A Jammer-Resilient 2.87 mm^2 1.28 MS/s 310 mW Multi-Antenna Synchronization ASIC in 65 nm

Flurin Arquint, Oscar Castañeda, Gian Marti, and Christoph Studer

Department of Information Technology and Electrical Engineering, ETH Zurich, Switzerland

***Abstract*—We present the first ASIC implementation of jammer-resilient multi-antenna time synchronization. The ASIC implements a recent algorithm that mitigates jamming attacks on synchronization signals using multi-antenna processing. Our design supports synchronization between a single-antenna transmitter and a 16-antenna receiver while mitigating smart jammers with up to two transmit antennas. The fabricated 65 nm ASIC has a core area of 2.87 mm^2, consumes a power of 310 mW, and supports a sampling rate of 1.28 mega-samples per second (MS/s).**

I. Introduction

Our aspirations for a wireless future make jammer-resilient communications an imperative. Much attention has been paid to data transmission under jamming [1]–[3], including the design of jammer-resilient data detectors as integrated circuits [4]. However, to deploy these methods, the transmitter and receiver must be synchronized in time [5]. This synchronization problem has received only little attention [6], [7] and, to our knowledge, no corresponding integrated circuit has been developed so far.

Contributions: We present an application-specific integrated circuit (ASIC) implementation of the recent JASS (short for Jammer-Aware SynchroniSation) algorithm [7] for time synchronization between a single-antenna transmitter and a 16-antenna receiver under jamming. The jammer is mitigated by fitting an adaptive spatial filter to a time-windowed sequence of the receive signal. Our design can mitigate jammers with up to two antennas and, by using a secret synchronization sequence (consisting of 16 BPSK symbols), it is also able to mitigate smart jammers (i.e., jammers that rely on more sophisticated strategies than pure noise-like barrage jamming). To our knowledge, our design is the first ASIC implementation of a jammer-resilient synchronization algorithm of *any* kind.

II. Prerequisites

A. System Model

We consider a multi-antenna receiver with a single-antenna transmitter (as in, e.g., a single-user uplink) under attack from a jammer that can potentially have multiple antennas. We model the receive signal at sample index $k = 0, 1, 2, \dots$ as follows:

$$\mathbf{y}[k] = \mathbf{h}s[k] + \mathbf{J}\mathbf{w}[k] + \mathbf{n}[k]. \tag{1}$$

This work has received funding from the Swiss State Secretariat for Education, Research, and Innovation (SERI) under the SwissChips initiative, and has also been supported by the European Commission within the context of the project 6G-REFERENCE (6G Hardware Enablers for Cell Free Coherent Communications and Sensing), funded under EU Horizon Europe Grant Agreement 101139155. Contact author: O. Castañeda (e-mail: caoscar@ethz.ch)

Here, $\mathbf{y}[k] \in \mathbb{C}^B$ is the receive vector at the B-antenna receiver; $s[k]$ is the legitimate transmit signal, which corresponds to

$$s[k] = \begin{cases} 0 & : k < L \\ \breve{s}_{k+1-L} & : L \le k < L+K \\ \text{undefined} & : L+K \le k, \end{cases} \tag{2}$$

where $\breve{\mathbf{s}} = [\breve{s}_1, \dots, \breve{s}_K]^T \in \{\pm 1\}^K$ is the length-K synchronization sequence and $L \ge 0$ represents the time at which the legitimate transmitter sends the start of the synchronization sequence; $\mathbf{h} \in \mathbb{C}^B$ is the channel between the legitimate transmitter and the receiver; $\mathbf{J} \in \mathbb{C}^{B \times I}$ is the channel between the I-antenna jammer and the receiver; $\mathbf{w}[k] \in \mathbb{C}^I$ is the jammer transmit vector; and $\mathbf{n}[k] \sim \mathcal{CN}(\mathbf{0}, N_0\mathbf{I}_B)$ is additive white Gaussian thermal noise with variance N_0 per entry.

