IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 9, 2018, accepted December 2, 2018, date of publication January 9, 2019, date of current version January 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2891597

Toward High-Performance Implementation of 5G SCMA Algorithms

ALIREZA GHAFFARI[1], MATHIEU LÉONARDON[2], ADRIEN CASSAGNE[2,3], CAMILLE LEROUX[2], AND YVON SAVARIA[1], (Fellow, IEEE)

[1]École Polytechnique de Montréal, Montreal, QC H3T 1J4, Canada
[2]CNRS IMS Laboratory, Bordeaux INP, University of Bordeaux, 33400 Bordeaux, France
[3]Inria, Bordeaux Institute of Technology, LaBRI/CNRS, 33405 Bordeaux, France

Corresponding author: Alireza Ghaffari (seyed-alireza.ghaffari@polymtl.ca)

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada, in part by the Huawei Canada, and in part by the Prompt Quebec.

ABSTRACT The recent evolution of mobile communication systems toward a 5G network is associated with the search for new types of non-orthogonal modulations such as sparse code multiple access (SCMA). Such modulations are proposed in response to demands for increasing the number of connected users. SCMA is a non-orthogonal multiple access technique that offers improved bit error rate performance and higher spectral efficiency than other comparable techniques, but these improvements come at the cost of complex decoders. There are many challenges in designing near-optimum high throughput SCMA decoders. This paper explores means to enhance the performance of SCMA decoders. To achieve this goal, various improvements to the MPA algorithms are proposed. They notably aim at adapting SCMA decoding to the single instruction multiple data paradigm. Approximate modeling of noise is performed to reduce the complexity of floating-point calculations. The effects of forwarding error corrections such as polar, turbo, and LDPC codes, as well as different ways of accessing memory and improving power efficiency of modified MPAs are investigated. The results show that the throughput of an SCMA decoder can be increased by 3.1 to 21 times when compared to the original MPA on different computing platforms using the suggested improvements.

INDEX TERMS 5G, BER, exponential estimations, intel advanced vector extensions (AVX), iterative multi-user detection, knights corner instruction (KNCI), log-MPA, maximum likelihood (ML), message passing algorithm (MPA), power efficiency, SCMA, single instruction multiple data (SIMD), streaming SIMD extension (SSE).

I. INTRODUCTION

Non-orthogonal Multiple Access (NOMA) mechanisms are investigated as means to improve the fifth-generation mobile communication systems (5G) [1] to realize massive connectivity and to reduce bit error rates. Sparse Code Multiple Access (SCMA) is a NOMA mechanism that offers better bit error rate performance and higher spectral efficiency, while the sparsity of the codebooks ensures lower complexity of decoding compared to other non-orthogonal modulations [2]. SCMA is a promising candidate for 5G communication systems since it provides up to 300% more connectivity by spreading information of each user's codebook over sets of shared OFDM frequency tones [3]. According to the NGMN white paper [4], 5G is seriously considered to fulfill more diverse scenarios compared to 4G. Applications can be broadband support in dense areas, low latency connectivity for Augmented Reality (AR) and reliable communication for intelligent industrial controls, Internet of Things (IoT) or Internet of Mission Critical Things (IoMCT). Unfortunately, massive connectivity and spectral efficiency of SCMA come at the cost of high complexity in the decoder, making the design of high throughput and low complexity decoders a challenge for systems exploiting SCMA [5].

Exploiting sparsity of the codebooks, Belief Propagation (BP) or Message Passing Algorithm (MPA) decoders were introduced to achieve near Maximum Likelihood performance with lower complexity [6]. Substantial research works were conducted on improving SCMA decoders to satisfy the uplink requirements of 5G. Indeed, MPA is populated with many exponential computations to calculate the extrinsic information and probabilities of the received signal. This is based on modeling the channel noise with a Gaussian

probability density function (PDF). A classical improvement to this bottleneck is the computation of extrinsic information in the logarithm domain, which led to develop the log-MPA decoder. In [7], fixed point and floating-point implementations of the MPA and log-MPA on FPGA are studied. The bit error rate performance and complexity of the MPA and log-MPA are compared and it is concluded that using log-MPA with 4 message passing iterations achieves a good tradeoff between performance and complexity. In [8], several complexity reduction techniques are proposed to increase the system throughput. These techniques are 1) SCMA codebook design with minimum number of projections, 2) clustered MPA (CMPA) which defines sub-graphs in MPA and runs MPA on them, and 3) selected exponential computations. In [9] an adaptive Gaussian approximation is used to unselect the edges of the graph with smaller modulus. In addition, mean and variance feedbacks are employed to compensate information loss caused by unselected edges. User's codebooks play an important role for fast convergence of the MPA or log-MPA. As investigated in [10]–[12], revisiting codebook design can help to reduce the number of iterations needed for MPA decoding of SCMA. In [13], an improved MPA is proposed which eliminates determined user codewords after certain number of iterations and continue the iterations for undetermined user's codewords. Similarly, in [14], a belief threshold is set to choose the most reliable edge probabilities and continue the iterations for the others. A Shuffled MPA (S-MPA) is introduced in [15]. S-MPA is based on shuffling the messages between function nodes and variable nodes. As a result, the convergence rate is accelerated. A Monte Carlo Markov Chain Method is proposed in [16] to decode SCMA signals and sphere decoding is also explored in [17] and [18] for SCMA receiver design.

The main difference between this work and previously cited works is that the present paper combines an analytic view of MPA complexity with hardware and resource aware programming, exploiting hardware features available in general purpose processors (GPPs). The SCMA decoding algorithms are revised in light of the needs of Cloud Radio Access Networks (C-RANs) and to take full advantage of key hardware features available in GPPs such as their SIMD engine. In the early 2000s, the performance of many processors improved significantly due to clock rate increases. This increase of performance needed very minimal if any programming effort, however the drawbacks of high clock rate was more power and energy consumption, overheating of processors, leakage currents and signal integrity problems. These disadvantages led designers to follow new paradigms such as thread-level and data-level parallelisms that provide good performance at lower clock speeds. Another challenge was data access efficiency in cache and RAM for performance critical algorithms. Higher performance also came from improved cache access efficiency of heterogeneous processors and parallel access to the L1 cache through vectorized instructions. Therefore, complicated and control heavy algorithms such as MPA have to be adapted for efficient execution on heterogeneous architectures and their exploitable parallelism must be expressed at every level of the code, whether in arithmetic or memory access instructions. Particularly, various Single Instruction Multiple Data (SIMD) extensions and thread-level parallelism are used to increase the throughput of MPA decoding on various platforms.

This paper reports on two contributions that can be useful for any variation of the aforementioned MPA. First, MPA and log-MPA have been adapted to use SIMD extensions leveraging the available data-level parallelism. The algorithms are revised to have aligned and contiguous access to memory, which is crucial to achieve high memory throughput. Various SIMD instruction set architectures (ISAs) such as Advanced Vector Extensions (AVX), Streaming SIMD Extension (SSE), Knights Corner Instruction (KNCI) and ARM NEON are used to enhance the performance of various parts of the algorithm. Multi-threaded programming technique and power efficiency are also studied in this paper.

Second, efforts have been made to reduce the high dynamic ranges and high storage burden that are induced by the numerous calculations of the exponential function embedded in MPA, which is one of its main points of congestion. To eliminate calculations of the exponentials in the MPA, this paper uses approximate modeling of noise. Indeed, a Gaussian Probability Density Function (PDF) is estimated with suboptimal, bell shaped, polynomial PDFs. Using polynomial PDFs enables a significant throughput improvement with a very small degradation on the bit error rate performance. In addition, this technique enables to use vectorized instructions for the calculation of the probabilities, as opposed to log-MPA. Details will be explained in the sequel. The impacts of turbo codes [19], polar codes [20] and LDPC codes [21] are investigated.

