HESSLE-FREE: Heterogeneous Systems Leveraging Fuzzy Control for Runtime Resource Management

KASRA MOAZZEMI, BISWADIP MAITY, SAEHANSEUL YI, AMIR M. RAHMANI, and NIKIL DUTT, University of California, Irvine

As computing platforms increasingly embrace heterogeneity, runtime resource managers need to efficiently, dynamically, and robustly manage shared resources (e.g., cores, power budgets, memory bandwidth). To address the complexities in heterogeneous systems, state-of-the-art techniques that use heuristics or machine learning have been proposed. On the other hand, conventional control theory can be used for formal guarantees, but may face unmanageable complexity for modeling system dynamics of complex heterogeneous systems. We address this challenge through HESSLE-FREE (Heterogeneous Systems Leveraging Fuzzy Control for Runtime Resource Management): an approach leveraging fuzzy control theory that combines the strengths of classical control theory together with heuristics to form a light-weight, agile, and efficient runtime resource manager for heterogeneous systems. We demonstrate the efficacy of HESSLE-FREE executing on a NVIDIA Jetson TX2 platform (containing a heterogeneous multi-processor with a GPU) to show that HESSLE-FREE: 1) provides opportunity for optimization in the controller and stability analysis to enhance the confidence in the reliability of the system; 2) coordinates heterogeneous compute units to achieve desired objectives (e.g., QoS, optimal power references, FPS) *efficiently* and with *lower complexity*, and 3) eases the burden of system specification.

CCS Concepts: • **Computer systems organization → System on a chip**;

Additional Key Words and Phrases: Fuzzy control, heterogeneous systems, runtime resource management, heterogeneous multi-core processor, MIMO control, DVFS, power management, QoS

ACM Reference format:
Kasra Moazzemi, Biswadip Maity, Saehanseul Yi, Amir M. Rahmani, and Nikil Dutt. 2019. HESSLE-FREE: Heterogeneous Systems Leveraging Fuzzy Control for Runtime Resource Management. *ACM Trans. Embed. Comput. Syst.* 18, 5s, Article 74 (October 2019), 19 pages.
https://doi.org/10.1145/3358203

1 INTRODUCTION

Modern processors are designed to support diverse workloads that exhibit various characteristics such as memory-bound, compute-bound, or a mixture across applications executing in parallel.

This article appears as part of the ESWEEK-TECS special issue and was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2019. This work was partially supported by NSF grant CCF-1704859.

Authors' addresses: K. Moazzemi, B. Maity, S. Yi, A. M. Rahmani, and N. Dutt, University of California, Irvine, Irvine, CA 92697; emails: {moazzemi, maityb, saehansy, a.rahmani}@uci.edu, dutt@ics.uci.edu.

1539-9087/2019/10-ART74 $15.00
https://doi.org/10.1145/3358203

These applications demand various resources often with conflicting constraints. These issues in homogeneous architectures create a challenge in the design and implementation of resource managers for these systems (e.g, task mapping, scheduling, core configuration, etc.) [15, 20, 55, 78]. This problem exacerbates in Heterogeneous Multi-Processor (HMPs) where multiple heterogeneous compute elements are integrated on the same chip. ARM big.LITTLE architecture [3] is one case of widely-used HMPs where high-performance big cores work alongside low-power LITTLE cores with the same instruction set architecture (ISA). This opens the opportunity to switch between multiple objectives such as maximizing performance or minimizing energy consumption. Many modern computing platforms leverage the compute power of general purpose GPUs to execute massively parallelized applications or speed up a portion of program execution that can leverage parallelism. Heterogeneous systems with both CPU and GPU compute units require sophisticated yet efficient resource management policies due to these architectural differences in the compute elements and nature of their applications runtime execution.

While heuristics [27, 58, 59, 64, 71, 73, 76, 79] or machine learning (ML) [7, 11, 35, 40, 89] solutions exist to help resource management of these heterogeneous systems, there are scenarios where response of the target system to these resource management mechanisms and its behaviour requires robustness and stability analysis. In such cases, control theoretic approaches [1, 26, 30, 37, 38, 61] can provide formal guarantees in order to increase the robustness of management mechanism and stability of the system in the presence of unpredictable workload variance. But considering the complexity of emerging heterogeneous systems, the conventional approach of deploying a control theoretic controller for modeling a system using system identification misses the opportunity to benefit from the designer's knowledge of system behavior and thus may result in system inefficiencies and high runtime complexity. It is therefore important to design techniques to manage these complex heterogeneous systems both efficiently and robustly.

Traditional Single-Input Single-Output (SISO) controllers such as proportional integral (PI) and proportional integral derivative (PID) have proven capable of tracking design objectives efficiently but come short in the management of emerging computer systems due to their limitation in coordinating multiple goals simultaneously. Multiple-Input Multiple-Output (MIMO) control theory has recently been proposed for runtime resource management of unicore processors [53, 54]. However, MIMO controllers suffer from *i)* scalability issues in multi- and many-core systems and *ii)* controller design challenges when the system is heterogeneous which can lead to possible instability in the system or sluggish response [48]. In order to design a controller, conventional control theory uses an explicit model (obtained either through mathematical analysis or system identification) of a process to be controlled and specifications of the desired closed-loop behavior. The problem arises from the difficult task of modeling and simulation of a complex real-world system such as modern computer systems. In addition, these controllers require an auxiliary component that dictates the tracking references (e.g., tracking certain power consumption or frames per second). This can come from a supervisor, a user, or an optimizer. In runtime management, the need for this optimizer adds another level of complexity in terms of design space exploration and training the optimization algorithm. Another critical, but often overlooked property of classical control theory is that it models the system as a black-box: the controller is designed using system identification theory [43][18], but does not incorporate domain semantics or designer intuition and expertise. While this property eases the task of controller design, it can lead to system inefficiencies in more realistic settings, for instance when multiple knobs with different behaviors are used (e.g., DFVS, idle cycle injector, power gating in a MIMO controller) and the system is heterogeneous (e.g., CPU-GPU or big.LITTLE architectures). Managing complex computing systems with a variety of knobs calls for incorporating designer's expertise in the management scheme to reduce the runtime complexity of the control algorithm and increase its efficiency.

We believe fuzzy control theory can offer an *efficient* and *robust* controller development methodology by leveraging both designer's heuristics and control theory foundation. Fuzzy control theory provides a formal methodology for representing, modifying, and implementing a human's heuristic knowledge about how to control a system. Fuzzy control employs qualitative descriptions of systems so as to make it easier to specify controller actions. Towards this end, we present *HESSLE-FREE*: a fuzzy control based approach for efficient and coordinated management of heterogeneous systems. We demonstrate the utility of HESSLE-FREE on a NVIDIA Jetson TX2 development platform [10] that incorporates a hexa-core HMP with a quad-core ARM cortex A57, a dual-core high performance NVIDIA denver, and a contemporary embedded GPU based on the NVIDIA Pascal architecture. We present experimental results to show the efficacy of HESSLE-FREE toward a light-weight and rapid resource management of heterogeneous architectures using fuzzy control theory. We have implemented a light-weight monitoring system that captures power and performance metrics from each computing unit in the system and makes runtime resource allocation and tuning decisions using fuzzy control theory with low overhead. HESSLE-FREE controls various knobs in this system such as the operating frequency of CPU clusters and GPU separately and decides on the number of active core in HMP. We show that HESSLE-FREE is capable of adapting to various objectives with minimal effort while achieving higher energy efficiency and meeting quality of service (QoS) targets over state-of-the-art heuristics and control theoretic controllers. We evaluate the candidate resource managers on their performance, energy efficiency and tracking accuracy. Our results show that HESSLE-FREE matches or exceeds the state-of-the-art techniques in the majority of the cases and show promising results in managing the complexity and heterogeneity of the system.

The key contributions of this paper are:

- **Inherent optimization:** We demonstrate the inherent benefit of fuzzy control in incorporating optimization as part of the control system in order to better adapt to system dynamics while achieving system objectives.
- **Addressing complexity:** We describe the complexity overhead of classical control theoretic approaches such as MIMO controller in managing resources of complex heterogeneous systems. This is performed through analyzing the imposed computational overhead of such controllers at runtime. We demonstrate the benefits of fuzzy control in managing the complexity of control problems and design optimization techniques to reduce runtime complexity and efficiently manage large heterogeneous systems.
- **Efficiently managing heterogeneity:** We show that the absence of architectural knowledge regarding the nature of heterogeneous elements and their impact on the system status can cause under-utilization of the shared resources or violation in design constraints. We deploy the idea of rule-base inference from fuzzy control to incorporate varying dynamics of heterogeneous compute units (i.e., heterogeneous CPU cores and GPU) in the runtime policies deployed by the resource manager.
- **Experimental case study:** We demonstrate the practical implementation of our approach on a NVIDIA Jetson-TX2 development board as a contemporary SoC showing HESSLE-FREE's ability in distributing power budget across compute units in an energy efficient manner and responsively providing the desired QoS.

