Architecture and Cross-Layer Design Space Exploration

8

Santanu Sarma and Nikil Dutt

Abstract

The task of architectural Design Space Exploration (DSE) is extremely complex, with multiple architectural parameters to be tuned and optimized, resulting in a huge design space that needs to be explored efficiently. Furthermore, each architectural parameter and/or design point is critically affected by decisions made at lower levels of abstraction (e.g., layout, choice of transistors, etc.). Ideally designers would like to perform DSE incorporating information and decisions made across multiple layers of design abstraction so that the ensuing design space is both feasible and has good fidelity. Simulation-based methods alone can not deal with this incredibly large and complex design space. To address these issues, this chapter presents an approach for cross-layer architectural DSE that efficiently prunes the large design space and furthermore uses predictive models to avoid expensive simulations. The chapter uses a single-chip heterogeneous single-ISA multiprocessor as an exemplar to demonstrate how the large search space can be covered and evaluated efficiently. A cross-layer approach is presented to cope with the complexity by restricting the search/design space through the use of cross-layer prediction models to avoid too costly full system simulations, coupled with systematic pruning of the design space to enable good coverage of the design space in an efficient manner.

Acronyms

CLDSE	Cross-Layer Design Space Exploration
DoE	Design of Experiments

S. Sarma (✉)
University of California Irvine, Irvine, CA, USA
e-mail: santanus@uci.edu

N. Dutt
Center for Embedded and Cyber-Physical Systems, University of California Irvine, Irvine, CA, USA
e-mail: dutt@ics.uci.edu

S. Ha, J. Teich (eds.), *Handbook of Hardware/Software Codesign*,
DOI 10.1007/978-94-017-7267-9_9

DSE	Design Space Exploration
EDP	Energy-Delay Product
EDSP	Energy-Delay Square Product
HMP	Heterogeneous Multi-core Processor
ILP	Instruction-Level Parallelism
ISA	Instruction-Set Architecture
RSM	Response Surface Modeling
SA	Simulated Annealing

Contents

8.1 Introduction 248
8.2 Design Space Exploration of Heterogeneous Multi-core Processors 251
  8.2.1 Design of Experiments 252
  8.2.2 Response Surface Models 253
8.3 Cross-Layer Predictive Model Building Approach 253
  8.3.1 Problem Formulation 254
  8.3.2 Application and Workload Models 256
  8.3.3 Heterogeneity-Aware Task Allocation 256
  8.3.4 Predictive Modeling of Performance and Power of Different Core Types 258
  8.3.5 Training Methodology and Benchmarks 261
  8.3.6 Selecting the HMP Configuration 261
8.4 Case Study: Experimental Evaluation of Cross-Layer DSE of HMPs 262
8.5 Conclusions 266
References 267

8.1 Introduction

The task of architectural Design Space Exploration (DSE) is extremely complex, with multiple architectural parameters to be tuned and optimized, resulting in a huge design space. Furthermore, each architecture parameter and/or design point is critically affected by decisions made at lower levels of abstraction (e.g., layout, choice of transistors, etc.). Ideally, designers would like to perform DSE incorporating information and decisions made across multiple layers of design abstraction so that the ensuing design space is both feasible and has good fidelity. Simulation-based methods alone can not deal with this incredibly large and complex design space. To address these issues, this chapter presents an approach for cross-layer architectural DSE that efficiently prunes the large design space, and furthermore uses predictive models to avoid expensive simulations. The chapter uses a single-chip heterogeneous single-ISA multiprocessor system as an exemplar to demonstrate how the large search space can be covered and evaluated efficiently. This chapter complements other chapters in this book that give additional insights on specific optimization and exploration strategies. For instance: Chapter 6 by Scuito et al. describes optimization strategies for DSE; Chapter 7 by Glass et al. details advanced hybrid DSE techniques; and Chapter 10 by Henkel et al. incorporates power-aware run-time adaptions in DSE.

Single-chip-single-ISA-based Heterogeneous Multi-core Processors (HMPs) are increasingly considered as an attractive design alternative to homogeneous multiprocessor systems because of their superior performance, power, and energy efficiency while providing the flexibility of using the same software (binaries) and development tools across cores for a range of applications. HMPs can effectively address complex requirements of diverse applications by executing workloads (or tasks) in the most appropriate core types to meet competing and conflicting objectives and figures of merit (e.g., performance, power, energy, throughput, area, cost etc.) [3, 19, 29, 30]. Since different workloads (e.g., CPU bound, integer-intensive, floating-point intensive, memory intensive, etc.) require different resources, a key issue is to determine and select the right types and number of cores (processing elements) for an allocation strategy that maps the workload (or tasks) to right core type such that the type of workload will best benefit from the given platform. The selection of number and type of cores is not straightforward when the applications executed by these HMPs exhibit diverse workload characteristics. When designing such a system, a chip architect must decide how to distribute the available limited system resources, such as area and power, among all the processor cores.

HMPs that integrate a mix of small power-efficient cores and big high-performance cores are attractive alternatives to homogeneous multiprocessor systems because they have the potential for higher performance and reduced power consumption. Contemporary mobile phones have already embraced hardware core heterogeneity, for instance, ARM's big.LITTLE architecture [17] and NVIDIA's Kal-El [39] that have cores of different strengths in one cache coherence domain with the same Instruction-Set Architecture (ISA). Architectures with more than two core types are already a reality (e.g., NVIDIA's Kal-El [39] that integrates four high performance cores, one low performance core, and many GPU cores), and this trend toward heterogeneity is only expected to grow further in the future. Processor cores in a heterogeneous multi-core system can differ in their static microarchitectures to dynamic behavior or modes of operation (e.g., frequency or operating voltage). Broadly, the vast space of HMPs can be classified by core type (strength/size/number/ISA) and heterogeneity levels. As an example, consider Fig. 8.1 that shows a sample space of HMP architectures using a combination of different ARM cores – ranging from big (A15) to medium (A11) to small (A7) – that vary in their performance, power, and energy efficiency.

HMPs provide architecturally diverse cores with drastically different power-performance trade-offs that can be exploited for system efficiency. Consequently, HMP architectures and their DSE is an active area of research. DSE is the process of discovering and evaluating design alternatives during system development that enable design optimization, system integration, and rapid prototyping prior to implementation. The main challenge of DSE for HMPs arises from the sheer size of the design space that must be explored because a typical HMP system has a huge number of possibilities (in the millions, if not billions), and so enumerating every point in the design space considering different layers of the system stack is prohibitive. Although several works have studied HMPs and their run-time systems [2, 4, 8, 24, 29, 36, 47], the topic of DSE of HMPs is still in its infancy and needs a principled approach as these architectures evolve in their diversity and complexity.

DSE of HMPs is critical to evaluate and architect a suitable processor platform configuration. Selection and composition of the platform is important at the early

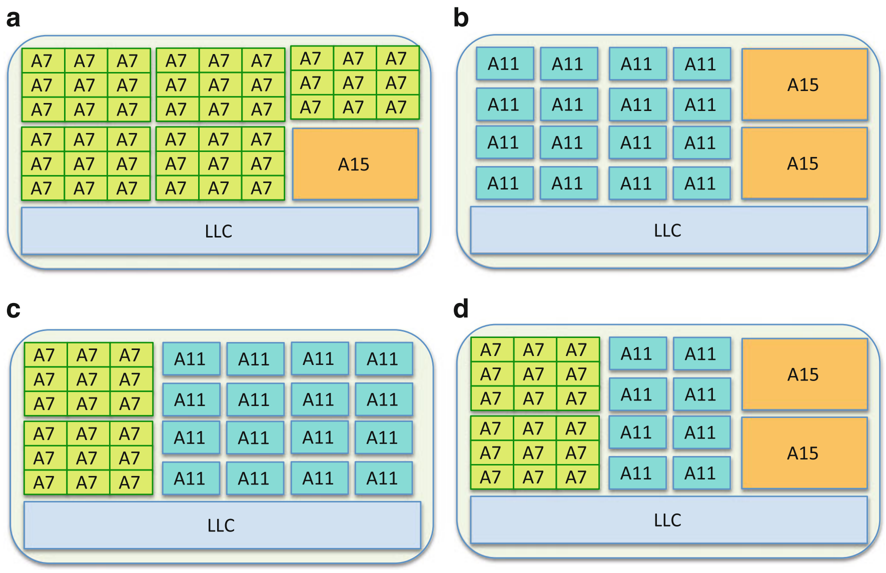

Fig. 8.1 Examples of heterogeneous architectures composition for the same die area using big (A15), medium (A11) , and little (A7) cores

