A 0.32 mm^2 100 Mb/s 223 mW ASIC in 22FDX for Joint Jammer Mitigation, Channel Estimation, and SIMO Data Detection

Jonas Elmiger*, Fabian Stuber*, Oscar Castañeda, Gian Marti, and Christoph Studer
Department of Information Technology and Electrical Engineering, ETH Zurich, Switzerland

***Abstract*—We present the first single-input multiple-output (SIMO) receiver ASIC that jointly performs jammer mitigation, channel estimation, and data detection. The ASIC implements a recent algorithm called siMultaneous mitigAtion, Estimation, and Detection (MAED). MAED mitigates smart jammers via spatial filtering using a nonlinear optimization problem that unifies jammer estimation and nulling, channel estimation, and data detection to achieve state-of-the-art error-rate performance under jamming. The design supports eight receive antennas and enables mitigation of smart jammers as well as of barrage jammers. The ASIC is fabricated in 22 nm FD-SOI, has a core area of 0.32 mm^2, and achieves a throughput of 100 Mb/s at 223 mW, thus delivering 3× higher per-user throughput and 4.5× higher area efficiency than the state-of-the-art jammer-resilient detector.**

I. Introduction

In a jamming attack, a hostile interferer transmits interference signals to disrupt a wireless communication system. Safety-critical communication systems must be able to mitigate such jamming attacks [1]. The proliferation of multi-antenna receivers provides a promising avenue for jammer mitigation based on spatial filtering. However, spatial filtering requires accurate estimates of a jammer's spatial signature, which is difficult to acquire for smart jammers that try to avoid estimation. Nonlinear methods have recently been proposed for mitigating such smart jammers [2], [3]. Of these, only the SANDMAN algorithm from [2] has so far been implemented in hardware [4], which is necessary to support the throughput requirements of modern wireless systems. However, SANDMAN suffers from suboptimal error-rate performance as it separates channel estimation from jammer mitigation and data detection [5]. Moreover, its hardware efficiency is orders of magnitude below that of non-jammer-resilient data detectors [4].

Contributions: We present an application-specific integrated circuit (ASIC) implementation of the MAED (short for siMultaneous mitigAtion, Estimation, and Detection) algorithm from [3] for an 8×1 single-input multiple-output (SIMO) system. MAED is the first receiver ASIC that unifies jammer estimation and mitigation, channel estimation, and data detection. Our design mitigates smart jammers and—by leveraging joint channel estimation and data detection (JED)—outperforms the only other jammer-resilient receiver implementation [4] in terms of error-rate performance. In addition, our ASIC measurements demonstrate an improved area efficiency compared to [4]. Moreover, our design is the first silicon-proven ASIC for JED.[1]

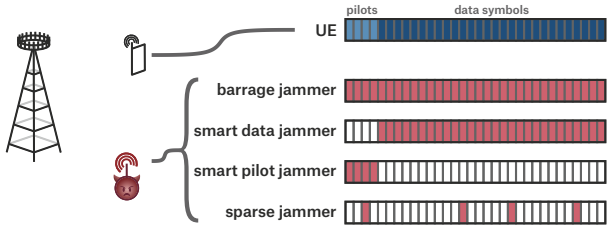


Fig. 1. Considered scenario of a SIMO uplink attacked by a (smart) jammer.

II. Prerequisites

A. System Model

We consider a SIMO system, which can model a single-antenna user equipment (UE) transmitting to a B-antenna receiver under a potentially smart single-antenna jammer (Fig. 1). We assume a flat-fading, block-fading channel with a coherence block of K channel uses. The input-output relation is

$$\mathbf{Y} = \mathbf{h}\mathbf{s}^T + \mathbf{j}\mathbf{w}^T + \mathbf{N}, \tag{1}$$

