Performance Monitoring for Run-time Management of Reconfigurable Devices

Ryan A. DeVille, Ian A. Troxel and Alan D. George
{deville, troxel, george}@hcs.ufl.edu

High-performance Computing and Simulation (HCS) Research Laboratory
Department of Electrical and Computer Engineering, University of Florida
Gainesville, Florida 32611-6200

Abstract

High-performance computing (HPC) systems with hardware-reconfigurable devices have the potential to achieve major performance increases over parallel computing systems based solely on traditional processors. However, providing services upon which users of traditional HPC systems have come to depend is essential for large-scale reconfigurable computing (RC) systems to become mainstream. Along with critical needs for management services, core libraries, user-friendly interface, etc., mechanisms for system resource monitoring to support debug and performance analysis and optimization is an important feature in conventional HPC systems that is currently lacking in their RC counterparts. This paper presents the concept of hardware monitoring probes within the CARMA framework for RC-based HPC and examines several design options. Experimental results analyze probe design considerations on a case-study application.

Keywords: Reconfigurable Computing, High-Performance Computing, Performance Monitoring

1. Introduction

A trend toward RC-based HPC focused largely on Commercial-off-the-Shelf (COTS) technologies has recently developed within the RC community. The creation of RC clusters at Virginia Tech, the University of Florida, and the Air Force Research Laboratory at Rome, NY [1] among others is evidence of this trend. Indeed, the emergence of RC-enhanced "big iron" systems from SGI, SRC, and Cray marks the potential beginning of the mainstream acceptance of RC as a viable technology for HPC. However, while the addition of RC hardware has improved the performance of many stand-alone applications, achieving the full potential of RC computing platforms has been challenging due to the lack of a versatile multiuser and multitasking environment.

One particular challenge is the task of providing performance monitoring and debug of hardware tasks executing on FPGAs in the system. As RC systems gain widespread adoption, tools must be developed to provide verification of system behavior, ensure efficient use of RC resources, and remove application performance bottlenecks. Moving and adapting traditional monitoring techniques to RC systems to provide an effective, flexible, lightweight monitoring infrastructure is a key challenge in this arena.

To address hardware performance monitoring and debug and a wide variety of key issues facing RC-based HPC designs, researchers at the University of Florida have proposed the Comprehensive Approach to Reconfigurable Management Architecture (CARMA) framework [2]. CARMA provides a framework to develop and integrate key components as shown in Figure 1. With CARMA, we seek to provide and integrate all the services upon which users of traditional HPC systems have come to depend in a versatile multitasking and multiuser environment. This paper focuses on the new hardware monitoring portion of the *Performance Monitor* module in CARMA in order to highlight the design and development of FPGA hardware probes.

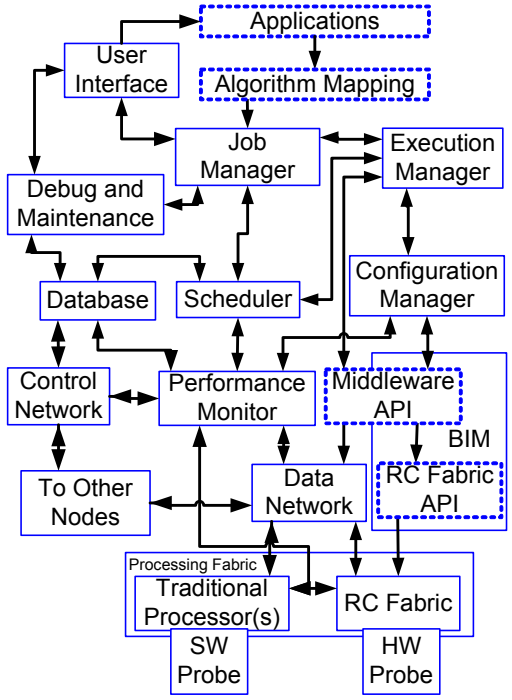


Figure 1. CARMA Framework

The remaining sections of this paper are organized as follows. Section 2 provides a background of past work in performance monitoring, while Section 3 describes the design and development of the hardware monitoring

subsystem for CARMA. Section 4 provides a detailed discussion of the experimental setup, while Section 5 presents the experimental results with analysis. Finally, Section 6 offers conclusions and directions for future work.

