An FPGA-based Approach to Evaluate Thermal and Resource Management Strategies of Many-core Processors

MARCEL METTLER, Chair of EDA, Technical University of Munich, Germany

MARTIN RAPP and HEBA KHDR, Chair for ES, Karlsruhe Institute of Technology, Germany

DANIEL MUELLER-GRITSCHNEDER, Chair of EDA, Technical University of Munich, Germany

JÖRG HENKEL, Chair for ES, Karlsruhe Institute of Technology, Germany

ULF SCHLICHTMANN, Chair of EDA, Technical University of Munich, Germany

31

The continuous technology scaling of integrated circuits results in increasingly higher power densities and operating temperatures. Hence, modern many-core processors require sophisticated thermal and resource management strategies to mitigate these undesirable side effects. A simulation-based evaluation of these strategies is limited by the accuracy of the underlying processor model and the simulation speed. Therefore, we present, for the first time, an field-programmable gate array (FPGA)-based evaluation approach to test and compare thermal and resource management strategies using the combination of benchmark generation, FPGA-based application-specific integrated circuit (ASIC) emulation, and run-time monitoring. The proposed benchmark generation method enables an evaluation of run-time management strategies for applications with various run-time characteristics. Furthermore, the ASIC emulation platform features a novel distributed temperature emulator design, whose overhead scales linearly with the number of integrated cores, and a novel dynamic voltage frequency scaling emulator design, which precisely models the timing and energy overhead of voltage and frequency transitions. In our evaluations, we demonstrate the proposed approach for a tiled many-core processor with 80 cores on four Virtex-7 FPGAs. Additionally, we present the suitability of the platform to evaluate state-of-the-art run-time management techniques with a case study.

CCS Concepts: • **Hardware** → Chip-level power issues; **Temperature simulation and estimation**; **On-chip resource management**;

Additional Key Words and Phrases: ASIC emulation, many-core processor, thermal management

ACM Reference format:

Marcel Mettler, Martin Rapp, Heba Khdr, Daniel Mueller-Gritschneider, Jörg Henkel, and Ulf Schlichtmann. 2022. An FPGA-based Approach to Evaluate Thermal and Resource Management Strategies of Many-core Processors. *ACM Trans. Arch. Code Optim.* 19, 3, Article 31 (May 2022), 24 pages.

<https://doi.org/10.1145/3516825>

New article, not an extension of a conference paper.

Partly funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), Project No. 146371743, TRR 89 Invasive Computing.

Authors' addresses: M. Mettler, D. Mueller-Gritschneider, and U. Schlichtmann, Chair of EDA, Technical University of Munich, Arcisstrasse 21, Munich, Germany, 80333; emails: {marcel.mettler, daniel.mueller, ulf.schlichtmann}@tum.de; M. Rapp, H. Khdr, and J. Henkel, Chair for ES, Karlsruhe Institute of Technology, Haid-und-Neu-Strasse 7, Karlsruhe, Germany, 76131; emails: {martin.rapp, heba.khdr, henkel}@kit.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.

1544-3566/2022/05-ART31 \$15.00

<https://doi.org/10.1145/3516825>

1 INTRODUCTION

Embedded systems show an ever-increasing demand for computational power in application fields such as robotics, telecommunication and autonomous driving. To meet this rising demand, many-core processors have been proposed, which consist of many simple cores instead of few high performance cores [10]. Enabled by the continuous technology scaling, this architecture comes with challenges in high power densities and high operating temperatures that affect the performance and the reliability of the processors [22].

To minimize these undesirable side effects, thermal and resource management strategies are needed. Besides **dynamic voltage frequency scaling (DVFS)**, thermal-aware mapping and scheduling strategies have been proposed to improve the reliability of the processor [34, 42]. For instance, it is possible to change the mapping of compute intensive applications at run-time to prevent thermal hot spots on the processor [15]. As these techniques become increasingly sophisticated, they need to be integrated and tested from early design stages on. At this point in time, the real platform is not available. Thus, management strategies must either be evaluated by pure simulation-based approaches or by **field-programmable gate array (FPGA)**-based approaches. While full-system simulators, such as gem5 [6], are orders of magnitudes slower than the real platform and fast simulators, such as Sniper [8], introduce inaccuracies by multiple levels of abstraction, FPGA prototypes are a common approach to drastically reduce the simulation time and to provide a platform for early software development. As FPGAs use lookup tables to implement combinational logic while **application-specific integrated circuits (ASICs)** use gates, their power and temperature consumption differ highly from each others. Thus, thermal sensors on the FPGA [26–28, 31–33, 47] can not be used to emulate the ASIC behavior. Instead ASIC power and temperature emulation [2–4, 9, 16, 17, 25, 39] is required. However, the existing works either completely neglect DVFS or scale the power consumption independently from the processor speed. This is a major limitation, since DVFS, the application mapping and scheduling are important control knobs for a resource manager. Furthermore, the centralized design of the existing transient temperature emulators [2] does not scale for many-core processors. Additionally, the evaluation of resource and thermal management strategies requires a non-intrusive monitoring infrastructure to analyse the performance impacts of each management decision and a large set of benchmark applications, on which the management strategies can be tested.

This article proposes the first FPGA-based evaluation approach for resource and thermal management strategies to address the above mentioned challenges in the design of many-core processors. The approach comprises an actor-based benchmark generator, an ASIC emulation platform and a monitoring infrastructure. With the automatic benchmark generator, it is possible to replicate applications using the actor-based programming model. This model is especially popular for many-core processors [20], as these do not necessarily implement cache coherency [43]. Hence, this method enables an application-independent evaluation of run-time management strategies. The data for the evaluation is generated based on the ASIC emulation platform consisting of a per core power, temperature and DVFS emulator. Thereby, we present the first distributed thermal emulator design of transient temperatures, including also the first emulation approach that scales linearly with the number of integrated cores. Furthermore, the emulation platform consists of the first DVFS emulator design that considers the overhead of switching frequencies for DVFS in terms of energy and delay. To enable a precise performance analysis, a non-intrusive monitoring infrastructure is used to trace the system status with the power and the temperature of the processor. Thus, the approach is able to compare different run-time management strategies in terms of power, temperature, execution time, and more advanced metrics such as maximal violation time of a thermal threshold.

The key contributions of this work can be summarized as follows:

- The combination of benchmark generation, ASIC emulation and run-time monitoring to enable an FPGA-based evaluation approach for resource and thermal management strategies of many-core processors.
- An actor-based benchmark generation method for many-core processors to enable an application-independent evaluation of run-time management strategies.
- A new distributed ASIC temperature emulator design that scales linearly in size with the number of integrated cores.
- A configurable DVFS emulator design that precisely models the timing and power overhead of voltage and frequency transitions.

We demonstrate the capabilities of the proposed approach for the evaluation of a state-of-the-art thermal management strategy on a tiled 80-core processor, which is implemented on a proFPGA platform [13] consisting of four Virtex-7 FPGAs. Thereby, we show how the evaluation approach helps to identify the strengths and weaknesses of the management strategy. The remainder of this article is structured as follows: First, we present the related work in Section 2. Subsequently, Section 3 introduces the evaluation approach. The implementation of the benchmark generation flow is discussed in Section 4, followed by the temperature emulator design in Section 5 and the DVFS emulator design in Section 6. Experimental results are discussed in Section 7, and Section 8 concludes the article.

2 RELATED WORK

The proposed evaluation approach for resource and thermal management strategies combines three major components: A benchmark generator, an ASIC emulation platform and a run-time monitor. Each component for itself has been studied in different research communities. Therefore, we first present the related work on evaluation approaches of run-time management strategies, followed by the related work on the novel components of our approach: the benchmark generation, the design of the temperature emulator and the design of the DVFS emulator.

