INVITED: The Case for Embedded Scalable Platforms

Luca P. Carloni
Department of Computer Science
Columbia University in the City of New York, New York, NY 10027
luca@cs.columbia.edu

ABSTRACT

Heterogeneous system-on-chip (SoC) architectures are emerging as a fundamental computing platform across a variety of domains, from mobile to cloud computing. Heterogeneity, however, increases design complexity in terms of hardware-software interactions, access to shared resources, and diminished regularity of the design. Embedded Scalable Platforms are a novel approach to SoC design and programming that addresses these design-complexity challenges by combining an architecture and a methodology. The flexible socketed architecture simplifies the integration of heterogeneous components by balancing regularity and specialization. The companion methodology raises the level of abstraction to system-level design, thus promoting closer collaboration among software programmers and hardware engineers. The architecture is supported by a scalable communication infrastructure. The methodology leverages compositional design-space exploration with high-level synthesis. The case for Embedded Scalable Platforms is made based on experiments on the development of various full-system prototypes and experience in teaching these concepts in a new graduate course.

1. INTRODUCTION

The natural progression towards the emergence of the heterogeneous SoC as the main computing platform across many application domains [21, 24, 26] is a consequence of the end of Dennard's ideal CMOS scaling. Facing the inability of continuing to scale down supply voltages, the designers of integrated circuits have made two consecutive moves. First, they moved towards parallelism by building homogeneous multicore architectures that integrate multiple, relatively simpler, processor cores instead of a single, more complex, hyper-pipelined processor. Then, they moved towards heterogeneity by combining the processor cores with an increasing number of specialized-hardware accelerators, each capable of executing a dedicated function in a way that is orders of magnitude more efficient than its corresponding software execution [6, 17]. As a result, a state-of-the-art SoC features a rich mix of heterogeneous components, including multiple general-purpose processor cores, graphics processing units, accelerators, memory subsystems, and I/O peripherals. To a large extent, many of these components

DAC '16, June 05 - 09, 2016, Austin, TX, USA

ISBN 978-1-4503-4236-0/16/06…$15.00

DOI: http://dx.doi.org/10.1145/2897937.2905018

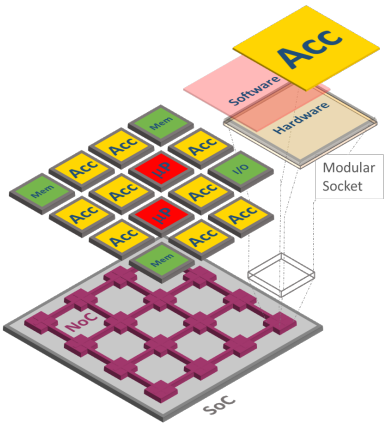

Figure 1: Integrating heterogeneous components in an Embedded Scalable Platform.

can work in parallel, either jointly or independently. In particular, each accelerator may be activated by a software application only when it is necessary to execute its particular function. Thanks to new technologies for fine-grain dynamic power management [1, 7, 29, 30], it is possible to power up only those SoC components needed to run the current workload. It is this ability of independent activation of selected specialized components that makes parallel heterogeneous architectures a value proposition to preserve the progress of the semiconductor industry in the so-called "age of dark silicon", when power limitations are stringent and homogeneous multicore scaling has hit diminished returns [13].

On the other hand, the more heterogeneous components are integrated into an SoC, the more complex becomes the integration process in terms of hardware-software interactions, access to shared resources, and diminished regularity of the design. Indeed, the critical challenges of SoC design are in the integration, programming, and management of many heterogeneous components more than in the design of any individual component. Addressing these challenges will require to move away from processor-centric architectures towards more distributed architectures that are developed in combination with scalable and compositional design methodologies. By combining a flexible socketed architecture with a companion system-level design (SLD) methodology, *Embedded Scalable Platforms (ESP)* is a step in this direction.

2. A SCALABLE ARCHITECTURE

Embedded Scalable Platforms seek to balance hardware specialization and design regularity by means of a tile-based heterogeneous multi-core architecture. As illustrated in Fig. 1, an SoC is an instance of an ESP architecture that is obtained by specifying a mix of tiles. Each tile may im-

plement a processor core (capable of running an operating system like Linux), a hardware accelerator, or some auxiliary functionality like I/O access. The number and mix of tiles of a particular architecture varies depending on its target application domain. The choice of a specific tile combination is the result of an application-driven *design-space exploration* that is guided by the methodology presented in Section 3.

