Hardware/Algorithm Co-design of Natural Language Processing Tasks

Literature Search & Review Presentation

Yihua Liu

UM-SJTU Joint Institute

December 9, 2021

Motivation

Why Hardware Acceleration for Neural Networks?

- Neural network are getting increasingly popular for a wide range of applications
- More embedded devices are deployed for neural network computation due to limitations of data transfer
- Resources are wasted when computation is done by traditional architecture

Challenges

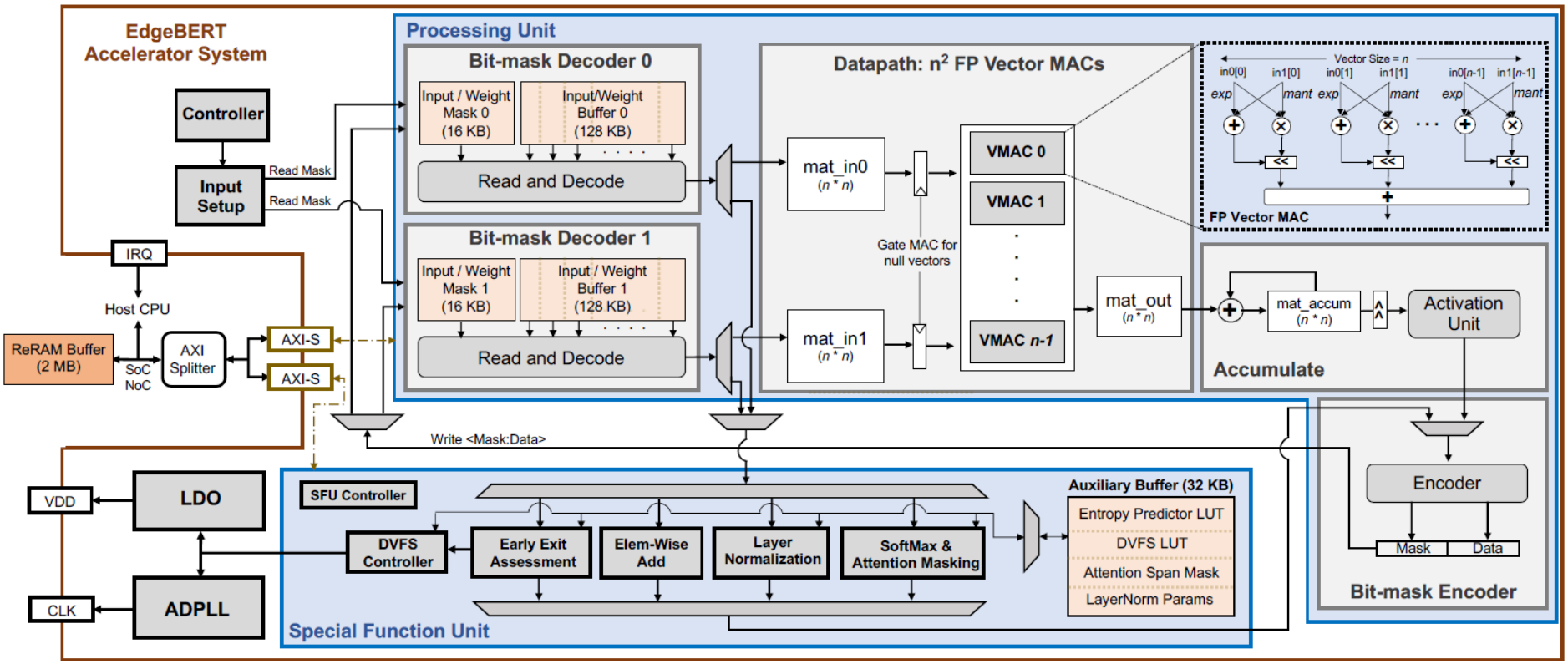
- Computation resources of embedded devices are strictly limited
- Computation load of neural network are getting heavier because of larger datasets and more complex structure
- Improve energy efficiency while accuracy not decreased
- Need to be flexible for different applications and scalable for different computation load

Novelty

The first complete general hardware/algorithm co-design for state-of-the-art natural language processing tasks

- Specialized datapath support for early-exit assessment, softmax and attention span masking, and layer normalization
- Non-volatile and high density storage of the shared multi-task parameters
- On-demand dynamic voltage/frequency scaling aided by integration of fast-locking all-digital phase-locked loop and low-dropout regulator
- Compressed sparse matrix execution via bitmask encoding [1]

