Adaptive Thermoregulation for Applications on Reconfigurable Devices, by
Phillip H. Jones, James Moscola, Young H. Cho, and John W. Lockwood,
17th IEEE International Conference on Field Programmable Logic and Applications (FPL),
Amsterdam, Netherlands, August 27-29, 2007.

ADAPTIVE THERMOREGULATION FOR APPLICATIONS ON RECONFIGURABLE DEVICES

Phillip H. Jones, James Moscola, Young H. Cho, John W. Lockwood

Applied Research Laboratory
Washington University
St. Louis, MO, USA
email: {phjones, jmm5, young, lockwood} arl.wustl.edu
http://www.arl.wustl.edu/arl/projects/fpx/reconfig.htm

ABSTRACT

A biological organism's ability to sense and adapt to its environment is essential to its survival. Likewise, environmentally aware computing systems avail themselves to a longer operational life and a wider range of applications than traditional systems. In this paper, we propose a novel circuit design methodology that allows parameterizable hardware to self-regulate its temperature. We apply this methodology to an image recognition system on an Xilinx Virtex 4 FX100 field programmable gate array (FPGA). The image recognition system sustains a safe operational temperature by automatically adjusting its frequency and output quality. The circuit sacrifices output performance and quality to lower its internal temperature as the ambient temperature increases, and can leverage cooler temperatures by increasing output performance and quality. Furthermore, the circuit will shutdown if the ambient temperature becomes too hot for the device to function properly. A performance evaluation of our adaptive circuit under various thermal conditions shows up to a 4x factor increase in performance and a 2x factor increase in quality over a system without dynamic thermal control.

1. INTRODUCTION

Developers of high performance computers constantly strive to build efficient systems that obtain the highest performance per area. As transistors shrink in size and are more densely packed, one of the biggest concerns for system developers today is system temperature management. As the ambient temperature rises, the heat generated by circuits can cause devices to become unstable or damaged due to internal temperatures rising above maximum operating thresholds.

A growing number of embedded computing systems are used outside of environmentally controlled locations. In locations such as remote parts of deserts, deep ocean floors, and outer space, it is not only difficult to predict environmental effects on a system, they also allow very limited access once a system is deployed. Therefore, it is often necessary for system parameters to be over provisioned to guarantee correct functionality under worst case environmental conditions. This often leads to an end system that is suboptimal for typical conditions.

Our work attempts to overcome the performance loss in such systems due to over-provisioning. In this paper, we present an adaptive mechanism that automatically adjusts system operating parameters to yield the best performance for the given environmental conditions. Real-time feedback from sensors is used to tune our circuits to run at near-optimal performance.

1.1. Contributions

Our earlier work [1] described how a design using a thermally adaptive frequency mechanism could operate faster than a fixed design by a factor of 2.4. In this paper, we successfully apply two new ideas to our thermally adaptive system. First, the adaptiveness of the circuit is extended to allow the system to make an application specific trade off between output quality and generated heat. We evaluate the performance to quantify the difference between our adaptive system and a static system under different environmental conditions. Second, we show how to overcome the problem of supply voltage variation associated with the use of ring-oscillators as an embedded temperature sensor. The key idea behind our solution is to time multiplex the system between running the application and making a temperature measurement. This allows the ring oscillator to sample temperature without interference from application dependent supply voltage variations.

1.2. Overview

In the following section, we discuss related research from the areas of power and temperature measurement and man-

agement. Section 3 then describes our adaptive self regulating architecture. First a generic architecture is described, followed by examples of applications that benefit from the use of adaption. Section 4 gives implementation details of our work. Section 5 presents a performance evaluation that compares our adaptive system to a static system under different thermal conditions. Section 6 concludes this paper.

2. RELATED WORK

Research has been conducted in several areas that are essential building blocks for our work. We divide these related works into three groups. First, we discuss work from the field of temperature measurement. Next, we discuss dynamic thermal management (DTM) techniques for embedded computing systems. Then we summarize our previous work, upon which this work builds.

2.1. Temperature Measurement Schemes

The ability to dynamically manage temperature of a device is only as effective as the speed and accuracy of its temperature measurement sensor. Lopez-Buedo [2] surveyed several techniques for measuring temperature. Thermal couples, thermal imaging cameras, embedded sense diodes, and ring oscillators were discussed.

On many of today's FPGAs, an embedded sense diode is built on to the die. Therefore, temperature can be determined by measuring current between the anode and cathode of this diode. Most of these sensors require an external analog to digital converter chip to make use of this data for reactive control. However, the latest family of FPGAs from Xilinx (Virtex 5 series) has a built-in temperature sensing diode that can be read directly by the FPGA logic [3].