Socket Interfaces. The integration of heterogeneous components is simplified because they can be "plugged in" a communication infrastructure through the instantiation of *modular socket interfaces*. A socket interface, which is parameterized and synthesizable starting from a generic template, encapsulates a component and implements the communication mechanisms together with other *ESP services*. In particular, for the case of an accelerator, the hardware-part of a socket typically implements: a configurable direct-memory access (DMA) engine, interrupt-request logic, and memory-mapped registers. The registers can be accessed through the software-part of the socket, which is running on a processor. Each processor is encapsulated in a tile through a socket instance implementing another set of ESP services. Some of these services include: an *ESP Linux module*, which allows the operating system to recognize all accelerators in the SoC, some *ESP Linux device drivers* to configure them, and an *ESP user-level library* which simplifies accelerator programming. In particular, the integration of a new accelerator simply requires instancing a device driver from a predefined template with very few modifications (typically less than 2% of the driver code) to specify the behavior of its control registers [23]. In a processor tile, the DMA engine is replaced by a cache, which gives the illusion of a traditional homogeneous system and decouples the processor bus from the rest of the SoC. Hence, legacy software can execute transparently on the processor.

On-Chip Communication. The platform services are supported by the sockets through a *scalable communication and control infrastructure (SCCI)*. The infrastructure, which is instrumental to accommodate heterogeneous concurrency and computing locality in ESP, is also automatically synthesized from predesigned templates. The SCCI has a two-fold role: at design time it simplifies the integration of heterogeneous tiles through the socket interfaces; at run time, it provides efficient inter-tile data communication with the integrated support for the ESP services. In particular, it supports the realization of fine-grained power management with multiple independent voltage/frequency domains. Thanks to the progress in developing integrated voltage regulators [1, 7, 29, 30], each domain can correspond to an individual ESP tile, thus providing fine granularity both in time and space. Each regulator is managed through a dedicated controller for dynamic voltage-frequency scaling. The controller is part of the tile hardware socket but is configurable through memory-mapped registers. Hence, it can continuously set the optimal operating point for its tile by enforcing either a local policy or allowing reconfiguration from software, which can override the local decisions in favor of a system-level policy [22]. The policy actuation is based on the information provided by local performance counters and collected by the SCCI, which acts as the "ESP nervous system" as it continuously senses the current state of each tile and distributes the actuation of the policy decisions.

The design complexity of the SCCI depends on the scale of the ESP instance and may vary from a simple bus to a packet-switched network-on-chip (NoC), which can be further scaled up by adding more virtual channels and/or multiple physical planes [31]. In any case, the low-level details of its circuit mechanisms are transparent to both the SoC architects and component designers, who work at a higher level of abstraction relying on transaction-level modeling [15]. The main goal here is to provide modularity and flexibility by decoupling computation tasks from communication tasks, following the Protocols and Shells Paradigm of latency-insensitive design [8, 9]. A practical demonstration of the fact that the SCCI implementation is transparent to both the processor and the accelerators is that an ESP bus-based design can be easily migrated to become part of a more complex ESP NoC-based design: no changes to the accelerators are required and the exact same operating system, device drivers, and application software can run on both the bus-based and NoC-based designs.

Accelerator Coupling. ESP accelerators are based on a *loosely-coupled accelerator model* [11]. They are designed independently from the processors so that they can work with any general-purpose processor core that is capable of running an operating system like Linux. In addition to improving reusability, this design decoupling gives more freedom to the accelerator designers. In particular, it enables the design of coarse-grained accelerator logic blocks with complex datapaths that yield an efficient implementation for a complete application kernel, like the Fast Fourier Transform or the Debayer algorithm. This implementation includes a large *private local memory (PLM)* which is carefully tailored to the accelerator's specific needs because it is critical for its performance. Typically, the PLM features aggressive SRAM banking that provides multi-ported memory accesses to match the multiple parallel blocks of the accelerator datapath. Since a large portion of the accelerator area consists of the PLM banks, the opportunity cost of investing die real estate on specialized accelerators can be efficiently mitigated by reusing it as a non-uniform cache architecture (NUCA) substrate [10, 12]. Being "out of core" and encapsulated in its own tile, an accelerator is simply accessed through the SCCI. With regards to software, abstracting an accelerator with device drivers similar to those for SoC on-chip devices is a low-complexity task and the runtime overhead becomes negligible as soon as the workset size becomes non trivial [11]. This is often the case since loosely-coupled accelerators are designed to process large worksets (tens to hundreds of megabytes).

