Formal Probabilistic Analysis of Distributed Dynamic Thermal Management

Shafaq Iqtedar*, Osman Hasan*, Muhammad Shafique†, Jörg Henkel†
*School of EE and CS, National University of Sciences and Technology (NUST), Islamabad, Pakistan
†Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany
Email: {10besesiqtedar, osman.hasan}@seecs.nust.edu.pk; {muhammad.shafique, henkel}@kit.edu

Abstract—The prevalence of Dynamic Thermal Management (DTM) schemes coupled with demands for high reliability motivates the rigorous verification and testing of these schemes before deployment. Conventionally, these schemes are analyzed using either simulations or by running on real systems. But these traditional analysis techniques cannot exhaustively validate the distributed DTM schemes and thus compromise on the accuracy of the analysis results. Moreover, the randomness due to task assignments, task completion times and re-mappings, is often ignored in the analysis of distributed DTM schemes. We propose to overcome both of these limitations by using probabilistic model checking, which is a formal method for modeling and verifying concurrent systems with randomized behaviors. The paper presents a case study on the formal verification of a state-of-the-art distributed DTM scheme using the PRISM model checker.

I. INTRODUCTION

With fabrication technologies entering the nano era, elevated temperatures resulting from increased on-chip power densities are jeopardizing the reliability and performance of on-chip systems. Dynamic Thermal Management (DTM) schemes [6] have been introduced to control these elevated temperatures. Conventionally, DTM schemes were based on centralized approaches [8]. But with the prevalence of many-core architectures, the overhead of monitoring the entire chip has significantly been increased. Therefore, distributed DTM schemes came forth to tackle the complexity and scalability issues by transforming the problem space from a global one to many smaller regional ones, thus exploiting locality when making DTM decisions. However, for a distributed DTM (dDTM) scheme to achieve the same performance as is possible from a global DTM scheme, an exchange of state information across regions becomes necessary in order to negotiate a near optimal system state configuration [5].

The widespread usage of DTM schemes [6] along with high demands for reliability require the rigorous verification of these schemes before deployment. Traditionally, these schemes are analyzed using either simulations or by running on real hardware systems. However, simulation based verification is not complete in terms of error detection. A simulator does not exhaustively traverse through all possible system paths. Even if some corner cases can be specifically targeted, there is no proof that these represent all the scenarios in which the DTM scheme would not be able to achieve a stable system configuration, and it is never possible to consider or even foresee all corner cases.

Moreover, the workloads distribution and in some cases the initial task assignment across the many-core system is random [5]. This randomness in the behavior, due to task assignments, task completion times and re-mappings, is often ignored in the analysis of DTM schemes. This fact compromises the analysis results as well by ignoring certain real time scenarios in which a DTM scheme may not be able to provide thermal balancing, resulting into the creation of thermal hotspots. These limitations in the analysis may in turn lead to delays in deployment of DTM schemes as happened in the case of Foxton DTM that was designed for the Montecito chip [1].

To overcome the above-mentioned limitations, model checking [7] can be used in the analysis of DTM schemes. Model checking is based on models describing the given systems behavior in a mathematically precise and unambiguous manner. A model checker, exhaustively and automatically checks whether a model meets the given specification while ensuring 100% completeness of analysis. Given the safety-critical nature of DTM schemes, a traditional model checking technique has also been recently proposed to verify dDTM schemes in [3]. However, traditional model checking techniques provide answers to simple non-probabilistic (Yes/No) functional properties. For instance, in case of a dDTM, conventional formal analyses can verify if the thermal hotspots can be created or not. However, an important piece of knowledge is about how often this can happen and in case if a thermal hotspot is created, it is important for a system designer to evaluate the duration that the DTM algorithm would take to balance the temperatures across the chip. These statistics can play a vital role in the design of the next generation DTM schemes for many-core systems. To the best of our knowledge, no formal probabilistic verification method, including detailed quantitative analysis while covering the randomized behaviors, has been used in the context of verifying dDTM schemes for many-core systems so far.