2. Related Research

A general background of RC is not included here since such a discussion has been covered by many sources such as Compton and Hauck [3]. As outlined in Section 1, monitoring of FPGA resources and applications is one of the critical new components within the CARMA framework that will be built upon several techniques used in modern processors and parallel systems as discussed below.

System performance monitoring has traditionally been undertaken through one of three approaches: hardware, software, or hybrid monitoring. The hardware approach typically consists of connecting physical probes from devices such as logic analyzers to system components in order to unobtrusively gather data. While highly accurate, this approach is limited due to the low-level nature of gathered data. This method is most appropriate for debugging stand-alone hardware systems and is rather inconvenient for profiling application code. Software monitoring consists of adding instrumentation code to an application to collect useful information about the system such as memory access patterns. While this approach collects hardware-independent information to perform a high-level performance and debug analysis, there is a significant performance penalty due to additional software overhead, typically resulting in inaccurate measurements than at the hardware probe level. Hybrid monitoring attempts to balance the two approaches by providing a few software instructions to access select hardware counters. This approach has the advantage of allowing a relatively high abstraction level while introducing low overhead and high accuracy. However, portability can be an issue with this implementation-specific approach [4, 5]. CARMA's performance monitor subsystem takes a hybrid approach by deploying tightly-coupled probes at the hardware level to provide accurate data to software agents.

In general, a performance monitor is an information processing agent, with the information describing time-varying relationships of select signals [6]. Information processing provided by performance monitors can be categorized into three functional components: (1) an event trigger, (2) an event handler, and (3) an accumulation or sampling storage facility [7]. An example of a simple performance monitor in a traditional processor is a cache miss counter. The event trigger for this monitor is a signal resulting from a cache miss. If the trigger is tripped, a handler responds by performing a preset action. In the miss counter example, a handler's action would be to clear the cache miss interrupt signal. After the handler processes the trigger, the state information is updated. For the miss counter, the state information would be the contents of the counter register and this value is incremented after the cache miss interrupt is cleared [7].

Triggering is the critical step in performance monitoring to ensure the accuracy of information. There are two main types of monitors based on their trigger schemes. Time-driven monitors collect state data at fixed time intervals, and event-driven monitors sample state data at predetermined events. The former approach is also known as sampling and follows the philosophy that the behavior of the system can be approximated by snapshots. The latter technique allows complete behavioral information to be captured, but with the additional overhead of storing relational information [8].

In addition to various design philosophies, performance monitors also typically employ at least three data collection methods including counting, statistical sampling, and event tracing. Counting records the number of times an interesting event occurs. While counting reduces the amount of data collected, contextual information is typically lost. Statistical sampling consists of periodic sampling of application resources to assess performance. Sampling often generates more data than the counting scheme, but has potential for inaccurate reporting as the accuracy of the data depends primarily on the sampling rate.

Event tracing is a more general approach to monitoring that merges the former two philosophies. In this scheme, the monitor generates a sequence of event records consisting of an action and its attributes [4]. Event tracing provides a wealth of information but has several drawbacks. The gathered information is typically retrieved after the application has completed and relevant information is extracted from a trace file. Therefore, bottlenecks cannot be immediately explored, forcing multiple runs for each slightly modified data set. Also, the large volume of data produced tends to limit the scheme's scalability in parallel systems [4]. Section 3 details how the design for the performance monitor probe in CARMA addresses these issues.

3. Performance Monitor Probes

Based on a thorough review of traditional processor and system monitoring schemes, several design goals have been determined for CARMA's FPGA monitor probes: 1) *provide accurate information*, failure to do so invalidates the idea of monitoring; 2) *minimize delay and area overhead*, limited resources and tight design constraints require a compact design; 3) *keep generated data to a manageable size*, it is imperative that the performance monitor strike a balance between data collection and system overhead; 4) *deliver data in a timely fashion*, critical for run-time monitoring and optimization. Insuring statistics are updated in a timely manner while not imposing a sizeable I/O overhead is a tradeoff that must be carefully balanced [4]. Having elaborated the design degrees of freedom in Section 2 and identified necessary goals for the

design of a performance monitor above, the remainder of this section describes CARMA's performance monitor.

