A Mixed-Signal RISC-V Signal Analysis SoC Generator With a 16-nm FinFET Instance

Steven Bailey, *Member, IEEE*, Paul Rigge, *Student Member, IEEE*, Jaeduk Han, *Member, IEEE*, Richard Lin, Eric Y. Chang, Howard Mao, Zhongkai Wang, *Student Member, IEEE*, Chick Markley, Adam M. Izraelevitz, *Student Member, IEEE*, Angie Wang, *Member, IEEE*, Nathan Narevsky, *Member, IEEE*, Woorham Bae, *Member, IEEE*, Steve Shauck, Sergio Montano, Justin Norsworthy, Munir Razzaque, *Member, IEEE*, Wen Hau Ma, Akalu Lentiro, Matthew Doerflein, Darin Heckendorn, Jim McGrath, *Senior Member, IEEE*, Franco DeSeta, Ronen Shoham, Mike Stellfox, Mark Snowden, Joseph Cole, Daniel R. Fuhrman, Brian Richards, *Member, IEEE*, Jonathan Bachrach, *Member, IEEE*, Elad Alon, *Fellow, IEEE*, and Borivoje Nikolić, *Fellow, IEEE*

***Abstract*— This paper demonstrates a signal analysis system-on-chip (SoC) consisting of a general-purpose RISC-V core with vector extensions and a fixed-function signal-processing accelerator. Both the application core and the accelerators are design instances produced through an agile design-space exploration process by generators that allow for a wide range of parameter configurations. The signal processing chain consists of generated instances of a time-interleaved analog-to-digital converter (ADC) followed by a digital tuner, a finite-impulse response (FIR) filter, a polyphase filter, and a fast Fourier transform (FFT) all connected to the five-stage, in-order RISC-V Rocket processor via an AXI4 bus. The generator-based design methodology is detailed, along with the agile design process of producing the fabricated design instance. The 5 mm × 5 mm chip is implemented in a 16-nm FinFET process and operates at 410 MHz at 750 mV drawing 600 mW. Presented applications show coupled functionality of the application processor and accelerator performing spectrometry and radar receive processing, and a comparison with other state-of-the-art application-specific integrated circuits (ASICs) proves that generators can produce performance-competitive designs.**

Manuscript received January 29, 2019; revised May 5, 2019; accepted June 6, 2019. Date of publication July 17, 2019; date of current version September 24, 2019. This paper was approved by Guest Editor Chen-Hao Chang. This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) through the Circuit Realization at Faster Timescales (CRAFT) Program under Grant HR0011-16-C0052, in part by the Berkeley Wireless Research Center (BWRC), and in part by the Berkeley Agile Design of Efficient Processing Technologies (ADEPT) lab (Intel iSTC). The views, opinions, and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. *(Corresponding author: Steven Bailey.)*

S. Bailey, P. Rigge, J. Han, R. Lin, E. Y. Chang, H. Mao, Z. Wang, C. Markley, A. M. Izraelevitz, A. Wang, N. Narevsky, W. Bae, B. Richards, J. Bachrach, E. Alon, and B. Nikolić are with the Electrical Engineering and Computer Sciences (EECS) Department, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: stevo.bailey@gmail.com).

S. Shauck, S. Montano, J. Norsworthy, M. Razzaque, W. H. Ma, A. Lentiro, and M. Doerflein are with Northrop Grumman Corporation, West Falls Church, VA 22042 USA.

D. Heckendorn, J. McGrath, F. DeSeta, R. Shoham, M. Stellfox, M. Snowden, J. Cole, and D. R. Fuhrman are with Cadence Design Systems, Inc., San Jose, CA 95134 USA.

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

Digital Object Identifier 10.1109/JSSC.2019.2924090

***Index Terms*— Agile hardware design, analog-to-digital converter (ADC), CMOS, fast Fourier transform, filter, FinFET, generators, radar signal processing, spectrometer.**

I. Introduction

EMBEDDED digital signal processing (DSP) systems require high performance and high energy efficiency, often necessitating application-specific integrated circuit (ASIC) implementations. However, designing custom chips imposes a burden on both the budget and the schedule of a project. Typically quoted non-recurring engineering (NRE) costs in modern process technologies are in the tens of millions of dollars. The complexity of design, verification, and validation takes many engineer years with high costs arising from engineering labor, prototyping, tools, and fabrication costs.

Design specifications frequently change because of new customer needs, physical design limitations discovered along the way, and bugs exposed through verification, resulting in multiple design iterations. Reuse in this paradigm is limited. Shims are used to repurpose hardened IP, which after several layers culminate in so much added complexity that restarting the design is necessary. Using design generators promotes the fastest response to change since new specifications only require changing generator parameters instead of hacking changes on top of finished designs. If the generator does not currently support the required parameters, modifying the generator imposes a one-time burden that may be reused for the same and future designs.

This paper demonstrates the design of a complex, domain-specific system-on-chip (SoC) by using a generator-based agile design methodology. A signal-processing accelerator and a general-purpose processor (GPP) were instantiated from a set of fully featured, open-source digital and analog generators [1] written in Chisel [2] and Berkeley Analog Generator (BAG) [3]. The agile design process establishes a functional yet incomplete system early in the design process, and subsequently adds features and improves the performance through a series of design iterations until the design's

completion [4]. Generators enable these iterations, since complexity is added in each iteration to the generator itself, and parameterization enables increasing the design complexity.

This paper is organized as follows. Section II explores the prior art of design using generators. Next, Section III presents a model of a generalized signal processing system. A generator-based framework supporting agile system development is given in Section IV, and this framework is used to design a novel radar processing system in Section V. The remaining sections discuss the results of this effort. Section VI details the implemented architecture, which required integration with IP as discussed in Section VI-A. The system was verified at multiple design levels, as presented in Section VII. Finally, Section VIII discusses the general testing results, and Section IX shows the system running real applications and compares them to other work. Section X concludes this paper.

II. Generator-Based Design

Signal analysis determines in real time frequency- and time-domain properties of waveforms for wireless, medical, and instrumentation applications. The application domains range from radar, where the signals give the position and motion of objects in the field of view, to radio astronomy, where signals are radiation from distant stars that carry valuable information. This analysis can be performed by using GPPs, field-programmable gate arrays (FPGAs), or fixed-function ASICs. Many space, government, and commercial signal processing systems require efficient ASIC designs to meet size, power, and area requirements. High-performance signal analyzers perform tens of trillions of operations per second, which is practical only with dedicated hardware.

Decoupled from the implementation platform, a signal analysis datapath involves the conversion of an analog signal to the digital domain, followed by a sequence of programmable filtering operations in the time and frequency domains, after which a decision is made about the signal being observed. The signal processing function is typically described by a dataflow model. Our design is also described as a dataflow model, partitioned between the GPP and a dedicated DSP accelerator. The fixed-function accelerator represents a dataflow graph, described as a Chisel generator.

Chisel is a domain-specific extension to the Scala programming language [2]. Chisel gives a hardware designer abstractions of primitive hardware components, such as registers, multiplexers, and wires. A designer in Chisel constructs a set of libraries whose classes represent those hardware primitives. Execution of a Chisel program generates a graph of such hardware components to provide a blueprint to fabricate hardware designs; the back-end of the Chisel compiler translates this graph of hardware components into a synthesizable Verilog representation that maps to standard emulation (Cadence Palladium), FPGA, or ASIC flows, as well as verification and integration collaterals, such as IP-XACT [5]. Chisel by itself does not raise the level of design abstraction above register-transfer level (RTL), but the software abstractions enabled by the expressiveness of the Scala host language allows for the design to be modular and highly parameterized,

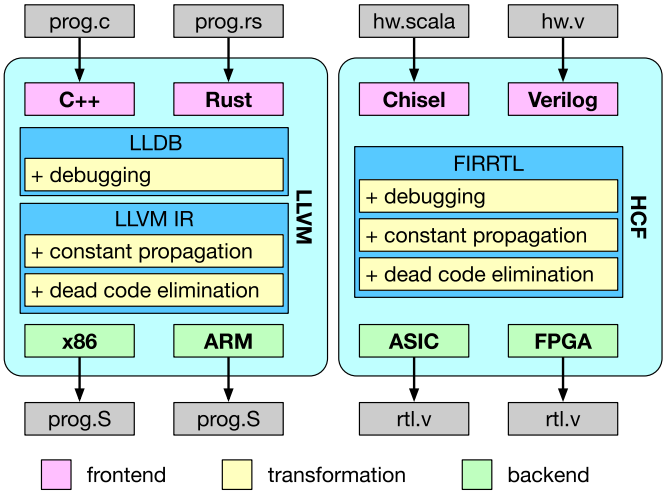


Fig. 1. Just like LLVM, the Chisel HCF interfaces transformations and optimizations with different front-end code bases and backend compilers [6].

thus enhancing reusability. Generators written in Chisel also preserve the designer's intent (i.e., control over the generated output), in contrast to high-level synthesis (HLS) tools, which usually provide limited control over output quality.

In the spirit of modern compilers that use an intermediate representation (IR), Chisel uses its own IR suitable for hardware, called flexible intermediate representation for RTL (FIRRTL) [7]. Typical software frameworks transform code written in a particular programming language into a compiler-specific IR, such as LLVM [8], where IR-to-IR transformations (optimizations) such as constant propagation or dead-code elimination modify the program's structure. Finally, a compiler backend converts the IR into code for the target instruction set architecture (ISA), e.g., Arm or x86. This structure of translating an input language into an IR enables reuse of transformations among multiple designs and languages. The Chisel hardware compiler framework (HCF) is similarly structured as shown in Fig. 1: Chisel and Verilog front ends translate designs into FIRRTL; transformation passes provide simplification, optimization, and instrumentation; and the resulting FIRRTL is simulated directly or passed to one of many Verilog backends tailored for simulators, emulators, FPGAs, or ASICs [7].

This approach sharply contrasts previous work on designing generators, where Perl and python scripts are wrapped around Verilog (or even inside Verilog, such as Genesis2 [9]). New domain-specific languages (DSLs) bridging the performance gap between HLS and Verilog, such as Chisel, enable powerful processor generators but come with a steep learning curve [10].

Complex mixed-signal interfaces are often a key performance-critical component of an SoC. Development of analog- and mixed-signal modules is a time-consuming process of refining design specifications through layout iterations and post-layout simulations. Analog design automation in the form of circuit synthesis and automatic layout generation has been a subject of exploration for many years but has had limited traction. BAG2 is a Python-based framework for process-portable development of analog

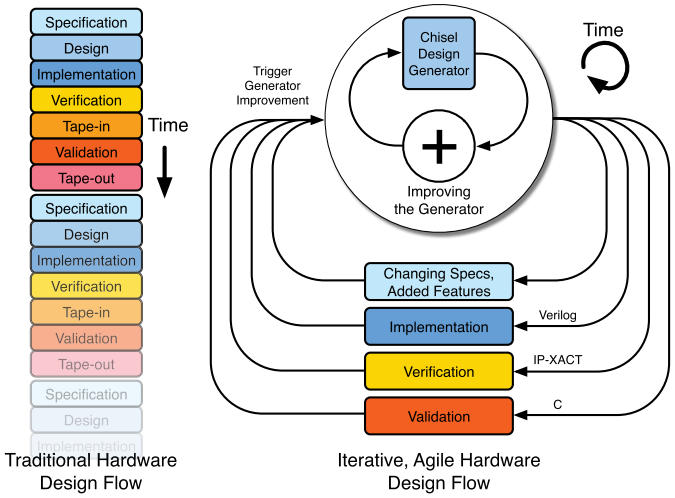


Fig. 2. Comparison of traditional, open-loop design flow with our agile, iterative design flow. Changes to specifications and bug fixes are incorporated into the generator. Design generators are portable across designs, so these enhancements persist.

generators [3]. The specification-to-verification framework encapsulates a schematic-generation application programming interface (API), a sizing routine (design script), and two layout generation engines, Laygo and XBase. The BAG2 framework has been used in this project to generate an analog-to-digital converter (ADC) and integrated with the digital components through the agile design flow.

