The Era of LLMs on ASICs

Locally Intelligent Machines

Outline

- Motivation
- Hardware Bottlenecks
- Directions and Opportunities for Collaboration

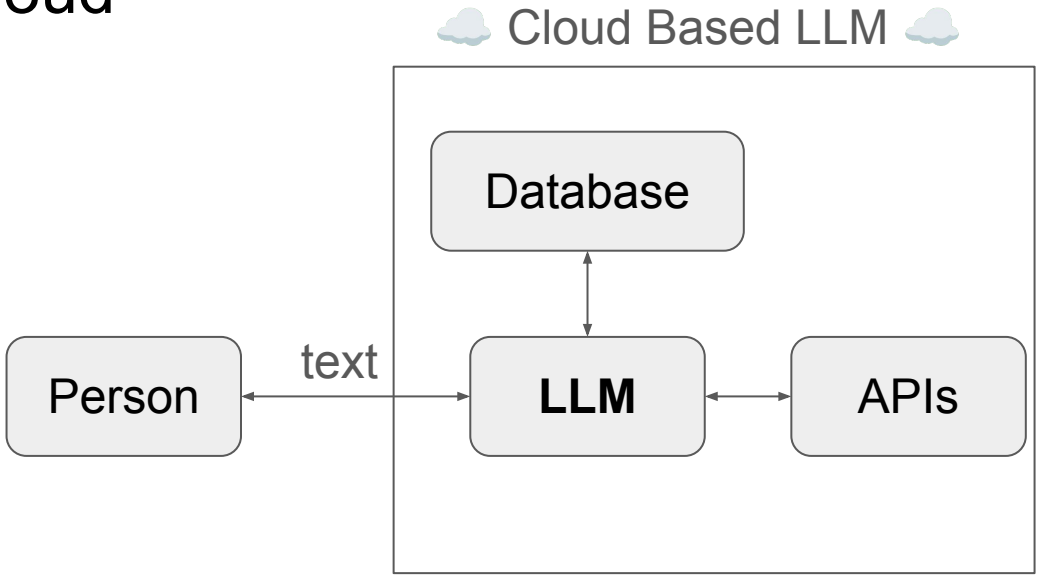
LLMs + Hardware = AI Robots

- Memory
- Multimodality
- Speech Capability

LLMs Currently in the Cloud

- AI Currently Resides in Remote Servers

Image of Servers

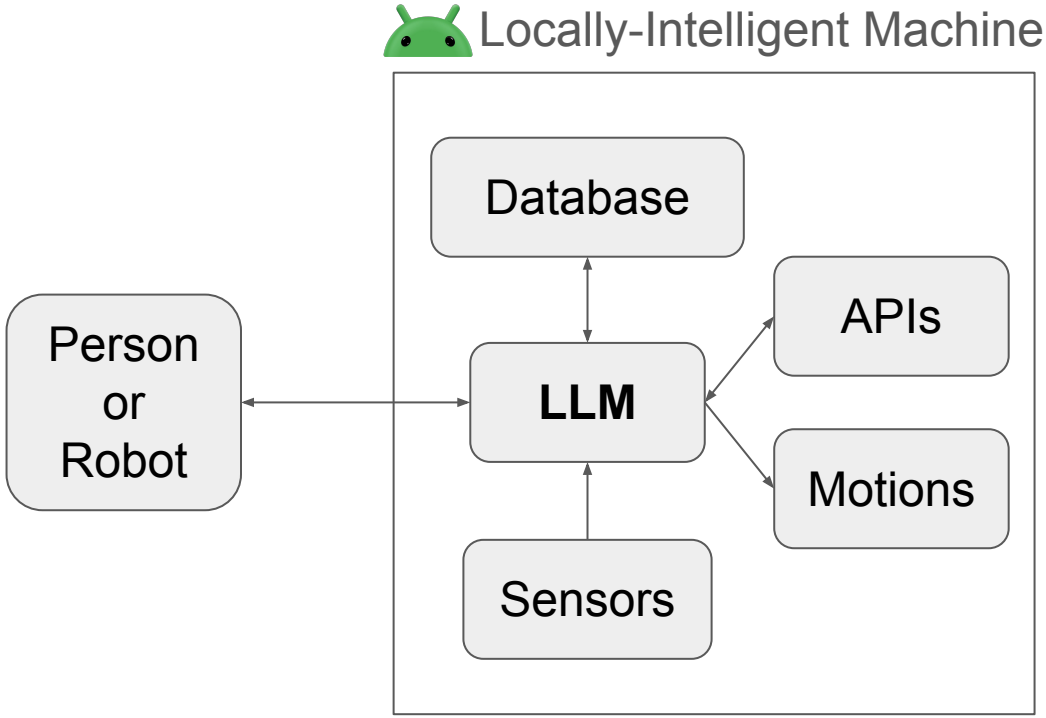


Simplified Modern LLM Diagram

Local Intelligence Enables *Future* Applications

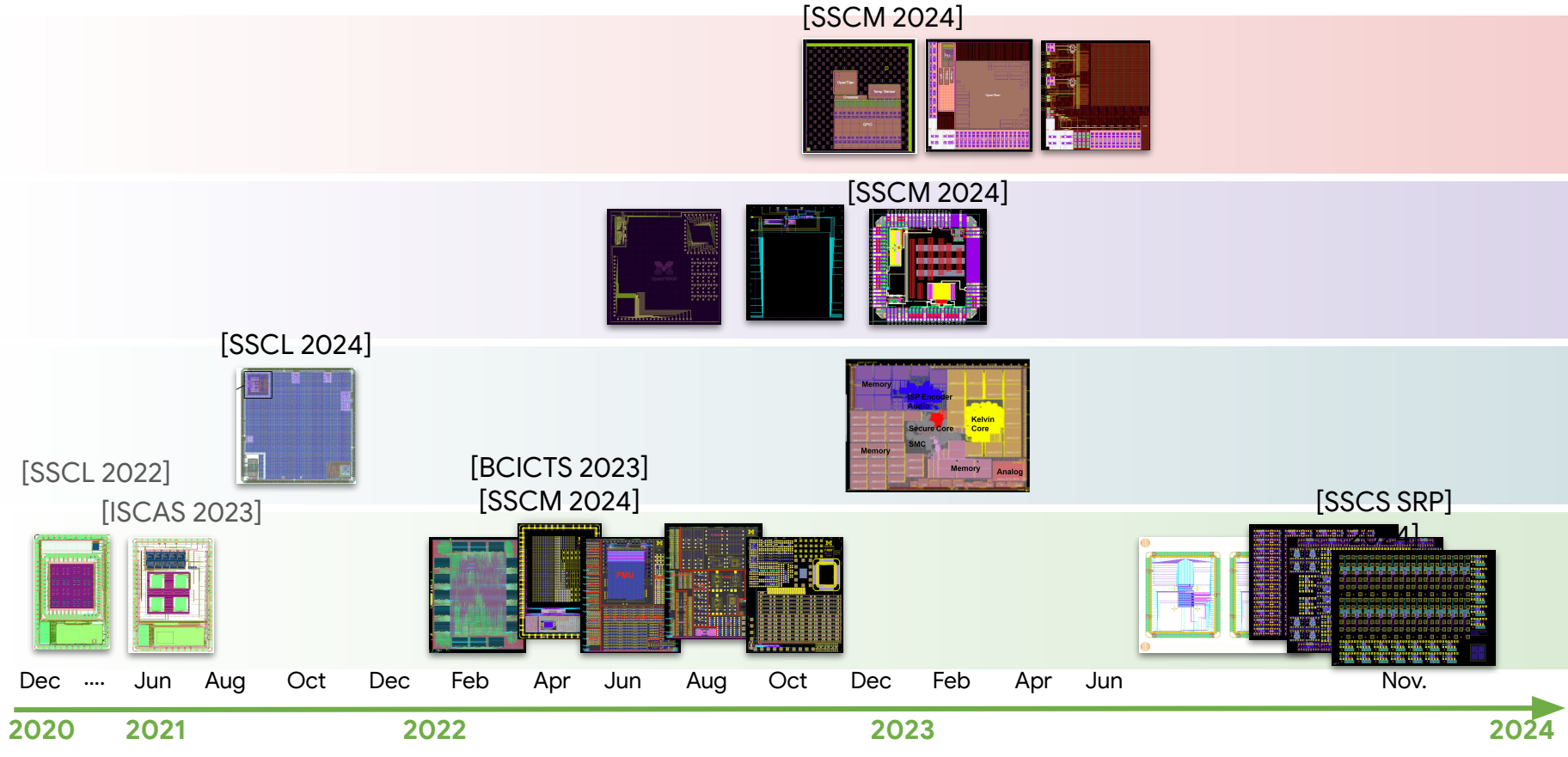
Benefits:

- Internet-Free Operation
- Real-Time Interaction
- Physical Interfaces



Missing Piece of the Puzzle: LLMs on ASICs

ASICs are the Key to *Efficient* On Device Operation



Dr. Saligane and Team's Collaborative Tapeouts with Partnerships

Open-Source Foundations for ASIC Collaborations

README Apache-2.0 license

OpenFASoC

OpenFASoC: Fully Open-Source Autonomous SoC Synthesis using Customizable Cell-Based Synthesizable Analog Circuits

docs failing

OpenFASOC is focused on open-source automated analog generation from user specification to GDSII with fully open-sourced tools. This project is led by a team of researchers at the University of Michigan and is inspired by FASoC, that sits on proprietary tools. (See more about FaSoC at website)

