A VLSI DESIGN PERSPECTIVE ON NON-CENTRALIZED APPROACH TO THERMAL AWARE TASK MIGRATION IN MANY-CORE SYSTEMS

Ershad. S. B and Sreejith. S. Nair
Department of Electronics and Communication Engineering,
Nehru College of Engineering and Research Centre, Pampady, Thrissur, Kerala, India.

ABSTRACT
In deep sub-micrometer era, thermal hot spots, and large temperature gradients significantly impact system reliability, performance, cost, and leakage power. The increasing chip complexity and power density elevate peak temperatures of chip and unbalance the thermal gradients. Raised peak temperatures reduce lifetime of the chip, deteriorate its performance, affect the reliability and increase the cooling cost. As the system complexity increases, it is more and more difficult to perform thermal management in a centralized manner because of state explosion and the overhead of monitoring the entire chip. A proposal of framework for distributed thermal management in many-core systems is presented where balanced thermal profile can be achieved by proactive task migration among neighbouring cores. The framework has a low cost agent residing in each core that observes the local workload and temperature and communicates with its nearest neighbour for task migration and exchange. By choosing only those migration requests that will result in balanced workload without generating thermal emergency, the proposed framework maintains workload balance across the system and avoids unnecessary migration. This paper shows light on VLSI design areas of the framework as different from previous papers and details on different cases possible with the design. The results section is enriched with simulation results and shows a gradual development from single core to multi-core and then to many-core system. The design perspective presented over here is very simple and helps to understand on development of a logical design from available theoretical details. The paper also presents maximum conditions that could possibly be shown in the simulation of a 9-core system.

KEYWORDS: *Distributed control, prediction, task migration, thermal management.*

I. INTRODUCTION

1. 1. Background

The Multiprocessor System-on-Chip (MPSoC) is becoming a major system design platform for general purpose and real-time applications, due to its advantages in low design cost and high performance. With the scaling of CMOS devices, this technology is progressing from the multi-core era to the many-core era. With the unprecedented number of transistors integrated on a single chip, the current multi-core technology may soon progress to hundreds or thousands of cores era [3]. Examples of such system are the 80-tile network-on-chip that has been fabricated and tested by Intel [28] and Tilera’s 64 core TILE64 processor [1]. While the multi-core or many-core technology delivers extraordinary performance, they have to face the significant power and thermal challenges. The increasing chip complexity and power density elevate peak temperatures of chip and unbalance the thermal gradients. Raised peak temperatures reduce lifetime of the chip, deteriorate its performance, affect the reliability [27] and increase the cooling cost. Dynamic thermal management (DTM) approaches such as core throttling or stalling which are widely used in today’s computer systems usually have negative impact on the performance. The adverse positive feedback between leakage power and raised temperature creates the potential of thermal runaway [27]. When mapped on a many-core system, diverse workload of applications may lead to power and temperature imbalance among different cores. Such temporal and spatial variations in temperature create local temperature maxima on the chip called the hotspot [11], [27]. An excessive spatial temperature variation, which is also referred to as the thermal gradients, increases clock skews and decreases performance and reliability [35].

1. 2. Present Scenario

Many dynamic thermal management techniques such as clock gating, dynamic voltage and frequency scaling, thread migration have been proposed for multi-core systems. All these techniques aim to ensure the system running under a fixed safe temperature constraint [8], [9], [16], [19], [21], [22], [26], [29]. Most of these existing techniques are centralized approaches. They require a controller that monitors the temperature and workload distribution of each core on the entire chip and make global decisions of resource allocation. Such centralized approaches do not have good scalability. First of all, as the number of processing elements grows, the complexity of solving the resource management problem grows exponentially. Second, a centralized resource management unit that monitors the status and issues DTM commands to each core generates a huge communication overhead in many-core architecture, as communication between the central controller and cores will increase exponentially with the number of cores [12]. Such overhead will eventually affect the speed of data communication among user programs and also consume more power on the interconnect network. Finally, as the size and the complexity of the many-core system increase the communication latency between the central controller and the cores increases, this leads to a delayed response and sub-optimal control.

1. 3. Proposed Framework

A framework of distributed thermal management where balanced thermal profile can be achieved by proactive thermal throttling as well as thermal-aware task migrations among neighbouring cores is proposed in [35]. The framework has a low cost agent residing in each processing element (PE). The agent observes the workload and temperature of the PE while exchanging tasks with its nearest neighbours through negotiation and communication. The goal of the proposed task migration is to match the PE‟s heat removal capability to its workload (i.e., the average power consumption) and at the same time create a good mix of high power (i.e., “hot”) tasks and low power (i.e., “cool”) tasks running on it. As each agent monitors only the local PE and communicates with its nearest neighbours, the proposed framework achieves much better scalability than the centralized approach. The proposed technique is referred to as distributed thermal balancing migration (DTB-M) as it aims at balancing the workload and temperature of the processors simultaneously. A steady-state temperature-based migration (SSTM) scheme as well as a temperature prediction-based migration (TPM) scheme is proposed. The first migration scheme considers the long term thermal behaviour of tasks and distributes tasks to PE’s based on their different heat removal capabilities. The second migration scheme predicts the thermal impact of different workload combinations and adjusts the task allocation in a neighbourhood so that all the PE’s get a good mixture of hot tasks and cool tasks. The two migration schemes are complementary to each other with the first considers long term average thermal effect and the second considers short term temporal thermal variations. Both SSTM and TPM methods are proactive migration schemes. Together they provide progressive improvement that reduces thermal gradients and prevents thermal throttling events. As part of the thermal management agent, a neural network-based temperature predictor is also proposed. It predicts the future peak temperature based on the workload statistics of the local PE and some preliminary information from the neighbouring PE’s. Comparing to the temperature predictors proposed in previous works [9], [31], proposed neural network predictor in [36] has several advantages. First of all, it only has to be trained once and after that the recall process has very low computation complexity. Second, because it takes the workload information as one of the input parameters, it can give accurate prediction right after task migration. This is the major difference between proposed prediction model and the previous prediction models [9] and [31] which need an online adaptation phase when workload changes. Finally, proposed model can be used to predict the temperature impact of a migration before the migration physically takes place, as long as the power consumption of the task to be migrated in or out is known. Therefore, the predictor is used not only to determine when to trigger a proactive task migration but also to evaluate whether a migration is beneficial.

The rest of this paper is organized as follows. Section-2 reviews the previous work. Section-3 gives the semantics of the underlying many-core system and the application model. We give an overview of our thermal management policy, with the details on prediction model and migration schemes in Section-4. Experimental results are reported in Section-5. Finally, we conclude this paper in Section-6. Also the future scope of development is presented in Section-7.

II. RELATED WORK

Modern day microprocessors handle thermal emergencies through various DTM mechanisms. Power density continues to increase exponentially with each new technology generation, posing a major challenge for thermal management in modern processors. For thermal purposes, choosing a distributed policy may be well worth the necessary added design complexity [11].