We assume that the synchronization sequence $\breve{\mathbf{s}}$ is unknown a priori to the jammer (i.e., $\mathbf{w}[k]$ can only depend on $\breve{s}_{k'}$ if $k - L \ge k'$), while L is unknown a priori to the receiver and has to be estimated based on the receive vectors. Specifically, the receiver has to solve the following problem: Based on a running receive sequence $\mathbf{y}[0], \dots, \mathbf{y}[\ell + K - 1]$, the receiver must decide—for each ℓ—if ℓ is equal to L or not. A *false alarm* occurs when the receiver erroneously decides that $\ell = L$. A *miss* occurs when the receiver erroneously decides that $\ell \ne L$. After the $(L+K-1)$th sample has been processed, the receiver has either successfully found L or made an error (false alarm or miss), so there is no need to define $s[k]$ for $k \ge L + K$.

B. Jammer-Resilient Synchronization

Our design is based on the JASS algorithm proposed recently in [7], which approximately solves the optimization problem

$$\arg\min_{\ell \ge 0} \ell \quad \text{such that} \quad \max\nolimits_{\tilde{\mathbf{P}} \in \mathcal{P}_{I_{\max}}} \frac{\|\tilde{\mathbf{P}}\mathbf{Y}_\ell \breve{\mathbf{s}}^*\|_2^2}{\|\tilde{\mathbf{P}}\mathbf{Y}_\ell\|_F^2} \ge \tau, \tag{3}$$

where $\mathbf{Y}_\ell = [\mathbf{y}[\ell], \dots, \mathbf{y}[\ell + K - 1]]$ is the windowed receive signal, $\tau \ge 0$ is a detection threshold tunable for an optimal tradeoff between false alarms and misses, $(\cdot)^*$ is the complex conjugate, and $\mathcal{P}_{I_{\max}} = \{\mathbf{I}_B - \mathbf{A}\mathbf{A}^\dagger : \mathbf{A} \in \mathbb{C}^{B \times I_{\max}}\}$ is the set of orthogonal projections onto the $(B - I_{\max})$-dimensional subspaces of $\mathbb{C}^B$, with $I_{\max}$ being the maximum number of jammer antennas that can be mitigated ($I \le I_{\max} < B$); $\mathbf{I}_B$, the $B \times B$ identity matrix; and $(\cdot)^\dagger$, the Moore-Penrose inverse.

To approximately solve (3), JASS performs the following operations for $\ell = 0, 1, \dots, \ell_{\max}$: First, it estimates the interference subspace by computing the $I_{\max}$ principal eigenvectors of $\mathbf{\Lambda} = \|\breve{\mathbf{s}}\|_2^2 \mathbf{Y}_\ell \mathbf{Y}_\ell^H - \mathbf{Y}_\ell \breve{\mathbf{s}}^* \breve{\mathbf{s}}^T \mathbf{Y}_\ell^H$. These principal

Algorithm 1 Jammer-Aware Synchronisation (JASS)

