# A Scalable Methodology for Agile Chip Development with Open-Source Hardware Components

**(Invited Paper)**

Maico Cassel dos Santos[1,†], Tianyu Jia[2,†], Martin Cochet[3], Karthik Swaminathan[3], Joseph Zuckerman[1], Paolo Mantovani[1], Davide Giri[1], Jeff Jun Zhang[2], Erik Jens Loscalzo[1], Gabriele Tombesi[1], Kevin Tien[3], Nandhini Chandramoorthy[3], John-David Wellman[3], David Brooks[2], Gu-Yeon Wei[2], Kenneth Shepard[1], Luca P. Carloni[1], and Pradip Bose[3]
Columbia University[1], Harvard University[2], IBM Research[3]

## ABSTRACT

We present a scalable methodology for the agile physical design of tile-based heterogeneous system-on-chip (SoC) architectures that simplifies the reuse and integration of open-source hardware components. The methodology leverages the regularity of the on-chip communication infrastructure, which is based on a multi-plane network-on-chip (NoC), and the modularity of socket interfaces, which connect the tiles to the NoC. Each socket also provides its tile with a set of platform services, including independent clocking and voltage control. As a result, the physical design of each tile can be decoupled from its location in the top-level floorplan of the SoC and the overall SoC design can benefit from a hierarchical timing-closure flow, design reuse and, if necessary, fast respin. With the proposed methodology we completed two SoC tapeouts of increasing complexity, which illustrate its capabilities and the resulting gains in terms of design productivity.

## KEYWORDS

System-on-Chip, Agile Design, Open-Source Hardware, Heterogeneous Computing, Computer Architecture, Network-on-Chip.

## 1 INTRODUCTION

One of the major consequences of the slowdown of Moore's Law [9, 11] and the end of Dennard's scaling [4, 14] has been a continuous growth in the complexity of chip design. Heterogeneous system-on-chip (SoC) architectures that combine multicore processors and specialized hardware accelerators [13] on the same die have emerged as the preferred solution to achieve both performance and energy efficiency across all main application domains [22]. As the costs of developing a new leading-edge SoC continue to rise, new methodologies and platforms that support design reuse are needed to reduce them by an order of magnitude [35]. In this context, open-source hardware (OSH) can play a unique role to support design reuse by promoting entrepreneurial innovation and collaborative engineering across industry and academia [21]. The success

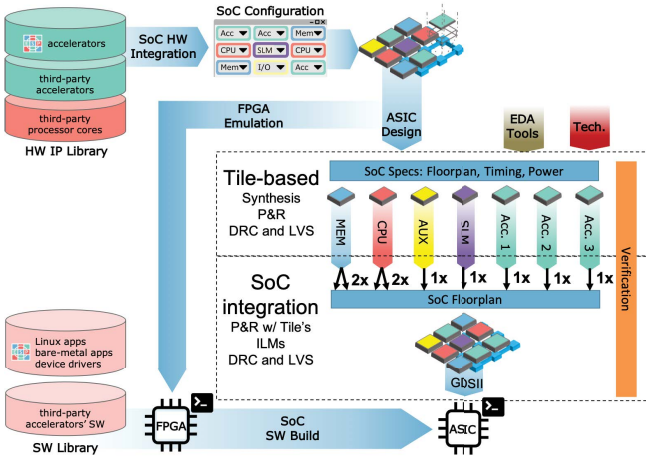


**Figure 1: Proposed methodology for agile chip development.**

of the RISC-V open standard Instruction Set Architecture (ISA) [32] has triggered a wave of new SoC architectures [37]. As more OSH components become available, the open-source community needs CAD methodologies that support turning these components into a variety of SoC designs that are specialized for target domain applications. Together with methodologies that simplify the logic design and system-level integration of OSH components, it is necessary to develop agile methodologies for the physical design of these SoCs. Key properties of these methodologies are flexibility, robustness, and scalability.

A methodology is *flexible* when it supports the smooth integration of a heterogeneous set of OSH components from distinct development sources and can adapt to use multiple projects' target technologies and preferred EDA tools while meeting performance, power, and area (PPA) requirements. Large and complex SoCs, for instance, often require multiple power and clock domains and their design requires high-end tools and advanced technologies to meet performance goals while keeping power dissipation under control. In contrast, smaller or simpler SoCs may have more relaxed performance and power constraints that can be met with mature technologies and use of open-source EDA tools [31].

A physical design methodology is *robust* when it not only achieves the target quality-of-results (QoR) metrics but also verifies functional correctness at each step from RTL specification to the GDS

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
*ICCAD '22, October 30-November 3, 2022, San Diego, CA, USA*
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9217-4/22/10.
https://doi.org/10.1145/3508352.3561102

† These authors have equal contributions.

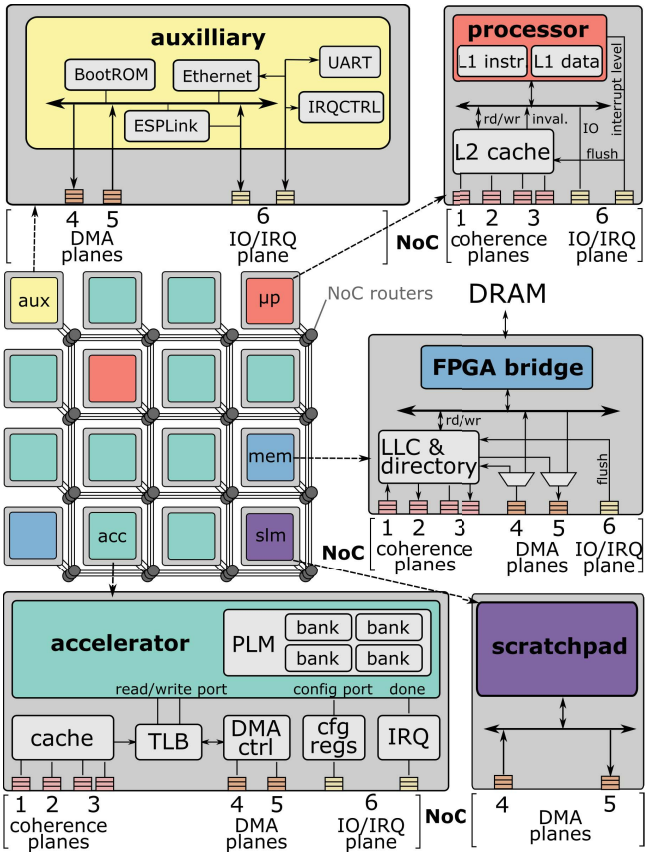


**Figure 2: The ESP architecture and its five main types of tiles.**

implementation. While EDA vendors provide their own recommended flows, a variety of problems can still arise, including, but not limited to, logical or physical design issues, poor or inaccurate settings of the CAD tools, and inconsistencies with the technology models. The more a design flow can avoid these mistakes and promptly detect issues, the more robust is the methodology in its support of design reuse.

