A Benchmark Suite for Evaluating Configurable Computing Systems - Status, Reflections, and Future Directions

S. Kumar, L. Pires, S. Ponnuswamy,
C. Nanavati, J. Golusky, M. Vojta,
S. Wadi, D. Pandalai
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
(612) 951-7107
skumar@htc.honeywell.com

H. Spaanenburg
Mercury Computer Systems, Inc.
199 Riverneck Road
Chelmsford, MA 01824-2820
(978) 256-0052
hspaanenburg@mc.com

ABSTRACT

This paper presents a benchmark suite for evaluating a configurable computing system's infrastructure, both tools and architecture. A novel aspect of this work is the use of *stressmarks*, benchmarks that focus on a specific characteristic or property of interest. This is in contrast to traditional approaches that utilize *functional benchmarks*, benchmarks that emphasize measuring end-to-end execution time. This suite can be used to assess a broad range of configurable computing systems, including single configurable devices, multiple configurable devices, and mixed architectures, such as fixed-plus-variable devices and hybrid systems. In addition, aspects that are particularly relevant to the domain of configurable computing, such as run-time reconfiguration and variable precision arithmetic, are considered. The paper provides an overview of the benchmark suite, presents some implementation results on an Annapolis Micro Systems WILDFORCE board, reflects on the benchmark suite developed, and briefly describes future work.

Keywords

Adaptive computing systems, configurable computing systems, benchmarks, stressmarks, methodology, specifications.

1. INTRODUCTION

A *configurable computing system* refers to a system that can be changed in basic computational structure, either statically or dynamically, without adding physical hardware devices. In its current realizations, such a system consists of a set of general-purpose processors augmented with a set of coprocessors, for example, uncommitted field-programmable gate arrays (FPGAs). An overview of current configurable computing technology can be found in a recent article [26].

Configurable computing architectures can be used to speed up dedicated activities, such as inner-loop computations. Using such architectures, operations traditionally performed by software running on a general-purpose processor can be implemented by an FPGA. The decision to put certain algorithms in an FPGA versus software on a general-purpose processor is based on such factors as system cost, overall performance, cost of dynamically loading the FPGAs, and the degree of interaction between the general-purpose processors and the FPGAs. Research efforts in the area of hardware/software co-design [20] are investigating approaches for performing this hardware/software partitioning.

In addition to being an approach that allows computational elements (for example, FPGAs) to be programmed to perform different tasks, configurable computing provides a niche in the processing spectrum, which includes general-purpose processors, application-specific processors (such as digital signal processors, graphics processors, and butterfly processors), FPGAs, and dedicated application-specific integrated circuits (ASICs). It offers additional degrees of design freedom, due to the presence of numerous space-time trade-off opportunities. Also, as a technology, it has the potential to reduce overall system life-cycle and maintenance costs.

Although much attention has been given to the development of new, innovative architectures for configurable computing [3][30] and investigating applications of configurable computing [13][27], little effort has been invested in developing assessment techniques for configurable computing systems. These techniques can be used to determine which systems will best satisfy a user's overall requirements. Benchmarking is commonly used for evaluating the hardware and software of general-purpose computer systems and parallel machines. As configurable computing technology matures and as new configurable architectures emerge, the use of appropriate benchmarks will play an increasingly important role in evaluating configurable computing systems. The need for objective configurable computing benchmarks has been expressed by several researchers [21].

This paper presents a benchmark suite for assessing configurable computing systems. An important goal is to provide benchmarks that expose as much information as possible about a configurable computing system's infrastructure, both tools and architecture. The intent of these benchmarks is not solely to compare competing architectures, but rather to provide insight regarding specific properties of configurable computing systems. The benchmark suite being developed leverages the work performed in the C^3I Parallel Benchmark Suite (C3IPBS) program [19], which addressed the development of a suite of benchmarks for a variety of critical C^3I (command, control, communications, and intelligence) applications on various parallel machines. Several benchmarking concepts from the C3IPBS program have been applied to the domain of configurable computing, such as the development of a benchmarking methodology, benchmarking procedures, unbiased specifications, and acceptance tests for ensuring that an implementation satisfies a benchmark specification.

In addition to filling a critical need and helping the development and propagation of configurable computing into embedded high-performance computing (HPC) systems, the benchmark suite provides the following benefits:

- Establishes an initial suite of benchmarks across a broad range of applications and HPC technologies;
- Establishes a uniform benchmarking methodology to allow meaningful comparison of results across implementations and solutions;
- Provides a resource for the adaptive computing systems (ACS) community, through the public dissemination of benchmarks and related information;
- Provides a tool for developers and vendors to evaluate their products, prototypes, and systems.

The remaining sections of this paper are organized as follows. Section 2 describes related work. In Section 3, our approach to the development of benchmarks for configurable computing systems is described in more detail. In Section 4, the benchmarks being developed are discussed, along with various aspects of the methodology. Section 5 provides a few implementation results and presents status. Some thoughts on the benchmark suite are provided in Section 6. Finally, in Section 7, conclusions are discussed.

