香港中文大學
The Chinese University of Hong Kong

EDA for AI Chip Designs

Bei Yu
Chinese University of Hong Kong
byu@cse.cuhk.edu.hk

July 30, 2023

Autonomous drive

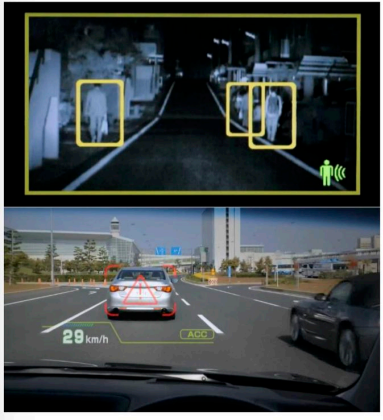
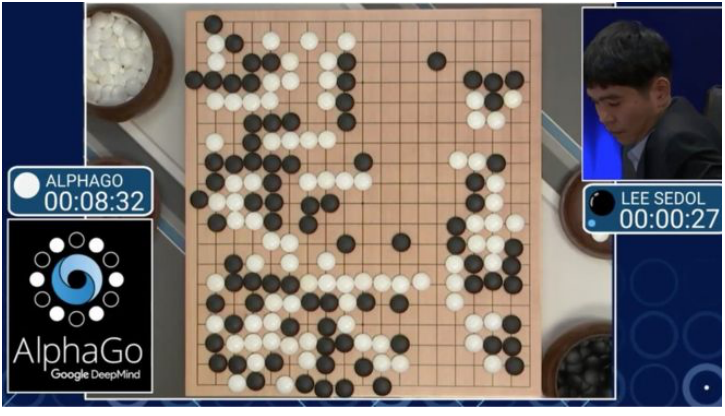


Image recognition

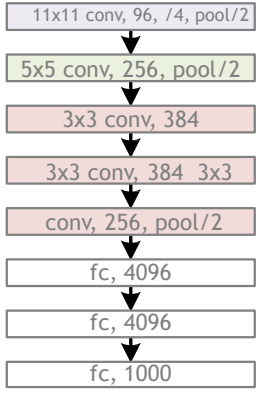
Renaissance of Deep Learning (2006 –)

- A fast learning algorithm for deep belief nets. [Hinton et.al 1996]
- Data + Computing + Industry Competition
- NVidia's GPU, Google Brain (16,000 CPUs)
- Speech: Microsoft [2010], Google [2011], IBM
- Image: AlexNet, 8 layers [Krizhevsky et.al 2012] (26.2% -> 15.3%)



Revolution of Depth

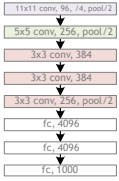
AlexNet, 8 layers (ILSVRC 2012)



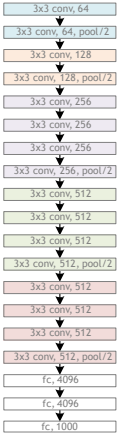
Slide Credit: He et al. (MSRA)

Revolution of Depth

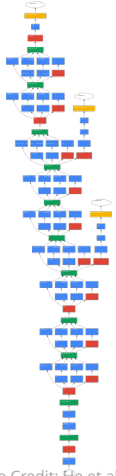
AlexNet, 8 layers (ILSVRC 2012)



VGG, 19 layers (ILSVRC 2014)



GoogleNet, 22 layers (ILSVRC 2014)



Slide Credit: He et al. (MSRA)

Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)

VGG, 19 layers (ILSVRC 2014)

ResNet, 152 layers (ILSVRC 2015)

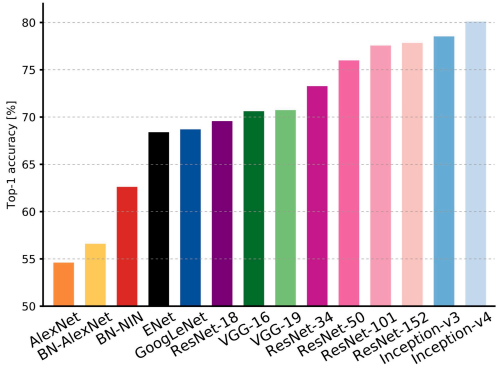
Slide Credit: He et al. (MSRA)

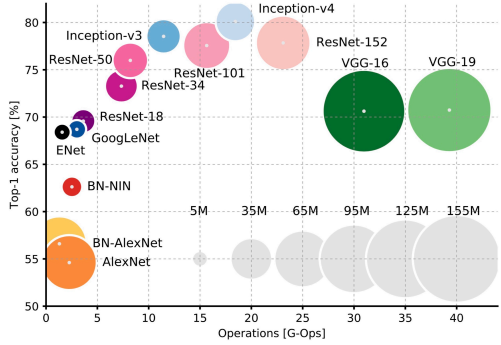
Some Recent Classification Architectures

