OPDB: A Scalable and Modular Design Benchmark

Georgios Tziantzioulis, Ting-Jung Chang, Jonathan Balkind, Jinzheng Tu, Fei Gao, David Wentzlaff

***Abstract*—Progress in Electronic Design Automation (EDA) has enabled us to manage the exponential growth of complexity in integrated circuits (IC) allowed by Moore's Law, and translate it into performance improvements. Furthermore, as Moore's law slows down and with the collapse of Dennard scaling over the past decade, EDA tools become increasingly important in addressing inefficiencies in ICs. As new proposals and techniques are put forward to address the current and future issues of IC design, a concrete set of contemporary benchmarks need to be used for their evaluation. Traditionally, EDA researchers have mainly relied on industry provided design benchmarks in evaluating the performance of their tools. However, due to their effort to maintain their competitive advantage, industry releases include older designs with limited information about the details of each design. This means that in multiple cases new proposals are evaluated with significantly dated designs, often more than a decade old, especially in terms of scale.**

In this work, we describe OPDB: a scalable, modular, heterogeneous, and extensible design benchmark for the EDA community. OPDB leverages and extends the OpenPiton open-source, tile-based research infrastructure to create a surplus of design benchmarks that target different components. Due to the tiled nature of this architecture, OPDB benchmarks can be made arbitrarily large in order to evaluate the efficiency of EDA tools across different design scales and configurations. OPDB contains several accelerators enabling full SoC designs to be used as benchmarks for EDA tools.

***Index Terms*—Electronic Design Automation (EDA), Design Benchmark, Scalable, Modular.**

I. Introduction

Over the past 50+ years, the field of Electronic Design Automation (EDA) has played a key role in the growth we have experienced in computational capabilities [1]. Innovations and research in EDA tools and techniques have enabled designers to manage the exponential increase in complexity of Integrated Circuits (ICs), from the first microprocessor which contained a few thousand transistors, to the latest ICs that can number up to trillions of transistors [1], [2], [3].

Despite the slowdown of Moore's law and the collapse of Dennard scaling, transistor counts in ICs have continued to increase. The increased complexity with each generation keeps pushing the limits of tools and algorithms. The combination of these issues means that EDA tools will become increasingly important in addressing inefficiencies in ICs.

As new algorithms and techniques are introduced to improve existing EDA tools, benchmarks play a crucial role in quantitatively evaluating the proposed techniques. Commercial EDA tools are able to assess their effectiveness and stay relevant through their use in the design of new IC products. In contrast, academic tools have had to rely on industrial partners to provide industry-strength and scale designs. Due to the desire of commercial companies to maintain a competitive advantage against competitors, industrial designs are not released in a timely manner to the public and academia, and though a number of research teams may be able to form collaborations with industry and gain access to newer designs, the results are not usable by others as the reported data cannot be reproduced by other teams [4].

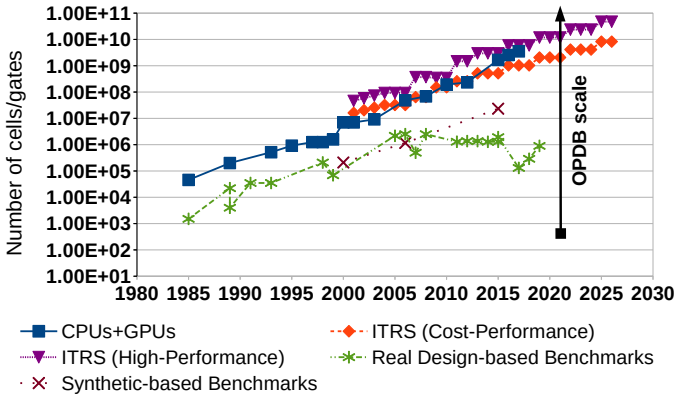


Fig. 1: Timeline of the number of cells/gates for commercial CPUs/GPUs, ITRS projections, and EDA benchmarks from real and synthetic designs. For CPUs/GPUs and ITRS projection the number of cells/gates was computed assuming that each cell/gate was composed of 6 transistors.

As a result, most new academic proposals are evaluated with designs that are significantly dated, especially in terms of scale. This problem has been exacerbated over the years as commercial designs kept scaling, while EDA benchmarks and test cases have struggled to scale commensurately. Fig. 1 presents historical data, between 1985 and 2019, regarding the number of cells/gates for commercial CPUs and GPUs [2], [5], [6], [7], the projected number of cells/gates for Multi-Processor Units based on ITRS 2001-2011 [8], together with the number of gates/cells in different EDA benchmarks (detailed in Table I).

As can be seen in Fig. 1, the difference between the number of cells/gates in commercial products and EDA benchmarks has been on the scale of multiple orders of magnitude. Furthermore, we can observe that the difference between the size of benchmarks and that of real designs has been increasing since 2005.

To address some of the existing problems of current EDA benchmarks, we created OPDB. OPDB is a scalable, modular, heterogeneous, and extensible collection of design benchmarks based on industrial strength, FPGA- and ASIC-tested designs from OpenPiton [9] and other open-source projects, such as, accelerators from OpenCores [10], [11], [12] and the MIAOW GPGPU [13].

0278-0070 ©2021 IEEE. https://doi.org/10.1109/TCAD.2021.3096794

TABLE I: Circuit Benchmark Characteristics

Benchmark	Type	Max Gate/Cells or (Nodes	Scalable	Modular	Synthesis	Placement	Routing
74X-series [16]	R	61		✓		✓	✓
ISCAS'85 [17]	R	1,512			✓	✓	✓
ISCAS'89 [18]	R	22,179			✓	✓	✓
LGSynth'89 [19]	R	4,000			✓		
LGSynth'91 [20]	R	35,000			✓		
IWLS'93 [21]	R	35,000 (est.)			✓		
ISPD'98 [22], [23], [24]	R	210,341				✓	
ITC'99 [25]	R	98,726	✱	✓	✓	✓	✓
Inacio et al. [26]	R	14,550		✓	✓	✓	✓
PEKO/PEKU [27]	S*	210,341				✓	✓
IWLS'05 [28]	R	899,632	✱	✓	✓	✓	✓
ISPD'05 [29]	R	2,177,353				✓	✓
LEKO/LEKU [30]	S*	1,166,655 (N)			✓		
ISPD'06 [29]	R	2,507,954				✓	✓
ISPD'07 [29]	R	494,011					✓
ISPD'08 [29]	R	2,507,954					✓
ISPD'11 [31]	R	1,293,433				✓	✓
DAC'12 [32]	R	1,364,958				✓	✓
ICCAD'12 [33]	R	1,364,958				✓	✓
ISPD'12 [34]	R	958,780				✓	✓
ICCAD'13 [33]	R	1,364,958				✓	✓
ISPD'13 [35]	R	982,258				✓	✓
ICCAD'14 [33]	R	958,792				✓	✓
ISPD'14 [36]	R	1,286,948				✓	✓
EPFL'15 [37]	R	214,335			✓	✓	✓
	S	23,339,737			✓	✓	✓
Matos et al. [38]	R	200,762			✓	✓	✓
ICCAD'15 [33]	R	1,931,639				✓	✓
ISPD'15 [39]	R	1,286,948				✓	✓
ICCAD'17 [33]	R	130,661				✓	✓
ISPD'18 [40]	R	290,386				✓	✓
ISPD'19 [41]	R	899,404				✓	✓
OPDB	R	*arbitrary*	✓	✓	✓	✓	✓

