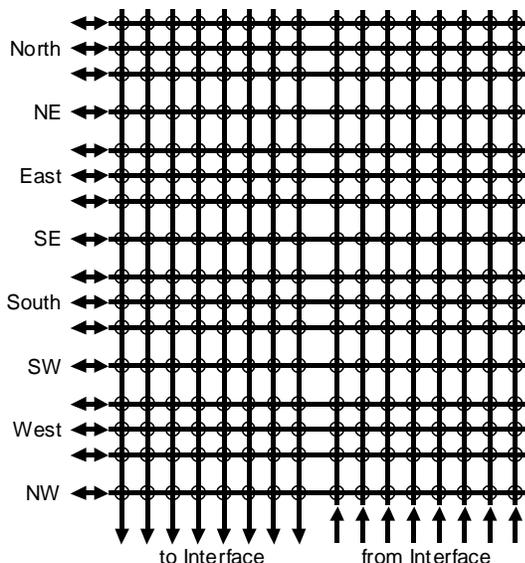


Figure 6. Structure of switch

To reduce total area, each connection in the switch uses a dynamic memory cell to store the state of the link. As illustrated in Figure 7, each link contains four transistors. The stored value is refreshed every 32 clock cycles.

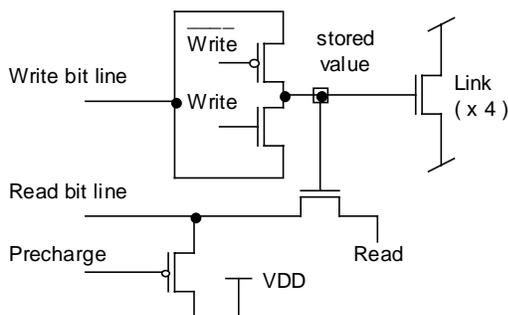


Figure 7. Switch connection

4. FEATURES OF THE DESIGN

The reconfigurable cell array incorporates a number of features that enhance the performance and utility of the design. These features include the following: flexible structures, variable word length, pipeline latches, and

error correction circuitry. The remainder of this section discusses these features.

Flexible Structures

The core of the cell contains a 4×4 matrix of reconfigurable elements that can perform memory operations or calculate mathematical functions. This design combines the flexibility of a fine-grain architecture with the parallelism of a coarse-grain architecture. With suitable configurations, the cell can implement a wide variety of operations, some of which are listed in Table 2.

Table 2. Examples of cell operations

Operation	Options
MAC: $(A \times B) + C + D$	Unsigned or signed, add or subtract
Addition: $A + B + C + D$	Unsigned or signed, add or subtract
Logic: AND, OR, XOR, etc.	Function specified by lookup table
Multiplexing	4-way (4-bit) or 2-way (8-bit)
Lookup table	64-entry×8-bit
State machine	States × inputs = 64
Random-access memory	64-entry×8-bit

Variable Word Length

In coarse-grain architectures, each cell contains a fixed number of functional units. In addition to limiting utilization and flexibility, this approach also incurs unnecessary delays and power consumption. For instance, if the device provides a 16-bit multiplier and the application only requires 8- or 12-bit multiplications, the extra hardware slows down the calculation and requires more power. The medium-grain architecture, on the other hand, implements a multiplier of the desired size.

The choice of 4-bit cells follows standard practice in digital circuit design. Most systems are partitioned into 4-bit building blocks, with buffer circuitry in between. Having larger groupings increases the fan-in and fan-out of the gates, creates signal integrity problems, and slows the datapath. In addition, some DSP algorithms may not require high precision. The variance of the algorithm reported in [7] did not decrease significantly for word lengths larger than 12 bits. Obviously, choosing a word length that optimizes precision requirements versus hardware complexity is an important design goal.

Pipeline Latches

The interface module in each cell contains a set of pipeline latches. The latches allow DSP algorithms to execute in a pipelined, and even superpipelined, fashion. Superpipelining helps increase the system clock rate substantially, as it can be maintained independently of the word length or operation type.

Figure 8 shows a superpipelined implementation of the 8-bit MAC unit. The slashes across a data bus indicate one pipeline latch. Now the operation executes in three

clock cycles, but the system can begin a new MAC operation every clock cycle.