2. RELATED WORK

Four types of benchmarks [16] are typically used in the evaluation of a system's performance: 1) "real" programs - actual programs to be executed on the system; 2) kernels - small, key pieces extracted from real programs; 3) "toy" benchmarks - simple, easy to run programs whose result is known a priori; and 4) synthetic benchmarks - programs which are constructed to be representative of a larger set of programs. These benchmark types are listed in decreasing order of accuracy for predicting performance.

A variety of benchmarks have been developed to evaluate computer systems. Widely used benchmarks include the Standard Performance Evaluation Corporation (SPEC) benchmarks [15], LINPACK [12], and the NAS parallel benchmarks [5]. These benchmarks are used to measure the performance of a computer's integer and floating-point computations, among other things. Such benchmarks are representative of *functional benchmarks*, since they emphasize measuring end-to-end execution times.

Several benchmarks have been developed to evaluate Electronic Design Automation (EDA) tools [7]. These benchmarks are used to assess circuit partitioning, high-level synthesis, logic synthesis, physical design, and circuit simulation tools. Some of these benchmarks have also been utilized to evaluate various aspects of configurable elements as well, such as place and route tools.

Within the domain of configurable computing systems benchmarking, the following works are particularly relevant. The Programmable Electronics Performance Corporation (PREP) benchmarks [23] are used to evaluate both synthesis tools and programmable logic devices. The RAW Benchmark Suite [4] is a collection of benchmarks used to evaluate overall performance and scalability of a configurable computing system. Brown et al. [9] have used a logic synthesis benchmark suite to investigate FPGA routing architectures and CAD tools.

The benchmarks presented in this paper differ from these works in several respects. Our approach introduces the concept of *stressmarks*, benchmarks that focus on a specific characteristic or property of a configurable computing system's infrastructure. This approach requires identifying a specific property of interest to be assessed, developing a stressmark that adequately measures this property, and deriving associated metrics of interest. The approach is fundamentally different from existing benchmarking approaches for configurable computing. It arises from the need to consider the numerous aspects typically associated with configurable computing tools and architectures.

Our benchmarking approach addresses a broad class of configurable architectures and issues. We are looking at configurable architectures that consist of a single configurable device, multiple configurable devices, and mixed architectures, such as fixed-plus-variable devices and hybrid systems that contain, for example, general-purpose processors, digital signal processors, and FPGAs. In addition, we address issues such as run-time reconfiguration and variable precision arithmetic, which are important aspects of configurable computing systems.

Our effort leverages existing benchmarking approaches for parallel computers and adds rigor to the benchmarking process. We are utilizing many of the concepts and ideas developed in the Rome Labs C3IPBS program and applying them to the configurable computing domain. Specifically, we are developing a benchmarking methodology and unbiased benchmark descriptions, which specify inputs, outputs, acceptance tests, metrics, and information to be reported.

3. BENCHMARK DEVELOPMENT APPROACH

Our structure for the development of configurable computing benchmarks is driven by the three main users of the benchmarks, and their respective motivations:

- ***Agencies that invest in configurable computing technology*** - to evaluate the progress of configurable computing technology;

- ***The developers of configurable computing technology*** - to inexpensively evaluate the designs of their components against a broad set of application requirements;
- ***Application system developers and procurement agencies*** - to evaluate the performance of applications and select the right technologies for their systems.

The overall benchmarking approach for configurable computing systems consists of developing a benchmarking methodology, benchmark specifications, and public dissemination of the benchmarks. Specifically, the approach for developing the benchmark suite includes performing the following tasks:

- ***Requirements Summary*** - review a broad range of applications and metrics to identify benchmarks that are relevant to configurable computing;
- ***Methodology Development*** - develop a rigorous methodology which can be applied across all benchmarks, including procedures, metrics, measurement, and reporting of information;
- ***Benchmark Development*** – develop unbiased benchmark specifications that focus on what is to be implemented, not how;
- ***Benchmark Evaluation*** - implement each benchmark on a target configurable computing platform to validate the specification;
- ***Public Dissemination*** - address the public availability of the benchmarks.

4. BENCHMARK DETAILS

This section discusses the benchmark suite and some elements of the benchmarking methodology. At this time, six benchmarks have been developed. The current benchmark suite consists of five stressmarks and a computer-aided design (CAD) benchmark. First, an overview of the benchmark suite is provided. Next, a benchmark specification template used to describe the benchmarks is discussed. Benchmark metrics and their measurement are described, followed by a discussion on submission of results.

