Adaptive Machine Learning-Based Proactive Thermal Management for NoC Systems

Kun-Chih Chen, *Senior Member, IEEE*, Yuan-Hao Liao, Cheng-Ting Chen, and Lei-Qi Wang

***Abstract*— Because of the high-complex interconnection in contemporary multicore systems, the network-on-chip (NoC) technology has been proven as an efficient way to solve the communication problem in multicore systems. However, the thermal problem becomes the main design challenge in the current NoC systems due to the high-diverse workload distribution and large power density. Therefore, proactive dynamic thermal management (PDTM) is employed as an efficient way to control the system temperature. Based on the predicted temperature information, the PDTM can control the system temperature in advance to reduce the performance impact during the temperature control period. However, conventional temperature prediction models are usually built based on specific physical parameters, which are usually temperature-sensitive. Consequently, the current temperature prediction models still result in significant temperature prediction errors. To solve this problem, a novel adaptive machine learning (ML)-based PDTM is proposed in this work. The adaptive ML-based PDTM first uses an adaptive single layer perceptron (ASLP), which is composed of a single-neuron operation and a least mean square (LMS) adaptive filter technology, to precisely predict the future temperature. Afterward, the proposed adaptive reinforcement learning (RL) is used to find the proper throttling ratio to control the system temperature. In this way, the proposed adaptive ML-based PDTM can adapt to the hyperplane of the temperature behavior of the NoC system and provide a proper temperature control strategy at runtime. Compared with related works, the proposed approach reduces average temperature prediction error by 0.2%–78.0% and improves the system performance by 2.4%–43.0% with smaller hardware overhead.**

***Index Terms*— Machine learning (ML), network-on-chip (NoC), neural network, reinforcement learning (RL), temperature prediction, thermal management.**

I. Introduction

With the advantage of semiconductor technology, a huge amount of transistors can be integrated into a single chip, which catalyzes the multicore system era. Because the required number of involved processing cores is increased to support multiple tasks simultaneously, the interconnection complexity in multicore systems becomes considerable. To mitigate the interconnection problem in multicore systems, the network-on-chip (NoC) technology has been proven as an efficient way to connect each processing element (PE) in multicore systems [1]. However, because of the high power density and high-diverse workload distribution, NoC systems usually suffer from severe thermal problems [2]. The thermal issue leads to many negative impacts, such as lower system reliability and longer system latency.

To mitigate the thermal problem of NoC systems, it is necessary to aware of the system temperature at runtime. The temperature can be obtained through either the physical thermal sensors embedded in each NoC node [3] or by estimating it based on the temperature sensing results from other NoC nodes [4]. While the system temperature reaches an alarming level, the involved dynamic thermal management (DTM) will be triggered to prevent the system from overheating [2], [4], [5]. Among the different DTM approaches, the dynamic voltage frequency scaling (DVFS) scheme [6] is proven as the most efficient way to control the system temperature. The DVFS scheme is used to control the operating frequency of the NoC nodes (i.e., throttle the NoC nodes), in which the node temperature becomes thermal emergency until the node temperature becomes thermal safety. Please note that each NoC node includes a PE, a memory, and a router. The DTM can be classified into reactive dynamic thermal management (RDTM) and proactive dynamic thermal management (PDTM). The conventional RDTM is triggered while the temperature of the NoC nodes reaches the triggering temperature [7]. When the RDTM is activated, the overheated node will be fully throttled to decrease the temperature. Although the RDTM can cool down the thermal-emergent NoC nodes quickly, it causes a significant performance impact. The reason is that the involved fully throttling scheme is used to shut down the thermal emergent NoC nodes. In contrast, to solve the problem of RDTM, the PDTM controls the system temperature in advance based on the information on temperature prediction [8]. With the partial throttling scheme (i.e., the NoC nodes are not fully throttled), the PDTM can mitigate the performance impact during the temperature control period compared with the RDTM. Consequently, PDTM has been proven as an efficient way to control the system temperature.

The efficiency of the PDTM depends on the precision of the temperature prediction and the throttling ratio selection. In a practical way, most temperature prediction models are formulated based on several physical parameters, such as capacitance, resistance, and power [9], [10]. However, these

Manuscript received 4 December 2022; revised 26 March 2023 and 14 May 2023; accepted 1 June 2023. Date of publication 15 June 2023; date of current version 26 July 2023. This work was supported by the Ministry of Science and Technology, Taiwan, under Grant NSTC 110-2221-E-110-026-MY3 and Grant NSTC 111-2628-E-110-011-MY3. *(Corresponding author: Kun-Chih Chen.)*

Kun-Chih Chen is with the Institute of Electronics, National Yang Ming Chiao Tung University, Hsinchu 300093, Taiwan (e-mail: kcchen@nycu.edu.tw).

Yuan-Hao Liao, Cheng-Ting Chen, and Lei-Qi Wang are with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 804, Taiwan (e-mail: jack@cereal.cse.nsysu.edu.tw; aa860407@cereal.cse.nsysu.edu.tw; laichi609@cereal.cse.nsysu.edu.tw).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2023.3282969.

Digital Object Identifier 10.1109/TVLSI.2023.3282969

physical parameters are usually temperature-sensitive (i.e., the value is different according to the temperature change) [11]. Besides, the temperature behavior on NoC systems is different according to various workload distributions on NoC systems, which worsens the difficulty of catching a precise value for these physical parameters at runtime. Therefore, these physical parameters are usually seen as constants in the involved temperature prediction models, which leads to large temperature prediction errors. The reason is that the involved constant physical parameters cannot satisfy every kind of temperature-changing situation. In contrast, the machine learning (ML)-based prediction methods for data estimation have received much attention in recent years because they can dynamically satisfy the hyperplane of a physical system behavior [14]. However, the precision of the ML-based prediction method highly depends on the involved training data. Hence, the temperature prediction error is still considerable by applying ML-based prediction methods because it is difficult to collect all kinds of temperature situations on NoC systems. With the imprecise temperature prediction information, adopting PDTM to control the system temperature becomes challenging. Therefore, the involved PDTM still needs to consider the pessimistic temperature control policy, leading to a dramatic performance impact, as shown in Fig. 1(a).

In addition to the imprecise temperature prediction result, the involved control policy (i.e., the selected throttling ratio) of the adopted PDTM affects the system performance significantly. In [9], Chen et al. proposed a partial throttling mechanism to control the system temperature early. Using the temperature prediction information, the authors partially throttle the potential thermal-emergent NoC (i.e., the predicted node temperature exceeds the thermal limit). On the contrary, the NoC node will be fully throttled as soon as the node temperature reaches an alarming level. However, this kind of deterministic temperature control method considers the worst case situation to prevent the system from overheating because of the involved fixed thermal control strategy. Therefore, applying this method to manage each NoC node's temperature is improper, which may lead to an over-control problem and performance degradation. Unfortunately, it is an NP-hard problem to find the optimal throttling ratio for each NoC node [12]. To solve this problem, Chen and Marculescu [13] applied distributed reinforcement learning (RL) to find the proper throttling ratio to maintain the system performance under a limited power budget. In contrast, Kim et al. [33] adopted imitation learning (IL) to control the voltage/frequency of the involved many-core system. Although the RL and IL methods can determine the proper control policy according to the different situations in the training phase, the deterministic control policy cannot adapt to the different workloads on NoC systems. The reason is that the decision-making policy highly depends on the training dataset at the training phase. Consequently, using current ML-based throttling control methods, the NoC system still suffers from a large performance impact.

After investigating the state-of-the-art, the current design challenges of PDTM methods for thermal-aware NoC systems are: 1) the imprecise temperature prediction results and 2) the deterministic temperature control policy. To solve the problems above, we propose an adaptive ML-based PDTM in this work. The adaptive ML-based PDTM first considers an adaptive single layer perceptron (ASLP)-based temperature prediction method by involving the least mean square (LMS) adaptive filter theory [15], as shown in Fig. 1(b). The LMS adaptive filter is used to find the proper weights for the involved ASLP-based temperature prediction to fit the hyperplane of the temperature behavior based on the temperature prediction error. Furthermore, to solve the problem of the conventional deterministic PDTM methods (i.e., fixed thermal control strategy), we propose an adaptive reinforced learning (RL) method to assist with the proper throttling ratio selection for further temperature control. Different from the conventional feedback mechanism (i.e., the throttling level is determined according to the current and predicted temperature), the RL method considers the current rewards (i.e., the information about the current temperature, the predicted temperature, and the throughput) to adjust the throttling ratio dynamically, as shown in Fig. 1(b). In this way, the proposed adaptive ML-based PDTM will ensure thermal safety and system throughput simultaneously. The main contributions of this article are summarized as follows:

1) *Adaptive SLP (ASLP)-Based Temperature Prediction Model for Precise Temperature Prediction:* The LMS adaptive filter theory [15] is employed to dynamically adjust the involved weights in the proposed ASLP-based temperature prediction model based on the information about the prediction error.
2) *Adaptive RL-Based Dynamic Throttling Ratio Selection for Precise Temperature Control:* The Q-learning method [17] is adopted to propose an adaptive RL-based temperature control policy in this work. The adaptive RL-based method is used to dynamically select and adjust the proper throttling ratio based on the temperature control efficiency, which helps to control the system temperature more precisely.
3) *Architecture of Adaptive ML-Based PDTM Unit Design:* We combine the proposed adaptive ASLP-based temperature prediction and the adaptive RL-based dynamic throttling ratio selection to design a low hardware-cost adaptive ML-based PDTM unit.

To sum up, the proposed adaptive ML-based PDTM approach provides a precise temperature prediction and helps to control the system temperature precisely. Compared with the related works, we can reduce the average temperature prediction error by 0.2%–78.0%, which helps to improve the system performance by 2.4%–43.0% with smaller hardware overhead.

