Center for Embedded Computer Systems
University of California, Irvine

Cross-Layer Design Space Exploration of Heterogeneous Multicore Processors With Predictive Models

Santanu Sarma and N. Dutt

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-2620, USA

{santanus, dutt}@uci.edu

CECS Technical Report No: CECS TR-14-02

Jan 30, 2014

Cross-Layer Design Space Exploration of Heterogeneous Multicore Processors With Predictive Models

Santanu Sarma and Nikil Dutt

***Abstract*—Heterogeneous multicore processors (HMP) present significant advantages over homogenous multiprocessor due to their improved power, performance, and energy efficiency for a given chip/die area. However, to fully tap their potential, a systematic exploration of their diverse and vast design space is necessary. In this paper, we present a cross-layer (across application, operating system, and hardware architecture layer) design space exploration (DSE) of single-ISA (Instruction Set Architecture) heterogeneous multicore processors. We deploy predictive models to investigate the interactions and influence of heterogeneity (configurations, number and types of cores), multi-objective allocation strategies, along with diverse types of workloads under system level constraints (such as equal area or power budget). We perform a cross-layer DSE study of heterogeneous multicore processors with four simulated annealing (SA) based task allocation strategies that are more generic and efficient in comparison to a baseline homogenous allocation. Our cross-layer DSE enables the designer to comparatively evaluate and select the most promising (energy and performance efficient) heterogeneous architecture. We illustrate the usefulness and applicability of our DSE approach especially during the early design and verification stages of HMP's when the design space is at its largest.**

***Index Terms*—Design Space Exploration, Cross-Layer Design, Heterogeneous multicore processor, Task Allocation and Scheduling, Multi-Processor Systems-on-Chip.**

I. Introduction

SINGLE-CHIP single-ISA based heterogeneous multicore processors (HMP) are increasingly considered as an attractive design alternative to homogeneous multiprocessors 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 / figures of merits (e.g., performance, power, energy, throughput, area, cost etc.) [3], [16], [14], [9]. Since different workloads (e.g. CPU bound, integer-intensive, floating-point intensive, memory intensive etc.) require different resources, some key issues are to determine and select the right types and number of cores (processing elements), allocation strategies that assign the workload (or tasks) to right core type, and the type of workload that 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 can have unprecedented forms of heterogenous parallelism, time varying program phases [26], and dynamic as well as diverse workloads. Consequently, there exists a rich, complex, and large design space, incomprehensible by mere intuition and past experience, to be explored and exploited.

In this paper, we perform cross-layer (across application, operating system, and hardware architecture layer) design space exploration (DSE) of single-ISA heterogeneous multicore processors to investigate the interactions and influence of heterogeneous hardware architectures (configurations, number and types of cores), multi-objective allocation or mapping strategies with diverse types of workloads under system level constraints (such as equal area or power budget). Heterogeneity essentially introduces added flexibility at different layers of the system stack especially in the operating system (OS) [25]. We believe cross-layer DSE is critical during the early design and verification stages of the HMP when the design space is at its largest. Cross-layer DSE of HMPs enable designers to quantitatively compare alternative designs, fine tune configurable system parameters, select candidate architectures, explore the interactions of tradable solutions with respect to the design criteria, and exploit the impact (sensitivity) of a design parameter to quickly steer the design to optimal (or approximated Pareto-optimal) solutions. Without cross-layer DSE, a limited exploration (e.g., fixed platform, selected workload, and allocation policy) may only reveal suboptimal design points and completely miss more beneficial and possibly counter intuitive design points.

The authors are with the Department of Computer Science, University of California, Irvine, USA, e-mail: santanus,dutt@uci.edu.

A. Contributions

We perform cross-layer DSE of heterogeneous multicore processors using predictive modes that considers multiple hardware architecture platform configurations, allocation strategies, and diverse set of workloads under system-level constraints. Our cross-layer DSE is complemented with four generalized simulated annealing (SA) based task allocation strategies that are not limited by the types of objectives (can handle both scalar/single [14], [15], [17], [10], [20] as well as vector/multiple objective functions without the need for linearity as in ILP based techniques [28]), number of core types (more than 2 types, unlike only small and big as considered in [10]), task to core ratio (even when > 1 unlike [14], [15], [4], [29], [25], [19], [13]), and complexity of the allocation strategy as the system scales with the number of cores (which is a major scalability concern [25] for the schemes in [4], [14]). Our SA based strategies perform better in terms of energy delay product (EDP) and energy delay square product (ED^2) by as much as 50 % and 70% respectively across all the architectures in comparison to a heterogeneity agnostic allocation policy. Besides, our cross-layer DSE reveals relative merits of one architectural configuration and allocation strategy over the other and enables selection of the most promising heterogeneous architectures. This helps in selecting the most performance and energy efficient architecture for a given allocation strategy as well as assessing robustness of a candidate architecture with workload variability.

