The Missing Pieces of Open Design Enablement: A Recent History of Google Efforts

Invited Paper

Tim Ansell
tansell@google.com
Google
Mountain View, California

Mehdi Saligane
mehdi@umich.edu
University of Michigan
Ann Arbor, Michigan

ABSTRACT

In an initiative to advance the open-source electronic design automation (EDA) and hardware design community, Google has been spearheading a global collaborative effort involving investigators from academia, start-ups as well as foundries. Open-source silicon being the end goal, multiple blossoming projects are supported to drive the renewed open-source wave to break down the barriers of EDA tooling and ultimately hardware design. This push toward the democratization of hardware also aims to develop and release an open-source platform of silicon-proven analog and digital IP blocks to serve as a foundation for rapid design of complex, secure systems-on-chip (SoCs) at bleeding edge technology nodes. This paper details the different efforts that constitute the missing pieces standing in the way of open design enablement such as: OpenROAD for EDA tooling, digital libraries such as OpenRAM and standard cells, and finally analog and mixed-signal (AMS) building blocks for SoCs such as: BAG and FASoC.

CCS CONCEPTS

• **Hardware → Software tools for EDA**.

KEYWORDS

OpenROAD, Open-Source, VLSI, EDA, Silicon Compilers, Yosys, SystemVerilog, Verible, UHDM, Surelog

ACM Reference Format:
Tim Ansell and Mehdi Saligane. 2020. The Missing Pieces of Open Design Enablement: A Recent History of Google Efforts: Invited Paper. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '20), November 2–5, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3400302.3415736

1 INTRODUCTION

Nathan Myhrvold's Four Laws of Software [1] state that software expands to consume any available compute power. This consumption has enabled the computer industry to create ever more complex and compelling products with fewer software engineering resources [2, 3]. In the last few years, the previously reliable Moore's Law, which enabled this increasing expansion, has slowed while the demand for ever more computing power has not. This continued growth can be found at most technology companies, including at companies like Google, as shown by Figure 1 which tracks the number of cycles spent transcoding highly compressed YouTube videos compared to compute performance [4].

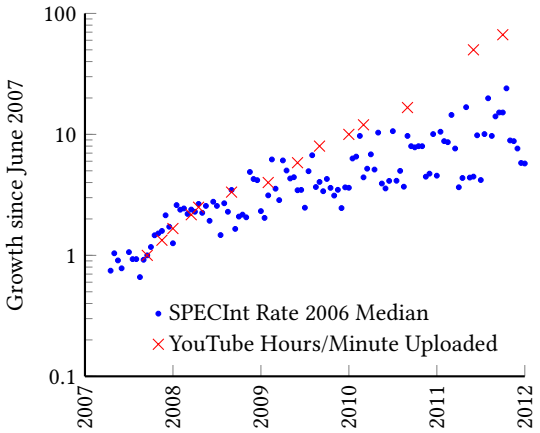

Figure 1: YouTube video uploads growth compared to CPU performance.

The impact of previous slowdowns around Moore's Law on unprepared companies is documented at [5]. Thus forward looking companies have already started to investigate new approaches to tackle this growing demand, such as the creation of hardware accelerator ASICs. Google in particular has been very successful with the development of hardware accelerators like the tensor processing unit (TPU) [6]. With this increasing involvement in the development of ASICs at Google, the gap shown in Figure 2 between theoretically achievable transistor density and the actual realized transistor density that EDA tooling is able to provide will become increasingly important [7].

2 REQUIREMENTS FOR A COLLABORATIVE RESEARCH ECOSYSTEM

Google's research philosophy starts with the following:

> *First and foremost, sharing knowledge accelerates progress for everyone.* [2]

This principle provides the base framework for our efforts in the hardware development space. Being able to reproduce and build on top of other researchers' results is a fundamental aspect required

ICCAD '20, November 2–5, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8026-3/20/11.
https://doi.org/10.1145/3400302.3415736

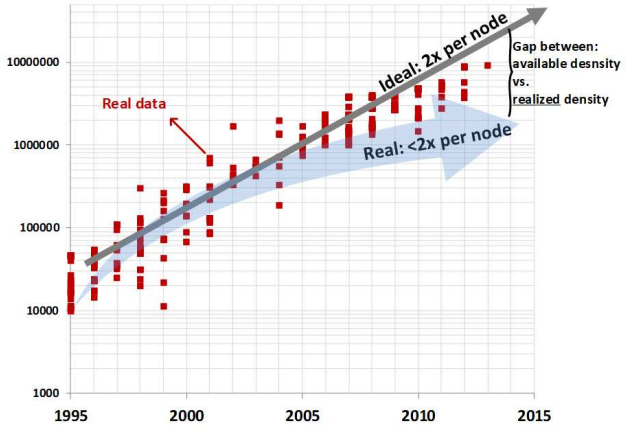

Figure 2: The growing EDA and Design capability gap.

for a reliable knowledge sharing ecosystem [8]. Fields of research that are computationally intensive, like those required for moving forward the computer and electronics industries, suffer from the additional complexity around managing both the code and data required to reproduce results. To address this, numerous groups have produced recommendations around best practices that should be followed. These include

> *[r]eproducibility [which] requires having the complete software environment available in other laboratories and the full source code available for inspection, modification* [9]

The Reproducible Research Standard [10] (RRS) (Stodden, 2009) mentions one the fundamental principles that enable reproducibility in science:

> *Principle of Scientific Licensing: Legal encumbrances to the dissemination, sharing, use, and re-use of scientific research compendia should be minimized, and require a strong and compelling rationale before application.*

The use of data only available under restrictive NDA, closed source proprietary tools, and hardware designs with restrictive licenses found frequently in all aspects of hardware design goes directly against this recommendation. The fact that research and development in the hardware design space is heavily dependent on these restrictive resources was highlighted by Andrew Kahng in [9, 11].