The rest of this article is organized below. The related works are investigated in Section II. In Section III, the proposed adaptive ASLP-based temperature prediction method will be introduced. Then, we will introduce the proposed adaptive RL-based dynamic throttling ratio selection method in Section IV and analyze the experimental results in Section V. The corresponding adaptive ML-based PDTM architecture will be described in Section VI. At last, we conclude this article in Section VII.

II. Background and Related Works

A. RC-Based Temperature Prediction Model

Because the thermal behavior can be modeled by an analogous electrical resistor–capacitor (RC) network, Chen et al. [9] employed the thermal RC model to propose

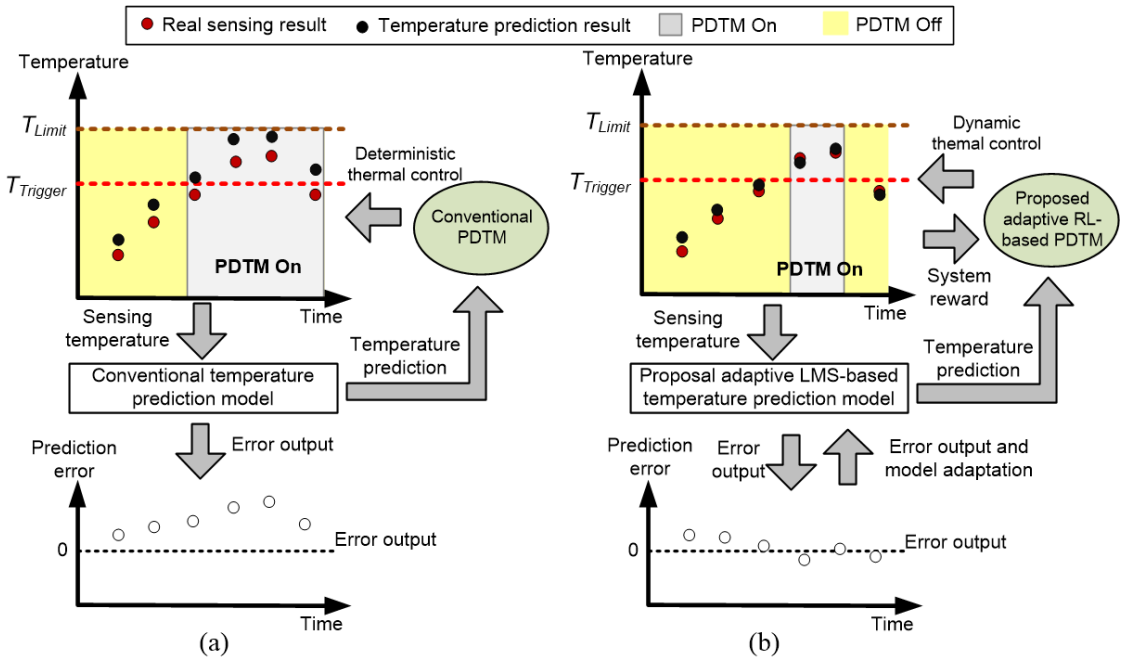

Fig. 1. (a) Conventional PDTM uses a deterministic thermal control policy due to the imprecise temperature prediction and (b) proposed adaptive ML-based PDTM can control the system temperature properly because it can adapt to the temperature behavior.

an *RC*-based temperature prediction model. The authors distinguish the temperature prediction models into temperature increasing mode and decreasing mode to reduce the temperature prediction error. The increasing mode is selected when the system temperature trend is increasing monotonously. In this case, the linear approximation way is used to predict the future temperature. On the contrary, the decreasing mode is used to predict the future temperature while the involved DTM starts to control the system temperature. Because of the worst case consideration, the authors assume that the temperature always increases after each temperature drop. Hence, the error of the temperature prediction is still very large, which counteracts the benefit of the PDTM. Besides, the involved constant *RC* parameters worsen the problem of significant temperature prediction error when the temperature change.

B. Second Derivative-Based Thermal Prediction Model

Lei et al. apply the second derivative to analyze the involved thermal *RC* model in [10] to find the temperature increasing rate to predict the future temperature. Therefore, the variation tendency of the temperature behavior can be observed. In this work, the authors assume that the trend of the system temperature behavior is increasing monotonously. Therefore, this model does not consider the time-varying system workload distribution, which includes increasing and decreasing temperature changing trends. In this way, this approach still suffers from a significant temperature prediction error.

C. Linear Regression-Based Temperature Prediction

Because the future temperature depends on the current temperature and the power consumption within a period, Weber et al. proposed to use a linear regression approach to predict the future temperature based on the information on the successive sensing temperature and the power consumption in [16]. The authors first use the power model to estimate future power consumption. Then, the temperature predictor uses the information about the current temperature and the estimated future power consumption to forecast the system temperature. To observe a proper linear regression model, the authors need to collect enough data by running different applications on the target platform. Hence, it cannot adapt to the high-diverse temperature behavior while the running applications are beyond the preloaded applications. Besides, acquiring information about the transient power consumption at runtime is hard.

D. Kalman Filter-Based Thermal Prediction Model

In [18], Kahng et al. proposed to use multiple linear regression to build a power model for NoC systems and employ the Kalman filter to formulate a thermal prediction model. The Kalman filter is a well-known method to estimate the unknown variable by considering the joint distribution of the measurement data at each measurement time. In this work, the authors first use the thermal-capacitance matrix and thermal-conductivity matrix to derive the state matrix, input matrix, and output matrix. With these matrices, the future temperature can be predicted after several computing iterations. However, this method usually requires performing matrix multiplication at runtime. Hence, the computing complexity and the area overhead are increasing with respect to the larger NoC size. Besides, the involved matrices with fixed physical parameters still lead to imprecise temperature prediction results.

After investigating the state-of-the-art temperature prediction methods, the current design problems are as follows:

1) *Fixed Physical Parameters Cannot Present the Real System Situation:* Because most physical parameters (e.g., thermal capacitance, thermal resistance, etc.) are temperature-sensitive, the physical parameters are not constant at every different temperature status [11].

2) *Exact System Running Status Is Hard to Acquire at Runtime:* Although some related works collect the actual system running information (e.g., transient power consumption, etc.) to evaluate the temperature status, the design complexity becomes dramatically considerable.

Hence, the current temperature prediction methods still suffer from imprecise temperature prediction results, which leads to improper temperature control. To solve this problem, we propose an adaptive single layer perception to precisely predict the temperature behavior and dynamically select the proper temperature control policy by using the proposed adaptive RL method, which will be discussed later.

III. ASLP-Based Temperature Prediction

As mentioned before, the conventional temperature prediction methods cannot adapt to the high-diverse workload on NoC systems. The reason is that the involved physical parameters are usually temperature-sensitive. In this section, we first employ a single layer perceptron (SLP) to learn the temperature behavior of NoC systems. Afterward, we further propose an ASLP-based method to adapt to the value of the involved temperature-sensitive parameters in the model. In this way, we can improve the precision of the temperature prediction results.

A. SLP-Based Temperature Prediction Model

The conventional way to model the temperature behavior is by considering the physical knowledge, such as thermal conductance, thermal resistance, or thermal capacitance. Therefore, Zhang et al. formulate a temperature prediction model in [19] as

$$T(k+N|k) = B^N T(k|k) + \sum_{i=k}^{q} B^{q-i} D u(i) \tag{1}$$

where B and D are both coefficients depending on the parameters of the thermal conductance and capacitance circuit; $u(i)$ is the power consumption model. Besides, the $T(k|k)$ is the temperature at time k, and the $T(k+N|k)$ represents the future temperature after N time periods. Note that the q is the time after $N-1$ time periods at time k (i.e., $q = k+N-1$). Because it is necessary to analyze the details of the circuits to capture the values of these physical parameters, it is hard to find them when the system scale becomes large.

To leverage the temperature prediction, it is necessary to observe the hyperplane of temperature changing behavior on NoC systems at runtime. Because multiply-accumulation operations are adopted to compute (1), it can be seen as a single neuron operation in a traditional artificial neural network (ANN). The ANN is a popular way to estimate the possible output results based on the receiving inputs. Therefore, it is proper to approximate a hyperplane of temperature behavior. Consequently, (1) can be reformulated to

$$T(k+N|k) = \sum_{n=1}^{k-q+2} w_n \cdot X_n \tag{2}$$

where w_n is an integrated factor combing B and D in (1). Besides, the X_n in (2) is the input data, including $T(k|k)$ and $u(i)$. Through the ANN approach in (2), we can not only calculate a similar computation in (1) but also use the well-trained weights [i.e., w_n in (2)] instead of the physical coefficients in (1) to estimate the future temperature.

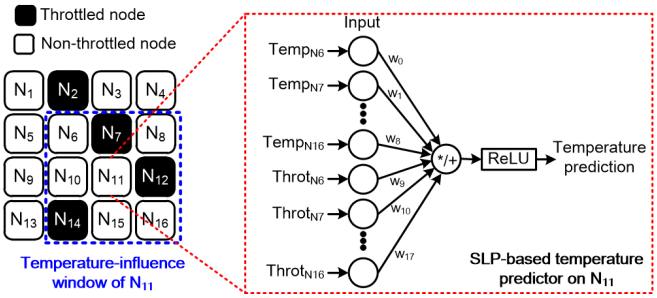

Fig. 2. Proposed single layer perception (SLP)-based temperature prediction model.

