Designing Hardware for Machine Learning

John Wawrzynek

johnw@berkeley.edu
University of California, Berkeley

Machine Learning has been Great for Computer Architects!

- With the slowing of Moore's Law and the end of Dennard Scaling, architects have turned to "accelerators" and purpose built processors as a way to continue to scale performance, energy efficiency, and cost.
- For ML, relatively easy to achieve high efficiency (compared to general purpose code) - simpler control flow.
- Embarrassing parallel (loads of data-level parallelism), lower precision requirements.
- Highly impactful!
- No end in sight!

Outline

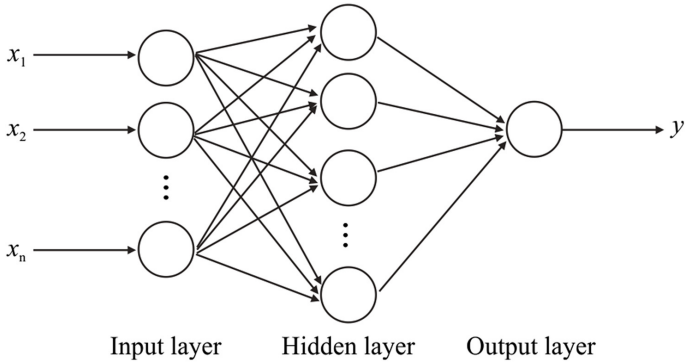
Past

Present

Future

Work at Berkeley - Early 1990's

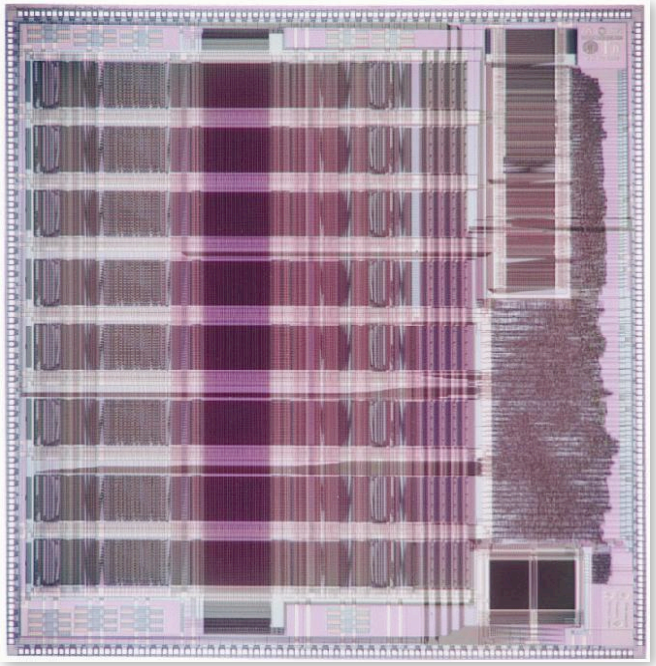
- Community wide keen interest in *parallel processing*
  - *How to we find parallelism in problems and exploit in hardware?*
- ICSI was successful at using ANN's trained with back-propagation for front-end signal processing of speech signals for speech recognition tasks. MLPs not DNNs!
- Training was taking months on CPUs.



- Our pitch: connectionist models (neural networks) could be a general model for many computations and are naturally parallel.
- Got funded from the ONR to design and build a "connectionist network supercomputer (CNS)".
- Quickly realized that training and inference is dominated by multiply/add operations and these vectorize => designed a vector processor.

T0 Vector Microprocessor (1995)

- World's first single chip vector processor
- MIPS CPU + vector lanes
- Three graduate students, 1 year
- 15X the performance of workstation