1: $\mathbf{\Phi} = \mathbf{Y}_0\mathbf{Y}_0^H$
2: **for** $\ell = 0, 1, 2, \ldots, \ell_{\max}$ **do**
3: $\quad \mathbf{c}_\ell = \mathbf{Y}_\ell \breve{\mathbf{s}}^*$
4: $\quad \mathbf{\Lambda} = \|\breve{\mathbf{s}}\|_2^2\mathbf{\Phi} - \mathbf{c}_\ell\mathbf{c}_\ell^H$
5: $\quad$ **for** $i = 1, \ldots, I_{\max}(= 2)$ **do**
6: $\quad\quad \mathbf{a}_i \leftarrow$ PRNG
7: $\quad\quad$ **for** $t = 1, \ldots, t_{\max}(= 2)$ **do**
8: $\quad\quad\quad \mathbf{a}_i' = \mathbf{\Lambda}\mathbf{a}_i$
9: $\quad\quad\quad \mathbf{a}_i = \mathbf{a}_i'/\|\mathbf{a}_i'\|_2$
10: $\quad\quad \mathbf{\Lambda} \leftarrow \mathbf{\Lambda} - \mathbf{a}_i'\mathbf{a}_i^H$
11: $\quad \mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2];\ \tilde{b} = \mathbf{a}_1^H\mathbf{a}_2;\ \tilde{\mathbf{B}} = \begin{bmatrix} 1 & -\tilde{b} \\ -\tilde{b}^* & 1 \end{bmatrix}$
12: $\quad \mathbf{v} = \mathbf{A}^H\mathbf{c}_\ell;\ \mathbf{W} = \mathbf{A}^H\mathbf{\Phi}$
13: $\quad N = (1 - |\tilde{b}|^2)\|\mathbf{c}_\ell\|_2^2 - \mathbf{v}^H\tilde{\mathbf{B}}\mathbf{v}$
14: $\quad D = (1 - |\tilde{b}|^2)\,\mathrm{tr}(\mathbf{\Phi}) - \mathrm{tr}(\tilde{\mathbf{B}}\mathbf{W}\mathbf{A})$
15: $\quad$ **if** $N - D\tau \geq 0$ **then**
16: $\quad\quad$ **return** ℓ
17: $\quad \mathbf{\Phi} \leftarrow \mathbf{\Phi} - \mathbf{y}[\ell]\mathbf{y}[\ell]^H + \mathbf{y}[\ell + K]\mathbf{y}[\ell + K]^H$

eigenvectors are collected into a matrix $\mathbf{A}$, so that the estimated interference subspace can be removed through the projection matrix $\tilde{\mathbf{P}} = \mathbf{I}_B - \mathbf{A}\mathbf{A}^\dagger \in \mathcal{P}_{I_{\max}}$. Then, JASS computes the score $\|\tilde{\mathbf{P}}\mathbf{Y}_\ell\breve{\mathbf{s}}^*\|_2^2/\|\tilde{\mathbf{P}}\mathbf{Y}_\ell\|_F^2$. If the score reaches the threshold τ, then JASS terminates and declares ℓ to be the delay index L at which the start of the synchronization sequence $\breve{\mathbf{s}}$ was received. If the threshold τ is not reached after evaluating $\ell_{\max}$ candidate indexes, then JASS terminates and declares a miss. For a proper motivation of JASS, its algorithmic details, and corresponding success guarantees, see [7].

To enable an efficient hardware implementation, the JASS algorithm from [7] was reorganized into Alg. 1. Instead of explicitly computing $\mathbf{\Phi} = \mathbf{Y}_\ell\mathbf{Y}_\ell^H$ for each ℓ, $\mathbf{\Phi}$ is iteratively updated (line 17). Each of the $I_{\max}$ principal eigenvectors of $\mathbf{\Lambda}$ is computed using $t_{\max}$ iterations of the power method (lines 5 to 10). Our ASIC is dimensioned to mitigate jammers with up to $I_{\max} = 2$ antennas. This parameter choice simplifies the computation of the matrix inverse $(\mathbf{A}^H\mathbf{A})^{-1} = (1 - |\tilde{b}|^2)^{-1}\tilde{\mathbf{B}}$ (line 11), which is needed to compute $\mathbf{A}^\dagger = (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H$. To avoid the division required to compute the score, we instead take the numerator N (line 13) and denominator D (line 14) of the score and check whether $N - D\tau \geq 0$ (line 15) instead of $N/D \geq \tau$. Note that in these computations, the projection matrix $\tilde{\mathbf{P}} = \mathbf{I}_B - \mathbf{A}\mathbf{A}^\dagger$ is never computed explicitly.

