An 8.93 TOPS/W LSTM Recurrent Neural Network Accelerator Featuring Hierarchical Coarse-Grain Sparsity for On-Device Speech Recognition

Deepak Kadetotad, *Member, IEEE*, Shihui Yin, *Student Member, IEEE*, Visar Berisha, *Member, IEEE*, Chaitali Chakrabarti, *Fellow, IEEE*, and Jae-sun Seo, *Senior Member, IEEE*

***Abstract*— Long short-term memory (LSTM) is a type of recurrent neural networks (RNNs), which is widely used for time-series data and speech applications, due to its high accuracy on such tasks. However, LSTMs pose difficulties for efficient hardware implementation because they require a large amount of weight storage and exhibit computation complexity. Prior works have proposed compression techniques to alleviate the storage/computation requirements of LSTMs but elementwise sparsity schemes incur sizable index memory overhead and structured compression techniques report limited compression ratios. In this article, we present an energy-efficient LSTM RNN accelerator, featuring an algorithm-hardware co-optimized memory compression technique called hierarchical coarse-grain sparsity (HCGS). Aided by the HCGS-based blockwise recursive weight compression, we demonstrate LSTM networks with up to 16× fewer weights while achieving minimal error rate degradation. The prototype chip fabricated in 65-nm LP CMOS achieves up to 8.93 TOPS/W for real-time speech recognition using compressed LSTMs based on HCGS. HCGS-based LSTMs have demonstrated energy-efficient speech recognition with low error rates for TIMIT, TED-LIUM, and LibriSpeech data sets.**

***Index Terms*— Hardware accelerator, long short-term memory (LSTM), speech recognition, structured sparsity, weight compression.**

I. Introduction

THE emergence of the Internet-of-Things (IoT) devices that require edge computing with severe area/energy constraints has garnered substantial interest in energy-efficient ASIC accelerators for deep learning applications. Automatic speech recognition (ASR) is one of the most prevalent tasks that allow such edge devices to interact with humans and have been integrated into many commercial edge devices. Recurrent neural networks (RNNs) are very powerful for speech recognition, combining two properties: 1) distributed hidden state that allows them to store a lot of information about the past efficiently and 2) non-linear dynamics that allow them to update their hidden state in complicated ways. Long short-term memory (LSTM) is a type of RNN with internal gates to scale the inputs and outputs within the cell. LSTM gates avoid vanishing/exploding gradients issue that plagues RNNs [1], but it requires 8× weights compared with a multi-layer perceptron (MLP) that has the same number of hidden neurons per layer.

Due to the large size of the LSTM RNNs that enable accurate ASR, most of these speech recognition tasks are performed in the cloud servers, which requires a constant internet connection, involves privacy concerns, and incurs latency for speech recognition tasks. The particular challenge of performing on-device ASR is that state-of-the-art LSTM-based models for ASR contain tens of millions of weights [2], [3]. Weights can be stored on-chip (e.g., SRAM cache of mobile processors), which has fast access time (nanoseconds range) but is limited to a few megabytes (MBs) due to cost [4]. Alternatively, weights can be stored off-chip (e.g., DRAM) up to a few gigabytes (GBs), but access is slower (tens of nanoseconds range) and consumes ∼100× higher energy than on-chip counterparts [5]. To improve the energy efficiency of neural network hardware, off-chip memory access and communication need to be minimized [6]. To that end, it becomes crucial to store most or all weights on-chip through sparsity/compression, weight quantization, and network size reduction.

Recent works presented methods to reduce the complexity and memory requirements of RNNs for ASR. Load-balance-aware pruning was proposed for LSTM hardware in [7], resulting in 20× model size reduction, but elementwise sparsity incurs considerable index memory and irregular memory access, negatively affecting both performance and power.

To overcome such issues, structured sparsity techniques have been proposed with rowwise/columnwise sparsity for RNNs [8], with blockwise sparsity for MLPs [9] and with block-circulant weight matrix for RNNs [10], [11] in speech applications. However, these works exhibit limited weight compression of ∼4× [8], [9] and high error rate [10], [11] or have not been implemented in ASIC [7]–[10].

While recent ASIC designs targeting RNNs focus on improved energy efficiency [11]–[16], they do not incorporate

Manuscript received December 20, 2019; revised March 13, 2020 and April 29, 2020; accepted April 30, 2020. Date of publication May 18, 2020; date of current version June 29, 2020. This article was approved by Associate Editor Sylvain Clerc. This work was supported in part by NSF under Grant 1652866, in part by Samsung, in part by the Office of Naval Research (ONR), and in part by the Center for Brain-inspired Computing (C-BRIC), one of six centers in the Joint University Microelectronics Program (JUMP), an Semiconductor Research Corporation (SRC) program sponsored by the Defense Advanced Research Projects Agency (DARPA). *(Corresponding author: Jae-sun Seo.)*

Deepak Kadetotad was with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85281 USA. He is now with Starkey Hearing Technologies, Eden Prairie, MN 55344 USA.

Shihui Yin, Visar Berisha, Chaitali Chakrabarti, and Jae-sun Seo are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85281 USA (e-mail: jaesun.seo@asu.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2020.2992900

compression techniques and do not report RNN accuracy beyond the simpler TIMIT data set [17], while it is necessary to benchmark larger data sets (e.g., TED-LIUM [18] and LibriSpeech [19]) to evaluate enabling practical ASR tasks on small-form-factor edge devices.

In this article, we present a new hierarchical coarse-grain sparsity (HCGS) scheme that structurally compresses LSTM weights by 16× with minimal error rate degradation. We prototyped HCGS-based LSTM accelerator in 65-nm LP CMOS, which executes two-/three-layer LSTMs for real-time speech recognition [20], [21]. It consumes 1.85-/3.43-/3.42-mW power and achieves 8.93/7.22/7.24 TOPS/W for TIMIT/TED-LIUM/LibriSpeech data sets, respectively. Our main contributions include the following.

1) A novel hierarchical blockwise sparsity scheme was proposed and applied to LSTM RNNs, which shows favorable error rate and memory compression tradeoffs compared with prior works.
2) Beyond simpler TIMIT data set [17], our LSTM accelerator is benchmarked against larger-scale TED-LIUM [18] and LibriSpeech [19] data sets with low error rates, demonstrating practical feasibility.
3) Aided by 16× HCGS compression and with 6-bit weight quantization, all parameters of LSTMs for TIMIT/TED-LIUM/LibriSpeech are stored on-chip in <300-kB SRAM.

The remainder of this article is organized as follows. Section II presents the proposed HCGS algorithm for LSTMs. Section III describes the HCGS-based LSTM accelerator architecture and chip design optimization. In Section IV, the prototype chip measurement results and comparison are presented. This article is concluded in Section V.

II. LSTM and Hierarchical Coarse-Grain Sparsity

A. LSTM-Based Speech Recognition

LSTM RNNs have shown state-of-the-art accuracy for speech recognition tasks [2], [3]. Fig. 1 shows the computation flow for each layer of an LSTM [1], which is a specialized recurrent structure. Each layer of an LSTM consists of neurons, which computes the final output h_t through four intermediate results called gates. In addition to the hidden state h_t used as a transient representation of state at timestep t, LSTM introduces a memory cell c_t, intended for internal long-term storage. The parameters c_t and h_t are computed via input, output, and forget gate functions. The forget gate function f_t directly connects c_t to the memory cell c_{t-1} of the previous timestep via an elementwise multiplication. Large values of the forget gates cause the cell to remember most (if not all) of its previous values. Each gate function has a weight matrix and a bias vector; we use subscripts i, o, and f to denote parameters for the input, output, and forget gate functions, respectively. For example, the parameters for the forget gate function are denoted as W_{xf}, W_{hf}, and b_f.