3.1. Probe Design Overview

Two performance monitor probe types have been designed for different configuration approaches. The *Integrated RC Monitor* design is integrated into the application prior to synthesis and is designed for applications with access to the entire chip. The *Resident RC Monitor* maintains a presence on the FPGA and is designed for applications loaded via partial reconfiguration. The integrated and resident approaches are described in Sections 3.2 and 3.3 respectively and their links to other CARMA components are described in Section 3.4.

3.2. Integrated Monitor

The integrated performance monitor consists of a number of simple yet powerful probes coupled to critical points within an application prior to insertion into the RC fabric. This method builds on the common application debugging method of introducing registers to the design at critical points to sample data at run-time [5]. As noted previously, sampling has inherent errors due to the dependence of the monitor on the sampling rate, therefore a more elaborate scheme is required. Xilinx's ChipScope also uses similar probes to gain an assessment of the internal workings of an FPGA application; however, this tool is designed specifically for application prototype debugging rather than run-time system performance monitoring [9].

The integrated monitor includes the ability to sample registers, perform accurate hardware-level counts and uses a scheme based on the Intel Pentium approach to provide event traces. A block diagram of the integrated probe is shown in Figure 2. The probe includes a set of *Triggers* that are compared to input *Comparison Data* to determine when an interesting event occurs. If the data and triggers match, the *Decoder Logic* enables appropriate signals to perform various monitoring tasks. Any number of triggers (defined as the trigger depth) of any width (in bits) can be combined by the decode logic to mark distinct events. The trigger depth can be used to filter through sequences of state machines or other data in any permutation of sequences up to the trigger depth to find complex patterns.

One task the probe can perform is to sample data via the *Sample Enable* signal. All data values from the signals of interest are brought into the *Hold Register*. When a sample is taken, the value in the hold register is written to the *Sample Register* and then passed to CARMA's performance monitor. Another type of task the probe can perform is a counting. The *Count Register* is a variable-bit counter that can be configured as a free-running counter (to count elapsed time between events) or as an event counter (to log the number of times an event occurs). When an interesting event occurs, the counter is either incremented for event counting or sampled for timing. It should also be noted that both types of monitoring – sampling and counting – can occur simultaneously.

The *Global Trigger* enables coordinated monitoring across all probes. When any data sample observed by any probe matches the trigger mask, all probes capture the current sample and counter data. This scheme follows the Intel Pentium method of precise event-based sampling.

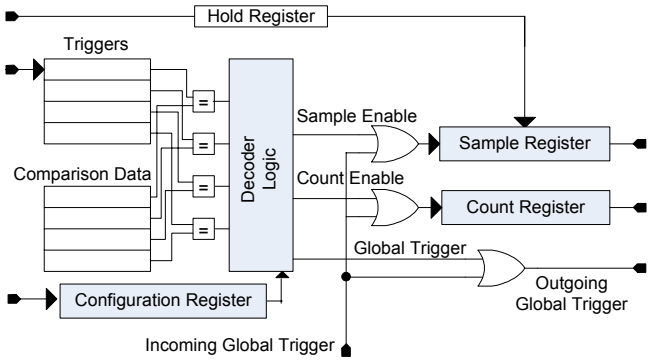


Figure 2. Integrated Probe

3.3. Resident Monitor

Another approach to RC performance monitoring provides a monitoring core constantly loaded on the FPGA to which applications loaded via partial reconfiguration attach. A resident design provides a standard interface specification through which users can monitor signals of interest. A resident monitor abstracts the user's low-level monitor design burden as compared to using the integrated monitor but at a potential cost in terms of efficient monitoring. Several resident monitor designs are currently under investigation at Florida and have not been fully developed at the time of submission. We are examining the limitations of partial reconfiguration in today's devices yet see it as an emerging technology likely to play a greater role in future RC systems.

3.4. Links to Other CARMA Modules

CARMA's hardware probes receive configuration information and pass monitor values to the Performance Monitor software agent via each particular board's Board Interface Module (BIM). The BIM handles host processor to FPGA communication, and limiting the monitor overhead on this link is critical. Data collected by the local performance monitor agent is used by a few local and numerous remote agents. For example, the local debug agent performs requested debug operations or simply collects and presents the data to the user, the maintenance agent uses this data to monitor FPGA card health, and the task scheduler uses the monitored data to perform enhanced scheduling based on information such as likely release time. CARMA's distributed information transmission system, GEMS [10], ensures that each node's local agents have up-to-date information system-wide, allowing efficient scheduling and distributed debug of parallel applications.