4.1 Benchmark Suite Overview

In this effort, the following characteristics of a configurable computing system have been identified: versatility, capacity, timing sensitivity, scalability, and interfacing. A stressmark has been developed for evaluating each one of these properties. It should be noted that this list is not necessarily "complete," and others characteristics may be relevant. Effective implementation of these benchmarks evaluates both the configurable computing architecture and CAD tools, such as those used for synthesis as well as place and route. A description of these stressmarks, along with a CAD benchmark, is provided below.

- ***Versatility*** - An important characteristic of configurable computing systems is the ability to perform several functions using a single architecture. This stressmark is a space versus time trade-off benchmark that evaluates how efficiently an infrastructure performs a distinct sequence of computations. The benchmark is based on a wavelet-based image compression algorithm [2][6][17], which accepts an input image in PGM format and outputs a compressed image. This algorithm consists of the following steps: wavelet transform, quantization, run-length encoding, and entropy coding.
- ***Capacity*** - This stressmark attempts to assess the usable capacity of a configurable computing infrastructure with a Huffman encoding algorithm [14][18][22][25]. This is accomplished by measuring the size of the largest variable bit-length look-up table that can be placed in the architecture. The stressmark reads a series of 16-bit characters from an input file, and outputs a continuous string of bits that encode all input characters using a predefined look-up table.
- ***Timing Sensitivity*** - This stressmark evaluates how well a time-critical application can be implemented by a configurable computing infrastructure. Within the context of this stressmark, a time-critical application is one that contains computational steps that must be implemented efficiently; otherwise, the overall performance will suffer. The computation to be implemented in this benchmark is a COordinate Rotation DIgital Computer (CORDIC) two-dimensional vector rotation algorithm [28][29], which has several pipeline stages. The input to this stressmark consists of a list of 16-bit vector coordinate pairs and rotation angles. The output consists of a collection of rotated vectors expressed using the same fixed-point format as used for the input.
- ***Scalability*** - The objective of this stressmark is to assess how well a configurable computing architecture can harness multiple configurable devices to solve a single problem. The application chosen for this purpose is the Fast Fourier Transform (FFT) [8][24]. In particular, this stressmark is based on a decimation in frequency fixed-point FFT. It specifies data sets of 3 different sizes and is concerned with the time required to perform an FFT of a given size as the number of devices used increases. Specifically, the input vectors are classified as small (512 complex elements), medium (4,096 complex elements) and large (16,384 complex elements).
- ***Interfacing*** - The interfacing stressmark evaluates how well a "mixed" platform, consisting of several types of hardware resources, addresses a complex, multi-step application. The specific kernel application chosen for this stressmark is a Constant False Alarm Rate (CFAR) algorithm [10], an adaptive, multiple stage, signal processing procedure. The algorithm accepts synthetic aperture radar (SAR) frames as input and outputs information regarding the number of objects detected, the size of the objects, and the centroids of the objects. CFAR consists of the following steps: local statistics computation, anomaly detection, erosion, dilation, and centroid calculation.
- ***CAD*** – This functional benchmark evaluates how well a configurable computing infrastructure can perform a time-consuming CAD application. The specific kernel application chosen for this benchmark is Boolean satisfiability (SAT) [1][31]. Given a Boolean formula in conjunctive normal form (CNF), the objective is to determine an assignment of values to the input variables such that the formula evaluates to true, or to determine that it is not possible.

4.2 Benchmark Specification

The specification of a benchmark requires considerable care. A benchmark specification must unambiguously and completely state the operations to be performed, the data to be used as input, and the measurements to be performed. No other documentation should be required to implement the benchmark. To allow for public distribution, all benchmark technical data should consist

of unclassified information. The selected benchmarks should represent more than a mere functional definition. They must describe a sequence of steps that will represent the use of configurable computing in actual applications.

The benchmarks are specified algorithmically using a combination of natural language text, mathematical notation, diagrams, and pseudo code where necessary to avoid ambiguity. A specification focuses on the problem to be solved, rather than how it should be solved. This approach is similar to that used on both the NAS parallel benchmarks [5] and the C3IPBS programs [19]. The strength of this approach over specifying benchmarks as code is that the resulting specifications are more portable and less biased toward any particular architecture. Also, implementers can tailor the implementations to their platforms.

Each specification is available in a separate document. Each contains a brief introduction to the benchmark, providing information about rationale and intent. It also contains an algorithmic description of the computations to be performed, including any restrictions. The specification describes input data formats, output data formats, acceptance tests, metrics and timing procedures, information to be reported, notes, and references. The components of the benchmark specification are described in more detail below.