Lopez-Buedo presented a novel temperature measurement technique without using special sensors. Ring oscillators were implemented using available reconfigurable logic, to infer device temperature from changes in signal oscillation frequencies. This work with ring oscillators is later extended in [4] using arrays of such oscillators to detect hot spots and thermal gradients in FPGAs.

2.2. Dynamic Thermal Management

Microprocessors have been built that allow their voltage and frequency to be scaled to extend the battery life of mobile computers. Companies that include Intel and AMD have extended this concept to manage the heat dissipated by servers [5]. By introducing power management features, software running on the CPU can scale voltage and frequency to lower power usage before the device overheats. This technology is critical for servers located in large data centers that house hundreds or thousands of computation nodes.

Low-power embedded processors like the Intel Xscale [6] have hooks that allow voltage and frequency scaling to manage power. Work presented by Wirth in [7] makes use of these features to present a DTM system that scales processor frequency in response to temperature readings from an external thermal couple.

Shang performed power measurement experiments on the Xilinx Virtex-II FPGA to determine the distribution of dynamic power [8]. For the applications analyzed, it was found that as much as 22% of dynamic power was consumed by clock resources. This result implies that managing clock resources could result in significant power savings. The Virtex-II, and later Xilinx FPGAs, have entities called BUFGMUXs [9] that can shut down part of a clock tree or switch the device to a low frequency during idle times [10]. Meng showed a 25% power savings through low-level simulation of a Wireless Channel Estimator application mapped to a Virtex-II, by disabling the clock for portions of the application not in use[11].

Chow [12] presented a dynamic voltage scaling mechanism that uses gate delay feedback to minimize the voltage supplied to internal FPGA logic, thereby reducing power consumption. The main idea of this work was to supply the minimum voltage to the FPGA that still allowed the critical path of an application to meet timing. Since the gate delays of a circuit are dependent on device temperature, a secondary benefit of this method was that voltage would scale with changes in temperature.

Peddersen [13] used estimations of power as feedback to adapt the compression ratio of an image processing application so that it would operate within a given power budget. Events such as cache misses were counted and used to estimate the power consumption of the application. It was shown in simulation that this method could successfully take advantage of the observation that the power needed to process images varied from image to image, even if they were of the same size and resolution. By adapting the compression ratio of the input image, the application could maximize image quality for a given power budget.

The DTM approach used in this case study builds upon work by Jones [1], in which a maximum operating temperature is associated with an application. A temperature feedback mechanism adaptively scales the clock frequency of the application. The goal was to maximum application performance under changing thermal conditions.

2.3. Our Previous Work

While profiling the thermal behavior of the FPGA on our reconfigurable platform, we discovered an opportunity to make use of the relatively fast measurements of junction temperature as compared to the rate of change in temperature. A relatively large amount of time is available to operate a circuit at a high frequency while the package slowly warms

as compared to the period at which the platform performs computation on data [14]. Seeing this as an opportunity to improve the performance of our reconfigurable hardware platform in transient conditions, we devised a novel scheme that dynamically adjusts the operation of the reconfigurable logic device between two clock frequencies using an upper and lower temperature threshold. This mechanism generates a thermally-adaptive frequency that maximizes the computational throughput for a specified maximum application temperature, which we refer to in this paper as the application's *thermal budget* [1].

The main idea of this approach is to modulate the duty cycle at which the application runs with a given base clock and a fast (e.g. 4x) clock. As the external thermal environment changes, the duty cycle will automatically adjust keeping the application temperature between the upper and lower bounds. By selecting thresholds appropriately and switching quickly between modes, the application can maintain a target average temperature within tight bounds. The objective is to achieve maximum computational performance for a given thermal budget by adaptively adjusting the duty cycle as the thermal operating environment changes.

We achieved up to a 2.4x factor improvement in throughput over using a thermally safe fixed frequency by applying this mechanism to an image processing application [1]. This work was further extended to be workload aware, and evaluated under bursty workload conditions. Results showed up to a 2x factor improvement in latency and up to a 32% saving in power under highly bursty conditions [15].

Our current work extends on our previous work by additionally enabling the image processing application to tradeoff quality of computation so as to gracefully degrade system performance as thermal conditions degrade. This mechanism allows the system to successfully operate over a wider range of thermal conditions, than our previous work. The current adaptive system is implemented on a single chip (Virtex-4 FX100).

