Minimizing Thermal Variation in Heterogeneous HPC Systems with FPGA Nodes

Yingyi Luo, Xiaoyang Wang, Seda Ogrenici-Memik, Gokhan Memik
Department of Electrical Engineering and Computer Science
Northwestern University
{yingyi, xiaoyangwang2018}@u.northwestern.edu,
{seda, memik}@eecs.northwestern.edu

Kazutomo Yoshii, Pete Beckman
Mathematics and Computer Science Division
Argonne National Laboratory
kazutomo@mcs.anl.gov,
beckman@anl.gov

Abstract—The presence of FPGAs in data centers has been growing due to their superior performance as accelerators. Thermal management, particularly battling the cooling cost in these high performance systems, is a primary concern. Introduction of new heterogeneous components only adds further complexities to thermal modeling and management. The thermal behavior of multi-FPGA systems deployed within large compute clusters is little explored. In this paper, we first show that the thermal behaviors of different FPGAs of the same generation can vary due to their physical locations in a rack and process variation, even though they are running the same tasks. We present a machine learning based model to capture the thermal behavior of a multi-node FPGA cluster. We then propose to mitigate thermal variation and hotspots across the cluster by proactive task placement guided by our thermal model. Our experiments show that through proper placement of tasks on the multi-FPGA system, we can reduce the peak temperature by up to 11.50°C with no impact on performance.

I. INTRODUCTION

Across many industries, a deluge of data is being generated and processed each day. This has resulted in a dramatic growth in the supercomputer and data center industry. However, these systems are power hungry and generate excessive amounts of heat. It is estimated that over 2% of the overall energy consumption in the US is allocated to data centers. Tianhe-2, one of the top super computing systems, consumes up to 17 MW with an additional 7 MW for cooling. Other studies [1]–[3] show that the cooling power takes up to 30% to 50% of the total power usage, which creates a significant waste of electricity.

One of the trends in the design of data centers and high-performance computing (HPC) systems is the increasing diversity of computational elements. While earlier systems primarily consisted of CPUs, newer systems increasingly make use of GPUs and FPGAs. Inclusion of FPGAs within HPC systems is motivated by their superior performance and power efficiency for certain workloads. However, thermal management within heterogeneous systems requires further attention, in order to fully harness the anticipated improvements while guaranteeing reliable systems with systematic control of thermal stress.

Previous studies [4]–[6] found that uneven temperature distribution in multi-CPU systems can raise system peak temperature. While there have been a number of studies focusing on modeling and managing temperature of CPUs, FPGA temperature dynamics within HPC systems are less explored. Considering the upcoming CPU-FPGA hybrid platforms, such as Intel’s Xeon-FPGA platform, the temperature management of FPGAs becomes even more challenging and needs to be addressed directly.

In this paper, we show that thermal events vary significantly between nodes hosting identical FPGAs. On the one hand, higher temperature leads to higher power consumption, especially impacting the cooling power. Reliability is similarly temperature-dependent. Controlling peak temperature in FPGA devices requires additional methods, because unlike microprocessors, FPGA chips are not equipped with dynamic thermal management features. This makes them particularly susceptible to thermal stress. Previous studies on the effects of thermal stress on FPGA device failure mechanisms have found that chip temperatures in the range of 64°C-75°C can cause more than 18% of wires to fail due to electromigration in less than 5 years [7]. The Mean Time to Failure (MTTF) relationship has an exponential dependency on temperature. Our analysis in a cluster of FPGAs revealed that operating temperatures exceeding 80°C for the majority of the execution time of an application can be observed in HPC workloads. Negative Bias Temperature Instability (NBTI) is another failure mechanism for modern FPGA chips operating in the temperature range of 70°C and above. NBTI mainly affects the threshold voltage levels of PMOS transistors and consequently many structures within the FPGA, such as the configuration SRAMs, routing multiplexors, and buffers. Finally, we note that managing peak temperatures in FPGA devices will become essential in future CPU-FPGA integrated chips, where thermal coupling can cause either chip to become a thermal aggressor towards the other depending on the workload distribution. In this paper, we first present a methodology for performing temperature characterization of FPGAs through empirical data analysis and machine learning techniques. No domain-specific knowledge (e.g., model of the FPGA chip, geometry, material properties) pertaining to thermal modeling is needed by our methodology. The peak temperature of the system is predicted purely based on the FPGA resource usage. This results in a lightweight system that has a small cost as we will discuss later. We then demonstrate the effectiveness of the model for making a static decision on task placement in our target system, which has four
Nallatech 385A FPGA accelerator cards. For each deployment request for a batch of tasks, the placement decision is made statically, prior to execution. A new placement decision will be made for each subsequent incoming batch of tasks managed by the central resource manager of the global cluster. This helps us create a lightweight tool to guide the global resource management decision with zero impact on performance and yet still allows to reap significant benefits in reducing the peak temperature with minimal intervention to the system management software infrastructure.

