A Real-Time, 1.89-GHz Bandwidth, 175-kHz Resolution Sparse Spectral Analysis RISC-V SoC in 16-nm FinFET

Angie Wang, *Member, IEEE*, Woorham Bae, *Member, IEEE*, Jaeduk Han, *Member, IEEE*, Stevo Bailey, *Member, IEEE*, Orhan Ocal, *Student Member, IEEE*, Paul Rigge, *Student Member, IEEE*, Zhongkai Wang, *Student Member, IEEE*, Kannan Ramchandran, *Fellow, IEEE*, Elad Alon, *Fellow, IEEE*, and Borivoje Nikolić, *Fellow, IEEE*

***Abstract*— A 1.89-GHz bandwidth, 175-kHz resolution spectral analysis system-on-chip (SoC), integrating a subsampling analog-to-digital converter (ADC) frontend with a digital reconstruction backend and implementing a 21 600-point sparse Fourier transform based on the fast Fourier aliasing-based sparse transform (FFAST) algorithm has been co-designed by using the Constructing Hardware in a Scala Embedded Language (Chisel) and Berkeley Analog Generator (BAG) circuit generator frameworks in 16-nm CMOS. Three sets of 25×, 27×, and 32× subsampling successive approximation register (SAR) ADCs acquire signal with ∼5.4–6.3 effective number of bits (ENOB)/slice. The digital backend consists of mixed-radix 864-, 800-, and 675-point fast Fourier transforms (FFTs), a signal location estimator, and a peeling decoder that recovers aliased signals from a sparsely populated spectrum. A single-issue, in-order, fifth-generation reduced instruction set (RISC-V) Rocket processor interacts with the spectrum analyzer for post-processing and calibration. The ADC consumes 49.8 mW with a 3.78-GHz reference clock. At 400 MHz and 0.7-V digital supply voltage (VDD), the Rocket core and the FFAST digital signal processing (DSP) together consume 133.5 mW.**

***Index Terms*— Analog-to-digital converters (ADCs), Berkeley Analog Generator (BAG), Constructing Hardware in a Scala Embedded Language (Chisel), fast Fourier transform (FFT), hardware generators, fifth-generation reduced instruction set computer (RISC-V), spectrum sensing.**

I. Introduction

As WITH radar and radio spectrometry, wideband spectrum sensing for the detection of unknown signals typically requires either a high sampling rate, high-power analog-to-digital converter (ADC), or scanning and tuning a narrowband filter across a wide (>GHz) frequency range, making real-time operation challenging. Because particular bands are only monitored for a small fraction of time [1], the scanning approach potentially misses short-duration signals. In practice, there are many applications that:

1) deal with signals that are sparse in the frequency domain;
2) greatly benefit from data compression early in the processing chain; and
3) do not require perfect spectrum reconstruction, tolerating probabilistic errors on the order of a percent.

Recently, compressed sensing techniques [2], [3] have been used to reduce the ADC sampling rate and analog power, but the complexity and overhead of hardware reconstruction have not been adequately addressed [4]. Compressed sensing typically requires analog mixing at gigahertz speeds [1], which, given stringent jitter requirements, results in power numbers comparable to those of high-speed ADCs. An on-chip, all-digital, sparse fast Fourier transform (FFT) accelerator [5], based on the algorithms in [6] and [7], reconstructs signals sparse in frequency. However, it targets a sparsity of only 0.1% and does not address algorithm degradation at low signal-to-noise ratios (SNRs), making it impractical for use in many applications. Agarwal *et al.* [8] and López-Parrado and Velasco-Medina [9] also present field-programmable gate array (FPGA) implementations of sparse FFT algorithms.

We demonstrate an analog/digital co-designed sparse spectral analysis system-on-chip (SoC) supporting real-time signal detection for frequency spectra with sparsities up to 3.2% and input SNRs down to 9.7 dB. The overall 1.89-GHz bandwidth signal acquisition and analysis time is 17.5 μs. This represents a significant improvement over the capabilities of commercial spectrum monitors like RFeye or off-the-shelf Universal Software Radio Peripheral (USRP) devices, which scan smaller bandwidths over the course of tens of milliseconds [1]. The SoC's output data are compressed to 4.4%, accounting for the storage of bin indices.

The premise of this chip is similar to that of MIT's BigBand [1]. Low-speed ADCs are used for uniform subsampling, saving power. Generated mixed-radix FFTs are used for spectrum reconstruction. In addition, the chip is

Manuscript received November 5, 2018; revised February 20, 2019; accepted April 9, 2019. Date of current version June 26, 2019. This paper was approved by Guest Editor Stefan Rusu. This work was supported in part by the Defense Advanced Research Projects Agency's Circuit Realization at Faster Timescales (DARPA CRAFT) Program under Grant HR0011-16-C-0052, in part by the National Science Foundation's Graduate Research Fellowship Program under Grant DGE-1106400, in part by the Berkeley Wireless Research Center, and in part by ADEPT (Intel Science and Technology Center for Agile Hardware Design). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies. *(Corresponding authors: Woorham Bae; Angie Wang.)*

A. Wang, J. Han, S. Bailey, O. Ocal, P. Rigge, Z. Wang, K. Ramchandran, E. Alon, and B. Nikolić are with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: angie.wang@eecs.berkeley.edu).

W. Bae is with Ayar Labs, Santa Clara, CA 95054 USA (e-mail: wrbae@berkeley.edu).

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

Digital Object Identifier 10.1109/JSSC.2019.2913099

capable both of detecting occupied frequency bands and also decoding the signals within them, assuming a sufficiently sparse spectrum. Unlike BigBand, which uses off-the-shelf components, the ADCs and digital reconstruction backend are fully integrated on-chip. Whereas BigBand is capable of sensing signals within a 900-MHz bandwidth, this chip is able to observe signals within a 1.89-GHz bandwidth.

II. FFAST Algorithm Overview

A. Noiseless FFAST

The fast Fourier aliasing-based sparse transform (FFAST) algorithm [10] relies on controlled aliasing via Chinese-remainder-theorem-guided (CRT) subsampling to: 1) reduce the effective input sampling rate, simplifying ADC design; 2) reduce the computational complexity of large FFTs; and 3) minimize data storage and memory bandwidth requirements via data compression, while still supporting a high probability of spectrum recovery for sufficiently sparse spectra. For a k-sparse spectrum, denoting that only k frequency bins contain non-zero data, where $k \ll n$, the computational complexity is reduced from $\mathcal{O}(n \log n)$ to $\mathcal{O}(k \log k)$, corresponding to FFT computation on $\mathcal{O}(k)$ subsampled time-domain data.

The CRT can be used to show that a frequency bin $j \in \{0, \ldots, n-1\}$ is uniquely mapped to subsampled frequency bins b_1 and b_2 such that $j \equiv b_1 \bmod n_1$ and $j \equiv b_2 \bmod n_2$ for $n = n_1 n_2$. n_1 and n_2, either the subsampling factor and length of the resulting spectrum or vice versa in this simple example, are coprime. Subsampling typically poses a problem: because signals at various frequency locations may fold on top of each other, they cannot be distinguished. Aliasing is governed by the equation

$$Y_{n_i}[b] = \sum_{j \equiv b \bmod n_i} X[j] \tag{1}$$

where $\vec{Y}_{n_i}$ represents the subsampled n_i-point spectrum. However, as shown in Fig. 1, when a spectrum is sufficiently sparse, most subsampled frequency bins will contain either noise (zeroton bins) or a signal at a single frequency (singleton bins), and only a few bins will contain multiple aliased signals (multiton bins).

Consider the $n = 20$ time-domain sequence $\vec{x}$ shown in Fig. 2. Its frequency-domain representation $\vec{X}$ has sparsity $k = 5$, where the non-zero signals are $X[3]$, $X[7]$, $X[9]$, $X[12]$, and $X[14]$ (Fig. 1). Fig. 3 illustrates a frontend consisting of d pairwise-coprime subsampling stages that use the CRT to guide the collection of time-domain samples. The aliased frequency spectra of the subsampled data are calculated by n_i-point FFTs, for $i \in \{1, \ldots, d\}$. Fig. 1 shows the FFT outputs—after subsampling by 5 (left) and 4 (right)—of a $d = 2$ frontend. Although simple, it highlights how coprime subsampling decreases the likelihood that a specific frequency collision in one stage (e.g., the $X[3]$ and $X[7]$ collision in stage 1) occurs in the other stage. In general, the number of signals recoverable by FFAST is of the same *order of magnitude* as the sum of the sub-FFT n_is: $\sum_{i=1}^{d} n_i$. Hence, the number of stages d and the n_i used per stage (where n_is

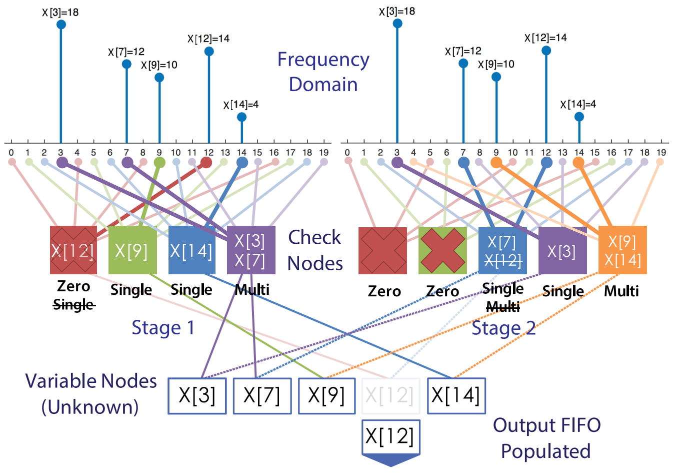


Fig. 1. Signal recovery from subsampled inputs, where $n_1 = 4$ and $n_2 = 5$ are coprime. The $j = 12$ signal recovered from the red singleton bin in stage 1 is peeled off (i.e., subtracted) from the associated multiton bin in stage 2 (blue, location determined by $j \equiv b_2 \bmod n_2$), such that the multiton bin becomes a recoverable singleton bin when processing the next stage (or on the next peeling iteration). Zeroton bins, which can be determined via simple thresholding, are not included in the graph, because they are already known.

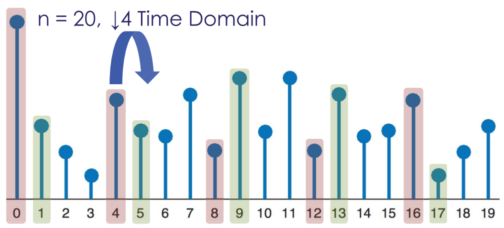


Fig. 2. $n = 20$ time-domain sequence, subsampled by 4 ($n_i = 5$). A unit delay $\tau = 1$ is applied to the subsampling clock to disambiguate singletons from multitons.

do not have to be pairwise coprime) need to be chosen to accommodate for the targeted k. $d = 3$ and n_is of 864, 800, and 675 are chosen to recover up to ~1000 non-zero frequency bins from an $n = 21\,600$ point spectrum. These n_is correspond to subsampling factors of 25, 27, and 32, respectively.

The pairwise-coprime subsampling scheme described in the previous paragraph leads to "peeling-friendly" aliasing patterns that aid in multiton recovery. Assume that singleton bins can be distinguished from multiton bins and that the location and value of a signal in a singleton bin can be easily identified. As these signal properties are iteratively discovered, they can be "peeled off" of the bipartite graph resulting from FFAST subsampling to recover additional signals (Fig. 1) [10]. The bipartite graph contains "variable nodes," which are associated with unknown, non-zero frequency bins in the n-length $\vec{X}$. It also includes "check nodes" with unresolved singleton and multiton bins that contain aliased discrete Fourier transform (DFT) coefficients. If, after subsampling, a particular bin j in $\vec{X}$ (variable node) contributes to a subsampled singleton or multiton bin b in $\vec{Y}_{n_i}$ (check node), the corresponding variable node and check node are connected. Singleton bins only have a single associated variable node, whereas multiton bins are connected to multiple variable nodes. In a "peeling

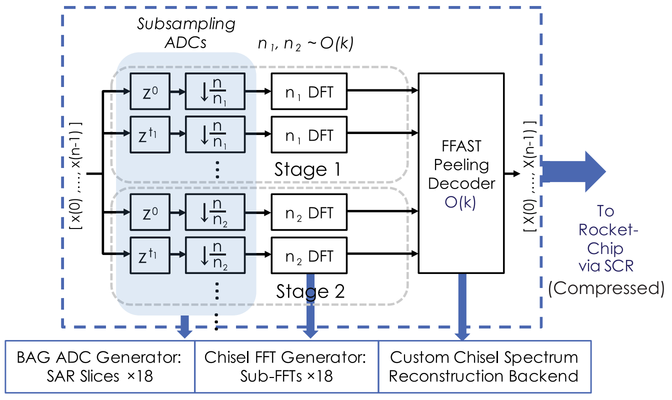

Subsampling ADCs
$n_1, n_2 \sim O(k)$
z^0
z^{t_1}
n_1 DFT
n_2 DFT
Stage 1
Stage 2
FFAST Peeling Decoder O(k)
[x[0] ..... x[n-1]]
To Rocket-Chip via SCR (Compressed)
BAG ADC Generator: SAR Slices ×18
Chisel FFT Generator: Sub-FFTs ×18
Custom Chisel Spectrum Reconstruction Backend

Fig. 3. High-level FFAST architecture supporting k-sparse spectra and consisting of delayed frontend subsampling, $\mathcal{O}(k \log k)$ complexity sub-FFTs, and an $\mathcal{O}(k)$ peeling reconstruction backend. Subsampling ADCs are generated using the BAG framework, and DSP blocks are written in Chisel.

step," a variable node attached to a singleton bin is identified. The variable node's signal contribution is removed from all connected check nodes, effectively severing the connections while potentially uncovering new singletons. The number of peeling iterations needed to resolve $\vec{X}$ is dependent on the input sparsity; additional iterations are needed with increasing k. However, above a particular k threshold dictated by user-selected FFAST parameters, peeling becomes stuck and signals become irrecoverable, as shown in Fig. 6.

