IEEE Open Journal of
Circuits and Systems

Received 9 August 2020; revised 9 November 2020; accepted 29 November 2020. Date of current version 28 January 2021.

Digital Object Identifier 10.1109/OJCAS.2020.3043643

Massive Machine-Type Communication Pilot-Hopping Sequence Detection Architectures Based on Non-Negative Least Squares for Grant-Free Random Access

NARGES MOHAMMADI SARBAND (Student Member, IEEE),
EMA BECIROVIC (Graduate Student Member, IEEE), MATTIAS KRYSANDER,
ERIK G. LARSSON (Fellow, IEEE), AND OSCAR GUSTAFSSON (Senior Member, IEEE)

Department of Electrical Engineering, Linköping University, 581 83 Linköping, Sweden

This article was recommended by Associate Editor C. Studer.

CORRESPONDING AUTHOR: O. GUSTAFSSON (e-mail: oscar.gustafsson@liu.se)

This work was supported in part by the ELLIIT Strategic Research Environment.

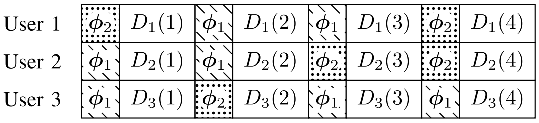
ABSTRACT User activity detection in grant-free random access massive machine type communication (mMTC) using pilot-hopping sequences can be formulated as solving a non-negative least squares (NNLS) problem. In this work, two architectures using different algorithms to solve the NNLS problem is proposed. The algorithms are implemented using a fully parallel approach and fixed-point arithmetic, leading to high detection rates and low power consumption. The first algorithm, fast projected gradients, converges faster to the optimal value. The second algorithm, multiplicative updates, is partially implemented in the logarithmic domain, and provides a smaller chip area and lower power consumption. For a detection rate of about one million detections per second, the chip area for the fast algorithm is about 0.7 mm^2 compared to about 0.5 mm^2 for the multiplicative algorithm when implemented in a 28 nm FD-SOI standard cell process at 1 V power supply voltage. The energy consumption is about 300 nJ/detection for the fast projected gradient algorithm using 256 iterations, leading to a convergence close to the theoretical. With 128 iterations, about 250 nJ/detection is required, with a detection performance on par with 192 iterations of the multiplicative algorithm for which about 100 nJ/detection is required.

INDEX TERMS 5G mobile communication, base stations, Internet of Things, machine-to-machine communications, MIMO.

I. INTRODUCTION

Massive machine type communication (mMTC) is a core use case in 5G and later generations wireless communication systems, where a large number of users are expected to be served [1]–[6]. These users are expected to intermittently send small amounts of data, primarily in the uplink, i.e., from the users to the base station. As the mMTC users send a limited amount data, the impact of overhead signaling such as random access or scheduling requests is large. Therefore, a grant-free random access scheme is beneficial for these types of users. In a grant-free random access scheme all the active users will transmit their data at the same time without waiting for a grant from the base station. To be able to be distinguished by the base station, the active users also need to transmit a unique identifier.

Grant-free random access with massive MIMO, i.e., base stations with a large number of antennas, has been widely studied in many papers [6]–[16]. Conventionally in massive MIMO, the users transmit a pilot in each coherence interval that is used by the base station to estimate the channel. Often these pilots are mutually orthogonal such that the channel estimates will be good. When a grant-free random access scheme is applied to massive MIMO, the same pilots are used to detect the active users. If the number of users exceeds the dimension (in samples) of the channel coherence interval, the users cannot be allocated mutually orthogonal



User 1	ϕ_2	$D_1(1)$	ϕ_1	$D_1(2)$	ϕ_1	$D_1(3)$	ϕ_2	$D_1(4)$
User 2	ϕ_1	$D_2(1)$	ϕ_1	$D_2(2)$	ϕ_2	$D_2(3)$	ϕ_2	$D_2(4)$
User 3	ϕ_1	$D_3(1)$	ϕ_2	$D_3(2)$	ϕ_1	$D_3(3)$	ϕ_1	$D_3(4)$

FIGURE 1. **Three users transmitting pilot-hopping sequences of length four using two pilots, ϕ_1 and ϕ_2. Additionally, in coherence interval *t* the user *k* also transmits uplink data $D_k(t)$.**

pilots, which leads to pilot collisions. There are two fundamentally different approaches for solving this problem. In the first, pilots are drawn at random and allowed to be non-orthogonal [8], [9], [11], [12], [14]–[16]. In the other, the pilots are orthogonal, but the data transmission is spread over many coherence intervals [7], [13], i.e., pilot-hopping sequences are detected instead of pilots. To reduce the complexity of the detection problem, a common approach for grant-free user detection with massive MIMO is to utilize the sparsity of the solution, i.e., that only a small subset of users are active at the same time. Hence, many of the existing solutions use methods developed in the field of compressed sensing [8]–[13], [16].

The scenario for detecting pilot-hopping sequences for grant-free random access was introduced in [7]. However, in [7] perfect detection of the pilot-hopping sequences was assumed. In [13], it was suggested that the non-negative least-squares (NNLS) is an appropriate problem formulation for the pilot-hopping sequence detection problem.

The active users transmit one pilot per coherence interval (the time-frequency interval in which the channel can be assumed to be constant), according to a predetermined sequence. The pilot-hopping sequences are unique for every user but more than one user can transmit the same pilot in a coherence interval. The pilots in each coherence interval are accompanied by an uplink data part. An example of this scheme can be found in Fig. 1, with three users, two pilots and a pilot-hopping sequence of length four.

In this article, two architectures for solving this problem using the approach in [13] is proposed. To the best of the authors' knowledge, the only other implementations of similar detectors are [17], implementing the algorithm from [11], and [18], implementing the algorithm from [12]. However, these are based on the other principle: having non-orthogonal pilot sequences drawn from random sources.

A preliminary version of this work was presented in [19]. In addition to a more detailed presentation and more detailed results, an alternative NNLS algorithm is also considered here. The implementation of the alternative algorithm is shown to require smaller chip area and less energy, although it has slightly worse detection performance for hard[1] channel conditions.

In the next section, the detection algorithm is reviewed, algorithms for solving NNLS problems discussed, and the general proposed architectures introduced. Then, in

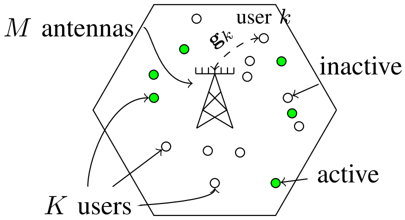


FIGURE 2. **System model: a cell with a base station using *M* antennas serving *K* users, where only a subset is active. The channel between user *k* and the base station is denoted by $\mathbf{g}_k$.**

Section III, implementation results and decisions are presented for a considered scenario. Finally, some concluding remarks are given in Section IV.

II. METHODS AND PROCEDURES

A. ALGORITHM TO DETECT PILOT-HOPPPING SEQUENCE

We consider a single-cell massive MIMO system where the base station is equipped with M antennas and serves K mMTC users, each with a single antenna, see Fig. 2. We assume that only a subset of these users is active at once. We consider a block fading channel model where the channel of user k in coherence interval t is denoted by $\mathbf{g}_k^t \in \mathbb{C}^M$. The users transmit T coherence interval long pilot-hopping sequences. There are τ_p mutually orthogonal pilots. Each pilot consists of τ_p symbols and have unit norm. At the pilot phase of coherence interval t, the base station receives

$$\mathbf{Y}^t = \sum_{k=1}^{K}\sum_{j=1}^{\tau_\text{p}} \alpha_k S_{j,k}^t \sqrt{\tau_\text{p} p_k}\, \mathbf{g}_k^t \boldsymbol{\phi}_j^H + \mathbf{N}^t, \tag{1}$$

where

$$\alpha_k = \begin{cases} 1 & \text{if user } k \text{ is active,} \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

$$S_{j,k}^t = \begin{cases} 1, & \text{if user } k \text{ sends pilot } j \text{ at pilot phase } t, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

p_k is the transmit power of user k, $\boldsymbol{\phi}_j \in \mathbb{C}^{\tau_\text{p}}$ is the jth pilot, and $\mathbf{N}^t \in \mathbb{C}^{M\times\tau_\text{p}}$ is noise with i.i.d. $\mathcal{CN}(0,\sigma^2)$ elements. The model uses the assumption that the users start their pilot-hopping sequences at the same coherence interval.