Our model receives as input a batch of tasks and a group of nodes that are allocated for this batch’s deployment by the global resource manager. Then, the corresponding peak temperatures for all possible task placements are predicted within this group of nodes. The placement with the lowest predicted peak temperature is chosen for deployment of this batch. Compared to a thermal oblivious placement method, our thermal-aware task placement method is capable of reducing the peak temperature by up to 11.50°C and the peak power by 3.45 W. Drawing from our observations on user statistics of a 650-node cloud computing cluster, we also propose a partitioning strategy ensuring scalability within well-defined partitions. Finally, we note that although we mainly leverage existing machine learning tools to create the prediction models, their application to focus on FPGA temperature modeling, the observation of large temperature variation between nodes in a real system, and the demonstration of mitigating this variation with a real system implementation are the key novel contributions of our work.

The rest of this paper is organized as follows. Section II gives an overview of related work. In Section III, we present our motivational experiments. We present our thermal characterization methodology and the machine learning models in Section IV. Section V presents the experimental results. We discuss the scalability of the framework in Section VI. The main findings are summarized in Section VII.

II. RELATED WORK

Zhang et al. [4], [8] proposed a framework to characterize thermal behavior of HPC tasks running on multi-CPU systems and reduce the overheating on the systems without CPU performance degradation. The framework utilized a variety of performance counters, like instruction count, branch miss rate, etc., to build the thermal model. However, due to the lack of hardware performance counters on FPGAs, the framework cannot be utilized for FPGA systems. Topcuoglu et al. [9] proposed two novel scheduling algorithms for a bounded number of heterogeneous processors to increase the performance and reduce the algorithm complexity. Their study focused on the system performance and did not take thermal behaviors into consideration. Jing et al. [10] proposed an energy-efficient scheduling algorithm based on ant colony optimization for multi-FPGA systems with a task placement scheme to reduce the energy consumption. The main idea is to scale down task frequency while preserving time constraints. Energy is reduced at the expense of performance. Tang et al. [11] proposed an approach for thermal-aware task scheduling to reduce the cooling cost in data centers by minimizing peak inlet temperature. Their approach requires knowledge of the heat recirculation model, which in turn, requires detailed information of the data center layout and relevant thermodynamics. Furthermore, it mainly focused on controlling the cold air supplied by computer room air conditioners not the compute nodes of the system.

There are runtime studies for efficient scheduling of parallel tasks [12]–[14] on HPC systems to exploit heterogeneous multi-nodes architectures. However, none of them took thermal behaviors into consideration.

In this paper, our main contributions are:

1) We collect data in a representative system. By analyzing the data, we show the presence of temperature variation and optimization opportunities for FPGA nodes in HPC systems;
2) To the best of our knowledge, this is the first attempt to present temperature models for FPGA systems using machine learning techniques.
3) We identify three relevant machine learning models to do predictions and analyze trade-offs between those models in terms of prediction accuracy and overheads;
4) We validate our prediction methodology’s effectiveness via a thermal-aware task placement study, which can reduce system peak temperature.