The impact of the hardware probe's data collection methods on other CARMA modules and services is the subject of ongoing research and beyond the scope of this paper.

As opposed to cycle-accurate simulation tools, CARMA's runtime performance monitor and probe system provides for scalable, system-wide application debug and performance monitoring on large-scale systems (i.e. a hundred or more nodes). Such runtime performance characteristics cannot be examined via simulation. Also, FPGA runtime status as required by numerous CARMA services can only be achieved via hardware probes. The next section describes the case-study experimental setup.

4. Experimental Setup

Several experiments have been performed to examine some of the tradeoffs presented in this paper to develop a viable, general-purpose, integrated monitor design. Generic probe results are discussed in Section 5.1. In addition to the generic tradeoff studies, two specialized designs are created for a case study with a representative application, Conway's Game of Life algorithm. In two separate experiments, an implementation of Game of Life is monitored, first with a centralized scheme and then a distributed scheme using a varying number of probes. The accuracy and overhead imposed by each design is measured and compared.

Game of Life mimics the birth and death progression of organisms living in a colony [11] with numerous events occurring in parallel. Our design is based on a 64×40 universe size, using the traffic light blinker at the test data set in which we monitor the state of each organism in the universe. The Game of Life probe results are shown and discussed in Section 5.2.

The probes and applications have been synthesized with Xilinx's XST tool and placed and routed by ISE Foundation 6.3. The configurations are run on a Xilinx Virtex-II 6000, speed grade -4, located on a Nallatech BenNUEY board. The BenNUEY is hosted in a Linux server with dual 2.4GHz Intel Xeon processors operating at 2.4GHz, 1GB of memory, and connected through a 32MHz, 64-bit PCI bus. A communication module between the host processor and the FPGA has been developed through Nallatech's proprietary communication scheme DIMETalk. The development software for DIMETalk provides a set of high-level data transfer components providing host processor to communication via the DIMETalk API.

5. Results and Analysis

Several experiments are performed with various generic integrated performance monitor designs to examine aforementioned tradeoffs. The generic experiments highlight the relationship between three degrees of freedom (i.e. sample, count and trigger register width) and probe overhead. After the generic probe experiments, the probes are tailored for the Game of Life for further tradeoff analysis. Game of Life is inherently distributed and parallel with a measure of randomness and is a good test case to compare the accuracy and overhead of the integrated monitor's centralized and distributed storage subsystem.

5.1. Generic Integrated Monitor Experiments

Numerous generic experiments were performed but omitted due to page limitations in this revised edition. An interesting experiment shows how the width and depth of the trigger register of a single probe affects maximum operating frequency and area overhead. The data is divided into a set of small triggers (e.g. each triggering on a signal or state) and a set of large triggers (e.g. each triggering on a data value) for clarity. The area and maximum frequency results, each with sets of small and large triggers, are shown in Figures 3 and 4, respectively. The count and sample register widths are set to 32 bits for this set of experiments.

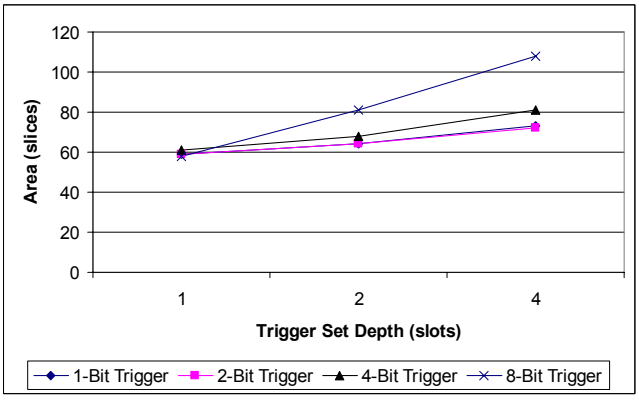
The small-trigger results show that one-, two- and four-bit triggers have about the same overhead, and the eight-bit trigger begins to impose additional area and maximum frequency overhead. The large-trigger results show that additional triggers and larger bit-widths impose additional area overhead. Also, the rate of reduction in maximum operating frequency imposed on the design tends to decrease as trigger depth increases. The critical path is directly affected by the trigger width, as the comparator and decode logic scale in direct proportion to the trigger width.