- ***Temperature sensor -***

sky130hd_temp-sense-generator failing

Build and test with the latest version of tools set failing

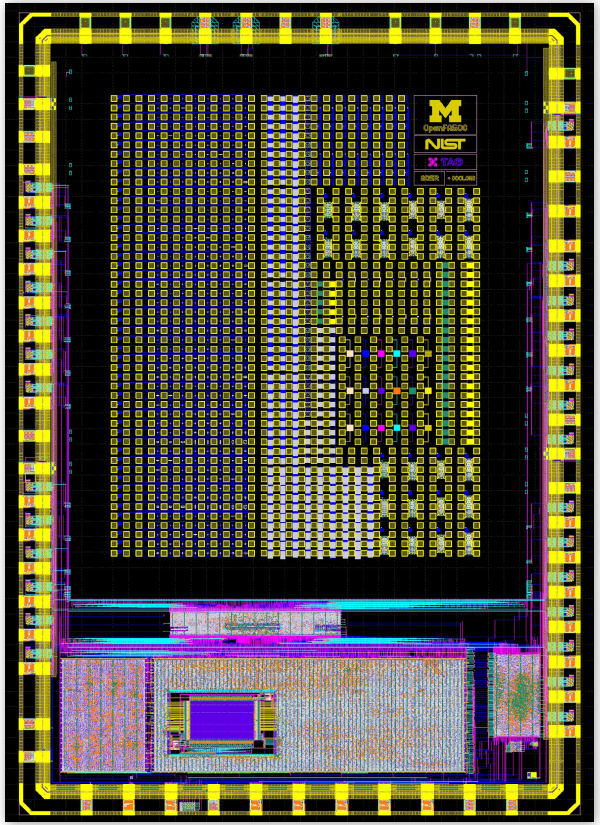
Open in Colab

Contributors 33

+ 19 contributors

Languages

Python 39.6% Tcl 26.0%
SourcePawn 18.3%
Makefile 9.5% Verilog 4.1%
SystemVerilog 1.4%
Other 1.1%



https://github.com/idea-fasoc/OpenFASOC

HW and Software Co-Design for LLM Silicon Development

ReaLLMASIC

Overview

ReaLLMAsic aims to bridge the gap between theoretical model design and practical hardware implementation, ensuring efficient, scalable, and robust ML model development.

Our project stands out for its extensive exploration of various model configurations and modules, catering to a diverse range of use cases.

Key exploration features include:

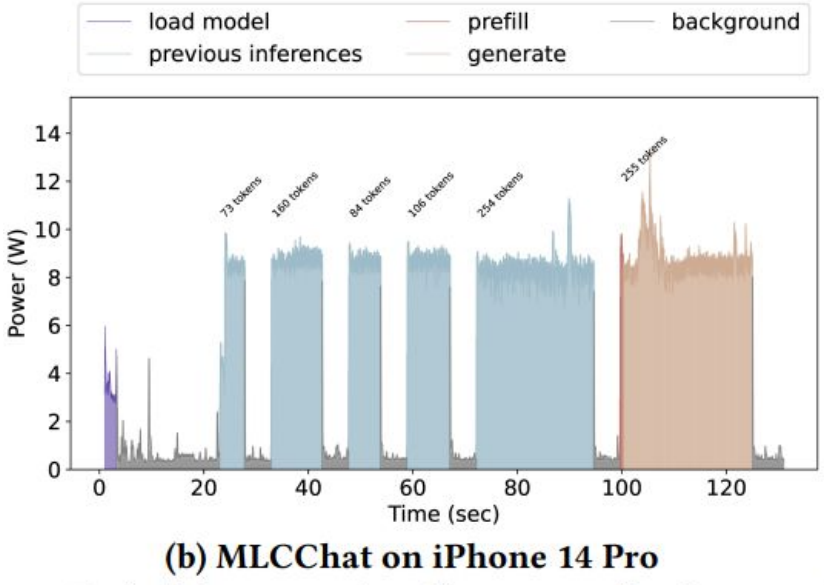
- `Module Variation`: Explore with different module types -- e.g. Softmax, Softermax, ConSmax, and SigSoftmax -- discover which is best suited (PPA) to your application.
- `Flexible Tokenization`: Explore different tokenization: tiktoken, sentencepiece, phonemization, character level, custom tokenization, etc.
- `Diverse Dataset Performance Testing`: Evaluate model efficacy across various languages and datasets including: csv-timeseries, mathematics, music, lyrics, literature, and webtext.
- `Standard and Custom Hyperparameters`: Fine-tune models using conventional hyperparameters and explore the impact of custom settings on model performance and PPA impacts.

[ReaLLMASIC on Github]