Though it is hard to predict the composition of future designs, tying a benchmark suite to an extensible, modular, and actively developed research infrastructure can provide – in a timely manner – test cases with contemporary size, modules, and designs. Moreover, the benefits can extend across disciplines to enable collaborative research across fields. Specifically, it can further push the envelop for across-the-stack optimizations down to the EDA tool level. Furthermore, Machine Learning enabled EDA tools and techniques [14] will benefit from OPDB's surplus of provided configurations and published data relating to design-point instances (tapein/tape-out), potentially making OPDB the ImageNet [15] of EDA. Our work makes the following contributions:

- Creation and characterization of a scalable, modular, heterogeneous, and extensible benchmark based on FPGA- and ASIC-tested real designs written in Standard Verilog.
- Release of the OPDB benchmark, which includes hundreds of designs ranging from tens to billions of transistors, to enable more EDA research opportunities.
- Development of Tursi, a tool for further scaling, configuration, and extension of the OPDB benchmark suite.

II. Related Work

In this section we discuss previously published related work and released benchmarks. Overall, benchmarks can be broadly grouped into two categories [42]: the first includes benchmarks based on real designs, either from industry or academia, while the second includes artificial (i.e., randomly generated or synthetic) designs generated through some algorithmic process. The synthetic category can be further split into two subcategories, those of synthetic designs and synthetic designs with known optimal/upper bounds (also referred as *exact benchmark circuits*). Table I provides a high-level overview of previous EDA benchmark suites. Table I is composed of six columns. The first column, `Benchmark`, provides the name of the benchmark, that be a name provided by the authors or the conference it was published, and one or more references that include further information for the specific benchmark. The `Type` column specifies if the benchmark is based on real designs (R), synthetic (S), or synthetic with known optimal/upper-bounds (S*). Additionally, the number of cells or gates for the largest design in the benchmark is included. Columns `Scalable` and `Modular` respectively indicate if the designs can be scaled arbitrarily, and if sub-modules can be targeted from the provided designs. Finally, columns `Synthesis`, `Placement`, and `Routing` mark which levels of benchmarking can be achieved with each benchmark suite. The ITC'99 and IWLS'05 benchmarks are marked with a ✱ symbol to indicate that they are limited in their scalability. For ITC'99 a single design can be scaled through replication. In IWLS'05 the leonmp design can be scaled through changing the core number parameter in the VHDL file. In both cases, enabling scalability is not a goal.

The majority of existing benchmarks are based on real designs, with circuits mostly provided by the industry. More recently, with the adoption of open source practices in hardware development, public open-source and academic designs have been included.

The MCNC and LGSynth benchmarks [17], [18], [19], [20], [21], together with the 74X-series circuits [16], were early benchmark suites which found wide use in academia [22]. In 1998, the year Intel released the Intel Celeron processor with 7.5M transistors [2], the largest public benchmark numbered 100K standard cells, while the second largest was just over 25K cells [43], [22]. To address this gap the ISPD'98 and ITC'99 benchmarks were constructed, providing a more gradual increase in complexity between each test case [22], [23], [24], [25].

Due to the rapid pace of the IC industry, industrial benchmarks were quickly becoming obsolete in terms of complexity, and artificial benchmarks were introduced to address this issue [44], [37], [45], [46], [47], [30], [27]. Apart from addressing issues of scale, synthetic benchmarks have been proposed as a way to assess how close to an optimal solutions can contemporary EDA tools and algorithms get [30], [48], [49]. Specifically, while traditionally new proposals are evaluated based on how they perform in relative terms compared to previous work, synthetic benchmarks with known optimal solutions (also known as exact benchmarks) can be constructed thus providing the opportunity to evaluate how a technique performs in terms of achieving a known optimal point. Despite the flexibility that artificial benchmarks provide, such approaches have also received criticism on their modelling precision [50], [22].

Over the last two decades, multiple benchmark suite iterations have been released, most in the context of workshop

or conference contests, such as the ISPD and ICCAD contests [29], [33], with the designs predominantly coming from industry. An important release for the open-source hardware community was the IWLS'05 benchmark [28], where multiple designs from OpenCores [12] were included.

OPDB attempts to combine the best attributes of previous benchmarks. In one hand, it includes open-source based designs and frameworks that enable free modification, distribution, and collaboration. At the same time it does not compromise on the quality of the chosen designs which are of industrial strength, contemporary scale, and are FPGA- and ASIC- tested.

Benchmarking of EDA tools and libraries can occur in two axes. Inacio et al. [26] describe the concept of *Vertical Benchmarking*, which requires that designs are provided in multiple formats and stages of the flow (behavioral, structural, and gate for behavioral synthesis, logic synthesis, and physical design, respectively). OPDB attempts to satisfy the requirements of Vertical Benchmarking by providing files in Verilog and BLIF formats. Moreover, it includes example scripts for converting Verilog to BLIF and AIGER formats using open-source EDA tools. On the alternate axis, Kahng et al. [42] proposed *Horizontal Benchmarking*, which attempts to provide an accurate comparison across tools at a given stage. In OPDB, to ease comparison between different tools at a given stage we choose to follow standard formats that are accepted both by commercial and open-source tools.

Finally, though some past benchmarks include modules which can be scaled in size (such as an ALU with a sliced design, a replicated module, or a multicore processor), such test cases are not representative of contemporary designs. Many of today's industrial designs resemble the tiled design of OpenPiton, instances of which are included in the OPDB design benchmark. OPDB thus emphasizes scalability in a way that directly reflects designs that are under active industrial development.

III. OPDB: A Scalable and Modular Design Benchmark

To address the scalability gap between contemporary designs and EDA benchmarks, we introduce OPDB: a scalable and modular design benchmark suite. Our goal is to provide a design benchmark suite that: covers a wide range of scales, enables modularity (so that specific design blocks can be targeted), allows for heterogeneity, and provides the ability to easily extend the existing set of benchmarks.

OPDB is composed of two parts: the first, and more visible, is a large collection of design benchmarks with varying size and functions based on modules and designs extracted or generated from the OpenPiton framework and other open-source hardware projects [9], [13], [10], [11]. Each of those design benchmarks is contained inside a "pickled" file; i.e., a single file which encloses post-processed all dependencies and modules for the specific design instance. The second part is Tursi: a tool for further scaling, configuration, and extension of the benchmark suite. Tursi was the tool used to generate the "pickled" (standalone, pre-processed, single-file) versions of the benchmark modules and designs that compose the first part.

A. Hardware Components

OPDB includes IP from a number of open-source hardware frameworks and projects: OpenPiton [9], MIAOW [13], and OpenCores [12].

OpenPiton is the first open-source, general-purpose, multi-threaded, manycore processor, developed by Balkind et al. [9]. Its main codebase is written in industry standard Verilog HDL (some newer inclusions such as the Ariane core [51] are written in SystemVerilog), and its design enables easy scalability, module configuration, extension, and modularity.