To differentiate between singleton and multiton bins, the following relationship is used: given $x[a] \xrightarrow{\mathcal{F}} X[j]$, a time shift by τ results in:

$$x[a+\tau] \xrightarrow{\mathcal{F}} X[j]e^{i\omega\tau}. \tag{2}$$

Ignoring phase wrapping, $\omega = 2\pi j/n$, and the phase rotation associated with τ is $\theta = \omega\tau$. Without accounting for noise, if only a single tone is present in a subsampled frequency bin b, the magnitude in that bin should remain unchanged, regardless of whether the sampling time of the associated time-domain input is delayed. This does not hold true for multiton bins, since aliased frequency components rotate by different amounts before folding. Therefore, to resolve singleton bins from multiton bins, each FFAST subsampling stage contains D delay chains. In the noiseless case, $D = 2$ can be used. The delay chains of a given stage use the same subsampling rate. However, as shown in Figs. 2 and 3, prior to subsampling, the signal is circularly time-shifted by a different amount—achieved by delaying the sampling clock—at the input of each chain.

In Fig. 2, $Y_{n_i,b,0}$ corresponds to the DFT coefficient at subsampled bin $b \in \{0, \ldots, n_i - 1\}$ for the original input, and $Y_{n_i,b,1}$ corresponds to the DFT coefficient at the same bin b for the sample-delayed input. Therefore, an estimate of the phase rotation of a singleton bin due to the delay can be determined via

$$\theta_{\text{est}} = \angle[Y_{n_i,b,1}\overline{Y_{n_i,b,0}}]. \tag{3}$$

From (3), a location estimate can be calculated as $j_{\text{est}} = \theta_{\text{est}} n/(2\pi\tau)$. Since $\tau = 1$, phase wrapping can be ignored.

B. Noise-Robust FFAST

1) Problem Setup: In practical systems, time-domain samples are corrupted by white Gaussian noise [10]. The actual observation is then given by $\vec{y} = \vec{x} + \vec{z}$, although only $\vec{x}$ is desired. To improve the robustness of the FFAST algorithm to noise, the subsampling stages' delay chains use a number of *pseudo-random* time shifts rather than singular consecutive shifts. Random delay shifts are used, so that, as in compressed sensing [11], the sensing matrix satisfies the restricted isometry property (RIP) [12] and has reasonable mutual incoherence [10]. This implies that the subsampling clock is delayed by a random (but fixed, for ease of hardware implementation) amount per delay chain. Because of hardware limitations, in practice, redundant time samples are collected, making the sample complexity worse than the theoretical best case.

Assume that the time-domain input in Fig. 2 is now corrupted by noise and $D > 2$ delay chains are used. From the Fourier relationship described in Section II-A, the signal contribution $X[j]$ from frequency bin j is "steered" by the vector

$$\vec{a}_j = [e^{i2\pi\tau_0\ j/n}\ \cdots\ e^{i2\pi\tau_{D-1}j/n}]^{\mathsf{T}}, \tag{4}$$

resulting in a set of phase-shifted signals that contribute to measurements associated with subsampled bin b if $j \equiv b \bmod n_i$. The set of observations associated with a *zeroton* bin like $Y_{n_i,b} = Y_{5,0}$ (in stage 2 shown in Fig. 1) is given by $\vec{z}_{n_i,b} = [z_{n_i,b,\tau_0}\ \cdots\ z_{n_i,b,\tau_{D-1}}]^{\mathsf{T}}$, whose elements are mostly uncorrelated—owing to the use of different delay chains—and result from noise at higher frequencies folding down due to subsampling. The set of observations associated with the *singleton* bin $Y_{5,3}$ is given by $\vec{Y}_{5,3} = \vec{a}_3\ X[3] + \vec{z}_{5,3}$. Finally, the set of observations associated with the *multiton* bin $Y_{5,2}$, containing two aliased tones, is given by $\vec{Y}_{5,2} = \vec{a}_7\ X[7] + \vec{a}_{12}X[12] + \vec{z}_{5,2}$.

2) FFAST Peeling Decoding: Algorithm 1 presents hardware-adapted pseudocode for the noise-robust FFAST algorithm, which is used to recover the non-zero $X[j]$s from noise-corrupted observations. In an actual hardware implementation, each check_nodes$_i$ is implemented as an $\mathcal{O}(k)$ circular buffer (CB) that keeps track of the non-zeroton bin locations b found in stage i. Observed bins are used for peeling—one-at-a-time—and early termination is implemented to save energy. Fig. 4 illustrates how this is accomplished. A CB's read pointer is updated for each b read. When a zeroton or singleton bin b is discovered, it is removed from the corresponding CB. If a subsampled bin is determined to be an unresolvable multiton, the write pointer is updated, replacing bin location b back into the CB for a decoding attempt during the next peeling iteration. In this example, the stage 2 CB is empty at the end of iteration 0 so peeling can be terminated early.

The noise-robust FFAST algorithm is able to reconstruct a sparse frequency spectrum with a probability of at least $1 - \mathcal{O}(1/k)$ using $\mathcal{O}(k \log^3\ n)$ time-domain samples [10]. The noise-robust algorithm has a computational complexity of $\mathcal{O}(k \log^4\ n)$.

3) Singleton Estimator: To outline the function of the singleton estimator, first consider a set of evenly spaced

Algorithm 1: Noise-Robust FFAST [10], [13] with Early Termination and Optimizations for Hardware

```
Input : Noise-corrupted singleton and multiton bin
        observations $\vec{Y}_{n_i,b}$ whose locations $b$ are stored
        in $check\_nodes_i$ per subsampling stage $i$.
Input : Noise threshold $T_{noise}$. Bins with power levels
        below this only contain noise.
Output: Estimate $\vec{X}$ of the $n$-point FFT compressed to
        $\mathcal{O}(k)$.

▷ Continue if signal information was recovered on the
previous peeling iteration and no CBs are empty.
while length of any $check\_nodes_i$ was updated do
    for each subsampling stage $i \in \{1, \ldots, d\}$ do
        for each bin $b$ in $check\_nodes_i$ do
            if $\|\vec{Y}_{n_i,b}\|^2 < T_{noise}$ then
                bin $b$ is a zeroton
                remove $b$ from $check\_nodes_i$
            else
                ▷ $v_j$ is the zero-delay signal estimate at
                $X[j]$.
                (isSingleton, $v_j$, $j$) =
                  SingletonEstimator($\vec{Y}_{n_i,b}$, $T_{noise}$)
                if isSingleton then
                    ▷ Peel the signal off at all stages.
                    Assumes all stages use the same delays.
                    $\vec{Y}_{n_i,b} \leftarrow \vec{0}$
                    for $l \in \{1, \ldots, d\}, l \neq i$ do
                        $j \equiv q \bmod n_l$
                        if $\|\vec{Y}_{n_l,q}\|^2 < T_{noise}$ then
                            $\vec{Y}_{n_l,q} \leftarrow \vec{0}$
                        else
                            $\vec{Y}_{n_l,q} \leftarrow \vec{Y}_{n_l,q} - v_j \vec{a}_j$
                        end
                    end
                    add $X[j] = v_j$ to the output FIFO
                    remove $b$ from $check\_nodes_i$
                else
                    ▷ No new information from bin $b$.
                    bin $b$ is a multiton
                end
            end
        end
        if $check\_nodes_i$ is empty then
            ▷ Done.
            break out of while loop
        end
    end
end
return $\vec{X}$, compressed
```

observations of a single complex sinusoid corrupted by white Gaussian noise. When the input SNR is sufficiently high (i.e., $\geq$ 5–7 dB) [14], the observations can be approximated as

$$y(t) \approx Ae^{i(\omega t+\phi+u(t))}. \tag{5}$$

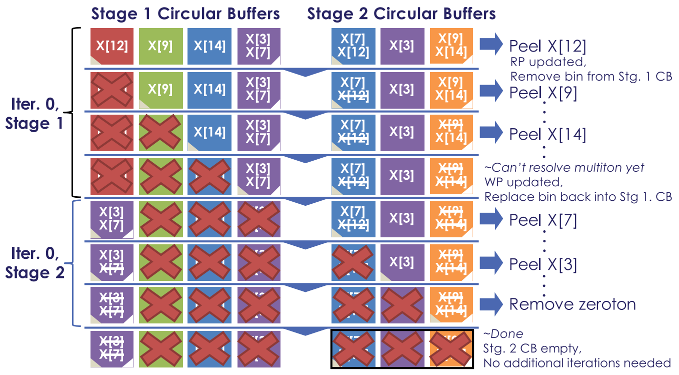


Fig. 4. Peeling using CBs. Read pointer = RP (brown). Write pointer = WP (white).

The amplitude A, angular frequency ω, and phase ϕ are fixed but unknown, and $u(t)$ represents the zero-mean white Gaussian noise sampled at time t. The phase of $y(t)$ is thus $\angle y(t) = \omega t + \phi + u(t)$. When $N = 2$ samples with a delay of τ_s between them are used, $\omega\tau_s$ can be estimated via

$$\angle y(t+\tau_s) - \angle y(t) = \omega\tau_s + u(t+\tau_s) - u(t). \tag{6}$$

Since $u(t+\tau_s)$ and $u(t)$ come from different delay chains in the hardware FFAST implementation, they are minimally correlated. Thus, the minimum mean square error (MMSE) estimate of the potentially phase-wrapped $\omega_s = \langle\omega\tau_s\rangle_{2\pi}$ (note: $\langle x\rangle_n$ denotes $x \bmod n$) can be directly obtained from θ_{est} in (3) (where τ_s, which we consider the new "unit delay," is factored into ω_s), even with a noise-corrupted input.

As described in Section II-B1, each stage uses $D > 2$ delay chains. For noise robustness, D is split into C clusters. Singleton estimation occurs in two steps. First, the MMSE frequency estimator generates estimates of ω_s, for $s \in \{0, \ldots, C-1\}$, from sets of $N = 2$ bin observations with delays of τ_s between observation pairs ($D = 2C$). In the presence of noise, the range around the estimate $\omega_{s,\text{est}}$ containing the true $\langle\omega\tau_s\rangle_{2\pi}$ with high probability is given by

$$\Omega_s = \left(\omega_{s,\text{est}} - \frac{\pi}{c}, \omega_{s,\text{est}} + \frac{\pi}{c}\right) \tag{7}$$

for some constant c [10]. The length of the interval is denoted as $|\Omega_s| = 2\pi/c$. When $\tau_0 = 1$, an estimate of ω can be directly obtained within some region of uncertainty. More generally, all $\omega + 2\pi a/\tau_s, a \in \mathbb{Z}$ map to the same frequency after multiplication by τ_s. That is,

$$\langle(\omega + 2\pi a/\tau_s)\tau_s\rangle_{2\pi} = \langle\omega\tau_s\rangle_{2\pi}, \quad a \in \mathbb{Z}. \tag{8}$$

Therefore, if noise across observations is uncorrelated and relatively insignificant (i.e., c is sufficiently large), the true ω may be isolated to τ_s regions of smaller interval size $|\Omega_s|/\tau_s$ using the calculated $\omega_{s,\text{est}}$, for $\tau_s > 1$. Ambiguity around which of the smaller regions may be mapped to ω can be resolved by looking only at regions of overlap for different τ_ss:

$$\left|\cap_{s=0}^{C-1}\Omega_s/\tau_s\right| \leq \frac{2\pi}{c\tau_{C-1}}, \tag{9}$$

where $\tau_{C-1} = \max \tau_s$. As more delay clusters using larger τ_ss are added, the estimate of ω is isolated to an ever smaller

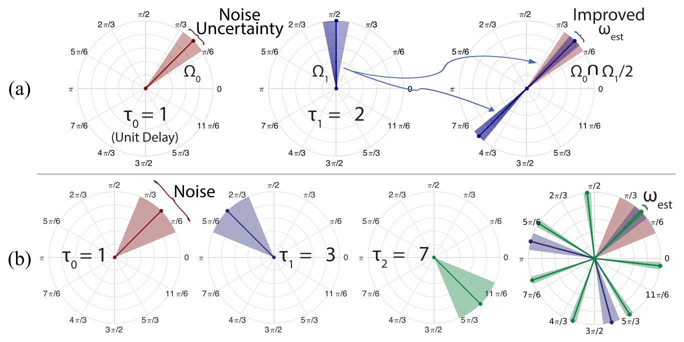


Fig. 5. (a) Successive refinement of ω_{est} for $\tau_s = 2^s$ [15]. (b) Successive refinement of ω_{est} using $\tau_0 = 1$, $\tau_1 = 3$, and $\tau_2 = 7$. The ω estimate is improved over (a), despite higher noise uncertainty.