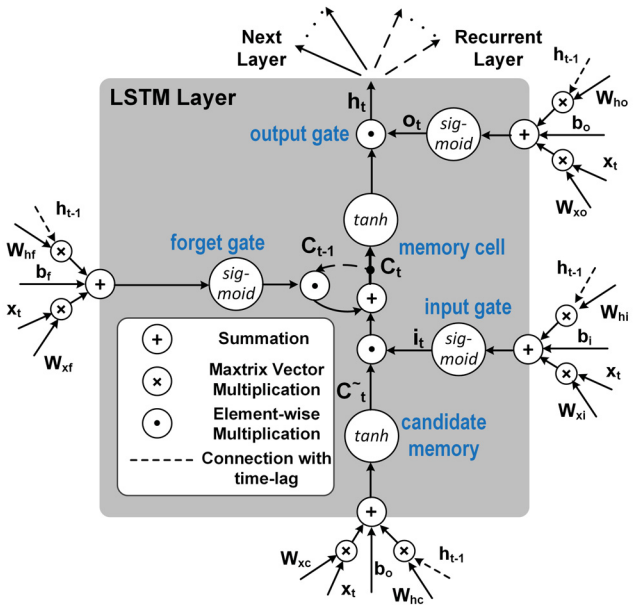


Fig. 1. LSTM computation flow for each layer. Each of the four gates of the LSTM layer receives the input sequence x_t and the recurrent hidden state sequence h_{t-1}, along with the corresponding weights and biases.

With the abovementioned notations, an LSTM is defined as

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (1)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (2)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (3)$$
$$\tilde{c}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (4)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (5)$$
$$h_t = o_t \odot \tanh(c_t) \quad (6)$$

where $\sigma(\cdot)$ represents the sigmoid function and $\odot$ is the elementwise product. From the abovementioned LSTM equations, we see that the weight memory requirement of LSTMs is 8× compared with MLPs with the same number of neurons per layer.

The LSTM-based speech recognition typically consists of a pipeline of a pre-processing or feature extraction module, followed by an LSTM RNN engine and then by a Viterbi decoder [22]. A commonly used feature for pre-processing of speech data is feature-space maximum likelihood linear regression (fMLLR). fMLLR features are extracted from Mel frequency cepstral coefficients (MFCCs) features, obtained conventionally from 25-ms windows of audio samples with a 10-ms overlap between adjacent windows. The features for the current window are combined with those of past and future windows to provide the context of input speech data. In our implementation, we merge five past windows, one current window, and five future windows to generate an input frame with 11 windows, leading to a total of 440 fMLLR features per frame. These merged set of features become inputs to the ensuing LSTM RNN. The output layer of the LSTM consists of probability estimates that are conveyed to the subsequent Viterbi decoder module to determine the best sequence of phonemes/words.

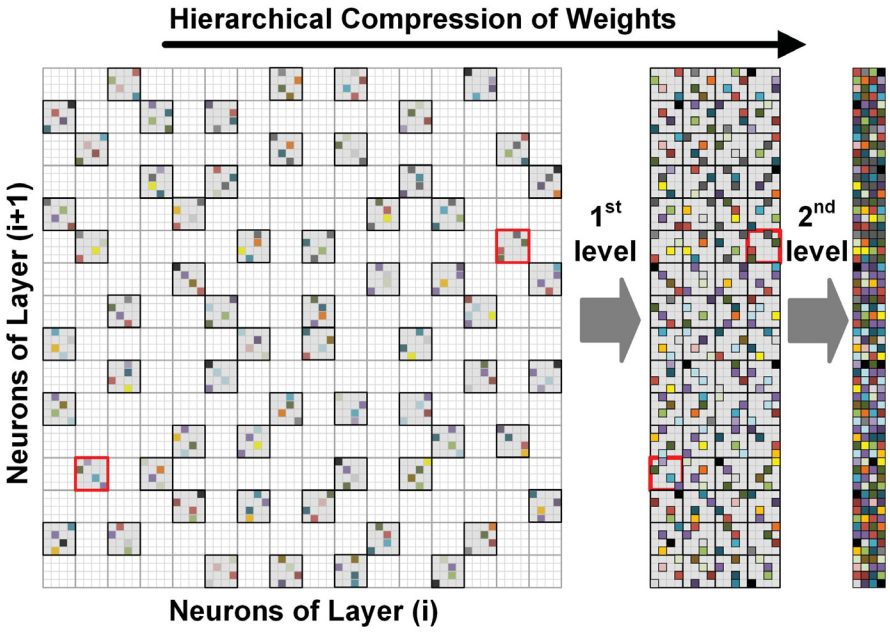


Fig. 2. Illustration of LSTM RNN weight compression featuring the proposed HCGS.

B. Hierarchical Coarse-Grain Sparsity

The proposed HCGS scheme maintains coarse-grain sparsity while further allowing fine-grain weight connectivity, leading to significant energy and area reduction. Fig. 2 illustrates two-level HCGS, where the first level compresses weights (e.g., 4× compression) using a larger block size (e.g., 32×32), and the remaining weights in the large blocks go through the second level of compression (e.g., 4×) with a smaller block size (e.g., 8 × 8). Beyond two levels, the HCGS hierarchy can be expanded to have multiple levels of blockwise sparse structure, recursively selecting even smaller blocks within smaller blocks (e.g., three-level and four-level HCGSs).

The hierarchical structure of blockwise weights is randomly selected before the RNN training process starts, and this pre-defined structured sparsity is maintained throughout the training and inference phases. We apply the constraint that HCGS always selects the same number of random blocks for every block row (see Fig. 2); hence, the selected blocks fit efficiently in SRAMs, enhancing regular memory access and hardware acceleration. The unselected blocks remain at zero and do not contribute to the physical memory footprint during both training and inference. While we focus on the HCGS-based LSTM inference accelerator in this work, due to the pre-defined and static nature of HCGS-based sparsity, training hardware acceleration [23], [24] could also become more efficient with significantly fewer weights and computation involved for the training process of deep neural networks.

We used three well-known benchmarks for speech recognition applications, TIMIT [17], TED-LIUM [18], and Libri-Speech [19], to train our proposed HCGS-based LSTMs and evaluate the corresponding error rates. The baseline three-layer, 512-cell LSTM RNN that performs speech recognition for TED-LIUM/LibriSpeech data sets requires 24 MB of weight memory in floating-point precision. Aided by the proposed HCGS that reduces the number of weights by 16× and low-precision (6-bit) representation of weights, the compressed parameters of a three-layer, 512-cell LSTM RNN is reduced to only 288 kB (83× reduction in model size compared with 24 MB). The resultant LSTM network can be fully stored on-chip, which enables energy-efficient acceleration without costly DRAM access.

