Journal of VLSI Signal Processing 28, 7–27, 2001

Reconfigurable Computing for Digital Signal Processing: A Survey*

RUSSELL TESSIER AND WAYNE BURLESON
Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003, USA

Received July 1999; Revised December 1999

Abstract. Steady advances in VLSI technology and design tools have extensively expanded the application domain of digital signal processing over the past decade. While application-specific integrated circuits (ASICs) and programmable digital signal processors (PDSPs) remain the implementation mechanisms of choice for many DSP applications, increasingly new system implementations based on *reconfigurable computing* are being considered. These flexible platforms, which offer the functional efficiency of hardware and the programmability of software, are quickly maturing as the logic capacity of programmable devices follows Moore's Law and advanced automated design techniques become available. As initial reconfigurable technologies have emerged, new academic and commercial efforts have been initiated to support power optimization, cost reduction, and enhanced run-time performance.

This paper presents a survey of academic research and commercial development in reconfigurable computing for DSP systems over the past fifteen years. This work is placed in the context of other available DSP implementation media including ASICs and PDSPs to fully document the range of design choices available to system engineers. It is shown that while contemporary reconfigurable computing can be applied to a variety of DSP applications including video, audio, speech, and control, much work remains to realize its full potential. While individual implementations of PDSP, ASIC, and reconfigurable resources each offer distinct advantages, it is likely that integrated combinations of these technologies will provide more complete solutions.

Keywords: signal processing, reconfigurable computing, FPGA, survey

1. Introduction

Throughout the history of computing, digital signal processing applications have pushed the limits of compute power, especially in terms of real-time computation. While processed signals have broadly ranged from media-driven speech, audio, and video waveforms to specialized radar and sonar data, most calculations performed by signal processing systems have exhibited the same basic computational characteristics. The inherent data parallelism found in many DSP functions has made DSP algorithms ideal candidates for hardware implementation, leveraging expanding VLSI capabilities. Recently, DSP has received increased attention due to rapid advancements in multimedia computing and high-speed wired and wireless communications. In response to these advances, the search for novel implementations of arithmetic-intensive circuitry has intensified.

While application areas span a broad spectrum, the basic computational parameters of most DSP operations remain the same: a need for real-time performance within the given operational parameters of a target system and, in most cases, a need to adapt to changing data sets and computing conditions. In general, the goal of high performance in systems ranging from low-cost embedded radio components to special-purpose ground-based radar centers has driven the development of application and domain-specific chip sets. The development and financial cost of this approach is often large, motivating the need for new

*This article is abridged from the forthcoming Marcel Dekker, Inc. publication, *Programmable Digital Signal Processors*, Y. Hu, editor.

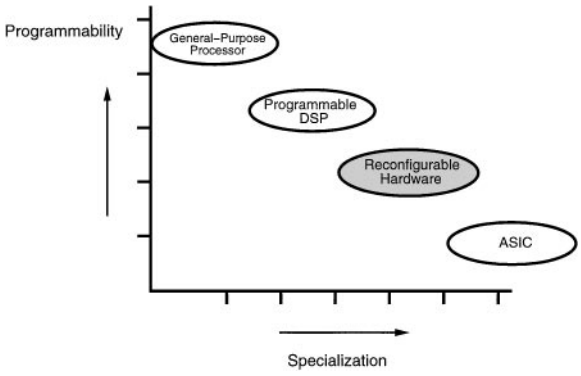

Figure 1. DSP implementation spectrum.

approaches to computer architecture that offer the same computational attributes as fixed-functionality architectures in a package that can be customized in the field. The second goal of system adaptability is generally addressed through the use of software-programmable, commodity digital signal processors. While these platforms enable flexible deployment due to software development tools and great economies of scale, application designers and compilers must customize their processing approach to available computing resources. This flexibility often comes at the cost of performance and power efficiency.

As shown in Fig. 1, *reconfigurable* computers offer a compromise between the performance advantages of fixed-functionality hardware and the flexibility of software-programmable substrates. Like ASICs, these systems are distinguished by their ability to directly implement specialized circuitry directly in hardware. Additionally, like programmable processors, reconfigurable computers contain functional resources that may be easily modified after field deployment in response to changing operational parameters and data sets. To date, the core processing element of most reconfigurable computers has been the field-programmable gate array (FPGA). These bit-programmable computing devices offer ample quantities of logic and register resources that can easily be adapted to support the fine-grained parallelism of many pipelined DSP applications. With current logic capacities exceeding one million gates per device, substantial logic functionality can be implemented on each programmable device. While appropriate for some classes of implementations, FPGAs represent only one possible implementation in a range of possible reconfigurable computing building blocks. A number of reconfigurable alternatives are presently under evaluation in academic and commercial environments.

In this survey, the evolution of reconfigurable computing with regard to digital signal processing is considered. This study includes a historical evaluation of reprogrammable architectures and programming environments used to support DSP applications. The chronology is supported with specific case studies which illustrate approaches used to address implementation constraints such as system cost, performance, and power consumption. It is seen that as technology has progressed, the richness of applications supported by reconfigurable computing and the performance of reconfigurable computing platforms have improved dramatically. Reconfigurable computing for DSP remains an active area of research as the need for integration with more traditional DSP technologies such as PDSPs becomes apparent and the goal of automated high-level compilation for DSP increases in importance.

The organization of this paper is as follows. In Section 2 a brief history of the issues and techniques involved in the design and implementation of DSP systems is described. Section 3 presents a short history of reconfigurable computing. Section 4 describes why reconfigurable computing is a promising approach for DSP systems. Section 5 serves as the centerpiece of the paper and provides a history of the application of various reconfigurable computing technologies to DSP systems and a discussion of the current state-of-the-art. We conclude in Section 6 with some predictions about the future of reconfigurable computing for digital signal processing. These predictions are formulated by extrapolating the trends of reconfigurable technologies and describing future DSP applications which may be targeted to reconfigurable hardware.

1.1. Definitions

The following definitions are used to describe various attributes related to reconfigurable computing:

- *Reconfigurable* or *Adaptive*—In the context of reconfigurable computing this term indicates that the logic functionality and interconnect of a computing system or device can be customized to suit a specific application through post-fabrication, user-defined programming.
- *Run-time (or Dynamically) Reconfigurable*—System logic functionality and/or interconnect

connectivity can be modified during application execution. This modification may be either data-driven or statically-scheduled.
- *Fine-grained parallelism*—Logic functionality and interconnect connectivity is programmable at the bit level. Resources encompassing multiple logic bits may be combined to form parallel functional units.
- *Specialization*—Logic functionality can be customized to perform exactly the operation desired. An example is the synthesis of filtering hardware with a fixed constant value.

2. Background in DSP Implementation

2.1. DSP System Implementation Choices

Since the early 1960's three goals have driven the development of DSP implementations: *1. data parallelism*, *2. application-specific specialization*, and *3. functional flexibility*. In general, design decisions regarding DSP system implementation require tradeoffs between these three system goals. As a result, a wide variety of specialized hardware implementations and associated design tools have been developed for DSP including associative processing, bit-serial processing, on-line arithmetic, and systolic processing. As implementation technologies have become available, these basic approaches have matured to meet the needs of application designers.

As shown in Table 1, various cost metrics have been developed to compare the quality of different DSP implementations. Performance has frequently been the most critical system requirement since DSP systems often have demanding real-time constraints. In the past two decades, however, cost has become more significant as DSP has migrated from predominantly military and scientific applications into numerous low-cost consumer applications. Over the past ten years, energy consumption has become an important measure as DSP techniques have been widely applied in portable, battery-operated systems such as cell-phones, CD players, and laptops [1]. Finally, flexibility has emerged as one of the key differentiators in DSP implementations since it allows changes to system functionality at various points in the design life cycle. The results of these cost tradeoffs have resulted in four primary implementation options including application-specific integrated circuits (ASICs), programmable digital signal processors (PDSPs), general-purpose microprocessors, and reconfigurable hardware. Each implementation option presents different trade-offs in terms of performance, cost, power and flexibility.

For many specialized DSP applications, system implementation must include one or more application-specific integrated circuits to meet performance and power constraints. Even though ASIC design cycles remain long, a trend toward automated synthesis and verification tools [2] is simplifying high-level ASIC design. Since most ASIC specification is done at the behavioral or register-transfer level, the functionality and performance of ASICs have become easier to represent and verify. Another, perhaps more important, trend has been the use of pre-designed cores with well-defined functionality. Some of these cores are in fact PDSPs or RISC microcontrollers, for which software has to be written and then stored on-chip. ASICs have a significant advantage in area and power and for many high-volume designs the cost-per-gate for a given performance level is less than that of high-speed commodity FPGAs. These characteristics are especially important for power-aware functions in mobile communication and remote sensing. Unfortunately, the fixed nature of ASICs limit their *reconfigurability*. For designs that must adapt to changing data sets and operating conditions, software-programmable components must be included in the target system, reducing available parallelism. Additionally, for low-volume or prototype implementations, the NRE costs related to an ASIC may not justify its improved performance benefits.

Table 1. DSP implementation comparison.

	Performance	Cost	Power	Flexibility	Design effort (NRE)
ASIC	High	High	Low	Low	High
Programmable DSP	Medium	Medium	Medium	Medium	Medium
General-purpose processor	Low	Low	Medium	High	Low
Reconfigurable hardware	Medium	Medium	High	High	Medium

The application domain of programmable digital signal processors can be identified by tracing their development lineage. Thorough summaries of programmable DSPs can be found in [3–5] and [6]. In the 1980's, the first programmable digital signal processors (PDSPs) were introduced by Texas Instruments. These initial processor architectures were primarily CISC pipelines augmented with a handful of special architectural features and instructions to support filtering and transform computations. One of the most significant changes to second generation PDSPs was the adaptation of the Harvard architecture, effectively separating the program bus from the data bus. This optimization reduced the von Neumann bottleneck, thus providing an unimpeded path for data from local memory to the processor pipeline. Many early DSPs allowed programs to be stored in on-chip ROM and supported the ability to make off-chip accesses if instruction capacity was exceeded. Some DSPs also had coefficient ROMs, again recognizing the opportunity to exploit the relatively static nature of filter and transform coefficients.