III. VLSI ARCHITECTURE

Our ASIC is designed to execute Alg. 1 for a receiver with $B = 16$ antennas, using a programmable synchronization sequence consisting of $K = 16$ BPSK symbols under the interference of a jammer with at most $I_{\max} = 2$ antennas. The detection threshold τ, as well as the maximum number $\ell_{\max}$ of candidate indexes to be evaluated, is programmable. The number of iterations for the power method is fixed to $t_{\max} = 2$, since it delivers practically the same performance as an exact eigenvalue decomposition, as used in the original algorithm [7].

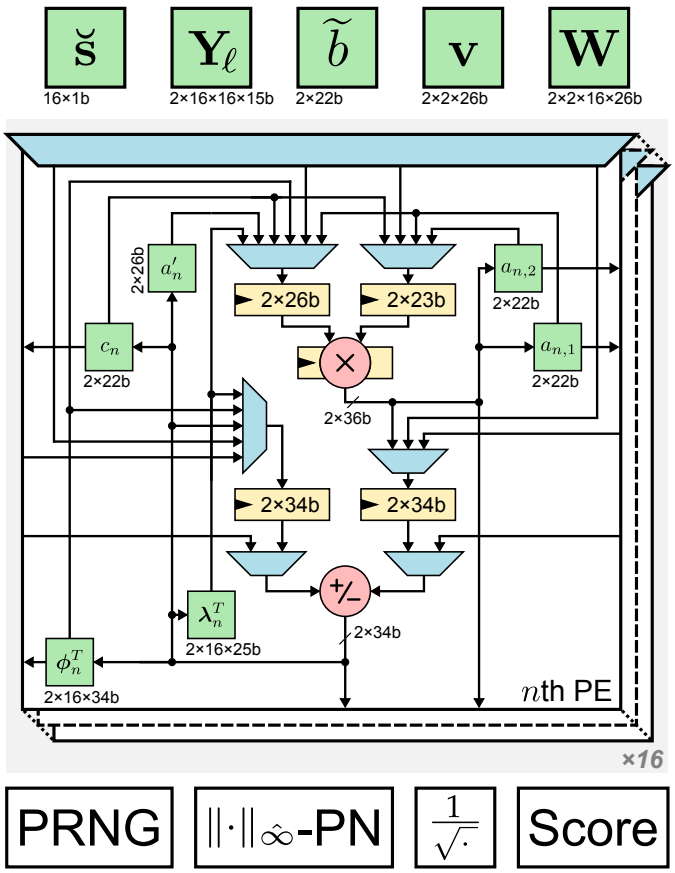
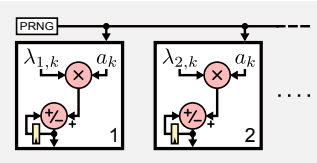


Fig. 1. Overview of the JASS architecture, consisting of 16 reconfigurable processing elements (PEs), a pseudorandom number generator (PRNG), a $\|\cdot\|_\infty$–pseudonormalization (PN) module, an inverse square root module, and the score module. Blocks in green correspond to flip-flops (FFs) and FF arrays. The $\boldsymbol{\lambda}_n^T$ and $\boldsymbol{\phi}_n^T$ FF arrays store the nth row of $\mathbf{\Lambda}$ and $\mathbf{\Phi}$, respectively.

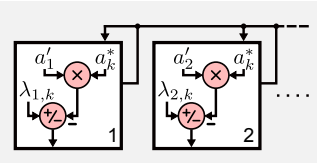
Fig. 1 provides an overview of the top-level hardware architecture, which consists primarily of 16 reconfigurable processing elements (PEs). Each PE includes a pipelined complex-valued multiplier, a complex-valued adder, and flip-flop (FF) arrays for local storage. The number of PEs was chosen to match the largest matrix dimension (in our case, $B = K = 16$) in order to reduce latency. Besides the PEs, the architecture contains a module for complex-valued pseudo-random number generation (PRNG; cf. Sec. III-B), a $\|\cdot\|_\infty$–pseudonormalization (PN) module (cf. Sec. III-C), an inverse square root module (cf. Sec. III-D), and a score module (cf. Sec. III-A). Finally, the architecture integrates FF arrays to store $\breve{\mathbf{s}}$, $\mathbf{Y}_\ell$, $\tilde{b}$, $\mathbf{v}$, and $\mathbf{W}$. Every PE can access the contents of these FF arrays, as well as the outputs from other PEs, through an interconnect network (represented by the large multiplexers at the top of each PE in Fig. 1).