The rest of this paper is organized as follows. We position our work with related efforts in Section 2. Section 3 presents the background and motivation for fuzzy control design in computer systems. Section 4 describes HESSLE-FREE as a methodology for the adaptation of fuzzy control in resource management of heterogeneous systems. Section 5 outlines the cases study. Section 6 reports experimental results and Section 7 presents conclusions.

2 RELATED WORK

Runtime resource management techniques have been explored extensively in the literature. These methods have been applied to a variety of computer systems ranging from cloud servers to embedded systems and sensor networks. In this work, we focus on on-chip resource management mechanisms. While some of these works focus on communication subsystem [8, 45, 72], there have been a plethora of approaches for managing computing units and their corresponding objectives (e.g., performance, QoS, power, thermal). In this context, we can classify the resource management approaches in the literature into three main classes: *heuristics*, *machine learning*, and *control theoretic* approaches. Below, we summarize representative efforts and describe their advantages and shortcomings.

Heuristic approaches cover a vast body of work on ad-hoc resource management techniques. Some use *optimization* [21–23, 28, 44, 52, 77] that attempts to minimize/maximize an objective. This optimization can be subject to some constraints. Although these techniques can be a natural choice for simple architectural tuning, they provide no mechanism for a feedback thus being rigid to the addition of learning mechanism. Other approaches might use *model-based heuristics* [4, 13, 16, 36, 85, 88] where decisions are guided using a system model that defines the relation between inputs and possible outputs. These methods lack guarantees or formal methodology making the runtime resource manager prone to errors in particular when encountering corner cases that might lead to system instability.

Machine Learning approaches [6, 12, 17, 25, 33] for runtime resource management have gained lots of traction in the past few years, especially in management of high performance systems and cloud servers. These methods need special tailoring before deployment on embedded and real-time systems in order to reduce their high computational overhead at runtime [32, 50]. Specifically, machine learning techniques have been a promising trend for modeling the complexity of interaction among different on-chip resources and the corresponding effect on resource metrics [24]. Further, these techniques have targeted beyond the conventional fixed single and multi- objective allocation policies, towards dynamically varying goals [35, 69, 70]. Conventional machine learning methods require extensive training to learn the correlation between inputs and outputs of the system. In the case of a new situation at runtime which they were not trained for, they might provide an inaccurate solution. To address this issue, online learning methods [11, 31, 39, 40] show promising results in learning new scenarios at runtime compared to a complete, expensive re-training of the weights and parameters.

Control Theory provides a formal methodology for design, implementation, and testing of a control system. There have been efforts to leverage the robustness of control theoretic approaches in managing computer systems [1, 30, 47, 56, 63, 68, 81, 83, 84, 88]. Conventional control theory such as lead-lag, PI, and PID controllers have proven successful and efficient in managing straightforward problems such as per-core DVFS. Recently, more advanced control theoretic approaches such as state-space MIMO control have been leveraged to manage on-chip resources. Pothukuchi et al. [54] provide a guide for designing MIMO formal controllers for tuning architectural parameters in processors to enhance coordination, and demonstrate the coordinating management of multiple goals for unicore processors [54]. These conventional controllers might be prone to exponential computational overhead and in general being agnostic to control parameters. In [48], the authors demonstrate that current MIMO control approaches suffer from scalability issues due to input/output sizes and infeasibility of dynamic system model identification for large MIMO systems, thus requiring the deployment of multiple controllers to achieve responsiveness. Additionally, conventional controllers such as PID and state-space are designed to track references (i.e., regulatory control). This ability suffices when the system's objective is only to track a certain

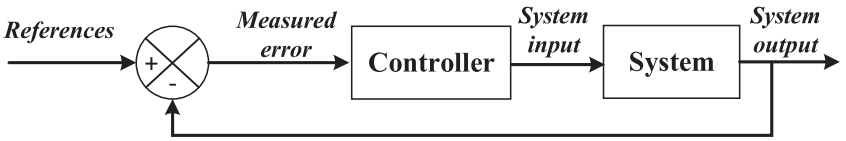

Fig. 1. Overview of Feedback control.

target references, but in many cases, resource management algorithm in computer systems has an optimization objective (e.g., minimizing energy consumption, maximizing performance, etc.). This requires the design of an additional optimizer to set the references for conventional controllers.

Fuzzy Resource Management: There have been a number of approaches to apply fuzzy control theory to resource management in virtual or cloud environment [9, 29, 62, 65, 82]. However, only a few approaches have been proposed to manage on-chip resources. Najam et al. [67] propose fuzzy logic based dynamic voltage and frequency scaling (DVFS) in a quad-core desktop setting. Authors in [49] using fuzzy controller to achieve power efficiency in a mobile scenario. However, they only consider homogeneous cores with a single design objective. Majority of these solutions show partial benefits of the fuzzy control by using SISO type controllers. On the other hand, we extend the usage of the fuzzy controller to HMPs to leverage fuzzy's capabilities in handling heterogeneity and managing systems with multiple inputs and outputs (MIMO). In this work, we propose fuzzy control to reduce the complexity and overhead of the runtime resource manager and to capture dynamic behaviour of the complex heterogeneous systems.

3 BACKGROUND AND MOTIVATION

Feedback control is a form of closed-loop control that can be defined as a management mechanism which regularly monitors a system to make modifications to meet a desired output response. Figure 1 depicts an overview of a feedback control loop. The system/plant[1] in this figure represents the target system managed by the controller. The controller can be designed and deployed using different types of controllers such as proportional-integral-derivative (PID), state space, fuzzy, adaptive control, etc. In the remainder of this section, we skip the background on the well-established PID control; we first describe MIMO controllers as a state-of-the-art control theoretic technique in managing processors and then focus on fuzzy control theory background with an example.

When considering single-input single-output controllers (SISO), proportional-integral (PI) and proportional-integral-derivative (PID) are well-known and widely used in various domains and industries. Given a system requires coordinated resource management of multiple objectives, Multiple-input Multiple-Output (MIMO) controllers can provide rapid response and formal guarantees. The MIMO is implemented using a Linear Quadratic Gaussian (LQG) controller [74]:

$$x_{t+1} = A \times x_t + B \times u_t \tag{1}$$

$$y_t = C \times x_t + D \times u_t \tag{2}$$

where x, y, and u are vectors representing the system state, the measured outputs, and the control inputs, respectively. Coefficient matrices A, B, C, and D capture the system behavior, and their values are obtained through *system identification*. Matrix sizes are determined by both the number of inputs and outputs of the controller as well as the *order* of the controller. Growth in order or number of inputs and outputs for complex systems will increase the size of the MIMO matrices to the point that the size of the controller will grow to unmanageable sizes, complicating the

[1]We interchangeably use the terms system and plants.

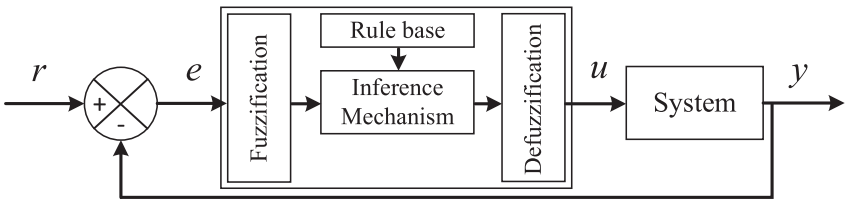

Fig. 2. Overview of Fuzzy control.

controller design through system identification, and impacting the computational complexity of the controller implementation [60].

3.1 Fuzzy Control Background

A fuzzy controller can be viewed as an artificial decision maker that operates in a closed-loop system in real time. Figure 2 depicts a simple fuzzy controller in the context of feedback control. As shown in this figure, system output is sensed and represented by y which is compared against the references set by the designer or the user r. This difference is fed to the fuzzy controller to decide system input(s) u (i.e., control output(s)) to guide the system towards the desired goals. The fuzzy controller consists of four main components: (1) the fuzzification component that interprets the inputs to be matched with the rules, (2) the rule base which is a set of rules that defines the knowledge on how to control the system in different situations, (3) the inference mechanism which matches the rules with the current situation and determines the fuzzy set for control outputs, and (4) the defuzzification component which converts the output of the inference to actual actuator values[2].