Specific Challenges

Better HW/SW Efficiency LLMs needed for Edge Devices

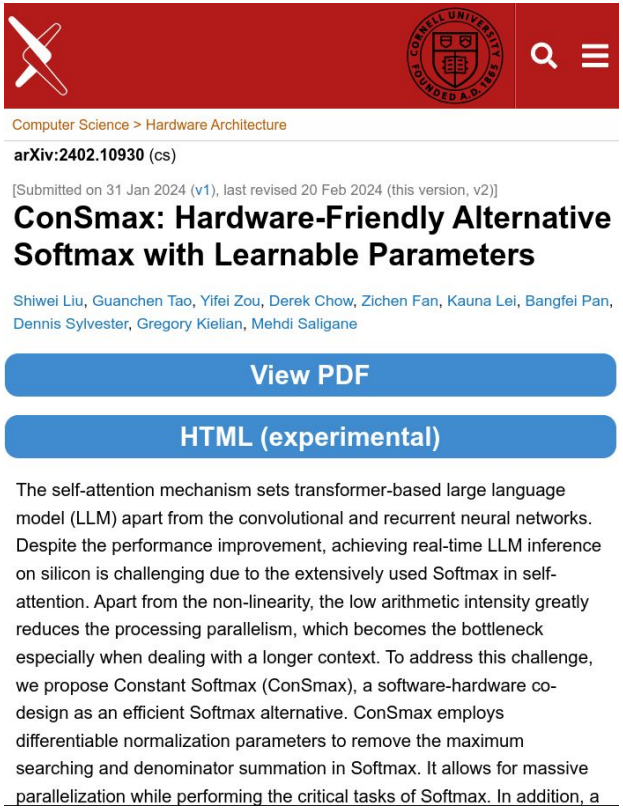
An IPhone 14 47.9C (118.22F) after just a 2 minute conversation with Zephyr-3B-4bit at 10 tokens/sec



Present Hardware Cannot Run LLMs Efficiently

More Efficient Algorithms: Softmax

Co-Design and Hardware And Software Collaborations



Computer Science > Hardware Architecture

arXiv:2402.10930 (cs)

[Submitted on 31 Jan 2024 (v1), last revised 20 Feb 2024 (this version, v2)]

ConSmax: Hardware-Friendly Alternative Softmax with Learnable Parameters

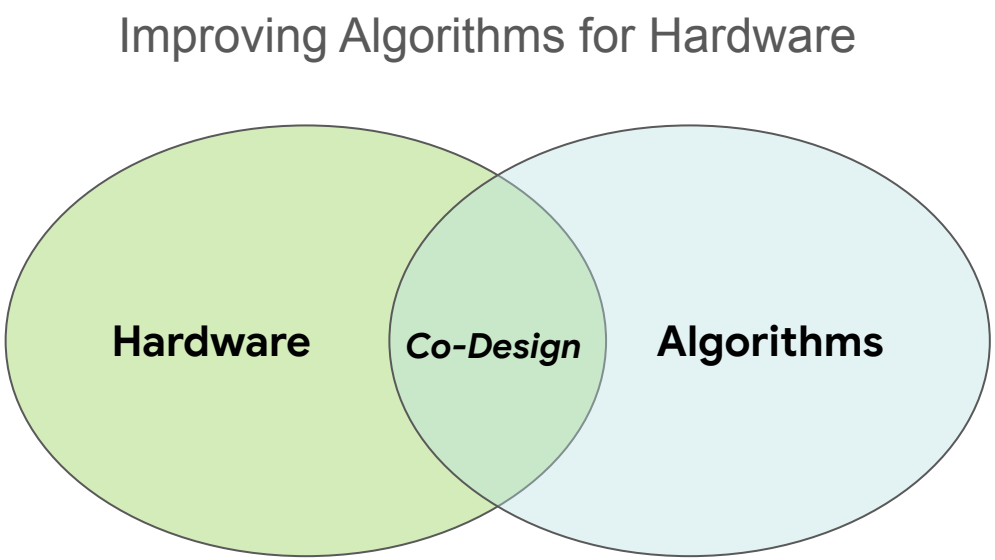
Shiwei Liu, Guanchen Tao, Yifei Zou, Derek Chow, Zichen Fan, Kauna Lei, Bangfei Pan, Dennis Sylvester, Gregory Kielian, Mehdi Saligane

View PDF

HTML (experimental)

The self-attention mechanism sets transformer-based large language model (LLM) apart from the convolutional and recurrent neural networks. Despite the performance improvement, achieving real-time LLM inference on silicon is challenging due to the extensively used Softmax in self-attention. Apart from the non-linearity, the low arithmetic intensity greatly reduces the processing parallelism, which becomes the bottleneck especially when dealing with a longer context. To address this challenge, we propose Constant Softmax (ConSmax), a software-hardware co-design as an efficient Softmax alternative. ConSmax employs differentiable normalization parameters to remove the maximum searching and denominator summation in Softmax. It allows for massive parallelization while performing the critical tasks of Softmax. In addition, a