Our version of the agile hardware flow involves iterative improvements of design generators and the addition of new design generators until the final design is reached, as shown in Fig. 2. Often DSP applications are flexible, and un-implemented processing elements (PEs) or features may be pushed to software running on a coupled GPP. As complications are exposed through changing specifications, implementation bottlenecks, verification tests, and validation checks, the design generator is refined and improved. These improvements are retained after tape-out, so future designs benefit from tool and generator reuse.

III. System Model

Abstractions that support composable, reusable components are critical for agile design. The system model here couples an application-specific, fixed-function accelerator with a GPP, allowing for both performance and flexibility. As the design matures and the generators improve, the scope of processing performed on the accelerator increases, while a portion of application processing can be relegated to the general-purpose machine. This model also allows for the reuse of existing CPU and compiler frameworks.

To support mapping an abstract signal processing dataflow graph into hardware, we model the fixed-function DSP processor as a set of PEs, or actors, which are combined to form a processing chain [11], as shown in Fig. 3. Each PE contains a custom DSP function of arbitrary complexity. Streaming inputs and outputs comprise the data interfaces for each PE, and an additional interface supports control. System-level connectivity and verification are simplified by using

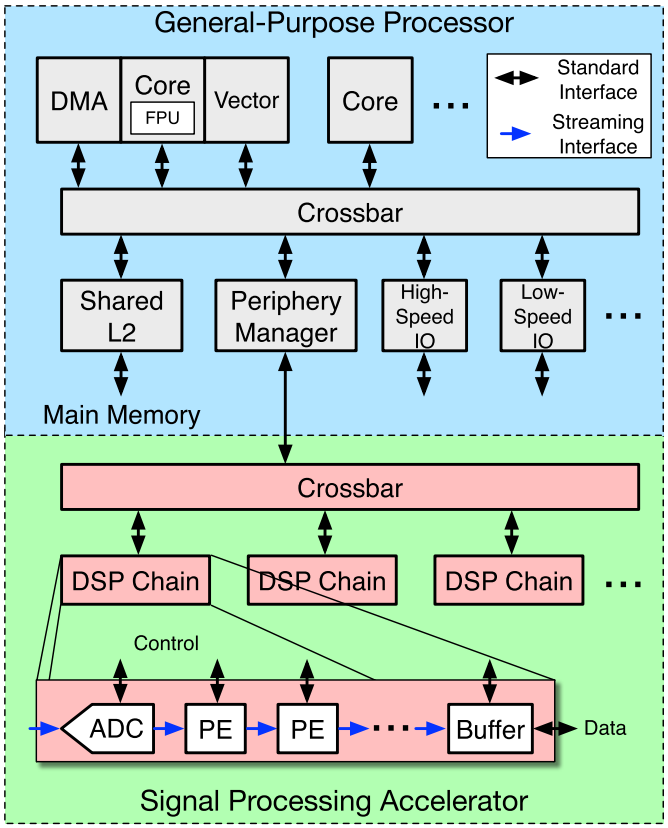


Fig. 3. DSP system abstracted as a separate GPP and signal processing accelerator. Composable streaming PEs comprise processing chains.

standardized data and control interfaces, such as streaming data input/output (IO) and memory-mapped control interfaces. A collection of PEs connected together comprises a DSP chain. On both ends of the DSP chain, there are special PEs, required to convert streaming data interfaces into some other format. At the input of the chain is typically an ADC, though a streaming memory interface could be used. At the output of the chain is typically a buffer storing streaming data for intermittent access by the GPP. Transmit chains terminated in a digital-to-analog converter (DAC) are also possible.

The full system model includes any number of DSP chains communicating with a processor through a crossbar, as shown in Fig. 3. The GPP abstracts the signal processing accelerator as a set of memory-mapped peripherals. It may feature a number of co-processors, such as direct memory access (DMA) and vector co-processors, as well as multiple cores, a cached memory model, and IO interfaces.

IV. DSP System Generator

By modeling a DSP processor in the way presented in Section III, we are able to construct a suite of tools that support streamlined design, verification, and programming of *any* system conforming to the model. The rationale for this framework comes from the agile methodology's emphasis on tools and generators [4]. Chosen framework features underscore the rest of the agile principles, such that small, collaborative design teams can focus on quickly constructing functional prototypes and easily respond to changing specifications.

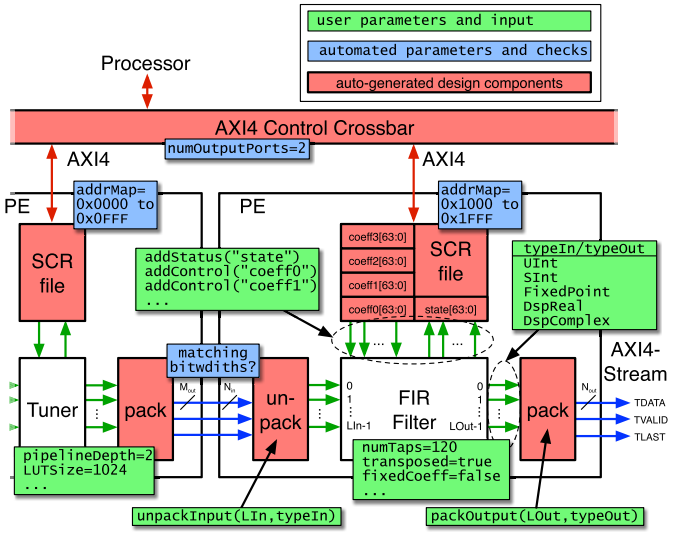


Fig. 4. DSP chain generator features. Processing element interfaces are semantically abstracted, leaving the designer free to choose desired datatypes.

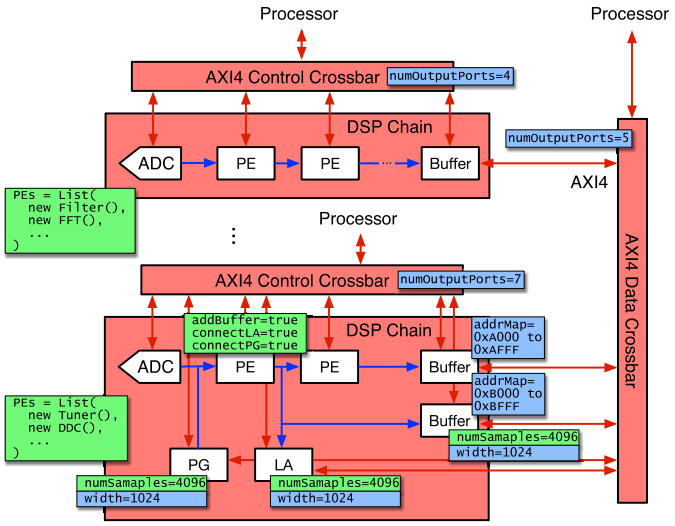


Fig. 5. Signal processing accelerator generator. The framework provides automatic component generators and checkers, allowing the design to scale easily. The colors scheme is the same as in Fig. 4.

A. Design Generation

The presented methodology automates the design of custom DSP co-processors. It provides a set of generators that handle the GPP, interconnect, and hooking up of the desired PEs into chains. The remaining user tasks are the creation and optimization of PEs. This section discusses the system generator's design features.

Designers create custom PEs or choose from a selection of library components. Each PE semantically unpacks its streaming interface into a set of parallel inputs, and packs its parallel outputs into a streaming interface, as shown in Fig. 4. This process is automatic, and the choices of datatype and number of parallel values are parameters. Currently, the supported datatypes include real and complex values, integers and fixed-point values, and signed and unsigned values. These numeric operations and datatypes come from the ACED hardware library [12] or are natively supported by Chisel. Aggregate datatypes such as vectors and multi-dimensional arrays are also supported natively. Custom datatypes and data structures may be easily added by expert Chisel users. For control, any number of inputs and outputs can be created and connected to generated registers in a status and control register (SCR) file. This SCR file is memory mapped for processor access. These two abstractions, the data and control shims, translate standardized interfaces, here chosen to be AXI4-Stream and AXI4, into more accessible formats for fast design creation. The choice of standard is decoupled from the design, so future frameworks could support more standards without needing to modify the existing processing element designs. With these interfacing elements and protocols defined, the methodology user can create a custom PE like any other Chisel design.

Chisel permits writing functional programming constructs to generate different hardwares, allowing a single parameterized generator to produce different PE instances. Constructing PEs in this way simplifies system design interconnect, design space exploration, and tailoring the design to an application domain. Input and output data types are parameters decided at compile time and easily changed. Implemented parameters for a library finite-impulse response (FIR) filter include the number of taps, microarchitecture (transposed or direct form I or II), fixed or variable coefficients, and pipelining style.

Since PEs contain the same interfaces, connecting them to each other and to the GPP is automated. Bitwidth checkers ensure every AXI4-Stream data connection between PEs matches, so invalid connections are caught during compilation. PEs may optionally be connected to independent buffers (named Stream-to-AXI4 Memories, or SAMs) and shared debugger structures, here memories acting as a logic analyzer (LA) and pattern generator (PG). The crossbar was separated into data and control crossbars to reduce fan-out. These crossbars automatically adjust to changing design sizes, such as when PEs are added or removed. Memory-mapped addresses are automatically calculated during compilation, adjusting as needed for different design sizes. Fig. 5 shows how DSP chains are parameterized and connected.

RISC-V Rocket was chosen as the GPP generator. Details on the processor generator can be found in [10], and the Chisel generator code is available on GitHub [13]. The Rocket generator meets the system model requirements, and it provides a TileLink periphery interface that may be converted to AXI4.

Chisel supports the inclusion of black boxes in the design. These black boxes can be Verilog or SystemVerilog RTL IP, or Verilog wrappers for analog IP. Thus, existing design components can be included without the need for porting to Chisel, reducing the NRE cost. Legacy PEs or smaller IP blocks may be directly added to the flow. Larger IP, such as full signal processing chains or other processors, may require manual integration if their interfaces and verification differ from the model presented.