C. HCGS-Based Training

We trained LSTM RNNs by minimizing the cross-entropy error, as described in the following equation:

$$E = -\sum_{i=1}^{N} t_i \times \ln y_i \tag{7}$$

where N is the size of the output layer, y_i is the ith output node, and t_i is the ith target value or label. The mini-batch stochastic gradient method [25] is used to train the network. The change in weight for each iteration is the differential of the cost function with respect to the weight value, as follows:

$$\Delta W = \frac{\delta E}{\delta W}. \tag{8}$$

The weight W_{ij} in the $(k+1)$th iteration is updated using the following equation:

$$\left(W_{ij}\right)_{k+1} = \left(W_{ij}\right)_k + \left\{\left(\Delta W_{ij}\right)_k + m \times \left(\Delta W_{ij}\right)_{k-1}\right\} \times \mathrm{lr} \times C_{ij} \tag{9}$$

where m is the momentum, lr is the learning rate, and C_{ij} is the binary connection coefficient between two subsequent neural network layers, which is introduced for the proposed HCGS, and only the weights in the network corresponding to $C_{ij} = 1$ are updated.

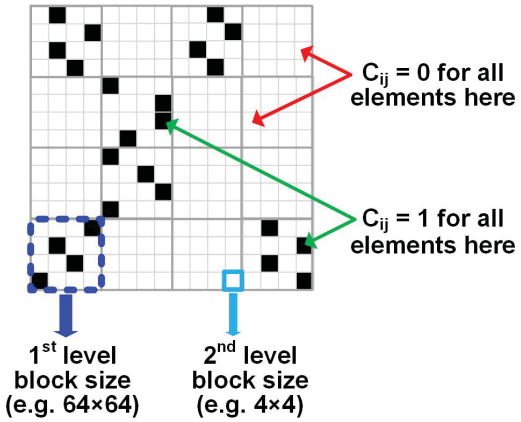


Fig. 3. Binary connection matrix used for HCGS-based neural network training.

Fig. 3 illustrates an example C_{ij} matrix for two levels of HCGS. Since existing neural network training frameworks (e.g., PyTorch and TensorFlow) cannot efficiently support this type of hierarchical blockwise structure, we simply prevent updating the non-selected weights from pre-defined sparsity by setting $C_{ij} = 0$. However, if such training frameworks or ASIC accelerators for neural network training can support this type of blockwise sparsity structure, we anticipate that the training time and energy will decrease substantially.

Proposed LSTM training has been performed using the PyTorch framework, and the code for this work is available at https://github.com/razor1179/pytorch-kaldi-CGS.

D. Design Space Exploration

There are several important design parameters for HCGS-based LSTM hardware design, including activation/weight precision, HCGS compression ratio, the number of CGS levels, and width of LSTM RNN (i.e., the number of LSTM cells in each layer). For this design space exploration, we investigated a number of different LSTM RNNs and the corresponding simulation results are shown in Fig. 4. Starting from the LSTM trained with 32-bit floating-point precision (phoneme error rate or PER = 16.6% for uncompressed 512-cell LSTM), we first reduced the weight precision down to 6 bit to keep all weights on-chip with minor PER loss. With 6-bit weights, we subsequently reduced the activation precision to 13-bit, which overall resulted in small PER degradation of 2.1% (PER = 18.7% for uncompressed fixed-point precision 512-cell LSTM). As illustrated in the precision study shown in Fig. 5, reducing the weight precision below 6 bits (e.g., 3-bit precision) aggravated the error rate degradation for HCGS-based LSTMs compressed by 16×.

Compared with single-level CGS [9], the two-level HCGS scheme shows a favorable tradeoff between weight compression and PER of two-layer LSTM RNN for TIMIT data set (see Fig. 4), aided by capability to balance coarse and fine connection granularity for high levels of compression. A similar trend has been found for the three-layer LSTM RNNs for TED-LIUM and LibriSpeech data sets.

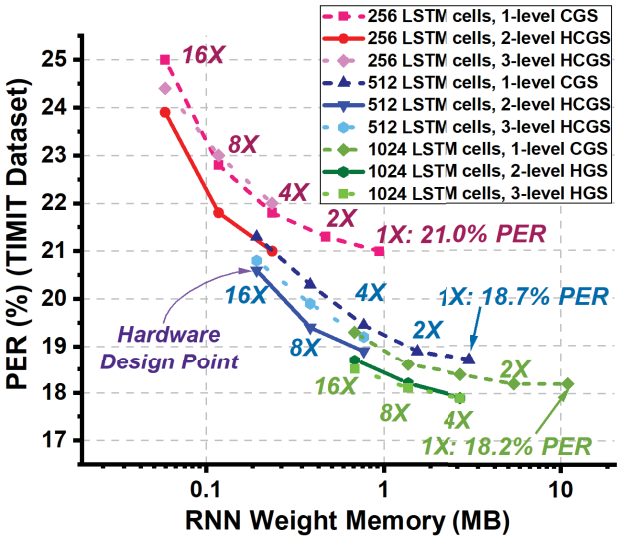


Fig. 4. HCGS design space exploration of two-layer LSTM RNNs across different RNN widths and number of CGS levels. The 6-bit weight precision and 13-bit activation precision are used for all data points.

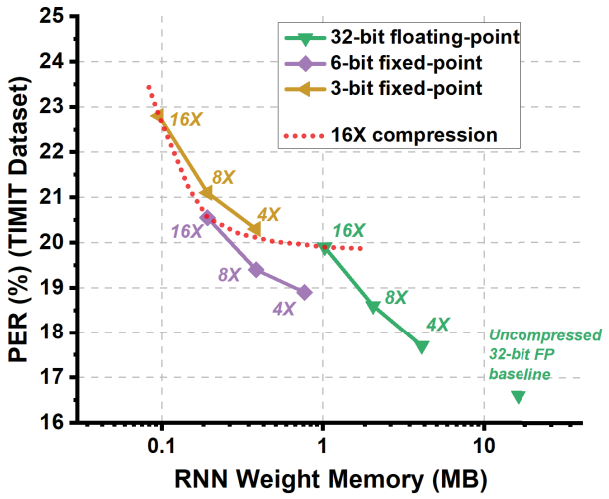


Fig. 5. Weight precision investigation with HCGS-based compression for 512-cell two-layer LSTM RNNs.

We have also experimented with three- and four-level HCGS schemes. As shown in Fig. 4, the three-level HCGS scheme resulted in 0.5%/0.2% worse PER for LSTMs with 256/512 cells and obtained marginal 0.2% PER improvement over two-level HCGS for LSTMs with 1024 cells. Four-level HCGS resulted in even worse PER results compared with three-level HCGS; hence, the four-level HCGS results were not included in Fig. 4. On the hardware side, even if three-level HCGS had shown marginally better error rate than two-level HCGS, given the additional hardware overhead of the selector and logic for additional levels of HCGS, two levels would be the optimal choice of levels for HCGS implementation.