A. Operation

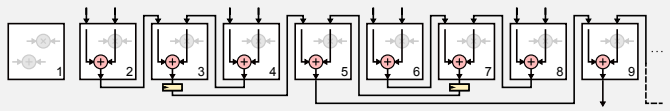
Most operations of Alg. 1 boil down to matrix-vector products (lines 3, 8, 12, 13, and 14) and rank-one matrix updates of the form $\mathbf{\Lambda} - \mathbf{a}'\mathbf{a}^H$ (lines 4, 10, and 17). For the matrix-vector product $\mathbf{\Lambda}\mathbf{a}$ on line 8 (which is executed four times per delay index ℓ), the PEs are configured as depicted in Fig. 2(a): In each clock cycle k, a new column $\boldsymbol{\lambda}_k$ of $\mathbf{\Lambda}$



(a) Accumulator configuration of the PEs for the operation $\mathbf{\Lambda a}$.



(b) PEs configured to multipy-and-subtract for the operation $\mathbf{\Lambda} - \mathbf{a}'\mathbf{a}^H$.



(c) Adder tree configuration of the PEs with extra pipeline registers.

Fig. 2. Illustration of three different PE configurations for the execution of the operations of the JASS algorithm.

is scaled by a_k and accumulated to the previous result, until $\mathbf{\Lambda a} = \sum_{k=1}^{16} \boldsymbol{\lambda}_k a_k$ is completed. Due to pipelining registers, computing $\mathbf{\Lambda a}$ takes 19 clock cycles. A rank-one matrix update $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} - \mathbf{a}'\mathbf{a}^H$ also takes 19 clock cycles and is performed by calculating, in clock cycle k, the column $\boldsymbol{\lambda}_k \leftarrow \boldsymbol{\lambda}_k - \mathbf{a}' a_k^*$ by using the 16 PEs in parallel as visualized in Fig. 2(b).

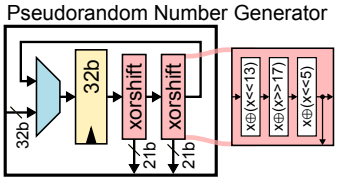
The computation of $\mathbf{c}_\ell$ on line 3 follows the configuration in Fig. 2(a), but the signs of the columns are simply adjusted instead of using the multipliers, since $\check{\mathbf{s}}$ is a BPSK sequence. For the same reason, the matrix rescaling $\|\check{\mathbf{s}}\|_2^2 \mathbf{\Phi}$ on line 4 amounts to a left shift by $\log_2 16 = 4$ bits. The PRNG operation from line 6 is described in Sec. III-B. The vector normalization from line 9 is performed in two steps: The first step consists of a pseudonormalization (described in Sec. III-C) to reduce the bitwidth of the entries of $\mathbf{a}_i'$ for the subsequent exact normalization. The exact normalization is performed by multiplying the pseudonormalized $\mathbf{a}_i'$ with the inverse square root (see Sec. III-D) of its squared ℓ_2-norm $\|\mathbf{a}_i'\|_2^2$, with $\|\mathbf{a}_i'\|_2^2$ being computed using the adder tree configuration shown in Fig. 2(c). The adder tree configuration is also used to compute the inner products on lines 11 and 13, the matrix-vector and matrix-matrix products on line 12, and the trace operations $\mathrm{tr}(\cdot)$ on line 14. Due to pipelining, a single inner product has a latency of 5 clock cycles. Finally, the score module utilizes its own real-valued multipliers and subtractors to complete the computation of N, D, and $N - D\tau$ (lines 13–15), to determine if the programmable threshold τ was reached. In total, 268 clock cycles are required to process one delay index ℓ, i.e., to evaluate whether or not the synchronization sequence occurred within the windowed receive signal $\mathbf{Y}_\ell$.