A compelling feature of the OpenPiton ecosystem is that apart from a codebase and build infrastructure, its designs have been validated through both FPGA and ASIC implementation, and their power and performance have been evaluated [52], [53], with all evaluation data being openly available.

OpenPiton's P-Mesh Network-on-Chip (NoC) and cache coherence system IP underlies OPDB's ability to achieve significant scalability, with the original OpenPiton environment providing the potential for scaling up to 500 million cores in a system (or 65K cores in a single chip) [9]. A recent expansion of OpenPiton, BYOC (Bring Your Own Core) [54], has demonstrated in practice that P-Mesh can further act as a glue for connecting heterogeneous components. The BYOC extension (now part of OpenPiton) supports the replacement of many of OpenPiton's default components like its cores and NoCs and provides standard ways to connect new accelerators. We have exploited this scalability and heterogeneity in the creation of OPDB to enable the instantiation of heterogeneous cores and accelerators into a single, scalable system on chip (SoC). Users of Tursi can create a configuration of their choice for internal evaluation purposes, featuring a mix of components with suitable scalability in number of tiles as needed for their use case. In addition, as a number of components have been already integrated the existing infrastructure make available AXI-Lite and Wishbone interfaces to the NoC that can be used for introducing further extensions and custom hardware. We have tested this in the scope of this work by integrating two accelerators (hosted by OpenCores [12]) with the P-Mesh NoC. The two accelerators we used were FFT (Fast Fourier Transformation) [10] and GNG (Gaussian Noise Generation) [11]. At this point we want to highlight that while integration of components in a tiled fashion is relatively easy, it does require editing of the OpenPiton codebase and knowledge of the NoC interfaces. Specifically, for integrating a new component, a wrapper module that performs the integration with the P-Mesh NoC needs to be created. Furthermore, the "chip" and "tile" modules will need to be modified so that the new design is instantiated in the appropriate coordinates. As detailed explanation of these steps is beyond the scope of this work we direct all interest readers to the BYOC work [54] and the OpenPiton code repository [55].

The above make OpenPiton an excellent source of benchmark designs, providing a plethora of reference points from real-world, industrial-strength implementations. Moreover, its

demonstrated extensibility and active development continuously expand the available modules and possible combinations.

In addition to the OpenCores accelerator cores, the MIAOW GPGPU is also included in OPDB. MIAOW is the first open-source RTL implementation of an OpenCL compatible GPGPU. It is based on the Southern Islands ISA and was developed by the Vertical Research Group at the University of Wisconsin-Madison [13].

B. Tursi

After identifying OpenPiton as a suitable source of designs and a framework that allows for scalability and which can act as a core of our benchmark suite, we worked to meet the rest of our target goals: modularity, and extensibility. These two features are significantly impacted by the scheme used for dependency description.

The OpenPiton build system follows the industry-standard convention of relying on file lists (Flists) for dependency description – Flists are text files that contain a list of all HDL source code files that a design is composed of. Flists are a simple and straightforward approach for collecting and enumerating all components that a design builds upon, and can be useful in a relatively static design. While Flists could be used to create and describe a benchmark suite, achieving modularity and extensibility for the generated benchmarks would come at a significant overhead to the user. This is because the flat description provided by Flists hides dependencies (e.g. common header files) and the naturally occurring design hierarchies, thus extracting or isolating sub-modules becomes impossible. To address these limitations, we developed Tursi: an extensible framework for "pickling" IP blocks based on a given HDL code base. Leveraging FuseSoC [56] and Icarus Verilog [57], Tursi "pickles" the chosen designs, bringing all of the dependent RTL files together into one industry-standard Verilog file. Any necessary pre-processing (including for macros) is done in advance by FuseSoC and Icarus Verilog to remove the need for EDA researchers and developers to understand the design and its build infrastructure. Tursi maintains the module hierarchy of the targeted design and does not collapse the modules to flattened design.

The main design goals for Tursi were to develop a configurable and extensible framework for creating "pickled" instances of different module/design configurations. To meet these goals we used the FuseSoC framework and its Core API (CAPI) description format to describe and process dependencies across modules. FuseSoC is "a package manager and build system for HDL" [56]. FuseSoC's CAPI description format uses the YAML language and contains all information regarding which files compose the specified module/core, possible dependencies to other cores, and EDA tool parameters for a specific task, it also allows dynamic code generation.

To introduce a new module into the Tursi framework (and thus OPDB), a CAPI file needs to be created to describe the target module. Besides the files that compose the module and possible dependencies on other CAPI files, the user needs to define a build target that performs the "pickling". This includes the call to Icarus Verilog and may also include pre-processing steps for configuration or file generation, like the PyHP preprocessor used by OpenPiton or the FFT generator. Listing 1 presents an example of a Core API file for the SPARC core that is part of the OpenPiton infrastructure. The first line defines the Core API version used in the document, while the second line defines the tag used to uniquely identify a core in VLNV format (Vendor Library Name Version). Lines 5 to 22 describe the composition of the design. A typical instance would includes a list of the HDL source code files, dependencies to other modules (common headers or sub-designs), and possibly initialization files for memories. In the specific example, lines 8 to 15 describe the dependencies on other submodules, notice that dependencies to other designs are described through the use VLNV tags. Lines 17-22 enumerates the files that extend and connect the submodules from the dependency list to implement the targeted design; for this instance it is a full SPARC T1 Core. Finally, lines 29-35 describe a build target. The build target is composed from the definition of the tools that would be used their command line configuration, the top-level module of the design, and the fileset to be used. The specific instance describes the "pickling" process which uses iverilog to pre-process the fileset and generate a single "pickled" file. The "pickled" output file is the merge of all the input files in the specified fileset(s) with their inclusions and macros expanded.

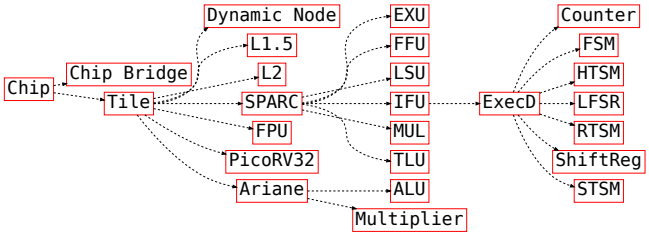


Fig. 2: OPDB module hierarchy.

Describing the dependencies of a given code base using CAPI allows the generation a natural hierarchy of IP blocks. By utilizing the dependency description feature of the CAPI format each component can be broken to several subcomponents that match the sub-blocks of a design schematic. This hierarchy can be utilized to generate a wide range of design benchmarks moving from simpler to more complex designs as we move towards the root of the dependence tree. CAPI files by clearly describing the dependencies between a module and its sub-components eliminate redundancy that would have been unavoidable in other descriptions, such as Flists. Figure 2 presents a visualization of the hierarchy of IP blocks that naturally arises from our CAPI description of OpenPiton's dependencies. Notice how the graphical representation of the SPARC Core and its submodules correlates with the description of Listing 1. After describing the dependencies between modules and the files that compose them, the task of generating the different "pickled" instances is reduced to invoking Tursi with each of the available VLNV tags; appropriate commandline flags can be used to customize an instance if wanted.