Contemporary digital signal processors are highly programmable resources that offer the capability for in-field update as processing standards change. Parallelism in most PDSPs is not extensive but generally consists of overlapped data fetch, data operation, and address calculation. Some instruction set modifications are also used in PDSPs to specialize for signal processing. Addressing modes are provided to simplify the implementation of filters and transforms and, in general, control overhead for loops is minimized. Arithmetic instructions for fixed point computation allow saturating arithmetic which is important to avoid overflow exceptions or oscillations. New hybrid DSPs contain a variety of processing and I/O features including parallel processing interfaces, VLIW function unit scheduling, and flexible data paths. Through the addition of numerous, special-purpose memories, on-chip DSPs can now achieve high-bandwidth and, to a moderate extent, reconfigurable interconnect. Due to the volume usage of these parts, costs are reduced and commonly-used interfaces can be included. In addition to these benefits, the use of a DSP has specific limitations. In general, for optimal performance, applications must be written to utilize the resources available in the DSP. While high-level compilation systems which perform this function are becoming available [7, 8], often it is difficult to get exactly the mapping desired. Additionally, the interface to memory may not be appropriate for specific applications creating a bandwidth bottleneck in getting data to functional units.

The 1990's have been characterized by the introduction of DSP to the mass commercial market. DSP has made the transition from a fairly academic acronym to one seen widely in advertisements for consumer electronics and software packages. A battle over the DSP market has ensued primarily between PDSP manufacturers, ASIC vendors, and developers of two types of general-purpose processors, desk-top microprocessors and high-end microcontrollers. General-purpose processors, such as the Intel Pentium, can provide much of the signal processing needed for desk-top applications such as audio and video processing, especially since the host microprocessor is already resident in the system and has highly optimized I/O and extensive software development tools. But general-purpose desk-top processors are not a realistic alternative for embedded systems due to their cost and lack of power efficiency in implementing DSP. Another category of general-purpose processors is the high-end microcontroller. These chips have also made inroads into DSP applications by presenting system designers with straightforward implementation solutions that have useful data interfaces and significant application-level flexibility.

One DSP hardware implementation compromise that has developed recently has been the development of domain-specific standard products in both programmable and ASIC formats. The PDSP community has determined that since certain applications have high volume it is worthwhile to tailor particular PDSPs to domain-specific markets. This has led to the availability of inexpensive, commodity silicon while allowing users to provide application differentiation in software. ASICs have also been developed for more general functions like MPEG decoding in which standards have been set up to allow a large number of applications to use the same basic function.

Reconfigurable computing platforms for DSP offer an intermediate solution to ASICs, PDSPs, and general and domain-specific processors by allowing reconfigurable and specialized performance on a per-application basis. While this emerging technology has primarily been applied to experimental rather than commercial systems, the application-level potential for these reconfigurable platforms is great. Following an examination of the needs of contemporary DSP applications, current trends in the application of reconfigurable computing to DSP are explored.

2.2. The Changing World of DSP Applications

Over the past thirty years the application space of digital signal processing has changed substantially, motivating new systems in the area of reconfigurable computing. New applications over this time span have changed the definition of DSP and have created new and different requirements for implementation. For example, in today's market, DSP is often found in human-computer interfaces such as sound cards, video cards, and speech recognition systems; application areas with limited practical significance just a decade ago. Since a human is an integral part of these systems, different processing requirements can be found, in contrast to communications front-ends such as those found in DSL modems from Broadcom [9] or CDMA receiver chips from Qualcomm [10]. Another large recent application of DSP has been in the read circuitry of hard-drive and CD/DVD storage systems [11]. Although many of the DSP algorithms are the same as in modems, the system constraints are quite different.

Consumer products now make extensive use of DSP in low-cost and low-power implementations [12]. Wireless and multimedia, two of the hottest topics in consumer electronics, both rely heavily on DSP implementation. Cellular telephones, both GSM and CDMA, are currently largely enabled by custom silicon [13], although trends toward other implementation media such as PDSPs are growing. Modems for DSL, cable, LANs and most recently wireless, all rely on sophisticated adaptive equalizers and receivers. Satellite set-top boxes rely on DSP for satellite reception using channel decoding as well as an MPEG decoder ASIC for video decompression. After the set-top box, the DVD player has now emerged as the fastest growing consumer electronics product. The DVD player relies on DSP to avoid intersymbol interference, allowing more bits to be packed into a given area of disk. In the commercial video market, digital cameras and camcorders are rapidly becoming affordable alternatives to traditional analog cameras, largely supported by photo-editing, authoring software, and the Web.

Development of a large set of DSP systems has been driven indirectly by the growth of consumer electronics. These systems include switching stations for cellular, terrestrial, satellite and cable infrastructure as well as cameras, authoring studios, and encoders used for content production. New military and scientific applications applied to the digital battlefield, including advanced weapons systems and remote sensing equipment, all rely on DSP implementation that must operate reliably in adverse and resource-limited environments. While existing DSP implementation choices are suitable for all of these consumer and military-driven applications, higher performance, efficiency, and flexibility will be needed in the future, driving current interest in reconfigurable solutions.

In all of these applications, data processing is considerably more sophisticated than the traditional filters and transforms which characterized DSP of the 1960's and 1970's. In general, performance has grown in importance as data rates have increased and algorithms have become more complex. Additionally, there is an increasing demand for flexible and diverse functionality based on environmental conditions and workloads. Power and cost are equally important since they are critical to overall system cost and performance.

While new approaches to application-specific DSP implementation have been developed by the research community in recent years, their application in practice has been limited by the market domination of PDSPs and the reluctance of designers to expose schedule and risk-sensitive ASIC projects to non-traditional design approaches. Recently, however, the combination of new design tools and the increasing use of intellectual property cores [14] in DSP implementations have allowed some of these ideas to find wider use. These implementation choices include systolic architectures, alternative arithmetic (RNS, LNS, digital-serial), wordlength optimization, parallelizing transformations, memory partitioning and power optimization techniques. Design tools have also been proposed which could close the gap between software development and hardware development for future hybrid DSP implementations. In subsequent sections it will be seen that these tools will be helpful in defining appropriate application of reconfigurable hardware to existing challenges in DSP. In many cases, basic design techniques used to develop ASICs or domain-specific devices can be re-applied to customize applications in programmable silicon by taking the limitations of the implementation technology into account.

3. A Brief History of Reconfigurable Computing

Since their introduction in the mid-1980's field-programmable gate arrays (FPGAs) have been the subject of extensive research and experimentation. In this section, reconfigurable device architecture and system integration is investigated with an eye towards

identifying trends likely to affect future development. While this summary provides sufficient background to evaluate the impact of reconfigurable hardware on DSP more thorough discussions of FPGAs and reconfigurable computing can be found in [15–17] and [18].

3.1. Field-Programmable Devices

The modern era of reconfigurable computing was ushered in by the introduction of the first commercial SRAM-based FPGAs by Xilinx Corporation [19] in 1986. These early reprogrammable devices and subsequent offerings from both Xilinx and Altera Corporation [20] contain a collection of fine-grained programmable logic blocks interconnected via wires and programmable switches. Logic functionality for each block is specified via a small programmable memory, called a *lookup table*, driven by a limited number of inputs (typically less than five) which generates a single boolean output. Additionally, each logic block typically contains one or more flip flops for fine-grained storage. While early FPGA architectures contained small numbers of logic blocks (typically less than 100), new device families have quickly grown to capacities of tens of thousands of lookup tables containing millions of gates of logic. As shown in Fig. 2, fine-grained lookup table/flip flop pairs are frequently grouped into tightly-connected coarse-grained blocks to take advantage of circuit locality. Interconnection between logic blocks is provided via a series of wire segments located in channels between the blocks. Programmable pass transistors and multiplexers can be used to provide both block-to-segment connectivity and segment-to-segment connections.

Much of the recent interest in reconfigurable computing has been spurred by the development and maturation of field-programmable gate arrays. The recent development of systems based on FPGAs has been greatly enhanced by an exponential growth rate in the gate capacity of reconfigurable devices and improved device performance due to shrinking die sizes and enhanced fabrication techniques. As shown in Fig. 3, reported gate counts [21–23] for LUT-based FPGAs, from companies such as Xilinx Corporation, have roughly followed Moore's Law over the past decade.[1] This increase in capacity has enabled complex structures such as multi-tap filters and small RISC processors to be implemented directly in a single FPGA chip. Over this same time period the system performance of these devices has also improved exponentially. While in the mid-1980's system-level FPGA performance of 2–5 MHz was considered acceptable, today's LUT-based FPGA designs frequently approach performance levels of 60 MHz and beyond. Given the programmable nature of reconfigurable devices, the performance penalty of a circuit implemented in reprogrammable technology versus a direct ASIC implementation is generally on the order of a factor of five to ten.

3.2. Early Reprogrammable Systems

Soon after the commercial introduction of the FPGA, computer architects began devising approaches for

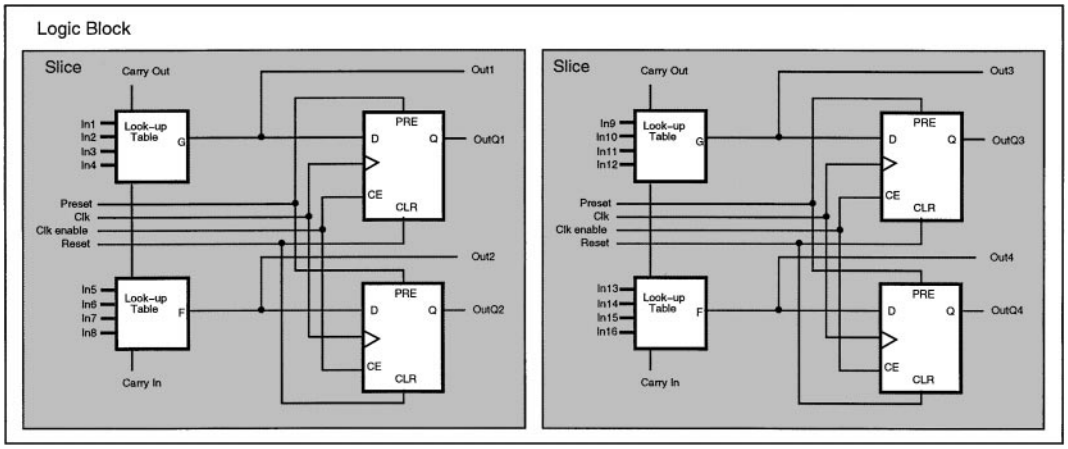

Figure 2. Simplified Xilinx Virtex logic block [23]. Each logic block consists of two 2-LUT slices.