Before describing each component in detail, we cover the terminology used in fuzzy control. To specify rules in the rule-base, an expert uses a *linguistic description*. These descriptions are usually in the form of *condition* $\rightarrow$ *action*. In this description, *linguistic variables* are used to describe fuzzy system inputs and outputs, and exist in one-to-one correspondence with numeric variables. For example, QoS-error is a linguistic variable corresponding to the numeric variable for the change in FPS. A linguistic variable takes on linguistic values such as positive-large and negative-large. Such variables indicate, among other information, the direction and magnitude of a variable. To better elaborate on the design components of a fuzzy controller, we use a simple DVFS example for single core power management. This is a simple and classic control problem for which many efficient techniques already exist such as a simple PID controller or a regression model. However, we use this example simply to illustrate the design and basic mechanics of a fuzzy control system. Here, y denotes the power consumption of the core (in Watts), and u is the frequency of the core (in MHz). We will use r to denote the desired power of the processor. The goal is to track this target power reference either specified by the system design manual or imposed by a higher system objective.

3.1.1 Fuzzification. The role of the fuzzification interface is to convert controller inputs into information that can be easily used to process, activate, and apply rules. Fuzzification can be simply defined as the mapping process between an obtained value for an input variable (e.g., IPS value, QoS metrics, execution time) to its numeric value defined in the corresponding membership function (MF). Membership function values can be interpreted as the *encoding* of the fuzzy controller numeric input values. The encoded information is then used in the fuzzy inference process.

[2]We limit our description of fuzzy control to cover the basis of practical control applications. We encourage the avid reader to find a more detailed analysis of fuzzy logic, sets, and systems to consult [42][41][90].

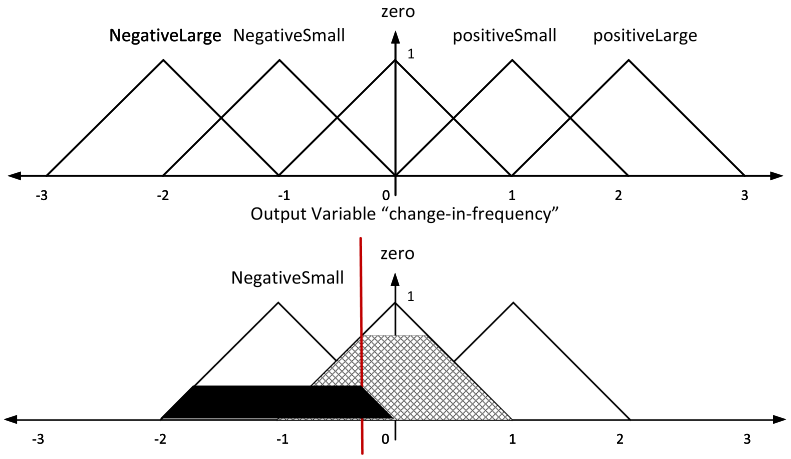

Fig. 3. (top) Sample membership function for *change-in-frequency*. (bottom) Implied fuzzy sets for two rules in DVFS example.

Depending on the application and the designer's preference, many different choices of membership functions are possible. Membership functions (e.g., trapezoid-shaped, Gaussian-shaped, Sharp peak Skewed triangle, etc.) quantify the meaning of the linguistic statements that experts used in defining the rules in the rule base [51].

In our DVFS example, we use $e(t) = r(t) - y(t)$ as the input to the fuzzy controller which denotes the error between the reference power and the current power value. As $e(t)$ takes on a value of, for example, 100mW at t = 2 (e(2) = 0.1W), linguistic variables assume "linguistic values." That is, the values that linguistic variables take on over time change dynamically. Suppose for the DVFS example that *error* and *change-in-frequency* take on the following values (*negativeLarge*, *negativeSmall*, *zero/hold*, *positiveSmall*, and *positivelarge*). Top part of Figure 3 shows membership function for *change-in-frequency* with corresponding values as a sample. This membership function can be used in the next steps to process and determine the output of the fuzzy control or simply decide how much frequency needs to be changed to achieve the target reference. We define similar functions for error in power with an adjusted range. Note that we mainly use triangles and skewed triangles for membership functions in this work due to its low computation overhead.

3.1.2 Rule Base. Construction of the rule base is where the experience and domain knowledge of experts prove to be beneficial. A deep understanding of the target system dynamics can increase the success of the designed controllers in the deployment process. In this step, values for input and output variable are described. Expert's knowledge alongside the above quantification is then used to specify a set of rules on how to control the system. Fuzzy rules are expressed in terms of linguistic variables. It is important to note that these rules are defined in a way that is easy to understand and interpret by humans. At this point, designer does not need to focus on details of control parameters and can simply define the rule structure of the control process. This comes from the raised abstraction level in the rule base definition that simply specifies the general idea on how to control a process. Although this might appear to be different from designers utilizing heuristics, this difference is one of the strong points of fuzzy control as these rules are defined in a clear and understandable way which can be subject to test and improvement.

For instance, in our DVFS example if the power consumption is just slightly higher than our target, we want to reduce the frequency a bit to reduce the core power. This can simply be added as a rule that says "if power error is a small positive value then change the frequency by a negative and

small value" (equivalent to rule 4 in pseudocode 1). In the field of fuzzy control, there has been a vast body of work on how to automatically tune fine parameters of controller after the initial rule base structure has been defined by the expert designer [14, 34, 86]. In addition, fuzzy control provides multiple analytical methodologies (e.g., The Lyapunov Method, Absolute Stability, and the Circle Criterion [2]) for stability analysis that can analyze the deployed heuristics. Pseudocode 1 shows a sample rule-base for the DVFS example. The intuition behind the rules is simple, assuming the frequency variations change power consumption, based on the *error*, the fuzzy inference decide on the number of steps in the change of frequency. To summarize, the rule base keeps a record of linguistic variables, values, and their associated member functions in addition to the set of all the rules. These rules have the general format of conditional statements making them easy to understand and computationally lightweight, for example, when compared to the matrix algebra used in state-space based MIMO control.

ALGORITHM 1: DVFS rule-base example

Input: error: difference between the current power and the target power
Outputs: change-in-frequency: actuation to the next frequency
(1) if error is *negativeLarge* then change-in-frequency is *positiveLarge*
(2) if error is *negativeSmall* then change-in-frequency is *positiveSmall*
(3) if error is *zero* then change-in-frequency is *zero*
(4) if error is *positiveSmall* then change-in-frequency is *negativeSmall*
(5) if error is *positiveLarge* then change-in-frequency is *negativeLarge*

3.1.3 Fuzzy Inference. In the inference mechanism component, the expert's *decision making* is emulated by interpreting and applying knowledge about how best to control the plant. This mechanism is also often called fuzzy inference or inference engine. The inference comprises of two steps. In the first step, the current situation is determined based on the comparison of the premises of all the rules and control inputs. Note that, in this *matching* process, more than one rule can be applied to a situation. Based on the membership functions and the control inputs, we determine the certainty that each rule applies. This simply means the rules that are more relevant to the current status of the system will have a stronger influence on the inference conclusion. This certainty is denoted by $\mu_{premise}$ of that rule. To perform inference, each of the applied rules must first be quantified by extracting the value of each fuzzy controller input terms present in that rule and then applying the fuzzy logic (and/or) operation on them. Usually minimum or product operations can be used here which will lead to a fact for the rules that include multiple input statements, we can be no more certain about the conjunction of two or more statements than we are about the individual terms that make them up. In our example we only have one input stat, if power consumption is not close to the target reference, the matching decides the certainty of rules such as the ones that starts with *negativeLarge* and *negativeSmall* values for *error* in power. If we get a small negative value between zero and one for the error rules 2 and 3 will be picked in the matching process where rule 2 states "error is *negativeSmall* then change-in-frequency is *positiveSmall*" and rule 3 is defined as "if error is *zero* then change-in-frequency is *zero*".

The second step involves determining the controller actions or the *conclusion* process. Every rule that can be applied to the current situation in the control system has a corresponding action which defines a controller action or a conclusion. Based on the number of active rules in each situation, there can be one or more conclusions with different levels of certainty. The conclusions are characterized by a fuzzy set that represents the certainty that the control inputs had in the matching process. Next, we consider each conclusion separately to determine what is the action

recommended by the associated rule. Bottom part of Figure 3 shows an example that the implied fuzzy sets of the inference that matched with rules (2) and (3). We can see that certainty of the rule (3) ($\mu_{premise} = 0.75$) is higher compared to the second rule ($\mu_{premise} = 0.25$) which means that conclusion of this rule will have a stronger influence on the inference conclusion. Based on this we define the conclusion of each rule as:

$$\mu_{(2)}(u) = min\left\{0.25, \mu_{negsmall}(u)\right\} \tag{3}$$

$$\mu_{(3)}(u) = min\left\{0.75, \mu_{zero}(u)\right\} \tag{4}$$

In the next step, every recommendation from all the rules are combined to calculate the final controller action value.

