Design Space Exploration for Chiplet-Assembly-Based Processors

Saptadeep Pal, *Student Member, IEEE*, Daniel Petrisko, *Student Member, IEEE*, Rakesh Kumar, *Senior Member, IEEE*, and Puneet Gupta, *Senior Member, IEEE*

***Abstract*— Recent advancements in 2.5-D integration technologies have made chiplet assembly a viable system design approach. Chiplet assembly is emerging as a new paradigm for heterogeneous design at lower cost, design effort, and turnaround time and enables low-cost customization of hardware. However, the success of this approach depends on identifying a minimum chiplet set which delivers these benefits. We develop the first microarchitectural design space exploration framework for chiplet assembly-based processors which enables us to identify the minimum set of chiplets to design and manufacture. Since chiplet assembly makes heterogeneous technology and cost-effective application-dependent customization possible, we show the benefits of using multiple systems built from multiple chiplets to service diverse workloads (up to 35% improvement in energy-delay product over a single best system) and advantages of chiplet assembly approaches over system-on-chip (SoC) methodology in terms of total cost (up to 72% improvement in cost) while satisfying the energy and performance constraints of individual applications.**

***Index Terms*— 2.5-D integration, chiplet assembly, microarchitectural design space exploration (DSE), multichiplet optimization.**

I. Introduction

MICROARCHITECTURAL design space exploration (DSE) for a general-purpose processor is traditionally aimed at determining the microarchitectural parameter values of *one* processor system that has the highest performance or efficiency for a set of representative applications. The goal is to arrive at the *one* system that has the highest performance or efficiency for these applications. However, applications are often targeted by using a family of processors where each processor in the family targets a subset of the applications. Consider Intel Xeon Product Family [1], for example. The product family targets nonconsumer workstations [2], servers [3], and embedded applications [4] using different processors in the family. Even within the Xeon Processor E7 family [2], there are a large number of processors each targeting a different subset of workstation and server applications. Using multiple systems to target applications provides an effective way to address heterogeneity in workloads, objective functions (power versus performance), compute conditions (battery-powered versus wall-powered), and cost.

Unfortunately, a countervailing trend—increasing processor design, verification, manufacturing, and management costs [5], [6]—is putting immense pressure on the number of systems that can be used to target applications. As these costs rise, designing and manufacturing a large number of systems-on-chip (SoCs) may become infeasible.

As a result, new design and assembly methods [7]–[11] are being developed and commercialized where different chiplets can be connected using SoC-like low-latency and high-bandwidth interconnect substrates. Thus, a large processor SoC can now be disintegrated into multiple, smaller component chiplets and then reintegrated into a full processor system. In fact, critical processor components can be partitioned into separate chiplets and integrated on these high-performance interconnects without significant performance degradation (<5%) compared to SoCs [11]. Since smaller chiplets often achieve better yield, this approach may decrease overall system cost. Moreover, one can create many systems using different combinations from a set of chiplets, allowing a designer to tailor each system toward a particular subset of applications. Therefore, reuse of chiplets across several systems can help amortize the cost of chiplet design and improve turnaround time. Furthermore, chiplets implemented using different technology nodes can be integrated into the same system—this may allow further reduction in design and manufacturing costs. Overall, chiplet assembly is emerging as a new paradigm for heterogeneous design at lower cost, design effort, and turnaround time and enables low-cost customization of hardware. However, the success of this approach depends on identifying a minimum chiplet set which delivers these benefits. Algorithmically exploring the design space and then investigating cost, performance, and energy tradeoffs of such chiplet-based customization are the goals of this article.

In this article, we ask three questions.

1) How should one set up the microarchitectural DSE problem when a set of component chiplets will be used to build a set of systems to target different applications?

Manuscript received June 6, 2019; revised November 22, 2019; accepted December 29, 2019. This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) through ONR under Grant N00014-16-1-263, in part by UCOP under Grant MRP-17-454999, in part by the GuruKrupa Fellowship, in part by the UCLA CHIPS Consortium, and in part by the Center for Design Enabled Nanofabrication. *(Corresponding author: Saptadeep Pal.)*

Saptadeep Pal and Puneet Gupta are with the Department of Electrical and Computer Engineering, University of California at Los Angeles, Los Angeles, CA 90095 USA (e-mail: saptadeep@ucla.edu; puneetg@ucla.edu).

Daniel Petrisko was with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Champaign, IL 61801 USA. He is now with the University of Washington, Seattle, WA 98195 USA (e-mail: petrisko@cs.washington.edu).

Rakesh Kumar is with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Champaign, IL 61801 USA (e-mail: rakeshk@illinois.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2020.2968904

2) What are the microarchitectural characteristics of different chiplets and the corresponding systems when each system targets only a subset of applications instead of the entire application set?
3) When total cost of the design and manufacturing is concerned, what are the benefits of chiplet-based assembly method and what chiplets and corresponding systems need to be built?

We show that the DSE problem of identifying the best m chiplets to build k system configurations for a set of n applications is qualitatively and computationally very different from conventional DSE where $k = 1$. We present integer-linear programming (IntLP)-based formulations for solving this selection problem. This is the first work on systematically performing a microarchitectural DSE when multiple systems will be used to target applications. This is also the first study on a methodology to determine the number and characteristics of chiplets required to target a set of applications. This article makes the following contributions.

1) We develop the first optimization framework for solving the multiple chiplet/system selection problem.
2) We show how chiplet assembly can deliver near-custom system performance for *every* application with relatively small number of chiplets.
3) We consider minimization of total design/manufacturing cost by simultaneously solving chiplet and technology selection. We also discuss chiplet characteristics when cost-aware optimization is performed and show the conditions under which chiplet assembly can be a major cost-saver compared to SoCs.
4) We demonstrate the value of performing the chiplet DSE across different suites of applications (high performance to embedded applications) rather than performing the exploration per suite. Chiplets can be reused across different benchmark suites, thus maximizing opportunity of design cost amortization.

The rest of this article is organized as follows. Section II discusses representative work on processor DSE and IP-reuse-based design. Section III motivates the value of multiple systems as well the need for an effective multisystem optimization strategy. Section IV presents our optimization framework and cost model. Section V presents the experimental setup where we discuss our methodology, workloads, system/chiplets evaluated, and the cost components. Section VI discusses the results and the applicability of our framework in different scenarios such as multicore and heterogeneous systems. Finally, Section VII concludes this article.

II. Related Work

A. Processor DSE

A large body of prior work on DSE has focused on exploring the various parameters of a processor microarchitecture to maximize overall performance of a processor while minimizing its power and energy overhead. Papaworth [12] discusses optimizing the microarchitecture of the Intel Pentium Pro processors. Kin *et al.* [13] propose a fast but accurate processor DSE approach that estimates the performance bottlenecks in a single-core design and predicts the performance effects of tuning the bottlenecks. Frameworks such as *ArchExplorer* [14], *MULTICUBE* [15], *Magallan* [16], and *FADSE* [17] enable automatic DSE for multi-core architectures. Kumar *et al.* [18] and Choudhary *et al.* [19] try to find configurations of cores for a heterogeneous chip multiprocessor (CMP) where each core can be tuned for a class of applications sharing common attributes. However, in all these studies, the goal is to still find the one, best performing processor, instead of a selection of processors.

The closest related work is by Li *et al.* [20] and Kin *et al.* [13]. The authors explore a multidimensional design space for multicore architectures focusing on different related parameters of a processor: area, power, pipeline depth, superscalar width, and so on. They argue that it is challenging to accommodate different application classes (e.g., memory bound and compute bound) on one processor system and, therefore, one needs to find the optimized parameters for each class of applications. While these studies only focus on media processors, they motivate us to look deeper into the paradigm of multisystem processor DSE.

B. IP-Reuse-Based Design

Intellectual property (IP)-based design has been a dominant driving factor in the integrated system design where predesigned modules are reused as plug-and-play system components [21], [22]. SoC design methodology often uses available IPs to decrease the total design time. These IPs come in different forms: soft, firm, and hard. They vary in terms of configurability, soft IPs being fully configurable, while hard-IPs come as nonreconfigurable layouts. Automatic selection of IPs from a library of IPs to match requirements has received attention [23].

