Digit-Serial DA-Based Fixed-Point RNNs: A Unified Approach for Enhancing Architectural Efficiency

Digit-Serial DA-Based Fixed-Point RNNs: A Unified Approach for Enhancing Architectural Efficiency

Mohd Tasleem Khan and Mohammed A. Alhartomi, *Member, IEEE*

***Abstract*— The next crucial step in artificial intelligence involves integrating neural network models into embedded and mobile systems. This requires designing compact and energy-efficient neural network models in silicon for optimized performance. This article introduces a unified approach for enhancing the architectural efficiency of long short-term memory (LSTM) recurrent neural networks (RNNs). Precisely, two new structures (I and II) based on the two's complement (TC) digit-serial distributed arithmetic (DSDA) technique are presented. The block-circulant matrix–vector multiplications (MVMs) and element-wise multiplications (EWMs) are formulated using TC DSDA. In addition, a fixed-point (FxP) training procedure for quantized LSTM RNNs is considered and validated for speech recognition tasks. Both structures leverage the circular rotation of weights and generate partial products with input digit slices. A new partial-product generator (PPG) and partial-product selector (PPS) designed to work with both unsigned and signed digits is introduced. In Structure I, a nonpipelined MVM is realized with a few PPGs and PPSs, followed by a shift-accumulate unit (SAU). Conversely, in Structure II, a suitably chosen depth-pipelined MVM is achieved with multiple PPGs and PPSs, followed by a shift-to-add tree (SAT). A critical path delay (CPD) analysis for both the proposed structures is also presented. Compared with previous works, post-synthesis results on 28-nm fully depleted silicon-on-insulator (FDSOI) technology reveal that for a model size of 128 × 128, Structures I and II provide 39.87%, 95.63%, and 30.95%, 91.18% more area and energy efficiencies, respectively.**

***Index Terms*— Digit-serial distributed arithmetic (DSDA), long short-term memory (LSTM), matrix–vector multiplication (MVM), recurrent neural network (RNN).**

I. Introduction

RECURRENT neural networks (RNNs) are extensively used for numerous sequence labeling tasks, such as speech recognition [1] and machine translation [2]. They struggle to propagate useful gradient information from output to input layers due to the vanishing gradient problem [3]. The long short-term memory (LSTM) RNN addresses this limitation by incorporating multiple gates to regulate information flow [4]. Notably, it has found applications in virtual assistant user interfaces, such as *Apple Siri*, *Amazon Alexa*, and *Google Assistant*. While these applications typically run on cloud servers due to their compute-intensive nature, the rise of Internet of Things with increasing prevalence of mobile devices makes this approach nonscalable. Consequently, there is a demand to offload some or all of the LSTM RNN computations to energy-constrained and performance-limited mobile devices. This shift poses intricate challenges in achieving both high energy efficiency and throughput.

The implementation of LSTM RNNs on field-programmable gate arrays (FPGAs) [5], [6] is unsuitable for embedded systems and mobile devices due to their relatively high power consumption [7]. Application-specific integrated circuit (ASIC) solutions are more focused on developing compact and energy-efficient RNNs [7], [8], [9], [10], [11], [12], [13], [14]. The technology in ASICs plays a pivotal role in determining design efficiency, e.g., fully depleted silicon-on-insulator (FDSOI) technology has demonstrated superior performance capabilities compared to complementary metal–oxide–semiconductor (CMOS) [15]. This is further driven by real-time feasibility of a task on mobile devices. For instance, the basic requirement of speech recognition task necessitates a clock frequency of at least 6.4 MHz [5], [13].

The gates in LSTM RNNs are considered as independent neural networks comprising several matrix–vector multiplications (MVMs) and element-wise multiplications (EWMs). In addition, they require large memory to store the network parameters [16], [17]. The efficiency of an LSTM RNN engine is limited by the number of MVMs/EWMs and the memory size. To improve the efficiency, effective quantization in weights/activation functions (AFs) could be utilized [18]. This would reduce the memory size, thereby enabling the accommodation of large network parameters. Among different methods [19], [20], [21], fixed-point (FxP) quantizations employ moderate bitwidths, ensuring the preservation of accuracy comparable to that of 32-bit floating point (FlP). Low-bit FxP weight quantization leads to the unavailability of loss function for backpropagation during training. Using the straight-through estimator (STE) [20] through suitable activation quantization can help avoid the issue of unavailable gradients. While alternating direction method of multipliers (ADMM) [22] can be employed to solve the weight quantization problem during training with a specified FxP scheme. Interestingly, the necessity to backpropagate through quantizer in ADMM is eliminated as it performs the iterative optimization using an augmented Lagrangian [23], [24].

While weight quantization plays a significant role in reducing computational costs, researchers are concurrently

Manuscript received 15 May 2023; revised 20 February 2024 and 23 May 2024; accepted 5 July 2024. This work was supported in part by the Deanship of Scientific Research, University of Tabuk under Grant S-1443-0195. *(Corresponding author: Mohd Tasleem Khan.)*

Mohd Tasleem Khan is with the Institute of Sensors, Signals and Systems, School of Engineering and Physical Sciences, Heriot-Watt University, EH14 4AS Edinburgh, U.K. (e-mail: M.T.Khan@hw.ac.uk).

Mohammed A. Alhartomi is with the Department of Electrical Engineering, University of Tabuk, Tabuk 71491, Saudi Arabia.

Digital Object Identifier 10.1109/TNNLS.2024.3425569

exploring diverse network compression schemes [25], [26], [27], [28]. For instance, in [26], a weight-sharing method was introduced to expedite sparse MVM. The effectiveness of FP quantization, as demonstrated in [25] and [27], showcased its tolerance to the accuracy of the compressed network. Utilizing low-displacement rank matrices, such as circulant matrices, emerges as a viable approach to pruning redundant parameters [29], [30]. This approach results in minimal loss of accuracy while reducing computational costs and memory. Further redundancy can be introduced using block-circulant variants, albeit with a slight degradation in accuracy [31]. In [32], circulant matrices in tandem with ADMM have been employed to accelerate MVMs.

Although circulant/block-circulant matrices are widely used to compress the LSTM RNN, efforts are being taken in tandem to reduce the multipliers [12], [13], [14], [28]. The sparsity in the weights was successfully exploited to reduce MVM computation [28]. In [12], a method based on circulant MVMs was presented to reduce memory access. Recently, circulant MVMs were implemented using the distributed arithmetic (DA) technique [13], [14]. Specifically, the partial products of one input vector are stored in a lookup table (LUT), with bit slices of the other vector used as addresses in inner product computation (IPC). This is followed by a shift-accumulate unit (SAU) to produce the output in specific clock cycles. Precisely, offset-binary coding (OBC) has been used to reduce the LUT size at the expense of an offset term in SAU [33].

In contrast to traditional OBC-DA, partial products in [13] and [14] were generated using adders, logic gates, and registers with digit slices (or bit slices) of weights as the selection lines instead of addresses. For example, in [13], partial products in radix-2 OBC form were realized in parallel, but its computational cost was found to be higher. This was partially mitigated by the serial generation of partial products in radix-2^{γ} OBC form with intermediate processing blocks in [14], where γ is the digit size. Due to digit-set symmetries in OBC-DA, the partial products for both signed and unsigned bits (or digits) are the same except for a different order. Thus, the same partial-product generator (PPG) was utilized, and however, a separate offset term generator was required. Moreover, fine-grained pipelining in [13] and [14] was employed to reduce the critical path delay (CPD). For the realization of EWMs, binary multipliers were used in [13], while a shift-and-add approach was employed in [14]. Recently, Alhartomi et al. [34] presented four different parallel LUTs based on radix-2 two's complement (TC) scheme to trade the computational cost and speed on FPGA. Notably, all the existing DA-based designs were not unified, as MVMs utilized PPGs/partial-product selectors (PPSs) and EWMs employed a different realization approach.

To the best of knowledge, efficient architectures for LSTM RNNs based on TC digit-serial DA (DSDA) have yet to be explored. Moreover, the development of FxP LSTM RNNs based on circulant matrices, alongside STE [19], [20], [21], [35] and ADMM [22], [23] quantization algorithms for speech recognition tasks, has not been studied. This motivates us to formulate a unified approach that systematically develops efficient accelerators for LSTM RNNs to cater to both training and inference tasks. Unlike OBC scheme, TC DSDA obviates the need for an offset term and eliminates its explicit need on hardware. However, it lacks symmetry in the digit set, preventing the use of the same PPG for both signed and unsigned digits. This challenge is addressed through the introduction of a new PPG/PPS unit. Considering the partial products of weights with digit slices of inputs as the selection lines eliminates the need for intermediate processing blocks and reduces the circular shifter complexity compared to [14]. In addition, the proposed methodology also includes a detailed CPD analysis and a consistent realization of MVMs and EWMs. Our contributions are summarized as follows.

1) The mathematical formulation of the compressed LSTM RNN using TC DSDA is presented, followed by its FxP training procedure.
2) A new PPG and PPG based on TC DSDA is proposed, which can efficiently handle both unsigned and signed digits.
3) Nonpipelined (Structure I) and pipelined (Structure II) LSTM RNNs are introduced, including a new circular shifter design for row-wise IPC and their CPD analysis. Both Structures I and II can be extended to their EWM counterparts. A comprehensive framework for developing LSTM RNN with various design parameters based on Structures I and II is discussed.
4) Structures I and II are evaluated on the ASIC platform, demonstrating their effectiveness with suitably chosen design parameters.

The rest of this article is organized as follows. Section II presents the unified mathematical formulation of LSTM RNN using TC DSDA followed by its training strategy. Section III discusses the details of the proposed structures. Section IV evaluates and compares the performance of the proposed and existing designs. Our conclusions are provided in Section V.

II. Formulation of Compressed LSTM RNN and Training Procedure

In Fig. 1, a typical LSTM RNN layer is depicted, distinct from conventional RNNs due to its use of memory cells to store activation values from preceding words in long sequences. The system controls information flowthrough specialized gates [3]. Specifically, the input gate, in tandem with the block gate, updates the information of the cell using current activation values. Concurrently, the forget gate determines how much information should be retained or removed from the previous cell state. A typical LSTM RNN layer is governed by the following equations:

$$\text{Input Gate: } \mathbf{i}^t = \sigma\left(\mathbf{W}_i \circledbullet \mathbf{x}^t + \mathbf{R}_i \circledbullet \mathbf{y}^{t-1} + \mathbf{b}_i\right) \quad (1)$$

$$\text{Forget Gate: } \mathbf{f}^t = \sigma\left(\mathbf{W}_f \circledbullet \mathbf{x}^t + \mathbf{R}_f \circledbullet \mathbf{y}^{t-1} + \mathbf{b}_f\right) \quad (2)$$

$$\text{Output Gate: } \mathbf{o}^t = \sigma\left(\mathbf{W}_o \circledbullet \mathbf{x}^t + \mathbf{R}_o \circledbullet \mathbf{y}^{t-1} + \mathbf{b}_o\right) \quad (3)$$

$$\text{Block Gate: } \mathbf{z}^t = \tanh\left(\mathbf{W}_z \circledbullet \mathbf{x}^t + \mathbf{R}_z \circledbullet \mathbf{y}^{t-1} + \mathbf{b}_z\right) \quad (4)$$

$$\text{Memory Cell: } \mathbf{c}^t = \mathbf{z}^t \odot \mathbf{i}^t + \mathbf{c}^{t-1} \odot \mathbf{f}^t \quad (5)$$

$$\text{Hidden State: } \mathbf{y}^t = \tanh\left(\mathbf{c}^t\right) \odot \mathbf{o}^t \quad (6)$$

where $t \in \{1, 2, \ldots, T\}$ indicates the time step, $\mathbf{W}_s$ $(s = i, f, o, z)$ represents parameter matrices of size $N_2 \times N_1$ for

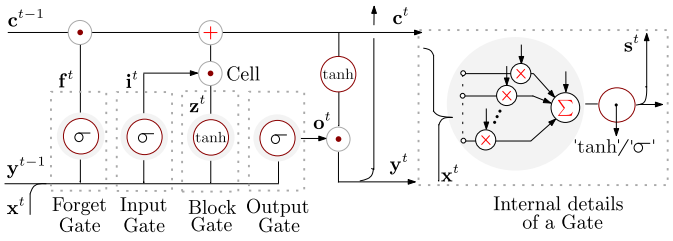


Fig. 1. Typical LSTM RNN layer. ◉: an MVM and ⊙: an EWM.

input weight connections, and $\mathbf{R}_s$ denotes the parameter matrices of size $N_2 \times N_2$ for recurrent connections. The dimensions of the input vector $\mathbf{x}^t$ and the recurrent vector $\mathbf{y}^{t-1}$ are $N_1 \times 1$ and $N_2 \times 1$, respectively. The vector $\mathbf{b}_s$ of size $N_2 \times 1$ corresponds to the bias terms. From these dimensions, the total number of network parameters to be learned during LSTM RNN training is $4(N_1N_2 + N_2^2 + N_2)$. Assuming equal dimensions for input and output nodes (i.e., $N_1 = N_2 = N$), the simplified total number of network parameters to be learned becomes $8N^2 + 4N$. The operators "◉" and "⊙," as specified in (1)–(6), denote an MVM and an EWM, respectively. Although "⊙" is commonly used for an EWM, "◉" is intentionally used for an MVM. By doing so, equivalence between operators "⊙" and "◉" can be established when the size of matrices and vectors is reduced to unity as $\circledcirc\,|_{N_1,N_2=1} \iff \odot\,|_{N_2=1}$. This equivalence would allow the designing of an EWM from an MVM, as discussed in Section III. The nonlinear AFs given in (1)–(6) i.e., sigmoid, $\sigma(\cdot)$, and hyperbolic tangent, $\tanh(\cdot)$, for any input x are, respectively, defined as

$$\sigma(x) = \frac{1}{1+e^{-x}} \in \{0, 1\}, \quad \tanh(x) = \frac{1-e^{-x}}{1+e^{-x}} \in \{-1, 1\}. \tag{7}$$

In the subsequent discussion, the superscript "t" for time step in different variables is omitted without loss of generality.