In each coherence interval the base station computes an estimate of the received signal energy over each pilot as

$$E_{i,t} = \frac{(\mathbf{Y}^t\boldsymbol{\phi}_i)^H(\mathbf{Y}^t\boldsymbol{\phi}_i)}{M} - \sigma^2 = \frac{\left\|\mathbf{Y}^t\boldsymbol{\phi}_i\right\|^2}{M} - \sigma^2. \tag{4}$$

With block Rayleigh fading,

$$\mathbf{g}_k \sim \mathcal{CN}(\mathbf{0}, \beta_k \mathbf{I}_M), \tag{5}$$

we have the channel hardening and favorable propagation properties of massive MIMO [20]. The channel harding comes from

[1] Unfavorable channel hardening and unfavorable propagation such that the linearization in (8) is not a good approximation.

$$\frac{\|\mathbf{g}\|^2}{M} \rightarrow \beta_k, \quad \text{as} \quad M \rightarrow \infty, \tag{6}$$

while the favorable propagation comes from

$$\frac{\mathbf{g}_k^H \mathbf{g}_{k'}}{M} \rightarrow 0, \quad \text{as} \quad M \rightarrow \infty, k \neq k'. \tag{7}$$

Using these properties, we can state the "asymptotic energies" as

$$E_{i,t} = \frac{\left\|\mathbf{Y}^t \boldsymbol{\phi}_i\right\|^2}{M} - \sigma^2 \rightarrow \sum_{k=1}^{K} \alpha_k S_{i,k}^t \tau_{\mathrm{p}} p_k \beta_k \quad \text{as} \quad M \rightarrow \infty. \tag{8}$$

From this limit, we can see that the asymptotic energies are linear in the user activity parameters. We assume that the users are using statistical channel inversion power control [21],

$$p_k = p \frac{\min_{k'} \beta_{k'}}{\beta_k} = p \frac{\beta_{\min}}{\beta_k}, \tag{9}$$

where $\beta_{\min} = \min_{k'} \beta_{k'}$ is the large-scale fading to the weakest user and p is a system wide power parameter. With statistical channel information, the received signal power at the base station from each user is the same.

We use the asymptotic energies and the fact that the users perform statistical channel inversion to form the column vector

$$\mathbf{b} = \frac{1}{\tau_{\mathrm{p}} p \beta_{\min}} \begin{pmatrix} E_{1,1} \\ \vdots \\ E_{\tau_{\mathrm{p}},1} \\ \vdots \\ E_{i,t} \\ \vdots \\ E_{1,T} \\ \vdots \\ E_{\tau_{\mathrm{p}},T} \end{pmatrix} \tag{10}$$

and the $T\tau_p \times K$-matrix

$$\mathbf{A} = \begin{pmatrix} S_{1,1}^1 & \cdots & S_{1,K}^1 \\ \vdots & \ddots & \vdots \\ S_{\tau_{\mathrm{p}},1}^1 & \cdots & S_{\tau_{\mathrm{p}},K}^1 \\ \vdots & \ddots & \vdots \\ S_{1,1}^T & \cdots & S_{1,K}^T \\ \vdots & \ddots & \vdots \\ S_{\tau_{\mathrm{p}},1}^T & \cdots & S_{\tau_{\mathrm{p}},K}^T \end{pmatrix}. \tag{11}$$

One can note that $\sum_{j=1}^{\tau_p} S_{j,k}^t = 1, \ \forall k, t$, as user k uses exactly one pilot sequence in coherence time interval t.

The $\mathbf{A}$-matrix for the three users in Fig. 1 is

$$\begin{array}{c} \text{User} \\ \begin{array}{ccc} 1 & 2 & 3 \end{array} \end{array}$$

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{array}{l} \left.\begin{array}{l} \\ \end{array}\right\} t = 1 \\ \left.\begin{array}{l} \\ \end{array}\right\} t = 2 \\ \left.\begin{array}{l} \\ \end{array}\right\} t = 3 \\ \left.\begin{array}{l} \\ \end{array}\right\} t = 4 \end{array} \tag{12}$$

With this vector and matrix, we can state the asymptotic energy equation as

$$\mathbf{A}\mathbf{x} \rightarrow \mathbf{b} \quad \text{as} \quad M \rightarrow \infty, \tag{13}$$

where

$$\mathbf{x} = \begin{pmatrix} \alpha_1 & \ldots & \alpha_k & \ldots & \alpha_K \end{pmatrix}^T.$$

We use the fact that not all the users are active at the same time, i.e., that $\mathbf{x}$ is sparse, to solve the pilot-hopping sequence detection problem. In [13], it was shown that formulating this as an NNLS problem produce sparse solutions that solve the pilot-hopping sequence detection problem. It was also shown that using additional sparsity promoting techniques, such as NNLS combined with L_1-minimization, do not improve the results.

B. ALGORITHMS FOR SOLVING NNLS

The non-negative least squares (NNLS) problem can be stated as

$$\begin{aligned} &\text{minimize} \quad \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \\ &\text{subject to} \quad \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{14}$$

where the design matrix, $\mathbf{A}$ and an observation vector, $\mathbf{b}$, are given. $\mathbf{0}$ denotes a vector with all zeros. The solution vector, $\mathbf{x}$, minimizes the error of $\mathbf{A}\mathbf{x} - \mathbf{b}$ in the least squares sense subject to all elements in $\mathbf{x}$ being non-negative. Over the years, many different algorithms have been proposed to solve NNLS problems. For an overview of different NNLS algorithms, we refer the reader to [22].

The probably most well-known approach is the active set method by Lawson and Hanson [23]. However, an active set-based algorithm relies heavily on branching making it unsuitable for a parallel high-speed implementation since parts of the execution will be sequential in nature. Besides, the execution is data dependent making it non-deterministic.

Instead, we here use two algorithms with a much higher degree of computational parallelism. As they are iterative, there are still limitations, but as later seen, a very high degree of parallelism is still obtained. The execution in each iteration is deterministic simplifying the design of any control. In addition, since these algorithms rely heavily on matrix-vector multiplications, the structure of $\mathbf{A}$, being a binary matrix consisting of only 0/1 entries as illustrated in (12), can be readily utilized.

1) FAST PROJECTED GRADIENT FOR SOLVING NNLS–FAST

The fast projected gradient algorithm [24], is described as

$$\mathbf{x}^{(0)} = \mathbf{0},\ \ \mathbf{p}^{(0)} = \mathbf{0} \tag{15}$$

$$\mathbf{x}^{(k+1)} = \max\left\{\mathbf{0},\ \mathbf{p}^{(k)} - \frac{1}{L}\mathbf{A}^T\left(\mathbf{A}\mathbf{p}^{(k)} - \mathbf{b}\right)\right\} \tag{16}$$

$$\mathbf{p}^{(k+1)} = \mathbf{x}^{(k+1)} + s^{(k)}\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right), \tag{17}$$

where

$$c^{(k)} = \frac{1 + \sqrt{1 + 4\left(c^{(k-1)}\right)^2}}{2},\ \ s^{(k)} = \frac{c^{(k-1)} - 1}{c^{(k)}}, \tag{18}$$

with $c^{(-1)} = 0$. L is the Lipschitz constant which controls the step size in each iteration. It should be larger or equal to the spectral radius of $\mathbf{A}^T\mathbf{A}$, i.e., the largest absolute eigenvalue. For a given $\mathbf{A}$, this can be computed off-line. Replacing $\mathbf{p}^{(k)}$ with $\mathbf{x}^{(k)}$ in (16) and not using (17) and (18) gives the standard projected gradient algorithm for NNLS. The fast part comes from (17), where the previous iteration value is used to accelerate the convergence. $s^{(k)}$ is the step length of the fast part and $\mathbf{p}^{(k)}$ is now denoted the predictor [24].