https://arxiv.org/abs/2402.10930



More Efficient Algorithms: Normalization

Layernorm Occurs Twice Per Layer

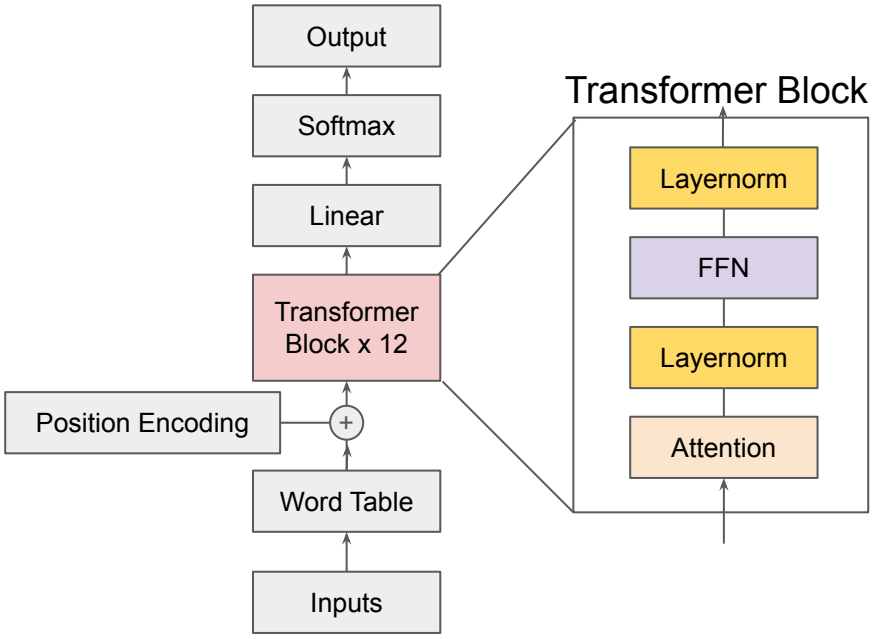
- Algorithm has steps with O(sequence_length) operations
- "**Normalized**" by dividing by variance and/or recentering mean of vectors.

$$\bar{a}_i = \frac{a_i - \mu}{\sigma} g_i$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} a_i, \quad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - \mu)^2}$$

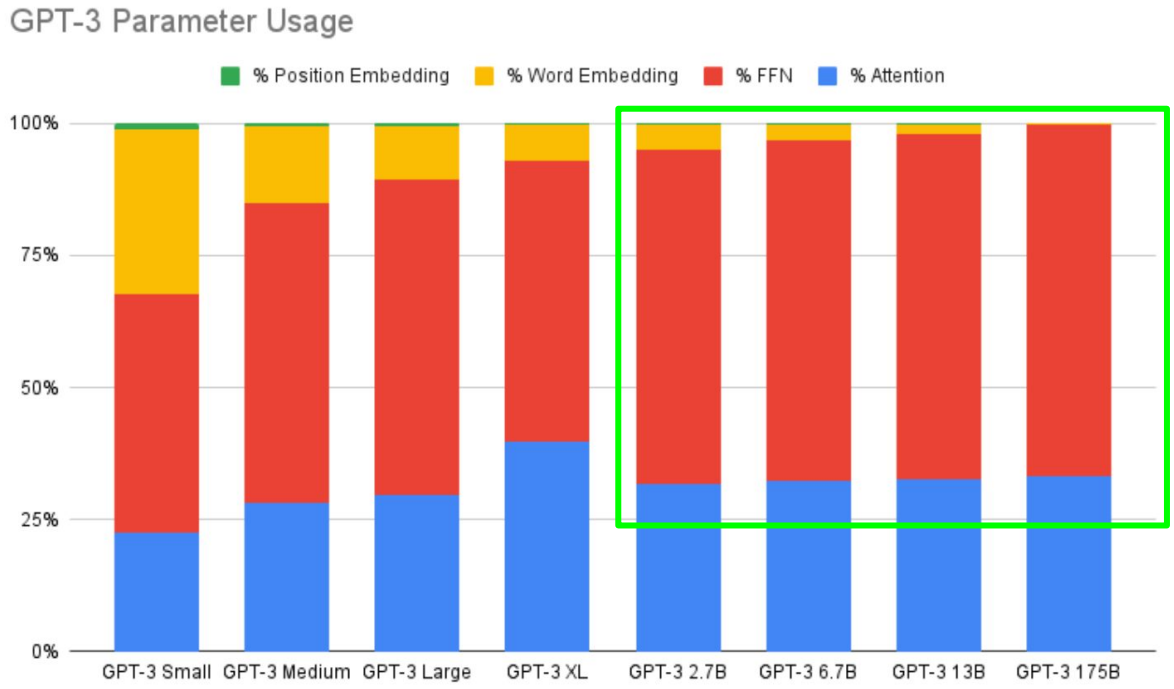
https://arxiv.org/abs/1607.06450
https://arxiv.org/abs/1910.07467