In this paper, symbols $\mathbb{B}$, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$ and $\mathbb{C}$ denote binary, natural, integer, real and complex numbers. Scalar, vector and matrix are presented as $x, \boldsymbol{x}, \boldsymbol{X}$ respectively. The n'th element of a vector denoted by $\boldsymbol{x}_n$ and $\boldsymbol{X}_{n,m}$ is the element of n'th row and m'th column of matrix $\boldsymbol{X}$. Notation diag(x) shows a diagonal matrix where its n'th diagonal element is $\boldsymbol{x_n}$. In addition, the transpose of a matrix is expressed as $\boldsymbol{X^T}$. The paper is organized as follows, Section II introduces the SCMA algorithm. Maximum Likelihood, MPA and log-MPA decoding methods are explained in this section as a background to this research. Section III elaborates on proposed improvements such as vectorizing the algorithm, exponential estimations, contiguous access to memory and other hardware oriented techniques. Section IV explores the bit error rate performance as well as the throughput, the latency, the power consumption, and the energy efficiency of the proposed MPA and log-MPA implementations. Some profiling metrics are given to better understand the results. Section V is dedicated to study

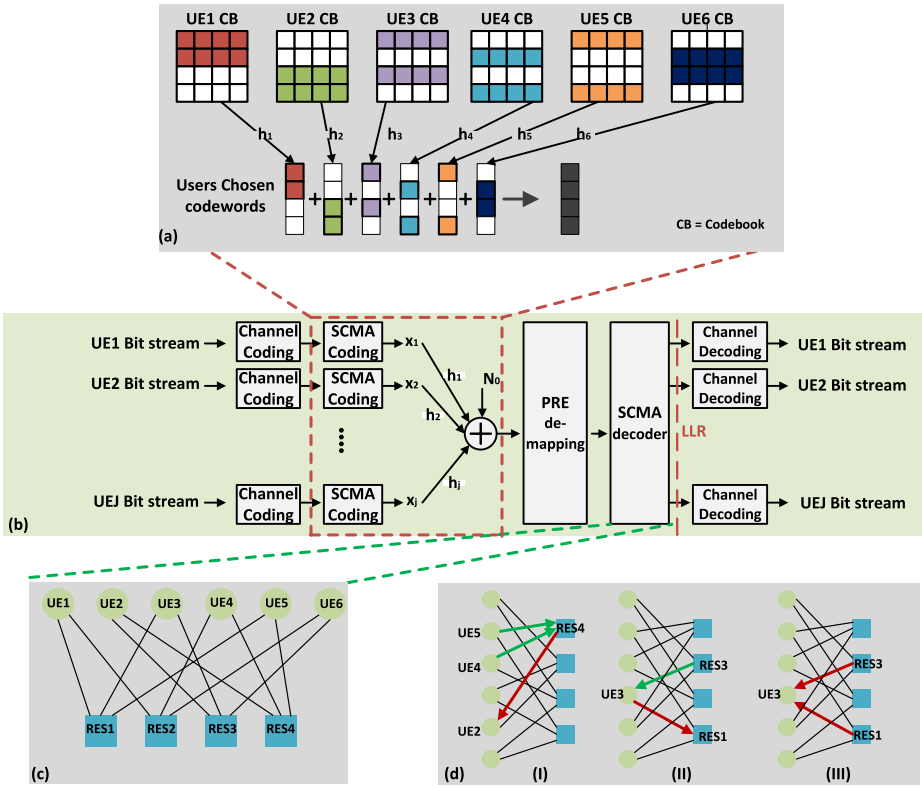


FIGURE 1. a) SCMA encoder with 6 users (layers) and 4 physical resources, b) SCMA uplink chain with channel coding, c) Factor graph representation of a decoder, d) Message Passing Algorithm based on Bayesian factor graph: (I) Resource to user message, (II) Guess swap at each user and user to resource message, (III) Final guess at each user.

the effects of suggested techniques on block error rate after channel coding. Finally, the main findings of this research are summarized in Section VI.

II. BACKGROUND

A. OVERVIEW OF THE SCMA SYSTEM MODEL

An SCMA encoder with J users (layers) and K physical resources is a function that maps a binary stream of data to K-dimensional complex constellations $f : \mathbb{B}^{log_2(M)} \rightarrow \mathbb{X}, x = f(\boldsymbol{b})$ where $\boldsymbol{X} \subset \mathbb{C}^k$. The K-dimensional complex codeword x is a sparse vector with $N < K$ non-zero entries. Each layer $j = 1, \ldots, J$ has its own codebook to generate the desired codeword according to the binary input stream. Fig. 1 shows SCMA uplink chain with $J = 6$, $K = 4$ and $N = 2$. SCMA codewords are spread over K physical resources, such as OFDMA tones. Fig. 1a shows that in the multiplexed scheme of SCMA, all chosen codewords of the J layers are added together after being multiplied by the channel coefficient $\boldsymbol{h}_j$. Then, the entire uplink chain is shown in Fig. 1b. The output of the SCMA encoder is affected by the white additive noise $\boldsymbol{n}$.

$$\boldsymbol{y} = \sum_{j=1}^{J} \text{diag}(\boldsymbol{h}_j)\boldsymbol{x}_j + \boldsymbol{n}, \tag{1}$$

where $\boldsymbol{x}_j = (x_1, \ldots, x_{Kj})^T$ and $\boldsymbol{h}_j = (h_1, \ldots, h_{Kj})^T$ are respectively codeword and channel coefficients of layer j.

B. SCMA DETECTION SCHEMES

1) MAXIMUM LIKELIHOOD

For an arbitrary codeword, the optimum decision, i.e. the one that minimizes the likelihood of transmission errors after decoding, is the one resulting from the Maximum Likelihood (ML) estimation, which can be described as:

$$\boldsymbol{x}\hat{_{ML}} = \underset{c \in \boldsymbol{X}}{arg\,min}\ ||\boldsymbol{y} - \boldsymbol{c}||^2, \tag{2}$$

given the received codeword. In (2), the soft outputs $\hat{\boldsymbol{x}}$ are also called Log-Likelihood Ratios (LLRs) that can be calculated with the following equation:

$$LLR_x = \ln\left(\frac{\sum_{C \in \mathcal{L}_x^0} \text{P}(\boldsymbol{y}|\boldsymbol{c})}{\sum_{C \in \mathcal{L}_x^1} \text{P}(\boldsymbol{y}|\boldsymbol{c})}\right), \tag{3}$$

where LLR_x is the log likelihood ratio of bit x obtained from codeword $\hat{\boldsymbol{x}}$. This codeword comes from $\mathcal{L}_x^1$ the set of codewords in which bit x is 1 and $\mathcal{L}_x^0$ the set of codewords in which bit x is 0. The probability function $\text{P}(\boldsymbol{y}|\boldsymbol{c})$ can be expressed as in (4) when a signal is transmitted over an additive white

Gaussian channel with σ^2 variance:

$$\mathrm{P}(\boldsymbol{y}|\boldsymbol{c}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{||\boldsymbol{y} - \boldsymbol{c}||^2}{2\sigma^2} \right). \tag{4}$$

Although the ML method provides the best guess for $\boldsymbol{\mathcal{C}_{ML}}$, performing the computation with this method requires unacceptable complexity in real applications. In the case of six users and codebooks matrices size 4 × 4 as in Fig. 1a, the calculation of the soft output for each bit in (4) needs 4096 exponential function computations, which is unacceptable. Nevertheless, in this article the result of this method is used to compare with practical methods to characterize the BER performance degradation of MPA and log-MPA.

2) MESSAGE PASSING ALGORITHM (MPA)

Fig. 1c shows a Bayesian factor graph representation of an MPA decoder with six users and four physical resources. Thanks to sparsity of the codebooks, exactly three users collide in each physical resource. There are four possible codewords for each of the three connected user's codebooks which gives 64 possible combined codewords in each physical resource. In the first step of the MPA, the 64 distances between each possible combined codewords and the actual received codeword are calculated.

$$d_{RES\beta}(\boldsymbol{m}, \boldsymbol{H}) = ||\boldsymbol{y}_\beta - \sum_{l \subset \zeta, m_u \in \{1,\ldots,K\}} \boldsymbol{h}_{l,m_u} \boldsymbol{x}_{l,m_u}||, \tag{5}$$

ζ is the set of users connected to resource β and the considered codeword is denoted as m. For instance, (5) can be re-written for resource 4 as:

$$\begin{aligned} &d_{RES4}(m_2, m_4, m_6, \boldsymbol{h}_2, \boldsymbol{h}_4, \boldsymbol{h}_5) \\ &\quad = ||\boldsymbol{y_4} - \big(\boldsymbol{h}_2\boldsymbol{x}_2(m_2) + (\boldsymbol{h}_4\boldsymbol{x}_4(m_4) + (\boldsymbol{h}_5\boldsymbol{x}_5(m_5))\big)||. \\ &\qquad\qquad m_{2,4,6}=1,2,3,4 \end{aligned} \tag{6}$$

In which m_2, m_4, m_5 indicate the different codewords for users 5, 4, and 2 in (6). Assuming perfect channel estimation and Gaussian noise, these Euclidean distances can be expressed as probabilities using (7):

$$\Psi(d_{RES\beta}) = \exp\left(- \frac{d^2_{RES\beta}}{2\sigma^2} \right). \tag{7}$$

