Programming and Development Environments for Configurable Computing Systems[1]

S. Kumar, D. Bhatt, S. Vestal, B. Wren, J. Shackleton, H. Shirley,
R. Bhatt, J. Golusky, M. Vojta, C. Nanavati, P. Zumsteg, P. Symosek
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
612-951-7107
skumar@htc.honeywell.com

S. Crago, B. Schott, R. Parker
University of Southern California/Information Sciences Institute
4350 N. Fairfax Dr., Suite 770
Arlington, VA 22203

G. Gardner
Honeywell Space and Strategic Systems Operation
13350 U.S. Highway 19 North
Clearwater, FL 33764

Abstract—Heterogeneous configurable computing systems containing general-purpose processors, special-purpose processors, field-programmable gate arrays (FPGAs), and application-specific integrated circuits (ASICs) present several programming and development challenges. The ADAPTERS effort is addressing these challenges by focusing on three technology areas: 1) a unified data flow programming environment, 2) a hardware/software partitioning and mapping tool, along with a system level modeling tool, and 3) a dynamic reconfiguration run-time environment for FPGAs. To investigate automatic mapping techniques, a constant false alarm rate (CFAR) application has been implemented on a combined PC/WILDFORCE platform. The combined platform containing FPGAs provided a 20% improvement in the end-to-end latency compared to a software solution running on a PC. An image compression example has also been developed to explore mode-based dynamic FPGA reconfiguration. The dynamic reconfiguration technology being developed can be used to adapt the behavior of systems, such as satellites, to changing internal and external conditions.

TABLE OF CONTENTS

1. INTRODUCTION

Configurable computing technologies [1], such as field-programmable gate arrays (FPGAs), have added another alternative to the already complex system design space for implementing application functions. The use of FPGAs has progressed from implementation of glue code logic and various types of controllers to implementation of complex algorithms that have been performed traditionally in software, offering higher performance than general-purpose computing platforms for many applications. Due to their programmable nature, FPGAs have further blurred the distinction between hardware and software. Thus, FPGAs offer a cost-effective implementation technology, one that blends the flexibility of general-purpose processors with the high performance of application-specific integrated circuits (ASICs).

In most applications today, FPGAs employ static (compile-time) reconfiguration, where the function to be implemented on the FPGA remains the same throughout the application's execution. Recently, there has been interest in developing techniques to support dynamic (run-time) reconfiguration, in which the function to be performed by the FPGA varies over time. As an example of exploiting reconfiguration ideas, evolvable hardware (EHW) approaches [2] often combine genetic algorithm concepts in conjunction with FPGAs to produce systems with pre-specified behaviors or that solve specific problems.

Dynamic reconfiguration techniques are a key enabling technology for adaptive computing systems, allowing different algorithms to be employed under diverse scenarios and conditions, for example, during different modes of system operation. Many types of commercial and military systems, for example, avionics, unmanned combat air

[1] 0-7803-5846-5/00/$10.00©2000 IEEE

vehicles, missiles, robotics, and spacecraft, can benefit from adaptation by making the system more robust, supporting more autonomous behavior, achieving performance objectives, and satisfying power demands.

As configurable computing technologies evolve and are embedded within larger systems, the complexity of these systems will increase. Applications that perform complex tasks utilize a diverse collection of resources to meet system objectives. Such heterogeneous configurable computing systems consisting of general-purpose processors, special-purpose processors (for example, digital signal processors), FPGAs, and ASICs present several challenges:

- The programming of such systems is a time-consuming process, requiring considerable effort to address interfaces between application functions executing on heterogeneous processing elements, integration of configurable computing elements, and application portability.
- The design and analysis of such systems is difficult due to the large design space and additional design trade-offs that arise with the introduction of configurable computing technology.
- The implementation of applications is a complex activity, requiring run-time support for data flow transparency, including interfaces, and dynamic FPGA reconfiguration.

These difficulties offset many of the potential benefits the technology has to offer, and as a result, hinders its insertion into many applications. Thus, in order to take advantage of configurable computing technology, new *system level* tools and techniques are required to support the programming and development of heterogeneous configurable computing systems. These considerations have led to our current effort: **A DomAin-specific Programming and development TEchnology for run-time Reconfiguration at the System level (ADAPTERS)**. As a part of this effort, we are introducing software engineering concepts and ideas to support the programming and development of FPGAs, and building upon our work on parallel and real-time embedded systems to support heterogeneous system integration.

The remaining sections of this paper are organized as follows. Section 2 provides an overview of our approach, which attempts to address the challenges stated earlier. An example application, constant false alarm rate (CFAR), is described in Section 3. This application is used to illustrate the technologies being developed. In Section 4, a unified data flow programming environment is described, which is part of an environment that allows both software and hardware descriptions of a system to be captured. In Section 5, hardware/software partitioning and mapping techniques that employ genetic algorithms are discussed, along with system level modeling capabilities. Section 6 provides a description of run-time environments being developed for FPGAs. Other applications and platforms being considered for demonstration and evaluation of the technology are mentioned in Section 7. Section 8 summarizes the paper, discusses some issues associated with configurable computing technologies, and describes future directions.

2. THE ADAPTERS APPROACH

At a high level, our effort is addressing the problem of implementing a complex application onto a heterogeneous collection of resources consisting of general-purpose processors, digital signal processors, and FPGAs, where some FPGAs may support either full or partial reconfiguration. The development and implementation of such systems mandate an integrated hardware/software approach, which is a key element of the ADAPTERS effort.

An overview of our approach is shown in Figure 1. The idea is to address both the programming and development of heterogeneous configurable computing systems using a hardware/software co-design [3] approach that integrates a development environment with a run-time environment. The three primary technology areas being emphasized are application development, hardware/software partitioning and modeling, and the run-time environment. These three technology areas are being developed as the domain-specific application development environment (DADE), partitioning, mapping, and trade-off analysis environment (PMTE), and the dynamic reconfiguration run-time environment (DRRTE), respectively.