2.1 Evaluation Approaches for Thermal and Resource Management Strategies

The dominating evaluation approaches for thermal and resource management strategies are based on multi-core simulators. The simulators run benchmark applications on a configurable processor model to estimate its run-time behavior, including the power consumption and the execution time. Based on the power consumption, thermal simulators, such as HotSpot [48] and MatEx [35], estimate the processor temperature. Popular examples for multi-core simulators are among others: Sniper [8], gem5 [6], and ZSim [45]. Although Sniper does not perform cycle-accurate simulations of the processor pipeline and uses an abstracted memory model, it shows together with ZSim the highest accuracy in a survey of multi-core simulators [1]. Furthermore, Sniper can be combined with the power simulator McPat [24] and the thermal simulator HotSpot [48] to HotSniper [38], making it a popular approach to evaluate thermal and resource management strategies. However, the configuration options of the target platform and the number of supported target **instruction set architectures (ISA)** are limited. Here, the cycle-accurate gem5 simulator provides more flexibility with the additional support to run a complete **operating system (OS)**. Nevertheless, the configurability of gem5 is also limited, which makes it difficult to accurately model custom processor architectures. Furthermore, all simulators show a significant variation in their accuracy over different benchmark applications [1]. This limitation might lead to wrong conclusions in the design of a run-time management strategy. Furthermore, the simulation speed is a major limitation of simulation-based approaches. Even though Sniper does not perform cycle-accurate simulations,

it only provides a simulation speed of ~ 2 **million instructions per second (MIPS)** [19]. Cycle-accurate simulators, such as Gem5, are even slower with a speed of ~ 0.3 MIPS [18]. Thus, it is impractical, and for some simulators even impossible, to boot the operating system on many-core processors. Hence, the run-time management strategy, which is typically integrated in the operating system, cannot be sufficiently tested.

FPGA-based ASIC emulations are an additional approach to evaluate run-time management strategies. Here, ASIC power and temperature emulators are integrated on an FPGA prototype. In Reference [9], Cochet et al. present an FPGA-based power emulator for a LEON3 processor, which is based on the correlation between the power consumption and the statistics unit of the processor. Similarly, Bhattacharjee et al. [4] evaluate the power consumption based on activity counters. Glocker et al. [16, 17] present eTPMon, an ASIC power and steady-state temperature emulator for FPGA prototypes. Listl et al. [25] extend this approach by a DVFS interface for the power monitor and an emulation of chip aging. Similarly, Alam and Garcia-Ortiz [2] introduce an emulator for the transient temperature of a multi-core processor. A common shortcoming in the works on FPGA-based ASIC emulation are the missing capabilities to non-intrusively monitor the system behavior of the processors. Hence, performance metrics, such as the maximal core temperature and the execution time of the application, are not available to analyze the performance of run-time management strategies. While simulation-based approaches provide sufficient monitoring capabilities, either their simulation speed or their simulation accuracy is limited by the level of abstraction of the underlying processors model. Furthermore, evaluations on standard industrial benchmarks might not reflect the run-time characteristics of the target application. Therefore, we propose to combine approaches from the area of benchmark generation, ASIC-emulation and run-time monitoring to an evaluation approach that combines the tracing advantages of simulation-based approaches with the accuracy and the simulation speed of FPGA-based approaches.

2.2 Benchmark Generation

Benchmark applications have been designed to evaluate the performance of processors. Here, it is important that the applications show an execution behavior comparable to the actual workload of the system to obtain representative performance metrics. In the past, two major benchmark types have been established in the community: Standard industrial benchmarks and synthetic benchmarks. Standard industrial benchmarks are selected applications that are reasonable in size and in complexity. Popular benchmark suites of this category are the CoreMark [11], SPEC [12], Splash-3 [44], and PARSEC [5] benchmark suites. All provide a set of benchmarks that should ideally cover the execution behavior of most applications. While standard industrial benchmarks are the closest to real workloads, they are fixed in their structure and are not easily adjustable to the permanently changing architecture of computing systems and new computing paradigms.

Here, synthetic benchmarks show their advantages. They are fully generated out of basic operations and therefore scale easily with the compute power of today's and future systems. Dujmović describes in Reference [14] two generation approaches for synthetic benchmarks. The first approach uses a **recursive expansion (REX)** process to generate an application out of basic operations according to a parameterizable depth (i.e., level of nesting) and a parameterizable breadth (i.e., the number of operations per code block). Using this technique, it is possible to rapidly generate applications of arbitrary size. The second approach uses a kernel insertion generator to synthesize an application according to a parameterizable run-time behavior. For this purpose, the insertion generator selects kernels out of a pre-characterized kernel library such that the combination of kernels matches the requested execution behavior.

The benchmark generation approach presented in this article extends the REX process presented by Dujmović [14] to generate benchmarks for many-core processors by embedding the generated tasks into configurable actor graphs. Thus, we are able to scale the size of the application with the number of cores, integrated in the processors, to evaluate resource and thermal management strategies on large-scale many-core processors.

2.3 ASIC Temperature Modelling and Emulation

The thermal model of an integrated circuit is commonly expressed by an electrical equivalent RC-circuit, which is implemented in most popular thermal simulators, such as HotSpot [48] and MatEx [35]. In References [16, 17] Glocker et al. use this model to generate a **lookup table (LUT)** that stores the steady-state core temperatures for different power consumption values of a multi-core processor. To consider the neighboring effect between adjacent cores, Listl et al. [25] use a linear regression model to calculate the steady-state core temperatures at run-time. While this approach gives a good estimate for the worst-case temperatures of the processor, an evaluation of resource and thermal management strategies is only possible with an emulation of the transient core temperatures. Otherwise, the time to heat a core up and cool it down is neglected, which changes the run-time behavior of the **dynamic thermal manager (DTM)** and thereby, the execution behavior of the target application. Hence, Alam and Garcia-Ortiz [2] compare two different methods to calculate the full thermal model at run-time, which yields an accurate emulation of the transient temperatures. However, a major drawback of this approach is the limited scalability. The number of required multiplications per iteration and the number of model parameters increases quadratically with the number of cores implemented on the processor. Furthermore, the power consumption of each core, which is commonly evaluated in a distributed fashion, must be sent to the central temperature monitor and the emulated temperatures must be sent back. This communication over the entire chip introduces a significant implementation overhead. In this article, we present a distributed temperature emulator design. Thus, the transient temperatures can be emulated locally and the emulator scales linearly in terms of multiplications and model parameters with the number of integrated cores.

2.4 DVFS Modelling and Emulation

ASIC implementations do not only differentiate from FPGA prototypes in power and temperature but also in DVFS overheads, which are oscillator- and voltage-regulator-dependent. In Reference [7], Burd and Brodersen introduce a model for the energy and delay transition overhead between different V/f levels, which assumes a voltage-controlled oscillator for the generation of the clock. This model has later been updated by Park et al. [37] for **Phase-locked Loops (PLL)**, which are still the standard in today's processors systems.

An FPGA prototyping platform by Mantovani et al. [29] evaluates the effects of DVFS with loosely coupled accelerators on an FPGA prototype. Here, the DVFS capabilities of the FPGA platform are used. This approach limits the evaluation of run-time management strategies by the fixed DVFS characteristics of the FPGA platform that are not adjustable to the characteristics of the target ASIC platform. In this article, we present a configurable DVFS emulation that considers energy and delay overheads during the transition between different V/f levels. Thus, the DVFS characteristics of the ASIC platform can accurately be emulated on the FPGA prototype.

3 SYSTEM OVERVIEW

In this section, we give an overview of the proposed evaluation approach and the considered system architecture before we discuss the implementation details of the novel benchmark generation

Fig. 1. FPGA-based evaluation framework for thermal and resource management strategies.

method and the temperature and DVFS emulator designs in the following sections. The evaluation approach, illustrated in Figure 1, combines methods from the area of benchmark generation, FPGA-based ASIC emulation and run-time monitoring. The benchmark generator provides an arbitrary number of application actor graphs with various run-time characteristics, on which the resource or thermal management strategy can be tested. This feature enables the identification of the strengths and weaknesses of a management strategy in different run-time scenarios already at the design time of an ASIC.

The run-time behavior of these applications is influenced by the resource manager and the DTM. In the considered system architecture, the resource manager is a software component that allocates pending jobs from the application to cores and requests an associated frequency to improve the performance of the system. In contrast, the DTM is a hardware component that protects the system from overheating. Hence, it either throttles a core when its temperature reaches a configured threshold or it approves the requested V/f level of the resource manager. When designing these management components, it is important to maximize performance under the consideration of thermal safety. While measures to increase the safety margin and the processor performance are sometimes contradicting, the presented emulation approach helps to identify a good trade-off.

As ASIC power, temperature and DVFS characteristics either explicitly or implicitly influence the decisions of the resource manager, and thereby, influence the overall execution, they must be emulated on FPGA prototypes. Hence, the DVFS emulation manipulates the processor speed such that its run-time behavior adjusts to the emulated frequency and to the transition overheads between V/f levels. As a result, a change of the V/f level does not only manifest itself in the emulated power and temperature but also in the execution time. The power consumption of the processor is emulated by the system status and the emulated V/f level. Based on the power consumption, the temperature is emulated per core. This feature is not only helpful for the evaluation of run-time management strategies but also a required input parameter for most of them.

Finally, the evaluation framework consists of a non-intrusive run-time monitoring architecture. Thus, the system status, and the evaluated power and temperature can be processed to gain deep insights into the system behavior. Based on power, temperature, execution time and more advanced metrics, the performance of different run-time management strategies can be evaluated and compared to each other.