II. Related Work

HMPs that integrate a mix of small power-efficient cores and big high-performance cores are very attractive alternative to homogenous multiprocessors because they have the potential for higher performance and reduced power consumption. Recent work illustrating the potential of the heterogeneous multicore processor for dramatic improvement in energy and power efficiency [15], [7], [5], [13], [4], [19], [20] have influenced adoption in several commercial products.

Some of these products include CPU and GPU integration (e.g. NVidia Tegra, Intel Sandy Bridge, and AMD Fusion), while others incorporate some form of CPU and accelerators combinations (e.g. IBM's Cell). More recent examples of commercial offerings include integration of processors of different types with same-ISA (e.g., ARMs big.LITTLE chip [8], and NVidias Kal-El [21]). Processor cores in a heterogeneous multicore system can differ in static micro-architecture to dynamic behavior or modes of operation (e.g., frequency or operating voltage). We are seeing a plethora of devices that integrate a range of processors, from single-ISA cores (that only varying in clock frequency), to single-ISA cores differing in macro-(e.g., in-order, out-of-order) to micro-architecture (e.g., cache size, branch predictors, issue width, etc), to cores with non-identical ISAs. However, since we focus on single-ISA architectures, we consider only this class of heterogeneity as illustrated in Fig. 1.

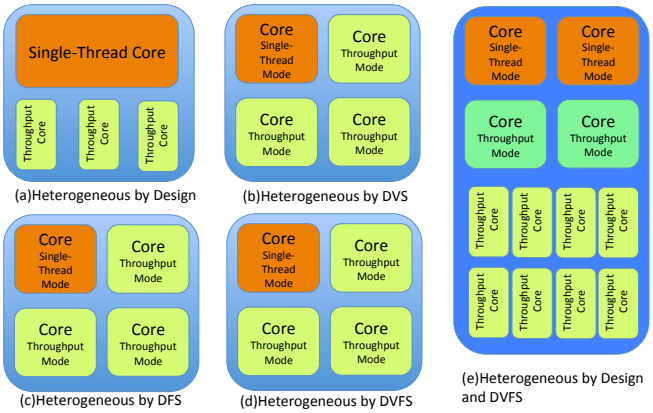


Figure 1. Examples of Heterogeneous Architectures: (a) by design (b) by DVS (c) by DFS (d) by DVFS (e) by design and DVFS.

The case for single-ISA HMPs was made by Kumar et al. [15] where they demonstrated that scheduling an application across core types based on the time-varying execution behavior can yield substantial energy saving. They evaluated both static and dynamic scheduling using multi-programmed workloads along with a sampling based approach [14]. The sampling based dynamic scheduling approach explored in [14] samples the workload to find the most energy-efficient core suitable for the application. Becchi et al. [4] also explored a sampling based dynamic scheduling approach with the aim of performance improvement. On the other hand, a program's bias towards a big versus small core based on memory stall counts has been used in Bias scheduling [13]. However, the schemes in [15], [14], [16], [4], [13] only consider few distinct heterogeneous architectural configurations and a specific task allocation strategy that is limited to a scalar objective of either performance or power optimization for tasks less than or equal to the number of cores. Our approach is more exhaustive in terms of architectural configurations, multi-objective allocations, and diverse spectrum of workloads in comparison to [15], [14], [16], [4], [13] ; We are not limited by the types of objectives, number of core types, and scalability concerns. It has been shown by Shelepov et al. [25], that a static scheduling scheme, HASS, with the primary motivation of scalability, can outperform the schemes in [4], [14] by avoiding frequent migration and periodic sampling overheads. Unlike the dynamic schemes in [4], [14], it uses a static scheduling policy and offline profiling. Even though the offline schemes are simple to design and implement, they lack the flexibility to adapt to dynamic execution modes or multiple usage scenarios. Consequently, we use a combination of measurement and online prediction as opposed to sampling to achieve the performance and power signatures used in [25], [4], [15] of each workload for use in our task allocation strategies.

Several other dynamic scheduling heuristics [17], [19], [7] have also been recently studied for heterogeneous systems along with few studies with dynamic clock frequency variations[25], [27]. The effect of micro-architecture on memory level parallelism has also been evaluated for different core types [23]. However, none of these existing bodies of work have investigated in-depth DSE of HMP's to address the cross-layer interactions and influence of heterogeneity of architectures (configurations, number and types of cores, and micro-architectural features such as cache size, branch predictors, etc), multi-objective allocation or mapping strategies with diverse types of workloads under system level constraints. In this paper, we address this aspect and illustrate the use and applicability of the approach by using realistic workloads and case studies.