The ADAPTERS environment can be used for systems ranging from single boards to multiple boards. Although not shown explicitly, information can flow from later stages to earlier stages, for example, to improve the partitioning and mapping process. An overview of the DADE, PMTE, and DRRTE is provided below. More detail on each of these technologies is provided in subsequent sections.

The DADE supports a unified data flow programming model, and allows both the software and hardware architecture to be represented in a common environment. An important objective is to transparently integrate configurable computing elements into an application. This important feature is supported through a "plug-and-play" capability, referred to as an application programming interface (API) wrapper, that allows a designer to specify multiple implementations of an application function, for example, software running on a processor or an FPGA. It also allows a complex application to be ported to other configurable computing platforms, an important consideration for heterogeneous systems. This feature allows a designer to take advantage of new improvements in technology, such as faster FPGAs.

The DADE interfaces to off-the-shelf hardware synthesis and place and route technologies, not being developed under this effort, which are used to generate FPGA configurations. Also, code generation facilities create software functions, glue code for interfacing different components, application programming interfaces, and run-time tables, which are employed by the run-time environment.

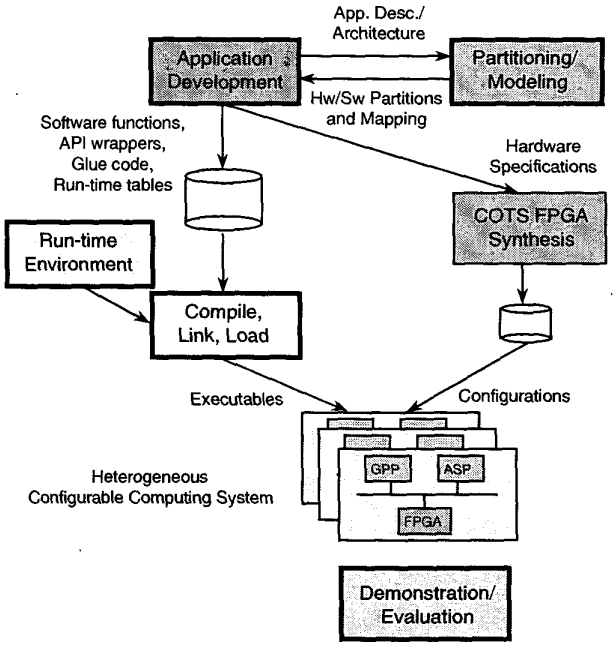

Figure 1 Overview of ADAPTERS Approach

The PMTE addresses the design and analysis of heterogeneous configurable computing systems including automatic partitioning and mapping of a large-scale application onto a heterogeneous architecture. Two primary technologies comprise the PMTE: 1) a tool that supports design trade-off exploration by considering different hardware/software partitions and mappings using genetic algorithms, and 2) a system level, token-based, hardware/software performance modeling environment.

The DRRTE performs several functions. It provides data flow transparency among functions implemented in software running on processors and FPGAs. In other words, it seamlessly implements the data flows specified in the application, regardless of whether the functions are being performed on a general-purpose processor or an FPGA. It supports scheduling and loading of FPGA configurations to allow for system mode changes and time-sharing of different functions on the same FPGA. It also supports the analysis and implementation of real-time scheduling. Finally, partial reconfiguration ideas are being integrated into the environment.

The ADAPTERS effort is related to some other programming and development environments being researched for configurable computing systems. Our research shares a "system level" perspective with the DEFACTO [4], MATCH [5], and model integrated environment for adaptive computing [6] efforts. The latter is particularly relevant, since it also addresses mode-based behavior. ADAPTERS is synergistic with that effort since our efforts are both implementing missile-based applications. DEFACTO and MATCH are more strongly oriented towards compilation-based approaches for implementing applications than our work. Some efforts are developing languages to support programming of configurable computing systems [7]. Our work is also synergistic with the System Level Applications of Adaptive Computing (SLAAC) effort and the application programming interfaces (APIs) being developed [8] since we are focusing on system level technologies that can be utilized for the distributed, heterogeneous architecture being developed under that effort. A board being developed under SLAAC is described in Section 7.

3. An Example Application – Constant False Alarm Rate

The interfacing stressmark [9], based on a constant false alarm rate (CFAR) application, is used to illustrate the ideas discussed in this paper. CFAR is a technique for processing sensor information adaptively in different regions to keep the rate at which false alarms occur nearly constant. The basic principle is to use local statistics, estimated in real-time, to adapt the processing that occurs in a region. In synthetic aperture radar (SAR) target classification, CFAR detection is used to adaptively separate clutter from man-made objects based on the difference in intensity between the measured signals and the local background.

The objective of CFAR detection processing is to determine, in each of several SAR intensity maps, the likely locations of man-made objects. The outputs of CFAR detection processing feed a more sophisticated automatic target recognition (ATR) algorithm. By isolating the areas where objects are likely to be found, small "chips" (regions) of SAR imagery can be analyzed using pattern recognition algorithms to classify the objects located in each chip. Thus, CFAR detection is an integral part of a SAR ATR algorithm used to identify areas worthy of further examination.

Referring to Figure 2, the major "logical" computations in the application are compute local statistics, detect anomalies, erode, dilate, and determine centroids. The input data set consists of a sequence of SAR intensity maps, and the outputs are a collection of centroid locations for the objects.