4 BENCHMARK GENERATION

The performance impact of a certain resource and thermal management strategy is application-dependent. Therefore, it is necessary to evaluate strategies on a large set of benchmark applications to verify their universal applicability. While industrial benchmarks are only available to a limited extent or are confidential, synthetic benchmarks can be rapidly generated to evaluate the performance of a management strategy. Dujmović [14] presents such a benchmark generator based on a REX process that generates a procedural application with a predefined depth D (i.e., the maximal level of nesting) and a predefined breadth B (i.e., the maximal number of statements per code block). In this work, we extend the generation process by two additional parameters, which allow us to accurately model run-time characteristics. The first parameter is the floating point instruction probability P_{fp} , which defines the probability that a variable assignment uses floating-point arithmetic instead of fixed-point arithmetic. Thus, the benchmarks can account for the typically higher power consumption and execution time of floating point instructions compared to integer instructions. The second parameter N_M defines the size of the memory on which the benchmark operates, which allows the benchmark generator to account for various cache hit and miss rates. Furthermore, we extend the application code by an input/output interface for message passing. This actor-based programming model [20] is especially suited for the high number of cores integrated in a many-core processor and overcomes the scalability limitations of shared memory models [23]. This trend is also reflected by Intel's **Single-chip Cloud Computer (SCC)** [43], where each tile is equipped with a special message passing interface.

The pseudo code of the extended REX procedure is illustrated in Algorithm 1. First, the REX process reserves the memory on which the benchmark operates. In this process, the task loads the input data with the size of N_I from its predecessors and copies it into the defined memory range. If a task has no predecessors, then the memory remains uninitialized. Subsequently, the actual task code is generated. For this purpose, the REX process calls the *get_statement* procedure, which randomly selects one out of N different procedural control structures. Thereby, the conditional statement is generated by a procedure *get_condition*, such that it evaluates to true after calling it n times. In this context, n is randomly generated to support varying levels of code locality. The code block of the control structure in turn is generated out of B statements. Thus, the REX process recursively calls the *get_statement* procedure until either the level of nesting is equal to D or until a *selector* equal to zero is chosen. In this case, the expansion process is terminated by an arithmetic statement. During the post-processing, the output data with the size of N_O is assigned to a subset of the local memory of the benchmark. Furthermore, the output data is transferred, if necessary, to the target location of the successor tasks.

Using the presented REX process, it is possible to rapidly generate tasks with various run-time behaviors. In this process, we describe the behavior of each task t by a performance vector $C_t = (c_{IC}, c_{DC}, c_{int}, c_{fp})$, where c_{IC} corresponds to the hold rate of the instruction cache, c_{DC} to the hold rate of the data cache, c_{int} to the integer instruction rate and c_{fp} to the floating-point instruction rate. Thus, it is possible to identify the best suited tasks for arbitrary test scenarios of run-time management strategies. To simplify the match between the execution time of the benchmark and the test scenario, a single benchmark can be executed multiple times. With this technique, also the variability of the execution time can be modeled. Furthermore, it is possible to split large test scenarios with phases of different run-time behaviors up and match each phase with a benchmark individually. Thus, the benchmark generator can either be used to model a specific application, which has not been ported to the target architecture or whose source code is confidential, or to generate a large amount of random test scenarios for run-time management techniques.

ALGORITHM 1: Recursive Expansion Process

```

string rex(D, B, Pfp, NM, NI, NO)
1  code = get_preprocessing(NM, NI)           // Initialize and Load Input Data
2  code += get_statement(D, B, Pfp, NM, 1+rand()%(N-1)) // Generate Application Code
3  code += get_postprocessing(NM, NO)       // Store Output Data
4  return code

string get_statement(D, B, Pfp, NM, selector)
5  D = D - 1
6  if D == 0 then
7    selector = 0
8  switch selector do
9    case 0 return get_assignment(Pfp, NM)           // Terminate Expansion
10   case 1 return "if" + get_condition() + get_block(D, B, Pfp, NM)
11   ...
12   case N - 1 return "do" + get_block(D, B, Pfp, NM) + "while" + get_condition()

string get_block(D, B, Pfp, NM, selector)
13  block = ""
14  for 1 ← 0 to B - 1 do
15    block += get_statement(D, B, Pfp, NM, rand() % N)
16  return block

```

5 TEMPERATURE EMULATION

In an ASIC, the temperature monitor uses physical sensors to measure the chip temperature. However, this measurement is not possible on an FPGA prototype, because its thermal behavior differs from an ASIC. Therefore, our approach emulates the compact RC-thermal model, which is also used by state of the art thermal simulators such as HotSpot [48], on the FPGA. The thermal model is composed of four vertical conductive layers, corresponding to the die, the **thermal interface material (TIM)**, the heat spreader and the heat sink. Each of the layers is divided into N thermal nodes corresponding to the components on the processor plus twelve extra nodes, which are located on the heat spreader and the heat sink. Furthermore, the nodes are interconnected through thermal conductances and thermal capacitances to model the thermal behavior of the system. Considering this thermal network, it is possible to express the behavior of all $M = 4N + 12$ thermal nodes by a first-order differential equation. Using the mathematical notation from Reference [35], the following equation is obtained:

$$AT' + BT = P + T_{amb}G. \quad (1)$$

Thereby, matrix $A = [A_{ij}]_{M \times M}$ describes the thermal capacitances, matrix $B = [B_{ij}]_{M \times M}$ describes the thermal conductances between nodes, vector $G = [G_i]_M$ describes the thermal conductances between nodes and ambiance, vector $P = [P_i]_M$ describes the current power consumption of all nodes, vector $T = [T_i]_M$ describes the temperature of each node, vector $T' = [T'_i]_M$ describes the first order derivative of the the temperature of each node and T_{amb} the ambient temperature. Note that the subscript of the matrices and vectors indicate their dimensions. While it is possible to implement the numerical solution of this differential equation directly in hardware, such an centralized implementation comes with several undesirable implications. First, the size of the matrices and thus, the computational effort scales quadratically with the number of thermal nodes on the processor. As a result, either the latency or the hardware overhead of the temperature emulator

Fig. 2. Temperature emulation approach.

scales quadratically as well. Second, the temperature is not emulated where it is needed. Especially on large-scale multi-FPGA ASIC prototypes, many routing and IO resources would be required to transfer the power consumption of each thermal node to the temperature emulator and the temperatures back to the thermal nodes. Additionally, multiple pipeline stages would be required to reduce the path delays all over the prototype.

These major limitations motivated us to develop a decentralized emulation approach that scales linearly in complexity with the number of thermal nodes. Instead of considering the temperature and the power consumption of all thermal nodes, we consider only the temperature of the direct neighbors of a thermal node. Thus, only the temperatures of the thermal nodes that are in any case placed next to each other need to be exchanged. Furthermore, it should be noted that a thermal node is still indirectly affected by the temperature of all other nodes. This is possible, since the temperature of each emulator affects the temperature of its neighbors, who then again affects the temperature of their neighbors. As a result, the emulated heat transfer on the many-core processor is similar to actual physical effect. In this process, we model the temperatures of a thermal node $T_i(n) = [T_i^j(n)]_4$ on the conductive layers j at iteration n by its power consumption $P_i(n)$, its temperature from the previous iteration $T_i(n-1) = [T_i^j(n-1)]_4$ and the temperatures of all neighboring nodes from the previous iteration, as illustrated in Figure 2. On a heterogeneous floorplan, where the thermal nodes vary in size, it is possible that one node has multiple neighbors in one of the four cardinal directions. In this case, the temperatures of all neighbors from the previous iteration are incorporated as input for the linear model. As a result, the size of the input vector depends on the number of neighbors. Hence, the following linear model for the temperature emulation of a thermal node i is obtained, where $\beta_i = [\beta_j]_{4 \times (4n_i+5)}$ corresponds to the regression matrix and n_i to the number of neighbors of node i :

$$T_i(n) = \beta_i \begin{bmatrix} P_i(n) \\ T_i(n-1) \\ T_{i,north}(n-1) \\ T_{i,east}(n-1) \\ T_{i,south}(n-1) \\ T_{i,west}(n-1) \end{bmatrix}. \quad (2)$$

Whenever a node is located at the border of the floorplan and no direct neighbor is available, the ambient temperature T_{amb} is chosen instead. As the regression matrices of the cores might vary in size, we fit the linear model for each of the nodes independently based on the results of the thermal simulator MatEx [35]. The linear model is then implemented in hardware using a **multiply and**

Fig. 3. The voltage and frequency transition between two V/f levels.

accumulate (MAC) unit. Similar to the power emulation also the temperature emulation will be updated every 256 cycles, such that a single MAC unit per core is sufficient to compute the temperatures.