Because the temperature prediction unit is an extra computing unit in NoC systems, it is necessary to have the features of low computing complexity and low power consumption. In this way, we can achieve the goal of timely temperature prediction with a light temperature injection effect (i.e., the operation of temperature prediction does not increase system temperature significantly). Based on this design concept, we select an SLP to calculate (2) because SLP performs only one neuron operation. In this work, we embed an SLP-based temperature prediction unit in each NoC node, as shown in Fig. 2. Because the results of SLP computation depend on the input features significantly, it is crucial to find the proper factors for the proposed SLP-based temperature prediction model. To find the proper factors, the temperature change within a time unit can be modeled as

$$\frac{dT(t)}{dt} = \frac{P(t)}{C} - \frac{T(t)}{RC} \tag{3}$$

according to the Fourier's law. Note that the $T(t)$ is the current temperature, and $P(t)$ is the total power consumption of a specific NoC node at time t. Besides, the R and C represent the effective thermal resistance and thermal capacitance, respectively. Obviously, the temperature gradient depends on the total power consumption and current temperature. In other words, the temperature change from the time t to the time $(t+\Delta t_s)$ is determined based on the information on the total power consumption from the time t to the time $(t+\Delta t_s)$. The Δt_s represents the temperature sensing period. Because the power consumption linearly depends on the clock frequency, the node temperature will be affected by throttling activity. In addition to the current temperature and the throttling activity of the current local NoC node, each NoC node temperature is affected by the neighboring NoC nodes surrounding the current local NoC node because of the spatial correlation of the temperature distribution [20]. Consequently, we consider the current temperature and throttling information about the current local NoC node and the neighboring NoC nodes surrounding the current local node as features of the proposed SLP-based prediction model, as shown in Fig. 2. As mentioned before, the temperature information of the surrounding environment can be obtained either through physical embedded sensors [3] or by estimating it using temperature sensing data from other nodes [4]. In contrast, the throttling status can be notified by using a 1-bit signal.

The scale of the proposed SLP-based temperature prediction model depends on the size of the temperature-influence

window (i.e., the NoC nodes in this window significantly contribute temperature impact to the target node). To find the proper size of the temperature-influence window, we employ the Fourier's law and formulate the one-dimensional (1-D) heat flow as

$$Q = -kA\frac{dT}{dx} = -kA\frac{\Delta T}{\Delta x} \tag{4}$$

where Q is the rate of heat flow in the x-direction through a surface of area A. Besides, $(\Delta T/\Delta x)$ is the temperature gradient in the x-direction, and the k is the thermal conductivity, which is a property of the material. By analyzing (4), we realize that the heat flow becomes smaller when the Δx becomes larger. In other words, the temperature impact is negligible when the distance between the two NoC nodes is far. Therefore, to consider the size of the proposed SLP-based temperature prediction model, we define a temperature-influence window including one target NoC node (e.g., N_{11} in Fig. 2) and eight NoC nodes (e.g., N_6, N_7, N_8, N_{10}, N_{12}, N_{14}, N_{15}, and N_{16} in Fig. 2) surrounding the target NoC node. The reason is that the eight NoC nodes surrounding the target NoC node affect the temperature of the target NoC node significantly.

Because the temperature behavior is not a linear function, it is necessary to involve a nonlinear function to make the result of (2) approximate it. Regarding the involved nonlinear function, which is the so-called activation function in SLP, in the proposed SLP-based temperature prediction model, the ReLU is selected. The reason is that the chip's temperature is positive, and the temperature change within a very short period can be seen as a linear function behavior. In this way, the proposed SLP-based temperature prediction model can be converged easily to fit the hyperplane of the temperature behavior on NoC systems, which can be further used to predict the future temperature based on the current system status. During the temperature prediction at runtime, we can eliminate the ReLU function and use these trained weights to evaluate the future temperature. In this way, we can use a low-cost linear equation to predict the future temperature.

B. Adaptive SLP-Based Temperature Prediction Model by Using LMS Adaptive Filter Theory

Although the proposed SLP-based temperature prediction model can satisfy the hyperplane of the temperature behavior of a system through the training phase, it still suffers from a large temperature prediction error due to the fixed weights for the SLP computing. The reason is that the workload on NoC systems is usually time-varying, but the trained weights highly depend on the involved training dataset. Therefore, the proposed SLP-based temperature prediction model cannot cover every situation of temperature change and is hard to adapt to the behavior of system temperature under time-varying workload distribution. With this motivation, adjusting the weights during the runtime is necessary according to the different temperature situations.

To adjust the weights at runtime, we apply the LMS-based adaptive filter theory in this work. The LMS-based adaptive filter theory is widely used to calibrate the involved coefficients to estimate the future signal trend based on the history of the signal. As the example in Fig. 2, the predicted temperature of

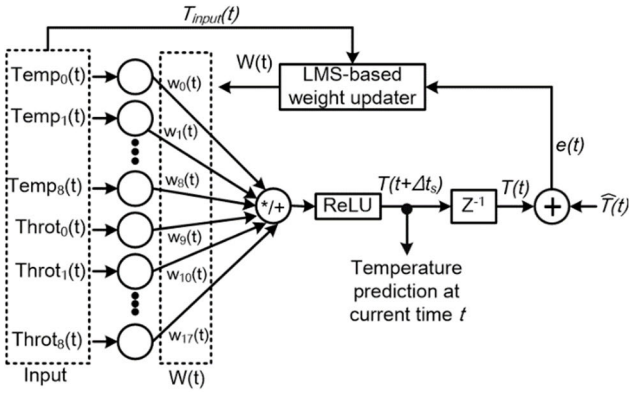

Fig. 3. Block diagram of proposed LMS-based thermal prediction model.

N_{11} at the time $(t + \Delta t_s)$ by using the proposed SLP-based temperature prediction model can be formulated to

$$\begin{aligned}
&\mathrm{ReLU}[T(t+\Delta t_s)] \\
&= \mathrm{ReLU}\big[w_0(t)\mathrm{Temp}_{N_6}(t) + w_1(t)\mathrm{Temp}_{N_7}(t) + \cdots \\
&\qquad + w_8(t)\mathrm{Temp}_{N_{16}}(t) + w_9(t)\mathrm{Throt}_{N_6}(t) \\
&\qquad + w_{10}(t)\mathrm{Throt}_{N_7}(t) + \cdots + w_{17}(t)\mathrm{Throt}_{N_{16}}(t)\ \big] \\
&= \mathrm{ReLU}\left[\sum_{n=0}^{17} w_n(t)T_{\mathrm{input}}(t)\right]
\end{aligned} \tag{5}$$

where $w_n(t)$ includes the trained weights in the model and $T_{\mathrm{input}}(t)$ collects the inputs of the proposed SLP-based temperature prediction model (i.e., the information about the temperature and the throttling state of the nodes in the temperature-influence window). Besides, the Δt_s is the temperature sensing period. The trained weights in (5) can be seen as the adjustable coefficients in the LMS-based adaptive filter. Therefore, the involved trained weights can be fine-tuned based on the information on the estimation error (i.e., the temperature prediction error). Fig. 3 shows the block diagram of the proposed adaptive SLP (ASLP)-based temperature prediction model by using LMS-based adaptive filter theory. Based on the information about the current sensing temperature $\hat{T}(t)$ and the predicted temperature $T(t)$ at the previous sensing time, the temperature estimation error $e(t)$ can be formulated to

$$e(t) = \hat{T}(t) - T(t). \tag{6}$$

Based on the temperature sensing information from the placed thermal sensors, each NoC node can be informed of its local temperature information [4]. Hence, we can apply the LMS-based adaptive filter theory to fine-tune $w_n(t)$, used to calculate the predicted temperature at the next thermal sensing time, by using

$$w_n(t_s) = w_n(t - \Delta t_s) + \mu e(t)T_{\mathrm{input}}(t) \tag{7}$$

where the $T_{\mathrm{input}}(t)$ is a vector including all required input features for the proposed ASLP-based temperature prediction model. The μ in (7) is the step-size parameter, which affects the convergence speed of the adopted weight adjustment method. In this work, we set μ to the sensing period Δt_s. Besides, $w_n(t - \Delta t_s)$ is the current weight before weight adjustment. Because we can find the proper weights based on the information on each temperature estimation error, the

proposed ASLP-based temperature prediction model can adapt to different temperature behavior at runtime.

Because we only use operations of multiplication and addition to update the involved weights in each NoC node, the computing complexity is O(1) and the computing frequency is aligned with the involved process technology. In contrast, the thermal profile of chip takes hundreds of milliseconds to change [19], [22], and thereby the computation of weights updating in (7) can be performed in time.

IV. Adaptive RL-Based Temperature Control Mechanism

A. RL-Based Temperature Control Process

By using the proposed ASLP-based temperature prediction model, the involved PDTM can control the system temperature based on the information about the predicted temperature. Although the PDTM uses a partial throttling scheme to control the system temperature without a significant performance impact, it is difficult to determine the proper throttling ratio at runtime. Hence, the practical way adopts fixed temperature control strategy to control the system temperature. However, the fixed temperature control strategy (i.e., deterministic PDTM) cannot adapt to the time-varying temperature behavior on the NoC system at runtime. Hence, the conventional deterministic PDTM still suffers from system throughput degradation because it cannot find the proper throttling ratio at runtime.

To determine the proper temperature control strategy by considering the real-time temperature situation on the NoC system, we employ the RL method in this work. The reason is that the RL method aims to find the optimal action based on the current environment status [25]. Among the different RL methods, we employ the Q-learning algorithm to realize the RL method because it is model-free. Because the temperature and throttling actions between different nodes affect each other, in this work, each NoC node has an agent to select the throttling ratio individually for the purpose of distributed thermal management based on the information in the embedded Q-table. In this work, the inputs of each NoC node's RL-based dynamic temperature control process are the information on the temperature and throttling status of the nodes in the temperature-influence window. To select the proper action (i.e., the adopted throttling ratio), we consider the information about the predicted temperature and the current temperature as the inputs for querying the involved RL-based temperature control process. Fig. 4 illustrates an example to show the proposed RL-based dynamic temperature control process in Node N_{11}, and the other NoC nodes have an identical process to control their temperature separately.