3.1.4 Defuzzification. Defuzzification component operates on the implied fuzzy sets produced by the inference mechanism and combines their effects to provide the "most certain" controller output [51]. Basically, Defuzzification is the process of converting the degrees of membership of output linguistic variables within their linguistic terms into crisp numerical values. There are various defuzzification methods that can be used to find these numeric values such as Center of Gravity (COG), Center of Area (CoA), Modified Center of Area (mCoA), Center of Maximum (CoM), Mean of Maximum (MoM), Center of Sums (CoS) [75]. If one considers fuzzification as an "encoding" process, defuzzification can be seen as a "decoding" mechanism for the fuzzy set(s) obtained from the inference engine to generate numeric values.

Finally, going back to our example, we decode the result of the inference step from something such as a fuzzy set of *negativeSmall* and *zero* change-in-frequency (Equations (3), (4)). We use COG defuzzification mechanisms to find the crisp output for change-in-frequency (e.g., decrease in current frequency by 100 MHz if possible). Note that we can have another input to the fuzzy controller that checks the current frequency and makes sure that change-in-frequency will not lead to an out of range frequency value. We can also do this as part of post processing or a filter after the controller. Adding gains before and after the controller is also a common practice to tune the effect of the control decision on the system.

4 HESSLE-FREE: ON-CHIP RESOURCE MANAGEMENT

In this section, we first present HESSLE-FREE's fuzzy control architecture and describe a methodology for depicting how to control a computer system (Section 4.1). We then describe an experimental case study demonstrating the design and verification of HESSLE-FREE on NVIDIA Jetson TX2 heterogeneous platform (Section 4.2).

4.1 Fuzzy Control in Computer Systems

Figure 4 depicts an abstract view of HESSLE-FREE for runtime resource management in modern computer systems. The information regarding application dynamics can be specified by either the user or the system software. An example can be QoS required by one or more application. This can be a certain frame-per-second (FPS) for a video processing application. System goal(s) need to be met or optimized by HESSLE-FREE. If the controller has more than one objective, it will attempt to achieve all of them. In the cases that a situation imposes a compromise on the system, the priorities in objectives embedded in the rule base will decide the controller's actions. A physical system can be managed as a single entity or can be broken down to sub-systems (sub-plants). If we can capture an estimated system dynamics with high accuracy, we can try to design a controller (either a conventional or fuzzy controller) for the identified system. Although in many cases due to a high order of dynamics in the system, the designed controller might suffer from sluggishness or maybe be impossible to meet the requirements of runtime systems. Some of such cases will be

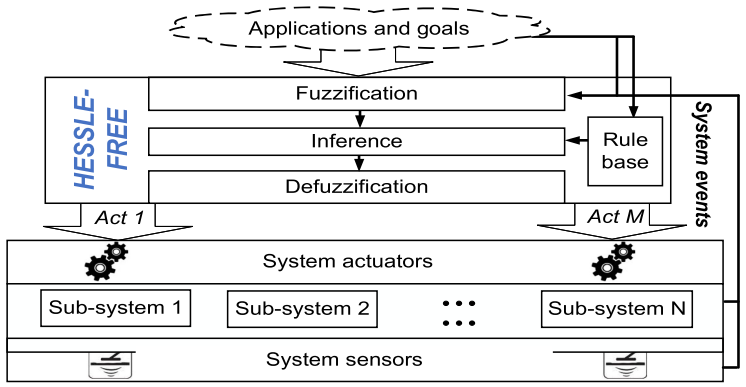

Fig. 4. Architecture of HESSLE-FREE.

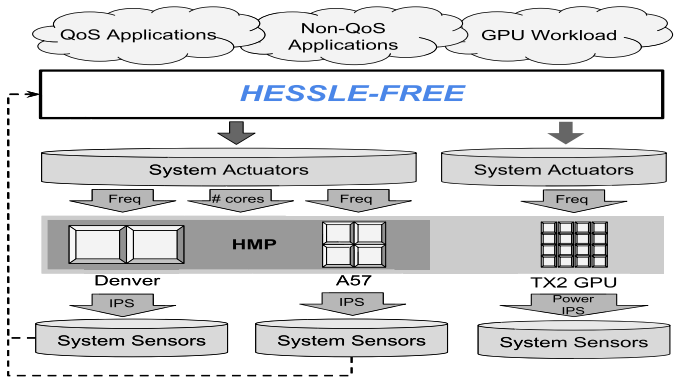

Fig. 5. HESSLE-FREE experimental setup.

highlighted in the experiments. In this situation, designers may decide to use modular decomposition to mitigate the complexity of control problems. This break down can simplify the control of the system and reduce the management algorithm's complexity but requires coordination between each sub-system. HESSLE-FREE handles such situation by obtaining information from each sub-system's sensors and the full-system information. This information is used in the inference mechanism to make decisions for actuating each sub-system in a way that will lead the full-system towards the system objectives.

4.2 HESSLE-FREE Case Study

Figure 5 shows an overview of our case study. We use the NVIDIA Jetson TX2 development board [10], which contains an HMP and a NVIDIA GPU. HMP contains a quad-core ARM Cortex A57 cluster and a dual-core NVIDIA Denver cluster. Similar to Cortex A57 cores, Denver cores implement ARMv8 instruction set and are designed as a processor with 7-way superscalar execution pipeline. The GPU is powered by NVIDIA Pascal CUDA cores. We consider multiple scenarios that are common in mobile devices where CPU runs multiple tasks (possibly one foreground with QoS requirements and others in background) and in full system scenarios GPU is executing a highly parallel kernel concurrently.

We use the following two scenarios to demonstrate how HESSLE-FREE can handle different system goals: i) Optimize Energy consumption under dynamic application behavior. Here we execute

workloads on CPU cores to demonstrate HESSLE-FREE's ability to dynamically optimize energy. ii) In a full system scenario, the CPU and GPU simultaneously execute their workloads, while HESSLE-FREE optimizes the user metric *frames per second* delivered by the GPU or QoS metric delivered by the CPU, as well as the *power consumption* of the entire system.

5 EXPERIMENTAL SETUP

As described in Section 4.2 we use the JetsonTX2 platform for our evaluations. The controllers used in our experiments are implemented as Linux userspace daemons that execute in the background with the applications. CPU and GPU runtime power are separately measured on-board alongside current and voltage using sensors present on the JetsonTX2 development board. Power measurements are made at the same time increments as performance metrics are gathered. Controller invocation is performed periodically every 200ms. In terms of overhead, the framework runtime on average adds 1.8% to the execution time for accessing PMU registers. A lightweight kernel module is used to collect instruction and cycle counters from ARM's Performance Monitor Unit (PMU) on each CPU. For GPU performance metric measurements, NVIDIA provides CUDA Profiling Tools Interface (CUPTI) library which includes API for attaching callback functions to GPU kernels. The callback functions enable measuring GPU metrics in application run-time. To avoid modifying target applications, we put necessary CUPTI functions into a shared library which is pre-loaded (LD_PRELOAD) to attach the callbacks when the application begins. This non-intrusive GPU profiling is hooked to our runtime resource management framework to capture the kernel information with low-overhead on the execution of the workload which in average adds 2.2% to GPU kernel executions. This delay is a fraction of imposed overhead by NVIDIA's native profiler (nvprof).

5.1 Evaluated Workloads

We use the PARSEC benchmark suite [5] to evaluate the performance of the resource managers. To better represent a real-world scenario where every element of a CPU/GPU system is involved in the computation, we select a face detection algorithm for the GPU workload. We used the implementation in [87] as a standalone application, which is easily portable to an embedded environment, based on the Viola-Jones face detection framework [80] with three GPU kernels for compute-intensive part and some CPU computation for pre- and post-processing of the frame. This application has a frame-per-second (FPS) requirement which can be an objective for the controller. Our target platform is a modern heterogeneous platform that can execute various multi-threaded application simultaneously on HMP while concurrently running massively parallel kernels on the GPU. Depending on user preference and system state on any point of time, priorities of runtime system might change which will update the objective of the resource management mechanism. We will demonstrate three scenarios in Section 6 to show the ease and efficiency of HESSLE-FREE in adapting to various objectives.