- ***Introduction*** - This section describes the benchmark's rationale and intent. The rationale provides justification for the choice of the benchmark, and the intent describes the characteristics of the configurable infrastructure to be exposed.
- ***Algorithmic Description*** - In this section, a functional description of all operations to be performed is provided, including algorithmic restrictions, mandatory steps, and references to relevant literature.
- ***Input*** - This section includes a discussion of the input to be used by the benchmark implementation, its format, and where the input is to come from.
- ***Output*** - This section includes a discussion of the output to be generated by a benchmark implementation, its format, and where the output is to be stored.
- ***Acceptance Test*** - This section describes the conditions that have to be met in order to consider the results of the benchmark valid. It defines tests that ensure the benchmark results are within acceptable tolerances.
- ***Timing*** - This section describes how timing is to be performed, what must be reported about timing services, when timing is to start, and when timing is to end.
- ***Reporting of Results*** - This section describes what information is to be reported. The primary goal is reproducibility. The information that needs to be reported includes specifics regarding hardware and software configuration used, software tools, metrics tabulated, source code used as a basis for the implementation, and outputs of the implementation.
- ***Notes*** - This section contains any notes that should be recorded, for example, any tips or caveats regarding the benchmark implementation.
- ***References*** - This section contains citations of any relevant articles and papers.

Some features have been added to the benchmark specifications that are worth mentioning. For example, some benchmarks, such as the versatility stressmark, have low, medium, and high complexity versions. This approach allows implementers considerable flexibility regarding how much effort should be invested in implementing the benchmark. Also, in many benchmarks, implementers can provide "automated" solutions, "manual" solutions, or a combination of both. For example, an implementer may choose to evaluate the quality of a solution by performing "automatic" synthesis and then manually performing place and route.

Each benchmark specification is accompanied by sample "C" and VHDL code. The "C" code aids in clarifying the specification and provides a first order validation of the specification. The VHDL code aids in ensuring that the specification can be implemented with reasonable effort. Also, the "C" and VHDL code can be used to reduce the effort required to implement the benchmarks by providing a possible starting point. In some benchmarks, the VHDL code is used to provide a required starting point from which to derive an implementation.

A few points are worth emphasizing. The benchmark document is the specification, not the "C" or VHDL code. The VHDL code provided with the benchmark is not developed with the intent to provide an optimized solution. The primary purpose is to perform validation of the specification and assess "level of effort" considerations. Although implementers may use the VHDL code provided, they are encouraged to develop their own to take advantage of any features present in their configurable computing platform.

4.3 Benchmark Metrics

Each benchmark defines two classes of metrics: primary and secondary. *Primary metrics* should be optimized. *Secondary metrics* provide supplemental information. Some metrics apply to all benchmarks, for example, the area utilized by the implementation. Benchmark specific metrics to be tabulated are described in the respective documents.

At a finer level of granularity, three types of metrics are tabulated when reporting results. The first type of metric applies to the benchmark implementation. This type of metric consists of both primary and secondary metrics. For example, in the capacity stressmark, the primary metric is the largest look-up table size, since the focus is to evaluate "space" considerations. Secondary metrics associated with the implementation are the speed of performing the table look-up operation and area utilized on the FPGA.

The second type of metric is associated with tools, for example, the time to perform synthesis, as well as place and route. The last type of metric is concerned with the amount of "effort" required to implement the benchmark. This metric is difficult to tabulate and interpret, since it depends on several factors: the experience of the FPGA implementer, the type and quality of the tools, and the computing environment, among others. These last two types of metrics are considered secondary.

4.4 Measurement of Benchmark Metrics

Because projects interested in benchmarking may be at different stages of development in terms of tools and/or architecture, it is desirable to support a flexible approach for measuring benchmark metrics, particularly timing information. In support of this objective, the required benchmark measurements can be made in one of three different ways:

- ***Simulation*** - Timing information, along with other relevant metrics, can be obtained using a simulation model of the configurable platform. If this approach is used, the following information needs to be provided: 1) the model's level of abstraction (for example, register-transfer level, logic level), 2) the basis for particular timing values (for example, estimates, back-annotated results from place and route). Specifics regarding tools and corresponding platforms employed should also be provided, along with the language used for simulation, although this is addressed in a separate section of reporting (see Section 4.5 – Software Tools).
- ***Place and Route*** - Timing information, along with other relevant metrics, such as area utilization and (in some cases) power estimates, can be obtained through the output of place and route software. Specifics regarding tools and corresponding platforms employed should also be provided, although this is addressed in a separate section of reporting (See Section 4.5 – Software Tools).
- ***Actual Platform*** - Timing information, along with other relevant metrics, can be obtained by implementing the benchmark on the platform and recording information. Time stamps can be obtained using timing routines or by implementing counters within the configurable architecture. Regardless of which approach is used, the technique for performing measurement must be explained in detail.

In general, timing for worst-case commercial operating conditions should be reported, based on the longest timing path. Any deviations from this operating condition should be noted. The definition of worst case timing in terms of process, voltage, and temperature should be provided.