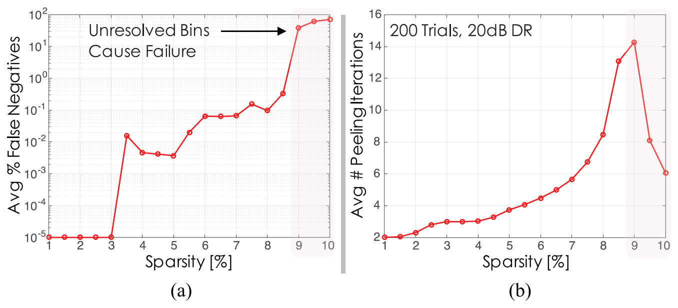


Fig. 6. (a) Percentage of false negatives versus input sparsity (% of occupied spectrum), averaged across 200 trials. (b) Average number of peeling iterations required to decode the frequency spectrum as a function of input sparsity. In both (a) and (b), noisy inputs with a DR of 20 dB are used.

region (Fig. 5) [10], illustrating the successive refinement of ω_{est}. Although the final region of uncertainty is determined by max τ_s, intermediate τ_ss are required to handle noise large enough such that Ω_0 overlaps with > 1 regions of Ω_{C-1}.

For typical software implementations, $\tau_s = 2^s$, which monotonically increases with increasing s. However, to reduce wiring congestion and ease ADC layout [Fig. 8(b)], instead of powers of two, $\tau_0 = 1$, $\tau_1 = 3$, and $\tau_2 = 7$ are chosen for hardware implementation. Successive approximation is able to produce refined estimates of ω, even when non-power-of-two τ_ss are used. Fig. 5 illustrates how it is possible to produce a better estimate of ω using τ_ss that increment at a faster rate than 2^s (as in our implementation), despite higher input uncertainty.

Finally, the 0th delay chain in each cluster delays the subsampling clock by an amount d_s, a number pseudo-randomly chosen between 0 and $n-1-\max\tau_s$ to satisfy RIP. Therefore, the rth delay chain ($r = 0, 1$) in the sth cluster has a total sample delay of $d_s + r\tau_s$. In our $C = 3$ hardware implementation, $d_0 = 0$, $d_1 = 6$, and $d_2 = 12$, and a total of $D = 6$ delay chains are used per stage, with sample delays of 0, 1, 6, 9, 12, and 19.

Hardware to perform successive refinement and obtain an estimate of a potential signal location j_{est} is described in Section IV-D and illustrated in Fig. 15(a). Once j_{est} is found for a given stage i and bin b, a signal estimate is obtained by rotating the time-shifted bin observations back and averaging; that is,

$$v_j = \vec{a}^{\dagger}_{j_{\text{est}}} \vec{Y}_{n_i,b}/D\,, \tag{10}$$

where $\dagger$ indicates the conjugate transpose. Finally, the bin contains a singleton if

$$\|\vec{Y}_{n_i,b} - v_j \vec{a}_{j_{\text{est}}}\|^2 < T_{\text{noise}}. \tag{11}$$

Otherwise, it is a multiton, because the location estimate does not explain the actual bin observations.

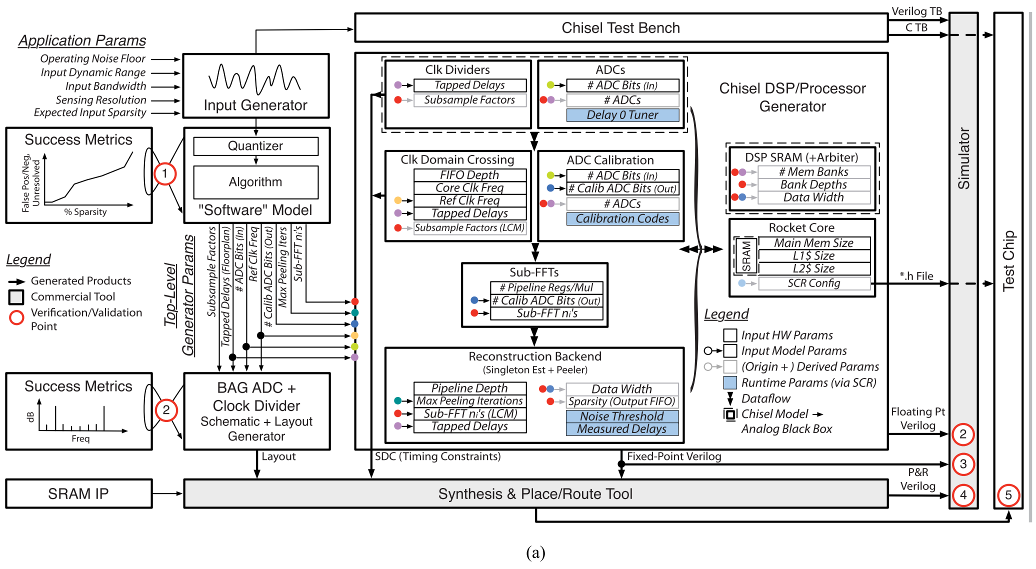
4) Simulated Algorithm Limits: When the FFAST architecture is fixed, the designer must provision for the worst case sparsity. To support a more realistic sparsity of ~3.6% for $n = 21\,600$, $n_1 = 2^5 \times 3^3 = 864$, $n_2 = 2^5 \times 5^2 = 800$, and $n_3 = 3^3 \times 5^2 = 675$ are used. A floating-point FFAST was implemented in MATLAB to determine upper bounds on algorithm performance. The results using noisy inputs are shown in Fig. 6. The percentage of false negatives remains at a reasonable level for input sparsities below 8.5%. Above 8.5% sparsity, the percentage of false negatives dramatically increases due to the FFAST architecture's inability to further resolve multiton bins. The subsampling factor can be reduced to improve this metric, at greater hardware cost. In addition, as more tones are introduced, the number of frequency collisions increases, requiring more peeling iterations to decode the spectrum. However, above the ~8.5% sparsity "cliff," the number of peeling iterations rapidly drops off, because the algorithm has been written to terminate early when no additional multiton bins can be resolved.

III. System-Level Generator Methodology

A. Parameterized Hardware Templates

The noise-robust FFAST algorithm described in Section II-B is mapped onto a fixed-function hardware accelerator for a general-purpose processor by using a generator-based design methodology. Digital hardware generators are built using the Constructing Hardware in a Scala Embedded Language (Chisel) hardware construction language [16], A Chisel Environment for DSP (ACED) library for hardware signal processing [17], and the Flexible Intermediate Representation for RTL (FIRRTL) compiler [18], whereas analog generators are built using the Berkeley Analog Generator (BAG) framework [19].

Designing an accelerator for a complex algorithm involves an iterative procedure of tuning the software-mapped algorithm and its hardware realization, trading off supported operating conditions and performance (e.g., compute time and false positive/negative rates per Fig. 24) for hardware complexity. With a traditional design methodology, this involves a cumbersome process of design-space exploration, requiring the manual creation of a series of design instances to match different algorithm realizations. In contrast, this endeavor uses hardware generators, which are highly parameterized models of building blocks, to rapidly explore the design space, validate performance, and verify the designed instance. FFAST is modeled in software to determine the system's operating conditions [e.g., input bandwidth, sparsity, dynamic range (DR), noise, and so on, as shown in Fig. 7(a)], which are used to tune hardware parameters at compile time to meet



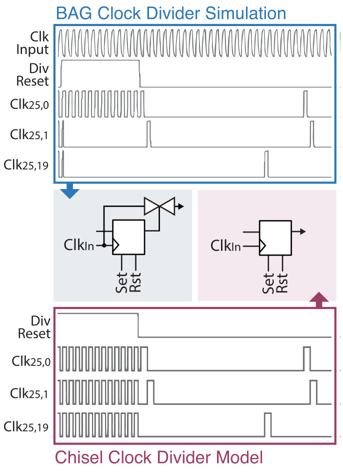


Fig. 7. (a) Hierarchically constructed FFAST hardware generator, consisting of a software model, analog ADC and clock divider templates, Chisel-based DSP generators, and the Rocket processor generator. (b) An analog simulation of clock divider inputs/outputs is used to generate a Chisel clock divider model.

performance goals. For FFAST, such parameters include the ADC reference clock frequency, ADC bitwidth, sub-FFT n_is, and the number of tapped delays required to achieve noise robustness.

Fig. 7(a) indicates that top-level model parameters and generic hardware parameters (e.g., pipeline depth) are propagated down to templates of key hardware blocks: clock dividers, ADCs, mixed-radix FFTs, coordinate rotation digital computer (CORDIC) units, and so on for hardware generation. To prevent infinite loops in the generated instance, Fig. 6(b) sets the maximum number of peeling iterations to attempt before the input spectrum is deemed unrecoverable. The number of ADCs is a function of the number of tapped delays and sub-FFT n_is, where n_is are chosen to support the expected input sparsity (Section II-A). The number of memory banks used for digital signal processing (DSP) and the banks' corresponding depths are calculated from the number of parallel memory accesses required by the sub-FFTs [20]. The data width along a processing chain is determined by the corresponding ADC bitwidth and FFT processing gain, such that quantization does not significantly affect the system's ability to recover sparse spectra. Finally, as mentioned in Section II-B3, particular tapped delay values are chosen for noise resiliency and set by floorplan and routing constraints.

While many system parameters can be defined at compile time, some "runtime" parameters can only be defined after characterizing the fabricated silicon. For example, ADC calibration codes are computed after implicitly measuring the nonlinearity and gain/offset error—affected by the fabrication process—of each ADC. Similarly, instead of attempting to simulate routing-induced lane-to-lane skew between ADC clocks (affected by parasitics and process variations), the relative delays of known signals are measured and used to tune peeling parameters. Finally, the noise threshold used by the singleton estimator can be adapted to different runtime operating conditions. The ability to support these parameters is provided by mapping the DSP templates' runtime configuration options to the Rocket processor's [21] status and control registers (SCRs), which are automatically generated.

B. Propagation of Designer Intent

When developing complex mixed-signal systems, it is important to propagate designer intent down the design hierarchy in a way that minimizes algorithm-to-hardware mismatches [17]. This is achieved with a unified generator framework that supports the automatic generation of Verilog and Synopsys Design Constraints (SDC) for timing from system parameters. Generated products are directly used by synthesis tools or applied in post-silicon verification. As an example, C header files that describe SCR address mappings are used to expedite the verification process.

To ensure the compatibility of analog and digital blocks, Chisel modules that model the behaviors of the ADCs, clock dividers, and static random access memory (SRAM) are used in simulation. Parameterization and a FIRRTL compiler pass [18] allow the simulation models to be replaced by black-box analog/SRAM layouts during place and route. Individual successive approximation register (SAR) ADCs are modeled by simple sample-delayed quantizers. The quantizer is created from an ACED floating-point-to-fixed-point converter [17] that wraps simulation-only SystemVerilog constructs. As shown in Fig. 13, to trigger data capture and coordinate ADC-to-DSP data alignment, a FIFO (first-in, first-out buffer) alignment circuit generates "valid" signals using inputs/outputs to/from the associated analog clock divider. Fig. 7(b) shows that the outputs of the flip-flop-based clock divider share the reference clock's pulsewidth. Because the FIFO alignment circuit shown in Fig. 13 only uses the clocks' rising edges, the Chisel model is able to directly use the flip-flop outputs (with twice the reference clock's pulsewidth) while guaranteeing