stage of the design, and a chip architect must decide how to distribute the available limited system resources, such as area and power, among all the processor cores. Moreover, since HMPs are inherently designed as a multilayered system, either gross approximation or complete neglect of any layer and its features can affect the behavior, misrepresent the intricate multilayer trade-offs and interactions, as well as misguide the design and composition process. However, due to their diverse and vast design space, selecting a suitable HMP configuration with different core types within a given area-power budget is an extremely challenging task. This complexity challenge can be overcome in an intelligent manner by (a) restricting the search/design space and (b) using cross-layer prediction models to selectively avoid too costly full system simulations all the time, making a good coverage of the design space affordable. The problem of exploring and configuring an HMP for a given system goal under system-level constraints (such as equal area or power budget) can be cast as a cross-layer optimization problem. This chapter illustrates an approach that jointly consider cross-layer features of the application, operating system (task allocation strategies), and hardware architecture while deploying computationally efficient predictive models (of performance and power) in configuring the HMP platform resources (number and types of cores) in an evolutionary optimization framework. The predictive cross-layer approach enables the designer to comparatively evaluate and select the most promising (e.g., energy and performance efficient) HMP configuration in over two order of magnitude less simulation time compared to a full system simulation especially during the early design and verification stages when the design space is at its largest.

8.2 Design Space Exploration of Heterogeneous Multi-core Processors

Design space exploration needs two important components (a) a simulation infrastructure to evaluate different configurations and (b) predictive models to assess the quality of new configurations with a metric of goodness. System designers often build performance, power, and area (PPA) models of a microarchitecture to predict these metrics as a function of the design parameters $\mathbf{x}$. These models may be often represented as $y = J(\mathbf{x})$, where $J(\mathbf{x})$ represents a cycle accurate simulator or empirical model fit to simulated [33, 51] or prototype system data [10]. Detailed simulation is the most widely adopted approach in evaluating $J(\mathbf{x})$. However, the significant computational costs of simulation often hinder the design process leading to approaches that aim at reducing the simulation cost by either reducing the number of simulation runs of an evaluated program [14,22,41,51] or reducing the number of simulated architectures by building predictive models [7,9,11,20,23,31,33,40,50]. A combination of jointly reducing both program and architecture simulations is also possible [13, 27].

Design space exploration of HMPs is much more complex and challenging than that of homogeneous architectures since it involves reevaluating architecture and application options along with the operating system (OS) implications [10]. A straightforward extension of the abovementioned works is not directly applicable for HMPs as they are targeted toward homogeneous multi-core processors without considering the operating system. The exploration is performed using architectural and program space parameters without considering the OS scheduling by either using microarchitecture or cycle-accurate simulation. In order to consider the operating system in the DSE, full system cycle-accurate architectural simulators are indispensable tools for evaluating complicated and subtle design trade-offs with respect to large design spaces and handling various design constraints. As an exemplar, this chapter presents an extended full system cycle accurate HMP simulator [46] based on Gem5 [6] along with the Linux OS as the infrastructure for the cross-layer DSE. A predictive model of the full system is built, considering parameters from all the system stacks in order to capture the HMP performance and power characteristics. Unlike the state-of-the-art mentioned earlier that focused either on uni-core or homogeneous multi-core processors, the predictive approach specifically focuses on the cross-layer predictive-model-based DSE of the heterogeneous multi-core processor while considering key parameters of the application, architecture, and the operating system in the evaluation of the HMP configurations.

This chapter specifically presents a cross-layer approach for exploring and configuring a HMP for a given goal under system-level constraints (such as equal area or power budget) based on recent work [45]. Unlike the state-of-the-art approaches, the presented approach jointly considers features of the application, operating system (task allocation strategies), and hardware architecture while deploying computationally efficient predictive models (of performance and power) in composing the HMP platform resources (number and types of cores). The predictive cross-layer approach enables the designer to comparatively evaluate and select the most promising (e.g., energy and performance efficient) HMP configuration in over two order of magnitude less simulation time especially during the early design and verification stages when the design space is at its largest.

8.2.1 Design of Experiments

The term "experiment" [28, 37] concerns situations where we have to organize a systematic scientific procedure to obtain some meaningful information about an object of interest. Design of Experiments (DoE) [1, 43] is an efficient and scientific approach that considers all factors simultaneously, applied to an experimentation to obtain meaningful information and to determine the relationship between factors affecting a process and the output of that process. DoE is a powerful tool that can be used in a variety of experimental platforms where more than one input factor is suspected of influencing an output. DoE allows for multiple input factors to be manipulated determining their effect on a desired output. By manipulating multiple inputs at the same time, DoE can identify important *interactions* that may be missed when experimenting with one factor at a time. Likewise, DoE provides a full insight of interaction between design elements; therefore, it helps turn any ad hoc design process into a robust, predictable process.

The DoE methodology may involve *controllable* and *uncontrollable* input factors. *Controllable* input factors are those input parameters that can be modified in an experiment while *uncontrollable* input factors cannot be changed. These factors need to be recognized to understand how they may affect the output or *response*. DoE provides information about the interaction of these factors and the total system work flow, something not obtainable through testing one factor at a time while maintaining other factors constant. Additionally, DoE shows how interconnected factors respond over a wide range of values, without requiring the testing of all possible values directly. Often, hypothesis testing is performed to determine the significant factors using statistical methods in combination with *orthogonal* vectors or sets of *orthogonal* vectors that are uncorrelated and independent [1]. Before doing the actual experiment, experimental design requires careful consideration of several factors such as number of factors that influence the design, fixed or random levels of these factors, control conditions required in the design process, sample size, number of units collected for the experiment to be generalizable, the relevance of interactions between factors, noise, etc., for the establishment of validity, reliability, and replicability [1]. In order to predict the output responses for any given combination of input values, DoE fits response data to mathematical models or predictive models. With these models, it is possible to optimize critical responses and find the best combination of input values. As measurements are usually subject to variation and measurement uncertainty, *blocking* and *replication* of experiments are adopted to overcome these shortcomings. Blocking is the arrangement of experimental units into groups consisting of units that are similar to one another in order to avoid any unwanted variations in the input or experimental process and thus allows greater precision in the estimation of the source of variation under study. A *randomization* process is used to assign individuals at random to groups or to different groups in an experiment so that each individual of the population has the same chance of becoming a part in the process. Similarly, replication of the experiments (i.e., perform the same combination run more than once) is done in order to identify the sources of variation, to get an estimate for the amount of random error that could be part of the process, and to further strengthen the experiment's reliability and validity.

8.2.2 Response Surface Models

Response Surface Modeling (RSM) techniques allow determining an analytical relationship or dependence between several design parameters and one or more response variables into a mathematical framework typically to rapidly evaluate a system-level metric. The working principle of RSM exploits a set of simulations generated by DoE in order to obtain a predictive model that is also called a response model. A typical RSM flow involves a training phase in which known data (or training set) is used to identify the RSM configuration and a prediction phase in which the RSM is used to forecast or predict unknown system response. These predictive models can be developed using established techniques [40] such as interpolations, linear regression, or artificial neural networks. RSM methodology has been used extensively to study the design space exploration of homogeneous architectures [35, 40]. The next section outlines the methodology specifically adapted for emerging HMP architectures.

8.3 Cross-Layer Predictive Model Building Approach

This section presents a Cross-Layer Design Space Exploration (CLDSE) approach, a methodology that allows evaluation of large architectural design spaces at different levels of abstraction to achieve efficiency (e.g., reducing simulation time by trimming down the large design space into a small finite set of points) and accuracy (gradual refinement of the abstraction models). The cross-layer-based DSE for the HMPs is motivated by the platform-based approach [26, 42] with the difference that the hardware architecture platform and the mapping strategy are varied along with diverse spectrum of applications for given system-level constraints. The presented methodology combines the DoE [43] and predictive model [40] techniques to predict the quality of the nonsimulated design points thereby speeding up the exploration process while reducing the number of required simulations. While the DoE phase generates an initial plan of experiments used to create a coarse view of the target design space to be explored by simulations, the predictive model – a closed-form expression of objective (figure of merit) space as a function of the parameter space – is useful during the DSE phase to quickly converge to the Pareto set of the multi-objective problem without executing lengthy simulations. The modeling and optimization techniques proposed in [33, 40] are used to iteratively update the predictive models (as shown in Fig. 8.2) while simulating different parts of the system stack.