3. ADAPTIVE CIRCUIT DESIGN

This section first presents a general model for dynamic adaption of application parameters using sensor feedback. This is followed by a short discussion of some application classes that benefit from the use of adaptation.

3.1. General Architecture

Adaptive systems can be described generically as systems that take feedback from their external and internal environment. Then based off of this feedback use a policy for adjusting their behavior through the use of actuators. Figure 1 illustrates this concept.

Starting from the top of this figure, feedback from the platform (e.g. platform temperature, power usage), and external environmental conditions (e.g. ambient temperature, atmospheric pressure) are measured by sensors. These measurements are then used as inputs into control logic that decide how to adjust system parameters in order to maintain/regulate system attributes within a specified range. Figure 1 separates system parameters into two sub groups; (1) Platform (i.e. application independent) parameters such as platform input frequency and supply voltages, and (2) application specific parameters, such as mode of operation, and number of concurrent execution engines. These system parameters can be thought of as actuators for controlling attributes of the system (e.g. power consumption, temperature). Changes in system attributes can then be detected by sensors, thereby closing the control loop.

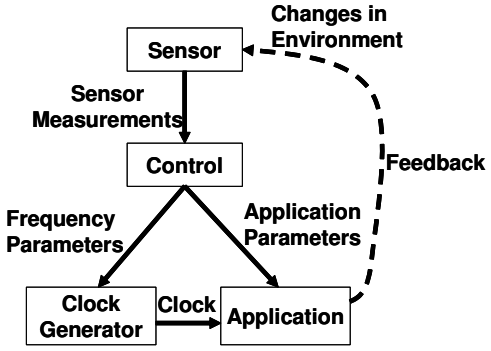

Fig. 1. Regulating operational conditions by adjusting application parameters and clock frequency

3.2. Environment Sensor Feedback

Many applications can and have benefited from the use of feedback to adapt to changing conditions. Three classes of such applications are autonomous vehicles, sensor networks and DTM.

Autonomous vehicles make extensive use of feedback for navigation and interacting with the external world. One well-known example is the Mars rovers. These rovers can autonomously execute complex commands sent from Earth. Traveling from point A to point B navigating through obstacles is one example. During navigation, video input is used to help detect and avoid hazards. The rover also has temperature sensors to help maintain a temperature of -40 to 40 C for electronic devices [16].

There are a number of sensor network applications that use feedback to adjust the behavior of a distributed system. For example Wang [17] explored the use of sensor networks to help stabilize buildings during earthquakes. A network of motion sensors were distributed throughout a structure. In the event of an emulated earthquake these sensors detected the motion of the structure, and sent this information to a controller that computed how to move actuators in order to

help reduce the effect of this motion.

DTM uses feedback from internal and external conditions to regulate temperature and/or power consumption of a system. Section 2.2 gives several examples of DTM.

4. IMPLEMENTATION

This section first introduces the development platform used in this work. Next, details on the implementation of temperature measurement are provided. This is followed by a description of an image processing application used for the case study evaluation. This section concludes with a discussion of the adaption policy used by the system.

4.1. Reconfigurable Development Platform

In this work we used a Virtex-4 100FX based development board available from High Tech Global [18]. The thermally self-regulating system is completely contained within the Virtex-4. Figure 2 shows the three main components of the system. The Thermal Manager is responsible for measuring the FPGA temperature. The Frequency and Quality Controller implements the policy for adapting the application. Finally the application implemented for this work is an image recognition application.

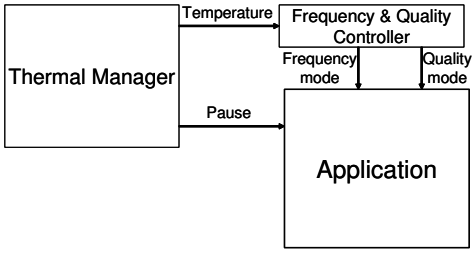

Fig. 2. System Architecture

4.2. Temperature Measurement

A ring oscillator thermometer was implemented to measure the internal temperature of the FPGA. As noted in [2], ring oscillators are sensitive to changes in supply voltage. If the supply voltage drifts, so does the period of the ring oscillator. In this work it was observed that when our application made a sudden change in its modes of operation, the supply voltage varied. Though these variations were relatively small (.001 V) as compared to the core voltage (1.2 V), this caused the ring oscillator thermometer to give measurements that were unusable. Figure 3 illustrates this behavior. In this example the FPGA is at 60 C. Depending if the application is in mode A, B, or C the thermometer will give a wide range of count values. This made tracking the FPGA temperature infeasible.