Listing 1: SPARC Core Core API file excerpt

```
CAPI=2:
name: openpiton::sparc_core:0.1
description: SPARC Core

filesets:
    rtl:
        depend:
            - openpiton::piton_include
            - openpiton::sparc_srams
            - openpiton::sparc_exu
            - openpiton::sparc_ffu
            - openpiton::sparc_lsu
            - openpiton::sparc_ifu
            - openpiton::sparc_mul
            - openpiton::sparc_tlu
        files:
            - sparc_core.v
            - sparc.v
            - cpx_spc_rpt.v
            - cpx_spc_buf.v
            - cfg_asi.v
        file_type: verilogSource-2001

...

targets:
    default:
        filesets: [rtl]
    pickle:
        default_tool: icarus
        filesets: [rtl]
        toplevel: [sparc_core]
        tools:
            icarus:
            iverilog_options: [-g 2001 -E]

...
```

C. The OPDB Design Benchmark

As previously mentioned, the OPDB Design Benchmark is composed of a large number of pre-generated benchmark files based on OpenPiton and other open-source hardware frameworks and codebases, and Tursi, our configurable infrastructure for generating "pickled" designs.

Tursi enables us to extract sub-modules of the available tile components or scale up to multi-tile designs. We have pre-generated hundreds of design examples for our initial release of OPDB. In addition, the user can utilize Tursi to generate different configurations of these designs, scale them up, or create new designs, either based on new components that are integrated into OpenPiton or a completely new code base. Table II and Table III present an overview of the modules that can be configured and in which ways. In Table II the first two columns provide the design name and top-level module name for each of the designs that we expose through Tursi. The third column, "Macros", in Table II identifies the modules that contain macros for register files or memory arrays, while the remaining columns identify the configuration knobs that are exposed to the user for each design. Table III expands the information of Table II by providing a description of the knobs, what they corresponds to, and the range of configuration values.

While scalability and level of complexity are important factors that need to be addressed by emerging EDA tools, the development and optimization of new approaches is easier on smaller, simpler circuits [25]. By exploiting the hierarchy of CAPI files, we are able to describe all dependencies in OpenPiton and generate a wide range of designs, from large ones that number billions of transistors to smaller components of tens or hundreds of gates (see examples in Table IV).

Another important capability of the OPDB benchmark is its extensibility and heterogeneity. Tursi leverages the "by-design" scalability and modularity of the OpenPiton codebase to populate a contemporary, scalable, modular design benchmark suite. While the initial release of OPDB focused on OpenPiton, the infrastructure and methodology for creating the design benchmarks is independent of it, meaning that OPDB can be easily extended to other code bases. New open-source designs which provide CAPI descriptions can be easily incorporated into OPDB and decomposed into their sub-components to immediately provide a variety of new benchmarks. As an example of this we created CAPI files and pickled a number of other open-source hardware codebases. These include the MIAOW GPGPU [13] codebase which was already integrated with OpenPiton, and the GNG [11] and FFT [10] accelerator cores hosted by OpenCores [12].

The choice of OpenPiton as the core base for our design benchmark further enables heterogeneity and extensibility at scale. Though OpenPiton was initially released as a tiled manycore framework focusing on general purpose processors, it has evolved into a scalable heterogeneous system, which supports the integration of different type of tiles, including multiple processors with heterogeneous-ISA [54], and accelerators like the general-purpose Graphic Processing Unit (GPGPU) [13] or the FFT [10] and GNG [11] accelerators we integrated for the purpose of this work.

Overall, based on the above demonstrated growing heterogeneity, extensive scalability, and a multitude of configuration options and easy extensibility, we believe that the OPDB Design Benchmark can provide continuous support for evaluating the performance of EDA tools and enable new research opportunities. In regard to performance evaluation of existing or new EDA tools, as hundreds of design points is not a practical target for evaluating a new technique, we suggest that subsequent work that uses OPDB focuses on submodules and design points that match the Piton processor [58]. The use of these configuration points have the additional benefit that the instances of these modules have been extensively verified and taped-out, in addition an extensive power and energy characterization exists for them [52]. As our target is for OPDB to be a dynamic benchmark suite which evolves over time, the suggested comparison baseline will be updated as new larger instances are evaluated and characterized through tapeins and tapeouts. In regard to new opportunities, Machine Learning enabled EDA tools and techniques [14] will benefit from OPDB's surplus of provided configurations, allowing training in different design scales and heterogeneous designs.

TABLE II: OPDB module configuration options

Module name	Top module	Macros	X-dim	Y-dim	Topology	L1-I	L1-D	L1.5	L2	Hardware Verified
chip	chip	✓	✓	✓	✓	✓	✓	✓	✓	FPGA/Tapeout
chip_bridge	chip_bridge									FPGA/Tapeout
dynamic_node	dynamic_node_top_wrap				2dmesh					FPGA/Tapeout
	dynamic_node_top_wrap_para				xbar					FPGA
fpga_bridge_rcv_32	fpga_bridge_rcv_32									FPGA
fpu	fpu									FPGA/Tapeout
ifu_esl	sparc_ifu_esl									FPGA/Tapeout
ifu_esl_counter	sparc_ifu_esl_counter									FPGA/Tapeout
ifu_esl_fsm	sparc_ifu_esl_fsm									FPGA/Tapeout
ifu_esl_htsm	sparc_ifu_esl_htsm									FPGA/Tapeout
ifu_esl_lfsr	sparc_ifu_esl_lfsr									FPGA/Tapeout
ifu_esl_rtsm	sparc_ifu_esl_rtsm									FPGA/Tapeout
ifu_esl_shiftreg	sparc_ifu_esl_shiftreg									FPGA/Tapeout
ifu_esl_stsm	sparc_ifu_esl_stsm									FPGA/Tapeout
l15	l15_wrap	✓			✓		✓	✓		FPGA/Tapeout
l2	l2	✓			✓				✓	FPGA/Tapeout
MIAOW (GPGPU)	neko	✓								FPGA/Tapein
pico	picorv32									FPGA/Tapeout
sparc_core	sparc_core	✓				✓	✓			FPGA/Tapeout
sparc_exu	sparc_exu_wrap									FPGA/Tapeout
sparc_ffu	sparc_ffu_nospu_wrap	✓								FPGA/Tapeout
sparc_ifu	sparc_ifu	✓				✓				FPGA/Tapeout
sparc_lsu	lsu	✓					✓			FPGA/Tapeout
sparc_mul	sparc_mul_top_nospu_wrap									FPGA/Tapeout
sparc_tlu	tlu_nospu_wrap									FPGA/Tapeout
tile	tile	✓			✓	✓	✓	✓	✓	FPGA/Tapeout
FFT	fftmain									FPGA
GNG	gng									FPGA/Tapein

TABLE III: OPDB module configuration options details

Attribute	ID in Table II	Value 1	Value 2
L1 data cache	L1-D	size (Bytes)	associativity
L1 instruction cache	L1-I	size (Bytes)	associativity
L1.5 cache	L1.5	size (Bytes)	associativity
L2 cache	L2	size (Bytes)	associativity
Network topology	Topology	2dmesh, xbar	-
# tiles in X dimension	X-dim	width [1,256]	-
# tiles in Y dimension	Y-dim	width [1,256]	-