Note here that the core of the algorithm, the matrix-vector multiplications, will be simple additions since the elements of $\mathbf{A}$ are 0 or 1 as seen from (12). As such, the architectures can not be directly used for NNLS problems without this property.

2) MULTIPLICATIVE UPDATES FOR SOLVING NNLS–MULT

In the multiplicative updates algorithm, the updates in each iteration are instead performed using multiplications and divisions and is described as [25]–[27]:

$$\mathbf{x}^{(0)} = \mathbf{1} \tag{19}$$

$$\mathbf{d} = \mathbf{A}^T\mathbf{b} \tag{20}$$

$$\mathbf{e}^{(k+1)} = \mathbf{A}^T\mathbf{A}\mathbf{x}^{(k)} \tag{21}$$

$$\mathbf{x}^{(k+1)} = \mathbf{d} \odot \mathbf{x}^{(k)} \oslash \mathbf{e}^{(k+1)}, \tag{22}$$

where $\mathbf{1}$ denotes a vector with all ones and $\odot$ and $\oslash$ represent component-wise (Hadamard) vector multiplication and division, respectively.

Similar to the Fast algorithm, the matrix-vector multiplications consist of additions.

C. PROPOSED ARCHITECTURES

The proposed architectures are iso-morphic mappings of a single iteration of the respective algorithms. Hence, each clock cycle one complete iteration of the algorithms is performed. The primary reason to do this is to obtain a high degree of parallelism and to easily benefit from the fact that $\mathbf{A}$ only consists of 0 and 1 entries, leading to additions only. Since the matrices $\mathbf{A}$ and $\mathbf{A}^T$ are constant in all iterations, these constant matrix-vector multiplications can be converted to summation structures where sub-expression sharing can be used to reduce the area and power consumption of the implementation.

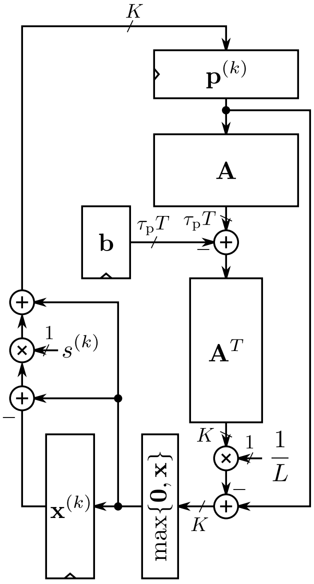


FIGURE 3. Proposed architecture of Fast algorithm with number of parallel signals annotated.

1) FAST

The resulting architecture is shown in Fig. 3. It should be noted that most signals are vectors. There are registers to store $\mathbf{b}$, which is given at the start of a detection process, and $\mathbf{p}^{(k)}$ and $\mathbf{x}^{(k)}$, which are initialized to zero at the start as seen in (15).

To implement multiplication by $s^{(k)}$, the value of it should be calculated in each iteration and then multiplied. As visible from (18), the calculation of $s^{(k)}$ is costly in hardware, but it is possible to compute the $s^{(k)}$ values off-line and save them in a memory. Furthermore, $s^{(k)}$ converges to one after some time, so for k and all later iterations, the values will saturate to the largest representable number less than or equal to one.

2) MULT

To avoid the relatively costly multiplications and divisions of the Mult algorithm, a logarithmic number system (LNS) [28] is used for those parts. This reduces the multiplications and divisions to additions and subtractions, respectively, at the expense of converting the numbers between the linear and logarithmic domains.

The multiplicative updates algorithm, partly in the logarithmic domain, is described as:

$$\hat{\mathbf{x}}^{(0)} = \mathbf{0} \tag{23}$$

$$\hat{\mathbf{d}} = \log\left(\mathbf{A}^T\mathbf{b}\right) \tag{24}$$

$$\hat{\mathbf{e}}^{(k+1)} = \log\left(\mathbf{A}^T\mathbf{A}\mathbf{x}^{(k)}\right) \tag{25}$$

$$\hat{\mathbf{x}}^{(k+1)} = \hat{\mathbf{d}} + \hat{\mathbf{x}}^{(k)} - \hat{\mathbf{e}}^{(k+1)} \tag{26}$$

$$\mathbf{x}^{(k+1)} = \exp\left(\hat{\mathbf{x}}^{(k+1)}\right). \tag{27}$$

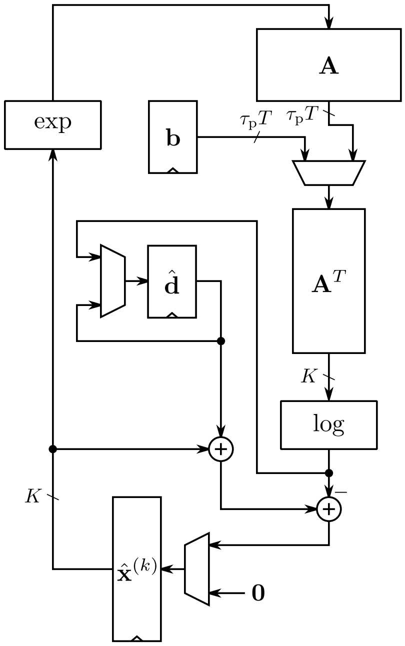


FIGURE 4. Proposed architecture of Mult algorithm partially implemented in the logarithmic domain with number of parallel signals annotated.

The resulting architecture is shown in Fig. 4. Note that $\hat{\mathbf{d}}$ is computed in the first clock cycle and that the first iteration is performed in the second clock cycle. The multiplexers are used for storing the computed value of $\hat{\mathbf{d}}$ and to initialize $\hat{\mathbf{x}}^{(0)}$ in the first clock cycle. Hence, one more clock cycle is required here compared to the Fast algorithm for the same number of iterations.

To compute the logarithm and exponentiation, either a table or using Mitchell's logarithm approximation [29] is considered. There are many methods suggested to either perform approximations of the logarithm and exponent functions using smaller tables or to improve the accuracy of Mitchell's algorithm. However, as seen in the results section, the accuracy of the Mitchell approximation is enough.

The idea of Mitchell's logarithm approximation is to utilize the approximation

$$\log(x) \approx x - 1 \tag{28}$$

in the range $1 \leq x \leq 2$.

To benefit from this, the position of the leading one, i.e., the most significant bit being one, of the input x is determined[2] and then shifted to the range $1 \leq x < 2$. The position of the first one also determines the integer part of the logarithm. Then, the leading one is removed and the fractional part of the logarithm is obtained as per (28).

For computing the exponent, the opposite procedure is performed. A one is put in front of the fractional part and the final result is obtained by shifting the fractional result with the leading one according to the number of integer bits.

TABLE 1. Parameters for the implemented example.

Parameter	Value
Number of users, K	1024
User activity	2^{-6}
Number of antennas, M	96
Number of orthogonal pilots, τ_p	16
Number of transmitted pilots, T	8

3) A NOTE ON PIPELINING

It should be noted here that pipelining will not provide any advantage as we need the result of one iteration before being able to start the next one. If required, one may interleave several detections by introducing pipelining, although that is not considered here since it will increase the detection latency and storage requirements. Unfolding may provide a small increase of detection rate as unfolding N times will most likely not decrease the clock frequency N times since the synthesis tool probably can optimize some parts better. Naturally, fully unfolding the architecture the required number of iterations will enable pipelining and streaming processing. However, the obtained detection rate is most likely more than enough for most practical current and envisioned scenarios.

III. RESULTS

For the implementation example, we consider a system with parameters as shown in Table 1. These parameters are selected based on the number of users and user activity, where the number of antennas, orthogonal pilots, and transmitted pilots are found by simulation to obtain a reasonable detection performance. For more results on how these parameters relate, we refer to [13]. Synthesis results are for 28 nm FD-SOI standard cells with a power supply voltage of 1.0 V, using Design Compiler. Power estimates for the final designs are obtained by simulating the synthesized netlist using SDF data to produce switching activities that are back annotated into the synthesis tool. Compared to the preliminary version [19], here the low leakage library with a slow-slow characterization at 125°C is used.

A. NUMBER OF ITERATIONS

Determining the number of iterations required for robust activity detection is important since this relates the clock frequency of the implementation with the number of detections per second. Here, we will consider floating-point implementation of the algorithms, while fixed-point implementation is further discussed in Section III-B.