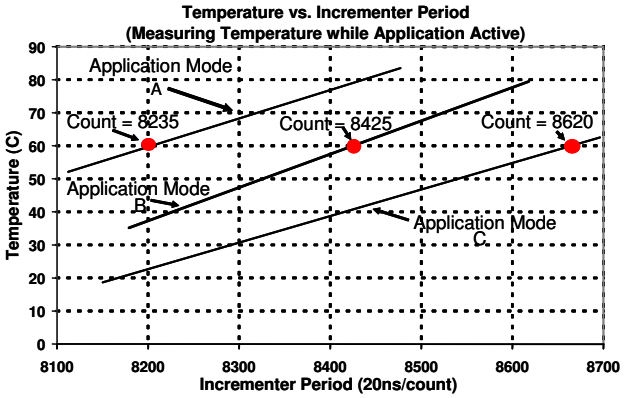

Fig. 3. Unstable readings while application active

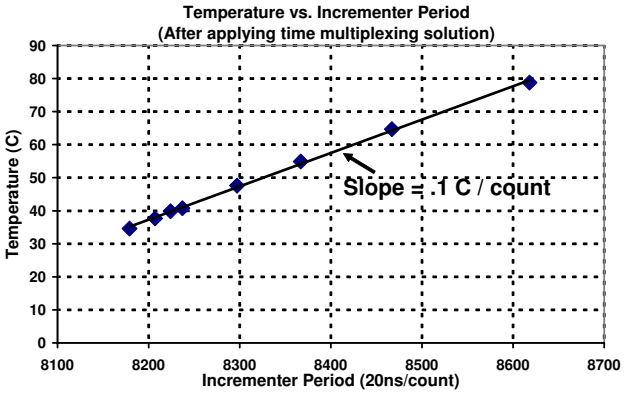

Fig. 4. Sample mode stabilizes temperature readings

This problem was solved by using the observation that changes in power consumption modes occur instantaneously relative to the changes in the FPGA temperature (e.g. in microseconds vs. milliseconds). Based on this observation, we used a special temperature sample mode to periodically measure the FPGA temperature. This allowed the ring oscillator sensor output to be read while in a electrically consistent state (i.e. constant supply voltage). During the temperature sample mode, the application is effectively paused. Figure 4 shows stable temperature measurements after implementing the temperature sample mode. Even though the application is cycling through different modes, having a temperature sample mode removes the unstable thermometer behavior.

The impact of the sample mode on application performance is dependent on the ratio between the rate at which temperature measurements are made, and the amount of time needed to make a measurement. For our implementation, it was found that .5 ms was needed to make a reliable measurement, and the period between measurements was 50 ms. This ratio results in a 1% decrease in application performance.

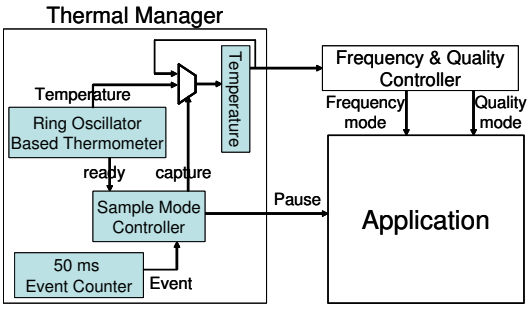

Fig. 5. Thermal Manager Architecture

Figure 5 shows a high level view of the logic used to implement the periodic sample circuit. The Event Counter signals an event every 50 ms. On detection of an event the Sample Mode Controller issues a signal to pause the application, thereby placing the FPGA in a consistent electrical state. The application remains paused until 3 measurements have been made by the thermometer ($<$.5ms), at which point the measured temperature is deemed stable. This measurement is then passed to the Frequency and Quality Controller and the application continues.