A methodology is *scalable* if it can handle the growth in size and complexity of SoC designs with a sublinear growth in terms of computation infrastructure, engineering effort, and design time. As the SoC complexity scales, more data must be processed by the EDA tools' algorithms and the machines running these tools need more cores and memory to reach the final GDS implementation efficiently. Without a scalable design methodology, both computing power and design effort can grow exponentially, thus causing the design time to skyrocket.

This paper presents a flexible, robust and scalable methodology for the agile physical design of heterogeneous SoC architectures. As shown in Fig. 1, the methodology builds on the ESP platform for SoC design and programming [28] by augmenting its original capabilities for FPGA-based SoC prototyping with support for chip development up to the final tapeout of the SoC implementation. As discussed in Section 2, the ESP platform combines a tile-based architecture and a system-level design methodology for the integration of OSH components and the derivation of the RTL implementation of a complete SoC instance. Thanks to the contributions described in this paper, a GDS implementation of each tile of this SoC instance can now be individually obtained from the RTL implementation by using the tile-based physical-design flow presented in Section 4. This flow relies on the enhancements to the ESP architecture described in Section 3. The top-level SoC integration, described in Section 5, uses the tile's Interface Logic Model (ILM) and Layout Exchange Format (LEF) views for top-level placement and timing closure. Top-level Design Rule Check (DRC) and Layout Versus Schematic (LVS) confirm the chip is ready for manufacturing. These steps can be performed in a more agile way with the verification and testing framework described in Sections 6 and 7, respectively. Our methodology was used to complete the chip tapeouts for two SoC architectures of growing complexity, as detailed in Section 8.

## 2 BACKGROUND ON ESP

ESP is a platform for heterogeneous SoC design and programming [28]. Developed over the course of more than a decade of research and teaching at Columbia University [6, 7], ESP combines a scalable architecture and a flexible system-level design methodology. As shown in Fig. 2, the ESP architecture is structured as a heterogeneous tile grid build upon a 2D mesh, multi-plane network-on-chip (NoC) [38]. Each type of tile serves a different purpose in the SoC, but every tile is encapsulated in a modular *socket*. The socket decouples the design of the tile from the NoC following the principles of communication-based system-level design [5]. It also provides several platform services to the intellectual property (IP) block of each given tile (e.g. dynamic voltage frequency scaling, performance counters, coherence, DMA, etc.). The ESP architecture hence strikes a balance between *regularity* and *specialization*. Currently, there are five main types of tiles in the ESP architecture.

The **Processor Tile** contains a CPU developed by a third-party vendor, instantiated off-the-shelf with its own private L1 cache. The processor options include the 32-bit SPARC Leon3 core [15], the 32-bit RISC-V Ibex core [26], and the 64-bit RISC-V CVA6 (formerly known as Ariane) core [39]. In the case of the Leon3 and CVA6 cores, the processor tile can instantiate the ESP L2 cache, which allows the core to transparently participate in the ESP coherence protocol, supporting multicore execution and booting Linux SMP [19, 41]. The development of the methodology described in this paper was driven by the design of chips that use the CVA6 core, but it could be easily extended to support also the other cores.

The **Memory Tile** provides a channel to external memory. ESP allows for the seamless instantiation of multiple memory tiles to satisfy the bandwidth requirements of large SoC designs; in such case, each memory tile serves a discrete partition of the global address space and the logic to route requests to the appropriate tile is automatically generated. When the ESP cache hierarchy is enabled, the memory tile contains the ESP last-level cache (LLC). The LLC, together with the L2, implements a standard directory-based MESI coherence protocol, adapted to work over a NoC [19]. The ESP LLC additionally can handle DMA requests directly from accelerators in an LLC-coherent manner, as proposed in [12].

The **Auxiliary Tile** hosts the non-memory I/O interfaces of the SoC, such as Ethernet and UART, as well as miscellaneous

components, such as the bootROM and interrupt controller. The Ethernet connection supports remote connection through SSH and enables the *ESPLink* debug application, which is an important part of the testing framework described in Section 7.

The **Shared Local Memory (SLM) Tile** can be instantiated to add to the on-chip memory capacity of SoCs. It provides a software-managed scratchpad that lies outside of the coherent address space and can be shared by multiple CPUs and accelerators that access it via DMA. The SLM tile is especially important in ASIC prototypes that may not contain a DDR controller and thus pay an increased penalty for accessing external memory.

As ESP embraces heterogeneity in SoC design, the **Accelerator Tile** is a key component of the architecture. In ESP, accelerators are given equal importance as processors in the SoC and hence occupy their own tile. They are *loosely-coupled* accelerators [12], executing coarse-grained tasks when invoked by a processor core through a device driver. When communicating with the memory hierarchy an accelerator utilizes one of several *coherence modes* – ranging from bypassing the cache hierarchy entirely with DMA to participating in the system's coherence protocol when equipped with a private L2 cache – that can be selected at runtime based on the workload characteristics and dynamic status of the system [18, 20, 40]. ESP provides several design flows for new hardware accelerators: at the RTL level; with C, C++, or SystemC with a high-level synthesis (HLS) tool; or directly from high-level machine learning models using the open source HLS4ML tool [16, 33]. When utilizing these flows, the socket of an accelerator tile provides platform services for address translation, DMA, configuration registers, and coherence. Hence, designers can focus on the optimization of their accelerators without having to "reinvent the wheel" with respect to these capabilities. ESP also provides a flow to integrate pre-designed third-party accelerators (e.g., the NVIDIA NVDLA [30]), so long as they comply with a standard interface, such as AXI [17].

The **Network-on-Chip** comprises six physical planes [38] and uses dimension-order look-ahead routing. Three planes are dedicated to coherence messages, two to DMA, and the last to access memory-mapped registers and interrupts. The NoC is synchronous and achieves single-cycle latency between adjacent routers. Each tile has a set of *proxies* that convert bus requests to NoC messages and vice versa. Thanks to the proxies, components in different tiles can exchange messages as if they were connected to the same bus.

## 3 ENHANCEMENTS TO ESP FOR ASIC DESIGN

For most of its history, the ESP project has focused on system-level design up to RTL implementations and SoC prototyping with FPGAs. Thanks also to the contributions described in this paper, ESP can now be used to realize ASIC implementations of SoCs. The physical-design flow for agile chip development presented in the next sections is supported by many enhancements to the ESP architecture. This section summarizes the main ones.

**Hierarchy Restructuring.** Previously, the NoC in ESP existed as its own entity – instantiated along with all of the tiles in the top level of the design. This design choice can pose problems for the top-level design and integration of a chip, as it can be difficult for EDA tools to achieve good QoR placing and routing of the NoC around the tiles. For this reason, we restructured the ESP RTL hierarchy, such that each tile now has a *wrapper* that instantiates the ESP socket, the encapsulated IP, and a single NoC router for each plane; this combination will henceforth be referred to as a *tile*. This approach better supports design partitioning (Section 4) and simplifies the top-level chip integration to routing connections between tiles (Section 5).