6 DVFS EMULATION

DVFS is a well-know technique that aims to reduce the power consumption of a processor by scaling its supply voltage and frequency. On a multi- and many-core processor, this technique can be used to balance the trade-off between the chip temperature and the speed of the running applications. Thus, it is not sufficient to only scale the power consumption on an emulation framework, as presented by Listl et al. [25] but also the processor speed. Additionally, each transition between V/f levels introduces an energy and timing overhead that needs to be considered as well. Therefore, we first model the DVFS overhead based on Park et al. [37] and present our emulator design afterwards.

6.1 DVFS Overhead Model

Figure 3(a) illustrates the supply voltage and the frequency over time during the transition between different V/f levels. In the case of up-scaling, the DC-DC converter needs to increase the supply voltage before the frequency can be changed. Yet in the case of down-scaling, the PLL first changes the frequency before the DC-DC converter reduces the supply voltages. As a result, a static and dynamic energy overhead E_{uc} is introduced during the voltage transitions, because the supply voltage is higher than necessary (i.e., the processor is underclocked) and a static overhead E_{PLL} is introduced during the lock time of the PLL. The combination of both overheads is expressed in Equation (3):

$$E_O = \begin{cases} E_{PLL} + E_{uc,up} & \text{up-scaling,} \\ E_{PLL} + E_{uc,down} & \text{down-scaling.} \end{cases} \quad (3)$$

As down-scaling is typically faster than up-scaling, we differentiate between the respective overheads $E_{uc,down}$ and $E_{uc,up}$. In addition to the energy overhead, the transition between V/f levels also introduces a timing overhead τ_O . In the case of up-scaling, the overhead is introduced by the underclocking during the time of the voltage transition τ_x and the lock time of the PLL τ_{PLL} . However, in the case of down-scaling, the overhead is only introduced by the lock time of the PLL τ_{PLL} . Thus, the timing overhead can be summarized as illustrated in Equation (4), where f_s corresponds to the current frequency before the transition and f_e corresponds to the frequency

after the transition:

$$\tau_O = \begin{cases} \tau_{PLL} + \frac{f_e - f_s}{f_e} \tau_x & \text{up-scaling,} \\ \tau_{PLL} & \text{down-scaling.} \end{cases} \quad (4)$$

6.2 Design of the DVFS Emulator

Figure 3(b) illustrates the behavior of our voltage and frequency emulation approach during a transition between different V/f levels. For the voltage emulation, we increase or reduce the voltage level in steps until the target voltage V_e is reached. To calibrate the slew rate of the DC-DC converter, the transition time τ_e between the voltage levels can be configured individually for up- and down-scaling. As this method does not account for voltage overshoots during up-scaling, we additionally increase the power consumption of the processor by a one-time energy overhead of E_{os} . Thus, the modelled energy consumption of the processor during the voltage transition can be described by Equations (5) and (6) for up- and down-scaling, respectively. Here, l_e and l_s describe the V/f level before and after the voltage transition, the interval $[l_e, l_s]$ describes the set of V/f levels between l_e and l_s including l_s , and P_l the power consumption of the processor at V/f level l :

$$E_{uc,up} = \int_0^{\tau_x} P(t) dt \approx \left(\sum_{l \in [l_s, l_e]} P_l \tau_{e,up} \right) + P_{l_e} (\tau_x - |[l_s, l_e]| \tau_{e,up}) + E_{os}, \quad (5)$$

$$E_{uc,down} = \int_0^{\tau_x} P(t) dt \approx \left(\sum_{l \in [l_s, l_e]} P_l \tau_{e,down} \right) + P_{l_e} (\tau_x - |[l_s, l_e]| \tau_{e,down}). \quad (6)$$

The PLL-induced energy overhead E_{PLL} is implicitly considered by the power monitor, since the processor pipeline is stalled during a frequency transition. Furthermore, we model the voltage transition time by a linear function, illustrated in Equation (7), with two parameters τ_m and τ_o that model the slope and the intercept of the function:

$$\tau_x \approx \tau_m |[l_s, l_e]| + \tau_o. \quad (7)$$

As a result, the FPGA model for DVFS emulation comprises five parameters $\tau_{e,up}$, $\tau_{e,down}$, E_{os} , τ_m , and τ_o , which can be calibrated for a specific DC-DC converter. Therefore, it is necessary to measure or simulate the energy consumption of the processors and the voltage transition time during up- and down-scaling between all possible V/f levels. Using these measurements, we obtain the model parameters by linear regressions.

For the frequency emulation, we stall the processor pipeline to slow it down. In this process, each of the supported frequencies is mapped to a number of active cycles, in which the processor executes the application, and a number of stall cycles, in which the processor pipeline is stalled. Consequently, the highest emulated frequency corresponds to one active cycle followed by zero stall cycles. To reduce the processor speed by one third, two active cycles would be followed by one stall cycle. With this method, the fine-grained emulation of any frequency level is possible. Furthermore, the frequency emulator accounts for the transition overhead introduced by the DC-DC converter by initiating a frequency transition to f_x only if the current supply voltage is either higher or equal to the corresponding voltage level V_x . Additionally, the frequency scaler inserts $f_{max} \tau_{PLL}$ stall cycles at each frequency transition to account for the timing overhead introduced by the PLL. While this emulation approach does not cover all effects of frequency scaling on the micro-architecture or the memory system of the processor, the simplicity and the flexibility of this approach outweighs these effects. In contrast to actual frequency scaling on the FPGA, this method enables to configure τ_{PLL} , τ_x and the number of supported V/f levels. Thus, the timing

Fig. 4. System architecture of the experimental setup.

overhead and the number of V/f levels of DVFS are not defined by the FPGA but can be configured instead.

7 EXPERIMENTAL RESULTS

In this section, we demonstrate the proposed evaluation framework. First, we discuss the experimental setup. Subsequently, we demonstrate the performance of the benchmark generator, the temperature emulator and the DVFS emulator, followed by the performance of the complete evaluation approach. Finally, we present the suitability of the approach in a case study.

7.1 Experimental Setup

For the following demonstrations, we integrate the evaluation approach into a tiled many-core processor, illustrated in Figure 4. The processor is implemented on a proFPGA platform [13] consisting of four Virtex-7 FPGAs. Thereby, the processor consists of 16 tiles and five cores per tile, yielding an 80-core processor. Each processor has a dedicated 8 kB L1 instruction and an 8 kB L1 data cache. Furthermore, all cores on a tile share a 512 kB L2 cache for remote tile memory accesses and an 8 MB tile-local memory. While the caches are implemented as **block RAM (bRAM)** on the FPGA, all tile local memories are physically located on an SRAM extension board, where each tile is mapped to a different bank. The FPGA platform runs at 50 MHz and emulates an ASIC target frequency of 4 GHz. For the ASIC emulation, we generate the temperature model using the default thermal chip characteristics provided by the state of the art thermal simulator Hotspot. The power emulator is based on the concepts of Listl et al. [25]. For this, we synthesize a LEON3 processor and run gate-level simulations to characterise the switching activity of the processor for all instructions individually. This information can then be used by the Synopsys tool PrimePower to simulate the power consumption of the processor based on its system status. In the power emulator, we store the simulation results in LUTs. Based on the current status of the cores, the respective power consumption is then loaded and scaled according to the selected V/f level.

The monitoring aspect of the evaluation approach extends the monitoring architecture of Mettler et al. [30]. The architecture is composed of three types of components: a set of probes, a set of tile monitors, and a tracing interconnect. A *probe* is assigned to each core in the system. It extracts events from the trace data provided by the core, the power monitor and the temperature monitors to reduce the data volume. For instance, it is possible to detect events based on the program counter address of the processor. Thus, a probe could inform a monitor about the start or the end of an executed task. Furthermore, the probes support to define events on power and temperature ranges such that the violation of a power corridor or a temperature threshold could be detected. The detected events are then sorted and distributed by the *tracing interconnect* to all tile monitors. As a result, all tile monitors have a consistent view on a globally sorted trace of events. The *tile monitors* support temporal and logical supervision. Using the temporal supervision, it is not only possible to non-intrusively evaluate the execution time of an application but also to measure the time a core violates its predefined power corridor or thermal threshold. Furthermore, the logical supervision can be used to verify the control flow of the application or the behavior of the resource manager. Thus, the monitoring architecture can not only be used to collect run-time data of the system but also to verify the implementation of the management strategy simultaneously.