LLM Transformer Architecture



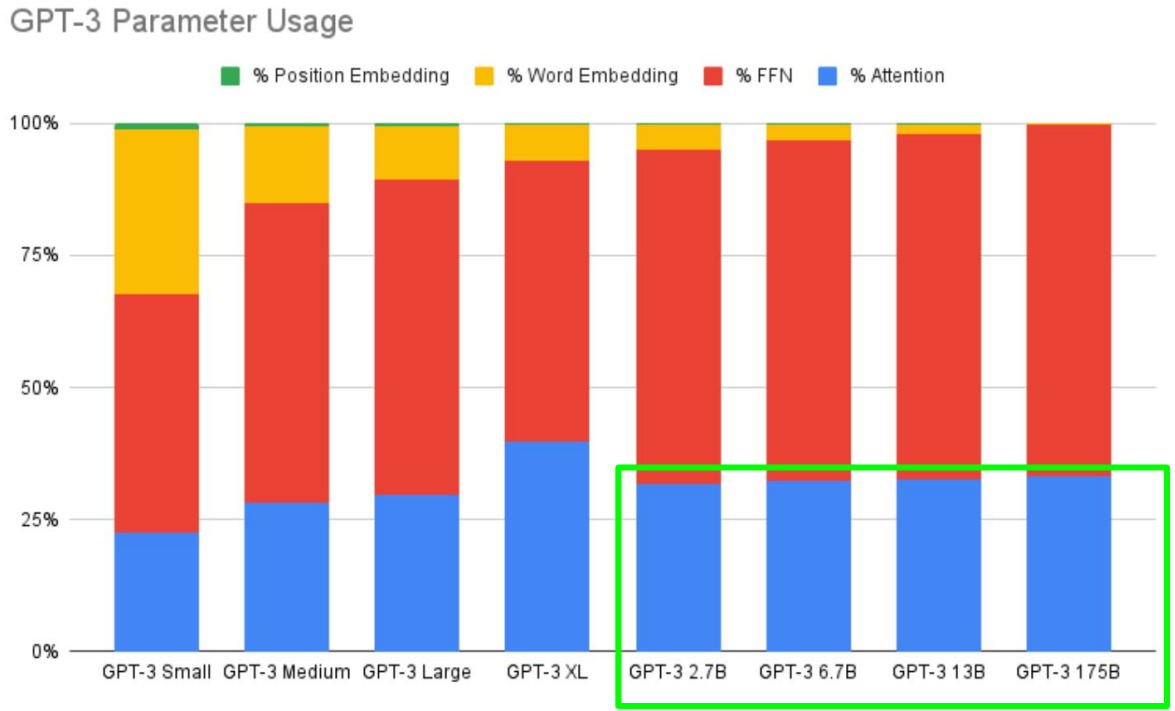
Memory Efficiency

FFN (aka MLP) Holds ~50-75% of Parameters



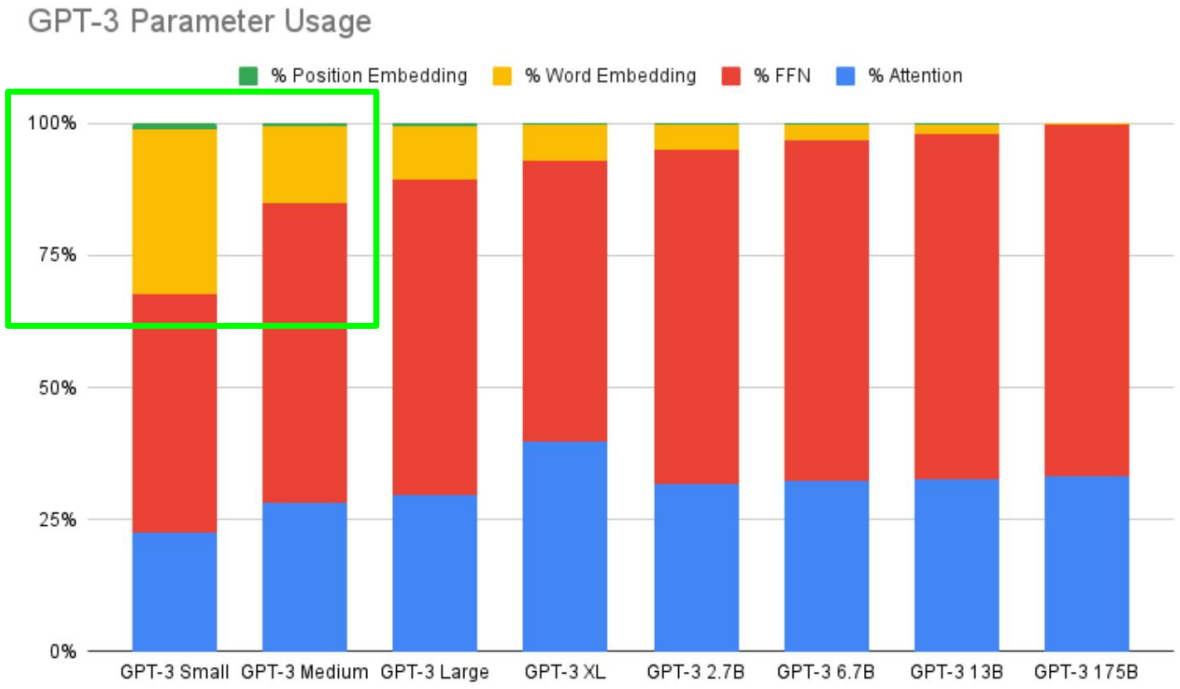
https://aizi.substack.com/p/how-does-gpt-3-spend-its-175b-parameters