B. Pseudorandom Number Generator (PRNG)

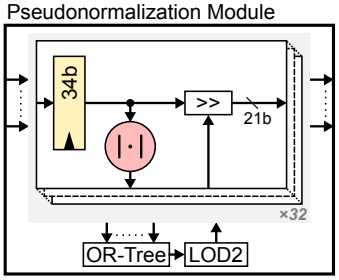
The PRNG consists of two 32-bit xorshift [8] blocks as visible in Fig. 3(a) to produce both real and imaginary parts of the output within one clock cycle. For the first xorshift block, the initial state for the first delay index ℓ can be programmed, while for all subsequent indices, the output state from the second xorshift is fed back to the input of the first xorshift.

C. $\|\cdot\|_\infty$–Pseudonormalization (PN) Module

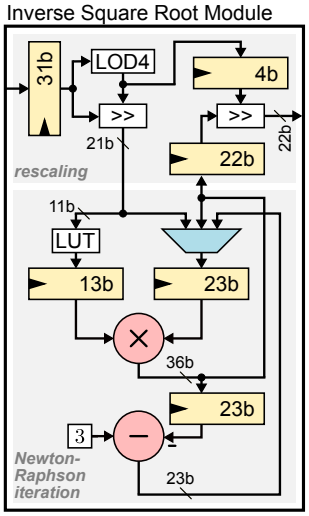
The jammer's large dynamic range requires the PEs' datapath to have larger bitwidths [9]. This issue is particularly



(a) PRNG producing complex-valued numbers using xorshift.



(b) $\|\cdot\|_\infty$–pseudonormalization module to rescale complex-valued vectors.



(c) Module for computing the inverse square root using a LUT and a Newton-Raphson iteration.

Fig. 3. Internal structure of the (a) PRNG module, (b) $\|\cdot\|_\infty$–PN module, and (c) inverse square root module of the JASS hardware architecture.

exacerbated when computing quantities directly associated to the jammer's power, such as $\|\mathbf{a}_i'\|_2^2$. Instead of using additional PEs with even larger bitwidths to compute $\|\mathbf{a}_i'\|_2^2$ as it was done in [4], we use the $\|\cdot\|_\infty$–PN module to scale the entries of the vector $\mathbf{a}_i'$ down into a known range, thereby reducing the bitwidth of $\|\mathbf{a}_i'\|_2^2$. To arrive at a low-complexity, hardware-friendly normalization, we use the pseudonorm $\|\cdot\|_\infty \triangleq 2^n$, where $n = \lfloor \log_2(\max\{\|\Re\{\cdot\}\|_\infty, \|\Im\{\cdot\}\|_\infty\}) \rfloor$ and $\|\cdot\|_\infty$ is the infinity norm. As illustrated in Fig. 3(b), normalization with respect to this pseudonorm amounts to an arithmetic right shift by n bits, where n is simply computed using an OR-tree to combine the absolute values of the real and imaginary parts of all entries of $\mathbf{a}_i'$, followed by a leading-one detector (LOD) with base 2 (LOD2). After the $\|\cdot\|_\infty$–PN module, the real and imaginary parts of the entries of $\mathbf{a}_i'$ are in the range $[-2, 2)$ and are represented with 21 bits instead of 34 bits.