An emerging method of hard-IP reuse is chiplet assembly. With advancement in interconnect technologies, novel integration schemes are now possible. Interposers [24], EMIB [10], silicon-interconnect fabric [11], and wafer-level-fan-out [25] technologies enable high-bandwidth, low-latency communication between individual chiplets mounted on these interconnects. Recent effort toward building systems have been reported in the works by Schulte *et al.* [26] and Kannan [27]. Schulte *et al.* [27] proposed an exascale computing system using chiplet-based integration. They argue that such a large system is not possible on one single silicon chip due to yield issues. Smaller high-yielding known-good-chiplets can be integrated on highly efficient interconnect at reasonable costs. Also once a chiplet is designed, it can be reused in a multitude of systems. *MoCHI* [28] is another technology to achieve an integration scheme where SoCs can be split into multiple smaller cost-optimized modules and reintegrated without compromising system performance.

In both these cases, the question of which chiplets to design and which systems to construct from them to serve a set of workloads remains unaddressed. In this article, we provide a framework to choose the best m chiplets to manufacture and target a wide variety of applications and the optimal set of systems to build from them.

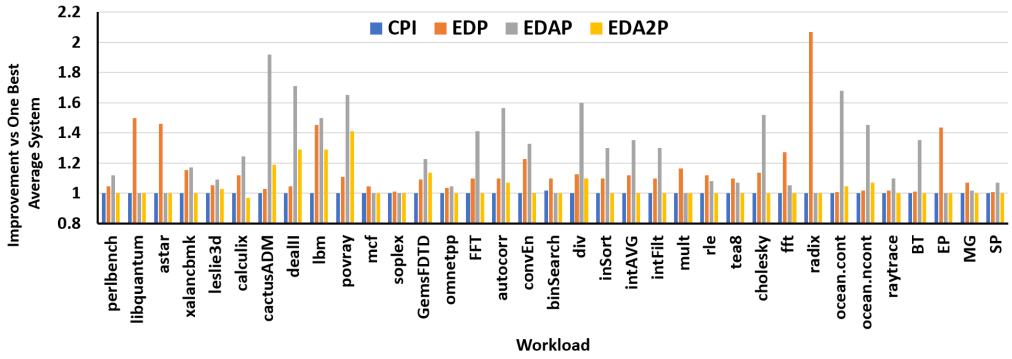

Fig. 1. Benefit of assigning each workload to its best processor versus a processor optimized across all workloads.

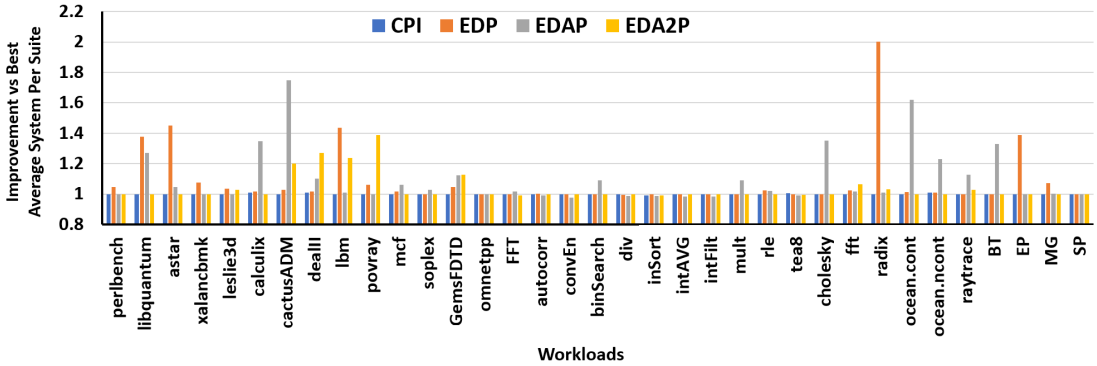

Fig. 2. Benefit of assigning four processors optimized across all workloads versus four processors each optimized for one of the four benchmark suites.

III. Motivation

Increasing the number of systems used to target performance and efficiency can have significant benefits. Fig. 1 shows the benefits across 35 workloads selected from four benchmark suites—SPEC2006 [29], EEMBC [30], SPLASH-2 [31], and NAS Parallel Benchmark (NPB) [32]—from assigning to each workload the best system available from a design space of 652 systems with private L1s, private L2s, and containing both in-order and out-of-order cores. The best system chosen for each workload depends, of course, on the metric. Benefits are normalized to a single system that performed the best across *all* workloads for that metric. As expected, without constraints on power or area, cycles-per-instruction (CPI) is minimized by selecting only one system, the largest issue width out-of-order core with the biggest caches. For other metrics, while a selection of one system per metric covers a large fraction of the workloads within 10% of optimal performance, there are significant improvements which can be achieved by picking a custom system per workload. For example, *radix* shows a 2.1× energy delay product (EDP) improvement compared to the best average system, *cactusADM* shows a 1.9× improvement in energy-delay-area product (EDAP)[1] and *povray* shows a 1.6× improvement in EDA^{2}P. As such, an effective exploration of the processor design space to allow selection of the (near) best system for a given workload could be very beneficial.

Since there is typically smaller variation within a single benchmark suite than across suites, one may be inclined to subset the workloads into their respective benchmark suites and select a system for each suite. Unfortunately, with this strategy, there are still certain outlier applications which are inadequately covered for some metric (see Fig. 2). This observation is consistent with prior works [19], which suggested that in the context of heterogeneous multicores, after selecting five cores to service average workloads, outlier workloads dominate core selection. When four systems are selected per metric using our optimization algorithm (see Section IV-A), all of the outlier workloads are covered within 10% of their optimal performance. Therefore, a systematic selection method is necessary to maximize the benefits of having multiple systems while minimizing the number of chiplets required.

Reducing the number of chiplets, processors, or processor component IPs can reduce the design, manufacturing, assembly, and management costs. Section IV-A describes how we formulate the different multisystem processor optimization problems under different constraints for the number of processors, processor component IPs, or chiplets.

IV. Optimal Selection of Chiplets

DSE of single-core [12], [13], multicore [14], [17], [34], and heterogeneous multicore [18], [19] processors has received a lot of attention in the past decades. However, in all these works, the goal is to find the one, best performing processor,

[1]The metrics of energy-delay-area^2 product (EDA^{2}P) and EDAP are examples of comprehensive metrics that consider both performance and cost, including both the operational cost (energy) and the capital cost (area). While a chip vendor may favor EDA^{2}P as area^2 provides an approximation to die cost in practice, a system vendor could prefer EDAP as other fixed system costs such as memory and I/O reduce the overall system cost dependence on CMP cost. The new metrics are shown to reveal new design sweet spots that cannot be found using other current metrics [33].

instead of a selection of processors. The potential gains from a multisystem approach, albeit in context of media processors, have been discussed previously [13], [20]. This article identifies the systems, constituent chiplets, and technologies to build them in order to deliver adequate per-application performance.

In this section, we describe a framework to select the m chiplets and the k systems to be built with these chiplets with the objective of either optimizing the EDP of the applications, or minimizing total cost of the systems. In our experiments, we considered a system is composed of core + L1 and L2 chiplets; given a performance estimator, other chiplets with different functionality (L3, accelerator, and so on) can also be considered in the framework. Here, we assumed that all chiplets would use standardized interface (physical layer, PHY) protocols to communicate with each other (e.g., Intel's AIB protocol [35]). These protocols can usually be extended to support any width of the interface and adds only one additional cycle of latency to the interface. The chiplets can still use the same higher level protocols (e.g., packetization schemes) as in an SoC implementation.

When $k = 1$ (systems) or $m = 2$ (chiplets), the problem becomes the conventional DSE problem. The nature of the problem changes for $m > 2$ or $k > 1$. For n possible systems, the search space increases from $O(n)$ to $O(2^n)$ going from single system to multisystem DSE. First, we describe an IntLP framework to optimally solve the multichiplet selection problem. When the number of chiplets/systems/workloads are large ($\gg$10 000), the IntLP formulation can become computationally challenging where well-known heuristic methods can be used (e.g., LP relaxations). We focus on optimal solutions as, for our problem sizes, IntLP can be solved by commercial solvers [36] in few minutes. Furthermore, formulating the problem as an IntLP gives us immense flexibility in setting up a variety of constraints and objectives within the same optimization framework as would be illustrated by the experiments in the subsequent sections. The notation used in this section are described in Table I. First, in Section IV-A, we describe an IntLP framework to optimally solve the multisystem/chiplet selection problem. Our chiplet assembly cost model used for cost-aware optimization is described next in Section IV-B.