These results show that using the smallest trigger width necessary is a good design practice while choosing the optimal trigger depth to use for a given design must be explored. Also, it may be more efficient to use several smaller probes as opposed to a single large probe for some designs. It should be noted that results produced when running many of these same experiments for a Virtex-II 6000 with a -6 speed grade (the best offered for the device) instead of -4 grade produced modest maximum frequency increases on the order of 5MHz for smaller designs and 25MHz for the largest design. Area overhead is not affected by speed grade changes within a given Xilinx FPGA family. Also, it can be assumed that designs routed for newer FPGAs (e.g. Virtex 4) would likely impose less area overhead and maximum operating frequency limitations.

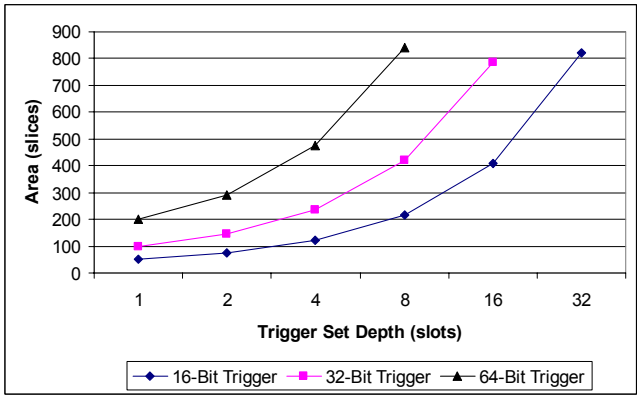
5.2. Game of Life Experiments

Several experiments for monitoring signals of interest in the Game of Life application are performed to examine accuracy and overhead tradeoffs between the centralized and distributed monitoring schemes. Figures 5 and 6 provide area and block RAM utilization results, respectively. Data points with zero number of probes represent the application without monitoring probes. As the number of probes increases, the centralized scheme uses slightly fewer block RAMs and substantially less area compared to the distributed scheme. This outcome shows that the centralized scheme is very area-efficient and scales well in terms of capacity due to a reduction in arbitration complexity and logic. It should be noted that 4000 slices

(the area overhead imposed by the 8-probe distributed design) is roughly 10% of some of the largest Xilinx FPGAs built today and would likely be close to the upper limit of an acceptable area overhead.