**Control and Status Registers.** Since our chip implementations require a greater degree of configuration at run time, we created a new set of *Control and Status Registers (CSRs)* in each tile. Each set of CSRs is an APB subordinate that can be accessed either through ESPLink or from software running on the SoC. CSRs provide the configuration for components, such as a digitally controlled oscillator (DCO) and pads. Another important CSR is the tile ID register. The tile ID identifies the tile's position in the SoC to logic within the tile. This is needed to form NoC packets and correctly route packets that pass through the tile. Previously, the tile ID was a parameter passed to the RTL of the tile at design time. However, in order to ensure that each type of tile only needs to be implemented once, we changed the tile ID to be in a CSR that gets set at runtime. At reset, a simple component in the auxiliary tile sends the tile ID over the NoC to each tile in a specific sequence that guarantees the routability of all of these messages. The CSRs also include an extensive set of performance counters that monitor events such as NoC injections, coherence messages, cache hits and misses, external memory accesses, and more.

**Local Clock Generation.** Most ESP FPGA-based prototypes of SoCs utilize two primary clock sources: one for the NoC and memory tiles and another for processors, accelerators, and the auxiliary tile. Depending on the FPGA, the clocks can come from DDR controller IPs included in the design or from clocks provided by the FPGA's board. It is possible to instantiate PLLs to provide an independent clock source to each tile [27]. However, depending on the number of tiles selected to use this feature, the available clocking resources of the FPGA can either be prohibitive or negatively impact the QoR of the system. Since ASIC implementations may desire independent voltage and frequency control for each tile and the NoC, we added the support for a DCO in each tile and a NoC DCO in the auxiliary tile. The DCO is designed for the target technology and is configured with a dedicated CSR. Reset-generation logic is added to each tile with a DCO, so that reset is only deasserted after the DCO has been running for a certain number of cycles. This per-tile clocking scheme enables a *communication-synchronous, globally asynchronous locally synchronous (CS-GALS)* approach, where all tiles run at an independent frequency and communicate through a synchronous NoC [24]. This strategy is key to the physical design flow, as detailed in Sections 4 and 5.

**Memory Integration.** ESP includes a variety of on-chip memories for SoC designs [28]: 1) caches (L1, L2, and LLC); 2) shared local memories 3) accelerators' private local memories; 4) BootROM in the Auxiliary tile; and 5) dual-port register files in the Ethernet MAC. These memories, which are usually mapped to BRAMs in FPGA-based prototypes, had to be remapped to technology-specific SRAMs.

**FPGA-link for DRAM Access.** Due to the limited availability of open-source DDR controllers and the difficulty of including one in an academic tapeout, we chose to utilize an FPGA link as the primary source of external memory access. In each memory tile, the

DDR controller that exists in FPGA-based ESP designs is replaced by a bridge from the SoC to an FPGA board, which hosts DDR controllers and the external DRAM. The FPGA bridge follows a simple credit-based protocol, with the FPGA host providing a clock to synchronize communication with the ESP chip. Messages are transmitted in the form of ESP NoC packets, allowing the reuse of components from the SoC in the design of the FPGA host.

**JTAG for Single-Tile Test.** In the context of chip manufacturing, ESP demands a robust pre- and post-silicon tile-based test strategy, decoupled from the system interconnect. The need for a NoC-independent *Test Access Mechanism (TAM)* of the SoC tiles is tightly related to the complex multi-plane NoC architecture of ESP: in case of logical bugs or physical design-based artifacts affecting the functionality of the NoC, an alternative way to communicate with the tiles is critical for diagnosing issues or independently testing the contents of the tile. Hence, we designed a *JTAG-based Debug Unit* to provide a new platform service, integrated in the ESP tile's socket, which enables a direct access to the tile leveraging the standard's four dedicated pins: TDI, TDO, TCLK, and TMS [1]. The TMS is asserted to connect the tile to the JTAG module in test mode and is deasserted to restore the standard connection to the NoC router in normal mode. The JTAG module is designed to bypass the NoC and, at the same time, mimic its latency-insensitive protocol [5, 8] while communicating to the tile. No additional modifications to the tile's internal logic are required to support the test mode.

## 4 TILE-BASED PHYSICAL DESIGN FLOW

A physical design flow has well defined steps (logic synthesis, floorplan, placement, routing, DRC, and LVS) that remain similar across the different offerings of EDA vendors. This section explains how our methodology provides productivity gains to a critical subset of these steps through flexibility, robustness, and scalability.

We start by adopting design partitioning, i.e., dividing the top-level design into smaller components. Design partitioning has several advantages in physical design compared to a full design implementation, including: the ability to parallelize the implementation by distributing the partitions to less powerful machines (scalability), a shorter time to execute the flow, and an easier way to detect and solve issues (robustness). The main drawback of design partitioning is its negative impact on the difficult task of achieving timing closure of the top-level synchronous interfaces. Since ESP tiles share the same tile-to-tile synchronous NoC router interface, the interface delays from one tile to another can be made very similar by using the right timing and floorplan constraints. As Section 5 discusses, this approach enables easier timing closure of a synchronous NoC and avoids the need to adopt an asynchronous NoC [25], which might incur a higher latency penalty.

**Timing Constraints.** Independent of the total number of tiles in ESP-based SoC, there are only four types of constraint files in the flow: `AUX`, `MEM`, `SLM/CPU/Acc`, and the `Top-Level`. The `AUX` and `MEM` files must include external interface constraints that are design dependent because their external delay values must be set according to the environment (e.g. packaging and board) to which the chip will be exposed. The `Top-Level` constraint file shares the external delays set into the `AUX` and `MEM` files with a minor discount to account for the delay from the tile to the IO pad. The remaining external

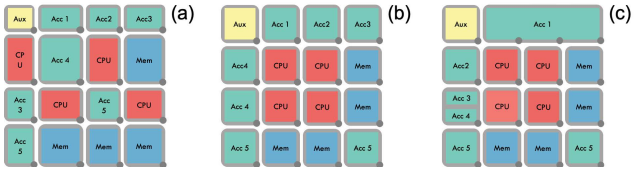


**Figure 3: Three approaches to top-level floorplanning.**

interface constraints can be set according to protocol's standards (e.g. Ethernet, UART), with the exception of the clock frequency for the FPGA bridge in the Memory tile, which varies according to the test environment. Since the tiles use the same clock strategy and NoC routers, the setting of the clock relationships and all tile-to-tile external delays is design independent. There are only two design dependent values common to each constraint file type: the NoC clock frequency and the individual IP clock frequency. All technology-dependent constraints (e.g. driving cells, clock transition, clock buffers, etc.) must be set, but do not change across the top level and all tiles. In the constraint files, these constraints are specified with variables that are the mapped to the technology-specific parameters in another file. These settings are reusable across different designs that target the same technology process.

An architecture that allows for pre-defined constraints significantly reduces the chances of mistakes in defining timing constraints such as false paths, interface clock relationships, and incomplete constraints (thus enhancing the robustness of the design flow). Moreover, as the design scales, the only values that must be specified are the IPs' clock frequencies (highlighting scalability). The small number of constraints that must be changed for a new design or technology demonstrates the flexibility of the flow.