Helping to move research ecosystems towards the goal of transparency through open source code and dataset releases is an area where Google efforts have been so successful that it is also highlighted in Google's philosophy on research [2]:

> *We have always seen scientific publications as an important component for much of our research work, but for fundamental research projects, open-source code releases and new datasets can be particularly valuable. They help us evaluate and contextualize the research we do in addition to stimulating new collaborations.*

An example of effectiveness of fully open source code and dataset releases can be seen in Google's efforts in the machine learning space. In 2015, Google released the open source code for TensorFlow [12] with the following reasoning mentioned by Jeff Dean, a key software engineer behind many of Google's most significant advances who now is the Head of Research at Google:

> *What we're hoping is that the community adopts this as a good way of expressing machine learning algorithms of lots of different types, and also contributes to building and improving [TensorFlow] in lots of different and interesting ways* [13]

Since then, Google has continued to accelerate the development of TensorFlow in the open and combined it with releasing numerous open datasets for usage in the machine learning space. TensorFlow is an example of a commercial company coming to similar conclusions as the recommendation by [9] and

> *investing in research and development of open-source, usable tools and infrastructure that support reproducibility for a broad range of studies across different domains in a seamless fashion*

This investment by Google and also many other companies like Facebook, Nvidia and Microsoft has seen an explosion of research in this space [14].

Google's efforts in the TensorFlow space also highlight another important aspect of Google's research philosophy, that of *connecting research with development teams* and aiming to *minimize or even eliminate the traditional technology transfer process* [2]. Google has released numerous products based on machine learning technologies using the same technologies that researchers have used for development. This type of high speed technology transfer and integration of research and development teams are two areas that the hardware design industry has struggled on achieving as highlighted in [11, 15].

RISC-V is another eloquent example of the success of the aforementioned open dataset and code trend [16]. Freed from the previous restrictions around proprietary licenses to the instruction set architecture (ISA), legions of designers have contributed to defining novel implementations and have built on top of each others' work. This has also led to direct technology transfer from the research space to industry adoption at an outstanding pace. With a common base of openness, support for the RISC-V architecture in open source software has also happened with significantly lower friction.

Both the machine learning and RISC-V examples also demonstrate that all aspects of research are increasingly software development connected and this is reflected in our approach to the roadblocks we discovered. In 2016 there were 800,000 software engineers in the USA compared with only 70,000 hardware engineers, and in the next 10 years software engineers are expected to increase by 30% compared with only 5.4% for hardware engineers [17]. It is clear that software and software engineers will need to play an important role in hardware design going forward. The authors of this paper have divided the hardware design into three major components each with overlapping and interrelated fields of research shown in Figure 3. Parallels can be drawn between EDA tooling and TensorFlow, PDK Data and ML datasets, and IP & Libraries as ML fields of research. A thriving ecosystem needs all three areas to participate to get significant change to happen. Roadblocks in

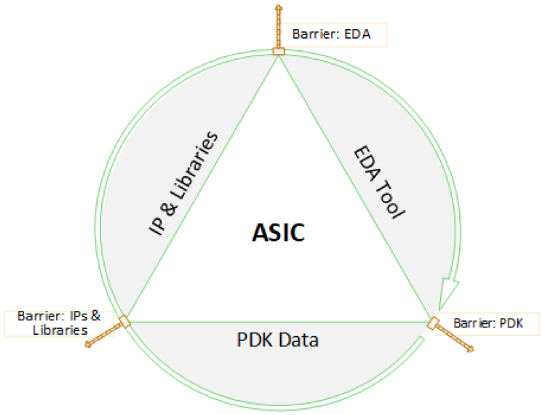

Figure 3: PDK, EDA and IP libraries are the three main components of a hardware design ecosystem.

each of these components have been discovered and efforts made to remove them.

Section 3 covers the effort to release an manufacturable open source PDK. In a very similar way to the lack of high quality, openly sharable datasets hampering the advancement of the ML space, the lack of an open source PDK has hampered all other aspects.

Sections 4 and 5 covers the efforts to improve the EDA tooling ecosystem, an effort which shares many parallels with Google's TensorFlow but is primarily focused on collaborating and enhancing existing projects like OpenROAD.

Section 6 covers the efforts to fix key holes in the IP and libraries and enable new approaches to generating IP and libraries. This again focuses on collaborating and enhancing existing efforts such as OpenRAM, BAG and FASoC.

3 ROADBLOCK - MANUFACTURABLE PROCESS DESIGN KIT

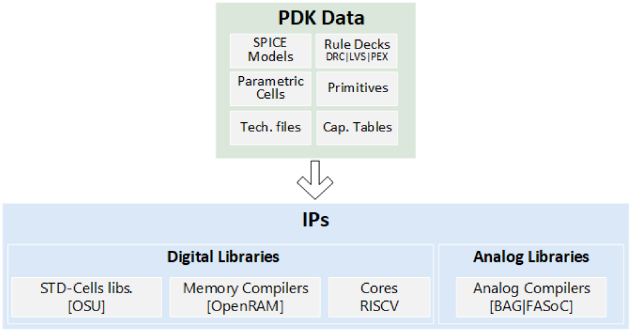

Figure 4: Open Sourced Process Design Kit Data is the first layer of enablement and allows producing building block libraries necessary for ASIC design.