Attention layer holds ~25-30% of Parameters



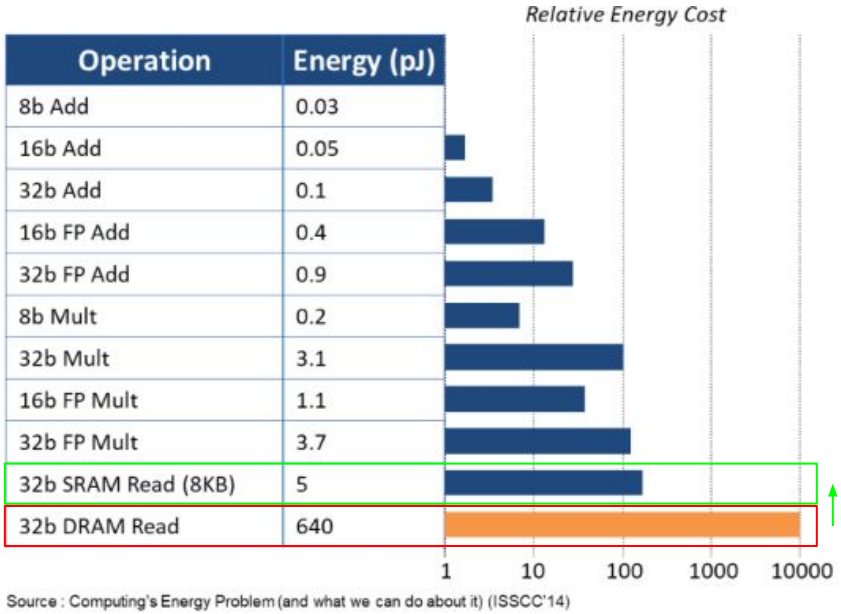
https://aizi.substack.com/p/how-does-gpt-3-spend-its-175b-parameters

In Small Networks, *Word Embeddings Have Large %*



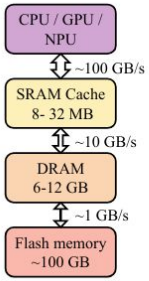
https://aizi.substack.com/p/how-does-gpt-3-spend-its-175b-parameters

DRAM and Off-Chip Memory Reads Largest Energy Sink



Operation	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640

Source : Computing's Energy Problem (and what we can do about it) (ISSCC'14)



Hardware	Device	SoC last level memory size	DRAM size
Apple A16	iPhone 15	24 MB	6 GB
Apple A15	iPhone 14	32 MB	6 GB
Google	Pixel 8	8 MB	8 GB / 12 GB (pro)
QCOM	Snapdragon 8	10 MB	8-12 GB

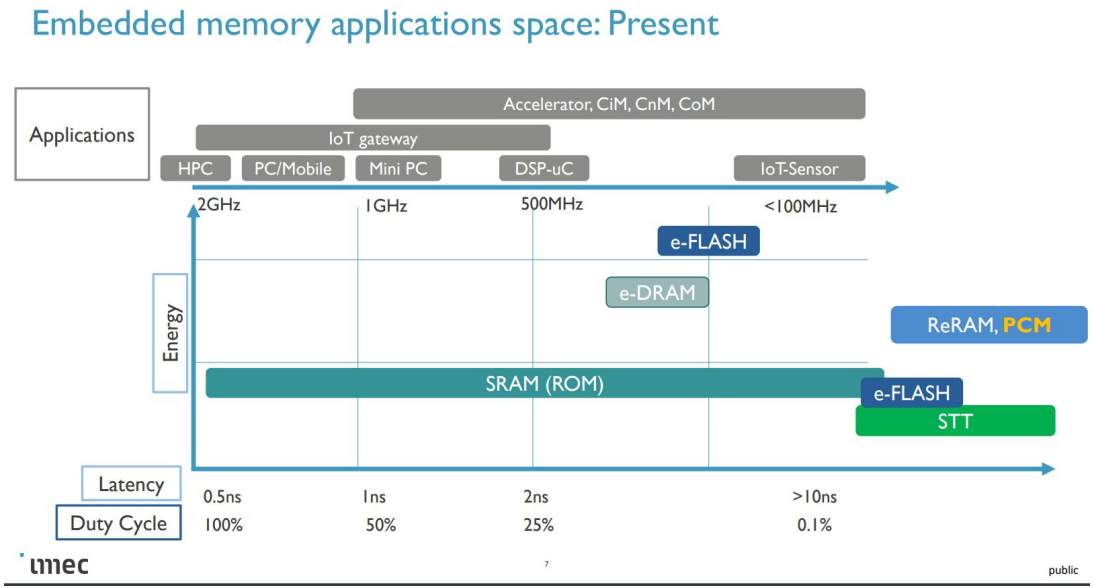
Figure 2: Memory hierarchy in prevalent mobile devices. Despite adequate Flash storage, the operational memory for executing high-speed applications predominantly resides in DRAM, typically constrained to 6-12 GB.

https://arxiv.org/abs/2402.14905

Iphone 14 47.9C (118.22F) after 2 minutes of conversation with Zephyr-3B-4bit at 10 tokens/sec