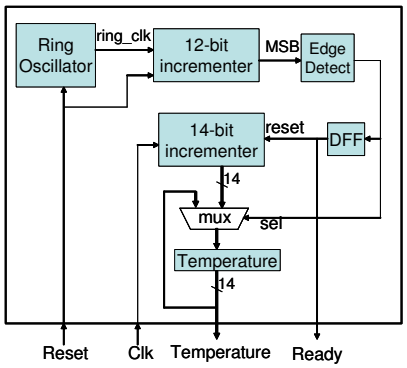

Fig. 6. Ring Oscillator Thermometer Architecture

Figure 6 shows the structural layout of the ring oscillator thermometer. The thermally dependent period of the ring oscillator is used as the clock input to a 12-bit incrementer. As the period of the ring oscillator changes, this is reflected in the amount of time it takes the 12-bit incrementer to count to its maximum value. An incrementer with a fixed frequency is used to precisely count the thermally dependent period of the 12-bit incrementer. This is done by storing, then reseting the count value of the fixed frequency incrementer each time the most significant bit (MSB) of the thermally dependent incrementer transition from 1 to 0, using a negative edge detector. The resource utilization and other characteristics of the ring oscillator based thermometer are shown in figure 7.

Inverters are used to construct the ring oscillator portion of the thermometer. These inverters were manually placed

Thermometer size	Ring oscillator size	Oscillation period	Incrementer Cycle Period	Temperature resolution
~100 LUTs	48 LUTs (47 NOT + 1 OR)	~40 ns	~.16 ms (40ns * 4096)	.1ºC/Δcount Or .1ºC/Δ20ns

Fig. 7. Ring Oscillator Based Thermometer Characteristics

using relative location attributes (RLOCs). These manual constraints are used to help maintain a consistent oscillation period between different instantiations of the application.

4.3. Image Recognition Application

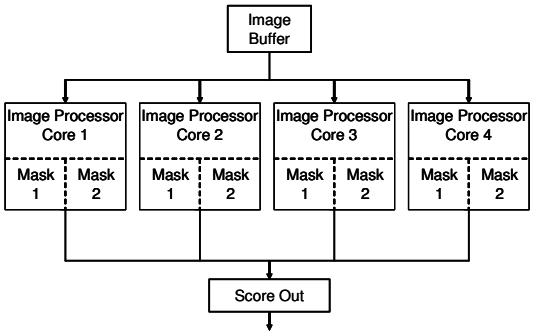

Fig. 8. High-level Application Architecture

Image recognition is an application that is well-suited for hardware implementation [19]. It is a highly parallel application that allows multiple image processing cores to run simultaneously using the same image data. We utilize an image recognition application with multiple processing engines to evaluate our thermally adaptive techniques.

Our image recognition application uses four parallel feature-based image processors. Each processor scans for up to 2 features, thereby allowing the system to scan an image for up to 8 features at a time. A block diagram of the basic image recognition application is shown in figure 8, and the resource utilization of this application is given in Figure 9.

Each image processing core implements an image processing algorithm. A 64x64 bit-mask pattern (each mask corresponds to a feature) is scanned over incoming images. A score is computed for each possible offset of the pattern. This score is the sum of the product of each bit of the pattern with a corresponding pixel value. Figure 10 illustrates this algorithm. Template T scans image I from left to right and from top to bottom.

4.4. Adaption Policy

Under optimal thermal conditions, the image recognition application processes images at the maximum frequency using both feature patterns for all four image processing cores. As

Virtex-4 100FX Resource Utilization

Lookup Tables (LUTs)	D Flip Flops (DFFs)	Occupied Slices	Block RAM	Max Frequency
57,461 (68%)	49,148 (58%)	32,868 (77%)	44 (11%)	200 MHz

a.)

Image Correlation Characteristics

Image Size (# pixels)	Pixel Resolution	# of Features	Image Processing Rate (Frames per second)
320x480	8-bit (grey scale)	1 - 8	40.6 (at 200 MHz)

b.)

Fig. 9. a.) FPGA Utilization, b.) Application Details

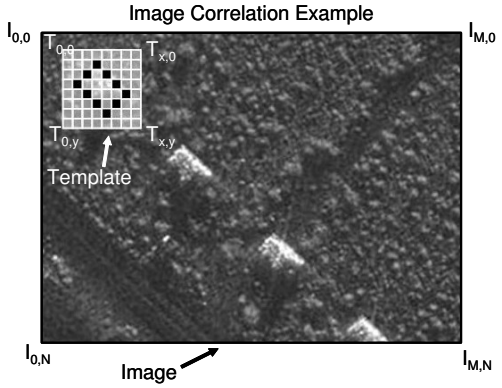

Fig. 10. Image Correlation Algorithm Example

the FPGA heats up and thermal conditions become more adverse, the application parameters change to prevent the FPGA from overheating.

The image recognition application parameters are modified in two ways. The first method decreases the temperature of the FPGA by decreasing the clock frequency. Decreasing the clock frequency decreases the amount of work performed by the chip, thus lowering its temperature. Since some image recognition applications may require that some minimum number of frames be processed per second, we assume that there is a minimum frequency that the circuit must operate above. If the circuit is already operating at its minimum frequency and the thermal conditions continue to degrade, other parameters of the application are modified.