In hardware design, the basic building blocks are provided by the foundry in the process design kit (PDK), as shown in Figure 4. The PDK data is essential to all aspects of hardware design research. The PDK data being restrictively licensed means that even with available open source RTL and EDA tools, the final taped-out

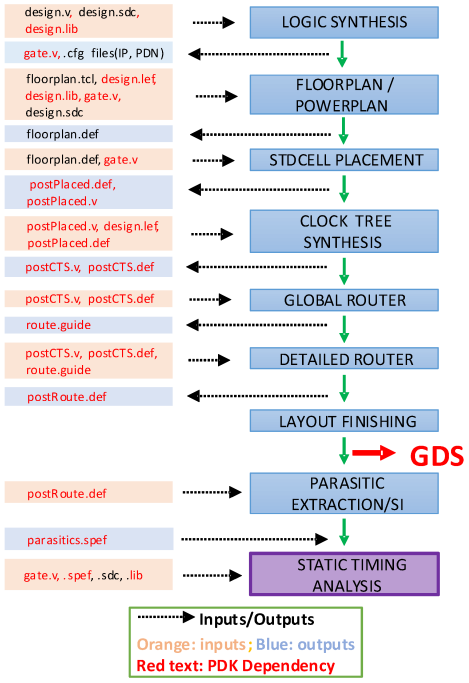

Figure 5: PDK impact shown in red with reference to the stages of a typical EDA tooling flow.

design isn't publicly available [18]. The PDK data is essential to almost all steps required to convert input design specifications into a manufacturable design, as shown in Figure 5. Figure 5 details how the data from electronic design automation (EDA) tooling outputs is tightly coupled with the PDK, even when the employed IP and RTL libraries are open. The absence of open data thus directly affects the development of both the open source RTL and the EDA tooling. EDA software can only be tested and evaluated for correctness in this context. Thus, the lack of PDK data means that open source EDA tools are not able to have public continuous integration (CI) systems; this considerably hinders productivity and dramatically reduces the number of people able to contribute to the tooling. Modern software development practices make heavy use of continuous integration, and hence this limitation drives away many of the best software engineers.

While the invention of languages like Verilog and VHDL have allowed digital RTL to be developed independently from a PDK, it remains impossible to verify that RTL changes are actually improving the result without linking the RTL to a standard-cell library and thus a PDK. In fact, changes to the RTL can have a huge impact on the final results in non-obvious ways [19]. On the other side, due to the unavailability of an equivalent high level description languange for analog design, currently the development of analog IPs without access to the PDK data is nearly impossible. These aspects further highlight the importance of fully open source code and datasets to a growing and thriving ecosystem, as highlighted in Section 6. This is only achievable if the fundamental building blocks are not held as closely guarded secrets.

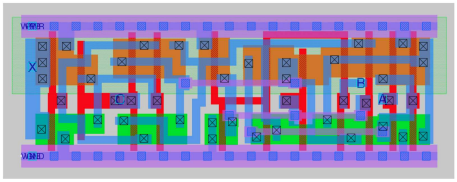

Figure 6: Standard-Cell Layout from SKY130 Libraries visualized in Magic Layout Editor

After extensive evaluation, it was identified that no open manufacturable open PDK existed. What PDKs that do exist do not follow the open guidelines and do not actually match real manufacturable data. This was again highlighted by Andrew Kahng in [11, 15]. To address this limitation, Google has been working with SkyWater Technology Foundry to release openly their 130nm process node (SKY130) [20]. While this node is 20 years old it was identified as being extremely affordable and still in heavy production use due to having good properties for analog design. The SKY130 node takes 130nm further by including an extremely diverse selection of features and deluxe options available by default, such as local interconnect, Silicon-Oxide-Nitride-Oxide-Silicon (SONOS) functionality, MiM capacitors, etc. enabling a wide range of research and development opportunities. SkyWater Technology Foundry also specializes in enabling production volume of even further customizations of this process, including the addition of specialized materials and devices such as Nb, Ge, V2O5 and Carbon Nanotubes. This aligns with the goal of increasing the high speed technology transfer from research into production volumes.

The SKY130 PDK is available on GitHub, free from any Non-Disclosure Agreement (NDA). A permissive Apache 2.0 license is used and the content of the platform is treated similarly to any software. As shown in Table 1, the SKY130 PDK supports all necessary files for both open source and proprietary design flows; primitive cell libraries and models for creating analog designs as well as SPICE models; and rule decks to verify design rule checks (DRC), layout versus schematic (LVS) and sign-off parasitic extraction. Hardware designers along with software developers are thus emancipated from traditional restrictions and are able to share, reproduce, and incrementally build on top of previous projects new IPs and tools.

	Rules Decks	Behavioral Models	Spice models	Pcells	Technology Files
PDK Data	DRC LVS PEX	Analog Digital	Monte Carlo Process corners	Transistors Capacitors Inductors Parasitics	LEF Cap. tables Layer Mapping

Table 1: SKY130 Process Design Kit Content.

The SKY130 PDK comes with an initial set of standard-cell libraries shown in Table 2 that covers various flavors such low-power, multi-speed, high-density and high-voltage. An example of a high-density standard-cell layout is shown in Figure 6. While the SKY130 PDK does provide the resources required to develop analog IP, including a customized SRAM and flash build spaces, it does not include any analog IP. How that issue is being tackled is covered in section 5.