4.5 Submission of Results

Two goals drive the requirements for submission of benchmark results: 1) readers should have all the data needed to interpret the results, and 2) the submitted results should be reproducible. Each benchmark specification contains a description of the information that needs to be reported (recall Section 4.2 – Reporting of Results). Specifically, the following items should be included in every submission:

- ***Hardware/software Architecture*** - A detailed description of the configurable hardware employed, and if applicable, any information about the software environment, for example, operating system. When describing hardware, particular attention should be paid to characteristics that convey information about the performance of the architecture, for example, device type, speed grade of a device, general organization of the hardware (particularly for boards), bandwidth associated with busses, communication between configurable elements, memory, and input/output information.
- ***Software Tools*** - A detailed description of the software tools employed to program the configurable architecture, including any vendor-specific libraries, along with any tools used for simulation and timing analysis, should be provided. When describing tools, platform information should be mentioned as well.
- ***Metrics*** – This information corresponds to measured values for various metrics. As indicated earlier, some metrics will be common across all benchmark implementations. Other metrics will be specific to a particular benchmark. In the latter case, the benchmark specification document contains the metrics to be reported. Any information regarding how time was measured should also be provided. One or more of the approaches described in Section 4.4 should be stated clearly, and any additional information required that is particular to the approach should be provided.
- ***Source Code*** - Any source code that is used as a basis for the benchmark implementations, and any instructions for building and running the code should be included. Details of the algorithms used in the implementation should also be provided.
- ***Output Listings*** - Output information from the benchmark runs should be provided.

5. RESULTS AND STATUS

We have developed specifications for all of the benchmarks described in Section 4.1 and have implemented the stressmarks on a configurable computing platform. Some results are presented in this section on the interfacing stressmark and the scalability stressmark. Space restrictions prevent a detailed discussion on all of the implementations and results.

Several tools were utilized to implement the benchmarks, including simulation, synthesis, and place and route. The Synopsys VHDL simulator and Design Compiler were used to perform simulation and synthesis, respectively. Also, the XACT and M1 toolsets were used to perform place and route.

The platform employed was a PCI-based Annapolis Micro Systems WILDFORCE board. The board contained a supervisor Xilinx XC4025 FPGA and four Xilinx XC4013 FPGAs. Each XC4013 had access to 2 Megabytes of local memory.

The interfacing stressmark is used to evaluate how well an infrastructure implements a complex application across a heterogeneous mix of resources. The resources may consist of general-purpose processors, special-purpose processors (for example, digital signal processors), and FPGAs. As implied by the name, one aspect being stressed is "interfaces." Interfaces include input/output and interactions between heterogeneous components. Another objective of this benchmark is to evaluate the benefit of employing configurable computing elements, such as FPGAs. Beyond synthesis and place/route tools, automatic hardware/software partitioning and mapping capabilities can be evaluated as well.

The interfacing stressmark is based on a constant false alarm rate (CFAR) kernel. 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" 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 1, 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.

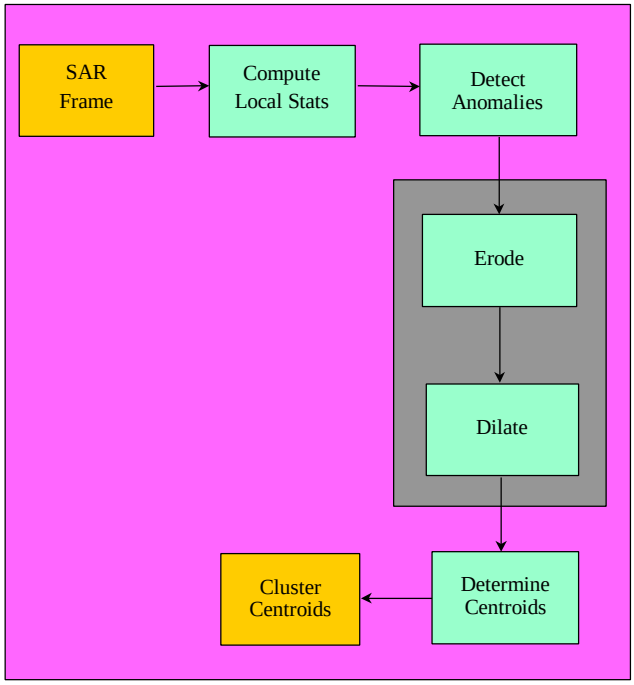

Figure 1. Interfacing Stressmark

The interfacing stressmark was implemented on a 120 MHz Pentium-based PC with the WILDFORCE board. The erode and dilate functions were manually mapped onto a single Xilinx XC4013 FPGA within the WILDFORCE board, while the remaining functions were mapped onto the Pentium. The erode and dilate functions ran at 10 MHz and consumed approximately 38% of the FPGA. It took a senior level engineer who was familiar with erosion and dilation about 2 weeks to perform VHDL coding, simulation, synthesis, and place and route of these functions.

This mapping decision was based on the knowledge that the computations performed by these functions would map well to an FPGA. No effort was made to try and optimize the FPGA implementations. The input data was stored on the Pentium, and the Pentium interfaced with the WILDFORCE board using the FIFOs available on the board.