First, the behavior of the algorithms is illustrated. When iterating the algorithms, the values in $\hat{\mathbf{x}}$ corresponding to active users will be close to one, while the values corresponding to inactive users will be close to zero. A typical sequence of values for a successful decoding is shown in Fig. 5, where the values of $\hat{\mathbf{x}}$ are shown after each iteration. Here, one can see that the Mult algorithm initially has a faster convergence for active users, but that around iteration 50, the Fast algorithm has caught up and reaches the final

[2] Here, the input is always non-negative, so the negative case where the leading zero should be detected does not have to be considered.

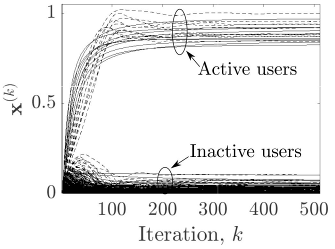


FIGURE 5. Convergence of $\hat{\mathbf{x}}$ using Fast (dashed) and Mult (solid) algorithms successfully performing an activity detection. Large values corresponds to active users and small values to inactive users. There is a clear separation between the active and inactive users.

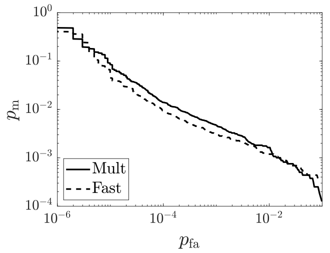


FIGURE 6. Missed detection, p_m, versus false alarm rates, p_fa, for 4000 iterations results of floating-point realizations of Fast and Mult algorithms.

values slightly faster. One can see that with about 100 iterations, it is possible to clearly separate the active and inactive users.

Typically, one does not use a fixed-threshold value, but studies the effect of the number of active users that are not detected (missed detections) and number of inactive users that are detected as active (false alarms) with a varying threshold. The probability of missed detections, p_m, and false alarm, p_fa, are calculated as

$$p_\mathrm{m} = \frac{\#\,\text{undetected active users}}{\#\,\text{active users}} \tag{29}$$

$$p_\mathrm{fa} = \frac{\#\,\text{detected inactive users}}{\#\,\text{inactive users}} \tag{30}$$

and plotted against each other in a receiver operating characteristic (ROC) curve. The ROC curves for the two considered algorithms are shown in Fig. 6 when using 4000 iterations, clearly more than in Fig. 5, and a large number of instances. Here, it can be seen that the Fast algorithm has a slightly better behavior compared to the Mult algorithm. Each point on the curve corresponds to a threshold and the position, the number of missed detections and false alarms if that threshold was used. Hence, the values in the top left corner corresponds to large thresholds and the values in the bottom right corner corresponds to small thresholds. For example, if a threshold of, say, 0.9 is used, there will almost never be any false alarms, but it is quite common that there are missed detections.

From Fig. 5, one can realize that for good convergence cases, it is possible to get a good separation of the active and inactive users is a few hundred iterations.

Instead, consider a case with bad convergence, as shown in Fig. 7. Here, the channel conditions and the selection of active users interacts in such a way that it is very hard to distinguish the active and inactive users. In these cases, one can expect that the curves of some active users are actually below the curves of some inactive users and that there are no distinct value that will separate the two groups.

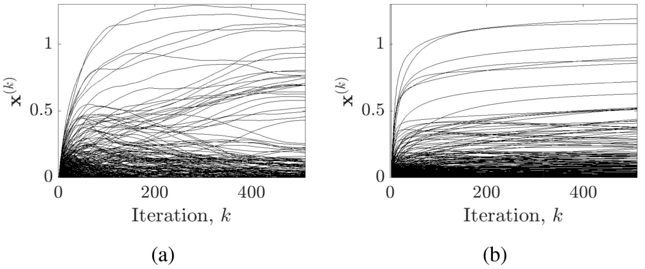


FIGURE 7. Convergence of $\hat{\mathbf{x}}$ using (a) Fast and (b) Mult algorithms where the active and inactive users are not clearly separated.

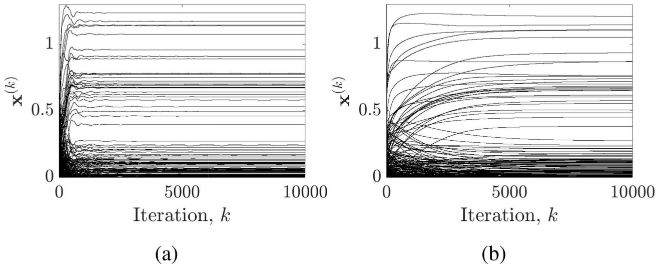


FIGURE 8. Convergence of $\hat{\mathbf{x}}$ using (a) Fast and (b) Mult algorithms where the active and inactive users are not clearly separated. Extended version of Fig. 7 covering 10000 iterations.

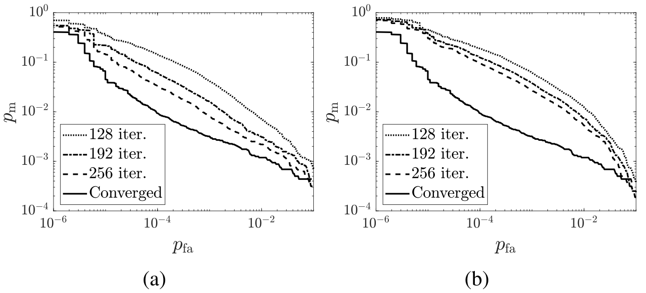


FIGURE 9. Missed detection, p_m, versus false alarm rates, p_fa, for floating-point implementations, 128 iterations, 192 iterations and 256 iterations for (a) Fast algorithm, and (b) Mult algorithm.

To study this further, the same case as in Fig. 7, was solved using 10000 iterations, with the results shown in Fig. 8. Here, it can be seen that the Fast algorithm in Fig. 8(a) reaches a more or less stable value in about 1000 iterations, while for the Mult algorithm in Fig. 8(b) some variables have not reached a stable value even after 10000 iterations.

Therefore, it can be argued that although Mult gives worse results in Fig. 6, the degradation comes primarily from cases where the results will contain detection errors anyway. Using even more iterations will make the curves in Fig. 6 identical. Hence, we will use the Fast algorithm curve in Fig. 6 as a reference curve for a converged solution.

In Fig. 9, the missed detection, p_m, and false alarm rates, p_fa, are shown for the floating-point implementations with

128, 192, and 256 iterations, for the Fast and Mult algorithms. As seen, the results are better for the Fast algorithm, but we know from previous discussions that the Mult algorithm converges faster in the good cases, so this will also be further considered.

B. WORD LENGTH OPTIMIZATION

A careful word length optimization has been performed to determine the minimum resolution of the signals while still being able to operate correctly. In addition, the maximum values have been determined by a combination of interval arithmetic and simulations. One example of the latter is the output of the $\mathbf{A}$ block that theoretically may require seven additional MSBs compared to the input as up to 87 $\mathbf{p}^{(k)}$-values are summed for the selected $\mathbf{A}$ since the maximum number of ones in a row of the selected $\mathbf{A}$ is 87. However, simulation show that at most five additional MSBs are required in practice as not all the $\mathbf{p}^{(k)}$-values will be close to one. Similarly, for the Fast algorithm, the subtraction of $\mathbf{x}^{(k)}$ from $\mathbf{x}^{(k+1)}$ results in a shorter output word length than the input word length. It can be seen from Figs. 5 and 7 that the values are correlated and do not change much between two iterations.

In the following, S and U denote signed and unsigned representation, respectively. The numbers following S and U are the number of integer and fractional bits, respectively. A negative number of integer bits denotes that the corresponding number of fractional bits are not required. As examples, $S(2, 2)$ denotes a four-bit signed number with values between $-\frac{2^{4-1}}{2^2} = -2$ and $\frac{2^{4-1}-1}{2^2} = 1.75$ and $U(-2, 4)$ denotes a two-bit unsigned number with values between 0 and $\frac{2^2-1}{2^4} = 0.1875$. In general, a $S(I, F)$ number goes from $-\frac{2^{I+F-1}}{2^F} = -2^{I-1}$ to $\frac{2^{I+F-1}-1}{2^F} = 2^{I-1} - 2^{-F}$ and a $U(I, F)$ number goes from 0 to $\frac{2^{I+F}-1}{2^F} = 2^I - 2^{-F}$.