In feature-based image recognition applications, some features may have a higher priority than others. If this is the case, then features of the image recognition application can be run in order of their priority and we selectively disable processing units to decrease the amount of work performed by the FPGA and hence lower its temperature. Figure 11 illustrates the priority of features that need to be processed. Lower priority features are disabled first, followed by higher priority features as thermal conditions degrade. In figure 11, both mask 1 and mask 2 of image processing core 4 are disable, effectively disabling this image processing core. Mask 2 of image processing core 3 is also disabled.

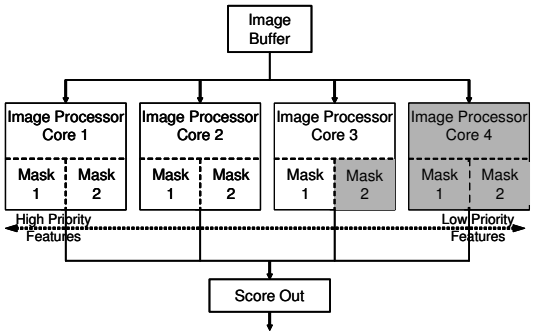

Fig. 11. Application Adaption Example

It is well known that the temperature of a device is proportional to the power it consumes, and power can be related to the clock frequency, activity rate, switching capacitance, and supply voltage of a device by the relation:
$Power \sim (activity\ rate) * CV^2F$
This relation presents four fundamental parameters of the FPGA that can be adjusted in order to regulate its temperature. Our adaption policy uses three of these four parameters; clock frequency, activity rate, and switching capacitance.

First, the frequency is adjusted by our adaption policy. Second, switching capacitance is controlled by the activation and deactivation of image processing cores, which modifies the active circuit size. Finally a time sharing approach was used to implement multiple masks per processing core. The removal of a single mask from a processing core reduces the activity rate of the corresponding core.

5. EXPERIMENTATION

This section first describes the experimental setup used to evaluate the adaptive image processing application. This is followed by an analysis of the experimental results.

5.1. Test Setup

The image recognition application was deployed on the Virtex-4 FX100 FPGA of our development platform. This platform was installed into a 3U rackmount case, shown in figure 12. The case is equipped with 2 fans that each supply approximately 250 Linear Feet per Minute (LFM) of air flow. During the execution of an experiment, the top of the case was closed and a thermally isolating cover was placed over the case to isolate the thermal conditions inside the case from the external laboratory conditions. An external heat source was used to raise ambient temperatures inside the case greater than that of the laboratory. A temperature probe was installed to monitor the temperature inside

the case. This enabled the us to verify that the temperature inside the case was correctly maintained during each experiment.

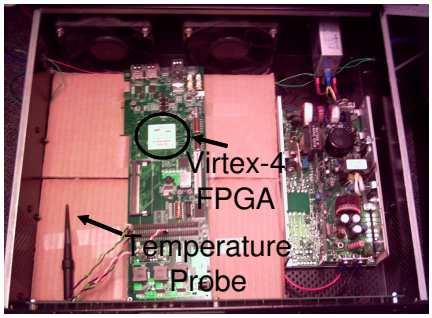

Fig. 12. Experiment Setup

Experiments were conducted to compare the performance of the adaptive image processing application to a static version of the same application. These experiments were executed under six different thermal conditions, listed in figure 13. The clock frequency and number of processing cores used for the static application were determined by finding the frequency and number of cores, under worst case thermal conditions (scenario S1), for a thermal budget of 65 C.

For this circuit the frequency was found to be 50 MHz, and the number of cores was found to be two. This will be referred to as the thermally safe static configuration for the image processing application. The adaptive version of the application uses a thermally controlled frequency that can switch between 50 MHz and 200 MHz. The adaptive application can also use up to four processing cores.

Scenario S1 – S6

	Ambient Temperature	# of Fans
S1	40 C (104 F)	0
S2	35 C (95 F)	0
S3	30 C (86 F)	0
S4	25 C (77 F)	0
S5	25 C (77 F)	1
S6	25 C (77 F)	2

Fig. 13. Thermal conditions used for evaluation

5.2. Results and Analysis

Figure 14 gives a summary of the results obtained from experiments for scenarios S1-S6. The adaptive version of the application reduces its frequency and number of cores in order to operate safely under worst case thermal conditions (S1). As thermal conditions improve, the adaptive application takes advantage of these conditions by first increasing the number of cores to improve the quality of the application, then by increasing the effective frequency. Under best case thermal conditions (S6), the adaptive application can operate at 200 MHz which is 4 times the frequency of the static application and it scans twice the number of features.