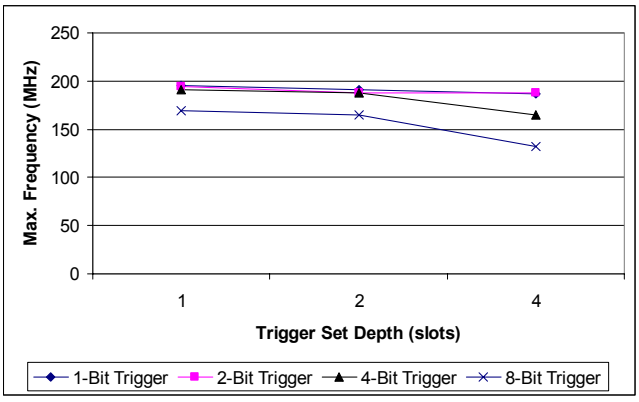


a) Small Trigger

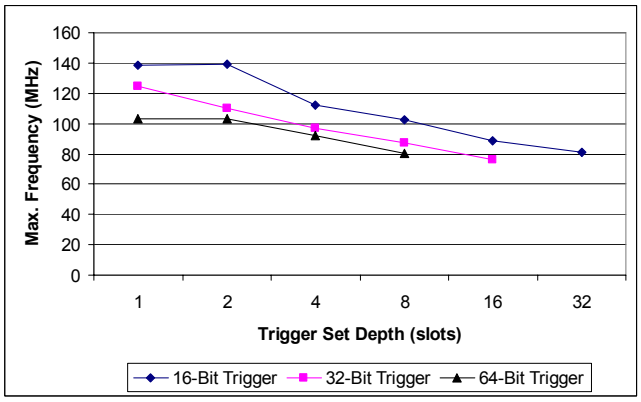


b) Large Trigger

Figure 3. Area Overhead of Trigger Probe



a) Small Trigger



b) Large Trigger

Figure 4. Maximum Frequency Overhead of Trigger Probe

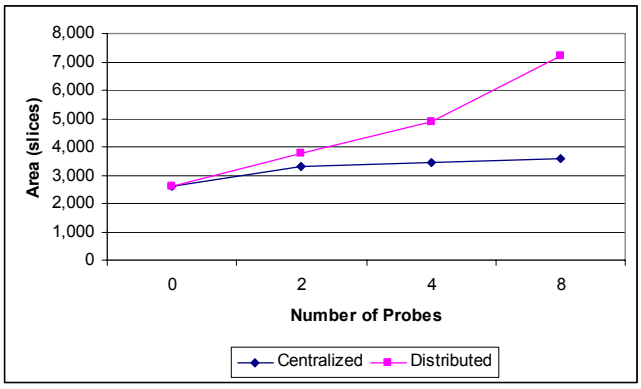


Figure 5. Area Overhead of Centralized vs. Distributed Monitoring Schemes

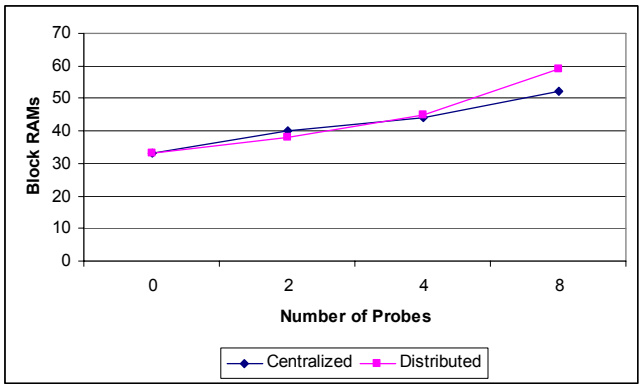


Figure 6. Block RAM Overhead of Centralized vs. Distributed Monitoring Schemes

The maximum operating frequency limits and data collection latencies are shown in Figures 7 and 8, respectively. Collection latency is defined as the time to pull all values from the central storage or all distributed storage locations. The results show that the centralized scheme's balanced routing lines provide better scalability in

terms of maximum operating frequency. The additional arbitration and routing complexity affect the distributed scheme's maximum operating frequency as seen in Figure 7. In addition, the centralized scheme provides a more efficient DMA solution, since all sample values are pulled out in one transfer, while the distributed approach requires multiple transfers (one to each probe's sample buffer) and thus incurs additional transmission overhead for each additional probe. Minimizing collection latency is critical, as this delay is pure overhead across the interconnect used for application data.

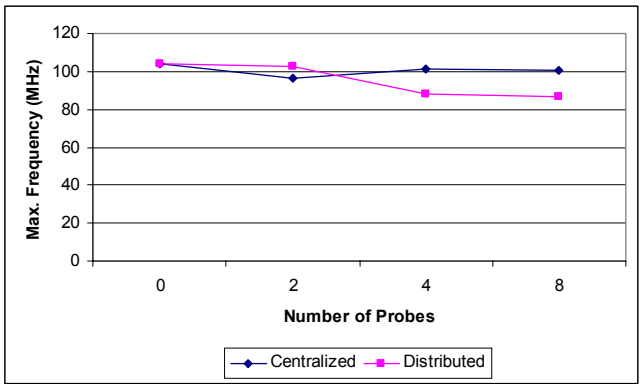


Figure 7. Maximum Operating Frequency of Centralized vs. Distributed Monitoring Schemes