We propose to formally verify the above-mentioned probabilistic properties for dDTM schemes by using probabilistic model checking. In order to illustrate the effectiveness of our approach, we present a formal analysis of a state-of-the-art agent-based dDTM scheme, namely Thermal-aware Agent-based Power Economy (TAPE) [5]. Our verification process is based on a widely used model checking tool, i.e., the PRISM model checker [2]. We ascertain the correctness of TAPE by constructing its Markovian model and analyzing the formally specified quantitative properties using PRISM. Moreover, we have also performed a verification feasibility analysis of TAPE DTM scheme. One of the main reasons for using TAPE as our case study is the ability to compare our results with the recent state-of-the-art formal verification technique, that
uses traditional model checking, for a fair comparison and
technology advancement [3]. Our results show the usefulness
of probabilistic model checking for analyzing dDTM schemes,
as a number of critical issues with TAPE have been identified
in this paper, which could not have been caught in [3].

II. PROBABILISTIC MODEL CHECKING AND PRISM

Probabilistic model checking [4] is a formal method for
verifying systems that exhibit randomized behaviors. A model
checker exhaustively traverses the entire state-space of a design
in ascertaining correctness. The 100% completeness of analysis
coupled with the consideration of randomized behaviors and
provision of a detailed quantitative knowledge makes probabilistic model checking a very powerful framework for
verifying dDTM schemes. One limitation of model checking is
that the state-space of a system can be very large, or sometimes
even infinite. This problem, termed as state-space explosion, is
usually resolved by developing abstract, less complex, models of the system.

PRISM [2] is a widely used probabilistic model checker. It
provides support for analyzing different kind of probabilistic
models, such as discrete-time Markov chains (DTMCs) [4], in
which time is modeled as discrete steps. Models are specified
using the PRISM modeling language, i.e., a simple, state-
based language based on the reactive modules formalism [2].

Distributed DTM schemes can be described in PRISM as the
parallel composition of a set of modules that are running over
different cores. The state of each module can be given by
a set of finite-ranging variables and its behavior by a set of
probabilistic guarded commands. PRISM also allows models
to be augmented with costs and rewards, i.e., the real values
assigned to states and transitions of the model. This permits
reasoning about a much wider range of quantitative measures
of a system, e.g., “completion time” or “energy consumption”.
Such statistics are very useful in the analysis of dDTM
schemes. For instance, it allows the designer to evaluate the
efficiency of dDTM scheme without the overhead of modeling
time separately. We chose PRISM for the proposed work since
it allows us to calculate actual probabilities, rewards based
timing properties and supports a wide variety of probabilistic
models. Thus, we can evaluate both the effectiveness and the
efficiency of dDTM schemes using PRISM.

III. TAPE DISTRIBUTED DTM SCHEME

TAPE [5] is a fully distributed agent-based DTM approach,
which focuses on distributing power and the resulting heat
evenly over a many-core architecture. The approach incorporates thermal penalties in the agent negotiations to deal pro-
avtively with potentially developing thermal hotspots explicitly.
Each core is assigned its own agent, which individually dis-
tributes power units between tiles of a many-core architecture
through local trade, i.e., only with its neighbors. Initially, a
predefined number of power units are distributed evenly among
all agents. Each of these power units belong to one of the
following two categories: used units are currently deployed
by the Processing Element (PE) to run tasks and free units
are power units available on a tile but not currently needed
by the tile to run its allocated tasks. These units are used by
each agent to calculate the buy and sell values for trading
power units. These buy and sell values vary depending on the
supply/demand of power units. At start-up, a tile is randomly
chosen to map a task as all tiles are equally suited to run the
task. Otherwise the task is mapped to a tile with most available
units (In TAPE it is the tile with maximum \(sell_{T_n} - buy_{T_n}\)
value). At each time interval, the agent \(n\) first calculates its
own buy/sell values and then compares them with the buy/sell
values of its neighbors as follows:

\[
(sell_n - buy_n) - (sell_i - buy_i) < T_n
\]  

(1)

If any of the neighbors \(i\) does not fulfill the above equation,
then the agent \(n\) relinquishes a power unit to the neighbor with the
maximum \(sell_i - buy_i\) value. If the agents have free power units
then this is simply a matter of decrementing its number
of free power units otherwise, the agent must give up one of
its used power units and re-mapping needs to be invoked.

The most important challenge in analysis of TAPE DTM
scheme is to ensure that the system will always attain a stable
power budget distribution without any redundant trading (i.e.,
two agents continuously trading one free unit back and forth
preventing the system from stabilizing). Moreover, even if a
stable power budget distribution is attained, it is important to
verify that no thermal hotspots are created and temperatures
are distributed evenly across the chip. Another major chal-
gen is to determine the optimal values of unknown weight
parameters (i.e., \(a, b, w_{u,s}, w_{f,s}, w_{u,b}, w_{f,b}\)) in TAPE, as
the performance of the algorithm depends highly on the values
of these parameters. We have successfully addressed all the above
mentioned challenges in this paper.

