E-LSTM: Efficient Inference of Sparse LSTM on Embedded Heterogeneous System

Runbin Shi, Junjie Liu, Shuo Wang, Yun Liang and Hayden So
The University of Hong Kong, Peking University
{rbshi, jjliu, hso}@eee.hku.hk, {shvowang, ericlyun}@pku.edu.cn

Problem

Various models with Long Short-Term Memory (LSTM) network have demonstrated outstanding performance in sequential information processing. Previous LSTM-specific architectures set large on-chip memory for weight storage to alleviate the memory-bound issue and facilitate the LSTM inference in cloud computing. For embedded scenario, however, the resource constraints should be considered while deploying LSTM on CPU-Accelerator heterogeneous system. As follows,

- The weight values should be stored off-chip, even if the compressed sparse model is larger than 1 MB.
- The limited interface bandwidth between CPU and accelerator (64 bit/cycle in E-LSTM) requires a high-efficient data representation of sparse weight matrix and high reuse rate of fetched weight.
- The parallelism obtained from *batch-processing* is not suitable as it introduces long latency to the system.

Background

LSTM Algorithm

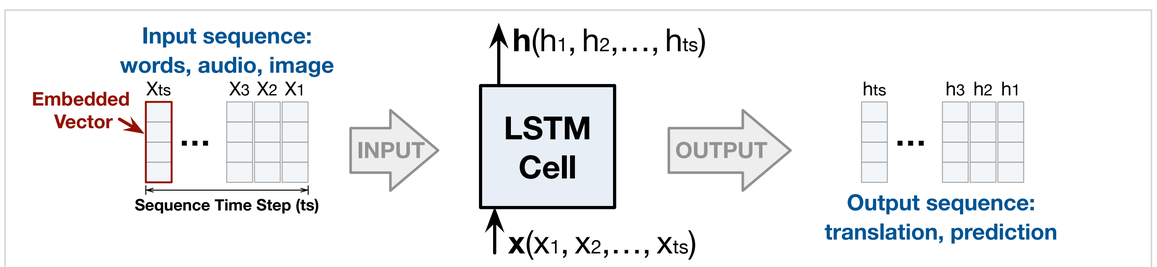


Figure 1: The LSTM-cell accepts the temporal sequence **x** consisting of embedded vectors (x_t) and outputs sequence **h** that represents classification / translation / prediction information.

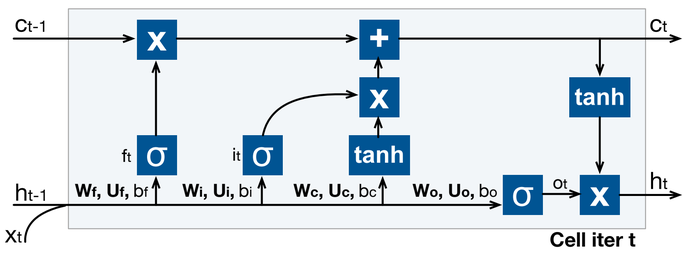


Figure 2: The main computational workload in LSTM cell-iteration is matrix-vector multiplication ($\mathbf{W}x_t$, $\mathbf{U}h_t$). Strong dependency exists between successive cell iterations.

RISC-V based Heterogeneous System

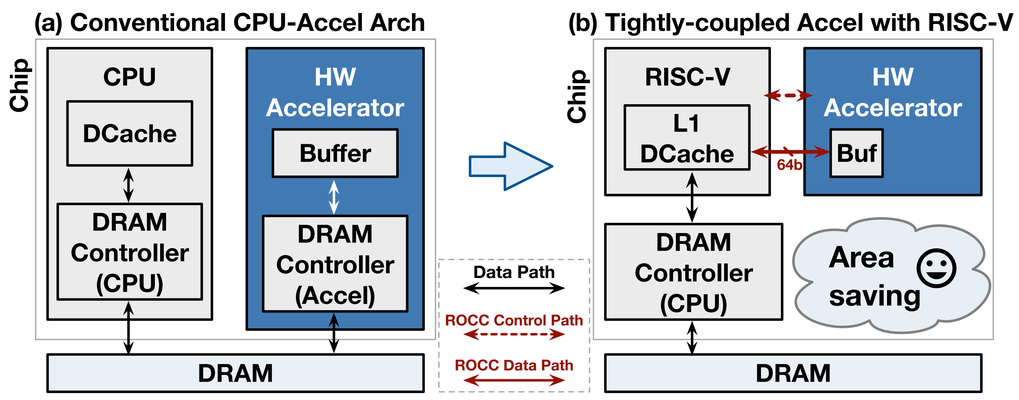


Figure 3: RISC-V provides ROCC interface for accelerator that facilitates lower latency and smaller chip-area cost.