**Power Constraints.** All tiles can support *Multi-Supply Voltage (MSV)* and *Power Shut-Off (PSO)*. The use of a CS-GALS design naturally defines the boundaries of the different power domains inside the tiles. The power domains follow the clock domains: the IP operates on the adjustable $V_{ip}$, while the NoC operates on a global $V_{noc}$. As a result, a configurable power constraint file [2] can be used across all tiles. If, for example, tile A has MSV and tile B has PSO capabilities, a simple parameter in this file is enough to make it reusable, thus promoting scalability and robustness.

**Top-Level Floorplan.** Once all tiles are mapped to the target technology, the top-level floorplan and the shape of the tiles can be defined based on their sizes. There are several trade-offs in play in this step. One option is to utilize a grid structure and allow different heights and widths for rows and columns (Fig 3 (a)), thus allowing for flexibility in tile sizes. While this approach can optimize area, more irregular dimensions make the tiles less flexible. For example, if the same tile is instantiated four times in the design and its locations have different shapes, four physical implementations are required (and four respins in case of some issue being detected). We avoided this approach because it reduces the productivity gains of the proposed methodology.

In contrast, a simple approach is to define the same shape for all tiles (Fig 3 (b)). This approach facilitates top-level timing closure because every route between tiles has a similar length. Further, it allows for easily swapping the position of any tiles in the SoC and even substituting tiles in case a last-minute change is needed before

tapeout. A uniform tile size also best promotes reusability, since the same GDS can be replicated multiple times for the same tile (i.e., only one processor tile implementation is needed for an SoC with four processor cores). Similarly, if the verification team detects an issue in a tile that is instantiated multiple times, only one respin is required to fix all of those tiles. This regularity, however, forces all tile sizes to match the size of the tile hosting the largest SoC component, resulting in wasted area and degraded NoC frequency, especially if many tiles can be smaller. To prevent this waste, smaller SoC components can share the same tile and bigger ones can use a cluster of multiple tiles (Fig. 3 (c)), thus keeping the advantages of a uniform floorplan. However, this approach may bring some contention in accessing the NoC router among components that share a tile and reduced flexibility of the overall floorplan due to positioning of multi-tile clusters.

Finally, if the use of relay stations [5] is not an option, then the supportable NoC frequency depends on the longest tile's edge. Hence, the tile's shape must be as close to a square as possible, particularly for high performance SoCs.

**Pin Assignment.** To reduce wire length at the top level of the chip, all tiles' pins are placed in the same position with respect to the corners of the tile: north ports, south ports, west ports, and east ports are in the top-right, bottom-right, bottom-left, and bottom right corners, respectively (Fig. 4 (a)). These locations are not design-dependent and can be reused in other designs.

**Macro Placement.** Despite great progress with the introduction of machine-learning techniques [29], macro placement remains among the hardest steps of physical design to automate. In ESP, there are two macro-cells types that require manual placement: register files and SRAMs. Even though the DCO uses standard cells, it is essential to constrain its height and width and use the same dimensions in all tiles to achieve similar frequency characteristics; the DCO location and routing, however, can vary from tile to tile.

**Power Strategy.** Due to the need for connections of the NoC on all 4 sides (N/W/S/E) of a tile, the IP and NoC power domains have square and inverted-L shapes, respectively (Fig. 4 (b)), which is reusable across all tiles. Its width depends on the density of the NoC-domain logic and the routing congestion to the output pins.

SRAM power supply can easily be partitioned into core voltage, supplying the bitcells, and periphery voltage, supplying the surrounding logic. The sense amplifiers and line drivers naturally provide a level-shifter between the two domains. Typically, the SRAM minimum voltage is limited by the array rather than by periphery logic. Hence, we add a third voltage $V_{mem}$ for the SRAM array, while the SRAM periphery is connected to $V_{ip}$. In this way, no explicit level shifters are needed between the SRAM and IP logic.

For technologies that have several metal layers, the power stripes are distributed as a grid in most layers. The top level uses the highest layer to connect all tiles to their corresponding power domain. Section 5 elaborates our top-level striping methodology. At the tile level, higher layer levels form a grid including $V_{ip}$, $V_{noc}$, $V_{mem}$, and $V_{ss}$ to maximize the power integrity and pin area for connection to the chip top level. Medium layers' stripes are split between the IP and NoC domains. While $V_{ip}$, $V_{mem}$, and $V_{ss}$ form the power grid in the IP domain, $V_{ip}$, $V_{noc}$, and $V_{ss}$ form the power grid over the NoC domain – $V_{ip}$ is needed to supply the level shifters. We consider the lowest medium layer as the lowest layer that supplies memory

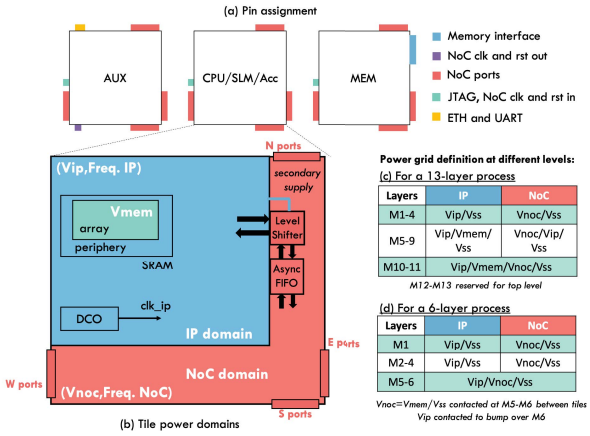


**Figure 4: Breakdown of power domains and pin assignment.**

macros. Finally, lower layers form a two-stripe grid, $V_{noc}/V_{ss}$ and $V_{ip}/V_{ss}$, to supply the standard cells. In these levels, the $V_{mem}$ grid is no longer needed to supply memories, and level shifters can automatically connect to the $V_{ip}$ of medium layers, thus freeing routing tracks for logic signals. Fig. 4 (c) exemplifies this strategy for the 13-metal-layer process technology we used in both chips discussed in Section 8.

Similar power design can be adapted to technology processes with fewer metal layers. For example, Fig. 4 (d) shows how a 6-layer grid could be built: a shared $V_{noc}/V_{ss}$ grid is made with metals M5 and M6 to connect to the other tiles, while M6 $V_{ip}$ is connected to a bump directly over the tile to minimize routing congestion for the top-level power. The grids between M1 and M5 are optional.

## 5 TOP-LEVEL PHYSICAL DESIGN FLOW

The inherent regularity of the ESP tile-based architecture forms a unified array in the top-level floorplan. The locations of the AUX and MEM tiles, the only tiles with external interfaces, should be set close to their respective IO pads. All other tiles can freely change locations at any point in the design cycle. We adopted a power-domain array package and a hierarchical timing-closure flow that match the flexibility of the ESP architecture and support agile design-respin cycles.

Package design affects the top-level power strategy. In our methodology we focused on flip-chip packages. Compared to conventional wire-bond packages, flip-chip packages scale better to the needs of larger SoC designs, which have a large number of required I/Os and power supplies. A similar power strategy can still be used, however, for smaller SoCs that utilize wire-bond packages.