III. Cross-Layer Design Space Exploration (CLDSE) of HMP's

Design Space Exploration (DSE) involves 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). Our cross-layer based DSE for the HMPs as illustrated in Fig. 2 is motivated by the platform based approach [11], [24] with the difference that the hardware architecture platform and the mapping strategy is varied along with diverse spectrum of applications for given system level constraints. In other words, we can say that the platform based approach is a specific instance of our cross-layer DSE approach which is more generic and broad. This generic nature of the cross-layer DSE results in a wide spectrum of requirements with a large design space consisting of several parameters at different layers of the system stack. Our methodology combines the design of experiments (DoEs) [1] and precitive model [22] 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, which is a closed-form expression of objective (figure of merit) space as a function of the parameter space is useful during the design space exploration (DSE) phase to quickly converge to the Pareto set of the multiobjective problem without executing lengthy simulations. We use the modeling and optimization techniques proposed in [22] to iteratively update the predictive models (as shown in Fig. 2b) while simulating different parts of the system stack as discussed in the subsequent sections.

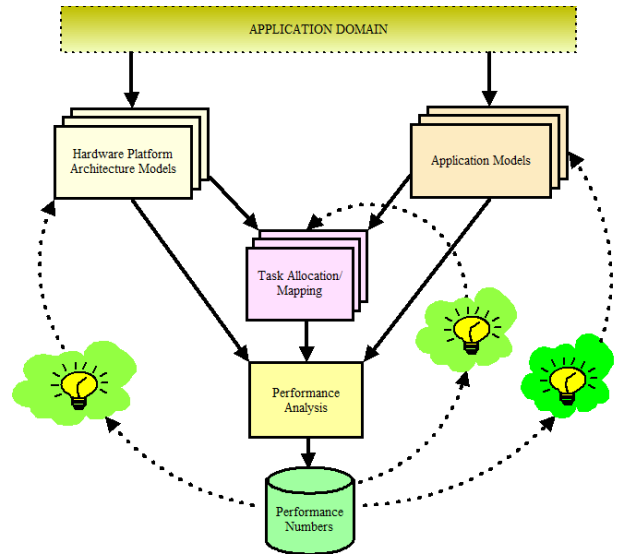
We now describe different aspects of cross-layer heterogeneity (architecture, applications, allocation strategies) for our problem statement.

A. Hardware Architecture Models

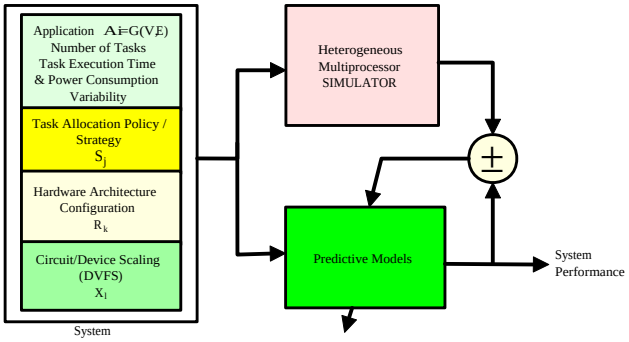
The heterogeneous hardware architecture consists of different classes of processors represented using a set $PC = \{\pi_1, \pi_2, ..., \pi_K\}, \pi_i \neq \pi_j$ where $K > 1$. We assume that all the processor cores are connected through a high-speed on-chip network. We define the set of all processing elements as $PE = \{p_1, p_2, ..., p_n\}$ where p_j is an instance of a processor class in PC. The total number of processors in the heterogenous architecture is $n = \sum n_l$, where n_l is the number of processors of class π_l. When all the cores in the set PE is identical, then the architecture is homogeneous by design but can again be heterogeneous due to dynamic voltage and frequency scaling as illustrated in Fig.1.

B. Application Models

The workload diversity and and existence of different phases within a single workload [26] are well know and is often exploited. Such diversities are usually referred to as either CPU or memory



(a) Y-Chart model



(b) Cross-Layer DSE with Predictive Models

Figure 2. Cross-Layer Design Space Exploration Approach.

intensive workloads. We model this diversity in workload with heterogeneous tasks using a directed acyclic graph (DAG) $G = (V, E)$ where:

- $V = \{v_i\}$ is the set of vertices, representing all the tasks. v_i stands for task i where $1 \leq i \leq m$ and m is the number of tasks which is varied to model workload diversity. $E = \{e_{ij}\}$ is the set of directed edges, representing the order and the data dependence among tasks.
- $N = \{N_i\}$ is the set of computational workload of tasks. N_i represents the computational workload (measured in terms of number of instructions) of task v_i.