B. Verification of Generated Instances

A set of readily interconnected PEs and their interface to the application processor requires a robust verification approach

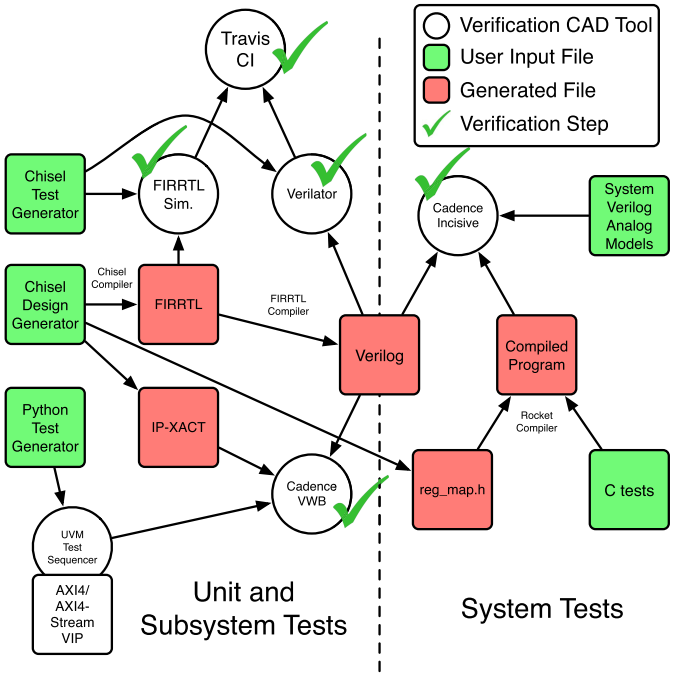


Fig. 6. Agile verification flow, split into unit and subsystem tests on the left and system tests on the right. VIP is provided. Check marks indicate where verification is performed.

to achieve an accelerated design closure. A combination of model features and methodology constructs allows for verification simplifications. Support for third-party verification tools permits multiple verification platforms as well. Writing testers as generators requires collaboration between the designer and a verification engineer. Verification is split into Chisel unit tests, unified verification methodology (UVM) test-bench generators in Cadence Verification Workbench (VWB), and system-level simulations in Verilog, SystemVerilog, and C. Post-implementation verification is performed on the Verilog RTL with Cadence Incisive. The separation of unit-level and system-level tests is shown in Fig. 6.

Chisel supports cycle-accurate testers in both FIRRTL and Verilator, an open-source Verilog simulator. For PE unit tests, tester constructs are written to automatically pack and unpack the input and output data, as well as convert between floating-point doubles in simulation to the specific data types of the PE. Thus, the testbench generators input a 1-D array of streaming data into the block and output a 1-D array of the result. Since the tester is embedded in Scala, existing numerical libraries, like Breeze, may be leveraged to build DSP testbench generators for PEs [14]. Also included are methods to read and write the SCR file, indexed by register name, so the user does not have to understand the details of AXI4 or know the generated address map. PEs are stored on GitHub, and any committed design change triggers a Travis continuous integration (CI) job that runs the PE tester.

To facilitate design reuse through sharing as well as verification with third-party tools, Chisel generates IP-XACT along with the Verilog for PEs and the crossbars. IP-XACT is an XML description of the interfaces, memory mapping, and parameters of a design component. Since PEs use AXI4-Stream and AXI4 interfaces only, existing verification IP (VIP) may be reused to verify that these interfaces and crossbars meet advanced microcontroller bus architecture (AMBA) specifications. We use VWB to run these checks. VWB also supports UVM testbenches, which we write as generators along with the PEs.

The Rocket CPU generator includes a C compiler and a host server to deliver programs to the processor. Thus, for both Verilog verification and application programming, C programs may be written. The SCR file and SAM memories automatically get assigned memory-mapped addresses, but accessing these values by name is preferred. Therefore, a C header file is generated from the address map. It provides a mapping between address names and values, and it also provides functions for reading and writing these registers. The Verilog testbench runs these C programs in a Verilog simulator, such as Cadence Incisive. It consumes the Chisel-generated Verilog along with SystemVerilog models of the analog components to run full system simulations.

Chisel maintains wire and register names between the Chisel code and generated Verilog when possible. When not possible, Chisel supports custom naming of generated wires and registers. Also, Chisel-generated Verilog contains comments to facilitate cross-referencing between generated wire and register names and Chisel wire and register names. These features simplify debugging post-synthesis and post-place-and-route simulations.

C. Physical Design of Generated Instances

Properly constraining clocks and power networks requires a good understanding of the underlying design, but generated designs may have changing constraints with changing parameters. Specifically, adding new DSP chains to the signal processing accelerator may add additional clocks. The Chisel compiler can generate design-constraint files to describe these clocks, which was used by a chip generated from this methodology [15]. Clock buffers, clock roots, and architectural clock gates are all added as Verilog black boxes in Chisel. This preserves naming and simplifies cross-referencing in manual constraint files.

Similarly, power management and design-for-test (DFT) instances are added as Verilog black boxes, with power and DFT constraint files generated by Chisel along with the design. Because the generator knows the design hierarchy, it generates constraints with proper hierarchical references to instances. These features are currently experimental and under development for the future use in mainstream Chisel.

V. Example Design Process

To evaluate the utility of the presented methodology, two chips were designed and fabricated through the generator flow. A radar analysis SoC [16] is described in this paper. The second chip, a sparse spectral analysis SoC [15], reuses the methodology and some of the Chisel and BAG generators to enable different system functionalities. The agile design methodology was exercised between two teams at two different locations. Each design iteration presented a *tape-in*, which was

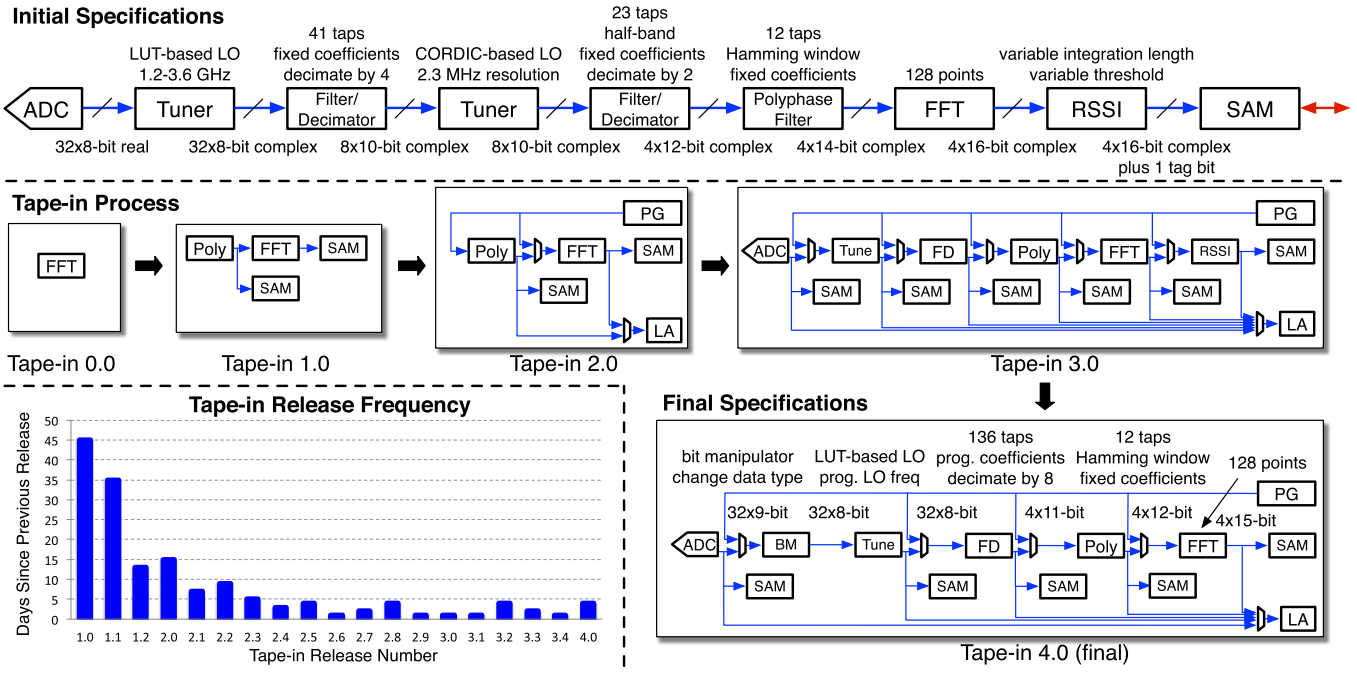


Fig. 7. Radar processor design flow. Top: initial customer specifications. Middle: agile design evolution. Bottom-left: timeline showing how the tape-in schedule converged. Version numbers increased when large design changes were folded in.

an RTL release that was ready for physical design and tape-out. When new features were added, such as more PEs, IP, or GPP accelerators, the design was verified then pushed through the physical design flow. Any feedback from verification triggered an improvement in the generator and a new tape-in release. As the handoff and feedback procedure matured, so did the frequency of tape-in releases, as shown in Fig. 7. This convergence suggests a coupling of process flows is possible, allowing for a near continuous evolution of design improvements, verification improvements, and physical design improvements, all living in the same generator framework.

Initial specifications included a high-bandwidth ADC with the deserialized output data feeding into the DSP chain. The chain consists of multiple programmable tuners, filters, and decimators followed by a polyphase filter and fast Fourier transform (FFT). Spectral bins are tagged when surpassing a threshold in the receive signal strength indicator (RSSI). Results are buffered in an SAM for readout. Fig. 7 shows more complete initial specifications, omitting control interfaces and the rest of the system for simplicity.

The need for tuner and filter generators is clear from the specification, since each copy will have different data types, parallelization, and coefficients. The input tuner accepts real data but outputs complex data, and later tuners all operate on the complex data. Bitwidth growth is desired for the filters and FFT. Therefore, instead of designing unique tuners, we designed one parameterized tuner generator that works in any context. Many parameters in the specification were adjusted, highly favoring generator development to the tuning of instances.

The tape-in process (Fig. 7) shows how the design evolved and drifted from the initial specifications. To simplify verification and the design, the tuner and filter stages were combined into one. Changing the filter design to 136 taps with programmable coefficients resulted in a massive physical design, yet implementation was feasible. The RSSI block was too large for achieved performance gains, so its function was performed in the application processor in the final design. Deleting only three lines of code resulted in a new DSP chain without the RSSI block, which was implemented in software instead. Memory-mapped addresses were recalculated, crossbars adjusted their size, and connections to the PG and LA were deleted, all automatically. Extra SAM blocks were added for more testability, and a new PE, a bit manipulator (BM), was added for data-type conversion. The result was a verified and validated signal analysis system that, despite being substantially different from the initial design, met the requirements.

The overall design took about 14 000 engineering hours, a significant reduction from the expected effort.[1]

VI. SoC Architecture

The rest of this paper discusses the features of the signal analysis SoC implementation [16]. Fig. 8 shows the system architecture. The SoC is divided into a GPP and a dedicated signal processing accelerator. Communication between the processor and the accelerator is handled through a memory-mapped IO manager.