D. Essential Attributes of OPDB Modules

Table IV presents a high-level overview of the basic modules provided by OPDB and their essential attributes. The OpenPiton Dynamic Node, L1.5 cache and L2 cache are part of the "P-Mesh" uncore (caches, cache-coherence protocol, NoCs, NoC-based I/O bridges, *etc.*) of OpenPiton [9]. Modules prefixed with OST1 are part of the OpenSPARC T1 core [59], an industrial strength core originally developed by Sun Microsystems. Modules prefixed with ExecD are part of the Execution Drafting [60] hardware included with OpenPiton. MIAOW is the first open-source RTL implementation of an OpenCL compatible GPGPU [13]. The GNG [11] and FFT [10] modules are hosted by the OpenCores project [12]. Modules prefixed with Ariane are part of the Ariane RISC-V core [53], an open-source 64-bit RISC-V application class processor. The Ariane core is written in SystemVerilog and is currently not a part of OPDB. We are working to enable the full inclusion of SystemVerilog codebases in future releases. As part of this effort, the Ariane sub-modules listed on Table IV (but not the core) were synthesized after being automatically translated from SystemVerilog to Verilog using sv2v [61]. Finally, if multiple prefixes exist (e.g. OpenPiton OST1) it means both components were included in the design benchmark.

The modules in Table IV are sorted based on the number of cells they contain, smallest to largest. All numbers were collected using commercial EDA tools and contemporary technology libraries. We run the tools on a 14 core Intel® Xeon® E5-2680 v4, utilizing only six of the cores running at 2.4 GHz with 192 GB of memory.

The following list describes each column in Table IV:

* `Cells`: The total number of logic cells used to synthesize the module; memory macros count as one cell per macro.
* `Nets`: The wires that connect ports to pins and/or pins to each other; starred (*) numbers are estimates.
* `Ports`: The input, output, or inout ports of the module.
* `Area (kGE)`: The area of the synthesized design reported in kilo Gate Equivalent (kGE). The designs that contain large memory macros, like OpenPiton L2 cache, would report a bigger number. Note that we turn the memory macros into blackboxes for the MIAOW module (marked with ◆), therefore the area number excludes the memory macros.
* `Delay (FO4)`: Delay of the longest path in the design.
* `Runtime`: Time taken to synthesize a given module. Short indicates runtime of less that one hour, medium indicates a runtime between one and six hours, long indicates a runtime of more than six hours. Due to hierarchical synthesis, the runtime for a Tile, which includes an OST1 core, L1.5 and L2 caches, and three NoC routers, is much shorter than a core itself.
* `Memory Macros`: The number of memory macros con-

TABLE IV: OPDB designs essential attributes

Module	Cells	Nets	Ports	Area (kGE)	Delay (FO4)	Runtime	Memory Macros
ExecD multiple sub-components	<1000	<1000	<1000	<10	<20	short	0
OST1 Instruction Fetch Unit (IFU)	1,900	8,500	4,500	<10	<50	short	5
Ariane Arithmetic and Logic Unit (ALU)	2,900	3,000	270	<10	<40	short	0
ExecD Execution Drafting	3,000	3,400	400	<10	<30	short	0
OST1 Load/Store Unit (LSU)	3,600	14,000	8,000	<10	<50	short	5
GNG Gaussian Noise Generator	6,800	7,000	24	14	<40	short	0
OST1 Floating-point Frontend Unit (FFU)	5,300	5,700	560	33	<40	short	1
OpenPiton Dynamic Node	8,300	8,800	890	24	<40	short	0
OST1 Multiplier Unit (MUL)	11,000	11,000	200	26	<20	short	0
PicoRV32 RISC-V Core	11,000	11,000	620	27	<20	short	0
Ariane Multiplier	14,000	15,000	210	44	<30	short	0
OpenPiton Chip Bridge	24,000	25,000	1,700	87	<20	short	0
OST1 Floating-Point Unit (FPU)	26,000	28,000	1,100	75	<50	short	0
OST1 Trap Logic Unit (TLU)	31,000	33,000	3,100	110	<30	short	0
OpenPiton L1.5 cache	34,000	40,000	7,700	260	<50	short	7
OpenPiton L2 cache	42,000	47,000	6,700	820	<70	short	8
FFT Fast Fourier Transform	80,000	83,000	4,000	285	<30	short	0
OST1 EXecution Unit (EXU)	75,000	76,000	2,300	260	<40	short	0
Ariane RISC-V Core	130,000	150,000	14,000	860	<50	short	32
OpenSPARC T1 Core (OST1)	180,000	190,000	6,300	990	<60	medium	11
OpenPiton OST1 Tile	310,000	420,000*	23,000	2200	<70	short	26
MIAOW GPGPU	658,000	808,000	152,000	1800	<50	medium	212
OpenPiton OST1 Chip 3x3 (9 Tiles)	2,800,000	4,500,000*	210,000	20000	<70	short	234
OpenPiton OST1 Chip 5x5 (25 Tiles)	7,800,000	12,000,000*	570,000	56000	<70	medium	650
OpenPiton OST1 Chip 10x10 (100 Tiles)	31,000,000	50,000,000*	2,300,000	220000	<70	medium	2600
OpenPiton OST1 Chip 15x15 (225 Tiles)	70,000,000	110,000,000*	5,100,000	500000	<70	long	5850
OpenPiton OST1 Chip 19x19 (361 Tiles)	110,000,000	180,000,000*	8,200,000	810000	<70	long	9386

tained in the module. Larger memory macros instantiated in RTL may be split into smaller ones during implementation process due to technology limitations.

All reported numbers are extracted directly from the reports generated by the synthesis tool, except starred (*) entries. Starred (*) entries represent manually calculated estimates. Manual estimates were required as the hierarchical flow we used to synthesize larger designs only counts the interface logic between modules to optimize the process. Therefore, we add up numbers for sub-modules and provide an estimate. Furthermore, `Cell`, `Nets`, and `Ports` numbers use two significant figures of accuracy. Cell counts do not include the cells included inside memory macros, making our reported numbers conservative. For example, the Piton chip [58], which is essentially a 25-tile instance of OpenPiton, was reported to have 460 million transistors (∼77M cells) [52], while our synthesized 25-tile chip instance is reported with 7.8M cells excluding memory macro cells. As Power calculations require the use of activity factors which can vary significantly based on the use case, we did not include such projections in this work. Instead, the publicly available data from the Piton chip [58] can be used to extract Power and Energy information for a number of the available OPDB designs.

From Table IV we can see that OPDB provides a wide range of benchmark designs. The smallest is a 49-bit shift register numbering only tens of cells, while the largest design – we provide numbers for – is a 19x19 OpenPiton chip with 110 million cells. This is 44× more cells than the largest previous benchmark based on real designs, even without the full cell count for the memory macros. Availability of large designs is essential for benchmarking new EDA tools. To paraphrase Alpert [22], a tool which achieves a 5% improvement on a small design is not as interesting or relevant as a tool which achieves a 5% improvement on a large design.