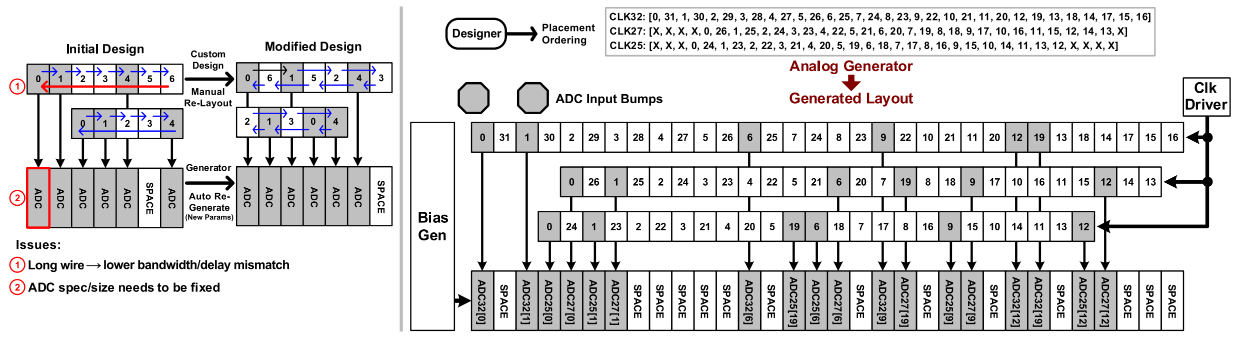

Initial Design
Custom Design
Modified Design
Manual Re-Layout
Generator
Auto Re-Generate (New Params)
ADC
SPACE
Issues:
① Long wire → lower bandwidth/delay mismatch
② ADC spec/size needs to be fixed
Designer
Placement Ordering
CLK32: [0, 31, 1, 30, 2, 29, 3, 28, 4, 27, 5, 26, 6, 25, 7, 24, 8, 23, 9, 22, 10, 21, 11, 20, 12, 19, 13, 18, 14, 17, 15, 16]
CLK27: [X, X, X, X, 0, 26, 1, 25, 2, 24, 3, 23, 4, 22, 5, 21, 6, 20, 7, 19, 8, 18, 9, 17, 10, 16, 11, 15, 12, 14, 13, X]
CLK25: [X, X, X, 0, 24, 1, 23, 2, 22, 3, 21, 4, 20, 5, 19, 6, 18, 7, 17, 8, 16, 9, 15, 10, 14, 11, 13, 12, X, X, X, X]
Analog Generator
Generated Layout
ADC Input Bumps
Clk Driver
Bias Gen
(a)
(b)

Fig. 8. (a) Example FFAST analog frontend layout when ÷7 and ÷5 subsampling factors are used, and delays 0, 1, and 4 are tapped out. (b) Floorplan of the FFAST analog frontend, which is designed to minimize routing congestion. A layout is automatically generated using delay cell orderings provided by the designer.

downstream functional correctness and timing compatibility. In addition, the Chisel clock divider model accounts for the fact that $\text{clk}_{x,0}$ oscillates at the reference clock frequency during reset. Although the analog and digital blocks are designed independently, Chisel models enable the verification of the entire analog and digital processing chain.

C. Unified System Validation and Verification

To verify that designer intent is properly passed down, the FFAST generator framework supports the same set of tests at all phases of design. Input stimuli used in the initial system modeling can be applied to the functional verification of the generated hardware. System- or block-level verification is performed using a custom DSP testing interface [17] that interprets the inputs/outputs of Verilog modules with non-synthesizable floating point constructs. By changing a single top-level parameter, the generator framework and testing infrastructure can also produce and simulate Verilog with fixed-point signals, helping to validate or tune the choice of DSP data widths. The Chisel DSP tester generates a stand-alone Verilog test bench that mirrors sequentially applied data inputs and output verification steps for post-place-and-route gate-level verification. Because top-level hardware tests involve reading from and writing to SCRs, the tests can be directly replicated in C programs using the generated SCR header file and C write/read-verify helper functions. The C programs are loaded onto the Rocket processor for pre-silicon digital verification on an FPGA—using a block-RAM-centric hardware instance—or chip testing.

IV. Implementation Details of Generator-Based Hardware Blocks

A. Analog Generator Design Considerations

Section III highlights the ease of algorithm-to-hardware translation, design-space exploration, and system verification afforded by a generator-based design methodology. An additional benefit of generator design—given sufficiently granular hardware templates—is the ability to "evolve hardware" by quickly re-customizing various levels of hierarchy to accommodate new features or mitigate discovered design flaws and non-idealities. For example, the generator approach supports the optimization of analog floorplanning late into the design cycle.

As described in Section II-B3 and illustrated in Fig. 10, to improve the noise resiliency of the FFAST architecture, ADC subsampling is performed with sampling delays of 0, 1, 6, 9, 12, and 19. These delays are obtained by tapping outputs of various delay cells in a flip-flop-based clock divider. Although the choice of sampling delays constrains the divider's floorplan to eliminate wiring congestion and simplify routing, an optimal floorplanning strategy is not immediately obvious. If the illustration shown in Fig. 10 is naively used to direct floorplanning, a delay mismatch penalty would be incurred, and a suboptimal clock divider bandwidth would be achieved. Although wires connecting delay cells in the forward direction have equal length, the return wire is significantly longer. For a simpler example with delay offsets of 0, 1, and 4, Fig. 8(a) shows how the divider's delay cells can be rearranged to improve circuit performance. Wire lengths within a loop can be (roughly) equalized by folding and interleaving high- and low-offset delay cells and shifting the lower set of delay cells horizontally, while clock distribution wires from the tapped outputs remain congestion-free.

A BAG analog template encapsulates complex physical design rules (e.g., in a 16-nm FinFET process), so that analog cells—such as ADCs and delay elements—may be automatically placed on a grid and signal wires can be routed to/from them [22]. The placements of the cells and lengths of the routing wires are automatically calculated when the cells' dimensions are known and a placement strategy is provided. The dimensions of the ADC slices and delay elements are determined by an application's input resolution and bandwidth requirements. However, because the placement order of the clock divider's delay cells is a tunable parameter, a naive floorplan can be easily replaced to mitigate design issues. The analog generator automatically produces the final layout illustrated in Fig. 8(b).

The flip-flop-based delay cells in the FFAST analog template are derived from those used in [22], although several FFAST-specific changes have been made to the clock divider architecture. Fig. 7(b) shows that, as in [22], tapped delays are not taken directly from the flip-flop outputs. Instead, the flip-flops control transmission gates that pass the

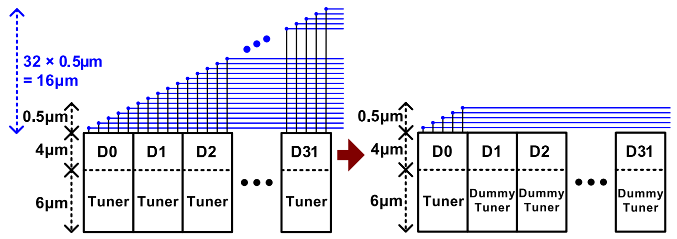


Fig. 9. Since non-ideal sampling delays are not tuned at the analog frontend (and are compensated for directly in the FFAST backend), delay tuners used in [22] are not needed. The generator can programmatically replace the tuners with dummies and remove corresponding control wires to reduce routing congestion and save area.

reference clock through when enabled. This reduces the flip-flops' jitter contributions at the divider outputs. The circuit in [22] uses a double data rate (DDR) divider structure that supports a differential reference clock running at half the effective ADC sampling frequency. In order to accommodate odd delay offsets, the FFAST clock divider instead uses a single-ended, full-rate reference clock. The performance of the time-interleaved ADC in [22] is improved with custom circuitry that finely tunes out delay mismatch across ADC lanes. As shown in Fig. 9, the delay tuner is roughly $1.5\times$ larger than the nominal delay cell, and control wires add approximately 0.5 μm of routing headroom per cell. If the tuners are used in the FFAST clock dividers, the overall routing penalty for a $\div 32$ clock divider would be 16 μm. Because delay mismatches are compensated digitally rather than corrected at the analog frontend, as alluded to in Section III-A, delay tuners are replaced with unconnected dummy cells—for alignment purposes—at all but the D0 delay cell, resulting in a routing overhead of only 0.5 μm and reduced wiring congestion.

Like the aforementioned delay cells, individual ADC slices are reused from the generator in [22]; however, the FFAST design has a lower input bandwidth, because 18 ADCs load the input, compared with 8 ADCs in [22]. In general, because of the sheer number of ADCs used by the FFAST architecture, the variation in distances between input sources (the reference clock driver, ADC input, and bias generator) and the clock divider/ADC blocks contributes more significantly to the spread of the ADCs' effective number of bits (ENOBs) and delay mismatches. As an example, the ADCs furthest from the bias generator have noticeably lower ENOBs than ADCs closest to it.

B. Analog Frontend From a System Perspective

In the analog frontend illustrated in Fig. 10, a flip-flop-based clock divider generates three different clock frequencies from a 3.78-GHz reference for CRT-guided subsampling. Six unique phase offsets are used per clock frequency for singleton/multiton disambiguation and noise resilience. The clock frequencies have (pairwise-coprime) $\div 25$, $\div 27$, and $\div 32$ relationships with the reference frequency. The τ_0, τ_1, and τ_2 delay deltas create the three delay clusters (themselves containing two delays) used for the successive approximation procedure in singleton estimation. These clocks run 18 lanes of asynchronous, constant common mode (V_{CM}), switched-capacitor SAR ADC slices generated using the BAG template [19] (Fig. 11) described in Section IV-A. The ADCs have 9-wire outputs, where <radix-2 is used for the 3 most significant bits (MSBs) to provide redundancy against missing decision levels, which can only be corrected in the analog domain with capacitor tuning. The resultant missing output codes, corresponding to the discontinuities in the mean input versus raw ADC code plot in Fig. 12(a), are corrected digitally by re-weighing the output bits [23]. To calibrate the ADCs, calibration lookup tables (LUTs) are initially programmed so that data and address values have a 1:1 correspondence. A known sinusoidal input is fed into the FFAST analog frontend, and sampled outputs are passed through to the DSP memory. In the calibration mode, the sequence of ADC subsampling $\rightarrow$ running sub-FFTs $\rightarrow$ reconstruction is interrupted immediately after ADC readout, so that the Rocket processor can collect ADC outputs for analysis. Coefficients that maximize the signal-to-noise-and-distortion-ratio (SNDR), determined via a least-squares fit to the sinusoidal input, are then used to generate calibration LUT parameters. Thus, during normal operation, mismatch corrections (in addition to gain/offset corrections between ADC slices) can be performed on-the-fly before padded 8-bit data are stored into memory. The 9-bit raw ADC outputs address the LUTs (totaling 10.4 kB of SRAM), and 8-significant-bit remapped outputs with the improved mean input versus calibrated ADC code characteristics shown in Fig. 12(b) are captured. Note that calibration accuracy can be improved if more memory is allocated for data averaging.

Because the divided clocks are aligned once every 21 600 cycles—the least common multiple (LCM) of the subsampling factors—an alignment circuit (Fig. 13) generates valid signals which control the asynchronous FIFOs at the 18 subsampling-clock-to-core-clock boundaries. The digital core clock runs at 400 MHz regardless of the ADC clock frequencies, necessitating data transfer across the FIFOs. The state of the alignment circuit stabilizes sometime after the initial ADC reference clock reset without regard for the DSP state, so handshake bits are used to ensure that downstream processing is able to receive properly aligned frames of ADC data. The data are mapped to SRAM banks and addresses via a LUT version of the index vector generator described in [20]. Accounting for delay redundancy added to enhance noise robustness (i.e., the six delay shifts), the ADC set samples 35% fewer time-domain points compared to a 3.78-Gs/s full-rate ADC. There is a tradeoff between the number of delay offsets used for noise robustness and performance degradation due to mismatches in bandwidth from process, voltage, and temperature (PVT) variations and spatial locality design limitations spanning the ADC lanes.

C. Sub-FFTs

The subsampled ADC outputs are fed into complex (20, 20)-bit, *mixed-radix*, 864-, 800-, and 675-point FFTs, corresponding to n_1, n_2, and n_3, respectively. Due to delay redundancy and the bitwidth growth (to 10 integer bits and additional fractional guard bits) needed to accommodate FFT processing gains, 70.2 kB of memory are required.