III. MOTIVATION

Our work is motivated by our temperature analysis of a real multi-FPGA system. We performed our experiments in the Chamaleon cloud computing cluster [15]. The cluster has 650 multi-core nodes. Among those 650 nodes, four of them are equipped with FPGA accelerators. Each accelerator is a Nallatech 385A board with an Altera Arria 10 1150 GX FPGA and 8 GB DDR3 on-card memory. They are all 2U nodes located next to each other in the top of a 48U rack. We refer to these four nodes as U41-42, U43-44, U45-46, and U47-48, respectively, based on the actual unit number they are located in. For the temperature measurement, Arria 10 provides an internal temperature sensing diode with a built-in analog-to-digital converter circuitry that can read out the temperature. For measuring the FPGA board power consumption, Nallatech OpenCL board support package has memory-mapped device library functions that can be called to monitor the board power consumption.

We used HPC benchmarks written in OpenCL that are designed and provided by Intel [16]. These benchmarks capture different patterns of parallel computation and communication algorithms. The mapping of these benchmarks to Berkeley’s Dwarf’s kernel classification [17] is listed in Table I.

Let us first take a close look at the temperature behavior for each application. Figure 1 shows the temperature traces of the tasks while running on node U43-44. The sampling frequency is 1Hz. We can see that each task requires different amount of time to reach steady state temperature and all of them reach peak temperatures after 10 minutes.
Figure 2 shows the peak temperature variations among the four nodes running the same benchmark. We can observe that the temperature variations are significant for all the ten benchmarks. The maximum difference between the hottest and coolest node is 11.50°C (asian_option benchmark) and the average difference is 9.20°C. We also observe up to 5.94 W (3.31 W on average) peak power difference. In addition, there is a consistent thermal characteristic among the four nodes. Node U43-44 is always the hottest and U45-46 is always the coolest. In a HPC system, certain components can display distinct thermal behavior due to physical/geometric attributes like heat conduction between nodes, airflow of cooling systems, and physical characteristics. Our results show that these differences can be considerable.

With the acquired data, we try task placements on this multi-FPGA system for all possible four-task combinations using four distinct tasks (no task is repeated in a given combination) and calculate the difference in peak temperatures between different placements for each task combination. Figure 3 shows the minimum peak temperature, average peak temperature, and maximum peak temperature of the system for each task combination. On average, the difference between the minimum peak temperature and average peak temperature is 4.55°C and the difference between minimum peak temperature and maximum peak temperature is 10.02°C. This indicates that if these tasks are scheduled with an optimal thermal-aware scheduler, we can expect a 4.55°C temperature reduction on average compared to thermal-oblivious scheduler and a 10.02°C temperature reduction on average compared to the worst case.

Next, we analyze the performance of each benchmark on different nodes. Performance is defined as the reciprocal of the execution time. We observe that the performance variations are no more than 0.1% on different nodes. There is no thermal throttling mechanism on the FPGA boards and the frequency of each design is pre-defined at OpenCL kernel compilation time, hence the performance variation is minimal. With the temperature and power data we collected at run time, we extracted both the peak temperature and peak power for each benchmark. Figure 4 shows the correlation between the two. A positive linear correlation is observed. This indicates that the peak power consumption can be reduced by reducing the node peak temperature.

These observations, as well as the trend of increasing diversity of computational elements in future HPC systems, motivated us to design tools for making thermal-aware decisions to minimize thermal variation and mitigate the hotspots in this system. Specifically, we develop a new methodology to characterize the thermal behavior of HPC systems with FPGA nodes. We then leverage this model to perform task placement to reduce the peak temperature of the system.