After calculating the residual probability of each codeword with (7), iterative MPA starts exchanging beliefs (probabilities) on possible received codewords among the users and resources nodes of the factor-graph. According to Fig. 1d(I), a message from resources to users has been defined to contain extrinsic information of two other connected users. For instance, a message from resource 4 to user 2 containing the probability information of codeword i can be expressed as:

$$\begin{aligned} \mu_{RES4\to UE2}(i) = &\sum_{j=1}^{4}\sum_{k=1}^{4} \Psi\big(d_{RES4}(i,j,k,\boldsymbol{H})\big) \\ &\times \mu_{UE4\to RES4}(j) \times \mu_{UE5\to RES4}(k). \end{aligned} \tag{8}$$

As shown in Fig. 1d(II) there are only two resources connected to each user. A message from a user to a resource is a normalized guess swap at the user node:

$$\mu_{UE3\to RES1}(i) = \frac{\mu_{RES3\to UE3}(i)}{\sum_i \mu_{RES3\to UE3}(i)}, \tag{9}$$

message passing between users and resources (see (8) and (9)) will be repeated three to eight times to reach the desired decoding performance. The final belief at each user B (i) is the multiplication of all incoming messages as illustrated in Fig. 1d(III) and (10) for UE4 and codeword i. Finally, (11) is used to calculate soft outputs for bit b_x:

$$B_3(i) = \mu_{RES1\to UE3}(i) \times \mu_{RES3\to UE3}(i), \tag{10}$$

$$\begin{aligned} LLR_x &= \ln\left(\frac{\mathrm{P}(\boldsymbol{y}|b_x = 0)}{\mathrm{P}(\boldsymbol{y}|b_x = 1)} \right) \\ &= \ln\left(\frac{\sum_m B_m(i)_{\ when\ b_x=0}}{\sum_m B_m(i)_{\ when\ b_x=1}} \right). \end{aligned} \tag{11}$$

3) LOG-MAP

Since calculation of exponentials in (7) requires relatively high computational effort, changing the algorithm to log domain using the Jacobi formula (12) is a classical improvement of MPA:

$$\ln\left(\sum_{i-1}^{N} \exp(f_i) \right) \approx \max\{f_1, f_2, \ldots, f_N\} \tag{12}$$

using Jacobi formula, (8) can be reduced to:

$$\begin{aligned} \mu_{RES1\to UE5}(i) = &\max_{j,k=1,\ldots,4} \left(- \frac{d^2_{RES1}(i,j,k,\boldsymbol{H})}{2\sigma^2} \right) \\ &+ \mu_{UE2\to RES1}(j) + \mu_{UE3\to RES1}(k), \end{aligned} \tag{13}$$

due to elimination of exponential's high dynamic ranges, there is no need to normalize the guess swap and (9) will be:

$$\mu_{UE3\to RES1}(i) = \mu_{RES3\to UE3}(i). \tag{14}$$

The rest of the algorithm can be expressed as follows:

$$B_3(i) = \mu_{RES3\to UE3}(i) + \mu_{RES1\to UE3}(i), \tag{15}$$

$$LLR_x = \max_i(B_m(i)_{\ when\ b_x=0}) - \max_i(B_m(i)_{\ when\ b_x=1}). \tag{16}$$

III. PROPOSED IMPROVEMENTS

Besides methodical improvements of the MPA such as log-MPA, hardware oriented improvements are important to take full benefit of C-RAN servers capabilities. Since MPA and log-MPA are control heavy algorithms, mishandling of data can induce huge performance losses. This section explores how MPA can be reformulated: 1) to improve data locality in cache and to reduce cache misses and branch mispredictions 2) to reorder the data paths in order to help exploiting data-level parallelism at each step of the MPA and log-MPA algorithms and 3) to exploit approximated modeling of additive white Gaussian noise in order to eliminate exponential

calculations and to drastically reduce the number of instructions for SSE, NEON, AVX and KNCI ISAs.

A. FLATTENING MATRICES TO REDUCE CACHE MISSES AND BRANCH MISSES

Considering (6) and (7), there are 64 calculations of distances and probabilities for each resource (256 for all resources). Using a multidimensional array ($4 \times 4 \times 4$) should be avoided, because it typically causes bad data locality, which leads to an increased number of cache misses. These misses negatively affect the throughput, and this is significant, since this process must be repeated in the decoder for each received 12-bit block of data. Flattening a d-dimensional array to a vector using (17) is appropriate to prevent cache misses and improve the spatial locality of data. This is done with the help of an index defined as:

$$index = \sum_{i=1}^{d} \left(\prod_{j=i+1}^{d} N_j \right) n_i. \tag{17}$$

where N_j is the size of the j^{th} dimension of the array and n_i is the location of a target element in that dimension. Improving data locality with a stride of a single floating-point number in each element makes it easier for the processor to have aligned and contiguous accesses to the memory through SIMD ISA. Utilizing SIMD instructions helps to reduce the total number of mispredicted branches in the algorithm. Contiguous accesses to the L1 cache are performed by chunks of 128-bit, 256-bit or 512-bit. It reduces the number of iterations in the `for`-loops and consequently it reduces the number of branches. On the other hand, for a vector of sixty four 32-bit floating-point numbers, 64 iterations are needed in the scalar mode, while only 16, 8 or 4 iterations are required in the vectorized modes using respectively SSE (or NEON), AVX or KNCI ISAs.

B. ADAPTING THE ALGORITHMS TO IMPROVE DATA-LEVEL PARALLELISM

SSE, NEON, AVX and KNCI ISAs handle SIMD operations [22]. KNCI and AVX use 512-bit and 256-bit registers, while SSE and NEON use 128-bit registers. For instance, an AVX operation can process eight 32-bit floating-point numbers simultaneously. The AVX instructions also provide high-performance loads and stores to the cache memory due to data vectorization. Flattening matrices to vectors is a prerequisite to enable AVX contiguous accesses to the memory. Vectorized instructions such as AVX are accessible in C++ through intrinsic functions. An intrinsic is a function that directly maps to an assembly instructions (for some rare exceptions it can be more than one instruction). Nowadays, AVX units use sixteen 256-bit YMM registers and a 32-bit MXCSR control register to handle vectors of eight 32-bit or four 64-bit floating-point numbers. The AVX ISA allows to perform SSE instructions using the lower 128-bit lane of the YMM registers. For MPA, the SIMD instructions are used to 1) compute the complex norm ||.|| in (5) and (6), 2) calculate the exponentials in (7), 3) perform users to resources messaging and final guesses at each user.

1) SIMD COMPUTATION OF COMPLEX NORMS

Equations (5) and (6) use a complex norm function ||.||, it can be optimized by using SIMD instructions. There are two ways to perform this computation: Fig. 2a depicts how to implement the norm function using an Array of Structures (AoS) for complex numbers. In this method, the complex numbers are represented as two consecutive floating-point numbers. The implementation with AoS uses

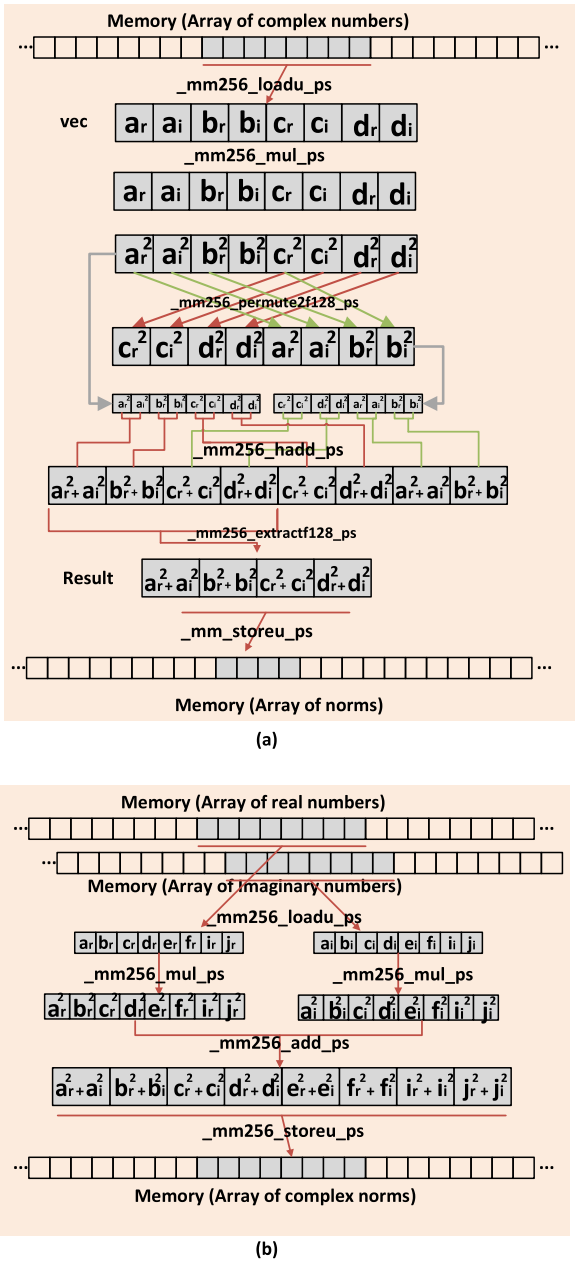


FIGURE 2. Complex norm AVX algorithm using a) Array of Structures (AoS), b) Structure of Arrays (SoA).

six intrinsic functions: one load (_mm256_loadu_ps), one store (_mm256_storeu_ps), one multiplication (_mm256_mul_ps), one permutation of the lanes (_mm256_permute2f128_ps), one horizontal addition (_mm256_hadd_ps) and one extraction of the highest lane of the AVX register (_mm256_extractf128_ps). Fig. 2b sketches the computation of the complex norm using a Structure of Array (SoA) data layout. This implementation also uses six intrinsic functions: two loads (_mm256_loadu_ps), one store (_mm256_storeu_ps), two standard multiplications (_mm256_mul_ps), one addition (_mm256_add_ps).