To prove the PDK was complete enough for anyone to develop completely new standard cells rather than being restricted to use

	Digital Standard Cells	Memory Compilers	Build Space	IO & Periphery
Libraries	High-Density High|Medium|Low-Speed Low-Leakage High-Voltage Low-Power	OpenRAM	SRAM bitcells DRC Exceptions SONOS Flash bitcells	Digital & Analog IOs

Table 2: Libraries and Digital Design Space in SKY130.

only the foundry-provided cells, a grant was given to Oklahoma State University (OSU) to create and test a new set of standard cells.

Lastly, to further reduce the roadblock of getting access to the SKY130 technology, and to encourage the release of new RTL and IP specifically targeted at this process node, Google has also funded Efabless and SkyWater to create a totally free open source shuttle program [20] for fully open source designs and will cover multiple runs over two years. This program is open to any open source design created by an academic, hobbyist or commercial entity. Since all the IP on the shuttle runs will be available, multiple shuttle runs will help designers to build on top of successful results of previous shuttle runs. This is key since it is expected that the combination of a multitude of efforts would result in incremental improvements. As noted from the example of the release of machine learning datasets in the ML space, Google hopes that other foundry houses would become more comfortable in also releasing their PDK information into the open source domain.

4 ROADBLOCK - PLACE AND ROUTE TOOLING

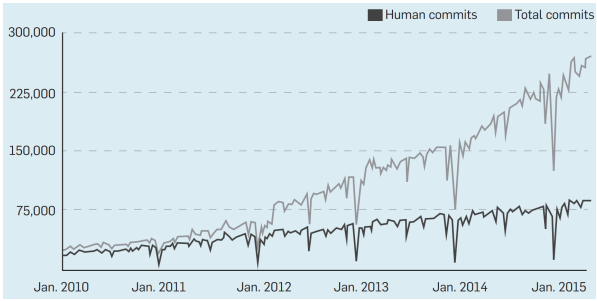

Figure 7: Commits per week within Google.

Another piece of the puzzle in creating a fully open and reproducible hardware design ecosystem is the tools used in the creation of the hardware design. While many academics have free access to the commonly used proprietary tooling from popular vendors, the tooling cannot be modified or adapted. Even when the tooling is used with open source RTL and open source PDK data, frequently the licenses for these tools also prevent distribution of the raw results and data produced, making verification even harder. The licenses provided to academics almost always include terms which prevent the direct technology transfer to companies interested in their application and prevents connecting with development teams at commercial companies which would otherwise have significant financial incentives to help improve the tooling. The harmful impact of this inability to freely modify and improve tooling has been previously demonstrated by Google in other similar areas. For example, in the GPU compiler space it was demonstrated that up to 51% performance improvement was possible [21].

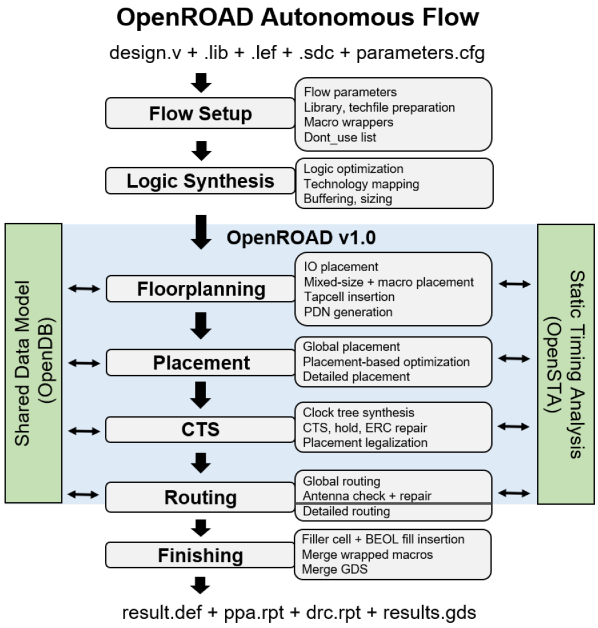

Figure 8: OpenROAD Autonomous Flow.

To further emphasize the critical nature of this roadblock, DARPA as part of its ERI initiative identified that in order to close the EDA performance gap and help deal with the end of Moore's Law the time needed to reach hardware design fabrication as well as to perform design iterations must be dramatically reduced [22]. Another group working on applying ML to ASIC place and route [23] also concluded that significantly reducing the runtime of EDA tools could dramatically enhance hardware design productivity.

The need to optimize runtime of the EDA tooling was identified as a major roadblock to DARPA's challenge to produce a "silicon compiler" as part of the IDEA program [24]. The aggressive 24 hour turnaround time with no-human-in-the-loop requires extensive changes to all parts of the EDA flow by multiple research teams [15]. This isn't trivial with the existing EDA tools status quo and led to the creation of the OpenROAD project, an effort led by Andrew Kahng [25]. The OpenROAD project builds a core open source base which can be easily customized and aims to tackle hardware problems by aggressively decomposing them into bite-sized subproblems and applying parallel and distributed search solutions.

Google has developed technologies like *borg* [26], *bigtable* [27] and *mapreduce* [28] to lead the field in enabling to split problems into multiple small chunks running them at scale on clusters of unreliable commodity compute hardware. Google has also demonstrated the ability to scale tooling beyond what was previously thought possible to increase the productivity of its software engineers [29]. These advancements have also allowed Google to continually advance its software ecosystem and increase developer productivity as shown by 7. Given how closely the OpenROAD project aligns with Google's core strengths and previous work it was easy for us to choose to support the OpenROAD project with our expertise, compute resources and supplemental research grants to the OpenROAD project.