**SoC Power Allocation.** Fig. 5 (a) exemplifies the flexible power domain allocation strategy by using a symmetric power domain array. Each power domain consists of at least 20 power bumps to sustain sufficient workload current. During the implementation, the power domain array is flexibly grouped into a few power domain clusters based on the SoC floorplan. For example, for SoC Case 1, the power domains are grouped based on the tile functionality and physical location to support flexible power sequencing while maintaining low IR drop. In this case, there are 16 $V_{dd}/V_{ss}$ arrays plus a global $V_{dd}/V_{ss}$ supplying the NoC. As the design scales, more power domain clusters are allocated to provide even power

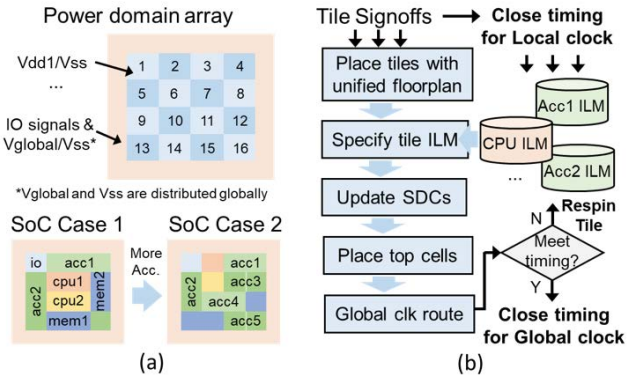


**Figure 5: (a) Symmetric power domain array for flexible power cluster allocation, (b) Hierarchical timing closure flow.**

distribution and testing flexibility, as shown Fig. 5(a) for SoC Case 2. During the physical design, the power grids of all power domains are distributed across the chip to decouple the SoC floorplan with the power striping step. The IR drop to each tile is carefully analyzed to guarantee a balanced power distribution.

**SoC Timing Closure.** Achieving timing closure for a large SoC design is a hard task. We adopted a hierarchical timing-closure flow for independent timing signoff between the local clock frequency of each tile and the global NoC clock frequency, as shown in Fig. 5(b). With our CS-GALS clocking strategy, each tile has its own clock domain and connects to the global synchronous NoC clock via an asynchronous interface. This allows the physical design of all tiles to be conducted in parallel, while the global timing is closed later based on the interface logic model (ILM) timing models. During the top-level global clock timing closure, only the clock skews between neighboring tiles need to be constrained, which significantly relaxes traditional timing closure requirements. Such a timing closure flow also supports flexible reuse and respin of pre-existing IPs, further trimming design time.

## 6 VERIFICATION

In an agile chip design framework, verification can prove challenging due to small team sizes and short design cycles. To make verification of complex, heterogeneous SoCs tractable given personnel and time constraints, we adopt a *system-level verification strategy* that focuses on testing the *integration* of all new components and complete system functionality. This strategy is based on two main assumptions. First, we assume that any OSH components taken from third-parties and any new in-house designed IPs (i.e. a new hardware accelerator) to be included in a tapeout are already *thoroughly verified* with unit-level testbenches by their designers. Second, we leverage the fact that ESP's NoC, buses, sockets, and platform services are *pre-verified* and do not require additional scrutiny from the verification team (unless the current design modifies some of these components). Our strategy utilizes full-system RTL simulation of bare-metal applications, netlist-level simulation with simplified bare-metal applications, and FPGA emulation to run longer tests that would be intractable in simulation. We also leverage logical equivalence checking (LEC) at multiple stages of the physical design process for verification of the implementation.

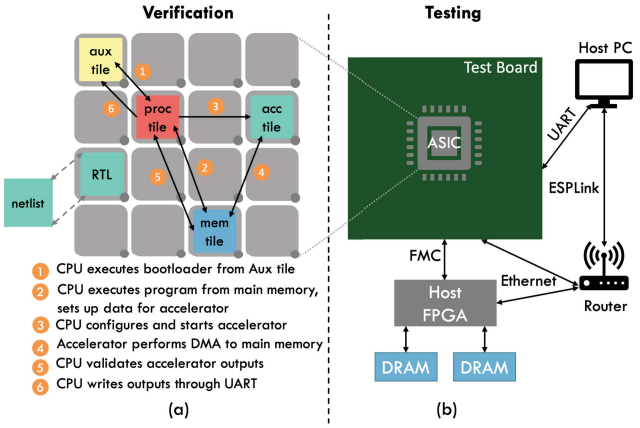


**Figure 6: (a) Example of a full-system bare-metal simulation for verification, (b) The chip-testing environment.**

Each full-system **RTL Simulation** test consists of a bare-metal application, written in C, which will run on one of the SoC's CPUs to stimulate a component of interest. As shown in Fig. 6 (a), to test a hardware accelerator, the C application will have the CPU prepare a dataset for the accelerator to operate on, configure and start the accelerator, wait for the accelerator to complete, and then validate the results compared to the golden values, computed from software. Since the accelerator is already verified by its designer, the test is designed to verify the aspects of the accelerator that are key to its integration, such as configuration registers, DMA, and interrupts.

The C application is cross-compiled for the chosen cores' ISA and is loaded into a simulation model of external memory. After reset, the CPU executes ESP's first-stage bootloader and then jumps to the start address of the program in memory. In addition to testing the target component, each program also tests the CPUs, memory interfaces, I/O, and NoC, providing further in-context verification of these critical components of the SoC. The software-level testing approach greatly lowers the effort required from the verification team to write new tests. Furthermore, all accelerator tests – which constitute a majority of the tests in the chips we taped out – follow the same template, and only the input data, configuration parameters, and golden outputs must change. The development of these exact tests serves multiple purposes, as they can also be used to test these IPs in the fabricated silicon.

**Netlist-Level Simulation** is used to verify the synthesized netlist for each tile of the SoC. In order to keep simulation times reasonable, we perform this simulation for one target tile at a time. Thanks to the hierarchical physical design flow, it is easy to instantiate the synthesized netlist for one tile in the top-level design, while keeping the other tiles as RTL. For each tile, we conduct a similar test as for RTL simulation, but modify the test to be as minimal as possible (using small datasets, removing `printf`s, hard-coding parameters that are normally discovered, etc.) to reduce simulation times. We also conduct one small test of the reset sequence and check each tile's ID CSR using the entire SoC's netlist.

**FPGA Emulation** complements RTL and netlist-level simulation by allowing the execution of long complex workloads that would not be feasible in simulation. Thanks to ESP's rapid FPGA-based design flow, we can quickly generate prototypes of the target chip for one or more FPGAs. Once the designs are generated, we

can boot Linux and run applications with large datasets that invoke accelerators. These tests give confidence in the stability and robustness of the target RTL. While being very efficient, FPGA emulation cannot fully replace RTL and netlist-level simulations due to the fundamental differences between the RTL of the target ASIC and that of the emulated FPGA design: (a) the emulated FPGA design contains DDR controllers, while the ASIC contains a custom bridge to an FPGA that provides DRAM access; (b) IPs that are generated with HLS will have vastly different RTL for different technologies; (c) FPGA designs use an external clock globally, whereas ASIC designs can instantiate internal clock generators for the NoC and each tile; and (d) the ASIC design relies on technology-specific IPs, such as memories, DCOs, and pads. Furthermore, verifying the entire SoC design by emulating it on a single FPGA device is often impossible due to space limitations. Multiple FPGA prototypes that each contain a subset of the chip design may be needed to achieve full coverage in this branch of the verification plan.