In this work, the Q-table in Fig. 4 records the Q value to determine the throttling ratio (i.e., 0%, 50%, and 100%). The proper action means the corresponding Q value is the biggest according to different states [i.e., arg max $Q(State, Action)$], which indicates the bold red-underlined Q value in Fig. 4. Note that the state represents the predicted temperature [i.e., $T(t+\Delta t_s)$], and the action means the possible throttling ratio. Obviously, the Q-table size depends on the number of states in the Q-table (i.e., the number of entries of the Q-table). Because the involved PDTM is used to prevent the system temperature from overheating (i.e., the system temperature

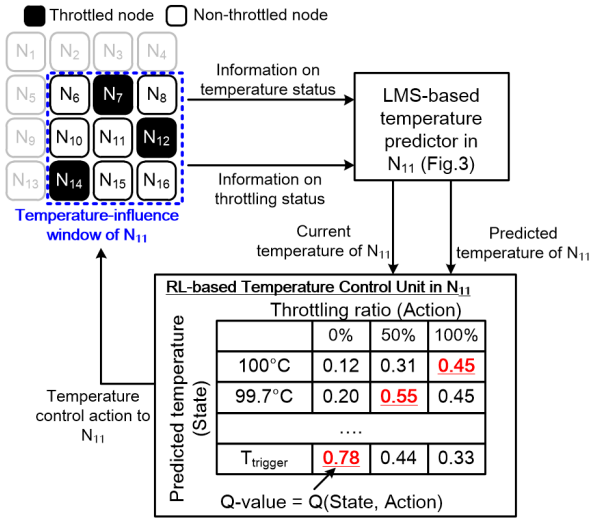

Fig. 4. Proposed RL-based dynamic temperature control process in node N_{11}.

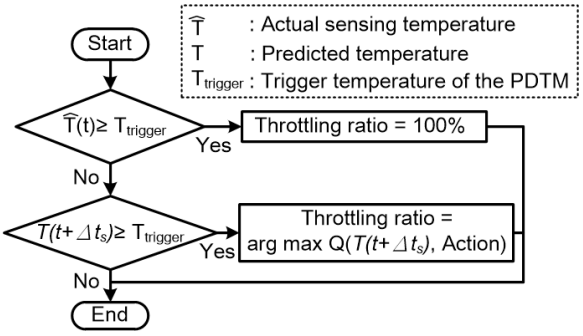

Fig. 5. Flowchart of the PDTM by using the proposed RL-based temperature control process.

exceeds the hard thermal limit), the PDTM is usually triggered as soon as the predicted temperature reaches the triggering temperature. In this work, the upper and lower bound of the Q-table states is set to the hard thermal limit (i.e., 100 °C in this work) of the system and the triggering temperature of the involved PDTM, respectively. The reason is that the Q-table is used to control the involved PDTM. To further decide the interval segment width between each state in Q-table, we align it with the sensing resolution of the involved thermal sensors. In this work, the sensing resolution is set to 0.3 °C by employing the setting of the thermal sensor [21]. Fig. 5 shows the flowchart of the proposed RL-based temperature control process to determine the proper throttling ratio based on the information about the current temperature and the predicted temperature.

As mentioned before, the PDTM agent will select the proper action based on the Q value in the Q-table. Therefore, it is crucial to obtain the Q value in the Q-table, which is trained by using the Q-learning method [17] during the offline training phase. Q-learning is a widely used training method for RL, and the pseudo-code is shown in Fig. 6. Initially, we determine two vectors to record the states and actions. As mentioned before, the upper bound of the state represents the hard thermal limit of the system, and the lower bound is the triggering temperature of the involved PDTM. The actions record the possible strategy (i.e., the involved throttling ratio) to control the system temperature according to the current state. During the training phase of the Q-learning, we iteratively focus on

Algorithm: Q-learning Algorithm
Input: State set X = {1, …, S_m}
Action set A = {1, .., A_n}
Target Q(s,a), ∀s ∈ X, ∀a ∈ A
Learning rate $\alpha \in [0,1]$
Discount factor $\gamma \in [0,1]$
Reward function R(s,a), ∀s ∈ X, ∀a ∈ A
Output: Converged Q ∈ $\mathbb{R}^{mxn}$
Initialize: Q is a zero matrix

1: **for** each episode **do**
2: Select a state $s \in X$
3: **while** Q is not converged **do**
4: Explore an action a for the current state s to maximize the Q value (i.e., $a = \arg\max\ Q(s,a)$)
5: $Q(s,a) = Q(s,a) + \alpha \times (R + \gamma \times \max Q(s',a') - Q(s,a))$, where s' is the next state and a' ∈ A
6: Set the next state s' as the current state s (i.e., $s = s'$)
7: **end while**
8: **end for**

Fig. 6. Pseudo-code of the Q-learning algorithm.

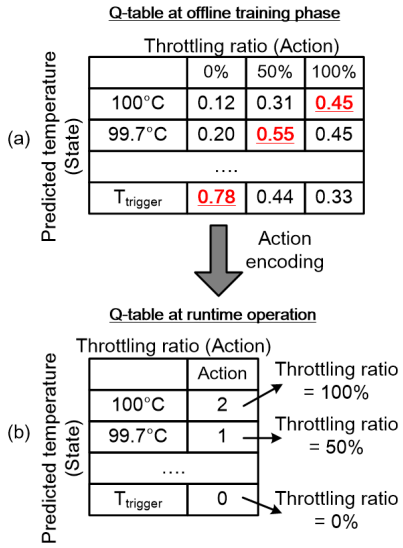

Fig. 7. (a) Q-table is trained at offline time and (b) proper actions are encoded for the runtime operation to reduce the Q-table size.

our target state s and select an action a, which leads to the largest Q value (i.e., $Q(s, a)$ in Fig. 6). The Q value means the reward to the current state as long as selecting the action. Therefore, the proper action to the current state is to select the action, which causes the largest Q value.

In addition to the current adopted action, the Q value depends on the reward at the next state. After selecting an action, we apply the Q-learning training algorithm [23] to update the Q value by using

$$Q(s,a) = Q(s,a) + \alpha \times (R(s,a) + \gamma \times \max Q(s',a') - Q(s,a)) \tag{8}$$

where α is the learning rate; γ is the discount factor; $R(s,a)$ is the reward function. The learning rate determines the extent to replace the old information with new information. In contrast, the discount factor represents the importance of the future reward. To define the reward function $R(s,a)$ in (8), we determine the reward output based on the current node temperature, which can be formulated as

$$R(s,a) = \begin{cases} -1, & \text{overheated} \\ \text{Throughput of the node}, & \text{non-overheated.} \end{cases} \tag{9}$$

Because the Q-table is trained at the offline training phase, the transient status of each NoC node (e.g., throughput and temperature) can be recorded via high-level system simulation. With a precise power model like Intel TeraFlop 80-core processor [27], we can estimate each NoC node's transient temperature for the Q-table training. In contrast, the transient throughput can be obtained statistically through a high-level system simulator [26]. With this transient information on node status, the Q value can be updated by using (8) and (9) until the whole Q-table is converged.

The hardware cost is considerable if each NoC node has its owned Q-table. To reduce the hardware cost, we only select and encode the proper action according to different states in the Q-table to design the involved Q-table at runtime. Fig. 7 shows an example. We encode the fully throttling action to 2; half throttling action to 1; nonthrottling action to 0. Then, the agent uses this encoded Q-table to control each NoC node's temperature at runtime. Obviously, the hardware cost of the Q-table can be reduced significantly. Besides, the complexity of action searching is O(1), which fits the criteria of real-time temperature control policy.

B. Adaptive RL-Based Temperature Control Process

Although the RL-based temperature control process can suggest a proper throttling ratio to control the system temperature, the deterministic temperature control policy cannot adapt to different workloads on NoC systems. The reason is that the recorded proper action in the involved Q-table is based on the fixed Q value, which is determined based on the simulation results under a certain workload. Therefore, the fixed temperature control policy cannot adapt to high-diverse temperature behavior on NoC systems at runtime. The improper temperature control policy further causes a significant performance impact.

To solve the aforementioned problem, it is necessary to update the temperature control policy according to the current NoC node temperature behavior. Therefore, we propose a dynamic temperature control policy adjustment to adapt to the current temperature situation in this work, and the flowchart is shown in Fig. 8. If the current NoC node temperature still becomes higher after selecting the temperature control policy suggested by Q-table, it means that the adopted temperature control strategy at the previous sensing time is not effective. Therefore, the controlling level of thermal management should be raised. On the contrary, the selected temperature control strategy may be too strict if the current NoC node temperature drops significantly, which may result in a severe performance impact. Consequently, the controlling level of thermal management should be eased. Because each temperature control policy (i.e., the action in the Q-table) is encoded to 0, 1, or 2, the temperature control policy can be updated by using a simple addition or subtraction at runtime. Note that the current NoC node temperature can be obtained either through physical embedded sensors [3] or by estimating it using temperature sensing data from other NoC nodes [4] at runtime.