5.2 Manager Configurations

HESSLE-FREE provides a framework to efficiently design, implement, deploy and tune fuzzy controllers. This framework comprises of three main components: (1) Design and initial evaluation for the controller, (2) Mapping and optimization of the controllers for portable deployment, (3) Middleware that accommodates the controller and provides APIs for monitoring and actuating configuration settings of the system in various software and hardware layers. In our evaluations, we use Matlab for design and initial test of the resource manager approaches. Fuzzy controllers can be designed using Matlab's fuzzy designer as Mamdani (linguistic) or with neuro fuzzy designer as Sugeno-type controllers with singleton consequents which leads to a simple yet efficient controller

with low computational overhead. For defuzzification, we use the centriod method. For each of our experiments with unique objectives, number of membership functions for each input and output range from 3 to 7 and deployed number of rules in the rule base did not exceed one hundred. The design process in HESSLE-FREE starts with the designer defining inputs and outputs of the system and their corresponding number of membership functions. Next, structure of the rule base is defined using linguistic variables. From here, the designer has the option to check the sanity of the inference system, simulate the system based on experimental data gathered from the target platform or use adaptive neuro-fuzzy toolbox to tune rule base and membership functions parameters. Subtractive clustering [66] provided by this toolbox may be used as a rapid one-pass algorithm in this process for estimating the number of rule clusters and the cluster centers in the rule base. The cluster estimates obtained from this function can be used to reduce the size of the rule base and consequently the runtime overhead of the matching and inference process. After initial test and evaluation, each controller is ported into fuzzylite library [57] for an optimized implementation. The resulting controller is integrated with our middleware. We have devised this component to be easily portable to platforms running Linux operating system while providing customizable APIs for accessing system metrics (IPS, power, utilization, etc.) and changing the configuration knobs (e.g., DVFS settings, number of cores, task mapping, etc.).

In order to design a valid MIMO controller for each scenario, a system model is created using Matlab System Identification Toolbox [46] by generating test waveforms from training applications. A common practice to build a model for complex systems is to use black-box methods based on System Identification Theory [43] for isolating the deterministic and stochastic components. Then, these controllers are tuned using Matlab's Simulink toolbox. Afterwards, they are deployed on the target system for experiments using test workloads. A detailed report on MIMO design can be found in [53]. Considering the large design space of configuration settings and impacting factors in our evaluations that involve both CPU and GPU applications, system identification requires fine tuning and repeated evaluations to ensure the extracted model can appropriately reflect the target platform. **MIMO's complexity:** To depict a picture regarding exponential growth of MIMO controllers runtime complexity we use the number of required matrix multiplications depending on *order* and *size* of the system. For a simple second order MIMO controller with two inputs and outputs approximately 300 operations are needed. Increase in order of the same controller to 4th and 8th order controller will result in 1500 and 10000 operations. If instead of the order, size of the system grows to 4 inputs and outputs, number of required runtime operations will reach 15000. For a bigger 8x8 system this number exceeds 200000 operations per decision making. In some cases, runtime overhead of MIMO controllers for such large complex systems can put a burden on the runtime framework. NVIDIA Performance modes and resource management governs are pre-loaded into the operating system that is ported specifically for jetson platform. Based on the target evaluation some of which are used in the appropriate scenario experiments.

6 EVALUATION RESULTS

We now demonstrate the advantages of fuzzy control in runtime resource management with optimization objectives along side system goals that might require certain guarantees. Our goal is to evaluate our HESSLE-FREE with respect to the state-of-the-art control theory controllers in terms of both ability to capture the system's dynamics and achieve the system's objectives. In order to make a fair comparison, we start with a simple multi-core system and work our way towards a more complex heterogeneous system. In this manner we are able to highlight the challenges faced by MIMO controller design for complex system dynamics, while showing the ease of using HESSLE-FREE. In addition, the efficacy of HESSLE-FREE is compared against a correspondent MIMO controller and state-of-the-art algorithms towards a certain objective. We show the

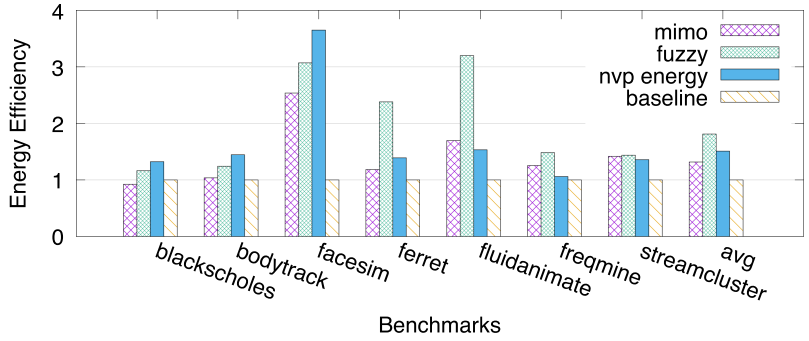

Fig. 6. MIPS per Watt for CPU workloads. This value is normalized to default linux values.

capability of HESSLE-FREE towards achieving various system's objectives in order to demonstrate ease in design and flexibility of fuzzy controllers.

6.1 Uniform Multi-core

In this section, we demonstrate the inherent optimization of fuzzy controller compared to conventional controllers by evaluating the efficacy of our approach for system energy minimization of a quad-core CPU. We use million instructions per second (MIPS) per Watt to represent energy consumption of this ARM A57 multi-core at runtime. MIMO has shown to be effective in tracking power and performance references for such systems with low-level of heterogeneity [48]. However, when the optimization happens at run-time in reaction to the dynamic behavior of the application, traditional MIMO faces challenges. Fuzzy control enables the embedding of optimization algorithms inside the control mechanism, which in turn naturally allows the system to react to this dynamic behavior.

Figure 6 shows the normalized MIPS per Watt for CPU workloads. We evaluate HESSLE-FREE in comparison to a MIMO controller and a NVIDIA performance model designed for energy efficient execution on Jetson TX2 platform. The actuations in the system are the number of active cores and core frequency. The intuition behind the designer's expertise used in design of this fuzzy controller is to adjust the computation power of the system to the dynamic behavior of the application. This will allow the fuzzy controller to increase the frequency and active cores as long as it adds to performance of the system in a meaningful manner and reduce the computation power to avoid energy waste while the performance is bounded either by memory access or disturbance of background applications. Alteration of frequency settings has a much smaller threshold compared to change in number of active cores. This threshold is extracted through experimental evaluations done in the initial phase of the fuzzy controller design. Overall, HESSLE-FREE demonstrates efficiency in managing the system's objectives. On average, fuzzy controller shows improvements of 81.3% over the Linux governor, 37.7% over the MIMO controller, and 20.0% over NVIDIA's state-of-the-art energy efficient governor.

6.2 CPU-GPU Resource Management

To evaluate the efficacy of HESSLE-FREE with respect to MIMO solutions, we perform two experiments on a full system exercising both the CPU clusters (executing PARSEC benchmarks) and the GPU (executing the Face detection application). We perform our full system evaluations comparing HESSLE-FREE with MIMO controller and a variety of Linux governors (*interactive*, *ondemand*, *performance oriented*, *power saving*). Each of these governors targets a certain objective in the system like maximizing performance or minimizing power consumption. For each of the experiments

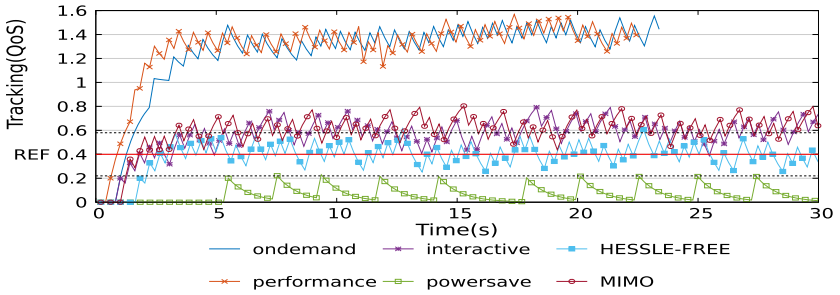

Fig. 7. Tracking QoS metric (ref = 0.4) for fluidanimate benchmark with different resource managers.

we perform system identification and control design process for MIMO controller. Target references are obtained through Sensors and actuators are fixed for each experiment. MATLAB System Identification toolbox also recommends a suitable order for the system. If possible, we pick the order with the best accuracy. However, the *order* of a controller model determines how observed output history is stored in the model, and directly impacts both the controller size and complexity. We begin with smaller size of the system (2 inputs x 2 outputs) where efficient MIMO controller is feasible and show the decrease in system identification accuracy and controller design efficiency when the number of the inputs (e.g, frequency knobs) and target outputs (e.g, QoS metric, power consumption) grows. During our full system experiments, We evaluated that for higher numbers of systems inputs (e.g, frequency and number of active cores in each cluster) and outputs (e.g, simultaneous FPS and QoS metrics with each units power consumption) with heterogeneous compute units the complexity of the system grows to the point that the system dynamics cannot be captured in the system identification phase with acceptable accuracy and also Matlab can only provide very large orders for MIMO controllers that cannot be computed at runtime. However, HESSLE-FREE without the need for system model was able to achieve system goals using expertise knowledge and tuning. Here, we report on the cases that MIMO control design was possible.