For the nodes where quantization is performed, i.e., the number of fractional bits are reduced, rounding is used.

The resulting ROC curves for the resulting word lengths, further discussed below, are shown in Fig. 10. Here, one can see that 256 and 192 iterations are required for the Fast and Mult algorithms to reach close to the floating-point converged performance, respectively. On the other hand, 128 iterations are enough for the Fast algorithm to reach a performance similar to the Mult algorithm.

1) FAST

The value of L for the considered $\mathbf{A}$ is 520.74. Strictly, this should be rounded up to guarantee convergence under all possible inputs, but simulations show that selecting $L = 512$ works well, resulting in a shift instead of a multiplication. The values of $s^{(k)}$ are pre-computed and stored in a lookup table (all values for $k \geq 58$ are $1 - 2^{-5}$). The values for $\mathbf{p}^k$ closely follow the values of $\mathbf{x}^k$. As seen from Fig. 7(a), sometimes the values for $\mathbf{x}^k$, and therefore $\mathbf{p}^k$, can take on values above one. Hence, we saturate the values to at most one, using a single guard-bit. This operation is combined with the max operation in (16) to the min(max(.))-operation in Fig. 11. It can also be noted that (17) on rare occasions result in a small negative value, which is confirmed by simulations. Hence, min(max(.))-operations are performed before storing $\mathbf{p}^k$ as well to keep the word length limited and enabling storage of unsigned values only.

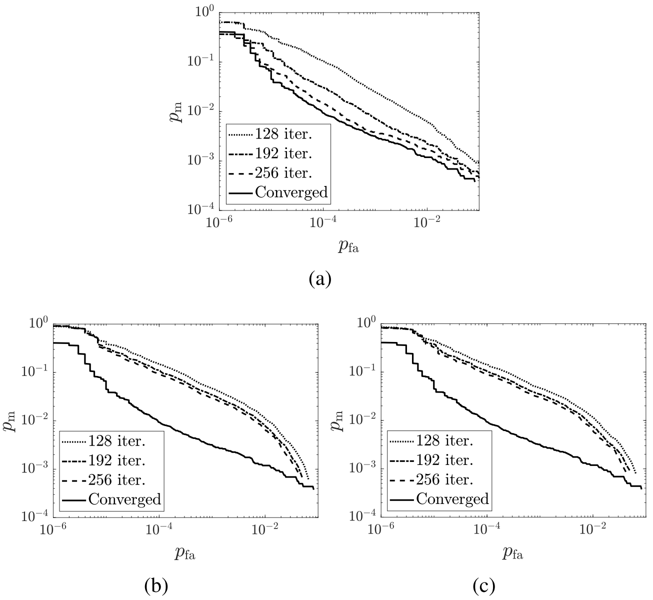


FIGURE 10. Missed detection, $p_{\rm m}$, versus false alarm $p_{\rm fa}$, rates for converged floating-point and fixed-point with 128, 192, and 256 iterations, for (a) Fast algorithm, (b) Mult algorithm using lookup tables for logarithm and exponent computations, and (c) Mult algorithm using Mitchell's approximated logarithm.

The resulting architecture with annotated word lengths is shown in Fig. 11. Rounding is here also performed for the input to the matrix-vector multiplication by $\mathbf{A}$, which significantly reduces the complexity of the computation by halving the input word length.

The implemented Fast algorithm can therefore be described as

$$\begin{aligned}
\mathbf{x}^{(0)} &= \mathbf{0},\ \ \mathbf{p}^{(0)} = \mathbf{0}\\
\mathbf{x}^{(k+1)} &= \min\left\{\mathbf{m}, \max\left\{\mathbf{0},\ \mathbf{p}^{(k)} - \frac{1}{L}\mathbf{A}^T\left(\mathbf{A}\mathbf{p}^{(k)} - \mathbf{b}\right)\right\}\right\}\\
\mathbf{p}^{(k+1)} &= \min\left\{\mathbf{m}, \max\left\{\mathbf{0},\ \mathbf{x}^{(k+1)} + s^{(k)}\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right)\right\}\right\}
\end{aligned}$$

where $\mathbf{m}$ is a vector of the largest positive value representable, $1 - 2^{-12}$.

2) MULT

As discussed earlier, the Mult architecture is partially based on computation in the logarithmic domain. The results in Figs. 10(b) and 10(c) show that the Mitchell logarithm approximation gives the same activity detection performance as using a lookup table.

As part of the word length optimization, the initial value of $\mathbf{x}^{(0)}$ was reduced from 1 to 2^{-7}. This reduces the values of $\mathbf{e}^{(1)}$, allowing a reduced word length, without reducing the

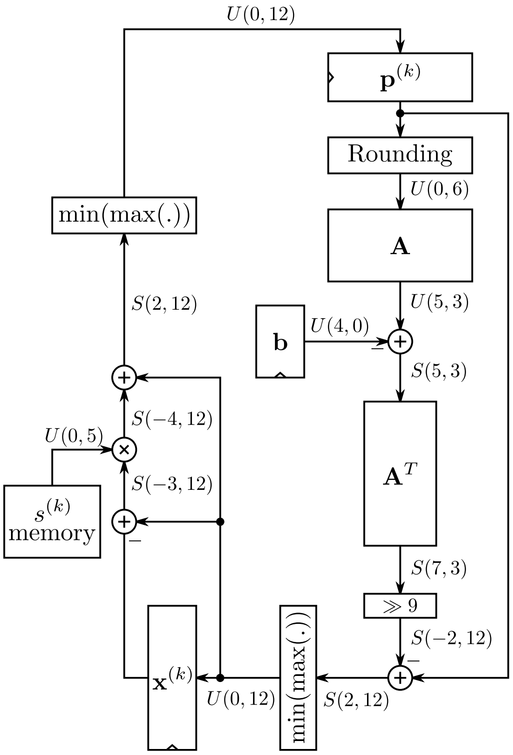


FIGURE 11. Modified proposed architecture for the Fast algorithm with annotated word lengths for the implemented example.

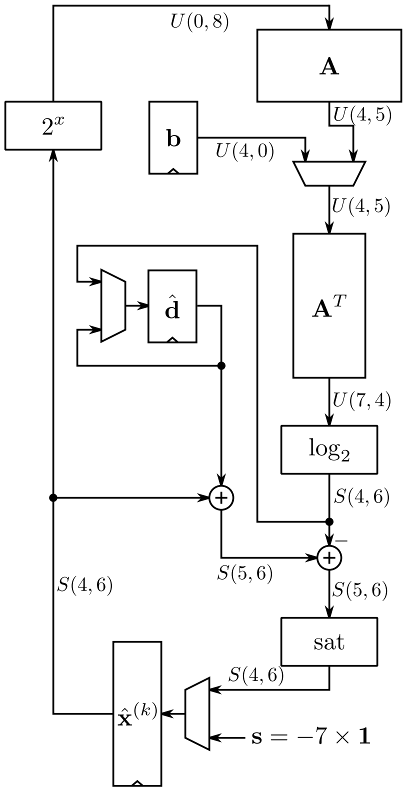


FIGURE 12. Modified proposed architecture for the Mult algorithm partially implemented in the logarithmic domain with annotated word lengths for the implemented example.

detection performance. In addition, saturation was introduced in order to handle potential overflows of $\hat{\mathbf{x}}^{(k+1)}$.

This leads to that the implemented Mult algorithm, partially in the logarithmic domain, is described as

$$\hat{\mathbf{x}}^{(0)} = \mathbf{s} \tag{31}$$

$$\hat{\mathbf{d}} = \log_2\left(\mathbf{A}^T\mathbf{b}\right) \tag{32}$$

$$\hat{\mathbf{e}}^{(k+1)} = \log_2\left(\mathbf{A}^T\mathbf{A}\mathbf{x}^{(k)}\right) \tag{33}$$

$$\hat{\mathbf{x}}^{(k+1)} = \text{sat}\left(\hat{\mathbf{d}} + \hat{\mathbf{x}}_i^{(k)} - \hat{\mathbf{e}}^{(k+1)}\right) \tag{34}$$

$$\mathbf{x}^{(k+1)} = 2^{\hat{\mathbf{x}}^{(k+1)}} \tag{35}$$

where $\mathbf{s}$ is a vector with the initial value, $\log_2(2^{-7}) = -7$, and sat corresponds to saturating the result in case of overflow. The resulting architecture with annotated word lengths is shown in Fig. 12.