### IV. Temperature Prediction Model

In this section, we describe our temperature prediction model. The model is targeted at predicting the steady-state thermal peak of a given set of tasks on a multi-FPGA system. We expect a task to reside on its assigned node for a suffi-
Fig. 3. System peak temperature across different combinations

Fig. 4. Relationship between peak temperature and peak power

ciently long duration of time, which is typical of long-running computation scenarios in HPC systems.

The effectiveness of a prediction model depends on its ability to capture the thermal characteristics of the system accurately. On the other hand, it is desirable to have a general model that is easily adapted to different architectures. Note that we make no assumptions of any detailed knowledge about the underlying physical system. For instance, the framework is unaware of the specific locations of the nodes (e.g., which machine is located far from the cool air inlet) or the geometry of the system. Also, the framework has no knowledge of the thermal transfer properties of the materials involved. It operates as a mapping function between features that are readily accessible to an operating system and the expected steady peak temperature at a node. This approach makes machine learning approach a natural candidate for our purposes.

Our methodology is comprised of the following five steps: (1) We select a series of benchmarks and gather benchmark-related properties, like logic utilization, on each FPGA device in the system. (2) For that specific system, we run the selected benchmarks on each FPGA node, and collect real time temperature and power data. (3) Using the cumulative data collected from steps (1) and (2), we develop a system-specific thermal model with machine learning methods that can predict the system’s peak temperature for a given task assignment on the FPGA nodes. (4) When a task set needs to be scheduled, we compare different possible task assignment scenarios with the generated model. The system software is configured to suggest an assignment that is expected to result in the lowest peak temperature for the system. As we will discuss in the experimental results section, other objective functions can be easily incorporated into the framework. (5) Task assignment is performed according to the suggestion made at step (4).

A. Feature Selection

To generate a good machine learning model, selecting representative features is an essential step. In our framework, the raw features we considered are static information about the resource usage on the FPGA. Specifically, they are logic utilization, DSP blocks, RAM blocks, memory bits, I/O pins, and frequency. These statistics can be easily gathered from Intel FPGA SDK for OpenCL tool at kernel compilation stage.

After analyzing the correlation between these features and the steady-state temperature (depicted in Figure 5), we find that logic utilization is the most correlated feature. Frequency has a negative correlation, which might be counter intuitive at first. However, further analysis shows that tasks with higher frequency have lower logic utilization. The compiler (synthesis tool) always aims for the highest possible frequency. In a smaller design with less logic complexity, the wire length tends to be shorter, which helps meet a higher frequency goal, but the logic utilization tends to be lower due to the low logic complexity. To find the best set of features to use for the machine learning model, we start by using logic utilization alone in the prediction model. We then add other features one by one until we encounter a degradation in the cross-validation accuracy. Ultimately, logic utilization, RAM blocks, and clock frequency features have been determined to be the most effective ones for building the model.

B. Machine Learning Models

With the selected features, our framework utilizes a set of representative benchmarks to characterize the system and obtain a system-specific model. Each sample in the training set is composed of the above task features for each FPGA node in the system and the system peak temperature. System peak temperature is defined as the highest temperature among all the nodes. We evaluated a large number of machine learning methods via Scikit-learn [18]. From these, we selected three
representative models and performed an in-depth analysis on each. They are linear regression (LR), multilayer perceptron (MLP), and random forest (RF) models.

LR is a linear approach for modeling the relationship between dependent variables and independent variables. It is easy to build. As for the prediction overhead, it is lightweight. In addition, the trained model can reflect system characteristics. However, the model might not be able to make accurate predictions when the system environment becomes more complex.

MLP is highly suitable for modeling non-linear data sets. However, it requires more effort and expertise on tuning model parameters like the number of layers and learning rate. Also, the training overhead is much higher than other models. When the dataset is small, it may result in overfitting.

RF works by constructing a multitude of random trees and outputs the mean value of the individual trees. It is resistant to overfitting [19], [20]. The downside is that it can only predict values within the range of observations for each target. In our case, if a new task that is never encountered before causes the system to dissipate significantly more heat than ever measured before, the model will fail to make an accurate prediction.