We model the diversity in workload, platform, etc. using a set of matrices:

- $\mathbf{S} = [\mathbf{s_i}] = \{s_{ij}, 1 \leq i \leq m, 1 < j \leq n\}$ is the average speed or throughput matrix (measured in terms of instructions per sec) when executing the tasks on different processors. $s_{ij} (= IPC_{ij} * F_j)$ 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 [6].
- $\mathbf{\Gamma} = [N_i/s_{ij}] = \{\tau_{ij}, 1 \leq i \leq m, 1 < j \leq n\}$ is the average execution time (or the time span) matrix. τ_{ij} in average execution time of task v_i on processor p_j. The execution time of each task varies with time (due to program phase and IPC changes) and can be modeled using a statistical distribution [12] with mean $\mu_\tau = \tau_{ij}$ and variance σ_j^2.
- $\mathbf{P} = [\mathbf{pw_i}] = \{pw_{ij}, 1 \leq i \leq m, 1 < j \leq n\}$ 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 as in [6].
- $\mathbf{\Xi} = [\varepsilon_{ij}] = \{pw_{ij} \times \tau_{ij}\}$ is the average energy consumption defined as the product of the average power consumption and execution time.

C. Estimation of the Execution Time and Power Matrices

In order to concisely encapsulate the effects of performance, power, and workload variability we need an effective approach to determine and represent the performance, power, and the energy matrices as described above. We use combination of measurement and on-line prediction 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 execution time behavior of a task on one core is correlatable (proportional) to the execution time in another core (with same-ISA and memory hierarchy) with a good degree of accuracy. By measuring the execution time of the task in one processor we can predict the execution time in the other processors. The execution time τ_{ij} of a task v_i on the processor p_j can be defined as

$$\begin{gathered}\tau_{ij} = CPI_{ij} * N_i * T_{cj} = \frac{CPI_{ij} * N_i}{F_j} = \frac{N_i}{IPC_{ij} * F_j} = \frac{N_i}{s_{ij}} \\ CPI_{ij} = \frac{1}{IPC_{ij}};\ T_{cj} = \frac{1}{F_j}\end{gathered} \tag{1}$$

where CPI_{ij} is the cycle per instructions, T_{cj} is the clock time period. The execution time of the task v_i on processor p_k can thus be computed as

$$\tau_{ik} = \frac{CPI_{ik} * N_i}{F_k} = \frac{CPI_{ij} * N}{\bar{C}_{jk} * F_k} = \frac{\tau_{ij}}{\bar{C}_{jk}} * \frac{F_j}{F_k}. \tag{2}$$

In ideal scenarios where the processor frequencies F_j, F_k are known and $\bar{C}_{jk}$ is known to be the ratio of the CPI of processor p_j to the another processor $p_k, j \neq k$. Thus, the execution time of the task v_i on all the other processors can easily be computed by measuring the execution time in one processor on-line using built-in performance counters[6]. Similarly, we also compute the power consumption of each task for all the cores by measuring the power consumption in a core and suitably scaling it by the scaling factor among the cores.

D. Allocation Strategies

The task allocation problem of multicore processor consists of finding an optimal distribution of tasks on a set of processors $PE = \{p_1, p_2, ..., p_n\}$. It is assumed that each processor runs independently, but can only run one task at any instant of time. We call an assignment of all tasks $V = \{v_1, v_2, ..., v_m\}$ to available processors $PE = \{p_1, p_2, ..., p_n\}$ a "schedule" $\mathbf{\Psi}$ represented as:

$$\begin{gathered}\Psi = \{\Psi_j, 1 \leq j \leq n\} \\ \Psi_j = \{\psi_i, 1 \leq i \leq m\}, \forall \psi_i \in V = \{v_1, v_2, ..., v_m\},\end{gathered} \tag{3}$$

where Ψ_j represent the schedule of set of task for the processor p_j and ψ_i represents a task among the set of tasks $V = \{v_1, v_2, ..., v_m\}$ that is mapped to processor p_j. A schedule as defined in (3) will result in a total execution time and power distribution consumption as a function of the task allocation taking into account the heterogeneity of processing elements and workload. In other words, for different allocation strategies, the total execution time and energy consumptions in the multicore processor system will be different. Thus, the cross-layer DSE determines a schedule $\mathbf{\Psi}$ for the given set of tasks that meets an objective or a performance index as defined in Table I.