3. A SCALABLE METHODOLOGY

The ESP design methodology is motivated by one main consideration: in order to continue sustaining the progress of the semiconductor industry in the face of growing SoC design complexity, it is necessary to raise the level of abstraction above register-transfer level (RTL) and realize the whole potential of *system-level design (SLD)* [8, 27].

System-Level Specification. For the design-entry point, the ESP methodology replaces RTL specifications in VHDL or Verilog with the use of SystemC, an IEEE-standard object-oriented language based on C++ [5, 18]. This reduces the gap between software and hardware, enables fast full-system simulation with virtual platforms under more significant environment conditions, and leverages *high-level synthesis (HLS)* to explore a broader space of al-

ternative implementations.

With a high-level language like SystemC, engineers can specify an accelerator while abstracting away all low-level logic and circuit details to focus instead on the relationships between the data structures and computational tasks that characterize the given algorithms. The benefits are higher productivity, less chances of errors, and more options for performance and power optimizations.

With SystemC and virtual platforms, engineers can simulate an accelerator together with the whole system in which it will operate. Differently from a cycle-accurate RTL simulator, a virtual platform allows the execution of realistic application scenarios on top of the actual software stack that will be deployed as part of the final system, including the operating system and any middleware layer. This allows engineers to develop the SoC under the guidance of the target applications that it will have to sustain when deployed in the field. The combination of SystemC and virtual platforms reduces the gap between application-software development and circuit-hardware design by providing a framework for collaboration: programmers can run and refine their software on the hardware model while designers can test and optimize their hardware accounting for the inputs from the programmers.

With SystemC and HLS tools, engineers can optimize an accelerator by exploring a broad design space. Admittedly, only a subset of SystemC is currently synthesizable by commercial HLS tools. This subset, however, is already sufficient to specify complex components and quickly synthesize many RTL implementations that are competitive with those that can be manually designed by an experienced engineer. Further, thanks to the much richer set of configuration knobs provided by HLS tools, starting from the same high-level specification it is possible to synthesize many alternative RTL implementations that differ in terms of their cost and performance. To obtain the same number of implementations with manual RTL design is prohibitive in terms of non-recurring engineering costs.

In summary, by using SystemC and SLD methods a hardware component like an accelerator can be designed more efficiently with a focus on its algorithmic properties, can be simulated faster under more significant environment conditions, and can be optimized more productively through the HLS-driven exploration of a broader design space.

Virtual Platforms. Fig. 2 shows the relationship between the ESP architecture and methodology. The methodology is supported by a combination of state-of-the-art commercial CAD tools with complementary tools that have been developed in-house to overcome some critical limitations of the commercial ones. These include tools to leverage compositional and learning methods in the application of HLS [19, 20] and tools for optimizing communication and PLM during the synthesis of accelerators [16, 25]. The premise of the ESP approach is that a set of key target application workloads must drive the software-programming and hardware-design efforts throughout all stages of the SoC realization. This is supported by an in-house virtual platform that runs the same software stack of the final system. As the bulk of the SoC design effort consists in the integration of many heterogeneous components, full-system simulation becomes increasingly important to test and evaluate their design together with the operating system.

IP Block Development and Design Reuse. The

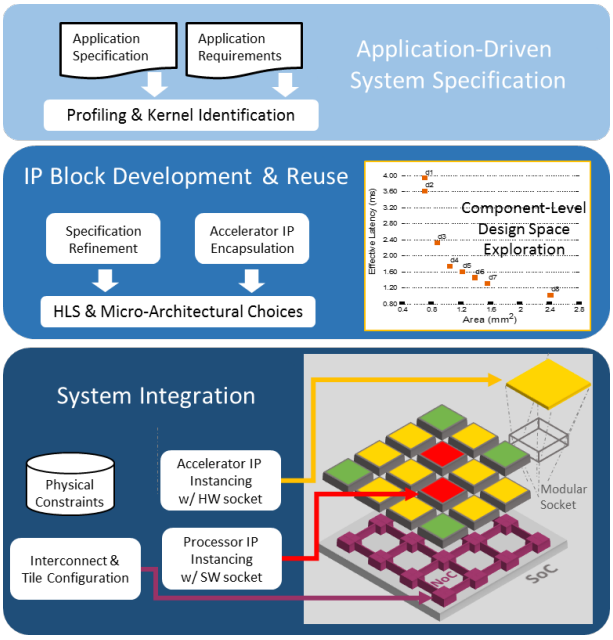

Figure 2: SLD methodology for ESP.