A. LSTM RNN Compression With Circulant Matrices

In this section, parameter matrices are replaced with circulant matrices [36] specified in (1)–(4). The total number of network parameters in the LSTM RNN, utilizing circulant matrices [12], is $2N^2 + 10N$. However, this substitution requires an additional step since circular matrices are typically square, i.e., $N_1 = N_2$, while the parameter matrices may not be equal $N_1 \neq N_2$. For example, in a fully connected layer, the number of inputs and outputs is the same, whereas in a softmax layer, they are not equal [12]. Consequently, when $N_2 > N_1$, it is necessary to append zeros to the input $\mathbf{x}^t$, and when $N_2 < N_1$, the multiplication result must be truncated to retain only the first N_2 elements. It is interesting to note that circulant matrices achieve a compression factor of nearly $4\times$ (for large N). Any circulant matrix, $\mathbf{W}$, is characterized by a primitive vector, $\mathbf{w}_1 = \{w_m\}_{m=1}^N$, corresponding to its first column. The remaining columns of $\mathbf{W}$ are obtained by leftward rotations of the elements of $\mathbf{w}_1$, as illustrated in Fig. 2(a). Observably, the consecutive rows of $\mathbf{W}$ are circular versions of each other. For any column vector $\mathbf{w}_n$ of $\mathbf{W}$, it can be generalized as

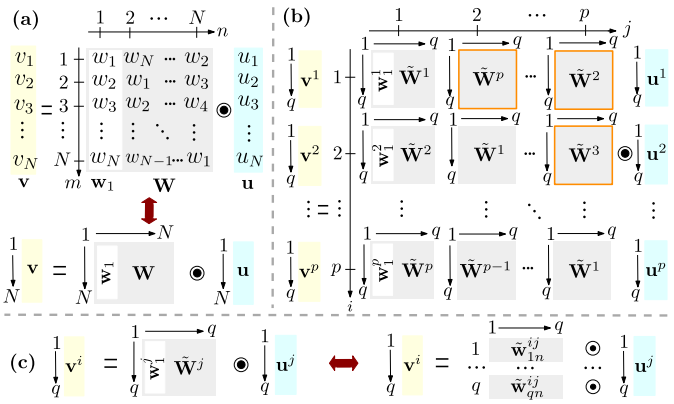


Fig. 2. Illustration of (a) $N \times N$ circulant MVM with its pictograph. (b) Block-circulant MVM with $N = q \times p$. (c) Equivalence between a subcirculant MVM and q IPCs.

$\mathbf{w}_n = [w_{1n}, w_{2n}, \ldots, w_{Nn}]^T$, where $mn = (m-n)\text{mod}(N)+1$, and $1 \leq m, n \leq N$. Therefore, (1)–(4) can be written as

$$\mathbf{v} = \mathbf{W} \circledcirc \mathbf{u} = \{v_m\}_{m=1}^N \quad \text{and} \quad v_m = \sum_{n=1}^{N} w_{mn}u_n \tag{8}$$

where $\mathbf{W}$ is an $N \times N$ circulant matrix and $\mathbf{u} = \{u_n\}_{n=1}^N$ is an input vector.

For large-sized LSTM RNN models, circulant matrices need to be split into p^2 smaller submatrices of size $q \times q$ with $q = N/p$ using block-circulant formulation [31]. They are distributed uniformly across all rows and columns, as illustrated in Fig. 2(b). Corresponding to submatrices, p primitive subvectors $\mathbf{w}_1^1, \mathbf{w}_1^2, \ldots, \mathbf{w}_1^p$ are obtained from the primitive vector $\mathbf{w}_1$, where $\mathbf{w}_1^p = \{w_{p+(m-1)p}\}_{m=1}^q$. Each submatrix $\mathbf{W}^p$ with a primitive vector $\mathbf{w}_1^p$ can be expressed as $\mathbf{W}^p = \{\mathbf{w}_{mn}^p\}_{m,n=1}^q$, where $mn = (m-n)\text{mod}(q)+1$ and (m, n) is (row index, column index) of a sub-MVM, with $1 \leq m, n \leq q$. Similarly, the input vector $\mathbf{u}$ is split into p input subvectors $\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^p$, where $\mathbf{u}^p = \{u_{p+(n-1)p}\}_{n=1}^q$, and the output vector $\mathbf{v}$ is split into p output subvectors $\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^p$, such that $\mathbf{v}^p = \{v_{p+(m-1)p}\}_{m=1}^q$. Thus, the sub-MVM with output $\mathbf{v}^i$ $(1 \leq i \leq p)$ in IPC form can be written as

$$\mathbf{v}^i = \sum_{j=1}^{p} \tilde{\mathbf{W}}^{ij} \circledcirc \mathbf{u}^j \tag{9}$$

where $ij = (i-j)\text{mod}(p)+1$ and $\tilde{\mathbf{W}}^{ij}$ is defined as

$$\tilde{\mathbf{W}}^{ij} = \begin{cases} \mathbf{W}^{ij}, & i > j \\ \mathbf{W}^1, & i = j \\ \mathbf{\Phi}\mathbf{W}^{ij}, & i < j \end{cases} \tag{10}$$

where $\mathbf{\Phi}$ is the circular-shift matrix used to obtain $\tilde{\mathbf{W}}^{ij}$ from $\mathbf{W}^{ij}$. Any element ϕ_{mn} of $\mathbf{\Phi}$ can be expressed as

$$\phi_{mn} = \begin{cases} 1, & (m-n)\text{mod}(q)+1 == 2 \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

By utilizing (11) and (10), one can compute (9). It is clear from (9) that p input vectors $\mathbf{u}^j$ and p submatrices $\tilde{\mathbf{W}}^{ij}$ $(1 \leq j \leq p)$ are involved in computing the output vector $\mathbf{v}^i$,

as shown in Fig. 2(b). Therefore, any $\mathbf{v}^i$ can be computed using $\tilde{\mathbf{W}}^{ij}\mathbf{u}^j$ successively in p clock cycles, for instance, $\mathbf{v^1}$ is given by

$$\mathbf{v}^1 = \tilde{\mathbf{W}}^1 \circledcirc \mathbf{u}^1 + \tilde{\mathbf{W}}^p \circledcirc \mathbf{u}^2 + \cdots + \tilde{\mathbf{W}}^2 \circledcirc \mathbf{u}^p. \tag{12}$$

From (12), it is implied that the sub-MVM, i.e., $\tilde{\mathbf{W}}^{ij}\mathbf{u}^j$ for any i, can be computed in parallel. Hence, the entire MVM given in (9) can be completed in p clock cycles. It can be observed that the block-circulant MVM for $p = 1$ and $q = N$ reduces to the circulant MVM, leading the primitive vector $\mathbf{w}_1^p = \{w_m\}_{m=1}^q$ to $\mathbf{w}_1$, submatrices $\mathbf{W}^p = \{\mathbf{w}_{mn}^p\}_{m,n=1}^q$ to $\mathbf{W}$, input vector $\mathbf{u}^p = \{u_{p+(n-1)p}\}_{n=1}^q = \{u_n^p\}_{n=1}^q$ to $\mathbf{u}$, and output vector $\mathbf{v}^p = \{v_{p+(m-1)p}\}_{m=1}^q = \{v_m^p\}_{m=1}^q$ to $\mathbf{v}$. In a scalar form, (12) can be reexpressed as

$$v_m^i = \sum_{j=1}^{p}\left(\sum_{n=1}^{q} \tilde{w}_{mn}^{ij} u_n^j\right) = \sum_{j=1}^{p} \tilde{v}_{\tilde{m}}^{ij} \tag{13}$$

where $\tilde{m} \in \{(m-1)\text{mod}(q)+1, (m-2)\text{mod}(q)+1, \ldots, (\text{m-q})\text{mod}(q)+1\}$ and $\tilde{v}_{\tilde{m}}^{ij}$ is defined as

$$\tilde{v}_{\tilde{m}}^{ij} = \sum_{n=1}^{q} \tilde{w}_{mn}^{ij} u_n^j = \tilde{\mathbf{w}}_m^{ij} \circledcirc \mathbf{u}^j \tag{14}$$

where $\tilde{\mathbf{w}}_m^{ij} = \{\tilde{w}_{mn}^{ij}\}_{n=1}^q$, $\mathbf{u}^j = \{u_n^j\}_{n=1}^q$, and "$\circledcirc$" denotes the operator for an IPC. Likewise, one can establish the relationship between IPC and MVM operators as $\circledcirc\,|_{N_1=1} \Longleftrightarrow \odot$. Clearly, an equivalence for deriving an q IPC from a block-circulant MVM can be established, as shown in Fig. 2(c).

B. Formulation of IPCs in Circulant MVMs Using TC DSDA

There are two potential methods to formulate the IPCs as given in (14) using TC DSDA as follows.

P1): Weights are expressed in TC form with inputs as such.
P2): Inputs are represented in TC form with weights as such.

The existing approaches [13], [14], [34] relied on the P1 method followed by the OBC scheme. In this work, the P2 method is considered to enhance the LSTM RNN efficiency. The justification for choosing P2 method from an architectural perspective is discussed in the next section. For clarity, they are illustrated in Fig. 3(a) and (b). By representing the B-bit inputs u_n^j in radix-2 TC form as

$$u_n^j = -u_{n,B-1}^j + \sum_{k=0}^{B-2} u_{n,k}^j 2^k \tag{15}$$

where $u_{n,k}^j \in [0, 1]$. In radix-2^γ TC form, γ-adjacent bits $\{u_{n,\gamma(k+1)-l}^j\}_{l=1}^{\gamma}$ of every digit in $u_{n,k}^j$ need to be considered. This would allow to rewrite (15) in radix-2^γ TC form as

$$u_n^j = -u_{n,\gamma B_\gamma-1}^j + \sum_{k=0}^{B_\gamma-2} u_{n,\gamma k}^j 2^{\gamma k} \tag{16}$$

where $u_{n,\gamma k}^j \in [0, 1, \ldots, 2^\gamma - 1]$ indicates the kth digit of u_n^j with $0 \le k \le B_\gamma - 1$ and $B_\gamma = \lceil B/\gamma \rceil$ represents the number of digits. It can be noted from (16) that $u_{n,\gamma B_\gamma-1}^j$ refers to the most significant digit. Each $u_{n,\gamma k}^j$ in terms of γ-adjacent bits can be expressed as

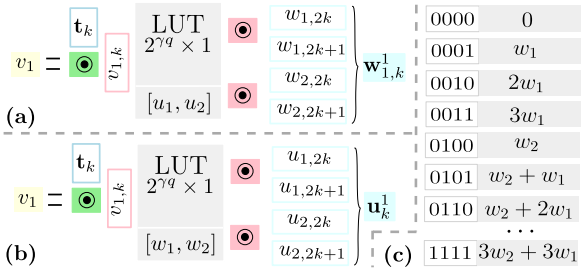


Fig. 3. (a) P1 method, (b) P2 method, and (c) LUT data with P2 method employing TC DSDA for $q = N = 2$, $\gamma = 2$, $B = 8$, and $0 \le k \le 3$.

$$u_{n,\gamma k}^j = \sum_{l=1}^{\gamma} u_{n,\gamma(k+1)-l}^j 2^{\gamma-l}. \tag{17}$$

For clarity, let us consider an example of block-circulant MVM for $q = N = 2$, $\gamma = 2$, and $B = 8$. It has two 8-bit input samples $\{u_1, u_2\}$ for $1 \le n \le 2$. These are processed with their two adjacent bits $\{u_{1,2k}, u_{1,2k+1}\}$ and $\{u_{2,2k}, u_{2,2k+1}\}$ at a time, where $0 \le k \le 3$.

Substituting (16) into (14) and interchanging the order of summation results in

$$\tilde{v}_{\tilde{m}}^{ij} = \sum_{k=0}^{B_\gamma-1} (-1)^{\lfloor k/(B_\gamma-1)\rfloor} \tilde{v}_{\tilde{m},\gamma k}^{ij} 2^{\gamma k} = \tilde{\mathbf{v}}_{\tilde{m},k}^{ij} \odot \mathbf{t}_k \tag{18}$$

with $\tilde{\mathbf{v}}_{\tilde{m},k}^{ij} = \{\tilde{v}_{\tilde{m},\gamma k}^{ij}\}_{k=0}^{B_\gamma-1}$ and $\mathbf{t}_k = \{(-1)^{\lfloor k/(B_\gamma-1)\rfloor} 2^{\gamma k}\}_{k=0}^{B_\gamma-1}$. Let us define each $\tilde{v}_{\tilde{m},\gamma k}^{ij}$ as

$$\tilde{v}_{\tilde{m},\gamma k}^{ij} = \sum_{n=1}^{q} \tilde{w}_{mn}^{ij} u_{n,\gamma k}^j = \tilde{\mathbf{w}}_m^{ij} \circledcirc \mathbf{u}_k^j \tag{19}$$

where $\mathbf{u}_k^j = \{u_{n,\gamma k}^j\}_{k=0}^{B_\gamma-1}$ with $u_{n,\gamma k}^j = \{l\}_{l=0}^{\gamma-1}$. It can be observed that the IPC in (14) is transformed into two different IPCs for TC DSDA computations, as shown in Fig. 3(b). The one shown in green as per (18) corresponds to an SAU with its input are power-of-two shifts, $\mathbf{t}_k$, and LUT output, $\tilde{\mathbf{v}}_{\tilde{m},k}^{ij}$. The other shown in pink as per (19) corresponds to an LUT with binary combinations of weights, $\tilde{\mathbf{w}}_m^{ij}$, and addresses as input digit slices, $\mathbf{u}_k^j$. For clarity, LUT data for an IPC based on TC DSDA with weight vector $\mathbf{w}_1^1 = \{w_1, w_2\}$, $\mathbf{u}_k^1 = \{u_{n,2k}, u_{n,2k+1}\}_{n=1}^2$, and $0 \le k \le 3$ are illustrated in Fig. 3(c). This analysis of IPC of an MVM can be extended for an EWM by setting $q = N = 1$.