D. Inverse Square Root Module

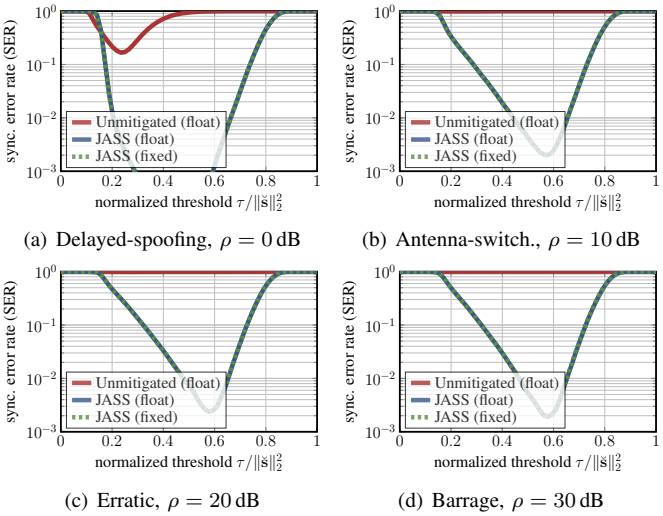
The inverse square root module is shown in Fig. 3(c). First, the input x is rescaled as $x' = x/2^{2\alpha}$, where $\alpha \in \mathbb{Z}$ is found using a LOD with base 4 (LOD4), so that $x' \in [0.25, 1)$. Then, an initial estimate of $1/\sqrt{x'}$ is fetched from a look-up table (LUT) and refined with one Newton-Raphson iteration: $y = y_{\mathrm{LUT}}(3 - y_{\mathrm{LUT}}^2 x')/2$. Here, a dedicated real-valued multiplier and subtractor are used to avoid the less efficient complex-valued datapath of the PEs. Finally, $y \approx 1/\sqrt{x'} = 2^\alpha/\sqrt{x}$ is scaled back to obtain the desired $1/\sqrt{x}$.

IV. ASIC Implementation Results

A. Synchronization Error Rate (SER) Performance

Fig. 4 shows the synchronization error rate (SER) as a function of the threshold[1] τ for our fixed-point hardware

[1] A large threshold τ entails a low probability of a false alarm but a high probability of a miss, while the opposite holds for a small τ—the optimal τ balances between false alarms and misses.



(a) Delayed-spoofing, $\rho = 0\,\text{dB}$ (b) Antenna-switch., $\rho = 10\,\text{dB}$

(c) Erratic, $\rho = 20\,\text{dB}$ (d) Barrage, $\rho = 30\,\text{dB}$

Fig. 4. Synchronization error rate (SER) performance against a two-antenna jammer at a signal-to-noise ratio (SNR) of 5 dB, for different jammer types and jammer-to-signal ratios ρ. The delayed-spoofing jammer repeats the synchronization sequence $\check{\mathbf{s}}$ with one sample delay; the antenna-switching jammer transmits Gaussian symbols using sometimes one antenna, sometimes the other; the erratic jammer transmits Gaussian symbols at random times and is otherwise silent; and the barrage jammer transmits white Gaussian noise.

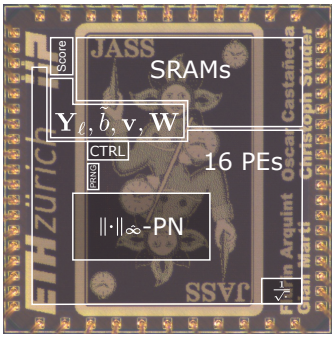


Fig. 5. Micrograph of the 2 mm × 2 mm JASS ASIC in TSMC 65 nm LP with highlighted modules.

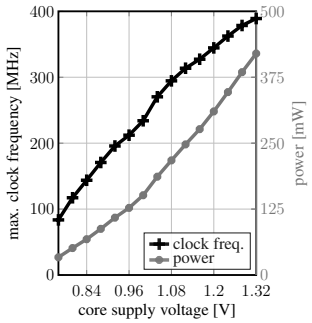


Fig. 6. Measured maximum clock frequency and power consumption versus core supply voltage.

implementation of JASS under different types of two-antenna jammers with different jammer-to-signal ratios ρ. In all four jammer scenarios, JASS consistently and substantially outperforms an unmitigated synchronization approach (i.e., setting the matrix $\mathbf{A}$ to the all-zeroes matrix), which fails due to jamming. It is also evident from Fig. 4 that the fixed-point arithmetic[2] implemented in the JASS ASIC incurs in virtually no performance loss compared to a floating-point baseline.