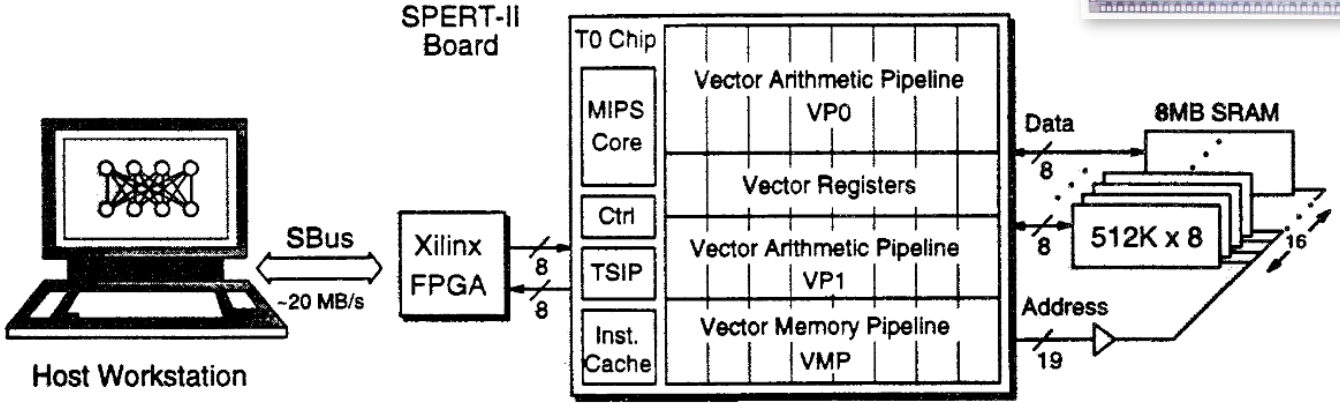


SPERT-II: A Vector Microprocessor System and its Application to Large Problems in Backpropagation Training

John Wawrzynek, Krste Asanović, & Brian Kingsbury
University of California at Berkeley
Department of Electrical Engineering and Computer Sciences
Berkeley, CA 94720-1776

James Beck, David Johnson, & Nelson Morgan
International Computer Science Institute
1947 Center Street, Suite 600
Berkeley, CA 94704-1105

Proceedings of MicroNeuro '96



Libraries for Speech researchers

System lived on for a decade!

Lessons from the 90's

1. NN training/inference dominated by multiply/add operations
    - Full precision rarely needed
    - Data-level parallelism (vectors/matrices)

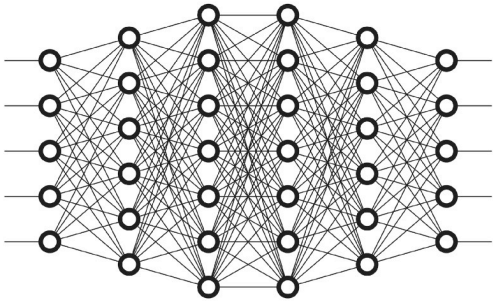
Obviously lives on today with TPUs, etc.

2. ANN's good general computation model
    - Wide range of function approximation, regression, classification, etc., useful tool in optimization.

3. Software is key to adoption

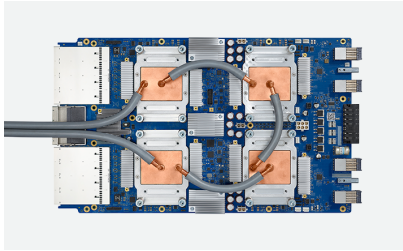
GPUs with Cuda/openCL, now TensorFlow/PyTorch

Learning Model Capabilities Scaled Directly with Hardware Advances



- Late 2000's - renewed interest in NNs, now **deep**
- Driven by availability of high-performance hardware (GPUs)
- ML models and HW development fundamentally linked:
- Success in LLMs tied directly to massive hardware compute capability (Even more important than algorithm details?)

Example: LLMs
GPT-4 trained on ~25,000 Nvidia A100 GPUs for 90-100 days,
~1.8 trillion parameters across 120 layers (~13T tokens in training)
[https://archive.md/2RG8X]



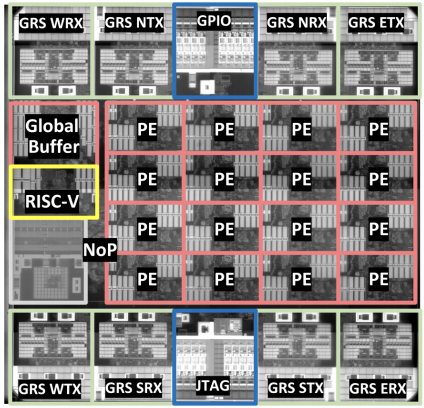
Cloud TPU v3 (45 TFLOP/s)

- How can we continue to scale HW performance (efficiency) to the benefit of ML?
  - *Scalable HW architectures + HW/Algorithm co-design*

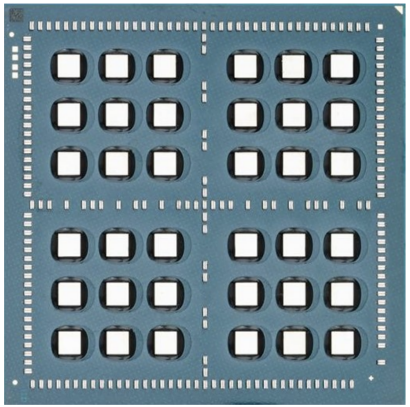
Increasing Number of Parallel Resources

Many PEs with Network on Chip/Package (NoC/NoP)