Our experiments demonstrated that these two methods have similar performances, however we used the Structure of Arrays (SoA) since it is 1) easier to port for the ISAs that lack from shuffle instructions and 2) trivial to extend for different register lengths.

2) SIMD COMPUTATION OF EXPONENTIAL

To speedup the computational time of the exponentials used in (7), the MIPP wrapper [23] has been used. MIPP proposes a vectorized implementation of the exponential based on a series expansion. Many intrinsic functions are encapsulated to compute the exponential. MIPP also allows to write portable intrinsic codes. A single SIMD code is written for multiple ISAs such as SSE, NEON, AVX, AVX512 and KNCI thanks to the meta-programming techniques.

The flattened complex and normalized numbers are calculated as shown in Fig. 2a and Fig. 2b to produce the preliminary values used to compute the probabilities. Fig. 3a illustrates the full process on a vector of eight floating-point numbers. First the values are loaded into the YMM registers, then they are multiplied by $-1/2\sigma^2$ and finally the exponential function is performed according to (7).

3) SIMD MESSAGE PASSING

Some remaining parts of the MPA can be vectorized too. Especially, the guess swaps and the computation of the final guesses at each user node can be vectorized using SSE instructions. Fig. 3b shows the computation of final guesses for user 4. There are four messages from a resource to a user containing the probabilities of four different codewords, which are the elements of the SSE vectors. According to Fig. 3b these vectors of probabilities are loaded into SSE, NEON or the lowest lane of the AVX registers.

C. ESTIMATED-MPA (E-MPA)

Computation of the exponentials in (7) is one of the most important bottlenecks of the MPA algorithm. It is possible to further accelerate the computation by using proper estimations. The exact exponential computation is not essential to produce a satisfying estimation in the MPA algorithms. Considering that (7) represents a Gaussian PDF, it can be replaced by sub-optimal bell-shaped polynomial distributions to model the noise. It will be shown in Section IV-B that using a polynomial estimation can increase the throughput while

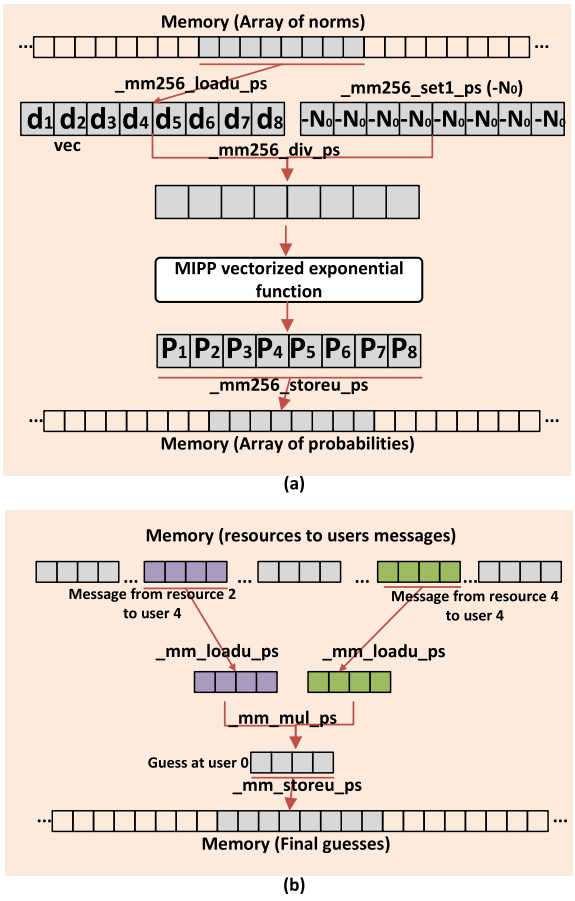


FIGURE 3. a) Vectorized Exponentials ($N_0 = 2\sigma^2$), b) Vectorized calculation of final guess at user 4.

leading to marginal bit error rate degradation after the MPA decoding. However, these estimated probabilities cause small degradations of the block error rate (BLER) performance after the channel decoding (cf. Section V). The proposed PDF must satisfy two conditions to be valid: 1) it must be positive and lower bounded at zero, 2) its integral over $(-\infty, \infty)$ must be equal to 1. The following function is suggested to estimate the exponentials:

$$\Psi'_{d_{RES\beta}} = \frac{2/\pi}{2\sigma^2 + 4d^4_{RES\beta}}. \tag{18}$$

The computation of Ψ' is faster than the original Ψ [24]. The probabilities produced using (7) and (18) are normalized according to (9). Furthermore, the numerator $2/\pi$ does not play an important role in MPA and can be uniformly eliminated from all calculations to reduce the computational effort. Thus,

$$\Psi'_{d_{RES\beta}} \approx \frac{1}{2\sigma^2 + 4d^4_{RES\beta}}, \tag{19}$$

can be used as a systematic replacement to the vectorized exponential MIPP function used in Fig. 3a. It reduces the overall number of instructions to three intrinsic functions:

two multiplications (`_mm256_mul_ps`) and one addition (`_mm256_add_ps`).

D. ACCURACY OF FLOATING-POINT COMPUTATIONS

The finite precision of floating-point calculations induces losses in the results. Thus, technical standards such as IEEE 754 define rounding rules, precision of calculations, exception handling and underflow behavior. However, the MPA delivers approaching bit error rate results with less precise floating-point models. For instance, in the GNU compiler, `-Ofast` is a high-level compiler option which includes fast math libraries to handle floating-point calculations (`-ffast-math`). The compiler uses various mathematical simplifications as explained in [25] and uses approximated libraries for the division and the square root functions. The compiler also forces the value to zero in the case of an underflow. Using `-Ofast` can improve the throughput of the MPA algorithm as will be shown in Section IV.

In this work, other well-known optimization techniques, such as loops unrolling, using references instead of pointers, avoiding type conversions, preferring prefixed operators, and functions inlining have been used to enhance the throughput of the various message passing algorithms.

IV. PERFORMANCE ANALYSIS

In this section, the effects of the various optimizations considered in Section III are investigated. A key concern is to ensure that the decoding error performance is not affected by the execution time improvements, particularly when approximations are involved. Energy efficiency and power consumption, throughput, memory access efficiency, hardware complexity analysis are all important aspects that must be considered.

A. EVALUATION OF ERROR PERFORMANCE

Fig. 4a shows the performance comparison of a maximum likelihood (ML) decoder, an MPA decoder performing 5 iterations and an estimated-MPA (E-MPA) decoder as explained in Section III also performing 5 iterations. There are very small differences in the bit error rate performance of the three decoders (less than 0.10 dB). Although both MPA and E-MPA show their optimum behavior with 5 iterations, the convergence behavior of the two methods are different as illustrated in Fig. 4b. E-MPA has a slower convergence rate for less than three iterations. This phenomenon is expected as the probability functions produced by bell-shaped polynomial PDF do not have the quality of probabilities produced by exponentials. However, the convergence behavior is almost identical for more than 4 iterations. The other optimizations like loops unrolling, fast math libraries and vectorization were not found to degrade the BER performance or the convergence rates.