A factor of 4x to 6x speed-up was observed 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, compared to the software solution. Although there are several ways of improving this performance, our intent was to illustrate how the benchmark could be implemented.

The scalability stressmark is used to evaluate how well an infrastructure implements an application across a multi-device platform. The devices can be any type of configurable element. The purpose of this benchmark is to evaluate how well the infrastructure addresses scaling with respect to the number of processing elements and the data size. The metrics of interest include speed-up and efficiency, where efficiency is defined as the speed-up divided by the number of configurable elements. Although this benchmark is not specific to configurable computing technology, it is of interest since many boards are used as hardware accelerators. Tools of interest include multi-device (for example, multi-FPGA) partitioning tools.

At the time that this benchmark was implemented, no multi-FPGA partitioning tools were available for the WILDFORCE board. As a result, the FFT partitioning was performed manually. VHDL was developed for 1, 2, and 4 FPGAs. The resulting description was simulated, synthesized, and placed and routed, but not fully tested on the WILDFORCE platform. The implementations consumed about 98% of each FPGA and required about 4 weeks worth of effort.

In Figure 2, simulation results are presented for the largest data set consisting of 16,384 elements. Two lines are shown. The top line corresponds to linear speed-up, and the line below represents the simulation results with corresponding speed-up and efficiency values for 2 and 4 FPGAs.

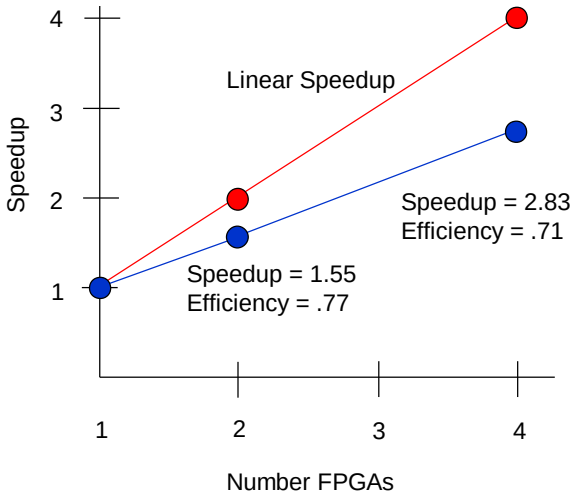

Figure 2. Scalability Stressmark Simulation Results

Note that the efficiency drops off slightly when considering a 4 FPGA implementation. This is due to a reduction in the maximum frequency of the implementation compared to a 2 FPGA implementation. The maximum frequency was affected by

an increased amount of routing resources required for communications and interactions between the FPGAs.

We are in the process of developing four more benchmarks. These benchmarks are a micro-kernel benchmark based on a discrete cosine transform, an information security benchmark based on the secure hashing algorithm (SHA-1), a variable precision arithmetic benchmark based on finite impulse response filters, and a data dependent computation benchmark based on an electronic counter-measure assessment application.

6. DISCUSSION

This section discusses some of the difficulties associated with developing benchmarks for configurable computing systems and some general insights gained as a result of performing our benchmarking activity.

Before discussing issues specific to configurable computing systems, the difficulties associated with benchmarking in general are presented. Although benchmarks are widely used in academic and industrial circles, their results are treated with skepticism. In part, this is due to the lack of rigor associated with benchmarking activities. In many circumstances, specifications are not stated precisely enough, allowing implementers too much flexibility. Also, the manner in which results are to be tabulated and reported is often not stated precisely enough, leaving too much room for interpretation. Timing is not always straightforward and requires considerable care. A sufficient granularity of computation needs to be provided, since it affects the ability to measure time accurately.

The development of configurable computing benchmarks poses several challenges. It is often difficult to separate the tools from the architecture. For example, the notion of "usable capacity" is affected by both the quality of the tools (synthesis and place and route) and the architecture. Because numerous types of configurable computing architectures exist (different logic, interconnect, routing, memory, I/O), it is difficult to develop a comprehensive suite of benchmarks that can address all of them sufficiently well.

It is a challenge to develop "scalable" benchmarks, ones that will scale with technology improvements, both in terms of capacity and speed. In fact, in the development of a benchmark specification, it is important not to be biased by the platform being used for implementation. For example, the capacity of the FPGA may influence the type of benchmark being developed.

Metrics need to be well thought out. This issue is related to benchmark development. For example, should reconfiguration time always be included as part of the overall execution time? Also, if I/O, computation, and reconfiguration are overlapped, it may make individual elements difficult to measure. In general, more work is required to try and develop uniform metrics that can be used broadly across a variety of configurable architectures.

The concept of "space" within configurable elements poses several issues. First, there is the issue of how to measure "capacity" in a uniform manner across various configurable architectures. At this time, the approach has been to extract utilization information from place and route tools. Clearly, more work is required here. The work by De Hon [11] provides some insights.