NoC/NoP Chip



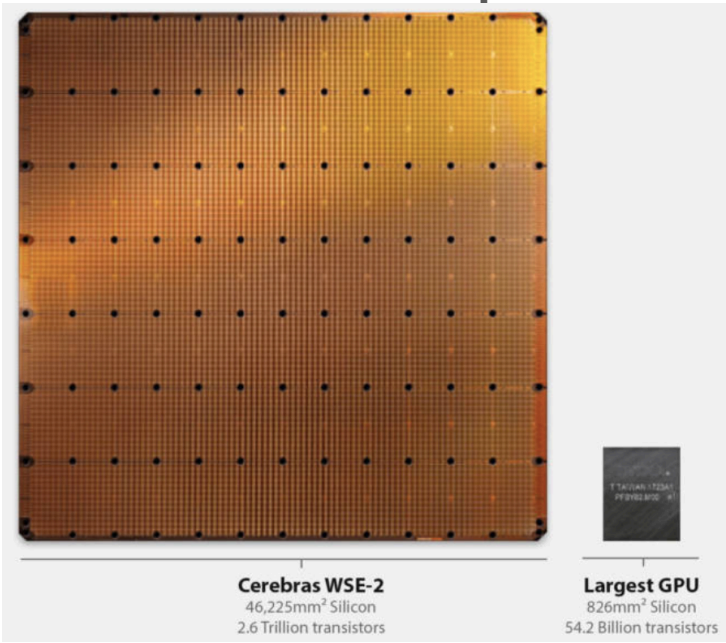
(a) Simba chiplet



(b) Simba package

[Simba]
16PEs x 36 Chiplets

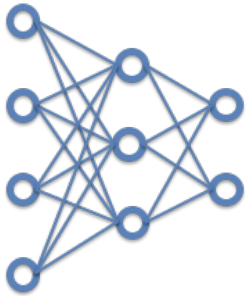
Wafer-scale Chip

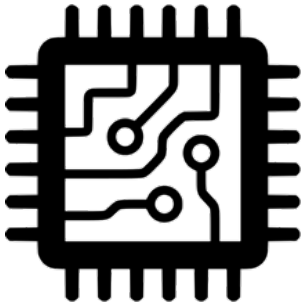


Cerebras
84 Interconnected Chips

Scheduling a constant challenge:

Particularly for multi-core architectures. How to partition and schedule execution to efficiently use parallel resources.





- Algorithm

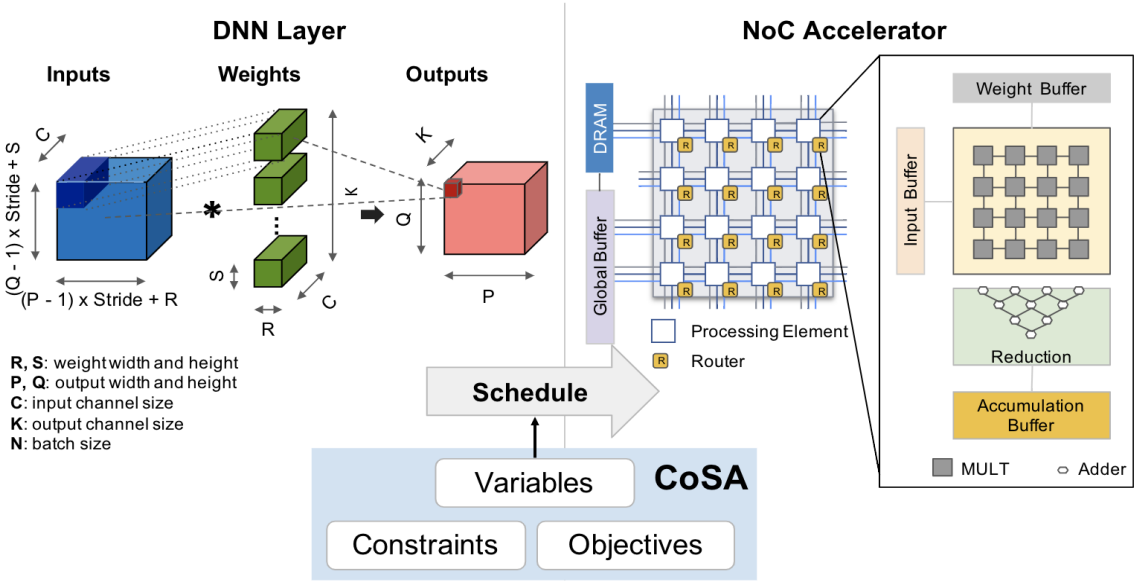
Problem instances are huge (large amount of state, large number of operations)

- Hardware

Relatively small amount of fast hardware resources (memory, computational units)

Perhaps the most important part of support software.