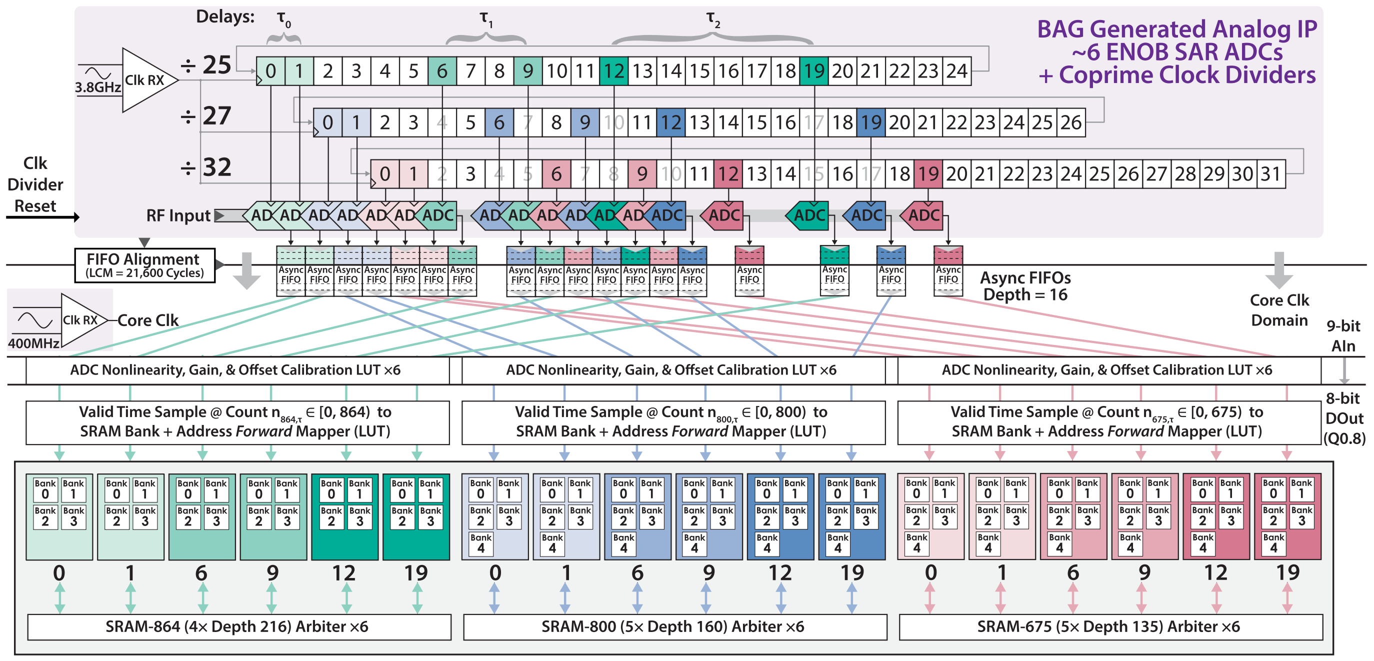

BAG Generated Analog IP
~6 ENOB SAR ADCs
+ Coprime Clock Dividers
Delays: τ0 τ1 τ2
3.8GHz
Clk RX
÷ 25
÷ 27
÷ 32
Clk Divider Reset
RF Input
FIFO Alignment (LCM = 21,600 Cycles)
Async FIFO
Async FIFOs Depth = 16
Core Clk Domain
400MHz
Core Clk
9-bit AIn
ADC Nonlinearity, Gain, & Offset Calibration LUT ×6
8-bit DOut (Q0.8)
Valid Time Sample @ Count n864,τ ∈ [0, 864) to SRAM Bank + Address Forward Mapper (LUT)
Valid Time Sample @ Count n800,τ ∈ [0, 800) to SRAM Bank + Address Forward Mapper (LUT)
Valid Time Sample @ Count n675,τ ∈ [0, 675) to SRAM Bank + Address Forward Mapper (LUT)
SRAM-864 (4× Depth 216) Arbiter ×6
SRAM-800 (5× Depth 160) Arbiter ×6
SRAM-675 (5× Depth 135) Arbiter ×6

Fig. 10. Analog frontend with shift-register-based clock dividers that generate (non-50% duty cycle) 151.2, 140, and 118.125-MHz subsampling clocks with delay offsets of 0, 1, 6, 9, 12, and 19 from a 3.78-GHz source. Data from associated 9-wire, <radix-2, SAR ADC slices are time synchronized and stored in memory using 8 significant bits, following capacitor mismatch, offset, and gain correction.

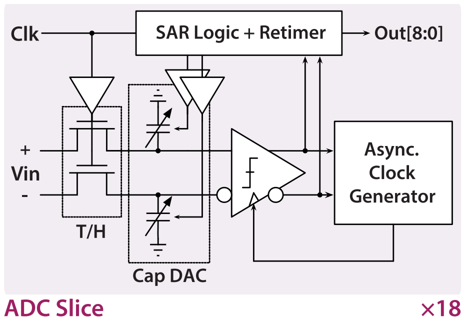


Fig. 11. Asynchronous SAR ADC slice. The slice design is described in [22] and reused as the unit cell in the FFAST subsampling ADC template.

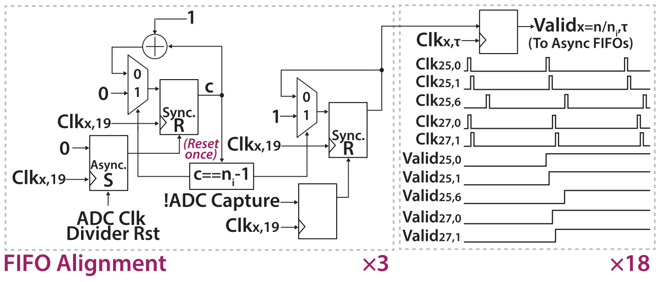


Fig. 13. ADC-to-core FIFO alignment circuit (aligned every 21 600 ADC reference clock cycles). The subsampling factor for stage i is denoted $x = n/n_i$.

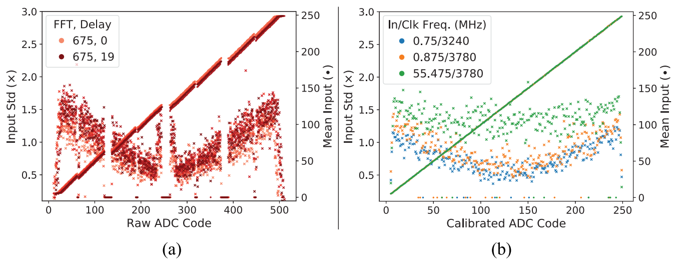


Fig. 12. (a) Fitted input (and its standard deviation) versus *raw* ADC code. Missing codes are due to reduced-radix capacitor weights. Gain and offset mismatches across different ADC slices are evident. (b) Fitted input (and its standard deviation) versus *calibrated* ADC code. The post-processed output has 8 significant bits, and missing codes are mostly eliminated. A 0.875-MHz sinusoidal input and a 3.78-GHz reference clock are used for calibration.

The decimation-in-frequency, non-2^n sub-FFTs are generated via a Chisel DSP [16], [17] FFT generator [20]. As shown in Fig. 14, each sub-FFT stage utilizes a conflict-free memory access scheme supporting one iterating processing element (PE) per FFT lane. Each lane's runtime reconfigurable butterfly unit reallocates hardware blocks to support multiple radices (e.g., 4, 2, and 3 for the 864-point FFT). Control logic is shared between the six lanes associated with a subsampling stage, and memory banks are calculated via mixed-radix counters and simple subtract/mux-based modulo units. FFT input–output unscrambling (i.e., the index-to-memory-bank/address mapping) is done by muxing between LUTs whose values are associated with in-order inputs and outputs indexed in quasi-digit-reversed order, as described in [20]. The fixed-point representations of the complex outputs are normalized by the sub-FFT size and reinterpreted (so that the location of the binary point is changed, but the bitwidth stays the same). Therefore, rather than dividing by n_i for re-normalization, the location of the binary point is shifted to the left by 9 places (corresponding to a divide by 512), and the output is instead multiplied by $512/n_i$. Normalization is necessary for digital reconstruction.

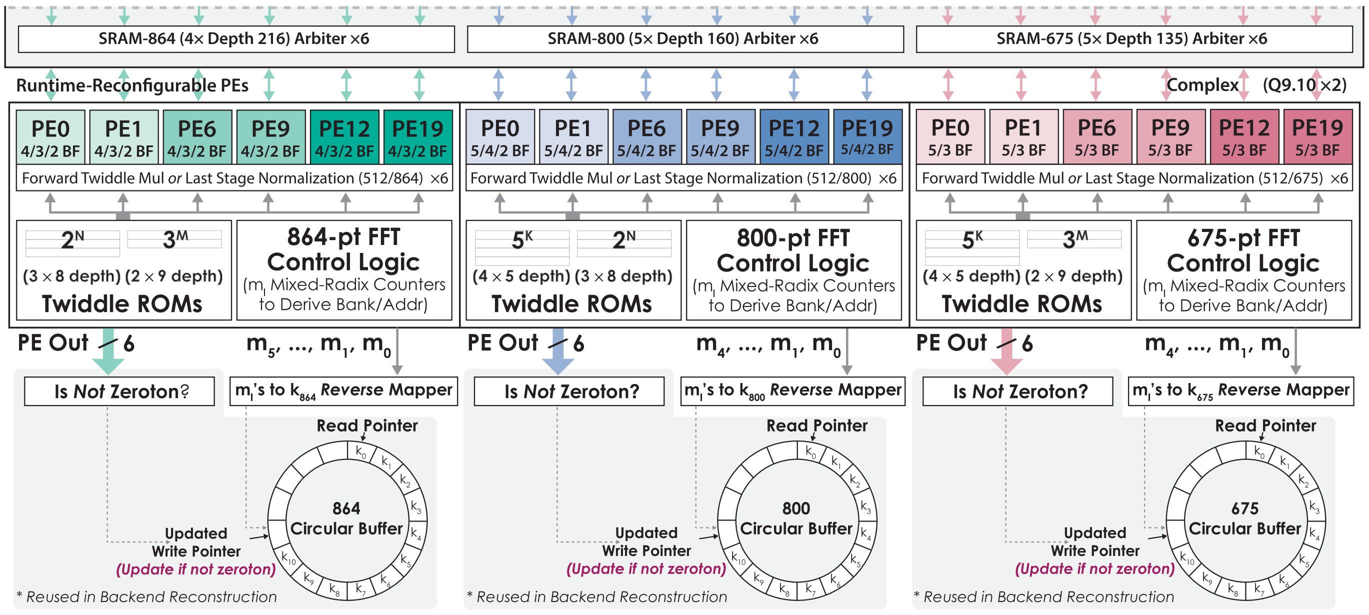
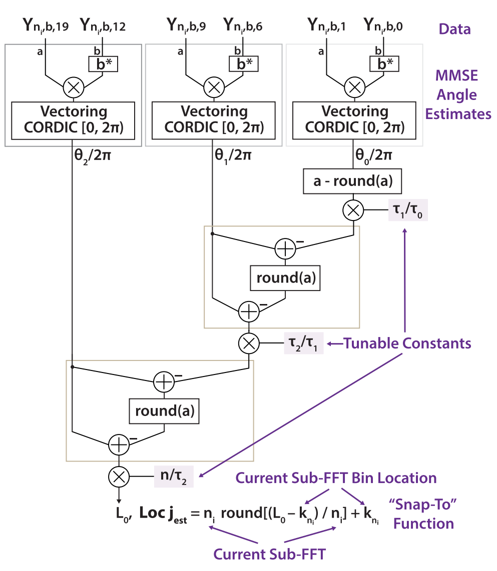
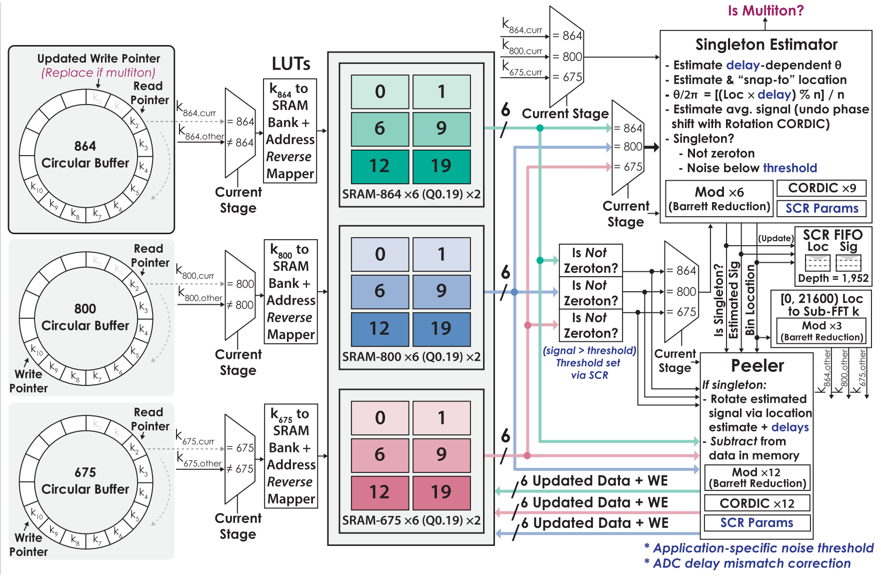

SRAM-864 (4× Depth 216) Arbiter ×6
SRAM-800 (5× Depth 160) Arbiter ×6
SRAM-675 (5× Depth 135) Arbiter ×6
Runtime-Reconfigurable PEs
Complex (Q9.10 ×2)
PE0 PE1 PE6 PE9 PE12 PE19 4/3/2 BF
PE0 PE1 PE6 PE9 PE12 PE19 5/4/2 BF
PE0 PE1 PE6 PE9 PE12 PE19 5/3 BF
Forward Twiddle Mul or Last Stage Normalization (512/864) ×6
Forward Twiddle Mul or Last Stage Normalization (512/800) ×6
Forward Twiddle Mul or Last Stage Normalization (512/675) ×6
2^N 3^M (3 × 8 depth) (2 × 9 depth) Twiddle ROMs
864-pt FFT Control Logic (m_i Mixed-Radix Counters to Derive Bank/Addr)
5^K 2^N (4 × 5 depth) (3 × 8 depth) Twiddle ROMs
800-pt FFT Control Logic (m_i Mixed-Radix Counters to Derive Bank/Addr)
5^K 3^M (4 × 5 depth) (2 × 9 depth) Twiddle ROMs
675-pt FFT Control Logic (m_i Mixed-Radix Counters to Derive Bank/Addr)
PE Out 6
m_5, ..., m_1, m_0
m_4, ..., m_1, m_0
Is Not Zeroton?
m_i's to k_864 Reverse Mapper
m_i's to k_800 Reverse Mapper
m_i's to k_675 Reverse Mapper
Read Pointer
864 Circular Buffer
800 Circular Buffer
675 Circular Buffer
Updated Write Pointer
(Update if not zeroton)
* Reused in Backend Reconstruction

Fig. 14. Memory-based, decimation-in-frequency, 864-, 800-, and 675-point FFT blocks with shared control logic and runtime-reconfigurable butterflies, derived from the generator in [20]. Abusing notation, $b = k_{n_i}$ corresponds to the current sub-FFT bin; the same bin is also represented as k_u, where u indicates the bin's position in the CB associated with stage i.