Obviously, the temperature controlling level will be switched frequently if we only compare the temperature

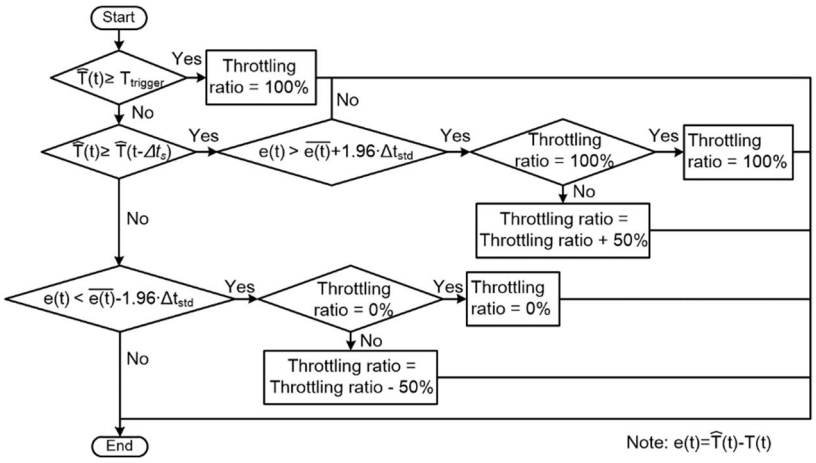

Fig. 8. Proposed dynamic temperature control policy adjustment for the adaptive RL-based temperature control process.

difference (TD) between the current temperature and the temperature at the previous sensing time. To avoid this problem, we aim to control the temperature prediction error [i.e., the difference between the predicted temperature and the current temperature, $e(t)$ in (6)] under consideration of a 95% confidence level, which is usually adopted in statistic theory [24]. Therefore, the probability distribution of the $e(t)$ can be defined as

$$\text{Prob}(\overline{e(t)} - 1.96 \cdot \Delta t_{\text{std}} \le e(\text{t}) \le \overline{e(t)} + 1.96 \cdot \Delta t_{\text{std}}) = 0.95 \tag{10}$$

where $\overline{e(t)}$ indicates the average temperature prediction error, and Δt_{std} is the standard deviation of the temperature error. In this way, the current temperature controlling level can be updated only when the $e(t)$ is outside the confidence interval in (10), which helps to avoid frequent temperature controlling level updating at runtime. Note that the information about the $\overline{e(t)}$ and the Δt_{std} can be obtained in the offline training phase.

V. Experimental Results and Discussion

A. Efficiency Analysis of the Proposed ASLP-Based Temperature Prediction

To evaluate the proposed adaptive ML-based PDTM, we modify a thermal-traffic co-simulator [26] to simulate synthetic traffic patterns and the traffic flow in real applications on an 8-by-8 NoC system. By using this modified thermal-traffic co-simulation, each NoC node's temperature is estimated based on the power model of the Intel TeraFlop 80-core processor [27]. In addition, we use the MCSL benchmark [28] to simulate the realistic NoC traffic patterns. The MCSL benchmark considers both communication and computation requirements of real-world applications and provides detailed communication traces between each NoC node. In this way, the simulation results consider the computing and temperature behaviors of each NoC node. In this work, we set the buffer depth of each router is 8 flits without virtual channels. To evaluate the efficiency of different dynamic thermal management methods and simplify the routing problem, the *XY* routing algorithm is adopted to deliver packets in this work. Because the *XY* routing algorithm is deadlock-free routing, we can ensure every packet can be delivered successfully [29]. Similar to [30], the temperature sampling period is set to 10 ms in this work. In the following experiments, we consider three synthetic traffic patterns (i.e., uniform random, transpose-1, and hotspot) and nine real traffic patterns (i.e., FFT, LDPC-802.11n, LDPC-802.16e, Fppp, Robot, RS-32_28_8_enc, RS-32_28_8_dec, Sparse, and H264-720p_dec) to analyze the efficiency of the proposed ASLP-based temperature prediction method. The uniform random traffic pattern means the source-destination pairs are randomly assigned. Regarding the transpose-1 traffic pattern, a source node (i, j) only sends packets to the destination node (7-j, 7-i) in an 8-by-8 NoC system where the range of i and j are both between 0 and 7. At last, the hotspot traffic pattern refers to a scenario where some particular hotspot NoC nodes have a higher packet injection rate (PIR) than other normal NoC nodes. In this section, we aim to compare the temperature prediction efficiency between different temperature prediction models. In this work, we implement three different temperature prediction models: 1) *RC*-based temperature prediction model [9], 2) second derivative-based thermal predictive model [10], and 3) linear regression-based temperature prediction model [16].

Fig. 9 shows the comparison results between the actual temperature and predicted temperature under the different traffic patterns using different approaches. Compared with the related works, the proposed ASLP-based temperature prediction approach can achieve more precise temperature prediction results. The reason is that conventional approaches usually require physical parameters or collect enough data to predict the temperature. However, these temperature-sensitive physical parameters cannot efficiently adapt to the temperature behavior at runtime. Besides, collecting enough data to represent every kind of temperature behavior is challenging. To solve these problems, the proposed ASLP-based temperature prediction method is able to adapt to the temperature behavior by adjusting the involved parameters for temperature prediction. Therefore, the proposed ASLP-based temperature prediction method can reduce the temperature prediction error.

Fig. 10 shows the comparison of average errors and maximum errors by using the proposed ASLP-based temperature prediction method and the related works under the synthetic traffic patterns and real traffic patterns. Compared with conventional temperature prediction methods, the proposed

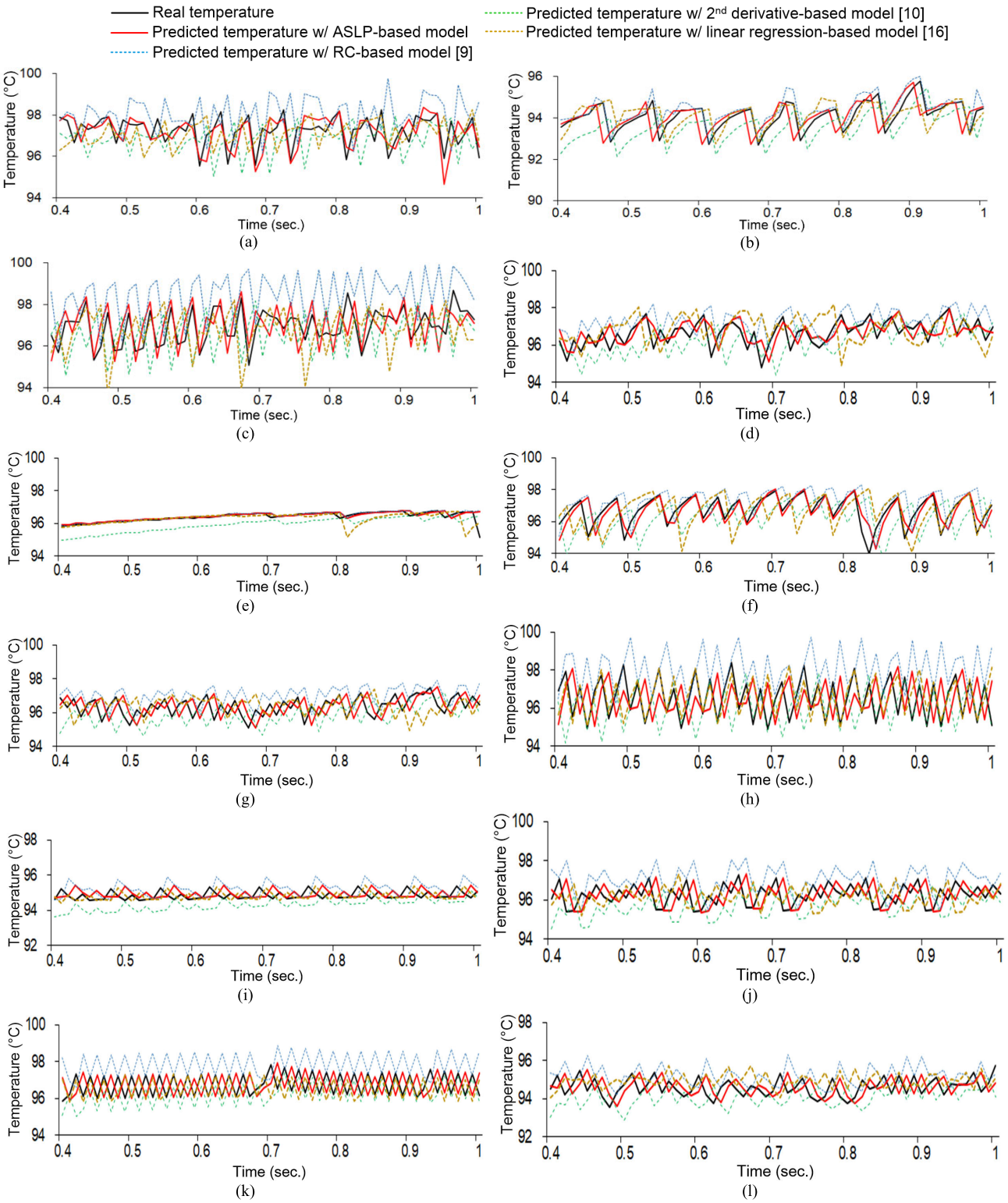

Fig. 9. Comparison of predicted temperature under (a) uniform random, (b) transpose-1, (c) hotspot, (d) FFT, (e) LDPC802_11n, (f) LDPC_802_16e, (g) Fppp, (h) Robot, (i) RS-32_28_8_enc, (j) RS-32_28_8_dec, (k) Sparse, and (l) H264-720p_dec traffic patterns.

approach reduces the average error of 0.2%–78.0% and the maximum error of 0.6%–74.1% over the related works. Note that we consider every NoC node's temperature to calculate the maximum and average errors in this work. As mentioned before, the proposed ASLP-based temperature prediction method can adjust the involved weights for temperature forecasting computing based on current temperature behavior. Therefore, the proposed ASLP-based temperature prediction method can adapt to different traffic patterns as well.