rich set of configuration knobs provided by commercial HLS tools allows a broad design-space exploration. Engineers can choose a particular knob configuration before invoking the HLS engine, which returns a corresponding optimized micro-architecture expressed at the RTL, e.g. in synthesizable Verilog. Different configurations produce different micro-architectures, thus enabling the choice among many alternative RTL implementations. As these implementations represent alternative tradeoffs in the multi-objective design space, HLS promotes Intellectual Property (IP) design reuse and exchange. For instance, a team of computer-vision experts can devise an innovative algorithm for object detection, use SystemC to design a specialized accelerator for this algorithm, and license it as a synthesizable IP block to many different SoC architects; each architect can then use HLS to derive automatically the particular implementation that provides the best tradeoff (e.g. higher performance or lower power) for a particular SoC. For example, the diagram at the center of Fig. 2 shows the result of the design-space exploration for a Debayer accelerator [23]: the 8 points correspond to 8 Pareto-optimal distinct micro-architectures that are synthesized from a SystemC specification using a commercial HLS tool and an in-house tool for PLM optimization [25]. All the HLS-synthesized implementations are not strictly equivalent from an RTL viewpoint because they do not produce exactly, i.e. clock by clock, the same sequence of output signals for any valid sequence of input signals. On the other hand, they are all valid RTL implementations of the original SLD specification, which is given as an untimed SystemC model. In this sense, they belong to a latency-equivalent class and any of them can be integrated within an ESP instance because its tile socket implements a latency-insensitive protocol [8, 9].

System Integration. The ESP system integration relies on the modular socket and infrastructure described in Section 2 and transaction-level modeling primitives [15]. These primitives follow the Protocols and Shells Paradigm in using point-to-point channels, which are inherently latency insensitive, combined with modular socket interfaces, which can

be instanced to connect the processes to the channels. CAD-tool vendors provide libraries of such primitives that offer abstracted functions to specify the communication and synchronization mechanisms among computation processes at the system level as well as synthesizable implementations of these mechanisms that can be combined with the implementation of SystemC processes in a modular fashion [14, 28]. By decoupling the computation and communication parts, transaction-level modeling enables a more efficient design of the SCCI. The SCCI itself can be synthesized from its parameterized specification while optimizing its properties (e.g. number of physical planes, number of virtual channels) to satisfy the communication requirements of the given SoC design. In summary, the decoupling of computation from communication reduces the complexity of the design effort at the specification level and supports the realization of SoC architectures that are highly scalable because they balance the specialization of their components with the modularity of the overall organization.

Rapid Full-System Prototyping. The methodology enables the rapid realization of complete prototypes of Embedded Scalable Platforms using both FPGA and ASIC technologies. For instance, as part of a project to build an accelerator for the Wide-Area Motion Imagery computer-vision application from the PERFECT Benchmark Suite [4], we performed an extensive design-space exploration by generating many alternative SoCs [23]. Each SoC is an ESP instance featuring one LEON-3 processor tile, two I/O tiles connected to DRAM banks, a multi-plane NoC, and a set of 12 to 15 accelerator tiles. For each ESP instance we built a prototype on a XILINX VIRTEX7 FPGA. In addition to the number of accelerators, these SoCs differ also for the choice of the Pareto-optimal implementations of some key accelerators. In another project, we used some of these prototypes together with the prototypes of other systems (an SoC featuring 10 independent heterogeneous accelerators from the PERFECT suite and an SoC featuring 12 copies of an FFT-2D accelerator) to drive the design of an FPGA-based infrastructure for rapid prototyping and emulation of the ESP services for fine-grain power management [22]. This project confirms that the ability to realize rapidly full-system prototypes is critically important to analyze the complex interactions between hardware and software when an SoC executes multiple applications with very large worksets, a task that simulation cannot adequately support [2].