where $\mathbf{Y} \in \mathbb{C}^{B \times K}$ is the receive signal, $\mathbf{h} \in \mathbb{C}^B$ and $\mathbf{j} \in \mathbb{C}^B$ are the UE and jammer channels, $\mathbf{s} \in \mathbb{C}^K$ and $\mathbf{w} \in \mathbb{C}^K$ are the UE and jammer transmit signals, and $\mathbf{N} \overset{\text{i.i.d.}}{\sim} \mathcal{CN}(0, N_0)$ models noise with per-entry variance N_0. The UE signal $\mathbf{s} = [\mathbf{s}_T^T, \mathbf{s}_D^T]^T$ consists of T pilots $\mathbf{s}_T \in \mathbb{C}^T$ and $D = K - T$ data symbols $\mathbf{s}_D \in \mathcal{S}^D$ from the QPSK constellation $\mathcal{S}$. Every entry s_i of $\mathbf{s}$ satisfies $|s_i|^2 = 1$. We consider Rayleigh fading: $\mathbf{h} \overset{\text{i.i.d.}}{\sim} \mathcal{CN}(0, 1)$ and $\mathbf{j} \overset{\text{i.i.d.}}{\sim} \mathcal{CN}(0, 1)$. Moreover, we assume that the jammer can be one of the following types (cf. Fig. 1):

1) Barrage jammer: This jammer sends $\mathbf{w} \overset{\text{i.i.d.}}{\sim} \mathcal{CN}(0, \sigma^2)$.

2) Smart data jammer: This jammer sends $\mathbf{w} = [\mathbf{w}_T^T, \mathbf{w}_D^T]^T$ with $\mathbf{w}_T = \mathbf{0} \in \mathbb{C}^T$ and $\mathbf{w}_D \overset{\text{i.i.d.}}{\sim} \mathcal{CN}(0, \sigma^2) \in \mathbb{C}^D$.

3) Smart pilot jammer: This jammer sends $\mathbf{w} = [\mathbf{w}_T^T, \mathbf{w}_D^T]^T$ with $\mathbf{w}_T \overset{\text{i.i.d.}}{\sim} \mathcal{CN}(0, \sigma^2) \in \mathbb{C}^T$ and $\mathbf{w}_D = \mathbf{0} \in \mathbb{C}^D$.

4) Sparse jammer: This jammer sends $\mathbf{w} = \mathbf{c} \odot \mathbf{z}$, where $\odot$ is the Hadamard product, $\mathbf{c}$ is drawn from the distribution $\text{Unif}\big[\tilde{\mathbf{c}} \in \{0,1\}^K : \sum_k \tilde{c}_k = 4\big]$, and $\mathbf{z} \overset{\text{i.i.d.}}{\sim} \mathcal{CN}(0, \sigma^2)$.

*Equal contribution. Contact author: O. Castañeda (e-mail: caoscar@ethz.ch)
This work has received funding from the Swiss State Secretariat for Education, Research, and Innovation (SERI) under the SwissChips initiative. The authors thank GlobalFoundries for providing silicon fabrication through the 22FDX University Program. The authors also thank Darja Nonaca and Jérémy Guichemerre for their assistance with backend design and testing.

[1]To our knowledge, the only other ASICs for JED are [6], [7], which are not fabricated in silicon, require preprocessing, and do not mitigate jammers.

Algorithm 1 SIMO MAED [3]

1: **input**: $\mathbf{Y}, \tau_{(0)}, ..., \tau_{(t_{\max}-1)}, \mathbf{s}_T$
2: $\tilde{\mathbf{s}}_{(0)} = [\mathbf{s}_T^T, \mathbf{0}_{D\times 1}^T]^T$
3: **for** $t = 0$ **to** $t_{\max} - 1$ **do**
4: $\mathbf{E}_{(t)} = \mathbf{Y}\left(\mathbf{I}_K - \frac{\tilde{\mathbf{s}}_{(t)}^* \tilde{\mathbf{s}}_{(t)}^T}{\|\tilde{\mathbf{s}}_{(t)}\|_2^2}\right)$
4a: $\mathbf{x}_{(t)} = (1/\|\tilde{\mathbf{s}}_{(t)}\|_2^2)\mathbf{Y}\tilde{\mathbf{s}}_{(t)}^*$
4b: $\mathbf{E}_{(t)} = \mathbf{Y} - \mathbf{x}_{(t)}\tilde{\mathbf{s}}_{(t)}^T$
5: $\mathbf{u}_{(t)} = \text{PRNG}(\,)$
6: $\tilde{\mathbf{j}}_{(t)} = \mathbf{E}_{(t)}\mathbf{E}_{(t)}^H\mathbf{u}_{(t)}$
6a: $\mathbf{v}_{(t)} = \mathbf{E}_{(t)}^H\mathbf{u}_{(t)}$
6b: $\tilde{\mathbf{j}}_{(t)} = \mathbf{E}_{(t)}\mathbf{v}_{(t)}$
7: $\mathbf{P}_{(t)} = \mathbf{I}_B - \frac{\tilde{\mathbf{j}}_{(t)}\tilde{\mathbf{j}}_{(t)}^H}{\|\tilde{\mathbf{j}}_{(t)}\|_2^2}$
7a: $\mathbf{z}_{(t)} = \mathbf{x}_{(t)} - \tilde{\mathbf{j}}_{(t)}\left(\frac{\tilde{\mathbf{j}}_{(t)}^H\mathbf{x}_{(t)}}{\|\tilde{\mathbf{j}}_{(t)}\|_2^2}\right)$
8: $\nabla_{(t)} = -\frac{1}{\|\tilde{\mathbf{s}}_{(t)}\|_2^2}\tilde{\mathbf{s}}_{(t)}^T\mathbf{Y}^H\mathbf{P}_{(t)}\mathbf{E}_{(t)}$
8a: $-\tau_{(t)}\nabla_{(t)}^H = \mathbf{E}_{(t)}^H(\tau_{(t)}\mathbf{z}_{(t)})$
9: $\tilde{\mathbf{s}}_{(t+1)} = \text{prox}\left(\tilde{\mathbf{s}}_{(t)} - \tau_{(t)}\nabla_{(t)}^T; \mathbf{s}_T\right)$
10: **end for**
11: **output:** $\tilde{\mathbf{s}}_{(t_{\max})}$