C. Training Procedure for FxP Compressed LSTM RNN

The compressed LSTM RNN discussed in the previous subsections is quantized in terms of parameter matrices, $\mathbf{W}_s/\mathbf{R}_s$, and nonlinear AFs, $\sigma(\cdot)/\tanh(\cdot)$, to a target B-bits as

$$\mathcal{Q}_{B,\alpha} = \pm\alpha \times \left\{0, \frac{1}{2^{B-1}-1}, \frac{2}{2^{B-1}-1}, \ldots, 1\right\} \tag{20}$$

where α is the scaling factor, and terms inside the braces indicate the quantization levels. The weights $\hat{w}_{mn}$ in parameter

Algorithm 1 Training Procedure for FxP LSTM RNN

Input : 32-bit floating-point LSTM RNN model ($\mathcal{M}$) given in (1)–(6); FxP quant. scheme (20).
Output: Quantized LSTM RNN model ($\hat{\mathcal{M}}$)
Initials: $\mathbf{Z}_s^0 = \mathbf{W}_s$, $\mathbf{R}_s$; $\mathbf{U}_s^0 = \mathbf{0}$, $loss = 0$.
for *each Epoch* **do**
  //Update of $\mathbf{Z}_s^t$ and $\mathbf{U}_s^t$ (ADMM)
  $\mathbf{Z}_s^t \leftarrow \{proj_s(\mathbf{W}_s + \mathbf{U}_s^{t-1})\}$
  $\mathbf{U}_s^t \leftarrow \mathbf{W}_s - \mathbf{Z}_s^t + \mathbf{U}_s^{t-1}$
  for *each Batch* **do**
    //Activation quantization (STE)
    $input \leftarrow proj_s(input)$;
    $loss \leftarrow \tilde{M}(input)$;
    $loss \leftarrow loss + \frac{1}{2}\sum ||\mathbf{W}_s - \mathbf{Z}_s^t + \mathbf{U}_s^{t-1}||^2$;
    Backpropagate *loss* and update $\mathbf{W}_s$;
  end
end
return $\hat{\mathcal{M}} \leftarrow \mathcal{M}\{proj_s(\mathbf{W}_s)\}$

matrices are quantized and projected from its 32-bit FlP counterpart as

$$\text{proj}_s(w_{mn}) \overset{\circ}{=}_{B,\alpha} \alpha \cdot f^{-1}\left(\frac{rd\big((2^B-1)\cdot f(\lceil w_{mn}, \alpha \rfloor)\big)}{2^B-1}\right) \tag{21}$$

where $\text{proj}_s(\cdot)$ denotes the mapping and quantizing onto an FxP quantization; $\text{rd}(\cdot)$ indicates the rounding operation; $f(\cdot)$ maps any value from $[-1, 1]$ to $[0, 1]$, i.e., $f(\cdot) = \tanh(\cdot)/2 + 0.5$; and $\lceil\cdot\rfloor$ indicates the clipping of w_{mn}, according to

$$\lceil w_{mn}, \alpha \rfloor = \begin{cases} -1, & w_{mn} < -\alpha \\ w_{mn}, & -\alpha \le w_{mn} \le \alpha \\ 1, & w_{mn} > \alpha. \end{cases} \tag{22}$$

Usually, LSTM RNNs have a certain limit to tolerate numerical errors, as their nonlinear AFs are approximated with piecewise linear functions [37]. The training procedure under consideration involves weight quantization based on (20). The network parameters are randomly initialized for training from scratch and incorporated circulant matrices and forward approximation. The gradients of the loss function become unavailable with AFs' quantization during backpropagation. This can be solved by employing the STE algorithm [19], [20], [21], [35]. It enables batch quantization training by setting the gradient to the constant value of 1

$$\textbf{Forward}: y = rd(x) \quad \text{and} \quad \textbf{Backward}: \frac{\partial y}{\partial x} = \mathbf{1}_{x \in R}. \tag{23}$$

While ADMM [22], [23] is incorporated to solve the weights of parameter matrices using the FxP quantization scheme given in (20)–(22) as the optimization constraint. This work jointly employs STE and ADMM approaches for AFs' and weights' quantization respectively, as described in Algorithm 1.

To demonstrate the effectiveness of training procedure, a speech recognition task with TIMIT dataset is examined [38]. The dataset features 630 speakers from eight distinct American English dialects, each reading ten sentences rich in phonetic variety. In addition, we utilize an extensive speech recognition dataset with about 500 h dedicated to training, 50 h for validation, and 5 h for testing. The number of features of the input speech is the same as in [28]. The raw audio recordings are initially converted into spectrograms using a Hamming window with 50% overlap before being input into the network model. In speech recognition tasks [30], the performance is evaluated based on phoneme error rate (PER); a smaller PER implies better prediction accuracy. Depending on the PER requirement, an appropriately sized network model can be used [9]. Similar to [12] and [39], the complete system comprises convolutional neural network (CNN) layers, bidirectional LSTM layers, and a fully connected layer. Note that the birectional LSTM layers can be considered as a combination of two standard LSTMs. The entire network is implemented in Pytorch and trained using the connectionist temporal classification loss [39]. For implementation and training purposes, a random pair of data is considered, with a batch size equal to kernel size. One may also choose a different batch or kernel size by trading off clock cycles and PER requirements [34]. During the testing phase of the network model, it is ensured that its implementation in Verilog hardware description language (HDL) exactly corresponds with its PyTorch model. Various measures have been taken into consideration to avoid overflow [14].

To assess the performance of FxP compressed LSTM RNN, two separate Shaprio–Wilk tests [18] with random initialization of weights for 128 × 128 layer and 4 × 4 kernel are conducted. The random initialization of weights is characterized by the Shaprio–Wilk test statistic, denoted by W. In almost all the simulations, W is in the range of 0.96–0.99, which indicates that the distribution of weights is normal. Corresponding to different initialization of weights, two W's are selected with different quantization cases in terms of mean (μ) and standard deviation (σ), as illustrated in Fig. 4. These cases are discussed as follows:

1) 32-bit FlP weights with AFs given in (7);
2) 32-bit FlP weights with 8-bit quantization of AFs [37];
3) 8-bit FxP weights with 8-bit quantization of AFs [37];
4) 4-bit FxP weights with 4-bit quantization of AFs [37].

Note that all the cases have a single training phase of 70 epochs. It can be observed from Fig. 4(a) and (b) that the distribution of weights for a given W is unique, and it is further distinguishable in terms of the quantization scheme for the weights and AFs. For both W's, it can be seen that the weights' distribution in cases 1) and 2) almost follow each other. This is because case 2) utilizes 8-bit FxP quantization of AFs, ensuring an adequate number of bits during the training process. For cases 3) and 4), 8 and 4 bit are considered for the FxP quantization of both weights and AFs. It is found that the weights' distribution is affected by the choice of bits and a value of W. For instance, the distribution with 8-bit FxP quantization is slightly affected with $W = 0.97$ and almost identical with $W = 0.99$ compared to case 1). Unlike case 3), case 4) has a different distribution compared to case 1), irrespective of W. This is because the number of bits is not sufficient to represent all the weights and AFs, thereby impacting the weights' distribution in terms of μ and σ. This is

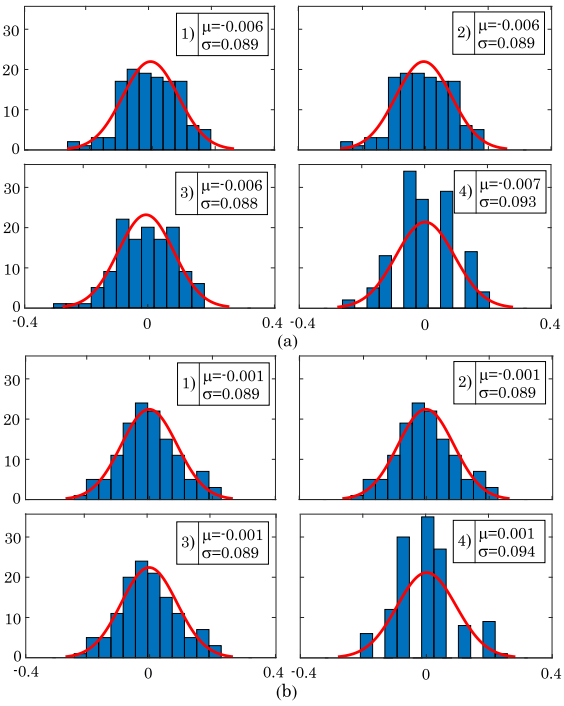


Fig. 4. Distribution of weights for 128 × 128 layer under different random initialization with (a) $W = 0.97$ and (b) $W = 0.99$.

TABLE I

ACCURACY RESULTS OF FXP COMPRESSED LSTM RNN FOR 128 × 128 LAYER AND 4 × 4 KERNEL AT SHAPIRO–WILK TEST STATISTIC, $W = 0.97$

Case	$\sigma(\cdot)$, $\tanh(\cdot)$	$\mathbf{W}_s$, $\mathbf{R}_s$	$\mathbf{c}^t$, $\mathbf{y}^t$	$\mathbf{i}^t$, $\mathbf{f}^t$ $\mathbf{o}^t$, $\mathbf{z}^t$	$\mathbf{b}_i$, $\mathbf{b}_f$ $\mathbf{b}_o$, $\mathbf{b}_z$	PER	CR
1)†	(8)	FlP32	FlP32	FlP32	FlP32	24.65%	–
2)†	[37]	FlP32	FlP32	FlP32	FlP32	24.81%	–
3)†	[37]	FxP8	FxP8	FxP8	FxP8	27.35%	93.53%
4)†	[37]	FxP4	FxP4	FxP4	FxP4	30.43%	96.76%
5)‡	[37]	FxP8	FxP8	FxP8	FxP8	21.69%	93.53%
6)‡	[37]	FxP4	FxP4	FxP4	FxP4	23.95%	96.76%

†: Random initialization, ‡: Re-trained from case 2), PER: Phone-error rate, CR: Compression ratio, FlPB/FxPB: B-bit floating/fixed-point quantization.

indicated by histogram bins, as some of them become emptied or occupied by their nearest neighbors.

The accuracy results of the trained FxP compressed LSTM RNN for $W = 0.97$ are listed in Table I. It can be seen that the prediction accuracy in terms of PER of case 2) is slightly decreased due to the aforementioned reason, while 8-bit FxP quantization of the weights in circulant parameter matrices can degrade the PER by about 2.54%. Notably, reducing the bits for both weights and AFs can further impact the prediction accuracy. Compared with case 1), cases 3) and 4) result in a 2.69% and 5.68% decrease in prediction accuracy, and 93.53% and 96.76% increase in compression, respectively. Two additional cases, namely, case 5) and case 6), are also considered to demonstrate the effectiveness of the training procedure when retrained from case 2), similar to [12]. It is found that the accuracy is improved at the expense of more epochs. Precisely, the PER improvement in cases 5) and 6) compared to cases 3) and 4) is 5.66% and 6.48%, respectively.

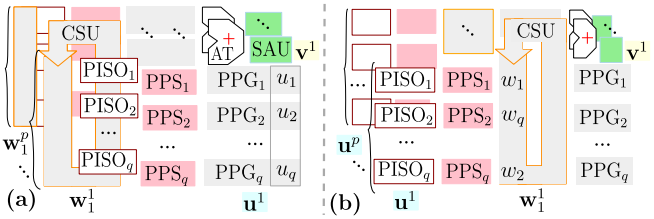


Fig. 5. Illustration of LUT-less topologies with (a) P1 and (b) P2 methods.

III. PROPOSED ARCHITECTURES

As discussed, the traditional TC DSDA for each IPC in block-circulant MVM requires an LUT of size $2^{\gamma q}$. For a given N, a high q (or low p) and/or a high γ results in large-sized LUT, which is not desirable from an implementation perspective. An alternative way to implement the IPCs based on P1 and P2 methods is to opt for an LUT-less approach, involving several PPGs and PPSs [13], [14], [34]. Corresponding to P1 and P2 methods shown in Fig. 3(a) and (b), there exists two LUT-less topologies shown in Fig. 5(a) and (b), respectively. Note that the select lines for PPSs are obtained by slicing inputs/weights into digits using a parallel-in serial-out (PISO) converter. Circular-shift unit (CSU) is employed to perform the rotations on inputs/weights. An adder tree (AT) is needed to accommodate the partial products.

A. Rationale of Structure Selection for MVM/EWM

For given LUT-less topologies, it is important to analyze them from an energy consumption perspective. For any digital circuit, the energy needed to perform a task is given by

$$\text{Energy} = \text{Power}/\text{Sample rate}. \tag{24}$$

As observed from the topologies, their energy consumption is primarily determined by CSU, PISO, and SAU since they are driven by the clock. Notably, the location of SAU in both topologies is the same, while the positions of CSU and PISO differ. Consequently, they play a crucial role in determining the energy consumption. Note that CSU holds a block of weights for every input sample to be processed in multiple digits since topology based on the P1 method rotates the weights and slices them into digits sequentially. Thus, its energy consumption, E_{P1}, is given by

$$E_{\text{P1}} = \left(P_{\text{CSU}} + B_\gamma \cdot P_{\text{PISO}}\right)/f_{\text{samp,clk}} \tag{25}$$

where P_{CSU} and P_{PISO} are the power consumption of CSU and PISO, respectively, and $f_{\text{samp,clk}}$ is the sample rate. The PISO unit has to finish its operation by running $B_\gamma\times$ faster in each sample iteration. While the topology based on the P2 method rotates the weights and slices the inputs into digits concurrently. Thus, its energy consumption, E_{P2}, is given by

$$E_{\text{P2}} = P_{\text{CSU}}/f_{\text{samp,clk}} + P_{\text{PISO}}/f_{\text{samp,clk}}. \tag{26}$$

In DSDA, $f_{\text{samp,clk}}$ is $1/B_\gamma$ × clock frequency to complete digit-serial computations. It is clear from (26) that PISO inherently works B_γ times faster to finish its operation. This implies that $E_{\text{P2}} < E_{\text{P1}}$, indicating that a more energy-efficient engine for LSTM RNNs can be derived using the topology based on the P2 method, the details of which are discussed next.

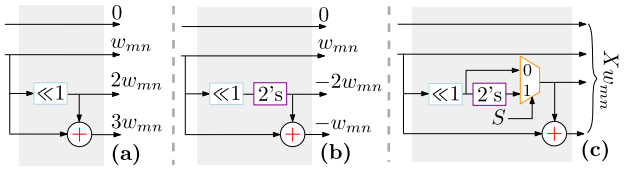


Fig. 6. Schematic of PPG for (a) unsigned digit, (b) signed digit, and (c) unsigned/signed digit.