B. ASIC Measurements

Fig. 5 presents the micrograph of the $4\,\text{mm}^2$ JASS ASIC fabricated in TSMC 65 nm LP, with its different modules, including the control unit and SRAMs, highlighted within the $2.87\,\text{mm}^2$ core area. The SRAMs are used to store up to 1024 receive samples $\mathbf{y}[k]$. At nominal core supply voltage of 1.2 V and a room temperature of 300 K, the ASIC achieves a maximum clock frequency of 344 MHz, which corresponds to a sampling rate of 1.28 mega-samples per second (MS/s). At this operating point, the ASIC (including SRAMs) consumes 310 mW when facing a two-antenna barrage jammer with $\rho = 30\,\text{dB}$. Table I summarizes the performance metrics of the JASS ASIC, while Fig. 6 shows how the clock frequency and power scale with the core supply voltage.

[2]The bitwidths of the fixed-point implementation are indicated in Figs. 1 and 3, where the $(2\times)$ accounts for real and imaginary components.

TABLE I
MEASUREMENT RESULTS FOR THE JASS ASIC

Receiver antennas B	16	Core area [mm^2]	2.87
Sync. seq. length K	16	Frequency [MHz]	344
Jammer antennas I	0–2	Sampling rate [MS/s]	1.28
Technology [nm]	65	Power [mW]	310
Supply [V]	1.2	Area eff. [MS/s/mm^2]	0.45
		Energy/sample [nJ/S]	242

As JASS is the first jammer-resilient synchronization ASIC, a comparison with other designs is not possible. Nevertheless, we contextualize our results by noting that, after technology normalization, our JASS ASIC (including SRAMs) occupies 9% of the area of the jammer-resilient MIMO detector from [4], while consuming 8% of its power when operating at the same clock frequency. We emphasize that jammer resilience comes at the cost of a significantly lower hardware efficiency and throughput [4]. This remains true for our JASS ASIC, especially when considering that it is able to counteract smart jammers with more than one antenna. Still, higher sampling rates could be achieved at the cost of silicon area by having several instances of the proposed JASS architecture operating in parallel on different delay indexes ℓ, or simply by reimplementing JASS in a more advanced technology node.

V. Conclusions

We have presented the *first* ASIC for jammer-resilient time synchronization reported in the open literature. Even when confronted with smart and strong jammers, our JASS ASIC performs accurate time synchronization at 1.28 MS/s when implemented on $2.87\,\text{mm}^2$ in a 65 nm technology node.

References

[1] T. T. Do, E. Björnsson, E. G. Larsson, and S. M. Razavizadeh, "Jamming-resistant receivers for the massive MIMO uplink," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 210–223, Jan. 2018.

[2] L. M. Hoang, J. A. Zhang, D. N. Nguyen, X. Huang, A. Kekirigoda, and K.-P. Hui, "Suppression of multiple spatially correlated jammers," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10 489–10 500, Oct. 2021.

[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] T. M. Schmidl and D. C. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Trans. Commun.*, vol. 45, no. 12, pp. 1613–1621, Dec. 1997.

[6] D. W. Bliss and P. A. Parker, "Temporal synchronization of MIMO wireless communication in the presence of interference," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1794–1806, Mar. 2010.

[7] G. Marti, F. Arquint, and C. Studer, "Jammer-resilient time synchronization in the MIMO uplink," *IEEE Trans. Signal Process.*, vol. 73, 2025.

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

[9] G. Marti, A. Stutz-Tirri, and C. Studer, "Fundamental limits for jammer-resilient communication in finite-resolution MIMO," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Oct. 2024.