7.2 Benchmark Generation

For an application-independent analysis of a resource or thermal management strategy, it is important to evaluate its performance on benchmarks with various execution behaviors. Therefore, we evaluate the run-time characteristics of the synthetic benchmarks generated by the REX process using a depth parameter $D \in \{1, 2\}$, a breadth parameter $B \in \{2, 3\}$, a floating-point probability parameter $P_{fp} \in \{0, 0.25, \dots, 1\}$ and a memory size of $N_M \in \{256 B, 2028 B, 32768 B\}$. We generate 50 benchmarks for each combination of the input parameters, such that we evaluate the run-time characteristics of 3,000 benchmarks.

The instruction cache hold rate and the data cache hold rate of the generated benchmarks are illustrated in Figures 5(a) and 5(b) over the execution time. Thereby, the point cloud of the instruction cache hold rate decreases until a run-time of 10 μs . This behavior can be explained by the low number of instructions of such short benchmarks. In contrast, benchmarks with a higher run-time show varying instruction cache hold rates between 100 and 600 $\frac{1}{\mu s}$. In exceptional cases, the instruction cache hold rate reaches a value of up to 1,600 $\frac{1}{\mu s}$. In contrast, the data cache hold rate of the benchmarks ranges between 300 and 1500 $\frac{1}{\mu s}$ for most execution times. However, it is noticeable that short benchmarks tend to have a higher data hold rate than long benchmarks. This behavior can be attributed to the fact that data cache first needs to be filled before the locality of the data can be exploited. Similarly, benchmarks with a long execution time contain many loop statements that are more likely to operate on local data. The integer and the floating-point instruction rate are illustrated in Figures 5(c) and 5(d), respectively. The integer instruction rate varies between 300 and 1,400 $\frac{1}{\mu s}$. Furthermore, an increase over the execution time is visible. This behavior matches the decreasing cache hold rates, as a lower cache rate enables an higher instruction count. In contrast to that, the floating-point rate varies uniformly over all execution times between 0 and 200 $\frac{1}{\mu s}$. Overall the generate benchmark show a great diversity in the memory access and compute intensity.

Finally, we demonstrate that the generated benchmarks cover a wide range of run-time characteristics by mimicking the behavior of real workloads. Therefore, we choose an object detection algorithm, whose actor graph is illustrated in Figure 6(a). All actors run in parallel on different tiles of the many-core processor and forward the data of the input images through the object detection pipeline. The application is implemented using the ActorX10 library [41] of the X10 programming language [46]. As the language implements the **asynchronously partitioned global address**

Fig. 5. The run-time behavior of 3,000 tasks generated by the recursive expansion process using different input parameters.

space (APGAS) programming model, it is especially suited for many-core processors. To mimic the behavior of the application, we first characterize each actor independently using the statistics unit of the LEON cores. The run-time characteristics can then be used to identify the best suited benchmark for each actor, respectively. As X10 is a managed programming language, the task characteristics may change when running several tasks together on a single tile, which would lead to a mismatch compared to the benchmark that tries to mimic this application. This can be remedied by post-tuning the benchmark. Thus, it is possible to iteratively match the run-time characteristics of the benchmarks with the characteristics of the real applications, even considering contentions. In Table 6(b), we compare the execution time t_{exe} , the performance vector C_t and the maximal temperature T_{max} of each actor of the real applications with each actor of the emulated application. Thereby, the maximal temperature of the emulated actors differs on average by less than 2 °C. Furthermore, the table shows that the accuracy of the thermal behavior depends on the accuracy of the performance vectors. For example, the performance vector of the emulated **corner detection (CD)** actor matches well with the actor of the real application and thus, also the maximal temperatures match well with each other. However, the performance vector of the emulated **SIFT Orientation (SO)** actor especially differs in the integer instruction rate c_{int} from the real actor and thus, also the maximal temperatures differ noticeably. As a result, it is expected that the accuracy of the thermal behavior further improves with the size of generated benchmarks, which increases the size of potential candidates to match the behavior of an actor. In summary, the results show

(a) Actor Graph of the Application

(b) Thermal Characteristics of the Object Detection Chain and Its Generated Copy

Actor	Tile	Object Detection Chain			Emulated Application			$ \Delta T_{max} $ [°C]
		t_{exe} [us]	$(c_{IC}, c_{DC}, c_{int}, c_{fp})$ [$\frac{1}{us}$]	T_{max} [°C]	t_{exe} [us]	$(c_{IC}, c_{DC}, c_{int}, c_{fp})$ [$\frac{1}{us}$]	T_{max} [°C]	
GSC	1	1167	(7.71, 633, 480, 139)	76	1268	(129, 620, 421, 124)	74	2
ED	1	1042	(10.9, 1111, 877, 0)	67	1065	(28.3, 1153, 691, 0)	71	4
CD	2	11160	(3.49, 542, 773, 128)	81	11331	(9.66, 550, 603, 132)	81	0
SO	3	4509	(37.5, 403, 1031, 109)	88	4503	(50.6, 488, 597, 146)	82	6
SDE	4	52.3	(187, 549, 403, 0.13)	51	107	(139, 630, 578, 24.5)	52	1
SD ₀	4	4510	(362, 1291, 587, 13)	77	4636	(144, 1555, 670, 46)	73	4
SD ₁	4	4334	(356, 1309, 585, 13)	75	4636	(142, 1543, 670, 46)	75	0
SD ₂	4	4434	(364, 1374, 567, 13)	75	4636	(144, 1540, 669, 46)	73	2
SD ₃	4	4251	(360, 1395, 554, 13)	74	4651	(143, 1530, 667, 46)	73	1
SME	5	42	(258, 536, 270, 0.71)	48	57	(210, 687, 304, 30.2)	49	1
SM ₀	5	1148	(183, 1652, 5.65, 506)	66	1402	(54.9, 1581, 742, 34)	67	1
SM ₁	5	1150	(177, 1580, 5.65, 526)	67	1401	(56.9, 1578, 743, 34)	66	1
SM ₂	5	1237	(190, 1785, 5.25, 475)	65	1401	(54.5, 1576, 743, 34)	67	2
SM ₃	5	1280	(171, 1521, 5.08, 481)	65	1426	(54.0, 1544, 730, 33)	67	2
RSC	6	6.98	(812, 685, 589, 0.86)	48	1269	(1075, 1185, 481, 0.75)	47	1

Fig. 6. The ability of the benchmark generator to represent real applications, shown on the example of an object detection chain.

that the generated benchmarks cover the run-time behavior of the object detection chain and additionally allow one to generate a large variety of run-time behaviors for the evaluation of run-time management strategies.

7.3 ASIC Temperature Emulation

The key performance indicators of an ASIC temperature emulation approach are its scalability, its accuracy and its hardware overhead. First, we compare the scalability of our **distributed temperature emulation (DTE)** model with the scalability of the numerical solutions of the RC-thermal network by the **Runge-Kutta method (RK4)** and the **time-invariant linear thermal system (TILTS)**. Both numerical methods have been used by Alam et al. [2] to implement a temperature emulator on an FPGA prototype. The scalability comparison between the different methods is illustrated in Figure 7(a) for the number of model parameters, and in Figure 7(b) for the number of multiplications per iteration. Since only our approach scales linearly with the number of processors in terms of model parameters and multiplications, we outperform the other approaches by

Fig. 7. A comparison in scalability of different thermal emulation approaches.¹

Fig. 8. The temperature emulation error for a 96 core many-processor measured within periods of $2 \mu\text{s}$ intervals over a total emulation time of 64 ms compared to MatEx.

more than an order of magnitude on many-core processors. This comparison depicts the strengths of the decentralized emulation approach.

In addition to the scalability, also the accuracy of the emulation approach is important. Therefore, we compare the emulated temperature of our decentralized approach with the numerical solutions of the temperature model on a 80-core processor over 1,000,000 iterations, corresponding to an execution time of 64 ms. The maximal emulation error, measured within intervals of $2 \mu\text{s}$, is illustrated in Figure 8(a). With a maximal emulation error below 0.04°C , the accuracy of the emulation approach is more than sufficient. Furthermore, the histogram shows a clear maximum at an emulation error of 0°C , which is desirable as well. Additionally, we verify the accuracy of the 32 bit fixed point implementation of the temperature mode in Figure 8(b). Here, the histogram shows a comparable behavior such that we can conclude that the emulation accuracy on the FPGA implementation is sufficient as well. Additionally, we evaluate the emulation accuracy

¹The authors of Reference [2] state that the two matrix vector multiplications of TILTS and RK4 scale cubically with the number of thermal nodes, i.e., cores. In our opinion, this assumption is overly pessimistic. Therefore, we consider a quadratic scaling in this evaluation, which makes TILT and RK4 even more competitive.