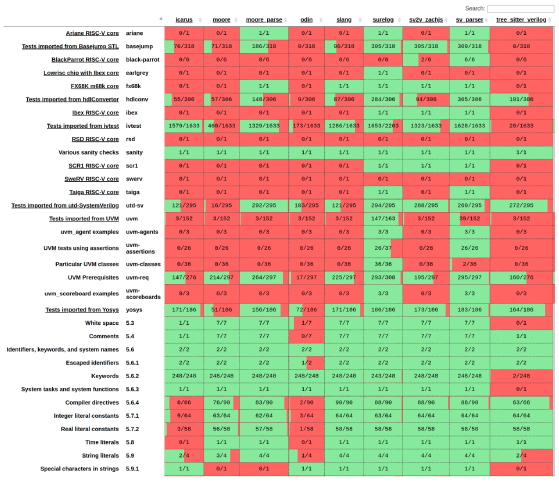

Figure 9: Screenshot of the sv-tests dashboard.

Over the past two years, major milestones have been reached by the OpenROAD project. Achieving a tapeout database using a 12nm finFET technology has been announced as part of the recent 2020 Electronic Resurgence Initiative (ERI) Summit; this entailed generating a DRC-clean result with over 535K placeable design elements including 49 SRAM macros with no human involvement.

Google has provided funding to efabless to collaborate with the OpenROAD project, to use the tooling combined with the open source PDK into the *OpenLANE* flow and to test this flow with existing open source RTL to tape out multiple open source designs that comprise the *striVe* family of SoCs – all with the target of completely DRC clean, no human involvement tapeouts. Three designs have already been produced, with more coming in the future [18].

5 ROADBLOCK - EXISTING SYSTEMVERILOG ECOSYSTEM

While the OpenROAD project is successfully able to place and route complex ASIC designs, Google identified another roadblock in the EDA tooling space that prevented a large section of the existing open IP ecosystem from being used with open source tooling. A significant portion of digital design, both proprietary and open source, is written in the SystemVerilog language which was called out as being poorly supported by open source tooling. To first verify this claim, Google in collaboration with Antmicro developed in the open a fully open source SystemVerilog compliance suite called sv-tests [30] under the SymbiFlow [31] project. The test suite follows the same philosophy as the CSS ACID tests which were developed to help drive browser CSS compliance closer together. A screenshot of the sv-tests dashboard can be seen in Figure 9. With the test suite well in progress Google was able to evaluate the compliance of various tooling both open source and proprietary; for the open source tooling we were able to publish complete compliance information whenever the test suite or tool was updated. The test suite has seen major contributions from a wide variety of different people and allowed us to discover work in and around SystemVerilog support that we had not discovered before. With

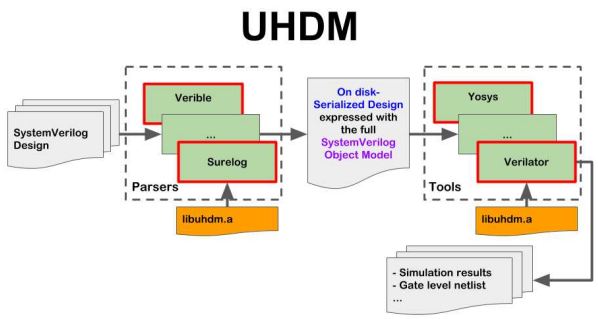

Figure 10: Block diagram of the UHDM architecture.

this information we discovered that our internal SystemVerilog linter and formatter called *Verible* had good compliance [32], and hence it was decided to release this as open source [33, 34]. Verible is used as a linter for a number of projects internally at Google as well as for the public OpenTitan project [35], and is starting to see adoption by other companies doing RTL development such as Western Digital.

Through the OpenROAD project we were also put in contact with a small company, Data Model Solutions, that had been developing its own ANTLR-based SystemVerilog parser called Surelog. With sv-tests able to accurately evaluate the quality of the parser, Google was able to fund this project to release SureLog under an open source license. Surelog is now available [36] and provides an extensive set of features including:

- one of the highest pass rates for almost all the sv-test suite tests;
- a multithreaded preprocessor and parser with file, module and packages splitting for multi-threaded compilation; and
- a comprehensive Python API enabling the creation of custom linting rules and other similar features.

Building on of the Surelog work, Google engineers are now working with Data Model Solutions and Antmicro to create the Universal Hardware Data Model (UHDM) project [34, 37] allowing the Surelog parser to be integrated into leading open source hardware tools such as Yosys and Verilator [38, 39], as shown in Figure 10.

6 ROADBLOCK - MISSING SECTIONS OF IP ECOSYSTEM

Open source IP and RTL has had a long history with online communities like OpenCores [40] having existed since 1999. Other groups at Google have already been making contributions to this space with projects like OpenTitan [35]. With digital designs being easily portable between various tools, process nodes and even between FPGA and ASIC technologies, the open source digital IP ecosystem has seen some progress. Recent successes such as RISC-V and significant decrease in the cost of FPGA-based platforms has led to a snowballing effect of open source digital design contributions. Other areas such as analog design have not been as fortunate and even basic open source analog IP blocks are nonexistent.