Implementation

E-LSTM on Github

Implementation methods

- Sparse LSTM benchmark: PyTorch
- LSTM accelerator coupled with RISC-V: cpp model in Spike (RISC-V simulator)

eSELL: Sparse Format for Smaller SRAM Area

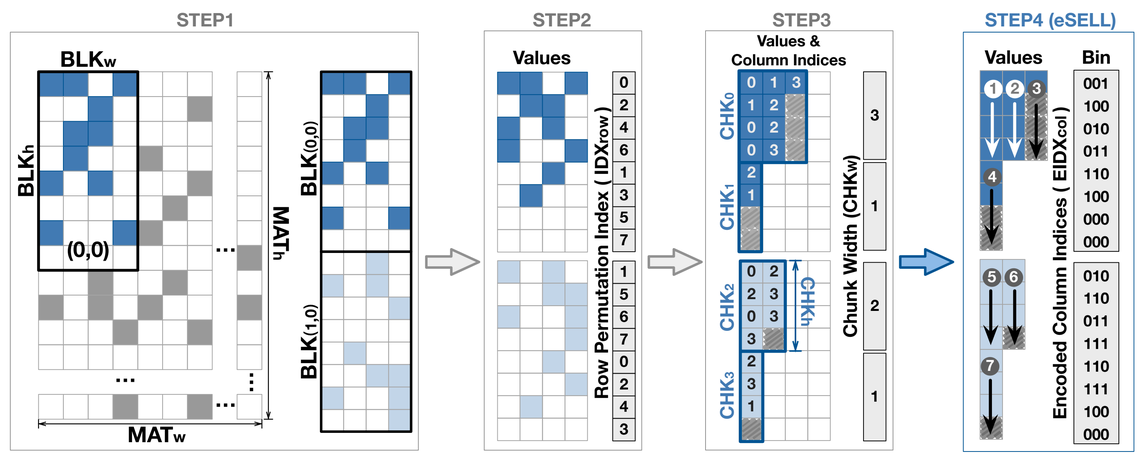
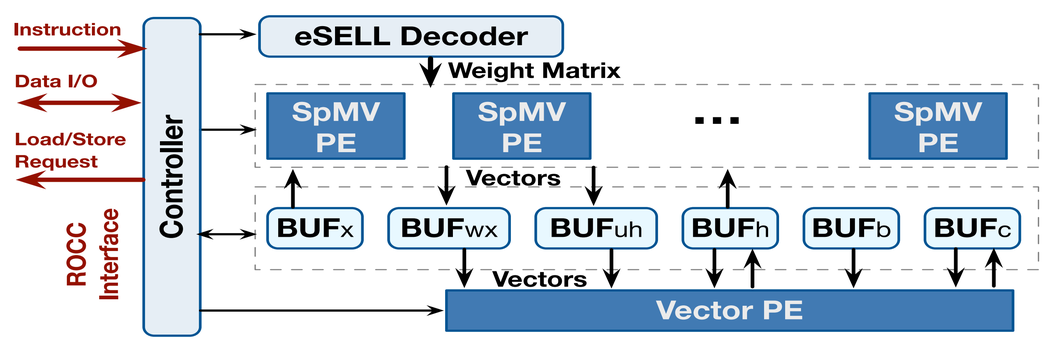


Figure 4: In eSELL representation, the non-zero matrix elements of each row in a block are coalesced that contributes to a fewer port number of on-chip result buffer. Besides, the encoding of permutation reduces the storage overhead.

Accelerator Architecture in E-LSTM



Cell-fusion based on Inherent h_t Sparsity

Scheduling with Dense h_t

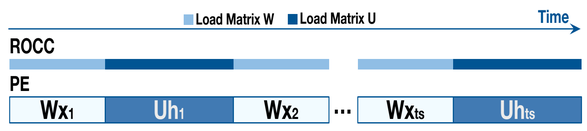


Figure 5: Both the PE and ROCC interface are fully utilized in 1-PE case.