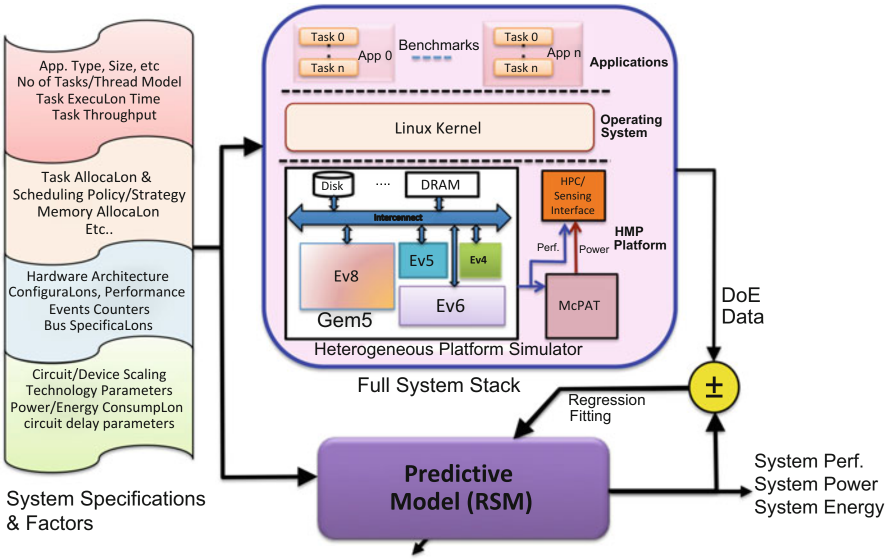

Fig. 8.2 Training phase: cross-layer predictive model building approach

The cross-layer predictive modeling methodology for HMP architectures is divided in two phases. In the training phase, known data (from a training set) are used to identify the predictive model configuration as depicted in Fig. 8.2. A special set of benchmarks are used for coverage of the design space. On the other hand, during the prediction phase, a predictive model is used to forecast the unknown system response as illustrated in Fig. 8.3. The training phase of the cross-layer predictive modeling approach captures the architectural design spaces and behaviors at different levels of abstraction to achieve efficiency (e.g., reducing simulation time by trimming down the large design space into a small finite set of points) and accuracy (gradual refinement of the abstraction models). This approach, illustrated in Fig. 8.2, is motivated by the platform-based approach [26, 42] with the difference that the hardware architecture platform and the mapping strategy are varied along with diverse spectrum of applications for given system-level constraints. The modeling and optimization techniques proposed in [40] are deployed to iteratively update the predictive models (as shown in Fig. 8.2) of different parts of the system stack as discussed in the subsequent sections.

8.3.1 Problem Formulation

Consider a shared memory HMP architecture as shown in Fig. 8.1 consisting of K types of core represented using a set $\Pi = \{\pi_1, \pi_2, \ldots, \pi_K\}, \pi_i \neq \pi_j, K > 1$ having

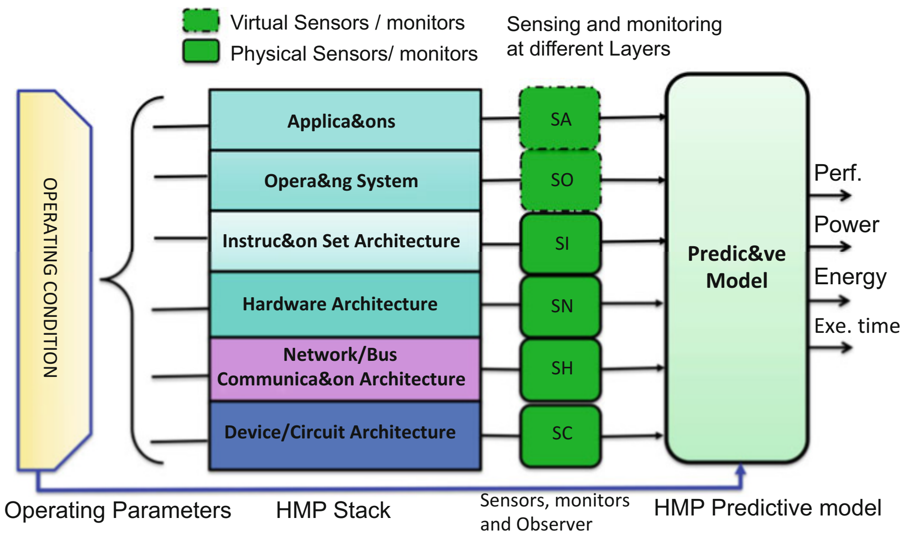
Virtual Sensors / monitors
Physical Sensors/ monitors
Sensing and monitoring at different Layers
OPERATING CONDITION
Applica&ons
Opera&ng System
Instruc&on Set Architecture
Hardware Architecture
Network/Bus Communica&on Architecture
Device/Circuit Architecture
SA
SO
SI
SN
SH
SC
Predic&ve Model
Perf.
Power
Energy
Exe. time
Operating Parameters
HMP Stack
Sensors, monitors and Observer
HMP Predictive model

Fig. 8.3 Prediction phase: use of the cross-layer predictive model using features from different layers of the stack during prediction

corresponding areas $\Lambda = \{a_1, a_2, \ldots, a_K\}$. Let the set of all processing elements be $PE = \{p_1, p_2, \ldots., p_n\}$ where p_j is an instance of a core type in Π. The HMP consists of the core combinations $\mathbf{C} = \{n_1, n_2, .., n_K\}$ such that the total number of processors in the HMP is $n = \sum_{l=1}^{K} n_l$, where n_l is the number of processors of type π_l and the total area $A = \sum_{i=1}^{K} a_i * n_i$. Let $A_{\max}$, $P_{\max}$ be the respective maximum die area and allowable power consumption in the design of the HMP. The goal is to find the HMP core combinations $\mathbf{C}$ such that a platform objective $\mathbf{J}$ (e.g., power/energy efficiency) is optimized under system constraints (such as area or power) as below:

$$\begin{aligned} &\underset{\mathbf{C}}{\text{Maximize}} \quad (\mathbf{J}) \quad . \\ &s.t \quad A \leq A_{\max} \end{aligned} \tag{8.1}$$

As the design space for HMP composition problem in (8.1) is extremely large, a few assumptions and approximations are made to reduce the design space. First, assume that the number of core types K is small (<5). Second, as there is a hard constraint on system area and power resources, the composition space of $\mathbf{C}$ can be reduced to a set of feasible configurations $\mathbb{C} = \{C_1, C_2, .., C_N\}$. Then the problem is to chose one of these C_i for a given set of workloads and OS level workload allocation strategy that optimizes the system goal $\mathbf{J}$.

8.3.2 Application and Workload Models

The workload and their diversity (program phases , CPU, memory, and IO intensive workloads) are modeled using a task-based (thread-based) model and are exposed to the OS using the performance and power/energy characterization matrices where:

- $V = \{v_i\}$ is the set of tasks (or interchangeably used for threads). v_i stands for task i where $1 \leq i \leq m$ and m is the number of tasks. Without loss of generality, a task or group of tasks represents the workload/application. Let N_i represent the computational workload (measured in terms of number of instructions) of task v_i.
- $\mathbf{S} = [\mathbf{ips_i}] = \left\{\text{ips}_{ij}, 1 \leq i \leq m, 1 < j \leq n\right\}$ is the average throughput matrix (measured in terms of instructions per second) when executing the tasks on different processors. $\text{ips}_{ij} (= \text{IPC}_{ij} * F_j)$ represents the average throughput when task v_i executes on processor p_j and is the product of the IPC_{ij} (instruction per cycle) and the frequency of the core F_j. The IPC_{ij} can be measured directly from the processor's built-in performance counters [15].
- $\varGamma = \left[N_i/s_{ij}\right] = \left\{\tau_{ij}, 1 \leq i \leq m, 1 < j \leq n\right\}$ is the average execution time (or the time span) matrix. τ_{ij} is the average execution time of task v_i on processor p_j.
- $\mathbf{P} = [\mathbf{pw_i}] = \left\{pw_{ij}, 1 \leq i \leq m, 1 < j \leq n\right\}$ is the average power consumption matrix of tasks executing on different processors. $\mathbf{pw}_i = \{pw_{ij}, 1 \leq j \leq n\}$ represents a vector of all the average powers of task v_i executing on each processor. pw_{ij} represents the average power of task v_i executing on processor p_j, and it varies with time. The power consumption pw_{ij} of a task v_i can be computed by using combination of performance counters [15].
- $\varXi = \left[\varepsilon_{ij}\right] = \left\{pw_{ij} \times \tau_{ij}\right\}$ is the average energy consumption defined as the product of the average power consumption and execution time.