In many applications, analog IPs have a dramatic impact on the final ASIC design. Embedded SRAM, which is partly an analog system, makes up a significant portion of almost all modern ICs [41]. Power supply stability affects the switching frequency and power supply efficiency has a direct impact on power consumption [42, 43]. Based on the internet-of-things (IoT) market predictions [44], mixed-signal systems with *big A and small d* including software-defined radios [45] are expected to be the predominant type of ASIC design. Analog high-speed SERDES and IOs are fundamental to fulfill modern inter-IC data transfer needs [46]. Thus, the absence of open source analog IPs is clearly a major roadblock to replicability and application of research to real world designs, as highlighted, e.g., by DARPA in the IDEA initiative [24].

A clear but simple example is demonstrated by the Ravenna IC from efabless where while the rest of the IC was publishable open source IP, the analog blocks such as ADC, power supply and SRAM block remained proprietary and provided by the foundry. The striVe family of SoCs has also yet to reach feature parity to the Ravenna IC line due to the current lack of analog IP on the SKY130 platform. This lack of analog IP also reduces the ability to port existing high-quality FPGA based open source digital IP, like LiteDRAM [47] and the LiteX ecosystem [48], to the SKY130 platform.

As embedded SRAM plays such an important part of most modern ASIC designs, we identified the need for a fully open source memory compiler to help enable people to take full advantage of the SKY130 platform and especially the SRAM build space provided. There are numerous open source memory generators but of the projects investigated by Google, all but OpenRAM required closed source tools for some part of the generation process. OpenRAM supports not only the closed source proprietary tooling but also fully open source replacements [49] and was therefore selected for a grant to build a memory generator for the SKY130 platform. The first test SRAM design has been sent out for manufacturing and, as can be seen in Figure 11, use of the OpenRAM SRAM reduces the size of the striVe design by a significant amount. This is still only the first memory generated and is only taking advantage of a small number of the customized cells in the SKY130 build space. Further optimization is planned by the OpenRAM team.

With the [50] and shuttle program for open source IP [20] enabling open source analog IP to be created, we expect to see a lot more activity in this area. Due to the increasing importance of analog IP, Google has provided funding to enable two different analog generation approaches targeting the SKY130 platform.

BAG: Berkeley Analog Generator [51] is a tool designed to make it easier for designers to write Python-based generators which are portable across multiple technology nodes. To do this a base level primitive library needs to be created which the generators are then built on top of. The BAG generator approach has gone through three iterations along with the creation of a startup called Blue Cheetah dedicated to commercializing the tools [52]. Blue Cheetah demonstrated success using the BAG technology by creating an open source Advanced Interface Bus (AIB) version 2.0 targeting Intel's chiplet-style manufacturing [53]. Google contracted Blue Cheetah to create the required fundamental primitives for BAG which enable anyone to use the BAG technology to generate analog IP for the SKY130 platform. BAG however still requires the use of proprietary tools in a number of stages which has prevented its wide adoption by the community.

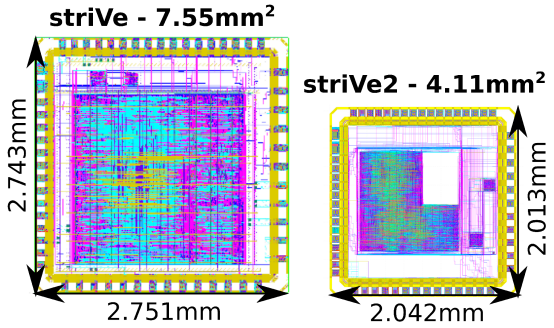

Figure 11: To-scale view of the striVe and striVe 2 test-chips showing size reduction through usage of OpenRAM generated memory block.

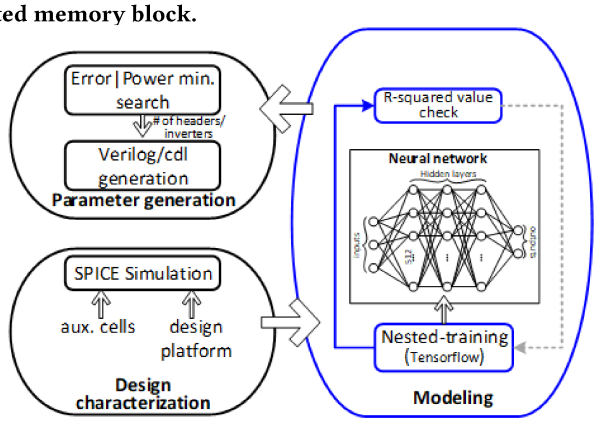

Figure 12: An automated neural network flow using nested training and parameter search based on minimum inaccuracy or power.

FASoC: Fully Autonomous System-on-Chip (FASoC) [54] tool is a DARPA-funded project within the IDEA program [24]. A main objective is to address the need for analog compilers. While the digital design flow has long been extensively automated, most analog automation flows are still extremely reliant on manual design. This lack of analog automation tools leads to long design cycles and costs, which is a major hurdle in the realization of fully autonomous design. Therefore, the realization of silicon compilers that include automated analog blocks generation, while a major challenge, could potentially alleviate the cumbersome time-to-market costs of meticulous analog design. The FASoC effort [55] uses a cell-based analog design generation methodology to generate different analog and mixed-signal (AMS) blocks (i.e., PLL, LDO, DC-DC converters, ADC, etc.). Key analog functions for each block-level generator are identified and called as *auxiliary cells*. Therefore, by building a library with all the required auxiliary cells in a specific design kit, it is possible to use these cells in the standard digital synthesis flow, allowing the automation of analog circuit design. This project is supported to port the analog generators to the SkyWater technology as well as to swap its current closed source EDA tooling to an open source EDA flow.