The increasing processing capability of Multi-Processor Systems on- Chips (MPSoCs) is leading to an increase in chip power dissipation, which in turn leads to significant increase in chip temperature. At system level, voltage/frequency scaling, task scheduling, task allocation and thread migration can be combined to leverage the temperature reduction on MPSoCs. Frequency assignment was formulated as a convex optimization problem and optimum solutions can be solved offline [22]. The thermal behavior of an MPSoC was modeled as a control theory problem which enables the design of an optimum frequency controller without depending on the thermal profile of the chip [32].

A light weight probability based scheduling method which could achieve better temporal and spatial thermal profile was proposed in [9]. The benefits of thermally aware task scheduling for multiprocessor systems-on-a-chip were also explored.

An algorithm was proposed in [33] to map and schedule tasks based on the thermal conductivity of different processors. The highlight of the algorithm were to minimize the time of heat staying on the chip and the resultant adverse impact; to avoid detrimentally heating up the cores located on the heat conduction path; to reduce the temporal temperature fluctuation and improve the reliability; to maximize the performance by reducing the thermal management event.

Temperature has become an important constraint in high-performance processors, especially multi-cores. Thread migration will be essential to exploit the full potential of future thermally constrained multi-cores. High temperature is mostly a consequence of high power density. Power density has increased relentlessly over technology generations and will probably continue to increase. A clock gating and thread migration based method was proposed in [19] which maximize system performance and minimize the number of migrations while maintaining the temperature under a desired constraint and guaranteeing fairness between threads.

If some (combinations of) workloads consume more power than the TDP, the processor is dynamically throttled to keep the chip temperature below a safe threshold. This mechanism is called dynamic thermal management (DTM). The throughput of an MPSoC system under a maximum temperature constraint was explained in [24], and derived an approximate analytic expression of system throughput depend on several parameters of interest.

Thermal management of on-chip interconnect network is addressed in [25]; first proposed an architecture thermal model for on-chip networks. Based on this model, they further proposed Thermal Herd, a framework which uses distributed thermal throttling and thermal aware routing to tackle thermal emergencies. Chip power consumption is expected to increase with each new generation of computers while the geometries continue to shrink, resulting in higher power densities. High power densities cause thermal hot spots and large temperature variations on the die and introduce a number of significant challenges, including higher cooling costs, increase in leakage power, lower device performance and reliability degradation.

Proactive temperature balancing (PTB) technique allocates incoming jobs to cores to minimize and balance the temperature on the die. In multithreaded systems, PTB performs balancing by first moving threads that are currently waiting, as opposed to stalling the executing threads. A proactive temperature management approach for MPSoCs, which can adapt to changes in system dynamics at runtime, had been presented in [9].

Predictive Dynamic Thermal Management (PDTM) was proposed in the context of multi-core systems in [31]. PDTM scheme utilizes an advanced future temperature prediction model for each core to estimate the thermal behavior considering both core temperature and applications temperature variations and take appropriate measures to avoid thermal emergencies. Unlike the prediction model proposed in [9] and [31], neural network-based prediction model can overcome the limitations mentioned previously. The proposed model does not rely on the history temperature. Instead it reveals the relation between temperature and workload. It is trained offline; and does not need an online adaption phase. As the model is trained separately for each core on the chip, it inherently takes into account the core location and heat dissipation ability.

III. SYSTEM INFRASTRUCTURE

A tile-based network-on-chip (NoC) architecture is targeted here. Each tile is a processor with dedicated memory and an embedded router. It is also referred to as core or PE. All the processors and routers are connected by an on-chip network where information is communicated via packet transmission. The cores that can reach to each other via one-hop communication are referred to as the nearest neighbors. The proposed DTB-M algorithm moves tasks among nearest neighbors in order to reduce overhead and minimize the impact on the communication bandwidth. In a NoC, the latency and energy for transferring a data packet from one PE to another is proportional to the number of hops along the path. If the congestions are considered, this relation could be super linear due to the buffering overhead at each router. Limiting the communication to nearest neighbors cuts the communication cost (including both latency and energy) by reducing the communication distances and eliminating congestions. An existence of temperature sensor on each core is assumed. A temperature sensor can be a simple diode with reasonably fast and accurate response.
A dedicated OS layer is assumed to be running on each core that provides functions for scheduling, resource management as well as communication with other cores. This is a trend pointed out by some literatures in OS research for many-core and NoC systems. Examples of such system are Intel's single-chip cloud computing (SCC) platform and research accelerator for multiple processors (RAMP). The proposed DTB-M algorithm is implemented as part of the OS-based resource management program which performs thermal-aware task migration. Each core is assumed to be a preemptive time-sharing/multitasking system. The focus is on batch processing mode, where pending processes/tasks are enqueued and scheduled by the agent. Each task occupies an equal slice of operating time. Between two execution slices is the scheduling interval in which the agent performs the proposed DTB-M algorithm and the OS switches from one task to another. The scheduling intervals of different cores do not have to be synchronized. Because the context switch overhead is very small compare to the execution interval (e.g., in Linux), and algorithm has very low overhead, the duration of the scheduling interval is assumed to be negligible comparing to that of the execution interval.

IV. DISTRIBUTED THERMAL BALANCING POLICY

4. 1. Master-Slave Communication

Each PE_i is a preemptive system and has a set of tasks LT_i. Each task occupies an equal slice of execution time t_{slice}. Between two execution intervals is the scheduling interval. The DTB-M policy is performed in scheduling interval. The PE also switches from one task to the next task at the scheduling interval. It is assumed that each task in LT_i will be running for a relatively long period of time and its power consumption has been profiled or can be estimated. It was reported that more than 95% accuracy can be achieved in power estimation using information provided by performance counters that are available in many modern processors. The power consumptions of all tasks in LT_i are referred as the "workload" of PE_i and different combinations of tasks in LT_i are referred as different "workload patterns" of PE_i. Both the information can easily be observed by OS.

Table 1. List of Symbols and Their Definitions

Symbol	Definition
LT_i	The list of tasks running on core *i*
$\|LT_i\|$	The number of tasks running on core *i*
$\tau_{i,}$	A task in LT_i
$P\tau_i$	The power of $\tau_{i,}$
T_i	Current temperature of core *i*
N_i	The set of nearest neighbors of core *i*
T_m	Temperature threshold to trigger the DTB-M algorithm
T_{diff}	Threshold to trigger thermal balancing
nt_{diff}	Threshold to trigger workload balancing
t_{slice}	Execution interval

The DTB-M policy basically can be divided into 3 phases: temperature checking and prediction, information exchange and task migration. Figure 1 shows the flowchart of the DTB-M execution in the ith core. A DTB-M agent could be in either master mode or slave mode. A DTB-M master initiates a task migration request while the DTB-M slave responds to a task migration request. A DTB-M agent is in slave mode by default. It will enter the master mode if and only if any of the following three scenarios are true. First of all when the local temperature T_i reaches a threshold Tm in the last execution interval and in this case, hotspots are generated, and the DTB-M agent will first throttle the processor to let it cool down before continue to execute. Second of all, the predicted future peak temperature T_m exceeds the threshold and the current peak temperature is larger than T_m-µ, where µ is a temperature margin and then actions are not taken unless the difference between the current peak temperature and the threshold is less than the margin. Final condition is that when the temperature difference between the local core and the neighbor core exceeds the thermal balancing threshold T_{diff}. Any of the above three scenarios could cause adverse effects. The first two scenarios indicate (potential) hotspots generation while the last scenario indicates high thermal gradients. Therefore, a task migration request will be initiated. A DTB-M master sends task migration requests to its nearest neighbors. Because the scheduling intervals in all processors are not synchronized, the requests are not likely to be checked and responded by the slave agents right away. On the other hand, because all cores adopt the same execution and scheduling interval, it is guaranteed that all slave agents will respond within one t_{slice} after the requests are issued.