V. EXPERIMENTS

In this section, we present our experimental evaluation. The benchmarks we used are listed in Table I. Each task is run for 10 minutes. We have confirmed that all tasks perform the major portion of their main body of computations within this duration. We have also verified that, 10 minutes duration is sufficient for all the tasks to run through their setup phase and reach their steady-state behavior and temperature. Table II shows the resource usage for each benchmark (the I/O Pin usage is the same for all the benchmarks: 41%).

In our experimental system with 4 FPGA nodes, each batch that needs to be assigned on FPGAs is composed of 4 tasks. With 10 benchmarks, we can have \( \binom{10}{4} \) different task combinations. There are 4! possible assignments of one task combination. Since each sample in the data set corresponds to one specific assignment, there are 5040 samples in total that we can use to build the prediction model.

A. Tuning Model Parameters

For each model we discussed in Section IV.B, we first tune the parameters for each machine learning model. After data normalization, we apply 10-fold cross validation to help the selection of model parameters, e.g., the number of layers in the MLP model, the number of trees and corresponding tree depth in the RF model. The accuracy of the model is evaluated by the root mean square error metric. Since LR is simple, it does not have additional parameters to be tuned. The details for the other two models are discussed below:

1. Multilayer Perceptron: it consists of seven layers, one input layer, five hidden layers and one output layer. The number of neurons in each hidden layer is set to 9. We use a constant learning rate of 0.1 and a batch size of 100. Other parameters are kept as default values.

2. Random Forest: 120 random trees with maximum depth of 6 are used in our model. Other parameters are set to default. It should be noted that the above parameters are tuned particularly for our target multi-FPGA system. If users want to apply our method on other systems with different FPGA devices, they may need to tune the model parameters.

B. Accuracy of Prediction Models

We first evaluate the task-dependent models, which means that all of the ten benchmarks may appear in some task combinations in both training and testing sets. We randomly select 80% of the data as training set to generate the models. The remaining 20% of the data is used for testing. Predicted temperature values are then compared against the actual measurements collected from runtime. This process is repeated 10 times to avoid data selection bias. The resulting root mean square errors are 3.11°C, 2.40°C and 1.43°C, respectively. RF performs the best, while LR delivers relatively worse accuracy.
We then test the task-independent models. Six tasks are selected randomly to form the training set and training samples are created in the same way described earlier. The remaining four tasks are reserved for testing. The prediction results of the three machine learning methods for a specific task combination (mandelbrot, matrixMult, sobel_filter, video_downscaling) are shown in Figure 6. Note that in this experiment, the training set and testing set do not have any common tasks, hence, the predictor has no knowledge of the tasks it is predicting for. After completing the experiment for all 210 different task combinations, the root mean square errors of LR, MLP, and RF are 7.31°C, 12.87°C, and 7.66°C, respectively. In this case, although the accuracy is considerably lower, the errors of all the predictors seem to exhibit a constant bias. Since our method uses the predictions only to compare different placements with each other, a bias does not interfere with achieving consistently well-guided decisions. In fact, for the task combination depicted in Figure 6, all prediction models can pick one of the lowest temperature placements (for example, the models would pick placements indexed 6, 6, and 1, respectively).

C. Thermal-Aware Task Placement

Previous studies [21]–[23] showed that circuit reliability, especially the reliability of interconnects, depends exponentially upon the operating temperature. Hotspots and thermal variations lead to over 50% of electronic failure. This is captured by the electromigration failure mechanism. Reliability studies specifically focusing on FPGA devices have also identified electromigration and NBTI failures as primary thermal-induced failure mechanisms [7]. As the number of FPGA nodes increases in large scale clusters (e.g., Microsoft’s large scale deployment of FPGA accelerators [24]), failure rates become increasingly relevant.