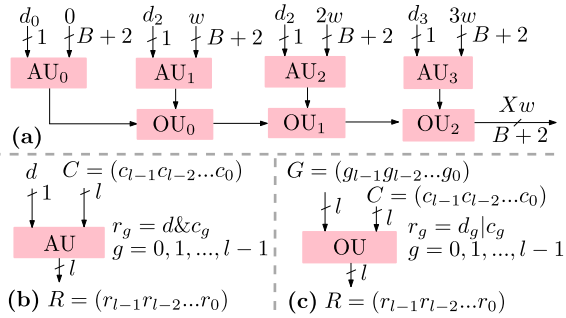


Fig. 7. Schematic of (a) PPS with (b) AU and (c) OU details. The operators "&" and "|" indicate the bitwise AND and OR, respectively.

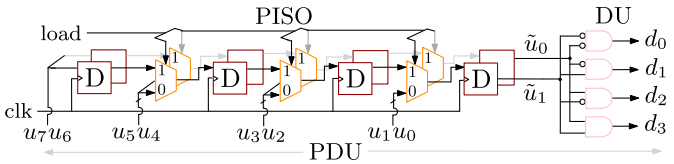


Fig. 8. Schematic of PDU with PISO and DU details.

B. Details of the Proposed Structures

1) PPG Unit: The schematics of the proposed PPG unit with input unsigned and/or signed digits are shown in Fig. 6(a)–(c). Each input digit has 2 bits, $u_{i,2k+1}u_{i,2k}$ ($0 \leq k \leq B_\gamma - 1$), which assume four possible combinations. For instance, $u_{i,2k+1}u_{i,2k}$ takes either 0, 1, 2, and 3 for unsigned digit ($k \neq B_\gamma - 1$) or 0, 1, -2, and -1 for signed digit ($k = B_\gamma - 1$), as per (16). These combinations, along with weights w_{mn}, result in partial products of the form Xw_{mn}. Due to the different digit-set of unsigned digit ($X \in \{0, 1, 2, 3\}$) and signed digit ($X \in \{0, 1, -2, -1\}$), they require separate PPGs. However, it is possible to use the same PPG for both unsigned and signed digits. The partial products with unsigned input digit are $0, w_{mn}, 2w_{mn}$, and $3w_{mn}$, where 0 can be obtained through an internally generated reset signal, $2w_{mn}$ is generated with a left shift on the weight w_{mn}, and $3w_{mn}$ needs to be generated with an adder, i.e., $3w_{mn}$ can be obtained by w_{mn} and $2w_{mn}$, as shown in Fig. 6(a). For any γ, there would be $(2^{\gamma-1} - 1)$ adders to generate the odd multiples.

The partial products with the input signed digit are $0, w_{mn}, -2w_{mn}$, and $-w_{mn}$, where the first half is the same as partial products with an unsigned digit. The partial products $-2w_{mn}$ and $-w_{mn}$ can be obtained through two separate 2's complementers with inputs as $2w_{mn}$ and w_{mn}, respectively. However, this would require a separate PPS for the input signed digit. However, the 2's complementer to obtain $-w_{mn}$ from w_{mn}, and extra PPS can be avoided. This is achieved by using the same adder to add $-2w_{mn}$ and w_{mn}, as shown in Fig. 6(b). By doing so, the same PPG can be utilized for both unsigned and signed digits. However, a 2-to-1 multiplexer is needed for bypassing the partial products corresponding to the unsigned digits, as shown in Fig. 6(c). The select signal "S" of a 2-to-1 multiplexer can be generated through a counter, and when the counter reaches a count value of B_γ, it sets $S = 1$, and otherwise, $S = 0$.

2) PPS Unit: The schematic of the proposed PPS for $\gamma = 2$ is shown in Fig. 7. It comprises four AND units (AUs) and three OR units (OUs). The details of AU and OU are also explained in Fig. 7. In general, it comprises 2^γ AUs and $(2^\gamma - 1)$ OUs. Each AU takes an l-bits input $C = c_{l-1}, c_{l-2}, \ldots, c_0$ with $0 \leq l \leq B + \gamma - 1$ and 1-bit inputs $d_0, d_1, \ldots, d_{2^\gamma-1}$. Thus, it requires "$l$" two-input AND gates, which distribute all the l-bits of input C to the AND gates, while the other input of each AND gate is fed with 1-bit inputs $d_0, d_1, \ldots, d_{2^\gamma-1}$. Similarly, the details of each OU can be explained, where "l" denotes the number of two-input OR gates in OU. The 1-bit inputs d_0, d_1, d_2, and d_3 can be obtained through a 2-to-4 decoder unit (DU), as shown in Fig. 8. It takes two adjacent bits $(\tilde{u}_1\tilde{u}_0)$ of the input digit from a PISO converter at a time. These are utilized to produce all four outputs of DU, i.e., $d_0 = \bar{\tilde{u}}_0$ & $\bar{\tilde{u}}_1$, $d_1 = \tilde{u}_0$ & $\bar{\tilde{u}}_1$, $d_2 = \bar{\tilde{u}}_0$ & $\tilde{u}_1$, and $d_3 = \tilde{u}_0$ & $\tilde{u}_1$, where "&" indicates the bitwise AND operator.

It can be noted that $0, w_{mn}, 2w_{mn}$, and $3w_{mn}$ are in the TC form and sign-extended to $(B+2)$-bits. Any digit $u_{n,\gamma k}$ of input u_n is obtained through a PISO converter. For instance, each of the two-adjacent bits $(\tilde{u}_1\tilde{u}_0)$ from a PISO is obtained after every clock cycle. The outputs $(\tilde{u}_1\tilde{u}_0)$ indicate the adjacent bits $\{u_1u_0; u_3u_2; u_5u_4; u_7u_6\}$. The PISO and DU together are referred to as PDU. Note that PDU, in general, consists of B_γ number of γ-bit registers in PISO, and 2^γ AND gates involved in DU with some inputs are inverted. In the following, the proposed Structures I and II are discussed.

3) Architectural Details of Structure I: The MVM based on Structure I for $N = 4$ is shown in Fig. 9. It computes the MVM output row-wise one at a time. The weight w_{mn} of any mth row is a right-circular shifted version of its $(m-1)$th row. It comprises a CSU, N PPGs $\{\text{PPG}_n\}_{n=1}^N$, N PPSs $\{\text{PPS}_n\}_{n=1}^N$, an AT, an SAU, a tapped-delay line unit (TDLU), and a bias-term load unit (BTLU). The CSU rotates the first-row weight vector $\mathbf{w} = \{w_{1n}\}_{n=1}^N$ with $1n = (1-n)\text{mod}(N)+1$ to make the next weight vector available for the successive row. This is achieved with a set of N 2-to-1 multiplexers and N registers. The multiplexers in CSU either load the first-row weight vector or rotate it to perform the row-wise IPC depending on the select enable (SE) signal. For instance, SE $= 0$ corresponds to the loading of the first-row weight vector and SE $= 1$ enables its rotation using the registers in successive clock cycles.

The IPC is then performed with the corresponding weight vector from CSU. The partial products Xw_{mn} are first computed using PPGs, and then, one of them is selected using PPSs. Subsequently, the outputs of PPSs are added together through an AT, as per (19). Finally, an SAU is used to perform its operation on the AT output for B_γ clock cycles to produce the output v_m, as per (18). Note that the operations of SAU and PDU are performed concurrently. After every $(B_\gamma - 1)$th

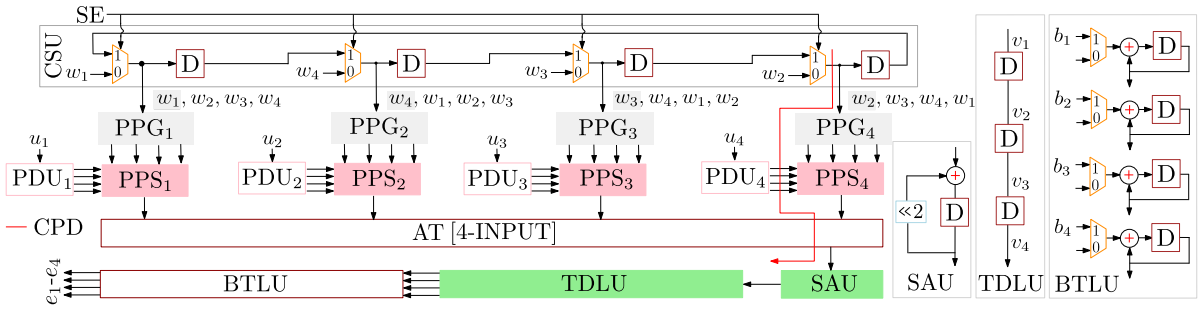


Fig. 9. Schematic of Structure I for $N = 4$ and $\gamma = 2$, where SE $= 1$ during CSU operation, and otherwise, SE $= 0$.

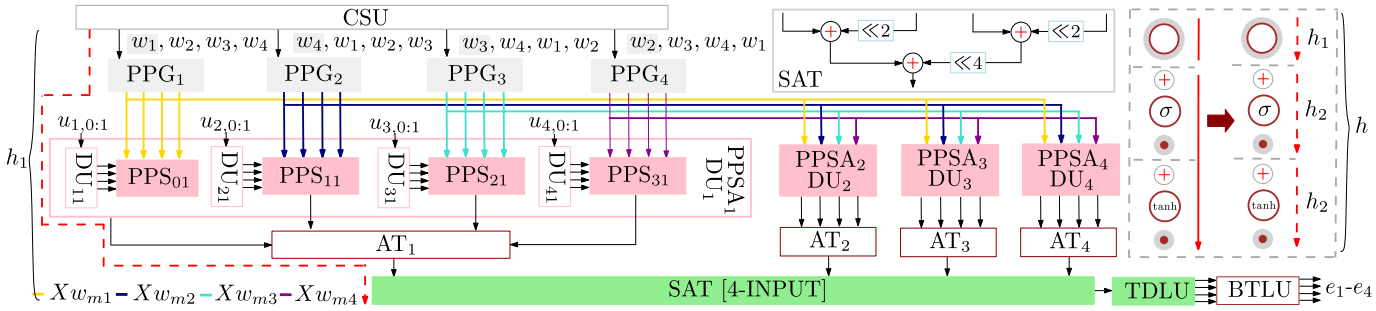


Fig. 10. Schematic of Structure II for $N = 4$, $B = 8$, and $\gamma = 2$, where PPSA$_n$ is the array of PPS$_{B_\gamma n}$ and $1 \le n \le 4$ and $0 \le B_\gamma \le 3$.

clock cycle, the output v_m is stored in TDLU one at a time and subsequently tapped out to produce all the MVM outputs. It is clear, from (1) to (4), that the bias terms $\mathbf{b}_s(s = z, i, f, o) = \{b_m\}_{m=1}^{N}$ must be added to the actual MVM outputs along with recurrent MVM outputs. These can be combined and given as input e_m to AFs, according to

$$e_m = \begin{cases} v_m + \mathbf{h}^T\mathbf{t}, & \text{if RST} = 0 \\ v_m + b_m, & \text{otherwise} \end{cases} \tag{27}$$

where RST stands for the reset, which loads either bias term or recurrent vector. The vectors $\mathbf{h}$ and $\mathbf{t}$ in (27) can be defined as $\mathbf{h}, \mathbf{t} = \mathbf{w}, \mathbf{u}$ or $\mathbf{r}, \mathbf{y}$. This operation can be performed by feeding the actual MVM outputs to the BTLU with $\mathbf{b}_s$ through a 2-to-1 multiplexer, as shown in Fig. 9, where the inputs to BTLU are either coming from TDLU or the bias terms. From Structure I of MVM, the details of EWM can be extracted through their operator relation. Specifically, an EWM would consist of a PPG, a PPS, a PDU, and an SAU.

4) Architectural Details of Structure II: In Structure I, SAU takes the AT output and performs its operation for B_γ clock cycles to produce v_m. This implementation introduces several clock cycles of latency, which worsens with higher input word lengths, particularly with lower digit sizes. It can be overcome by increasing γ until it reaches $\gamma = B$, i.e., a bit-parallel DA. However, it requires a lot of adders in the PPG, which is not desirable. Unrolling the SAU to a shift-to-add tree (SAT) in Structure II addressed the aforementioned problem, albeit at the expense of an array of PPSs (PPSA) and ATs. In contrast, it eliminates the need for PISO in PDU, as all the input digits are used simultaneously.

The partial products Xw_{mn} with input digits $u_{n,(\gamma k:\gamma k+\gamma-1)}$ are added together through an AT$_{k+1}$ for each k such that $0 \le k \le B_\gamma - 1$. Each output of AT$_{k+1}$ would experience an effective left shift of $2\lfloor (B_\gamma + 1)/2 \rfloor$ along the (B_γ)th branch of the SAT, as shown in Fig. 10. In general, SAT possesses $\lceil \log_2 B_\gamma \rceil$ stages to add the outputs from all AT$_{k+1}$ to produce v_m. Note that CSU, AT, TDLU, and BTLU are identical to Structure I, as shown in Fig. 10. In this case, the details of EWM can be extracted from Structure II of MVM based on their operator relation. Specifically, it would consist of a PPG, a PPSA, a DU, and an SAT.

C. Top-Level Architecture

The hardware architecture of the considered LSTM RNN is shown in Fig. 11. It comprises MVMs, EWMs, AFs, and block RAMs. The parameter matrices $\mathbf{W}s$ or $\mathbf{R}s$, the input $\mathbf{x}^t$, and the recurrent output $\mathbf{y}^{t-1}$ are involved in the MVMs. The output of MVMs is added to their bias terms $\mathbf{b}_s$ $(s = z, i, f, o)$. These are subsequently passed through $\sigma(\cdot)$ and $\tanh(\cdot)$ to produce the gate outputs $\mathbf{i}^t$, $\mathbf{f}^t$, $\mathbf{o}^t$, and $\mathbf{z}^t$. Similarly, the candidate memory output $\mathbf{c}^t$ is computed. The weights for each gate are stored in a block RAM (RAM$_s$); similarly, previous cell and output values are stored in separate block RAMs (RAM$_{c,y}$), as shown in Fig. 11. The AFs are assumed to be identical as in [37] for reasonable prediction accuracy. It may be noted that once $\mathbf{i}^t$, $\mathbf{f}^t$, $\mathbf{o}^t$, and $\mathbf{z}^t$ are updated, $\mathbf{c}^{t-1}$ is read from RAM$_c$. This allows the computation of $\mathbf{c}^t$, as per (5), which is then written back to RAM$_c$.