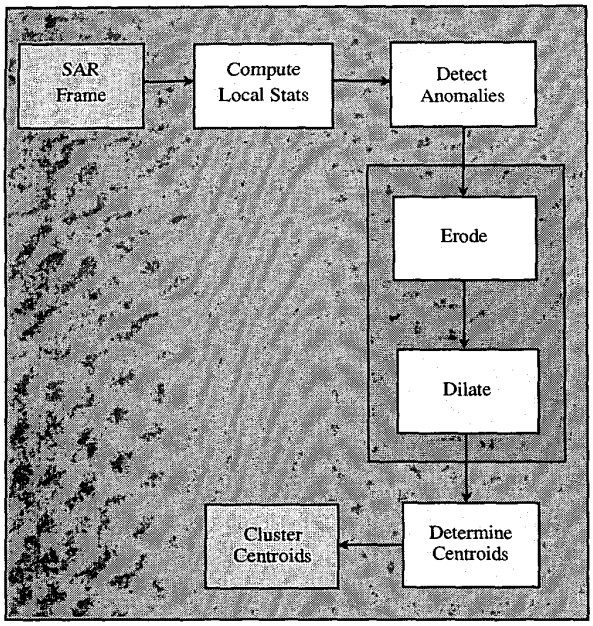

Figure 2 Constant False Alarm Rate Application

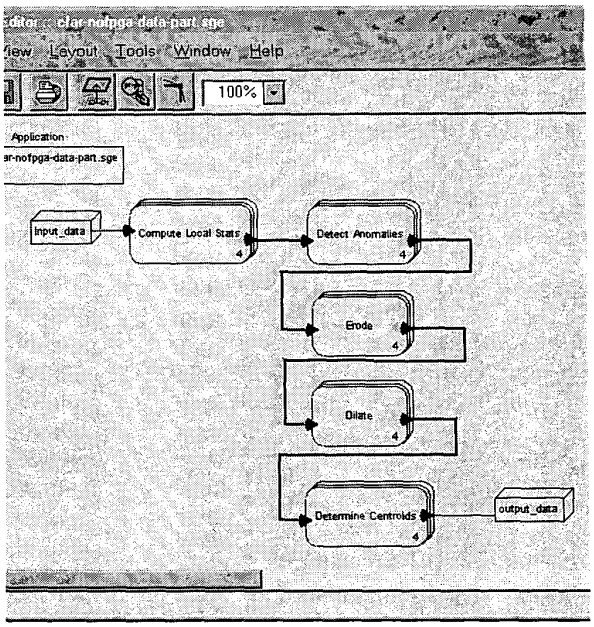

Figure 3 Multi-thread Constant False Alarm Rate Application

4. Domain-Specific Application Development Environment

The time-consuming process of programming heterogeneous configurable computing systems is addressed in several ways within the DADE. A unified data flow programming model and system level abstractions, using a library of domain-specific primitives, are employed that allow an application to be constructed quickly and mapped onto multiple platforms. The automated construction of application programming interface (API) wrappers hides the complexity of underlying configurable elements and supports portability of applications. The use of domain-specific styles constrains the choices of parallelism and synchronization [10], and imposes certain composition rules, allowing the resulting application to be mapped efficiently onto the platform. Auto-generation of interfaces between heterogeneous components reduces the amount of programming effort as well.

A multi-board CFAR example is used to illustrate the representation of application software and hardware architectures, and integration of the partitioning and mapping technology with the DADE. Figure 3 shows the application represented within the DADE using a data flow programming model. The basic model consists of independent threads of control that communicate with other threads using data flows. Specific synchronization is associated with each flow.

The boxes with rounded corners correspond to functions, and the arcs represent flows. A function contains a particular implementation of the behavior, whether it is for a configurable device or a general-purpose processor. Functions can be characterized in terms of their computational requirements, such as fixed-point operations, floating-point operations, and memory usage. Also, the physical resource mapping, scheduling properties, and the number of threads can be specified, shown in the lower right hand corner of the function. Scheduling properties are associated with each function, such as "event driven" or "periodic." Properties associated with the flows govern the type of synchronization to be employed, such as sequential or pipelined execution, and the latency constraints.

Task-level or data-level parallelism is supported. Task-level parallelism refers to the concurrent execution of independent threads of control. Data-level parallelism refers to multiple parallel threads that operate on individual slices of multi-dimensional data. As an example of data-level parallelism, in Figure 3, four parallel threads are associated with the erode function, with each thread independently operating on a specific slice of data. Thus, each thread of the erode function can be executed on a separate processing resource.

Beyond a pure data flow representation, mode-based behavior can be expressed in this environment. This capability allows mode-based applications to be described in conjunction with data flows. Mode-based systems can be

found in telephony, avionics, missiles, and other application domains. Such applications are characterized by one or more modes (alternative run-time configurations) of system operation, each of which consists of one or more processes, which contain one or more data flow application functions. A system is in only one mode of operation at any given instant. Transitions between modes occur on explicitly defined conditions, based upon events. Each mode is characterized by a set of processes together with a specification of how they are to be scheduled and how data is to be exchanged between them.

Various features are available to allow application functions to be mapped onto FPGAs. VHDL behaviors can be described for implementation on FPGAs. Constructs are available within the environment to express VHDL processes, constants, ports, signals, and generate statements using a mixture of graphical and textual notation. The VHDL can then be synthesized using the Synopsys Design Compiler, and placed and routed using the Xilinx M1 toolset. These constructs have been employed to describe the second-level detection (SLD) portion of a synthetic aperture radar (SAR) automatic target recognition (ATR) application using VHDL. This SLD description can be implemented on a single Xilinx XC4062 FPGA.

The DADE also provides templates, language-independent library abstractions that allow the representation and customization of FPGA implementations. Templates can be parameterized and reused, reducing the time to develop applications. They can represent multiple implementations of a given function, such as a 16-bit filter, a 12-bit filter, and an 8-bit filter. This idea is useful for supporting variable precision arithmetic. They can also be used to represent implementations at different levels of abstraction and implementations for different FPGA vendors.