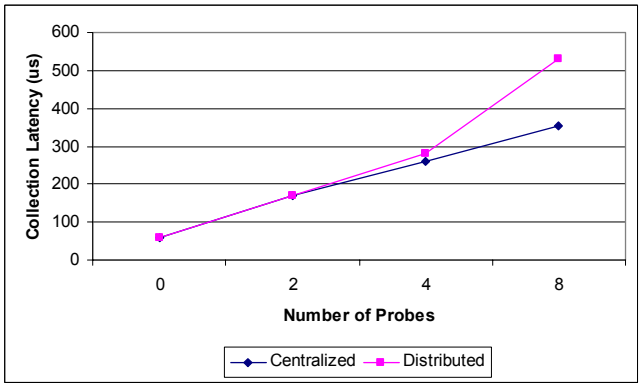


Figure 8. Collection Latency of Centralized vs. Distributed Monitoring Schemes

Unfortunately, due to contention at the central probe data storage, the centralized approach tends to drop sample values and thus may provide inaccurate monitoring results. While a probe is waiting for its turn to write to the central storage, it is unable to sample any additional data. For the Game of Life experiments, the two-probe centralized monitor did not drop any data, but the four-probe version dropped 36% of potential samples and the eight-probe version dropped 68%. A priority scheme has been added to the central storage in order to improve probe reliability in select regions, but data loss remains likely with the centralized scheme. The distributed scheme did not drop any data values in these experiments but could potentially do so if values are produced faster than they can be pulled out of a given probe's buffer.

Based on the Game of Life results, the centralized approach is found to provide a scalable design with less resource, operating frequency, and collection latency overhead, but it may not provide the level of accuracy required for some applications. The distributed version sacrifices performance to gain monitoring accuracy and is best suited for small designs or when resources are not at a premium. A hybrid scheme may be produced where distributed probes periodically dump data from their individual buffers into a centralized location for DMA transfer out of the FPGA, but such a scheme may have the worst characteristics of both the central and distributed designs. Another option may be to pipeline the probe design, which may provide speed enhancements for high-performance designs.

Evaluating the *Integrated RC Performance Monitor* probes against the design criteria presented in Section 3 shows that the criteria can be met by performing design tradeoffs:

1) *Provide accurate information*. The accuracy of sampling, triggering, and timestamps can be improved by increasing the bit-widths of the sample, trigger and counter registers, respectively. Increasing these values tends to increase resource utilization, and care must be taken to determine the level of accuracy a design requires and provide that level and no more. The centralized approach may or may not provide accurate data depending upon the amount of data generated and the speed at which that data is generated. The distributed approach may provide better accuracy for some designs.

2) *Minimize delay and area overhead*. Limiting the number of probes and register bit-widths is important in keeping overhead to a minimum. Also, choosing the best mix of sample, count, and trigger widths as well as trigger set depth is an important tradeoff to minimize delay and reduce resource usage. The centralized monitoring scheme provides a compact design and should be used whenever possible to make best use of device resources.

3) *Keep generated data to a manageable size* and 4) *Deliver data in a timely fashion*. The inherent memory and I/O limitations of RC boards force the amount of data generated by monitor designs to be relatively small. The centralized approach provides a compact design with good DMA capability to minimize data transfer impact and thus provide data in a timely fashion to other CARMA agents.

6. Conclusions

Providing traditional monitoring techniques and other run-time management services on scalable RC systems is critical for their widespread adoption. The University of Florida's Comprehensive Approach to Reconfigurable Management Architecture (CARMA) seeks to provide a versatile execution and management environment for scalable, RC-augmented systems. This paper proposes

several hardware monitoring probe designs that fit within the CARMA framework, based in part on performance monitoring capabilities present in modern microprocessors.

Experimental results outline numerous tradeoffs that must be performed to optimize the integrated probe design for a particular application. Three generic probe designs are developed but not included in the final draft due to page limitations including: 1) one probe used to sample data based on an event without regard for when that event occurs (i.e. no timestamp); 2) one probe used to determine the amount of time that passes between two events (i.e. no sampling); and 3) one probe used to determine the time between two events that also samples at each occurrence.

In general, care should be given to ensure the width of the count register is large enough to cover the area of interest but no larger, since otherwise the design's maximum operating frequency may be adversely affected. Also, the width of the sample register affects area utilization, and care should also be given to limit samples to only the region of interest as appropriate. For probes with simultaneously active sample and count registers, as register bit-widths increase, maximum operating frequency reductions are dominated by the count register probe's operating frequency limitation and area is dominated by the sample register probe area overhead due to the PAR tool's inability to condense the design as much.