CoSA: Scheduling by Constrained Optimization for Spatial Accelerators [21'ISCA]



Scheduling Decisions:

1. Loop tiling
2. Loop permutation
3. Spatial mapping

Objective is to minimize latency and energy

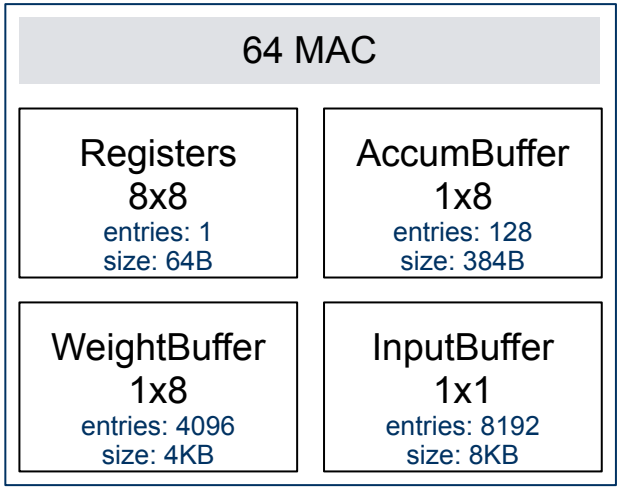
Mixed Integer Programming (MIP) for scheduling DNN on NoC accelerator with multi-level memory hierarchies

Three operation-level scheduling decisions

- **Inputs Constraints:**

Problem: ----- 7 nested loops
R=3, S=3, P=28, Q=28, C=128, K=128, N=1

Architecture: ----- 5 levels of memory



- **Output Schedule:**

```
DRAM [ Weights:147456 Inputs:115200 Outputs:100352 ]
-----------------------------------------------------
| for P in [0:4)
|   for S in [0:3)
|    for C in [0:16) (Spatial-X)
InputBuffer [ Inputs:2016 ]
--------------------------
|      for N in [0:1)
|       for R in [0:3) (Spatial-X)
WeightBuffer [ Weights:1024 ]
----------------------------
|         for Q in [0:28)
|          for P in [0:7)
AccumulationBuffer [ Outputs:128 ]
----------------------------------
|             for K in [0:128)
|              for C in [0:8)
Registers [ Weights:1 ]
-----------------------
|                for N in [0:1)
```

1. Loop Permutation
2. Spatial Mapping
Temporal Mapping
3. Tiling Factors

State-of-the-art DNN accelerator schedulers

Scheduler	Search Algorithm
Brute-force Approaches:	
Timeloop [57]	Brute-force & Random
dMazeRunner [28]	Brute-force
Triton [75]	Brute-force over powers of two
Interstellar [81]	Brute-force
Marvel [17]	Decoupled Brute-force
Feedback-based Approaches:	
AutoTVM [19]	ML-based Iteration
Halide [65]	Beamsearch [4], OpenTuner [9], [52]
FlexFlow [42]	MCMC
Gamma [45]	Genetic Algorithm
Constrained Optimization Approaches:	
Polly+Pluto [15], [16], [35]	Polyhedral Transformations
Tensor Comprehension [77]	
Tiramisu [11]	
CoSA	**Mixed Integer Programming (MIP)**