The structure of the used logarithm approximation with base 2 is shown in Fig. 13. It consists of a leading one detector (LOD) that outputs the position of the first one in the input word, with the MSB corresponding to 0. As there in general are I integer bits, the value must be offset by $I-1$ to determine the correct integer part of the logarithmic value. The LOD value is also used to left-shift the input such that the leading one comes in the MSB position. This leading one is removed and the remaining part is used as the fractional part of the logarithm and combined with the integer part. Finally, as a zero input value will output the most negative output value, a simple saturation multiplexer is added. The programmable shift is realized using a 11-to-1-multiplexer, which provided the best synthesis results.

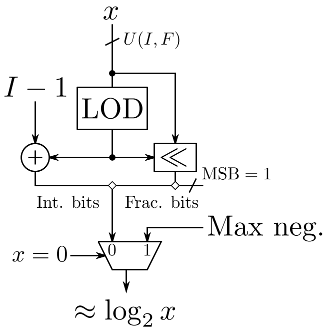


FIGURE 13. Realization of the Mitchell logarithm approximation with a $U(I, F)$ input format and zero input saturating to the most negative output value.

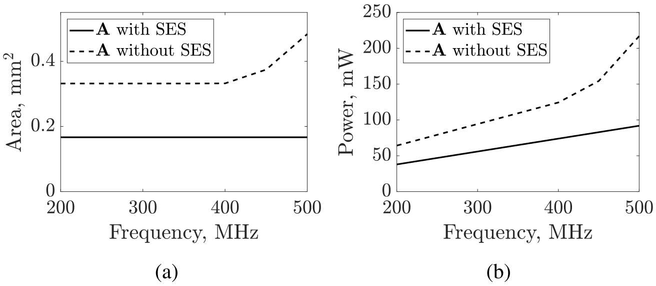
The logarithm approximation in Fig. 13 and the corresponding exponentiation require a significantly smaller area than using lookup tables and allow significantly higher clock frequencies. As the detection results are similar for both approaches, as seen from Figs. 10(b) and 10(c), only the Mitchell approximation version is considered in the following.

C. SUB-EXPRESSION SHARING

As mentioned in Section II-C, the number of adders for performing the matrix-vector multiplications by $\mathbf{A}$ and $\mathbf{A}^T$

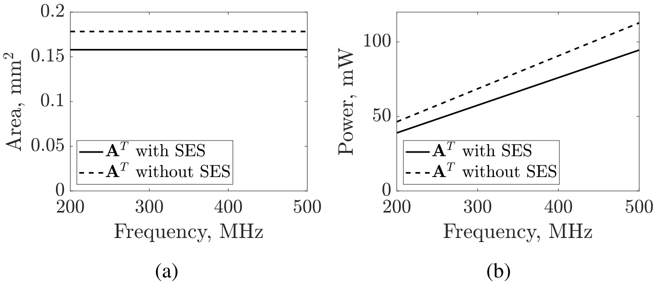
TABLE 2. Number of adders for the matrix-vector multiplications without and with sub-expression sharing in the implemented example.

	Adders		
Matrix	Without	With	Reduction
$\mathbf{A}$	8064	5699	29.3%
$\mathbf{A}^T$	7168	4709	34.3%
Total	15232	10408	31.7%



(a) (b)

FIGURE 14. (a) Area and (b) power results at different clock frequencies for the matrix-vector multiplication with A with and without using sub-expression sharing.



(a) (b)

FIGURE 15. (a) Area and (b) power consumption at different clock frequencies for the matrix-vector multiplication with $\mathbf{A}^T$ with and without using sub-expression sharing.

can be reduced. Here, we use an iterative two-term sub-expression sharing approach with the constraint that each addition should be performed at the minimum depth to keep the power consumption low. The results of this are shown in Table 2. As seen, a significant reduction is obtained compared to implementing the additions straightforwardly. For comparison, the other computations for the Fast approach are, as can be seen from Fig. 3, $3K + \tau_\mathrm{p}T = 3200$ additions/subtractions and $2K$ multiplications, of which K are reduced to a simple shift, so $K = 1024$ multiplications. For the Mult approach, $2K = 2048$ additions/subtractions, $K = 1024$ logarithm approximations, and $K = 1024$ exponent approximations are used, as evident from Fig. 4.

To further clarify the consequences of sub-expression sharing, the $\mathbf{A}$ and $\mathbf{A}^T$ matrix-vector multiplications have been synthesized separately for using the word lengths of the Fast implementation, i.e., the word lengths in Fig. 11. The area and power results are shown in Figs. 14 and 15 for $\mathbf{A}$ and $\mathbf{A}^T$, respectively. It can be seen that although the reduction in area is not the $\approx 30\%$ expected, more for $\mathbf{A}$ and less for $\mathbf{A}^T$, sub-expression sharing provides a significant area and power reduction.

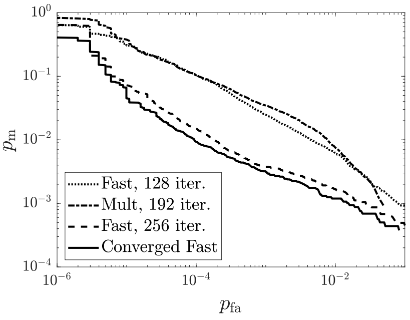


FIGURE 16. Missed detection, p_m, versus false alarm rates, p_fa, of the three considered implementation examples and the converged floating-point results.

1) SELECTING A

The number of ones in $\mathbf{A}$ is always KT, as each user uses T pilots. This leads to that the number of adders without sharing is the same independent of the pilot hopping sequences: $KT - \tau_\mathrm{p}T$ for $\mathbf{A}$ and $KT - K$ for $\mathbf{A}^T$. However, by selecting different pilot hopping sequences, different sub-expression sharing results can be obtained. Therefore, one can try to optimize $\mathbf{A}$ to give fewer adders, while still having good detection performance. One may note that qualitatively, detection performance increases the more different the pilot hopping sequences are, while the sharing increases with similarities in pilot hopping sequences.

Similarly, one can see that decreasing the length of the pilot hopping sequences T will lead to fewer adders, assuming that the number of users is constant. In [13], it was shown that detection performance scales with $\tau_\mathrm{p}T$. Hence, to decrease the implementation complexity with similar detection performance, one may decrease T and increase τ_p. However, this will lead to that less time is available to transmit data.

D. IMPLEMENTATION RESULTS

Based on the previous analyses, three different design cases are considered. A design using the Fast algorithm and 256 iterations provides detection results close to the converged floating-point results as shown in Fig. 10(a). For the Mult algorithm, 192 iterations are used since increasing the number of iterations beyond that only gives a limited detection gain. As discussed earlier, the Mult algorithm converges faster for the good cases, but slower for the bad cases, so depending on channel conditions etc, the actual detection performance may differ, and the different approaches may be beneficial in different scenarios. Finally, the Fast algorithm with 128 iterations gives about the same detection performance as the Mult algorithm with 192 iterations and is also included. The design is identical, and it is only the number of iterations in the power simulations that differ. The corresponding ROC curves are shown in Fig. 16.

The designs were synthesized for increasing clock frequency and power simulated. The results in terms of area and power are shown in Fig. 17. It can be seen that the area and power is significantly smaller for the Mult algorithm,

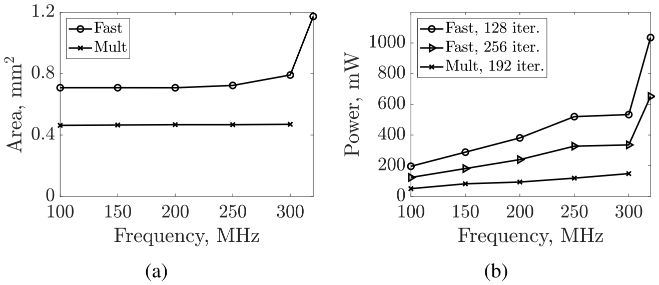


FIGURE 17. (a) Area and (b) power consumption at different clock frequencies for the implementations of Fast and Mult algorithms.