Various libraries can be incorporated into the environment, such as the Vector Signal Image Processing (VSIP) libraries being developed through the Cameron effort [11]. Also, various compilation tools, particularly those supporting multi-FPGA partitioning, may be incorporated in the future.

The DADE can be used to describe a configurable architecture consisting of a mix of hardware components. One possible hardware architecture is shown in Figure 4. It consists of a controller (hidden) and four DSP/FPGA boards on a VME backplane, with each board containing a SHARC DSP and a Virtex FPGA. A "databook" within the DADE allows a user to characterize the compute and bus elements in terms of performance, such as the clock speed for various processors and bandwidth of busses, as well as size, weight, power, and cost. For an FPGA, this characterization would include the number of configurable logic cells within the device and the reconfiguration time.

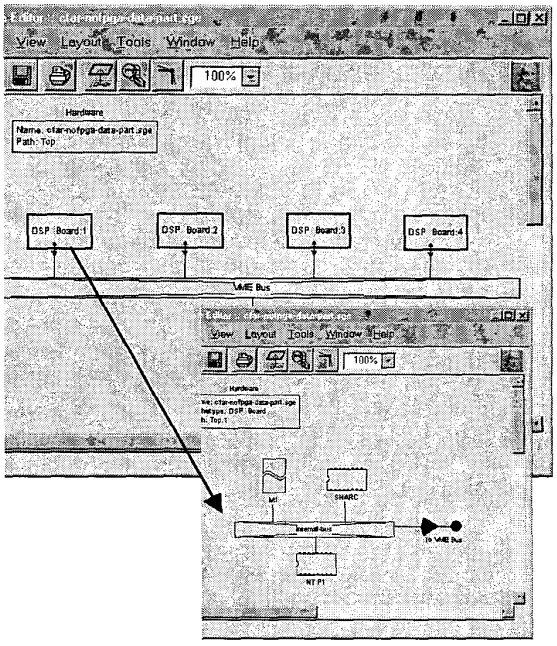

Figure 4 Constant False Alarm Rate Hardware Architecture

5. Partitioning, Mapping, and Trade-off Analysis Environment

In the design of complex, heterogeneous systems, a large design space exists, consisting of several possible alternatives. Tools and techniques are necessary that can rapidly explore this design space and evaluate alternatives with respect to multiple metrics, such as performance, size, weight, and power. This activity needs to be performed early in the design process, before a system is implemented. Also, new space-time trade-offs and implementation options must be assessed when considering configurable computing elements. To support these objectives, two tools are being developed: 1) a hardware/software partitioning and mapping tool [12], and 2) a system level modeling tool that supports performance analysis using VHDL [13].

Using the software and hardware architectures captured in the DADE, the partitioning and mapping tool utilizes genetic algorithms to examine several different mappings of software functions onto hardware resources, subject to various constraints. The tool supports a trade-off analysis between general-purpose processors, digital signal processors, FPGAs, and ASICs. Decisions regarding whether application functions should be performed on an FPGA are influenced by numerous factors. These factors include performance, communication overhead, reconfiguration overhead, frequency of reconfiguration, suitability, and flexibility, such as the identification of

functions that are likely to change in the future. Other design decisions include the type and number of processors and the topology.

The partitioning and mapping tool populates its internal data structures with information from the DADE, such as component and connectivity information as well as performance parameters. The software functions are mapped onto specific hardware resources in an iterative fashion, and a series of analyses are performed to evaluate the chosen architecture. Several criteria are included in the evaluation, including latency and schedulability. The current candidate architecture is then compared to past candidates, and the candidate is kept or thrown out based on its merits. The partitioning and mapping tool iterates until a certain number of iterations have been reached, or until a candidate architecture satisfies a set of criteria defined by the designer. Each candidate corresponds to a member of the population. User parameters can be used to control the genetic algorithm's operation. For example, parameters can be supplied that affect the maximum number of generations, the population size, the mutation rate, and the crossover rate.

The tool supports a variety of analyses: compute resource loading, bus loading, combined compute resource/bus loading, and latency. Using the multi-board CFAR application as an example, Figure 5 shows the compute resource loading analysis. This figure shows the CPU, in this case a SHARC DSP, and FPGA resource utilization for one possible mapping. This design alternative assumes that each DSP/FPGA board executes all five CFAR functions, operating on its data slice. Within each board, the DSP performs compute local statistics and detect anomalies, and the FPGA implements erode, dilate, and determine centroids. After the genetic algorithm completes, a recommended mapping is provided. We are in the process of further validating the approach using additional examples.

In addition to the partitioning and mapping technology, a VHDL-based hardware/software system level modeling environment has been developed that supports performance analysis of heterogeneous systems in terms of such metrics as throughput and latency. A comprehensive collection of library elements is available to support modeling and analysis, including FPGA models. These library elements use "tokens" to represent information and a handshaking protocol to communicate. Functionality exists to allow the simulation of time-sharing of tasks on a single FPGA, which supports either full or partial reconfiguration. These FPGA models can be simulated with other processor models, network models, and communication subsystems as part of a system level simulation.