6.2.1 QoS Focused. In this experiment, we use a QoS metric to evaluate the progress of the CPU foreground application. This is done by the Heartbeats API [19] monitor to measure QoS. By periodically issuing *heartbeats*, the application informs the system about its current performance. The user provides a performance reference value using the Heartbeats API. Figure 7 shows tracking of QoS value by different resource management mechanisms for fluidanimate benchmark and target reference of 0.4. In this scenario, one foreground QoS application is running while there are many non-QoS applications are running in the background both on CPU and GPU. The goal of the resource manager is to keep the heart beats (QoS metric) in the specified range (0.2–0.6) by the application while consuming minimum power. Because of the heterogeneity in the CPU clusters MIMO controller has a hard time following the target reference while fuzzy controller is able to meet the QoS in a steady manner using minimum energy compared to other resource managers. Expertise used in this experiment for fuzzy controller was to not only consider the error from reference QoS but also the speed of change in measured Heartbeats. Meanwhile, as we reach and pass the target reference, we reduce the frequency of the compute unit incrementally to the extent that QoS drops to half point of target reference and lower boundary. Moving forward, CPU frequency is increased to the point that we exceed target reference again. Figure 8 shows the comparison of energy consumption of each resource manager. HESSLE-FREE achieves and tracks the QoS metric while being in average 26%, 46%, 43%, 66% and 65% more energy efficient than MIMO controller, interactive ondemand, performance and power oriented governors, respectively. This is due to the designer expertise incorporated in the fuzzy controllers that favors energy efficient

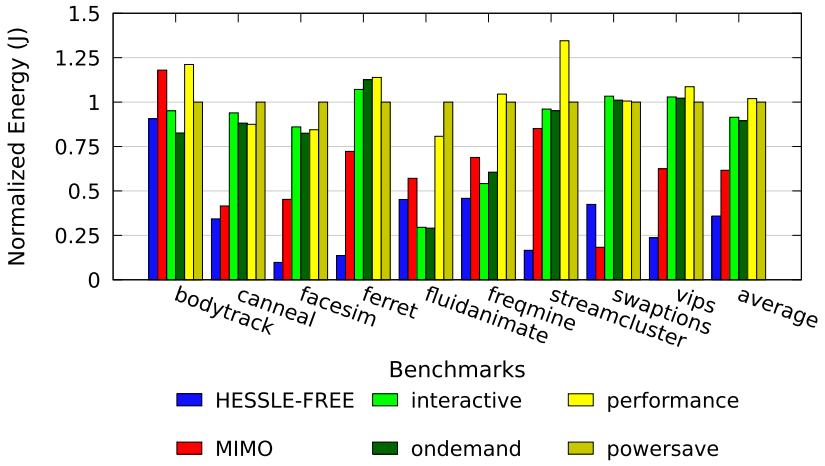

Fig. 8. Total Energy consumption for CPU+GPU for tracking QoS metric (normalized to power saver energy).

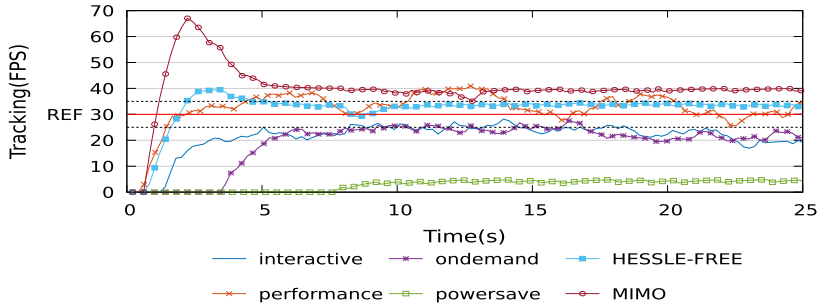

Fig. 9. Delivered FPS for Face detection.

cores actuations and only increases frequency for high-performance cores when higher QoS is needed.

6.2.2 FPS Focused. In this experiment our target metric is to meet our desired FPS while consuming the minimum power required. Target FPS is defined as 30 frames per second for our platform with threshold of ±5 frames. We capture the number of frames processed in each measuring window. In order to stress test the management policy, we execute the PARSEC benchmarks on the CPU cluster in parallel with the face detection GPU workloads. The rationale for this mix is that the CPU workloads can demonstrate dynamic phasic behavior (e.g., compute-bound, memory-bound) that can affect the GPU performance and consequently the FPS of the system. Figure 9 shows the FPS tracking for each resource manager for the facesim CPU benchmark. The rest of workloads follow a similar trend where: i) the performance oriented governor provides high FPS with no regard to power consumption, ii) the ondemand and interactive governor provides moderate performance based on application demand which sometimes results in high power consumption of CPU units, iii) the power saving governor executes in the lowest configuration of each compute unit without any regard to the system status, iv) MIMO generally tracks FPS but abrupt changes to power consumption can cause deviation from the target reference, and v) our HESSLE-FREE fuzzy controller is able to follow the reference FPS while minimizing energy consumption. The intuition behind fuzzy rules is to observe the error in FPS and take actions according to value of this error to

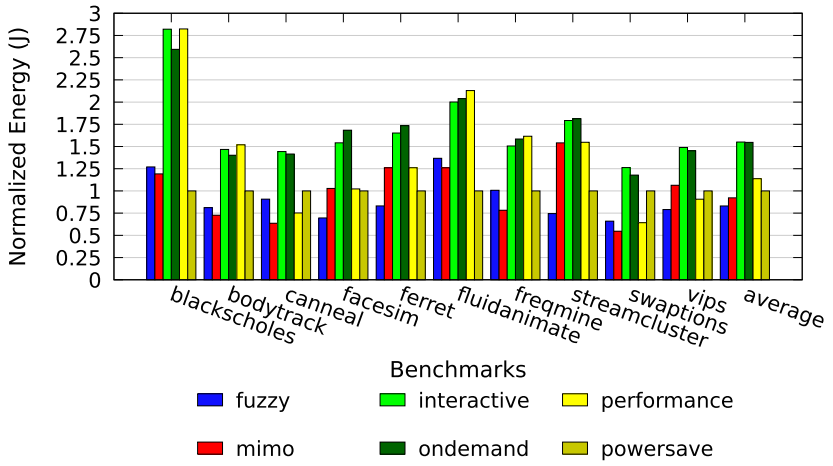

Fig. 10. Total Energy consumption for CPU (PARSEC) plus GPU (face detection) for FPS metric.

set the configuration knobs. As the error gets closer to the target FPS, GPU frequency change slows down. Meanwhile, we try to minimize the energy consumption of the entire platform by reducing the energy consumption of the CPU cores while avoiding any drops in FPS measurement. Also, In the case that increase in GPU frequency shows no improvement in a consecutive windows, we increase the CPU cores frequency. This is done to reduce the frame pre-processing bottleneck. Figure 10 shows the energy consumption of CPU clusters executing PARSEC benchmarks plus GPU cores executing face detection through this experiment. HESSLE-FREE's fuzzy governor is able to achieve the desired objective tracking the system's FPS while in average preserving 9% more energy than MIMO controllers.

7 CONCLUSION

We presented HESSLE-FREE, a fuzzy control mechanism for on-chip runtime resource management in heterogeneous systems. HESSLE-FREE leverages fuzzy control theory to combine heuristic approaches with the strengths of classic control theory to efficiently manage complex heterogeneous systems with a variety of system objectives. We demonstrated the simplicity and effectiveness of HESSLE-FREE with implementation on the NVIDIA Jetson TX2 platform with heterogeneous (CPU+GPU) compute units. Our evaluations show that HESSLE-FREE can successfully manage complex systems in an energy efficient manner while achieving QoS targets. We believe that our HESSLE-FREE approach highlights the promise of fuzzy control for resource management in heterogeneous computer systems, and paves the way for managing the increasing complexity of newer heterogeneous computing platforms.