**Equivalence Checking** executes in parallel to functional verification. It formally confirms whether the synthesis' netlist output matches the RTL. A similar comparison assures the logic equivalence of the netlist after place and route. Because gate-level simulations are abbreviated, these checks raise our confidence in the implementation. For the tiles with multi-supply voltage, formal checking is executed between the power constraint file and the netlist to assure that the low-power cells were properly included.

## 7 TESTING

Our ASIC design flow leverages the testing infrastructure developed over many years of FPGA prototyping with ESP, along with some additional new features to improve testability, which enables rapid bring-up and post-silicon validation. There are a few requirements for the test board that mounts the packaged SoC, as shown in Fig. 6 (b). The primary debug interface for ESP uses Ethernet, which is enabled with the instantiation of a MAC and EDCL debug unit from Cobham Gaisler in the auxiliary tile [10, 23]. A UART connection to a host PC is also required for serial output from the SoC. Finally, a connection to an FPGA – we use an FPGA Mezzanine Card (FMC) connector – provides access to external DRAM for SoCs without a DDR controller. Our testing flow is described as follows.

**1. Power Up.** Upon initial power-up of the chip, we test the full system for any major electrical issues (e.g. shorts) and check the output from the available DCOs with SMA connections on the test board – another recommended feature for board design. Once the DCOs are validated, we proceed with trying to establish an Ethernet connection to the SoC. In our test setup, we connect the test board to a router on the local network, allowing remote testing from any PC. Once an Ethernet connection is established, we can use the *ESPLink* debug application. ESPLink is an application that runs on a host PC that utilizes the Ethernet connection to access any memory-mapped region of the SoC. ESPLink has a simple command line interface and can be used to read/write individual registers or dump/load contiguous regions of memory to/from a binary or hex file. With ESPlink, for example, the Ethernet connection can be verified by reading a register with a known value (e.g. the NoC DCO CSR) in the auxiliary tile.

**2. NoC Status and Memory Access.** With an Ethernet connection established, we can begin to test the operation of the NoC by trying to read registers in each tile from ESPLink. By accessing the tile ID register in each tile, we verify that we can read from all tiles, and if the value is correct, it means the tile IDs were successfully sent from the auxiliary tile at reset. Success in this test gives strong indication of the functionality of the NoC Plane 6. We can then try to write to and read from external DRAM through the link to the FPGA device, also stimulated by ESPLink.

**3. Running Bare-Metal Programs.** Once memory access is established, we try to run some bare-metal programs on the processor tiles. To do this, the compiled bootloader is first written to the bootloader memory in the auxiliary tile with ESPLink. Then, the compiled program is written to external DRAM through the FPGA host, also utilizing ESPLink. Finally, ESPLink sends the soft reset to the SoC, which triggers the processor cores to start executing. They initially execute the bootloader from the RAM in the auxiliary tile and then jump to the address of the program in external DRAM. A simple *"Hello World"* program can be used to further validate the external memory access, test NoC Planes 1-3, and check functionality of the UART interface. Following this, we can run bare-metal tests that invoke accelerators to validate the remaining IPs and check NoC Planes 4 and 5. As mentioned before, the same bare-metal programs run during verification can be reused during bring-up, thus eliminating the need to write new programs at this stage.

**4. Boot Linux and Data Collection.** At this point, the SoC is ready to boot Linux. For RISC-V, the device tree (which is included with the bootloader binary) must be configured with the correct frequencies for the processors and UART connection. If there are issues with any of the memory tiles, then the device tree must also be set to only use addresses within memory ranges that have been validated successfully. We have found it to be faster and easier to collect data while running an operating system. Creating V/F curves can require many runs of the same program, continuously increasing frequency of the local clock until failure. With Linux, this can be done without resetting the chip each time.

**Backup: JTAG for Single-Tile Test.** The JTAG logic bypasses the connection between the tile and the NoC and packets are serialized/deserialized over two pins connected to the FPGA host. The FPGA host can then stimulate the tile with NoC packets, and the tile reacts as if it was connected to the NoC. Upon receiving the response from the tile, the FPGA host can check the outputs against the expected NoC packets to validate the execution. This is useful for debugging and communicating directly with the tile in case of manufacturing issues in the NoC.

## 8 EXPERIENCE WITH TWO CHIP DESIGNS

The proposed methodology was first employed in the development of a 4.65x4.65mm chip with a 12nm technology. With a 4x4-tile ESP architecture, the SoC targets the swarm-based perception domain [24]. In the span of 4 months, a team of 10 engineers installed and configured the technology, set the EDA tools' scripts, enhanced the ESP platform to support ASIC development, integrated the OSH components, developed applications utilizing the accelerators, and performed the physical design flow. Fig 7(a) shows the breakdown of the design cycle of this chip. The most time-consuming part of

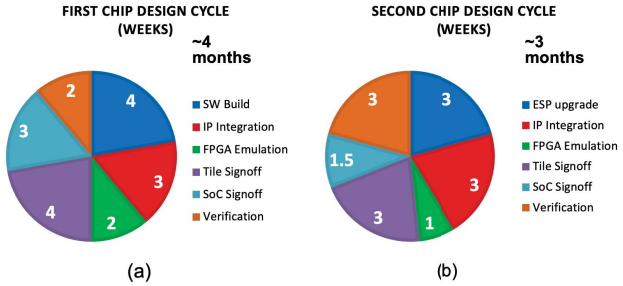


**Figure 7: Design time breakdown.**

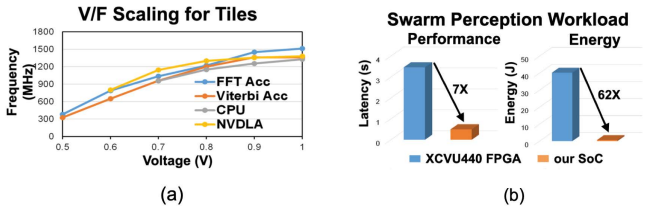


**Figure 8: Performance of the first chip [24].**

the tile signoff step was the implementation of the first tile. Once we achieved a physical design flow without human intervention that met the constraints and resulted in less than 20 DRC violations in the layout signoff tool, repeating this step on all other tiles was mostly a matter of machine runtime, showing the robustness and reusability of the methodology. The physical implementation of each tile, from RTL to GDS, took 12 hours on a 16-core, 64GB RAM machine. The SoC integration and final physical verification took 51 hours on a 64-core, 376GB RAM machine.