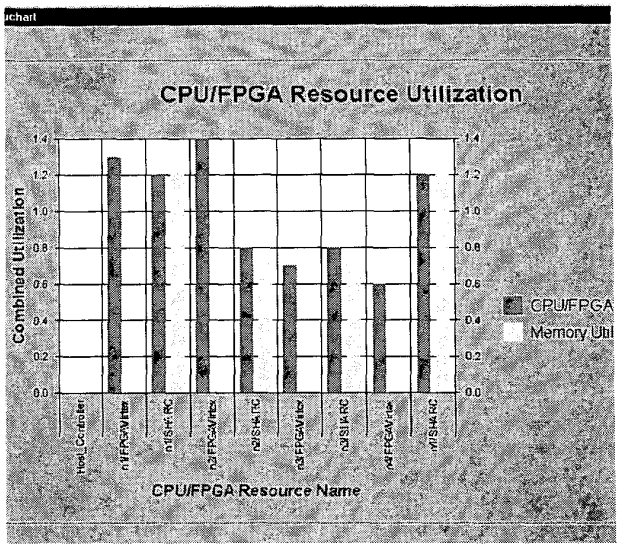

Figure 5 CPU/FPGA Resource Utilization for CFAR Application

6. Dynamic Reconfiguration Run-Time Environment

The implementation of applications is a complex activity, requiring run-time support for data flow transparency, including interfaces, and dynamic FPGA reconfiguration. In order to efficiently use configurable computing elements for large scale applications, it is essential to have a run-time environment that provides a plug-and-play transparency to the application developer for distributing functions over a heterogeneous architecture. It also must support dynamic scheduling and loading of configurations onto specific hardware elements.

A high-level view of our run-time environment is shown in Figure 6. The run-time environment is a layer of software that resides on top of the operating system and platform-specific libraries. To gain further insight into run-time environments required for heterogeneous configurable computing systems, we are exploring two related aspects. The first addresses orchestrating data flow and synchronization among application functions. The second addresses mode-based application processing and hard real-time scheduling to support dynamic FPGA reconfiguration. As a part of the latter, partial reconfiguration ideas are being explored as well. This run-time environment is being demonstrated on a variety of platforms.

One important function of the DRRTE is to provide data flow transparency among application functions implemented in software running on processors and FPGAs. In other

words, it must seamlessly implement the data flows specified in the application, regardless of whether the functions are being performed on a general-purpose processor or an FPGA. Three important, related elements that support data flow transparency are glue code generation, data flow sequencing using run-time tables, and application programming interface (API) wrappers.

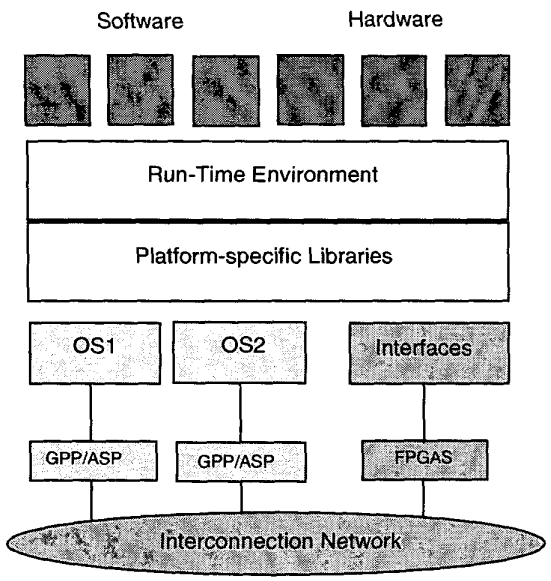

Figure 6 Dynamic Reconfiguration Run-Time Environment

As part of the data flow and synchronization exploration, we have implemented a CFAR application consisting of single thread functions. This application was implemented on a 120 MHz Pentium-based PC with a PCI-based Annapolis Micro Systems WILDFORCE board employing a Xilinx XC4025 and four XC4013 FPGAs. The application was captured using the DADE, and run-time tables along with glue-code were automatically generated. An important aspect of this design and run-time environment is that it allows for flexible mapping of any function in the application to any processing element (Pentium or FPGA) by using menu options in the application editor. Based upon the desired mapping, the DADE generates the appropriate tables and glue-code, and the relevant hardware interface libraries are linked. This allows for the development of portable applications.

Using this approach, we implemented two versions of CFAR. In the first version, all functions were implemented using library primitives in software on the Pentium. In the second version, the erode and dilate functions were mapped onto a single Xilinx XC4013 FPGA within the WILDFORCE board, while the remaining functions were mapped onto the Pentium. No effort was made to try and optimize the FPGA implementations. Although the erode and dilate functions collectively consume approximately 20% of the computation's execution time, this mapping decision was based on the knowledge that the computations performed by these functions would map well to an FPGA. We observed a factor of 4x to 6x speedup for the combined erode and dilate functions, across the eight SAR frames of input data, which included the communication overhead. The end-to-end execution time (latency) improvement was about 20% for the largest SAR frame, which contained 76 objects. Although there are several ways of improving the performance, our intent was to illustrate the ease with which applications could be implemented across multiple platforms.

Another aspect of run-time environments that we are investigating is mode-based dynamic FPGA reconfiguration. As mentioned earlier, a mode defines an alternative run-time configuration for a system [14]. It consists of a collection of processes that are concurrently available for dispatching and execution at run-time. The set of processes that are active during a given instant in time may change as the mode changes. For example, suppose a system is in mode A of operation, and processes P1, P2, and P3 are active. Now, suppose an event occurs that causes the system to transition to mode B. In this mode, another set of processes may be active, for example, P1, P4, and P5.

Within the context of heterogeneous configurable computing systems, it is possible that the process which is currently active contains a function implemented on an FPGA. In the example above, during mode A, process P2 may contain a function implemented on an FPGA. When the system transitions to mode B, process P4 may contain a function that must be downloaded into an FPGA. Thus, there is a need to support mode-based dynamic reconfiguration of FPGAs within the context of such systems. Another motivation for exploring mode-based dynamic reconfiguration is that it allows an application's execution to be tailored to the set of resources available during a particular mode of operation. Although the notion of a mode has been described within the context of system modes of operation, in general, a mode can be defined arbitrarily by the user.