8.3.3 Heterogeneity-Aware Task Allocation

The task allocation problem of multi-core processors within an HMP architecture consists of finding an optimal distribution of tasks on a set of processors $PE = \{p_1, p_2, \ldots, p_n\}$. It is assumed that each processor runs independently but can only run one task at any instant of time. An assignment of all tasks $V = \{v_1, v_2, \ldots., v_m\}$ to available processors $PE = \{p_1, p_2, \ldots, p_n\}$ a "schedule" $\boldsymbol{\varPsi}$ is represented as:

$$\begin{gathered} \boldsymbol{\varPsi} = \left\{\lambda_j, 1 \leq j \leq n\right\} \\ \lambda_j = \{v_i, 1 \leq i \leq m\}, \forall v_i \in V = \{v_1, v_2, \ldots, v_m\}, \end{gathered} \tag{8.2}$$

where λ_j represents the schedule of set of task for the processor p_j and v_i represents a task among the set of tasks $V = \{v_1, v_2, \ldots, v_m\}$ that is mapped to processor p_j. A schedule as defined in (8.2) will result in a total execution time and power distribution consumption as a function of the task allocation taking into account the

Table 8.1 Heterogeneity-aware task allocation strategies for a given HMP composition

Sl No	Platform design goal	Allocation	Problem definition	Objective function	Nomenclature
1	Performance maximization (PerfMax)	$\min D$	Find Ψ_D $\exists\ J_D$ is minimized	$t_{\text{opt}} = \min\{J_D\} = \min\{\max\{t_j\}\}$ $J_D = \max\{t_j\};\ t_i = \sum_{i=1}^{k} \tau_{ij}$ $1 \le j \le n$	t_j represents total execution time of the tasks in processor p_j
2	Energy minimization (EnergyMin)	$\min E$	Find Ψ_E $\exists\ J_E$ is minimized	$\Xi_{\text{opt}} = \min\{J_E\};\ J_E = \sum_{j=1}^{n} \xi_j$ $\xi_j = \sum_{i=1}^{k} \varepsilon_{ij} = \sum_{i=1}^{k} pw_{ij}.\tau_{ij};$ $1 \le j \le n$	ξ_j represents sum of total energy consumed by k task in processor p_j
3	Power minimization (PowerMin)	$\min ED$	Find Ψ_{ED} $\exists\ J_{ED}$ is minimized	$J_{ED} = \min\{J_E.J_D\}$	Energy delay product
4	Energy efficiency maximization (EEMax)	$\min ED^2$	Find Ψ_{ED^2} $\exists\ J_{ED^2}$ is minimized	$J_{ED^2} = \min\{J_E.J_D^2\}$	Energy delay square product

heterogeneity of processing elements and workload. In other words, for different allocation strategies, the total execution time and energy consumption in the multi-core processor system will be different. Thus, the CLDSE determines a schedule $\boldsymbol{\Psi}$ for the given set of tasks that meets an objective or a performance index as defined in Table 8.1.

8.3.3.1 Optimization Methodology

Finding the optimal allocation that maximizes or minimizes the multiple objective functions is a combinatorial problem; therefore, a solution based on brute force search is not suitable even for a small number of cores and threads due to combinatorial explosion; furthermore, this problem is shown to be NP-hard [49]; thus, polynomial time optimal solutions are not available at all. However, heuristics that exploit specific characteristics of the problem can be adopted to reach acceptable solutions within a reasonable amount of time. Owing to the tremendous diversity of heuristics, a judicious choice of heuristics is critical for finding efficient solutions. Many heuristics often converge to local minima resulting in poor results [12], while others cannot be applied due to the nonlinear nature of the thread allocation objective function (e.g., linear-programming-based approaches [48]). In general, problem structure-dependent heuristics does not provide a generic solution, and a new heuristic formulation has to be obtained for every change in the problem structure (e.g., objective function or constraints).

A more generic approach to such optimization problems have used probabilistic strategies such as *Simulated Annealing (SA)* that have demonstrated the ability to produce close-to-globally-optimal solutions with a moderate complexity in terms of execution time [12]. SA can easily accommodate changes in the problem nature without significant modifications and provide tunable parameters to trade-off computational complexity for solution quality. Furthermore, SA can also be parallelized and distributed to control the computation complexity for extreme scalability.

8.3.3.2 Simulated Annealing-Based Optimization

Simulated Annealing (SA) is a method for solving unconstrained and bound-constrained optimization problems [12]. The method models the physical process of heating a material and then slowly lowering the temperature to decrease defects, thus minimizing the system energy. At each iteration of the SA algorithm, a new point is randomly generated. The distance of the new point from the current point, or the extent of the search, is based on a probability distribution with a scale proportional to the temperature. The algorithm accepts all new points that lower the objective but also accepts points that raise the objective with a certain probability. By accepting points that raise the objective, the algorithm avoids being trapped in local minima and is able to explore globally for more possible solutions. An annealing schedule is selected to systematically decrease the temperature as the algorithm proceeds. As the temperature decreases, the algorithm reduces the extent of its search to converge to a minimum. The SA-based algorithm outlined in Fig. 8.4 is used as the optimization engine for exploring the cross-layer design space of HMPs.

8.3.4 Predictive Modeling of Performance and Power of Different Core Types

The predictive models based on RSM as described in [40, 43] are closed-form analytical expressions suitable for predicting the quality of nonsimulated design points. Predictive model techniques are typically introduced to decrease the time due to the evaluation of the system-level objective function $\mathbf{J}(\mathbf{x})$ for each architecture $\mathbf{x}$. A response surface model for the function $\mathbf{J}(\mathbf{x})$ is an analytical function $\mathbf{r}(\mathbf{x})$ such that

$$\mathbf{J}(\mathbf{x}) = \mathbf{r}(\mathbf{x}) + \epsilon, \tag{8.3}$$

where ϵ is the estimation error. Typically, an appropriate predictive model for $\mathbf{J}(\mathbf{x})$ is such that it has some desired statistical properties such as a mean of zero and small variance. The working principle of a predictive model is to use a set of simulations generated by DoE in order to build the response model of the system. A typical predictive model-based flow involves a) a training phase, in which known data (or training set) are used to identify the predictive model configuration and b) a prediction phase, in which the predictive model is used to forecast the unknown system response. This chapter demonstrates the use of linear regression techniques to construct the predictive model by taking into account the interaction between the

SA Input Params: Temperature T, Temperature schedule c, Maximum number of iterations $Iter_{max}$
Input Data: HMP config **C**, Throughput Matrix **S**, Power Matrix **P**, Execution Time Matrix Γ, Energy Matrix Ξ
Output: Allocation $\boldsymbol{\Psi}$

1. Set an initial solution $\boldsymbol{\Psi} = \boldsymbol{\Psi}_0$
2. Obtain a new solution $\boldsymbol{\Psi'} = \boldsymbol{\Psi}$ and randomly perturb one of the elements $\boldsymbol{\Psi'}_{ji}$ of $\boldsymbol{\Psi'}$. The so-called Bolz-mann generating scheme accomplishes this:

$$\begin{aligned}
&idx = j * m + i \\
&idx = [idx + \sqrt{T} \times rand()] \ \mathbf{mod}\,(n*m) \\
&j' = idx \,\mathbf{mod}\, n,\ i' = (idx - j')/m \\
&swap(\boldsymbol{\Psi'}_{ji}, \boldsymbol{\Psi'}_{j'i'})
\end{aligned}$$

   where $rand()$ generates uniformly distributed random integer numbers.
3. Evaluate the objective function $\boldsymbol{J(C,S,P,\Gamma,\Xi)}$ for $\boldsymbol{\Psi'}$
4. Accept (set $\boldsymbol{\Psi} = \boldsymbol{\Psi'}$) or reject $\boldsymbol{\Psi'}$. If the value of the objective function is lower than before the perturbation, always accept. If it is higher, then accept according to the probabilistic rule

$$accept\ if\ rand() < exp\left(\frac{E_0 - E_p}{T}\right)$$

   where $E_0 - E_p$ is the difference in objective function values before and after the perturbation.
5. Decrease the temperature according to the cooling schedule:

$$T = c \times T$$

   where $0 < c < 1$ is a constant.
6. The algorithm stops when the average change in the objective function is small relative to the tolerance, or when it reaches the maximum number of iterations $Iter_{max}$, otherwise, it goes back to step 2.