IV. FORMAL MODELLING OF TAPE IN PRISM

1) Modeling distributed agents and the (re-) mapping
software: The first step in our verification is to describe
the behavior of the TAPE algorithm running on a many-core chip
as a DTMC. The grid is modeled as a two dimensional array
of distributed nodes such that the TAPE algorithm runs on
all these nodes concurrently. We developed a generic node
model so that it can be repeatedly used to formally express
any grid of arbitrary dimension. Using our approach, the agents
(associated with the nodes) are realized as separate modules
[5] executing in parallel. Every agent module is associated with
the following set of variables: \(free_n, used_n, Buybase_n, sellBase_n, buy\_T_n, sell\_T_n\), where, \(n\) represents the
number of the corresponding core. As mentioned in [5], the
software entity that is responsible for application (re-)mapping
may be executed in any of the existing processing elements
inside the many-core architecture. Therefore for the selection
of the mapping-core, a separate module is constructed. This
module selects a core on the basis of following two conditions:

1) If all the tiles are equally suited for mapping the new
task then one tile is randomly chosen among all the
tiles with probability equal to one divided by the total
number of cores (Line 1 Algorithm 1).

2) Otherwise, the task is (re-)mapped to a tile where most
free units are available and this is the tile with the
maximum \(sell_{T_n} - buy_{T_n}\) value. Again if more than
one tile fulfills this criterion, then one of them is chosen randomly (Line 4 Algorithm 1).

Thus, whenever a new task is initiated, this module selects a tile and invokes the corresponding agent module of that tile. The new tasks are initiated in a separate module, which randomly selects a task execution time and just sends a signal to the mapping software entity. Once a core is selected for mapping, its corresponding agent is invoked which is a separate module and it increments the used units and decrements the free units, depending on the task execution time. The higher the value of task-time, the more power units will be consumed and it is assumed that only 1 power unit is deployed for executing a task, the free power unit is converted into a used power unit and the temperature for mapping, its corresponding agent is invoked which is a separate module and it increments the used units and decrements the free units, depending on the task execution time. The higher the value of task-time, the more power units will be consumed and it is assumed that only 1 power unit is deployed for executing a task, the free power unit is converted into a used power unit and the temperature.

2) Discretization of Temperature Variable: Whenever a free power unit is deployed for executing a task, the free unit is converted into a used power unit and the temperature variable (Tm) of the corresponding tile is incremented by $4^\circ K$ (as done in [3]). At start-up, an initial value of $30^\circ K$ is assigned to the temperature variable.

3) Reward-Structure for Modeling Time: For the verification of timing properties, we extend the probabilistic model with rewards in order to compute the time-based properties. In every execution of the algorithm, the reward is accumulated and the final value of the accumulated reward determines the number of steps in attaining a stable system configuration. This is a major advantage of using PRISM as it removes the overhead of modeling time separately, as was done with Lamport time stamps in [3].

4) Modeling Agent based negotiations: As discussed earlier that in order to avoid the creation of thermal hotspots, agent based negotiations take place and the power units are exchanged among the agents. PRISM does not provide a mechanism to pass messages between different modules. Therefore, in order to implement the agent-based negotiations, we developed a separate module to perform the exchange of information. At every trade interval, this module compares the modified buy/sell values of all nodes with the buy/sell values of their neighboring nodes and ensures that the stability equation is satisfied. If the equation is not satisfied and the differences among the buy and sell values goes beyond the threshold, this module sends a signal for the trading of power units to the respective increment-and-decrement agents. For every node, there is a power trading agent as well (see Algorithm 2) for the exchange of power units. Whenever a signal is received, one agent relinquishes a power unit to the other agent. These power trading agents are realized in separate modules from mapping-agent’s modules (as mentioned in TAPE DTM scheme [5]). The agents will keep trading the power units until the stability equation is satisfied again. Moreover, if during the power-trade, re-mapping is invoked the power trading agent sends a signal to the corresponding (re-) mapping module.