Table I
TASK ALLOCATION STRATEGIES

Sl No	Allocation Strategy	Problem Definition	Objective Function	Nomenclature
1	Dealy Only Minimization (minD)	Find $\mathbf{\Psi_D}$ $\exists\ J_D$ is minimized	$t_{opt} = min\{J_D\} = min\{max\{t_j\}\}$ $J_D = max\{t_j\};\ t_i = \sum_{i=1}^{k} \tau_{ij}, 1 \leq j \leq n$	t_j represents total execution time of the tasks in processor p_j
2	Energy Only Minimization (minE)	Find $\mathbf{\Psi_E}$ $\exists\ J_E$ is minimized	$\Xi_{opt} = min\{J_E\};\ J_E = \sum_{j=1}^{n} \xi_j$ $\xi_j = \sum_{i=1}^{k} \varepsilon_{ij} = \sum_{i=1}^{k} pw_{ij}.\tau_{ij}; 1 \leq j \leq n$	ξ_j represents sum of total energy consumed by k task in processor p_j
3	Energy $\times$ Dealy Minimization (minED)	Find $\mathbf{\Psi_{ED}}$ $\exists\ J_{ED}$ is minimized	$J_{ED} = min\{J_E.J_D\}$	Energy Delay Product (EDP)
4	Energy $\times$ Dealy ^{2} Minimization (minED^2)	Find $\mathbf{\Psi_{ED^2}}$ $\exists\ J_{ED^2}$ is minimized	$J_{ED^2} = min\{J_E.J_D^2\}$	Energy Delay Square Product (EDSP)

IV. PREDICTIVE MODELING

The predictive models based on response response surface model (RSM) as described earlier is a closed-form analytical expression 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

$$\mathbf{J}(\mathbf{x}) = \mathbf{r}(\mathbf{x}) + \epsilon \tag{4}$$

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 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 training phase, in which known data (or training set) are used to identify the predictive model configuration; and a prediction phase, in which the predictive model is used to forecast the unknown system response. In this paper, we use linear regression techniques to construct the predictive model by taking into account the interaction between the parameters and the quadratic behavior with respect to a single parameter with the following general model [22]:

$$\mathbf{r}(\mathbf{x}) = \alpha_0 + \sum_{j=1}^{n} \alpha_j x_j^2 + \sum_{l=1}^{n} \sum_{j=1, j\neq 1}^{n} \beta_{l,j} x_l x_j + \sum_{j=1}^{n} \gamma_j x_j. \tag{5}$$

V. SIMULATION AND EXPERIMENTAL RESULTS

To demonstrate the power and efficacy of our cross-layer DSE approach we carryout several experiments that are representative of recent heterogeneous multicore architectures (for example ARMs big.LITTLE chip [8]) and quantify the benefits of different architectural configurations. We use different classes of Alpha Processors [15], [16] whose specifications are given in Table II to construct realistic HMP like the big.LITTLE. Note that the performance of the processors in terms of average IPC, Area, and Power are normalized w.r.t to the smallest EV4 (Alpha 21064) core. Also observe that the asymmetric increase of $82\times$ approx. in chip area just to double the performance of a EV8 core w.r.t 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.

We perform the DSE to illustrate the cross-layer approach by considering chip/die area budget of four Alpha 21264 (EV6) processors as the system-level constraint. We find all the possible distinct combination with three class of processors (EV4, EV5, and EV6) that meets the area budget and number the 37 possible candidate configurations as shown in Fig. 3. To represent a diverse set of workload we select 8 Mediabench-II algorithms [18] as representative workloads and measure their execution time and power consumption as shown in Fig. 4 using a cycle accurate simulator (SimpleScalar and Wattch respectively). We measure the performance and power values for the EV4 processor and scale them by the scale factors as in Table II. To simulate the effect of varying workload and other micro-architectural effects, we randomly vary the execution time of the benchmark program, with a specified variance σ^2 across the measured mean value obtained earlier with the cycle accurate simulation. We consider each of these benchmarks as a single threaded task. To simulate the effect of diverse multithreaded workloads on the platform we gradually increase the number of tasks and perform the allocation across all the platforms. We use the combination of these 8 benchmarks to form new composite tasks (e.g. JPEG compression followed by AES encryption) and compute the execution time and power consumption as the sum of the individual benchmarks respectively. This allows us to test architecture configurations with more than 100 cores (for e.g. area budget of 4 EV8 results in 46428 configuration with as many as 330 EV4 cores). With the variability in number of tasks, the objective functions of each architectural combination with delay only minimization allocation strategy $minD$ is shown in Fig. 5.