4. A SLD ECOSYSTEM

An important objective of the ESP project is to develop the capabilities that are necessary to realize a new ecosystem for SLD. The new ecosystem will promote the collaboration between two main professional figures: the SoC architect and the IP core designer. The goal of the SoC architect is to design an innovative system that is optimized for a target application domain or a specific class of applications while minimizing the non-recurring engineering costs. The goal of the IP core designer is to design and validate a specialized component that, thanks to unique intellectual properties, offers superior performance in delivering a particular functionality while being sufficiently flexible to be integrated in a variety of different SoCs. While distinct, the goals of SoC architects and IP core designers are clearly compatible as they compete with their peers in different fields and need each other's help to succeed. But what they need is also an

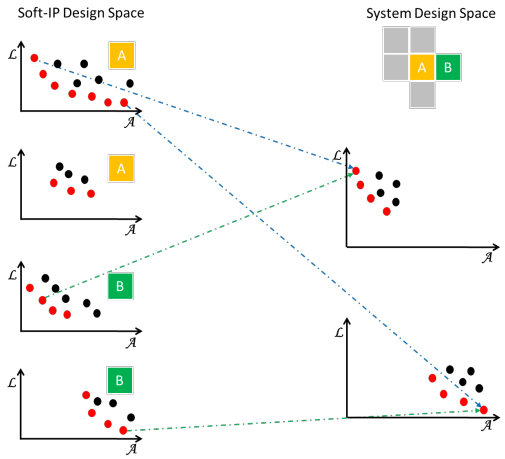

Figure 3: Compositional SLD and design reuse.

environment that allows them to interact effectively. On one hand, it should enable the rapid evaluation of many alternative IP cores through a set of formal metrics for performance and cost so that the architects can select the right one for their specific SoCs. On the other hand, it should collect precious feedback from many possible system-integration attempts so that the designers can refine their IP cores.

In this context, the notion of Pareto-optimal frontier provides a sound metric for reusability that applies to both individual IP cores and complete SoC designs. The four diagrams on the left-hand side of Fig. 3 represent the results of the design-space exploration performed by four distinct IP designers. The first two designers compete in producing accelerator A and the second pair of designers compete in produced accelerator B; instead, the two diagrams on the right-hand side represent the results of the design-space exploration performed by two distinct architects for an SoC that includes an instance of both A and B. The results are shown in a bi-objective optimization space, where the metric for performance $\mathcal{L}$ is the latency taken by the accelerator to execute a task and the metric for cost $\mathcal{A}$ is its area occupation. The red dots correspond to Pareto-optimal designs. In this simple example, the architect of the "top SoC" focuses on realizing a low-cost implementation while the architect of the "bottom SoC" aims at a high-performance one. In doing so, they select implementations of the accelerators that best contribute to their goals. For example, Fig. 3 shows that while they license two distinct implementations of accelerator B from two different IP designers, for accelerator A they license two distinct implementations from the first IP designer, who has been able to realize a more reusable design for A than the second IP designer. ESP facilitates this kind of compositional SLD with IP reuse because: different implementations of any given accelerator (1) can be automatically generated via HLS and evaluated through rapid full-system prototyping and (2) can be replaced in an ESP architecture without changing the rest of the system.

5. TEACHING SLD WITH ESP

The ideas described in the previous sections are the foundation of *System-on-Chip Platforms*, a new course that I developed at Columbia University over the last five years. In this course the students learn the hardware and software aspects of: (1) integrating heterogeneous components into a