This first chip has 16 tiles (4 CPUs, 4 MEMs, 3 Fast Fourier Transform (FFT) accelerators, 3 copies of the NVIDIA NVDLA [30], 1 AUX, and 1 Viterbi Decoding accelerator) and multi-supply voltage in all accelerators. The NoC reaches 800 MHz at 0.8V and the tiles can reach up to 1.5GHz as the voltage/frequency curves in Fig. 8(a) show. Fig. 8(b) shows the chip's performance for the application compared to an FPGA implementation of the same SoC. To keep the methodology simple in this first proof of concept, we use the same size for all tiles, which results in 15% extra area overhead.

With the design of a second chip, we scaled from 16 to 36 tiles, resulting in a 8x8mm area in the same target technology. New features were added to the ESP architecture such as new CSRs in the NoC domain to support power shut-off and integration of a power management unit. Seven new accelerators were added to the previous three. The physical design flow was adjusted to accommodate more components in the NoC domain. To save area, one cluster of four tiles was used for a larger accelerator and three smaller accelerators were combined into a single tile. The machine runtime for the implementation of the tiles did not change. The SoC integration, however, took 66 hours to complete, 29% more than the first chip. Fig 7(b) shows the design cycle's breakdown.

Our methodology was tested on many fronts during the design cycle. The most critical was a bug identified by the verification team five days before tape-out that affected eleven tiles. Thanks to the tiles' regularity and a robust autonomous flow, we were able to fix the bug in all the tiles (only seven respins were needed due to tile duplications) and integrate them in the SoC before the deadline. Our testing methodology allowed for a complete bring-up and analysis of the first chip in three weeks, including booting Linux and running complex applications that invoke accelerators.

## 9 RELATED WORK

In recent years, several methodologies have been proposed to tackle the complexity of SoC design. Khailany et al. presented a modular physical design methodology for high productivity SoC design that combines some architectural decisions such as asynchronous NoC, GALS, and latency-insensitive channels with physical design partitioning [25]. OpenROAD leverages the OpenLane tool-set [31] to offer a full open-source physical design flow by providing scripts and documentation for chip design. Chipyard is an IP library for agile SoC design based on the Chisel hardware description language for SoC integration [3]. Chipyard offers the Hammer [36] physical design flow to guide non-experts with a set of APIs that hide the complexity of EDA tools and their interactions with the technology. Hammer supports multiple EDA vendors and technologies.

## 10 CONCLUDING REMARKS

We presented an agile chip-design methodology that builds on top of the open-source ESP platform to deliver gains in terms of flexibility, robustness, scalability and, ultimately, design productivity for complex heterogeneous SoCs. To develop our methodology, we made various enhancements to the ESP tile-based architecture (e.g., support for a CS-GALS clocking scheme) and made strategic decisions about the physical design (e.g., defining a robust power plan). Concrete benefits of our methodology include smooth integration of SoC components, fast respin, reduction of design time through tile reuse and parallel machine distribution, usage of affordable machines for most stages of the flow, comprehensive integration verification, and an adaptable flow across multiple designs and technologies. The methodology was used to successfully design and test first a heterogeneous SoC with 16 tiles and then a more complex SoC with 36 tiles. The two chips were designed in essentially the same time by the same team of 10 engineers working remotely.

Our methodology is under continuous development, with many new features under consideration. The support of an open-source tool set and PDK, such as OpenLane [31] and Skywater [34], respectively, would enable an open-source release of the methodology to the community. An automatic top and tile-level floorplan would improve productivity in one of the most time-consuming tasks in physical design. Improvements in design for testing (DFT) would increase post-silicon observability and controllability, while keeping the pin count low as the design size and complexity scales.

**Acknowledgments.** This research was developed, in part, with funding from the Defense Advanced Research Projects Agency (DARPA), and in part with funding from the Army Research Office under Grant Number W911NF-19-1-0476. The views, opinions and/or other findings expressed are those of the authors and should not be interpreted as representing the official views or policies (either expressed or implied) of the Department of Defense, the Army Research Office, or the U.S. Government. Distribution Statement"A": Approved for Public Release, Distribution Unlimited. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

# REFERENCES


[1] 2013. IEEE Standard for Test Access Port and Boundary-Scan Architecture - Redline. *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001) - Redline* (2013), 1–899.

[2] 2019. IEEE Standard for Design and Verification of Low-Power, Energy-Aware Electronic Systems. *IEEE Std 1801-2018* (2019), 1–548. https://doi.org/10.1109/IEEESTD.2019.8686430

[3] Alon Amid, David Biancolin, Abraham Gonzalez, Daniel Grubb, Sagar Karandikar, Harrison Liew, Albert Magyar, Howard Mao, Albert Ou, Nathan Pemberton, Paul Rigge, Colin Schmidt, John Wright, Jerry Zhao, Yakun Sophia Shao, Krste Asanović, and Borivoje Nikolić. 2020. Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs. *IEEE Micro* 40, 4 (2020), 10–21. https://doi.org/10.1109/MM.2020.2996616

[4] Mark Bohr. 2007. A 30 Year Retrospective on Dennard's MOSFET Scaling Paper. *IEEE Solid-State Circuits Society Newsletter* 12, 1 (2007), 11–13. https://doi.org/10.1109/N-SSC.2007.4785534

[5] Luca P. Carloni. 2015. From Latency-Insensitive Design to Communication-Based System-Level Design. *Proc. IEEE* 103, 11 (2015), 2133–2151. https://doi.org/10.1109/JPROC.2015.2480849

[6] Luca P. Carloni. 2016. Invited: The Case for Embedded Scalable Platforms. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1145/2897937.2905018

[7] Luca P. Carloni, Emilio G. Cota, Giuseppe Di Guglielmo, Davide Giri, Jihye Kwon, Paolo Mantovani, Luca Piccolboni, and Michele Petracca. 2019. Teaching Heterogeneous Computing with System-Level Design Methods. In *Proceedings of the Workshop on Computer Architecture Education*. Article 4, 8 pages. https://doi.org/10.1145/3338698.3338893

[8] Luca P. Carloni, Kenneth L. McMillan, and Alberto L. Sangiovanni-Vincentelli. 2001. Theory of latency-insensitive design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 20, 9 (2001), 1059–1076. https://doi.org/10.1109/43.945302

[9] Ralph K. Cavin, Paolo Lugli, and Victor V. Zhirnov. 2012. Science and Engineering Beyond Moore's Law. *Proc. IEEE* 100, Special Centennial Issue (2012), 1720–1749. https://doi.org/10.1109/JPROC.2012.2190155

[10] Cobham Gaisler. [n.d.]. GRLIB IP Library. https://www.gaisler.com/index.php/downloads/leongrlib.

[11] Robert Colwell. 2013. The chip design game at the end of Moore's law. In *2013 IEEE Hot Chips 25 Symposium (HCS)*. 1–16. https://doi.org/10.1109/HOTCHIPS.2013.7478302

[12] Emilio G. Cota, Paolo Mantovani, Giuseppe Di Guglielmo, and Luca P. Carloni. 2015. An Analysis of Accelerator Coupling in Heterogeneous Architectures. In *Proceedings of the Design Automation Conference (DAC)* (San Francisco, California) *(DAC'15)*. ACM, New York, NY, USA, Article 202, 6 pages. https://doi.org/10.1145/2744769.2744794

[13] William J. Dally, Yatish Turakhia, and Song Han. 2020. Domain-specific hardware accelerators. *Commun. ACM* 63, 7 (2020), 48–57. https://doi.org/10.1145/3361682

[14] Robert H. Dennard, Fritz H. Gaensslen, Hwa-Nien Yu, V. Leo Rideout, Ernest Bassous, and Andre R. LeBlanc. 1974. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits* 9, 5 (1974), 256–268. https://doi.org/10.1109/JSSC.1974.1050511

[15] Cobham Gaisler. [n.d.]. LEON3 Processor. www.gaisler.com/index.php/products/processors/leon3.