In fact, there are a couple of different notions of capacity: logic and routing. Both of these aspects need to be evaluated when considering configurable elements. In addition to the existing capacity stressmark, a stressmark that focuses on routing would be useful.

Also, evaluating capacity may not be as significant as time progresses. Although there will always be a need for high capacity devices, with the ability to perform run-time reconfiguration, users will essentially have unlimited capacity available.

Some general insights gained as a result of our benchmarking experiences include the following. Although performance is important, other system metrics, such as size, weight, and power, are equally important for some applications. This implies that when evaluating implementations, it may be worthwhile to consider combining metrics. For example, one can derive metrics such as MOPS/kilogram-watt. Focusing on a single metric may not reveal the benefit provided by a particular technology. For example, a factor of 2x increase in performance may not seem significant, but when combined with a factor of 10x improvement in power, it may.

Debugging implementations on configurable computing platforms is difficult and time-consuming. When an implementation does not perform as intended, it is difficult to know where to begin looking for the source of the error. Environments are needed to help support the debugging process, although some work in this area has started to appear.

FPGA experience makes a significant difference in the amount of effort required to implement application functions and in the quality of the implementation. It is for this reason that "level of effort" metrics are provided as supplemental information and should not be used as a basis for assessing benchmark implementation complexity. More tools are required to map complex applications onto a configurable computing platform.

7. CONCLUSIONS

Benchmarks play an important role in the assessment of hardware/software systems. Although many different types of benchmarks are available, very few address the needs of new, configurable computing technologies. These technologies include configurable architectures, such as fixed-plus-variable devices and devices that support partial reconfiguration, and software used to perform automatic synthesis, hardware/software partitioning, multi-device partitioning, and run-time reconfiguration. Benchmarks will become increasingly important as these technologies evolve.

This paper has described our effort in the development of a benchmark suite for assessing configurable computing systems. It is the first effort to specify a set of characteristics relevant to configurable computing (versatility, capacity, timing sensitivity, interfacing, and scalability). It employs a novel approach to benchmarking, which includes the use of stressmarks in conjunction with functional benchmarks. Issues pertinent to configurable computing systems, such as run-time reconfiguration and variable precision arithmetic, are being

considered. Our approach addresses a broad range of configurable architectures and tools. Also, it is based on an unbiased and technology independent formal benchmark specification methodology. It is the intent of the benchmarks to challenge implementers (hence, the name stressmarks) and critically evaluate configurable computing technology.

The benchmark suite is a critical element in the development, evaluation, and insertion of configurable computing technology in applications of interest. It allows users the ability to examine various trade-offs, refine their configurable computing platforms, and select appropriate configurable computing elements. In addition, it can be used to quantify the benefits of configurable computing technology.

Release 1.0 of the benchmark suite, which contains all of the benchmarks described in this paper, is publicly available. It can be downloaded from the DARPA/AFRL benchmarking program web site located at: www.rl.af.mil/programs/hpcbench. This benchmark suite is being used by several universities and companies to assess new configurable computing technologies, both hardware and software, being developed under the adaptive computing systems program, which is sponsored by the Defense Advanced Research Projects Agency (DARPA). A new release of the benchmark suite, Release 1.1, containing revisions to Release 1.0 and drafts of the micro-kernel and information security benchmarks is available on CD.

8. ACKNOWLEDGMENTS

We wish to thank DARPA for supporting this research under contract number DABT63-96-C-0085. We would also like to thank the following individuals for their valuable inputs and feedback regarding various aspects of the work, and their active participation: José Muñoz, Robert Parker, Sherman Karp, Ralph Kohler, Al Scarpelli, Kerry Hill, Greg Tumbush, Al Reynolds, John Villasenor, Brad Hutchings, Seth Goldstein, Herman Schmit, Srihari Cadambi, Dave Schwartz, Anant Agarwal, Walter Lee, Margaret Martonosi, Peixin Zhong, Randy Harr, Dannie Lau, Stephen Smith, André De Hon, Walid Najjar, Ranga Vemuri, Dinesh Bhatia, Praveen Chawla, Scott Hauck, Prith Banerjee, David, Zaretsky, Todd Haverkos, Zhiyuan Li, David Martinez, Bob Ford, Cory Meyers, Steve Scalera, Bill Mangione-Smith, Peter Athanas, Rhett Hudson, Wayne Dai, Tim Garverick, Charlé Rupp, Jack Jean, Karen Tomko, Don Bouldin, Maya Gokhale, Jeffrey Arnold, Mary Hall, Heidi Ziegler, Ian Page, Martyn Edwards, Gary Gardner, John Prewitt, and Kevin Driscoll. Finally, we appreciate the comments of the reviewers, which helped improve the quality of the paper.

9. REFERENCES

[1] Abramovici, M., and Saab, D. Satisfiability on Reconfigurable Hardware. Seventh International Workshop on Field Programmable Logic and Applications. September 1997.

