

Broadband Digital Over-the-Air Computation for Wireless Federated Edge Learning

Lizhao You, Xinbo Zhao, Rui Cao, Yulin Shao, and Liqun Fu

Abstract—This paper presents the first orthogonal frequency-division multiplexing (OFDM)-based digital over-the-air computation (AirComp) system for wireless federated edge learning, where multiple edge devices transmit model data simultaneously using non-orthogonal OFDM subcarriers, and the edge server aggregates data directly from the superimposed signal. Existing analog AirComp systems often assume perfect phase alignment via channel precoding and utilize uncoded analog transmission for model aggregation. In contrast, our digital AirComp system leverages digital modulation and channel codes to overcome phase asynchrony, thereby achieving accurate model aggregation for phase-asynchronous multi-user OFDM systems. To realize a digital AirComp system, we develop a medium access control (MAC) protocol that allows simultaneous transmissions from different users using non-orthogonal OFDM subcarriers, and put forth joint channel decoding and aggregation decoders tailored for convolutional and LDPC codes. To verify the proposed system design, we build a digital AirComp prototype on the USRP software-defined radio platform, and demonstrate a real-time LDPC-coded AirComp system with up to four users. Trace-driven simulation results on test accuracy versus SNR show that: 1) analog AirComp is sensitive to phase asynchrony in practical multi-user OFDM systems, and the test accuracy performance fails to improve even at high SNRs; 2) our digital AirComp system outperforms two analog AirComp systems at all SNRs, and approaches the optimal performance when $\text{SNR} \geq 6$ dB for two-user LDPC-coded AirComp, demonstrating the advantage of digital AirComp in phase-asynchronous multi-user OFDM systems.

Index Terms—Federated edge learning, over-the-air computation, multiple access, channel codes, real-time implementation.

I. INTRODUCTION

DUE to the increased concern in data privacy-preserving, federated edge learning (FEEL) has become a popular distributed learning framework for mobile edge devices. A FEEL system often consists of an edge parameter server and multiple edge devices with non-independent and identically distributed (non-iid) local datasets. The parameter server periodically aggregates the locally-trained model parameters that are transmitted over the wireless channel.

Since large numbers of model parameters have to be transmitted in the model aggregation step, wireless communication becomes a bottleneck that restricts the practical development of FEEL [1]. Over-the-air computation (AirComp) has been proposed to address this problem [2], [3], as shown in

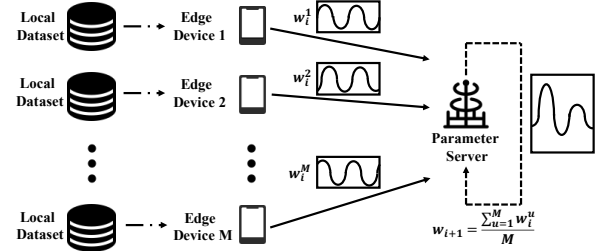


Fig. 1. Model aggregation in wireless federated edge learning via over-the-air computation where edge devices transmit simultaneously and the average data is decoded directly from the superimposed signal.

Fig. 1. In AirComp systems, selected edge devices transmit simultaneously using non-orthogonal frequency resources, and the parameter server decodes the arithmetic sum of edge devices' data directly from the superimposed signal. The non-orthogonal channel access allows the access of more devices given the same frequency resource, and improves the communication efficiency. Decoding aggregation data directly at the physical layer avoids decoding individual data first and then aggregation, and may improve the computation performance.

Existing AirComp systems [2]–[12] are all analog AirComp systems that use discrete-time analog transmission for communication. Specifically, the edge devices transmit their model parameters in an analog fashion without channel coding. The overlapped signals in the air naturally produce the arithmetic sum. The parameter server then extracts the arithmetic sum directly from the superimposed signal. However, the performance of analog AirComp is sensitive to the phase offsets among the superimposed signal: they can add up destructively if their signals are asynchronous, resulting in large aggregation errors. Therefore, these analog AirComp systems assume perfect channel precoding that can compensate all channel impairments, such as path loss, shadowing, small-scale fading, time offset (TO), and carrier frequency offset (CFO), and receive the clean arithmetic sum of the transmission signals.

However, realizing such perfect phase-aligned transmissions is quite challenging in practice, especially for random-access orthogonal frequency-division multiplexing (OFDM) systems such as WiFi. First, initial channel phase misalignment must be provided via a feedback protocol from the receiver, inducing some protocol overhead [13]. Second, the residual CFO is unavoidable due to estimation errors. Even with perfect phase alignment initially, the residual CFO rotates subcarriers' phases over time (symbols), and causes phase misalignment eventually. For a multi-user OFDM system, different users may have different residual CFOs, leading to rotating phase offsets

L. You, X. Zhao, R. Cao, and L. Fu are with the Department of Information and Communication Engineering, School of Informatics, Xiamen University, China. E-mails: {lizhaoyou, liqun}@xmu.edu.cn, {xbzhao, ruicao}@stu.xmu.edu.cn.

Y. Shao is with the Department of Electrical and Electronic Engineering, Imperial College London, UK. E-mail: y.shao@imperial.ac.uk.

over symbols. As a consequence, the performance of analog AirComp deteriorates.

To tackle the problem, this paper puts forth the first OFDM-based digital AirComp system where multiple users use non-orthogonal subcarriers for data transmission. Unlike analog AirComp that hinges on accurate channel precoding, digital AirComp leverages digital modulation and channel codes to combat channel impairments and phase misalignments, thereby achieving accurate model aggregation even when the signal phases of multiple edge devices are misaligned at the parameter server. Specifically, in channel-coded digital AirComp, model parameters are first quantized into bits, and then transmitted by a conventional OFDM encoder. Their signals get superimposed in the air. Then, the receiver aims to decode the aggregated source bits (which we refer to as *SUM bits*) instead of the individual source bits transmitted from different edge devices. To obtain SUM bits, the receiver first demodulates phase-asynchronous symbols, and then removes the protection of channel codes to get aggregation results. The latter process is referred to as joint channel decoding and aggregation (Jt-CDA). Similar to the analysis in wireless quantized federated learning [14], we prove that FEEL with digital AirComp can converge despite of SUM bit errors and quantization errors.

Convolutional codes and LDPC codes are two common codes used in OFDM systems. We target supporting them in our digital AirComp system. However, it is challenging to implement low-complexity and real-time Jt-CDA decoders. For convolutional-coded AirComp, the SUM packet(codeword)-optimal decoder has prohibitively high computational complexity, since several codewords (and corresponding source bits) may lead to the same SUM bits, and their probability must be aggregated first before decoding. Thus, we design two reduced-complexity Jt-CDA decoders for convolutional-coded AirComp. The first Jt-CDA decoder, *full-state joint decoder (FSJD)*, is a nearly optimal decoder that leverages the log-max approximation to realize maximum likelihood (ML) decoding. The second Jt-CDA decoder, *reduced-state joint decoder (RSJD)*, is a simplified version of FSJD that greatly reduces the number of states without compromising much performance for most SNRs.

For LDPC-coded AirComp, we present a new belief propagation algorithm that realizes the SUM bit-optimal decoder. We reuse the Tanner graph and the decoding framework of message passing for conventional single-user LDPC codes, but redefine the passing message as a vector and the corresponding message update equations. The vector represents the probability of all users' bit combinations, and the update equations sum the probability of combinations that generate the same output. After fixed rounds of iterations, we sum the probability of the bit combination that generates the same SUM bit, and choose the SUM bit with the maximal sum probability. The iterative BP algorithm and the per-bit mapping have low complexity and render it possible for real-time implementation.

We prototype these Jt-CDA decoders for digital AirComp on the USRP software-defined radio platform [15] with the GNU Radio software [16], and perform extensive simulations and experiments to evaluate their performance against analog

AirComp in phase-asynchronous FEEL systems. Simulation results show that phase offsets have a strong impact on the Jt-CDA performance. Trace-driven simulation results on test accuracy verify that the analog AirComp fails to improve the performance in the presence of phase misalignments even at a high signal-to-noise ratio (SNR) regime. In contrast, digital AirComp completely avoids the problem, and outperforms analog AirComp at all SNRs. The proposed digital AirComp can approach optimal learning performance at high SNRs. Furthermore, we implement a real-time LDPC-coded AirComp system with up to four simultaneous transmissions. The real-time system leverages the graphic processing unit (GPU) hardware for highly parallel signal processing, and achieves a network throughput of 1.16 Mbps for four-user LDPC-coded AirComp considering the protocol overhead.

The main technical contributions of this paper are summarized as follows:

- 1) We present the first digital AirComp system that exploits the joint design of channel decoding and aggregation to overcome phase asynchrony, and prove the convergence of FEEL with digital AirComp.
- 2) We put forth three low-complexity Jt-CDA decoders for digital AirComp: two decoders (i.e., FSJD and RSJD) for convolutional-coded AirComp and one decoder for LDPC-coded AirComp.
- 3) We study the physical-layer performance and the learning performance of FEEL with digital AirComp using experiment traces, and provide a real-time implementation of the LDPC-coded AirComp system.

The remainder of this paper is organized as follows. Section II presents the background and the motivation for digital AirComp. Section III overviews the proposed system. Section IV and Section V introduce the decoders for convolutional codes and LDPC codes. Experiment results are shown in Section VI. Section VII provides some discussions. Section VIII gives the related work. Section IX concludes the paper.

II. BACKGROUND AND MOTIVATION

In this section, we first review the federated edge learning model, and the multi-user OFDM model without channel precoding. Then, we discuss the difficulties in realizing phase-aligned transmissions via channel precoding, and advocate digital AirComp. Finally, we show the convergence and benefit of federated learning under digital AirComp.

A. Wireless Federated Edge Learning

In FEEL, multiple edge devices (e.g., Q devices) collaboratively learn a joint model with the help of a parameter server without transmitting raw datasets to each other. The federated model training requires many iterations to converge. Each iteration contains the following four steps:

- 1) Model broadcast: P edge devices are randomly selected by the parameter server and each of them is broadcasted with the current global model w_t in the t -th iteration;
- 2) Local training: every edge device u trains the received global model with their local data and gets a new local model w_t^u ;

- 3) Model update: the local models of selected edge devices are transmitted to the parameter server;
- 4) Model aggregation: the parameter server takes the average of the received models to update the global model for the next $t+1$ -th iteration: $w_{t+1} = \sum_{u=1}^P w_t^u / P$.

For mobile edge devices, FEEL is often realized by wireless networks (e.g., WiFi, 5G). In the following, we focus on step 3) and 4), and study how to apply over-the-air computation to improve the model aggregation performance.

B. Multi-User OFDM System Model

The model update and aggregation steps require a large bandwidth, since each edge device needs to transmit a large number of model coefficients to the parameter server. Therefore, in this paper, we consider OFDM wideband systems for the model update, especially WiFi OFDM systems. Different from traditional OFDM systems, AirComp uses non-orthogonal channel access where all edge devices transmit data at the same OFDM subcarriers.

For user u , our system first quantizes the updated model parameters with a quantization length of b bits per parameter. Then we pack several parameters in a packet for transmission. Assuming an OFDM packet can transmit n source bits, we can pack $\lceil n/b \rceil$ parameters in a packet. Let $S^u = \{s_0^u, \dots, s_{n-1}^u\}$ be the source bits fed to the OFDM transmitter. After channel encoding, the coded bits C^u are modulated in the frequency domain. We assume that binary phase shift keying (BPSK) is used in this paper, and each bit $C[l]$ is modulated to $X^u[l]$ with $X^u[l] \in \{+1, -1\}$. Then after inverse discrete Fourier transform (IDFT), cyclic prefix, preambles padding, and digital-to-analog conversion, X^u is transformed into a time-domain continuous signal $x^u(t)$.

Let \mathcal{U} be the set of users that transmit simultaneously. Let τ^u be the time offset (TO) with respect to the start of receiving window at the parameter server, and δ^u be the carrier frequency offset (CFO) between user u and the receiver. Then, the received baseband signal $y(t)$ can be represented as

$$y(t) = \sum_{u \in \mathcal{U}} (h^u(t) * x^u(t - \tau^u)) e^{j2\pi\delta^u t} + n(t), \quad (1)$$

where $h^u(t)$ is the impulse response of the channel between user u and the receiver, $*$ denotes the convolution operation, and $n(t)$ is the zero-mean circularly-symmetric complex additive white Gaussian noise (AWGN). Here $h^u(t)$ includes large-scale and small-scale fading, and thus is a complex vector under multipath channel. Let $h^u(t) = [h_1^u, \dots, h_L^u]$ where there are L paths and the l -th path has delay τ_l^u .

Most commercial multi-user OFDM systems (e.g., WiFi 6, 5G) specify the maximal time synchronization error and frequency synchronization error so that TO and CFO do not affect the decoding performance. For example, WiFi 6 enforces a timing accuracy of less than $\pm 0.4 \mu\text{s}$ and a CFO error less than 350Hz (i.e., 0.07 ppm at 5GHz) after the trigger frame for multi-user orthogonal transmissions [17]. Although the accuracy is specified, the TO and CFO of users are very likely to be different in practical systems, i.e., $\tau^{u_1} \neq \tau^{u_2}$ and $\delta^{u_1} \neq \delta^{u_2}$ ($u_1, u_2 \in \mathcal{U}$).

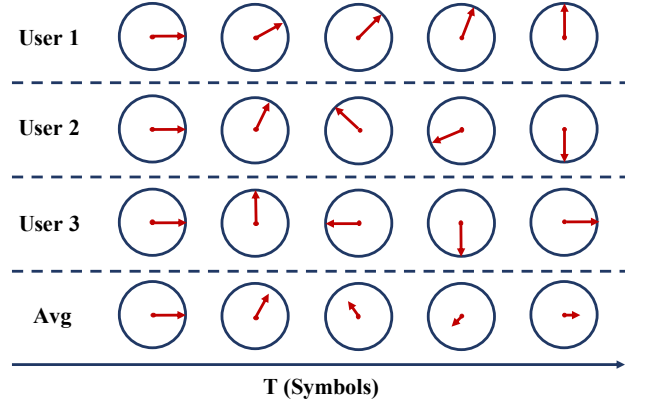


Fig. 2. The average operation of an analog AirComp system with imperfect channel precoding in one OFDM subcarrier over symbols.