[16] Davide Giri, Kuan-Lin Chiu, Giuseppe Di Guglielmo, Paolo Mantovani, and Luca P. Carloni. 2020. ESP4ML: Platform-Based Design of Systems-on-Chip for Embedded Machine Learning. *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*.

[17] Davide Giri, Kuan-Lin Chiu, Guy Eichler, Paolo Mantovani, and Luca P. Carloni. 2021. Accelerator Integration for Open-Source SoC Design. *IEEE Micro (Special Issue: FPGAs in Computing)* 41, 4 (2021), 8–14. https://doi.org/10.1109/MM.2021.3073893

[18] Davide Giri, Paolo Mantovani, and Luca P. Carloni. 2018. Accelerators and Coherence: An SoC Perspective. *IEEE Micro (Special Issue: Hardware Acceleration)* 38, 6 (Nov. 2018), 36–45.

[19] Davide Giri, Paolo Mantovani, and Luca P. Carloni. 2018. NoC-Based Support of Heterogeneous Cache-Coherence Models for Accelerators. *Proceedings of the Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*.

[20] Davide Giri, Paolo Mantovani, and Luca P. Carloni. 2019. Runtime Reconfigurable Memory Hierarchy in Embedded Scalable Platforms. *Proceedings of the Asia and South Pacific Design Automation Conference (ASPDAC)*.

[21] Gagan Gupta, Tony Nowatzki, Vinay Gangadhar, and Karthikeyan Sankaralingam. 2017. Kickstarting Semiconductor Innovation with Open Source Hardware. *Computer* 50, 6 (2017), 50–59. https://doi.org/10.1109/MC.2017.162

[22] Mark Horowitz. 2014. 1.1 Computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 10–14. https://doi.org/10.1109/ISSCC.2014.6757323

[23] Marko Isomäki. 2004. *Processor Debugging Through Ethernet.* Master's thesis.

[24] Tianyu Jia, Paolo Mantovani, Maico Cassel dos Santos, Davide Giri, Joseph Zuckerman, Erik Jens Loscalzo, Martin Cochet, Karthik Swaminathan, Gabriele Tombesi, Jeff Jun Zhang, Nandhini Chandramoorthy, John-David Wellman, Kevin Tien, Luca Carloni, Kenneth Shepard, David Brooks, Gu-Yeon Wei, and Pradip Bose. 2022. A 12nm Agile-Designed SoC for Swarm-Based Perception with Heterogeneous IP Blocks, a Reconfigurable Memory Hierarchy, and an 800MHz Multi-Plane NoC. In *European Solid-State Circuits Conference (ESSCIRC)*.

[25] Brucek Khailany, Evgeni Krimer, Rangharajan Venkatesan, Jason Clemons, Joel S. Emer, Matthew Fojtik, Alicia Klinefelter, Michael Pellauer, Nathaniel Pinckney, Yakun Sophia Shao, Shreesha Srinath, Christopher Torng, Sam Likun Xi, Yanqing Zhang, and Brian Zimmer. 2018. INVITED: A Modular Digital VLSI Flow for High-Productivity SoC Design. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1109/DAC.2018.8465897

[26] lowRISC. [n.d.]. Ibex RISC-V Core. https://github.com/lowRISC/ibex.

[27] Paolo Mantovani, Emilio G. Cota, Kevin Tien, Christian Pilato, Giuseppe Di Guglielmo, Ken Shepard, and Luca P. Carloni. 2016. An FPGA-based Infrastructure for Fine-grained DVFS Analysis in High-performance Embedded Systems. In *Proceedings of the Design Automation Conference (DAC)*. 157:1–157:6.

[28] Paolo Mantovani, Davide Giri, Giuseppe Di Guglielmo, Luca Piccolboni, Joseph Zuckerman, Emilio G. Cota, Michele Petracca, Christian Pilato, and Luca P. Carloni. 2020. Agile SoC Development with Open ESP : Invited Paper. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. 1–9.

[29] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter, and Jeff Dean. 2021. A graph placement methodology for fast chip design. *Nature* 594, 7862 (2021), 207–212. https://doi.org/10.1038/s41586-021-03544-w

[30] NVIDIA. 2017. NVIDIA Deep Learning Accelerator (NVDLA). www.nvdla.org.

[31] OpenLANE. [n.d.]. . https://github.com/The-OpenROAD-Project/OpenLane

[32] RISC-V. [n.d.]. . Retrieved July 31, 2022 from https://riscv.org/

[33] hls4ml. [n.d.]. https://fastmachinelearning.org/hls4ml.

[34] SkyWater. [n.d.]. . https://github.com/google/skywater-pdk

[35] Ed Sperling. 2014. *How much will that chip cost?* Retrieved July 31, 2022 from http://semiengineering.com/how-much-will-that-chip-cost/

[36] Edward Wang, Colin Schmidt, Adam Izraelevitz, John Wright, Borivoje Nikolić, Elad Alon, and Jonathan Bachrach. 2020. A Methodology for Reusable Physical Design. In *2020 21st International Symposium on Quality Electronic Design (ISQED)*. 243–249. https://doi.org/10.1109/ISQED48828.2020.9136999

[37] Rich Wawrzyniak. 2021. *Analyzing RISC-V CPU market for SiP, SoCs, AI, and Design Starts.* https://semico.com/content/analyzing-risc-v-cpu-market-sip-socs-ai-and-design-starts/

[38] Young Jin Yoon, Nicola Concer, Michele Petracca, and Luca P. Carloni. 2013. Virtual Channels and Multiple Physical Networks: Two Alternatives to Improve NoC Performance. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 12 (2013), 1906–1919. https://doi.org/10.1109/TCAD.2013.2276399

[39] Florian Zaruba and Luca Benini. 2019. The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology. *IEEE Transactions on Very Large Scale Integration Systems* 27, 11 (2019), 2629–2640.

[40] Joseph Zuckerman, Davide Giri, Jihye Kwon, Paolo Mantovani, and Luca P. Carloni. 2021. Cohmeleon: Learning-Based Orchestration of Accelerator Coherence in Heterogeneous SoCs. In *Proceedings of the IEEE/ACM Symposium on Microarchitecture (MICRO)*.

[41] Joseph Zuckerman, Paolo Mantovani, Davide Giri, and Luca P. Carloni. 2022. Enabling Heterogeneous, Multicore SoC Research with RISC-V and ESP. *Proceedings of the Workshop on Computer Architecture Research with RISC-V (CARRV)*.