A temperature sensor has been ported using a variant of the structure defined in [56]. The generator takes user input specifications such as Temperature Range (T_{RANGE}), Power, Inaccuracy Constraint and Optimization Strategy. During the initial steps of the proposed autonomous flow, the tool detects whether a modeling file has been already provided. If not already present, a new file is automatically generated by running SPICE simulations sweeping the internal parameters of the sensor as part of the sensor design space exploration. The results are then employed to train a Bayesian neural network (BNN) model, as shown in Figure 12. Doing this considerably reduces the simulation runtime, and the generator is able to predict and determine the configuration of design parameters that best matches a desired input specification. A hierarchically structured topology is employed to enable application-specific optimizations such as low power or high performance. The proposed generator is able to describe each of the analog functionalities in Verilog and is thus compliant with conventional digital flows. This last feature is highly attractive since any analog block can be treated as a soft macro and undergo place-and-route using minimal digital flow constraints, as shown in the example of Figure 13.

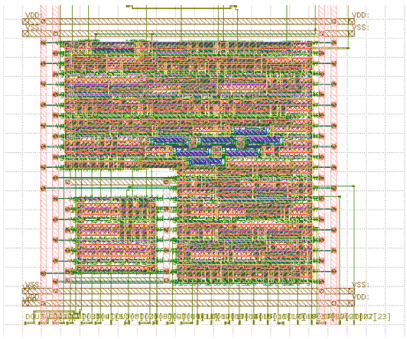

Figure 13: Top-view of a temperature sensor automated design generation [FASoC] in SKY130.

7 CONCLUSIONS

With the continuous explosion of compute requirements, Google is exploring multiple avenues to enable hardware design to keep up with ever-increasing demand. One such effort, led by Tim Ansell's group, is focused on enabling a rapidly growing hardware development ecosystem by using a targeted approach of collaboration with researchers and industry to remove key roadblocks preventing an open and collaborative research ecosystem from being built using the recognized best practices of science.

Following Google Research's philosophy of *sharing knowledge accelerates progress for everyone* and *connecting research with development teams*, and following the successful route demonstrated by Google's efforts around TensorFlow in the machine learning space, Google has released new open source code and datasets and is *investing in research and development of open-source, usable tools and infrastructure that support reproducibility for a broad range of studies across different domains in a seamless fashion.*

The efforts have covered a wide range of roadblocks in the three fundamental areas required to make hardware designs 3 including:

- the open source release of a manufacturable PDK [50] and funding OSU to verify usability of the PDK for standard-cell creation;

- contributing to tooling including collaboration with the OpenROAD project [25] and development and open source release of System Verilog related tools [32]; and
- helping to fund core IP such as OpenRAM [49] and helping to enable promising interesting new analog generators such as BAG [51] and FASoC [55].

The authors have already been amazed by what has happened so far and look forward to seeing more interesting results from these efforts.

ACKNOWLEDGMENTS

We would like to thank the OpenROAD Team members for their invaluable help in preparing and reviewing contents of this paper. We also thank Danny Berlin and Parthasarathy Ranganathan for their advice and support. We are thankful to the many collaborators and members of the open source community whose efforts led to the works reviewed here.