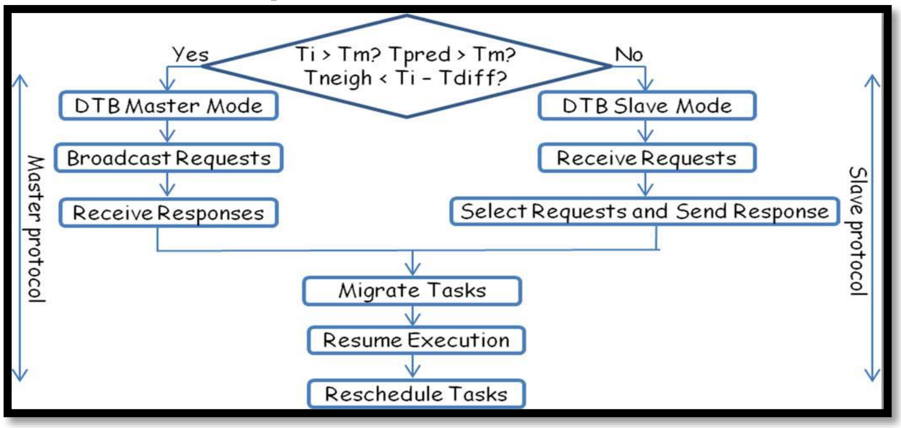


Figure 1. Master-slave Execution Protocol

The asynchronous communication between master and slave agents is explained by the example shown in figure 2. It shows a complete execution cycle of DTB-M policy starting from condition check phase to task migration. When an agent first enters its scheduling interval and becomes a master, it broadcasts a migration request in its neighborhood and then continues task execution. The slave will not respond until it reaches the next scheduling interval, when it checks its message queue for incoming requests. If there is no request, the PE resumes normal execution in next time slice. In case of multiple master requests, the slave selects a master which has the highest average power consumption. Response to this master PE includes its ID, details of slave's workload, its recent operating temperature etc. The slave is then locked to this master until it is released by the master. After receiving the response, the master decides which tasks to migrate during its next scheduling interval and sends the migration command to slave. The tasks are migrated from master to slave at this time. After sending a response, the slave ignores any possible incoming requests from other master agents until it receives the migration command from the original master. Tasks can be migrated from slave to master at this time, which marks the end of DTB-M. To make migration decisions, a master DTB agent considers both loads balancing as well as thermal balancing. First, a load balancing process is triggered which migrates tasks one way to balance the workload between the master and the slave if the workload difference between them exceeds the threshold nt_{diff}, which is measured by $||LT_i|- |LT_j|| > nt_{diff}$, j € N_i. If there is no workload imbalance, then the thermal balancing process is triggered.

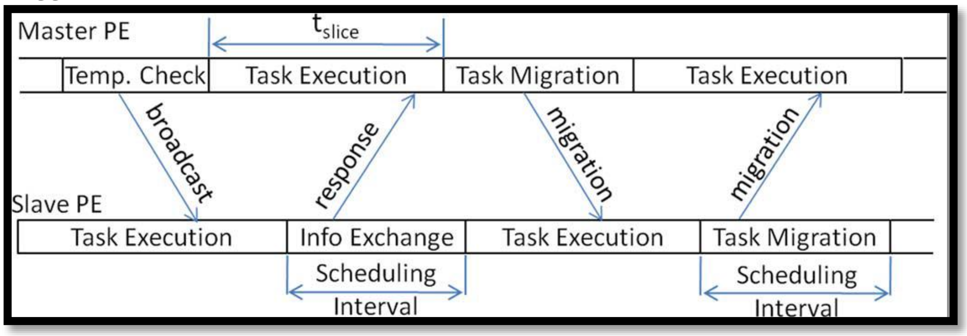


Figure 2. Master-slave Communication

The main idea of the DTB-M policy is to exchange tasks between neighboring PE's, so that each PE can get a set of tasks that produces fewer hotspots. The DTB-M policy is composed of two techniques. Both of the techniques have quadratic complexities to the number of tasks in the local task queue. The first technique is a SSTM. It distributes tasks to cores based on their different heat dissipation abilities. The second technique is a TPM, which relies on predicted peak temperatures of different task combinations to make migration decisions. It ensures that each core can get a good mixture of high power and low power tasks without having thermal emergency. The two techniques are complementary to each other with the SSTM focuses on long term average thermal effect and the TPM focuses on short term temporal variations. The main computation of the SSTM is performed by the masters while the main computation of the TPM is performed by the slaves. The DTB-M agent is not a separate task but resides in the kernel code. For example, it can be integrated with the Linux task scheduler, which will be called each time when a task finishes its current time slice and gives up the CPU.

4. 2. Temperature Prediction Model

Instead of projecting the future temperature based on a sequence of history temperatures, the peak temperature of a processor is modeled as a function of a set of features collected from local processor and its neighboring processors within a history window, and approximates this function using a neural network. The feature set includes not only the temperature information but also the workload information. Because the relation between temperature and workload is relatively stable when the layout and packaging style of the chip is given, the neural network needs to be trained only once. The peak temperature predictor will be used in the temperature checking/prediction phase to determine if a master mode DTB-M will be triggered and also in the information exchange phase to find out if a TPM migration is beneficial or not. Therefore, it should not only give accurate peak temperature estimation when the PE continues the current workload pattern, but also project the temperature change before dramatic workload changes.

Temperature prediction in a timesharing or multitasking system is challenging. Different tasks have different power consumptions and therefore display different thermal characteristics. When running the combination of these tasks, the temperature of a PE would oscillate rapidly, making accurate temperature prediction difficult. Fortunately, it is observed that the local peak temperature for a given set of tasks changes much slower compared to the instantaneous temperature. Because a high peak temperature causes the thermal emergency, the PE's peak temperature in the near future, given the set of tasks (i.e., the workload pattern) on the processor, is predicted. The neural network model for the peak temperature prediction could be used. Neural network has been widely used in pattern recognition and data classification because of their remarkable ability to extract patterns and detect trends through complex or imprecise data. It is composed of a number of interconnected processing elements (i.e., neurons) working together to solve a specific problem. A neural network model can be trained through a standard learning process. After the training process, the model can be used to provide projections on the new data of interest. It is important to point out that the neural network model is based on an assumption that the peak temperature of a core is a deterministic function of all the features aforementioned plus some white noise. A training set that covers all possible feature settings will yield the best model. Therefore, the longer training set gives better training quality. However, it also increases the training time.