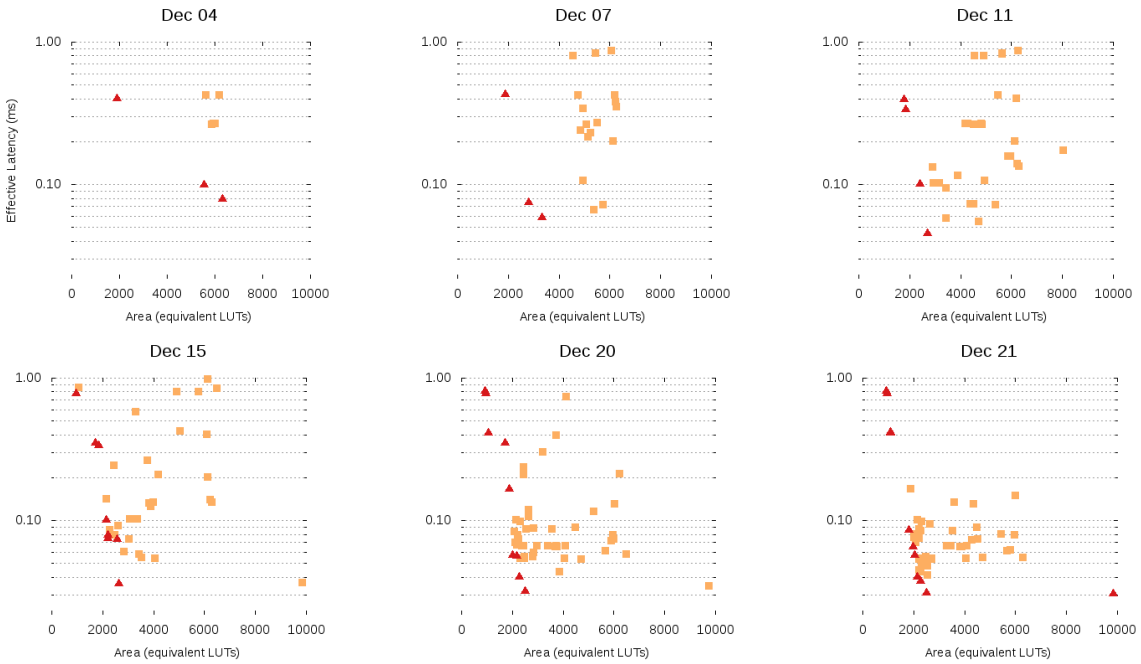

Figure 4: Progression of the design-space exploration for the project made by the F-15 SoC Platforms class.

complete system; (2) evaluating designs in a multi-objective optimization space; and (3) designing new components that are reusable across different systems, product generations, and implementation platforms.

Originally offered as a graduate-level course, System-on-Chip Platforms is now part of the regular upper-level curriculum for the Computer Engineering program. The course consists of two main tracks that run in parallel throughout the semester. A *theory track* covers the principles of system-level design including: models of computations, virtual prototyping, design-space exploration, hardware/software co-design, component integration, memory organization, communication infrastructure, and power management. A *practice track* includes the presentation of: the SystemC programming language, transaction-level modeling, programming of software applications and device drivers with virtual platforms, and design of hardware accelerators with HLS tools. The theory track is illustrated by the presentation of case studies based on recent chips from industry and academia. The practice track is supported by extensive use of commercial tools (e.g. for HLS) as well as in-house tools (e.g. virtual platform, accelerator memory generation).

The course requires the completion of a project in the second half of the semester, after a series of homework assignments have helped students to gain practice with both the preliminary material and the project infrastructure. The project is structured as a design contest. Most teams consist of two students, although there is also the option of working individually. For instance, in Fall 2015 twenty-one teams competed in designing a hardware accelerator for the Gradient algorithm, a computer-vision kernel from the Perfect Benchmark Suite [4]. Each team specified the accelerator design in SystemC, wrote a device driver to integrate it with the application running on an embedded processor, and validated the hardware/software co-design using an in-house virtual platform. Then, the teams performed a design-space exploration with the commercial HLS tool Cadence C-to-Silicon, targeting a Xilinx FPGA platform.

For each team, the goal is to obtain three distinct implementations of the given accelerator that must correspond to three different trade-off points in terms of performance and area. The quality of each final implementation is evaluated in the context of the work done by the entire class: Pareto-optimal implementations receive the highest score, while the penalty for Pareto-dominated implementations is proportional to the distance from the Pareto frontier. Additionally, a system of incentives encourages the students to keep improving their work by committing new versions into the design repository. Specifically, teams receive extra credits for each day when they commit at least one new improved design. Throughout the one-month duration of the project, a live *Pareto-efficiency plot* reporting the current position of the three best design implementations for each team in the bi-objective design space is made available on the course webpage. This allows students to continuously assess their performance with respect to the rest of the class (each given design point is identifiable through a label known only to the instructors and the corresponding team members).