Overall, 512-cell LSTMs show a good balance between error rate (compared with 256-cell LSTMs) and memory (compared with 1024-cell LSTMs) for various HCGS experiments. Based on these results, we selected the 512-cell LSTM

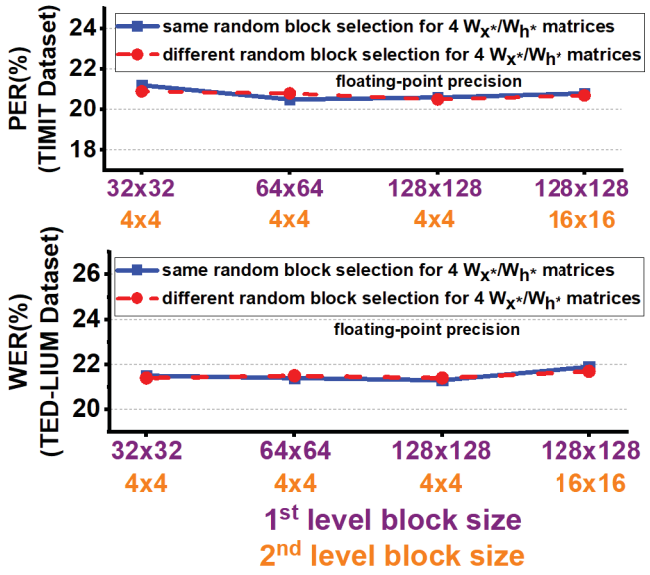


Fig. 6. Robustness of HCGS performance across various block sizes and random block selection.

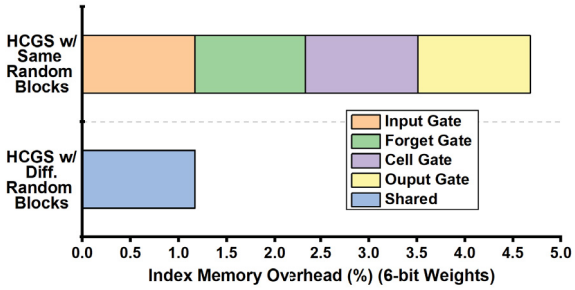


Fig. 7. Further reduction of index memory aided by using the same random block selection for four gates in each LSTM layer.

and two-level HCGS with 16× compression as the hardware design point (see Fig. 4), for speech recognition tasks using TIMIT, TED-LIUM, and LibriSpeech data sets.

E. Robustness Across Random Block Selection and Further Minimization of Index Memory

It is important to note that the block sizes chosen in both levels may affect the final accuracy of the trained network. Therefore, LSTM networks with varying block sizes in both levels must be evaluated to obtain the optimal compression and accuracy. These results are reported in Fig. 6, which shows similar PER and WER values for the cases of using the same and different random block assignments for four LSTM gates. Compared with cases of using different random block selection, sharing the same random block selection for four gates did not affect PER or WER by more than 0.2% across all our LSTM experiments.

Based on this result, to further increase the compression efficiency, we employed the same random block selection for weights associated with the four gates in each LSTM layer. As shown in Fig. 7, sharing the same random block selection results in 4× reduction of the index memory and reduces the computations for decompression by 4× as well. As a result, only 1.17% index memory overhead exists for our HCGS-based LSTM accelerator design. If the weight precision were to be reduced to 3-bit, the index memory overhead would double to 3.34%.

III. Architecture and Design Optimizations

A. Hardware Architecture

Fig. 8 shows the overall architecture of the proposed LSTM accelerator. Our LSTM accelerator consists of the input and output buffers, MAC unit, HCGS selector, H-buffer, C-buffer, two memory banks (144 kB each) for weight storage, bias/index memory bank (8.5 kB), and the global finite state machine (FSM). The proposed architecture facilitates the computation of one LSTM cell output per cycle after an initial latency period and reuses the MAC unit, as outputs are computed in a layer-by-layer manner. The reuse of the MAC unit leads to a compact design, allowing for storing weights, LSTM states, and configuration bits in densely packed memory structures without the need for complicated routing architectures, as shown in [6].

1) HCGS Selector: The HCGS selector (see Fig. 8, top left) has two levels, where the first level of selector only enables the propagation of activations associated with larger non-zero weights blocks and the second level further filters through the activations associated with smaller non-zero blocks. The selection of relevant blocks is done through the implementation of block multiplexers, which allows the propagation of the set of inputs that correspond to non-zero weights stored. With 16× HCGS compression, only 32 activation outputs are required from a total of 512 activations, and only the activations corresponding to non-zero weights propagate to the MAC unit, largely improving energy efficiency.

The selection input for the HCGS selector is a 48-bit vector, where 16 bits correspond to the first level of selection and the remaining 32 bits are used for the second level. The selector supports block sizes ranging from 128×128 to 32×32 for the first level and from 16×16 to 4×4 for the second level. This wide range of block sizes allows for flexibility to map arbitrary LSTM networks trained with HCGS onto our accelerator chip using different configurations.

2) Input and Output Buffers: An input frame consists of fMLLR features, as described in Section II-A. The input buffer is used to store the fMLLR features of an input frame, which streams in 13 bits each cycle over 512 cycles. The input buffer is essential for the continuous computation of the LSTM output as it enables the subsequent input frame to be ready for use as soon as the current frame computation is complete. This buffer ensures that there is no stall required to stream in the consecutive frames of the real-time speech input. The serial-in/parallel-out input buffer takes in 13-bit inputs sequentially and outputs all 6656 bits in parallel.

The output buffer consists of two identical buffers for double buffering, which enables continuous computation of the LSTM accelerator in conjunction with the input buffer while streaming out the final layer outputs. Each output buffer employs an HCGS selector and a 6656:416 multiplexer to feedback the

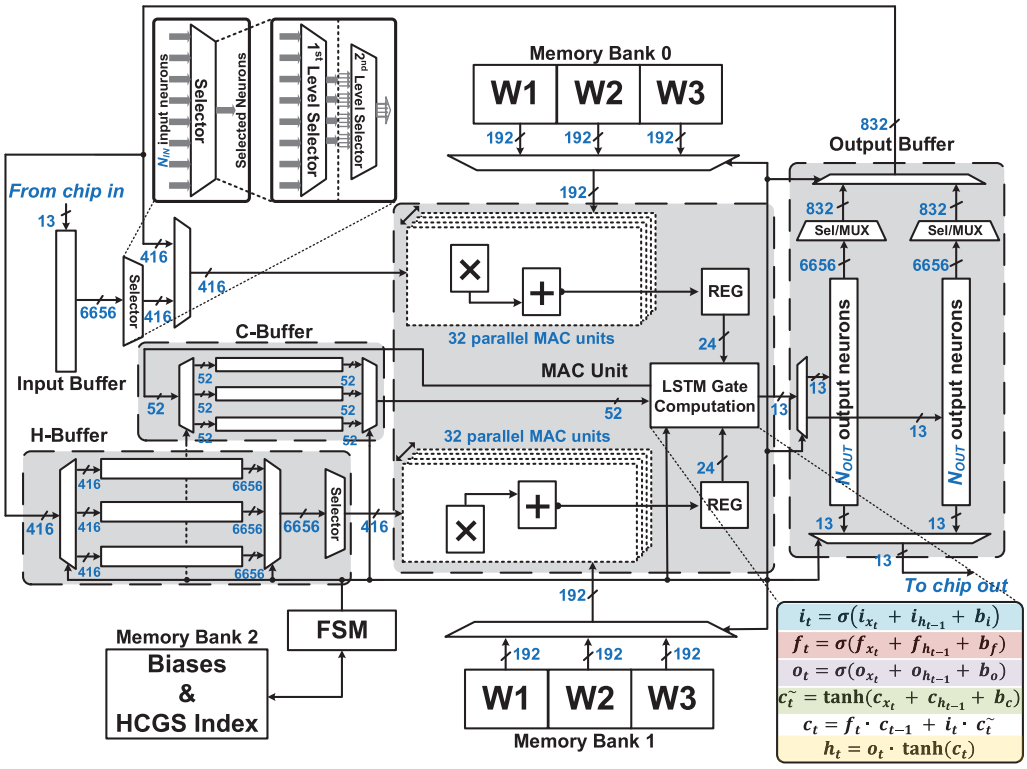


Fig. 8. Overall architecture of the proposed LSTM RNN accelerator.

current layer output to the next layer. The feedback path from the output buffer to the input of the MAC facilitates the reuse of the MAC unit. Each output buffer takes in a 13-bit LSTM cell output, and the correct buffer is chosen by the FSM by keeping track of whether the buffer is full or ready to stream data out of the chip.