D. CPD Analysis

In Sections II-B and II-C, the architectural details of Structures I and II to realize the MVMs and EWMs were discussed. Depending on the selected design parameters, MVMs/EWMs, along with AFs, will result in some computational delays, as shown in Fig. 11. For a systematic design procedure, it is

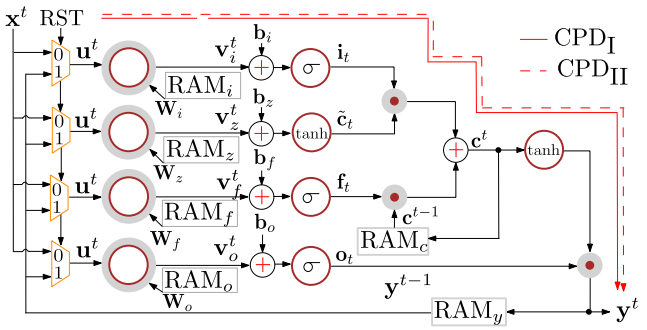


Fig. 11. Top-level architecture of the LSTM RNN. RAM_s/$\text{RAM}_{c,y}$ indicate the block RAMs to store weights for the gates/intermediate cell and output values respectively, where $s = i, f, o,$ and z.

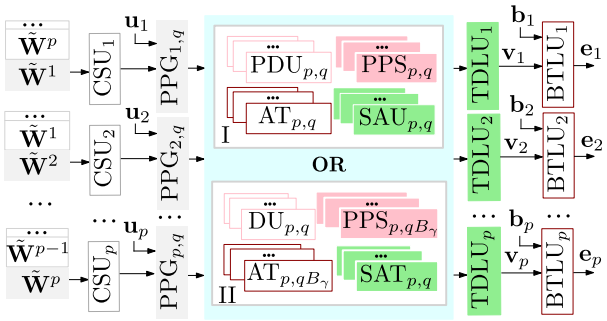


Fig. 12. Framework of the Structures I and II with block-circulant MVMs.

necessary to conduct a CPD analysis. Due to its folded nature, Structure I is considered nonpipelined, while Structure II is regarded as pipelined, allowing for exploration of the design space. In general, the CPD of Structures I and II based on Fig. 11 can be estimated as

$$\text{CPD}_{\text{K}} = \max\left\{\text{CPD}_{\text{MVM}_{\text{K}}}, 2\text{CPD}_{\text{EWM}_{\text{K}}} + T_\sigma + T_{\text{A}} + T_{\tanh}\right\} \tag{28}$$

where $\text{CPD}_{\text{MVM}_{\text{K}}}$ and $\text{CPD}_{\text{EWM}_{\text{K}}}$ are the CPDs of MVM and EWM of the Structures I and II, respectively, with $\text{K} \in \{\text{I}, \text{II}\}$. The computational delays of adder and AFs are denoted by T_{A} and $T_\sigma / T_{\tanh}$, respectively. In the following, the CPD analysis of Structures I and II is carried out.

1) Structure I: The CPD of Structure I shown in Fig. 9 includes the computational delays of CSU (T_{CSU}), PPG (T_{PPG}), PPS (T_{PPS}), AT (T_{AT}), and SAU (T_{SAU}), as shown by a red solid line in Fig. 9. Since SAU completes the operation in B_γ clock cycles, its effective CPD (i.e., the reciprocal of the sample rate) is given by

$$\text{CPD}_{\text{MVM}_{\text{I}}} = B_\gamma (T_{\text{CSU}} + T_{\text{PPG}} + T_{\text{PPS}} + T_{\text{AT}} + T_{\text{SAU}}) \tag{29}$$

where $T_{\text{CSU}} = T_{\text{MX}}$, $T_{\text{PPG}} = (\gamma - 1)T_{\text{A}}$, $T_{\text{PPS}} = T_{\text{AND}} + (\gamma - 1)T_{\text{OR}}$, $T_{\text{AT}} = \lceil \log_2 N \rceil T_{\text{A}}$, and $T_{\text{SAU}} = T_{\text{A}} + T_{\text{D}}$; and T_{MX}, T_{AND}, T_{OR}, and T_{D} are the computational delays of 2-to-1 multiplexer, AND, OR, and D flip-flop, respectively. Likewise, the effective $\text{CPD}_{\text{EWM}_{\text{I}}}$ can be estimated by setting $T_{\text{MX}} = T_{\text{AT}} = 0$ in (29). However, it needs to be multiplied by 2 and include the computational delays of the adder and AF. This is because both the EWMs in (5) can simultaneously perform their operation in B_γ clock cycles, while the EWM in (6) requires another set of B_γ clock cycles to perform its operation. Thus, EWMs require $2B_\gamma$ clock cycles in total to compute the LSTM RNN output based on Structure I.

2) Structure II: Due to the cascaded PPG, PPS, AT, and SAT in Structure II as shown in Fig. 10, the computational delay could be very high. This is overcome through the pipelining of depth h_1 by combining PPG/PPS, AT, and SAT modules. Corresponding to h_1, the number of pipelined registers and clock cycle latency can be decided. Unlike Structure I, the effective CPD of Structure II is the same as its actual CPD, which is

$$\text{CPD}_{\text{MVM}_{\text{II}}} = (T_{\text{CSU}} + T_{\text{PPG}} + T_{\text{PPS}} + T_{\text{AT}} + T_{\text{SAT}})/(h_1 + 1) \tag{30}$$

where $T_{\text{SAT}} = \lceil \log_2 B_\gamma \rceil T_{\text{A}}$. The computational delays of PPG, AT, and SAT are relatively higher than CSU and PPS. Therefore, the computational delays of CSU and PPS can be ignored in the CPD analysis for analytical purposes, i.e., $T_{\text{CSU}} + T_{\text{PPG}} + T_{\text{PPS}} + T_{\text{AT}} + T_{\text{SAT}} \approx \lceil \log_2(\gamma N B_\gamma) \rceil T_{\text{A}} = \lceil \log_2(N B) \rceil T_{\text{A}}$. Observably, for higher γ, PPG becomes more complex and the depth of SAT decreases. With pipelining depth h_1, Structure II comprises $(h_1 + 1)$ CPD sections with almost the same delays. Among them, the most critical is the one involving PPS. In general, these can be expressed as

$$\text{CPD}^1_{\text{MVM}_{\text{II}}} \approx \cdots \approx \text{CPD}^{h_1+1}_{\text{MVM}_{\text{II}}} = \left\lceil \log_{2^{(h_1+1)}}(NB) \right\rceil T_{\text{A}}. \tag{31}$$

Thus, the CPD of Structure II with pipelining depth h_1 is

$$\text{CPD}_{\text{MVM}_{\text{II}}} = T_{\text{PPS}} + \left\lceil \log_{2^{(h_1+1)}}(NB) \right\rceil T_{\text{A}}. \tag{32}$$

Likewise, $\text{CPD}_{\text{EWM}_{\text{II}}}$ with pipelining depth h_2 can be estimated by setting $N = 1$ in (32). This allows the CPD estimation of the second component in (28), which involves two adders, two AFs, and two EWMs. Therefore, it is appropriate to choose a pipelining depth of $2h_2$, as shown in Fig. 10. From the previous discussion, it is clear that the different path delays are set to be balanced if the pipelining depth h_1 in an MVM and pipelining depth h_2 in an EWM, along with an AF/adder, are chosen to make path delays approximately equal. In such a case, it is always recommended to pipeline an EWM first followed by an MVM due to its smaller size. The other approach to pipeline Structure II is to make the path delays unbalanced, i.e., not satisfying (31).

E. Block-Circulant MVMs Based on Structures I and II

Referring to (13), it can be noted that sub-MVMs $\tilde{\mathbf{W}}^{ij}\mathbf{u}^j$ for any j can be computed in parallel. Hence, the MVM given in (12) based on Structure I requires p CSUs, p PDUs, p PPSs, p PPGs, p ATs, p SAUs, p TDLUs, and p BTLUs. The CSU consists of q registers and q 2-to-1 multiplexers. The PDU comprises q PISOs and q DUs. The PISO consists of γB_γ registers, and the DU consists of 2^γ AND gates. The PPS consists of $q2^\gamma$ AUs and $q(2^\gamma - 1)$ OUs, where each AU and OU involve $(B + \gamma)$ AND and OR gates, respectively. The AT consists of $(q-1)$ adders. The SAU consists of an adder and a register. The TDLU consists of $(q-1)$ registers, while BTLU consists of q registers, q adders, and q 2-to-1 multiplexers. This estimation also holds for Structure II except that p PDUs are replaced by pB_γ DUs (i.e., without PISOs) and p PPSs are replaced by p PPSAs, where each PPSA involves B_γ PPSs

and an AT. A framework to obtain Structures I and II with any p and q is shown in Fig. 12. The subscript p, q indicates the size of the component involved in Structures I and II. Although Structures I and II need to be modified for block-circulant MVMs, the details of their EWMs remain the same. The integration of the proposed structures for MVM/EWM enables an optimal tradeoff between system throughput and circuit complexity through the appropriate selection of design parameters, as discussed in the following.

F. Design Methodology

To assess the effectiveness of Structures I and II, selecting appropriate design parameters is crucial. These parameters encompass the model size N, kernel size q, digit size γ, word length B, and clock frequency f_{clk}. In previous works [12], [13], [14], [34], the throughput (η) of an LSTM RNN is given by

$$\eta = q f_{\text{samp,clk}}/p \tag{33}$$

where $f_{\text{samp,clk}}$ represents the sample rate. In the context of DSDA, it is related to the clock frequency as $f_{\text{samp,clk}} = f_{\text{clk}}/B_\gamma$. For a given N, a higher q (or lower p) and/or $f_{\text{samp,clk}}$ consistently enhance throughput performance, while the opposite holds true. The determination of $f_{\text{samp,clk}}$ for Structures I and II depends on their effective CPDs, as per (28)–(32). Individual CPDs are influenced by the clock frequency f_{clk}, which can be optimized through the synthesis tool. Utilizing (33), η of Structures I and II in terms of f_{clk} and other design parameters can be expressed as

$$\eta_{\text{I}} = (q\gamma/pB) f_{\text{clk}}, \quad \& \quad \eta_{\text{II}} = (q(h+1)/p) f_{\text{clk}} \tag{34}$$

where h denotes the cumulative pipelining depth within Structure II, spanning an MVM (PPG/PPS, AT, and SAT), an EWM (PPG/PPS and SAT), and AFs/adders. By employing (34), the parameters q and γ in η_{I} can be consolidated into a composite factor denoted as $Q = q\gamma$. Similarly, the parameters q and $(h+1)$ in η_{II} can be fused into a unified factor $Q = q(h+1)$. These factors for Structures I and II are termed effective speed. Due to the nonpipelined characteristic of Structure I, it is apt for configuring a low-throughput LSTM RNN engine. In contrast, Structure II, with its clock rate aligning with the sample rate due to its unrolled nature, is conducive for designing a high-throughput LSTM RNN engine. According to (34), it is evident that the design parameters q and γ in Structure I can be adjusted to achieve the same η_{I}. Similarly, the design parameters q and h in Structure II can be tailored for the same η_{II}. One way to comprehend this is by examining Q takes on values from the set $\{1, 2, 4, 8\}$. For each value of Q, Structure I will consider combinations of γ and q from the set $(q, \gamma) \in \{(4, 1), (2, 2), (1, 4)\}$, while Structure II will consider combinations of h and q from the set $(q, h) \in \{(4, 0), (2, 1), (1, 3)\}$. Generally, for any effective speed Q, the design parameters (q, γ) for Structure I and (q, h) for Structure II will have $(\lceil \log_2 Q \rceil + 1)$ possible combinations. This will empower the designer to select the optimal design parameters (q, γ) for Structure I and (q, h) for Structure II based on specific requirements.

For designing an LSTM RNN with a model size of $2N$, the viable options involve either doubling p or q. This decision affects the choice of design parameters γ and h as follows.

1) *Structure I:* Doubling p yields $2N$ but halves η_{I}, a result equivalently achieved by reducing the effective speed Q to $Q/2$. Similarly, doubling q produces $2N$, a result also attained by increasing Q to $2Q$.
2) *Structure II:* Doubling q results in $2N$ but doubles η_{II}, an outcome similarly achieved by increasing the speedup Q to $2Q$. Likewise, doubling p leads to $2N$, a result also obtained by reducing Q to $Q/2$.

IV. Results and Discussion

A. Implementation Preliminaries

1) Synthesis Procedure: ASIC synthesis was conducted to extract information on the area, power, and timings of the proposed structures. The Synopsys Design Compiler (DC) was utilized for synthesis, taking input from: 1) a Verilog (.v) file; 2) an FDSOI 28-nm library (.lib); 3) a Synopsys design constraint (.sdc) file; and 4) a Technology library exchange format (.lef). The synthesis considered the worst case process corner with a temperature of 125 °C and a voltage of 1.0 V. Following training with the TIMIT dataset, the proposed structures underwent functional validation using Synopsys VCS, with the outputs generating a value change dump (.vcd) file for estimating switching activity. Subsequently, the tool performed a final synthesis to generate reports on area, power, and timing. The synthesis tool exhibits varying behavior for each clock frequency within a set range (100 MHz–4 GHz). For example, at lower clock frequencies, the area remains relatively constant, while at higher frequencies, the tool optimizes timing at the expense of area and power. To assess the efficiency of the proposed structures, two clock frequencies, $f_{c,\text{clk}}$ and $f_{m,\text{clk}}$, were considered. Here, $f_{c,\text{clk}}$ represents the clock frequency at which the area is roughly 10% of the constant area at lower frequencies, and $f_{m,\text{clk}}$ is the point at which the area is maximized at higher frequencies. Notably, the maximum area corresponds to timing violations by the synthesis tool. Generally, at $f_{c,\text{clk}}$, the hardware elements of the design primarily contribute to area and power. In contrast, at $f_{m,\text{clk}}$, synthesis tools make additional efforts to optimize timings, even at the cost of area and power. The inclusion of $f_{m,\text{clk}}$ alongside $f_{c,\text{clk}}$ serves to evaluate the proposed structures for sample rate improvement.