B. Evaluation of System Performance

To evaluate the system performance with our proposed adaptive ML-based PDTM, similar to [9], the distributed

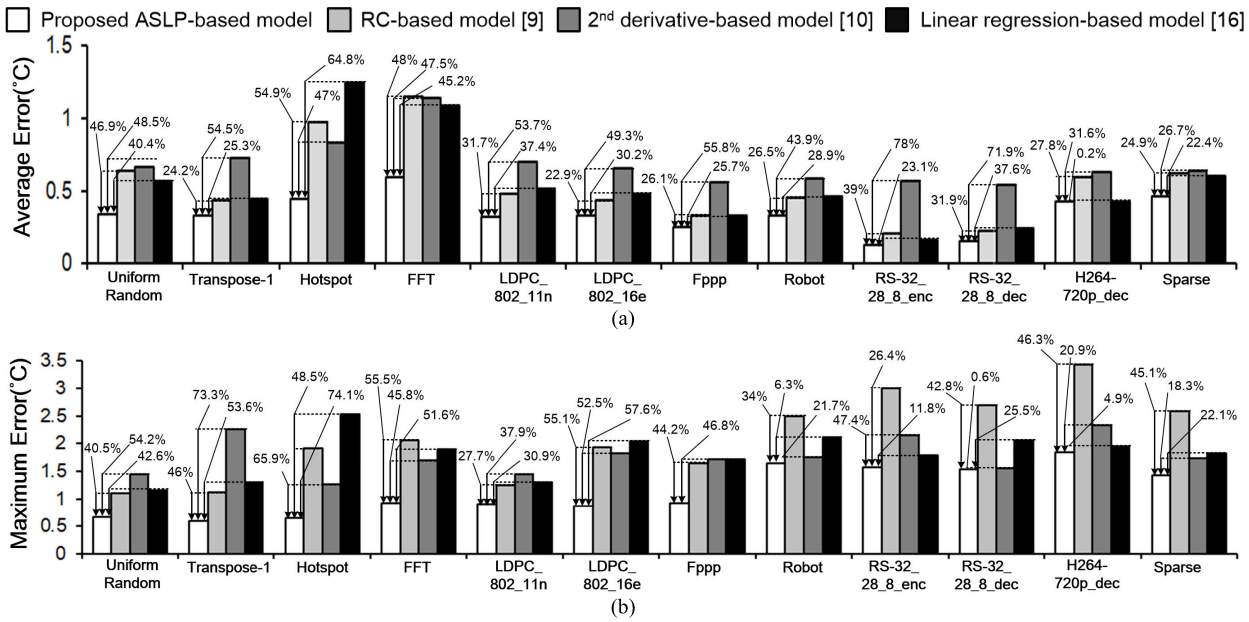

Fig. 10. Comparison of (a) the average and (b) maximum temperature prediction error.

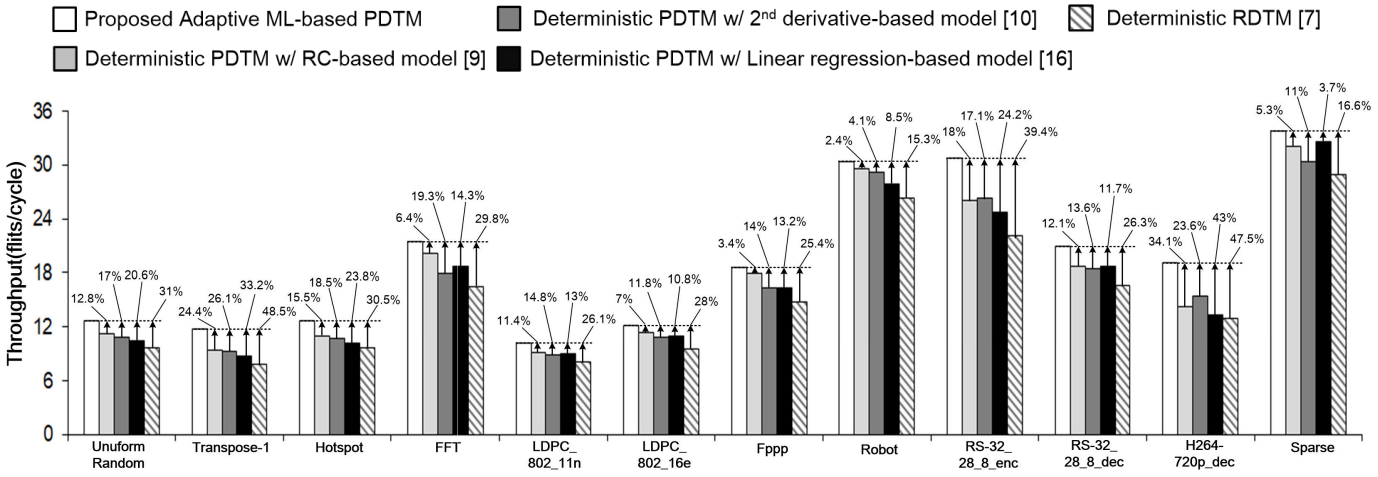

Fig. 11. Comparison of throughput between different approaches under different traffic patterns.

PDTM is adopted in this work. In other words, only the potential thermal-emergent NoC nodes (i.e., the predicted temperature of these NoC nodes exceeds the trigger temperature of the involved PDTM) are throttled to cool down the node temperature. In the following experiments, we implement the deterministic RDTM approach [7] as the baseline approach. Besides, three different deterministic PDTM approaches [9], [10], [16] are implemented as well. The three deterministic PDTM approaches use 1) *RC*-based temperature prediction model [9], 2) second derivative-based thermal predictive model [10], and 3) linear regression-based temperature prediction model [16], respectively. The deterministic approaches predetermine the temperature control strategy, which is fixed at runtime, based on the temperature information. At last, regarding the proposed adaptive ML-based PDTM, we set the learning rate α to 0.9 and the discount factor γ to 0.7 for the Q-learning algorithm in Fig. 6 as a design example in this work.

Fig. 11 shows the saturation throughput comparison under different traffic patterns. The definition of saturation throughput is where average latency equals to twice of the zero-load latency (i.e., the average packet latency when there is no network congestion) [31]. Because the PDTM approaches to control the system temperature in advance, the PDTM approaches help the system achieve better throughput than the RDTM approach. In contrast, the conventional deterministic PDTM approaches [9], [10], [16] determine the throttling ratio according to the reported predicted temperature, which usually leads to the over-control problem. Instead, the proposed adaptive ML-based PDTM can suggest a proper throttling ratio based on more precise temperature prediction results to control the system temperature efficiently. Hence, the proposed approach helps to improve the throughput by 2.4%–43.0% over related works under different traffic patterns. To analyze the efficiency of the proposed approach comprehensively, the throughput comparison according to different PIR is shown in Fig. 12. The experimental results show that the proposed approach still demonstrates the advantage and improves the throughput according to different PIR. The reason is that the conventional deterministic temperature control strategies cannot adapt to the time-varying temperature behavior at runtime.

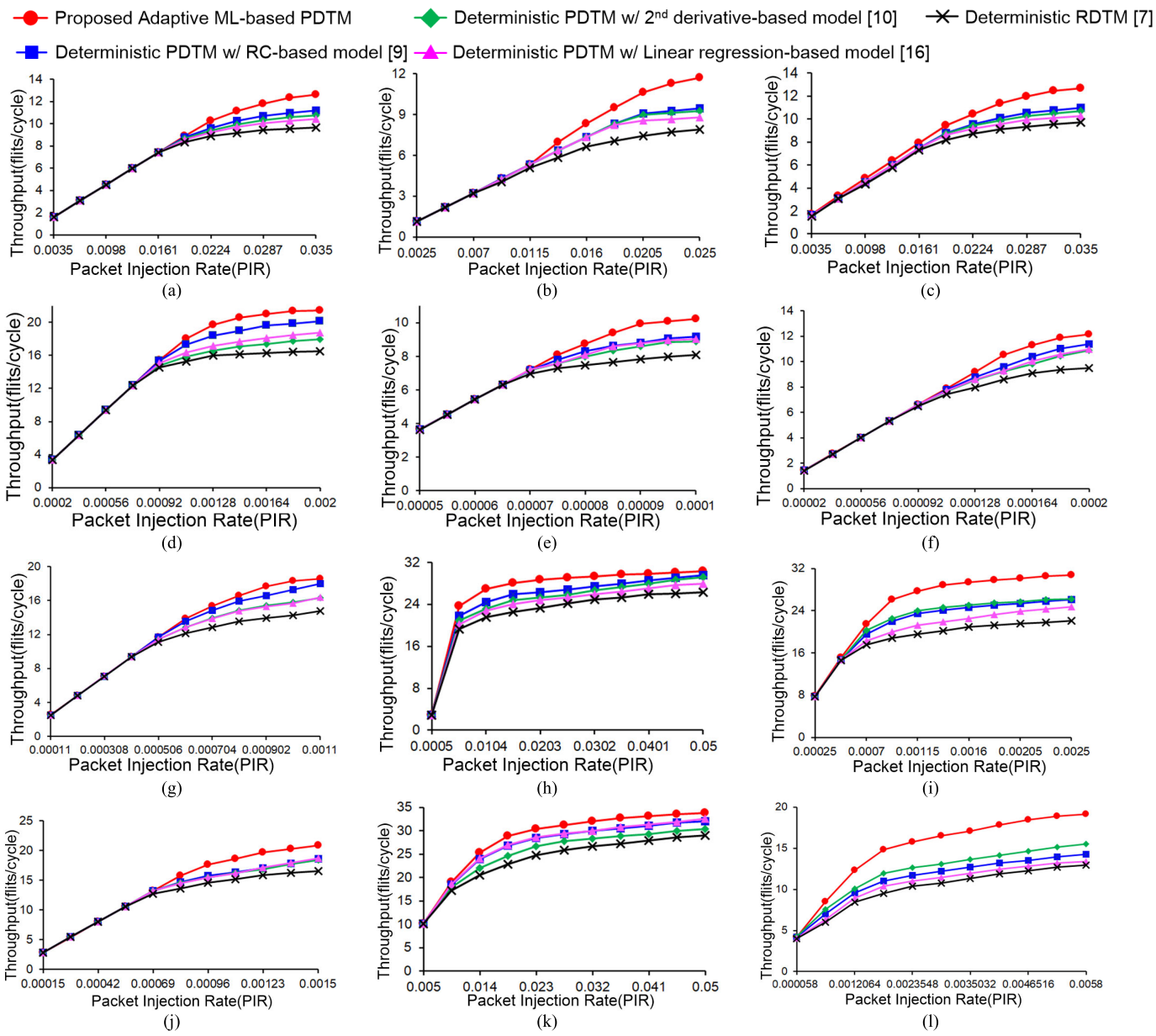

Fig. 12. Comparison of throughput according to different PIR under (a) uniform random, (b) transpose-1, (c) hotspot, (d) FFT, (e) LDPC802_11n, (f) LDPC_802_16e, (g) Fppp, (h) Robot, (i) RS-32_28_8_enc, (j) RS-32_28_8_dec, (k) Sparse, and (l) H264-720p_dec traffic patterns.