Given such accuracy, the inter-subcarrier interference (ICI) is negligible, and the receiver can find a proper cyclic prefix cut without affecting the decoding performance. However, TO introduces phase rotation in each subcarrier, and CFO also causes phase rotation in each subcarrier over symbols. Specifically, the received baseband signal of the i -th symbol and the k -th subcarrier in the frequency domain can be represented as

$$Y_i[k] = \sum_{u \in \mathcal{U}} H^u[k] X_i^u[k] e^{j2\pi \frac{\tau_l^u k}{N_s}} e^{j2\pi \delta^u i (N_s + N_{cp})} + N_i[k], \quad (2)$$

where H^u is the fast Fourier transform (FFT) of h^u , X_i^u is the modulated signal in the frequency domain, N_s is the number of FFT points, and N_{cp} is the number of cyclic prefix (CP) samples. Assume the demodulation window is aligned with the first path. Then we have

$$H^u[k] = \sum_{l=1}^L h_l^u e^{j \frac{2\pi \tau_l^u k}{N_s}}. \quad (3)$$

According to the above formula, the multi-path channel introduces phase misalignment even with the CP design in OFDM since h_l^u and τ_l^u are different from different users.

Note that TO causes a linear phase rotation across subcarriers. On the other hand, although the inter-subcarrier interference (ICI) caused by a CFO value of 350Hz is negligible, CFO still causes constant phase rotation across subcarriers, and phase rotation accumulates over symbols. Since TO does not cause phase accumulation over symbols, Eq. (2) can be simplified as

$$Y_i[k] = \sum_{u \in \mathcal{U}} A^u[k] X_i^u[k] e^{j2\pi \delta^u i (N_s + N_{cp})} + N_i[k], \quad (4)$$

where $A^u[k] \triangleq H^u[k] e^{j2\pi \frac{\tau_l^u k}{N_s}}$.

Our target is to compute the arithmetic sum $S^F \triangleq \sum_{u \in \mathcal{U}} S^u = \{\sum_{u \in \mathcal{U}} s_0^u, \dots, \sum_{u \in \mathcal{U}} s_{n-1}^u\}$ of all parameters from Y , where operator in \sum is the arithmetic sum. Then, we can easily compute the average.

C. Challenges of Phase-Aligned Transmissions

Existing OFDM-based AirComp systems [3], [7] assume near-perfect phase alignment using channel precoding. Such

channel precoding is implemented in an analog AirComp system recently [18]. Multi-user OFDM systems with channel precoding have been demonstrated in network MIMO systems [19]–[22] and a physical-layer network coding (PNC) system [13]. Nevertheless, considerable protocol overhead is required for the initial channel phase alignment from different users. Even if the channel phases are aligned initially, maintaining such a phase alignment is quite costly due to residual CFO after precoding, especially when the packet duration is large.

In Eq. (4), due to the existence of $A^u[k]$ and δ^u , their channels are not phase synchronous. The task of a channel precoding protocol is to eliminate the power difference and phase difference caused by $H^u[k]$, τ^u and δ^u . In particular, the receiver estimates composed channel $\hat{A}^u[k]$ and CFO δ^u , and sends them back to the transmitter. The transmitter precodes its frequency-domain signal $S_i^u[k]$ with a coefficient $P_i^u[k]$ in the k -th subcarrier of the i -th symbol (i.e., $X_i^u[k] = P_i^u[k]S_i^u[k]$), where

$$P_i^u[k] = \frac{e^{-j2\pi\hat{\delta}^u i(N_s+N_{cp})}}{\hat{A}^u[k]}.$$

Then the received signal with channel precoding is represented by

$$Y_i[k] = \sum_{u \in \mathcal{U}} \Delta A^u[k] S_i^u[k] e^{j2\pi\Delta\delta^u i(N_s+N_{cp})} + N_i[k], \quad (5)$$

where $\Delta A^u[k] = \frac{A^u[k]}{\hat{A}^u[k]}$ and $\Delta\delta^u = \delta^u - \hat{\delta}^u$.

In practical systems with channel precoding, the receiver needs to estimate $\hat{A}^u[k]$ on the time granularity of the channel coherence time, and quantize $\hat{A}^u[k]$ for all subcarriers for feedback. Moreover, $\Delta A^u[k] \neq 1$ and $\Delta\delta^u \neq 0$ due to estimation errors. The phase offset caused by $\Delta A^u[k]$ is constant over the whole packet, and thus it does not affect decoding with accurate precoding. However, the phase offset caused by $\Delta\delta^u$ accumulates as time goes, and can easily lead to severe phase misalignment for a long packet. Fig. 2 illustrates such a phenomenon and its impact on the average operation of analog AirComp in one OFDM subcarrier. As we can see, the phase misalignment over symbols degrades the performance of analog AirComp.

Highly-accurate CFO precoding is not easy to achieve in practical systems. For example, given the ± 0.07 ppm frequency error specified by WiFi 6 (assuming the central frequency 5GHz), 1.5ms will lead to a phase offset of around $2 \times 0.07 \cdot 10^{-6} \times 5 \cdot 10^9 \times 1.5 \cdot 10^{-3} \times 2\pi \approx \pi$ radian (i.e., totally misaligned) in the worst case. Assuming that the phase offset must be less than $\pi/4$ radian for constructive signal superimposition, a packet must be no more than 100 symbols (the duration of each symbol is $4\mu\text{s}$ for 20MHz bandwidth and 80 samples per symbol). The packet duration corresponds to 300Bytes (assuming BPSK and 1/2 coding rate), which is not adequate for FEEL.

To tackle the problem, many approaches have been proposed. A high-precision but expensive clock can be used (e.g., global position system disciplined oscillator clock with 0.1ppb [23]). Protocols with some signaling overhead are used in the two-way relay network [13] and the multiple access network [18], [24]. Extra radio [20], [21] or extra infrastructure [19],

[22], [25] are required for network MIMO. In this paper, we leverage the digital AirComp technique to overcome the phase asynchrony problem.

D. Convergence of Digital AirComp-enabled FEEL

Although our digital AirComp system can decode aggregate data, some sum bits are inevitably erroneous. Given that most FEEL algorithms work under the error-free assumption, the first question is whether such sum bit errors affect the convergence of federated learning.

For simplicity, we assume that the model broadcast step is error-free and only focus on finding out how the SUM bit error rate (SUM BER) of the received superimposed signal and quantization precision in the model aggregation step affect the convergence rate of our proposed system. The proof is similar to [14] that considers quantization errors, but differs in that we further consider the transmission errors in the uplink (i.e., the SUM BER).

We consider an FL process consisting of T communication rounds, and each round contains τ steps of the stochastic gradient descent (SGD) operation. Without loss of generality, we assume all edge devices participate in the training, and M is the number of edge devices. We assume the each device transmits d parameters of the local model in each communication round. The goal of the training process is to find the parameter w^* that minimizes the loss function of the whole dataset

$$F(w) = \sum_{n=1}^M p_n F_n(w), \quad (6)$$

where p_n is the fraction of local datasets among total datasets and $F_n(w)$ is the loss function of edge device n . That is, $w^* = \arg \min_w F(w)$.

Let $w(t)$ be the broadcasted model at the beginning of the t -th communication round. Take the n -th edge device for example. The i -th step of SGD can be represented as

$$w_n^i(t) = w_n^{i-1}(t) - \eta(n) \nabla F_n(w_n^{i-1}(t), \xi_n^{i-1}(t)), \quad (7)$$

where $w_n^0(t) = w(t)$, $\eta(n)$ represents the learning rate, and $\xi_n^i(t)$ represents the batch size. Then the local weight difference in the t -th communication round is defined as

$$\Delta w_n(t) = w_n^\tau(t) - w(t). \quad (8)$$

We define the global model at the parameter server of the t -th communication round as

$$w(t+1) = w(t) + \frac{1}{M} \sum_{n=1}^M p_n B(\Delta w_n(t)), \quad (9)$$

where $B(\Delta w_n(t))$ is the updated weight differential from selected edge device n .

The correctness of the global weight is determined by the quantization precision and SUM BER of the transmission, which can be represented as

$$B(\Delta w_n(t)) = Q(\Delta w_n(t)) + X(\Delta w_n(t)), \quad (10)$$

where $Q(\Delta w_n(t))$ is the updated weight error after quantization and $X(\Delta w_n(t))$ is the update weight error caused by communication.

To facilitate the convergence analysis, we use the single-user BER α for analysis in each communication round. The exact relationship between BER α and SUM BER β is shown in Appendix A. Let $B_n(t)$ be the number of quantization bits. Assume \mathcal{K} contains the erroneous positions where $\mathcal{K} \subset \{1, \dots, B_n(t)\}$ in the t -th communication round. The communication error induced by \mathcal{K} is given by

$$\begin{aligned} X^{\mathcal{K}}(\Delta w_n(t)) &= \sum_{k \in \mathcal{K}} X^k(\Delta w_n(t)) \\ &= \sum_{k \in \mathcal{K}} \frac{\Delta w_n^{\max}(t) - \Delta w_n^{\min}(t)}{2^{B_n(t)} - 1} \times 2^{k-1} \times m^k, \end{aligned} \quad (11)$$

where $\Delta w_n^{\max}(t)$ and $\Delta w_n^{\min}(t)$ are the maximum and minimum value of weight difference vector after edge device n performs local training, and m^k is an indicator that is -1 if the error of the k -th bit is $1 \rightarrow 0$, or 1 if the error is $0 \rightarrow 1$. Note that $\mathbb{E}(X^{\mathcal{K}}(\Delta w_n(t))) = 0$, since each bit has an equal probability for $0 \rightarrow 1$ and $1 \rightarrow 0$.

Let \mathbb{K} be all possible error patterns. Then

$$\mathbb{E}[X(\Delta w_n(t))] = \sum_{\mathcal{K} \in \mathbb{K}} p^{\mathcal{K}} \mathbb{E}[X^{\mathcal{K}}(\Delta w_n(t))] = 0, \quad (12)$$

where $p^{\mathcal{K}}$ is the probability that bits in \mathcal{K} are all erroneous.

Then, we have the following lemma.

Lemma 1. $B(\Delta w_n(t))$ is an unbiased estimator of $\Delta w_n(t)$, i.e.,

$$\mathbb{E}[B(\Delta w_n(t))] = \Delta w_n(t), \quad (13)$$

while it also holds that

$$\mathbb{E}[\|B(\Delta w_n(t)) - \Delta w_n(t)\|_2^2] \leq J_n^2(t) + K_n(t), \quad (14)$$

where

$$\begin{aligned} J_n^2(t) &\triangleq \frac{\delta_n^2(t)}{(2^{B_n(t)} - 1)^2}, \\ K_n(t) &\triangleq \alpha \left(\frac{4\delta_n(t)^2}{d(2^{B_n(t)} - 1)^2} \right)^2 \times \frac{1 - 4^{B_n(t)}}{1 - 4}, \\ \delta_n(t) &\triangleq \sqrt{\frac{d}{4} (\Delta w_n^{\max}(t) - \Delta w_n^{\min}(t))^2}. \end{aligned}$$

Proof. See Appendix B. \square

Next, we use the following commonly used assumptions to prove convergence.

Assumption 1. (μ -strongly convex). For each edge device n , function F_n is μ -strongly convex, which means, for all w and w' , we have $F_n(w') \geq F_n(w) + \langle w' - w, \nabla F_n(w) \rangle + \frac{\mu}{2} \|w' - w\|_2^2$.

Assumption 2. (L -smoothness). For each edge device n , function F_n is L -smooth, which means, for all w and w' , we have $F_n(w') \leq F_n(w) + \langle w' - w, \nabla F_n(w) \rangle + \frac{L}{2} \|w' - w\|_2^2$.

Assumption 3. (Uniformly bounded). For each edge device n , communication round t , and local SGD step i , the expected squared norm of local stochastic weight differences is uniformly bounded, $\mathbb{E}[\|\nabla F_n(w_n^i(t), \xi_n^i(t))\|_2^2] \leq G^2$.

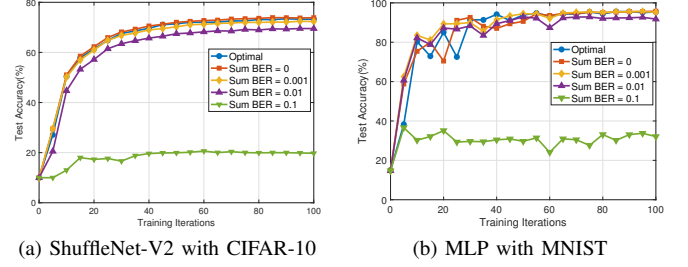


Fig. 3. Test Accuracy Performance of two digital AirComp-enabled FEEL systems under different SUM BERs.

Assumption 4. (Variance bounded). For each edge device n , communication round t , and local SGD step i , the estimate of local stochastic gradients has a bounded variance, $\mathbb{E}[\|\nabla F_n(w_n^i(t), \xi_n^i(t)) - \nabla F_n(w_n^i(t))\|_2^2] \leq \sigma_n^2$.

We define Γ as the degree of non-iid among user's datasets, $\Gamma \triangleq F(w^*) - \sum_{n=1}^M p(n) F_n^*$, where F_n^* represents the minimum value of F_n . With the above equations and assumptions, we can have the following theorem.

Theorem 1. By selecting a diminishing learning rate $\eta(t) = \frac{2}{\mu(\gamma+t)}$ and $\gamma > \max\{2, \frac{2}{\mu}, \frac{L}{\mu}\}$, $\mathbb{E}[F(w(T)) - F(w^*)]$ is upper bounded by

$$\begin{aligned} \mathbb{E}[F(w(T)) - F(w^*)] &\leq \\ &\frac{L}{2} \frac{1}{\gamma + T} \left(\frac{4U}{\mu^2} + \gamma \mathbb{E}[\|w(0) - w^*\|_2^2] \right) \\ &+ \frac{L}{2} \sum_{j=0}^{T-1} \left[\sum_{n=1}^M p_n (J_n^2(t) + K_n(t)) \prod_{i=j+1}^{T-1} \left(1 - \frac{2}{\gamma + i} \right) \right], \end{aligned} \quad (15)$$

where

$$U = \tau^2 \sum \sigma_n^2 + \tau G^2 + 2L\tau^2 \Gamma + (\mu + 2) \frac{\tau(\tau-1)(2\tau-1)}{6} G^2.$$

Proof. See Appendix B. \square

In Eq. (15), the first term of the upper bound tends to zero for large T . For the second term, compared with results in [14], the added element $K_n(t)$ is related to BER α . The second term related to quantization error and transmission error indeed causes a gap. However, following the same argument in [14], the term $\prod_{i=j+1}^{T-1} (1 - \frac{2}{\gamma+i})$ tends to zero for small j , which means the gap may be negligible in practice.