4. 3. SSTM and TPM Policies

Due to variant heat dissipation abilities, a task running on different processors have different steady state temperatures. The steady-state temperature-based migration (SSTM) policy balances high power tasks and low power tasks among neighbor PE's to lower the average steady state temperature of the whole chip. It considers the lateral heat transfer between neighbor PE's and their different heat dissipation capabilities. The SSTM reduces the average steady state temperature of the whole chip. Although very effective, it has several limitations. First, it is possible that the SSTM moves all high power tasks in a neighborhood to one core whose thermal conductance value is the minimum. Furthermore, if the thermal conductance value of a core is less than that of all its neighbors, then using SSTM policy the core will not be able to exchange its high power task with a low power task in its neighborhood when it is overheated because this will increase the average steady state temperature of the chip. To escape from the above mentioned situation, temperature prediction-based migration (TPM) is proposed. The TPM policy guides high temperature core to exchange tasks with its cooler neighbors as long as those task exchanges will not cause any thermal emergency in both cores.

4. 4. Overall Task Migration and Scheduling

DTB-M policy consists of both SSTM and TPM. The SSTM algorithm reduces the overall chip temperature by considering the thermal conductance of the chip. So that in a neighborhood, high power tasks can quickly be moved to the PE's that have better heat dissipation, while low power tasks can be moved to the PE's that are more easily to heat up. On the other hand, the TPM algorithm mitigates the local hot spots and balances the thermal gradient. After a master DTB agent triggers a migration request, it waits for the response from the slaves. In this request, the master sends out the list of its local task. When the slave receives the request, it performs the TPM algorithm. In the reply message, it sends the master the task pair that is found by the TPM algorithm and also the list of its local task. The master then performs SSTM. If SSTM algorithm found task pair whose exchange can reduce the average temperature of the whole chip, then the master will issue a task migration command. If the SSTM algorithm cannot find any task pair for exchange, the master DTB performs the TPM algorithm. After task migration finishes, a simple technique is employed to schedule the execution of tasks based on their average power consumption for both master and slave PE's. All tasks in a PE's run queue are sorted according to descending order of their average power consumption. The thermal aware scheduler will execute hot and cool tasks alternatively starting from the coolest and the hottest tasks, then the second coolest tasks and the second hottest, until all tasks have been executed once. It will start a new round of scheduling again. It is a simple yet effective scheduling technique that tries to reduce the operating temperature of the core. For a task with several phases that have different power and thermal characteristics, each phase is considered as a single task. New predictions will be made whenever a phase change is detected.

4. 5. Workload Balancing Policy

Workload balancing is triggered when a master PE_i finds the workload difference between itself and a slave PE_j exceeds the threshold nt_{diff} , that is,$||LT_i|- |LT_j|| > nt_{diff}$, $j€ N_i$. The goal of workload balancing is to maintain approximately equal number of tasks on each core and therefore improve worst case latency and response time. The master will pick the slave which gives the maximum workload difference. Then, tasks are migrated one by one from the PE with more tasks to the PE with fewer tasks until their difference is less than or equal to one. In every migration, the total steady-state temperature change of all processors after task migration is evaluated and the task which minimizes the same will be selected. It can be proved that if thermal conductance of master is greater than that of slave and number of tasks in master PE_i ($|LT_i|$) is greater than number of tasks running in slave PE_j ($|LT_j|$), the migration from PE_i to PE_j will start from the task with the highest power. On the other hand, if thermal conductance of master is less thermal conductance of slave and number of tasks in the master PE_i is less than number of tasks running in slave PE_j, the migration from PE_j to PE_i will start from the task with the lowest power.

V. Results and Discussion

RTL Schematic was obtained using Xilinx ISE 9.1i and simulation was done using ModelSim 6.2c. Programming was done using Verilog HDL. A gradual development from single core to multi-core and then to many-core systems with step by step application of the policies discussed is presented in this section. The rest of the section is organized as follows. Section-5.1 presents a single core using SSTM. Section-5.2 presents dual core using SSTM**.** Section-5.3 presents a 4-core system using both SSTM and TPM policies and section-5.4 presents an illustration to many core system through a 3x3 mesh or 9-core logic.