B. CHARACTERIZING THROUGHPUT GAINS, ENERGY EFFICIENCY AND POWER CONSUMPTION

Energy efficiency is of interest in the design of C-RAN servers. It is determined by the rate of computation that can be delivered by a processor. Joint optimization of the

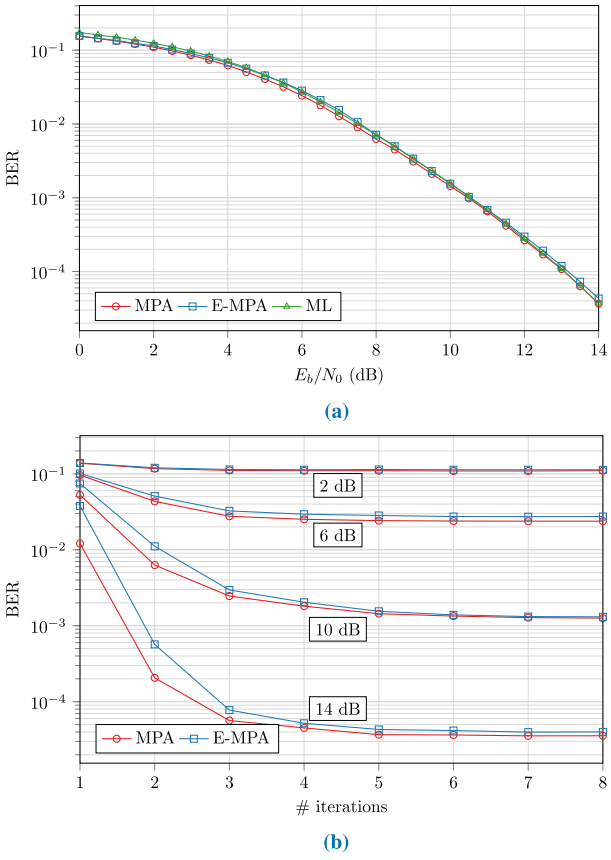


FIGURE 4. Performance of MPA compared with E-MPA. (a) BER performance comparison of ML, MPA and E-MPA for 5 iterations. (b) Convergence behavior of E-MPA and MPA.

throughput and energy consumption is a main goal of system designers. Energy optimization can reduce the cost of cloud services significantly while it can contribute to decrease the emission of greenhouse gases. Power utilization is also important because improved performance per Watt is useful to limit power demands. This section explores the power, energy efficiency and throughput of the various message passing algorithms suggested in this work. Tests have been conducted on three platforms running the Ubuntu Linux operating system. The three systems are : 1) an Intel™ Core-i7 6700HQ processor with AVX instructions (256-bit SIMD) and four physical cores using 2-way Simultaneous Multi-Threading (SMT or Intel Hyper-Threading™ technology) running at nominal frequency of 2.6 GHz, 2) an ARM™ Cortex-A57 with NEON instructions (128-bit SIMD) and four cores (no SMT) running at 2.0 GHz and 3) an Intel™ Xeon-Phi Knight-Corner 7120P with KNCI instructions (512-bit SIMD) and 61 cores using 4-way SMT and running at 1.2 GHz.

Table 1 shows the comparison of throughput, latency, power consumption and energy of different decoding algorithms that are executed on the three platforms to decode 768 Million bits. The average power and energy consumption measured on the Core-i7 processor were obtained with the turbostat software [26] which exploits the Intel

TABLE 1. Throughput, latency, power and energy characteristics.

	Algorithm	Throughput per Core (Mbps)	Throughput per Socket (Mbps)	Latency per Core (ns)	Power (W)	Energy per Bit (μJ)
Intel™ Core-i7 6700HQ	E-MPA+AVX (`-Ofast`)	17.46	75.46	57.2	40.02	0.66
	MPA+AVX (`-Ofast`)	15.06	67.83	66.4	40.53	0.73
	Log-MPA (`-Ofast`)	2.51	10.31	398.4	35.11	3.53
	Log-MPA (`-O3`)	1.11	6.37	900.9	33.11	6.02
	MPA (`-Ofast`)	3.58	14.85	279.3	33.01	2.49
	MPA (`-O3`)	0.55	3.51	1818.1	35.00	10.25
ARM™ Cortex-A57	E-MPA+NEON (`-Ofast`)	3.79	15.30	263.8	7.93	0.52
	MPA+NEON (`-Ofast`)	2.09	8.40	478.4	7.56	0.90
	Log-MPA (`-Ofast`)	1.20	4.70	833.7	6.99	1.46
	Log-MPA (`-O3`)	0.75	3.01	1333.3	6.99	2.33
	MPA (`-Ofast`)	1.03	4.07	970.8	7.18	1.76
	MPA (`-O3`)	0.41	1.60	2439.0	6.99	4.21
Xeon-Phi 7120P	E-MPA+KNCI (`-O2`)	0.90	114.60	1111.1	198.00	1.73
	MPA+KNCI (`-O2`)	0.67	82.32	1492.5	198.00	2.41
	Log-MPA (`-O2`)	0.36	53.38	2777.7	184.00	3.45
	MPA (`-O2`)	0.28	36.09	3571.4	196.00	5.44

performance counters in Machine Specific Registers (MSRs) to monitor CPU and RAM utilizations. However, in the case of ARM and Xeon Phi platforms, external current sensors were used to measure the energy and power consumptions.

1) INTEL™ CORE-i7 6700HQ

The baseline implementation of MPA with level 3 (`-O3`) optimization of the GNU compiler reaches 3.51 Mbps utilizing all four physical cores of the processor (SMT on). Log-MPA improves the performance to 6.37 Mbps benefiting from elimination of the exponential calculations, still in `-O3`. However, using the fast math libraries (`-Ofast`) and the loop optimizations from Section III-D increases the throughput to 14.85 Mbps for MPA and to 10.31 Mbps for log-MPA. It is important to observe that MPA outperforms the log-MPA with the fast math libraries and more aggressive optimizations, without compromising on the bit error rate performance. This is because log-MPA induces inefficient data accesses due to the messages passed from resources to users. This phenomenon will be investigated further in Section IV. Using the AVX and SSE SIMD ISAs reduces the branch mispredictions and the cache misses (cf. Section III-A). Consequently, the throughput is increased to 67.83 Mbps in MPA and to 75.46 Mbps for the E-MPA where the Ψ' estimated exponentials from (19) are performed. These results confirm significant throughput gains for the proposed implementations, while the energy consumption is reduced. Utilizing AVX increases the average power consumption of MPA and log-MPA from 35 to 40 Watts but throughput and latency are improved by much larger factors. It means that the overall energy consumption have been decreased with AVX.

2) ARM™ CORTEX-A57

On this platform [27], the throughput difference caused by the fast math libraries of the GNU compiler is still visible for MPA and log-MPA algorithms. With level three optimization (`-O3`), MPA and log-MPA run at 1.60 Mbps and 3.01 Mbps respectively. When using fast math libraries (`-Ofast`) the throughputs increased to 4.07 and 4.70 Mbps. It should be noted that the four physical cores of the ARM platform were utilized for those tests. Power consumption and energy used per decoded bit is lower on the ARM platform than on the Intel processors. The low power consumption of the ARM platform notably comes at the cost of less powerful floating-point arithmetic units (cf. MPA+NEON and E-MPA+NEON in Table 1). Eliminating the exponential computations almost doubled the performance in E-MPA (15.30 Mbps) as compared to MPA+NEON (8.40 Mbps), which shows the limits of low power processors when calculating many exponentials. Nevertheless, by using E-MPA, the ARM low power processors can be a good candidate for implementation of SCMA decoders on C-RAN servers as it allows significant energy savings.

3) INTEL™ XEON-PHI 7120P

The Xeon-Phi Knights Corner [28] benefits from the ability to execute four hardware threads per core, while having 61 cores and 512-bit SIMD registers. In this case, 244 threads can be run to handle the MPA decoding task. Despite these benefits, the Xeon-Phi Knight Corners suffers from two main disadvantages: 1) the KNC instructions diversity is reduced compared to AVX or AVX-512 ISAs and 2) the cores frequency is relatively low in order to keep reasonable power consumption and limits the heat dissipation. As an example of missing instruction, the KNCI ISA does not offer coalesced division (`_mm512_div_ps`) for floating-point numbers. Beside those limitations, the E-MPA+KNCI exhibits the highest throughput among the three mentioned platforms (up to 114.60 Mbps). However, it consumes almost three times more energy per bit compared to the ARM-based implementations. The MPA decoding algorithm exhibits its best performance on this platform when cross compiled using `-O2 -mmic` flags by an Intel icpc compiler. Using fast math options such as `-no-prec-div -no-prec-sqrt -fp-speculation=fast -fp-model-fast=2` do not change the results significantly with the Intel compiler.