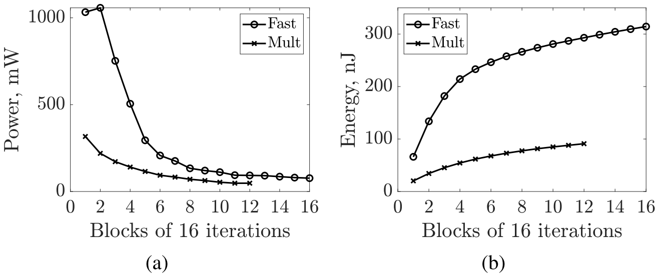


FIGURE 18. (a) Average power and (b) accumulated energy consumption over blocks of 16 iterations for Fast and Mult approaches.

although the maximum clock frequency is slightly higher for the Fast algorithm, 320 MHz, as compared to 300 MHz for the Mult algorithm.[3]

It may at first seem a bit unintuitive that the power consumption for the Fast approach using 128 iterations is higher than when using 256 iterations. However, it should be noted that these results are against clock frequency, so the approach with 128 iterations can perform twice as many detections. Also, as illustrated in the following, the switching activity and therefore power consumption is reduced when the results converge. Similarly, the Mult approach using 192 iterations can perform more detections compared to the Fast approach with 256 iterations, but fewer than the Fast approach with 128 iterations.

In the following, the 250 MHz clock frequency designs are used. Considering the area curves in Fig. 17, this should be close to the minimum area design, as synthesizing for a lower clock rate will not decrease the area significantly.

To provide further insights here, the average power consumption was analyzed in blocks of 16 iterations. The results are shown in Fig. 18(a). As expected, the power consumption is high in the first iterations and decreases as the results converge.

Similarly, one can compute the energy required for one detection instance. This is shown in Fig. 18(b), where the 16 iteration block averages are used as underlying data. Here, it is clear that the total energy required for one detection is significantly lower for the Mult algorithm compared to the Fast algorithm. Even if 128 iterations are used for the Fast algorithm, about 250 nJ is required, compared to about 100 nJ for the Mult algorithm. Still, the Fast algorithm has better detection performance when more iterations are performed.

Finally, it should be pointed out that the current approaches can perform about one million detections per second. This is most likely too much in the current settings. However, it is possible that the future brings scenarios with higher detection rate requirements. Hence, in the meanwhile one can reduce the energy consumption further by power supply voltage scaling, leading to even lower power and energy consumption.

E. COMPARISON WITH PREVIOUS RESULTS

As discussed earlier, to the best of the authors' knowledge, the only other existing previous implementations of mMTC activity detection algorithms are [17], [18]. However, this is for the other type of scenario, non-orthogonal sequences, instead of the pilot-hopping scheme considered here. Also, [17] considers the case where each user can send different sequences and therefore transmit a few bits of information (four bits). Hence, the comparison is not fully relevant as such. The reported detection performance is better for both [17] and [18]. However, this will apart from the algorithm also depend on, e.g., SNR, sequence length, number of users, user activity etc.

The implementations in [17] and [18] can perform about 11600 and 289 detections per second, respectively, so significantly less than the proposed architectures. The implementation occupies about 5.1 mm^2 and 1.14 mm^2 chip area for [17] and [18], respectively in the same standard cell process, memories not included. This should be compared to about 0.7 mm^2 and 0.4 mm^2 for the Fast and Mult algorithms, respectively. Hence, both the proposed architectures occupies significantly less chip area.

The energy consumption for one detection is about 0.13 mJ and 4.5 mJ for [17] and [18], respectively. The Fast algorithm presented here requires about 250 nJ and 300 nJ for 128 and 256 iterations, respectively. The Mult algorithm requires about 100 nJ. Hence, the proposed approaches are several orders of magnitude more energy efficient.

Another advantage of the approaches in [17], [18] is that the pilot sequences are easy to change. In the approaches proposed in the current work, they are hard coded into the structure of the matrix-vector multiplications with $\mathbf{A}$ and $\mathbf{A}^T$. While this provides a very efficient implementation, it may have other drawbacks. However, one may imagine that a system comes with prepared user modules. This will for example guarantee that all users have unique pilot-hopping sequences. One can also consider to synthesize the proposed architectures to FPGAs, in which case the sequences are readily configurable through a re-synthesis of the $\mathbf{A}$ and $\mathbf{A}^T$ matrix-vector multiplications.[4]

[3] Can be increased to 320 MHz by using a different programmable shift with a small area increase.

[4] The spectral radius is about the same for all matrices with the same parameters that have been tested.

IV. CONCLUSION

Two different architectures for activity detection in grant-free random access massive machine type communication based on pilot-hopping sequences have been proposed. They both solve a non-negative least squares problem, but with different algorithms. Both algorithms are iterative and with deterministic computations, enabling a high-speed implementation. It is shown that the fixed-point implementation of the fast projected gradient algorithm, Fast, provides detection results close to the converged floating-point realization. The multiplicative updates algorithm, Mult, is shown to converge faster for good cases, but slower for bad cases. It is partially implemented in the logarithmic domain to replace multiplications and divisions with additions and subtractions. Here, 192 iterations are selected, which gives a detection performance similar to 128 iterations of the Fast algorithm. However, the area and power/energy consumptions of the multiplicative updates algorithm is significantly lower. The energy consumed for one detection is about 100, 250, and 300 nJ for Mult with 192 iterations, Fast with 128 iterations, and Fast with 256 iterations, respectively, in a scenario with 1024 users and eight pilots, each with a length of 16.

The detection rate is more than one million detections per second, which is most likely too much for contemporary scenarios. However, future scenarios will most likely require an increased detection speed. It is also possible to reduce the clock frequency and power supply voltage to benefit from a lower energy consumption. Currently, the pilot hopping matrices are hard coded in the structure, requiring that the terminals must send the correct sequences. When implementing the same architecture on an FPGA, it is possible to change the pilot hopping matrices by reprogramming the FPGA. This is readily obtained as the architectures have code generators developed.

REFERENCES

[1] *IMT Vision—Framework and Overall Objectives of the Future Development of IMT for 2020 and Beyond*, ITU-R Standard M.2083-0, 2015.

[2] C. Bockelmann *et al.*, "Towards massive connectivity support for scalable mMTC communications in 5G networks," *IEEE Access*, vol. 6, pp. 28969–28992, 2018.

[3] K. Mikhaylov *et al.*, "Energy efficiency of multi-radio massive machine-type communication (MR-MMTC): Applications, challenges, and solutions," *IEEE Commun. Mag.*, vol. 57, no. 6, pp. 100–106, Jun. 2019.

[4] S. Dang, O. Amin, B. Shihada, and M.-S. Alouini, "What should 6G be?" *Nat. Electron.*, vol. 3, no. 1, pp. 20–29, 2020.

[5] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, May/Jun. 2020.

[6] X. Chen, D. W. K. Ng, W. Yu, E. G. Larsson, N. Al-Dhahir, and R. Schober, "Massive access for 5G and beyond," *IEEE J. Sel. Areas Commun.*, early access, Sep. 24, 2020, doi: 10.1109/JSAC.2020.3019724.

[7] E. de Carvalho, E. Björnson, J. H. Sørensen, E. G. Larsson, and P. Popovski, "Random pilot and data access in massive MIMO for machine-type communications," *IEEE Trans. Wireless Commun.*, vol. 16, no. 12, pp. 7703–7717, Dec. 2017.

[8] Z. Chen, F. Sohrabi, and W. Yu, "Sparse activity detection for massive connectivity," *IEEE Trans. Signal Process.*, vol. 66, no. 7, pp. 1890–1904, Apr. 2018.

[9] L. Liu and W. Yu, "Massive connectivity with massive MIMO—Part I: Device activity detection and channel estimation," *IEEE Trans. Signal Process.*, vol. 66, no. 11, pp. 2933–2946, Jun. 2018.