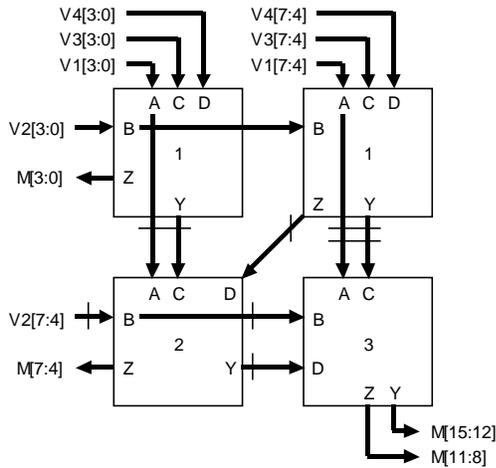


Figure 8. Superpipelined 8-bit Multiply-Accumulate Unit

Error Correction Circuitry

The reconfigurable function array includes an error correction system to protect the memory inside of every element. This error correction is provided by a cross-parity scheme [8], which divides the 16x2-bit memory in an element into two parallel 16-bit units arranged in a 4x4 configuration. The scheme will correct up to one error in each unit per read cycle. Even-parity bits are stored for the rows and columns in both memory units during write operations. When a memory read is performed, the parity bits of the selected rows and columns are XORed with the data in the rows and columns, which allows the detection of one error in each row and column. If an error is detected in both the row and column of the requested data bit, then the system determines that the bit has been corrupted. The data bit is then inverted, sent to the output line, and fed back to the memory. This feedback permanently corrects the erroneous bit. Multiple errors in the same unit can be fixed, as long as they are all separated by at least one read cycle.

Figure 9 shows the hardware organization of the error correction system in one 16-bit unit during a memory read operation. The selected row and column are directed to the XOR logic, where they are combined with the parity bits. The row bits are also sent to a multiplexer, which selects the data bit to be read based on the column address bits. If the row and column are inconsistent with the stored parity bits, then the correction unit inverts the data bit and feeds it back into memory. Otherwise, the correction unit passes the data bit unaltered.

Figure 10 illustrates a memory write operation. The data bit to be written is XORed with the corresponding row and column in the memory unit. The outputs of the XOR operations are then stored as the parity bits for that row and column.

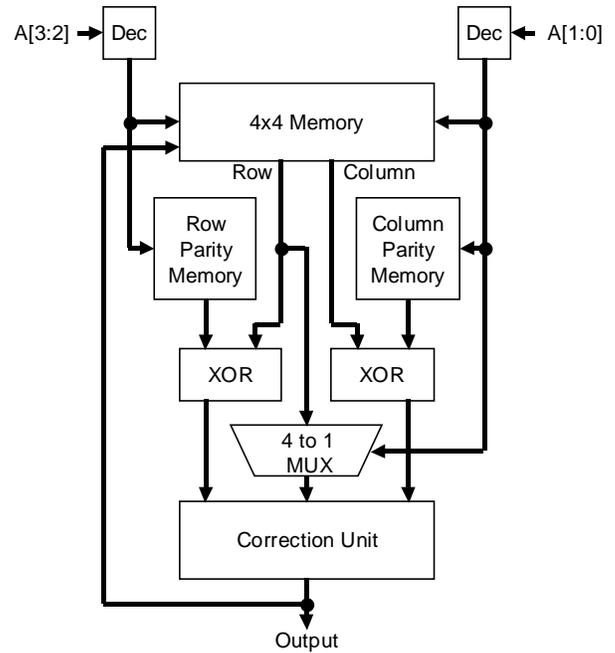


Figure 9. Error correction for memory read

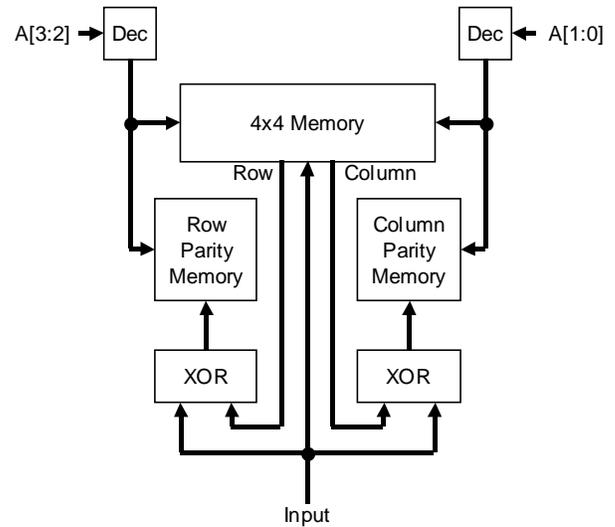


Figure 10. Error correction for memory write

5. PERFORMANCE AND PROTOTYPE

Initial simulations of the reconfigurable cell show that it can operate with a clock period of 10 ns using a modest 0.5- μm CMOS technology. This technology, supported by MOSIS and available for academic purposes, has been chosen to build the first prototype of the cell. Due to the simplicity of the basic cell and the availability of pipeline latches, the system can reach a clock frequency of 100 MHz even with this 0.5- μm technology.