TABLE I
NOTATION

Notation	Meaning
S	Initial Set of all systems under consideration
A	Set of applications
D	Set of all chiplets under consideration
V_a	Estimated relative volume of systems running application a
x_s^a	0 or 1 indicating whether s is used to service application a
x_s	0 or 1 indicating whether system s is present in the final system set
W_s^a	n-tuple of metrics (EDP, CPI etc) when application a runs on system s
N_p	p^{th} element of W_s^a, $p \in [1, n]$
c_s	Area of system s
α_a	Area constraint for application a
χ_{ap}	Threshold value of parameter p for application a
D_d^s	0 or 1 indicating whether chiplet d is a component of system s
y_d^t	0 or 1 indicating where chiplet d manufactured in technology node t is present in the final set of chiplets
T	Set of technology nodes in exploration
K	Maximum number of unique systems possible in the final set of systems
L	Maximum number of unique chiplets allowed in the final set of chiplets

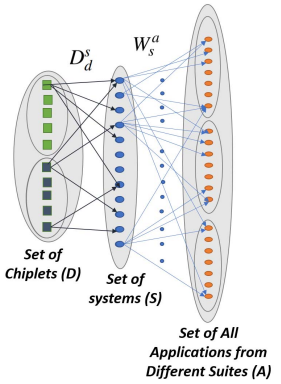

Fig. 3. Illustration of the multichiplet selection problem.

A. IntLP DSE Framework

One can visualize the problem as shown in Fig. 3. The microarchitecture design space is used to constitute the set of chiplets and systems. For example, in our DSE with single-core systems, a system is considered to be composed of two chiplets, core + L1 and L2. The initial set of microarchitectural parameters, such as size and associativity for L1 and L2 caches and size, type, execution units, and so on for the core, could be used to determine the set of chiplets and systems in the optimization problem. Here, the D_d^s edges indicate which chiplets are part of a system. Multiple copies of the same chiplet microarchitecture may be included in the initial set of the chiplets (D), representing different technology nodes. The set of system configurations (S) is the set of all systems that can be composed out of the chiplets. Each application can be assigned to run on any system and the cost [n-tuple containing the power, performance in CPI, energy-per-instruction (EPI), EDP, and total cost] associated with that is given by W_s^a. In the absence of chiplet constraints, the multisystem selection problem is related to minimum maximal bipartite matching.

For a given set of chiplets and applications, we formulate the IntLP optimization as follows:

$$\textbf{Minimize: } \sum_{a\in A}\sum_{s\in S} x_s^a * W_s^a.N_j$$

$$\textbf{Subject to: } \sum_{s\in S} x_s^a = 1, \quad a \in A \tag{1a}$$

$$\sum_{s\in S} x_s^a * W_s^a.N_p \le \chi_{ap}, \quad p \in [1,n];\ a \in A \tag{1b}$$

$$\sum_{s\in S} x_s^a * c_s \le \alpha_a, \quad a \in A \tag{1c}$$

$$x_s^a \le x_s, \quad s \in S;\ a \in A \tag{1d}$$

$$\sum_{a\in A} x_s^a \ge x_s, \quad s \in S \tag{1e}$$

$$\sum_{s\in S} x_s \le K \tag{1f}$$

$$y_d^t \ge D_d^s * x_s \quad s \in S;\ d \in D;\ t \in T \tag{1g}$$

$$\sum_{s\in S} D_d^s * x_s \ge y_d^t \quad d \in D;\ t \in T \tag{1h}$$

$$\sum_{d\in D, t\in T} y_d^t \le L. \tag{1i}$$

The objective is to minimize the sum of a normalized metric, such as EDP, or total cost, over all application-system assignments. Each metric is normalized for each application with respect to the best possible custom system for that application. Note that other objective functions such as maximizing performance, or different objectives for different applications can also be used.

Constraint (1a) assigns exactly one system to every workload. Constraint (1b) is used to ensure that for metric p, the system k chosen for an application a is better than χ_{ap} threshold. We primarily use it to enforce a minimum normalized CPI or EDP constraint in our optimizations. Constraint (1c) enforces application-specific area (or power) constraints (e.g., for low-cost or thermally constrained systems). Constraint (1d) ensures that a system selected to service an application is present in the final system set. Constraint (1e) guarantees that every system in the final set services at least one application. Constraint (1f) is an optional constraint which upper bounds the total number of systems selected to k. Constraints (1g) and (1h) ensure that a chiplet is chosen if and only if it is part of a chosen system. Constraint (1i) is optional and puts a cap on the maximum number of chiplets allowed.

B. Chiplet Assembly Cost Model

Cost brings in an interesting dimension to the multichiplet selection problem. Migrating to an advanced technology node provides benefits in terms of area and power but increases design and manufacturing costs. Cost of a chiplet can be broken down into two major components: nonrecurring engineering (NRE) cost and volume-dependent recurring cost. NRE costs include architecture, RTL design, IP validation, physical design, prototype, validation, and mask manufacturing cost. In our cost model, we consider the cost difference between the logic and SRAM as well. Due to its regular structure, memory design is typically less expensive than the logic of the same size. Moreover, SRAMs typically use fewer metal layers resulting in lower manufacturing costs as well.

We assume the recurring cost to be the cost of wafer fabrication. Yield and process complexity are the primary factors which determine the fabrication cost. Advanced technology nodes require larger number of process steps on more expensive equipment, increasing manufacturing costs.

We use the yield learning model given in [37] to model the yield improvement over time so as to account for process maturity

$$Y(t) = Y_{\text{asymp}} \times (1 - \exp(-ct)) \tag{2}$$

where $Y(t)$ is the yield at time t after launch of the technology node. We considered 50% yield at the time of process node introduction and that the yield reaches 95% when $t = 2$ years. Y_{asymp} is the asymptotic yield determined by the die area (A_{die}), defect density (D_0), and clustering factor (α) and is given by the well-known negative binomial model

$$Y_{\text{asymp}} = \left(1 + \frac{A_{\text{die}} D_0}{\alpha}\right)^{-\alpha}. \tag{3}$$

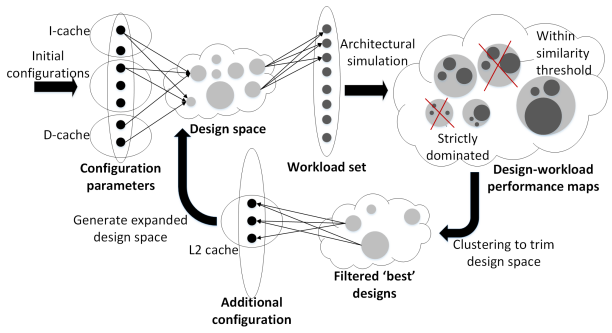

Fig. 4. Experimental methodology for design space generation.

Recurring cost of manufacturing (C^{mfg}) taking into consideration the volume of dies required (V_d), wafer cost (C^w), yield (Y), and the number of dies per wafer (N_{die}) is given by

$$C^{\text{mfg}} = \frac{V_d \times C^w}{N_{\text{die}} \times \text{Yield}}. \tag{4}$$

We use the following objective function when the total cost minimization is considered for chiplet-based assemblies:

$$\sum_{a \in A}\left(V_a \times \left(\sum_{s \in S} x_s^a \left(C^{\text{int}} + \sum_{d \in D} D_d^s * C_d^{\text{mfg}}\right)\right)\right) + \left(\sum_{d \in D}\sum_{t \in T} y_d^t * \text{NRE}_d^t\right) + \sum_{d \in D}\left(\vee_{t \in T}\, y_d^t * \text{NRE}_d^{\text{nt}}\right) \tag{5}$$

where V_a is the volume of systems required for application a. C^{int} is the chiplet assembly/integration cost, C_d^{mfg} is the cost of manufacturing of chiplet d and is estimated using (4). NRE_d^t is the technology-dependent NRE cost which includes physical design, IP validation, prototype, mask set cost, and so on. NRE_d^{nt} is the technology agnostic NRE cost which includes architecture, RTL development and verification, and so on. When a particular microarchitecture is used to build chiplets in two different technology nodes, NRE_d^{nt} is amortized; however, NRE_d^t expense is required for every technology node the chiplet is built.

The result of the IntLP solution is an optimal set of chiplets, and the systems constituted out of the chiplets and the application-system mapping.

V. Experimental Setup

A. Methodology

The methodology for our DSE is an iterative process consisting of three main steps (see Fig. 4). First, we identified a set of interesting initial system configurations to explore. This set of configurations (provided in Table II) must be diverse enough so that changing any parameter has a measurable impact on either power, area, or performance of the system running an application. In addition, these configurations should vary enough such that their range tightly covers the range of application behavior.