Y_{n_i,b,19} Y_{n_i,b,12} Y_{n_i,b,9} Y_{n_i,b,6} Y_{n_i,b,1} Y_{n_i,b,0}
Data
MMSE Angle Estimates
Vectoring CORDIC [0, 2π)
θ_2/2π
θ_1/2π
θ_0/2π
a - round(a)
τ_1/τ_0
round(a)
τ_2/τ_1
Tunable Constants
n/τ_2
Current Sub-FFT Bin Location
L_0, Loc j_est = n_i round[(L_0 − k_{n_i}) / n_i] + k_{n_i}
"Snap-To" Function
Current Sub-FFT

(a)


Updated Write Pointer
(Replace if multiton)
Read Pointer
864 Circular Buffer
800 Circular Buffer
675 Circular Buffer
Write Pointer
LUTs
k_864 to SRAM Bank + Address Reverse Mapper
k_800 to SRAM Bank + Address Reverse Mapper
k_675 to SRAM Bank + Address Reverse Mapper
Current Stage
SRAM-864 ×6 (Q0.19) ×2
SRAM-800 ×6 (Q0.19) ×2
SRAM-675 ×6 (Q0.19) ×2
Is Multiton?
Singleton Estimator
- Estimate delay-dependent θ
- Estimate & "snap-to" location
- θ/2π = [(Loc × delay) % n] / n
- Estimate avg. signal (undo phase shift with Rotation CORDIC)
- Singleton?
- Not zeroton
- Noise below threshold
Mod ×6 (Barrett Reduction)
CORDIC ×9
SCR Params
SCR FIFO Loc Sig
Depth = 1,952
[0, 21600) Loc to Sub-FFT k
Mod ×3 (Barrett Reduction)
Is Not Zeroton?
(signal > threshold) Threshold set via SCR
Is Singleton?
Estimated Sig
Bin Location
Peeler
If singleton:
- Rotate estimated signal via location estimate + delays
- Subtract from data in memory
Mod ×12 (Barrett Reduction)
CORDIC ×12
SCR Params
6 Updated Data + WE
* Application-specific noise threshold
* ADC delay mismatch correction

(b)

Fig. 15. (a) Successive approximation used to improve the singleton estimator's signal location estimate j_{est} from the angle deltas (obtained via CORDIC [24]) of time-shifted input samples. Data are retrieved from sub-FFT n_i memories for all clock delays. (b) Peeling reconstruction backend with a singleton estimator. $\vec{Y}_{n_i,b}$ used by the singleton estimator comes from the bin location $b = k_{n_i}$ associated with sub-FFT stage i.

D. Singleton Estimation and Peeling Reconstruction

Sub-FFT output bins that only contain noise are predetermined via thresholding and discarded from future analysis. The locations of the remaining bins, which may either be singleton or multiton, are stored in per-sub-FFT-stage CBs (as k_u in Fig. 14, where u represents the bin's index in a stage's CB). In total, the CBs occupy 2.9 kB of SRAM. As described in Section II, because signals at different frequencies do not rotate by the same amount when delayed ($\langle\theta\rangle_{2\pi} = 2\pi\langle j\tau\rangle_n/n$), time-shifted versions of the input are collected to distinguish between singleton bins and multiton bins. As shown in Fig. 15(a), a vectoring CORDIC is used to estimate the phase difference $\theta_s = 1\omega_s$ between two versions of the subsampled signal with a delay delta of τ_s. Successively larger non-power-of-two τ_ss are used to obtain a better estimate of the phase rotation $\theta_0 = \omega$ associated with $\tau_0 = 1$ and, thus, the signal's potential frequency location j. As shown in Fig. 15(a), the estimated frequency location

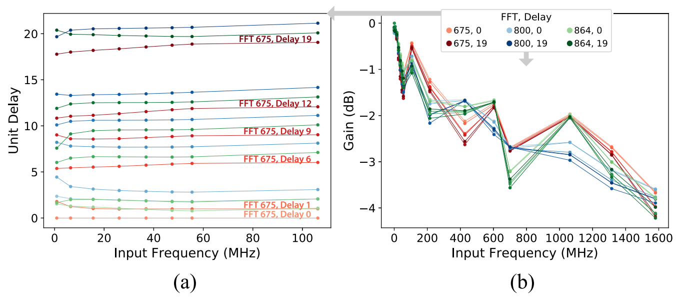


Fig. 16. Note—For plots where FFT and delay are labeled (where each ADC output is associated with an FFT and delay offset), darker shades of the same color correspond to larger unit delay offsets in the set {0, 1, 6, 9, 12, 19} used by a single FFT stage. (a) Measured sample delay offsets across different ADC slices and input frequencies using a 3.78-GHz ADC reference clock. Backend delay correction is required, as dc lane-to-lane skew shifts the delays away from ideal. (b) Gains across different ADC slices and input frequencies (including frequencies above an ADC slice's sampling rate) using a 3.78-GHz ADC reference clock.

is refined with knowledge of the subsampled bin location $b = k_{n_i}$ $(= k_u)$. The estimate, whose accuracy is affected both by thermal noise and quantization in the ADC/DSP chain, is snapped to the nearest feasible j, as dictated by aliasing. Fig. 15(b) shows that, subsequently, a set of rotation-mode CORDICs undoes the sampling-delay-based phase rotations applied to the observation set, so that the observations can be averaged to obtain a signal estimate. Finally, singletons are determined by a runtime-reconfigurable decision threshold.

As illustrated in Figs. 4 and 15(b) and described in Section II-B2, singletons discovered in stage i are iteratively peeled off by removing their bin locations b from the ith stage CB and subtracting their contributions from associated bins in other sub-FFT stages. If a subsampled bin is still unresolvable after singleton estimation, b is placed back into the CB. This peeling process is terminated early when any CB is empty or no new signal information is uncovered (i.e., when the lengths of all CBs have not changed after a peeling iteration). As singletons are found, their locations j and signal values are output into FIFOs readable by the processor via SCR. FIFO depths are sized to accommodate up to 9% sparsity, requiring 13.4 kB of SRAM. The unsorted frequency location/data output is compressed to the same order of magnitude as the input sparsity (4.4% for an input spectrum with 3.2% sparsity, accounting for the storage of 15-bit-wide j indices not required for normal FFTs).

Singleton estimation and peeling decoding do not require skew-specific layout optimization or timing calibration at the ADC outputs, because the (dc) lane-to-lane skew can be empirically determined after chip fabrication, as shown in Fig. 16(a). Lane-to-lane skew is compensated digitally via adjustments to the absolute delay shifts $(d_s + r\tau_s)$ and τ_ss stored in the Rocket processor's SCRs [see Fig. 15(a)]. A sine fit is used to calculate the real sample delay offsets given limited bandwidth mismatch between ADC slices. Note that the accuracy of an estimated delay offset is limited by thermal noise (mitigated by averaging) and quantization in the fixed-point processing steps used to calculate the offset. In addition, the amount of bandwidth mismatch is estimated by observing the gain differences between the 18 FFT outputs across the input bandwidth,

Algorithm 2: CORDIC [24]

```
Input  : $x$, n bits, signed.
Input  : $y$, n bits, signed.
Input  : $\theta$, n bits. The signed interpretation is
         $\theta_s \in [-\pi, \pi)$. It is equivalently represented as
         unsigned $\theta_u \in [0, 2\pi)$.
Input  : isRotation. If not, then in vectoring mode.
Output: $x$, $y$, $\theta$

▷ CORDIC can rotate an input by a maximum of $\pm\pi/2$.
if (isRotation and $\frac{\pi}{2} < \theta_u < \frac{3\pi}{2}$) or (!isRotation and $x < 0$) then
    $x \leftarrow -x$
    $y \leftarrow -y$
    $\theta_s \leftarrow \theta_s - \pi$
end
for $i \leftarrow 0$ to $n-1$ do
    if (isRotation and $\theta_s \geq 0$) or (!isRotation and $y < 0$)
    then
        $d = +1$
    else
        $d = -1$
    end
    $x' \leftarrow x - 2^{-i} dy$
    $y \leftarrow y + 2^{-i} dx$
    $x \leftarrow x'$
    ▷ Angle constant properly normalized to bitwidth.
    $\theta_s \leftarrow \theta_s - \text{round}(2^n \arctan(2^{-i})/2\pi)d$
end
return $x$, $y$, $\theta$
```

as shown in Fig. 16(b). Gains do not monotonically decrease with higher input frequency, since they are not completely determined by the input samplers' bandwidths, and signals above individual subsampling frequencies are folded down.

1) Calculating x mod n*:* Modulo operations are used extensively in the singleton estimator and peeling decoder. When $x \leq 2n - 1$, minimal hardware can be used:

$$x \bmod n = \text{Mux}(x - n < 0, x, x - n). \tag{12}$$

However, if x is arbitrarily large and n is a constant, x mod n can be calculated using Barrett reduction [25]. Barrett reduction is suitable for hardware, since a divider is not needed. Instead, multiplication is performed with a pre-computed $1/n$. Barrett reduction is heavily used in the peeling reconstruction backend.

2) CORDIC: CORDIC is an iterative algorithm—requiring only additions, subtractions, and bit shifts—used by the singleton estimator and peeling decoder to determine the phase difference between two delayed samples (in vectoring mode) and rotate observations for signal estimation (in rotation mode). Its computation converges to the desired value at a rate of 1 bit per iteration. n-bit signed angles used by CORDIC span the full $[-\pi, \pi)$ interval, although they can likewise be interpreted as unsigned $[0, 2\pi)$. The pseudocode for CORDIC is found in Algorithm 2.

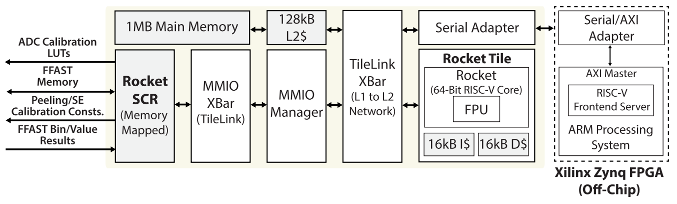


Fig. 17. Rocket RISC-V processor & Xilinx Zynq FPGA testing setup. FFAST status/control registers and data outputs are mapped to the memory space of the Rocket processor. C programs for analyzing data and testing the chip are uploaded to the processor through the Xilinx FPGA. Because main memory is limited to 1 MB of SRAM, compiled C code must also be < 1 MB.

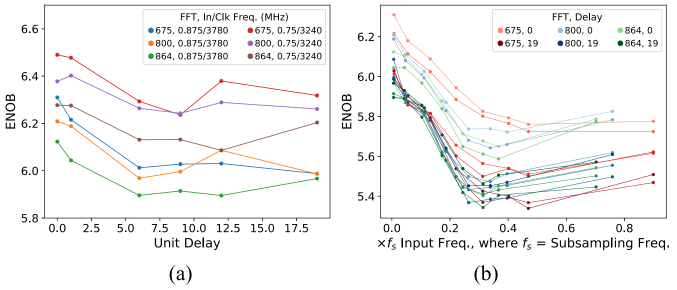


Fig. 18. (a) ENOB comparison using 3.24-GHz and 3.78-GHz reference clocks. ENOB degrades at higher clock frequencies. (b) ENOBs calibrated at different input frequencies (with a 3.78-GHz reference clock) for different ADC slices. ENOBs are higher at lower sample delay offsets due to lower V_{ref} noise. An ADC cell's ENOB is correlated with the distance between the cell and the bias generator, as shown in Fig. 8(b). In both (a) and (b), phase imbalance is tuned during testing.

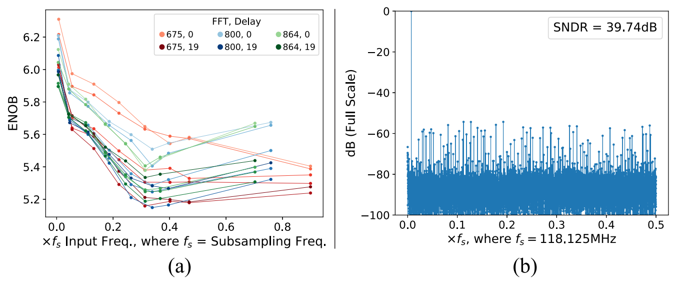


Fig. 19. (a) ENOBs with a 3.78-GHz reference clock, using calibration parameters taken with a single 0.875-MHz input. (b) Frequency spectrum at a sub-ADC output (0.875-MHz input). In (a) and (b), tuning the ADC's differential P and N phase imbalance reduces HD2.