On average, over the one-month period of the Fall-2015 project, each team committed over 30 design improvements. The Pareto frontier changed every day and a total of 99 times across the project duration. Snapshots of the status of the plot taken at the end of the day are shown in Fig. 4 for six distinct days. The x axis reports the cost metric in terms of area occupation measured as the number of equivalent FPGA look-up tables (LUT), while the y axis reports the effective latency (in ms) that each accelerator takes to execute the Gradient algorithm. The Pareto-optimal designs are denoted with triangles. The final plot contains a Pareto frontier with eleven designs that span a range of about 26× in performance and 10× in area cost, thus providing a rich set of alternative implementations for this accelerator. Table 1 reports additional statistics on the Fall-2015 project.

Plans for future editions of the course include the intro-

Table 1: The SoC Platforms Project by Numbers.

Number	Description
21	Number of student teams
661	Number of improved designs across all teams
31.5	Average number of improved designs per team
1.5	Average number of improved designs committed each day per team
99	Total number of changes of the Pareto curve over the project period
11	Final number of Pareto-optimal designs
26×	Performance range of Pareto curve
10×	Area range of Pareto curve

duction of incentives for component reuse and the application of learning-based [19] and compositional SLD exploration methods [20] to enable the design of more complex systems in the project contest. As the course enrollment continues to grow, the project can be scaled up to include the design of multiple accelerators for a target SoC, while balancing competition and collaboration aspects. This can be done by assigning the design of each accelerator to a subset of the student teams and offering incentives for collaboration across teams working on different accelerators. Besides providing students with the opportunity to design and evaluate a component in the context of a large system, scaling the project scope follows one of the recommendations emerged from the recent CCC Workshop Series on Extreme Scale Design Automation, i.e.: "As custom design is being displaced by more automated design styles, university courses should highlight abstractions (e.g., system-level design) needed to manage designs that far exceed those built at universities" [3].

6. CONCLUSIONS

Embedded Scalable Platforms provide the right balance between regularity and specialization to address the challenges of designing and programming heterogeneous system-on-chip architectures. The companion system-level design methodology improves the productivity of embedded designers and programmers by promoting the use of virtual platforms, high-level synthesis, and component reuse.

Acknowledgments. The author would like to thank the members of the System-Level Design Group at Columbia University, and particularly the ESP team, including: E. Cota, G. Di Guglielmo, P. Mantovani, and C. Pilato. This work was supported in part by DARPA PERFECT (C#: HR0011-13-C-0003), the National Science Foundation (A#: 1219001 and 1527821), and the Center for Future Architectures Research (C-FAR) (C#: 2013-MA-2384), one of the six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

7. REFERENCES

[1] Andersen, T. M., et al. A feedforward controlled on-chip switched-capacitor voltage regulator delivering 10W in 32nm SOI CMOS. In *ISSCC Digest of Technical Papers* (Feb. 2015), pp. 22–26.

[2] Arvind. Simulation is passé; all future systems require FPGA prototyping. *Keynote Address at Embedded System Week (ESWEEK)* (Oct. 2014).

[3] Bahar, I., et al. "Scaling" the impact of EDA education — preliminary findings from the CCC workshop series on extreme scale design automation. In *Intl. Conf. on Microelectronic Systems Education (MSE)* (June 2013), pp. 64–67.

[4] Barker, K., et al. *PERFECT (Power Efficiency Revolution For Embedded Computing Technologies) Benchmark Suite Manual*. PNNL and GTRI, Dec. 2013. http://hpc.pnnl.gov/perfect.

[5] Black, D. C., Donovan, J., Bunton, B., and Keist, A. *SystemC: From the Ground Up, Second Edition*. Springer, 2009.

[6] Borkar, S., and Chen, A. The future of microprocessors. *Communication of the ACM 54* (May 2011), 67–77.

[7] Burton, E. A., et al. FIVR – fully integrated voltage regulators on 4th generation Intel Core SoCs. In *Applied Power Electronics Conference and Exposition* (Mar. 2014), pp. 16–20.

[8] Carloni, L. P. From latency-insensitive design to communication-based system-level design. *Proc. of the IEEE 103*, 11 (Nov. 2015), 2133–2151.

[9] Carloni, L. P., McMillan, K. L., and Sangiovanni-Vincentelli, A. L. Theory of latency-insensitive design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 20*, 9 (Sept. 2001), 1059–1076.

[10] Cota, E., et al. Accelerator memory reuse in the dark silicon era. *Computer Architecture Letters 13*, 1 (Jan-Jun 2014), 9–12.