Finally, a multiplexer is used to decide whether the x_t input should be from the input or the output buffer, and this is done through the FSM that uses the frame complete flag to switch between the two buffers.

3) H-Buffer and C-Buffer: The H-buffer and C-buffer are rolling buffers and store the outputs of the previous frame (h_{t-1}) and cell state (c_{t-1}) for each LSTM layer, respectively. Each buffer has three internal registers corresponding to the maximum number of layers supported by the hardware. The C-buffer registers behave as shift registers, while the H-buffer registers operate similar to the input buffer where inputs are streamed in serially and outputs are streamed out in parallel.

4) MAC Unit: The MAC unit consists of 64 parallel MACs (computing vector-matrix multiplications) and the LSTM gate computation module (computing intermediate LSTM gate and final output values), which can perform 129 (=64 × 2.1) compressed operations equivalent to 2064 (=129×16) uncompressed operations effectively in each cycle, aided by the proposed HCGS compression by 16×. The non-linear activation functions of sigmoid and hyperbolic tangent (tanh) are implemented with piecewise linear (PWL) modules using 20 linear segments that exhibit maximum relative error [(PWL_output − ideal_output)/ideal_output] of 1.67×10^{-3} and average relative error of 3.30×10^{-4}.

5) Weight/Bias Memory: As described in Section III-B, weights are stored in the interleaved fashion, where each memory sub-bank (W1–W3) stores weights corresponding to a single layer. Since all weights of the two-/three-layer RNNs can be loaded on-chip initially, write operations are not needed for our LSTM accelerator during inference operations. The required read bandwidth of our LSTM accelerator is 192 bits/cycle from memory bank 0 and 192 bits/cycle from memory bank 1 (see Fig. 8). If there are more parallel MAC units, the required read bandwidth will proportionally increase.

The memory sub-banks that store weights of layers not currently being computed are put into "selective precharge" mode, which clamps the wordlines to a low value (0 V) and floats the bitlines for leakage power reduction. Getting into and out of this selective precharge mode each adds a small overhead of one extra cycle. Moreover, due to the nature of the LSTM, each weight in the memory and sub-banks are used only once, which makes the number of transitions between selective precharge mode and normal mode for each sub-bank to be minimal. Overall, adding the selective precharge mode resulted in 19% energy-efficiency improvement at the system-level for the LSTM accelerator.

B. Interleaved Memory Storage

Fig. 9 shows the timing diagram of LSTM computation and the necessary interleaved storage pattern of weights in on-chip SRAMs. The LSTM cell stores the intermediate products to compute the cell state (c_t) and output (h_t). Conventionally, the cell states and outputs of an entire layer are computed only after every intermediate gate output for the corresponding

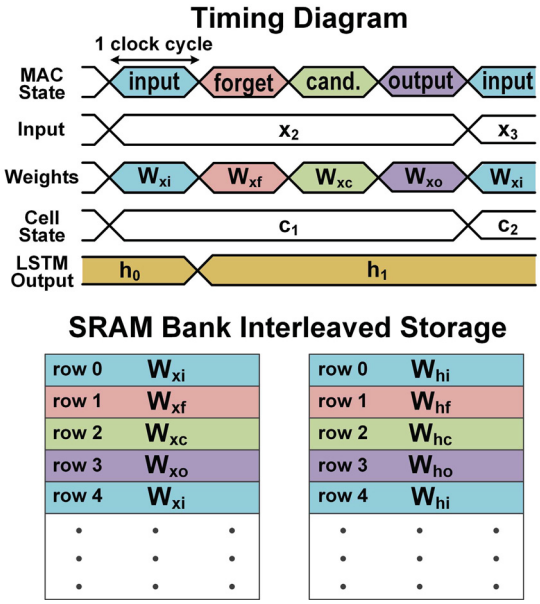
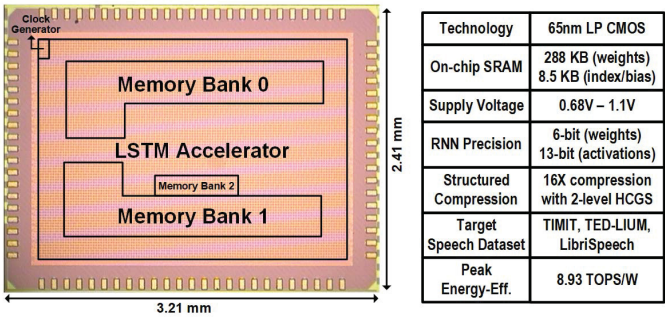


Fig. 9. Timing diagram of LSTM computation (top) and interleaved storage of weights in SRAM (bottom).



Technology	65nm LP CMOS
On-chip SRAM	288 KB (weights) 8.5 KB (index/bias)
Supply Voltage	0.68V – 1.1V
RNN Precision	6-bit (weights) 13-bit (activations)
Structured Compression	16X compression with 2-level HCGS
Target Speech Dataset	TIMIT, TED-LIUM, LibriSpeech
Peak Energy-Eff.	8.93 TOPS/W

Fig. 10. Prototype chip micrograph and performance summary.

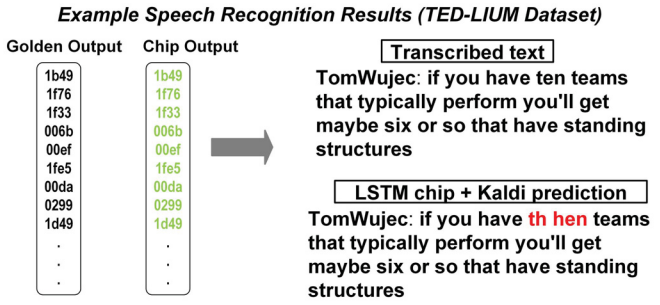


Fig. 11. Example speech recognition results for the TED-LIUM data set.

layer is completed. However, this leads to additional memory requirements to store the intermediate gate outputs for all the LSTM cells in the layer. To alleviate this issue, we take advantage of the structure of the LSTM cell. The proposed architecture cycles between the four states computing internal gates of the LSTM cell, namely, input gate (i_t), forget gate (f_t), output gate (o_t), and candidate memory ($\tilde{c}_t$). In addition, the vector-matrix multiplications of $x_t W_{x*}$ and $h_{t-1} W_{h*}$ can be computed in independent streams, effectively increasing throughput via parallel computing.

To enable this efficiently, we store each row of four matrices W_{xi}, W_{xf}, W_{xo}, and W_{xc} in a staggered manner (same for W_{h*}) in on-chip SRAM arrays (see Fig. 9, right bottom). This way, the computation of new c_t and h_t values can be completed after every four cycles, hence eliminating the requirement to store all intermediate gate outputs of the layer. Also, as described in Section II-E, since we used the same random hierarchical block selection for HCGS for all four matrices of W_{xi}, W_{xf}, W_{xo}, and W_{xc} (same for W_{h*}), the selector logic does not need to change through the interleaving process.