REFERENCES

[1] Nathan Myhrvold. The future of software, the software industry, and windows '47. In *ACM97: The Next 50 Years of Computing*, ACM '97, New York, NY, USA, 1997. Association for Computing Machinery.
[2] A. Spector et al. Google's hybrid approach to research. *Commun. ACM*, 55(7): 34–37, July 2012.
[3] L. Hatton et al. The long-term growth rate of evolving software: Empirical results and implications. *Journal of Software: Evolution and Process*, 29(5):e1847, 2017.
[4] A. Lottarini et al. Vbench: Benchmarking video transcoding in the cloud. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '18, page 797–809, New York, NY, USA, 2018. Association for Computing Machinery.
[5] Neil Thompson. The economic impact of moore's law: Evidence from when it faltered. *SSRN Electronic Journal*, 01 2017.
[6] N. P. Jouppi et al. In-datacenter performance analysis of a tensor processing unit. 2017. URL https://arxiv.org/pdf/1704.04760.pdf.
[7] Andrew B. Kahng. Reducing time and effort in ic implementation: A roadmap of challenges and solutions. In *Proceedings of the 55th Annual Design Automation Conference*, DAC '18, New York, NY, USA, 2018. Association for Computing Machinery.
[8] Roger D. Peng. Reproducible research in computational science. *Science*, 334 (6060):1226–1227, 2011.
[9] National Academies of Sciences Engineering and Medicine. *Reproducibility and Replicability in Science*. The National Academies Press, Washington, DC, 2019. URL https://www.nap.edu/catalog/25303/reproducibility-and-replicability-in-science.
[10] Victoria Stodden. Enabling reproducible research: Open licensing for scientific innovation. *Science*, 13, 03 2009.
[11] Andrew B. Kahng. Looking into the mirror of open source. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2019.
[12] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.
[13] Google. Tensorflow: Open source machine learning, 2015. URL https://youtu.be/oZikw5k_2FM.
[14] Y. Duan et al. Artificial intelligence for decision making in the era of big data - evolution, challenges and research agenda. *International Journal of Information Management*, 48:63 – 71, 2019. URL http://www.sciencedirect.com/science/article/pii/S0268401219300581.
[15] Andrew B. Kahng. Open-source eda: If we build it, who will come? In *Proc. 28th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, 2020.
[16] S. Hoover. Unleashing the potential of open-source silicon. Invited talk, VSDOpen Conference, 2019.
[17] Sokanu Interactive. The job market for computer hardware engineers in the united states, 2020. URL https://www.careerexplorer.com/careers/computer-hardware-engineer/job-markethttps://www.careerexplorer.com/careers/software-engineer/job-market.
[18] R. T. Edwards et al. Real silicon using open source eda. *IEEE Design & Test*, 2020.
[19] P. Davide Schiavone et al. Slow and steady wins the race? a comparison of ultra-low-power risc-v cores for internet-of-things applications. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–8, 2017.
[20] Tim Ansell. [fossi dial-up] tim ansell - skywater pdk: Fully open source manufacturable pdk for a 130nm process, 2020. URL https://youtu.be/EczW2IWdnOM.
[21] J. Wu et al. Gpucc - an open-source gpgpu compiler. In *Proceedings of the 2016 International Symposium on Code Generation and Optimization*, pages 105–116, New York, NY, 2016. URL http://dl.acm.org/citation.cfm?id=2854041.
[22] A. Olofsson. Silicon compilers - version 2.0, 2018. URL http://www.ispd.cc/slides/2018/k2.pdf.
[23] A. Mirhoseini et al. Chip placement with deep reinforcement learning, 2020.
[24] DARPA, 2020. URL https://www.darpa.mil/program/intelligent-design-of-electronic-assets.
[25] T. Ajayi et al. Toward an open-source digital flow: First learnings from the openroad project. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, New York, NY, USA, 2019. Association for Computing Machinery.
[26] A. Verma et al. Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, New York, NY, USA, 2015. Association for Computing Machinery.
[27] F. Chang et al. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26(2), June 2008.
[28] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: A flexible data processing tool. *Commun. ACM*, 53(1):72–77, January 2010.
[29] Rachel Potvin and Josh Levenberg. Why google stores billions of lines of code in a single repository. *Commun. ACM*, 59(7):78–87, June 2016.
[30] SymbiFlow, 2020. URL https://github.com/SymbiFlow/sv-tests.
[31] SymbiFlow, 2020. URL https://symbiflow.github.io.
[32] David Fang. Systemverilog productivity tools, 2018. URL https://woset-workshop.github.io/PDFs/2018/a10.pdf.
[33] Google, 2020. URL https://github.com/google/verible.
[34] Tim Ansell. Google's slides for eri posh + idea integration exercise, 2020. URL https://j.mp/eri20-goog.
[35] lowRISC, 2020. URL https://opentitan.org.
[36] Data Model Solution, 2020. URL https://github.com/alainmarcel/Surelog.
[37] Data Model Solution, 2020. URL https://github.com/alainmarcel/UHDM.
[38] D. Shah et al. Yosys+nextpnr: An open source framework from verilog to bitstream for commercial fpgas. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 1–4, 2019.
[39] Veripool, 2020. URL https://www.veripool.org/wiki/verilator.
[40] OpenCores, 2020. URL https://opencores.org/.
[41] M. Khayatzadeh et al. 17.3 a reconfigurable dual-port memory with error detection and correction in 28nm fdsoi. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 310–312, 2016.
[42] J. Lee et al. 19.2 a 6.4pj/cycle self-tuning cortex-m0 iot processor based on leakage-ratio measurement for energy-optimal operation across wide-range pvt variation. In *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 314–315, 2019.
[43] M. Saligane et al. An adaptive body-biasing soc using in situ slack monitoring for runtime replica calibration. In *2018 IEEE Symposium on VLSI Circuits*, pages 63–64, 2018.
[44] C. Perera et al. A survey on internet of things from industrial market perspective. *IEEE Access*, 2:1660–1679, 2014.
[45] M. Condoluci et al. Enabling the iot machine age with 5g: Machine-type multicast services for innovative real-time applications. *IEEE Access*, 4:5555–5569, 2016.
[46] S. Abdennadher et al. At speed testing challenges and solutions for 56gbps and 112gbps pam4 serdes. In *2020 IEEE Latin-American Test Symposium (LATS)*, pages 1–5, 2020.
[47] John Sully. Litedram vs mig - a viable alternative. ORCONF, 2018. URL https://j.mp/or18-litedram.
[48] F. Kermarrec et al. Litex: an open-source soc builder and library based on migen python dsl, 2020.
[49] M. R. Guthaus et al. Openram: An open-source memory compiler. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–6, 2016.
[50] Google, 2020. URL https://github.com/google/skywater-pdk.
[51] J. Crossley et al. Bag: A designer-oriented integrated framework for the development of ams circuit generators. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 74–81, 2013.
[52] Inc. Blue Cheetah Analog Design. Blue cheetah analog design, 2020. URL https://www.bcanalog.com/.
[53] K. Settaluri and D. Kehlet. Chiplet integration: Tools, methodology, requirement, infrastructure. Tutorial 10 DAC, 2020.
[54] FASoC, 2020. URL https://github.com/idea-fasoc.
[55] T. Ajayi et al. An open-source framework for autonomous soc design with analog block generation. In *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SoC)*, 2020.
[56] M. Saligane et al. All-digital soc thermal sensor using on-chip high order temperature curvature correction. In *2015 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4, 2015.