Simulation Results. We further study the convergence of the digital AirComp-enabled FEEL system, and the impact of SUM BER on the learning accuracy of FEEL applications by simulation. We choose two typical FL applications with non-iid data (the data allocation method can be found in Section VI-B): CIFAR-10 with ShuffleNet-V2 [26] and MNIST with multi-layer perceptron (MLP) [27], and investigate their performance under different SUM BERs. We adopt 8-bit probability quantization [28] for each model parameter. There are 40 edge devices in total, and 4 devices are randomly chosen for training each time. The test accuracy results are shown in Fig. 3(a) and Fig. 3(b), where the optimal curve shows the result when model parameters are not quantized and aggregated without error. They can converge to the optimal if

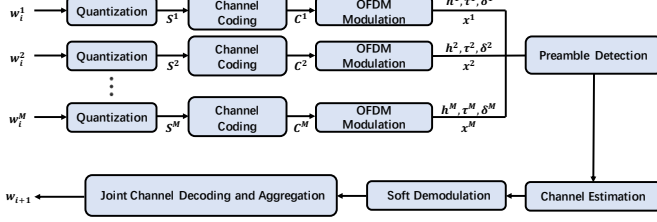


Fig. 4. The architecture of the OFDM-based digital AirComp transceiver.

SUM BER is smaller than 10^{-3} , and converge close to the optimal if smaller than 10^{-2} .

III. SYSTEM OVERVIEW

There are Q devices in the network. In each FEEL iteration, P edge devices are chosen to participate in model update. They use the digital AirComp system that allows M devices to transmit simultaneously, and require $\lceil P/M \rceil$ rounds of transmissions in each iteration. In the following, we focus on one round of the digital AirComp transmission.

Architecture. Fig. 4 shows the transceiver architecture of our digital AirComp system and the workflow. To be best compatible with the WiFi standards, our system reuses most components as in conventional WiFi OFDM systems and makes minimal modifications. In particular, the transmitter architecture is almost the same as the traditional OFDM transmitter. For the receiver, we follow the conventional frame detection and multi-user channel estimation design, but redesign the demodulation and channel decoding for the arithmetic sum computation. Demodulation is a soft demodulation block where the probabilities of all possible constellation combinations are preserved.

Channel decoding and aggregation are performed jointly, which is referred to as *Jt-CDA*. The Jt-CDA decoder is the newly designed block in digital AirComp. Given that both convolutional codes and LDPC codes are mandatory in WiFi 6 802.11ax [17], we aim to support these two codes. The design for convolutional codes is presented in Section IV, and the design for LDPC codes is presented in Section V.

Protocol. Fig. 5 shows the enabling medium access control (MAC) protocol for multi-user non-orthogonal channel access. Similar to the multi-user simultaneous transmission protocol in 802.11ax (e.g., UL-MIMO), the parameter server contends the channel on behalf of all users following the traditional random access protocol in Wi-Fi networks, and then transmits a trigger frame to notify the transmission users in each round. The trigger frame includes the coding scheme information (e.g., which channel codes and coding rate to use) and the power control information (e.g., the used transmission power and the expected reception power). All informed users synchronize to the trigger frame and then transmit OFDM frames simultaneously with their preambles orthogonal in time domain, pilots orthogonal in frequency domain, and data symbols overlapped in time domain. Furthermore, all informed users use the provided power information to perform fine-grained power control to ensure the reception powers of their signals are near-balanced. Such a power control scheme is valid due to the

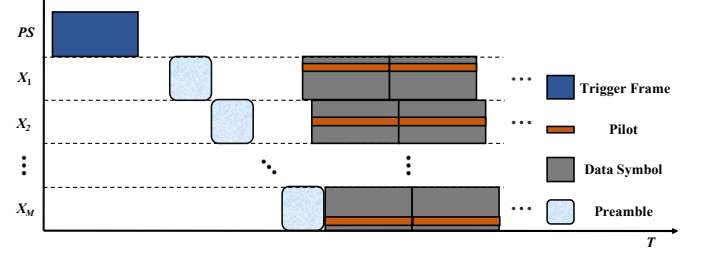


Fig. 5. A MAC protocol for digital AirComp where the parameter server contends the channel on behalf of all users following the traditional random access protocol in Wi-Fi networks.

existence of channel reciprocity where the uplink channel and the downlink channel go through almost the same attenuation in the same frequency, and has also been adopted for multi-user simultaneous transmissions (e.g., OFDMA and UL MU-MIMO) in IEEE 802.11ax [17].

Each OFDM frame consists of 10 short training sequences (STS) and 2 long training sequences (LTS), and some data symbols. Moreover, we allocate orthogonal pilots to each edge device in all OFDM data symbols. In this way, the receiver can estimate the initial frequency-domain channel through orthogonal LTSes, and track phase rotation through orthogonal pilots. Then the receiver performs Jt-CDA to compute average model parameter coefficients. In particular, for the t -th round, device u quantizes its weight w_t^u to S_t^u , encodes and modulates S_t^u to $x_t^u(t)$. The received signal is shown as Eq. (1) in time domain and Eq. (2) in frequency domain. In the end, the receiver extracts the average weight w_{t+1} for the next round.

IV. DESIGN FOR CONVOLUTIONAL CODES

Convolutional codes are widely used in wireless communication systems (e.g., WiFi, Cellular, Satellite). For single-user communication systems, convolutional codes are often decoded by the Viterbi algorithm that performs maximum likelihood (ML) decoding and achieves the codeword-optimal performance. However, for multi-user simultaneous transmission systems, such SUM codeword-optimal performance is not easy to achieve.

A. SUM Packet-optimal Decoder

For ease of illustration, we assume there are only two concurrent users A and B . The extension to more users is straightforward. Their source packets are S^A and S^B with length K . Let the encoding function be $\Pi(\cdot)$. Their encoded packets (codewords) are $C^A = \Pi(S^A)$ and $C^B = \Pi(S^B)$ with length N . Their modulated signals are denoted by X^A and X^B . Let Y denote the received signal. Our target is to get the arithmetic sum of source packets $S^F = S^A + S^B$. Note that there is no valid codeword corresponding to S^F since the arithmetic sum $+$ operation is not closed under \mathcal{F}_2 and $\Pi(\cdot)$. Therefore, we also call it *SUM packet-optimal decoder*. More specifically, our target is to compute

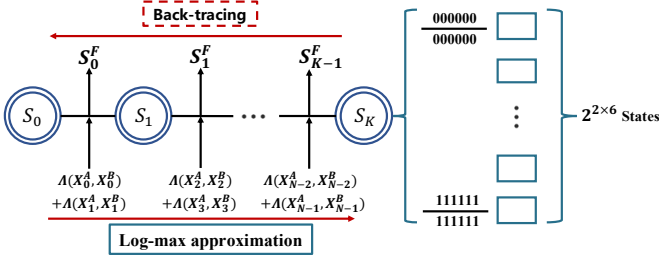


Fig. 6. Joint trellis of the two-user convolutional decoder, assuming K source bits, N coded bits, $1/2$ coding rate and constraint length $L=7$ for each user.

$$\begin{aligned} \hat{S}^F &= \arg \max_{S^F} \sum_{C^A, C^B: \Pi^{-1}(C^A) + \Pi^{-1}(C^B) = S^F} Pr(Y|C^A, C^B) \\ &= \arg \max_{S^F} \log \sum_{C^A, C^B: \Pi^{-1}(C^A) + \Pi^{-1}(C^B) = S^F} Pr(Y|C^A, C^B), \end{aligned} \quad (16)$$

where

$$\begin{aligned} Pr(Y|C^A, C^B) &= \prod_{n=1}^N Pr(Y[n]|C^A[n], C^B[n]) \\ &\propto \prod_{n=1}^N \exp\left(-\frac{\|Y[n] - H^A[n]X^A[n] - H^B[n]X^B[n]\|^2}{2\sigma^2}\right) \\ &\propto \exp\left(-\sum_{n=1}^N \frac{\|Y[n] - H^A[n]X^A[n] - H^B[n]X^B[n]\|^2}{2\sigma^2}\right). \end{aligned} \quad (17)$$

Eq. (16) means that we need to add up the probability of all possible codeword pairs (C^A, C^B) , aggregate the probability of pairs that yield the same S^F , and then make decision. The functional mapping from S^A and S^B to the SUM packet S^F can be expressed as

$$f_{packet} : \{0, 1\}^K \times \{0, 1\}^K \rightarrow \{0, 1, 2\}^K.$$

In fact, it is a $(4/3)^K$ -to-1 mapping. There is no known exact computation method for Eq. (16) except for exhaustively sum over all possible combinations, which has prohibitively high computational complexity.

B. Full-State Joint Decoder

Full-state joint decoder (FSJD) is an approximated decoder for the ML decoder. In particular, Eq. (16) can be simplified using log-max approximation (i.e., $\log(\sum_i \exp(x_i)) \approx \max_i x_i$) to

$$\begin{aligned} \hat{S}^F &\approx \arg \max_{S^F} \max_{C^A, C^B: \Pi^{-1}(C^A) + \Pi^{-1}(C^B) = S^F} Pr(Y|C^A, C^B) \\ &\approx \arg \max_{C^A, C^B: \Pi^{-1}(C^A) + \Pi^{-1}(C^B) = S^F} Pr(Y|C^A, C^B) \\ &\approx \arg \min_{C^A, C^B: \Pi^{-1}(C^A) + \Pi^{-1}(C^B) = S^F} \frac{\Lambda(X^A, X^B)}{2\sigma^2} \\ &\approx \arg \min_{C^A, C^B: \Pi^{-1}(C^A) + \Pi^{-1}(C^B) = S^F} \Lambda(X^A, X^B) \\ &\approx \arg \min_{C^A, C^B: \Pi^{-1}(C^A) + \Pi^{-1}(C^B) = S^F} \sum_{n=1}^N \Lambda(X_n^A, X_n^B), \end{aligned} \quad (18)$$

where

$$\Lambda(X_n^A, X_n^B) = \|Y[n] - H^A[n]X^A[n] - H^B[n]X^B[n]\|^2. \quad (19)$$

The computation of Eq. (18) includes two steps: 1) find the best pair of codewords \hat{C}^A and \hat{C}^B such that $(\hat{C}^A, \hat{C}^B) = \arg \min_{C^A, C^B} \Lambda(X^A, X^B)$; 2) map \hat{C}^A to \hat{S}^A and \hat{C}^B to \hat{S}^B , and get $\hat{S}^F = \hat{S}^A + \hat{S}^B$. Step 1) is equivalent to finding the minimum-cost path on the joint trellis of user A's and user B's encoders, and can be implemented by using the Viterbi algorithm. In Eq. (19), for BPSK-modulated received signal $Y[n]$, we can construct four possible constellations with $H^A[n]$ and $H^B[n]$ ($X^A[n], X^B[n] \in \{+1, -1\}$), and calculate its Euclidean distance to the received point as the edge cost. The demodulation is called *soft demodulation*. Since the state space of the joint trellis is the combination of user A's and user B's state space, it is referred to as *full-state* joint Viterbi decoder.

For single-user convolutional codes with constraint length L , the number of state in trellis is $2^{(L-1)}$. Each state branches out two edges to two states in the next stage, and each edge is associated with 1 input bit and r output bits for a convolution code with coding rate r . For the two-user joint trellis case, the number of states is $2^{2(L-1)}$, and each state branches out four edges to four states in the next stage. Moreover, each edge is associated with 2 input bits (each user owns 1 bit) and $2r$ output bits (each user owns r bits). Fig. 6 shows the joint trellis for two-user simultaneous transmission, where the joint state is represented by $2(L-1)$ bits with each user $L-1$ bits.

The Viterbi decoding algorithm is to find the minimum-cost path from the start state to the end state. In particular, for each decoding state, it performs the Add-Compare-Select (ACS) operation to compute and choose the minimum-cost path to the current state (i.e., path metric), and records the previous best state for back-tracing. The path metric value of the k -th state at the $i+1$ -th decoding stage can be represented as:

$$PM[k, i+1] = \min_{k' \in \text{set}(k)} (PM[k', i] + BM[k' \rightarrow k]), \quad (20)$$

where $\text{set}(k)$ is the set of four states that branch out to state k at the last stage i , and $BM[k' \rightarrow k]$ is the branch metric from state k' to k . Note that $BM[k' \rightarrow k]$ is the sum of corresponding bits' Euclidean distance in the constellation graph, while the distance for one bit is shown in Eq. (19).

After running the algorithm to the last decoding stage, we can determine the end state and identify the minimum-cost path from the start state to the end state. By tracking back the path, we can recover the joint source bits corresponding to the minimum-cost path and obtain the sum bits by summing the decoded source bits. The sum bits are then transformed back to floating-point values corresponding to each parameter, and average is performed to get the update model parameter value. Fig. 6 also shows the decoding procedure: by back-tracing, the decoder finds the minimal-cost path, and the found path corresponds to the optimal pair (C^A, C^B) ; by mapping the pair to corresponding source bits (S^A, S^B) , the decoder gets S^F .

C. Reduced-State Joint Decoder

The computational complexity of FSJD is proportional to the number of states in each decoding stage, since each state is associated with one ACS operation. The aforementioned FSJD has a high computational complexity, since there are $2^{2(L-1)}$ states in each decoding stage. For the convolution codes defined in WiFi, the constraint length is 7, and there are 4096 states for two-user simultaneous transmissions. The state number increases exponentially as the user number n increases (e.g., $2^{n(L-1)}$), leading to a high computation cost. For example, there are $2^{6 \times 3} = 262144$ states for three-user simultaneous transmissions.

Reduced-state joint decoder (RSJD) is a simplified (and approximated) version of FSJD. In particular, it limits the number of states participating in the ACS operation in the next decoding stage. At most R states with minimum path metric are selected to advance to the next decoding stage, and other states are ignored. Therefore, we call it *reduced-state joint Viterbi decoder*.

In our joint trellis, R states branch out to at most $2^n R$ states in the next decoding stage, but RSJD still selects at most R states for the following decoding stages. We use the quick-sort algorithm to select these R states in our implementation. In this way, RSJD reduces the computational complexity, although it may not be the optimal in terms of the minimum-cost path, since some paths with smaller cost may be ignored.

In Section VI-A, we study the performance loss under two-user simultaneous transmissions. Results show that the performance of RSJD approaches that of FSJD for most SNRs, especially for high SNR regimes. The performance loss may be larger for more users, since the total number of states increases exponentially. However, such reduced-state approximation is essential for real-time implementation. Therefore, we adopt RSJD and adapt the number of reserved states according to the number of simultaneous transmissions in our implementation.

D. Complexity Analysis

We have presented an optimal decoder and two reduced-complexity decoders. In the following, we give a complexity analysis of these decoders, and show how much complexity can be reduced.