Table 1. The Temperature Emulation Error for Different Architectures Measured Within Periods of $2 \mu\text{s}$ Intervals Over a Total Emulation Time of 64 ms

Processor	Cores	Thermal Nodes	Mean Absolute Emulation Error	Maximal Emulation Error
Alpha EV6	1	30	0.007°C	0.049°C
Emulation Platform	80	96	0.004°C	0.034°C
Heterogeneous Many-core	92	147	0.022°C	0.540°C
Intel Gainestown Dual-core	2	13	0.008°C	0.047°C

Table 2. The Hardware Overhead of the Temperature Monitor

	DTE [ours]		TTE [2]
	Single-core	16 Cores	16 Cores
Slice Registers	1,113	17,811	10,388
Slice LUTs	496	7,942	4,292
DSPs	4	64	6

of four architectures with various numbers of thermal nodes against Hotspot in Table 1. For each of the architectures, the mean average emulation error is below 0.03°C . Also most architectures achieve a maximal emulation error of less than 0.05°C . Even the maximal emulation error of the heterogeneous many-core processor, consisting of various core types and accelerators, is well acceptable at 0.55°C . Finally, we evaluate the hardware overhead of our temperature emulator in Table 2. Thereby, each thermal node requires a 32 bit MAC unit to compute the local temperatures. This unit has been synthesized into four **digital signal processing blocks (DSPs)** on the FPGA. A comparison with the hardware overhead of a 22 bit fixed-point implementation of **TILTs Thermal Emulation (TTE)** IP for 16 thermal nodes, yields that TTE required 10 times fewer DSPs. However, this comparison does not reflect the emulation latency. While the centralized approach requires 3,496 cycles to compute the temperature per iteration, our approach requires 84 cycles only. Especially, for many-processors, like on our evaluation platform, the 2-MAC design of TTE would already require 178,584 cycles per iteration, while our decentralized approach still requires 84 cycles only. Furthermore, the additional effort to route the power signals from the respective thermal node to the TTE IP and the effort to route the temperature signals back is impracticably large. Large-scale FPGA prototype span across multiple FPGAs. Thus, power and temperature signals do not only need to be pipelined to meet the timing requirements but also many IO resources are needed to send the signals to the IP, which can only be located on one FPGA. As a result, especially on large-scale prototypes, a decentralized temperature emulation approach is inevitable to minimize the design effort and to save hardware resources.

7.4 Dynamic Voltage Frequency Scaling Emulation

In this section, we first evaluate the proposed DVFS emulation approach in Table 3. Therefore, we calibrate the FPGA model for three exemplary processors and compare its accuracy with a macromodel, proposed by Park et al. [37]. In this process, we compute the accuracy of both methods based on SPICE simulations conducted by Park et al. It can be seen that the FPGA model outperforms the macro model in terms of **mean absolute error (MAE)** and **mean relative error (MRE)** for the voltage transition time τ_{uc} and the energy consumption during down-scaling $E_{uc,down}$. Even though the macro model achieves better results for the energy consumption of the processors during up-scaling $E_{uc,up}$, it should be noted that the absolute error of the FPGA model

Table 3. The Accuracy of the DVFS Model

Processor		FPGA-Model [ours]		Macromodel[37]	
		MAE	MRE	MAE	MRE
Intel Core2 Duo E6850 with LTC3733	$E_{uc,up}$	220 μJ	7.97%	30.5 μJ	1.22%
	$E_{uc,down}$	19.0 μJ	0.81%	36.3 μJ	1.52%
	τ_{uc}	6.35 μs	7.11%	10.8 μs	13.9%
Samsung Exynos 4210 with LTC3568	$E_{uc,up}$	1.63 μJ	10.9%	0.11 μJ	0.67%
	$E_{uc,down}$	0.27 μJ	1.80%	0.52 μJ	3.56%
	τ_{uc}	2.13 μs	9.99%	8.97 μs	47.3%
MSP430 with LTC3632	$E_{uc,up}$	8.31 μJ	1.25%	7.49 nJ	0.02%
	$E_{uc,down}$	0.29 μJ	1.81%	0.48 μJ	3.67%
	τ_{uc}	5.23 μs	5.01%	6.01 μs	4.90%

Table 4. The Effect of DVFS Overheads on the Execution Time of an Application for Different DTM Strategies

Lower Thermal Threshold	Upper Thermal Threshold	Throttling with DVFS Overhead	Throttling without DVFS Overhead	Relative Difference
69 °C	80 °C	257 ms	227 ms	11.7%
71 °C	80 °C	256 ms	221 ms	13.7%
73 °C	80 °C	257 ms	215 ms	16.3%
75 °C	80 °C	263 ms	211 ms	19.8%
77 °C	80 °C	266 ms	206 ms	22.6%
79 °C	80 °C	266 ms	201 ms	24.4%

is sufficient. Furthermore, we evaluate the impact of the DVFS overhead on the design of a state-of-the-practice hardware DTM [21]. The DTM monitors the temperature of a core and reacts on a thermal violation by throttling down the V/f level to a minimum value. Once the core temperature decreases below a lower thermal threshold, the V/f level of the core is re-set to their peak value. The challenge in the design of such a DTM is to define a suitable lower thermal threshold, since the upper threshold is already defined by the safe operating temperature of the processor (here 80 °C). Therefore, we evaluate the execution time of an application for different lower thermal thresholds without considering DVFS overheads and with considering DVFS overheads in Table 4. It can be seen that for a given lower thermal threshold, the execution time is always higher when the DVFS overheads are considered. This behavior is mostly introduced by the voltage transition timing overhead. Even though the transition time is in the order of micro seconds, the overhead accumulates as the the DTM continually switches between the highest and the lowest frequency.

The greater the difference between the lower and the upper thermal threshold the longer is the throttling period after each thermal violation. Hence, the implementation with and without DVFS show very similar execution times for the smaller lower thresholds. Here, the number of V/f transitions are limited. In contrast, a larger lower thermal threshold implies a large number of V/f transitions. Hence, the DVFS overheads impact the execution time of the application significantly. As a result, an evaluation of the execution time without DVFS overheads suggests the choice of a high lower thermal threshold. However, the evaluations with the consideration of the DVFS overhead show that a smaller lower thermal threshold is actually better suited to maximize the performance of the application. Thus, the DVFS overheads must be emulated in the evaluation approach.

Table 5. The Hardware Utilization of the Evaluation Platform

	Slice Registers	Slice LUTs	DSPs
80-core processor	1,604,265	2,567,082	1,184
Power Emulation	23,920	25,360	480
Temperature Emulation	39,680	89,056	384
DVFS Emulation	12,720	14,160	0
Runtime Verification	97,648	166,020	0
Utilization on 4×Virtex-7 FPGAs	16.4%	52.5%	13.7%
Utilization on 4×Ultrascale FPGAs	11.6%	23.2%	2.4%

7.5 Evaluation Platform

In this section, we evaluate the scalability and the performance of the evaluation approach. Therefore, we illustrate the hardware utilization of the proFPGA system in Table 5. The 80-core processor uses 1,604,265 slice registers, 2,567,082 slice LUTs, and 1,184 DSPs on the four FPGAs. While this is a lot, the system uses only 16.4% of the available slice registers, 52.5% of the available slice LUTs, and 13.7% of the available DSPs on the FPGA. Thus, the number of available LUTs limits the scalability of the approach. In a first-order approximation, one would could estimate that up to ~150 cores can be integrated on four Virtex-7 FPGAs. However, on four Xilinx Ultrascale FPGAs, the system could be scaled up further to a maximum of ~340 cores. As a result, FPGAs prototypes are also suitable to demonstrate many-core systems.

Besides the scalability, the evaluation performance is also important. Therefore, we compare the simulation speed of the different approaches in MIPS. For this comparison, we assume that a single LEON3 core executes 0.7 **instructions per cycle (IPC)**. Thus, the target ASIC design, consisting of 80 cores and a maximal frequency of 4 GHz, could provide a peak performance of 224,000 MIPS. On the FPGA prototype, the cores run at 50 MHz. Thus, it provides a peak performance of 2,800 MIPS. As a result, the FPGA prototype is only two order of magnitudes slower the the target system. Hence, it is still possible to do rapid evaluations and run the target application on top of an operating system. This is a performance that common simulation-based approaches can not achieve. Although, sniper does not perform cycle-accurate simulations it only achieves a performance of ~2 MIPS [8, 19], which is five orders of magnitudes slower than the target design and three orders of magnitudes slower than the FPGA prototype. A cycle-accurate simulator, such as GEM5, achieves only a performance of ~0.3 MIPS [18], making it six orders of magnitudes slower than the target design. In summary, simulation-based approaches do not provide the performance, which is needed to evaluation thermal and resource management strategies of many-core processors.