- AlexNet (Krizhevsky, Sutskever, and E. Hinton 2012) 233MB
- Network in Network (Lin, Chen, and Yan 2013) 29MB
- VGG (Simonyan and Zisserman 2015) 549MB
- GoogleNet (Szegedy, Liu, et al. 2015) 51MB
- ResNet (He et al. 2016) 215MB
- Inception-ResNet (Szegedy, Vanhoucke, et al. 2016)
- DenseNet (Huang et al. 2017)
- Xception (Chollet 2017)
- MobileNetV2 (Sandler et al. 2018)
- ShuffleNet (Zhang et al. 2018)

Some Recent Classification Architectures

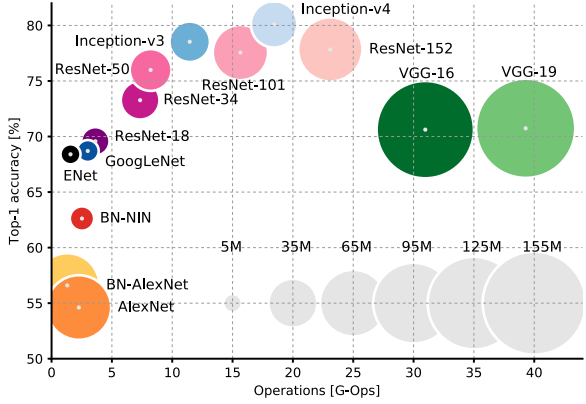
- AlexNet (Krizhevsky, Sutskever, and E. Hinton 2012) 233MB
- Network in Network (Lin, Chen, and Yan 2013) 29MB
- VGG (Simonyan and Zisserman 2015) 549MB
- GoogleNet (Szegedy, Liu, et al. 2015) 51MB
- ResNet (He et al. 2016) 215MB
- Inception-ResNet (Szegedy, Vanhoucke, et al. 2016) 23MB
- DenseNet (Huang et al. 2017) 80MB
- Xception (Chollet 2017) 22MB
- MobileNetV2 (Sandler et al. 2018) 14MB
- ShuffleNet (Zhang et al. 2018) 22MB





1

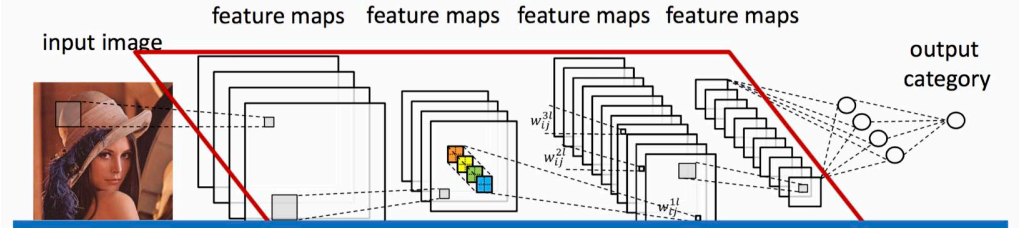
[1] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello (2017). "An analysis of deep neural network models for practical applications". In: *arXiv preprint*.



Why AlexNet is large in size, but small in operations?

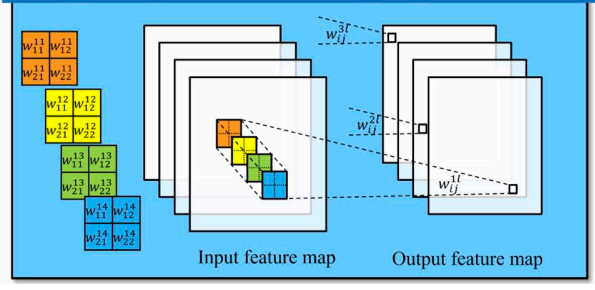
- Special FC layers
- Special Conv layers
- More channels
- Some redundant operators

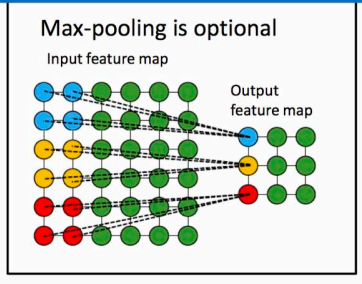
Convolutional Neural Network (CNN)



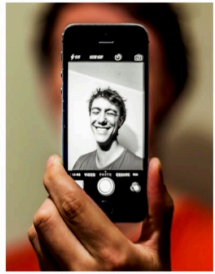
Convolutional layers account for over 90% computation

[1] A. Krizhevsky, etc. Imagenet classification with deep convolutional neural networks. NIPS 2012.
[2] J. Cong and B. Xiao. Minimizing computation in convolutional neural networks. ICANN 2014