First, the optimal decoder can also be viewed as computations in the joint trellis. Different from the Viterbi algorithm that only keeps one path in each state, the optimal decoder keeps all possible paths in each state. Therefore, we can focus on the number of performed ACS operations, and ignore the computation of branch metrics that are common for all three decoders.

For the general M -user digital AirComp system, the optimal decoder needs to consider 2^{MK} codeword pairs and $(M+1)^K$ SUM codewords. We need to exhaustively compute the probabilities for all codeword pairs, and sum the probabilities of corresponding pairs to compute the probability of a SUM codeword. Therefore, the optimal decoder needs 2^{MK} add operations, and 1 compare-select operation. Instead, FSJD needs $2^{M(L-1)} \cdot K$ ACS operations, and RSJD needs $R \cdot K$ ACS operations. If we ignore the complexity of the

compare-selection operation, the computation complexity of these decoders are $O(2^{MK})$, $O(2^{M(L-1)} \cdot K)$, and $O(R \cdot K)$, respectively. Given the packet length K is much larger than the constraint length L , FSJD reduces the complexity from exponential (i.e., 2^K) to linear (i.e., K). RSJD further reduces the complexity from $2^{M(L-1)}$ to R .

V. DESIGN FOR LDPC CODES

LDPC codes are linear block codes with a sparse generator matrix. Compared with convolutional codes, they have a better performance in approaching the Shannon capacity. For single-user communication systems, codeword-optimal (ML) decoding for LDPC codes is NP-hard, and the iterative belief propagation (BP) algorithm that achieves the bit-optimal performance is often adopted. Therefore, we also focus on the BP-based *SUM bit-optimal decoder* to decode the arithmetic sum. The difficulty lies in defining the passing message and the update equations. This section presents the decoding algorithm in details.

A. SUM Bit-optimal Decoder

The SUM bit-optimal decoder is defined as follows. The i -th SUM source bits \hat{S}_i^F , $i = 1, \dots, K$, is given by

$$\hat{S}_i^F = \arg \max_{S_i^F} \sum_{S_i^A, S_i^B: S_i^A + S_i^B = S_i^F} Pr(S_i^A, S_i^B | Y, \mathcal{C}), \quad (21)$$

where $Pr(S_i^A, S_i^B | Y, \mathcal{C})$ denotes the *a posteriori* probability of (S_i^A, S_i^B) given the received signal Y and the codebook \mathcal{C} . Eq. (21) is to compute the probability of $Pr(S_i^A, S_i^B | Y, \mathcal{C})$, map (S_i^A, S_i^B) to S_i^F according to the functional mapping method

$$f_{bit} : \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1, 2\},$$

and choose the S_i^F value with the maximal probability. It is a (4/3)-to-1 mapping, and the mapping complexity is significantly lower than the codeword-optimal decoding.

Then, the key is to compute the marginal probability distributions $Pr(S_i^A, S_i^B | Y, \mathcal{C})$. That is, we compute the probability of (S_i^A, S_i^B) with the assistance of coding structure in codebook \mathcal{C} . Fortunately, it can be computed using the classical BP framework, which is a general framework for generating inference-making algorithms for graphical models. It can find the ML SUM source bit without incurring exponential growth in complexity.

B. Bit-optimal LDPC Decoder

LDPC codes are linear block codes with a sparse generator matrix. Compared with convolutional codes, they have a better performance in approaching the Shannon capacity. For single-user communication systems, codeword-optimal (ML) decoding for LDPC codes is NP-hard, and the iterative belief propagation (BP) algorithm that achieves the bit-optimal performance is often adopted. Therefore, we focus on the BP-based *SUM bit-optimal decoder* to decode the arithmetic sum

for LDPC codes¹. The difficulty lies in defining the passing message and the update equations. This section presents the decoding algorithm in details.

Our multi-user LDPC SUM decoder reuses the Tanner graph structure for single-user LDPC, but redesigns the passing message and the update rules for decoding. For ease of illustration, we use the two-user LDPC SUM decoder for example. The extension to more users is straightforward.

Message Vector: The SUM decoder uses a message vector that contains the probability of multi-users' bit vector instead of a log likelihood ratio message for single-user. For example, for two-user LDPC, we define the information passed between nodes as a vector $(p_{00}, p_{01}, p_{10}, p_{11})$, in which

$$p_{ij} = \Pr(X^A[n] = 2i - 1, X^B[n] = 2j - 1 | Y[n]) \quad (22)$$

is the probability that the user A's n -th coded bit is i and the user B's n -th coded bit is j given the n -th received bit signal assuming the BPSK bit-to-signal mapping of $1 \rightarrow 1$ and $0 \rightarrow -1$. The message vector will be used to update the information in both variable nodes and check nodes.

Tanner Graph: Fig. 7(a) shows the Tanner graph for the multi-user simultaneous transmission system. Given the above message vector definition, we can reuse the Tanner graph for single-user LDPC. A variable node corresponds to an output bit vector, and the check nodes correspond to their redundancy relationship that their finite-field sum is zero. The evidence nodes correspond to the received signal associated with variable nodes.

Message Initialization: The information from an evidence node to a variable node is the initial messages. Note that the message remains the same in every iteration of the decoding process. The information from the evidence node n is computed from the received signal $Y[n]$ as

$$\begin{aligned} P &= (p_{00}, p_{01}, p_{10}, p_{11}) \\ &= \frac{1}{\beta_1} (\Pr(X^A[n] = -1, X^B[n] = -1 | Y[n]), \\ &\quad \Pr(X^A[n] = -1, X^B[n] = 1 | Y[n]), \\ &\quad \Pr(X^A[n] = 1, X^B[n] = -1 | Y[n]), \\ &\quad \Pr(X^A[n] = 1, X^B[n] = 1 | Y[n])) \\ &= \frac{1}{\beta_1} (\exp(\frac{-D(-1, -1)}{2\sigma^2}), \exp(\frac{-D(-1, 1)}{2\sigma^2}), \\ &\quad \exp(\frac{-D(1, -1)}{2\sigma^2}), \exp(\frac{-D(1, 1)}{2\sigma^2})), \end{aligned} \quad (23)$$

where β_1 is the sum of four probability for normalization and $D(X^A[n], X^B[n])$ is given by

$$D(X^A[n], X^B[n]) = \frac{\|Y[n] - H^A[n]X^A[n] - H^B[n]X^B[n]\|^2}{\alpha}$$

Here α represents the Euclidean distance normalization factor, since the power of superimposed signal in different subcarriers may be diverse due to heterogeneous phase offsets. We set α as the minimal distance over all subcarriers in one packet.

¹There also exists a SUM bit-optimal decoder for convolutional codes. However, its performance does not make a big difference compared with the approximated codeword-optimal decoders in Section IV despite higher complexity. Therefore, we do not show the design here, and put the details in Appendix C.

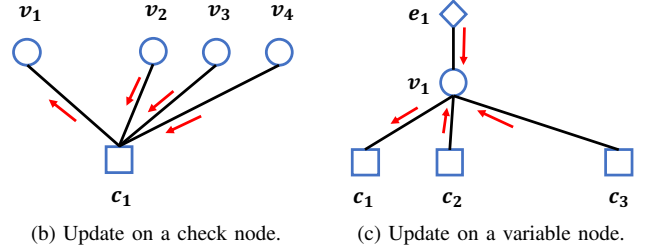
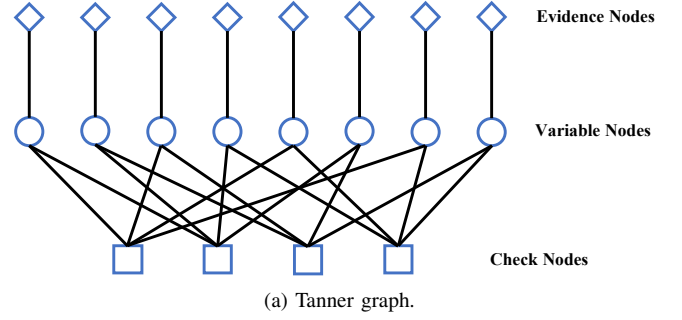


Fig. 7. LDPC decoder for multi-user simultaneous transmissions.

Update Equations on Check Nodes: We use function $CHK(\cdot)$ to denote the update equation that computes the output message $\mathcal{M}(\cdot)$ going out of a check node to a variable node. $CHK(\cdot)$ supports multiple inputs, and is computed sequentially, i.e.,

$$CHK(U, V, \dots) = CHK(CHK(U, V), \dots).$$

Fig. 7(b) shows an example of the updating process on a check node c_1 . It is to compute $\mathcal{M}(c_1 \rightarrow v_1)$ defined as

$$\mathcal{M}(c_1 \rightarrow v_1) = CHK(\mathcal{M}(v_2 \rightarrow c_1), \mathcal{M}(v_3 \rightarrow c_1), \mathcal{M}(v_4 \rightarrow c_1)).$$

To ease notation, we rewrite $\mathcal{M}(c_1 \rightarrow v_1)$ as $CHK(v_2 \rightarrow c_1, v_3 \rightarrow c_1, v_4 \rightarrow c_1)$ or $CHK(\{v_2, v_3, v_4\} \rightarrow c_1)$.

Since $CHK(\cdot)$ is computed sequentially, we focus on the minimal computation unit $CHK(I, J)$, where the message vectors of the two input information edges are $I = (i_{00}, i_{01}, i_{10}, i_{11})$ and $J = (j_{00}, j_{01}, j_{10}, j_{11})$. The probability that the composed temporary node v is 00 is obtained as

$$\begin{aligned} \Pr(v = 00 | I, J) &= \Pr(i = 00, j = 00 | I, J) + \Pr(i = 01, j = 01 | I, J) + \\ &\quad \Pr(i = 10, j = 10 | I, J) + \Pr(i = 11, j = 11 | I, J) \\ &= i_{00}j_{00} + i_{01}j_{01} + i_{10}j_{10} + i_{11}j_{11}. \end{aligned}$$

Similarly, we can obtain $\Pr(v = 01 | I, J)$, $\Pr(v = 10 | I, J)$ and $\Pr(v = 11 | I, J)$. Then the output message vector is given by

$$\begin{aligned} CHK(I, J) &= (i_{00}j_{00} + i_{01}j_{01} + i_{10}j_{10} + i_{11}j_{11}, \\ &\quad i_{00}j_{01} + i_{01}j_{00} + i_{10}j_{11} + i_{11}j_{10}, \\ &\quad i_{00}j_{10} + i_{01}j_{00} + i_{01}j_{11} + i_{11}j_{01}, \\ &\quad i_{00}j_{11} + i_{01}j_{00} + i_{01}j_{10} + i_{10}j_{01}). \end{aligned} \quad (24)$$

Then we can apply $CHK(\cdot)$ to include more input variable nodes and obtain the final output message.

In this way, we use $CHK(\cdot)$ to obtain the output message from the check node to one variable node. Similarly, we can

Algorithm 1: BP algorithm for LDPC Jt-CDA.

Input: Number of users: M ; Received baseband signal in frequency domain: Y ; Check nodes list: C ; Variable nodes list: V .

Output: Arithmetic sum of source bits: \hat{S}^F .

```

1 Calculate messages out of evidence nodes  $\mathcal{M}(e_i \rightarrow v_i)$ 
  according to Eq. (23);
2 Initialize messages from variable nodes to associated
  check nodes:  $\mathcal{M}(v_i \rightarrow c_j) = \mathcal{M}(e_i \rightarrow v_i)$ ;
3 while  $iter < MAX-ITER$  do
4   for check nodes  $c_i$  in  $C$  do
5      $V_i \leftarrow$  variable nodes linked with  $c_i$ ;
6     for variable node  $v_j \in V_i$  do
7        $\mathcal{M}(c_i \rightarrow v_j) = CHK(\mathcal{M}(V_i \setminus v_j \rightarrow c_i))$ ;
8   for variable nodes  $v_i$  in  $V$  do
9      $C_i \leftarrow$  check nodes linked with  $v_i$ ;
10    for check node  $c_j \in C_i$  do
11       $\mathcal{M}(v_i \rightarrow c_j) = VAR(\mathcal{M}(e_i \rightarrow v_i), \mathcal{M}(C_i \setminus c_j \rightarrow v_i))$ ;
12 for variable nodes  $v_i$  in  $V$  do
13    $C_i \leftarrow$  check nodes linked with  $v_i$ ;
14    $P = p_{s_1 \dots s_M} = VAR(\mathcal{M}(e_i \rightarrow v_i), \mathcal{M}(C_i \rightarrow v_i))$ ;
15   Obtain the sum bit  $s$ :
16    $\arg \max_{s=s_1+\dots+s_M} \sum p_{s_1 \dots s_M}$ ;
17 Extract the SUM source bits  $\hat{S}^F$  in selected variable
  nodes (for systematic codes).
```

use $CHK(\cdot)$ to compute output messages from the check node to other variable nodes.

Update Equations on Variable Nodes: We use function $VAR(\cdot)$ to denote the update equation that computes output message \mathcal{M} going out of a variable node to a check node. Note that the output message is used as input to $CHK(\cdot)$. For multiple inputs, the computation of $VAR(\cdot)$ is also sequential. Fig. 7(c) shows an example of the updating process on a variable node v_1 . It is to compute $\mathcal{M}(v_1 \rightarrow c_1) = VAR(\mathcal{M}(e_1 \rightarrow v_1), \mathcal{M}(c_2 \rightarrow v_1), \mathcal{M}(c_3 \rightarrow v_1))$.

We focus on the minimal computation unit $VAR(I, J)$, where the message vectors of the two input information edges are $I = (i_{00}, i_{01}, i_{10}, i_{11})$ and $J = (j_{00}, j_{01}, j_{10}, j_{11})$. The probability that the composed temporary node v is 00 is obtained as

$$\begin{aligned}
Pr(v = 00|I, J) &= \frac{Pr(I, J|v = 00)Pr(v = 00)}{Pr(I, J)} \\
&= \frac{Pr(I|J, v = 00)Pr(J|v = 00)Pr(v = 00)}{Pr(I, J)} \\
&= \frac{Pr(I|v = 00)Pr(J|v = 00)Pr(v = 00)}{Pr(I, J)} \\
&= \frac{Pr(v = 00|I)Pr(v = 00|J)Pr(I)Pr(J)}{Pr(I, J)Pr(v = 00)} \\
&= \alpha_2 i_{00} j_{00},
\end{aligned} \tag{25}$$

where $\alpha_2 = 4 \frac{Pr(I)Pr(J)}{Pr(I, J)}$ and the two input messages are

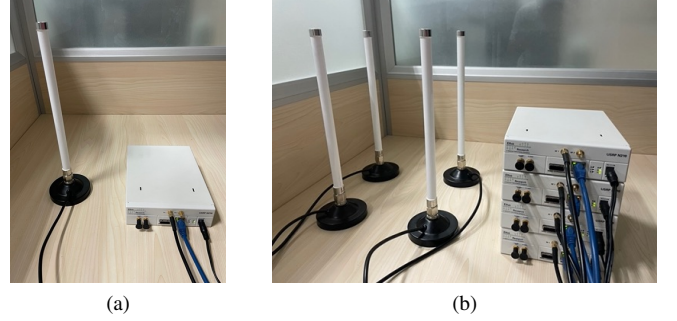


Fig. 8. A digital AirComp testbed on the USRP platform: (a) an edge parameter server; (b) four edge devices.

assumed to be independent given the value of the variable node, i.e., $Pr(I|J, v) = Pr(I|v)$.