Fig. 8.4 Heterogeneity-aware static task allocation using SA

parameters and the quadratic behavior with respect to a single parameter using the general model discussed in [40].

In order to concisely encapsulate the effects of performance, power, and workload behavior, an effective approach is required to determine and represent the performance, power, and the energy matrices as described above. A combination of measurement and on-line prediction is used to construct these matrices. Estimation or the prediction of the performance and power matrices are possible as there is a direct correlation between the behavior of different core types. The key idea behind the estimation and prediction of execution time and power matrix relies on the fact that the performance of a task on one core is correlatable to the performance in another core (with the same ISA and memory hierarchy) with a good degree of accuracy. By measuring the performance of the task in one processor, one can predict the performance in other processors. The execution time τ_{ij} of a task v_i on the processor p_j can be defined as,

$$\begin{aligned}
\tau_{ij} &= \frac{N_i}{\mathrm{IPC}_{ij} * F_j} = \frac{N_i}{\mathrm{ips}_{ij}} \\
\mathrm{IPC}_{ij} &= 1/CPI_{ij}.
\end{aligned} \tag{8.4}$$

Next, core specific performance (throughput) predictors are developed and then combined to obtain performance prediction of the combined total platform. The average throughput IPC_{ij} is for a given task v_i running on processor p_j is predicted by using a linear predictor

$$\text{IPC}_{ij} = \Phi_j * X_{ij}^T, \tag{8.5}$$

where Φ_j is constant vector of a predictive model [2, 24] and $X_{ij}^T = [x_{1i}, x_{2i}, .., x_{qi}]_j^T$ is a characterization vector of core architectural features and hardware counter (cycle counters, instruction counters, performance degradation events) values that is used to predict the performance for the core p_j for the task v_i. The cross-layer features and hardware architecture counters are used in the prediction. The following static features and dynamic hardware performance counters are used:

- **Hardware Architecture Features:** Issue width (I_w), LQ/SQ size (LSQ), IQ size (IQ), ROB size (ROB), Int/float Regs (IFR), L1\$I size (KB) (L_{1I}), L1\$D size (KB) (L_{1D}), Freq. (MHz) (F), voltage (V), core area (a).
- **Performance Events Counters:** The following events are measured that are known to drive the performance of a core [2, 24]: mispredicted branches, which are used to compute the branch misprediction rate (m_B) and instruction/data L1 caches and TLBs misses and hits, which are used to compute the L1 instruction miss rate (m_{L1I}), L1 data cache miss rate (m_{L1D}), instruction TLB miss rate (m_{ITLB}), data TLB miss rate (m_{DTLB}), and Context switch counters (Cw).
- **Cycle and Instruction Counters:** the following cycle counters are sampled: the amount of *busy cycles* (cy_{Busy}), *idle cycles* (cy_{Idle}), and *sleep cycles* (cy_{Sleep}) of a core. *Busy cycles* represent the time a core spends doing computation, *idle cycles* capture idling time due to pipeline stalls or cache misses, and *sleep cycles* capture the time a core spends in a quiescent state. Furthermore, the following instruction counters are sampled: total amount of *committed instructions* (I_{total}), *committed load and stores* (I_{mem}), and *committed branches* (I_{branch}).

Similarly, the power consumption of each task is computed for all the cores by measuring the power consumption in a core and suitably scaling it by the scaling factor among the cores using a linear predictor described below:

$$pw_{ij} = \Theta_j * X_{ij}^T, \tag{8.6}$$

where $\Theta_j = [\theta_1, \theta_2, \ldots, \theta_q]_j$ are constant vectors obtained by fitting the data of the benchmarks and $X_{ij}^T = [x_{1i}, x_{2i}, .., x_{qi}]_j^T$ is the architecture feature and hardware counter (cycle counters, instruction counters, performance degradation events).

The computational complexity (execution time) and accuracy of the predictors for sample benchmarks are shown in Table 8.3.

8.3.5 Training Methodology and Benchmarks

The process of training and the training data used for identifying the parameters is fundamental for creating reasonably accurate prediction models. For this specific approach, training of the predictive models leverages the DoE as discussed in sub-section 8.2.1 by using existing benchmarks such as PARSEC [5], Mediabench-II [32], SPEC 2006 [18], as well as their unique combinations; this training is guided by DoE such that the properties of these experiments are satisfied. These benchmarks and their combinations are used with different parameters, such as levels of parallelization, number of threads, computational load, memory requirements, etc., to excite the platform from different dimensions and systematically collect the response data for training. For instance, PARSEC benchmarks have good Instruction-Level Parallelism (ILP) diversity and are excellent for building predictive models that capture the computing and memory behaviors. However, these benchmark applications are CPU bound and mostly exhibit a constant high load, which may not be ideal for properly evaluating the impact of distinct load contribution patterns. For this reason, a set of synthetic microbenchmarks with attributes that reflect interactive behaviors (I/O dependent applications) and other cross-layer attributes are created and mixed with traditional benchmark suites during the training data collection for the predictive models [38, 46].

The use of specialized microbenchmarks [44, 46] can provide further diversity to accurately capture cross-layer characteristics. For example, in [46] a set of multi-threaded synthetic benchmarks – interactive microbenchmarks (IMB) – enable selective control of the workload, phasic/ bursty behavior, and interactivity (sleep and wait periods). These IMBs can be configured to have throughput (T) and interactivity (I) that control the sleep/wait periods for high(H), medium(M), and low(L) values. Using this approach, a diverse combination of synthetic benchmarks can be generated to stress various dimensions of cross-layer attributes. For instance, the combination “HTHI” represents a high throughput and high interactivity IMB configuration; all other combinations are similarly used in the experiments to capture the cross-layer behavior [38, 46].

8.3.6 Selecting the HMP Configuration

Once the response surface of the system goal is formed using the predictive models, different search heuristics can be used to find the most suitable HMP configuration. As an example, the configuration that performed the best in most cases as the number of threads (or load) increases can be selected by searching the feasible configurations. Other heuristics or optimization criteria can be used to select the configurations from the Pareto front [16, 21].

8.4 Case Study: Experimental Evaluation of Cross-Layer DSE of HMPs

In this section, an example case study of the presented cross-layer approach is illustrated for a contemporary heterogeneous multi-core architecture such as the ARM big.LITTLE [17]), with the goal of quantifying the benefits of different architectural configurations. To emulate different core types, different classes of publicly available Alpha processor models [25, 30] (Table 8.2) are used to construct a realistic HMP model in Gem5 [6] by specifying their multilayer architectural features. Note that the performance of the processors in terms of average IPC, area, and power are normalized with respect to the smallest EV4 (Alpha 21064) core. Also observe that the asymmetric increase of approximately 82× in chip area just to double the performance of an EV8 core with respect to an EV4 core. This asymmetry (or heterogeneity) in scaling is essentially exploited by HMPs to achieve performance, power, and energy efficiency for a given area budget.