[11] Cota, E., et al. An analysis of accelerator coupling in heterogeneous architectures. In *Proc. of the Design Automation Conf. (DAC)* (June 2015), pp. 202:1–202:6.

[12] Cota, E., Mantovani, P., and Carloni, L. P. Exploiting private local memories to reduce the opportunity cost of accelerator integration. In *Proc. of the Intl. Conf. on Supercomputing (ICS)* (June 2016).

[13] Esmaeilzadeh, H., et al. Dark silicon and the end of multicore scaling. In *Proc. of the Intl. Conf. on Computer Architecture ISCA)* (June 2011), pp. 365–376.

[14] Fingeroff, M. *High-level synthesis blue book*. Mentor Graphics Corp., 2010.

[15] Ghenassia, F. *Transaction-Level Modeling with SystemC*. Springer-Verlag, 2006.

[16] Guglielmo, G. D., Pilato, C., and Carloni, L. P. A design methodology for compositional high-level synthesis of communication-centric SoCs. In *Proc. of the Design Automation Conf. (DAC)* (June 2014), pp. 128:1–128:6.

[17] Horowitz, M. Computing's energy problem (and what we can do about it). In *ISSCC Digest of Technical Papers* (Feb. 2014), pp. 10–14.

[18] IEEE. SystemC Standardization Working Group. 1666-2011 - IEEE standard for standard SystemC reference manual.

[19] Liu, H.-Y., and Carloni, L. P. On learning-based methods for design-space exploration with high-level synthesis. In *Proc. of the Design Automation Conf. (DAC)* (June 2013).

[20] Liu, H.-Y., Petracca, M., and Carloni, L. P. Compositional system-level design exploration with planning of high-level synthesis. In *Proc. of the Conf. on Design, Automation and Test in Europe (DATE)* (Mar. 2012), pp. 641–646.

[21] Mair, H., et al. A highly integrated smartphone SoC featuring a 2.5GHz octa-core CPU with advanced high-performance and low-power techniques. In *ISSCC Digest of Technical Papers* (Feb. 2015), pp. 424–425.

[22] Mantovani, P., et al. An FPGA-based infrastructure for fine-grained DVFS analysis in high-performance embedded systems. In *Proc. of the Design Automation Conf. (DAC)* (June 2016).

[23] Mantovani, P., Guglielmo, G. D., and Carloni, L. P. High-level synthesis of accelerators in embedded scalable platforms. In *Proc. of the Asia and South Pacific Design Automation Conf. (ASPDAC)* (Jan. 2016).

[24] Mochizuki, S., et al. 20nm high-K metal-gate heterogeneous 64b quad-core CPUs and hexa-core GPU for high-performance and energy-efficient mobile application processor. In *ISSCC Digest of Technical Papers* (Feb. 2016), pp. 78–79.

[25] Pilato, C., et al. System-level memory optimization for high-level synthesis of component-based SoCs. In *Proc. of the Intl. Conf. on Hardware/Software Codesign and SystemSynthesis (CODES+ISSS)* (Oct. 2014), pp. 18:1–18:10.

[26] Pyo, J., et al. 20nm high-K metal-gate heterogeneous 64b quad-core CPUs and hexa-core GPU for high-performance and energy-efficient mobile application processor. In *ISSCC Digest of Technical Papers* (Feb. 2015), pp. 420–421.

[27] Sangiovanni-Vincentelli, A. L. Quo vadis SLD: Reasoning about trends and challenges of system-level design. *Proc. of the IEEE 95*, 3 (Mar. 2007), 467–506.

[28] Sanguinetti, J., Meredith, M., and Dart, S. Transaction-accurate interface scheduling in high-level synthesis. In *ESLsyn Conference* (2012), pp. 31–36.

[29] Sturcken, N., et al. A 2.5D integrated voltage regulator using coupled magnetic core inductors on silicon interposer delivering $10.8A/mm^2$. In *ISSCC Digest of Technical Papers* (Feb. 2012), pp. 400–402.

[30] Tien, K., et al. An 82%-efficient multiphase voltage-regulator 3D interposer with on-chip magnetic inductors. In *Symp. on VLSI Circuits* (June 2015), pp. C192–C193.

[31] Yoon, Y., Concer, N., and Carloni, L. P. Virtual channels and multiple physical networks: Two alternatives to improve NoC performance. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 32*, 12 (Dec. 2013), 1906–1919.