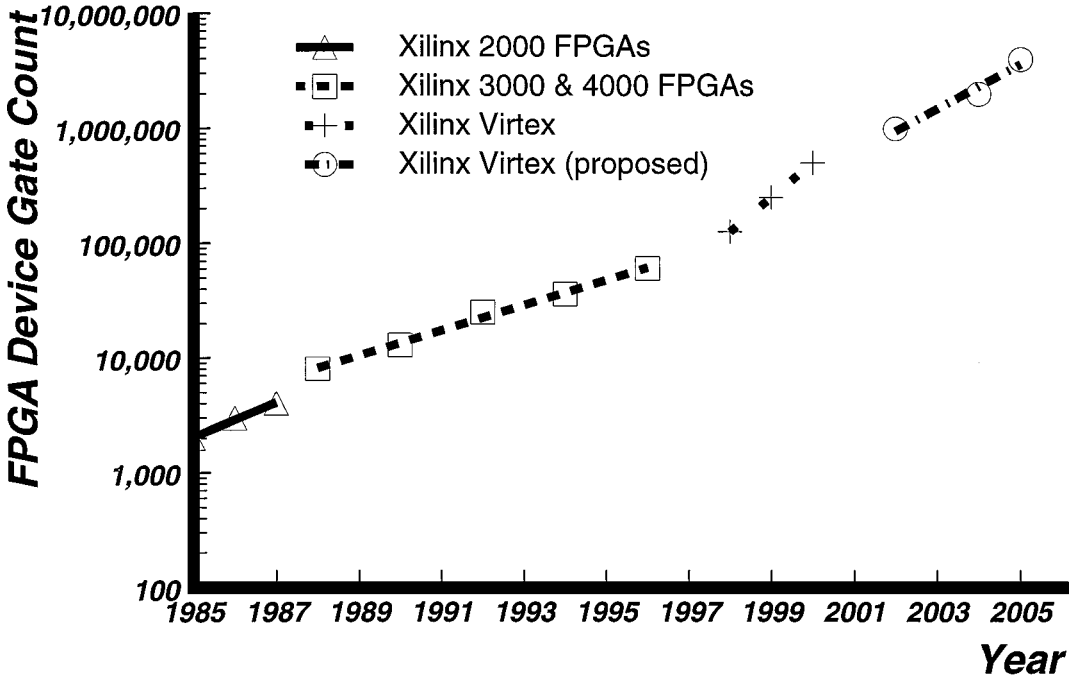

Figure 3. Growth of FPGA gate capacity.

leveraging new programmable technology in computing systems. As summarized in [18], the evolution of reconfigurable computing was significantly shaped by two influential projects: Splash II [24] and Programmable Active Memories (PAM) [25]. Each of these projects addressed important programmable system issues regarding programming environment, user interface, and configuration management by applying pre-existing computational models in the areas of special-purpose coprocessing and statically-scheduled communication to reconfigurable computing.

Splash II is a multi-FPGA parallel computer which uses orchestrated systolic communication to perform inter-FPGA data transfer. Each board of a multi-board Splash II system contains 16 Xilinx XC4000 series FPGA processors, each with associated SRAM. Unlike its multi-FPGA predecessor, Splash [26], which was limited to strictly near-neighbor systolic communication, each Splash II board contains inter-FPGA crossbars for multi-hop data transfer and broadcast. Software development for the system typically involves the creation of VHDL circuit descriptions for individual systolic processors. These designs must meet size and performance constraints of the target FPGAs. Following processor creation, high-level inter-FPGA scheduling software is used to ensure that system-wide communication is synchronized. In general, the system is not dynamically reconfigured during operation. For applications with SIMD characteristics, a compiler [27] has been created to automatically partition processing across FPGAs and to synchronize interfaces to local SRAMs. Numerous DSP applications have been mapped to Splash II, including audio and video algorithm implementations. These applications are described in greater detail in Section 5.

Programmable Active Memory DECPeRLe-1 systems [25] contain arrangements of FPGA processors in a two-dimensional mesh with memory devices aligned along the array perimeter. PAMs were designed to create the architectural appearance of a functional memory for a host microprocessor and the PAM programming environment reflects this. From a programming standpoint the multi-FPGA PAM can be accessed like a memory through an interface FPGA with written values treated as inputs and read values used as results. Designs are generally targeted to PAMs through handcrafting of design sub-tasks, each appropriately sized to fit on an FPGA. The PAM array and its successor, the Pamette [28], are interfaced to a host workstation through a backplane bus. Additional discussion of PAMs with regard to DSP applications appears in Section 5.

3.3. Reconfigurable Computing Research Directions

An important aspect of reconfigurable devices is the ability to reconfigure functionality in response to changing operating conditions and application data sets. While SRAM-based FPGAs have supported slow millisecond reconfiguration rates for some time, only recently have devices been created that allow for rapid device reconfiguration at run-time. Dynamically-reconfigurable FPGAs, or *DPGAs* [29, 30], contain multiple interconnect and logic configurations for each programmable location in a reconfigurable device. Often these architectures are designed to allow configuration switching in a small number of system clock cycles measuring nanoseconds rather than milliseconds. While several DPGA devices have been developed in research environments, none are currently commercially available due the large overhead costs associated with the required large configuration memory. To promote reconfiguration at lower hardware cost, several commercial FPGA families [23, 31] have been introduced that allow for fast, partial reconfiguration of FPGA functionality from off-chip memory resources. A significant challenge to the use of these reconfigurables is the development of compilation software which will partition and schedule the order in which computation will take place and will determine which circuitry must be changed. While some preliminary work in this area has been completed [32, 33], more advanced tools are needed to fully leverage the new hardware technology. Other software approaches that have been applied to dynamic reconfiguration include the definition of hardware subroutines [34] and the dynamic reconfiguration of instruction sets [35].

While high-level compilation for microprocessors has been an active research area for decades, development of compilation technology for reconfigurable computing is still in its infancy. The compilation process for FPGA-based systems is often complicated by a lack of identifiable coarse-grained structure in fine-grained FPGAs and the dispersal of logic resources across many pin-limited reconfigurable devices on a single computing platform. In particular, since most reconfigurable computers contain multiple programmable devices, design *partitioning* forms an important aspect of most compilation systems. Several compilation systems for reconfigurable hardware [36, 37] have followed a traditional multi-device ASIC design flow involving pin-constrained device partitioning and individual device synthesis using RTL compilation. To overcome pin limitations and achieve full logic utilization on a per-device basis using this approach, either excessive internal device interconnect [36] or I/O counts [38] have been needed. In [39], a hardware virtualization approach is outlined that promotes high per-device logic utilization. Following design partitioning and placement, inter-FPGA wires are scheduled on inter-device wires at compiler-determined time slices, allowing pipelining of communication. Inter-device pipelining also forms the basis of several FPGA system compilation approaches that start at the behavioral level. A high-level synthesis technique described in [40] outlines inter-FPGA scheduling at the RTL level. In [41] and [42] functional allocation is performed that takes into account the amount of logic available in the target system and available inter-device interconnect. Combined communication *and* functional resource scheduling is then performed to fully utilize available logic and communication resources. In [43], inter-FPGA communication and FPGA-memory communication are virtualized since it is recognized that memory rather than inter-FPGA bandwidth is frequently the critical resource in reconfigurable systems.

4. The Promise of Reconfigurable Computing for DSP

Many of the motivations and goals of reconfigurable computing are consistent with the needs of signal processing applications. It will be seen in Section 6 that the deployment of DSP algorithms on reconfigurable hardware has aided in the advancement of both fields over the past fifteen years. In general, the direct benefits of the reconfigurable approach for DSP can be summarized in three critical areas: functional specialization, platform reconfigurability, and fine-grained parallelism.

4.1. Specialization

As stated in Section 2.1, programmable digital signal processors are optimized to deliver efficient performance across a set of signal processing tasks. While the specific implementation of tasks can be modified though instruction-configurable software, applications must frequently be customized to meet specific processor architectural aspects, often at the cost of performance. Currently, most DSPs remain inherently

sequential machines, although some parallel VLIW and multi-function unit DSPs have recently been developed [44]. The use of reconfigurable hardware has numerous advantages for many signal processing systems. For many applications, such as digital filtering, it is possible to customize irregular datapath widths and specific constant values directly in hardware, reducing implementation area and power and improving algorithm performance. Additionally, if standards change, the modifications can quickly be reimplemented in hardware without expensive NRE costs. Since reconfigurable devices contain SRAM-controlled logic and interconnect switches, application programs in the form of device configuration data can be downloaded on a per-application basis. Effectively, this single, wide program instruction defines hardware behavior. Contemporary reconfigurable computing devices have little or no NRE cost since off-the-shelf development tools are used for design synthesis and layout. While reconfigurable implementations may exhibit a 5 to 10× performance reduction compared to the same circuit implemented in custom logic, limited manual intervention is generally needed to map a design to a reconfigurable device. In contrast, substantial NRE costs require ASIC designers to focus on high-speed physical implementation often involving hand-tuned physical layout and near-exhaustive design verification. Time-consuming ASIC implementation tasks can also lead to longer time-to-market windows and increased inventory, effectively becoming the critical path link in the system design chain.

4.2. Reconfigurability

Most reconfigurable devices and systems contain SRAM-programmable memory to allow full logic and interconnect reconfiguration in the field. Despite a wide range of system characteristics, most DSP systems have a need for configurability under a variety of constraints. These constraints include environmental factors such as changes in statistics of signals and noise, channel, weather, transmission rates, and communication standards. While factors such as data traffic and interference often change quite rapidly, other factors such as location and weather change relatively slowly. Still other factors regarding communication standards vary infrequently across time and geography limiting the need for rapid reconfiguration. Some specific ways that DSP can directly benefit from hardware reconfiguration to support these factors include:

- *Field customization*—The reconfigurability of programmable devices allows periodic updates of product functionality as advanced vendor firmware versions become available or product defects are detected. Field customization is particularly important in the face of changing standards and communication protocols. Unlike ASIC implementations, reconfigurable hardware solutions can generally be quickly updated based on application demands without the need for manual field upgrades or hardware swaps.
- *Slow adaptation*—Signal processing systems based on reconfigurable logic may need to be periodically updated in the course of daily operation based on a variety of constraints. These include issues such as variable weather and operating parameters for mobile communication and support for multiple, time-varying standards in stationary receivers.
- *Fast adaptation*—Many communication processing protocols [45] require nearly constant re-evaluation of operating parameters and can benefit from rapid reset of computing parameters. Some of these issues include adaptation to time-varying noise in communication channels, adaptation to network congestion in network configurations, and speculative computation based on changing data sets.