C. End-to-End Operation and Latency

Since all weights of target LSTM networks with HCGS compression are stored on-chip, there is no need to off-chip DRAM communication, and our chip performs the end-to-end operation of the entire LSTM in a pipelined fashion.

The initial delay of 512 cycles is consumed to load the input buffer. Once the input buffer is filled, each LSTM state computation takes three cycles, one for MAC, one for addition, and one for activation, which is all pipelined. The first neuron output takes a total of nine cycles, after which we obtain a new neuron output every cycle. The outputs of the current layer are stored in the output buffer. Once the output buffer is full, if the current layer is an intermediate layer, the output directly is conveyed to the input of the next layer, or if the current layer is the last layer of the LSTM, the output data is streamed out of the chip over 512 cycles.

IV. Measurement Results

The proposed LSTM RNN accelerator is fabricated in 65-nm LP CMOS. Fig. 10 shows the chip micrograph and performance summary. For chip testing, we initially load the weights, biases, and configuration bits to on-chip memory. To verify real-time operation, 13-bit input fMLLR features are streamed into the input buffer every cycle (see Section III-A), while LSTM outputs from the chip are streamed out and stored.

A. Pre-/Post-Processing Operations

The pre-processing steps are performed in Kaldi framework [26] using audio files from TIMIT, TED-LIUM, and LibriSpeech data sets. The same extracted input features were used for LSTM training and real-time inference based on HCGS compression. With LSTM outputs streamed out of the chip, we perform post-processing also using Kaldi framework [26] to obtain the final error rates for speech recognition. When 512 outputs (13-bit each) per frame are received from chip output, the hidden Markov model (HMM) states are calculated using a weighted finite-state transducer (WFST) that performs Viterbi beam search, finally obtaining phoneme error rate (for TIMIT data set) or WER (for TED-LIUM/LibriSpeech data sets). Example speech recognition results and the transcribed text for the TED-LIUM data set are shown in Fig. 11.

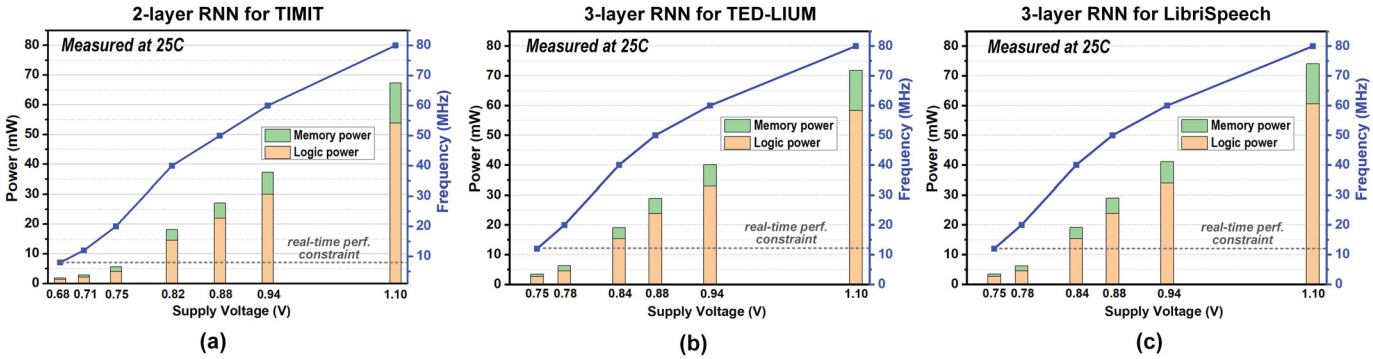


Fig. 12. Power and frequency measurement results with voltage scaling for (a) two-layer LSTM for the TIMIT data set, (b) three-layer LSTM for the TED-LIUM data set, and (c) three-layer LSTM for the LibriSpeech data set.

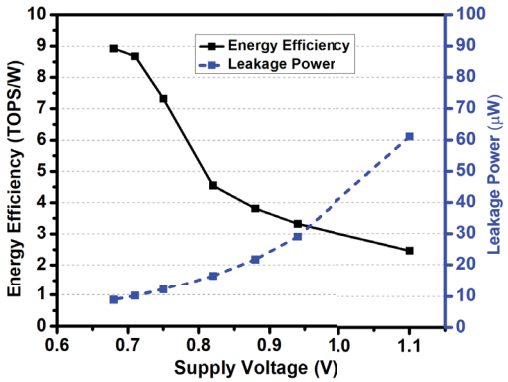


Fig. 13. Measurement results of energy efficiency (TOPS/W) and leakage power of two-layer LSTM for TIMIT data set.

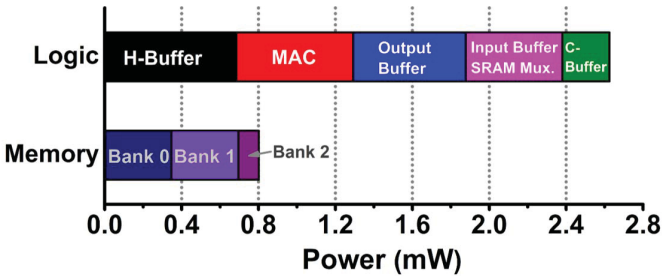


Fig. 14. Power breakdown of three-layer LSTM for TED-LIUM data set at 0.75-V supply. As logic supply and memory supply were separated, the logic power and memory power themselves are directly from chip measurement. The percentage of module-level breakdown within logic/memory power was obtained from post-layout simulation.

While our chip did not integrate pre-/post-processing engines, we can deduce the relative power consumption of them from other prior works. Regarding pre-processing, a recent work [27] proposed serial FFT computation and frame computation re-use for MFCC and reported the MFCC pre-processing power from 28-nm prototype chip as 340 nW at 0.41 V at 40 kHz. Even if we consider CMOS scaling from different technologies, supply voltage, and frequency, still, the MFCC pre-processing power will be a fraction of 1 mW for real-time speech recognition. Regarding post-processing, Price *et al.* [22] implemented both the deep neural network (only supported MLPs) and post-processing engine (Viterbi search) for speech recognition. For large MLPs (e.g., six layers of 512 neurons each), it has been reported that the MLP module consumes $>4\times$ more power than the WFST/Viterbi search module. As aforementioned, an LSTM RNN requires $8\times$ weights compared with an MLP with the same number of neurons per layer. Therefore, we anticipate that both the pre-processing and post-processing engines will consume relatively much smaller power than the LSTM RNN engine, and thus, power/energy reduction of the LSTM RNN would remain as a large benefit for the overall ASR system.

B. Performance, Energy, and Error Rate Measurements

Fig. 12 shows the chip measurement results for two-/three-layer LSTM RNNs for TIMIT/TED-LIUM/LibriSpeech data sets. With voltage scaling, the power consumption at 0.68 V for the two-layer RNN for TIMIT is 1.85 mW at 8 MHz [see Fig. 12(a)] and, at 0.75 V for the three-layer RNNs for TED-LIUM/LibriSpeech, is 3.43/3.42 mW at 12 MHz [see Fig. 12(b) and (c)]. In all cases, the accelerator satisfies the real-time speech recognition requirement of 100 frames/s (10 ms/frame).