5) Characteristics of weights ($a_s$, $a_b$, $w_{u,s}$, $w_{f,s}$, $w_{u,b}$, $w_{f,b}$): An important factor in ensuring that the system reaches a stable state in the TAPE DTM technique is the choice of weights used to determine the buy and sell values. These weights are constant values with respect to time. A challenge in the analysis of TAPE algorithm was to determine the optimal values of these parameters. Our proposed probabilistic analysis allowed us to come up with some interesting results during the calculation of optimal values for these parameters. For instance, we discovered certain combinations of these parametric values for which the TAPE algorithm would not be able to attain a stable system configuration from a random initial task mapping. Therefore, these results clearly indicate the shortcomings of ignoring randomness while analyzing dDTM schemes.

Algorithm 1 Mapping Software

```
Module (re-) mapping
1: // (Condition1 = true) ->
2: 1/tiles: (remap_core_no = 11) + 1/tiles: (remap_core_no' = 12)
3: +1/tiles: (remap_core_no = 21) + 1/tiles: (remap_core_no' = 22)
4: // (Condition2 = true & max_diff = diff_11) ->
5: (remap_core_no' = 11) ...
```

Algorithm 2 Power Trading Agent

```
Module power_trading_agent_n
1: // (dec_n = true) & (free_n > 0) ->
2: (free_n' = free_n - 1) & (dec_n' = false);
3: // (dec_n = true) & (free_n <= 0) ->
4: (used_n' = used_n - 1) & (dec_n' = false) &
5: (temp_n' = temp_n - 4) &
6: (remap_tasks' = remap_tasks + 1) & (remap_state' = 0);
7: // (inc_n = true) -> (free_n' = free_n + 1) & (inc_n' = false);
```
the optimal values of weight parameters, and higher when either

Fig. 3. Redundant trading of power units

where \( \text{diff} \_11 \) is the \( \text{sell}_{T_n} - \text{buy}_{T_n} \) value of core number

S is the steady state probability (i.e., probability of being in a state where stability equation holds true at time

T, as T goes to infinity). We also compared the verification times for these properties with the verification times given in [3], as illustrated in Fig.2. The difference in verification times can be a consequence of the higher number of states and transitions stored during model checking in [3]. For instance, using our Markov chain model, PRISM stored only 10800 states and 12601 transitions, whereas 4786929 states and 25244478 transitions were stored using the model checking approach in [3]. In order to ensure that no thermal hotspots are formed, we calculated the probability that in steady state the maximum temperature will be less than the threshold temperature at different values of weight parameters.

**Thermal hotspot:** \( S = ? [\text{MaxTemp} < 62] \)

**Optimal values of parameters and uniform distribution of temperatures:** After many iterative experiments, we discovered the optimal set of parameters and it has been formally verified, using PRISM, that with the discovered combinations of parametric values, the probability of maintaining lower temperatures is very high. Fig.4 represents the probability that temperature will remain within a certain threshold at different values of \( a_s \) and \( a_b \). It was observed that the probability is higher when either \( a_s \approx a_b \) or \( a_s \) is slightly greater than \( a_b \). At the optimal values of weight parameters, a uniform distribution

![Graph a: Effects of \( a_s \)](image)

![Graph b: Effects of \( a_b \)](image)

**Fig. 4. Effects of \( a_s \) and \( a_b \) of temperatures is achieved and no thermal hotspots are created.**

**Back and forth trading:** It was discovered that for some values of \( a_s \) and \( a_b \), the algorithm would grow continuously without attaining a stable state. In reality, the algorithm enters into a never ending loop due to redundant trading (See Fig.3).

The results of our verification shows that the performance of TAPE algorithm depends highly on the values of weight parameters. The above mentioned statistics provide very useful insights for the designer and indicates the usefulness of our approach.

**VI. CONCLUSIONS**

This paper presents a formal verification approach for dDTM schemes based on probabilistic model checking. The rigorous nature of the analysis coupled with the consideration of randomized behaviors of the dDTM schemes allows the identification of design problems that cannot be registered using traditional techniques. For illustration purposes, a successful probabilistic analysis of a state-of-the-art dDTM scheme is presented in this work. The identification of various loopholes in TAPE corroborates the benefits of our approach over other traditional techniques.

**ACKNOWLEDGMENT**

This work is supported in parts by the German Research Foundation (DFG) as part of the priority program “Dependable Embedded Systems” (SPP 1500 - spp1500.itec.kit.edu) and in parts by the DAAD “Deutsch-Pakistanische Forschungs kooperationen” project.

**REFERENCES**