Next, we performed a full factorial exploration of these initial system configurations. To evaluate performance of designs, we used Sniper [38], a fast trace-driven multicore

TABLE II
SYSTEM PARAMETERS AND THEIR VALUES

Issue-width	1, 2, 4	**ROB (OOO)**	32, 64 entries
I-Cache	8KB DM, 16 KB 2-way, 32KB 4-way, 64KB 4-way	**L2 Cache**	256KB, 1-way, 512KB, 2-way, 1MB, 4-way, 2MB, 8-way, all 12 cycle access
D-Cache	8KB DM, 16KB 2-way, 32KB 4-way, 64KB 4-way dual ported	**Memory Channel**	533 MHz, doubly-pumped, RDRAM
Execution Units	3, 6	**ITLB-DTLB**	64-28 entries
IQ (OOO)	48, 96	**Ld/St Queue**	32 entries

interval simulator. Sniper is significantly faster than gem5: this is important because the sheer number of simulations required for a full factorial exploration is large. We used the ROB-centric model in Sniper, which more accurately models in order cores and issue contention. We studied trade-offs related to multicore systems too. For the performance and energy calculation of multicore systems considered in Section VI-B and heterogeneous multicores in Section VI-H, we made a simplifying assumption that two single-threaded workloads running on separate cores of a dual-core system do not interfere and that their aggregate performance is the summation of the IPCs when each workload is run individually on a single core. Past works [18] have made such assumptions. However, we also performed an experiment (in Section VI-I) with homogeneous multicore system running multithreaded benchmarks with a shared-L2 cache.

L2 cache access usually takes $\sim$10 cycles. For the performance simulations, we assumed that the L2 caches have a 12-cycle access. Changing the latency by one cycle (due to interface protocol circuitry) resulted in a negligible performance difference (worst case $<$3%) as most of the memory accesses from the cores are served by the L1 cache which lies in the same chiplet as of the core. Therefore, we used the same performance numbers for SoC and chiplet in order to avoid duplicate simulation runs.

For area and power modeling, we modeled 22-, 32-, and 45-nm processes using McPat 1.0 [33], which is deeply integrated with Sniper. To calculate average power and energy for our primary results, we excluded DRAM energy. We performed a sensitivity analysis of our results when including DRAM and found that the trends and analysis were largely unchanged.

The design space of this initial set of systems is pruned using a canopy clustering technique similar to [39] such that systems which are worse (or within 5%) with respect to the rest of the systems for *every* metric of interest on *all* workloads are removed from consideration. We then added new configurations to our reduced design space. This clustering usually cuts down the design space by 3×–10× with negligible impact on eventual optimized metrics. This is important as DSE runtime is dominated by microarchitectural simulation.

Last, we used the optimization algorithm described in Section IV-A to select the best chiplets and systems to cover the given workloads.

Runtime: The total runtime of an IntLP instance lies between 1 and 4 min depending on the contraints, while a Sniper simulation run for a particular system–workload combination takes about 20–30 min.

TABLE III
WORKLOADS EVALUATED

Suite	Benchmarks
EEMBC	FFT, autocorr, convEn, binSearch, div, inSort, intAVG, intFilt, mult, rle, tea8
SPEC2006	perlbench, omnetpp, mcf, libquantum, astar, xalancbmk, leslie3d, calculix, GemsFDTD, cactusADM, dealII, soplex, lbm, povray
SPLASH-2	cholesky, fft, radix, raytrace, ocean.cont, ocean.ncont
NPB	BT, EP, MG, SP

B. Workloads Evaluated

A diverse set of workloads were chosen by subsetting four benchmark suites—SPEC2006 [29], EEMBC [30], SPLASH-2 [31], and NPB [32]. We chose 14 applications from SPEC2006 based on [40] in order to demonstrate the diversity of the suite without capturing redundant results between benchmarks. SPEC2006 applications were simulated using 100M-warmup, 30M-detail Pinballs [41] with $\max K = 10$ (up to ten SimPoint regions are simulated in total). Four benchmarks selected from NPB were simulated using 100M-warmup, 30M-detail Pinballs with $\max K = 10$ and input size W. Twelve benchmarks from the telecomm and automotive suites of EEMBC were simulated by running them in a loop in detailed mode for 30M instructions. SPLASH-2 benchmarks were run single-threaded with fast-forwarding until the region of interest (ROI) was simulated in full. A complete list of chosen benchmarks is shown in Table III.

C. Systems and Chiplets Evaluated

We relied on previous work [18] to make our initial design space selection with a few differences in order to more accurately model the systems in Sniper which is based on a Nehalem-like architecture [42]. For instance, Sniper does not model the number of functional units, but instead multipurpose instruction ports, which can be used to service a subset of instructions. We provide configurations for both a traditional Nehalem architecture and one with double the number of instruction ports. Additionally, due to a lack of accurate functional modeling in Sniper, we did not consider different physical register file sizes.

For this article, we considered that a single-core processor system is comprised of two chiplets: core with L1 cache (core + L1 chiplet) and L2 chiplet. We considered the interconnect characteristics to match those of the state-of-the-art chiplet assembly approach presented in [11]. With latency and energy-per-bit overheads comparable to those of traditional SoCs, core + L1 and L2 chiplet systems require no substantial microarchitectural changes compared to their single-die counterparts.

TABLE IV
NORMALIZED COST COMPONENTS AT DIFFERENT TECHNOLOGY NODES

Cost Type		Technology Node		
		22 nm	32 nm	45 nm
NRE Design Cost [44]	Logic (per mm^2)	849	502	332
	SRAM	965	695	483
NRE Mask Set Cost [45]	Logic	695	541	309
	SRAM	486	378	216
Recurring Wafer Cost [46]	Logic	2.43	2.01	1.41
	SRAM	1.70	1.41	1
Chiplet Assembly Cost [47]		0.25		

We first performed simulations on all core (includes L1 caches) configurations, with a constant 1-MB L2 cache. We fully explored these configurations and then trimmed the design space to a smaller set using the clustering approach described earlier. We then simulated this reduced set of cores with a wide range of L2 cache sizes to construct a broader design space for our diverse set of workloads. Finally, we used these results to generate power, energy, and area results for chiplets implemented in 22-, 32-, and 45-nm technology nodes. Because we assume the same microarchitectural parameters for a chiplet across technology nodes, performance characteristics of each chiplet remain the same. We also considered systems built out of heterogeneous technology chiplets for our evaluations. In all, we considered a total of 5868 system configurations in our homogeneous-core (single- and multicore) studies and 10 404 system configurations for the heterogeneous multicore case.

D. Cost Components and Volume

The cost model used in our formulation has been presented in Section IV-B. The various cost components used in the cost model for the evaluations are shown in Table IV [43]–[45]. Note that for SRAM, the design cost was considered to be the cost of a memory compiler license and a few engineers. SRAM chiplets would also usually use only five to six BEOL layers as compared to 10–12 metal layers for logic (both use all the FEOL layers). Therefore, we scaled the mask and recurring wafer cost accordingly.

We considered the chiplets to be integrated on 2.5-D substrates such as silicon interposers [7], silicon interconnect fabric [11], or EMIB [10]. On these substrates, the chiplets can have an inter-chiplet spacing of <1 mm (even <100 μm). Also because of fine pitch interconnections, the interfaces are usually wide and signaling is done at frequencies of 2–4 GHz, and therefore the I/Os driving these substrate wires can be simple multistage buffers (with small ESD circuitry) [10], [11] which are area and energy efficient. Therefore, considering $\sim$2000 IOs between a core and a cache chiplet, we added an overhead of 0.5 mm^2 (conservative estimate) to the chiplet areas.

The cost of integration and bonding, i.e., C^{int}, was assumed to be similar to the data in [46]. We also performed sensitivity analysis with $2\times$ and $4\times$ the assembly cost, because in reality, the cost of 2.5-D have been very high and is in fact used for a niche class of high-performance processors only. We used the ITRS data [47] for D_0 and α while calculating the yield. To estimate the per chip vendor volume per year, we used the global estimates of phone and tablet sales (180 million units per vendor) [48] for the EEMBC suite, aggregate PC and laptop sales (90 million units per vendor) [49] for SPLASH and SPEC suites combined and global $\times$86 server sales (10 million units per vendor) for the NPB suite [50]. We divided the total volume in each suite by the number of workloads in each suite to uniformly distribute volume demand among individual workloads. The cost analysis is done for different sizes of the system. System size of $N\times$ (in Fig. 6) is the one where N copies of core + L1 chiplet and L2 chiplet constitute a system. For such multicore systems, the NRE design cost is considered for only one copy of the core. As an example, for a system with N ARM-based cores connected using AXI interconnect, the same core IP is replicated multiple times and connected to the scalable AXI interconnect IP. Therefore, the design cost incurred would be roughly similar to that of a single-core system with the interconnect. For performance and energy simulations, we consider that N instances of the same workload are executed on the N-core system. The cost optimization is also performed for different production start years to understand the impact of process maturity on cost. As processes mature, yield increases and this brings down the manufacturing cost per good chiplet.