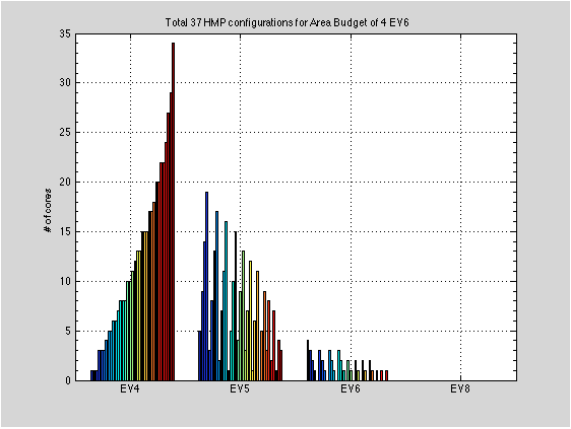


Figure 3. HMP configurations for area budget of 4×EV6. Total of 37 configurations numbered from 1 to 37 from left to right.

We perform an initial set of simulations generated by DoE in order to build the response surface model of the system for four design objectives i.e., make-span/delay, power, energy, energy delay product (EDP), and energy delay square product (EDSP) as listed in Table. I. We use these initial simulation points to construct the predictive models by using linear regression and obtain the coefficients of the expression (5) by performing least square fitting of the data. Our cross-layer DSE shows that an allocation strategy that performs excellently with one architecture configuration does

Table II
ALPHA PROCESSOR CORES PERFORMANCE, AREA AND POWER [16], [14]

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

* Normalized versus EV4; All cores scaled to 0.1 μm, at 2.1 GHz; IPC based on SPEC CPU benchmarks;

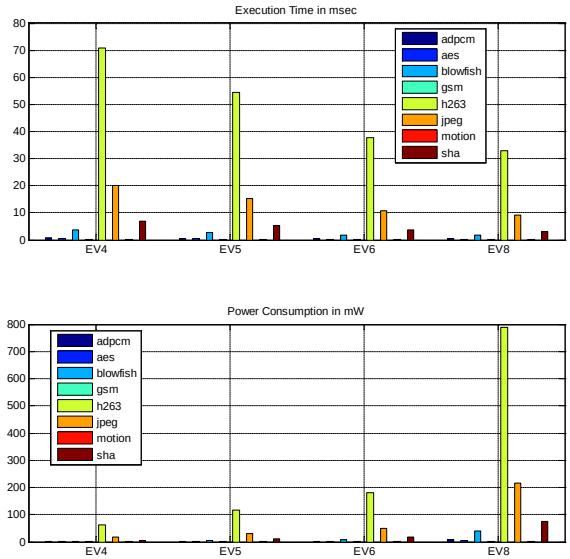


Figure 4. Power and execution time for 8 benchmarks for different Alpha processors.

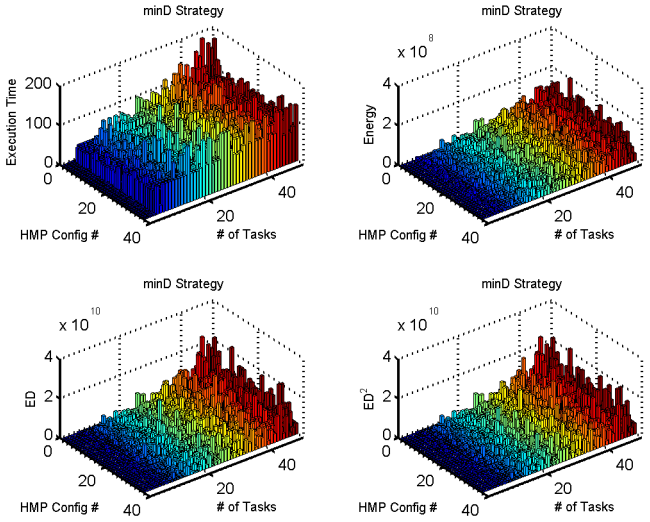


Figure 5. Objectives with variability in number of task for delay only task allocation strategy ($minD$). Lower is better.

not perform equally well for another architecture configuration and there is a rich design space to exploit for a specific solution. The predictive models demonstrating the relative merit for heterogeneous multicore processor configurations for the same area budget and different allocation strategies are shown in Fig. 6. Furthermore, we compare the allocation strategies with a heterogeneity oblivious random allocation with variability in number of task and the execution time as show in Fig. 7. The joint impact of considering the workload variability (with variations in number of tasks and intra task execution time variation) with allocation strategies shows that our SA based strategies performs better in terms of energy delay product (EDP) and energy delay square product (ED^2) by as much as 50 % and 70% respectively across all the architectures in comparison to the heterogeneity agnostic allocation policy. Considering these impacts and variability in the workload, we search for the best performing architectures and tabulate in Table III as the most preferable architectures (most frequently occurring) for a given allocation strategy for different objectives with the given equal area budget constraints. It is observed that for the given area budget the architectural combination #9(8×EV4,5×EV5,2×EV6) with 8 EV4 cores, 5 EV5 and 2 EV6 cores has superior performance in terms of EDP and ED^2. Our detailed technical report contains comprehensive DSE results and other discussions of explorations [2].