VI. Architecture Design and Implementation

The architecture of the proposed adaptive ML-based PDTM unit is shown in Fig. 13, which includes as follows:

1) *ASLP-Based Temperature Predictor:* It computes the predicted temperature of the current NoC node by using the proposed ASLP method.
2) *Adaptive RL-Based Action Decision Maker:* It is used to provide the suggested action to control the temperature of the current NoC node based on the information about the current temperature and predicted temperature.

In this section, the corresponding hardware architecture designs and the hardware cost will be discussed.

A. Architecture of the Proposed Adaptive ASLP-Based Temperature Prediction Unit

The ASLP-based temperature predictor can be further divided into the SLP computation unit and the LMS-based weight updater. Regarding the SLP computation unit, it is composed of a multiplication array and an accumulator. The multiplication array is used to do the operation of multiplying the input data (i.e., the $\text{Temp}_0(t)$ to $\text{Temp}_8(t)$ and the $\text{Throt}_0(t)$ to $\text{Throt}_8(t)$ in Fig. 2) with the weights (i.e., the $w_0(t)$ to $w_{17}(t)$ in Fig. 2). Afterward, the accumulator is used to accumulate the results from the multiplication array. Because we use ReLU as the action function and the chip temperature is always higher than zero, the ReLU function can be eliminated in this architecture to reduce the hardware overhead. The architecture of the SLP computation unit is shown in Fig. 14.

With the results of the temperature prediction calculation, the LMS-based weight updater is used to adjust the involved weights by using (6) and (7). First, the temperature estimation error is calculated by subtracting the predicted temperature $T(t)$ from the current real temperature $\hat{T}(t)$. Afterward, the result should be multiplied by the step-size parameter μ

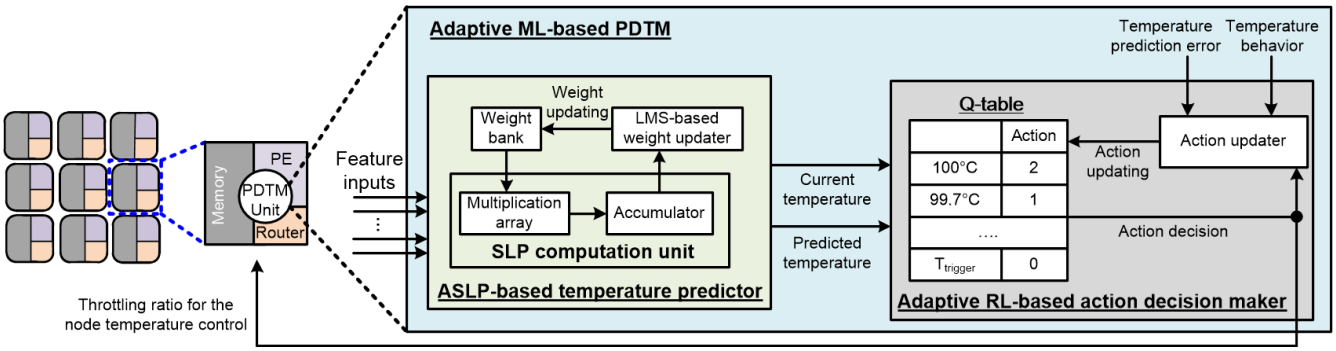

Fig. 13. Architecture of the proposed adaptive ML-based PDTM.

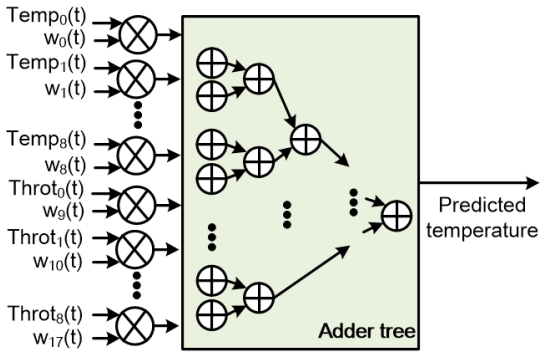

Fig. 14. Architecture of the SLP computation unit.

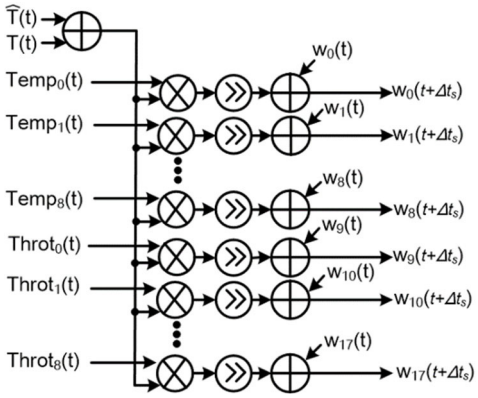

Fig. 15. Architecture of the LMS-based weight updater.

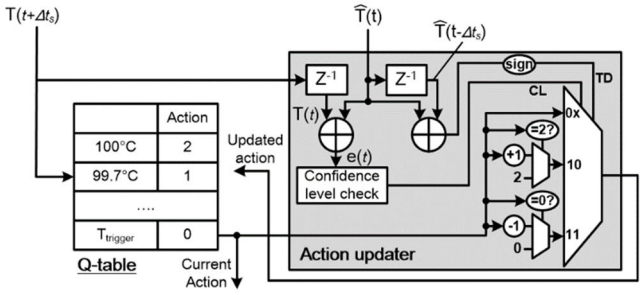

Fig. 16. Architecture of the adaptive RL-based action decision-maker.

TABLE I
ACTION UPDATING POLICY TABLE

CL	TD	Updated Action
0	X	Hold the current action because the temperature prediction error is at the 95% confidence level.
1	0	Upgrade the current action because the temperature prediction error is lower than the 95% confidence level. Besides, the temperature is not controlled well.
1	1	Downgrade the current action because the temperature prediction error is lower than the 95% confidence level. Besides, the temperature is over-controlled.

(i.e., the sensing period Δt_s in this work) and added with the corresponding weights by using (7). Because the sensing period Δt_s is 10 ms in this work, it is easy to approximate this value by shifting 6 bits right. Therefore, the second term [i.e., $\mu e(t)T_{\text{input}}(t)$] in (7) can be calculated by shifting the product of current temperature $T_{\text{input}}(t)$ in (7) and temperature estimation error $e(t)$ in (7) right by 6 bits. Finally, the updated weights can be obtained by adding the shift-right result and the currently involved weights. The architecture of the LMS-based weight updater is shown in Fig. 15.

B. Architecture of the Proposed Adaptive RL-Based Action Decision Maker

As mentioned before, the adaptive action decision-maker is used to adjust the predecided action (i.e., throttling ratio) to achieve the goal of precise temperature control. By following the flowchart of the dynamic temperature control policy adjustment in Fig. 8, the architecture of the adaptive RL-based action decision-maker is shown in Fig. 16. First, the temperature prediction error $e(t)$ in (7) is evaluated and check the confidence level. Besides, the current temperature behavior trend (i.e., temperature increase or decrease) can be determined by checking the sign bit of the TD between the current temperature $\hat{T}(t)$ and the temperature at the previous sensing time $\hat{T}(t - \Delta t_s)$. Based on the information on the current TD and the confidence level (CL) of the temperature prediction error, the two control signals (i.e., CL and TD in Fig. 16) are used to determine the updated action. One MUX is adopted to select the proper updated action.

Table I shows the control policy table to determine the updated action. If the CL is equal to 0, it means that the confidence level of the current temperature prediction error is at 95%, which fits the criteria in (10). Therefore, the current action is preserved for the next temporary control period. In contrast, if the CL is equal to 1, it presents that the current temperature prediction error is lower than the target confidence interval in (10) (i.e., 95% in this work). Hence, the adopted action may be upgraded while the current temperature trend is up (i.e., the TD is equal to 0). Otherwise, the downgraded action may be considered if the temperature trend is down (i.e., the TD is equal to 1).

TABLE II
COMPARISON OF THE HARDWARE COST AND HARDWARE EFFICIENCY

	RC-based PDTM [9]	Second derivative based PDTM [10]	Linear regression based PDTM [16]	Proposed adaptive ML-based PDTM
Area (Unit: μm^2)				
Router	45,015	45,015	45,015	45,015
Temperature predictor*	41,925	49,067	44,812	21,487
LMS-based weight updater*	-	-	-	17,514
Adaptive RL-based action decision maker*	-	-	-	219
Processing element	9,728	9,728	9,728	9,728
Total area	96,668	103,810	99,555	93,963
Power (Unit: mW)				
Router	12.83	12.83	12.83	12.83
Temperature predictor*	10.28	13.24	11.16	4.02
LMS-based weight updater*	-	-	-	3.65
Adaptive RL-based action decision maker*	-	-	-	0.02
Processing element	0.66	0.66	0.66	0.66
Total power	23.77	26.73	24.65	21.18
Hardware efficiency	7.65	6.14	6.86	9.81

*: hardware overhead over the baseline design

C. Hardware Implementation and Analysis

To analyze the hardware cost and efficiency of the proposed adaptive ML-based PDTM approach, we implement it and three other related works with the TSMC 40-nm process technology. Besides, we implement the baseline router to support *XY* routing [32] and the Intel TeraFlop processor [27] to process data. In this work, we assume the hard thermal limit is 100 °C. Because the involved weights may be negative, we use a 16-bit fixed-point operation (i.e., 8-bit signed integer and 8-bit fraction) to implement every work for a fair comparison.