Fig. 5 focuses on the energy consumed per decoded bit (also mentioned in Table 1). In summary, the SIMD algorithms have a higher energy efficiency per decoded bit. The processor resources are well stressed and the power does not increase too much. Among the obtained results, the Xeon-Phi obtains the best throughput while the Cortex-A57 has the lowest energy consumption. In the case where the number of users in the cloud is increased, the results presented in

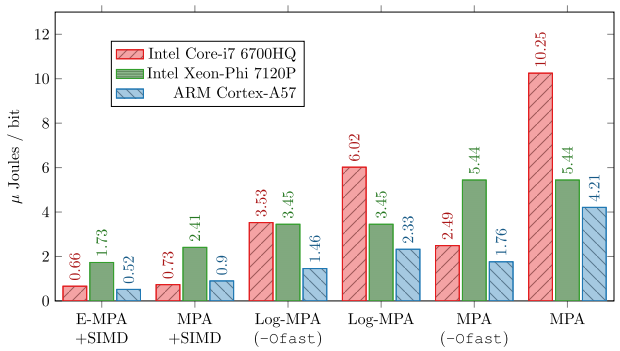


FIGURE 5. Graphical comparison of the energy consumed per decoded bit for three different platforms.

this section are scalable up to the number of processing units dedicated to them.

C. MEMORY (CACHE) ACCESS EFFICIENCY

Apart from SIMD operations and parallelization, cache access efficiency plays an important role in the high-performance implementation of algorithms on GPP. Table 2 shows the performance characterization of different MPA algorithms on the Core-i7 6700HQ processor for decoding 768 Million bits. As reported in Table 2, contiguous accesses to the memory using AVX instructions reduces the total number of branches and references to the cache. Reducing the number of branches and references to the cache increases the throughput of the algorithm.

TABLE 2. Cache performance characterization.

Algorithm	# of Branches (Million)	# of Branch Misses (Million)	# of Cache Ref. (Million)	# of Cache Misses (Million)	Instruction per Cycle
E-MPA+AVX	12267	422	275	70.83	1.23
MPA+AVX	12845	401	244	70.32	1.19
Log-MPA (`-Ofast`)	148867	17584	484	73.02	0.67
Log-MPA (`-O3`)	359967	18039	635	77.75	0.69
MPA (`-Ofast`)	126578	7093	397	72.58	1.12
MPA (`-O3`)	527075	9454	833	79.73	0.57

According to Table 2, MPA+AVX shows almost ten times fewer branches (12845 Million) versus MPA `-Ofast` (126578 Million) and consequently it offers better performance. For MPA+AVX, 401 Million branches have been mispredicted by the processors, compared to 7093 Millions for MPA. For cache misses MPA+AVX produced two Millions fewer cache misses when compared to MPA and the total number of cache references are also significantly (122 Millions) less than with MPA. The total number of cache misses for various algorithms in Table 2 are between 70 to 79 Millions, while the total number of branch mispredictions varies between 422 Millions to 6454 Millions. This high dynamic range of branch mispredictions shows that reducing the total number of branches and branch mispredictions have more impact on increasing throughput of the MPA algorithm in comparison to reducing cache misses. This phenomenon also shows that using optimization methods such as log-MPA which produces large number of branches due to the max(.) function is not ideal for multi-processor servers in C-RAN. These reported significant improvements have been brought by SIMD instructions. Improving data locality, contiguous access to memory and parallelizing loops are the main reasons that made SIMD algorithms exhibit better performance when it comes to cache interface.

Table 2 also reports the number of Instructions per Cycle (IPC) of each implementation. It is obvious that the number of IPC was reduced in MPA `-O3` and log-MPA due to poorer memory access efficiency. This reduces the throughput of those algorithms. On the other hand, without using contiguous access to memory, the processor spends more time for scalar load and stores. This can cause a bottleneck in interfacing memory while other resources of the processor are waiting to receive data and consequently it decreases the IPC. By contrast, in the case of contiguous access to memory (or cache) the processor can fetch sufficient data all at once to support sustained processing thus reducing the memory bottleneck and improving internal processing as reflected by better IPC indices.

D. PROFILING AND HARDWARE COMPLEXITY

Previous sections explored how processor parallel resources, efficient and contiguous memory access, and compiler optimizations play an important role in getting efficient implementation of the SCMA algorithms. In [6], [7], [13], and [15], computational complexity, measured as operation counts, was used to represent the complexity of the MPA. Operation counts can be misleading metrics when characterizing algorithmic complexity of algorithms executing on general purpose processors. Indeed, it misses significant factors such as cache misses, memory efficiency and precision of floating-point calculations. In this section, the time complexity of the various forms of SCMA decoders are investigated using the Intel Vtune Studio™ profiler [22].

Fig. 6 reports profiling results obtained with different SCMA decoders variations when applied to the decoding of 768 Million bits. Results were organized to show the existence of five bottlenecks i.e. logarithms in (11), exponentials in (7), complex norm and complex subtraction in (5) and messages passed from resources to users in (8).

Observing MPA and MPA (`-Ofast`) reveals the overhead of exponentials and complex norms in the algorithm. For example, the decoder spent more than 62 percent of its time (32.35 seconds) to calculate exponentials and norms in MPA (`-Ofast`). This led us to explore SIMD calculation of these two steps. Comparing E-MPA+SIMD and MPA+SIMD implementations to others such as MPA (`-O3` or `-Ofast`) shows a clear gain in throughput for calculation of the exponentials and norms. In more details, E-MPA+SIMD spends

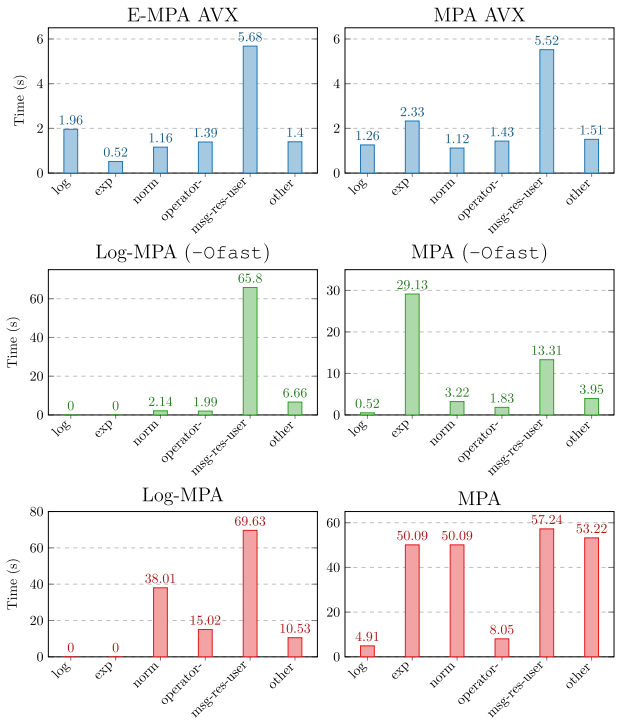


FIGURE 6. Profiling results of different MPA algorithms using Intel Vtune Profiler™ on Core-i7 6700HQ platform for decoding 768 Million bits.

1.68 seconds computing exponentials and norms which is more than 19 times faster than the initial computation of norms and exponentials in MPA (`-Ofast`). On the other hand, exponentials and norms computations are performing as fast as complex subtract. This profiling results show the efficiency of the proposed SIMD implementation methods. By contrast, log-MPA has not shown good performance using fast math library when compared to MPA. Inefficient memory access, cache misses and high number of branches are among the reasons that made log-MPA exhibits lower throughput than expected. Those phenomena are induced by comparison operations embedded in the max(.) function in (13). Nevertheless, without using fast math libraries, log-MPA still offers performance gains over MPA.

V. CHANNEL CODING

A. COMPLETE SIMULATION CHAIN

In the previous sections of this article, algorithmic improvements and implementation techniques have been proposed. These optimizations lead to drastic reductions of the processing time and to an increase of the processing power efficiency. This is done with approximately no degradation of the BER performance after SCMA decoding. Nevertheless, in a full communication chain, multiple access algorithms are closely linked to the Forward Error Correction (FEC) modules. Indeed, the input of the FEC decoder consists in the outputs of the SCMA decoder.

In order to claim that the proposed improvements do not degrade the overall error performance, it is necessary to embed the SCMA encoder and decoder in a full communication chain. To this purpose, we used the AFF3CT[1] software which is an ideal tool that provides the necessary simulation models and allows performing the desired verifications.

AFF3CT is Open-source and specifically designed to offer an efficient environment to the communication systems designers. Monte-Carlo simulations can be run to measure various metrics such as the BER and BLER performance, or the throughputs and latencies of each module, e.g. FEC encoders and decoders, modulation and demodulation blocks, or different channel models.