Note that the smallest and largest designs present in Table IV do not define the limits of OPDB. Smaller designs can be created by targeting a different module or a sub-module of the existing ones. In the most trivial case single leaf modules could be targeted. Similarly, the largest design we report is only limited by the resources of the machine on which we run synthesis.

The numbers reported in Table IV are provided as reference points to assist tool designers in making decisions on which modules they should target but not as numbers for comparison against the tools and libraries used in this work.

IV. Early Impact

Naturally, our team members were the first to use OPDB internally. In one instance we created an open-source-based flow for gate-level verification. Specifically, we synthesized the "pickled" version of the dynamic node from OpenPiton to the Nangate 45nm Open Cell Library [62] node using Yosys [63] and subsequently verified the gate-level output using Icarus Verilog [57] and an RTL-level generated stimuli file. We plan to release a skeleton of this flow together with OPDB as a showcase.

Furthermore, an early version of the OPDB benchmark was shared to a number of academic and industry groups. One group that received an early OPDB release is the Laboratory for NanoIntegrated Systems (LNIS) at the University of Utah. LNIS utilized designs from OPDB during the development and evaluation of the LSOracle logic optimization framework [64], [65]. By utilizing pickled design files the development team was able to concentrate on algorithmic development. Another

group that received early access to the OPDB design benchmarks was the OpenROAD development team, which decided to incorporated the dynamic_node module from OPDB on their tools' verification design suite [66]. Rovinski *et al.* [66] highlight that bigger designs can be too complex and include features that are missing from early development editions. OPDB by providing a wide range of designs both in scale and complexity allows early development to progress using smaller/simpler real-world design but also provides full scale configurations to stress the tools when completed.

V. Current Status

Through OPDB we provide an open source, scalable, modular, heterogeneous, and extensible design benchmark suite to the EDA community. The hundreds of pre-generated designs present a plethora of modules that can be used in all steps of EDA tool developement: from early development and testing, to algorithmic tuning, and extreme scale evaluation of a new technique. The Tursi framework is open source and is entirely based on open source tools. Moreover, all hardware codebases and frameworks used to create the OPDB suite are also open source, thus all designs of OPDB can be freely used, modified, and redistributed [55], [67]. As prior work has highlighted, the benefits of open source tools and benchmarks are significant [50]. Open-source benchmarks and tools enable researchers and developers to gain significant insight that can be then utilized to improve existing techniques. Furthermore, they enable modifications to tools and combinations of tools that can boost efficiency. In regard to benchmarks open source enables the identification and fixing of bugs in a continuous fashion, also it enables modifications and conversion of the designs that enable a broader use by the community. We believe that adopting an open source policy for development and evaluation will be beneficial to the overall community, and we encourage other teams to do the same with derived work.

Early use has shown the potential of OPDB across a range of different applications, from EDA tool development to scalability testing. To further ease the integration of OPDB with EDA tools, apart from pickled designs in Verilog, we will provide Yosys [63] scripts for translating to BLIF and AIGER formats; pre-generated BLIF files will be provided for a number of designs too. We encourage other teams to use OPDB with their tools and integrate their designs. We will assist teams who want to release their flows and designs and integrate with OPDB.

Finally, the current version of OPDB includes only Verilog designs. This is a by-choice feature of OPDB. Due to the lack of support for SystemVerilog in the open source community we decided not to include any SystemVerilog modules on our initial release. We are currently integrating an open source SystemVerilog to Verilog converter [61] to enable the creation of Verilog pickled designs from SystemVerilog codebases. We previewed the use of this converter to provide the results for the Ariane ALU and multiplier shown in Table IV.

VI. Acknowledgements

We sincerely thank Olof Kindgren (FuseSoC) and Zachary Snow (SV2V) for their help when using their tools, as well as the LSOracle, Synopsys, Xilinx, OpenROAD, and other DARPA IDEA performers. We also thank the anonymous reviewers for their feedback on earlier versions of this manuscript. We also thank all the contributors to the open-source projects utilized in OPDB. This work was supported in part by the Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under Agreement FA8650-18-2-7846. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory (AFRL), Defense Advanced Research Projects Agency (DARPA), or the U.S. Government.

References

[1] R. Brayton and J. Cong, "NSF workshop: Electronic design automation - past, present, and future," http://cadlab.cs.ucla.edu/nsf09/NSF_Workshop_Report_v2.pdf, 2009.

[2] Intel Corporation, "Transistors to transformations: from sand to circuits - how Intel makes chips," 2012. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/corporate-information/museum-transistors-to-transformations-brochure.pdf

[3] The New York Times, "To power A.I., start-up creates a giant computer chip," 2019. [Online]. Available: https://www.nytimes.com/2019/08/19/technology/artificial-intelligence-chip-cerebras.html

[4] P. H. Madden, "Reporting of standard cell placement results," in Proceedings of the 2001 International Symposium on Physical Design, ser. ISPD '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 30–35. [Online]. Available: https://doi.org/10.1145/369691.369727

[5] Nvidia, "Inside Pascal," 2016. [Online]. Available: https://devblogs.nvidia.com/inside-pascal/

[6] Nvidia, "Inside Volta," 2017. [Online]. Available: https://devblogs.nvidia.com/inside-volta/

[7] AMD, "Radeon RX Vega 64," 2017. [Online]. Available: https://www.amd.com/en/products/graphics/radeon-rx-vega-64

[8] Semiconductor Industry Association, "The international technology roadmap for semiconductors (ITRS)," 2001-2011.

[9] J. Balkind, M. McKeown, Y. Fu, T. Nguyen, Y. Zhou, A. Lavrov, M. Shahrad, A. Fuchs, S. Payne, X. Liang, M. Matl, and D. Wentzlaff, "Openpiton: An open source manycore research framework," in Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems, ser. ASPLOS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 217–232. [Online]. Available: https://doi.org/10.1145/2872362.2872414

[10] D. Gisselquist, "A generic pipelined FFT core generator," https://opencores.org/projects/dblclockfft, 2018.

[11] G. Liu, "Gaussian noise generator core specification," https://opencores.org/projects/gng, 2015.

[12] OpenCores Community, "OpenCores," https://opencores.org, 2020.

[13] R. Balasubramanian, V. Gangadhar, Z. Guo, C.-H. Ho, C. Joseph, J. Menon, M. P. Drumond, R. Paul, S. Prasad, P. Valathol, and K. Sankaralingam, "Enabling GPGPU low-level hardware explorations with miaow: An open-source rtl implementation of a GPGPU," ACM Trans. Archit. Code Optim., vol. 12, no. 2, Jun. 2015. [Online]. Available: https://doi.org/10.1145/2764908

[14] A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, S. Bae, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, A. Babu, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, and J. Dean, "Chip placement with deep reinforcement learning," 2020.

[15] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.

[16] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering," IEEE Design Test of Computers, vol. 16, no. 3, July 1999.

[17] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN," in International Symposium on Circuits and Systems, 1985, pp. 695–698.

[18] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in International Symposium on Circuits and Systems,, vol. 3, 1989, pp. 1929–1934.