4.3. Parallelism

An abundance of programmable logic facilitates the creation of numerous functional units directly in hardware. Many characteristics of FPGA devices, in particular, make them especially attractive for use in digital signal processing systems. The fine-grained parallelism found in these devices is well-matched to the high-sample rates and distributed computation often required of signal processing applications in areas such as image, audio, and speech processing. Plentiful FPGA flip flops and a desire to achieve accelerated system clock rates have led designers to focus on heavily pipelined implementations of functional blocks and inter-block communication. Given the highly pipelined and parallel nature of many DSP tasks, such as image and speech processing, these implementations have exhibited substantially better performance than standard PDSPs. In general, these systems have been implemented using both task and functional unit pipelining. Many DSP systems have featured bit-serial functional unit implementations [46] and systolic inter-unit communication [24] that can take advantage of the synchronization resources of contemporary FPGAs

without the need for software instruction fetch and decode circuitry. As detailed in Section 5, bit-serial implementations have been particularly attractive due to their reduced implementation area. As reconfigurable devices increase in size, however, more nibble-serial and parallel implementations of functional units have emerged in an effort to take advantage of data parallelism.

Recent additions to reconfigurable architectures have aided their suitability for signal processing. Several recent architectures [23, 47] have included 2-4K bit SRAM banks that can be used to store small amounts of intermediate data. This allows for parallel access to data for distributed computation. Another important addition to reconfigurable architectures has been the capability to rapidly change only small portions of device configuration without disturbing existing device behavior. This feature has recently been leveraged to help adapt signal processing systems to reduce power [48]. The speed of adaptation may vary depending on the specific signal processing application area.

5. History of Reconfigurable Computing and DSP

Since the appearance of the first reconfigurable computing systems, DSP applications have served as important test cases in reconfigurable architecture and software development. In this section a wide range of DSP design approaches and applications that have been mapped to functioning reconfigurable computing systems are considered. Unless otherwise stated, the design of complete DSP systems is stressed including I/O, memory interfacing, high-level compilation and real-time issues rather than the mapping of individual benchmark circuits. For this reason, a large number of FPGA implementations of basic DSP functions like filters and transforms that have not been implemented directly in system hardware have been omitted. While our consideration of the history of DSP and reconfigurable computing is roughly chronological, some noted recent trends were initially investigated a number of years ago. To trace these trends, recent advancements are directly contrasted with early contributions.

5.1. FPGA Implementation of Arithmetic

Soon after the introduction of the FPGA in the mid-1980's an interest developed in using the devices for DSP, especially for digital filtering which can take advantage of specialized constants embedded in hardware. Since a large portion of most filtering approaches involves the use of multiplication, efficient multiplier implementations in both fixed- and floating-point were of particular interest. Many early FPGA multiplier implementations used circuit structures adapted from the early days of LSI development and reflected the restricted circuit area available in initial FPGA devices [42]. As FPGA capacities have increased, the diversity of multiplier implementations has grown.

Since the introduction of the FPGA, bit-serial arithmetic has been used extensively to implement FPGA multiplication. As shown in Fig. 4, taken from [42], bit-serial multiplication is implemented using a linear systolic array that is well-suited to the fine-grained nature of FPGAs. Two data values are input into the multiplier including a parallel value in which all bits are input simultaneously and a sequential value in which values are input serially. In general, a data sampling rate of one value every M clock cycles can be supported where M is the input word length. Each cell in the systolic array is typically implemented using one to four logic blocks similar to the one shown in Fig. 2.

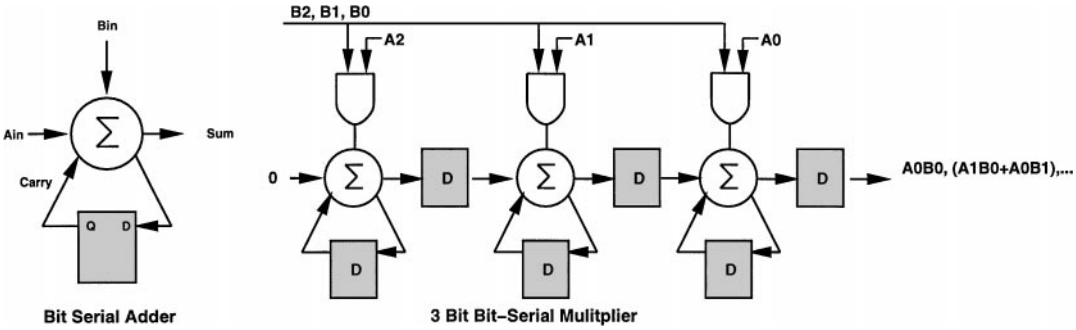

Figure 4. Bit-serial adder and multiplier [42].

Bit-serial approaches have the advantage that communication demands are independent of word length. As a result, low-capacity FPGAs can efficiently implement them. Given their pipelined nature, bit-serial multipliers implemented in FPGAs typically possess excellent area-time products. Many bit-serial formulations have been applied to finite impulse response filtering [49]. Special-purpose bit-serial implementations have included canonic signed digit [50] and power-of-two sum or difference [51].

Given the dual use of lookup tables as small memories, distributed arithmetic (DA) has also been an effective implementation choice for LUT-based FPGAs. Since it is possible to group multiple LUTs together into a larger fanout memory, large LUTs for DA can easily be created. In general, distributed arithmetic requires the embedding of a fixed-input constant value in hardware, thus allowing the efficient pre-computation of all possible dot-product outputs. An example of a distributed arithmetic multiplier, taken from [42], appears in Fig. 5. It can be seen that a fast adder can be used to sum partial products based on nibble lookup. In some cases it may be effective to implement the LUTs as RAMs so new constants can be written during execution of the program.

To promote improved performance, several parallel arithmetic implementations on FPGAs have been formulated [42]. In general, parallel multipliers implemented in LUT-based FPGAs achieve a speedup of 6X in performance when compared to their bit-serial counterparts with an area penalty of 2.5X. Specific parallel implementations of multipliers include a carry-save implementation [52], a systolic-array with cordic arithmetic [52], and pipelined parallel [49, 54, 55].

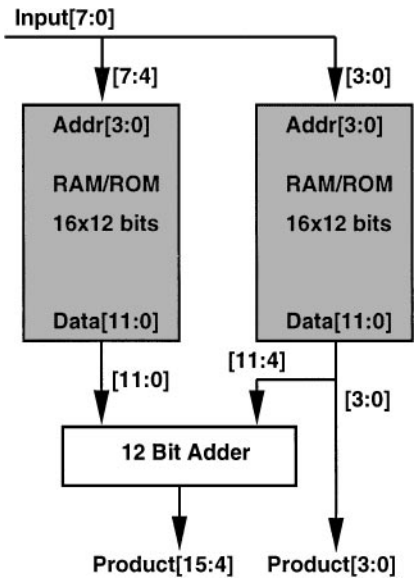

Figure 5. Distributed arithmetic multiplier [42].

As FPGA system development has intensified, more interest has been given to upgrading the accuracy of calculation performed in FPGAs, particularly through the use of floating point arithmetic. In general, floating point operations are difficult to implement in FPGAs due to the complexity of implementation and the amount of hardware needed to achieve desired results. For applications requiring extended precision, floating point is a necessity. In [56] an initial attempt was made to develop basic floating point approaches for FPGAs that met IEEE754 standards for addition and multiplication. Area and performance were considered for various FPGA implementations including shift-and-add, carry-save, and combinational multiplier. Similar work was explored in [57] which applied 18 bit wide floating point adders/subtractors, multipliers, and dividers to 2D FFT and systolic FIR filters implemented on Splash II to avoid overflow and underflow found in fixed point formats. This work was extended to full 32 bit floating point in [58] for multipliers based on bit-parallel adders and digit-serial multipliers. More recent work [59] re-examines these issues with an eye towards greater area efficiency.

5.2. Reconfigurable DSP System Implementation

While recent research in reconfigurable computing has been focused on advanced issues such as dynamic reconfiguration and special-purpose architecture, most work to date has been focused on the effective use of application parallelization and specialization. In general, a number of different DSP applications have been mapped to reconfigurable computing systems containing one, several, and many FPGA devices. In this section a number of DSP projects that have been mapped to reconfigurable hardware are described. These implementations represent a broad set of DSP application areas and serve as a starting point for advanced research in years to come.

Image Processing Applications. The pipelined and fine-grained nature of reconfigurable hardware is a particularly good match for many image processing applications. Real-time image processing typically requires specialized data paths and pipelining which

can be implemented in FPGA logic. A number of projects have been focused in this application area. In [60] and [61], a set of image processing tasks mapped to the Splash II platform, described in Section 3.2, are outlined. Tasks such as Gaussian pyramid-based image compression, image filtering with 1D and 2D transforms, and image conversion using DFT operations are discussed. This work was subsequently extended to include the 2D DCT implemented on the Splash II platform in [62]. The distributed construction of a standalone Splash II system containing numerous physical I/O ports is shown to be particularly useful in achieving high data rates. Since Splash II is effective in implementing systolic versions of algorithms that require repetitive tasks with data shifted in a linear array, image data can quickly be propagated in a processing pipeline. The targeted image processing applications are generally implemented as block-based systolic computations with each FPGA operating as a systolic processor and groups of FPGAs performing specific tasks.

Additional reconfigurable computing platforms have also been used to perform image processing tasks. In [63], a commercial version of PAM, the turbochannel-based Pamette [28], is interfaced to a CCD camera and a liquid crystal polarizing filter is used to perform solar polarimetry. The activity of this application is effectively synchronized with software on an Alpha workstation. In [37] a multi-FPGA system is used to process three-dimensional volume visualization data though ray casting. These implementations show favorable processing characteristics when compared to traditional microprocessor-based systems. In [64], a system is described in which a two-dimensional DCT is implemented using a single FPGA device attached to a backplane bus-based processing card. This algorithm implementation uses distributed arithmetic and is initially coded in VHDL and subsequently compiled using RTL synthesis tools. In [65], a commercial multi-FPGA system is described that is applied to spatial median filtering. In [66], the application of a PCI-based FPGA board to 1D and 2D convolution is presented. Finally, in [67] a system implemented with a single-FPGA processing board is described that performs image interpolation. This system primarily uses bit-serial arithmetic and exploits dynamic reconfiguration to quickly swap portions of the computation located in the reconfigurable hardware. Each computational task has similar computational structure, so reconfiguration time of the FPGA is minimal.

Video Processing Applications. Like image processing, video processing requires substantial data bandwidth and processing capability to handle data obtained from analog video equipment. To support this need, several reconfigurable computing platforms have been adapted for video processing. The PAM system [25], described in Section 3.2, was the first platform used in video applications. A PAM system programmed to perform stereo vision was applied to applications requiring three-dimensional elevation maps such as those needed for planetary exploration. A stereo matching algorithm was implemented that was shown to be substantially faster than programmable DSP-based approaches. This implementation employed dynamic reconfiguration by requiring the reconfiguration of programmable hardware between three distinct processing tasks at run time. A much smaller single-FPGA system, described in [68], was focused primarily on block-based motion estimation. This system tightly coupled SRAM to a single FPGA device to allow for rapid data transfer.