Another set of experiments is performed to determine how the trigger size and number of triggers a single probe monitors affect maximum operating frequency and area overhead. Increasing trigger width and depth tends to impose additional area and operating frequency overhead on the design, although this is more pronounced in larger-width triggers. Therefore, it may be more efficient to use several smaller probes than one large probe for many designs. Using the fewest trigger bits necessary is a good design practice while choosing the optimal trigger set depth for a given design must be explored.

Several experiments for monitoring organism states in a Game of Life application are performed to examine accuracy and overhead tradeoffs between the centralized and distributed monitoring schemes. The Game of Life results indicate that the centralized approach provides a scalable design with less resource, operating frequency, and collection delay overhead but it may not provide the level of accuracy required for some applications. The distributed version sacrifices performance for monitoring accuracy and should only be used when resources are not at a premium. A hybrid scheme may be produced where distributed probes periodically dump data from their individual buffers into a central location for DMA out of the FPGA, but further investigation is needed to ascertain if any benefit is achieved in this fashion. Another option may be to pipeline the probe design, which may provide speed enhancements for high-performance designs.

Directions for future work include exploring the additional degrees of freedom outlined throughout this paper including further development of a resident monitor and design of a hybrid or pipelined approach to overcome the limits encountered in the centralized and distributed integrated monitor approaches. Also, further case studies of monitoring with other scalable applications to determine the general applicability of our probe designs will be explored. Linking the monitor probes to other CARMA agents to provide enhanced scheduling and system repair methods is the subject of ongoing research.

7. Acknowledgments

This work was sponsored in part by the U.S. Department of Defense. We wish to thank the following vendors for key resources that made this research possible: Nallatech, Xilinx, Intel, and Cisco Systems.

8. References

[1] V. Ross, "Heterogeneous HPC Computing," *Proc. of 35th GOvernment Microcircuits Applications and Critical TECHnologies (GOMACTech)*, Tampa, FL, 2003.

[2] I. Troxel, A. Jacob, A. George, R. Subramaniyan, and M. Radlinski, "CARMA: A Comprehensive Management Framework for High-Performance Reconfigurable Computing," *Proc. of 7th International Conference on Military and Aerospace Programmable Logic Devices (MAPLD)*, Washington, DC, Sept. 8-10, 2004.

[3] K. Compton and S. Hauck, "Reconfigurable Computing: A Survey of Systems and Software," *ACM Computing Surveys*, Vol. 34, No.2, June 2002, pp. 171-210.

[4] K. Li and K. Zhang, "Supporting Scalable Performance Monitoring and Analysis of Parallel Programs," *The Journal of Supercomputing*, Vol. 13, No. 1, Jan./Feb. 1999, pp. 5-31.

[5] J. P. Calvez and O. Pasquier, "Performance Monitoring and Assessment of Embedded HW/SW Systems," *Proc. of 1995 International Conference on Computer Design (ICCD): VLSI in Computers and Processors*, Austin, TX, Oct. 2-4, 1995.

[6] R. Snodgrass, "A Relational Approach to Monitoring Complex Systems," *ACM Transactions on Computer Systems*, Vol. 6, No. 2, May 1988, pp 157-195.

[7] M. Martonosi, D. Ofelt, and M. Heinrich, "Integrating Performance Monitoring and Communication in Parallel Computers," *ACM SIGMETRICS Performance Evaluation Review*, Vol. 2, 1996, pp. 138-147.

[8] M. Harkema, D. Quartel, B.M.M. Gijsen, and R.D. Vander Mei, "Performance Monitoring of Java Applications," *Proc. of 3rd International Workshop on Software and Performance (WOSP)*, Rome, Italy, July 2002.

[9] D. Levi and S.A. Guccione, "BoardScope: a Debug Tool for Reconfigurable Systems," *Proc. of International Society for Optical Engineering: Reconfigurable Technology: FPGAs for Computing and Applications II*, Boston, MA, Oct. 2000.

[10] R. Subramaniyan, P. Raman, A. George, and M. Radlinski, "GEMS: Gossip-Enabled Monitoring Service for Scalable Heterogeneous Distributed Systems," *Cluster Computing* journal, accepted and in press.

[11] M. Gardner, "The Fantastic Combinations of John Conway's New Solitaire Game 'Life'," *Scientific American,* v. 223, Oct. 1970, pp. 120-123.