To obtain more precise area and power estimations, additional tools, such as Synopsys IC Compiler, PrimeTime, and Power Compiler, were employed. Before this, a formal verification of gate-level netlists for the proposed structures was conducted in ModelSim. Precisely, Synopsys IC Compiler received input from Synopsys DC, along with preliminary timing information, to execute the floorplan and place and route (PnR) processes. A hierarchical approach was implemented, performing a complete PnR routine across different submodules sequentially and reaching the top level. This process adhered to a set of plan groups for floorplanning. The tool then assessed the routability and congestion of the plan groups. I/O pin locations were assigned using an input–output

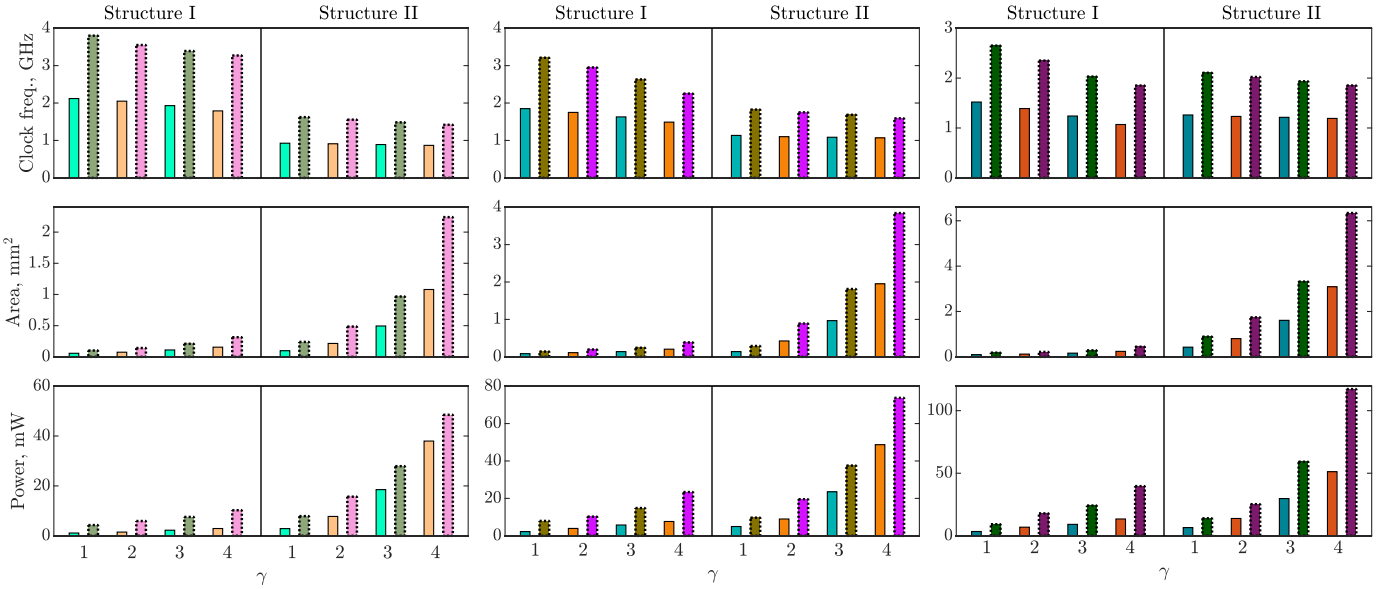


Fig. 13. Clock frequency, area, and power tradeoffs of Structures I and II for $N = 128$ with $\gamma \in \{1, 2, 3, 4\}$; $h = 2 + \lceil \log_2(2q) \rceil$ (in Structure II); and $q \times q = 2 \times 2$ (left), $q \times q = 4 \times 4$ (middle), and $q \times q = 8 \times 8$ (right) at $f_{c,\text{clk}}$ (□) and $f_{m,\text{clk}}$, (⬚).

constraint (.ioc) format, ensuring the check passed; otherwise, a strategy such as incremental spacing was adopted to make them routable. Following the floorplan and PnR, static timing analysis (STA) was conducted using Synopsys PrimeTime to identify any timing violations (setup or hold time). If violations were detected, the tool corrected them, introducing additional buffers in clock tree synthesis at the expense of increased area and power. Finally, Synopsys Power Compiler utilized the post-layout netlist and annotated information to derive fairly accurate post-layout power figures [40].

2) Setting of Design Parameters: The word lengths of inputs and weights were set to 8 bits for fair accuracy, as detailed in Table I. The proposed structures can also be adapted for a different word length, as outlined in (21). For speech recognition task with the TIMIT dataset, a model size of $N = 128$ is found to deliver satisfactory performance [38]. For implementation purposes, $\gamma \in \{1, 2, 3, 4\}$ and $q \in \{2, 4, 8\}$ are considered for both the proposed structures. To determine the pipelining depth h in Structure II, a balanced CPD strategy is adopted by placing registers in AT and/or SAT of MVM/EWM. This ensures that the CPD is approximately equal to $\lceil \log_2(B/2)T_A \rceil$. Under such circumstances, h_1 and h_2 based on the rule $h_1 = \lceil \log_2(2q) \rceil$ with $h_2 = 1$ are selected. Consequently, the total pipelining depth for Structure II is estimated as $h = h_1 + 2h_2 = \lceil \log_2(2q) \rceil + 2$.

B. Performance Evaluation

1) Clock Frequencies, Chip Area, and Power Consumption: For evaluating the proposed structures, $f_{c,\text{clk}}$ (solid bars) and $f_{m,\text{clk}}$ (dotted bars) are considered as sufficient benchmarks. These clock frequencies were derived from synthesis results, as illustrated in Fig. 13. Structure I, despite having a longer CPD than Structure II at higher h, can achieve higher clock frequencies for lower values of γ and q due to its smaller size. In the case of Structure I, the synthesis tool encounters a less pronounced CPD for higher γ and q, prompting more aggressive optimization attempts to meet timing requirements at the expense of area and power. Conversely, in Structure II, with its pipelined nature and smaller CPD, the tool is less inclined to prioritize timing optimization over area and power considerations. It is worth noting that the clock frequencies of Structure II increase with higher h to compensate for the delay of adders in PPG and AT. In contrast, the CPD of Structure I rises with higher γ and q, limiting the values of $f_{c,\text{clk}}$ and $f_{m,\text{clk}}$. Observably, $f_{c,\text{clk}}$ and $f_{m,\text{clk}}$ of Structure I decrease at a more pronounced rate than Structure II for each γ and q due to its nonpipelined nature. Meanwhile, in Structure II, there is a marginal reduction attributed to a slight increase in T_{PPS} for higher γ and q.

The area and power characteristics of Structures I and II, under the specified settings, at $f_{c,\text{clk}}$ (solid bars) and $f_{m,\text{clk}}$ (dotted bars) are depicted in the second and third rows of Fig. 13. Notably, the area and power of both structures at $f_{c,\text{clk}}$ are consistently smaller than those at $f_{m,\text{clk}}$. Reducing the operating clock frequency can effectively diminish the power consumption of Structures I and II. In the case of Structure II, power can also be managed by optimizing the pipelining depth appropriately. The area and power profiles of Structure I at $f_{c,\text{clk}}$ consistently outperform Structure II for various γ and q values, primarily owing to its fewer hardware elements. This discrepancy arises because, at $f_{c,\text{clk}}$, the synthesis tool has not yet initiated comprehensive timing optimization for each structure. However, at $f_{m,\text{clk}}$, the area of Structure I may not consistently be smaller than that of Structure II, as the synthesis tool behaves differently at higher clock frequencies. Notably, the tool exerts more effort for large CPDs, especially in the case of Structure I (particularly for higher γ and q), leading to increased area and power consumption. For instance, the area of Structure I at $f_{m,\text{clk}}$ approximately equals that of Structure II at $f_{c,\text{clk}}$ with $\gamma = 1$ and $q = 4$. In Structure II, both area and power are higher due to the timing optimization performed by the synthesis tool for higher γ and q. It is worth observing that the area and power

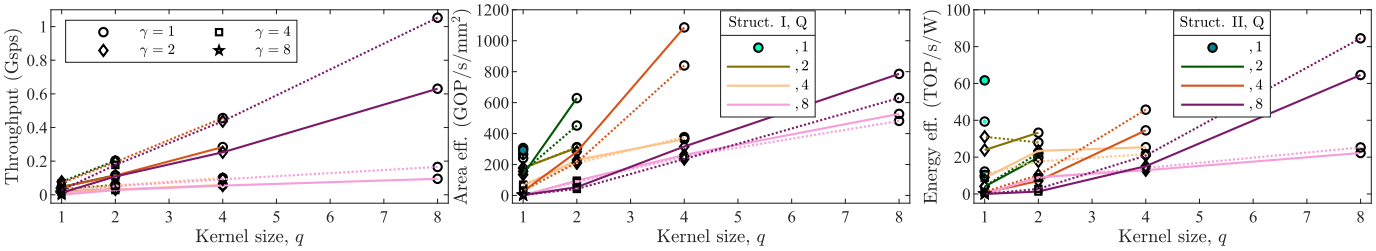


Fig. 14. Throughput, area, and energy efficiencies tradeoff of Structures I and II for various design parameters at $f_{c,\text{clk}}$ (—) and $f_{m,\text{clk}}$ ($\cdots$).

TABLE II

COMPARISON OF HARDWARE COMPLEXITIES AND LATENCY OF DIFFERENT BLOCK-CIRCULANT MVMS BASED LSTM RNNS

Design	**MULT**	**PPG/PPS$_\text{E}$** {ADD, CLG, REG}	**PPG/PPS$_\text{M}$** {ADD, CLG, REG}	**SAU/SAT** {ADD, MUX, REG}	**CSU** {MUX, REG}	**Latency**
LSTM$_1$ [12]	$pq(3+4q)$	–	$4pq \times \{q-1, \text{-}, q-1\}$	–	$-, 4pq(q-1)$	$\log_2 q$
LSTM$_2^\dagger$ [13]	$3pq$	–	$4p \times \{2^{\gamma q/2}(1+\log_2(\gamma q))$ $q2^{\gamma(q-1)}, 2^{\gamma q/2}(1+\log_2(\gamma q))\}$	$4pq \times \{1,1,2\}$	$-, 4pq(q-1)$	$\log_2 q+$ $2+B_\gamma$
LSTM$_{3,\text{I}}^\ddagger$ [14]	0	$3pq \times \{B_\gamma - 1 + 2^\gamma,$ $B_\gamma 2^{\gamma-1}, B_\gamma\}$	$4p \times \{q(2^\gamma-1), q2^\gamma, -\}$	$4p \times \{3,1,2\}$	$16p(q-1)B_\gamma,$ $8p(q-1)B_\gamma$	$q+6$
LSTM$_{3,\text{II}}^\ddagger$ [14]	0	$3pq \times \{B_\gamma - 1 + 2^\gamma,$ $B_\gamma 2^{\gamma-1}, B_\gamma\}$	$4p \times \{q(2^\gamma-1)+B_\gamma(q-1),$ $qB_\gamma 2^{\gamma-1}, B_\gamma(q+2^{\log_2 q}-1)\}$	$4p \times \{B_\gamma+q-1, -,$ $B_\gamma+q-2\}$	$4p(2\lceil\log_2 q\rceil-1),$ $4pq(q-1)$	$\log_2(qB_\gamma)+$ $q+7$
LSTM$_{4,\text{I}}^\diamond$ [34]	$3q$	–	$4p \times \{2^{\gamma(q-1)}-1, q2^{\gamma q}, 2^{\gamma q}-1\}$	$4pq \times \{1,-,1\}$	$-, 4pq(q-1)$	$B_\gamma+\gamma$
LSTM$_{4,\text{II}}^\diamond$ [34]	$3q$	–	$4p \times \{3.2^{\gamma(q-2)}, q2^{\gamma q}, q\}$	$4pq \times \{1,-,1\}$	$-, 4pq(q-1)$	B_γ
LSTM$_{4,\text{III}}^\diamond$ [34]	$3q$	–	$4p \times \{2.2^{\gamma(q/2-1)}+1, 2q.2^{\gamma q/2}, 2.2^{\gamma q/2}-2\}$	$4pq \times \{1,-,1\}$	$-, 4pq(q-1)$	$B_\gamma+\gamma$
LSTM$_{4,\text{IV}}^\diamond$ [34]	$3q$	–	$4p \times \{2.2^{\gamma(q/2-1)}+1, 2q.2^{\gamma q/2}, q\}$	$4pq \times \{1,-,1\}$	$-, 4pq(q-1)$	B_γ
Structure I$^{\diamond\diamond}$	0	$3pq \times \{2^\gamma-1, 2^\gamma, -\}$	$4p \times \{q(2^\gamma-1), q2^\gamma, -\}$	$p \times \{3q+4, -, 3q+4\}$	$4pq, 4pq$	$q-1$
Structure II$^{\diamond\diamond}$	0	$3pq \times \{2^\gamma-1, 2^\gamma, -\}$	$4p \times \{q(2^\gamma-1)+B_\gamma(q-1),$ $qB_\gamma 2^{\gamma-1}, 2(4^{h_3}-1)/3\}$	$p \times \{(B_\gamma-1)(3q+4),$ $-, 3q(2^{h_2}-1)\}$	$4pq, 4pq$	$q-1+h$

MULT: Multipliers, CLG: Combinational logic gates, †: radix-2 OBC, ‡: radix-2$^\gamma$ OBC, $^\diamond$: radix-2 TC, $^{\diamond\diamond}$: radix-2$^\gamma$ TC, B: Wordlength; subscripts 'E' and 'M' in PPG/PPS$_{\text{E,M}}$: indicate EWMs and MVMs respectively, γ: Digit-size, $B_\gamma = \lceil B/\gamma \rceil$, $N = p \times q$: Model size; q: kernel size; p: Number of sub-MVMs, $h_3 = \lceil \log_2(qB)/h_1 \rceil$. CLG in LSTM$_2$ involves $2^{\gamma(q-1)}$ multiplexers (or $2^{\gamma(q-1)}-1$ 2-to-1 multiplexers). CLG in LSTM$_{3,\text{I,II}}$ involves $2^{\gamma-1}$ multiplexers, which have $2^{\gamma-1} \times (B + \lceil \log_2 \gamma \rceil - 1)$ AND gates, $(2^{\gamma-1}-1) \times (B + \lceil \log_2 \gamma \rceil - 1)$ OR gates and $2 \times (B + \lceil \log_2 \gamma \rceil - 1)$ XOR gates. CLG in LSTM$_{4,\text{I/II/III/IV}}$ involves $2^{\gamma q}$ multiplexers, which have $(2^{\gamma q}-1)$ 2-to-1 multiplexers. CLG in Structure I/II involve $2^{\gamma-1}$ multiplexers, which have $2^\gamma \times (B + \lceil \log_2 \gamma \rceil)$ AND gates, $(2^\gamma - 1) \times (B + \lceil \log_2 \gamma \rceil)$ OR gates, $1 \times (B + \lceil \log_2 \gamma \rceil)$ XOR gates.

consumption of both structures generally increase for higher γ and q.