An interesting application of FPGAs for video computation is described in [69]. A stereo transform is implemented across 16 FPGA devices by aligning two images together to determine the depth between the images. Scan lines of data are streamed out of adjacent memories into processing FPGAs to perform the comparison. In an illustration of the benefit of a single-FPGA video system, in [70] a processing platform is described in which a T805 transputer is tightly coupled with an FPGA device to perform frame object tracking. In [71], a single-FPGA video coder, which is reconfigured dynamically between three different sub-functions (Motion Estimation, DCT, and Quantization), is described. The key idea in this project is that the data located in hardware does not move, but rather the functions which operate on it are reconfigured in place.

Audio and Speech Processing. While audio processing typically requires less bandwidth than video and image processing, audio applications can benefit from datapath specialization and pipelining. To illustrate this point, a sound synthesizer was implemented using the multi-FPGA PAM system [25] producing real-time audio of 256 different voices at up to 44.1 kHz. Primarily designed for the use of additive synthesis techniques based on lookup tables, this implementation included features to allow frequency modulation synthesis and/or nonlinear distortion and

was also used as a sampling machine. The physical implementation of PAM as a stand-alone processing system facilitated interfacing to tape recorders and audio amplifiers. The system setup was shown to be an order-of-magnitude faster than a contemporary off-the-shelf DSP.

Other smaller projects have also made contributions in the audio and speech processing areas. In [72], a methodology is described to perform audio processing using a dynamically-reconfigurable FPGA. Audio echo production is facilitated by dynamically swapping filter coefficients and parameters into the device from an adjacent SRAM. Third-party DSP tools are used to generate the coefficients. In [55], an inventive FPGA-based cross-correlator for radio astronomy is described. This system achieves high processing rates of 250 MHz inside the FPGA by heavily pipelining each aspect of the data computation. To support speech processing, a bus-based multi-FPGA board, Tabula Rasa [73], was programmed to perform Markov searches of speech phenomes. This system is particularly interesting because it allowed the use of behavioral partitioning and contained a codesign environment for specification, synthesis, simulation, and evaluation design phases.

Target Recognition. Another important DSP application that has been applied to Splash II is target recognition [74]. To support this application, images are broken into columns and compared to pre-computed templates stored in local memory along with pipelined video data. As described in Section 3.2, near-neighbor communication is used with Splash II to compare pass-through pixels with stored templates in the form of partial sums. After an image is broken into pieces, the Splash II implementation performs second-level detection by roughly identifying sections of sub-images that conform to objects through the use of templates. In general, the use of FPGAs provides a unique opportunity to quickly adapt target recognition to new algorithms, something not possible with ASICs. In another FPGA implementation of target recognition, researchers [75] broke images into pieces called chips and analyzed them using a single FPGA device. By swapping target templates dynamically, a range of targets may be considered. To achieve high performance design, templates were customized to meet the details of the target technology.

Communication Coding. In modern communication systems signal-to-noise ratios make data coding an important aspect of communication. As a result, convolutional coding can be used to improve signal-to-noise ratios based on the constraint length of codes without increasing the power budget. Several reconfigurable computing systems have been configured to aid in the transmission and receipt of data. One of the first applications of reconfigurable hardware to communications involved the PAM project [25]. On-board PAM system RAM was used to trace through 2^{14} possible states of a Viterbi encoder allowing for the computation of 4 states per clock cycle. The flexibility of the system allowed for quick evaluation of new encoding algorithms. A run-length Viterbi decoder, described in [76], was created and implemented using a large reconfigurable system containing 36 FPGA devices. This constraint length 14 decoder was able to achieve decode rates of up to 1Mbit/s. In [77], a single-FPGA system is described that supports variable-length code detection at video transfer rates.

5.3. Reconfigurable Computing Architecture and Compiler Trends for DSP

Over the past decade the large majority of reconfigurable computing systems targeted to DSP have been based on commercial FPGA devices and have been programmed using RTL and structural hardware description languages. While these architectural and programming methodologies have been sufficient for initial prototyping, more advanced architectures and programming languages will be needed in the future. These advancements will especially be needed to support advanced features such as dynamic reconfiguration and high-level compilation over the next few years. In this section, recent trends in reconfigurable computing-based DSP with regard to architecture and compilation are explored. Through near-term research advancement in these important areas, the breadth of DSP applications that are appropriate for reconfigurable computing is likely to increase.

5.3.1. Architectural Trends. Most commercial FPGA architectures have been optimized to perform efficiently across a broad range of circuit domains. Recently, these architectures have been changed to better suit specific application areas.

Specialized FPGA Architectures for DSP. Several FPGA architectures specifically designed for DSP have been proposed over the past decade. In [78], a fine-grained programmable architecture is considered that

uses a customized LUT-based logic cell. The cell is optimized to efficiently perform addition and multiplication through the inclusion of XOR gates within LUT-based logic blocks. Additionally, device inter-cell wire lengths are customized to accommodate both local and global signal interconnections. In [79], a specialized DSP operator array is detailed. This architecture contains a linear array of adders and shifters connected to a programmable bus and is shown to efficiently implement FIR filters. In [80], the basic cell of a LUT-based FPGA is augmented to include additional flip flops and multiplexers. This combination allows for tight inter-block communication required in bit-serial DSP processing. External routing was not augmented for this architecture due to the limited connectivity required by bit-serial operation.

While fine-grained look-up table FPGAs are effective for bit-level computations, many DSP applications benefit from modular arithmetic operations. This need has lead to an interest in reconfigurables with coarse-grained functional units. One such device, Paddi [81], is a DSP-optimized parallel computing architecture that includes 8 ALUs and localized memories. As part of the architecture, a global instruction address is distributed to all processors and instructions are fetched from a local instruction store. This organization allows for high instruction and I/O bandwidth. Communication paths between processors are configured through a communication switch and can be changed on a per-cyle basis. The Paddi architecture was motivated by a need for high data throughput and flexible datapath control in real-time image, audio, and video processing applications. The coarse-grained Matrix architecture [82] is similar to Paddi in terms of block structure but exhibits more localized control. While Paddi has a VLIW-like control word, which is distributed to all processors, Matrix exhibits more MIMD characteristics. Each Matrix tile contains a small processor including a small SRAM and an ALU which can perform eight bit data operations. Both near-neighbor and length-four wires are used to interconnect individual processors. Inter-processor data ports can be configured to support either static or data-dependent dynamic communication.

The ReMarc architecture [83], targeted to multimedia applications, was designed to perform SIMD-like computation with a single control word distributed to all processors. A two-dimensional grid of 16 bit processors is globally controlled with a SIMD-like instruction sequencer. Inter-processor communication takes place either through near-neighbor interconnect or through horizontal and vertical buses. The Chess architecture [84] is based on 4-bit ALUs and contains pipelined near-neighbor interconnect. Each computational tile in the architecture contains memory which can either store local processor instructions or local data memory. The Colt architecture [85] was specially designed as an adaptable architecture for DSP that allows interconnect reconfiguration. This coarse-grained architecture allows run-time data to steer programming information to dynamically determined points in the architecture. A mixture of both 1-bit and 16-bit functional units allows both bit and word-based processing.

While coarse-grained architectures organized in a two-dimensional array offer significant interconnect flexibility, often signal processing applications, such as filtering, can be accommodated with a linear computational pipeline. Several coarse-grained reconfigurable architectures have been created to address this class of applications. PipeRench [86] is a pipelined, linear computing architecture that consists of a sequence of computational *stripes*, each containing look-up tables and data registers. The modular nature of PipeRench makes dynamic reconfiguration on a per-stripe basis straightforward. Rapid [87] is a reconfigurable device based on both linear data and control paths. The coarse-grained architecture for this datapath includes multipliers, adders, and pipeline registers. Unlike PipeRench, the interconnect bus for this architecture is segmented to allow for non-local data transfer. In general, communication patterns built using Rapid interconnect are static although some dynamic operation is possible. A pipelined control bus that runs in parallel to the pipelined data can be used to control computation.

DSP Compilation Software for Reconfigurable Computing. While some high-level compilation systems designed to target DSP algorithms to reconfigurable platforms have been outlined and partially developed, few complete synthesis systems have been constructed. In [88], a high-level synthesis system is described for reconfigurable systems that promotes high-level synthesis from a behavioral synthesis language. For this system, DSP designs are represented as a high-level flow graph and user-specified performance parameters in terms of a maximum and minimum execution schedule are used to guide the synthesis process. In [46], a compilation system is described that converts a standard ANSI C representation of filter and FFT operations into a bit-serial circuit that can be applied to an

FPGA or to a field-programmable multi-chip module. In [89], a compiler, debugger, and linker targeted to DSP data acquisition is described. This work uses a high-level model of communicating processes to specify computation and communication in a multi-FPGA system. By integrating D/A and A/D converters into the configurable platform, a primitive digital oscilloscope is created.

The use of dynamic reconfiguration to reduce area overhead in computing systems has recently motivated renewed interest in reconfigurable computing. While a large amount of work remains to be completed in this area, some preliminary work in the development of software to manage dynamic reconfiguration for DSP has been accomplished. In [90], a method of specifying and optimizing designs for dynamic reconfiguration is described. Through selective configuration scheduling, portions of an application used for 2D image processing is dynamically reconfigured based on need. Later work [33] outlined techniques based on bipartite matching to evaluate which portions of an dynamic application should be reconfigured. The technique is demonstrated using an image filtering example.

Several recent DSP projects address the need for both compile-time and run-time management of dynamic reconfiguration. In [91], a run-time manager is described for a single-chip reconfigurable computing system with a large FIR filter used as a test case. In [32], a compile-time analysis approach to aid reconfiguration is described. In this work, all reconfiguration times are statically-determined in advance and the compilation system determines the minimum circuit change needed at each run-time point to allow for reconfiguration. Benchmark examples which use this approach include arithmetic units for FIR filters which contain embedded constants. Finally, in [48], algorithms are described that perform dynamic reconfiguration to save DSP system power in time-varying applications such as motion estimation. The software tool created for this work dynamically alters the search space of motion vectors in response to changing images. Since power in the motion estimation implementation is roughly correlated with search space, a reduced search proves to be beneficial for applications such as mobile communications. Additionally, unused computational resources can be scheduled for use as memory or rescheduled for use as computing elements as computing demands require.