In a similar way, we can obtain $Pr(v = 01|I, J)$, $Pr(v = 10|I, J)$ and $Pr(v = 11|I, J)$ as well. Thus, the output message at the variable node is

$$VAR(I, J) = \frac{1}{\beta_2} (i_{00}j_{00}, i_{01}j_{01}, i_{10}j_{10}, i_{11}j_{11}), \tag{26}$$

where β_2 is the sum of four probability for normalization. Note that β_2 does not include α_2 , since α_2 is a common factor that can be ignored during normalization.

In this way, we use $VAR(\cdot)$ to obtain the output message from the variable node to one check node. Similarly, we can use $VAR(\cdot)$ to compute output messages from the variable node to other check nodes.

Overall Algorithm: Algorithm 1 shows the BP algorithm for Jt-CDA in overall. Initially, we calculate the message vectors out of evidence nodes by received baseband signal Y . The message also initializes the message vectors from variables nodes to their associated check nodes. Then, following the update equations of Eq.(24) and Eq.(25), we can iterate the message passing process. After fixed rounds of iteration, we can calculate a probability vector in each variable node that represents the joint probability of each user's bit. Then, we can use the (4/3)-to-1 mapping to sum the probability for the SUM bit and determine the best SUM bit. For systematic LDPC codes, we can directly extract SUM bits in corresponding positions to get \hat{S}^F .

VI. EVALUATION

To test our digital AirComp systems in a practical environment, we use the USRP software-defined radio as the experiment platform. We implement a testbed using five USRP N210s with UHD driver 3.8.2 as shown in Fig. 8: four USRPs serve as edge devices, and one USRP serves as the parameter server. USRPs are equipped with the UBX daughterboards. All experiments are performed at 5.85GHz with a bandwidth 10MHz.

To support simultaneous transmissions, we use the implemented MAC protocol in [29]: a beacon triggers simultaneous transmissions; edge devices detect the beacon with precise hardware timestamp on USRP (on the order of ns), and

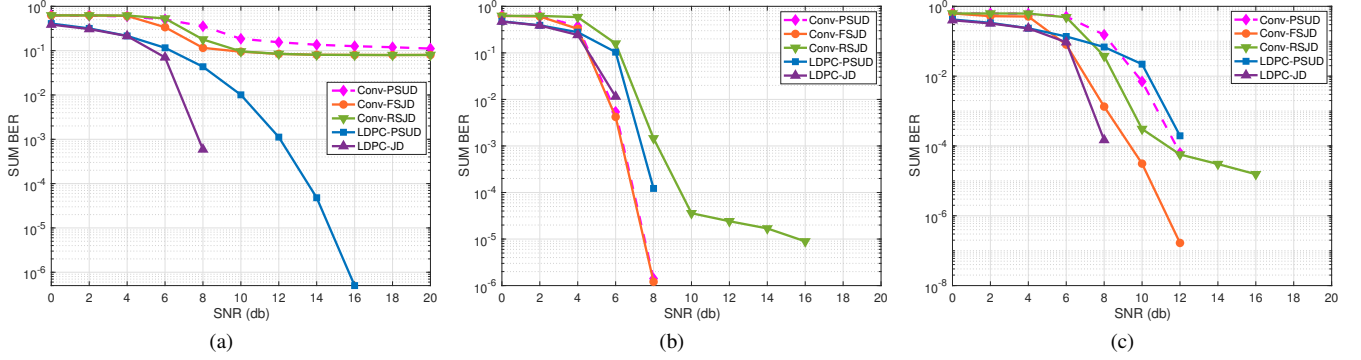


Fig. 9. SUM BER versus SNR under various phase settings in AWGN channel for two-user simultaneous transmissions: (a) perfectly aligned channel phase; (b) relative channel phase offset of $\pi/2$ radian; (c) relative channel phase offset of $\pi/4$ radian.

add sufficient large time that compensates the signal transfer time and the baseband processing time for transmission. Meanwhile, edge devices measure CFOs according to the beacon frame in the downlink, and perform phase precoding to reduce the CFO in the uplink. We measure the synchronization accuracy under the current hardware and protocol: TO is less than 2 samples, and CFO is within $[-2, +2]$ kHz.

We implement the transmitter and the receiver shown in Fig. 4 using GNU Radio 3.7.11. GNU Radio is a real-time signal processing framework written in C++ and Python. The core signal processing functionalities are all implemented in C++ for real-time operation. Each block in Fig. 4 is implemented as a GNU Radio block. Signal processing blocks are run in parallel using multiple threads, and are scheduled by the operating system using a default policy specified by GNU Radio. Each OFDM frame includes 500 data symbols. We adopt BPSK and 1/2 coding rate, and thus each OFDM frame has around 1500 Bytes, close to the maximal length of an Ethernet frame. For convolutional-coded AirComp, we adopt the code defined in WiFi standards with polynomials $(133_8, 171_8)$. For LDPC-coded AirComp, we adopt the block-structured code defined in WiFi standards with each block 1296 bits.

A. SUM BER Performance

Setup. We use the sum bit error rate (SUM BER) as a metric to evaluate the decoding performance of the proposed Jt-CDA decoders. For two devices, there are three possible SUM bit outcomes $\{0, 1, 2\}$, and any mismatch is treated as an error. We compare their SUM BER performance for different SNRs (less than 20 dB). For each SNR, we run 4000 times, and present the average result. SNR is defined as the reception power of the superimposed signal versus noise.

We consider the following proposed joint decoders: *Conv-FSJD*, *Conv-RSJD*, and *LDPC-JD*. *Conv-FSJD* and *Conv-RSJD* belong to *Conv-JD*. For *Conv-RSJD*, we consider $R=256, 1024, 4096$ for two-user, three-user and four-user simultaneous transmissions, respectively. Meanwhile, we also consider two parallel separate decoders (PSUD), *Conv-PSUD* and *LDPC-PSUD* for comparison. Different from joint decoders, PSUDs perform symbol-level probability marginalization to separate and decode each user's data, and thus also

have low implementation complexity. Appendix D shows their formal definitions.

To study the impact of relative phase offset on all Jt-CDA decoders, we consider the following ideal channel conditions with two-user simultaneous transmissions, and generate superimposed signals using simulations:

- 1) AWGN channel with unit amplitude and no relative channel phase offset.
- 2) AWGN channel with unit amplitude and relative channel phase offset of $\pi/2$ radian.
- 3) AWGN channel with unit amplitude and relative channel phase offset of $\pi/4$ radian.

We also consider an experimental realistic channel using USRPs. We use the aforementioned MAC protocol to trigger simultaneous transmissions, and log the over-the-air superimposed baseband signals. Then we run various decoders on the same signal offline. We control the transmission power of each user so that their reception powers are almost balanced (i.e., within 2dB) and emulate different SNRs. For the realistic channel, we consider two, three and four users.

Results. Fig. 9(a) shows the results of different decoders with no relative phase offset. Given four possible constellation points, two of them (i.e., $(+1, -1)$ and $(-1, +1)$) are almost overlapped, and are hard to differentiate. It may confuse decoders that need to classify them. We can see that *Conv-JDs* have an error floor even at high SNRs, and their performances are worse than *LDPC-JD* decoders. The key reason is that convolutional decoders rely on joint states, and inseparable constellation points would lead to a wrong survivor path and thus wrong bits. In contrast, *LDPC-JD* aggregates probability for SUM bit decision in the end despite inseparable constellation points during iterations. The poor performance of PSUDs is due to the treatment of the other user's signal as noise.

Fig. 9(b) shows the results of different decoders with a relative phase offset of $\pi/2$ radian. Four possible constellation points locate on the orthogonal coordinates and equally partition the complex plane into four equal decision regions. In this case, the performance of parallel decoders (PSUDs) is close to joint decoders (JDs), since signals are orthogonal. The performance of *Conv-FSJD* is close to *LDPC-JD*, since all constellations are separable for *Conv-FSJD*. However, due to the probability aggregation capability, *LDPC-JD* is still better

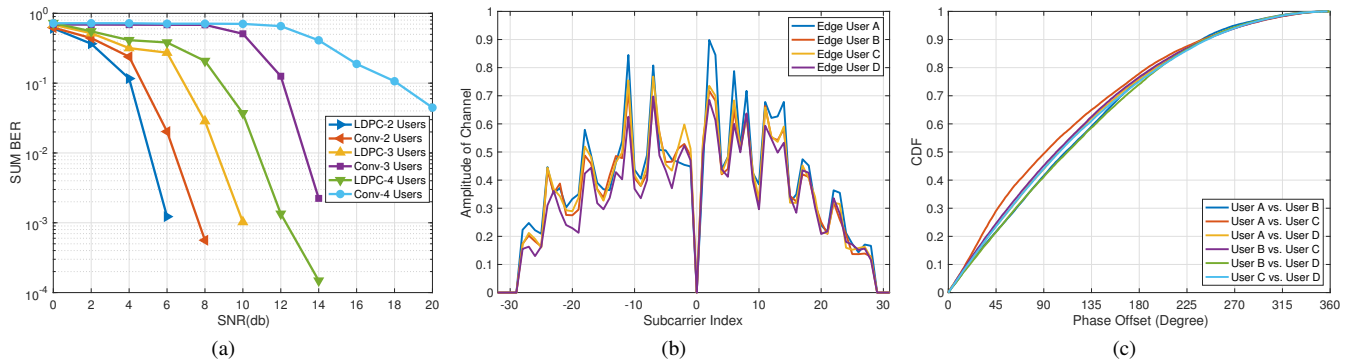


Fig. 10. (a) SUM SER versus SNR under experimental channel on USRP; (b) channel amplitude of transmission users in an experiment; (c) cumulative distribution function of relative phase offset between transmission users in an experiment.

than Conv-FSJD at $\text{SNR} \geq 6$ dB. We also observe that Conv-FSJD outperforms Conv-RSJD since RSJD loses some path information during decoding.

Fig. 9(c) shows the results of different decoders with a relative phase offset of $\pi/4$ radian. The performance of all decoders gets worse than that with relative phase offset $\pi/2$ radian, and is better than that without relative phase offset. Furthermore, LDPC-JD is less sensitive to the relative phase offset compared with Conv-JD. From Fig. 9(a), Fig. 9(b) and Fig. 9(c), we can observe that phase offset has a strong impact on the SUM bit decoding performance, and LDPC-JD is more robust than Conv-JD against various phase offsets.

Then we show the experiment results on USRPs. In the experiment, we only consider *Conv-RSJD* for practical convolutional-coded AirComp systems, since *Conv-FSJD* has unacceptable exponential states for more than two users. We consider *LDPC-JD* for LDPC-coded AirComp. A typical channel in our experiments is shown in Fig. 10(b) and Fig. 10(c). Fig. 10(b) shows the channel amplitudes of all subcarriers for different users. Fluctuating amplitudes in Fig. 10(b) indicate a multi-path channel for each user. Fig. 10(c) shows the cumulative distribution function of relative phase offset between any two users for subcarriers over all symbols. Due to the long packet duration with 1500 Bytes, the phase offset spreads over the whole plane, and the distribution is almost even although their initial phase offsets may be different. In overall, the realistic channel experiences all possible phase misalignments, and the decoder performance seems to be averaged over fixed phase misalignments as in Fig. 9.

Fig. 10(a) shows the SUM BER results of different decoders under a realistic channel. LDPC-JD outperforms Conv-RSJD for all SNRs given the same number of simultaneous transmissions in the realistic channel. The observation is consistent with results in simulated AWGN channels. Furthermore, the decoding performance of LDPC-JD degrades as the number of simultaneous transmissions increases. Given the 10^{-3} target for FL applications, two-user LDPC-coded AirComp works at $\text{SNR} \geq 6$ dB, three-user LDPC-coded AirComp works at $\text{SNR} \geq 10$ dB, and four-user LDPC-coded AirComp works at $\text{SNR} \geq 12$ dB.

B. Test Accuracy Performance

FEEL System Setup. Our FEEL system consists of a parameter server and forty edge devices. They communicate through the wireless medium under various AirComp systems. Digital AirComp systems follow the MAC protocol defined in Section III. We consider two FL applications: CIFAR-10 [26] and MNIST [27].

For the CIFAR-10 dataset, each edge device uses a convolution neural network (CNN) called ShuffleNet V2 network [30] to perform classification. CIFAR-10 contains 32×32 RGB color images in 10 categories. The size of the CIFAR-10 training dataset is 50,000 and the size of the testing dataset is 10,000. The training datasets are distributed in a non-iid manner: 1) we first equally distribute the 40,000 training images to 40 edge devices in random order; 2) the rest 10,000 images are sorted by labels and equally distributed to every device with size 250.

For the MNIST dataset, each edge device uses an artificial neural network (ANN) called Multiple-layer Perceptron [31] to perform classification. MNIST contains 28×28 images of handwriting numbers in 10 categories. The size of the MNIST training dataset is 60,000 and the size of the testing dataset is 10,000. The training datasets are also distributed in a non-iid manner.

Every parameter in the neural network is compressed using the probability quantization approach [28] with 8 bits. In each training iteration, the parameter server randomly selects some edge devices for training. We use test accuracy as the metric to evaluate the performance of different AirComp systems. The test accuracy is defined as the prediction accuracy of the global learned model on the test datasets.

In the following, we let the number of chosen edge devices be equal to the number of users that the digital AirComp decoder can support (i.e., $P = M$). The test accuracy results for the case $P \neq M$ are similar, so we only present results for the case $P = M$.