5. 1. Single Core using SSTM

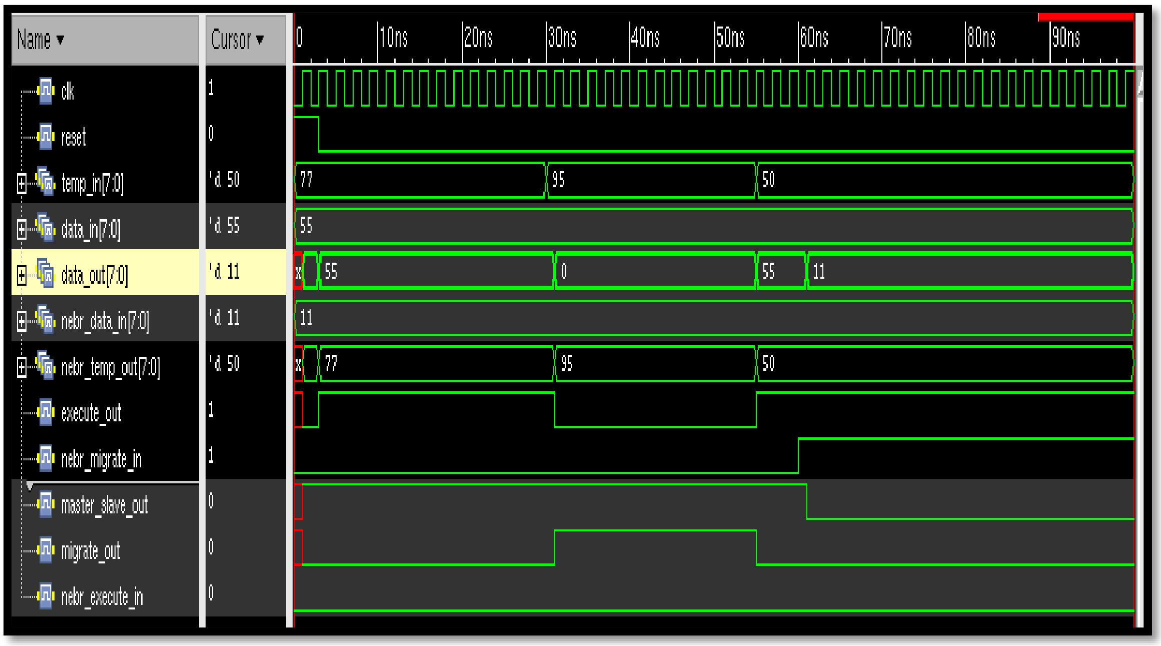
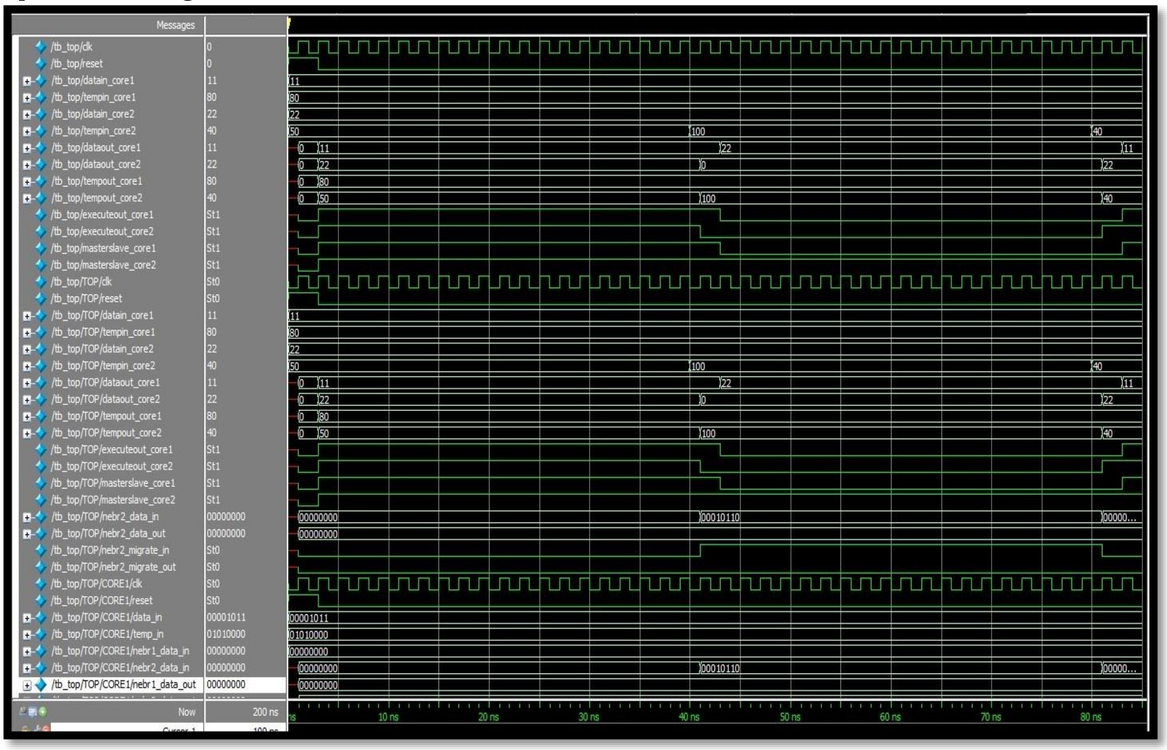


Figure 3. Screenshot of Output Waveforms of a Single Core

A single core was designed which takes inputs such as clock; reset; present temperature; input data; neighbour's temperature, data, migration requests etc. If it gets migrate request from the neighbour, then it stalls its present work and exercises neighbour's task. If its own temperature crosses the threshold temperature then it gives migration request to its neighbour and stalls its work. The temperature of the core was assigned as 77 at the beginning. The threshold temperature value was assigned as 90. As the temperature of the core exceeds the threshold value at 30ns, it gives a migration request to the neighbour and stalls its work. So the data out value could be seen as 0. Then when the temperature comes below the threshold value, it resumes task execution and gives output. Since a neighbour core is only assumed here, after sometime the neighbour data value of 11 was arbitrarily given to show migration to the core also.

5. 2. Dual Core using SSTM

Two cores were designed using the same logic for single core. Therefore each core takes inputs such as clock; reset; present temperature; input data; neighbour's temperature, data, migration requests etc. If it gets "migrate" request from the neighbour, then it stalls its present work and exercises neighbour's task. If its own temperature crosses the threshold temperature then it gives migration request to its neighbour and stalls its work.



```
Messages
/tb_top/clk                        0
/tb_top/reset                      0
/tb_top/datain_core1               11
/tb_top/tempin_core1               80
/tb_top/datain_core2               22
/tb_top/tempin_core2               40
/tb_top/dataout_core1              11
/tb_top/dataout_core2              22
/tb_top/tempout_core1              80
/tb_top/tempout_core2              40
/tb_top/executeout_core1           St1
/tb_top/executeout_core2           St1
/tb_top/masterslave_core1          St1
/tb_top/masterslave_core2          St1
/tb_top/TOP/clk                    St0
/tb_top/TOP/reset                  St0
/tb_top/TOP/datain_core1           11
/tb_top/TOP/tempin_core1           80
/tb_top/TOP/datain_core2           22
/tb_top/TOP/tempin_core2           40
/tb_top/TOP/dataout_core1          11
/tb_top/TOP/dataout_core2          22
/tb_top/TOP/tempout_core1          80
/tb_top/TOP/tempout_core2          40
/tb_top/TOP/executeout_core1       St1
/tb_top/TOP/executeout_core2       St1
/tb_top/TOP/masterslave_core1      St1
/tb_top/TOP/masterslave_core2      St1
/tb_top/TOP/nebr2_data_in          00000000
/tb_top/TOP/nebr2_data_out         00000000
/tb_top/TOP/nebr2_migrate_in       St0
/tb_top/TOP/nebr2_migrate_out      St0
/tb_top/TOP/CORE1/clk              St0
/tb_top/TOP/CORE1/reset            St0
/tb_top/TOP/CORE1/data_in          00001011
/tb_top/TOP/CORE1/temp_in          01010000
/tb_top/TOP/CORE1/nebr1_data_in    00000000
/tb_top/TOP/CORE1/nebr2_data_in    00000000
/tb_top/TOP/CORE1/nebr1_data_out   00000000
Now                                200 ns
```

Figure 4. Screenshot of Output Waveforms of the Dual Core System

The threshold value for temperature was assigned as 90. It could be seen that each core was given respective input data such as 11 to core 1 and 22 to core 2. They give respective output data till the temperature of core 2 goes above the threshold value. Then it gives migration request to core 1 and stalls its work till its own temperature goes below threshold. Core 1, on receiving request from neighbour executes its task and gives data output as 22 till core 2 resumes execution. After sometime temperature of core 2 was assigned to come down below threshold and then both cores gives their respective data outputs as normal. The predicted temperatures of cores were not considered and hence this system exclusively shows SSTM policy.