While the integration of DSP and reconfigurable hardware is just now being considered for single-chip implementation, several board-level systems have been constructed. GigaOps provided the first commercially-available DSP and FPGA board in 1994 containing an Analog Devices 2101 PDSP, 2 Xilinx XC4010s, 256KB of SRAM, and 4MB of DRAM. This PC-based system was used to implement several DSP applications including image processing [92]. Another board-based DSP/FPGA product line is the *Arix-C67* currently available from MiroTech Corporation [93]. This system couples a Xilinx Virtex FPGA with a TMS320C6701 DSP. In addition to supporting several PC-bus interfaces, this system has an operating system, a compiler, and a suite of debugging software.

6. The Future of Reconfigurable Computing and DSP

The future of reconfigurable computing for DSP systems will be determined by the same trends that affect the development of these systems today: system integration, dynamic reconfiguration, and high-level compilation. DSP applications are increasingly demanding in terms of computational load, memory requirements, and flexibility. Traditionally, DSP has not involved significant run-time adaptivity, although this characteristic is rapidly changing. The recent emergence of new applications that require sophisticated, adaptive, statistical algorithms to extract optimum performance has drawn renewed attention to run-time reconfigurability. Major applications driving the move toward adaptive computation include wireless communications with DSP in hand-sets, base-stations and satellites, multimedia signal processing [95], embedded communications systems found in disk drive electronics [11] and high-speed wired interconnects [96], and remote sensing for both environmental and military applications [97]. Many of these applications have strict constraints on cost and development time due to market forces.

The primary trend impacting the implementation of many contemporary DSP systems is Moore's Law, resulting in consistent exponential improvement in integrated circuit device capacity and circuit speeds. According to the National Technology Roadmap for Semiconductors, growth rates based on Moore's Law are expected to continue until at least the year 2015 [94]. As a result, some of the corollaries of Moore's Law will require new architectural approaches to deal with the speed of global interconnect, increased power consumption and power density, and system and

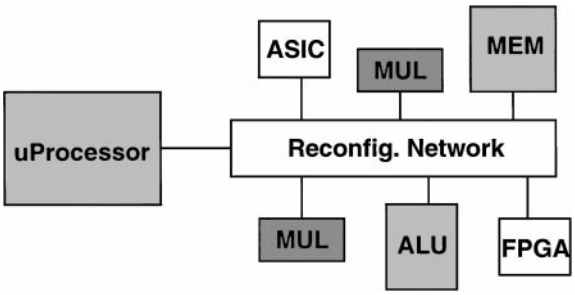

Figure 6. Architectural template for a single-chip Pleiades device [99].

chip-level defect tolerance. Several architectural approaches have been suggested to allow reconfigurable DSP systems to make the best use of large amounts of VLSI resources. All of these architectures are characterized by heterogeneous resources and novel approaches to interconnection. The term *system-on-a-chip* is now being used to describe the level of complexity and heterogeneity available with future VLSI technologies. Figures 6 and 7 illustrate various characteristics of future reconfigurable DSP systems. These are not mutually exclusive and some combination of these features will probably emerge based on driving application domains such as wireless handsets, wireless base-stations, and multimedia platforms. Figure 6, taken from [99], shows an architecture containing an array of DSP cores, a RISC microprocessor, large amounts of uncommitted SRAM, a reconfigurable FPGA fabric and a reconfigurable interconnection network. Research efforts to condense DSPs, FPGA logic, and memory on a single substrate in this fashion are being pursued in the Pleiades project [98, 99]. This work focuses on selecting the correct collection of functional units to perform an operation and then interconnecting them for low power. An experimental compiler has been created for this system [98] and testing has been performed to determine appropriate techniques for building a low-power interconnect. An alternate, adaptive approach [100] that takes a more distributed view of interconnection appears in Fig. 7. This figure shows how a regular tiled interconnect architecture can be overlaid on a set of heterogeneous resources. Each tile contains a *communication switch* which allows for statically-scheduled communication between adjacent tiles. Cycle-by-cycle communications information is held in embedded communication switch SRAM (SMEM).

The increased complexity of VLSI systems enabled by Moore's law presents substantial challenges in design productivity and verification. To support the continued advancement of reconfigurable computing, additional advances will be needed in hardware synthesis, high-level compilation, and design verification. Compilers have recently been developed which allow software development to be done at a high level enabling the construction of complex systems including significant amounts of design re-use. Additional advancements in multi-compilers [101] will be needed to partition designs, generate code, and synchronize interfaces for a variety of heterogeneous computational units. VLIW compilers [102] will be needed to find substantial amounts of instruction level parallelism in DSP code, thereby avoiding the overhead of run-time parallelism extraction. Finally, compilers that target the co-design of hardware and software and leverage techniques such as static inter-processor scheduling [43] will allow truly reconfigurable systems to be specialized to specific DSP computations.

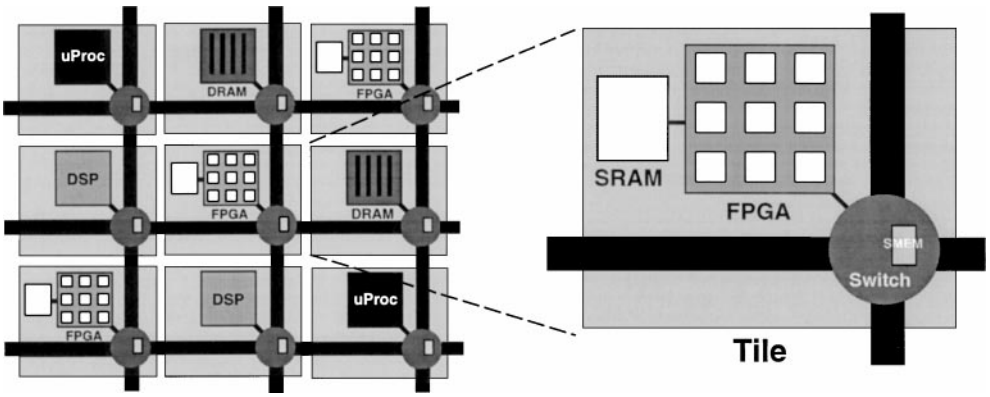

Figure 7. Distributed single-chip DSP interconnection network [100].

A critical aspect of high-quality DSP system design is the effective integration of re-usable components or cores. These cores range from generic blocks like RAMs and RISC microprocessors to more specific blocks like MPEG decoders and PCI bus interfaces. Trends involving core development and integration will continue and tools to support core-based design will emerge allowing significant user interaction for both design-time and run-time specialization and reconfiguration. Specialized synthesis tools will be refined to leverage core-based design and to extract optimum efficiency for DSP kernels while using conventional synthesis approaches for the surrounding circuitry [1, 103].

Verification of complex and adaptive DSP systems will require a combination of simulation and emulation. Simulation tools like Ptolemy [104] have already made significant progress in supporting heterogeneity at a high-level and will continue to evolve in the near future. Newer verification techniques based on logic emulation will emerge as effective mechanisms for using reconfigurable multi-FPGA platforms to verify DSP systems are developed. Through the use of new generations of FPGAs and advanced emulation software [105], new emulation systems will provide the capability to verify complex systems at near real-time rates.

Power consumption in DSP systems will be increasingly important in coming years due to expanding silicon substrates and their application to battery-powered and power-limited DSP platforms. The use of dynamic reconfiguration has been shown to be one approach that can be used to allow a system to adapt its power consumption to changing environments and computational loads [48]. Low-power core designs will allow systems to be assembled without requiring detailed power optimizations at the circuit level. Domain-specific processors [98] and loop transformations [106] have been proposed as techniques for avoiding the inherent power inefficiency of von Neumann architectures [107]. Additional computer-aided design tools will be needed to allow high-level estimation and optimization of power across heterogeneous architectures for dynamically-varying workloads.

The use of DSP in fields such as avionics and medicine have created high reliability requirements that must be addressed through available fault tolerance. Reliability is a larger system goal of which power is only one component. As DSP becomes more deeply embedded in systems, reliability becomes even more critical. The increasing complexity of devices, systems and software all introduce numerous failure points which need to be thoroughly verified. New techniques must especially be developed to allow defect-tolerance and fault-tolerance in the reconfigurable components of DSP systems. One promising technique which takes advantage of FPGA reconfiguration at various grain sizes is described in [108].

Reconfiguration for DSP systems is driven by many different goals: performance, power, reliability, cost and development time. Different applications will require reconfiguration at different granularities and at different rates. DSP systems that require rapid reconfiguration may be able to exploit regularity in their algorithms and architectures to reduce reconfiguration time and power consumption. An approach called *dynamic algorithm transforms* (DAT) [109, 110] is based on the philosophy of moving away from designing algorithms and architectures for worst-case operating conditions in favor of real-time reconfiguration to support the current situational case. This is the basis for reconfigurable ASICs (RASICs) [111] where just the amount of flexibility demanded by the application is introduced. Configuration cloning [112], caching, and compression [113] are other approaches to address the need for dynamic reconfiguration. Techniques from computer architecture regarding instruction fetch and decode need to be modified to deal with the same tasks applied to configuration data.

In conclusion, reconfiguration is a promising technique for the implementation of future DSP systems. Current research in this area leverages contemporary semiconductors, architectures, CAD tools, and methodologies in an effort to support the ever-increasing demands of a wide range of DSP applications. There is much work still to be done, however, since reconfigurable computing presents a very different computational paradigm for DSP system designers as well as DSP algorithm developers.

Note

1. In practice, usable gate counts for devices are often significantly lower than reported data book values (by about 20–40%). Generally, the proportion of per-device logic that is usable has remained roughly constant over the years indicated in Fig. 3.