[10] L. Liu, E. G. Larsson, W. Yu, P. Popovski, C. Stefanovic, and E. de Carvalho, "Sparse signal processing for grant-free massive connectivity: A future paradigm for random access protocols in the Internet of Things," *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 88–99, Sep. 2018.

[11] K. Şenel and E. G. Larsson, "Grant-free massive MTC-enabled massive MIMO: A compressive sensing approach," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6164–6175, Dec. 2018.

[12] S. Haghighatshoar, P. Jung, and G. Caire, "A new scaling law for activity detection in massive MIMO systems," 2018. [Online]. Available: http://arxiv.org/abs/1803.02288.

[13] E. Becirovic, E. Björnson, and E. G. Larsson, "Detection of pilot-hopping sequences for grant-free random access in massive MIMO systems," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Brighton, U.K., May 2019, pp. 8380–8384.

[14] Z. Chen, F. Sohrabi, Y.-F. Liu, and W. Yu, "Covariance based joint activity and data detection for massive random access with massive MIMO," in *Proc. IEEE Int. Conf. Commun.*, Shanghai, China, 2019, pp. 1–6.

[15] X. Shao, X. Chen, and R. Jia, "Low-complexity design of massive device detection via Riemannian pursuit," in *Proc. IEEE Global Commun. Conf.*, Waikoloa, HI, USA, 2019, pp. 1–6.

[16] J. Ahn, B. Shim, and K. B. Lee, "EP-based joint active user detection and channel estimation for massive machine-type communications," *IEEE Trans. Commun.*, vol. 67, no. 7, pp. 5178–5189, Jul. 2019.

[17] M. Tran, O. Gustafsson, P. Källström, K. Şenel, and E. G. Larsson, "An architecture for grant-free massive MIMO MTC based on compressive sensing," in *Proc. Asilomar Conf. Signals Syst. Comput.*, Nov. 2019, pp. 901–905.

[18] M. Henriksson, O. Gustafsson, U. K. Ganesan, and E. G. Larsson, "An architecture for grant-free random access massive machine type communication using coordinate descent," in *Proc. Asilomar Conf. Signals Syst. Comput.*, Nov. 2020.

[19] N. M. Sarband, E. Becirovic, M. Krysander, E. G. Larsson, and O. Gustafsson, "Pilot-hopping sequence detection architecture for grant-free random access using massive MIMO," in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 1–5, 2020.

[20] T. L. Marzetta, E. G. Larsson, H. Yang, and H. Q. Ngo, *Fundamentals of Massive MIMO*. Cambridge, U.K.: Cambridge University Press, 2016.

[21] E. Björnson, J. Hoydis, and L. Sanguinetti, "Massive MIMO networks: Spectral energy, and hardware efficiency," *Found. Trends Signal Process.*, vol. 11, nos. 3–4, pp. 154–655, 2017.

[22] M. Slawski. (Mar. 2013). *Problem-Specific Performance Analysis of Non-Negative Least Squares Solvers With a Focus on Instances With Sparse Solutions*. [Online]. Available: https://sites.google.com/site/slawskimartin/nnlsalgorithms.pdf

[23] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*. Philadelphia, PA, USA, SIAM, 1995.

[24] R. A. Polyak, "Projected gradient method for non-negative least square," *Contemp. Math.*, vol. 636, pp. 167–179, Jan. 2015.

[25] M. E. Daube-Witherspoon and G. Muehllehner, "An iterative image space reconstruction algorthm suitable for volume ECT," *IEEE Trans. Med. Imag.*, vol. 5, no. 2, pp. 61–66, Jun. 1986.

[26] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 2001, pp. 556–562.

[27] F. Sha, Y. Lin, L. K. Saul, and D. D. Lee, "Multiplicative updates for nonnegative quadratic programming," *Neural Comput.*, vol. 19, no. 8, pp. 2004–2031, 2007.

[28] E. E. Swartzlander and A. G. Alexopoulos, "The sign/logarithm number system," *IEEE Trans. Comput.*, vol. C-24, no. 12, pp. 1238–1242, Dec. 1975.

[29] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Trans. Electron. Comput.*, vol. EC-11, no. 4, pp. 512–517, Aug. 1962.

NARGES MOHAMMADI SARBAND (Student Member, IEEE) received the B.Sc. degree in computer hardware engineering from the Sadjad University of Technology, Mashhad, Iran, in 2006, and the M.Sc. degree in computer architecture from the University of Isfahan, Isfahan, Iran, in 2014. She is currently pursuing the Ph.D. degree in electrical engineering with specialization in computer engineering with Linköping University, Sweden. She has also worked on optimizing peer-to-peer algorithms in computer networking. Her research focuses on efficient design and hardware implementation of communication algorithms for beyond 5G systems in both ASIC and FPGA.

EMA BECIROVIC (Graduate Student Member, IEEE) received the M.Sc. degree from Linköping University, Sweden, in 2018, where she is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, Division of Communication Systems. Her research is mainly focused on massive MIMO and massive machine-type communications.

MATTIAS KRYSANDER is an Associate Professor with the Department of Electrical Engineering, Linköping University, Sweden. His research interests include model-based and data-driven diagnosis and prognosis. To address the complexity and size of industrial systems (mainly vehicle systems), he has used structural representations of models and developed graph theoretical methods for assisting the design of diagnosis systems and fault isolation and sensor placement analysis.

ERIK G. LARSSON (Fellow, IEEE) received the Ph.D. degree from Uppsala University, Uppsala, Sweden, in 2002.

He is currently a Professor of Communication Systems with Linköping University, Linköping, Sweden. He was with the KTH Royal Institute of Technology, Stockholm, Sweden; George Washington University, Washington, DC, USA; the University of Florida, Gainesville, FL, USA; and Ericsson Research, Sweden. He has coauthored the books *Space-Time Block Coding for Wireless Communications* (Cambridge University Press, 2003) and *Fundamentals of Massive MIMO* (Cambridge University Press, 2016). He is the co-inventor of 19 issued U.S. patents. His main professional interests are within the areas of wireless communications and signal processing.

Prof. Larsson received the *IEEE Signal Processing Magazine* Best Column Award in 2012 and 2014, the IEEE ComSoc Stephen O. Rice Prize in Communications Theory in 2015, the IEEE ComSoc Leonard G. Abraham Prize in 2017, the IEEE ComSoc Best Tutorial Paper Award in 2018, and the IEEE ComSoc Fred W. Ellersick Prize in 2019. He served as the Chair of the IEEE Signal Processing Society SPCOM technical committee from 2015 to 2016, the Chair of the IEEE WIRELESS COMMUNICATIONS LETTERS steering committee from 2014 to 2015, the General and the Technical Chair of the Asilomar SSC conference in 2012 and 2015, the Technical Co-Chair of the IEEE Communication Theory Workshop in 2019, and the member of the IEEE Signal Processing Society Awards Board from 2017 to 2019. He was an Associate Editor for IEEE TRANSACTIONS ON COMMUNICATIONS from 2010 to 2014 and IEEE TRANSACTIONS ON SIGNAL PROCESSING from 2006 to 2010. He is currently an Editorial Board Member of the *IEEE Signal Processing Magazine* and the member of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS steering committee.

OSCAR GUSTAFSSON (Senior Member, IEEE) received the M.Sc., Ph.D., and Docent degrees from Linköping University, Linköping, Sweden, in 1998, 2003, and 2008, respectively. He is currently an Associate Professor and the Head of the Division of Computer Engineering, Department of Electrical Engineering, Linköping University. His research interests include design and implementation of DSP algorithms and arithmetic circuits in FPGA and ASIC. He has authored and coauthored over 180 papers in international journals and conferences on these topics. He is a member of the VLSI Systems and Applications and the Digital Signal Processing technical committees of the IEEE Circuits and Systems Society. He has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: EXPRESS BRIEFS and *Integration, the VLSI Journal*. Furthermore, he has served and serves in various positions for conferences, such as ISCAS, PATMOS, PrimeAsia, ARITH, Asilomar, NorCAS, ECCTD, and ICECS.