To study the concept of mode-based dynamic reconfiguration using a non real-time application, an example was developed based on an image compression algorithm [9] containing the following steps: 2D-wavelet, quantization, run-length encoding, and entropy coding. Although the application is not inherently mode-based, it was structured in this manner to explore mode-based dynamic reconfiguration ideas. This application was implemented on the same platform as the CFAR application. The 2D-wavelet and quantization functions

were downloaded into a single FPGA sequentially, based on the mode of operation, and the remaining functions were implemented on the Pentium. Events triggered the execution of the functions. For those functions implemented on an FPGA, the arrival of an event would cause the download of an FPGA bitstream using a WILDFORCE API and subsequent execution of the function on the FPGA.

7. Demonstration and Evaluation

Additional applications and platforms are being considered to help demonstrate and evaluate the technologies under development. To further demonstrate the mapping technology, the CFAR application is being implemented on other platforms beyond the PC/WILDFORCE.

Work is underway to map the CFAR application onto a platform that consists of a modified CSPI board containing PowerPCs and a SLAAC-2 board [15]. In particular, a two node data parallel implementation is being developed (see Figure 7), where some of the computations will be executed on the PowerPC, and others will be executed on the FPGAs within the SLAAC-2 board. The computations within CFAR involve access to neighboring pixels of data. In parallel implementations, this data might be distributed across different processing elements (PEs). The trade-off in parallelizing these computations is between data overlap (redundant computations) and communication of intermediate results. A larger data overlap region implies more redundant computations but less communication of results. By contrast, a smaller data overlap region implies less redundant computations but more communication between the PEs.

SLAAC-2 is an embedded FPGA-based accelerator board. It is implemented as a 6U VME mezzanine board designed to plug into a modified CSPI M2621S baseboard carrier. The CSPI carrier includes two PowerPC processors, two Myrinet network interfaces, and a network switch that connects the two nodes and provides external network connections. As the SLAAC-2 architecture diagram in Figure 8 shows, there are actually two accelerators on the SLAAC-2 board, nodes A and B, each of which is controlled by one of the two PowerPCs on the M2621S.

The SLAAC-2 design uses Xilinx XC4085 FPGAs, labeled IF, for the PowerPC bus interface and to implement control registers and FIFOs visible to the application programmer. The PowerPC bus is a 40 MHz bus with 64 bits of data and 32 address lines. The IF FPGA is not intended to be programmed by the application programmer. The application programmer programs the FPGAs labeled X0, X1, and X2. X0 is a Xilinx XC4085 FPGA, and is intended to perform data distribution and computation. X1 and X2 are Xilinx XC40150 FPGAs and are intended to do the bulk of the computation. X0, X1, and X2 are all connected with a bi-directional ring, and there is also a shared bus between all three FPGAs. Each line of each interconnection bus can be programmed to send data in any direction. The board contains a clock synthesizer that allows user designs in X0, X1, and X2 to run at up to 100 MHz.

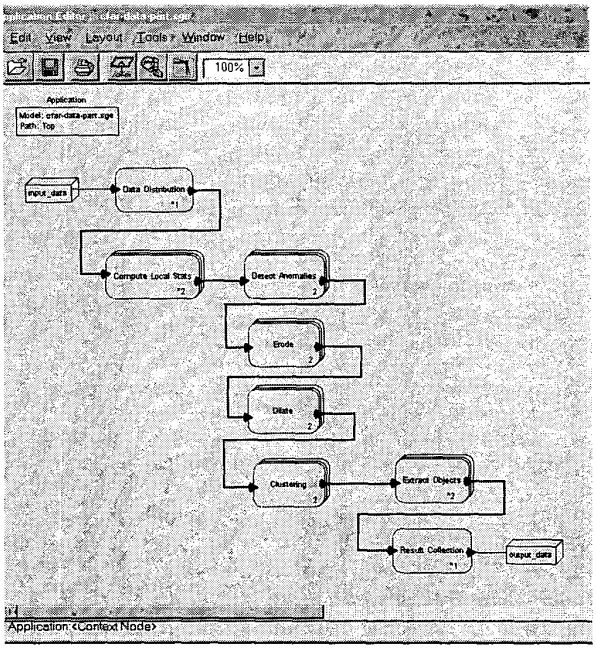

Figure 7 Data Parallel Representation of CFAR

Although the two nodes are intended to be independent, there are two 40-pin busses between X1A and X2B, and X1B and X2A. Also, although the A and B designs have separate tunable clock synthesizers, a side-effect of having a single reference oscillator on SLAAC-2 is that it will allow the two designs to operate synchronously with respect to each other. In any event, cross clocking was performed to spare pins in the compute FPGAs so that X1A and X2A have access to the B design's clock and vice versa with X2A and X2B. This permits cooperation between the two adjacent nodes.

Figure 9 is a photo of the component side of SLAAC-2. A total of six FPGAs is visible on this side. Two additional FPGAs on the back are not shown. The long connectors visible that nearly span the length of the board are the PowerPC bus connectors.

Another application being investigated is second-level detection (SLD). SLD, the most compute-intensive portion of Sandia National Laboratory's implementation of synthetic aperture radar (SAR) automatic target recognition (ATR), takes regions of interest from SAR imagery and produces hypothetical target matches. Current laboratory

approaches for implementing SLD employ general-purpose processors, such as a PowerPC. The programming environment will be used to develop multi-FPGA representations of SLD for the WILDFORCE board and the SLAAC-2 board.