Table II compares the hardware cost comparison between the proposed approach and other related works to analyze the area overhead and power consumption. Compared with the baseline design, which only includes the router and PE, the hardware overhead supporting PDTM is the additional designs beyond the router and PE architectures (i.e., temperature predictor, LMS-based weight updater, and adaptive RL-based action decision maker in Table II). In contrast, the proposed approach includes additional hardware components for the LMS-based weight updater and adaptive RL-based action decision-maker compared with the related works supporting PDTM. Regarding the proposed ASLP-based temperature predictor, it adopts a single neuron to forecast the temperature and a low-cost LMS-based adaptive filter. Besides, the involved adaptive RL-based action decision-maker only employs low-cost multiplexers and a simplified encoded Q-table. Hence, our proposed approach can reduce 3%–9% area overhead and 11%–21% power consumption compared with other related works. The reason is that the related works usually require a high-complex exponential operation. In addition to the hardware analysis, we consider hardware efficiency, which is defined as

$$\text{Hardware efficiency} = \frac{\text{Throughput}}{\text{Area} \times \text{Power}} \times 10^6. \tag{11}$$

Because the proposed adaptive ML-based PDTM predicts the future temperature precisely and provides a proper temperature control policy, the proposed approach improves the hardware efficiency by 28.1%–59.7% over the related works, as shown in Table II.

VII. CONCLUSION

In this work, we propose an adaptive ML-based PDTM to control the temperature of NoC systems. First, the proposed ASLP-based temperature is used to predict the future temperature of the system. Based on the involved LMS-based adaptive filter theory, the weights for the temperature prediction can be adjusted at runtime based on the temperature prediction error. Therefore, the proposed ASLP-based temperature prediction model helps to reduce the 0.2%–78.0% average error and 0.6%–74.1% maximum error. With precise information about the predicted temperature, we further employ the Q-learning method to propose an adaptive RL method to find the proper throttling ratio to control the system temperature. The experimental results show that the proposed approach helps to improve the system throughput by 2.4%–43.0% with smaller hardware overhead over the related works.

REFERENCES

[1] H. Zheng, K. Wang, and A. Louri, "Adapt-NoC: A flexible network-on-chip design for heterogeneous manycore architectures," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Feb. 2021, pp. 723–735.

[2] K. Chen, "Game-based thermal-delay-aware adaptive routing (GTDAR) for temperature-aware 3D network-on-chip systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 9, pp. 2018–2032, Sep. 2018.

[3] *Quad-Core Intel Xeon Processor 5400 Series Datasheet*, Intel Co., Santa Clara, CA, USA, Aug. 2008.

[4] K. Chen, H. Tang, C. Wu, and C. Chen, "Thermal sensor placement for multicore systems based on low-complex compressive sensing theory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 11, pp. 5100–5111, Nov. 2022.

[5] C.-H. Chao, K.-Y. Jheng, H.-Y. Wang, J.-C. Wu, and A.-Y. Wu, "Traffic- and thermal-aware run-time thermal management scheme for 3D NoC systems," in *Proc. 4th ACM/IEEE Int. Symp. Netw.-on-Chip*, May 2010, pp. 223–230.

[6] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Proc. 33rd Int. Symp. Comput. Archit. (ISCA)*, Jun. 2006, pp. 77–88.

[7] L. Shang, L. Peh, A. Kumar, and N. K. Jha, "Thermal modeling, characterization and management of on-chip networks," in *Proc. 37th Int. Symp. Microarchitecture (MICRO)*, Dec. 2004, pp. 67–68.

[8] T. Wegner, M. Gag, and D. Timmermann, "Impact of proactive temperature management on performance of networks-on-chip," in *Proc. Int. Symp. Syst. Chip (SoC)*, Oct. 2011, pp. 116–121.

[9] K. Chen, E. Chang, H. Li, and A. Wu, "RC-based temperature prediction scheme for proactive dynamic thermal management in throttle-based 3D NoCs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 1, pp. 206–218, Jan. 2015.

[10] Z. Lei et al., "A predictive thermal model for multiprocessor system-on-chip," *Proc. World Congr. Eng. Comput. Sci.*, Oct. 2015, pp. 27–32.

[11] K. Chen, H. Li, and A. A. Wu, "LMS-based adaptive temperature prediction scheme for proactive thermal-aware three-dimensional network-on-chip systems," in *Proc. Tech. Papers Int. Symp. VLSI Design, Autom. Test*, Apr. 2014, pp. 1–4.

[12] G. Liu, J. Park, and D. Marculescu, "Dynamic thread mapping for high-performance, power-efficient heterogeneous many-core systems," in *Proc. IEEE 31st Int. Conf. Comput. Design (ICCD)*, Oct. 2013, pp. 54–61.

[13] Z. Chen and D. Marculescu, "Distributed reinforcement learning for power limited many-core system performance optimization," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2015, pp. 1521–1526.

[14] J. M. N. Abad and A. Soleimani, "Novel feature selection algorithm for thermal prediction model," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 10, pp. 1831–1844, Oct. 2018.

[15] P. L. Feintuch, "An adaptive recursive LMS filter," *Proc. IEEE*, vol. 64, no. 11, pp. 1622–1624, Nov. 1976.

[16] E. W. Wächter, C. de Bellefroid, K. R. Basireddy, A. K. Singh, B. M. Al-Hashimi, and G. Merrett, "Predictive thermal management for energy-efficient execution of concurrent applications on heterogeneous multicores," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 6, pp. 1404–1415, Jun. 2019.

[17] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[18] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, Apr. 2009, pp. 423–428.

[19] Y. Zhang and A. Srivastava, "Accurate temperature estimation using noisy thermal sensors for Gaussian and non-Gaussian cases," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 9, pp. 1617–1626, Sep. 2011.

[20] K. Chen, H. Tang, Y. Liao, and Y. Yang, "Temperature tracking and management with number-limited thermal sensors for thermal-aware NoC systems," *IEEE Sensors J.*, vol. 20, no. 21, pp. 13018–13028, Nov. 2020.

[21] S. Wang and R. Bettati, "Reactive speed control in temperature-constrained real-time systems," *Real-Time Syst.*, vol. 39, nos. 1–3, pp. 73–95, Dec. 2007.

[22] H. Hanson, S. W. Keckler, S. Ghiasi, K. Rajamani, F. Rawson, and J. Rubio, "Thermal response to DVFS: Analysis with an Intel Pentium M," in *Proc. Int. Symp. Low Power Electron. Design*, Aug. 2007, pp. 219–224.

[23] M. A. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, Learn., Optim.*, vol. 12, no. 3, p. 729, 2012.

[24] D. Choudhary and P. K. Garg, "95% confidence interval: A misunderstood statistical tool," *Indian J. Surgery*, vol. 75, no. 5, p. 410, Oct. 2013.

[25] H. van Hasselt and M. A. Wiering, "Reinforcement learning in continuous action spaces," in *Proc. IEEE Int. Symp. Approx. Dyn. Program. Reinforcement Learn.*, Apr. 2007, pp. 272–279.

[26] K.-Y. Jheng, C.-H. Chao, H.-Y. Wang, and A.-Y. Wu, "Traffic-thermal mutual-coupling co-simulation platform for three-dimensional network-on-chip," in *Proc. Int. Symp. VLSI Design, Autom. Test*, Apr. 2010, pp. 135–138.

[27] Y. Hoskote, "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sep. 2007.

[28] W. Liu et al., "A NoC traffic suite based on real applications," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Jul. 2011, pp. 66–71.

[29] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 7, pp. 729–738, Jul. 2000.

[30] C.-H. Chao, K.-C. Chen, T.-C. Yin, S.-Y. Lin, and A.-Y. Wu, "Transport-layer-assisted routing for runtime thermal management of 3D NoC systems," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 1, pp. 1–22, Aug. 2013.

[31] L. Shang et al., "PowerHerd: Dynamic satisfaction of peak power constraints in interconnection networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, pp. 92–110, 2006.

[32] M. Ahn and E. J. Kim, "Pseudo-circuit: Accelerating communication for on-chip interconnection networks," in *Proc. 43rd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Jan. 2011, pp. 399–408.

[33] R. G. Kim et al., "Imitation learning for dynamic VFI control in large-scale manycore systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 9, pp. 2458–2471, Sep. 2017.

Kun-Chih (Jimmy) Chen (Senior Member, IEEE) is currently an Associate Professor and an Electric Junior Chair Professor at the Institute of Electronics, National Yang Ming Chiao Tung University (NYCU), Hsinchu, Taiwan. His research interests include MPSoC design, neural network learning algorithm design, reliable system design, and VLSI/CAD design.

Dr. Chen served as a Technical Program Committee (TPC) Chair and a General Chair of the International Workshop on Network on Chip Architectures (NoCArc) in 2018 and 2019. Besides, he was a Guest Editor of IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS (JETCAS). He received the IEEE Tainan Section Best Young Professional Member Award, TCUS Young Scholar Innovation Distinction Award, Exploration Research Award of Pan Wen Yuan Foundation, the Taiwan IC Design Society Outstanding Young Scholar Award, and the CIEE Outstanding Youth Electrical Engineer Award. He is an ACM member.

Yuan-Hao Liao received the B.S. degree in computer science and information engineering from the National Changhua University of Education, Changhua City, Taiwan, in 2018, and the M.S. degree from the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, in 2020.

His research fields interest in the thermal-aware multicore system design.

Mr. Liao was a recipient of the Best Student Paper Award of ISCAS 2020.

Cheng-Ting Chen received the B.S. degree in applicated mathematics and the M.S. degree in computer science and engineering from the National Sun Yat-sen University, Kaohsiung, Taiwan, in 2020 and 2022, respectively.

His research field interests are in machine learning and stochastic computing.

Lei-Qi Wang received the B.S. degree from the Department of Computer and Communication, National Pingtung University, Pingtung City, Taiwan, in 2021. She is currently working toward the M.S. degree at the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan.

Her research field interests are in thermal sensor placement for multicore system.