Main Approach



Evaluation

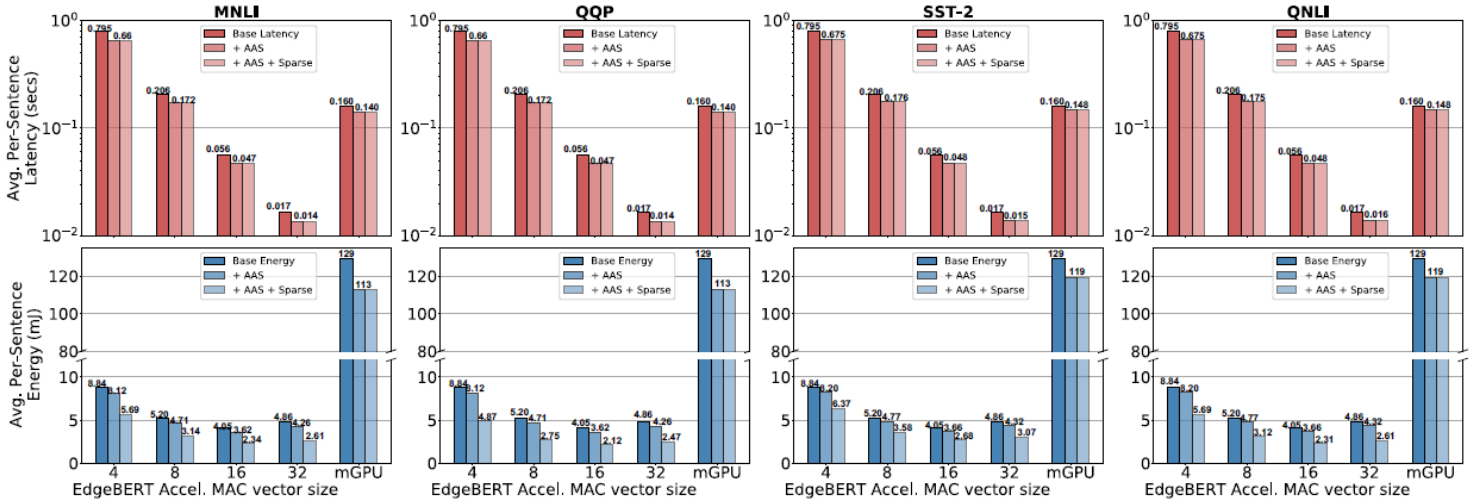


Figure 8: Average latency (top row) and energy (Bottom row) per sentence as the PU MAC vector size scales at max frequency (1GHz) and nominal voltage (0.8V), highlighting impact of adaptive attention span (AAS), and sparsity in weights and activations (Sparse) on the EdgeBERT accelerator and TX2 mGPU. MAC size of 16 yields the most energy efficient design.