Note that for some of the experiments where the goal is to capture both the energy and area cost, we used EDAP and EDA^{2}P as examples of composite metrics. This also shows that the framework is flexible and can be used in a variety of contexts.

VI. RESULTS

Here we discuss the characteristics of the chiplets, their usage and sharing across workloads and benchmark suites, cost implications, and so on. Note that we allow chiplets to be selected from different technology nodes in our optimizations. Also, the results shown here are for a small but interesting set of optimization objectives and constraints; however, the IntLP framework is flexible enough such that one can study many different objectives and constraint combinations.

A. Minimizing EDP With CPI Constraint

First, we consider EDP minimization as the objective with constraints on maximum CPI and number of unique chiplets. A CPI threshold t ensures that any selected system–workload pair have CPI less than t times the minimum CPI for that workload which can be obtained on the best performing system. As shown in Fig. 5, having more chiplets available allows significant reduction in average EDP. Initially, the benefit from adding more chiplets is high, as the first few systems selected out of the chiplets target broad workload classes such as memory-bound or compute-bound applications. As more chiplets are added, new systems primarily target outlier workloads, resulting in incremental average improvement. When the CPI threshold is very strict, the benefit quickly saturates since only a subset of systems are available for selection. However, for more relaxed CPI thresholds, seven to eight chiplets are required to attain near-optimal EDP. Since migrating to a newer technology node results in reduced EDP, all the chiplets selected in this cost-agnostic optimization were exclusively in 22-nm technology. Using multiple chiplets can

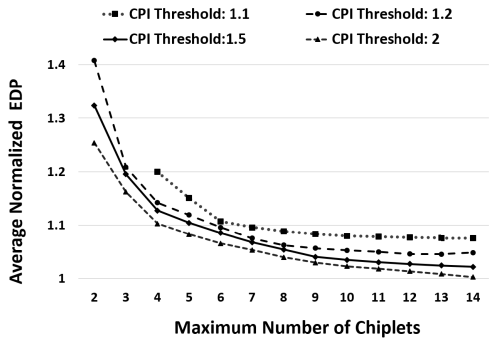

Fig. 5. Pareto curve showing average normalized EDP versus maximum number of chiplets allowed in the optimization. We show the Pareto curves for different CPI thresholds.

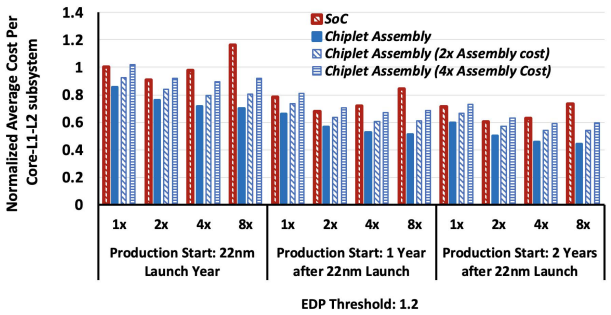

Fig. 6. Curve showing cost benefits of chiplet-based assembly over SoC for different sizes of the system and technology maturity level.

have greater than 35% reduction in EDP over a single average best system ($m = 2$).

Chiplet Microarchitecture: CPI threshold of 1.1 requires a minimum of four chiplets to be feasible. When only two unique chiplets are allowed (i.e., an average best system) for CPI thresholds of 1.2, 1.5, and 2.0, the same medium out-of-order core + L1 chiplet gets selected. However, the size of the selected L2 cache varies from 2 to 1 MB to 512 kB, respectively, for the three threshold values. When three chiplets are allowed, an additional cache gets added in all three cases (the smaller 512 kB for 1.2 and 1.5 CPI thresholds and the bigger 2 MB for the 2.0 case). This shows that cache variety has a major impact on EDP.

When four unique chiplets are allowed for CPI thresholds of 1.2 and 1.5, a smaller core chiplet with small L1 *D*-cache is added. Only when CPI is relaxed to 2.0, one in order core chiplet gets selected. With four chiplets, CPI threshold of 1.1 now becomes feasible and requires 1- and 2-MB L2 caches alongside two core + L1 chiplets, in which the core components are same but the L1 caches are large (64 kB) and small (16 kB), respectively. Initially when chiplets are added, having a variety of L2 cache sizes minimizes EDP. With more chiplet varieties, having large and small cores is more essential.

B. SoC Versus Chiplet-Assembly Cost Analysis

When systems are implemented via chiplet assembly rather than IP integration on an SoC, use of smaller chiplets results in higher yield and thus reduces cost. We perform DSE with total cost minimization objective with EDP threshold constraint. Chiplet-based assembly provides substantial cost benefit over an SoC approach as shown in Fig. 6. When the system size is small, SoC yield is good. This, along with the added cost of chiplet integration, results in a meager 10%–14% cost benefit over a single-core SoC system when the integration cost is considered to be the same as in [46]. However, in multicore systems as the system size increases, the SoC yield decreases according to (3). Because of the negative binomial nature of the yield curve, beyond a certain die size, the yield decreases rapidly. As a result, in Fig. 6, we observe that going from a single-core to a dual-core system, SoC integration results in per-core cost improvement, however increased yield issues result in increased cost as the system is scaled to a four-core and eight-core system. Using a chiplet assembly, copies of the same core + L1 and L2 chiplets are utilized and the yield of these smaller dies remains the same. Moreover, the NRE cost of developing these chiplets gets amortized as the system size grows. Though the interconnect substrate and chiplet assembly cost grows with the size of the system, it is a smaller fraction of the overall cost. Therefore, this results in reduced total cost per core + L1 and L2 chiplets using chiplet-based assembly despite a slight increase in system integration cost. We confirmed this trend by running our optimization across several EDP thresholds.

Apart from yield benefits, fewer chiplets are needed to assemble a large number of systems. The fact that relatively few unique chiplets can be mixed and matched to serve a wide array of applications bodes well for an era of customizable systems using chiplet assembly approach. In today's era of multicore processors, chiplet assembly can provide significant cost benefits, which will only improve as processors become larger and core count continues to increase.

Sensitivity to Assembly Cost: As mentioned earlier, there has not been a large improvement in 2.5-D integration cost and the overall cost remains quite high today. As seen in Fig. 6, when the integration cost is higher (4× assembly cost), the gap between the SoC and chiplet assembly becomes smaller. In fact, when the system size is small, chiplet assembly can be costlier than SoC. As the technology matures, the cost of SoC drops but integration cost does not drop much and therefore, larger dual-core (2×) systems have smaller (or diminished) gap in cost between SoC and chiplet assembly. However, when the system is even larger (4× or 8×), the worsened yield of larger SoC dies leads to rapid increase in SoC costs while even with 4× assembly costs, chiplet assembly-based systems come out ahead by as much 23%.

C. More Applications Share Chiplets Than Share Systems

Systems can share chiplets and, hence, m chiplets allow us to construct $\gg m$ systems. Systems serving different applications can share the same chiplets. We observe an example of chiplet sharing with the L2 caches. EEMBC workloads have smaller working sets, so they tend to use smaller L2 caches. However, a few workloads from SPEC and SPLASH-2 which also have smaller working sets can benefit from sharing these smaller L2 caches. We observe that when a large number of chiplets are available, most chiplets are shared across

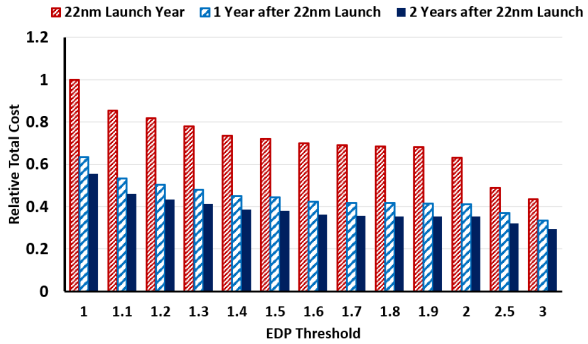

Fig. 7. Curve showing tradeoff for EDP and total cost at different technology maturity levels.