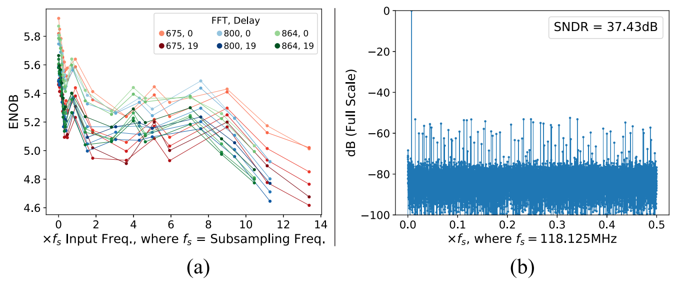


Fig. 20. (a) ENOBs with a 3.78-GHz reference clock, using calibration parameters taken with a 0.875-MHz input. A balun with ~5° of phase imbalance is used to measure SNDR at higher input frequencies. (b) Frequency spectrum at a sub-ADC output (0.875-MHz input). HD2 caused by the external balun's phase imbalance degrades SNDR.

E. Rocket Processor

To interact with the FFAST accelerator, the SoC includes a 64-bit fifth-generation reduced instruction set (RISC-V) Rocket core [21] with 1 MB of main memory and a 128-kB L2 cache (Fig. 17). In addition to supplying calibration and control information for adapting to the input environment and chip non-idealities, the Rocket core also post-processes data. It is simple to write a C program that collects and sorts the compressed FFAST frequency information and organizes it for human consumption (e.g., outputting signal bandwidths and locations). The processor can also aid the spectral analysis engine: if latency requirements are relaxed, the processor can window the ADC data to suppress spectral leakage. Finally, because Rocket has visibility into the sparse FFT memories, individual blocks in the signal processing chain like the SAR ADC slices and sub-FFTs can be repurposed for general applications.

The Rocket core interfaces with the FFAST DSP via memory-mapped SCRs. Interfaces heavily rely on the TileLink on-chip interconnect fabric. Although the main memory is on-chip for this application, the ARM frontend server on the Xilinx Zynq FPGA is used to asynchronously communicate with the chip (via a serial adapter) and load C tests.

V. Measurement Results

The ENOB was measured for ADC slices corresponding to different n_is and sampling delays using 3.24-GHz and 3.78-GHz reference clocks [Fig. 18(a)]. Initially, calibration LUT parameters were updated to maximize SNDR at individual input frequencies [Fig. 18(b)]. In addition, the positive and negative ADC inputs were tuned to remove phase imbalance and maintain a 180° phase offset.

Fig. 18(a) indicates that there is an ENOB degradation of approximately 0.2 bits at lower unit delays when using the higher frequency reference clock. This ENOB degradation is worse at higher unit delays, which are associated with ADC slices farther away from the reference supply. In general, as indicated in Fig. 18(b), ENOBs are better at lower delay offsets due to the corresponding ADCs' spatial proximity to V_{ref}, resulting in reduced noise.

Optimal calibration parameters are frequency-dependent, so SNDR can only be maximized at one frequency point in practice. To understand system performance in a real application setting, the ADCs are calibrated at an input frequency of 0.875 MHz. The calibration values are stored and used to determine ENOBs across the desired input bandwidth. Fig. 19(a) shows that the ENOB/SNDR is maximized around a 0.875-MHz input.

As in a real system, the analog frontend is characterized using a balun with approximately 5° of phase imbalance, resulting in SNDR degradation from the second harmonic distortion (HD2). The ENOBs achieved with a discrete balun are shown in Fig. 20(a). Fig. 19(b) shows the frequency spectrum at the output of one ADC when a balun is not used, whereas Fig. 20(b) shows the spectrum with a balun. In the

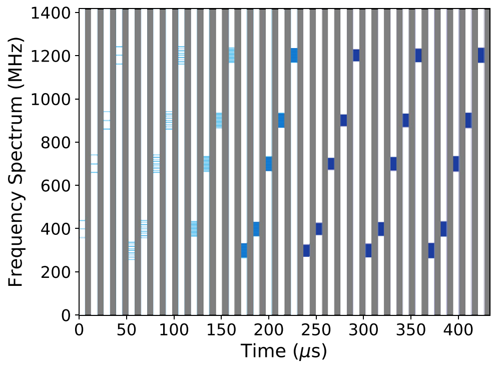


Fig. 21. Recovered spectra (blue) over time for 3- (left) to 37-tone (right, dense) vector signal generator outputs. The tones are spread out across an approximately 80-MHz bandwidth and have center frequencies ranging from 300 MHz to 1.2 GHz. Gray regions indicate observation "dead zones," when the FFAST hardware is busy analyzing the previous frame of data and unable to observe the current ADC input. A full signal acquisition and analysis cycle for up to 37 tones takes ~13.3 μs.

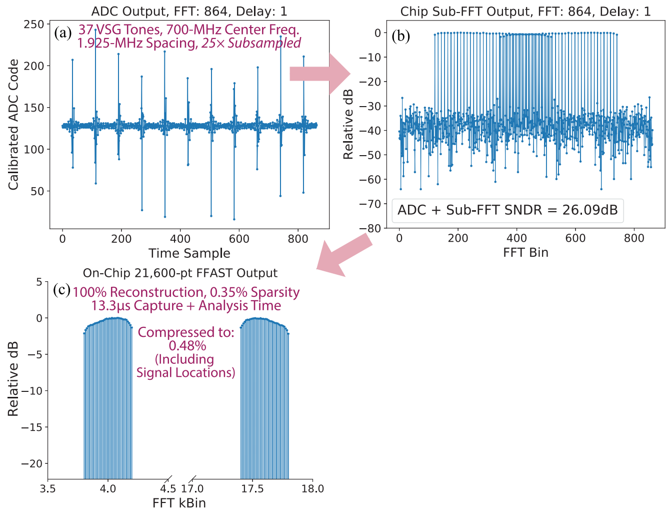


Fig. 22. Multi-tone signal at each FFAST stage. (a) 37 tones (corresponding to 0.35% sparsity), centered at 700 MHz, with a 1.925-MHz spacing, are generated from a vector signal generator. The time-domain waveform is subsampled 25× in the stage corresponding to an 864-point FFT. (b) Subsampled frequency domain result at the output of the 864-point FFT. Tones have been folded down to different bin locations due to aliasing. (c) FFAST is able to reconstruct 100% of the spectrum in 13.3 μs (including signal capture and analysis). The data are compressed to 0.48%, including 15-bit signal locations.

first FFT plot, the second harmonic of the input frequency is not readily observable, but it is one of the dominant contributors to the approximately 2.3-dB SNDR degradation seen in the latter plot.

Fig. 21 shows that functionality across the entire ADC/DSP chain has been verified by successfully recovering the spectrum of ~80-MHz bandwidth, 3- to 37-tone inputs generated by a vector signal generator at 300-MHz to 1.2-GHz center frequencies. The ADC input is not observed when analysis on a previous frame is being performed, corresponding to the gray periods in Fig. 21. The multi-tone signal at various stages of analog/digital processing is illustrated in Fig. 22(a)–(c). The subplots correspond to the time-domain output of one of the 18 ADC slices, the frequency-domain representation of the previous result (after the FFT), and the fully reconstructed

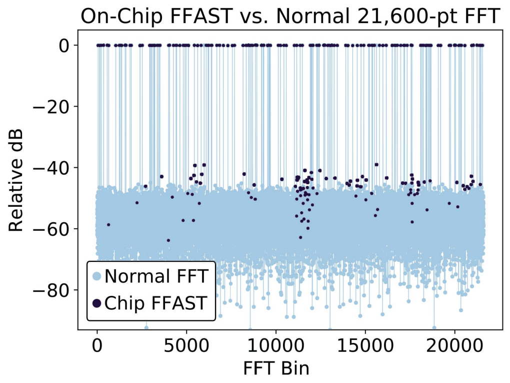


Fig. 23. FFAST versus normal FFT for C test vectors with 33.5-dB SNR and 0.79% sparsity. Reconstruction with 0% false negatives and 0.5% false positives. Note that although there are no false negatives, the magnitudes of two recovered points differ significantly from ideal.

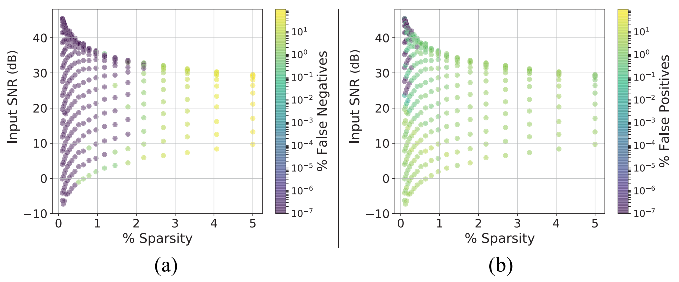
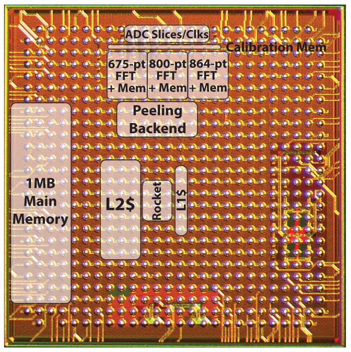


Fig. 24. Digital reconstruction false negative and false positive rates. (a) False negatives (worst case) remain below 5% for sparsities $< 2.7\%$. With respect to the number of false negatives, FFAST supports input SNRs > 9.7 dB for less populated spectra. (b) False positives remain below 5% for sparsities $< 5.0\%$. With respect to the number of false positives, FFAST supports input SNRs > 8.4 dB for less populated spectra.

sparse spectrum. Approximately 7.6 μs elapse between the end of signal acquisition and the start of spectrum availability. With 37 tones, the output is compressed to 0.48% when compared to the output of a full-length FFT.

The analysis of a synthesized 3.2% sparse spectrum (achieving 4.5% false negatives and 0.8% false positives) has been benchmarked. Approximately 4700 400-MHz Rocket core clock cycles elapse before the processor sees valid reconstruction data, corresponding to 11.75 μs of "real-time" processing. Approximately 1500 cycles are allocated for the sub-FFTs, and the remainder is used during peeling. The worst case number of clock cycles required by the sub-FFTs corresponds to the calculation of 864-point FFTs with single iterating butterflies (although 2 radix-2 butterflies operate simultaneously per FFT). The peeling time (and the number of iterations required) is a function of sparsity and increases when there are more frequency collisions. The total reconstruction time is 17.5 μs; signal acquisition accounts for 5.71 μs on top of the aforementioned processing time. This determines the minimum signal duration to guarantee SoC observability. An example of a spectrum generated from C test vectors is shown in Fig. 23. In this case, although the FFAST hardware is able to recover all real signals, some noise bins are also interpreted to be low-magnitude signal bins, resulting in false positives. These false positives can be cleaned up with an additional thresholding step during post-processing.

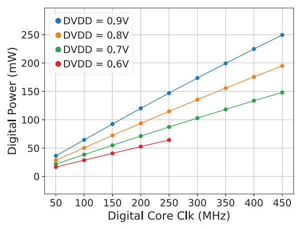


(a)

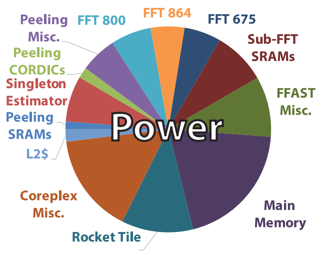
Technology	16-nm FinFET
Sensing BW/Resolution	1.89GHz/175kHz
Capture + Analysis Time	5.71μs + 11.8μs (3.2% sparsity, 4.5%/0.8% false -/+)
Compression (% Samples)	65% @ ADC, 4.4% @ FFAST Output (3.2% sparsity, signal dependent)
Sub-FFT SQNRs	675: 38-52dB, 800: 42-56dB 864: 41-56dB (signal dependent)
Area	0.27mm^2 Analog, 0.85mm^2 FFAST Digital, 0.37mm^2 Rocket Core (+L2$), 1.07mm^2 Main Memory
Memory	1MB Main Memory 128kB L2$, 2× 16kB L1$, 97kB FFAST
SAR ADC Power	49.8mW (@3.78Gs/s effective, 0.85V/0.95V ADC/Clk supplies, up to 6.3ENOB/slice)
FFAST DSP Power	68.3mW (@400MHz, 0.7V)
64-Bit Rocket-Chip Power	65.2mW (@400MHz, 0.7V)

(b)

Fig. 25. (a) Die photo with key hardware blocks highlighted. (b) Summary of FFAST hardware capabilities, area, and power consumption.



(a)



(b)

Fig. 26. (a) Power versus frequency/supply with an FFAST workload. A 0.7-V, 400-MHz digital operating point was chosen to reduce power consumption while maintaining a sufficiently fast compute time. (b) Post-place-and-route power breakdown, simulated with representative test vectors. The Rocket core and FFAST DSP consume similar amounts of power.