Communication System Setup. We adopt a trace-driven simulation approach to compare the test accuracy performance over SNRs. We compare digital AirComp with both traditional digital FL system that only allows one user to transmit, and analog AirComp. We adopt the SNR to SUM BER table measured in the realistic channel (including the single-user and

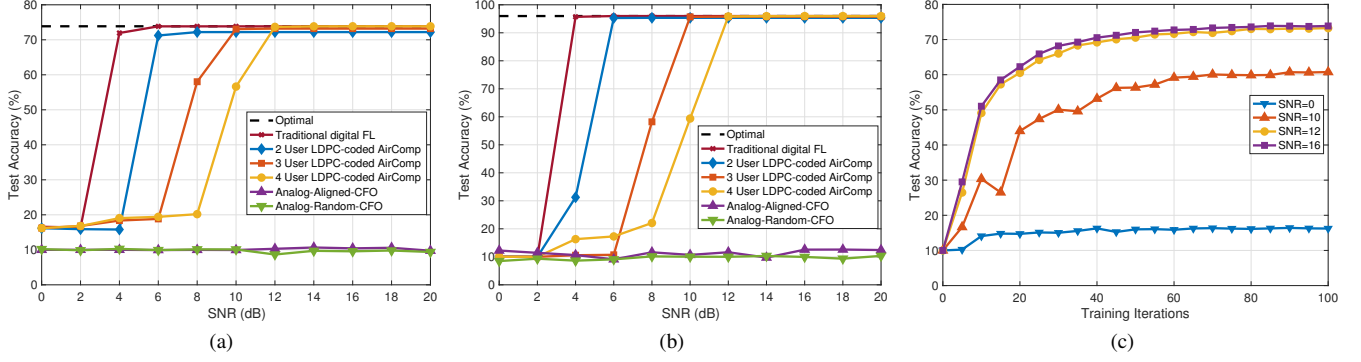


Fig. 11. Test accuracy performance of different AirComp systems under realistic channel: (a) test accuracy of CIFAR-10 with ShuffleNet-v2 for different AirComp systems; (b) test accuracy of MNIST with MLP for different AirComp systems; (c) test accuracy of CIFAR-10 over training iterations with different SNRs for four-user LDPC-coded AirComp.

multi-user systems). Then, we perform packet-level simulation to simulate the learning process where each frame has 500 data symbols, and the frame may have erroneous SUM bits. For digital AirComp, we only consider the LDPC-JD decoder given its superior performance over the Conv-JD decoder.

For analog AirComp [3], we assume an AWGN channel with perfect power alignment and time alignment, but with phase misalignment due to air-channel and CFO. For each learning instance, we first generate random air-channels and CFOs for all forty edge devices. Four edge devices are randomly chosen to participate in each training round, and they have different air-channels and CFOs. The final test accuracy results are averaged over 100 instances.

We compare digital AirComp with the following analog AirComp systems:

- 1) *Analog-Aligned-CFO*: It is the analog AirComp with precise precoding at the beginning, and with initial phases in all subcarriers aligned but having rotating phases caused by a random CFO within $[-350\text{Hz}, 350\text{Hz}]$ over symbols;
- 2) *Analog-Random-CFO*: It is the analog AirComp without precoding, and with initial phases in all subcarriers generated randomly due to air-channel and having rotating phases caused by a random CFO within $[-350\text{Hz}, 350\text{Hz}]$ over symbols.

Here the *Analog-Aligned-CFO* AirComp system is to emulate a channel-precoded system with perfect phase alignment at the beginning but with phase misalignment as time goes on. The *Analog-Random-CFO* AirComp system is to emulate a practical system without phase precoding.

In analog AirComp systems, the float-valued parameters are directly loaded on subcarriers. We enhance the scheme by allowing each subcarrier to transmit two parameters (in both I and Q planes). To keep the same protocol overhead for time synchronization, we assume the same frame duration as digital AirComp systems. Since analog AirComp does not need quantization, it leads to less overall data transmission time. To be fair, we allow an improved analog AirComp system² that transmits the same data several times so that the

overall transmission duration is the same with digital AirComp (i.e., 16 repeat transmissions for 8-bit quantization in digital AirComp), and picks the transmission where the aggregated data has the minimum mean squared error compared with the ground truth as the final result.

Results. Fig. 11(a) shows the test accuracy of CIFAR-10 with ShuffleNet-v2 for different systems. First, digital AirComp can realize almost the same performance as the ideal system (i.e., 73%) at high SNRs. The operating SNR regimes for different digital AirComp systems are consistent with results in Fig. 10(a). Second, analog AirComp without accurate phase precoding could not train the model successfully even at high SNRs³, although initial phases are aligned. The rotating phase offset in a frame induces a large aggregation error. Instead, digital AirComp could overcome the phase asynchrony, and achieve near-optimal learning performance. Third, the learning performance of traditional digital FL is better than digital AirComp, since it can approach the optimal performance at lower SNR. This phenomenon is as expected, since traditional digital FL can achieve the same BER at lower SNR. Fig. 11(b) shows similar results of MNIST with MLP for different systems. The above results reveal the tradeoff between reliability and throughput for various digital FL systems: we can choose a suitable digital FL scheme depending on the SNR ranges that the FEEL system operates.

Fig. 11(c) shows the test accuracy iteration results of CIFAR-10 for four-user digital AirComp systems in 100 training iterations. We can see that: 1) different SNRs have different convergence rate and results; 2) the learning process converges to the upper bound 73% test accuracy at $\text{SNR} \geq 12$ dB; 3) the learning process may not converge in a low SNR regime with test accuracy 10% as the lower bound. All results show that digital AirComp is more suitable for phase-asynchronous systems.

³The performance of analog AirComp is lower than the reported performance in our previous conference version [32]. The reason is that analog AirComp in [32] assumes the same phase offsets for all subcarriers and all symbols. The phase offsets are generated randomly. Therefore, good phase offsets are possible sometimes, leading to some learning improvement during iterations.

²We also try the analog AirComp system that uses the same duration to train the model with more rounds. However, the results are similar since the performance does not improve after convergence.

C. Real-time Performance

Implementation. We consider the real-time implementation of LDPC-coded AirComp due to its superior performance over convolutional-coded AirComp. Although the LDPC decoder utilizes the low-complexity BP algorithm for decoding, real-time implementation on the CPU remains a big challenge. Thanks to the structure of LDPC, parallel processing can be applied on check nodes and variable nodes in each iteration to accelerate the decoding. Moreover, parallel processing can also be applied on independent blocks for a long frame. Some previous work [33] has demonstrated superior decoding performance of single-user LDPC decoder on the graphics processing unit (GPU) hardware. Therefore, we also adopt GPU to implement LDPC Jt-CDA.

The digital AirComp receiver is implemented on a Ubuntu 18.04.6 LTS server with Intel Xeon Gold 5122 CPUs (four cores@3.6GHz), 64GB DDR4 DRAM, and an Nvidia GeForce RTX 2080 Ti GPU with CUDA 11.4. The LDPC decoding function is written in CUDA C, and is compiled to a dynamic library using *nvcc*. We modify an open-source single-user LDPC GPU decoder [34] to realize the multi-user joint decoder defined in Section V. There are 40 iterations during the LDPC decoding. Then we write a GNU Radio block in Python that loads the library during initialization and calls the decoding function when soft demodulation inputs of a complete frame are ready. The other blocks except the demodulation and decoding blocks are the same as in a real-time PNC system [35].

Setup. We first measure the *channel decoding time* that decodes the SUM bits using LDPC Jt-CDA. The block is executed for a complete frame, and thus we can add two timers in the decoding block to measure the processing time. Then we try to measure the *frame decoding time* that captures the processing time of all blocks in the receiver. However, it is challenging to measure the time in GNU Radio, since GNU Radio is a pipeline framework that runs in multiple threads, and there are queuing delays among blocks. If we simply compute the time gap between the time a frame is detected and the time bits are decoded, the frame decoding time will be over-estimated, because the time contains the queuing time. To character the processing time of the receiver, we use the *inter-packet interval* (IPI) in the transmitter to approximate the frame decoding time. In particular, we tune the IPI, and find the minimal IPI that the receiver is still stable. Here stable means that the inter-packet decoded time gap is equal to the inter-packet arrival time gap (i.e., IPI). The method has been adopted in measuring the real-time performance of a PNC system on USRP and GNU Radio in [35].

We also measure the *network throughput* in bits per second considering the protocol overhead. To improve the network throughput, we extend the designed MAC protocol with a burst transmission mode [29], where a trigger frame initiates a burst transmission consisting of multiple frames from each user. The transmission gap is equal to the IPI. The number of frames in a burst is chosen so that the last frame remains time synchronized despite clock drifts in each user. Note that the burst MAC reduces the synchronization overhead, since one

TABLE I
REAL-TIME PERFORMANCE OF THE LDPC-CODED AIRCOMP SYSTEM WITH TWO, THREE, AND FOUR CONCURRENT USERS. CD TIME MEANS CHANNEL DECODING (I.E., Jt-CDA) TIME, AND FD TIME MEANS FRAME DECODING TIME (I.E., IPI).

	CD Time	FD Time	Throughput
Two-User	17.2±1.6ms	20ms	1.19Mbps
Three-User	24.5±2.0ms	30ms	1.18Mbps
Four-User	38.9±3.1ms	40ms	1.16Mbps

trigger frame synchronizes multiple data frames. To take the number of concurrent transmissions into account, we define the network throughput as the product of the measured throughput in the edge server and the number of concurrent transmissions.

We measure and fix the IPI (i.e., frame decoding time) beforehand for different numbers of simultaneous transmissions. We also measure and fix the burst duration of 1s. Then we measure the throughput 30 times with each measurement 500s, and present the average result. Meanwhile, we measure the channel decoding time for each frame, and present the average result with standard deviation over all measurement. In each measurement, there are 25,000, 16,500, 12,500 frames for two-user, three-user, and four-user digital AirComp, respectively.

Results. Table I shows the overall real-time performance results. The Jt-CDA time is almost linear to the number of simultaneous transmissions. It is consistent with Algorithm 1 since the complexity increases with the message vector length. The frame decoding time is larger than the Jt-CDA time, since Jt-CDA is just only one step of the receiver. The network throughput is about 1.19 Mbps for two-user LDPC-coded AirComp, and 1.16 Mbps for four-user LDPC-coded AirComp. Given that the frame duration is around 4ms, and the physical throughput is 3Mbps for our system (i.e., around 500 data symbols, 10MHz bandwidth, BPSK and 1/2 coding rate), the achievable throughput almost reaches the upper bound for the two-user LDPC-coded AirComp (i.e., $4/20 \times 3\text{Mbps} \times 2 = 1.2\text{Mbps}$), meaning negligible protocol overhead. Although the achievable throughput is lower than the physical throughput, we note that the bottleneck is on the Jt-CDA block, and further optimization is possible with more powerful GPU.

VII. DISCUSSION

Real-time FEEL. We have presented a real-time digital AirComp system for FEEL. However, it is still far from a complete and real-time FEEL system. Although communication is a bottleneck that restricts the development of FEEL [1], device heterogeneity with different local training time and data heterogeneity with different data distribution also have a strong impact on the FEEL performance, and thus user selection that considers device and data heterogeneity has been widely studied in FEEL. Moreover, there exists communication heterogeneity in FEEL with digital AirComp: edge devices have heterogeneous channels and CFOs, and different combinations of simultaneous transmissions have different communication performances. Therefore, one important direction of future work is to investigate the user selection strategy to improve the overall learning performance, and develop a real-time FEEL system.

Joint Source-Channel Coding Design. We believe that channel codes are an indispensable part of any communication system supporting wireless FL, since bit errors are inevitable due to noise, frequency-selective fading and adverse phase offsets. However, as shown in Figure 3 in Section II-D, FL applications may not require a high degree of reliability, and may allow highly-compressed source codes for communication. The joint design of source codes and channel codes is an interesting direction for future work.

VIII. RELATED WORK

AirComp Systems. Over-the-air computation in multiple-access channels is first studied from an information-theoretic perspective [36], [37]. They show that joint communication and computation can be more efficient than separating communication from computation in terms of function computation rate. Recently, many AirComp systems [2]–[12] for FEEL have been proposed. However, all of them are analog AirComp systems, and require accurate power and phase synchronization.

Abari, Rahul and Katabi [38] try to realize analog AirComp for the first time. There are some analyses in [38], but the real system is not demonstrated. Guo et al., [18] provide the first implementation of two-user analog AirComp. It leverages dedicated frame/protocol design and accurate timing tracking to compensate for phase misalignment. Sahin [24] provides a new five-user analog AirComp implementation for SignSGD-based FL. Different from previous analog AirComp systems, it uses two subcarriers for the positive vote and the negative vote, and compares their energy for decisions based on the majority vote rule. The positive results are analyzed using statistical channel, and some preliminary results are presented via SDR experiments. In contrast, we take another way to deal with phase asynchrony following the standard physical layer, and demonstrate a four-user digital AirComp system.

Some analog AirComp systems are proposed to handle phase-misaligned transmissions. Goldenbaum et al., [39], [40] design and implement an analog-modulated AirComp system that only needs coarse symbol-level synchronization, and leverages direct spread spectrum sequences with random phases for modulation. Shao et al., [41], [42] leverages oversampling and sum-product ML estimators to estimate the arithmetic sum under symbol misalignment and phase misalignment. However, their designs only work for time-domain narrowband systems, and cannot directly apply to the frequency-domain OFDM broadband system.

Network MIMO Systems. Network MIMO systems are envisioned to improve network capacity, where distributed APs in different locations form a virtual big AP for simultaneous transmissions and receptions to several users (i.e., distributed MU-MIMO). The most important requirement is to maintain frequency synchronization among distributed APs during transmissions. Otherwise, CFOs between APs and users rotate the channel phases, and invalidate the orthogonality of inter-user signals which is achieved by the precoding matrix computed from the initial channel.

Many solutions have been proposed. However, all these solutions require extra resources and dedicated protocol de-

sign. Out-of-band radio [20], out-of-band power line infrastructure [25], out-of-band Ethernet infrastructure [19], [22], and dedicated hardware [21] are required to realize frequency synchronization. Moreover, receivers need to provide feedback information on the initial subcarrier phases for interference-free transmissions. Phase-aligned transmissions are not easy for random-access WiFi networks if low overhead is desired. Therefore, we target low-cost multi-user OFDM systems, and design signal processing algorithms for digital AirComp.

IX. CONCLUSION

This paper present the first digital AirComp system using OFDM modulation for phase-asynchronous FEEL systems. With a simple multi-user non-orthogonal channel access protocol with negligible overhead, our system leverages joint channel decoding and aggregation decoders tailored for convolutional and LDPC codes to overcome phase asynchrony. We implement a real-time digital AirComp system with up to four users on a software-defined radio platform. Trace-driven simulation results show that digital AirComp can achieve near-optimal learning performance under practical phase-asynchronous scenarios, while analog AirComp fails to do so even with high SNRs.

Moving forward, our digital AirComp system can be extended in the following directions: a) SUM bit-optimal decoder design for Turbo-coded AirComp, b) joint source-channel coding designs to directly improve the application performance, and c) user selection in FEEL with digital AirComp for improved network performance.