To illustrate the CLDSE approach, some experiments can be run by considering chip/die area budget of four Alpha 21264 (EV6) processors as the system-level constraint. Observe that all the possible distinct combinations with three classes of processors (EV4, EV5, and EV6) that meet the area budget are numbered for the 37 possible candidate configurations as shown in Fig. 8.5. To represent a diverse set of workloads, 8 Mediabench-II algorithms [32] and PARSEC benchmarks [5] are selected as representative workloads; their execution time and power consumption are generated using a combination of Gem5 [6] and McPAT[34], respectively, as shown in Fig. 8.6. The performance and power values for each processor core type are generated through full system simulation as shown in Table 8.2. To consider the effect of varying workload and other microarchitectural effects, the number of threads in the benchmark program are varied with different inputs in the cycle accurate full system Gem5 simulation. Here each benchmark is viewed as a single threaded task. The effect of diverse multi-threaded workloads on the

Table 8.2 Alpha processor cores performance, area and power [30]

Alpha core	Issue width	I-cache	D-cache	Branch prediction	# MSHRs	IPC[a]	Area[a]	Peak power(W)	Avg. power(W)	Power[a]
EV4	2	8 KB, DM	8 KB, DM	2 KB, 1-bit	2	1.00	1.00	4.97	3.73	1.00
EV5	4	8 KB, DM	8 KB, DM	2K-, gshare	4	1.30	1.76	9.83	6.88	1.84
EV6	6	64 KB, 2 Way	64 KB, 2 Way	Hybrid, 2 level	8	1.87	8.54	17.8	10.68	2.86
EV8	8	64 KB, 4 Way	64 KB, 4 Way	Hybrid, 2×EV6 size	16	2.14	82.2	92.88	46.44	12.45

[a]Normalized versus EV4; all cores scaled to 0.1 µm, at 2.1 GHz; IPC based on SPEC CPU benchmarks

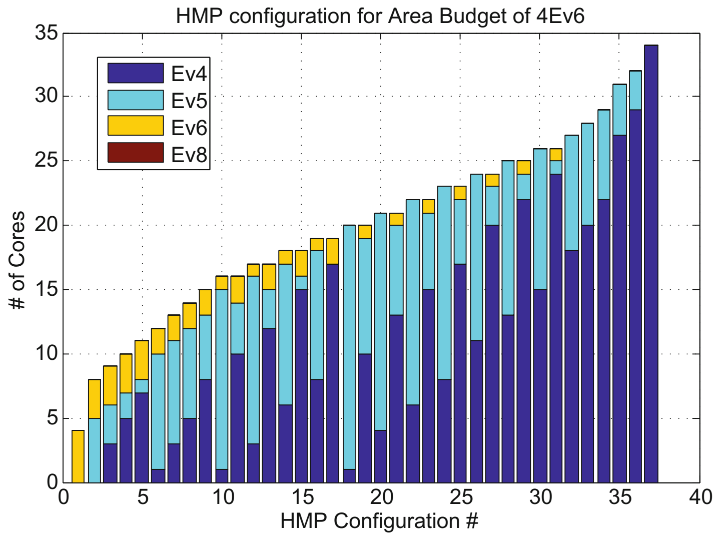

Fig. 8.5 HMP configurations for area budget of 4×EV6. Total of 37 configurations numbered from 1 to 37 from left to right

platform can be simulated by using PARSEC benchmarks or by gradually increasing the number of single threaded tasks and performing the allocation for a given platform. The combination of these benchmarks form new composite tasks (e.g., JPEG compression followed by AES encryption), and the execution time and power consumption of the composite task can be computed as the sum of execution time and power consumption of the individual benchmarks, respectively. This tests architectural configurations with more than 100 cores (e.g., area budget of 4 EV8 results in 46,428 configuration with as many as 330 EV4 cores). With the variability in number of tasks, the objective functions of each architectural combination with system goal $\min D$ is shown in Fig. 8.7.

To demonstrate the ability to cover a large design space, an initial set of simulations generated by DoE is used to build the response surface model of the system for the following design objectives: make-span/delay, power, energy, Energy-Delay Product (EDP), and Energy-Delay Square Product (EDSP) as listed in Table 8.1. These initial simulation points are used to construct the predictive models by using linear regression to obtain the coefficients of the expression (8.5) by performing least square fitting of the data. Table 8.3 shows the computational complexity (execution time) and accuracy of the predictors in comparison to a full system simulation (over two orders of magnitude at maximum prediction error of 10%) of the platform for some sample benchmarks. The presented CLDSE shows that an allocation strategy that performs well with one architectural configuration does not perform equally well for another architectural configuration and there is a rich design space to exploit for a specific solution. The predictive models demonstrate

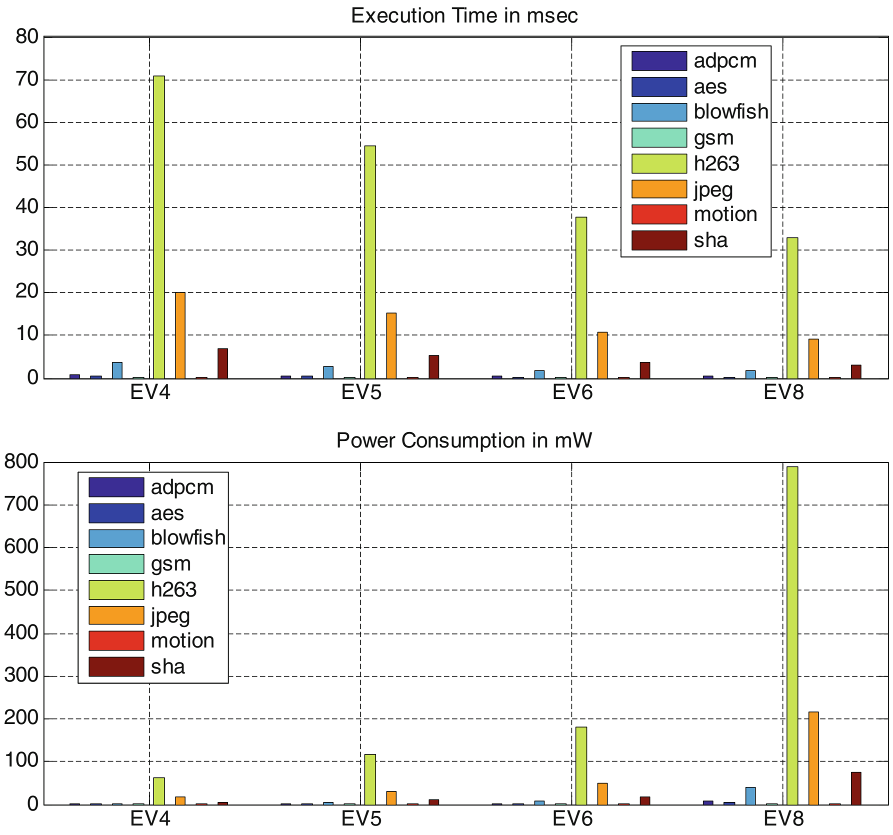

Fig. 8.6 Average power and execution time for eight benchmarks for different Alpha processors

the relative merit for heterogeneous multi-core processor configurations for the same area budget and different allocation strategies. Furthermore, the allocation strategies are compared with a heterogeneity oblivious random allocation with variability in number of task and the execution time as shown in Fig. 8.8. The joint impact of considering the workload variability (with variations in number of tasks and intra-task execution time variations) with allocation strategies shows that almost all the HMP configuration will under-perform by as much as 50 and 70%, respectively, in terms of energy delay product (EDP) and energy delay square product (ED^2) if a heterogeneity agnostic allocation policy (e.g., random policy as in vanilla Linux Kernel) is used. Thus, heterogeneity-aware allocation strategies are crucial for almost any HMP platform configurations, and their impact is significant as the system is loaded with more tasks. This approach can be used to search for the best performing architecture (Table 8.4) as the most preferable architecture (most frequently occurring) for different system goals using a given allocation strategy with the given equal area budget constraints. Observe that for the given area

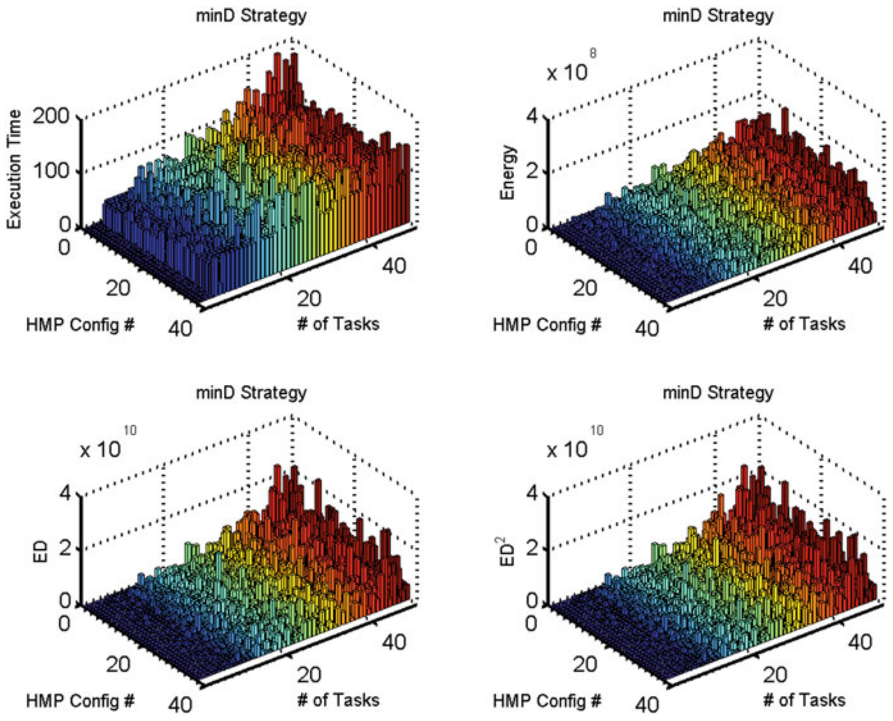

Fig. 8.7 Objectives with variability in number of task for delay only task allocation strategy ($minD$) using the predictive models. Lower is better