The percentages of false negatives and false positives for inputs with different sparsities and SNRs are illustrated in Fig. 24(a) and (b), respectively. In general, the system is more susceptible to false negatives, which remain below 5% for sparsities $<$ 2.7% and SNRs $>$ 9.7 dB. Unfortunately, false negatives tend to be the more application-critical metric. Therefore, further tuning of noise/signal thresholds can be performed to trade off a decrease in false negatives for an acceptable increase in false positives (which currently remain below 5% for sparsities $<$ 5%).

VI. Chip Summary

The described instance has been fabricated in a 16-nm FinFET process. Fig. 25(a) shows the chip die photo with key blocks highlighted, and Fig. 25(b) summarizes the FFAST hardware's capabilities, area, and power consumption. Fig. 26(a) has been used to choose a digital operating point of 400 MHz at a 0.7-V digital supply voltage (VDD), which allows for low power consumption and real-time operation. As highlighted in Fig. 26(b), the RISC-V Rocket core consumes 65.2 mW, and the FFAST DSP consumes 68.3 mW. The set of ∼6 ENOB ADCs (at 0.85 V) and associated clock dividers (at 0.95 V) consumes 49.8 mW/48.1 mW with a 3.78-GHz/3.24-GHz reference clock. At 3.78 GHz, the SAR ADC cores consume a total of 33.8 mW, while the clock dividers consume 16 mW.

TABLE I
Comparison With the State of the Art

	[5]	[26]	This Work
Technology	45nm	28nm	16nm FinFET
Input Bandwidth (GHz)	-	8.5	1.89
FFT Points	746,496	8,192	21,600
Integrated ADC	No	No	Yes
Power (mW), *incl. ADC	174.8	5,200*	133.5 + 49.8*
Supported Sparsity	0.1%	100%	3.2%
Can Post-Process	No	No	Yes
Data Compressed	Yes	No	Yes
Noise Robust	No	-	Yes
Approx. Runtime, *incl. ADC	6.8μs	-	17.5μs*
FOM (Pts. × BW) / Pwr.	-	13.4	222.7

VII. Discussion

Measurement results in Section V show that the supported sparsity in hardware is only ∼3.2%, although a floating-point MATLAB implementation achieves ∼8.5%. The ADCs and fixed-point arithmetic incur a quantization noise penalty. When noise is sufficiently high, k is effectively increased and sparsity degrades. However, this is not the primary source of the discrepancy.

Because of the computational complexity of the singleton estimator, individual operations are heavily pipelined. Since pipeline stalls were not implemented, occasionally, the value of a bin $b = k_{n_i}$, associated with a stage i, is requested before it has been updated in memory. The probability that this occurs increases as sparsity degrades, limiting the supportable sparsity of the system. Additional tones can be supported if pipeline stalls are properly inserted, although the addition of the stalls will slightly increase the analysis time. Eliminating the data hazard in the sparse FFT's reconstruction backend should roughly double the number of signals detectable in real time, as per Fig. 6.

To support more realistic, off-grid signals—which also effectively degrade sparsity—in real time, windowing can be performed at the frontend (either digitally or in the analog domain, using an architecture similar to that reported in [27]). Again, this requires accurate measurement of the dc lane-to-lane skew.

VIII. Conclusion

As described in [28] and [29], a set of highly parameterized analog and digital hardware generators enabled the design exploration and rapid 16-nm implementation of the FFAST algorithm in ∼2 months. This is, to the best of the authors' knowledge, the first automatically generated, fully integrated, sparse spectral analysis SoC, with a BAG-designed custom SAR ADC frontend, generated mixed-radix FFTs, digital reconstruction backend based off of the FFAST algorithm [30], and a RISC-V processor. The system targets realistic spectral sparsities and SNR, and the $<$ 20-μs runtime enables real-time adaptation in closed-loop systems. Table I shows the comparison of this work to a purely digital 746 496-point sparse FFT hardware implementation [5] and a recently published application-specific integrated circuit (ASIC) spectrometer using an 8192-point FFT [26].

REFERENCES

[1] H. Hassanieh, L. Shi, O. Abari, E. Hamed, and D. Katabi, "GHz-wide sensing and decoding using the sparse Fourier transform," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, May 2014, pp. 2256–2264.

[2] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[3] M. Mishali and Y. C. Eldar, "From theory to practice: Sub-Nyquist sampling of sparse wideband analog signals," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 2, pp. 375–391, Apr. 2010.

[4] J. Yoo, S. Becker, M. Loh, M. Monge, E. Candès, and A. Emami-Neyestanak, "A 100MHz–2GHz 12.5x sub-Nyquist rate receiver in 90 nm CMOS," in *Proc. IEEE Radio Freq. Integr. Circuits Symp.*, Jun. 2012, pp. 31–34.

[5] O. Abari *et al.*, "27.4 A 0.75-million-point fourier-transform chip for frequency-sparse signals," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2014, pp. 458–459.

[6] P. Indyk, M. Kapralov, and E. Price, "(Nearly) sample-optimal sparse fourier transform," in *Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2014, pp. 480–499,

[7] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Simple and practical algorithm for sparse Fourier transform," in *Proc. 23rd Annu. ACM-SIAM Symp. Discrete Algorithms*, Philadelphia, PA, USA: SIAM, Jan. 2012, pp. 1183–1194.

[8] A. Agarwal, H. Hassanieh, O. Abari, E. Hamed, D. Katabi, and Arvind, "High-throughput implementation of a million-point sparse Fourier transform," in *Proc. 24th Int. Conf. Field Program. Logic Appl. (FPL)*, Sep. 2014, pp. 1–6.

[9] A. López-Parrado and J. Velasco-Medina, "SoC-FPGA implementation of the sparse fast Fourier transform algorithm," in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2017, pp. 120–123.

[10] S. Pawar and K. Ramchandran, "R-FFAST: A robust sub-linear time algorithm for computing a sparse DFT," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 451–466, Jan. 2018.

[11] E. J. Candès, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Commun. Pure Appl. Math.*, vol. 59, no. 8, pp. 1207–1223, Aug. 2006.

[12] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[13] F. Ong, S. Pawar, and K. Ramchandran, "Fast sparse 2-D DFT computation using sparse-graph alias codes," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 4059–4063.

[14] S. Kay, "A fast and accurate single frequency estimator," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 12, pp. 1987–1990, Dec. 1989.

[15] X. Li, S. Pawar, and K. Ramchandran, "Sub-linear time compressed sensing using sparse-graph codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2015, pp. 1645–1649.

[16] J. Bachrach *et al.*, "Chisel: Constructing hardware in a Scala embedded language," in *Proc. DAC Design Automat. Conf.*, Jun. 2012, pp. 1212–1221.

[17] A. Wang, P. Rigge, A. Izraelevitz, C. Markley, J. Bachrach, and B. Nikolić, "ACED: A hardware library for generating DSP systems," in *Proc. 55th Annu. Design Autom. Conf.*, New York, NY, USA: ACM, Jun. 2018, Art. no. 61.

[18] 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)*, Nov. 2017, pp. 209–216.

[19] E. Chang *et al.*, "BAG2: A process-portable framework for generator-based AMS circuit design," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2018, pp. 1–8.

[20] A. Wang, J. Bachrach, and B. Nikolié, "A generator of memory-based, runtime-reconfigurable $2^N3^M5^K$FFT engines," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 1016–1020.

[21] 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. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html

[22] J. Han *et al.*, "A generated 7 GS/s 8 b time-interleaved SAR ADC with 38.2 dB SNDR at Nyquist in 16 nm CMOS FinFET," in *Proc. IEEE CICC*, Apr. 2019, pp. 1–4.

[23] D. Stepanovic and B. Nikolic, "A 2.8 GS/s 44.6 mW time-interleaved ADC achieving 50.9 dB SNDR and 3 dB effective resolution bandwidth of 1.5 GHz in 65 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 48, no. 4, pp. 971–982, Apr. 2013.

[24] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in *Proc. ACM/SIGDA 6th Int. Symp. Field Program. Gate Arrays*, Feb. 1998, pp. 191–200.

[25] P. Barrett, "Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor," in *Advances in Cryptology—CRYPTO*, A. M. Odlyzko, Ed. Berlin, Germany: Springer, 1987, pp. 311–323.

[26] S. Bailey *et al.*, "A 28nm FDSOI 8192-point digital ASIC spectrometer from a Chisel generator," in *Proc. IEEE CICC*, Apr. 2018, pp. 1–4.

[27] D. Bankman and B. Murmann, "An 8-bit, 16 input, 3.2 pJ/op switched-capacitor dot product circuit in 28-nm FDSOI CMOS," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Nov. 2016, pp. 21–24.

[28] 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. ESSCIRC–IEEE 44th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2018, pp. 322–325.

[29] B. Nikolic, E. Alon, and K. Asanovic, "Generating the next wave of custom silicon," in *Proc. IEEE 44th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2018, pp. 6–11.

[30] S. Pawar and K. Ramchandran, "FFAST: An algorithm for computing an exactly k-sparse DFT in $\mathcal{O}(k \log k)$ time," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 429–450, Jan. 2018.

[31] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[32] Y. Duan and E. Alon, "A 12.8 GS/s time-interleaved ADC with 25 GHz effective resolution bandwidth and 4.6 ENOB," *IEEE J. Solid-State Circuits*, vol. 49, no. 8, pp. 1725–1738, Aug. 2014.

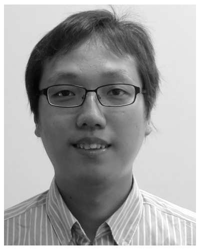
Angie Wang (S'11–M'18) received the B.S. degree in electrical engineering from the California Institute of Technology, Pasadena, CA, USA, in 2012 and the Ph.D. degree from the University of California at Berkeley, Berkeley, CA, USA, in 2018, where she was supported by the National Science Foundation's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

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, Seoul, South Korea. From 2017 to 2019, he was with the University of California at Berkeley, Berkeley, CA, USA, as a Post-Doctoral Researcher. He is currently a Senior SerDes Engineer with 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.



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.

Stevo Bailey (S'11–M'18) 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 from the University of California at Berkeley, Berkeley, CA, USA, in 2014 and 2018, respectively.

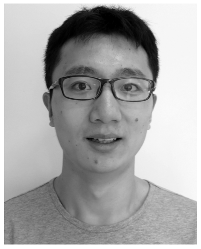
He was a Summer Research Intern with the Jet Propulsion Laboratory, Pasadena, CA, USA, in 2014, and Nvidia Corporation, Santa Clara, CA, USA, in 2015. He is currently working in the area of San Jose, CA, USA. His research interests include digital integrated circuit design methodologies, digital signal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

Orhan Ocal (S'14) received the M.Sc. degree in communication systems from the École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, in 2014, and the B.Sc. degree in electrical engineering from Boğaziçi University, Istanbul, Turkey, in 2012. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Sciences (EECS), University of California at Berkeley, Berkeley, CA, USA.

His current research interests include signal processing, information theory and coding, and machine learning.

Paul Rigge (S'09) received the B.S. degree in electrical engineering and in 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 with 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.



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.

Kannan Ramchandran (F'05) received the Ph.D. degree from Columbia University, New York, NY, USA, in 1993.

From 1993 to 1999, he was a Faculty Member with the University of Illinois at Urbana-Champaign, Champaign, IL, USA. Since 1999, he has been a Professor of electrical engineering and computer science with the University of California at Berkeley, Berkeley, CA, USA. He has published extensively in his field, and holds over a dozen patents. His current research interests are at the intersection of coding theory, statistical signal processing, and machine learning, with a particular emphasis on the use of sparse-graph coding theory for distributed machine learning, spectrum sensing, and imaging, and peer-to-peer content delivery for distributed video-on-demand systems.

Dr. Ramchandran was a recipient of the 2017 IEEE Kobayashi Computers and Communications Award for his pioneering contributions to the theory and practice of distributed storage coding and distributed compression. He has received several awards for his research and teaching including the IEEE Information Theory Society and Communication Society Joint Best Paper Award in 2012, the IEEE Communication Society Data Storage Best Paper Award in 2010, two Best Paper awards from the IEEE Signal Processing Society in 1993 and 1999, the Okawa Foundation Prize for outstanding research at Berkeley in 2001, and the Departmental Outstanding Teaching Award at Berkeley in 2009.

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

Dr. Alon was a recipient of 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, and 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* (Prentice-Hall, 2nd ed., 2003). His research activities 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 and the Anil K. Jain Prize for the Best Doctoral Dissertation in Electrical and Computer Engineering at University of California at Davis in 1999, and the City of Belgrade Award for the Best Diploma Thesis in 1992. For work with his students and colleagues, he was a recipient of the best paper awards at the IEEE International Solid-State Circuits Conference, Symposium on VLSI Circuits, IEEE International SOI Conference, European Solid-State Device Research Conference, European Solid-State Circuits Conference, S3S Conference, and the ACM/IEEE International Symposium of Low-Power Electronics. He was a Distinguished Lecturer of the IEEE Solid-State Circuits Society from 2014 to 2015.