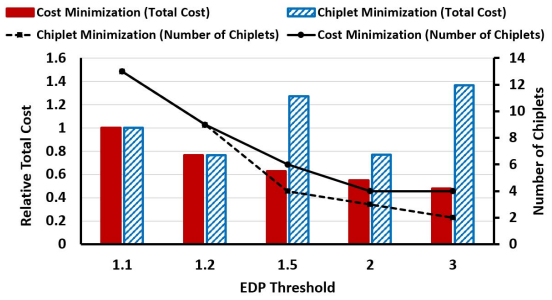

Fig. 8. Curve showing the difference between minimizing the number of chiplets versus total cost. The experiments were done for system size of 1× and production during 22-nm launch year.

benchmarks suites. The degree of reuse is much higher in this case than that when only system/SoC is considered because once the number of available systems increases, quite a few systems get selected that only cater to specific workloads from one workload suite.

D. EDP-Cost Tradeoff

Harsher EDP constraints (low threshold value) require more systems to minimize costs, essentially trimming the design space for each workload so that fewer systems are available for selection. This leads to excessive customization with little sharing of systems across workloads. Hence, the number of distinct chiplets required to build these systems increases, which results in higher NRE costs (shown in Fig. 7). As the EDP threshold constraint is relaxed, the optimization begins to select smaller cores and smaller L2 chiplets. Overall, fewer chiplets and systems are selected, increasing sharing across workloads. Increased sharing helps amortize NRE costs, resulting in reduced total costs. In fact, when the EDP threshold is low, the cost difference is much more dramatic (>34% for threshold of 1.1) between production start during 22-nm launch year and subsequent years than higher thresholds (>16% for threshold of 2.5).

For a particular total cost budget, as technology matures, one can achieve better overall EDP. This is because more chiplets and hence more systems tailored toward particular set of applications can be built for the same total cost when technology matures.

E. Impact of Technology Maturity

Technology selection of chiplets is a key factor which determines the overall cost of designing and fabricating multiple chiplets. Migrating to an advanced technology node reduces power and therefore EDP. Additionally, area per transistor decreases, leading to reduced chiplet size. However, lower wafer yield and higher NRE and manufacturing costs may outstrip the benefits of accommodating more chiplets per wafer in an advanced technology node. These costs are much higher during the early stages of a technology node. Fig. 6 shows how total cost varies with production start year. Our evaluations with 22-, 32-, and 45-nm technology show that although it is attractive to migrate to the 22-nm node because of the EDP and area benefits, a few chiplets are still chosen from 32-nm technology during the early stages of 22-nm node. However, it is interesting to note that more core + L1 chiplets are chosen from 32-nm technology than L2 chiplets. This is because the SRAM L2 chiplets are heavily shared across multiple workload suites and thus the high volume amortizes the NRE cost of the SRAM dies.

When the 22-nm technology matures (yield increases) and the fabrication cost decreases, building chiplets in 22 nm becomes attractive. Thus, all the chiplets are chosen from 22-nm technology when the production start is one or more years after 22-nm launch. In Fig. 6, depending on production start, up to 72% cost reduction can be achieved by chiplet assembly over SoC methodology while satisfying the EDP constraints per workload. Chiplet assembly's continuing cost benefits make it the ideal choice for building systems irrespective of production start year. Moreover, the cost benefits of chiplet assembly, especially through technology heterogeneity, are likely to be even higher as technology scales below 22 nm where manufacturing NRE costs are much higher (e.g., due to extensive use of multiple patterning lithography).

F. Minimizing Number of Chiplets Versus Cost

Here we show that optimization with the goal of minimizing the number of chiplets does not necessarily result in minimum total cost. As shown in Fig. 8, when EDP threshold is relaxed, chiplet minimization results in smaller number of chiplets than when cost is minimized. However, the total cost of design and manufacturing remains much higher than when cost minimization is the objective. This is because chiplet minimization chooses fewer but larger chiplets (lower yield and costly) to satisfy the EDP constraints of all the workloads while cost minimization chooses more number of smaller (less costlier) chiplets to build multiple systems, each tailored toward different types of workloads. For example, when EDP threshold is 2.0, cost minimization results in one in order, one small OOO, 256- and 512-kB L2 caches, whereas chiplet minimization returns two large OOO core and 512-kB L2 cache.

G. Efficacy of Proposed Optimization Framework

The IntLP framework solves the multisystem, multichiplet selection problem optimally. Here, we briefly compare our optimization to two other naive selection algorithms (as shown

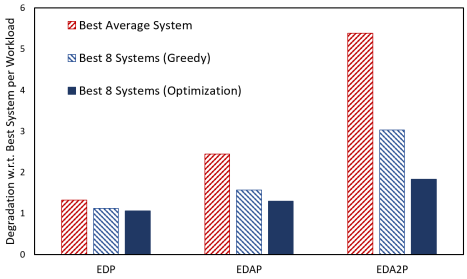

Fig. 9. Comparison between different system selection approaches for each metric when compared to the baseline case of best custom system per workload.

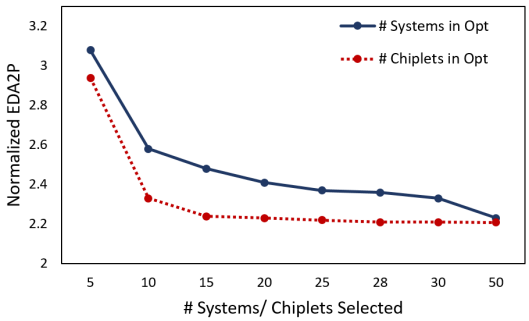

Fig. 10. Benefit in EDA^{2}P (normalized CPI threshold = 1.2) for heterogeneous CMP is shown for the cases where number of systems (SoCs) is taken into consideration versus when number of chiplets is considered in the optimization.

in Fig. 9). All selection algorithms are evaluated with a CPI constraint of 1.5, normalized to the best performance of each workload across all systems. "Best Average System" selects the system with the best average for a single metric which meets the CPI constraint across all workloads. "Greedy 8" selects first the best system which meets the CPI constraint across all workloads, then iteratively selects the next system which improves the metric, only using the new system to cover workloads for which it meets the CPI constraint. Our IntLP-based optimization framework can be up to 65% (EDA^{2}P) better than the naive heuristics.

H. Optimization for Homogeneous Multicores

Our methods of finding optimal set of chiplets can be applied directly to homogeneous multicores by considering, for example, shared L2 as a chiplet, multiple core + L1 chiplets in a system, and multithreaded benchmarks as workloads. For an evaluation with the SPLASH-2 suite, we find that five systems are still required to minimize metrics such as EDAP and EDA^{2}P for CPI-constrained systems. As observed in the previous experiments (without shared L2 cache), the set of final systems included a mix of system sizes, from small in order cores to medium-sized OoO cores to big OoO cores with a large cache. However, the progression toward a smooth gradient in system sizes that we observed in those system sets is not present in multicore system selection. Instead, we observed that an optimal system set contains a system with large OoO cores with a large shared cache and a set of systems with much smaller L2 shared cache size but a variety of smaller core sizes. This is because the larger system caters to the workloads with large working sets which otherwise would observe degraded performance if they are run on processors with smaller caches as interference between threads increases. We also observe that for such system, the positive effect of L1 cache sizes on system performance is diminished because of coherence overhead and false sharing.

Because of the disproportionate impact of L2 size on system performance, when selecting homogeneous shared-L2 multicore systems based on chiplet sharing, we see a very similar pattern of microarchitectures as when disregarding chiplets (i.e., SoC implementation). This suggests that the optimal system configurations for these homogeneous multicore systems are not as flexible as single-core systems. That is, the differences in relative performance across optimal systems for these workloads is large enough such that substituting a closely related chiplet for cost reasons will substantially degrade performance. While the sensitivity of workload performance to L2 size specifically may be an intrinsic property of SPLASH-2, we conclude that shared chiplets (L2 in this case) such as those present in a multicore system may have an especially large impact on the exact choice of chiplet microarchitectures.