2) Throughput, Area, and Energy Efficiencies: To achieve specified area and energy (power/frequency) constraints, the area and energy efficiencies in terms of performance can be estimated as

$$\text{Area eff.} = \frac{\text{Performance}}{\text{Chip-area}}, \quad \& \quad \text{Energy eff.} = \frac{\text{Performance}}{\text{Energy}}. \tag{35}$$

Utilizing the definitions of η for Structures I and II provided in (34), their performance [41] can be determined through the following relation:

$$\text{Performance} = \frac{2 \times \text{No. of Parameters in LSTM} \times \eta}{N}. \tag{36}$$

In prior results, the variations in chip area and power consumption of Structures I and II for $N \times N = 128 \times 128$, with varying q, γ, and/or h at $f_{c,\text{clk}}$ and $f_{m,\text{clk}}$ are demonstrated. It is crucial to discern the tradeoffs in area and energy efficiencies at $f_{c,\text{clk}}$ and $f_{m,\text{clk}}$ for Structures I and II, aiming to meet specific throughput (or performance) needs. To determine the throughput of Structures I and II, Q for both structures is again assumed to be in the same set, $Q \in \{1,2,4,8\}$. The corresponding throughput values for different $\gamma \in \{1,2,4,8\}$ values are illustrated in the leftmost section of Fig. 14. For Structure II, an unbalanced CPD strategy is employed, selecting the pipelining depth from the set $h \in \{0,1,3,7\}$ with an appropriate combination of h_1 and h_2. As per (34), both η values exhibit a linear variation with f_{clk} (i.e., $f_{c,\text{clk}}$ and $f_{m,\text{clk}}$) and an inverse relationship with p. In case of Structure I, both f_{clk} and p $(= N/q)$ decrease (see Fig. 13). The decrease in p surpasses the decrease in f_{clk}. In addition, the factor B in the denominator of η_I contributes to slower growth for q. Conversely, f_{clk} of Structure II increases with an increase in q (see Fig. 13), resulting in η_II growing faster than η_I.

As expected, the throughput of both structures at $f_{c,\text{clk}}$ (solid lines) is lower than at $f_{m,\text{clk}}$ (dotted lines), while the throughput remains the same for a given Q in each structure, area, and energy efficiencies that are contingent on the pairs (q,γ) in Structure I and (q,h) in Structure II. Based on throughput values, one can estimate the processing time $(=\eta/N)$ of an LSTM layer, a measure of real-time performance [5], [12]. For example, the estimated processing times of Structures I and II for $N = 128, q = 4, \gamma = 2$, and $B = 8$ are 0.19 and 3.91 μs, respectively. The proposed structures meet the minimum criteria essential for the speech recognition tasks [5]. The middle and rightmost sections of Fig. 14 depict the area and energy efficiencies of Structures I and II for $Q \in \{1,2,4,8\}$, respectively. According to (35), these efficiencies are gauged concerning unit area and unit energy, where lower area and energy values signify enhanced design efficiency. Observing Fig. 14, it is apparent that the area efficiencies of Structures I and II at $f_{c,\text{clk}}$ surpass those at $f_{m,\text{clk}}$ due to a smaller chip area. However, the converse holds true for energy efficiencies due to faster operations at $f_{m,\text{clk}}$ and $f_{c,\text{clk}}$. For a given Q, an increase in γ (or a decrease in q) and/or h will diminish both area and energy efficiencies. To achieve an area-efficient design, it is advisable to opt for Structure II with large q and small γ, and Structure I for small q and large γ, as they offer high performance. Conversely, for energy-efficient designs, the reverse choices are recommended.

C. Hardware Complexities and Latency

For clarity, the existing works in [12], [13], [14], and [34] are referred to as LSTM_{1-4}, respectively. Note that LSTM_3 encompasses two distinct designs, labeled as $\text{LSTM}_{3,\text{I/II}}$, while LSTM_4 incorporates four distinct designs, identified as $\text{LSTM}_{4,\text{I/II/III/IV}}$. The hardware complexities of different designs are summarized in Table II. To enhance clarity, these complexities are expressed in terms of PPGs/PPSs involved in MVMs and EWMs, denoted as PPG/PPS_M and PPG/PPS_E for MVMs and EWMs, respectively. These estimates are further detailed in terms of adders (ADD), combinational logic gates (CLGs), and registers (REG). For a fair comparison, the hardware elements in SAU/SAT and CSU are also presented. Note that CLG refers to the logic gates involved in PPGs/PPSs for different designs. The hardware involved in PISO, TDLU, BTLU, and AFs is assumed to be the same for all the designs. For clarity, various DA-based designs are classified based on radix size and coding scheme, as listed in the footnote of Table II. Although $\text{LSTM}_{2,4}$ were developed for radix-2 implementation, it is fair to compare their complexities for any radix-2^γ. Compared with $\text{LSTM}_{1,2,4}$, the proposed structures do not incorporate binary multipliers in MVMs, EWMs, or both.

In the proposed methodology, each EWM follows the MVM design based on Structures I and II according to their operator relation. Consequently, the overall complexities associated with ADD, REG, and MUX for Structures I and II are reduced compared to LSTM_3. It can be noted that $\text{LSTM}_{2,4}$ have lower CLG than TC, but they require explicit generation of the offset terms in contrast to the proposed structures. Furthermore, Structure I, as opposed to $\text{LSTM}_{3,\text{I}}$, does not require a time-multiplexing unit, partial-product sorter, modulo-cum interleaver, offset generator, multiple rotators, multiple PPSs, and multiple SAUs. This is because Structure I performs the IPC row-wise and computes the partial products of the weights with the input digit slices. As a result, the complexities of ADD, CLG, MUX, and REG in Structure I are lower than $\text{LSTM}_{3,\text{I}}$. The same argument applies to Structure II compared to $\text{LSTM}_{3,\text{II}}$. Moreover, Structures I and II have simpler CSU compared to LSTM_{1-3}. For more clarity, the hardware complexities of existing designs for $N = 128$, $q = 4$, $\gamma = 2$, and $B = 8$ are listed in Table III. Clearly, Structure I offers 28.21% fewer ADD, 60.00% fewer REG, 66.67% fewer MUX, and 37.5% more CLG compared to $\text{LSTM}_{3,\text{I}}$.

The latency of various DA-based designs is also presented in Table III. Due to fine-grained pipelining in LSTM_2 and $\text{LSTM}_{3,\text{II}}$, the number of registers is significantly high. In $\text{LSTM}_{4,\text{I/III}}$, few pipelined registers are strategically placed to maximize the gain in clock frequency, while $\text{LSTM}_{4,\text{II/IV}}$ are nonpipelined counterparts of $\text{LSTM}_{4,\text{I/III}}$. Compared with $\text{LSTM}_{3,\text{II}}$, Structure II has less number of pipelined registers and latency as it is relied on choosing the pipelining depth on CPD analysis. Since Structure I is nonpipelined, the only latency is due to TDLU registers, while Structure II has an additional latency of h due to its pipelined nature. From Table III, it is clear that $\text{LSTM}_{3,\text{I}}$ offers 33.33% less sample clock latency compared to $\text{LSTM}_{4,\text{IV}}$.

TABLE III

THEORETICAL ESTIMATES OF HARDWARE COMPLEXITIES AND LATENCY LISTED IN TABLE II FOR $N = 128$, $q = 4$, $\gamma = 2$, AND $B = 8$

Design	MULT	ADD	REG	MUX	*CLG	Latency
LSTM_1 [12]	2432	1536	1536	–	–	2
$\text{LSTM}_2^\dagger$ [13]	384	8192	10752	512	16192	8
$\text{LSTM}_{3,\text{I}}^\ddagger$ [14]	0	3546	2560	1536	13440	10
$\text{LSTM}_{3,\text{II}}^\ddagger$ [14]	0	6656	7424	896	36480	15
$\text{LSTM}_{4,\text{I}}^\diamond$ [34]	12	8586	34304	–	131072	6
$\text{LSTM}_{4,\text{II}}^\diamond$ [34]	12	6656	2560	–	131072	4
$\text{LSTM}_{4,\text{III}}^\diamond$ [34]	12	4736	5504	–	16384	6
$\text{LSTM}_{4,\text{IV}}^\diamond$ [34]	12	4736	2560	–	16384	4
Structure $\text{I}^{\diamond\diamond}$	0	2048	1024	512	21504	3
Structure $\text{II}^{\diamond\diamond}$	0	5760	4224	512	58368	8

†: radix-2 OBC, ‡: radix-2^γ OBC, $^\diamond$: radix-2 TC, *: estimated CLG of Structure I/II and $\text{LSTM}_{2,3,4}$ (listed at Table II footnote) are normalized with B.

D. Performance Comparison

Based on the specified synthesis methodology, a performance comparison of the proposed structures with LSTM_{1-4} baselines [12], [13], [14], [34] and other designs [7], [8], [9], [10], [11] is carried out. As reported in LSTM_3 [14], $\text{LSTM}_{1,2}$ were resynthesized under identical conditions on a 28-nm low-power standard cell FDSOI library. Hence, it was appropriate to synthesize the proposed structures on the same technology node/type. Note the designs in LSTM_4 were implemented on FPGA, and they are resynthesized on ASICs for a fair comparison. Furthermore, all the baselines and proposed structures were synthesized with design parameters $N = 128$ with $q = 4$ at 800 MHz and 8 bits (at $\gamma = 1$). In addition, the proposed structures were also synthesized for a larger network model with design parameters $N = 512$ with $q = 16$ at 600 MHz to evaluate their architectural efficiency for a better PER requirement in speech recognition tasks [9], [12].

The obtained post-synthesis results of Structures I and II are listed in Table IV. It can be noted that the pipelining depth in Structure II was chosen under a balanced CPD strategy. In LSTM_1, MVMs/EWMs were realized with bit-parallel multipliers. They were implemented in LSTM_2/LSTM_4 and LSTM_3 using PPGs/PPSs to realize a parallel LUT in radix-2 OBC/radix-2 TC and a serial LUT in radix-2^γ OBC, respectively. Notably, Structures I and II consume less area and power than their high-radix counterparts $\text{LSTM}_{3,\text{I}}$ and $\text{LSTM}_{3,\text{II}}$, respectively. While they offer significantly higher area and energy efficiencies compared to $\text{LSTM}_{4,\text{I/II/III/IV}}$. Compared with $\text{LSTM}_{3,\text{I}}$, the proposed Structure I occupies 28.50% less area, consumes 23.61% less power, provides 39.87% more area efficiency, and exhibits 30.95% more energy efficiency. In contrast, Structure II, when compared with $\text{LSTM}_{4,\text{IV}}$, offers 26.71% more area, consumes 57.91% more power, provides 95.73% more area efficiency, and exhibits 91.18% more energy efficiency.

The other ASIC-based LSTM RNNs [7], [8], [9], [10], [11] were either post-synthesized or fabricated on CMOS technology. To enable a fair comparison, their results were either scaled or resynthesized to 28-nm CMOS. It is noteworthy to mention that scaling to FDSOI technology cannot be applied since it relies on a different process flow. However, it offers superior performance than CMOS [15], necessitating the resynthesis of other designs. Unlike LSTM_{1-4}, the designs in [7], [8], [9], [10], and [11] had employed different