References

1. D. Singh, J. Rabaey, M. Pedram, F. Catthor, S. Rajgopal, N. Sehgal, and T. Mozdzen, “Power-conscious CAD Tools and

Methodologies: A Perspective," in *Proceedings of the IEEE*, vol. 83, no. 4, 1995, pp. 570–594.
2. J. Rabaey, R. Broderson, and T. Nishitani, "VLSI Design and Implementation Fuels the Signal-Processing Revolution," *IEEE Signal-Processing Magazine*, pp. 22–38, Jan. 1998.
3. Y.H. Hu, *Programmable Digital Signal Processors*. New York, N.Y.: Marcel Dekker, Inc. 2000.
4. E. Lee, "Programmable DSP Architectures, Part I," *IEEE Signal Processing Magazine*, vol. 5, no. 4, Oct. 1988, pp. 4–19.
5. E. Lee, "Programmable DSP Architectures, Part II," *IEEE Signal Processing Magazine*, vol. 6, no. 1, Jan. 1989, pp. 4–14.
6. J. Eyre and J. Bier, "The Evolution of DSP Processors: From Early Architecture to the Latest Developments," *IEEE Signal Processing Magazine*, vol. 17, no. 2, March 2000, pp. 44–51.
7. A. Kalavade, J. Othmer, B. Ackland, and K. Singh, "Software Environment for a Multiprocessor DSP," in *Proceedings of the 36th Design Automation Conference*, June 1999.
8. P. Schaumont, S. Vernalde, L. Rijnders, M. Engels, and I. Bolsens, "A Programming Environment for the Design of Complex High Speed ASICs," in *Proceedings of the 35th Design Automation Conference*, June 1998, pp. 315–320.
9. Broadcom Corporation, *www.broadcom.com*, 2000.
10. Qualcomm Corporation, *www.qualcomm.com*, 2000.
11. N. Nazari, "A 500 Mb/s Disk Drive Read Channel in .25 um CMOS Incorporating Programmable Noise Predictive Viterbi Detection and Trellis Coding," in *Proceedings of the IEEE International Solid State Circuits Conference*, 2000.
12. A. Bell, The Dynamic Digital Disk. *IEEE Spectrum*, vol. 36, no. 10, Oct. 1999, pp. 28–35.
13. G. Weinberger, "The New Millennium: Wireless Technologies for a Truly Mobile Society," in *Proceedings of the IEEE International Solid State Circuits Conference*, 2000.
14. W. Strauss, "Digital Signal Processing: The New Semiconductor Industry Technology Driver," *IEEE Signal Processing Magazine*, vol. 17, no. 2, March 2000, pp. 52–56.
15. W. Mangione-Smith, B. Hutchings, D. Andrews, A. Dehon, C. Ebeling, R. Hartenstein, O. Mencer, J. Morris, K. Palem, V. Prasanna, and H. Spaanenberg, "Seeking Solutions in Configurable Computing," *IEEE Computer*, vol. 30, no. 12, Dec. 1997, pp. 38–43.
16. J. Villasenor and W. Mangione-Smith, "Configurable Computing," *Scientific American*, vol. 276, no. 6, June 1997, pp. 66–71.
17. S. Hauck, "The Role of FPGAs in Reprogrammable Systems," in *Proceedings of the IEEE*, vol. 86, no. 4, April 1998, 615–638.
18. J. Villasenor and B. Hutchings, "The Flexibility of Configurable Computing," *IEEE Signal Processing Magazine*, Sept. 1998, pp. 67–84.
19. Xilinx Corporation, *www.xilinx.com*, 2000.
20. Altera Corporation, *www.altera.com*, 2000.
21. Xilinx Corporation, *The Programmable Logic Data Book*, 1994.
22. Xilinx Corporation, *The Programmable Logic Data Book*, 1998.
23. Xilinx Corporation, *Virtex Data Sheet*, 2000.
24. J. Arnold, D. Buell, and E. Davis, "Splash II," in *Proceedings, 4th ACM Symposium of Parallel Algorithms and Architectures*, San Diego, CA, 1992, pp. 316–322.
25. J. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati, and P. Boucard, "Programmable Active Memories: Reconfigurable Systems Come of Age," *IEEE Transactions on VLSI Systems*, vol. 4, no. 1, March 1996, pp. 56–69.
26. M. Gokhale, W. Holmes, A. Kopser, S. Lucas, R. Minnich, D. Sweeney, and D. Lopresti, "Building and Using a Highly Parallel Programmable Logic Array," *Computer*, vol. 24, no. 1, Jan. 1991, pp. 81–89.
27. M. Gokhale and R. Minnich, "FPGA Computing in a Data Parallel C," in *Proceedings IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1993, pp. 94–101.
28. M. Shand, "Flexible Image Acquisition Using Reconfigurable Hardware," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1995, pp. 125–134.
29. X.-P. Ling and H. Amano, "WASMII: A Data Driven Computer on a Virtual Hardware," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1993, pp. 33–42.
30. A. Dehon, "DPGA-Coupled Microprocessors: Commodity ICs for the 21st Century," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1994, pp. 31–39.
31. Atmel Corporation, *AT6000 Data Sheet*, 1999.
32. J.P. Heron, R. Woods, S. Sezer, and R.H. Turner, "Development of a Run-Time Reconfiguration System with Low Reconfiguration Overhead," *Journal of VLSI Signal Processing*, 2001.
33. N. Shirazi, W. Luk, and P.Y. Cheung, "Automating Production of Run-Time Reconfigurable Designs," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1998, pp. 147–156.
34. N. Hastie and R. Cliff, "The Implementation of Hardware Subroutines on Field Programmable Gate Arrays," in *Proceedings, IEEE Custom Integrated Circuits Conference*, vol. 3, no. 4, May 1990, pp. 1–4.
35. M. Wirthlin and B. Hutchings, "A Dynamic Instruction Set Computer," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1995, pp. 99–107.
36. R. Amerson, R. Carter, W.B. Culbertson, P. Kuekes, and G. Snider, "Teramac—Configurable Custom Computing," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1995, pp. 32–38.
37. W.B. Culbertson, R. Amerson, R. Carter, P. Kuekes, and G. Snider, "Exploring Architectures for Volume Visualization on the Teramac Computer," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1996, pp. 80–88.
38. J. Varghese, M. Butts, and J. Batcheller, "An Efficient Logic Emulation System," *IEEE Transactions on VLSI Systems*, vol. 1, no. 2, June 1993, pp. 171–174.
39. J. Babb, R. Tessier, M. Dahl, S. Hanono, D. Hoki, and A. Agarwal, "Logic Emulation with Virtual Wires," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 6, June 1997, pp. 609–626.
40. H. Schmit, L. Arnstein, D. Thomas, and E. Lagnese, "Behavioral Synthesis for FPGA-based Computing," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1994, pp. 125–132.
41. A. Duncan, D. Hendry, and P. Gray, "An Overview of the COBRA-ABS High Level Synthesis System," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1998, pp. 106–115.

42. R.J. Peterson, "An Assessment of the Suitability of Reconfigurable Systems for Digital Signal Processing," Master's Thesis, Brigham Young University, Department of Electrical and Computer Engineering, Sept. 1995.
43. J. Babb, M. Rinard, C.A. Moritz, W. Lee, M. Frank, R. Barua, and S. Amarasinghe, "Parallelizing Applications to Silicon," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1999.
44. Texas Instruments Corporation, *TMS320C6201 DSP Data Sheet*, 2000.
45. D. Goeckel, "Robust Adaptive Coded Modulation for Time-Varying Channels with Delayed Feedback," in *Proceedings of the Thirty-Fifth Annual Allerton Conference on Communication, Control, and Computing*, Oct. 1997, pp. 370–379.
46. T. Isshiki and W.W.-M. Dai, "Bit-Serial Pipeline Synthesis for Multi-FPGA Systems with C++ Design Capture," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1996, pp. 38–47.
47. Altera Corporation, *Flex10K Data Sheet*, 1999.
48. S.R. Park and W. Burleson, "Reconfiguration for Power Savings in Real-Time Motion Estimation," in *Proceedings, International Conference on Acoustics, Speech and Signal Processing*, 1997.
49. G.R. Goslin, "A Guide to Using Field Programmable Gate Arrays for Application-Specific Digital Signal Processing Performance," in *Xilinx Application Note*, 1998.
50. S. He and M. Torkelson, "FPGA Implementation of FIR Filters Using Pipelined Bit-Serial Canonical Signed Digit Multipliers," in *Custom Integrated Circuits Conference*, 1994, pp. 81–84.
51. Y.C. Lim, J.B. Evans, and B. Liu, "An Efficient Bit-Serial FIR Filter Architecture," in *Circuits, Systems, and Signal Processing*, May 1995.
52. J.B. Evans, "Efficient FIR Filter Architectures Suitable for FPGA Implementation," *IEEE Transactions on Circuits and Systems*, vol. 41, no. 7, July 1994, pp. 490–493.
53. C.H. Dick, "FPGA Based Systolic Array Architectures for Computing the Discrete Fourier Transform," in *Proceedings, International Symposium on Circuits and Systems*, 1996, pp. 465–468.
54. P. Kollig, B.M. Al-Hashimi, and K.M. Abbott, "FPGA Implementation of High Performance FIR Filters," in *Proceedings, International Symposium on Circuits and Systems*, 1997, pp. 2240–2243.
55. B.V. Herzen, "Signal Processing at 250 MHz using High Performance FPGAs," in *International Symposium on Field Programmable Gate Arrays*, Monterey, CA, Feb. 1997, pp. 62–68.
56. B. Fagin and C. Renard, "Field Programmable Gate Arrays and Floating Point Arithmetic," *IEEE Transactions on VLSI Systems*, vol. 2, no. 3, Sept. 1994, pp. 365–367.
57. N. Shirazi, A. Walters, and P. Athanas, "Quantitative Analysis of Floating Point Arithmetic on FPGA-based Custom Computing Machines," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1995, pp. 155–162.
58. L. Louca, W.H. Johnson, and T.A. Cook, "Implementation of IEEE Single Precision Floating Point Addition and Multiplication on FPGAs," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1996, pp. 107–116.
59. W.B. Ligon, S. McMillan, G. Monn, F. Stivers, and K.D. Underwood, "A Re-evaluation of the Practicality of Floating-Point Operations on FPGAs," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1998.
60. P. Athanas and A.L. Abbott, "Real-Time Image Processing on a Custom Computing Platform," *IEEE Computer*, vol. 28, no. 2, Feb. 1995, pp. 16–24.
61. A.L. Abbott, P. Athanas, L. Chen, and R. Elliott, "Finding Lines and Building Pyramids with Splash 2," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1994, pp. 155–161.
62. N. Ratha, A. Jain, and D. Rover, "Convolution on Splash 2," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1995, pp. 204–213.
63. M. Shand and L. Moll, "Hardware/Software Integration in Solar Polarimetry," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1998, pp. 18–26.
64. R. Woods, D. Trainer, and J.-P. Heron, "Applying an XC6200 to Real-Time Image Processing," *IEEE Design and Test of Computers*, vol. 15, no. 1, Jan. 1998, pp. 30–37.
65. B. Box, "Field Programmable Gate Array Based Reconfigurable Preprocessor," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1994, pp. 40–48.
66. S. Singh and R. Slous, "Accelerating Adobe Photoshop with Reconfigurable Logic," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1998, pp. 18–26.
67. R.D. Hudson, D.I. Lehn, and P.M. Athanas, "A Run-Time Reconfigurable Engine for Image Interpolation," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1998, pp. 88–95.
68. J. Greenbaum and M. Baxter, "Increased FPGA Capacity Enables Scalable, Flexible CCMs: An Example from Image Processing," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1997.
69. J. Woodfill and B.V. Herzen, "Real-Time Stereo Vision on the PARTS Reconfigurable Computer," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1997, pp. 242–250.
70. I. Page, "Constructing Hardware-Software Systems from a Single Description," *Journal of VLSI Signal Processing*, vol. 12, no. 1, 1996, pp. 87–107.
71. J. Villasenor, B. Schoner, and C. Jones, "Video Communications Using Rapidly Reconfigurable Hardware," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, Dec. 1995, pp. 565–567.
72. J. Ferguson and E. Lee, "Generating Audio Effects Using Dynamic FPGA Reconfiguration," *Computer Design*, vol. 36, no. 2, Feb. 1997.
73. D.E. Thomas, J.K. Adams, and H. Schmit, "A Model and Methodology for Hardware-Software Codesign," *IEEE Design and Test of Computers*, vol. 10, no. 3, Sept. 1993, pp. 6–15.
74. M. Rencher and B.L. Hutchings, "Automated Target Recognition on Splash II," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1997, pp. 192–200.