The jammer power is characterized in terms of the receive jammer-to-signal ratio $\rho \triangleq \frac{\|\mathbf{j}\mathbf{w}^T\|_F^2}{\|\mathbf{h}\mathbf{s}^T\|_F^2}$, where $\|\mathbf{A}\|_F$ is the Frobenius norm of a matrix $\mathbf{A}$.

B. MitigAtion, Estimation, and Detection (MAED)

Smart jammers can be mitigated using the recently developed MAED algorithm [3]. MAED detects data in jammer-resilient fashion by approximately solving the optimization problem

$$\min_{\tilde{\mathbf{j}}\in\mathbb{C}^B, \tilde{\mathbf{h}}\in\mathbb{C}^B, \tilde{\mathbf{s}}_D\in\mathcal{S}^D} \left\|\left(\mathbf{I}_B - \tilde{\mathbf{j}}\tilde{\mathbf{j}}^\dagger\right)\left(\mathbf{Y} - \tilde{\mathbf{h}}[\mathbf{s}_T^T, \tilde{\mathbf{s}}_D^T]\right)\right\|_F^2, \qquad (2)$$

which unifies jammer estimation and mitigation (by solving for $\tilde{\mathbf{j}}$), channel estimation (by solving for $\tilde{\mathbf{h}}$), and data detection (by solving for $\tilde{\mathbf{s}}_D$). The problem in (2) can be transformed into a problem that only depends on $\tilde{\mathbf{j}}$ and $\tilde{\mathbf{s}}_D$. It can then be solved approximately by alternating between power iterations in $\tilde{\mathbf{j}}$ and projected gradient descent steps in $\tilde{\mathbf{s}}_D$. The resulting method is shown in Alg. 1. The proximal operator $\text{prox}(\tilde{\mathbf{s}}_{(t)}; \mathbf{s}_T)$ on line 9 replaces the first T entries of $\tilde{\mathbf{s}}_{(t)}$ with $\mathbf{s}_T$ and clips the remaining D entries to the convex hull of the constellation $\mathcal{S}$.

C. Algorithm Rearrangements for Hardware Implementation

To arrive at an efficient hardware implementation, we rearrange the MAED algorithm as outlined on the right side of Alg. 1. These rearrangements avoid matrix-matrix products and instead perform sequences of matrix-vector products which reduce the number of multiplications required by Alg. 1. Consider the computation of $\tilde{\mathbf{j}}_{(t)}$ on line 6 as an example: Computing first the matrix-matrix product $\mathbf{E}_{(t)}\mathbf{E}_{(t)}^H$ and then the matrix-vector product $(\mathbf{E}_{(t)}\mathbf{E}_{(t)}^H)\mathbf{u}_{(t)}$ requires $(K+1)B^2$ complex-valued multiplications, whereas computing first the matrix-vector product on line 6a and then the one on line 6b requires only $2KB$ complex-valued multiplications. Furthermore, while the original MAED algorithm [3] applies the Barzilai-Borwein method to determine the step sizes $\tau_{(t)}$, we empirically tune these step sizes to powers of two. Doing so saves the computations of the Barzilai-Borwein method and simplifies multiplication by $\tau_{(t)}$ into a simple arithmetic shift.