VI. CONCLUSIONS

Recent research has highlighted the potential benefits of single-ISA heterogeneous multicore 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, to fully tap into the potential of these multicore processor, a design flow exploration is necessary. In this paper, we propose and perform cross-layer (across application, operating system, and hardware architecture layer) design space exploration (DSE) of single-ISA heterogeneous multicore processors using predictive models to investigate the interactions and influence of heterogeneity of hardware architectures (configurations, number and types of cores), multi-objective allocation strategies along with diverse types of workloads under system level constraints (such as equal area or power budget). The proposed cross-layer DSE approach reveals relative merits of one architectural configuration and allocation strategy over the other and helps in selecting most promising heterogenous architectures. We illustrate the usefulness and applicability of our DSE approach as a crucial tool especially during the early design and verification stages of the HMPs when the design space is at its largest by using an example case study. Our future work will investigate the joint impact of DVFS and allocation strategies for distinct heterogeneous architectural configurations for drastic performance, power, and energy efficiency using the proposed cross-layer DSE.

REFERENCES

[1] *The Design and Analysis of Computer Experiments*. Springer-Verlag, 2003.
[2] Anonymized. Detailed technical report. Technical report, 2012.
[3] Saisanthosh Balakrishnan et al. The impact of performance asymmetry in emerging multicore architectures. *SIGARCH Comput. Archit. News*, 33(2):506–517, May 2005.
[4] Michela Becchi et al. Dynamic thread assignment on heterogeneous multiprocessor architectures. In *Proceedings of the 3rd conference on Computing frontiers*, CF '06, pages 29–40, New York, NY, USA, 2006. ACM.

Table III
PREFERRED ARCHITECTURAL CONFIGURATIONS WITH DIFFERENT STRATEGIES

Strategy/Objective	J_D	J_E	J_{ED}	J_{ED^2}
$minD$	#1 (4×EV6)	#1(4×EV6)	#1(4×EV6)	#1(4×EV6)
$minE$	#37(34×EV4)	#9(8×EV4,5×EV5,2×EV6)	#37(34×EV4)	#37(34×EV4)
$minED$	#9(8×EV4,5×EV5,2×EV6)	#37(34×EV4)	#2(5×EV5,3×EV6)	#2(5×EV5,3×EV6)
$minED^2$	#9(8×EV4,5×EV5,2×EV6)	#37(34×EV4)	#9(8×EV4,5×EV5,2×EV6)	#9(8×EV4,5×EV5,2×EV6)

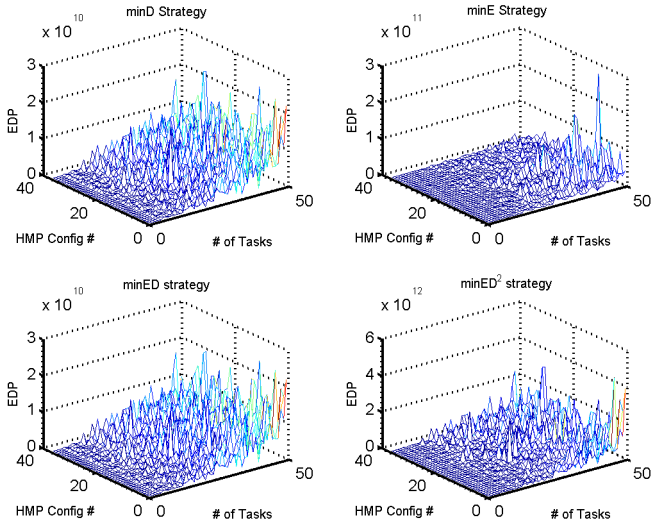


Figure 6. Energy Delay Product (EDP) with different allocation strategies and task variability using the predictive models. Lower is better.

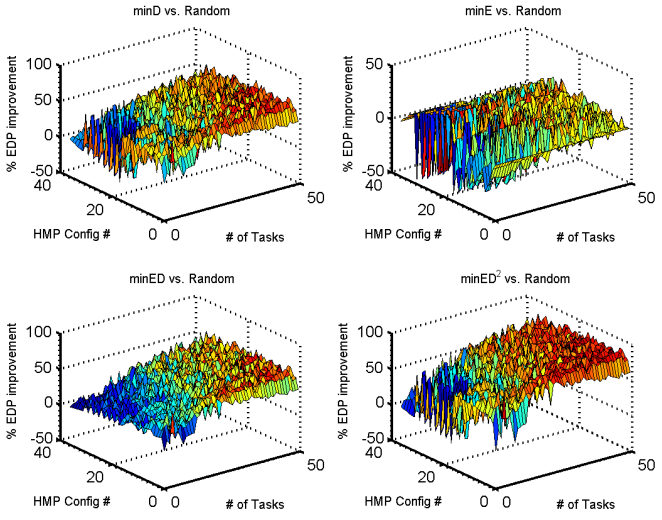


Figure 7. DSE with predictive models: Comparison of SA based allocation strategies with random allocation strategy. Higher is better.