Ideal Memory Characteristics

- Low Read Power
- More Memory
- High Area Efficiency



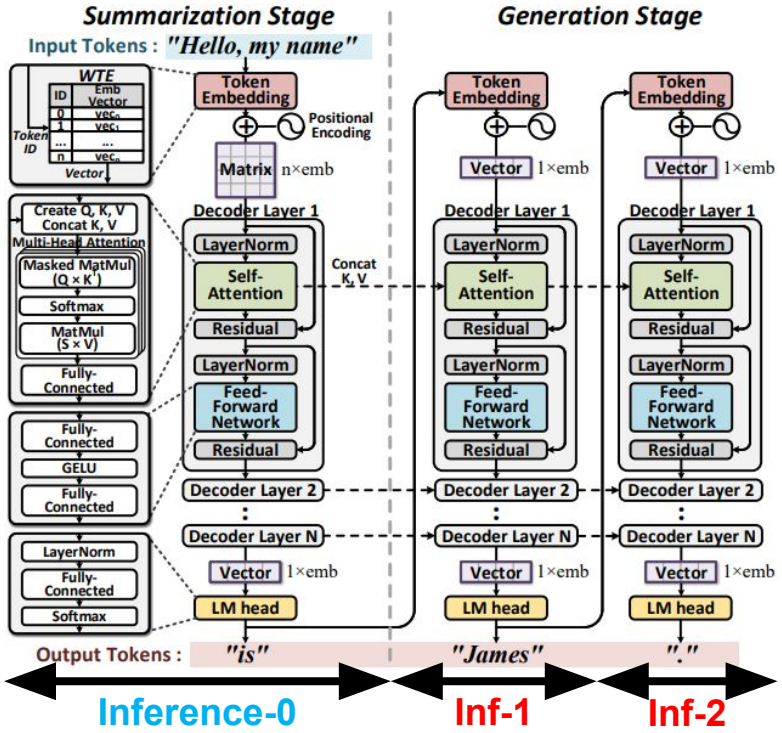
VLSI 2022: Slides imec

Hardware Acceleration

Edge AI LLM Accelerators

Google AI

❑ Summarization and Generation Stage in LLM



Summarization and Generation in LLM

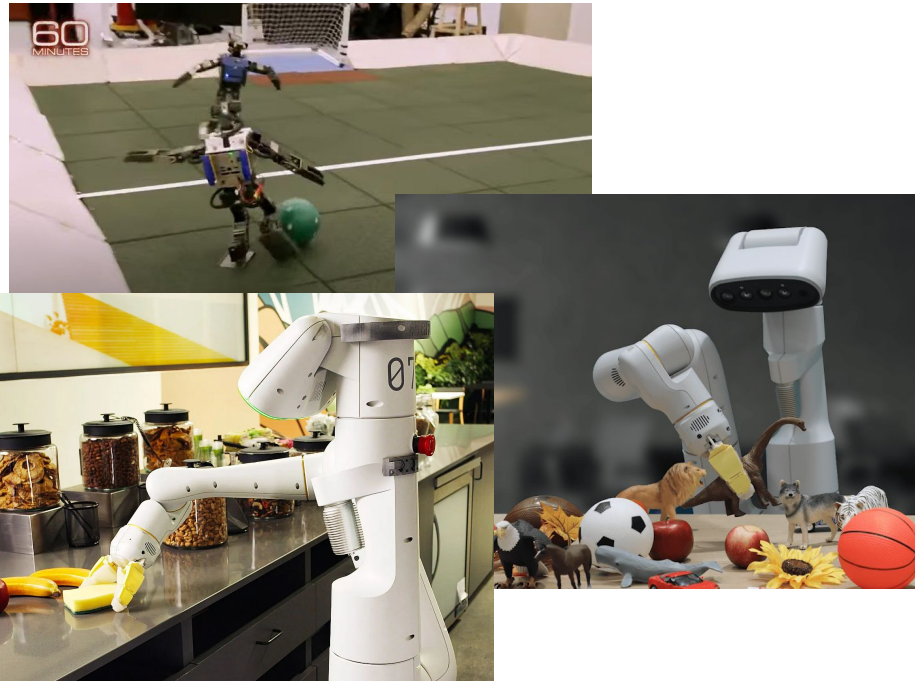
DFX, KAIST, MICRO'2022

- Summarization Stage:
  - ❑ Summarize and extract features from input prompt.
  - ❑ Model receives multiple input tokens from the given prompt.
  - ❑ Compute-Bounded by general matrix-matrix multiplication.
- Generation Stage:
  - ❑ Generate new tokens.
  - ❑ Each inference produces one new token.
  - ❑ Memory-Bounded by general matrix-vector multiplication at small batch size (= 1).

Directions and Collaborations

The Future is What We Make It

- Custom Silicon and EdgeLLM ASICs can make Robotics and AI Ubiquitous
- Partnerships are needed for bringing AI to Life.
- Let’s make the future!



Let’s build some silicon together!