I. Optimization of Heterogeneous CMP Systems

Modern processor systems are increasingly heterogeneous, incorporating a variety of core types, memories, and accelerators. Our system/chiplet selection framework is also easily adapted to heterogeneous settings by essentially enumerating candidate heterogeneous systems and application sets within the framework. Of course, we realize that computational scalability may become a concern here but we believe our clustering-based design-space pruning coupled with a smart initial design of experiments can provide a solution. Also to speedup the performance runs, if one has parameterizable RTL, FPGA emulation can be used which could potentially lead to 100× or more speedup compared to software simulation. As an example, we studied heterogeneous dual-core (i.e., big–little architecture) systems where each core runs a SPEC2006 benchmark application. Therefore, in this article, a total of 276 application pairs were considered. Fig. 10 shows the chiplet and system (SoC) exploration results. We make two observations. First, due to the dramatically increased diversity in workloads (pair of applications) to be run on a single dual-core system, the number of unique systems (composed out of chiplets) needed to achieve good performance is almost ten times higher compared to the single-core case. Second, the number of chiplets required to achieve the peak performance is significantly less than the number of distinct SoCs required to achieve the same metric. For example, more than 50 dual-core SoCs are required to achieve the best EDP with a CPI constraint of 1.2×. However, similar EDP requires less than 23 chiplets because a majority of chiplets are shared across the systems. This shows that chiplet-based assembly could be leveraged to generate a large number of these systems by manufacturing only a few types of chiplets. Thus, finding

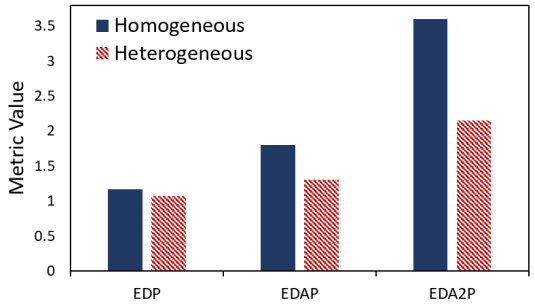

Fig. 11. Average normalized metric values are shown for homogeneous and heterogeneous constraints in area and CPI for different metrics for five systems. For heterogeneous constraints, normalized CPI thresholds are 1.2 for SPLASH-2 and NPB, 1.6 for SPEC and 2.4 for EEMBC. Area thresholds are 15 mm^2 for EEMBC, 25 mm^2 for SPEC, and no area constraint for SPLASH and NPB. Homogeneous constraints impose no area threshold and normalized CPI threshold 1.2 for all suites.

the optimal set of chiplets is even more important in this context.

J. Optimization With Heterogeneous Constraints

A likely common case for multisystem/chiplet optimization is when constraints and/or objectives for different application domains are very different. Our optimization framework can be constrained in almost arbitrary ways or setup different optimization objectives for different workloads or workload suites. As an example, embedded workloads may be severely cost limited while servers are performance limited. Fig. 11 considers one such case and compares the results with all applications uniformly constrained by a server performance constraint. One can see the dramatic reduction in overall cost-based metrics (EDAP, EDA^2P) as a result. EEMBC workloads are now mapped to much smaller cores with worse CPI but better EDP while SPLASH-2 and NPB continue to use bulkier systems.

VII. Conclusion

Traditionally, processor DSE has focused on identifying one system that maximized performance or efficiency. However, applications are often targeted using a family of processors, where each processor targets a subset of applications. As building multiple SoCs become costlier, chiplet integration technologies enable cost-effective system customization. In this article, we have developed the first multichiplet processor DSE framework. Our results clearly show benefits of using multiple chiplet assembly-based processors to service diverse workloads (35% improvement in EDP over a single best average system), and advantages of cognizance of chiplets during optimization (over 2× reduction in chiplets required for a heterogeneous CMP). We also identify the different factors that affect the cost of building multiple systems using chiplets as well as the characteristics of the systems/chiplets chosen under different use contexts. Using the framework, we show that emerging chiplet assembly approaches are very promising when total cost of design and manufacturing is considered (up to 72% benefit in cost over SoC) while satisfying energy and performance requirements of every workload. This cost benefit strongly depends on system size and technology maturity.

Acknowledgment

The authors would like to thank the anonymous reviewers and M. Tomei for their helpful feedback and suggestions.

References

[1] *Intel Xeon Processors*. Accessed: Apr. 25, 2019. [Online]. Available: http://www.intel.com/content/www/us/en/products/processors/xeon.html
[2] *Intel Xeon e7 Processors*. Accessed: Apr. 25, 2019. [Online]. Available: http://www.intel.com/content/www/us/en/products/processors/xeon/e7-processors.html
[3] *Intel Xeon e5 Processors*. Accessed: Apr. 25, 2019. [Online]. Available: http://www.intel.com/content/www/us/en/products/processors/xeon/e5-processors.html
[4] *Intel Xeon e3 Processors*. Accessed: Apr. 25, 2019. [Online]. Available: http://www.intel.com/content/www/us/en/products/processors/xeon/e3-processors.html
[5] (2014). *McKinsey on Semiconductors*. [Online]. Available: http://www.mckinsey.com/industries/semiconductors/our-insights/
[6] *SemiWiki-All Things Semiconductor*. Accessed: Apr. 25, 2019. [Online]. Available: https://www.semiwiki.com/forum/content/2991-faster-cooler-simpler-could-fd-soi-cheaper-too.html
[7] S. Y. Hou *et al.*, "Wafer-level integration of an advanced logic-memory system through the second-generation cowos technology," *IEEE Trans. Electron Devices*, vol. 64, no. 10, pp. 4071–4077, Oct. 2017.
[8] M. M. Kim, M. Mehrara, M. Oskin, and T. Austin, "Architectural implications of brick and mortar silicon manufacturing," in *Proc. 34th Annu. Int. Symp. Comput. Archit. (ISCA)*, San Diego, CA, USA, vol. 35, 2007, pp. 244–253, doi: 10.1145/1250662.1250693.
[9] T. Vijayaraghavan *et al.*, "Design and analysis of an APU for exascale computing," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 85–96
[10] R. Mahajan *et al.*, "Embedded multi-die interconnect bridge (EMIB)–a high density, high bandwidth packaging interconnect," in *Proc. IEEE 66th Electron. Compon. Technol. Conf. (ECTC)*, May 2016, pp. 557–565.
[11] S. Jangam, S. Pal, A. A. Bajwa, S. Pamarti, P. Gupta, and S. S. Iyer, "Latency, bandwidth and power benefits of the superchips integration scheme," in *Proc. IEEE Electron. Compon. Technol. Conf. (ECTC)*, May 2017, pp. 86–94.
[12] D. B. Papworth, "Tuning the Pentium Pro microarchitecture," *IEEE Micro*, vol. 16, no. 2, pp. 8–15, Apr. 1996.
[13] J. Kin, C. Lee, H. W. Mangione-Smith, and M. Potkonjak, "Power efficient mediaprocessors: Design space exploration," in *Proc. 36th Annu. ACM/IEEE Design Autom. Conf.*, New York, NY, USA, Jun. 1999, pp. 321–326.
[14] V. Desmet, S. Girbal, and O. Temam, "ArchExplorer.org: Joint compiler/hardware exploration for fair comparison of architectures," in *Proc. 13th Workshop Interact. Between Compil. Comput. Archit.*, Raleigh, NC, USA, 2009, p. 10.
[15] C. Silvano *et al.*, "MULTICUBE: Multi-objective design space exploration of multi-core architectures," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Jul. 2010, pp. 488–493.
[16] S. Kang and R. Kumar, "Magellan: A search and machine learning-based framework for fast multi-core design space exploration and optimization," in *Proc. Conf. Design, Autom. Test Eur.*, New York, NY, USA, 2008, pp. 1432–1437.
[17] H. Calborean and L. Vinţan, "An automatic design space exploration framework for multicore architecture optimizations," in *Proc. 9th RoEduNet IEEE Int. Conf.*, Jun. 2010, pp. 202–207.
[18] R. Kumar, D. M. Tullsen, and N. P. Jouppi, "Core architecture optimization for heterogeneous chip multiprocessors," in *Proc. 15th Int. Conf. Parallel Archit. Compilation Techn.*, 2006, pp. 23–32.
[19] N. K. Choudhary *et al.*, "FabScalar: Composing synthesizable RTL designs of arbitrary cores within a canonical superscalar template," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 3, p. 11–22, 2011.
[20] Y. Li, C. Benjamin Lee, M. David Brooks, Z. Hu, and K. Skadron, "CMP design space exploration subject to physical constraints," in *Proc. 12th Int. Symp. High-Perform. Comput. Archit.*, Feb. 2006, pp. 17–28.
[21] F. Remond and P. Bricaud, "Set top box SoC design methodology at STmicroelectronics," in *Proc. Design, Autom. Test Europe Conf. Exhib.*, Mar. 2003, pp. 220–223.