With the proposed HCGS scheme, our LSTM accelerator achieves an average energy efficiency of 8.93 TOPS/W for running end-to-end two-layer LSTM RNN for TIMIT data set (see Fig. 13) and 7.22/7.24 TOPS/W for running end-to-end three-layer LSTM RNNs for TED-LIUM/LibriSpeech data sets while meeting the real-time speech recognition requirement. If we relax the real-time performance constraint, higher energy efficiency can be achieved. The total leakage power at 0.68-V supply is less than 10 μW (see Fig. 13).

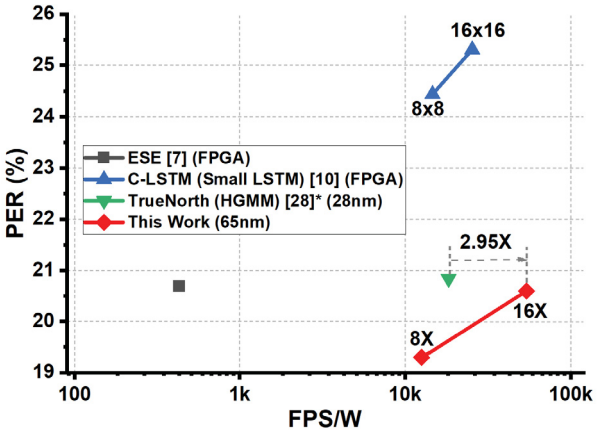
The memory and logic power breakdown for the three-layer RNN at 0.75-V supply is shown in Fig. 14. It can be seen that the logic power is dominant due to the highly compressed weight memory despite a large number of RNN weight matrices. Pipelined with the LSTM gate computation unit, the MAC engines exhibit a very high utilization ratio of 99.66%.

This high MAC efficiency is obtained because each of the layers in the two-/three-layer RNNs that we target has a regular structure to start with, and also HCGS compression still maintains a regular structure because we have the same number of

TABLE I
COMPARISON OF RNN PERFORMANCE WITH PRIOR WORKS

	[7]	[10]	[12]	[13]	[11]	This Work
Technology	FPGA	FPGA	65nm CMOS	65nm CMOS	65nm CMOS	**65nm CMOS**
Area (mm^2)	-	-	1.57	19.36	7.5	**7.74**
On-Chip Memory (KB)	4.2 MB	280	82	348	100	**297**
Number of MACs	-	-	96	-	256	**65**
Bit-Precision Weights / Activations	12/16	16/16	8/16	16/16	4/4 (FFT: 8-bit)	**6/13**
Core Voltage (V)	-	-	1.24/0.75	1.2/0.67	1.15/0.54	**1.1/0.68**
Frequency (MHz) [1]	200	200	168/20	200/10	200/25	**80/8**
Power (mW) [1]	41W	22W	29/1.2	447/4	339.2/13.3	**67.3/1.85**
Peak Performance (GOPS) [1]	2500	-	-	-	14.9	**164.95/24.60**
Energy-Efficiency (TOPS/W) [1]	0.061	2.08	1.11/3.08	1.06/5.09	14.4 q	**2.45/8.93**
PER (TIMIT)	20.7%	25.3%	-	-	worse by 1.93% compared to baseline	**20.6% (measured)**
WER (TED-LIUM)	-	-	-	-		**21.3% (measured)**
WER (LibriSpeech)	-	-	-	-		**11.4% (measured)**

[1] Performance, power, and energy-efficiency values all correspond for RNNs for TIMIT dataset.



* [28] classification error from HGMM.

Fig. 15. Comparison of PER for TIMIT data set and energy efficiency (FPS/W) with prior LSTM implementations.

blocks pruned/kept in the same block row (see Fig. 2). As long as the number of compressed LSTM operations for each RNN layer is an integer multiple of 64, all 64 MAC units in our chip will continuously perform operations without idle periods. In addition, the input and output buffers (see Fig. 8) are designed specifically to enable continuous operations without idle periods between layers.

We achieve measured accuracy results of 20.6% PER for TIMIT, 21.3% WER for TED-LIUM, and 11.4% WER for LibriSpeech data sets.

C. Comparison to Prior LSTM/RNN Works

Since most LSTM or RNN ASIC works only reported error rate results with the simpler TIMIT database, we compared the TIMIT PER and frames/second/power (FPS/W) in Fig. 15 of the proposed HCGS and prior works [7], [10], [28] that perform speech/phoneme recognition.

The RNN accelerator [29] reports low power consumption but can only support limited keyword spotting tasks and is not considered. Compared with 28-nm ASIC design supporting speech recognition [28], this work shows 2.95× higher energy efficiency (FPS/W) with slightly better PER. Although FPS/W in [10] is comparable to our work, we achieve considerably lower PER. Conversely, [7] has comparable PER to our work but poor FPS/W. The recent ASIC work based on block-circulant matrices [11] has neither reported the absolute PER for TIMIT nor the results necessary to calculate corresponding FPS/W. Overall, this demonstrates the effectiveness of our proposed design due to the algorithm-hardware co-optimization.

Table I shows a detailed comparison with prior ASIC and FPGA hardware designs for RNNs. Compared with the RNN ASIC works of [12] and [13], this work shows 2.90× and 1.75× higher energy efficiency (TOPS/W), respectively. Reference [11] presented higher TOPS/W than our work, but the end-to-end latency or FPS has not been reported. Moreover, only a simpler TIMIT data set has been benchmarked (while we also benchmarked against more complex TED-LIUM and LibriSpeech data sets), and the absolute TIMIT PER has been not shown.

V. CONCLUSION

This article presents an energy-efficient LSTM accelerator featuring hierarchical/recursive blockwise sparsity for ASR. HCGS enables large compression (16×) of LSTM weights with graceful error rate degradation while minimizing the

index memory cost (∼1%). Exploiting the hierarchical block-wise sparsity and low-precision quantization, our LSTM accelerator stores the entire compressed weights of three-layer, 512-cell LSTMs in 288 kB of on-chip SRAM and reduces the required computation by up to 16×. Experiments conducted on TIMIT, TED-LIUM, and LibriSpeech data sets demonstrated the effectiveness and applicability of HCGS across various LSTM RNNs.

The prototype chip fabricated in 65-nm LP CMOS achieves average energy efficiency of 8.93 TOPS/W for two-layer LSTM for TIMIT data set and 7.22/7.24 TOPS/W for three-layer LSTM for TED-LIUM/LibriSpeech data sets while meeting the real-time speech recognition performance constraint. Algorithm/hardware techniques demonstrated by HCGS and our prototype chip can help enable on-device ASR on edge devices with severe area and energy constraints.

REFERENCES

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The microsoft 2017 conversational speech recognition system," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 5934–5938.

[3] K. J. Han, A. Chandrashekaran, J. Kim, and I. Lane, "The CAPIO 2017 conversational speech recognition system," 2017, *arXiv:1801.00059*. [Online]. Available: http://arxiv.org/abs/1801.00059

[4] M. Halpern, Y. Zhu, and V. J. Reddi, "Mobile CPU's rise to power: Quantifying the impact of generational mobile CPU design trends on performance, energy, and user satisfaction," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Mar. 2016, pp. 64–76.

[5] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[6] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

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

[8] W. Wen *et al.*, "Learning intrinsic sparse structures within long short-term memory," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–14.