[5] Jian Chen et al. Efficient program scheduling for heterogeneous multi-core processors. In *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, pages 927 –930, july 2009.

[6] Rong Ge et al. Powerpack: Energy profiling and analysis of high-performance systems and applications. *IEEE Trans. Parallel Distrib. Syst.*, 21(5):658–671, May 2010.

[7] Soraya Ghiasi et al. Scheduling for heterogeneous processors in server systems. In *Proceedings of the 2nd conference on Computing frontiers*, CF '05, pages 199–210, New York, NY, USA, 2005. ACM.

[8] P. Greenhalgh. Big.little processing with arm cortex-a15 & cortex-a7: Improving energy efciency in high-performance mobile platforms. 2011.

[9] M.D. Hill et al. Amdahl's law in the multicore era. *Computer*, 41(7):33 –38, july 2008.

[10] Van Craeynest Kenzo et al. Scheduling heterogeneous multi-cores through performance impact estimation (pie). In *nternational Symposium on Computer Architecture*, ISCA'12, 2012.

[11] K. Keutzer et al. System-level design: orthogonalization of concerns and platform-based design. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 19(12):1523 –1543, dec 2000.

[12] Jungsoo Kim et al. Program phase and runtime distribution-aware online dvfs for combined vdd/vbb scaling. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 417 –422, april 2009.

[13] David Koufaty et al. Bias scheduling in heterogeneous multi-core architectures. In *Proceedings of the 5th European conference on Computer systems*, EuroSys '10, pages 125–138, New York, NY, USA, 2010. ACM.

[14] R. Kumar et al. Single-isa heterogeneous multi-core architectures for multithreaded workload performance. In *Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on*, pages 64 – 75, june 2004.

[15] Rakesh Kumar et al. Single-isa heterogeneous multi-core architectures: The potential for processor power reduction. In *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 36, pages 81–, Washington, DC, USA, 2003. IEEE Computer Society.

[16] Rakesh Kumar et al. Heterogeneous chip multiprocessors. *Computer*, 38(11):32–38, November 2005.

[17] Nagesh B. Lakshminarayana et al. Age based scheduling for asymmetric multiprocessors. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 25:1–25:12, New York, NY, USA, 2009. ACM.

[18] Chunho Lee et al. Mediabench: a tool for evaluating and synthesizing multimedia and communications systems. In *Microarchitecture, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on*, pages 330 –335, dec 1997.

[19] Tong Li et al. Efficient operating system scheduling for performance-asymmetric multi-core architectures. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, SC '07, pages 53:1–53:11, New York, NY, USA, 2007. ACM.

[20] J.C. Mogul et al. Using asymmetric single-isa cmps to save energy on operating systems. *Micro, IEEE*, 28(3):26 –41, may-june 2008.

[21] NVidia. Variable smp - a multi-core cpu architecture for low power and high performance. 2011.

[22] G. Palermo et al. Respir: A response surface-based pareto iterative refinement for application-specific design space exploration. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(12):1816 –1829, dec. 2009.

[23] George Patsilaras et al. Efficiently exploiting memory level parallelism on asymmetric coupled cores in the dark silicon era. *ACM Trans. Archit. Code Optim.*, 8(4):28:1–28:21, January 2012.

[24] A.D. Pimentel et al. A systematic approach to exploring embedded system architectures at multiple abstraction levels. *Computers, IEEE Transactions on*, 55(2):99 – 112, feb. 2006.

[25] Daniel Shelepov et al. Hass: a scheduler for heterogeneous multicore systems. *SIGOPS Oper. Syst. Rev.*, 43(2):66–75, April 2009.

[26] T. Sherwood et al. Discovering and exploiting program phases. *Micro, IEEE*, 23(6):84 – 93, nov.-dec. 2003.

[27] Sadagopan Srinivasan et al. Efficient interaction between os and architecture in heterogeneous platforms. *SIGOPS Oper. Syst. Rev.*, 45(1):62–72, February 2011.

[28] Radu Teodorescu et al. Variation-aware application scheduling and power management for chip multiprocessors. *SIGARCH Comput. Archit. News*, 36(3):363–374, June 2008.

[29] Jonathan A. Winter et al. Scalable thread scheduling and global power management for heterogeneous many-core architectures. In *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, PACT '10, pages 29–40, New York, NY, USA, 2010. ACM.