APPENDIX A

RELATIONSHIP BETWEEN BER AND SUM BER

For the multi-user digital AirComp system, the SUM bit is the arithmetic summation of all users' original bits, and thus the SUM bit error is different from the traditional bit error. In particular, for the n -user digital AirComp system, the corresponding mapping relationship of the original bits and the SUM bit is given as follows

$$\underbrace{\{0, 1\} \times \cdots \times \{0, 1\}}_n \rightarrow \{0, 1, \cdots, n\}$$

with the mapping function from size 2^n to size $(n + 1)$. The unequal size makes them different.

Let the traditional source bit error rate (BER) be α , and the SUM bit error rate (SUM BER) be β . This appendix analyzes their relationship. We assume the error in each bit is independent (channel codes can be used to make the assumption valid).

Take the two-user digital AirComp system for example. There are four kinds of two-users' bit sequences $\{00, 01, 10, 11\}$, while there are three kinds of SUM bits $\{0, 1, 2\}$. If the original bit sequence is 00 or 11, any bit error causes the SUM bit different. Therefore, the SUM bit error probability is given by $\beta = 1 - (1 - \alpha)^2$. If the original bit sequence is 01 or 10, one-bit error causes the SUM bit different. Thus, the SUM bit error probability is given by $\beta = 1 - (1 - \alpha)^2 - \alpha^2$. Given the equal probability of

sequences $\{00, 01, 10, 11\}$ (i.e., 1/4), the SUM bit error rate is given by $1 - (1 - \alpha)^2 - \frac{\alpha^2}{2}$. Obviously, $\beta < 1 - (1 - \alpha)^2$. Given α is typically much smaller than 10^{-3} in a communication system, $\beta > \alpha$ (it holds for $\alpha < 2/3$).

For the n -user digital AirComp system, we first consider two special multi-user' bit sequence with length n $\{0 \dots 0, 1 \dots 1\}$, their SUM BER is given by

$$\beta_{\underbrace{0 \dots 0}_n} = \beta_{\underbrace{1 \dots 1}_n} = 1 - (1 - \alpha)^n.$$

For other multi-user' bit sequences, we consider a particular sequence s with n_0 zeros and $n - n_0$ ones ($0 < n_0 < n$). The SUM BER is given by

$$\beta_s = 1 - (1 - \alpha)^n - \sum_{e=1}^{\min\{n_0, n-n_0\}} \alpha^{2e}. \quad (27)$$

Then the overall SUM BER is given by

$$\beta = 1 - (1 - \alpha)^n - \frac{1}{2^n} \sum_{n_0=1}^{n-1} \binom{n}{n_0} \sum_{e=1}^{\min\{n_0, n-n_0\}} \alpha^{2e}. \quad (28)$$

That is, BER α and SUM BER β are one-to-one correspondence. Obviously, $\beta < 1 - (1 - \alpha)^n$. Then we are to prove $\beta > \alpha$ for $0 < \alpha < 10^{-3}$.

Given $\min\{n_0, n - n_0\} \leq n/2$, we have

$$\sum_{e=1}^{\min\{n_0, n-n_0\}} \alpha^{2e} \leq \sum_{e=1}^{n/2} \alpha^{2e} = \alpha^2 \frac{1 - \alpha^n}{1 - \alpha^2} < \alpha. \quad (29)$$

The last inequality holds since $\alpha + \alpha^2 < 1 + \alpha^{n+1}$ for $0 < \alpha < 10^{-3}$. Then Eq. (28) becomes

$$\beta > 1 - (1 - \alpha)^n - \alpha. \quad (30)$$

Let $f(n) = 1 - (1 - \alpha)^n - 2\alpha$. Then we are to prove $f(n) > 0$ for $n \geq 3$ by induction, where $0 < \alpha < 10^{-3}$.

- For $n = 3$, $f(3) = 1 - (1 - \alpha)^3 - 2\alpha = \alpha^3 - 3\alpha^2 + \alpha = \alpha(\alpha^2 - 3\alpha + 1)$. Let $g(\alpha) = \alpha^2 - 3\alpha + 1$. It is easy to check that $g(\alpha)$ is a monotonically decreasing function for $\alpha \in (0, 10^{-3})$. Thus, $g(\alpha) > 10^{-6} - 3 \cdot 10^{-3} + 1 > 0$, and $f(3) > 0$ for $\alpha \in (0, 10^{-3})$.
- We assume $f(n) = 1 - (1 - \alpha)^n - 2\alpha > 0$ holds.
- We consider $f(n+1)$:

$$\begin{aligned} f(n+1) &= 1 - (1 - \alpha)^{n+1} - 2\alpha \\ &= (1 - 2\alpha) - (1 - \alpha)^{n+1} \\ &> (1 - \alpha)^n - (1 - \alpha)^{n+1} > 0, \end{aligned} \quad (31)$$

where $(1 - 2\alpha) > (1 - \alpha)^n$ holds due to the assumption for n .

Overall, SUM BER is larger than BER (i.e., $\beta > \alpha$), meaning the SUM bit indeed differs from the traditional bit.

APPENDIX B PROOFS OF CONVERGENCE ANALYSIS

A. Proof of Lemma 1

Proof. First,

$$\begin{aligned} \mathbb{E}(B(\Delta w_n(t))) &= \mathbb{E}[Q(\Delta w_n(t)) + X(\Delta w_n(t))] \\ &= \mathbb{E}[Q(\Delta w_n(t))] + \mathbb{E}[X(\Delta w_n(t))] \\ &\stackrel{(a)}{=} \mathbb{E}[Q(\Delta w_n(t))] \\ &\stackrel{(b)}{=} \Delta w_n(t), \end{aligned} \quad (32)$$

where (a) holds because of Eq. (13) in Lemma 1, and (b) holds due to Lemma 1 in [14]. Then,

$$\begin{aligned} &\mathbb{E}[\|B(\Delta w_n(t)) - \Delta w_n(t)\|_2^2] \\ &= \mathbb{E}[\|Q(\Delta w_n(t)) + X(\Delta w_n(t)) - \Delta w_n(t)\|_2^2] \\ &= \mathbb{E}[\|Q(\Delta w_n(t)) - \Delta w_n(t)\|_2^2] + \mathbb{E}[\|X(\Delta w_n(t))\|_2^2] \\ &\quad + 2\mathbb{E}[\langle Q(\Delta w_n(t)) - \Delta w_n(t), X(\Delta w_n(t)) \rangle] \\ &\stackrel{(c)}{=} \mathbb{E}[\|Q(\Delta w_n(t)) - \Delta w_n(t)\|_2^2] + \mathbb{E}[\|X(\Delta w_n(t))\|_2^2] \\ &\stackrel{(d)}{\leq} J_n^2(t) + K_n(t), \end{aligned} \quad (33)$$

where (c) holds because

$$\mathbb{E}[\langle Q(\Delta w_n(t)) - \Delta w_n(t), X(\Delta w_n(t)) \rangle] = 0,$$

which is due to (b) in Eq. (32).

We divide it into two parts to prove that (d) holds. First, we have the following inequality in [14]:

$$\mathbb{E}[\|Q(\Delta w_n(t)) - \Delta w_n(t)\|_2^2] \leq \frac{\delta_n^2(t)}{(2^{B_n(t)} - 1)^2} = J_n^2(t). \quad (34)$$

Second, we can have:

$$\begin{aligned} &\mathbb{E}[\|X(\Delta w_n(t))\|_2^2] \\ &= \frac{\alpha}{2} \sum_{i=0}^{B_n(t)-1} \left(\left(\frac{4\delta_n(t)^2}{d(2^{B_n(t)} - 1)} \times 2^i \times 1 \right)^2 + \left(\frac{4\delta_n(t)^2}{d(2^{B_n(t)} - 1)} \times 2^i \times (-1) \right)^2 \right) \\ &= \alpha \left(\frac{4\delta_n(t)^2}{d(2^{B_n(t)} - 1)} \right)^2 \sum_{i=0}^{B_n(t)-1} 2^{2i} \\ &= \alpha \left(\frac{4\delta_n(t)^2}{d(2^{B_n(t)} - 1)} \right)^2 \times \frac{1 - 4^{B_n(t)}}{1 - 4} \\ &\triangleq K_n(t). \end{aligned} \quad (35)$$

The proof is completed. \square

B. Proof of Theorem 1

Proof. We follow the method in [14]. We first introduce the lossless model at the $(t+1)$ -th communication round as

$$\hat{w}(t+1) = w(t) + \frac{1}{N} \sum_{n=1}^N p_n \Delta w_n(t). \quad (36)$$

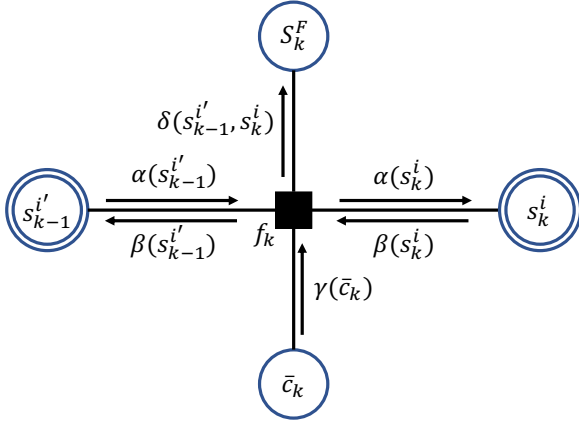


Fig. 12. The passed messages around a factor node during the operation of the BCJR algorithm.

With Eq. (36), we have

$$\begin{aligned}
 & \|w(t+1) - w^*\|_2^2 \\
 &= \|w(t+1) - \hat{w}(t+1) + \hat{w}(t+1) - w^*\|_2^2 \\
 &= \|w(t+1) - \hat{w}(t+1)\|_2^2 + \|\hat{w}(t+1) - w^*\|_2^2 \\
 &\quad + 2\langle w(t+1) - \hat{w}(t+1), \hat{w}(t+1) - w^* \rangle.
 \end{aligned} \tag{37}$$

We split the equation into three parts and analyze their property separately. The second part and the third part are the same with [14]. So we only prove the first part.

For the first part, we can have

$$\begin{aligned}
 & \mathbb{E} [\|w(t+1) - \hat{w}(t+1)\|_2^2] \\
 &= \mathbb{E} \left[\left\| \sum_{n=1}^N p_n (Q(\Delta w_n(t)) + X(\Delta w_n(t)) - \Delta w_n(t)) \right\|_2^2 \right] \\
 &\leq \sum_{n=1}^N p_n \mathbb{E} [\|Q(\Delta w_n(t)) + X(\Delta w_n(t)) - \Delta w_n(t)\|_2^2] \\
 &\leq \sum_{n=1}^N p_n (J_n^2(t) + K_n(t)).
 \end{aligned} \tag{38}$$

The first inequality follows from the convexity of $\|\cdot\|_2^2$ and $\sum_{n=1}^N p_n = 1$, and the second inequality follows from the Lemma 1.

Based on the above analysis, we can rewrite Eq. (37) as

$$\begin{aligned}
 \mathbb{E} [\|w(t+1) - w^*\|_2^2] &\leq (1 - \eta(t)\mu) \mathbb{E} [\|w(t) - w^*\|_2^2] \\
 &\quad + \eta(t)^2 U + \sum_{n=1}^N p_n (J_n^2(t) + K_n(t)).
 \end{aligned} \tag{39}$$

Then following the same proof in [14], Theorem 1 is proved. \square

APPENDIX C

BIT-OPTIMAL CONVOLUTIONAL DECODER

The goal of bit-optimal convolutional decoder is to find the maximum-likelihood (ML) SUM bit. The Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm is the widely used algorithm for bit-optimal decoding. It is a probabilistic iterative decoding

algorithm that utilizes message passing to compute the most probable original message sequence based on the received sequence. The core idea of the BCJR algorithm is to use a recursive algorithm to calculate the state transition metrics between each state by computing the branch metrics, and then calculate the probabilities of each state associated with the past, the present and the future of received signal. We modify the BCJR algorithm to support SUM bit decoding. In particular, a posterior probability (APP) of the SUM bits is computed based on these three probabilities.

A. Algorithm Design

Similar to the design for codeword-optimal convolutional decoder, we adopt the full-state joint trellis or the reduced-state joint trellis for two-user digital AirComp for bit-optimal convolutional decoder. However, for more than two users, we adopt the reduced-state joint trellis due to the exponential number of joint states.

Bit-optimal convolutional decoder passes probability message between states. Fig. 12 shows the message transmission graph between two states, where $s_{k-1}^{i'}$ represents the i' -th state in the $(k-1)$ -th stage and s_k^i represents the i -th state in the k -th stage. α , β and γ represent messages associated with the past, the present and the future, which can be calculated by forward and backward recursion. $\alpha(s_{k-1}^{i'})$ represents the forward probability of state $s_{k-1}^{i'}$. $\beta(s_{k-1}^{i'})$ represents the backward probability of state $s_{k-1}^{i'}$. $\gamma(s_k)$ represents the message from received signal Y . f_k is a virtual node, which validates the state transition of the trellis. If $f_k = 1$, it represents a valid transmission, and $f_k = 0$ otherwise. $\delta(\bar{u}_k)$ records the probabilities of different SUM bit cases. Overall, BCJR decoding algorithm consists of four steps: initialization, forward and backward recursion, and SUM bit decision. We will now explain them in details.

Initialization: We consider reduced state joint trellis as a cycle-free Tanner graph. Our sum-product algorithm starts at the first and last stages. Since we use the zero-tail convolutional code, the initial and terminal states of different edge device's convolutional encoders are zero state. Therefore, the forward probabilities α and backward probabilities β in different states can be initialized as:

$$\alpha(s_0^i) = \begin{cases} 1, & \text{if } i = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{40}$$

and

$$\beta(s_K^i) = \begin{cases} 1, & \text{if } i = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{41}$$

where i is the state index.

Forward and Backward Recursion: We first perform forward recursion from the 0-th stage to the K -th stage to calculate α and γ , and then perform backward recursion from the K -th stage to the 0-th stage to calculate β and δ .

We first introduce the forward recursion. In the k -th stage, the branch metric $\gamma(\bar{c}_k)$ is the difference between the received signal and the expected signal given the current state.