A. Application Processor

The 64-bit single-issue in-order RISC-V Rocket CPU includes a single-precision/double-precision (SP/DP) floating-point unit (FPU) [10]. Provisions in the RISC-V ISA support

[1]One estimate for a chip of similar complexity is 14.4 months for a 10-person team, or about 25 000 h [18]. We also analyzed this design using Software Evaluation and Estimation of Resources for Integrated Circuits (SEER-IC), an industrial tool for estimating the time and material costs of custom ASICs. The presented design was estimated to take between 25 000 and 34 000 h in 28 nm (16 nm was unavailable at the time).

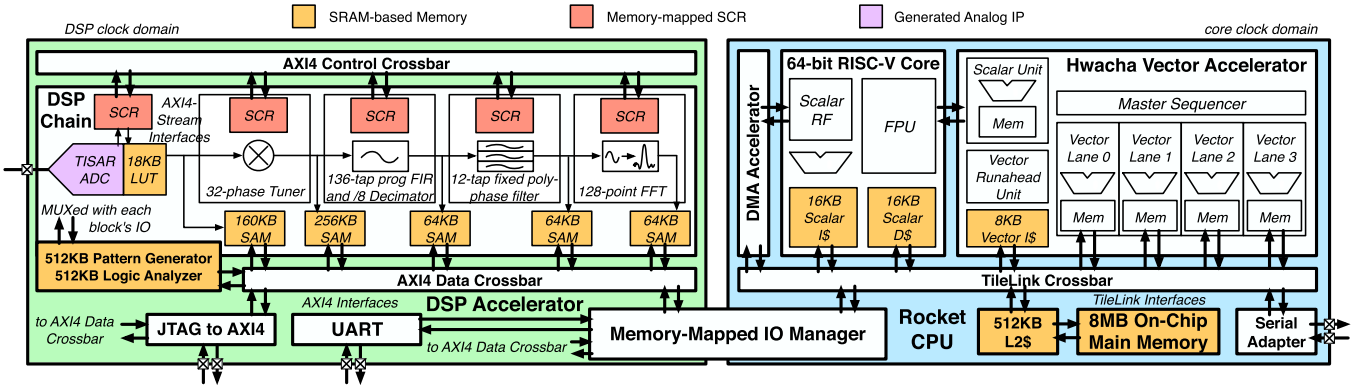


Fig. 8. Block diagram of the SoC architecture.

extensions to the instruction set, and this chip includes two accelerators using ISA extensions. A DMA accelerator extends the ISA through the Rocket Custom Coprocessor (RoCC) interface and offloads memory movement between the processors from the CPU. The four-lane high-performance Hwacha V4 vector accelerator implements a decoupled vector-fetch architecture and can perform compute-intensive parallel workloads not handled by the signal processing accelerator [19]. A serial adapter tethers the CPU to the FPGA host, which writes programs to the on-chip 8-MB main memory before booting the core. A custom clock receiver supplies the core clock, which clocks the CPU, accelerators, entire memory hierarchy, and TileLink interconnects.

B. Digital Signal Processing Accelerator

The DSP accelerator architecture is a streaming processor with PEs communicating through AXI4-Stream interfaces. The Chisel generator contains memory-mapped IO registers that take commands from either the CPU or a joint test action group (JTAG) debug port. Asynchronous FIFOs buffer data between the core clock, DSP clock, and JTAG clock domains. Separate AXI4 crossbars access SCRs and SAMs. For testing, a 512-KB PG and 512-KB LA allow direct access to inputs and outputs of individual PEs. A chain of PEs receives the data from a BAG-generated 8-bit time-interleaved successive approximation (TISAR) ADC with look-up table (LUT)-based static calibration, and it also provides a clock to the PEs and AXI4 crossbars. The universal asynchronous receiver–transmitter (UART) provides a backup interface into the RISC-V core, and the JTAG provides a backup interface into the accelerator.

C. Processing Elements

The selected PEs implement a signal analysis accelerator targeting spectral analysis or radar receive chain processing. Fig. 9 shows the parameterization of PEs in green atop the final implementation diagram. Sections VI-D–VI-H describe the individual elements and their generators. The ADC LUT outputs 9 bits for testing, so a custom BM PE truncates this to 8 bits. The next two PEs comprise a digital downconverter (DDC). A 32-entry LUT-based digital tuner mixes the input signal with a complex sinusoid, and a fully programmable 136-tap complex FIR filter shapes the signal and decimates it by 8. A 12-tap fixed-function polyphase filter multiplies the time-series data by a sinc function to window each FFT bin and reduces frequency-domain spectral leakage. A 128-point radix-2 FFT, comprised of 32-point biplex pipelined FFTs and a 4-point direct-form FFT, produces the complex spectrum output. The chain generator supports arbitrary ordering and duplication of PEs, so the chosen arrangement of PEs represents just one possible DSP accelerator configuration. The presented PE components are not architecturally or algorithmically new, but their novelty lies in their flexible parameterization, the scope of which is often unavailable in Verilog or SystemVerilog designs.

D. Bit Manipulator

The BM simply passes the input data through to the output. However, because the input and output datatypes may be distinct, automatic datatype conversion provides the utility of this block. For example, this chip used the BM to truncate the 9-bit ADC calibration output into 8 bits. Also, as all PEs support connections to the LA, PG, and a SAM, including a BM without any data-level modifications allows the designer to place these DFT connections anywhere. The accelerator includes two BMs right at the beginning of the chain. The first one connects the 9-bit ADC calibration RAM outputs to a SAM and the LA for direct monitoring of calibrated ADC data. The second block truncates these 9 bits into 8, as the ADC produces only eight real bits.

E. Tuner

The tuner processing element implements the first part of a digital downconversion. The inputs can be either complex or real. If the inputs are real, they are mapped to the real part of complex values before being multiplied by the mixer coefficients. The coefficients and outputs are always complex.

In the current PE generator version, the phase values are restricted to being one of $(2\pi/N)$ phases where N is defined by a parameter. This parameter must be an integer multiple of the number of lanes. The SCR file allows control of a multiplier factor, k, which specifies to use phases 0, $(2\pi k/N)$, $2 \times (2\pi k/N)$, $3 \times (2\pi k/N)$, and so on. This sequence will inherently wrap around and repeat after at most N phases and thus each input lane will always use the same value once k is chosen. Fig. 9 contains a block diagram for this tuner

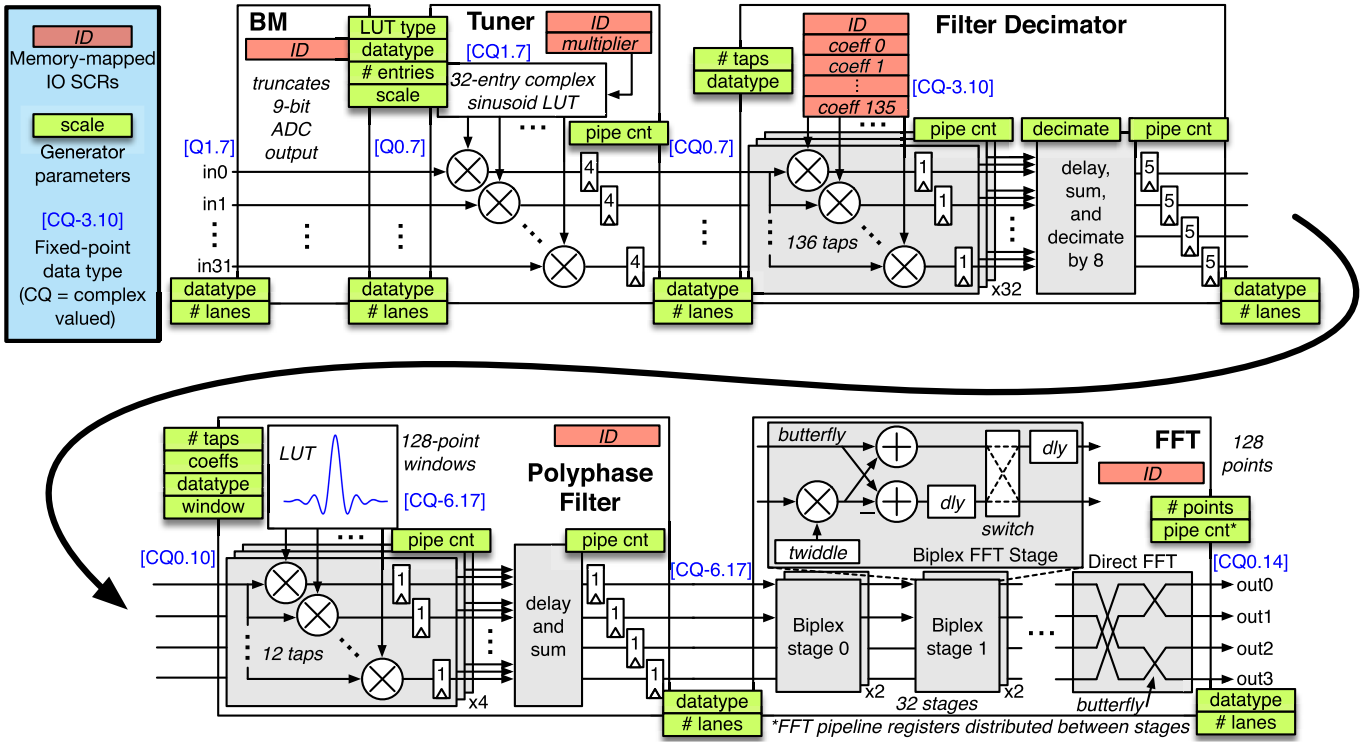


Fig. 9. Detailed diagram of the PEs in the DSP accelerator. Red boxes indicate memory-mapped IO SCRs. Green overlays show generator parameters. Blue text gives fixed-point data type parameters chosen. CQ = complex fixed-point number.

configuration. Note that this example uses mixed data types, so the input is real-valued but the output and coefficients are complex.

F. Decimating Filter

The filter PE implements an FIR filter, with decimation done by dividing down the number of parallel outputs. The user can choose any number of taps and parallel inputs. Multiplication is performed first, followed by summation. The number of outputs must be an integer divisor of the number of inputs. Tap coefficients are programmable through the SCR file and can be a distinct data type from the input and output. Pipeline registers and decimation are handled at the output. A highly parallel implementation will benefit from pipeline registers and retiming within synthesis tools. The processing delay parameter lets the user specify an explicit group delay. This value determines the number of pipeline registers applied to the TLAST and TVALID signals between the input and output in the AXI4-Stream protocol. Fig. 9 contains a simplified block diagram of this filter decimator.

G. Polyphase Filter

The polyphase filterbank (PFB) is based on the CASPER design [20]. In Chisel, the design is split into parallel lanes. Each parallel lane implements an FIR filter in transposed form, using Chisel Mems for delays, which may be mapped to SRAMs. Pipelining is split into registers placed at the output of each multiplier and registers placed at the PE output. The number of taps is an independent parameter but each lane has the same number of taps. When specifying a configuration, the tap datatype can be distinct from the input and output datatypes, but a conversion function must be supplied. Similar to the decimating filter, the processing delay parameter specifies a custom group delay for pipelining the TLAST and TVALID signals.