Table 8.3 Execution time and prediction model performance on Intel i7 2.4 GHz machines

Benchmarks	HMP config	Gem5 full system simulation time	Prediction model execution time	Prediction error
H.264	#1 (4 cores)	>2 days	< 1 s	<5%
Bodytrack	#2 (8 cores)	>4 days	< 1 s	<5%
Blackscholes	#10 (16 cores)	>7 days	< 1 s	<8%
Fluidanimate	#10 (16 cores)	>7 days	< 1 s	<8%
Mix of above	#36 (32 cores)	>10 days	< 1 s	<10%

budget, the architectural combination #9(8 × EV4, 5 × EV5, 2 × EV6) with 8 EV4 cores, 5 EV5 scores, and 2 EV6 cores has superior performance in terms of EDP and ED^2.

This section has outlined a case study demonstrating the benefits of cross-layer design space exploration of HMPs. These preliminary studies show that much more research is required to exploit this rich and complex space of cross-layer optimizations for emerging heterogeneous architectures.

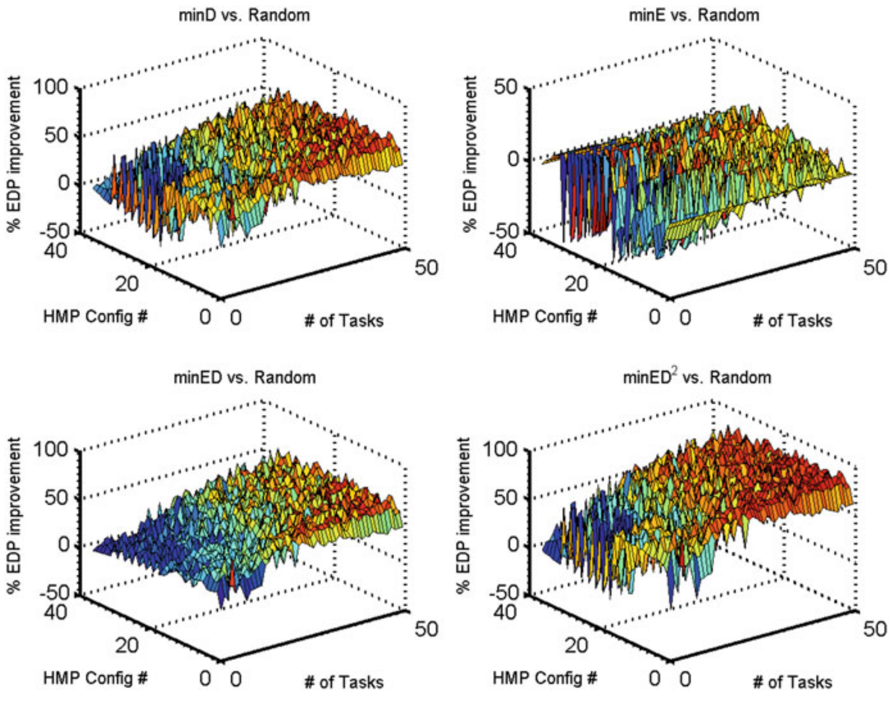

Fig. 8.8 DSE with predictive models: comparison of SA-based static allocation strategies with random allocation strategy (as in vanilla Linux). Higher is better

Table 8.4 Preferred architectural composition with different system goals and allocation strategies

Goals/objective	J_D (min$D$)	J_E (min$E$)	J_{ED}(minED)	J_{ED^2}(minED^2)
$\mathbf{C_{PerfMax}}$	#1 (4 × EV6)	#1(4 × EV6)	#1(4 × EV6)	#1(4 × EV6)
$\mathbf{C_{EnergyMin}}$	#37(34 × EV4)	#9(8 × EV4, 5 × EV5, 2 × EV6)	#37(34 × EV4)	#37(34 × EV4)
$\mathbf{C_{PowerMin}}$	#9(8 × EV4, 5 × EV5, 2 × EV6)	#37(34 × EV4)	#2(5 × EV5, 3 × EV6)	#2(5 × EV5, 3 × EV6)
$\mathbf{C_{EEMax}}$	#9(8 × EV4, 5 × EV5, 2 × EV6)	#37(34 × EV4)	#9(8 × EV4, 5 × EV5, 2 × EV6)	#9(8 × EV4, 5 × EV5, 2 × EV6)

8.5 Conclusions

Cross-layer architectural design space exploration presents significant opportunities for architects to comparatively evaluate design choices early in the design, while accounting for complex interactions between constraints at multiple abstraction levels. This chapter presented an exemplar case study for the exploration of single-chip, single-ISA heterogeneous multi-core processors.

Recent research has highlighted the potential benefits of single-ISA heterogeneous multi-core processors over cost-equivalent homogeneous ones, and it is likely that future processors will integrate cores that have the same ISA but offer different performance and power characteristics. However, there are few efforts that address the problem of HMP composition, constituting of different core types. This chapter presented a cross-layer (across application, operating system, and hardware architecture layer) approach of single-ISA heterogeneous multi-core processors using predictive models to investigate the interactions and influence of heterogeneity of hardware architectures (configurations, number, and types of cores) and multi-objective allocation strategies along with diverse types of workloads under system-level constraints (such as equal area or power budget). A versatile and realistic approach was outlined along with clear methodology to build cross-layer predictive models of application and system interactions that can be used in the HMP compositions. The presented cross-layer approach quantifies the relative merits of one architectural configuration and allocation strategy over others and helps in selecting most promising heterogeneous architectures. The predictive cross-layer approach enables the chip architect and designer to comparatively evaluate and select the most promising (e.g., energy and performance efficient) HMP configuration in over two order of magnitude less simulation time compared to a full system simulation especially during the early design and verification stages when the design space is at its largest. The approach embodied in this chapter should be applicable for design space exploration of many emerging programmable architectures.

Acknowledgments This work was partially supported by the NSF Variability Expedition award CCF-1029783.

References

1. Anderson MJ, Whitcomb PJ (2000) Design of experiments. Wiley Online Library. doi: 10.1002/0471238961.0405190908010814.a01.pub3. http://onlinelibrary.wiley.com/doi/10.1002/0471238961.0405190908010814.a01.pub3/abstract. Accessed Sep 2010
2. Annamalai A, Rodrigues R, Koren I, Kundu S (2013) An opportunistic prediction-based thread scheduling to maximize throughput/watt in amps. In: 2013 22nd international conference on parallel architectures and compilation techniques (PACT), pp 63–72. doi: 10.1109/PACT.2013.6618804
3. Balakrishnan S et al (2005) The impact of performance asymmetry in emerging multicore architectures. SIGARCH Comput Archit News 33(2):506–517. doi: 10.1145/1080695.1070012
4. Becchi M et al (2006) Dynamic thread assignment on heterogeneous multiprocessor architectures. In: Proceedings of the 3rd conference on computing frontiers, CF '06. ACM, New York, pp 29–40. doi: 10.1145/1128022.1128029
5. Bienia C et al (2008) The parsec benchmark suite: characterization and architectural implications. In: Proceedings of the 17th international conference on parallel architectures and compilation techniques. ACM, pp 72–81