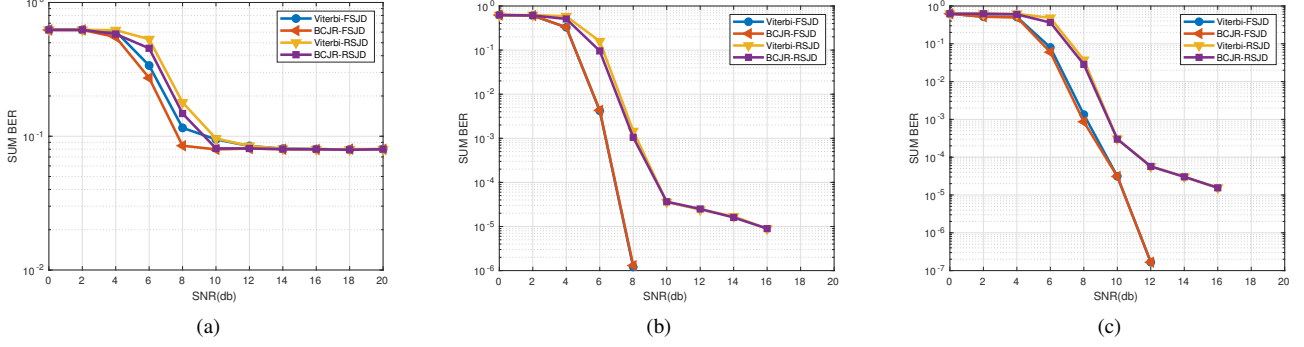


Fig. 13. SUM BER versus SNR results for the BCJR and Viterbi algorithms under various phase settings for two-user convolutional-coded AirComp: (a) perfectly aligned channel phase; (b) relative channel phase offset of $\pi/2$ radian; (c) relative channel phase offset of $\pi/4$ radian.

Assuming BPSK modulation with a code rate of 1/2, we obtain the encoded bits $\bar{C}_k = (C_{k,1}^A, C_{k,2}^A, C_{k,1}^B, C_{k,2}^B)$, where these channel-encoded bits correspond to the BPSK-modulated symbols $X^A[k] = (X_{2k-1}^A, X_{2k}^A)$ and $X^B[k] = (X_{2k-1}^B, X_{2k}^B)$. Therefore, the branch metrics can be represented as

$$\gamma(\bar{c}_k) \propto \prod_{n=2k-1}^{2k} \exp\left(-\frac{\|Y[n] - H^A[n]X^A[n] - H^B[n]X^B[n]\|^2}{2\sigma^2}\right), \quad (42)$$

which is actually the summation of the two likelihood probabilities of the encoded-bit pairs generated by source bits S_k^A and S_k^B , and $2\sigma^2$ is the variance of noise.

After calculating the branch metrics, we can compute the forward probabilities α between linked states. As for the i -th state in the k -th stage, the forward probability $\alpha(s_k^i)$ can be represented as

$$\alpha(s_k^i) = \sum_{\mathbf{L}\{i\}} f_k(s_{k-1}^{i'}, S_k^F, \bar{c}_k, s_k^i) \alpha(s_{k-1}^{i'}) \gamma(\bar{c}_k), \quad (43)$$

where $\mathbf{L}\{i\}$ is a set of states in the $(k-1)$ -th stage linked with the i -th state in the k -th stage.

Then, we perform backward recursion from K -th stage to 0-th stage. The backward probability $\beta(s_k^i)$ of the i -th state in the k -th stage can be represented as:

$$\beta(s_{k-1}^{i'}) = \sum_{i \in \mathbf{L}\{i'\}} f_k(s_{k-1}^{i'}, S_k^F, \bar{c}_k, s_k^i) \beta(s_k^i) \gamma(\bar{c}_k), \quad (44)$$

where $\mathbf{L}\{i'\}$ is a set of states in the k -th stage linked with the i' -th state in the $(k-1)$ -th stage.

To decode the SUM bits, decoder requires the APP of different SUM bit result S_k^F in every stage, which is determined by the APP of every state. We add the calculation of APP in backward recursion to reduce the decoding complexity. Specifically, for every linked state pairs $s_{k-1}^{i'}$ and s_k^i , we calculate the $\delta(s_{k-1}^{i'}, s_k^i)$ after β calculation. The APP $\delta(s_{k-1}^{i'}, s_k^i)$ can be represented as

$$\delta(s_{k-1}^{i'}, s_k^i) = f_k(s_{k-1}^{i'}, \bar{u}_k, \bar{c}_k, s_k^i) \alpha(s_{k-1}^{i'}) \gamma(\bar{c}_k) \beta(s_k^i). \quad (45)$$

SUM Bit Decision: Notice that we need APP of a stage instead of one state to determine the SUM bit. Therefore, we

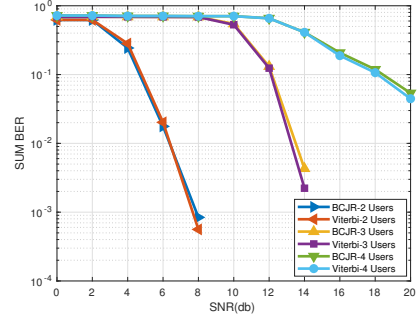


Fig. 14. SUM BER versus SNR results for the BCJR and Viterbi algorithms under realistic channel on USRP.

sum up $\delta(s_{k-1}^{i'}, s_k^i)$ of all possible linked state pairs and obtain the probabilities of SUM bit results. The SUM bit decision can be represented as

$$S_k^F = \arg \max \sum_{s_{k-1}^{i'}, s_k^i} \delta(s_{k-1}^{i'}, s_k^i). \quad (46)$$

We choose the SUM bit with the largest probability as the final decoding result.

B. Results

We first perform simulations to investigate the decoding performance of the designed BCJR decoder. The simulation conditions are identical to those specified in Section 6.1. We conduct separate tests for relative channel phase offset of 0, $\pi/2$ and $\pi/4$ radian. FSJD retains all path metrics and states in each iteration, while RSJD selectively retains the 256 states with the minimum path metrics in each iteration. The forward recursion in the BCJR algorithm is performed synchronously with the path metric calculation in the Viterbi algorithm. BCJR-RSJD utilizes the 256 states recorded by Viterbi-RSJD in each iteration for the forward recursion calculation. After completing all iterations, BCJR-RSJD utilizes the recorded states to perform backward recursion and calculate the posterior probabilities. Fig. 13 shows the the decoding performance of the BCJR and Viterbi algorithms in the case of two users. In the range of 4-10 dB, the simulation results show that BCJR is slightly better than Viterbi.

Then we perform experiments to investigate the decoding performance of the designed BCJR decoder under realistic channels. Fig. 14 shows the SUM BER results of BCJR and Viterbi for different decoders under a realistic channel measured on USRP. The data is identical to that specified in Section 6.1. Similar to the simulation results, BCJR is slightly better than Viterbi using the experiment data.

The above results turn out to be negative for the SUM bit-optimal decoder for convolutional codes. The proposed BCJR algorithm operates in the same trellis as the Viterbi algorithms (i.e., the approximated SUM codeword-optimal decoders), but it has a higher computation complexity due to message passing. However, their performances are similar. The phenomenon may be explained similar to the single-user communication system: the maximum a posteriori probability (MAP) decoder without a-priori information does not provide any performance gain against the maximum likelihood decoder (i.e., the Viterbi algorithm). The MAP decoder shows superior performance in the iterative decoder such as Turbo codes. We will study the decoding algorithm for Turbo-coded AirComp in the future.

APPENDIX D

PARALLEL SINGLE-USER DECODERS (PSUD)

Parallel single-user decoder (PSUD) is another type of channel decoding and aggregation approach that leverages the multi-user decoding (MUD) technique. Different from joint decoders, PSUDs decode each device's data separately. Take two-user simultaneous transmissions for example. We first decouple the superimposed signal in each subcarrier to each device's symbol decoding likelihood

$$Pr(Y[n]|C^A[n]) = \sum_{C^B[n]} Pr(Y[n]|C^A[n], C^B[n]), \quad (47)$$

and

$$Pr(Y[n]|C^B[n]) = \sum_{C^A[n]} Pr(Y[n]|C^A[n], C^B[n]). \quad (48)$$

The decoupled symbol likelihoods are fed into two conventional single-user decoders to find the codeword-optimal codewords for convolutional codes or the bit-optimal codewords for LDPC codes. Then codewords are mapped to source bits for each user, and SUM bits can be computed.

REFERENCES

- [1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*. JMLR, 2017, pp. 1273–1282.
- [2] M. M. Amiri and D. Gündüz, "Over-the-air machine learning at the wireless edge," in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2019, pp. 1–5.
- [3] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 491–506, 2019.
- [4] H. Xing, O. Simeone, and S. Bi, "Decentralized federated learning via SGD over wireless D2D networks," in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2020, pp. 1–5.
- [5] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, 2020.
- [6] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2155–2169, 2020.
- [7] G. Zhu, Y. Du, D. Gündüz, and K. Huang, "One-bit over-the-air aggregation for communication-efficient federated edge learning: Design and convergence analysis," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 2120–2135, 2020.
- [8] T. Sery, N. Shlezinger, K. Cohen, and Y. C. Eldar, "Over-the-air federated learning from heterogeneous data," *IEEE Transactions on Signal Processing*, vol. 69, pp. 3796–3811, 2021.
- [9] Z. Wang, J. Qiu, Y. Zhou, Y. Shi, L. Fu, W. Chen, and K. B. Letaief, "Federated learning via intelligent reflecting surface," *IEEE Transactions on Wireless Communications*, vol. 21, no. 2, pp. 808–822, 2022.
- [10] Y. Shao, S. C. Liew, and D. Gündüz, "Denosing noisy neural networks: A bayesian approach with compensation," *arXiv preprint arXiv:2105.10699*, 2021.
- [11] Z. Wang, Y. Zhou, Y. Shi, and W. Zhuang, "Interference management for over-the-air federated learning in multi-cell wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 8, pp. 2361–2377, 2022.
- [12] Y. Zou, Z. Wang, X. Chen, H. Zhou, and Y. Zhou, "Knowledge-guided learning for transceiver design in over-the-air federated learning," *IEEE Transactions on Wireless Communications*, vol. 22, no. 1, pp. 270–285, 2022.
- [13] Y. Tan, S. C. Liew, and T. Huang, "Mobile lattice-coded physical-layer network coding with practical channel alignment," *IEEE Transactions on Mobile Computing*, vol. 17, no. 8, pp. 1908–1923, 2018.
- [14] P. S. Bouzinis, P. D. Diamantoulakis, and G. K. Karagiannidis, "Wireless quantized federated learning: A joint computation and communication design," *arXiv preprint arXiv:2203.05878*, 2022.
- [15] (2022) The universal software radio peripheral USRP software defined radio device. [Online]. Available: <https://www.ettus.com/products/>
- [16] (2022) GNU Radio – the free and open software radio ecosystem. [Online]. Available: <https://github.com/gnuradio/gnuradio>
- [17] (2021) IEEE 802.11ax-2021. [Online]. Available: <https://standards.ieee.org/ieee/802.11ax/7180/>
- [18] H. Guo, Y. Zhu, H. Ma, V. K. N. Lau, K. Huang, X. Li, H. Nong, and M. Zhou, "Over-the-air aggregation for federated learning: Waveform superposition and prototype validation," *Journal of Communications and Information Networks*, vol. 6, no. 4, pp. 429–442, 2021.
- [19] H. S. Rahul, S. Kumar, and D. Katabi, "JMB: scaling wireless capacity with user demands," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 235–246, 2012.
- [20] H. V. Balan, R. Rogalin, A. Michaloliakos, K. Psounis, and G. Caire, "AirSync: Enabling distributed multiuser MIMO with full spatial multiplexing," *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 1681–1695, 2013.
- [21] O. Abari, H. Rahul, D. Katabi, and M. Pant, "Airshare: Distributed coherent transmission made seamless," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 1742–1750.
- [22] T. Wang, Q. Yang, K. Tan, J. Zhang, S. C. Liew, and S. Zhang, "DCAP: Improving the capacity of wifi networks with distributed cooperative access points," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 320–333, 2017.
- [23] (2022) LC-XO Kit. [Online]. Available: https://www.jackson-labs.com/index.php/products/lc_xo_plus_kit
- [24] A. Şahin, "A demonstration of over-the-air computation for federated edge learning," in *2022 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2022, pp. 1821–1827.
- [25] V. Yenamandra and K. Srinivasan, "Vidut: Exploiting power line infrastructure for enterprise wireless networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 595–606, 2014.
- [26] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," *Technical report*, 2009.
- [27] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.
- [28] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *International conference on machine learning*. PMLR, 2017, pp. 3329–3337.
- [29] L. Lu, L. You, Q. Yang, T. Wang, M. Zhang, S. Zhang, and S. C. Liew, "Real-time implementation of physical-layer network coding," in *Proceedings of the second workshop on Software radio implementation forum*, 2013, pp. 71–76.
- [30] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 116–131.

- [31] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.
- [32] X. Zhao, L. You, R. Cao, Y. Shao, and L. Fu, "Broadband digital over-the-air computation for asynchronous federated edge learning," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 5359–5364.
- [33] C. Tarver, M. Tonnemacher, H. Chen, J. Zhang, and J. R. Cavallaro, "GPU-based, LDPC decoding for 5G and beyond," *IEEE Open Journal of Circuits and Systems*, vol. 2, pp. 278–290, 2021.
- [34] (2015) LDPC Decoder powered by CUDA. [Online]. Available: <https://github.com/glymehrvrd/ldpc-gpu>
- [35] L. You, S. C. Liew, and L. Lu, "Reliable physical-layer network coding supporting real applications," *IEEE Transactions on Mobile Computing*, vol. 16, no. 8, pp. 2334–2350, 2016.
- [36] M. Gastpar and M. Vetterli, "Source-channel communication in sensor networks," in *Information Processing in Sensor Networks*. Springer, 2003, pp. 162–177.
- [37] B. Nazer and M. Gastpar, "Computation over multiple-access channels," *IEEE Transactions on Information Theory*, vol. 53, no. 10, pp. 3498–3516, 2007.
- [38] O. Abari, H. Rahul, and D. Katabi, "Over-the-air function computation in sensor networks," *arXiv preprint arXiv:1612.02307*, 2016.
- [39] M. Goldenbaum and S. Stanczak, "Robust analog function computation via wireless multiple-access channels," *IEEE Transactions on Communications*, vol. 61, no. 9, pp. 3863–3877, 2013.
- [40] A. Kortke, M. Goldenbaum, and S. Stańczak, "Analog computation over the wireless channel: A proof of concept," in *SENSORS, 2014 IEEE*. IEEE, 2014, pp. 1224–1227.
- [41] Y. Shao, D. Gündüz, and S. C. Liew, "Federated edge learning with misaligned over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 21, no. 6, pp. 3951–3964, 2022.
- [42] —, "Bayesian over-the-air computation," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 3, pp. 589–606, 2023.