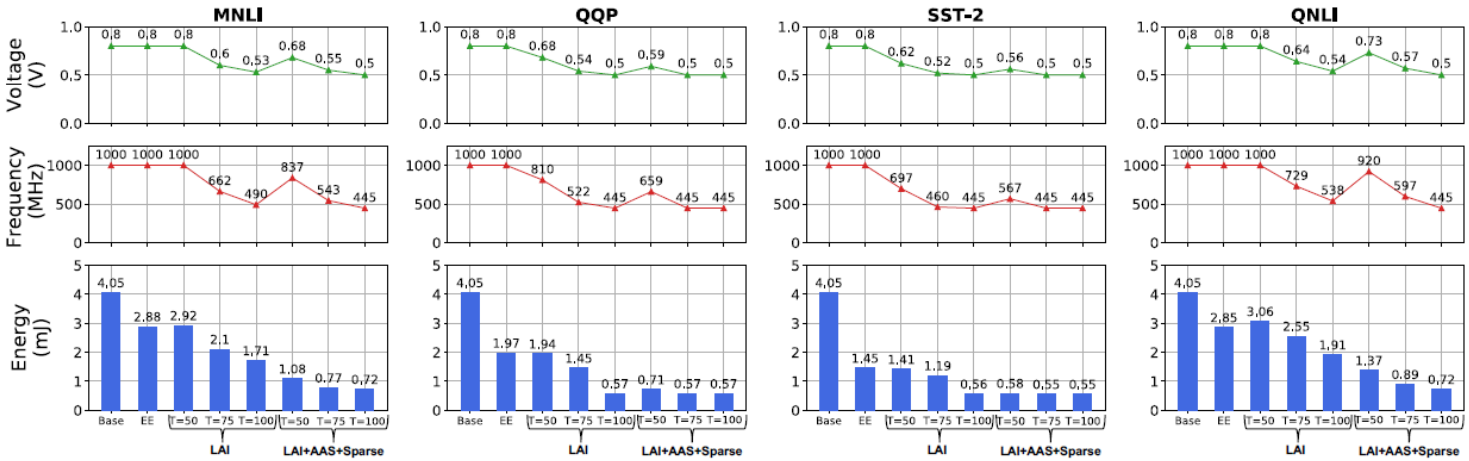
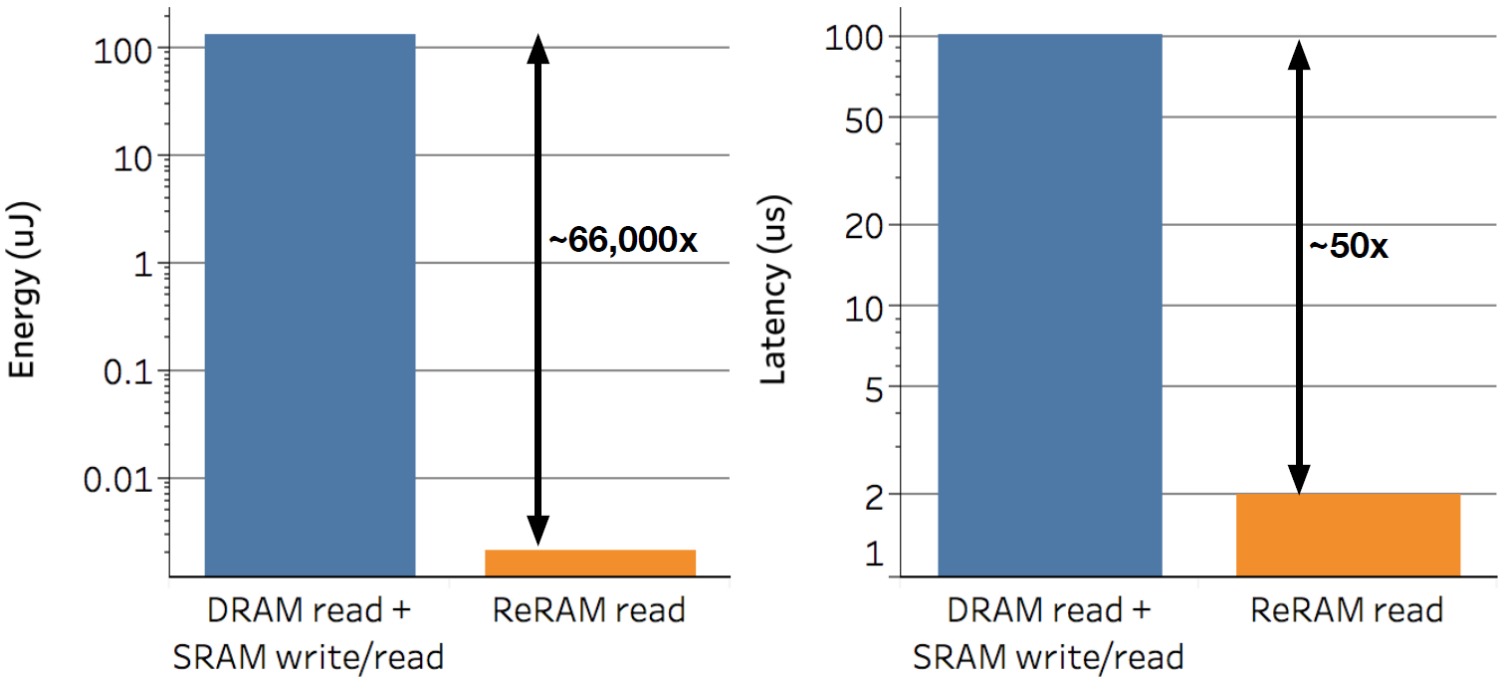


Figure 9: Average DVFS-driven supply voltage (top row) and clock frequency (middle row), as well as, generated energy expenditures (bottom row) of the EdgeBERT accelerator system with $n = 16$ during latency-aware inference (LAI), and latency-aware inference further improved with adaptive attention span and sparse execution (LAS+AAS+Sparse). Different latency targets of 50ms (T=50), 75ms (T=75), and 100ms (T=100) are used for LAI executions. Results are compared with the baseline 12-layer inference (Base) and the conventional early exit inference (EE).

Evaluation



- Energy savings (per-inference): 7× compared to conventional inference and 2.5× compared to early-exit inference.
- Latency advantage 50 × greater than conventional operation and energy advantage 66,000 × greater than conventional operation.

Limitations

- Only support GLUE (General Language Understanding Evaluation) benchmark, not supporting Hugging Face datasets or SQuAD (Stanford Question Answering Dataset).
- Only test on Nvidia Jetson Tegra X2 board, not tested on latest Jetson Xavier NX board.
- It once claimed in previous versions that its solution leads to only 1%-pt drop in accuracy [1], but it lacked data support, and it is removed from the latest version, so its accuracy drop is questionable.
- It admits that its eNVM modeling methodology is traditional ReRAM (Resistive RAM) arrays rather than emerging NVM technologies like PCM (Phase-Change Memory).

Reference

[1] Thierry Tambe et al. *EdgeBERT: Sentence-Level Energy Optimizations for Latency-Aware Multi-Task NLP Inference*. 2021. arXiv: 2011.14203 [cs.AR].

Thanks!