[9] D. Kadetotad, S. Arunachalam, C. Chakrabarti, and J.-S. Seo, "Efficient memory compression in deep neural networks using coarse-grain sparsification for speech applications," in *Proc. 35th Int. Conf. Comput.-Aided Design*, Nov. 2016, pp. 1–8.

[10] S. Wang *et al.*, "C-LSTM: Enabling efficient LSTM using structured compression techniques on FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2018, pp. 11–20.

[11] J. Yue *et al.*, "A 65 nm 0.39-to-140.3 TOPS/W 1-to-12b unified neural network processor using block-circulant-enabled transpose-domain acceleration with 8.1× higher TOPS/mm^2 and 6T HBST-TRAM-based 2D data-reuse architecture," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2019, pp. 138–140.

[12] F. Conti, L. Cavigelli, G. Paulin, I. Susmelj, and L. Benini, "Chipmunk: A systolically scalable 0.9 mm^2, 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.

[13] S. Yin *et al.*, "A 1.06-to-5.09 TOPS/W reconfigurable hybrid-neural-network processor for deep learning applications," in *Proc. Symp. VLSI Circuits*, Jun. 2017, pp. 26–27.

[14] C. Gao, D. Neil, E. Ceolini, S.-C. Liu, and T. Delbruck, "DeltaRNN: A power-efficient recurrent neural network accelerator," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2018, pp. 21–30.

[15] C. Gao, S. Braun, I. Kiselev, J. Anumula, T. Delbruck, and S.-C. Liu, "Real-time speech recognition for IoT purpose using a delta recurrent neural network accelerator," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[16] S. Dey and P. D. Franzon, "An application specific processor architecture with 3D integration for recurrent neural networks," in *Proc. 20th Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2019, pp. 183–190.

[17] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgrenm "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM," NISTIR, Gaithersburg, MD, USA, Tech. Rep. 4930, Feb. 1993.

[18] A. Rousseau, P. Deléglise, and Y. Esteve, "TED-LIUM: An automatic speech recognition dedicated corpus," in *Proc. Int. Conf. Lang. Resources Eval. (LREC)*, 2012, pp. 125–129.

[19] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 5206–5210.

[20] D. Kadetotad, V. Berisha, C. Chakrabarti, and J. Seo, "A 8.93-TOPS/W LSTM recurrent neural network accelerator featuring hierarchical coarse-grain sparsity with all parameters stored on-chip," in *Proc. IEEE Eur. Solid-State Circuits Conf. (ESSCIRC)*, Sep. 2019, pp. 119–122.

[21] D. Kadetotad, V. Berisha, C. Chakrabarti, and J.-S. Seo, "A 8.93-TOPS/W LSTM recurrent neural network accelerator featuring hierarchical coarse-grain sparsity with all parameters stored on-chip," *IEEE Solid-State Circuits Lett.*, vol. 2, no. 9, pp. 119–122, Sep. 2019.

[22] M. Price, J. Glass, and A. P. Chandrakasan, "A low-power speech recognizer and voice activity detector using deep neural networks," *IEEE J. Solid-State Circuits*, vol. 53, no. 1, pp. 66–75, Jan. 2018.

[23] S. Yin and J.-S. Seo, "A 2.6 TOPS/W 16-bit fixed-point convolutional neural network learning processor in 65-nm CMOS," *IEEE Solid-State Circuits Lett.*, vol. 3, no. 1, pp. 13–16, Jan. 2020.

[24] S. K. Venkataramanaiah *et al.*, "Automatic compiler based FPGA accelerator for CNN training," in *Proc. 29th Int. Conf. Field Program. Log. Appl. (FPL)*, Sep. 2019, pp. 166–172.

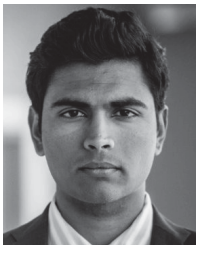
[25] W. A. Gardner, "Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique," *Signal Process.*, vol. 6, no. 2, pp. 113–133, Apr. 1984.

[26] M. Ravanelli, T. Parcollet, and Y. Bengio, "The pytorch-kaldi speech recognition toolkit," in *Proc. ICASSP-IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 6465–6469.

[27] W. Shan *et al.*, "A 510 nW 0.41 V low-memory low-computation keyword-spotting chip using serial FFT-based MFCC and binarized depthwise separable convolutional neural network in 28 nm CMOS," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2020, pp. 230–232.

[28] S. K. Esser *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci. USA*, vol. 113, no. 41, pp. 11441–11446, Oct. 2016.

[29] J. S. P. Giraldo and M. Verhelst, "Laika: A 5 uW programmable LSTM accelerator for always-on keyword spotting in 65 nm CMOS," in *Proc. IEEE Eur. Solid-State Circuits Conf. (ESSCIRC)*, Sep. 2018, pp. 166–169.



Deepak Kadetotad (Member, IEEE) received the B.E. degree in electronics and communication engineering from the M. S. Ramaiah Institute of Technology, Bengaluru, India, in 2013, and the Ph.D. degree in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2019.

In 2019, he joined Starkey Hearing Technologies, Eden Prairie, MN, USA, where he is working on speech enhancement using machine learning. His current research interests include the application of machine learning and neuromorphic algorithms on energy-constrained hardware.

Dr. Kadetotad was a recipient of the LSI Chairman's International Scholarship from 2009 to 2013.

Shihui Yin (Student Member, IEEE) received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2013, and the M.S. degree in electrical engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2015.

He is currently pursuing the Ph.D. degree in electrical engineering with Arizona State University, Tempe, AZ, USA. His research interests include low-power biomedical circuit and system design, and energy-efficient hardware design for machine learning and neuromorphic computing.

Mr. Yin was a recipient of the University Graduate Fellowship from Arizona State University in 2015 and the IEEE Phoenix Section Student Scholarship for the year 2016.

Visar Berisha (Member, IEEE) received the Ph.D. degree from Arizona State University (ASU), Tempe, AZ, USA, in 2007.

From 2007 to 2009, he was a Member of the Technical Staff with the Lincoln Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA. He is currently an Associate Professor with the School of Electrical Computer and Energy Engineering, College of Health Solutions and Fulton Entrepreneurial Professor, ASU. His research interests include computational models of speech production and perception, clinical speech analytics, and statistical signal processing.

Chaitali Chakrabarti (Fellow, IEEE) received the B.Tech. degree in electronics and electrical communication engineering from IIT Kharagpur, Kharagpur, India, in 1984, and the Ph.D. degree in electrical engineering from the University of Maryland, College Park, MD, USA, in 1990.

She is currently a Professor with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA. Her research interests include VLSI algorithm-architecture co-design of signal processing and communication systems and all aspects of low-power embedded systems' design.

Jae-sun Seo (Senior Member, IEEE) received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2001, and the M.S. and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2006 and 2010, respectively.

From 2010 to 2013, he was with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, where he worked on cognitive computing chips under the DARPA SyNAPSE Project and energy-efficient integrated circuits for high-performance processors. In 2014, he joined the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA, as an Assistant Professor. In 2015, he was with the Intel Circuits Research Lab, Hillsboro, OR, USA, as a Visiting Faculty. His current research interests include efficient hardware design of machine learning and neuromorphic algorithms and integrated power management.

Dr. Seo was a recipient of the Samsung Scholarship from 2004 to 2009, the IBM Outstanding Technical Achievement Award in 2012, and the NSF CAREER Award in 2017.