We set the thermal-oblivious placement as the baseline for evaluation. We make two comparisons: average behavior and worst case behavior. The average peak temperature of the baseline system itself helps to understand certain trends. In the absence of significant performance difference between the options, over a long time period, any placement can be considered to have equal probability to be selected by the thermal-oblivious method. Hence, we determine the average peak temperature of the 24 placements of 4 tasks to represent the average behavior of the baseline. On the other hand, the worst case choice of the thermal-oblivious placement illustrates the maximum thermal stress a system might endure in comparison to our proposed task placement. This can have significant consequences, even if the thermal-oblivious placement picks those particular placements occasionally. Transient hotspots and temporary thermal stress cause permanent-irreversible damage, which will accumulate over the lifetime of the system.

Average peak temperatures resulting from a placement guided by our task-dependent models are shown in Figure 7. Among three models, we observe that the MLP model performs the best. However, the other models also behave reasonably good. Specifically, the LR, MLP, and RF reduce the average peak temperature by 4.12°C, 4.21°C, and 4.07°C, respectively. Considering that each model has a different overhead of prediction, designers can choose the one that fits their requirements the best. We then analyzed the generated LR model, and found that the coefficients for the task’s logic utilization, RAM blocks and frequency running on node U43-44 are the largest among all the other features. This indicates that those three features are the most important ones to predict the peak temperature, which makes sense since it is usually the hottest node that contributes to the system peak temperature.

Similarly, the performance of the task-independent models are shown in Figure 8. The benefits of each model are 3.83°C, 2.22°C, and 3.34°C, respectively, compared to the average behavior of thermal-oblivious placement. It turns out that LR performs the best on the task-independent model. RF performs comparably good. We believe the reason that MLP performs the worst is that it is likely overfitting to the training data.

In comparison to the worst case behavior of the baseline solution, our method is able to reduce the peak system temperature by up to 11.50°C. Taking all the possible task combinations into consideration, the average peak temperature difference compared to the worst case is 9.31°C, 8.20°C, and 9.07°C for LR, MLP, and RF respectively.

The results show that our methods can robustly pick the best placement. We find the peak temperature is largely determined by one machine and one task. So among the 24 placements of 4 tasks in one combination, there are usually only about 4 different peak temperatures. The predictor may at times not place a task with low peak temperature in the optimal location, however, it still captures the opportunity to identify the best placement as it is able to predict the hottest task(s) correctly.
D. Impact on Power Consumption

The main goal of our task placement method is to reduce peak temperature in a system by suppressing variation across nodes. This, in turn, improves the lifetime and maintenance costs (e.g., cooling power) of the system. Fortunately, another side effect of our method is a reduction in the peak power consumption without any degradation in performance. Surges in peak power can lead to problems, such as the branch circuit overload and overheating, and it is the limiting factor for many systems that operate under stringent power caps [25].

The impact of our mechanism on power consumption is shown in Figure 9. This figure lists all 24 possible placements of a particular task combination (asian_option, compute_score, fd3d, video_downscaling). The origin of this plot is (average peak temperature, average of peak power) and the placement decision identified by our method is located at (4.60, 1.77) in that coordinate system. This demonstrates that reduction of peak temperature can lead to peak power reduction. In this specific task combination, we achieve a 1.77 W (3.5%) peak power reduction compared to the average, which reaches 73.91°C and consumes 50.28 W power.

We then evaluate the average and worst case behavior for all task combinations. The average peak power reductions achieved by LR, MLP, and RF are 1.48 W, 1.50 W and 1.49 W, respectively. For the worst case behavior, the peak power reductions are 3.45 W (2.40%), 3.44 W (2.29%), 3.40 W (2.40%), respectively for a system reaching 80°C peak temperature.

We note that, while the node power reduction, particularly for the average case, is modest, it is achieved at zero cost, without any impact on performance.