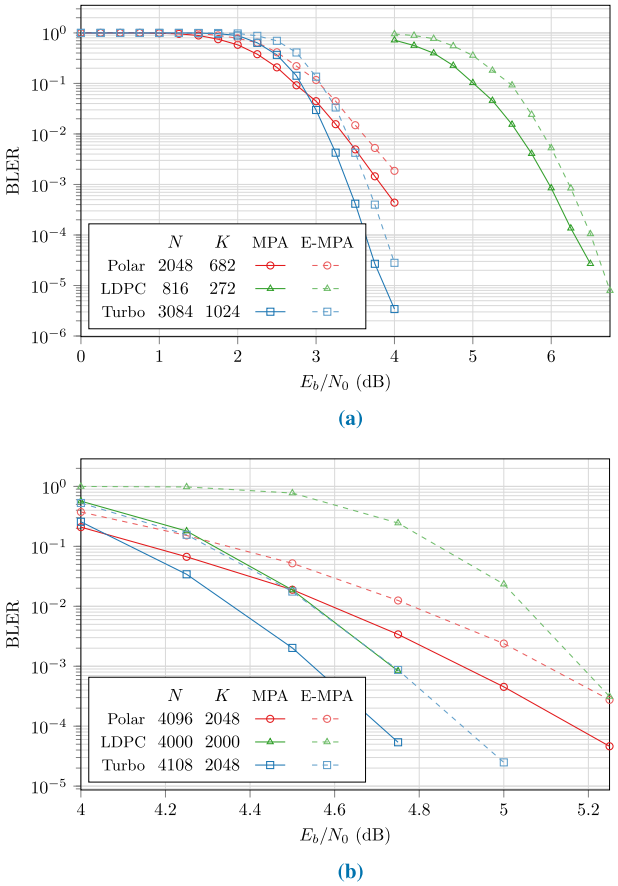


FIGURE 7. BLER evaluation of SCMA MPA and E-MPA decoders combined with LDPC, polar and turbo codes. (a) Code rate $R = 1 = 3$. (b) Code rate $R = 1 = 2$.

According to the latest 3GPP report [29], in the 5G standard, the two selected code families are the LDPC and polar codes. Being implemented in the AFF3CT software, it is possible to test our SCMA decoders in a complete communication chain, in conjunction with state-of-the art LDPC, polar and even turbo decoders that were used in the LTE standard [30]. Fig. 7 shows the BLER performances of MPA and E-MPA decoders when combined with different channel codes. For a matter of reproducibility, the full parameters of the FEC used are reported in the next section. This research

[1] AFF3CT is an Open-source software (MIT license) for fast forward error correction simulations, see `http://aff3ct.github.io`

does not claim any novelty in channel coding, however, we found crucial to validate our proposed SCMA optimizations in a sufficiently complete communication chain.

B. CHANNEL CODING CONFIGURATIONS

1) TURBO CODES

In a first validation, the turbo code from the LTE standard is used. In the decoder, 6 iterations are done. The two sub-decoders implement the max-log Maximum A Posteriori algorithm (max-log-MAP) [31] with a 0.75 scaling factor [32]. In Fig. 7a, the rate is $R \approx 1/3$, no puncturing is performed, the number of information bits K is 1024 and the codeword length N is 3084. In Fig. 7b, $R \approx 1/2$ with the puncturing of half of the parity bits, $K = 2048$, and $N = 4108$.

2) LDPC CODES

In a second set of validations, the LDPC codes used in this paper are based on MacKay matrices that have been taken from [33]. In Fig. 7a, the matrix used is ($K = 272, N = 816$), and in Fig. 7b the matrix is ($K = 2000, N = 4000$). In both figures, the decoder used is a Belief Propagation (BP) decoder with an Horizontal Layered scheduling [34]. For the update rules, the Sum-Product Algorithm (SPA) has been used [35]. The number of iterations is 100.

3) POLAR CODES

In the final validation, polar codes are built by suitably selecting the frozen bits. We used the Gaussian Approximation (GA) technique of [36]. The input SNR for the code construction with the GA is 1 dB, which apparently is very low considering that the SNR are 4 to 5 dB in the convergence zone. This is motivated by the fact that the GA algorithm is designed to work with the BPSK modulation. Using SCMA completely modifies the histogram of the LLR values for a given SNR. Therefore, a shift on the input SNR of the GA algorithm must be applied in order to efficiently select the frozen bits. If this shift is not applied, the decoding performances of the polar code degrades drastically. The number of information bits and the codeword length are ($K = 682$, $N = 2048$) in Fig. 7a and ($K = 2048, N = 4096$) in Fig. 7b. The decoder is a Successive Cancellation List (SCL) decoder with $L = 32$ and a 32-bit GZIP CRC that was proposed in [37].

C. EFFECTS OF E-MPA ON ERROR CORRECTION

In Fig. 7, the number of iterations of the SCMA demodulator is 5. The objective of simulating multiple channel codes is not to compare them with each other. A fair comparison of the different channel codes would indeed impose using the same code lengths and more importantly their computational complexity should be compared, which is not the case here. Our goal here is to study the impact of using E-MPA on the BER and FER performances when the channel codes are included in the communication chain. For each channel code, two curves are plotted: one for the E-MPA and the other for the MPA. Only 0.2 to 0.4 dB separate the two versions of the algorithm for all the considered channel codes. These results show the extent to which uncertainty of estimations affects channel coding. The decoding speed improvement brought by the E-MPA algorithm has a cost in terms of decoding performance. This trade-off should be considered in order to meet the system constraints.

VI. CONCLUSIONS

In this paper, in consideration of the potential of Cloud-RAN that would support 5G communication, we focused on improving the efficiency of 5G SCMA receivers on the type of multiprocessors that can be found in such servers. We provided test results using different platforms such as ARM Cortex, Xeon-Phi and Core-i7. The benefits of using SIMD and various algorithmic simplifications have been studied and test results were presented. Among the platforms, the ARM Cortex-A57 was shown to offer the lowest energy consumption per decoded bit, while many-core platforms such as Xeon-Phi Knight's Corner 7120P had the best throughput. In addition, an estimation of conditional probabilities using polynomial distributions instead of Gaussian distribution was proposed to increase throughput. This estimation has shown to offer throughput improvements of 15 to 90 percent depending on the platform used, while it causes a very small degradation of BLER after channel decoding. To support this claim, the error performance of telecommunication chains combining MPA and E-MPA with channel coding with LDPC, polar codes and turbo codes with code rates $R = 1/3$ and $R = 1/2$ were tested.

REFERENCES

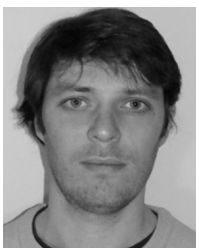
[1] S. M. R. Islam, N. Avazov, O. A. Dobre, and K.-S. Kwak, "Power-domain non-orthogonal multiple access (NOMA) in 5G systems: Potentials and challenges," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 721–742, 2nd Quart., 2017.
[2] H. Nikopour and H. Baligh, "Sparse code multiple access," in *Proc. IEEE Int. Symp. Pers. Indoor Mobile Radio Commun.*, London, U.K., Sep. 2013, pp. 332–336.
[3] 5G Algorithm Innovation Competition. (2015). *Altera Innovate Asia FPGA Design Contest*. [Online]. Available: http://www.innovateasia.com/5g/en/gp2.html
[4] NGMN Alliance, "5G white paper," Next Gener. Mobile Netw., Frankfurt, Germany, White Paper, 2015, pp. 1–125.
[5] L. Lei, C. Yan, G. Wenting, Y. Huilian, W. Yiqun, and X. Shuangshuang, "Prototype for 5G new air interface technology SCMA and performance evaluation," *China Commun.*, vol. 12, no. 9, pp. 38–48, Sep. 2015.
[6] S. Zhang, X. Xu, L. Lu, Y. Wu, G. He, and Y. Chen, "Sparse code multiple access: An energy efficient uplink approach for 5G wireless systems," in *Proc. IEEE Global Commun. Conf.*, Dec. 2014, pp. 4782–4787.
[7] J. Liu, G. Wu, S. Li, and O. Tirkkonen, "On fixed-point implementation of Log-MPA for SCMA signals," *IEEE Wireless Commun. Lett.*, vol. 5, no. 3, pp. 324–327, Jun. 2016.
[8] A. Bayesteh, H. Nikopour, M. Taherzadeh, H. Baligh, and J. Ma, "Low complexity techniques for SCMA detection," in *Proc. IEEE Globecom Workshops*, San Diego, CA, USA, Dec. 2015, pp. 1–6.
[9] Y. Du, B. Dong, Z. Chen, J. Fang, P. Gao, and Z. Liu, "Low-complexity detector in sparse code multiple access systems," *IEEE Commun. Lett.*, vol. 20, no. 9, pp. 1812–1815, Sep. 2016.
[10] M. Taherzadeh, H. Nikopour, A. Bayesteh, and H. Baligh, "SCMA codebook design," in *Proc. IEEE Veh. Technol. Conf.*, Las Vegas, NV, USA, Sep. 2014, pp. 1–5.