5. 3. Quad Core System using SSTM and TPM

As the name indicates, this system contains 4 cores. Each core takes inputs such as clock, reset, present temperature, predicted temperature, neighbour's data, temperature, migration requests etc. Here each core was designed to have two neighbouring cores. Neighbours to core 1 are 2 and 3; neighbours to core 2 are 1and 4; neighbours to core 3 are 1 and 4; and neighbours to core 4 are 2 and 3. Task migration takes places between the neighbouring cores following steady-state temperature-based migration (SSTM) and temperature prediction-based migration (TPM) policies. Core 1 gives t_1 and t_2 as outputs, and takes in t_3 and t_5; core 2 gives t_3 and t_4 as outputs, and takes in t_2 and t_8; core 3 gives t_5 and t_6 as outputs, and takes in t_2 and t_8; and finally core 4 gives t_7 and t_8 as outputs, and takes in t4 and t5. Each core also takes predicted temperature as one of its inputs. The conditions that has to be verified for effective task migration are- whether current temperature of core goes higher than the threshold temperature value; whether predicted temperature goes higher than threshold temperature and finally whether neighbour's temperature is less than the threshold temperature value. On analyzing the output waveforms, the data inputs to core 1, 2, 3 and 4 were assigned as 11, 22, 33 and

44 respectively. The threshold value of temperature for each core was assigned arbitrarily as 90. The temperature of core 2 was designed to go beyond threshold value after sometime and hence it gives out migration requests to neighbours (core 1 and core 4). The migration to core 4 is preferred over core 1 because the temperature input of core 4 is less than core 1. Hence data in core 2 migrates to core 4 whose current temperature is less than the threshold value for temperature. Here the predicted values of temperatures were also considered for task migration. Hence this system gives an illustration on how thermal-aware task migration takes place in many-core systems using SSTM and TPM policies.

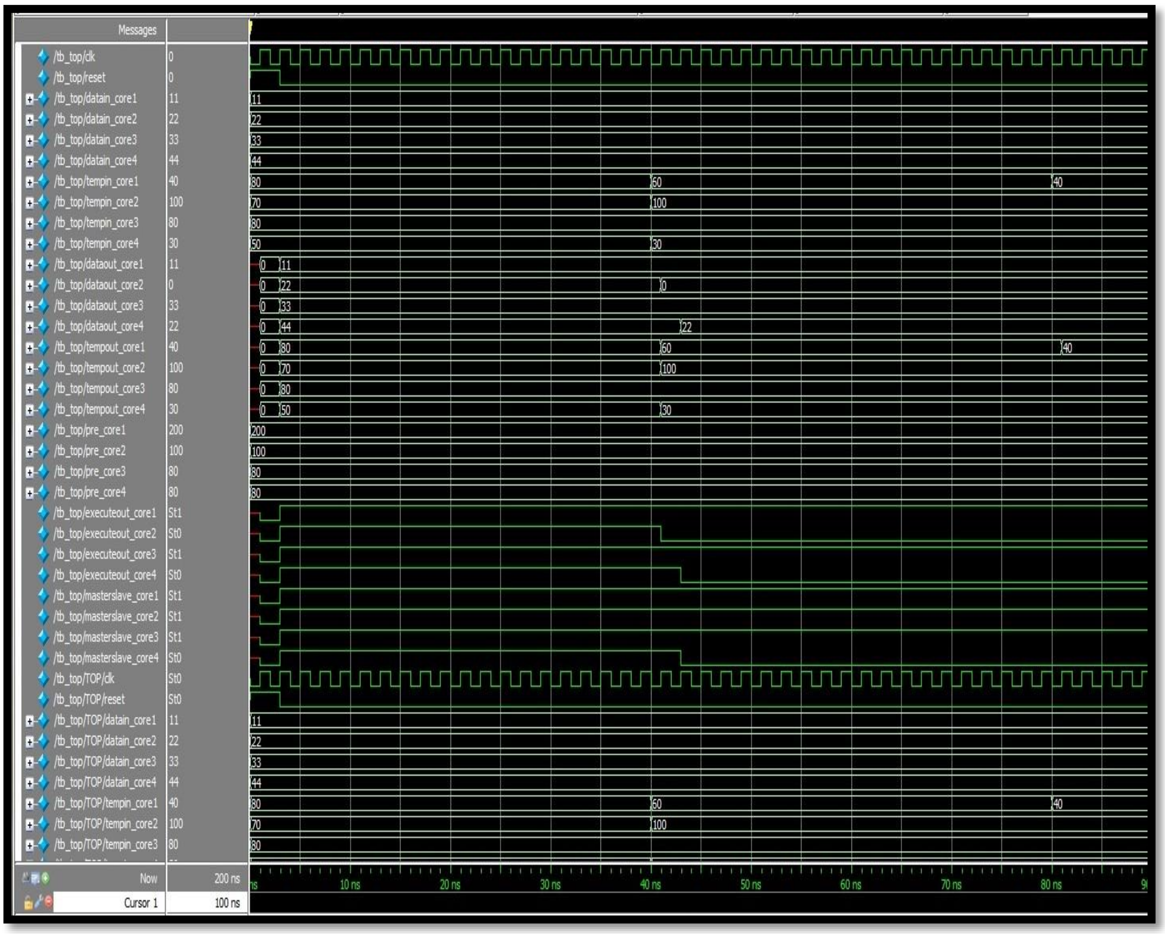


Figure 5. Screenshot of Output Waveforms of the Quad Core System

5. 4. 9-Core Logic

A 9-core system was designed and simulated as an illustration of a many-core system. The logic was developed in such a way as to explain thermal-aware task migrations. From the topology it could be seen that each core receives output data from two other cores. Core 1 receives data from cores 2 and 4; core 2 receives output data from cores 1 and 5; core 3 receives data from cores 2 and 6; core 4 receives output data from cores 1 and 7; core 5 receives data from cores 8 and 4; core 6 receives data from cores 5 and 9; core 7 receives data from cores 4 and 8; core 8 receives output data from cores 7 and 9 and finally core 9 receives data outputs of cores 6 and 8. In other words it could be said that each core has got at least two neighbours to which its task could get migrated.
Three conditions were checked to ensure effective task migration. First condition was that the current temperature of core should be higher than the threshold temperature value; second condition was that the predicted temperature of the core should also be higher than threshold temperature and finally the neighbour's temperature should be less than the threshold temperature. The screenshot of output waveforms obtained after simulation is shown in Figure. 6. 2. Cores from 1 to 9 were assigned with input data values as 11, 22, 33, 44, 55, 66, 77, 88 and 99 respectively. Temperatures of each core from 1 to 9 were initially assigned with values 80, 80, 80, 80, 80, 60, 80, 80 and 40 respectively. The

predicted temperature values of each core were initially assigned as 100, 200, 75, 200, 100, 100, 70, 70, and 200 respectively. Clock and rest pulses were given as input to each core. The threshold value was assumed to be 90 for each core.