Effective Frequency (MHz): Scenarios S1-S6

	S1	S2	S3	S4	S5	S6
Adaptive 50-200 MHz	50.0	50.0	64.8	106.4	183.7	200.0
Fixed 50 MHz	50.0	50.0	50.0	50.0	50.0	50.0

Quality Level (# of features scanned): Scenarios S1-S6

	S1	S2	S3	S4	S5	S6
Adaptive 50-200 MHz	4	6	8	8	8	8
Fixed 50 MHz	4	4	4	4	4	4

Fig. 14. a.) Effective Frequency, b.) Quality Level

We achieve a significant improvement over our previous work that only adapted the frequency. Using our previous approach, the application would have still operated a 200 MHz under best case conditions. However in order to safely operate under worst case conditions, our previous approach would have been limited to using 2 cores, since the number of cores was a fixed parameter. By enabling the quality of the application to adapt, our new circuit allows a factor of 2x increase in quality over our previous approach.

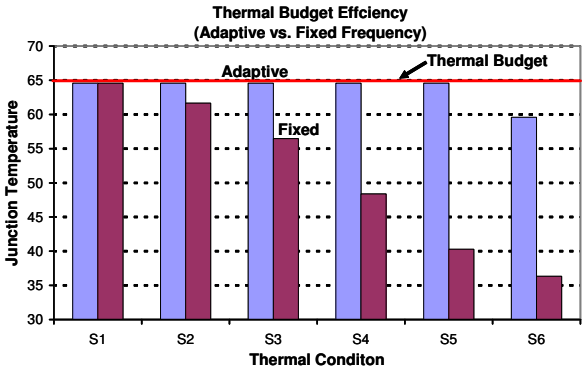

Fig. 15. Thermal budget efficiency

Figure 15 and 16 shows further analysis of the experimental data. In figure 15, it can be seen that as thermal conditions improve, the adaptive version of the application increases performance to make best use the thermal budget. The gap in efficiency between the adaptive and fixed version of the application increases as conditions improve. The only thermal condition for which the adaptive version of the application can not make full use of the thermal budget is thermal condition (S6). For this case, the application performance is limited by its maximum operating frequency of 200 MHz.

For conditions S1-S6, figure 16 shows the change in

junction temperature as each of the four processing cores are activated. The change in temperature per core is about 2 C for S1-S4. While for conditions S5 and S6, the change in temperature is about 1.4 C and 1.2 C respectively. The significance of this is that the increase in thermal energy per core is fairly independent of changes in ambient temperature when there is no airflow (S1-S4), but changes quickly for a constant ambient temperature with changing airflow (S4-S6). This behavior may be due to the fact that when there is no airflow, the FPGA can heat the air in its local area, thereby decreasing the dependency of thermal energy per core on ambient temperature. When there is airflow, the air heated by the FPGA is replaced by ambient air, thereby allowing the ambient temperature to have a greater impact on thermal energy per core.

It is also worth noting that the change in temperature per core being in the range of 1 - 2C is less than expected. This is most likely due to the fact that a global clock was used to drive all four cores. Therefore when a core is deactivated, the clock driving that core is still dissipating heat. The change in temperature per core would increase if each core was driven by a separate clock that could be disabled.

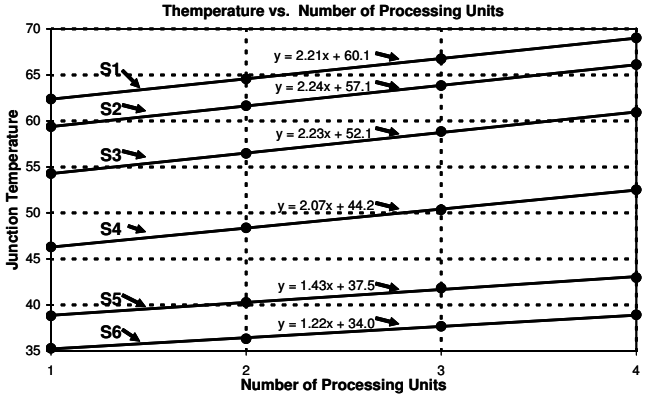

Fig. 16. Change in temperature per processing unit

6. CONCLUSION

This paper explored ways to build a single-chip system that obtains near-optimal performance while deployed in environments that are unpredictable. We applied a number of techniques to adapt an image recognition application implemented on an FPGA. This allowed the application to adaptively react to changing environmental conditions to obtain the highest possible performance while maintaining a safe temperature.