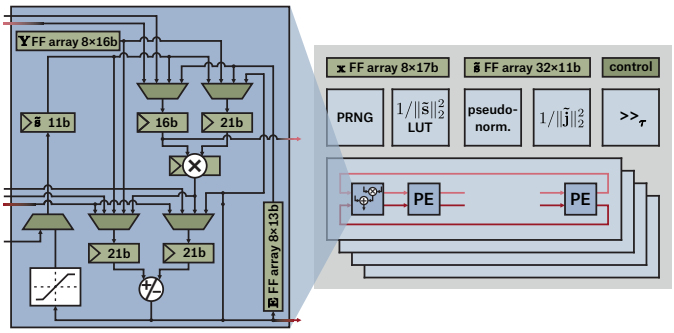
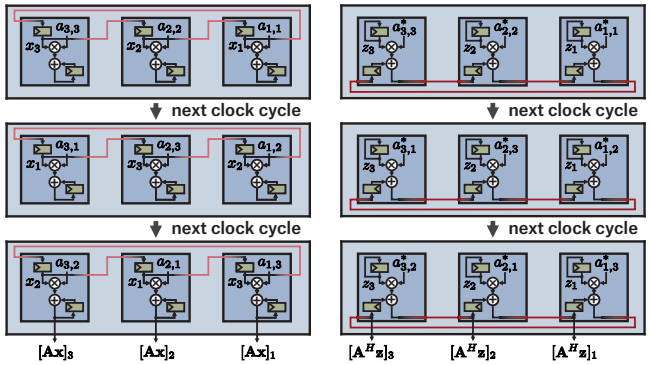


Fig. 2. Top view of the MAED architecture with zoom-in on the architecture of the processing elements (PEs). The bitwidths shown for registers and flip-flop (FF) arrays are per real and imaginary part of variables.



(a) $\mathbf{Ax}$ (b) $\mathbf{A}^H\mathbf{z}$

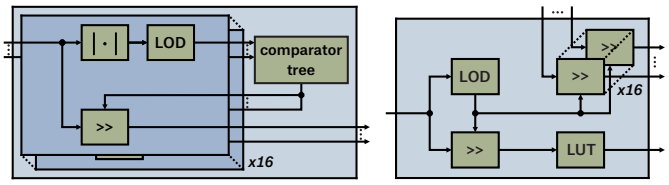
Fig. 3. Cannon's algorithm and its Hermitian variant illustrated for a 3×3 matrix-vector product: (a) To compute $\mathbf{Ax}$, the jth PE stores $\mathbf{a}_j$, the jth row of matrix $\mathbf{A}$. Each PE starts with x_k and multiplies it by $a_{j,k}$. The PEs then circularly exchange the entries of $\mathbf{x}$ and accumulate the partial products until completing $\mathbf{Ax}$. (b) In the Hermitian variant, the PEs compute $\mathbf{A}^H\mathbf{z}$ by circularly exchanging the accumulated partial products while keeping the entries of $\mathbf{z}$ fixed. The entries of $\mathbf{a}_j$ are read in the same order as for $\mathbf{Ax}$.

III. VLSI Architecture

Our ASIC is designed to execute the rearranged Alg. 1 for $B = 8$, $K = 32$ and $T = 4$. In what follows, we drop the iteration subscript "(t)" from the variables to simplify notation. Fig. 2 shows the top-level architecture, which consists of 32 processing elements (PEs) grouped in four slices of 8 PEs each, as well as auxiliary modules for pseudorandom number generation (PRNG), inversion, multiplication by the step size τ, and flip-flop (FF) arrays to store $\mathbf{x}$ and the output $\tilde{\mathbf{s}}$. Fig. 2 also shows a PE's internals, which includes complex-valued multiply-accumulate circuitry, local FF arrays, and a clipping unit for executing the prox$(\cdot)$ operator on line 9 of Alg. 1.