[19] S. Yang, "Logic synthesis and optimization benchmarks," 1988.

[20] S. Yang, "Logic synthesis and optimization benchmarks user guide: Version 3.0," Tech. Rep., 1991.

[21] K. McElvain, "IWLS'93 benchmark set: Version 4.0," Tech. Rep., 1993.

[22] C. J. Alpert, "The ISPD98 circuit benchmark suite," in Proceedings of the 1998 International Symposium on Physical Design, ser. ISPD '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 80–85. [Online]. Available: https://doi.org/10.1145/274535.274546

[23] M. Wang, X. Yang, and M. Sarrafzadeh, "Dragon2000: Standard-cell placement tool for large industry circuits," in Proceedings of the 2000 IEEE/ACM International Conference on Computer-Aided Design, ser. ICCAD '00. IEEE Press, 2000, p. 260–263.

[24] S. N. Adya and I. L. Markov, "Consistent placement of macro-blocks using floorplanning and standard-cell placement," in Proceedings of 2002 International Symposium on Physical Design, ISPD 2002, Del Mar, CA, USA, April 7-10, 2002. ACM, 2002, pp. 12–17.

[25] F. Corno, M. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," IEEE Design Test of Computers, vol. 17, no. 3, Jul 2000.

[26] C. Inacio, H. Schmit, D. Nagle, A. Ryan, D. E. Thomas, Y. Tong, and B. Klass, "Vertical benchmarks for CAD," in Proceedings of the 36th Annual ACM/IEEE Design Automation Conference, ser. DAC '99. New York, NY, USA: Association for Computing Machinery, 1999, p. 408–413. [Online]. Available: https://doi.org/10.1145/309847.309969

[27] C.-C. Chang, J. Cong, and M. Xie, "Optimality and scalability study of existing placement algorithms," in Asia and South Pacific Design Automation Conference, 2003, pp. 621–627.

[28] C. Albrecht, "IWLS 2005 benchmarks," 2005. [Online]. Available: https://ddd.fit.cvut.cz/prj/Benchmarks/IWLS2005.pdf

[29] "International Symposium on Physical Design Contest," 2005-2019. [Online]. Available: http://www.ispd.cc/?page=contests

[30] J. Cong and K. Minkovich, "Optimality study of logic synthesis for lut-based fpgas," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 26, no. 2, Feb 2007.

[31] N. Viswanathan, C. J. Alpert, C. Sze, Z. Li, G.-J. Nam, and J. A. Roy, "The ISPD-2011 routability-driven placement contest and benchmark suite," in Proceedings of the 2011 International Symposium on Physical Design, ser. ISPD '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 141–146. [Online]. Available: https://doi.org/10.1145/1960397.1960429

[32] N. Viswanathan, C. Alpert, C. Sze, Z. Li, and Y. Wei, "The dac 2012 routability-driven placement contest and benchmark suite," in Proceedings of the 49th Annual Design Automation Conference, ser. DAC '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 774–782. [Online]. Available: https://doi.org/10.1145/2228360.2228500

[33] "CAD contest at ICCAD," http://iccad-contest.org/2019/history.html, 2012-2017.

[34] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke, and C. Zhuo, "The ISPD-2012 discrete cell sizing contest and benchmark suite," in Proceedings of the 2012 ACM International Symposium on International Symposium on Physical Design, ser. ISPD '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 161–164. [Online]. Available: https://doi.org/10.1145/2160916.2160950

[35] M. M. Ozdal, C. Amin, A. Ayupov, S. M. Burns, G. R. Wilke, and C. Zhuo, "An improved benchmark suite for the ISPD-2013 discrete cell sizing contest," in Proceedings of the 2013 ACM International Symposium on Physical Design, ser. ISPD '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 168–170. [Online]. Available: https://doi.org/10.1145/2451916.2451959

[36] V. Yutsis, I. S. Bustany, D. Chinnery, J. R. Shinnerl, and W.-H. Liu, "ISPD 2014 benchmarks with sub-45nm technology rules for detailed-routing-driven placement," in Proceedings of the 2014 on International Symposium on Physical Design, ser. ISPD '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 161–168. [Online]. Available: https://doi.org/10.1145/2560519.2565877

[37] L. Amaru, P.-E. Gaillardon, and G. De Micheli, "The EPFL combinational benchmark suite," 24th International Workshop on Logic Synthesis, 2015.

[38] J. M. Matos, A. Neutzling, R. P. Ribas, and A. Reis, "A benchmark suite to jointly consider logic synthesis and physical design," in Proceedings of the 2015 Symposium on International Symposium on Physical Design, ser. ISPD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 185–192. [Online]. Available: https://doi.org/10.1145/2717764.2717785

[39] I. S. Bustany, D. Chinnery, J. R. Shinnerl, and V. Yutsis, "ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement," in Proceedings of the 2015 Symposium on International Symposium on Physical Design, ser. ISPD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 157–164. [Online]. Available: https://doi.org/10.1145/2717764.2723572

[40] S. Mantik, G. Posser, W.-K. Chow, Y. Ding, and W.-H. Liu, "ISPD 2018 initial detailed routing contest and benchmarks," in Proceedings of the 2018 International Symposium on Physical Design, ser. ISPD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 140–143. [Online]. Available: https://doi.org/10.1145/3177540.3177562

[41] W.-H. Liu, S. Mantik, W.-K. Chow, Y. Ding, A. Farshidi, and G. Posser, "ISPD 2019 initial detailed routing contest and benchmark with advanced routing rules," in Proceedings of the 2019 International Symposium on Physical Design, ser. ISPD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 147–151. [Online]. Available: https://doi.org/10.1145/3299902.3311067

[42] A. B. Kahng, H. Lee, and J. Li, "Horizontal benchmark extension for improved assessment of physical cad research," in Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI, ser. GLSVLSI '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 27–32. [Online]. Available: https://doi.org/10.1145/2591513.2591540

[43] K. Koźmiński, "Benchmarks for layout synthesis - evolution and current status," in Proceedings of the 28th ACM/IEEE Design Automation Conference, ser. DAC '91. New York, NY, USA: Association for Computing Machinery, 1991, p. 265–270. [Online]. Available: https://doi.org/10.1145/127601.127678

[44] J. Pistorius, E. Legai, and M. Minoux, "Generation of very large circuits to benchmark the partitioning of fpga," in Proceedings of the 1999 International Symposium on Physical Design, ser. ISPD '99. New York, NY, USA: Association for Computing Machinery, 1999, p. 67–73. [Online]. Available: https://doi.org/10.1145/299996.300026

[45] A. B. Kahng and S. Kang, "Construction of realistic gate sizing benchmarks with known optimal solutions," in Proceedings of the 2012 ACM International Symposium on International Symposium on Physical Design, ser. ISPD '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 153–160. [Online]. Available: https://doi.org/10.1145/2160916.2160949

[46] J. Darnauer and W. W.-M. Dai, "A method for generating random circuits and its application to routability measurement," in Proceedings of the 1996 ACM Fourth International Symposium on Field-Programmable Gate Arrays, ser. FPGA '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 66–72. [Online]. Available: https://doi.org/10.1145/228370.228380

[47] M. D. Hutton, J. Rose, J. P. Grossman, and D. G. Corneil, "Characterization and parameterized generation of synthetic combinational benchmark circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 17, no. 10, Oct 1998.