H. Fourier Transform

The FFT supports any power of two sizes of 4 or greater ($n \geq 4$). The input rate may be divided down, resulting in a number of parallel input lanes different from the FFT size. But the input lanes (p) must be a power of 2, greater than or equal to 2, but less than or equal to the FFT size. Twiddle factors are hard-coded and automatically calculated. The transform is split into pipelined biplex FFTs and a direct-form FFT to multiplex the logic for large FFT sizes [21]. When the number of parallel inputs equals the FFT size, a simple, direct-form, streaming FFT is used. Pipeline registers are placed at the output of butterflies, evenly distributed among biplex and direct-form stages based on the pipeline depth parameter.

When the input is serialized, the FFT may have fewer input lanes than the size of the FFT. In this case, the inputs are assumed to arrive in time order, time-multiplexed on the same wires. To accommodate this time multiplexing, the FFT architecture changes. Pipelined biplex FFTs are inserted before the direct form FFT. These FFTs efficiently reuse both computer hardware and memories to calculate the FFT at a slower clock rate but higher latency [22]. The input is assumed to be pipelined by the previous blocks, and the output of the biplex has shift registers already. A final direct-form FFT is placed at the output of the biplex FFTs, finishing the

Fourier transform. Pipeline registers favor the direct-form FFT slightly, though the critical path through this circuit is still through $\log_2(n)$ butterflies, so one pipeline register per stage (a pipeline depth of $\log_2(n)$) is recommended. Bitwidths grow immediately to the estimated output bitwidth assuming one bit growth per stage. This growth is split between the biplex and direct FFT stages, so bitwidth growth happens twice. If the output has more or fewer bits than normal growth would imply, the result is automatically truncated or sign-extended as needed. This architecture reuses significantly from the previous spectrometer design [23].

I. Integration of Mixed-Signal Blocks

The ADC is integrated as an analog black box. A Chisel black box contains only the interface, with the module itself being replaced with a model for simulation or the front-end physical design views for synthesis and place-and-route. The top-level IO includes differential clock and data inputs, supplies, and references. These are punched through the hierarchy by mixing in a special trait and manually declaring the connections. Digital configuration pins for the ADC are manually added to the floating SCR file.

The outputs of the ADC are eight 9-bit words, though the data rate requires a 1.2-GHz clock. To avoid such a high-speed signal processing path, the outputs are deserialized by a factor of 4 before being calibrated. The deserializer on the output of the ADC was written in Verilog to account for the intricate clocking, and black-boxed with the analog component of the ADC. The resulting block produces 32 9-bit words at 300 MHz.

Calibration is handled by a runtime-programmable SRAM memory bank, written in Chisel. It connects the deserialized ADC outputs to the address ports of 32 SRAMs and outputs the read port into the signal processing chain. Special modes of the calibration SRAMs support loading of the calibration data, storing ADC samples directly into the memory instead of using them as the address, and reading out of these SRAMs through the SCR file. To store calibrated ADC data samples directly for testing, a SAM must be connected after the calibration RAMs. This is done after a BM processing element that passes its input to its output without any modifications.

VII. Verification

Verification of the generators and instances was split into unit level tests and system level tests. The Rocket CPU comes with its own verification infrastructure, and this is considered at the system level. Design for test structures in the hardware supports simple translation of design-time verification tests into hardware instance validation tests.

A. Unit Tests

At the unit level, two verification paradigms are adopted. The first involved writing Chisel tests for the processing element generators. Since the PEs all contain the same interfaces (AXI4-Stream data inputs and outputs, and an AXI4 control interface), boilerplate functions simplify transaction definitions for verification. For example, SCR file registers are all 64-bit unsigned integers accessed through the AXI4 interface. But some PEs, such as the FIR filter, have programmable coefficients residing in the SCR file but converted to an alternate datatype internally. Manual number conversions between testbenches would dramatically complicate design verification and validation. Thus, custom functions provide the capability of writing and reading values represented by some arbitrary datatype (such as a DspComplex with underlying FixedPoint) to and from interfaces using an unsigned integer datatype (such as the AXI4 data signals). Another function added is the ability to read and write memory-mapped SCR file registers using their name reference rather than their physical address. This allows testbenches to scale correctly when new memory-mapped registers are added.

To simplify the verification of the streaming interface, AXI4-Stream input's and output's interface with software buffers. An input sequence is defined and subsequently *played* through the streaming input interface. To support AXI4 status and control interactions with the processing element without considering their latency overhead, the input stream can be *paused*, which invalidates input data and does not increment the read pointer of the input buffer. When the output data are valid, they are added to the output buffer. Conversion between processing element IO datatypes and AXI4-Stream unsigned bits is automatically handled through software packing and unpacking.

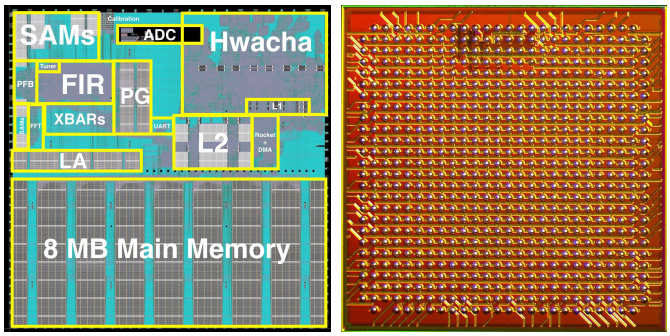
Programming through the AXI4 control interface and communication with the AXI4-Stream data interfaces still require useful tests and golden models to properly verify a processing element. Some PEs, such as the filters and FFT, leverage a Scala numerical processing library Breeze [14], and interface with Plot.ly for data visualization.

Separating the generator design and instance verification complies with the standard practice of separating the concerns. Metadata languages, such as IP-XACT, allow verification engineers to write their own unit tests in a common framework. For this chip instance, Chisel generates IP-XACT files for each PE, then the verification engineers used the IP-XACT to help verify the design. IP-XACT and Cadence VWB were used to generate a UVM testbench, then Python scripts read the IP-XACT parameters section to automatically generate appropriate test vectors.

B. System Tests

System-level testing of the application processor and signal analysis accelerator relies on running compiled C programs. The RISC-V compiler for Rocket systems was used. Building the toolchain produces cross compilers capable of compiling custom C code for both Verilog simulation and hardware testing. To simplify testing, Chisel produces a C header file that maps the names of all the SCR file registers to their addresses in *define* statements. This header file includes other helper functions, like functions for reading and writing the memory-mapped registers and SAM functions.

Running programs require first compiling the Verilog design and testbench in a simulator. The simulation side of the serial



Annotated Layout Die Photo

Technology	16nm FinFET	
Die Area	5mm x 5mm (25mm^2)	
	General-Purpose Processor	Signal-Analysis Processor
Area	1.1 mm^2 (gates) 9.2 mm^2 (SRAM)	1.5 mm^2 (gates) 0.8mm^2 (SRAM)
Total SRAM Size	71 Mbits	14 Mbits
Voltage	0.56 V - 0.98 V	0.56 V - 0.98 V
Max Frequency	410 MHz	417 MHz
Power	349 mW @ 0.75V, 410 MHz	210 mW @ 0.75 V, 417 MHz
Max Throughput (Mspectra/s)	0.46 (vector) 0.004 (scalar) @ 410 MHz	13 @ 417 MHz
Efficiency	23.4 GFLOPS/W (0.56V, 191MHz) (DGEMM on vector accelerator)	19.2 TOPS/W (0.56V, 192MHz) (1 op = 8-bit add ~ 17-bit mul)

Fig. 10. Chip layout, die photograph, and summary.

interface, written in C, connects the hardware to testbench software that writes programs into the chip. This enables any compiled program to be run in simulation. Design and testbench modifications allow integrated IP to be simulated in conjunction with the processor. First, all IP requires a Verilog or SystemVerilog model to simulate properly. The ADC has a custom SystemVerilog model, the clock receiver has a simple differential-to-single-ended Verilog model, the UART design is in Verilog already, the memory compiler generates Verilog models for simulation, and the IO cells have their own Verilog models as well. Second, certain IP, like the ADC, use real-valued analog pins. A noisy sine wave was driven into the top-level ADC inputs to simulate a real signal.

VIII. Testing Results and Measurements

The chip is implemented in Taiwan Semiconductor Manufacturing Companies (TSMC's) 16-nm FinFET technology and signed off at 300 MHz for both the core and DSP clock domains at 0.72 V and 125 °C. Fig. 10 shows the 5 mm × 5 mm annotated layout, die photograph, and chip summary. The 8-MB main memory, Hwacha vector accelerator, and various other memories comprise most of the area. Also visible is the 136-tap fully programmable FIR filter, composed of many complex multipliers and adders. By using the Hwacha vector accelerator, at these conditions, the GPP achieves 23.4 GFLOPS/W running double-precision matrix multiply on 256 × 256 matrices. For the GPP, throughput is measured by moving FFT output data from the SAM to the CPU memory and accumulating spectra using either the vector co-processor or the scalar arithmetic logic unit (ALU) and DMA. Throughput is measured for the signal-analysis processor at the maximum operating frequency. Efficiency for the signal-analysis processor accounts for all PEs, and one

TABLE I

Breakdown of Operations in the Signal-Analysis Processor. Each Operation Is Performed Once per Cycle. C Represents a Complex Number and R Represents a Real Number. It Is Assumed That $C \times R$ Is 2 Multiplies, $C \times C$ Is 3 Multiplies and 5 Adds, and C+C Is 2 Adds

Block	Operation	Bitwidth	Op Count
Tuner	C×R	8	32
Filter	C×C	8	32×136 / 8
Filter	C+C	11	32×135 / 8
PFB	C×C	12	4×12
PFB	C+C	12	4×11
FFT (Biplex)	C×C	17	10
FFT (Biplex)	C+C	17	20
FFT (Direct)	C×C	16	4
FFT (Direct)	C+C	16	8

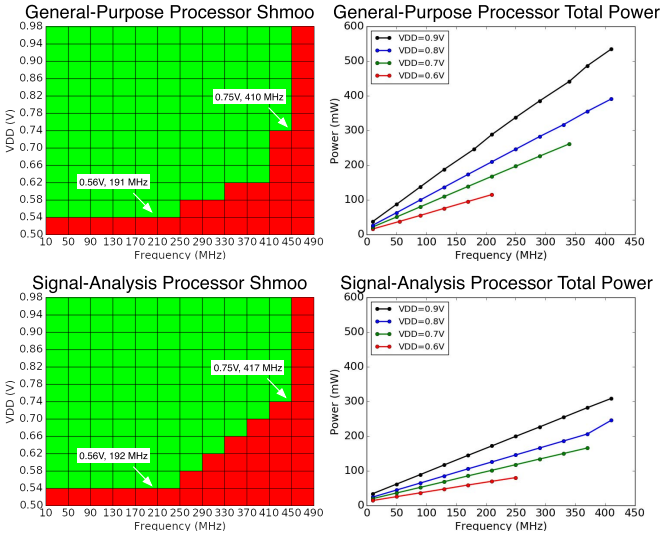


Fig. 11. Both processors function under similar operating condition ranges. The GPP consumes more power because of the 8-MB main memory.

operation is anything from a real 8-bit add to a 17-bit multiply, with complex adds and multiplies broken into their real operations. The signal-analysis processor achieves 19.2 TOPS/W with a breakdown of operations shown in Table I. It is assumed that decimation by eight in the filter reduces the number of operations by eight, since decimation is performed by disconnecting outputs and letting synthesis optimize away unnecessary operations. Also, the direct FFT operations are 19 bits but the output is truncated to 16 bits, so this is counted as a 16-bit operation.