6. Binkert N et al (2011) The gem5 simulator. SIGARCH Comput Archit News 39(2):1–7. doi: 10.1145/2024716.2024718
7. Chen T, Chen Y, Guo Q, Zhou ZH, Li L, Xu Z (2013) Effective and efficient microprocessor design space exploration using unlabeled design configurations. ACM Trans Intell Syst Technol (TIST) 5(1):20
8. Chen J et al (2009) Efficient program scheduling for heterogeneous multi-core processors. In: 46th ACM/IEEE design automation conference, 2009, DAC '09, pp 927–930
9. Chen T, Guo Q, Tang K, Temam O, Xu Z, Zhou ZH, Chen Y (2014) Archranker: a ranking approach to design space exploration. In: 2014 ACM/IEEE 41st international symposium on computer architecture (ISCA). IEEE, pp 85–96
10. Chitlur N, Srinivasa G, Hahn S, Gupta P, Reddy D, Koufaty D, Brett P, Prabhakaran A, Zhao L, Ijih N et al (2012) Quickia: exploring heterogeneous architectures on real prototypes. In: 2012 IEEE 18th international symposium on high performance computer architecture (HPCA). IEEE, pp 1–8
11. Cook H, Skadron K (2008) Predictive design space exploration using genetically programmed response surfaces. In: Proceedings of the 45th annual design automation conference. ACM, pp 960–965
12. Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester
13. Dubach C, Jones T, O'Boyle M (2007) Microarchitectural design space exploration using an architecture-centric approach. In: Proceedings of the 40th annual IEEE/ACM international symposium on microarchitecture. IEEE Computer Society, pp 262–271
14. Eeckhout L, Vandierendonck H, Bosschere K (2002) Workload design: selecting representative program-input pairs. In: Proceedings of the 2002 international conference on parallel architectures and compilation techniques. IEEE, pp 83–94
15. Ge R et al (2010) Powerpack: Energy Profiling and analysis of high-performance systems and applications. IEEE Trans Parallel Distrib Syst 21(5):658–671. doi: 10.1109/TPDS.2009.76
16. Givargis T, Vahid F, Henkel J (2002) System-level exploration for pareto-optimal configurations in parameterized system-on-a-chip. IEEE Trans Very Large Scale Integr Syst 10(4):416–422
17. Greenhalgh P (2011) Big.little processing with arm cortex-a15 & cortex-a7: improving energy efficiency in high-performance mobile platforms. http://www.arm.com/files/downloads/big.LITTLE_Final.pdf
18. Henning JL (2006) Spec cpu2006 benchmark descriptions. ACM SIGARCH Comput Archit News 34(4):1–17
19. Hill M et al (2008) Amdahl's law in the multicore era. Computer 41(7):33–38. doi: 10.1109/MC.2008.209
20. İpek E, McKee SA, Caruana R, de Supinski BR, Schulz M (2006) Efficiently exploring architectural design spaces via predictive modeling. SIGPLAN Not 41(11):195–206. doi: 10.1145/1168918.1168882
21. Ipek E, McKee SA, Singh K, Caruana R, Supinski BRd, Schulz M (2008) Efficient architectural design space exploration via predictive modeling. ACM Trans Archit Code Optim (TACO) 4(4):1
22. Jin Z, Cheng AC (2008) Improve simulation efficiency using statistical benchmark subsetting: an implantbench case study. In: Proceedings of the 45th annual design automation conference. ACM, pp 970–973
23. Joseph P, Vaswani K, Thazhuthaveetil MJ (2006) Construction and use of linear regression models for processor performance analysis. In: The twelfth international symposium on high-performance computer architecture. IEEE, pp 99–108
24. Kenzo VC et al (2012) Scheduling heterogeneous multi-cores through performance impact estimation (PIE). In: International symposium on computer architecture, ISCA'12
25. Kessler R (1999) The alpha 21264 microprocessor. IEEE Micro 19(2):24–36. doi: 10.1109/40.755465
26. Keutzer K et al (2000) System-level design: orthogonalization of concerns and platform-based design. IEEE Trans Comput-Aided Des Integr Circuits Syst 19(12):1523–1543. doi: 10.1109/43.898830

27. Khan S, Xekalakis P, Cavazos J, Cintra M (2007) Using predictivemodeling for cross-program design space exploration in multicore systems. In: Proceedings of the 16th international conference on parallel architecture and compilation techniques. IEEE Computer Society, pp. 327–338
28. Kleijnen JP (2008) Design and analysis of simulation experiments, vol 20. Springer, New York/London
29. Kumar R et al (2004) Single-ISA heterogeneous multi-core architectures for multithreaded workload performance. In: Proceedings of the 31st annual international symposium on computer architecture, pp 64–75. doi: 10.1109/ISCA.2004.1310764
30. Kumar R et al (2005) Heterogeneous chip multiprocessors. Computer 38(11):32–38. doi: 10.1109/MC.2005.379
31. Lee BC, Brooks DM (2006) Accurate and efficient regression modeling for microarchitectural performance and power prediction. In: ACM SIGPLAN notices, vol 41. ACM, pp 185–194
32. Lee C, Potkonjak M, Mangione-Smith WH (1997) Mediabench: a tool for evaluating and synthesizing multimedia and communicatons systems. In: Proceedings of the 30th annual ACM/IEEE international symposium on microarchitecture. IEEE Computer Society, pp 330–335
33. Lee BC, Collins J, Wang H, Brooks D (2008) Cpr: composable performance regression for scalable multiprocessor models. In: 2008 41st IEEE/ACM international symposium on microarchitecture, 2008, MICRO-41. IEEE, pp 270–281
34. Li S et al (2009) Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In: 42nd annual IEEE/ACM international symposium on microarchitecture, 2009, MICRO-42, pp 469–480
35. Liu HY, Carloni LP (2013) On learning-based methods for design-space exploration with high-level synthesis. In: Proceedings of the 50th annual design automation conference. ACM, p 50
36. Liu G et al (2013) Dynamic thread mapping for high-performance, power-efficient heterogeneous many-core systems. In: 2013 IEEE 31st international conference on computer design (ICCD), pp 54–61. doi: 10.1109/ICCD.2013.6657025
37. Montgomery DC: Design and analysis of experiments. Wiley, Hoboken (2008)
38. Mück T, Sarma S, Dutt N (2015) Run-DMC: runtime dynamic heterogeneous multicore performance and power estimation for energy efficiency. In: Proceedings of the 10th international conference on hardware/software codesign and system synthesis. IEEE, pp 173–182
39. NVidia (2011) Variable SMP – a multi-core CPU architecture for low power and high performance. http://www.nvidia.cn/content/PDF/tegra_white_papers/Variable-SMP-A-Multi-Core-CPU-\Architecture-for-Low-Power-and-High-Performance-v1.1.pdf
40. Palermo G, Silvano C, Zaccaria V (2009) Respir: a response surface-based pareto iterative refinement for application-specific design space exploration. IEEE Trans Comput-Aided Des Integr Circuits Syst 28(12):1816–1829. doi: 10.1109/TCAD.2009.2028681
41. Phansalkar A, Joshi A, John LK (2007) Subsetting the spec CPU2006 benchmark suite. ACM SIGARCH Comput Archit News 35(1):69–76
42. Pimentel A et al (2006) A systematic approach to exploring embedded system architectures at multiple abstraction levels. IEEE Trans Computers 55(2):99 – 112. doi: 10.1109/TC.2006.16
43. Santner TJ, Notz W, Williams B (2003) The design and analysis of computer experiments. Springer, New York
44. Sarma S (2016) Cyber-physical-system-on-chip (CPSoC): an exemplar self-aware SoC and smart computing platform
45. Sarma S, Dutt N (2015) Cross-layer exploration of heterogeneous multicore processor configurations. In: 2015 28th international conference on VLSI design (VLSID), pp 147–152. doi: 10.1109/VLSID.2015.30
46. Sarma S, Muck T, Bathen LAD, Dutt N, Nicolau A (2015) Smartbalance: a sensing-driven linux load balancer for energy efficiency of heterogeneous mpsocs. In: Proceedings of the 52nd annual design automation conference, DAC '15. ACM, New York, pp 109:1–109:6. doi: 10.1145/2744769.2744911
47. Shelepov D et al (2009) Hass: a scheduler for heterogeneous multicore systems. SIGOPS Oper Syst Rev 43(2):66–75. doi: 10.1145/1531793.1531804

48. Teodorescu R, Torrellas J (2008) Variation-aware application scheduling and power management for chip multiprocessors. SIGARCH Comput Archit News 36(3):363–374. doi: 10.1145/1394608.1382152
49. Vidyarthi DP et al (2009) Scheduling in distributed computing systems: analysis, design & models, a research monogram. Springer
50. Wu W, Lee BC (2012) Inferred models for dynamic and sparse hardware-software spaces. In: Proceedings of the 2012 45th annual IEEE/ACM international symposium on microarchitecture. IEEE Computer Society, pp 413–424
51. Yi JJ, Lilja DJ, Hawkins DM (2003) A statistically rigorous approach for improving simulation methodology. In: Proceedings of the ninth international symposium on high-performance computer architecture, 2003, HPCA-9 2003. IEEE, pp 281–291