A. Operation

The 8×32 matrix-vector product $\mathbf{Y}\tilde{\mathbf{s}}^*$ on line 4a of Alg. 1 can be rewritten as $[\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3, \mathbf{Y}_4][\tilde{\mathbf{s}}_1^T, \tilde{\mathbf{s}}_2^T, \tilde{\mathbf{s}}_3^T, \tilde{\mathbf{s}}_4^T]^H$, where $\mathbf{Y}_i \in \mathbb{C}^{8\times 8}$ and $\tilde{\mathbf{s}}_i \in \mathbb{C}^8$, and is thus computed as the sum $\sum_i \mathbf{Y}_i\tilde{\mathbf{s}}_i^*$ of four 8×8 matrix-vector products. The jth PE of the ith slice stores the jth row of $\mathbf{Y}_i$, which enables the application of Cannon's algorithm [8] (cf. Fig. 3(a)) to compute the products $\mathbf{Y}_i\tilde{\mathbf{s}}_i^*$ in parallel across the four PE slices. The PE slices exchange their $\mathbf{Y}_i\tilde{\mathbf{s}}_i^*$ to compute $\mathbf{Y}\tilde{\mathbf{s}}^*$, which is stored across the PEs of the first slice. The 8×32 matrix-vector



(a) Pseudonormalization module (b) $\|\tilde{\mathbf{j}}\|_2^2$-inversion module

Fig. 4. Modules of the MAED architecture.

product $\mathbf{Ev}$ on line 6b is computed by following the same procedure, which takes 13 clock cycles. The matrix-vector products involving $\mathbf{E}^H$ on lines 6a and 8a are each computed after 10 clock cycles by using a variant of Cannon's algorithm (cf. Fig. 3(b)), which avoids rewriting memory to transpose $\mathbf{E}$.

To complete line 4a, we need the reciprocal value of the 32-dimensional inner product $\|\tilde{\mathbf{s}}\|_2^2 = \tilde{\mathbf{s}}^H\tilde{\mathbf{s}}$. The multipliers of the 32 PEs compute the partial products $\tilde{s}_i^*\tilde{s}_i$, which are then summed by reconfiguring the PEs' adders into an adder tree with five pipelined stages. The resulting $\|\tilde{\mathbf{s}}\|_2^2$ is confined to the interval $[4, 32]$, and hence its inverse is simply computed via a look-up table (LUT). The PEs of the first slice then multiply $1/\|\tilde{\mathbf{s}}\|_2^2$ by $\mathbf{Y}\tilde{\mathbf{s}}^*$ to obtain $\mathbf{x}$, which is stored in a FF array that is accessible to all PE slices.

Since the jth PE of the ith slice stores the jth row of $\mathbf{Y}_i$, it can compute line 4b by scaling $\tilde{\mathbf{s}}_i$ by x_j. This is achieved in 12 clock cycles by circularly exchanging the entries of $\tilde{\mathbf{s}}_i$ across the 8 PEs of the ith PE slice, just as it is done for Cannon's algorithm (cf. Fig. 3(a)). The next step, given by line 5, is to use the PRNG module to generate $\mathbf{u}$, which is used to start the power iteration that follows on lines 6a and 6b. The PRNG was implemented using xorshift [9] with a 64-bit state.

The vector $\tilde{\mathbf{j}}$ of line 6b corresponds to an estimate of the jammer channel $\mathbf{j}$, up to an unknown scale factor proportional to the jammer power. Since the jammer might be jamming at very high power (e.g., the received jammer power could exceed that of the UE by $\rho = 30\,\text{dB}$), a large bitwidth is required to represent the entries of $\tilde{\mathbf{j}}$, leading to an even larger bitwidth to represent $\|\tilde{\mathbf{j}}\|_2^2$. In fact, the jammer-resilient detector from [4] uses dedicated PEs with extended bitwidths to compute an analogous quantity. We avoid this significant hardware overhead by noting that the subsequent algorithm operation (line 7a) is invariant to the scale of $\tilde{\mathbf{j}}$. This allows us to first rescale $\tilde{\mathbf{j}}$ with the pseudonormalization module (cf. Sec. III-B) to focus only on the bits that are relevant for the computation of $\|\tilde{\mathbf{j}}\|_2^2$.