7.6 Case Study

In this case study, we employ our proposed platform to evaluate a state-of-the-art system-level thermal management technique based on power budgeting [40] and compare it with the state-of-the-practice DTM evaluated in Section 7.4, as a baseline. Thereby, we assume a safe operating temperature of 80 °C for the emulated processor. The DTM technique is reactive, and non-predictable. Hence, it is not possible to give timing guarantees for real-time tasks at design time, since DTM can be triggered at any point at run-time. To provide predictability, the concept of **Thermal Safe Power (TSP)** [36] can be employed. Thermal Safe Power (TSP) is a per-core power budget that guarantees avoiding thermal violations. There are multiple variants for TSP; uniform and non-uniform ones. In this case study, we employ the non-uniform TSP, which means different power budgets per task are calculated. Moreover, the worst-case schedule of parallel running tasks w.r.t.

Fig. 9. Task graph and experimental results of the first use-case. Frequent triggering of DTM in the baseline reduces the performance compared to an execution at constant thermally safe frequencies with a power budgeting technique.

temperature is taken into account to provide guarantees at design time that thermal violations will not occur at run-time. The resulting power budget of each task is mapped to a thermally safe frequency f_{safe} based on the power profile of the task.

To evaluate this technique, we conduct two use-cases with applications generated using the benchmark generation infrastructure earlier introduced. In the first experiment, one application is employed (illustrated in Figure 9(a)) and it follows the scatter-gather pattern, which is often seen in parallel benchmarks. The thermally safe power budgets are calculated for all tasks. Since t_0 and t_{13} can never run in parallel to another task, they get a high power budget, which allows them to run at the peak frequency. Contrarily, for t_{01} to t_{12} , it is possible that all of these tasks are running in parallel (depending on the actual execution time of each task). As mentioned, calculating the thermally safe power budgets accounts for the worst-case schedule, thereby the power budgets of these tasks are lower, restricting them to run at lower frequencies; i.e., 3.0 and 3.3 GHz. The difference in the frequencies is because of the resulting non-uniform power budgets. Particularly, t_9 and t_{11} are mapped next to several idle cores, which allows a higher power budget for them, and thereby higher f_{safe} . Executing the tasks at their selected f_{safe} avoids thermal violations (i.e., the maximally emulated temperature T_{max} does not reach the thermal threshold of 80 °C) as our platform demonstrates (Figure 9(b)). However, when this power budgeting technique is not employed, all tasks are executed at the peak frequency, but then thermal violations occur. Consequently, DTM is triggered frequently and throttles down the frequencies to return the system to a thermally safe state. Hence, the frequencies are oscillating between peak and minimum levels, and the average frequency is less than f_{safe} . Therefore, execution times of most parallel-running tasks are higher than with thermally safe power budgeting, as shown in Figure 9(b).

The second use-case is shown in Figure 10, in which two applications are running in parallel. The first application has an early parallel phase, while the second application has a late parallel phase. Since the actual execution times of the tasks are not known at design-time, thermally safe power budgeting considers the worst-case schedule. For example, when calculating the power budget of task t_{01} , the worst-case schedule is that the tasks t_{02} to t_{04} and t_{12} to t_{15} all are running in

Fig. 10. Task graphs and experimental results of the second use-case. Without knowing the execution times of tasks at design-time, the worst-case schedule needs to be considered for calculating the power budgets as indicated for task t_{01} .

parallel to it, as illustrated in the figure. However, at run-time, for certain input data, these tasks do not overlap. As a result, the maximum emulated core temperatures T_{max} are far from the thermal constraint, as shown in the table in Figure 10(b). That demonstrates that there is a price in terms of performance that needs to be paid to provide the thermal-safety guarantees at design time. In case the power budgeting technique is not employed, thermal violations occur and DTM is triggered, throttling down the V/f levels. However, in this particular experiment, the negative impacts of triggering DTM on the execution times of the tasks are lower than the price paid for the thermally safe power budgets, as illustrated in Figure 10(b).

In summary, our approach enables a comprehensive analysis of a state-of-the-art thermally safe power budgeting technique that highlights its potential to guarantee a thermally safe execution and also its implied pessimism. Moreover, the approach demonstrates the negative impacts of the state-of-the-practice DTM on performance and how it hinders predictability. These findings could only be established by the combination of benchmark generation, ASIC emulation, and run-time monitoring. Thereby, the proposed benchmark generation approach enables the generation of applications, which reveals the advantageous and disadvantageous of both thermal management techniques. The proposed ASIC emulation approach enables the ASIC power, temperature and DVFS emulation, which makes the analysis possible in the first place and the run-time monitoring architecture, which provides performance indicators, such as the maximal emulated core temperature and the execution time of each task, to compare and analyse both techniques.

8 CONCLUSION

In this article, we presented an FPGA-based evaluation approach for resource and thermal management strategies of many-core processors. The approach combines methods from benchmark generation, ASIC emulation, and run-time monitoring to unite the tracing capabilities of

simulation-based evaluations with the high simulation speed and the high accuracy of FPGA-based evaluations. In this context, a benchmark generation approach has been presented, which enables an evaluation of run-time management strategies for various application scenarios. Furthermore, the ASIC emulation platform consists of a novel temperature emulator design, which scales linearly in terms of complexity with the number of integrated cores, and a novel DVFS emulator design, which considers timing and energy overheads between V/f level transitions. In our evaluations, we demonstrate the suitability of our approach by the evaluation of a state-of-the-art thermal management strategy in a case study.

REFERENCES

- [1] A. Akram and L. Sawalha. 2019. A survey of computer architecture simulation techniques and tools. *IEEE Access* 7 (2019), 78120–78145. <https://doi.org/10.1109/ACCESS.2019.2917698>
- [2] M. S. Alam and A. Garcia-Ortiz. 2017. An FPGA-based thermal emulation framework for multicore systems. In *Proceedings of the 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS'17)*. 1–6.
- [3] D. Atienza, P. G. Del Valle, G. Paci, F. Poletti, L. Benini, G. De Micheli, and J. M. Mendias. 2006. A fast HW/SW FPGA-Based thermal emulation framework for multi-processor system-on-chip. In *Proceedings of the 43rd Annual Design Automation Conference (DAC'06)*. Association for Computing Machinery, New York, NY, 618–623.
- [4] A. Bhattacharjee, G. Contreras, and M. Martonosi. 2008. Full-system chip multiprocessor power evaluations using FPGA-based emulation. In *Proceeding of the 13th International Symposium on Low Power Electronics and Design (ISLPED'08)*. 335–340.
- [5] C. Bienia. 2011. *Benchmarking Modern Multiprocessors*. Ph.D. Dissertation. Princeton University.
- [6] N. Binkert, B. Beckmann, et al. 2011. The gem5 simulator. *SIGARCH Comput. Archit. News* 39, 2 (Aug. 2011), 1–7.
- [7] T. D. Burd and R. W. Brodersen. 2000. Design issues for dynamic voltage scaling. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED'00)*. Association for Computing Machinery, New York, NY, 9–14.
- [8] T. E. Carlson, W. Heirman, and L. Eeckhout. 2011. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11)*. Association for Computing Machinery, New York, NY, Article 52, 12 pages.
- [9] M. Cochet, G. Bonnechere, J. Daveau, F. Abouzeid, and P. Roche. [n.d.]. Implementing the LEON3 statistics unit in 28nmFD-SOI: Power estimation by activity proxy. Retrieved from <https://www.gaisler.com/doc/antn/ext/implementing-leon3-statistics-final.pdf>.
- [10] Ampere Computing. 2020. 2020 Vision Leads to True Cloud Native in 2021. Retrieved from <https://amperecomputing.com/2020-vision-leads-to-true-cloud-native-in-2021/>.
- [11] EEMBC Embedded Microprocessor Benchmark Consortium. CoreMark. Retrieved from <https://www.eembc.org/coremark/>.
- [12] Standard Performance Evaluation Corporation. 2021. SPEC CPU 2017. Retrieved from <https://www.spec.org/cpu2017/>.
- [13] Pro Design. Retrieved from <https://www.profpga.com/products/systems-overview/virtex-7-based/profpga-quad-v7>.
- [14] J. Dujmović. 2010. Automatic generation of benchmark and test workloads. In *Proceedings of the 1st Joint WOSP/SIPEW International Conference on Performance Engineering (WOSP/SIPEW'10)*. Association for Computing Machinery, New York, NY, 263–274.
- [15] Y. Ge, Q. Qiu, and Q. Wu. 2012. A multi-agent framework for thermal aware task migration in many-core systems. *IEEE Trans. Very Large Scale Integr. Syst.* 20, 10 (2012), 1758–1771. <https://doi.org/10.1109/TVLSI.2011.2162348>
- [16] E. Glocker, Q. Chen, U. Schlichtmann, and D. Schmitt-Landsiedel. 2017. Emulation of an ASIC power and temperature monitoring system (eTPMon) for FPGA prototyping. *Microprocess. Microsyst.* 50 (2017), 90–101.
- [17] E. Glocker, Q. Chen, A. M. Zaidi, U. Schlichtmann, and D. Schmitt-Landsiedel. 2015. Emulation of an ASIC power and temperature monitor system for FPGA prototyping. In *Proceedings of the 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC'15)*. 1–8.
- [18] X. Guo and R. Mullins. 2020. Accelerate cycle-level full-system simulation of multi-core RISC-V systems with binary translation. In *Proceedings of the 4th Workshop on Computer Architecture Research with RISC-V*.
- [19] W. Heirman, S. Sarkar, T. E. Carlson, I. Hur, and L. Eeckhout. 2012. Power-aware multi-core simulation for early design stage hardware/software co-optimization. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques (PACT'12)*. 3–12.
- [20] Carl Hewitt. 2015. Actor Model of Computation: Scalable Robust Information Systems. Retrieved from <https://arXiv:cs.PL/1008.1459>.