Fig. 11 shows the shmoo and power plots for two processors. A shmoo plot indicates under which operating conditions, typically frequency and voltage, the processor functions correctly. Both function down to 0.56 V (the nominal supply is 0.8 V) and up to 410 MHz. A success on the shmoo plot requires the GPP to pass all ISA unit tests and the signal-analysis accelerator to pass all PE unit tests. Annotated on the shmoo plots are the corner values, i.e., the minimum voltage at the maximum frequency and the maximum frequency at the minimum supply voltage. The chip consumes less than 1 W total for all modes.

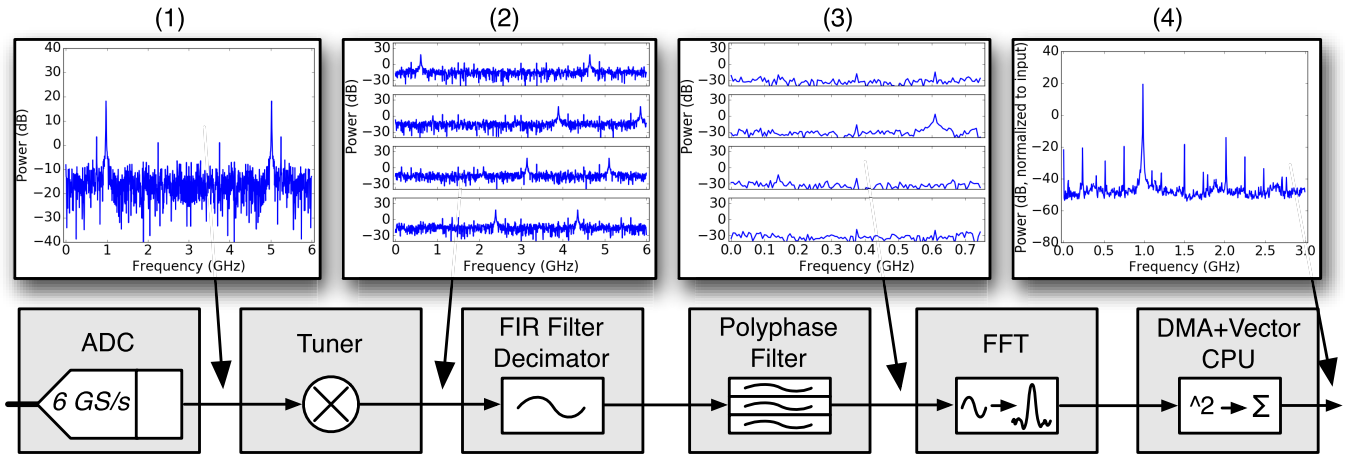


Fig. 12. Spectrometer signal processing example. Snapshots captured in the SAMs. (1) Real-valued signal is sampled through the calibrated ADC, producing a symmetric spectrum. (2) Four-tuner local oscillator (LO) frequencies are mixed with the input, producing four frequency-shifted spectra. (3) These spectra are low-pass filtered and downconverted by 8, resulting in four separate frequency bands. (4) Bands are Fourier transformed, accumulated, and combined on the CPU. This figure shows 100 accumulated spectra.

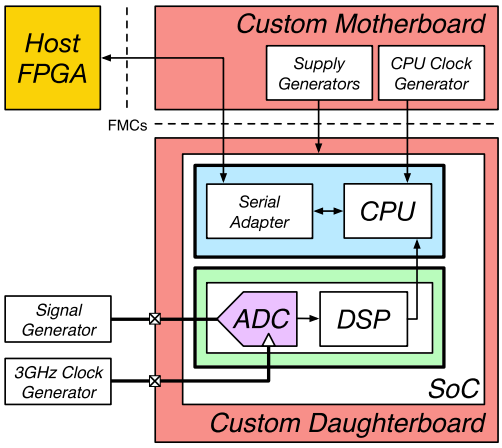


Fig. 13. Testing setup.

TABLE II
COMPARISON OF THE STATE-OF-THE-ART ASIC SPECTROMETERS

	CICC'09 [24]	**CICC'15** [25]	**CICC'18** [23]	**This Work**
Technology	90nm CMOS	65nm CMOS	28nm FDSOI	16nm FinFET
Bandwidth	0.75 GHz	1.1 GHz	**8.5 GHz**	3.0 GHz
FFT Size	**8192 pts**	512 pts	**8192 pts**	128~512 pts
Integrated ADC	No	**Yes**	No	**Yes**
Power	1500 mW*	**188 mW**	5200 mW	586 mW
ADC Output	**8 bits**	7 bits	3 bits	**8 bits**
Can post-process	No	No	No	**Yes**
On-chip Accum. Depth	16M Spectra	1024 Spectra	65520 Spectra	**Infinite**

*excludes ADC power

IX. Signal Analysis Applications

In this section, we present two applications running on the SoC. Using the PG to produce arbitrary waveforms and replace the ADC as the input source hastened the development and debugging of writing these programs by decoupling program errors from ADC miscalibration, input noise, or instrument errors. However, the presented applications and results include the use of external waveform generators and the ADC. These applications utilize many SoC features, including the complete signal processing chain, DMA, vector accelerator, and GPP. We compare the results with similar, fixed-function processors.

A. Spectrometry

Atmospheric spectrometers monitor molecule emissions to determine the composition of gases. Given the low SNR of these emissions, spectrometers with a wide bandwidth and long accumulation time are desired. A 512-pt FFT is formed by sweeping the tuner frequency to allow the signal processor to analyze up to four frequency bands with a 128-pt FFT engine (the filter decimates the data rate by eight, but half the bands are symmetric because the input is real-valued). Fig. 12 shows the signal processing path of a spectrometer input data set, tested via the setup shown in Fig. 13. The low-pass filter, with constrained equiripple coefficients designed in MATLAB, passes the lower eighth of the spectrum to avoid aliasing when down converting. Its stopband is at 40 dB below the passband, resulting in visible aliasing above the noise floor in the combined spectrum. For each band, the tuner is set and the FFT outputs are stored in the SAM before being moved into the GPP's memory by the DMA. The power is calculated and accumulated using the vector accelerator. The use of these accelerators boosts the data processing rate for this application by over 10×. While not designed specifically for spectrometry, this work is competitive with other published ASIC spectrometers, as given in Table II.

B. Radar

Unlike atmospheric spectrometry, the radar operates on fixed or variable short pulses or frequency-modulated

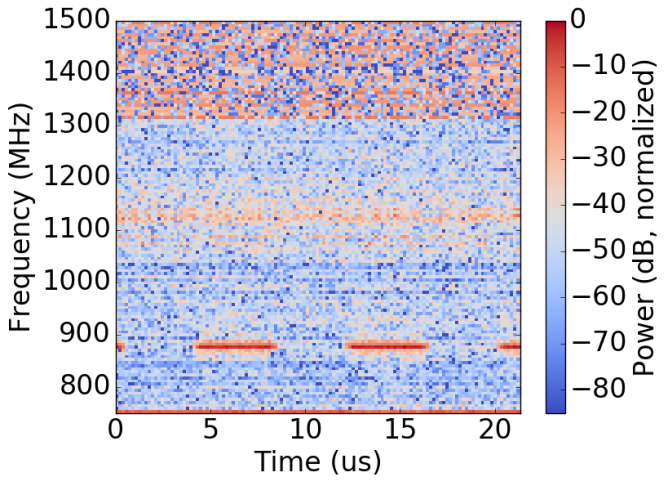


Fig. 14. Measured spectrogram of a 4-μs pulse at 876 MHz.

continuous-wave (FMCW) signals. These higher SNR signals require less accumulation, but processing speed limits the detectable range resolution. Fig. 14 shows an example measured spectrogram of 4 μm fixed-frequency pulses, repeating every 8 μm. The tuner is set to view frequency bands containing the expected signal, and the FFT outputs a spectrum every 171 ns when the ADC operates at 6 GS/s. For unmodulated pulsed signals as shown in Fig. 14, the minimum pulse repetition frequency (PRF) this SoC can resolve is 2.9 MHz (pulses per second), leading to a minimum range resolution of

$$\frac{c}{2 \times \text{PRF}} = 51.7 \text{ m}. \tag{1}$$

It is possible to increase the resolution by implementing pulse compression. The vector accelerator may be used to convolve the received signal with the expected signal, and the FFT may be reused to perform an inverse FFT (IFFT) to recover the compressed signal. At 6 GS/s and by using a single-frequency band with a 750-MHz wide linear frequency-modulated (LFM) chirp, this system has a minimum range resolution of 0.2 m.

X. Conclusion

This paper demonstrates an SoC, designed by using parameterized digital and analog generators, which achieves a peak efficiency of over 19 TOPS/W and 23 GFLOPS/W in 16-nm CMOS. The implemented RISC-V signal analysis SoC, generated from Chisel and BAG frameworks, performs spectrometry and radar signal processing with performance comparable to the state of the art. On-chip DFT facilitates quick bring-up and validation of the design instance. Generators used in this paper are open source and may be easily adjusted and reused for a variety of applications [1]. The process of designing the chip highlights how the proper application of agile hardware development principles contributes to the success of a project, in this case, a streaming DSP processor.