E. Overhead of the Machine Learning Methods

Since we target at making placement decision statically, we can train the models and do predictions offline. For 80% of the total samples (4056 samples), the LR, MLP, and RF require 0.09s, 0.55s, 0.47s to train the models. The prediction overheads are 0.07ms, 0.08ms and 0.67ms, respectively. As for making one placement decision for a task combination, in our 4-nodes system, we need to do predictions on 24 possible assignments. We see the prediction overheads of LR and MLP are comparable to each other, while RF takes longer time.

In terms of the models’ effectiveness to reduce system peak temperature, if users know in advance the tasks that they want to schedule, MLP may be the right choice. In contrast, if there are many unknown new tasks that may be deployed in the system, LR is a better option. RF is more complex computationally, but it is good at performing non-linear regression. If we could have some evidence that the new unknown tasks do not raise the system temperature beyond the ones already in the training data, then RF can be adapted to get better prediction accuracy.

VI. Scalability and Other Considerations

Our analysis on the Chameleon cloud computing platform, which has 650 multi-core nodes, showed that the average number of active users in a day is 7. This indicates that even if there may be thousands of applications running on the cluster over a long period, there will be only a small number of applications in a batch that needs to be scheduled at a given instance. Furthermore, each application usually requires more than one computing node to run. Hence, we propose to perform predictions within the boundaries of partitions when applying our framework to do task placement on a large scale system.

Suppose there are \( N \) nodes in the platform, we can partition them into \( S \) sets so that each set consists of \( N/S \) nodes. For each set, we then pick the hottest node (using a set of experiments that is similar to the one described in Section III) and let it represent the set. Since we have \( S \) sets, the prediction model that needs to be trained will only contain \( S \) nodes’ features, instead of \( N \) nodes’ features. In addition, the prediction model can be trained offline, which further reduces the overhead. When a new batch arrives, we do predictions on the pre-defined partitions. Instead of placing the applications over individual nodes, we can have the placement strategy over partitions to reduce the average peak temperature.

As long as we keep \( S \) small, our model is still lightweight. If a task does not need all the nodes in one partition, we can recursively do finer grain partitioning of nodes within that cluster and apply our framework to the partitions.

In terms of the applicability of our framework on other HPC systems, as long as the FPGAs support OpenCL, we can collect the different resource usage features mentioned in Section IV.A to train a machine learning model that can characterize the thermal behavior of the system. Hence, we argue that the proposed method is general.
Unfortunately, currently our test platform contains only 4 FPGA nodes. Hence, we cannot experiment with the above ideas at present time. There are, however, plans to significantly expand the number and type of FPGA nodes in the Chameleon testbed system. Hence, we leave these two aspects for future study.

VII. CONCLUSION

In this paper, we performed a thermal analysis on a FPGA HPC system. The results show that the temperature and power of the same benchmarks vary significantly across different machines due to their different physical characteristics: we have observed up to 11.50°C peak temperature and 5.94 W peak power difference across the machines in our testbed. Three different machine learning techniques have been lever-aged to characterize the thermal behaviors. We evaluated task-dependent and task-independent models. With a task-dependent model, we can reduce the peak temperature by 4.21°C and peak power by 1.50 W on average and by 11.50°C and 3.45W compared to the worst case placement. The task-independent model is less accurate in terms of absolute temperature prediction, but the error exhibits a constant bias. This leads to correct placement decisions.

The performance of the system is not affected through our task placement. In future FPGA systems with sophisticated CPU-like thermal throttling mechanisms, the reduction in peak temperature can also lead to improved performance.

Our next step would be to study how to apply our framework at run-time. Compared to the static task placement, other interesting directions are: (a) if a new task arrives at the system while there are already other tasks running for a while, should we take into consideration those scheduled tasks to do task migration accordingly and (b) whether we can utilize the new task’s data to calibrate the machine learning model online. We are also interested in investigating other platforms, like tightly-coupled heterogeneous systems and system-on-chips.

ACKNOWLEDGMENT

Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation.

REFERENCES