REFERENCES

[1] A. M. Rahmani et al. 2015. Dynamic power management for many-core platforms in the dark silicon era: A multi-objective control approach. In *ISLPED*.

[2] J. Aracil and F. Gordillo. 2000. *Stability Issues in Fuzzy Control*. Physica-Verlag HD. https://books.google.com/books?id=h6weAQAAIAAJ.

[3] ARM. 2013. *big.LITTLE Technology: The Future of Mobile*. Technical Report. https://www.arm.com/files/pdf/big_LITTLE_Technology_the_Futue_of_Mobile.pdf.

[4] Andrea Bartolini, Matteo Cacciari, Andrea Tilli, and Luca Benini. [2011]. A distributed and self-calibrating model-predictive controller for energy and thermal management of high-performance multicores. In *DATE, 2011*.

[5] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. The PARSEC benchmark suite. In *PACT*. ACM Press, New York, New York, USA, 72. DOI: https://doi.org/10.1145/1454115.1454128

[6] Ramazan Bitirgen et al. 2008. Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach. In *MICRO*.
[7] R. Bitirgen, E. Ipek, and J. F. Martinez. 2008. Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach. In *2008 41st IEEE/ACM International Symposium on Microarchitecture*.
[8] Tobias Bjerregaard and Shankar Mahadevan. 2006. A survey of research and practices of network-on-chip. *Comput. Surveys* 38, 1 (2006).
[9] Zhijia Chen, Yuanchang Zhu, Yanqiang Di, and Shaochong Feng. 2015. A dynamic resource scheduling method based on fuzzy control theory in cloud environment. *Journal of Control Science and Engineering* 2015 (2015), 1–10. DOI: https://doi.org/10.1155/2015/383209
[10] NVIDIA Corporation. 2017. NVIDIA Jetson TX2 embedded module. https://developer.nvidia.com/embedded/buy/jetson-tx2.
[11] L. Costero, A. Iranfar, M. Zapater, F. D. Igual, K. Olcoz, and D. Atienza. 2019. MAMUT: Multi-agent reinforcement learning for efficient real-time multi-user video transcoding. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*.
[12] Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: Resource-efficient and QoS-aware cluster management. In *ASPLOS*.
[13] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini. 2012. CoScale: Coordinating CPU and memory system DVFS in server systems. In *MICRO*.
[14] Dong Hwa Kim. 2002. Parameter tuning of fuzzy neural networks by immune algorithm. In *2002 IEEE World Congress on Computational Intelligence. 2002 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE'02. Proceedings (Cat. No.02CH37291)*.
[15] Bryan Donyanavard, Tiago Mück, Santanu Sarma, and Nikil Dutt. 2016. SPARTA: Runtime task allocation for energy efficient heterogeneous many-cores. In *CODES, 2016*.
[16] Christophe Dubach, Timothy M. Jones, and Edwin V. Bonilla. 2013. Dynamic microarchitectural adaptation using machine learning. In *TACO, 2013*.
[17] Christophe Dubach, Timothy M. Jones, Edwin V. Bonilla, and Michael F. P. O'Boyle. 2010. A predictive model for dynamic microarchitectural adaptivity control. In *MICRO, 2010*.
[18] Luc Pronzato Eric Walter. 1997. *Identification of Parametric Models from Experimental Results*. Springer.
[19] Hoffmann et al. 2013. A generalized software framework for accurate and efficient management of performance goals. In *EMSOFT*.
[20] Heirman et al. 2014. Undersubscribed threading on clustered cache architectures. In *HPCA*.
[21] Sui et al. 2016. Proactive control of approximate programs. In *ASPLOS*.
[22] Xing Fu, Khairul Kabir, and Xiaorui Wang. 2011. Cache-aware utilization control for energy efficiency in multi-core real-time systems. In *ECRTS, 2011*.
[23] Ujjwal Gupta, Raid Ayoub, Michael Kishinevsky, David Kadjo, Niranjan Soundararajan, Ugurkan Tursun, and Umit Ogras. 2017. Dynamic power budgeting for mobile systems running graphics workloads. In *TMSCS, 2017*.
[24] Ujjwal Gupta, Manoj Babu, Raid Ayoub, Michael Kishinevsky, Francesco Paterna, and Umit Y. Ogras. 2018. STAFF: Online learning with stabilized adaptive forgetting factor and feature selection algorithm. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.
[25] Ujjwal Gupta, Joseph Campbell, Umit Y. Ogras, Raid Ayoub, Michael Kishinevsky, Francesco Paterna, and Suat Gumussoy. 2016. Adaptive performance prediction for integrated GPUs. In *ICCAD, 2016*.
[26] M. H. Haghbayan, A. Miele, A. M. Rahmani, P. Liljeberg, and H. Tenhunen. 2017. Performance/reliability-aware resource management for many-cores in dark silicon era. *IEEE TC* (2017).
[27] Nejatollahi Hamid and Salehi Mostafa E. 2018. Reliability-aware voltage scaling of multicore processors in dark silicon era. *Advances in Parallel Computing* 33, Big Data and HPC: Ecosystem and Convergence (2018), 242–262. DOI: https://doi.org/10.3233/978-1-61499-882-2-242
[28] Vinay Hanumaiah, Digant Desai, Benjamin Gaudette, Carole-Jean Wu, and Sarma Vrudhula. 2014. STEAM: A smart temperature and energy aware multicore controller. In *TECS, 2014*.
[29] P. Haratian, F. Safi-Esfahani, L. Salimian, and A. Nabiollahi. 2018. An adaptive and fuzzy resource management approach in cloud computing. *IEEE Transactions on Cloud Computing* (2018).
[30] Henry Hoffmann et al. 2011. Dynamic knobs for responsive power-aware computing. In *ASPLOS*.
[31] J. Hu, W. Peng, and C. Chung. 2018. Reinforcement learning for HEVC/H.265 intra-frame rate control. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*.
[32] Connor Imes, Steven Hofmeyr, and Henry Hoffmann. 2018. Energy-efficient application resource scheduling using machine learning classifiers. In *Proceedings of the 47th International Conference on Parallel Processing (ICPP'18)*.
[33] Engin Ipek, Onur Mutlu, José F. Martínez, and Rich Caruana. 2008. Self-optimizing memory controllers: A reinforcement learning approach. In *ISCA, 2008*.