[48] L. Amarú, M. Soeken, W. Haaswijk, E. Testa, P. Vuillod, J. Luo, P. Gaillardon, and G. De Micheli, "Multi-level logic benchmarks: An exactness study," in 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), 2017, pp. 157–162.

[49] W. L. Neto, V. N. Possani, F. S. Marranghello, J. M. Matos, P. Gaillardon, A. I. Reis, and R. P. Ribas, "Exact benchmark circuits for logic synthesis," IEEE Design Test, vol. 37, no. 3, pp. 51–58, 2020.

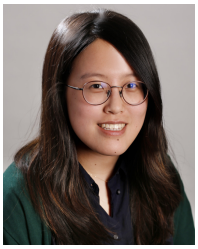
[50] S. N. Adya, M. C. Yildiz, I. L. Markov, P. G. Villarrubia, P. N. Parakh, and P. H. Madden, "Benchmarking for large-scale placement and beyond," in Proceedings of the 2003 International Symposium on Physical Design, ser. ISPD '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 95–103. [Online]. Available: https://doi.org/10.1145/640000.640022

[51] J. Balkind, M. Schaffner, K. Lim, F. Zaruba, F. Gao, J. Tu, D. Wentzlaff, and L. Benini, "OpenPiton+Ariane: the first SMP Linux-booting RISC-V system scaling from one to many cores," in Third Workshop on Computer Architecture Research with RISC-V, 2019.

[52] M. McKeown, A. Lavrov, M. Shahrad, P. J. Jackson, Y. Fu, J. Balkind, T. M. Nguyen, K. Lim, Y. Zhou, and D. Wentzlaff, "Power and energy characterization of an open source 25-core manycore processor," in

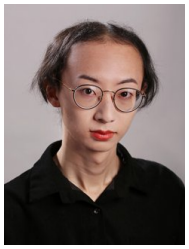
2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2018, pp. 762–775.
[53] F. Zaruba and L. Benini, "The cost of application-class processing: energy and performance analysis of a Linux-ready 1.7GHz 64bit RISC-V core in 22nm FDSOI technology," CoRR, vol. abs/1904.05442, 2019. [Online]. Available: http://arxiv.org/abs/1904.05442
[54] J. Balkind, K. Lim, M. Schaffner, F. Gao, G. Chirkov, A. Li, A. Lavrov, T. M. Nguyen, Y. Fu, F. Zaruba, K. Gulati, L. Benini, and D. Wentzlaff, "Byoc: A "Bring Your Own Core" framework for heterogeneous-isa research," in Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, ser. ASPLOS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 699–714. [Online]. Available: https://doi.org/10.1145/3373376.3378479
[55] Princeton Parallel Group, "OpenPiton Research Platform," https://github.com/PrincetonUniversity/OpenPiton, 2016.
[56] O. Kindgren, "A scalable approach to IP management with FuseSoC," in Workshop on Open Source Design Automation (OSDA), 2019.
[57] S. Williams and M. Baxter, "Icarus Verilog: open-source Verilog more than a year later," Linux Journal, vol. 2002, no. 99, Jul 2002.
[58] M. McKeown, Y. Fu, T. Nguyen, Y. Zhou, J. Balkind, A. Lavrov, M. Shahrad, S. Payne, and D. Wentzlaff, "Piton: A manycore processor for multitenant clouds," IEEE Micro, vol. 37, no. 2, pp. 70–80, 2017.
[59] Oracle, "OpenSPARC T1 microarchitecture specification," pp. 432–444, 2008. [Online]. Available: https://www.oracle.com/servers/technologies/opensparc-t1-page.html
[60] M. McKeown, J. Balkind, and D. Wentzlaff, "Execution Drafting: energy efficiency through computation deduplication," in 47th International Symposium on Microarchitecture, 2014.
[61] Z. Snow, "sv2v," https://github.com/zachjs/sv2v, 2019, accessed: 2019-10-12.
[62] Nangate, "Nangate 45nm Open Cell Library," 2010.
[63] C. Wolf, "Yosys Open SYnthesis Suite," 2019, accessed: 2019-09-12. [Online]. Available: http://www.clifford.at/yosys
[64] Laboratory for Nano Integrated Systems, "LSOracle," https://github.com/LNIS-Projects/LSOracle, 2019, accessed: 2019-09-12.
[65] M. Austin, S. Temple, W. L. Neto, L. Amarù, X. Tang, and P. Gaillardon, "A scalable mixed synthesis framework for heterogeneous networks," in 2020 Design, Automation Test in Europe Conference Exhibition (DATE), 2020, pp. 670–673.
[66] A. Rovinski, T. Ajayi, M. Kim, G. Wang, and M. Saligane, "Bridging academic open-source eda to real-world usability," in 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2020, pp. 1–7.
[67] Princeton Parallel Group, "OPDB: OpenPiton Design Benchmark Suite," https://github.com/PrincetonUniversity/OPDB, 2019.

Georgios Tziantzioulis is currently a Postdoctoral Research Associate with the Department of Electrical Engineering, Princeton University. His current research focus is on the design of power and energy efficient computer systems for datacenters and Cloud Services. Tziantzioulis received the Ph.D. degree in computer engineering from Northwestern University and the Diploma degree in computer and communication engineering from the University of Thessaly. Contact him at georgios.tziantzioulis@princeton.edu.



Ting-Jung Chang is currently working toward the Ph.D. degree with the Department of Electrical Engineering, Princeton University. Her research interests include computer architecture, memory systems, and emerging transistor technologies. Chang received the M.A. degree in electrical engineering from Princeton University. Contact her at tingjung@princeton.edu.

Jonathan Balkind is currently an Assistant Professor in the Department of Computer Science, UC Santa Barbara. His research interests include computer systems, programming languages, and computer architecture with the aim of improving the efficiency of modern multicore systems in mobile and datacenter environments. He received the M.Sci. degree in computing science from the University of Glasgow and the M.A. degree in computer science from Princeton University. Contact him at jbalkind@ucsb.edu.



Jinzheng Tu is currently working toward the Ph.D. degree with the Department of Electrical Engineering, Princeton University. Her research focuses on in-memory computing, digital circuits, and logical synthesis. Tu received the B.S. degree in electrical engineering from Tsinghua University. Contact her at jinzheng@princeton.edu.

Fei Gao is currently working toward the Ph.D. degree with the Department of Electrical Engineering, Princeton University. His research interests include in-memory compute, memory systems, and many-core processor design. Gao received the M.A. degree in electrical engineering from Princeton University. Contact him at feig@princeton.edu.

David Wentzlaff is currently an Associate Professor with the Electrical Engineering Department, Princeton University. His research interests include parallel computer architecture, architectures for cloud computing, and biodegradable computing systems. He has received the NSF CAREER award, the DARPA Young Faculty Award, the AFOSR Young Investigator Prize, and the Princeton E. Lawrence Keyes Faculty Advancement Award. Wentzlaff received the master's and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology. Contact him at wentzlaf@princeton.edu.