75. J. Villasenor, B. Schoner, K.-N. Chia, and C. Zapata, "Configurable Computing Solutions for Automated Target Recognition," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1996, pp. 70–79.
76. D. Yeh, G. Feygin, and P. Chow, "RACER: A Reconfigurable Constraint-Length 14 Viterbi Decoder," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1996.
77. G. Brebner and J. Gray, "Use of Reconfigurability in Variable-Length Code Detection at Video Rates," in *Proceedings, Field Programmable Logic and Applications (FPL'95)*, Oxford, England, Aug. 1995, pp. 429–438.
78. M. Agarwala and P.T. Balsara, "An Architecture for a DSP Field-Programmable Gate Array," *IEEE Transactions on VLSI Systems*, vol. 3, no. 1, March 1995, pp. 136–141.
79. T. Arslan, H.I. Eskikurt, and D.H. Horrocks, "High Level Performance Estimation for a Primitive Operator Filter FPGA," in *Proceedings, International Symposium on Circuits and Systems*, 1998, pp. v237–v240.
80. A. Ohta, T. Isshiki, and H. Kunieda, "New FPGA Architecture for Bit-Serial Pipeline Datapath," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1998.
81. D.C. Chen and J. Rabaey, "A Reconfigurable Multiprocessor IC for Rapid Prototyping of Algorithmic-Specific High Speed DSP Data Paths," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 12, Dec. 1992, pp. 1895–1904.
82. E. Mirsky and A. Dehon, "MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1996, pp. 157–166.
83. T. Miyamori and K. Olukotun, "A Quantitative Analysis of Reconfigurable Coprocessors for Multimedia Applications," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1998.
84. A. Marshall, T. Stansfield, I. Kostarnov, J. Vuillemin, and B. Hutchings, "A Reconfigurable Arithmetic Array for Multimedia Applications," in *International Symposium on Field Programmable Gate Arrays*, Monterey, CA, Feb. 1999, pp. 135–143.
85. R. Bittner and P. Athanas, " Wormhole Run-time Reconfiguration," in *International Symposium on Field Programmable Gate Arrays*, Monterey, CA, Feb. 1997, pp. 79–85.
86. S.C. Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R.R. Taylor, and R. Laufer, "PipeRench: A Coprocessor for Streaming Multimedia Acceleration," in *Proceedings, International Symposium on Computer Architecture*, Atlanta, GA, June 1999, pp. 28–39.
87. C. Ebeling, D. Cronquist, P. Franklin, J. Secosky, and S.G. Berg, "Mapping Applications to the RaPiD Configurable Architecture," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1997, pp. 106–115.
88. M. Leeser, R. Chapman, M. Aagaard, M. Linderman, and S. Meier, "High Level Synthesis and Generating FPGAs with the BEDROC System," *Journal of VLSI Signal Processing*, vol. 6, no. 2, 1993, pp. 191–213.
89. A. Wenban and G. Brown, "A Software Development System for FPGA-based Data Acquisition Systems," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1996, pp. 28–37.
90. W. Luk, N. Shirazi, and P.Y. Cheung, "Modelling and Optimising Run-Time Reconfigurable Systems," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1996, pp. 167–176.
91. J. Burns, A. Donlin, J. Hogg, S. Singh, and M. de Wit, "A Dynamic Reconfiguration Run-Time System," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1997, pp. 66–75.
92. P. Athanas and R. Hudson, "Using Rapid Prototyping to Teach the Design of Complete Computing Solutions," in *Proceedings, IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, CA, April 1996.
93. Mirotech Corporation, *Mirotech Data Sheet*, 1999. www.mirotech.com.
94. D. Sylvester and K. Keutzer, "Getting to the Bottom of Deep Submicron," in *Proceedings, International Conference on Computer-Aided Design*, Oct. 1998, pp. 203–211.
95. P. Pirsch, A. Freimann, and M. Berekovic, "Architectural Approaches for Multimedia Processors," in *Proceedings of Multimedia Hardware Architecture, SPIE*, 1997, vol. 3021, pp. 2–13.
96. W. Dally and J. Poulton, *Digital Systems Engineering*. Cambridge University Press, 1999.
97. M. Petronino, R. Bambha, J. Carswell, and W. Burleson, "An FPGA-based Data Acquisition System for a 95 GHz W-band Radar," in *Proceedings, International Conference on Acoustics, Speech, and Signal Processing*, 1997, pp. 3037–3040.
98. H. Zhang, V. Prabhu, V. George, M. Wan, M. Benes, A. Abnous, and J.M. Rabaey, "A 1V Heterogeneous Reconfigurable Processor IC for Baseband Wireless Applications," in *Proceedings of the IEEE International Solid State Circuits Conference*, 2000.
99. M. Wan, H. Zhang, V. George, M. Benes, A. Abnous, V. Prabhu, and J. Rabaey, "Design Methodology of a Low-Energy Reconfigurable Single-Chip DSP System," *Journal of VLSI Signal Processing*, 2001.
100. R. Tessier, "Adaptive Systems-on-a-Chip," in University of Massachusetts, Department of ECE Technical Memo, 2000. www.ecs.umass.edu/ece/tessier.
101. K. McKinley, S.K. Singhai, G.E. Weaver, and C.C. Weems, "Compiler Architectures for Heterogeneous Processing," *Languages and Compilers for Parallel Processing*, pp. 434–449, Aug. 1995, Lecture Notes in Computer Science, vol. 1033.
102. K. Konstantinides, "VLIW Architectures for Media Processing," *IEEE Signal Processing Magazine*, vol. 15, no. 2, March 1998, pp. 16–19.
103. Synopsys Corporation, *www.synopsys.com*, 2000.
104. J.T. Buck, S. Ha, E.A. Lee, and D.G. Messerschmitt, "Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems," *International Journal of Computer Simulation*, vol. 4, 1994, pp. 152–184.
105. R. Tessier, "Incremental Compilation for Logic Emulation," in *IEEE Tenth International Workshop on Rapid System Prototyping*, Clearwater, Florida, June 1999, pp. 236–241.
106. H. DeMan, J. Rabaey, J. Vanhoof, G. Goosens, P. Six, and L. Claesen, "CATHEDRAL-II-A Computer-Aided Synthesis System for Digital Signal Processing VLSI Systems," *Computer-Aided Engineering Journal*, vol. 5, no. 2, April 1988, pp. 55–66.

107. M. Horowitz and R. Gonzalez, "Energy Dissipation in General Purpose Processors," *Journal of Solid State Circuits*, vol. 31, no. 9, Nov. 1996, pp. 1277–1284.
108. V. Lakamraju and R. Tessier, "Tolerating Operational Faults in Cluster-Based FPGAs," in *International Symposium on Field Programmable Gate Arrays*, Monterey, CA, Feb. 2000, pp. 187–194.
109. M. Goel and N.R. Shanbhag, "Dynamic Algorithm Transforms for Low-power Adaptive Equalizers," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, Oct. 1999, pp. 2821–2832.
110. M. Goel and N.R. Shanbhag, "Dynamic Algorithm Transforms (DAT): A Systematic Approach to Low-power Reconfigurable Signal Processing," *IEEE Transactions on VLSI Systems*, vol. 7, no. 4, Dec. 1999, pp. 463–476.
111. J. Tschanz and N.R. Shanbhag, "A Low-power Reconfigurable Adaptive Equalizer Architecture," in *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, Monterey, CA, Oct. 1999.
112. S.R. Park and W. Burleson, "Configuration Cloning: Exploiting Regularity in Dynamic DSP Architectures," in *International Symposium on Field Programmable Gate Arrays*, Monterey, CA, Feb. 1999, pp. 81–89.
113. S. Hauck, Z. Li, and E. Schwabe, "Configuration Compression for the Xilinx XC6200 FPGA," in *Proceedings, IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1998, pp. 138–146.

Russell Tessier is an assistant professor of electrical and computer engineering at the University of Massachusetts at Amherst. He received the B.S. degree in Computer Engineering from Rensselaer Polytechnic Institute, Troy, N.Y. in 1989 and S.M. and PhD degrees in Electrical Engineering from MIT in 1992 and 1999, respectively. Tessier is a founder of Virtual Machine Works, a logic emulation company, and has also worked at BBN and Ikos Systems. Prof. Tessier currently leads the Reconfigurable Computing Group at UMass. His research interests include computer architecture, field-programmable gate arrays, and system verification.

Wayne Burleson is an associate professor of electrical and computer engineering at the University of Massachusetts at Amherst. He received B.S. and M.S. degrees from MIT in 1983 and the PhD degree in 1989 from the University of Colorado. He has worked as a custom DSP chip designer at VLSI Technology and Fairchild and currently consults with the Alpha Development Group of Compaq. He currently leads research in VLSI and Configurable Signal Processing, with emphasis in VLSI methods and CAD tools for low power and high performance. He also leads a research project in Educational Innovations using Multimedia. Prof. Burleson is an associate editor of IEEE Transactions on VLSI, ACM TODAES and Kluwer JVSP. He has served as program chair and general chair of the Signal Processing Systems Workshop and has served on the program committees of ISLPED, TAU, ASAP, CHES, ISCAS and VLSI. He recently won the Best Paper Award at the 1999 Frontiers in Education Conference.