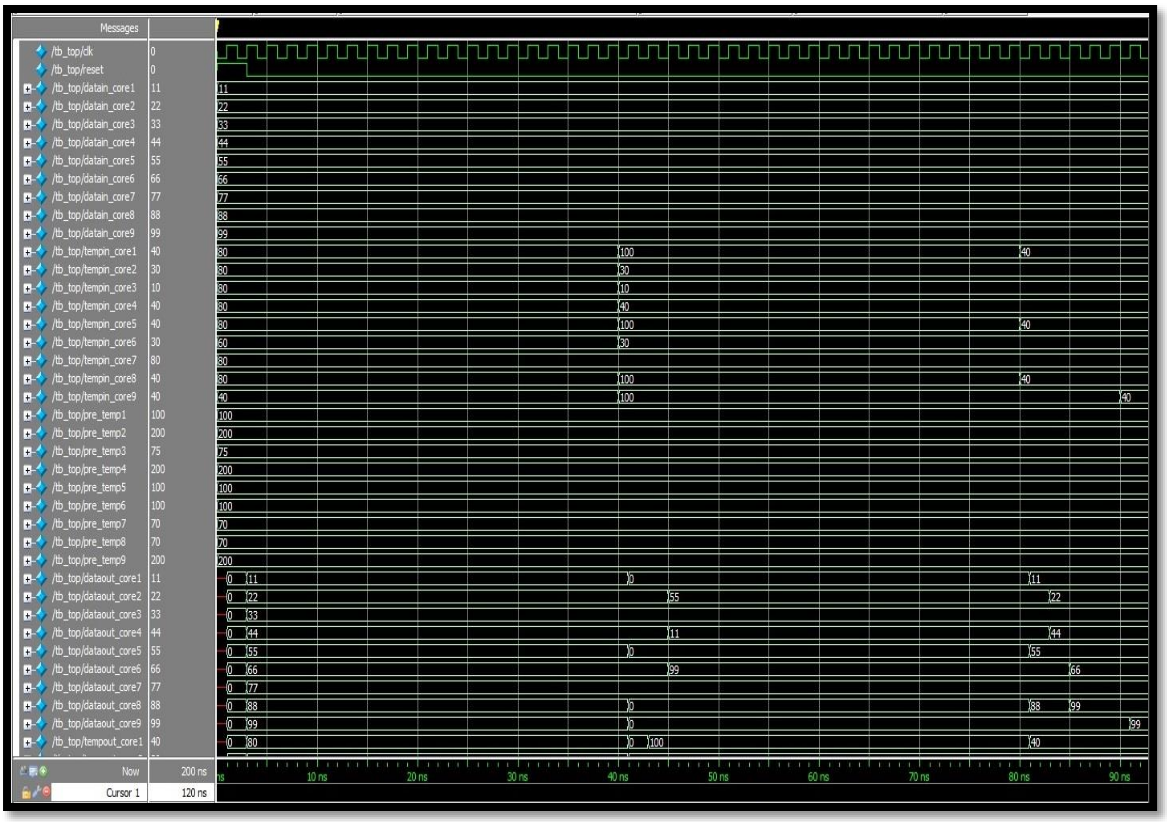


Figure 6. Screenshot of Output Waveforms of the 9-core System

At 40ns, when the temperature of core 9 reaches 100, it gives migration requests to its neighbours, which were core 8 and core 6. The predicted temperature of core 9 was 200 which was greater than the threshold value. Even though core 8 was the nearest neighbor to core 9, it could not consider the request because its own temperature reaches the value of 100 which was above the assigned threshold value. Hence the task from core 9, that is, the data value 99 gets migrated to core 6 whose current temperature value at that instant was below threshold. At the same instant, which is at 40ns, temperature of core 5 crosses the threshold value, and gives out migration requests to its neighbours, which were core 2 and core 6. But core 6 was locked by core 9 and the only available option was core 2. The predicted temperature of core 5 was assumed to be 100, which was greater than assigned threshold value. The temperature of core 2 was below threshold value and hence the task got migrated to core 2, that is, the output data of core 2 was 55 after 40ns. Again at 40ns, the temperature of core 1 could be observed as 100, and its predicted temperature value was assumed to be 100, and these two conditions initiates task migration. Migration requests were given to core 2 and core 4, which were its neighbours. Even though core 2 was the nearest neighbor to core 1, the same was locked by core 5 at that instant. Hence core 4 was considered, whose temperature was 40. Since the conditions were satisfactory, the data value 11 or task in core 1 could be seen as migrated to core 4.

At 80ns, the temperature of core 8 could be observed as 40, when it considers the migration request from core 9 which was given earlier. Then the task of core 9 could be seen received by core 8, and core 6 could be seen to have resumed the execution of its own task. At the same instant, the temperature of core 5 also could be observed as 40, and then it resumes its task execution which allows core 2 to resume its own task, which as the data value of 22. Similarly, the temperature of core 1 could be seen to have reached 40 at 80ns and then it resumes its own task execution, and its neighbour core would be free to execute its own task. At 90ns, the temperature of core 9 could be observed as below threshold and the data value of 99 could be seen as its output, and 88 at the output of core 8. The pin 'execute_out' was designed to give logic high when a core is executing its own task, and gives logic low value when a core is executing a neighbour's task. Hence both SSTM and TPM policies could be seen working complementary to each other, which results in effective task migration.

VI. CONCLUSION

A distributed thermal management framework for many-core systems was proposed. In this framework, no centralized controller was required. Each core had an agent which monitors the core temperature, communicates and negotiates with neighboring agents to migrate and distribute tasks evenly across the system. The agents use DTB-M policy for task migration. The DTB-M policy consisted of two parts. The SSTM migration policy distributes different tasks in a neighborhood based on their heat dissipation ability. The TPM migration policy ensures a good mixture of hot tasks and cool tasks on processors in a neighborhood. A neural network-based prediction model could be used not only for future temperature prediction but also for agents to evaluate the rewards of proposed migration offers. A 9-core system was designed and simulated as an illustration of many-core system. The logic was developed in such a way as to explain thermal-aware task migrations. Various conditions available as per the theory was illustrated in 9-core logic, such as by keeping nearest neighbor busy, both neighbours available, neighbor selection and also most importantly task scheduling. Thus the thermal-aware task migrations could avoid occurrence of hotspots and core throttling events to a great extent.

VII. FUTURE SCOPE

A standard networking stack uses services provided by the Operating System (OS) running on a single processor (single threaded). Performance improvements can be made to an OS networking stack by adapting the protocol stack processing software to support multiple processors (multi-threaded), either through the use of Symmetrical Multiprocessing (SMP) platforms or multi-core processor architecture. The cores that support for simultaneous multithreading (SMT) were not considered because it is anticipated that future many-core platform will be composed of large number of weaker and smaller cores with less transistors and power consumption, therefore, they are more likely to be single threaded cores. However, with some modification in the temperature prediction models, the same DTB-M algorithm could be applied to systems with SMT cores.

REFERENCES

[1]. Tilera Corporation, San Jose, CA, "Tile processor architecture: Technology brief," 2008. [Online]. Available: http://www.tilera.com/pdf/ ProductBrief_TileArchitecture_Web_v4.pdf.

[2]. Aleksander and H. Morton, *An Introduction to Neural Computing*. London, U.K.: International Thomson Computer Press, 1995.

[3]. S. Borkar, "Thousand core chips—A technology perspective," in *Proc. Design Autom. Conf. (DAC)*, 2007, pp. 746–749.

[4]. D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural level power analysis and optimizations," in *Proc. Int. Symp. Comput. Arch. (ISCA)*, 2000, pp. 83–94.

[5]. X. Chen, C. Xu, R. Dick, and Z. Mao, "Performance and power modeling in amulti-programmed multi-core environment," in *Proc. Design Autom. Conf. (DAC)*, 2010, pp. 813–818.