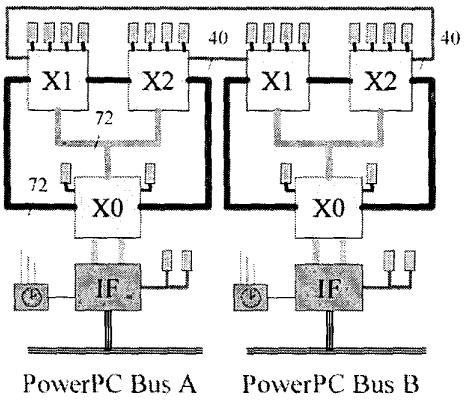

Figure 8 SLAAC-2 Architecture Diagram

To further explore mode-based dynamic FPGA reconfiguration, a missile tracking application is under investigation, loosely based on some work being performed at the U.S. Army Aviation and Missile Command (AMCOM) [16]. This application consists of three primary modes of operation: front-end processing, midcourse, and terminal tracking. Partial reconfiguration ideas are being investigated, such as the ability to reduce the reconfiguration overhead time between mode switches and the ability to overlap the computation and reconfiguration overhead through appropriate scheduling techniques. The application is being implemented on a VME-based platform consisting of Wind River System's VxWorks, a Motorola MVME2604 board (PowerPC 604), and an Annapolis Micro Systems WILDSTAR board with three Virtex FPGAs. A digital signal processing board will be added in the future. Our investigations of the run-time aspects described earlier will be leveraged in the implementation of this application.

Considerable interest exists in the use of configurable computing technology for space applications, such as earth observing satellites [17]. Unmanned satellites can benefit from configurable computing technology. The ability to adapt, evolve, and repair satellites "autonomously" is desirable, allowing them to operate more efficiently, to behave more intelligently, and to function in a more robust manner. For example, repair may be needed either to correct generic problems that escaped detection in design verification, or for problems due to a particular onboard component fault. In an effort to explore the domain of satellites, hardware/software partitioning and mapping trade-offs will be considered for a space-based radar application.

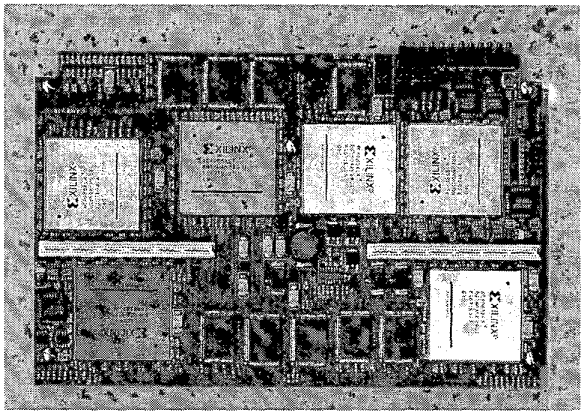

Figure 9 SLAAC-2 Photo

8. SUMMARY

Configurable computing technologies hold promise in their ability to provide cost-effective, flexible, high performance solutions and to serve as an important component of adaptive computing systems. As configurable computing technologies evolve, system level programming and development environments will be critical to harness the benefits this technology has to offer. The ADAPTERS effort is an approach to providing such an environment for heterogeneous configurable computing systems. The dynamic FPGA reconfiguration technology being developed has important applications, such as the repair, upgrade, and improved performance of satellites. An important technology to support such applications is the development of radiation-hardened reconfigurable FPGA (RHrFPGA) processing technology [18].

Broadly speaking, although configurable computing technologies are promising, several outstanding issues exist. From a practical standpoint, these issues include the acceptance of configurable computing technology for safety critical systems, particularly certification and verification of correct behavior. For many systems, the time at which reconfiguration is performed is an important concern. Another issue is the large reconfiguration time that currently exists for many FPGAs. In general, this problem is exacerbated as the size of the device increases. Some FPGAs, such as the Xilinx Virtex device, support partial reconfiguration. One benefit of partial reconfiguration is reduced reconfiguration time. We will be evaluating this technology, particularly for the SLD and missile tracking applications.

Also, evaluating the benefit of configurable computing technology is not always a simple matter of examining performance only. For heterogeneous configurable computing systems, it requires considering the hardware/software cost and complexity to support dynamic

reconfiguration. Finally, as hinted above, configurable computing technologies, particularly those based on FPGAs, should be considered within the context of other adaptive computing technologies, such as adaptive resource management [19].

ACKNOWLEDGMENTS

This work has been supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. F33615-98-C-1320 and agreement number F30602-97-2-0220. We would like to thank our colleagues for their interactions and many contributions: Subburajan Ponnuswamy, Jerry Kooyman, Kevin Driscoll, Dong-in Kang, Peter Bellows, and Shezad Okhai from the University of Toronto. We would also like to thank Kerry Hill and Ralph Kohler from AFRL.

REFERENCES

[1] J. Villasenor and B. Hutchings, "The Flexibility of Configurable Computing," *IEEE Signal Processing Magazine*, Vol. 15, No. 5, 67-83, September 1998.

[2] A. Thompson, *Hardware Evolution – Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution*, Springer-Verlag, 1998.

[3] S. Kumar, J. H. Aylor, B. W. Johnson, and Wm. A. Wulf, *The Codesign of Embedded Systems – A Unified Hardware/Software Representation*, Kluwer Academic Publishers, Boston, 1996.

[4] K. Bondalapati, P. Diniz, P. Duncan, J. Granacki, M. Hall, R. Jain, and H. Ziegler, "DEFACTO: A Design Environment for Adaptive Computing Technology," *Proceedings of the 6th Reconfigurable Architectures Workshop*, 570-578, April 1999.

[5] P. Banerjee, A. Choudhary, S. Hauck, N. Shenoy, C. Bachmann, M. Chang, M. Haldar, P. Joisha, A. Jones, A. Kanhare, A. Nayak, S. Periyacheri, M. Walkden, "MATCH: A MATLAB Compiler for Adaptive Computing Systems," submitted to *Computer*.

[6] T. Bapty, J. Scott, S. Neema, J. Sztipanovits, "Uniform Execution Environment for Dynamic Reconfiguration," *Engineering of Computer Based Systems Conference*, March 1999.

[7] P. Bellows and B. Hutchings, JHDL – An HDL for Reconfigurable Systems, *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, 175-184, April 15-17, 1998.