1. Expensive and time-consuming
2. Sample invalid space
3. Hard to generalize

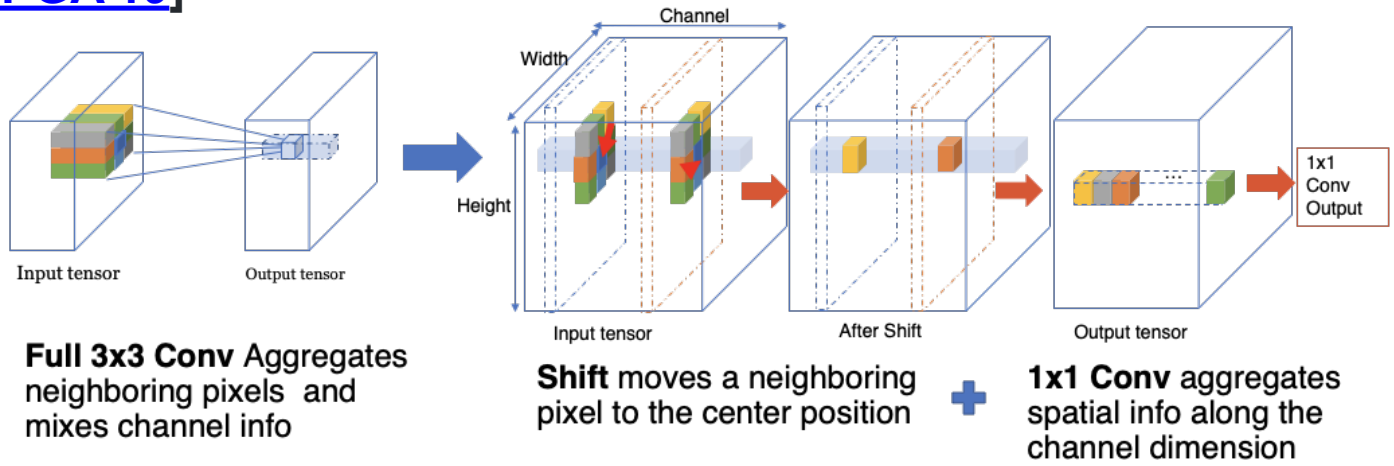
Unable to determine tiling factor sizes

One-pass solution

2x speedup compared to the state-of-the-art work with **116x** shorter time-to-solution

Hardware-Friendly Algorithm Design

Synetgy: Image Classification without 3x3 Convolution [FPGA'19]



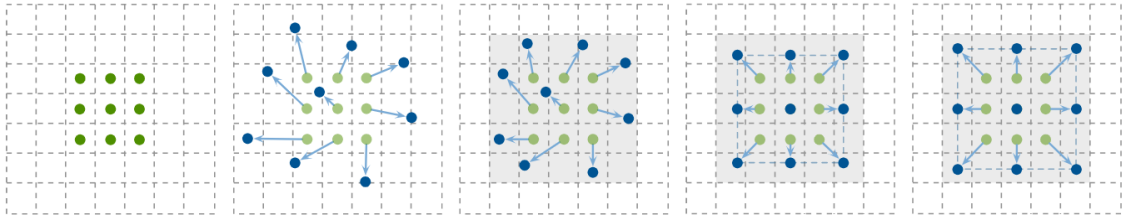
- 3x3 Conv → Shift and 1x1 Conv
- Dataflow accelerator on embedded FPGA

equal top-1 accuracy, **11.6x** higher frame-rate, **6.3x** better power efficiency, on ImageNet classification task

Hardware-Friendly Algorithm Design

CoDeNet: Object Detection with Deformable Convolution Codesign [FPGA'21]

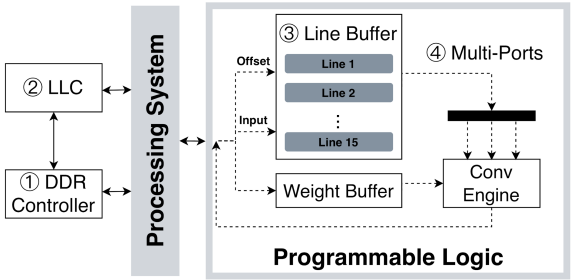
I. Algorithm Modifications



(a) Regular (b) Deformable (c) Bound d) Square. (e) Round

Deformable Conv samples inputs from variable offsets generated based on the input, and leads to the state-of-the-art accuracy for object recognition tasks.

II. Hardware Optimizations

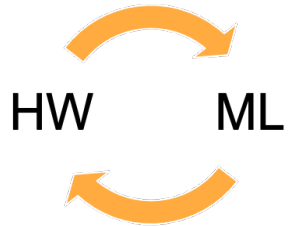


(2) Caching (3) Buffering (4) Parallel Ports

- **9.76×** speedup for the deformable conv on FPGA
- **20.9×** smaller model but **10%** higher accuracy than Tiny-YOLO