TABLE IV
PERFORMANCE COMPARISON OF THE PROPOSED AND EXISTING ASIC-BASED LSTM RNN ACCELERATORS

	Design	Tech. (nm) ($\mathcal{N}, \mathcal{P}, \mathcal{T}$)	$N \times N$	Volt. (V)	Quant. (bit)	Memory kB	MACs	f_{clk} (MHz)	Area (mm^2)	Power (mW)	η (Msps)	Perf. (GOP/s)	Energy eff. (TOP/s/W)	Area Eff. (GOP/s /mm^2)
Baselines	♣$LSTM_1$ [12]	28,FDSOI,*LP*	128 × 128*	1.0	8-12	33.625	608	800	0.251	12.048	100	53.20	25.97	211.95
	♣$LSTM_2$ [13]	28,FDSOI,*LP*	128 × 128*	1.0	8-12	33.625	384	800	0.158	5.632	12.5	6.65	1.18	42.09
	♣$LSTM_{3,I}$ [14]	28,FDSOI,*LP*	128 × 128*	1.0	8-12	33.625	0	800	0.107	1.808	12.5	6.65	2.94	62.15
	♣$LSTM_{3,II}$ [14]	28,FDSOI,*LP*	128 × 128*	1.0	8-12	33.625	0	800	0.173	10.944	500	266.00	24.31	1537.57
	♣$LSTM_{4,I}$ [34]	28,FDSOI,*LP*	128 × 128*	1.0	8-12	33.625	12	800	0.144	4.691	12.5	6.65	1.41	46.18
	♣$LSTM_{4,II}$ [34]	28,FDSOI,*LP*	128 × 128*	1.0	8-12	33.625	12	800	0.156	5.439	12.5	6.65	1.22	42.62
	♣$LSTM_{4,III}$ [34]	28,FDSOI,*LP*	128 × 128*	1.0	8-12	33.625	12	800	0.101	2.685	12.5	6.65	2.47	65.84
	♣$LSTM_{4,IV}$ [34]	28,FDSOI,*LP*	128 × 128*	1.0	8-12	33.625	12	800	0.096	2.494	12.5	6.65	2.66	69.27
Others	♣♣ [8]	65,CMOS,*LL*	96 × 96	1.24	8	84	96	168	0.93	29.03	20.92	32.3	1.11	34.4
	♣ [7]	65,CMOS,–	128 × 128	1.1	8-11	106	772	322	2.62	20.4	13.13	27.0	1.32	10.3
		28,CMOS,*S*	128 × 128	1.0	8-11	106	772	749	0.48	7.27	30.54	62.81	8.64	130.85
	♣♣ [9]	65,CMOS,*LP*	64 × 64	1.1	13	297	64	80	7.74	67.3	159.83	164.95	2.45	–
		28,CMOS,*S*	64 × 64	1.0	13	297	64	186	1.44	24.09	371.03	382.92	2.95	–
	♣ [10]	40,CMOS,*LL*	128 × 128	1.1	8-15	88.5	12	200	0.45	6.16	11.67	24	3.89	53.3
		28,CMOS,*S*	128 × 128	1.0	8-15	88.5	12	286	0.22	3.57	16.68	34.28	2.74	155.81
	♣♣ [11]	65,CMOS,*LL*	96 × 96	1.275	8	84	96	159	0.93	30.36	19.77	30.53	1.01	32.8
		28,CMOS,*S*	96 × 96	1.0	8	84	96	369	0.172	7.926	45.88	70.92	8.95	412.32
Proposed	♣Structure I	28,CMOS,*LP*	128 × 128**	1.0	8-12	33.625	0	800	0.0832	1.145	12.5	6.65	4.75	79.92
		28,FDSOI,*LP*	128 × 128**	1.0	8-12	33.625	0	800	0.0765	1.381	12.5	6.65	3.85	86.93
		28,CMOS,*LP*	512 × 512**	1.0	8-12	518.5	0	600	0.412	9.325	37.5	77.57	4.99	188.27
	♣Structure II	28,CMOS,*LP*	128 × 128**	1.0	8-12	33.625	0	800	0.148	4.767	400	212.80	35.72	1437.84
		28,FDSOI,*LP*	128 × 128**	1.0	8-12	33.625	0	800	0.131	5.925	400	212.80	32.48	1624.43
		28,CMOS,*LP*	512 × 512**	1.0	8-12	518.5	0	600	0.651	38.857	2100	4342.80	67.06	6670.96

LEGEND: MACs: Multiply-and-accumulate units, η : Throughput, Perf.: Performance, ♣: Post-synthesis, ♣♣: Fabricated, $\mathcal{N}$: Node, $\mathcal{P}$: Process, and $\mathcal{T}$: Type; *LP*: Low Power; *LL*: Low Leakage, *S*: Scaled. *: results reported in [14], **: results based on considered synthesis methodology. Frequency, area and energy metrics of [7], [9]–[11] are normalized to 28 nm CMOS for a fair comparison with scaling factors $1/S$, S^2, and U^2S^2 respectively [42], where $S = 28\text{nm}/65\text{nm} = 0.43$ and $^{\dagger}U_1 = 1.0\text{V}/1.1\text{V} = 0.91$, $^{\ddagger}U_2 = 1.0\text{V}/1.275\text{V} = 0.78$.

techniques to improve hardware efficiency. For instance, in [8], a single systolic array-based hardware accelerator was introduced to mitigate the need for extensive memory transfers. In [7], an approximate computing unit along with idea of computation overlapping was employed to aggressively reduce area and power consumption. In [9], an efficient approach to address memory storage limitations was presented, involving algorithm-hardware co-optimized memory. In [10], an energy-efficient scalable architecture for LSTM RNN with network compression was proposed. In [11], scalability and memory bandwidth issues arising from feedback in LSTM RNN were tackled. Compared with [11], Structure II occupies 13.95% less area, consumes 39.86% less power, provides 2.48× greater area efficiency, and exhibits 2.01× more energy efficiency.

V. CONCLUSION AND FUTURE DIRECTIONS

In this work, two efficient structures (I and II) for LSTM RNNs based on circulant matrices, leveraging TC DSDA, have been introduced. These structures were founded on a unified formulation for block-circulant MVM/EWM, coupled with an FxP training procedure for speech recognition tasks. Specifically, Structure I involved a nonpipelined MVM/EWM, while Structure II considered their pipelined counterparts, aiming to achieve a balance between area, energy, and throughput under constant speedup scenarios. A distinctive feature of Structures I and II was their utilization of circular rotation of weights and the generation of partial products with input digit slices, effectively reducing energy consumption. In addition, new PPGs and PPSs were introduced to accommodate both unsigned and signed digits. Furthermore, an in-depth CPD analysis of the proposed structures offered insights into their design considerations. Through post-synthesis evaluations with carefully chosen design parameters, both proposed structures demonstrated superior area and energy efficiencies compared to recent designs [14], [34]. This underscored the efficacy of the presented approaches in advancing the state-of-the-art designs of LSTM RNNs.

Expanding on this work, several promising research directions emerge. One avenue is exploring the integration of approximate computing techniques to optimize energy efficiency without compromising performance. In addition, investigating alternative architectures or optimization strategies tailored for real-time or edge computing could yield valuable insights. Extending the analysis to larger scale implementations or exploring parallelism may enhance throughput. Finally, studying the applicability of the proposed structures to other machine learning tasks could broaden the impact of this research.

REFERENCES

[1] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand. (ASRU)*, Dec. 2015, pp. 167–174.

[2] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, *arXiv:1406.1078*.

[3] M. Moradi, S. A. Sadrossadat, and V. Derhami, "Long short-term memory neural networks for modeling nonlinear electronic components," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 11, no. 5, pp. 840–847, May 2021.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[5] M. Lee, K. Hwang, J. Park, S. Choi, S. Shin, and W. Sung, "FPGA-based low-power speech recognition with recurrent neural networks," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2016, pp. 230–235.

[6] A. X. M. Chang and E. Culurciello, "Hardware accelerators for recurrent neural networks on FPGA," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4, doi: 10.1109/ISCAS.2017.8050816.

[7] E. Azari and S. Vrudhula, "ELSA: A throughput-optimized design of an LSTM accelerator for energy-constrained devices," *ACM Trans. Embedded Comput. Syst.*, vol. 19, no. 1, pp. 1–21, Jan. 2020.

[8] F. Conti, L. Cavigelli, G. Paulin, I. Susmelj, and L. Benini, "Chipmunk: A systolically scalable 0.9 mm2, 3.08Gop/s/mW @ 1.2 mW accelerator for near-sensor recurrent neural network inference," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2018, pp. 1–4.

[9] D. Kadetotad, S. Yin, V. Berisha, C. Chakrabarti, and J. Seo, "An 8.93 TOPS/W LSTM recurrent neural network accelerator featuring hierarchical coarse-grain sparsity for on-device speech recognition," *IEEE J. Solid-State Circuits*, vol. 55, no. 7, pp. 1877–1887, Jul. 2020, doi: 10.1109/JSSC.2020.2992900.

[10] J. Wu, F. Li, Z. Chen, and X. Xiang, "A 3.89-GOPS/mW scalable recurrent neural network processor with improved efficiency on memory and computation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 12, pp. 2939–2943, Dec. 2019.

[11] G. Paulin, F. Conti, L. Cavigelli, and L. Benini, "Vau da muntanialas: Energy-efficient multi-die scalable acceleration of RNN inference," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 1, pp. 244–257, Jan. 2022.

[12] Z. Wang, J. Lin, and Z. Wang, "Accelerating recurrent neural networks: A memory-efficient approach," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2763–2775, Oct. 2017.

[13] K. P. Yalamarthy, S. Dhall, Mohd. T. Khan, and R. A. Shaik, "Low-complexity distributed-arithmetic-based pipelined architecture for an LSTM network," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 2, pp. 329–338, Feb. 2020.

[14] M. T. Khan, H. E. Yantir, K. N. Salama, and A. M. Eltawil, "Architectural trade-off analysis for accelerating LSTM network using Radix-r OBC scheme," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 1, pp. 266–279, Jan. 2023.

[15] B.-Y. Nguyen et al., "A path to energy efficiency and reliability for ICs: Fully depleted silicon-on-insulator (FD-SOI) devices offer many advantages," *IEEE Solid-State Circuits Mag.*, vol. 10, no. 4, pp. 24–33, Jan. 2018, doi: 10.1109/MSSC.2018.2867405.

[16] T. Ergen, A. H. Mirza, and S. S. Kozat, "Energy-efficient LSTM networks for online learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 3114–3126, Aug. 2020.

[17] C. Gao, T. Delbruck, and S.-C. Liu, "Spartus: A 9.4 Top/s FPGA-based LSTM accelerator exploiting spatio-temporal sparsity," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 1–15, Jan. 2024.

[18] C. Baskin et al., "Uniq: Uniform noise injection for non-uniform quantization of neural networks," *ACM Trans. Comput. Syst.*, vol. 37, nos. 1–4, pp. 1–15, 2021.

[19] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.

[20] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," 2019, *arXiv:1903.05662*.

[21] S.-E. Chang et al., "Mix and match: A novel FPGA-centric deep neural network quantization framework," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Feb. 2021, pp. 208–220.

[22] Z. Li et al., "E-RNN: Design optimization for efficient recurrent neural networks in FPGAs," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2019, pp. 69–80.

[23] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin, "Extremely low bit neural network: Squeeze the last bit out with ADMM," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–19.

[24] J. Xu, X. Chen, S. Hu, J. Yu, X. Liu, and H. Meng, "Low-bit quantization of recurrent neural network language models using alternating direction methods of multipliers," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 7939–7943.

[25] A. Diro and N. Chilamkurti, "Leveraging LSTM networks for attack detection in fog-to-things communications," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 124–130, Sep. 2018.

[26] S. Han et al., "EIE: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 243–254, 2016.

[27] K. Guo et al., "Angel-eye: A complete design flow for mapping CNN onto customized hardware," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 24–29.

[28] S. Han et al., "ESE: Efficient speech recognition engine with sparse LSTM on FPGA," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2017, pp. 75–84.

[29] C. Cheng and K. K. Parhi, "Hardware efficient fast DCT based on novel cyclic convolution structures," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4419–4434, Nov. 2006.

[30] Z. Li, S. Wang, C. Ding, Q. Qiu, Y. Wang, and Y. Liang, "Efficient recurrent neural networks using structured matrices in FPGAs," 2018, *arXiv:1803.07661*.

[31] S. Liao, Z. Li, X. Lin, Q. Qiu, Y. Wang, and B. Yuan, "Energy-efficient, high-performance, highly-compressed deep neural network design using block-circulant matrices," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 458–465.

[32] A. Fiandrotti, S. M. Fosson, C. Ravazzi, and E. Magli, "GPU-accelerated algorithms for compressed signals recovery with application to astronomical imagery deblurring," *Int. J. Remote Sens.*, vol. 39, no. 7, pp. 2043–2065, Apr. 2018.

[33] M. Tasleem Khan and R. Ahamed Shaik, "Optimal complexity architectures for pipelined distributed arithmetic-based LMS adaptive filter," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 2, pp. 630–642, Feb. 2019.

[34] M. A. Alhartomi et al., "Low-area and low-power VLSI architectures for long short-term memory networks," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 13, no. 4, pp. 1000–1014, Dec. 2023.

[35] Z. Tang et al., "Automatic sparse connectivity learning for neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 1–15, Oct. 2023.

[36] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, "An exploration of parameter redundancy in deep networks with circulant projections," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2857–2865, doi: 10.1109/ICCV.2015.327.

[37] B. Zamanlooy and M. Mirhassani, "Efficient VLSI implementation of neural networks with hyperbolic tangent activation function," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 1, pp. 39–48, Jan. 2014.

[38] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1–1.1," *STIN*, vol. 93, p. 27403, Feb. 1993.

[39] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. Laurent Yoshua Bengio, and A. Courville, "Towards end-to-end speech recognition with deep convolutional neural networks," 2017, *arXiv:1701.02720*.

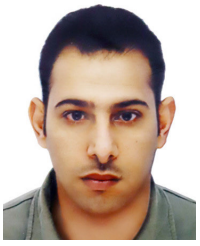
[40] A. Minwegen, D. Auras, and G. Ascheid, "A multimode decision-directed channel estimation ASIC for MIMO-OFDM," in *Proc. IEEE/IFIP 20th Int. Conf. VLSI Syst. Chip (VLSI-SoC)*, Oct. 2012, pp. 65–70.

[41] N. M. Rezk, M. Purnaprajna, T. Nordström, and Z. Ul-Abdin, "Recurrent neural networks: An embedded computing perspective," *IEEE Access*, vol. 8, pp. 57967–57996, 2020.

[42] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*, vol. 2. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

Mohd Tasleem Khan received the B.Tech. degree in electronics engineering from AMU, Aligarh, India, in 2013, and the Ph.D. degree in VLSI from IIT Guwahati, Guwahati, India, in 2019.

Currently, he serves as an Assistant Professor at Heriot-Watt University, U.K. Prior to this position, he served as a Postdoctoral Researcher at Linkoping University, Sweden, an Assistant Professor at the Department of Electronics Engineering, IIT Dhanbad, India, a Research Consultant for KAUST, Saudi Arabia, and a Principal Engineer at TSMC, Taiwan. His research focuses on developing efficient algorithms and architectures for machine learning, signal processing and communication systems. His work spans a variety of application domains, including high-speed and low-power mobile systems, machine learning platforms, and modern DSPs.



Mohammed A. Alhartomi (Member, IEEE) received the B.Sc. degree from King Abdulaziz University, Jeddah, Saudi Arabia, in 2004, the M.Sc. degree in wireless communications systems from Swansea University, Swansea, U.K., in 2011, and the Ph.D. degree from the University of Leeds, Leeds, U.K., in 2016.

He is currently an Associate Professor with the Electrical Engineering Department, University of Tabuk, Tabuk, Saudi Arabia. He is also a Visiting Scholar with the University of Central Florida, Orlando, FL, USA, where he is working on dense visible light communication (VLC) networks. His current research interests include mobile and wireless communications, optical communications, and signal processing.