[8] M. Jones, L. Scharf, J. Scott, C. Twaddle, M. Yaconis, K. Yao, P. Athanas, and B. Schott, "Implementing an API for Distributed Adaptive Computing Systems," *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, April 1999.

[9] S. Kumar, L. Pires, D. Pandalai, M. Vojta, J. Golusky, S. Wadi, and H. Spaanenburg, "Benchmarking Technology for Configurable Computing Systems," *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, 273-274, April 15-17, 1998.

[10] D. Bhatt and J. Shackleton, "A Design Notation and Toolset for High-Performance Embedded Systems Development," *Lectures on Embedded Systems*, LNCS Tutorial, Volume 1494, Springer-Verlag, 249-267, 1998.

[11] J. Hammes, B. Rinker, W. Bohm, W. Najjar, B. Draper, R. Beveridge, "Cameron: High Level Language Compilation for Reconfigurable Systems," *Conference on Parallel Architectures and Compilation Techniques*, Newport Beach, CA, October 12-16, 1999.

[12] B. Iyer et al., "Distributed, Embedded, Real-Time Computer System Analysis and Integration," *1999 INCOSE Symposium Proceedings*, Brighton, U.K.

[13] F. Rose, J. Shackleton, and C. Hein, "Performance Modeling of System Architectures," *Journal of VLSI Signal Processing*, 15, 97-109, 1997.

[14] S. Vestal, "Mode Changes in a Real-Time Architecture Description Language," *International Workshop on Configurable Distributed Systems*, March 1994.

[15] B. Schott, S. Crago, R. Parker, L. Carter, C. Chen, J. Czarnaski, M. French, I. Hom, T. Tho, and T. Valenti, "Reconfigurable Architectures for System-Level Applications of Adaptive Computing," *VLSI Design*, Special Issue on Reconfigurable Computing, to be published.

[16] Private conversations with Dr. Richard Sims and Dr. Monte Helton, AMCOM.

[17] M. Figueirido, P. H. Stakem, T. P. Flatley, and T. M. Hines, "Extending NASA's Data Processing to Spacecraft," *Computer*, Vol. 32, No. 6, 115-118, June 1999.

[18] J. McCabe, "Radiation Hard Reconfigurable Field Programmable Array," *Proceedings of MAPLD '98*, Greenbelt, Maryland, September 15-16, 1998.

[19] D. I. Rosu, K. Schwan, S. Yalamanchili, and R. Jha, "On Adaptive Resource Allocation for Complex Real-Time Applications," *Proceedings of the 18th IEEE Real-Time Systems Symposium*, San Francisco, California, December 1997.

Dr. Sanjaya Kumar *is a Principal Research Scientist at the Honeywell Technology Center in Minneapols, Minnesota. His interests include adaptive computing systems, hardware/software codesign, complex system modeling, computer architecture, fault-tolerant systems, and parallel and distributed systems. He is currently the Principal Investigator of the following DARPA Adaptive Computing Systems programs: "Benchmarking Tools and Assessment Environment for Configurable Computing" and "ADAPTERS." He is the principal author of a book on hardware/software codesign titled "The Codesign of Embedded Systems - A Unified Hardware/Software Representation." He is a co-editor for the Integrated Engineering Department in IEEE Computer and is a community faculty member at Metropolitan State University in Minneapolis.*

Dr. Devesh Bhatt *is a Research Staff Scientist at the Honeywell Technology Center. He is leading a team developing technology to efficiently create software for high-performance distributed computer systems. The approach is based on object-oriented domain-specific design notations and styles that help a system designer specify and constrain a high-performance software application design problem so that much of the development process can now be automated. Through a joint initiative, this same technology is being commercialized as the Systems and Applications Genesis Environment (SAGE™).*

Dr. Steve Vestal *is a Research Staff Scientist at the Honeywell Technology Center. He mostly works in the aerospace business, alternating between contract and internal research. He is the principal investigator of a DARPA/MICOM project developing languages for describing software architectures, methods and tools for formal modeling and analysis of software architectures, and methods and tools to build complex real-time systems from software components according to architectural specifications. He is also principal investigator of an AFOSR program that is developing methods to integrate real-time schedulability with concurrent process modeling and analysis.*

Dr. Stephen Crago *is a Computer Scientist in the Dynamic Systems Division of University of Southern California's Information Sciences Institute. He completed his Ph.D. in Computer Engineering from the University of Southern California in 1998, where he developed the HiDISC architecture, a microprocessor architecture designed to tolerate memory latency. He has been a Computer Scientist since October 1997 at ISI East, where he has worked on adaptive computing, SAR/ATR algorithm implementations, embedded parallel computing for space, and a memory-intensive multiprocessor architecture.*

Mr. Robert Parker *is Deputy Director of the Information Sciences Institute and Director of ISI-East, a newly formed laboratory focused on system integration issues drawing upon and augmenting the core research competencies of ISI. Mr. Parker was most recently Deputy Director of the Information Technology Office at DARPA where he developed the concepts for the existing DARPA program in Adaptive Computing Systems as well as helped form the technology vision in this area. Mr. Parker also managed several large-scale consortium activities in wireless systems, automotive electronics, and electronic packaging, demonstrating the ability to manage complex and diverse technology teams. He has 26 years of experience in electronics system design and line management.*

Mr. Gary Gardner *is the Program Manager for Honeywell's Discoverer II effort. He has been an engineer at Honeywell for 26 years. Mr. Gardner has contributed innovations in computer architecture, ASIC design, space-based communications, and imaging payload processing. This work has included Space Shuttle and several defense observation spacecraft. Most recently, Mr. Gardner has pioneered the development of ASIC designs and design processes, which translate some of the most recent computer architecture developments on the ground into radiation hardened implementations for space use.*