The rescaled $\tilde{\mathbf{j}}$ is then used to compute the 8-dimensional inner products $\tilde{\mathbf{j}}^H\mathbf{x}$ and $\|\tilde{\mathbf{j}}\|_2^2 = \tilde{\mathbf{j}}^H\tilde{\mathbf{j}}$ (line 7a). These inner products are each computed in 5 clock cycles using one PE slice. The $\|\tilde{\mathbf{j}}\|_2^2$-inversion module (described in Sec. III-C) then computes $1/\|\tilde{\mathbf{j}}\|_2^2$, which is multiplied by $\tilde{\mathbf{j}}^H\mathbf{x}$. Using these quantities, the first PE slice computes $\mathbf{z}$ (line 7a) and scales its entries by the step size τ through the $>>_\tau$ arithmetic shifters. The resulting $\tau\mathbf{z}$ is broadcasted to all PE slices in order to compute $\mathbf{E}^H(\tau\mathbf{z})$ (line 8a) using the Hermitian variant of Cannon's algorithm (cf. Fig. 3(b)). Finally, each PE performs the gradient descent step and proximal operation for one entry of $\tilde{\mathbf{s}}$ to obtain the new iterate (line 9). This completes one iteration of the MAED algorithm after 83 clock cycles.

B. Pseudonormalization Module

Naïvely, the potential dynamic range of $\tilde{\mathbf{j}}$ would require large bitwidths to faithfully represent $\tilde{\mathbf{j}}$ on line 7a (since it is not known *a priori* where the relevant bits of $\tilde{\mathbf{j}}$ are located). However, since line 7a is invariant to the scale of $\tilde{\mathbf{j}}$, the pseudonormalization solves this issue by rescaling $\tilde{\mathbf{j}}$ such that $\|\tilde{\mathbf{j}}\|_2^2 \in [1, 32)$, which enables subsequent operations to discard fractional bits not needed for the required computational precision. Our module finds the largest (in magnitude) entry $j_{\max}$ across the real and imaginary parts of $\tilde{\mathbf{j}}$ using leading one detectors (LODs), and divides all entries by $2^{\lfloor \log_2(j_{\max}) \rfloor}$.

C. $\|\tilde{\mathbf{j}}\|_2^2$-Inversion Module

The $\|\tilde{\mathbf{j}}\|_2^2$-inversion module (cf. Fig. 4(b)) starts by scaling $\|\tilde{\mathbf{j}}\|_2^2$ to be within the interval $[1, 2)$, to then compute its scaled inverse with a LUT. To obtain the desired $1/\|\tilde{\mathbf{j}}\|_2^2$, the result from the LUT should be scaled back, but this would require an increased bitwidth at the input of the multiplier computing $(\tilde{\mathbf{j}}^H\mathbf{x})/\|\tilde{\mathbf{j}}\|_2^2$ to not lose the LUT's precision. Instead, we scale $\mathbf{x}$ when computing $\tilde{\mathbf{j}}^H\mathbf{x}$, so that both $\tilde{\mathbf{j}}^H\mathbf{x}$ and $\|\tilde{\mathbf{j}}\|_2^2$ are scaled by the same factor. While this entails some loss in the precision of the entries of $\mathbf{x}$ for computing $\tilde{\mathbf{j}}^H\mathbf{x}$, our results demonstrate that the performance is not affected (cf. Sec. IV-A).

IV. ASIC Implementation Results

A. Bit Error-Rate (BER) Performance

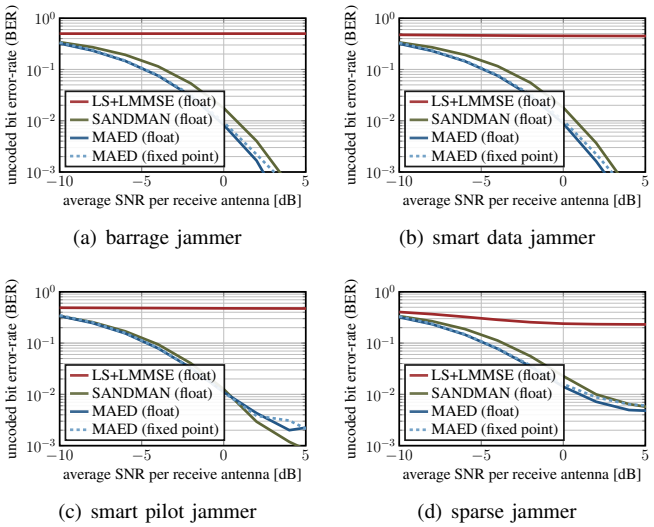
Fig. 5 shows the uncoded bit error-rate (BER) of different SIMO receivers that detect QPSK signals while under attack from one of four different jammers (cf. Sec. II-A). We vary the signal-to-noise ratio (SNR) per receive antenna, which is defined as $\text{SNR} = \|\mathbf{h}\|_2^2/(BN_0)$, whereas all jammers interfere with a jammer-to-signal ratio of $\rho = 30\,\text{dB}$. The compared receivers are a non-mitigating receiver with least squares (LS) channel estimation and linear minimum mean squared error (LMMSE) data detection as in [10], an adapted version of the jammer-mitigating SANDMAN method [2], [4] for SIMO,[2] and the MAED algorithm implemented in this work.