[22] R. Saleh *et al.*, "System-on-chip: Reuse and integration," *Proc. IEEE*, vol. 94, no. 6, pp. 1050–1069, Jun. 2006.

[23] G. Martin, R. Seepold, T. Zhang, L. Benini, and G. D. Micheli, "Component selection and matching for IP-based design," in *Proc. Conf. Design, Autom. Test Eur.*, Mar. 2001, pp. 40–46.

[24] T. G. Lenihan, L. Matthew, and E. J. Vardaman, "Developments in 2.5D: The role of silicon interposers," in *Proc. IEEE 15th Electron. Packag. Technol. Conf. (EPTC)*, Dec. 2013, pp. 53–55.

[25] C. F. Tseng, C. S. Liu, C. H. Wu, and D. Yu, "Info (wafer level integrated fan-out) technology," in *Proc. IEEE 66th Electron. Compon. Technol. Conf. (ECTC)*, May 2016, pp. 1–6.

[26] M. J. Schulte *et al.*, "Achieving exascale capabilities through heterogeneous computing," *IEEE Micro*, vol. 35, no. 4, pp. 26–36, Jul. 2015.

[27] A. Kannan, N. E. Jerger, and H. Gabriel Loh, "Enabling interposer-based disintegration of multi-core processors," in *Proc. 48th Int. Symp. Microarchitecture*, New York, NY, USA, 2015, pp. 546–558.

[28] *Mochi Architecture*. Accessed: Apr. 25, 2019. [Online]. Available: http://www.marvell.com/architecture/mochi/

[29] (2006). *SPEC-Standard Performance Evaluation Corporation*. [Online]. Available: http://www.specbench.org/osg/cpu2006/

[30] *EEMBC-Embedded Microprocessor Benchmarks*. Accessed: Apr. 25, 2019. [Online]. Available: https://www.eembc.org/

[31] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: Characterization and methodological considerations," in *Proc. 22nd Annu. Int. Symp. Comput. Archit.*, Jun. 1995, pp. 24–36.

[32] D. H. Bailey *et al.*, "The NAS parallel benchmarks," *Int. J. High Perform. Comput. Appl.*, vol. 5, no. 3, pp. 63–73, 1991.

[33] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore archit.," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Dec. 2009, pp. 469–480.

[34] S. Kang and R. Kumar, "Magellan: A search and machine learning-based framework for fast multi-core design space exploration and optimization," in *Proc. Design, Autom. Test Eur.*, Mar. 2008, pp. 1432–1437.

[35] Intel. *Advanced Interface Bus*. Accessed: Apr. 25, 2019. [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/programmable/heterogeneous-integration/overview.html/

[36] *IBM ILOG*. Accessed: Apr. 25, 2019. [Online]. Available: https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

[37] I. Tirkel, "Yield learning curve models in semiconductor manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 26, no. 4, pp. 564–571, Nov. 2013.

[38] E. Trevor Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations," in *Proc. Int. Conf. for High Perform. Comput., Netw., Storage Anal. (SC)*, Nov. 2011, pp. 52:1–52:12.

[39] A. McCallum, K. Nigam, and H. Lyle Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 2000, pp. 169–178.

[40] A. Phansalkar, A. Joshi, and L. K. John, "Analysis of redundancy and application balance in the SPEC CPU2006 benchmark suite," in *Proc. ACM SIGARCH Comput. Archit. News*, vol. 35, no. 2, May 2007, pp. 412–423.

[41] H. Patil and T. Carlson, "Pinballs: Portable and shareable user-level checkpoints for reproducible analysis and simulation," in *Proc. Workshop Reproducible Res. Methodol. (REPRODUCE)*, Orlando, FL, USA, 2014, p. 2.

[42] M. E. Thomadakis, "The architecture of the Nehalem processor and Nehalem-EP SMP platforms," *Resource*, vol. 3, no. 2, pp. 30–32, 2011.

[43] E. Sperling. *How Much Will That Chip Cost?* Accessed: Mar. 27, 2014. [Online]. Available: http://semiengineering.com/how-much-will-that-chip-cost/

[44] eSilicon. *How to Improve the Profitability of Fabless Semiconductor Companies*. Accessed: Apr. 25, 2019. [Online]. Available: http://esilicon.com/

[45] *Wafer Price is Hiking Up*, NVIDIA Corporation, Santa Clara, CA, USA, 2012.

[46] D. Stow, I. Akgun, R. Barnes, P. Gu, and Y. Xie, "Cost analysis and cost-driven IP reuse methodology for soc design based on 2.5D/3D integration," in *Proc. ICCAD*, Nov. 2016, pp. 1–6.

[47] *Yield Enhancement*, ITRS, 2007.

[48] *Gartner Says Worldwide Sales of Smartphones Returned to Growth in First Quarter of 2018*. Accessed: Apr. 25, 2019. [Online]. Available: http://www.gartner.com/newsroom/id/3876865

[49] *Quarterly Personal Computer (PC) Vendor Shipments Worldwide, From 2009 to 2018, by Vendor (in Million Units)*. Accessed: Apr. 25, 2019. [Online]. Available: https://www.statista.com/statistics/263393/global-pc-shipments-since-1st-quarter-2009-by-vendor/

[50] Gartner. *Newsroom, 3530117 and 3560517*. Accessed: [Online]. Available: http://www.gartner.com/newsroom/id/3560517 and http://www.gartner.com/newsroom/id/3530117

[51] O. Tange, "GNU parallel: The command-line power tool," *USENIX Mag.*, vol. 36, no. 1, pp. 42–47, 2011.

Saptadeep Pal (Student Member, IEEE) received the B.Tech. degree in electrical engineering from IIT Patna, Patna, India, in 2015 and the M.S. degree from the University of California at Los Angeles, Los Angeles, CA, USA, in 2017, where he is currently working toward the Ph.D. degree, where his research focuses on design and architecture of packageless and wafer-scale processor systems based on novel high-performance interconnect fabrics.

Daniel Petrisko (Student Member, IEEE) received the M.S. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2018, where his research focused on architecting wafer-scale systems and many-die processors. He is currently working toward the Ph.D. degree at the Paul G. Allen Center for Computer Science and Engineering, University of Washington, Seattle, WA, USA.

His current research focuses on designing flexible, reconfigurable microarchitectures, and accelerating open-source hardware design.

Rakesh Kumar (Senior Member, IEEE) received the B.Tech. degree from IIT Kharagpur, Kharagpur, India, in 2001, and the Ph.D. degree from the University of California at San Diego, La Jolla, CA, USA, in 2006.

He is currently an Associate Professor with the Electrical and Computer Engineering Department, University of Illinois at Urbana–Champaign, Champaign, IL, USA, with research and teaching interests in computer architecture, hardware design, and low power, trustworthy, and error resilient computer systems.

Dr. Kumar was a recipient of the Stanley H. Pierce Faculty Award, the Mahatma Gandhi Pravasi Samman, Ronald W. Pratt Faculty Outstanding Teaching Award, the ARO Young Investigator Award, and the UCSD CSE Best Dissertation Award. His public service has been recognized by the Nelson Mandela Leadership Award. He previously served as the Co-Founder and Chief Architect at Hyperion Core, Inc., a microprocessor startup aimed at bringing polymorphous grid processor technology to the market.

Puneet Gupta (Senior Member, IEEE) received the B.Tech. degree in electrical engineering from IIT Delhi, New Delhi, India, in 2000 and the Ph.D. degree from the University of California at San Diego, La Jolla, CA, USA, in 2007.

He is currently a Professor with the Electrical and Computer Engineering Department, University of California at Los Angeles, Los Angeles, CA, USA. He co-founded Blaze DFM Inc., Sunnyvale, CA, USA, in 2004, and served as its Product Architect until 2007. He has authored over 150 articles, a book, and a book chapter, and holds 17 U.S. patents. His current research interests include building high-value bridges across application-architecture-implementation-fabrication interfaces for lowered cost and power, increased yield, and improved predictability of integrated circuits and systems.

Dr. Gupta was a recipient of the NSF CAREER Award, the ACM/SIGDA Outstanding New Faculty Award, the SRC Inventor Recognition Award, and the IBM Faculty Award. He led the multiuniversity IMPACT+ Center which focused on future semiconductor technologies.