[11] J. Peng, W. Chen, B. Bai, X. Guo, and C. Sun, "Joint optimization of constellation with mapping matrix for SCMA codebook design," *IEEE Signal Process. Lett.*, vol. 24, no. 3, pp. 264–268, Mar. 2017.
[12] C. Yan, G. Kang, and N. Zhang, "A dimension distance-based SCMA codebook design," *IEEE Access*, vol. 5, pp. 5471–5479, 2017.
[13] M. Jia, L. Wang, Q. Guo, X. Gu, and W. Xiang, "A low complexity detection algorithm for fixed up-link SCMA system in mission critical scenario," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3289–3297, Oct. 2018.
[14] L. Yang, Y. Liu, and Y. Siu, "Low complexity message passing algorithm for SCMA system," *IEEE Commun. Lett.*, vol. 20, no. 12, pp. 2466–2469, Dec. 2016.
[15] Y. Du, B. H. Dong, Z. Chen, J. Fang, and L. Yang, "Shuffled multiuser detection schemes for uplink sparse code multiple access systems," *IEEE Commun. Lett.*, vol. 20, no. 6, pp. 1231–1234, Jun. 2016.
[16] J. Chen, Z. Zhang, S. He, J. Hu, and G. E. Sobelman, "Sparse code multiple access decoding based on a Monte Carlo Markov chain method," *IEEE Signal Process. Lett.*, vol. 23, no. 5, pp. 639–643, May 2016.
[17] L. Yang, X. Ma, and Y. Siu, "Low complexity MPA detector based on sphere decoding for SCMA," *IEEE Commun. Lett.*, vol. 21, no. 8, pp. 1855–1858, Aug. 2017.
[18] F. Wei and W. Chen, "Low complexity iterative receiver design for sparse code multiple access," *IEEE Trans. Commun.*, vol. 65, no. 2, pp. 621–634, Feb. 2017.
[19] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 2, Geneva, Switzerland, May 1993, pp. 1064–1070.
[20] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.
[21] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.
[22] Intel. (2018). *Intel C++ Compiler 18.0 Developer Guide and Reference*. [Online]. Available: https://software.intel.com/en-us/cpp-compiler-18.0-developer-guide-and-reference-fxsave64
[23] A. Cassagne, O. Aumage, D. Barthou, C. Leroux, and C. Jégo, "MIPP: A portable C++ SIMD wrapper and its use for error correction coding in 5G standard," in *Proc. Workshop Program. Models SIMD/Vector Process. (WPMVP)*, Vösendorf, Austria, Feb. 2018, pp. 1–8.
[24] A. Ghaffari, M. Léonardon, Y. Savaria, C. Jégo, and C. Leroux, "Improving performance of SCMA MPA decoders using estimation of conditional probabilities," in *Proc. 15th IEEE Int. New Circuits Syst. Conf. (NEWCAS)*, Jun. 2017, pp. 21–24.
[25] GCC. (2018). *Semantics of Floating Point Math in GCC*. [Online]. Available: https://gcc.gnu.org/wiki/FloatingPointMath
[26] L. Torvalds. (2018). *Turbostat*. [Online]. Available: https://github.com/torvalds/linux/tree/master/tools/power/x86/turbostat
[27] NVIDIA. (2018). *Jetson TX1*. [Online]. Available: https://www.nvidia.com/fr-fr/autonomous-machines/embedded-systems-dev-kits-modules/
[28] G. Chrysos, "Intel Xeon Phi coprocessor (codename Knights Corner)," in *Proc. IEEE Hot Chips 24 Symp. (HCS)*, Cupertino, CA, USA, Sep. 2012, pp. 1–31.
[29] *Multiplexing and Channel Coding (Release 15)*, document TS 38.212, 3GPP, Sep. 2017.
[30] *Multiplexing and Channel Coding (Release 11)*, document TS 136.212, 3GPP, Feb. 2013.
[31] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 2, Jun. 1995, pp. 1009–1013.
[32] J. Vogt and A. Finger, "Improving the max-log-MAP turbo decoder," *Electron. Lett.*, vol. 36, no. 23, pp. 1937–1939, Nov. 2000.
[33] D. J. MacKay. (2018). *Encyclopedia of Sparse Graph Codes*. [Online]. Available: http://www.inference.org.uk/mackay/codes/data.html
[34] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam, "High throughput low-density parity-check decoder architectures," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, vol. 5, Nov. 2001, pp. 3019–3024.
[35] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
[36] P. Trifonov, "Efficient design and decoding of polar codes," *IEEE Trans. Commun.*, vol. 60, no. 11, pp. 3221–3227, Nov. 2012.
[37] M. Léonardon, A. Cassagne, C. Leroux, C. Jégo, L. Hamelin, and Y. Savaria, "Fast and flexible software polar list decoders," *CoRR*, vol. abs/1710.08314, pp. 1–11, Oct. 2017. [Online]. Available: http://arxiv.org/abs/1710.08314

ALIREZA GHAFFARI received the B.Sc. degree from the Amirkabir University of Technology (Tehran Polytechnic), Iran, and the M.Sc. degree from Laval University Canada. He has wide range of industrial and research experience in firmware design, FPGA, hardware acceleration, and the IoT. Recently, he is more focused on hardware/software acceleration in heterogeneous platforms and clouds. He is currently an Enthusiast in electrical engineering and a Research Associate with the Electrical Engineering Department, École Polytechnique de Montréal.

MATHIEU LÉONARDON received the degree in engineering from Bordeaux INP, Bordeaux, France, in 2015. He is currently pursuing the Ph.D. degree with the École Polytechnique de Montréal and the University of Bordeaux under a co-directorship between both institutions. His research interests include the design of efficient and flexible implementations, both hardware and software, for decoding error-correcting codes, in particular polar codes.

ADRIEN CASSAGNE received the M.Sc. degree in computer science from the University of Bordeaux, France, in 2013, where he is currently pursuing the Ph.D. degree. His research interests include the design of efficient and flexible software implementations for modern decoding error-correcting codes such as LDPC, turbo, and polar codes. More precisely, he looks at different aspects of parallelism such as multi-node, multi-threading, or vectorization.



CAMILLE LEROUX received the M.Sc. degree in electronics engineering from the University of South Brittany, Lorient, France, in 2005, and the Ph.D. degree in electronics engineering from TELECOM Bretagne, Brest, France, in 2008. He was a Visiting Student with the Electrical and Computer Engineering Department, Aalborg University, Denmark, in 2004, and also with the University of Alberta, AB, Canada, in 2005. From 2008 to 2011, he was a Postdoctoral Research Associate with the Electrical and Computer Engineering Department, McGill University, Montreal, QC, Canada. He has been an Associate Professor with Bordeaux INP, since 2011. His research interests include encompass algorithmic and architectural aspects of channel decoder implementation. More generally, he is interested in the hardware and software implementation of computationally intensive algorithms in a real-time environment.

YVON SAVARIA (S'77–M'86–F'08) received the B.Ing. and M.Sc.A. degrees from the École Polytechnique Montreal, in 1980 and 1982, respectively, and the Ph.D. degree from McGill University, in 1985, all in electrical engineering. Since 1985, he has been with the École Polytechnique de Montréal, where he is currently a Professor with the Department of Electrical Engineering.

He has carried work in several areas related to microelectronic circuits and microsystems, such as testing, verification, validation, clocking methods, defect and fault tolerance, the effects of radiation on electronics, high-speed interconnects and circuit design techniques, CAD methods, reconfigurable computing and the applications of microelectronics to telecommunications, aerospace, image processing, video processing, radar signal processing, and digital signal processing acceleration. He is currently involved in several projects that relate to aircraft embedded systems, radiation effects on electronics, asynchronous circuits design and test, green IT, wireless sensor networks, virtual networks, machine learning, computational efficiency, and application specific architecture design. He holds 16 patents. He has published 140 journal papers and 440 conference papers. He was the thesis advisor of 160 graduate students who completed their studies.

He has been a Consultant or was sponsored for carrying research by Bombardier, Inc., CNRC, Design Workshop, DREO, Ericsson, Genesis, Gennum, Huawei, Hyperchip, ISR, Kaloom, LTRIM, Miranda, MiroTech, Nortel, Octasic, PMC-Sierra, Technocap, Thales, Tundra, and VXP. He is a Fellow of IEEE. He has been a member of CMC Microsystems Board, since 1999. He is a member of the Regroupement Stratégique en Microélectronique du Québec and of the Ordre des Ingénieurs du Québec. He was a Tier 1Canada Research Chair on design and architectures of advanced microelectronic systems, from 2001 to 2015. He received the Synergy Award of the Natural Sciences and Engineering Research Council of Canada, in 2006. He was a Program Co-Chairman of ASAP'2006 and NEWCAS'2018, and the General Co-Chair of ASAP'2007. He was a Chairman of CMC Microsystems Board, from 2008 to 2010.