References

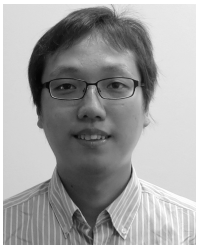
[1] *Craft 2 Top-Level Repository*. Accessed: May 15, 2019. [Online]. Available: https://github.com/ucb-art/craft2-chip
[2] J. Bachrach *et al.*, "Chisel: Constructing hardware in a Scala embedded language," in *Proc. DAC Design Autom. Conf.*, San Francisco, CA, USA, Jun. 2012, pp. 1212–1221.
[3] E. Chang *et al.*, "BAG2: A process-portable framework for generator-based AMS circuit design," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, San Diego, CA, USA, Apr. 2018, pp. 1–8.
[4] Y. Lee *et al.*, "An agile approach to building RISC-V microprocessors," *IEEE Micro*, vol. 36, no. 2, pp. 8–20, Mar./Apr. 2016.
[5] *IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows*, IEEE Standard 1685-2014, Sep. 2014.
[6] B. Nikolić, E. Alon, and K. Asanović, "Generating the next wave of custom silicon," in *Proc. IEEE 44th Eur. Solid State Circuits Conf. (ESSCIRC)*, Dresden, Germany, Sep. 2018, pp. 6–11.
[7] A. Izraelevitz *et al.*, "Reusability is FIRRTL ground: Hardware construction languages, compiler frameworks, and transformations," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Irvine, CA, USA, Nov. 2017, pp. 209–216.
[8] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *Proc. Int. Symp. Code Gener. Optim.*, San Jose, CA, USA, Mar. 2004, pp. 75–86.
[9] O. Shacham *et al.*, "Avoiding game over: Bringing design to the next level," in *Proc. DAC Design Autom. Conf.*, San Francisco, CA, USA, Jun. 2012, pp. 623–629.
[10] K. Asanović *et al.*, "The rocket chip generator," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2016-17, Apr. 2016.
[11] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proc. IEEE*, vol. 75, no. 9, pp. 1235–1245, Sep. 1987.
[12] A. Wang, P. Rigge, A. Izraelevitz, C. Markley, J. Bachrach, and B. Nikolić, "ACED: A hardware library for generating DSP systems," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2018, p. 61:1–61:6.
[13] *Rocket Chip Generator*. Accessed: Nov. 6, 2017. [Online]. Available: https://github.com/freechipsproject/rocket-chip
[14] D. Hall *et al.* (2009). *Breeze*. [Online]. Available: https://github.com/scalanlp/breeze
[15] A. Wang *et al.*, "A real-time, 1.89-GHz bandwidth, 175-kHz resolution sparse spectral analysis RISC-V SoC in 16-nm FinFET," *IEEE J. Solid-State Circuits*, vol. 54, no. 7, pp. 1993–2008, Jul. 2019.
[16] S. Bailey *et al.*, "A generated multirate signal analysis RISC-V SoC in 16 nm FinFET," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Tainan, Taiwan, Nov. 2018, pp. 285–288.
[17] A. Wang *et al.*, "A real-time, analog/digital co-designed 1.89-GHz bandwidth, 175-kHz resolution sparse spectral analysis RISC-V SoC in 16-nm FinFET," in *Proc. IEEE 44th Eur. Solid State Circuits Conf. (ESSCIRC)*, Dresden, Germany, Sep. 2018, pp. 322–325.
[18] *Circuit Realization at Faster Timescales*, DARPA, Arlington County, VA, USA, DARPA-BAA-15-55.
[19] C. Schmidt, A. Ou, and K. Asanović, "Hwacha v4: Decoupled data parallel custom extension," in *Proc. Inaugural RISC-V Summit*, Santa Clara, CA, USA, Dec. 2018, pp. 1–40.
[20] J. Chennamangalam. (Aug. 2011). *The Polyphase Filter Bank Technique*. [Online]. Available: https://casper.ssl.berkeley.edu/wiki/The_Polyphase_Filter_Bank_Technique
[21] A. Parsons, "The symmetric group in data permutation, with applications to high-bandwidth pipelined FFT architectures," *IEEE Signal Process. Lett.*, vol. 16, no. 6, pp. 477–480, Jun. 2009.
[22] R. Emerson, "Biplex pipelined FFT," in *Proc. Deep Space Netw. Prog. Rep. 42–34*, Aug. 1976, pp. 54–59. [Online]. Available: https://library.nrao.edu/public/memos/spm/SPM_007.pdf
[23] S. Bailey *et al.*, "A 28 nm FDSOI 8192-point digital ASIC spectrometer from a chisel generator," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, San Diego, CA, USA, Apr. 2018, pp. 1–4.
[24] B. Richards *et al.*, "A 1.5GS/s 4096-point digital spectrum analyzer for space-borne applications," in *Proc. IEEE Custom Integr. Circuits Conf.*, Rome, Italy, Sep. 2009, pp. 499–502.
[25] F. Hsiao, A. Tang, Y. Kim, B. Drouin, G. Chattopadhyay, and M.-C. F. Chang, "A 2.2 GS/s 188 mW spectrometer processor in 65 nm CMOS for supporting low-power THz planetary instruments," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, San Jose, CA, USA, Sep. 2015, pp. 1–3.

Steven (Stevo) Bailey (M'11) was born in Richmond, VA, USA, in 1989. He received the B.S. degree in engineering science and the B.A. degree in physics from the University of Virginia, Charlottesville, VA, USA, in 2012, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California at Berkeley, Berkeley, CA, USA, in 2014 and 2018, respectively.

He was a Summer Research Intern with the Jet Propulsion Laboratory, Pasadena, CA, USA, in 2014, and Nvidia Corporation, Santa Clara, CA, USA, in 2015. He is currently working in San Jose, CA, USA. His research interests include digital integrated circuit design methodologies, digital signal processing and algorithms, and machine learning.

Paul Rigge (S'09) received the B.S. degree in electrical engineering and computer science from the University of Michigan, Ann Arbor, MI, USA, in 2012. He is currently pursuing the Ph.D. degree with the University of California at Berkeley, Berkeley, CA, USA.

He held an internship position at the NASA Jet Propulsion Laboratory, Pasadena, CA, USA, where he was involved in signal processing for spectrometers. He also held an internship position at Google, Sunnyvale, CA, USA, where he was involved in digital logic design. His current research interests are agile hardware methodologies for wireless systems.



Jaeduk Han (S'15–M'17) received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Seoul, South Korea, in 2007 and 2009, respectively, and the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 2017.

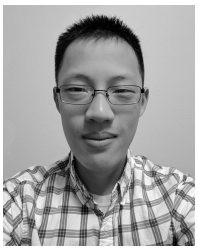
His research interests include high-speed analog and mixed-signal circuits, power electronics, and automatic generation of analog and mixed-signal circuits.

Richard Lin received the B.S. degree in electrical engineering and computer science from the University of California at Berkeley, Berkeley, CA, USA, in 2013, where he is currently pursuing the Ph.D. degree in computer science.

He is currently a Graduate Student Researcher with the University of California at Berkeley. His research interests include design tools for electronics, both at the digital logic and circuit board level, with a focus on human–computer interaction.

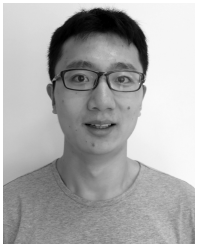
Eric Y. Chang received the B.S. degree in electrical engineering and computer science from the University of California at Berkeley, Berkeley, CA, USA, in 2011, where he is currently pursuing the Ph.D. degree in electrical engineering.

From 2011 to 2014, he was a Full-Time Researcher with the Oracle Labs, Redwood Shores, CA, USA, where he focused on silicon photonics. His current research interests include RF power amplifiers, high-speed wireline links, and design and layout automation of analog and mixed-signal circuits.



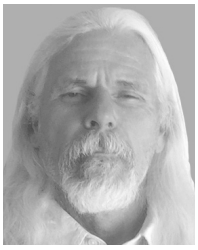
Howard Mao was born in Shanghai, China, in 1991. He received the B.S. degree in computer engineering from Columbia University, New York, NY, USA, in 2014, and the M.S. degree in electrical engineering from the University of California at Berkeley, Berkeley, CA, USA, in 2016, where he is currently pursuing the Ph.D. degree.

In 2014, he was a Software Development Engineer with Amazon.com, Seattle, WA, USA. He has coauthored a paper which will be published in MICRO top picks in 2019. His research interests include agile hardware design, disaggregated memory in datacenters, and novel memory system architectures.



Zhongkai Wang (S'16) received the B.S. degree in electrical engineering from Northwestern Polytechnical University, Xi'an, China, in 2011, and the M.S. degree in electrical engineering from Fudan University, Shanghai, China, in 2014. He is currently pursuing the Ph.D. degree in electrical engineering with the University of California at Berkeley, Berkeley, CA, USA.

His current research interests include mixed-signal circuits, high-speed wireline communication circuits, and analog circuit design automation.



Chick Markley was born in Lakehurst, NJ, USA, in 1958. He received the A.B. degree in physics from the University of California at Berkeley, Berkeley, CA, USA, in 1981.

From 2009 to 2013, he worked on a problem in general artificial intelligence for AzureSky, Alexandria, VA, USA, a private research company. Since 2013, he has been a member of the Staff with the Electrical Engineering and Computer Sciences Department, University of California at Berkeley. His research interests include high-performance computing, code generation, and hardware simulation.

Adam M. Izraelevitz (S'11) received the B.S. degree in electrical and computer engineering from Cornell University, New York, NY, USA, in 2013. He is currently pursuing the Ph.D. degree in electrical engineering in computer architecture with the University of California at Berkeley, Berkeley, CA, USA.

From 2009 to 2013, he held summer internships at the Los Alamos National Laboratory, Los Alamos, NM, USA, and Intel Corporation, Hillsboro, OR, USA, and undergraduate research with the Computer Systems Laboratory, Cornell University. His current research interests include developing a hardware compiler infrastructure, FIRRTL, to enable programmatic transformation of RTL designs.

Mr. Izraelevitz's awards and honors include the 2012 Barry Goldwater Scholarship, USA, the 2009 J. Robert Oppenheimer Scholarship (LANB), and the Los Alamos National Laboratory (LANL) Foundation Gold Scholarship.

Angie Wang (S'11–M'18) received the B.S. degree in electrical engineering from the California Institute of Technology, Pasadena, CA, USA, in 2012, and the Ph.D. degree from the University of California at Berkeley, Berkeley, CA, USA, in 2018, where she was supported by the National Science Foundation's Graduate Research Fellowship.

Her research interests include algorithm/architecture co-design for intelligent systems and design methodologies that enable agile ASIC and FPGA prototyping of next-generation software-defined radios, multisensor fusion, and beyond.

Nathan Narevsky (S'11–M'18) received the B.S. degree in electrical engineering and computer science from the University of California at Berkeley, Berkeley, CA, USA, in 2012, where he is currently pursuing the Ph.D. degree in electrical engineering with the Berkeley Wireless Research Center.

In 2012, he was an Intern at Intel, Hillsboro, OR, USA, where he is involved in integrated voltage regulators for next-generation microprocessor dynamic voltage and frequency scaling. In 2015, he was an Intern at Intel Labs, Hillsboro, OR, USA, where he is involved in mixed-signal equalizers for next-generation millimeter-wave communication systems. His research interests include energy-efficient analog and mixed-signal circuit and system design, optimization, and automation for various applications, including array radio systems, biomedical devices, and high-speed communication interfaces.

Woorham Bae (S'14–M'16) received the B.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2010 and 2016, respectively.

In 2016, he was with the Inter-University Semiconductor Research Center, Seoul National University. From 2017 to 2019, he was a Post-Doctoral Researcher with the University of California at Berkeley, Berkeley, CA, USA. He is currently a Senior SerDes Engineer with the Ayar Labs, Santa Clara, CA, USA. His current research interests include integrated circuits for silicon photonics, high-speed I/O circuits and architectures, non-volatile memory systems, and agile hardware design methodology.

Dr. Bae was a recipient of the IEEE Circuits and Systems Society Outstanding Young Author Award in 2018, the Distinguished Ph.D. Dissertation Award from the Department of Electrical and Computer Engineering, Seoul National University in 2016, the IEEE Circuits and Systems Society Pre-Doctoral Scholarship in 2016, the IEEE Solid-State Circuits Society STG Award in 2015, and the Best Poster Award at the IC Design Education Center Chip Design Contest, International SoC Design Conference, in 2014.