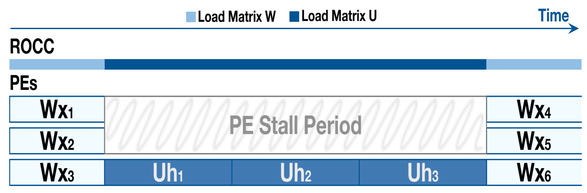


Figure 6: In 3-PE case, 2 PEs are stalled because ROCC is used by loading **U**. Due to the dependency between cell-iterations, $\mathbf{U}h_t$ can only be computed in sequence. Thus, it becomes the bottleneck in both computation and weight fetching communication.

Computation with Sparse h_t

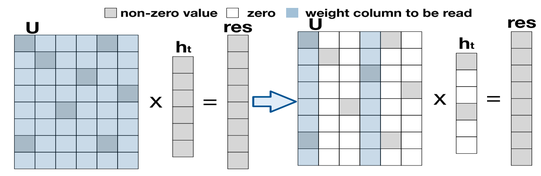


Figure 7: With the sparse h_t, computation workload can be decreased via skipping the **U** columns corresponding to zeros in h_t.

Cell-fusion with Sparse h_t

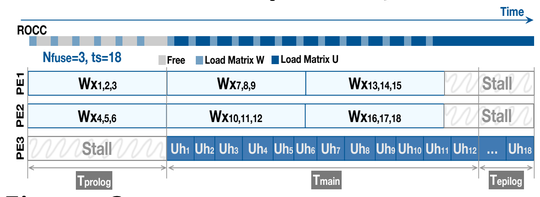


Figure 8: Fine-tuned cell-fusion scheduling effectively utilizes both the PE and ROCC.

Experiments and Results Comparison

Benchmark Layers

Name	Layer	len(x)	len(h)	ts	Sp_w	Sp_u	Sp_h	Score
OCR	LSTM1	28	128	28	0.3	0.5	0.22	98.68/98.61/98.11
(MNIST)	LSTM2	128	128	28	0.2	0.4	0.29	the higher, the better
LM	LSTM1	800	800	35	0.2	0.5	0.56	81.33/81.67/88.52
(PTB)	LSTM2	800	800	35	0.2	0.6	0.41	the lower, the better
LM	LSTM1	1500	1500	35	0.4	0.5	0.37	101.63/102.15/106.5
(Wikitext)	LSTM2	1500	1500	35	0.3	0.4	0.39	the lower, the better

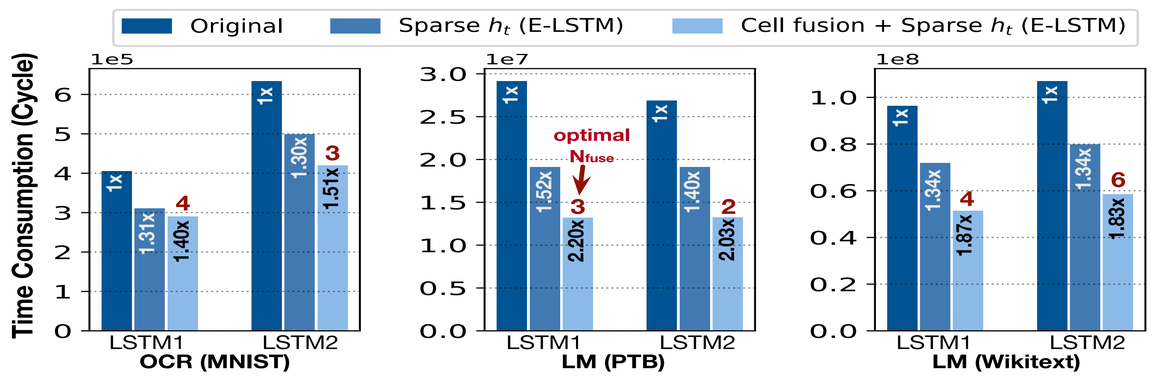


Figure 9: Performance with three scheduling schemes.