Fig. 5 shows that the BER of the non-mitigating LS+LMMSE receiver does not fall below 10% for any of the jammer types. In contrast, both SANDMAN and MAED succeed in mitigating all jammers and reach BERs below 1% at high SNR. However, MAED outperforms SANDMAN thanks to the gains afforded by JED. Finally, the BER curves of the fixed-point implementation of MAED closely follow those of the double-precision floating-point reference, which demonstrates that the selected numerical representation scheme of the silicon implementation is sufficiently precise for practical purposes.

B. ASIC Measurements and Comparison

Fig. 6 shows the micrograph of the $5\,\text{mm}^2$ 22 nm FD-SOI ASIC that contains our design along with other, unrelated designs. Our design occupies a core area of $0.32\,\text{mm}^2$. At $0.8\,\text{V}$ nominal core supply, zero body biasing, and $300\,\text{K}$ room

[2]SANDMAN resembles MAED in that it also mitigates the jammer jointly with detecting the receive data, and thus is able to mitigate smart jammers as well as barrage jammers. However, SANDMAN estimates the UE channel separately and thus fails to harvest the gains associated with JED [5], [7].



(a) barrage jammer (b) smart data jammer

(c) smart pilot jammer (d) sparse jammer

Fig. 5. Uncoded bit error-rate (BER) of different SIMO receivers as a function of the SNR for different jamming scenarios (cf. Sec. II-A). The receive jammer-to-signal ratio of all jammers is equal to $\rho = 30\,\text{dB}$.

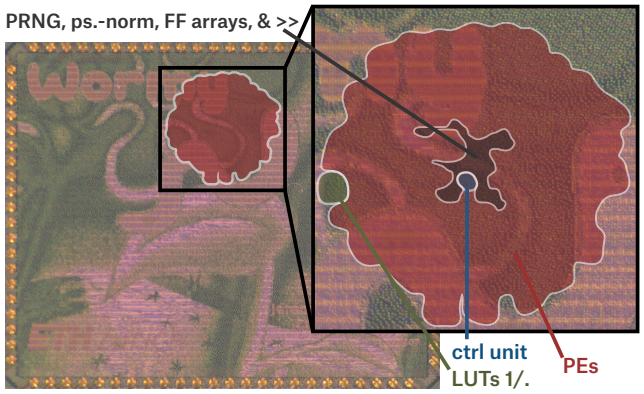


Fig. 6. Micrograph of the $2.5\,\text{mm} \times 2\,\text{mm}$ ASIC with the area containing the MAED design zoomed-in; the rest of the ASIC contains unrelated designs.

temperature, our design achieves a maximum clock frequency of $1.49\,\text{GHz}$, which, for $t_{\max} = 10$ iterations, corresponds to $100\,\text{Mb/s}$ throughput, $314\,\text{Mb/s/mm}^2$ area efficiency, and $2.2\,\text{nJ/b}$ energy when facing a $\rho = 30\,\text{dB}$ jammer. Fig. 7 shows the maximum clock frequency and energy as a function of the core supply voltage, with and without forward body biasing.

MAED is the first jammer-resilient JED ASIC, so a direct comparison to other detectors is not entirely fair. Table I shows such a comparison nonetheless. Compared to the only other jammer-mitigating detector in the literature [4], MAED achieves $3\times$ higher per-user throughput and $4.5\times$ higher area efficiency at the same energy efficiency. Compared to the most efficient SIMO-JED detector in the literature [7], MAED exhibits orders of magnitude lower efficiency, which is the cost of mitigating smart jammers; see also the efficiency difference between the (non-)jammer-resilient MIMO detectors [10] and [4].