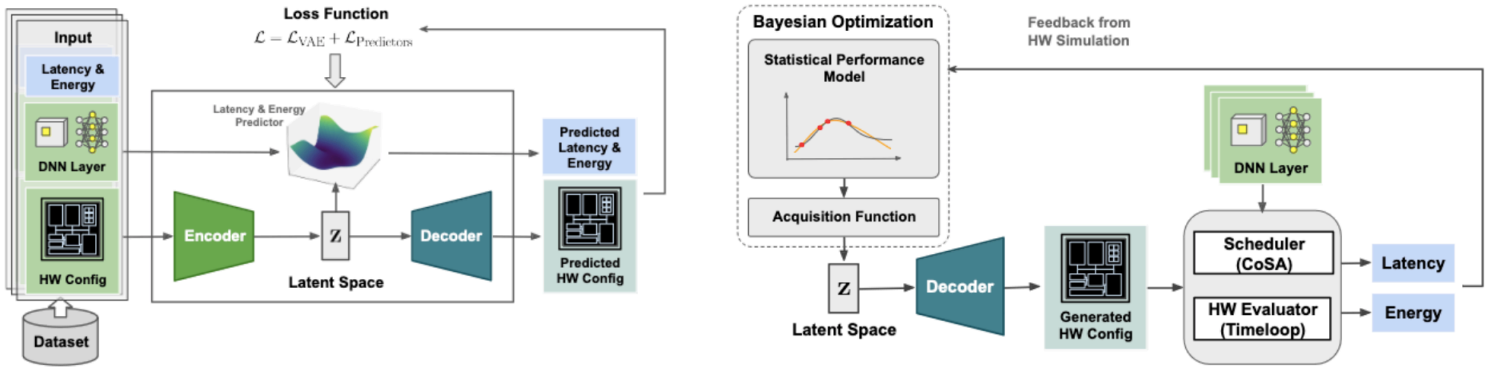
The Future

- Advances in ML will continue to be critically dependent on advances in Hardware Design.
  - To spur ML advances HW design must be agile: easy, fast, cheap
    - None of these true now!
    - Chip/accelerator design is slow, expensive (years, $10-100M)
    - Consequently, we use yesterday's application benchmarks to design tomorrow's HW!
- There's hope: HW design greatly benefits from ML
- ML methods can help in architecture/logic synthesis, optimization, and verification of ML hardware circuits and systems



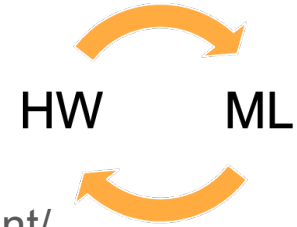
Learning A Continuous and Reconstructible Latent Space for Hardware Accelerator Design (VAESA)

2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)



- HW design space exploration (DSE) exponentially large in design parameters & discontinuous
- Design challenge: Design Simba-like (multicore) architectures for DNNs optimizing for latency and energy
  - parameters such as # of PEs, weight and input buffer sizes, … 3.6 x 10^{17} configurations!
- We use a *variational autoencoder (VAE)* to learn a compressed and continuous representation (latent) of the design space - new designs can be generated from the latent space
- Eases search (Bayesian optimization, and gradient-based search)
- Demonstrated on AlexNet, ResNet-50, ResNeXt-50, Deep Bench, …
- Significantly improves optimization results versus searches in original design space (5%) and 6.8X better sample efficiency

The Future



- Emerging body of work:
  - Many solid results on ML for physical design problems (placement/ routing)
  - Some in logic and high-level synthesis
  - Most build estimators of power, cost, or performance to aid in search
  - Few positive results on “generative techniques”
    - ex: “Design an LDPC decoder with 1Gsps throughput and 10mW for SK90FD”
    - LLMs probably not suitable
    - “Large Circuit-Models”?
      - Challenges: Training sets, and correctness guarantees, constraint satisfaction

Co-authors / Collaborators

- T0 Vector Microprocessor: *Krste Asanovic, Brian Kingsbury, James Beck, David Johnson, Nelson Morgan*
- Synetgy: *Yifan Yang, Qijing Huang, Bichen Wu, Tianjun Zhang, Liang Ma, Giulio Gambardella, Michaela Blott, Luciano Lavagno, Kees Vissers, Kurt Keutzer*
- Codenet: *Qijing Huang, Dequan Wang, Zhen Dong, Yizhao Gao, Yaohui Cai, Tian Li, Bichen Wu, Kurt Keutzer*
- CoSA: *Qijing Huang, Minwoo Kang, Grace Dinh, Thomas Norell, Aravind Kalaiah, James Demmel, Sophia Shao*
- VAESA: *Qijing Huang, Charles Hong, Mahesh Subedar, Sophia Shao*

Thanks!