[2] Antonini, M., Barlaud, M., Mathieu, P., and Daubechies, I. Image Coding using the Wavelet Transform. IEEE Transactions on Image Processing 1, 1992. 205-220.

[3] Athanas, P., and Silverman, H. An Adaptive Hardware Machine Architecture for Dynamic Processor Reconfiguration. Proceedings of the IEEE International Conference on Computer Design (Cambridge MA, October 1991).

[4] Babb, J., et al. The RAW Benchmark Suite: Computation Structures for General Purpose Computing. IEEE Symposium on Field-Programmable Custom Computing Machines. April 1997. 161-171.

[5] Bailey, D., Barton, J., Lasinski, T., and Simon, H. The NAS Parallel Benchmarks. NASA Technical Memorandum 103863. July 1993.

[6] Bradley, J. N., and Brislawn, C. M. The FBI Wavelet/Scalar Quantization Standard for Gray-scale Fingerprint Image Compression. Technical Report LA-UR-93-1659. Los Alamos National Lab, Los Alamos, N.M., 1993.

[7] Brglez, F. ACM/SIGDA Benchmarks Electronic Newsletter, DAC '93 Edition. June 1993, 1-22 (see http://www.cbl..ncsu.edu/benchmarks also).

[8] Brigham, O. The Fast Fourier Transform and Its Applications. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[9] Brown, S., Khellah, M., and Vranesic, Z. Minimizing FPGA Interconnect Delay. IEEE Design and Test of Computers. Winter 1996. 16-23.

[10] Castanon, D. A., Petersen, M., Ponnuswamy, S., VanVoorst, B., and Nanavati, C. Constant False Alarm Rate Benchmark Specification. Technical Information Report, Honeywell Technology Center. March 1998.

[11] De Hon, A. Balancing Interconnect and Computation in a Reconfigurable Computing Array (or, why you don't really want 100% LUT utilization). Proceedings of FPGA '99. February 1999.

[12] Dongarra, J. J. The LINPACK Benchmark: An Explanation. Supercomputing. Spring 1988. 10-14.

[13] Eldredge, J. G., and Hutchings, B. L., Run-time Reconfiguration: A Method for Enhancing the Functional Density of SRAM-based FPGAs. Journal of VLSI Signal Processing 12, 1996. 67-86.

[14] Glassey, C. R., and Karp, R. M. On the Optimality of Huffman Trees. SIAM J. Appl. Math 31, 2. September 1976. 368-378.

[15] Grace, R., The Benchmark Book. Prentice Hall PTR, Upper Saddle River, New Jersey, 1996.

[16] Hennessy, J. L., and Patterson, D. A., Computer Architecture - A Quantitative Approach. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[17] Hopper, T., Brislawn, C., and Bradley, J. WSQ Grey-scale Fingerprint Image Compression Specification, Version 2.0. Criminal Justice Information Services. Federal Bureau of Investigation, Washington, DC., 1993.

[18] Huffman, D. A. A Method for the Construction of Minimum-Redundancy Codes. Proceedings IRE 40. September 1952. 1098-1101.

[19] Jha, R., et al., The C3I Parallel Benchmark Suite – Introduction and Preliminary Results. Proceedings of the 1996 Supercomputing Conference, November 1996.

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

[21] Mangione-Smith, W. H., et al., Seeking Solutions in Configurable Computing. IEEE Computer, December 1997. 38-43.

[22] McIntyre, D. R., and Pechura, M. A. Data Compression Using Static Huffman Code-Decode Tables. Communications of the ACM 28. June 1985. 612-616.

[23] PREP Benchmarks. http://www.prep.org.

[24] Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. Numerical Recipes in "C." Cambridge University Press, 1988.

[25] Tanaka, H. Data Structure of Huffman Codes and its Application to Efficient Encoding and Decoding. IEEE Transactoins on Information Theory 33. January 1987. 154-156.

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

[27] Villasenor, J., Jones, C., and Schoner, B. Video Communications using Rapidly Reconfigurable Hardware. IEEE Transactions on Circuits and Systems for Video Technology. Vol. 5, December 1995, 565-567.

[28] Volder, J. E. The CORDIC Trigonometric Computing Technique. IRE Transactions on Electronic Computers. Vol. EC-8, 1959. 330-334.

[29] Walther, J. S. A Unified Algorithm for Elementary Functions. Spring Joint Computer Conference Proceedings. Vol. 38, 1971. 379-385.

[30] Wirthlin, M. J., and Hutchings, B. L. A Dynamic Instruction Set Computer. Athanas, P., and Pocek, K. L., editors, Proceedings of the IEEE Workshop on FPGAS for Custom Computing Machines (Napa CA, April 1995).

[31] Zhong, P., Martonosi, M., Ashar, P., Malik, S. Accelerating Boolean Satisfiability with Configurable Hardware. Proceedings of FCCM '98 (Napa Valley CA, April 15-17, 1998).