Table 3 shows the hardware requirements and execution times for two common DSP benchmarks: the

Fast Fourier Transform (FFT) and a finite-impulse response (FIR) filter. The benchmarks assume 16-bit fixed-point data. A 64-point complex FFT requires 116 cells and executes in 456 clock cycles. With a 100-MHz clock, the total latency is only 4.56 μ s. The last two rows in the table demonstrate how the designer can trade off hardware for performance. A parallel implementation of the 16-tap FIR filter requires over 5 times as many cells as the serial implementation, but reduces the execution time by a factor of 15.

Table 3. Hardware requirements and execution times for DSP benchmarks

Benchmark	Cells	Cycles
64-point complex FFT	116	456
256-point complex FFT	164	1120
16-tap, 256-point FIR filter (serial)	52	4256
16-tap, 256-point FIR filter (parallel)	296	280

Table 4 compares the performance of the 256-point complex FFT with four commercial DSP processors that operate on 16-bit fixed-point data. As shown, the execution time is comparable with the highest-performance processors today, notwithstanding the prototype 0.5- μ m technology.

Table 4. Execution time of 256-point complex FFT

Processor	Cycles	Frequency	Time
ADSP-2188N*	7423	80 MHz	92.8 μ s
ADSP-21532*	3176	300 MHz	10.6 μ s
TMS320VC5416-160**	8542	160 MHz	53.4 μ s
TMS320VC5502-300**	4786	300 MHz	16.0 μ s
Reconfigurable Cell Array	1120	100 MHz	11.2 μ s

* Source: Analog Devices, www.analog.com

** Source: Texas Instruments, www.ti.com

Figure 11 below depicts one of the prototype chips that will be used to test the architecture. This chip contains the processing core of the cell; two separate chips implement the switch and error correction circuitry. The chips are currently being fabricated by MOSIS; some test results will be presented at the conference.

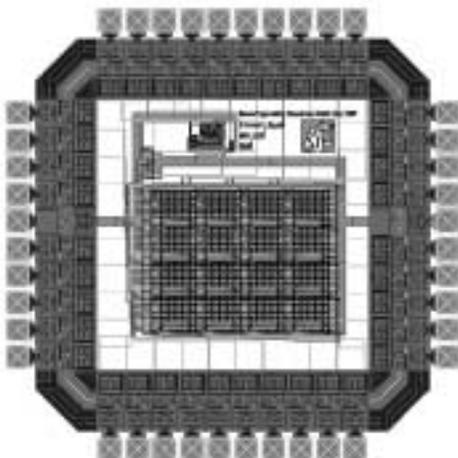


Figure 11. Prototype chip

6. Concluding Remarks

This paper presented the design of a medium-grain reconfigurable device for digital signal processing. The device contains an array of programmable 4-bit cells and interconnection structures. Each cell performs a small portion of the overall algorithm. Cells contain a 4x4 matrix of smaller reconfigurable elements, which may implement mathematical functions or act as a small memory. Cells also contain interconnection circuitry and an error correction scheme.

The reconfigurable cell array has numerous advantages over other implementations, including flexible structures, variable word lengths, pipeline registers, and error correction circuitry. These features enhance performance, flexibility, and reliability: key requirements for most applications. Although the performance of the device may not reach that of custom integrated circuits, the reduced development cost and ease of reuse make the architecture a promising alternative to current implementations. In fact, the execution time for the Fast Fourier Transform, using the modest clock rate of the prototype, compares with the highest-performance DSP processors today.

REFERENCES

- [1] R. Tessier and W. Burlison, Reconfigurable computing for digital signal processing: a survey, in Y. Hu (Ed.) *Programmable digital signal processors* (Marcel Dekker Inc., 2001).
- [2] R. Hartenstein, Coarse grain reconfigurable architectures, *6th Asia and South Pacific Design Automation Conference*, 2001.
- [3] N. Dutt and K. Choi, Configurable processors for embedded computing, *IEEE Computer*, 36(1), 2003, 120-123.
- [4] J. Smit et al, Low cost and fast turnaround: reconfigurable graph-based execution units, *Proc. 7th BELSIGN Workshop*, Enschede, the Netherlands, 1998.
- [5] P. Heysters et al, A reconfigurable function array architecture for 3G and 4G wireless terminals, *Proc. World Wireless Congress*, San Francisco, USA, 2002, 399-405.
- [6] C. Dick, B. Turney, and A. M. Reza, Configurable logic for digital signal processing, 1999. Available at: http://www.xilinx.com/products/logiccore/dsp/config_logic4_99.pdf
- [7] E. Pauer et al, Environment for implementing DSP algorithms in reconfigurable hardware, *Proc. High Performance Embedded Computing Workshop (HPEC)*, 2000.
- [8] M. Pflanz, H. T. Vierhaus, and K. Walther, On-line error detection and correction in storage elements with cross-parity check, *Proc. Eighth IEEE Int. On-Line Testing Workshop*, 2002, 69-73.