[6]. J. Choi, C. Cher, H. Franke, H. Hamann, A. Weger, and P. Bose, "Thermal aware task scheduling at the system software level," in *Proc. Int. Symp. Low Power Electron. Design*, 2007, pp. 213–218.

[7]. R. Cochran and S. Reda, "Consistent runtime thermal prediction and control through workload phase detection," in *Proc. Design Autom. Conf. (DAC)*, 2010, pp. 62–67.

[8]. A. Coskun, T. Rosing, and K. Whisnant, "Temperature aware task scheduling in MPSoCs," in *Proc. Design Autom. Test Euro. (DATE)*, 2007, pp. 1659–1664.

[9]. A. Coskun, T. Rosing, and K. Gross, "Utilizing predictors for efficient thermal management in multiprocessor SoCs," *IEEE Trans. Comput.- Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1503–1516, Oct. 2009.

[10]. W. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. Design Autom. Conf.*, 2001, pp. 684–689.

[11]. J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Proc. Int. Symp. Comput. Arch. (ISCA)*, 2006, pp. 78–88.

[12]. T. Ebi, M. Al Faruque, and J. Henkel, "TAPE: Thermal-aware agentbased power economy for multi/many-core architectures," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2009, pp. 302–309.

[13]. J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, F. Pailet, S. Jain, T. Jacob, S. Yada, S. Marella, P. Salihundam, V. Erraguntla, M. Konow, M. Riepen, G. Droege, J. Lindemann, M. Gries, T. Apel, K. Henriss, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, R. Van DerWijngaart, and T. Mattson, "A 48-Core IA-32 message-passing processor with DVFS in 45 nm CMOS," in *Proc. Int. Solid-State Circuits Conf. (ISSCC)*, 2010, pp. 108–109.

[14]. J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proc. Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2003, pp. 233–239.

[15]. W. Huang, M. Stan, K. Sankaranarayanan, R. Ribando, and K. Skadron, "Many-core design from a thermal perspective," in *Proc. Design Autom. Conf. (DAC)*, 2008, pp. 746–749.

[16]. R. Jayaseelan and T. Mitra, "Dynamic thermal management via architectural adaption," in *Proc. Design Autom. Conf. (DAC)*, 2009, pp. 484–489.

[17]. L. Ljung, *System Identification: Theory for the User (2nd Edition)*. Upper Saddle River, NJ: Prentice-Hal PTR, 1999.

[18]. R. Marculescu, U. Ogras, L. Peh, N. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives," *IEEE Trans. Comput.—Aided Design Integr. Circuits Syst.*, vol. 28, no. 1, pp. 3–21, Jan. 2009.

[19]. P. Michaud, A. Seznec, D. Fetis, Y. Sazeides, and T. Constantinou, "A study of thread migration in temperature-constrained multicores," *ACM Trans. Arch. Code Optim. (TACO)*, vol. 4, no. 2, pp. 9-1–9-28, Jun. 2007.

[20]. D.Wentzlaff, C. Gruenwald, N. Beckmann, K.Modzelewski, A. Belay, L. Youseff, J.Miller, and A. Agarwal, "A Unified operating system for clouds and manycore: FOS," MIT-CSAIL-TR-2009-059, Nov. 2009.

[21]. F. Mulas, M. Pittau, M. Buttu, S. Carta, A. Acquaviva, L. Benini, D. Atienza, and G. De Micheli, "Thermal balancing policy for streaming computing on multiprocessor architectures," in *Proc. Design Autom. Test Euro. (DATE)*, 2008, pp. 734–739.

[22]. S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. De Micheli, "Temperature-aware processor frequency assignment forMPSoCs using convex optimization," in *Proc. CODES+ISSS*, Sep. 2007, pp. 111–116.

[23]. V. Nollet, T. Marescaux, and D. Verkest, "Operating system controlled network on chip," in *Proc. Design Autom. Conf. (DAC)*, 2004, pp. 256–259.

[24]. R. Rao and S. Vrudhula, "Fast and accurate prediction of the steady state throughput of multi-core processors under thermal constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1559–1572, Oct. 2009.

[25]. L. Shang, L. Peh, A. Kumar, and N. Jha, "Thermal modeling, characterization and management of on-chip networks," in *Proc. Int. Symp. Microarch.*, 2004, pp. 67–78.

[26]. S. Sharifi, A. Coskun, and T. Rosing, "Hybrid dynamic energy and thermal management in heterogeneous embedded multiprocessor," in *Proc. Asia South Pacific Design Autom. Conf. (ASPDAC)*, 2010, pp. 873–878.

[27]. K. Skadron, M. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Arch. Code Opt.*, vol. 1, no. 1, pp. 94–125, Mar. 2004.

[28]. S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Lyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, "An 80-tile 1.28 TFLOPS network-on-chip in 65 nm CMOS," in *Proc. Int. Solid-State Circuits Conf. (ISSCC)*, 2007, pp. 98–589.

[29]. Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation," in *Proc. Int. Symp. Comput. Arch. (ISCA)*, 2009, pp. 314–324.

[30]. J.Wawrzynek, D. Patterson, M. Oskin, S. Lu, C. Kozyrakis, J. Hoe, D. Chiou, and K. Asanovic, "RAMP: Research accelerator for multiple processors," *IEEE Micro*, vol. 27, no. 2, pp. 46–57, Aug. 2007.

[31]. A. Yeo, C. Liu, and E. Kim, "Predictive dynamic thermal management for multicore systems," in *Proc. Design Autom. Conf. (DAC)*, 2008, pp. 734–739.

[32]. F. Zanini, D. Atienza, and G. De Micheli, "A control theory approach for thermal balancing of MPSoC," in *Proc. Asia South Pacific Design Autom. Conf. (ASPDAC)*, 2009, pp. 37–42.

[33]. S. Liu, J. zhang, Q. Wu, and Q. Qiu, "Thermal-aware job allocation and scheduling for three dimensional chip multiprocessor," in *Proc. Int. Symp. Quality Electron. Design (ISQED)*, 2010, pp. 390–398.

[34]. Y. Ge, P. Malani, and Q. Qiu, “Distributed task migration for thermal management in many-core systems,” in *Proc. Design Autom. Conf. (DAC)*, 2010, pp. 579–584.

[35]. Y. Ge, Q. Qiu, and Q. Wu, “Multi-agent framework for thermal-aware task migration in many-core systems,” in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2012, pp. 1758-1771.

AUTHORS

Ershad. S. B is working as Asst. Professor at Nehru College of Engineering & Research Centre, Thrissur, Kerala. He is an associate member of Institution of Engineers, India. He received M.Tech degree in VLSI Design from Amrita Vishwa Vidyapeetham, Coimbatore, Tamil Nadu.

Sreejith. S. Nair is doing M.Tech in VLSI Design at Nehru College of Engineering & Reseach Centre, Thrissur, Kerala under the University of Calicut. He received B.Tech degree from University of Calicut in 2011.