- [21] Intel. Intel Xeon Phi Processor 7250. Retrieved from <https://ark.intel.com/content/www/us/en/ark/products/94035/intel-xeon-phi-processor-7250-16gb-1-40-ghz-68-core.html>.
- [22] D. Kudithipudi, Q. Qu, and A. K. Coskun. 2013. *Thermal Management in Many Core Systems*. Springer, Berlin, 161–185.
- [23] Rakesh Kumar, Timothy G. Mattson, Gilles Pokam, and Rob Van Der Wijngaart. 2011. *The Case for Message Passing on Many-Core Chips*. Springer, New York, NY, 115–123.
- [24] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. 2013. The McPAT framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing. *ACM Trans. Archit. Code Optim.* 10, 1, Article 5 (Apr. 2013), 29 pages.
- [25] A. Listl, D. Mueller-Gritschneider, F. Kluge, and U. Schlichtmann. 2018. Emulation of an ASIC power, temperature and aging monitor system for FPGA prototyping. In *Proceedings of the IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS'18)*. 220–225.
- [26] S. Lopez-Buedo and E. Boemo. 2004. Making visible the thermal behaviour of embedded microprocessors on FPGAs: A progress report. In *Proceedings of the ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays (FPGA'04)*. Association for Computing Machinery, New York, NY, 79–86.
- [27] S. Lopez-Buedo, J. Garrido, and E. Boemo. 2000. Thermal testing on reconfigurable computers. *IEEE Design Test Comput.* 17, 1 (2000), 84–91.
- [28] S. Lopez-Buedo, J. Garrido, and E. I. Boemo. 2002. Dynamically inserting, operating, and eliminating thermal sensors of FPGA-based systems. *IEEE Trans. Comp. Packag. Technol.* 25, 4 (2002), 561–566.
- [29] P. Mantovani, E. G. Cota, K. Tien, C. Pilato, G. Di Guglielmo, K. Shepard, and L. P. Carloni. 2016. An FPGA-Based infrastructure for fine-grained DVFS analysis in high-performance embedded systems. In *Proceedings of the 53rd Annual Design Automation Conference (DAC'16)*. Association for Computing Machinery, New York, NY, Article 157, 6 pages.
- [30] M. Mettler, D. Mueller-Gritschneider, and U. Schlichtmann. 2021. A distributed hardware monitoring system for run-time verification on multi-tile MPSoCs. *ACM Trans. Archit. Code Optim.* 18, 1, Article 8 (Dec. 2021), 25 pages.
- [31] S. Mondal, R. Mukherjee, and S. O. Memik. 2006. Fine-grain thermal profiling and sensor insertion for FPGAs. In *Proceedings of the IEEE International Symposium on Circuits and Systems*. 4387–4390.
- [32] S. Mondal, R. Mukherjee, and S. O. Memik. 2006. Fine-grain thermal profiling and sensor insertion for FPGAs. In *Proceedings of the IEEE International Symposium on Circuits and Systems*. 4 pp.–4390.
- [33] R. Mukherjee, S. Mondal, and S. O. Memik. 2006. Thermal sensor allocation and placement for reconfigurable systems. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*. 437–442.
- [34] D. Olsen and I. Anagnostopoulos. 2017. Performance-Aware resource management of multi-threaded applications on many-core systems. In *Proceedings of the on Great Lakes Symposium on VLSI (GLSVLSI'17)*. Association for Computing Machinery, New York, NY, 119–124. <https://doi.org/10.1145/3060403.3060426>
- [35] S. Pagani, J. Chen, M. Shafique, and J. Henkel. 2015. MatEx: Efficient transient and peak temperature computation for compact thermal models. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'15)*. 1515–1520.
- [36] S. Pagani, H. Khdr, J.-J. Chen, M. Shafique, M. Li, and J. Henkel. 2017. Thermal safe power (TSP): Efficient power budgeting for heterogeneous manycore systems in dark silicon. *IEEE Trans. Comput.* 66, 1 (2017), 147–162.
- [37] S. Park, J. Park, D. Shin, Y. Wang, Q. Xie, M. Pedram, and N. Chang. 2013. Accurate modeling of the delay and energy overhead of dynamic voltage and frequency scaling in modern microprocessors. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 32, 5 (2013), 695–708.
- [38] A. Pathania and J. Henkel. 2019. HotSniper: Sniper-Based toolchain for many-core thermal simulations in open systems. *IEEE Embed. Syst. Lett.* 11, 2 (2019), 54–57.
- [39] S. Penolazzi. 2011. *A system-level framework for energy and performance estimation in system-on-chip architectures*. Ph.D. Dissertation. KTH Royal Institute of Technology.
- [40] M. Rapp, M. Sagi, A. Pathania, A. Herkersdorf, and J. Henkel. 2019. Power- and cache-aware task mapping with dynamic power budgeting for many-cores. *IEEE Trans. Comput.* 69, 1 (2019), 1–13.
- [41] S. Roloff, A. Pöpl, T. Schwarzer, S. Wildermann, M. Bader, M. Glaß, F. Hannig, and J. Teich. 2016. ActorX10: An actor library for X10. In *Proceedings of the 6th ACM SIGPLAN Workshop on X10 (X10'16)*. Association for Computing Machinery, New York, NY, 24–29.
- [42] A. Rudi, A. Bartolini, A. Lodi, and L. Benini. 2014. Optimum: Thermal-aware task allocation for heterogeneous many-core devices. In *Proceedings of the International Conference on High Performance Computing Simulation (HPCS'14)*. 82–87. <https://doi.org/10.1109/HPCSIm.2014.6903672>
- [43] MohammadSadeqh Sadri, Andrea Bartolini, and Luca Benini. 2011. Single-chip cloud computer thermal model. In *Proceedings of the 17th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC'11)*. IEEE, 1–6.
- [44] C. Sakalis, C. Leonardsson, S. Kaxiras, and A. Ros. 2016. Splash-3: A properly synchronized benchmark suite for contemporary research. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'16)*. 101–111.

- [45] D. Sanchez and C. Kozyrakis. 2013. ZSim: Fast and accurate microarchitectural simulation of thousand-core systems *Proceedings of the 40th International Symposium on Computer Architecture (ISCA'13)*. Association for Computing Machinery, New York, NY, 475–486.
- [46] V. Saraswat, B. Bloom, I. Peshansky, O. Tardieu, D. Grove, A. Shinnar, M. Takeuchi, et al. 2014. X10 Language Specification. <http://x10.sourceforge.net/documentation/languagespec/x10-latest.pdf>.
- [47] S. Velusamy, Wei Huang, J. Lach, M. Stan, and K. Skadron. 2005. Monitoring temperature in FPGA-based SoCs. In *Proceedings of the International Conference on Computer Design*. 634–637.
- [48] R. Zhang, M. R. Stan, and K. Skadron. 2015. Hotspot 6.0: Validation, acceleration, and extension. Technical Report, University of Virginia.

Received June 2021; revised December 2021; accepted February 2022