The thermally adaptive circuit outperforms a static version of the application by a factor of 4x in throughput, and by a factor of 2x for image processing quality (i.e. number of features that can be processed per image). Compared to our previous approach, which only used a thermally adaptive frequency, application-specific adaption allowed a more efficient use of the thermal budget.

Through this work we also developed a method that ensures a consistent supply voltage to our ring-oscillator during temperature measurements. This was accomplished by time multiplexing application execution and temperature measurements into different time slices. This eliminated the voltage variations caused by the application changing modes during temperature readings, thereby providing consistent readings.

7. REFERENCES

[1] P. H. Jones, Y. H. Cho, and J. W. Lockwood, "An adaptive frequency control method using thermal feedback for reconfigurable hardware applications," in *IEEE International Conference on Field Programmable Technology (FPT)*, Bangkok, Thailand, Dec. 2006.

[2] S. Lopez-Buedo, J. Garrido, and E. Boemo, "Thermal testing on reconfigurable computers," *IEEE Design and Test of Computers*, vol. 17, pp. 84–91, 2000.

[3] *Virtex-5 System Monitor User Guide*, Xilinx, 2006.

[4] S. Lopez-Buedo and E. I. Boemo, "Making visible the thermal behaviour of embedded microprocessors on FPGAs: a progress report." in *FPGA*, 2004, pp. 79–86.

[5] Intel Corporation, "Addressing power and thermal challenges in the datacenter," 2005.

[6] *Intel 80200 Processor based on Intel XScale Microarchitecture Developer's Manual*, 2003.

[7] E. Wirth, "Thermal management in embedded systems," Master's thesis, University of Virginia, 2004.

[8] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in virtex-ii fpga family," in *FPGA '02: Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*. New York, NY, USA: ACM Press, 2002, pp. 157–164.

[9] *Virtex-II Platform FPGA User Guide*, Xilinx, 2005.

[10] S. Choi, R. Scrofano, V. K. Prasanna, and J.-W. Jang, "Energy-efficient signal processing using fpgas," in *FPGA '03: Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays*. New York, NY, USA: ACM Press, 2003, pp. 225–234.

[11] Y. Meng, W. Gong, R. Kastner, and T. Sherwood, "Algorithm/architecture co-exploration for designing energy efficient wireless channel estimator," *Journal of Low Power Electronics*, vol. 1, pp. 238–248, 2005.

[12] C. T. Chow, L. S. M. Tsui, P. H. W. Leong, W. Luk, and S. J. E. Wilton, "Dynamic voltage scaling for commercial fpgas," in *ICFPT*, 2005, pp. 173–180.

[13] J. Peddersen and S. Parameswaran, "Energy driven application self-adaptation at runtime," in *20th IEEE/ACM International Conference on VLSI Design*, Bangalore, India, Jan. 2007.

[14] P. H. Jones, J. W. Lockwood, and Y. H. Cho, "A thermal management and profiling method for reconfigurable hardware applications," in *16th International Conference on Field Programmable Logic and Applications (FPL)*, Madrid, Spain, Aug. 2006.

[15] P. H. Jones, Y. H. Cho, and J. W. Lockwood, "Dynamically optimizing FPGA applications by monitoring temperature and workloads," in *20th IEEE/ACM International Conference on VLSI Design*, Bangalore, India, Jan. 2007.

[16] A. H. Mishkin, J. C. Morrison, T. T. Nguyen, H. W. Stone, B. K. Cooper, and B. H. Wilcox, "Experiences with operations and autonomy of the mars pathfinder microrover," in *IEEE Aerospace Conference*, 1998.

[17] Y. Wang, R. A. Swartz, J. P. Lynch, K. H. Law, K.-C. Lu, and C.-H. Loh, "Decentralized civil structural control using a real-time wireless sensing and control system," in *4th World Conference on Structural Control and Monitoring (4WCSCM)*, San Diego, CA, July 2006.

[18] "Hitech Global Webpage," http://www.hitechglobal.com/, 2007.

[19] Y. H. Cho, "Optimized automatic target recognition algorithm on scalable myrinet-field programmable array nodes," in *34th IEEE Asilomar Conference on Signals, Systems, and Computers*, Monterey, CA, Oct. 2000.