V. Conclusions

We have presented the *first* ASIC for joint data detection, jammer mitigation, and channel estimation in SIMO, which is also the first silicon-proven ASIC for JED. Our design

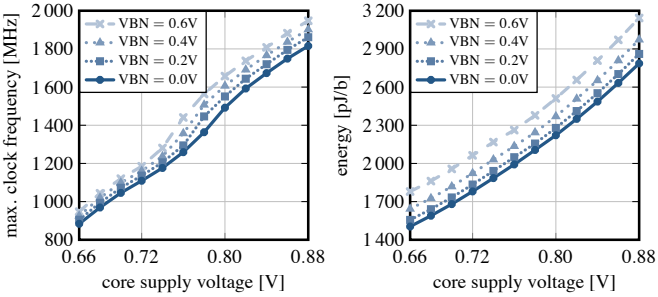


Fig. 7. Measured maximum clock frequency and energy per bit as the core supply and NMOS forward body biasing (VBN) vary; VBP is held at 0 V.

TABLE I
Measurement Results and ASIC Comparison

	This work	Bucheli [4]	Castañeda [7]	Prabhu [10]
Max. receiver antennas	8	32	128	128
Max. UEs	1	8	1	8
Modulation [QAM]	4	16	4	256
Algorithm	MAED	SANDMAN	PrOX	LMMSE
Jammer mitigation	**yes**	**yes**	**no**	**no**
Joint est. & det. (JED)	**yes**	**no**	**yes**[a]	**no**
Technology [nm]	22	22	40	28
Core supply [V]	0.8	0.78	1.1	0.9
Core area [mm^2]	0.32	3.78	0.30	–
Max. frequency [MHz]	1 492	320	695	300
Throughput [Mb/s]	100	267	412	300
Power [mW]	223	583	58	18
Area eff.[b] [Mb/s/mm^2]	314	70	8 150	–
Energy[b] [pJ/b]	2 218	2 300	41	37

[a]Not validated in silicon, [b]technology normalized to 22 nm at 0.8 V core supply.

outperforms the only jammer-resilient detector in the open literature in terms of error-rate performance, per-user throughput, and area efficiency, without degrading energy efficiency. Jammer resilience comes at a steep premium, but without it, communication in hostile environments is doomed to fail.

References

[1] H. Pirayesh and H. Zeng, "Jamming attacks and anti-jamming strategies in wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 2, pp. 767–809, 2022.

[2] G. Marti and C. Studer, "Joint jammer mitigation and data detection for smart, distributed, and multi-antenna jammers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2023, pp. 1364–1369.

[3] G. Marti, T. Kölle, and C. Studer, "Mitigating smart jammers in multi-user MIMO," *IEEE Trans. Signal Process.*, vol. 71, pp. 756–771, 2023.

[4] F. Bucheli, O. Castañeda, G. Marti, and C. Studer, "A jammer-mitigating 267 Mb/s 3.78 mm^2 583 mW 32×8 multi-user MIMO receiver in 22FDX," in *IEEE Int. Symp. VLSI Technol. Circuits*, Jun. 2024.

[5] H. Vikalo, B. Hassibi, and P. Stoica, "Efficient joint maximum-likelihood channel estimation and signal detection," *IEEE Trans. Wireless Commun.*, vol. 5, no. 7, pp. 1838–1845, Jul. 2006.

[6] O. Castañeda, T. Goldstein, and C. Studer, "Data detection in large multi-antenna wireless systems via approximate semidefinite relaxation," *IEEE Trans. Circuits Syst. I*, vol. 63, no. 12, pp. 2334–2346, Dec. 2016.

[7] O. Castañeda, T. Goldstein, and C. Studer, "VLSI designs for joint channel estimation and data detection in large SIMO wireless systems," *IEEE Trans. Circuits Syst. I*, vol. 65, no. 3, pp. 1120–1132, Mar. 2018.

[8] L. Cannon, "A cellular computer to implement the Kalman filter algorithm," Ph.D. dissertation, Montana State University, 1969.

[9] G. Marsaglia, "Xorshift RNGs," *J. Stat. Software*, vol. 8, Jul. 2003.

[10] H. Prabhu, J. N. Rodrigues, L. Liu, and O. Edfors, "A 60pJ/b 300Mb/s 128×8 massive MIMO precoder-detector in 28nm FD-SOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Feb. 2017, pp. 60–61.