Steve Shauck received the B.E.S. and M.S.E. degrees in electrical engineering and computer science from Johns Hopkins University, Baltimore, MD, USA, in 1982.

He is currently a Consulting Engineer with Northrop Grumman Corporation, Linthicum, MD, USA.

Sergio Montano was born in Cluj-Napoca, Romania, in 1960. He received the B.S.E.E. degree from the Polytechnic Institute of Cluj-Napoca, Cluj-Napoca, in 1984, and the M.S. degree in electrical engineering from George Mason University, Fairfax, VA, USA, in 1995.

After military service, from 1985, he held positions in Research and Development in the telecom and wireless industry with several companies in Romania, Israel, and USA. Since 2004, he has been a Principal ASIC Engineer with Northrop Grumman Corporation, Linthicum, MD, USA, where he is currently a Staff Engineer with the Advanced Technology Group involved in complex mixed-signal large-scale integrated circuit development. He has coauthored several published papers and holds two patents in the wireless communication area. His research interests include edge computing, focal-plane image processing, and cryogenic CMOS design.

Justin Norsworthy was born in 1979. He received the B.S. degree in electrical engineering from the Rose-Hulman Institute of Technology, Terre Haute, IN, USA, in 2002, and the M.S. degree in electrical engineering from Johns Hopkins University, Elkridge, MD, USA, in 2007.

Munir Razzaque (M'91) received the B.S. and M.S. degrees in electrical and computer engineering from The University of Texas at Austin, Austin, TX, USA, in 1991 and 1997, respectively.

From 1997 to 2014, he has held positions with IBM, Austin, TX, USA, and AMD, Austin, TX, USA, as Design and Verification Engineer, where he was involved in the development of high-performance microprocessors and system-on-chip ASICs. He has extensive industry experience in computer architecture, SoC integration, testbench development, methodology specification, and tools development. Since 2015, he has been a Senior Principal Engineer with Northrop Grumman Corporation, Linthicum, MD, USA, where he has been involved in complex mixed-signal large-scale ASIC development.

Wen Hau Ma received the B.S. degree in electrical engineering from State University of New York at Buffalo, Buffalo, NY, USA, in 1993, and the M.S. degree in electrical engineering from New York University, New York, NY, USA, in 2000.

He has been in the semiconductor field for 19 years as a Designer and Verification Engineer. From 2000 to 2016, he was with Lucent Technologies, Murray Hill, NJ, USA; Agere Systems, Allentown, PA, USA; iKanos Communications, Fremont, CA, USA; and Qualcomm Inc., San Diego, CA, USA. Since 2016, he has been with Northrop Grumman Corporation, Linthicum, MD, USA.

Akalu Lentiro received the B.Sc. degree in computer engineering technology from Southern Polytechnic State University, Marietta, GA, USA, in 2006, and the M.S. and Ph.D. degrees in electrical engineering from North Carolina State University, Raleigh, NC, USA, in 2008 and 2013, respectively.

From 2007 to 2011, he was a Graduate Research Assistant with North Carolina State University, and a Senior Hardware Engineer with Cisco, Herndon, VA, USA, from 2010 to 2014, mostly responsible for system-level pre- and post-route signal integrity analysis for high-speed interfaces and backplane designs. He is currently a Principal Engineer with Northrop Grumman Corporation, Linthicum, MD, USA, where he is involved in various gate-level ASIC, FPGA, and DSP designs and leading board design verification and system integration efforts.

Matthew Doerflein received the B.S. degree in electrical engineering from The Pennsylvania State University, State College, PA, USA, in 2006.

From 2006 to 2009, he was a Hardware Engineer with Lockheed Martin Corporation, Owego, NY, USA, with a focus on RF/microwave product development. In 2009, he joined Northrop Grumman Corporation, Linthicum, MD, USA, where he is currently a Program Manager. He has held multiple design, manufacturing, leadership, and management positions focused on microelectronics research and development. His research interests include RF/Microwave systems and subsystem product development, specifically MMIC, RFIC, and application-specific integrated circuit development.

Darin Heckendorn received the B.S.E.E. degree from The Pennsylvania State University, State College, PA, USA.

He is a 22-year veteran of the electronics industry specializing in digital SoC design as an Applications Engineer and an ASIC designer. He has extensive experience in multiple EDA tool suites and has developed and supported RTL to GDS flows in a variety of technologies from 350 to 7 nm. In addition, he has developed specialized flows for RAD Hard, 2.5-D/3-D, and superconducting electronics designs. As a respected member of the Cadence Aerospace and Defense Team, he is currently supporting the DARPA CRAFT, CHIPS, and IDEA programs.

Jim McGrath (M'76–SM'13) received the B.S. and M.S. degrees in electrical engineering from the New Jersey Institute of Technology, Newark, NJ, USA, in 1980 and 1983, respectively.

He held various digital hardware and software development positions. He was an Application Engineering with Cadence Design Systems, Cary, NC, USA, where he was specialized in digital ASIC verification.

Franco DeSeta, photograph and biography not available at the time of publication.

Ronen Shoham, photograph and biography not available at the time of publication.

Mike Stellfox, photograph and biography not available at the time of publication.

Mark Snowden was born in Birmingham, AL, USA, in 1960. He received the B.S. degree in electrical engineering from the University of Florida, Gainesville, FL, USA, in 1983.

He has held design and CAD engineering positions at Harris Semiconductor, Melbourne, FL, USA, Unitrode, Cary, NC, USA, and Rambus, Chapel Hill, NC, USA. He has held research and development and application positions at Synopsys and Cadence, Cary, NC, USA. Since 2015, he has been a Sr. Principal Application Engineer with Cadence Design Systems, Inc., Cary, NC, USA. He has published an article in the IEEE JOURNAL OF SOLID-STATE CIRCUITS and holds two patents. His specialization is in physical verification and extraction.

Joseph Cole, photograph and biography not available at the time of publication.

Daniel (Dan) R. Fuhrman received the A.S. degree in engineering science and the A.A.S. degree in computer engineering technology from the State University of New York, Alfred, NY, USA, in 1984, and the B.S. degree in computer science from Union College, Schenectady, NY, USA, in 1989.

From 1984 to 1991, he was an ASIC Design Engineer with IBM, Fishkill, NY, USA, and from 1991 to 1994, he was an ASIC Project Manager with IBM, Research Triangle Park, Raleigh, NC, USA. Since 1994, he has held various positions at Cadence Design Systems, Cary, NC, USA, where he is currently a Program Management Director. He holds one patent.

Brian Richards (M'09) received the B.S. degree in electrical engineering from the California Institute of Technology, Pasadena, CA, USA, in 1983, and the M.S. degree in electrical engineering and computer science from the University of California at Berkeley, Berkeley, CA, USA, in 1986.

He joined the Research Staff, University of California at Berkeley, in 1986, where he worked on large-scale digital system design projects, including the Infopad portable wireless multimedia terminal and several automated integrated circuit design flows. As a Founding Member of the Berkeley Wireless Research Center in 1999, he is continuing the development and support of several ASIC and FPGA system design CAD tool flows. He has also held internship or consulting positions at IBM Yorktown, Megatest, San Jose, CA, USA, Kodak, Rochester, NY, USA, Intel, Hillsboro, OR, USA, and BEEcube, Fremont, CA, USA.

Jonathan Bachrach (M'07) was born in Gastonia, NC, USA, in 1961. He received the B.S. degree in computer science and cognitive science from the University of California at San Diego, La Jolla, CA, USA, in 1985, and the M.S. and Ph.D. degrees in computer science from the University of Massachusetts, Amherst, MA, USA, in 1989 and 1992, respectively.

He was a Research Scientist with the MIT Artificial Intelligence Laboratory, Cambridge, MA, USA, from 2000 to 2008. He co-founded Otherlab in 2009, and since 2012, he has been an Adjunct Assistant Professor of computer science with the Electrical Engineering and Computer Sciences Department, University of California at Berkeley, Berkeley, CA, USA.

Elad Alon (M'06–SM'12–F'19) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 2001, 2002, and 2006, respectively.

He is currently a Professor of electrical engineering and computer sciences with the University of California at Berkeley, Berkeley, CA, USA, and also the Co-Director of the Berkeley Wireless Research Center (BWRC). He is also a Co-Founder and the Chief Scientist with Blue Cheetah Analog Design, San Francisco, CA, USA, which is commercializing generator technologies in order to enable analog/mixed-signal solutions at lower barrier to entry. He has also held advisory, consulting, or visiting positions at Ayar Labs, Emeryville, CA, USA, Locix, San Bruno, CA, USA, Lion Semiconductor, San Francisco, CA, USA, Cadence, Cary, NC, USA, Xilinx, San Jose, CA, USA, Wilocity (currently Qualcomm), San Diego, CA, USA, Oracle, Redwood City, CA, USA, Intel, Hillsboro, OR, USA, Santa Clara, CA, USA, AMD, Santa Clara, CA, USA, Rambus, Chapel Hill, NC, USA, Hewlett Packard, Palo Alto, CA, USA, and IBM Research, San Jose, CA, USA, where he worked on digital, analog, and mixed-signal integrated circuits for computing, test and measurement, power management, and high-speed communications. His research interests include energy-efficient integrated systems, including the circuit, device, communications, and optimization techniques used to design them.

Dr. Alon received the IBM Faculty Award in 2008, the 2009 Hellman Family Faculty Fund Award, and the 2010 and 2017 UC Berkeley Electrical Engineering Outstanding Teaching Awards. He has coauthored papers that received the 2010 ISSCC Jack Raper Award for Outstanding Technology Directions Paper, the 2011 Symposium on VLSI Circuits Best Student Paper Award, the 2012 and the 2013 Custom Integrated Circuits Conference Best Student Paper Awards, and the 2010–2016 Symposium on VLSI Circuits Most Frequently Cited Paper Award.

Borivoje Nikolić (S'93–M'99–SM'05–F'17) received the Dipl.Ing. and M.Sc. degrees in electrical engineering from the University of Belgrade, Belgrade, Serbia, in 1992 and 1994, respectively, and the Ph.D. degree from the University of California at Davis, Davis, CA, USA, in 1999.

In 1999, he joined the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, USA, where he is currently a National Semiconductor Distinguished Professor of Engineering. He has coauthored the book *Digital Integrated Circuits: A Design Perspective*—Second Edition (Prentice Hall, 2003). His research interests include digital, analog, and RF integrated circuit design and VLSI implementation of communications and signal processing systems.

Dr. Nikolić was a recipient of the NSF CAREER Award in 2003, the College of Engineering Best Doctoral Dissertation Prize, the Anil K. Jain Prize for the Best Doctoral Dissertation in Electrical and Computer Engineering at the University of California at Davis, in 1999, and the City of Belgrade Award for the Best Diploma Thesis in 1992. For the work with his students and colleagues, he received the best paper awards at the IEEE International Solid-State Circuits Conference, the Symposium on VLSI Circuits, the IEEE International SOI Conference, the European Solid-State Device Research Conference, the European Solid-State Circuits Conference, the S3S Conference, and the ACM/IEEE International Symposium of Low-Power Electronics. From 2014 to 2015, he was a Distinguished Lecturer of the IEEE Solid-State Circuits Society.