[34] J. R. Jang. 1993. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics* (1993).
[35] Biresh Kumar Joardar, Ryan Gary Kim, Janardhan Rao Doppa, Partha Pratim Pande, Diana Marculescu, and Radu Marculescu. 2018. Learning-based application-agnostic 3D NoC design for heterogeneous manycore systems. *IEEE Trans. Comput.* (2018).
[36] H. Jung et al. 2008. Stochastic modeling of a thermally-managed multi-core system. In *DAC*.
[37] A. Kanduri, M. H. Haghbayan, A. M. Rahmani, P. Liljeberg, A. Jantsch, N. Dutt, and H. Tenhunen. 2016. Approximation knob: Power capping meets energy efficiency. In *ICCAD, 2016*.
[38] A Kanduri, M. H. Haghbayan, A. M. Rahmani, M. Shafique, P. Liljeberg, and A. Jantsch. 2018. dBoost: Thermal aware performance boosting through dark silicon patterning. *IEEE Trans. Comput.* (2018).
[39] Umair Ali Khan and Bernhard Rinner. 2014. Online learning of timeout policies for dynamic power management. *ACM Trans. Embed. Comput. Syst.* 13, 4 (March 2014).
[40] R. G. Kim, W. Choi, Z. Chen, J. R. Doppa, P. P. Pande, D. Marculescu, and R. Marculescu. 2017. Imitation learning for dynamic VFI control in large-scale manycore systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2017).
[41] G. J. Klir and T. A. Folge. 1988. *Fuzzy Sets, Uncertainty and Information*. Prentice-Hall, Englewood Cliffs, NJ, USA.
[42] G. J. Klir and B. Yuan. 1995. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice-Hall, Englewood Cliffs, NJ, USA.
[43] L. Ljung. 1999. *System Identification: Theory for the User*. Prentice Hall PTR.
[44] D. Mahajan et al. 2016. Towards statistical guarantees in controlling quality tradeoffs for approximate acceleration. In *ISCA*.
[45] R. Marculescu, U. Y. Ogras, L. Peh, N. E. Jerger, and Y. Hoskote. 2009. Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives. *IEEE TCAD* (2009).
[46] MathWorks. 2017. *System Identification Toolbox*. Technical Report. https://www.mathworks.com/products/sysid.html.
[47] Asit K. Mishra et al. 2010. CPM in CMPs: Coordinated power management in chip-multiprocessors. In *SC*.
[48] T. Mück, B. Donyanavard, K. Moazzemi, A. M. Rahmani, A. Jantsch, and N. Dutt. 2018. Design methodology for responsive and rrobust MIMO control of heterogeneous multicores. *IEEE TMSCS* (2018).
[49] Shaheryar Najam, Jameel Ahmed, Saad Masood, and Chuadhry Mujeeb Ahmed. 2019. Run-time resource management controller for power efficiency of GP-GPU architecture. *IEEE Access* (2019). DOI : https://doi.org/10.1109/ACCESS.2019.2901010
[50] Mina Niknafs, Ivan Ukhov, Petru Eles, and Zebo Peng. 2019. Runtime resource management with workload prediction. In *Proceedings of the 56th Annual Design Automation Conference 2019 (DAC'19)*.
[51] Kevin M. Passino and Stephen Yurkovich. 1997. *Fuzzy Control*. Addison-Wesley.
[52] P. Petrica et al. 2013. Flicker: A dynamically adaptive architecture for power limited multicore systems. In *ISCA*.
[53] Raghavendra Pothukuchi et al. 2016. A guide to design MIMO controllers for architectures. http://iacoma.cs.uiuc.edu/iacoma-papers/mimoTR.pdf (2016).
[54] Raghavendra Pradyumna Pothukuchi et al. 2016. Using multiple input, multiple output formal control to maximize resource efficiency in architectures. In *ISCA*.
[55] A. Prakash, H. Amrouch, M. Shafique, T. Mitra, and J. Henkel. 2016. Improving mobile gaming performance through cooperative CPU-GPU thermal management. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*.
[56] A. Prakash, H. Amrouch, M. Shafique, T. Mitra, and J. Henkel. 2016. Improving mobile gaming performance through cooperative CPU-GPU thermal management. In *DAC*.
[57] Juan Rada-Vilela. 2018. The FuzzyLite Libraries for Fuzzy Logic Control. https://fuzzylite.com/.
[58] Ramya Raghavendra et al. 2008. No "power" struggles: Coordinated multi-level power management for the data center. In *ASPLOS*.
[59] A. M. Rahmani, P. Liljeberg, A. Hemani, A. Jantsch, and H. Tenhunen. 2016. *The Dark Side of Silicon*. Springer.
[60] Amir M. Rahmani, Bryan Donyanavard, Tiago Mück, Kasra Moazzemi, Axel Jantsch, Onur Mutlu, and Nikil Dutt. 2018. SPECTR: Formal supervisory control and coordination for many-core systems resource management. In *ASPLOS*.
[61] A. M. Rahmani, M. H. Haghbayan, A. Miele, P. Liljeberg, A. Jantsch, and H. Tenhunen. 2017. Reliability-aware runtime power management for many-core systems in the dark silicon era. *IEEE TVLSI* (2017).
[62] Jia Rao, Yudi Wei, Jiayu Gong, and Cheng Zhong Xu. 2011. DynaQoS: Model-free self-tuning fuzzy control of virtualized resources for QoS provisioning. *IWQoS* (2011). DOI : https://doi.org/10.1109/IWQOS.2011.5931341
[63] K. Rao, J. Wang, S. Yalamanchili, Y. Wardi, and Y. Handong. 2017. Application-specific performance-aware energy optimization on Android mobile devices. In *HPCA*.
[64] B. K. Reddy, A. K. Singh, D. Biswas, G. V. Merrett, and B. M. Al-Hashimi. 2018. Inter-cluster thread-to-core mapping and DVFS on heterogeneous multi-cores. *IEEE TMSCS* (2018).

[65] Leili Salimian, Faramarz Safi Esfahani, and Mohammad-Hossein Nadimi-Shahraki. 2016. An adaptive fuzzy threshold-based approach for energy and performance efficient consolidation of virtual machines. *Computing* (jun 2016). DOI: https://doi.org/10.1007/s00607-015-0474-5

[66] Seema Chopra, R. Mitra, and Vijay Kumar. 2004. Identification of rules using subtractive clustering with application to fuzzy controllers. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*.

[67] Najam Shaheryar, Yasir Qadri Muhammad, Najam Zohaib, Ahmed Jameel, and N. Qadri Nadia. 2018. A fuzzy logic based power-efficient run-time reconfigurable multicore system. *Chinese Journal of Electronics* (May 2018). DOI: https://doi.org/10.1049/cje.2018.02.005

[68] Sina Shahosseini, Kasra Moazzemi, Amir M. Rahmani, and Nikil Dutt. 2017. Dependability evaluation of SISO control-theoretic power managers for processor architectures. *NORCAS* (2017).

[69] Elham Shamsa, Anil Kanduri, Amir M Rahmani, Pasi Liljeberg, Axel Jantsch, and Nikil Dutt. 2018. Goal formulation: Abstracting dynamic objectives for efficient on-chip resource allocation. In *2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*. IEEE, 1–4.

[70] Elham Shamsa, Anil Kanduri, Amir M Rahmani, Pasi Liljeberg, Axel Jantsch, and Nikil Dutt. 2019. Goal-driven autonomy for efficient on-chip resource management: Transforming objectives to goals. In *Proc. of Conf. on Design, Automation Test in Europe (DATE)*. IEEE.

[71] Amit Kumar Singh, Alok Prakash, Karunakar Reddy Basireddy, Geoff V. Merrett, and Bashir M. Al-Hashimi. 2017. Energy-efficient run-time mapping and thread partitioning of concurrent OpenCL applications on CPU-GPU MPSoCs. *ACM Trans. Embed. Comput. Syst.* (Sept. 2017).

[72] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel. 2013. Mapping on multi/many-core systems: Survey of current and emerging trends. In *DAC*.

[73] G. Singla, G. Kaur, A. K. Unver, and U. Y. Ogras. 2015. Predictive dynamic thermal and power management for heterogeneous mobile platforms. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*.

[74] S. Skogestad and I. Postlethwaite. 2005. *Multivariable Feedback Control: Analysis and Design*. John Wiley & Sons.

[75] Janusz T. Starczewski. 2013. *Defuzzification of Uncertain Fuzzy Sets*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[76] P Tembey et al. 2010. A case for coordinated resource management in heterogeneous multicore platforms. In *ISCA*.

[77] R. Teodorescu and J. Torrellas. 2008. Variation-aware application scheduling and power management for chip multiprocessors. In *ISCA*.

[78] K. Van Craeynest others. 2012. Scheduling heterogeneous multi-cores through performance impact estimation (PIE). In *ACM SIGARCH Computer Architecture News*.

[79] Augusto Vega et al. 2013. Crank it up or dial it down: Coordinated multiprocessor frequency and folding control. In *MICRO*.

[80] P. Viola and M. Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *CVPR*. DOI: https://doi.org/10.1109/CVPR.2001.990517

[81] X. Wan et al. 2011. Adaptive power control with online model estimation for chip multiprocessors. *IEEE TPDS* (2011).

[82] Lixi Wang, Jing Xu, Ming Zhao, and José Fortes. 2011. Adaptive virtual resource management with fuzzy model predictive control. *ICAC* (2011). DOI: https://doi.org/10.1145/1998582.1998623

[83] Y. Wang, K. Ma, and X. Wang. 2009. Temperature-constrained power control for chip multiprocessors with online model estimation. In *ISCA*.

[84] Q Wu et al. 2005. Formal control techniques for power-performance management. *IEEE Micro* (2005).

[85] K. Yan, X. Zhang, J. Tan, and X. Fu. 2016. Redefining QoS and customizing the power management policy to satisfy individual mobile users. In *MICRO*.

[86] Z. Yang, L. Li, and B. Liu. 2014. Auto-tuning method of fuzzy PID controller parameter based on self-learning system. In *2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*.

[87] Saehanseul Yi, Illo Yoon, Chanyoung Oh, and Youngmin Yi. 2014. Real-time integrated face detection and recognition on embedded GPGPUs. In *ESTIMedia*.

[88] H. Zhang et al. 2016. Maximizing performance under a power cap: A comparison of hardware, software, and hybrid techniques. In *ASPLOS*.

[89] Y. Zhang, J. Yao, and H. Guan. 2017. Intelligent cloud resource management with deep reinforcement learning. *IEEE Cloud Computing* (2017).

[90] H. J. Zimmerman. 1991. *Fuzzy Set Theory–and Its Applications*. Kluwer Academic Press, Boston, MA, USA.

Received April 2019; revised June 2019; accepted July 2019