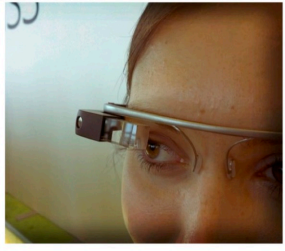






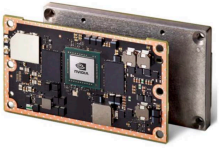




Embedded CV

Flexibility vs. Efficiency

CPU
(Raspberry Pi3)



GPU
(Jetson TX2)

FPGA
(UltraZed)

ASIC
(Movidius)

Flexibility

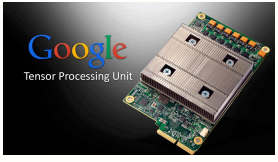
Power/Performance
Efficiency

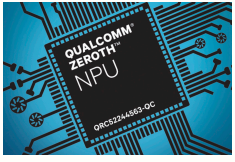
Convolution layer is one of the most expensive layers

- Computation pattern
- Emerging challenges

More and more end-point devices with limited memory

- Cameras
- Smartphone
- Autonomous driving



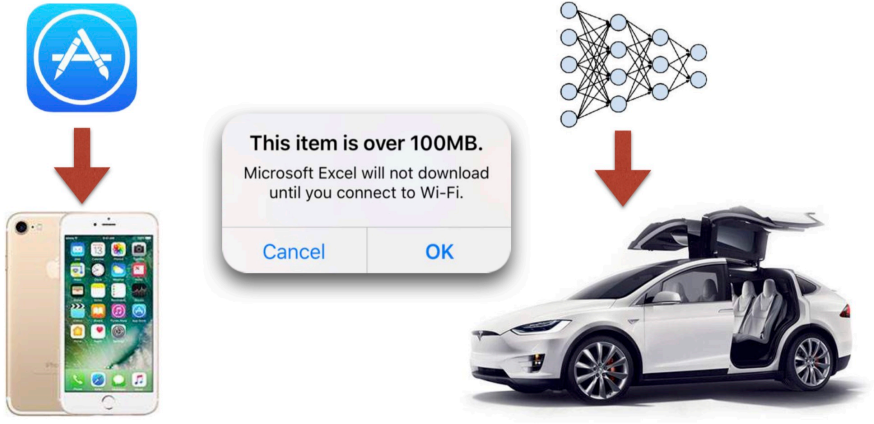


XILINX

MOBILEYE
An Intel Company

1st Challenge: Model Size

Hard to distribute large models through over-the-air update



[2]

[2] Song Han and William J. Dally (2018). "Bandwidth-efficient Deep Learning". In: *Proc. DAC*, 147:1–147:6.

2nd Challenge: Energy Efficiency

AlphaGo: 1920 CPUs and 280 GPUs,
$3000 electric bill per game

on mobile: drains battery
on data-center: increases TCO

[3]

[3]Song Han and William J. Dally (2018). "Bandwidth-efficient Deep Learning". In: *Proc. DAC*, 147:1–147:6.

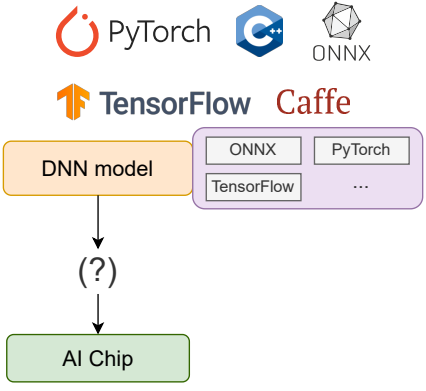
Application Category

Both	Datacenter	Edge
Intel, Nvidia, IBM, Xilinx, HiSilicon, Google, Baidu, Alibaba Group, Cambricon, DeePhi, Bitmain, Wave Computing	AMD, Microsoft, Apple, Tencent Cloud,Aliyun, Baidu Cloud, HUAWEI Cloud, Fujitsu, Nokia, Facebook, HPE, Thinkforce, Cerebras, Graphcore, Groq, SambaNova Systems, Adapteva, PEZY	Qualcomm, Samsung, STMicroelectronics, NXP, MediaTek, Rockchip, Amazon_AWS, ARM, Synopsys, Imagination, CEVA, Cadence, VeriSilicon, Videantis, Horizon Robotics, Chipintelli, Unisound, AISpeech, Rokid, KnuEdge, Tenstorrent, ThinCI, Koniku, Knowm, Mythic, Kalray, BrainChip, AImotive, DeepScale, Leepmind, Krtkl, NovuMind, REM, TERADEEP, DEEP VISION, KAIST DNPU, Kneron, Esperanto Technologies, Gyrfalcon Technology, GreenWaves Technology, Lightelligence, Lightmatter, ThinkSilicon, Innogrit, Kortiq, Hailo,Tachyum

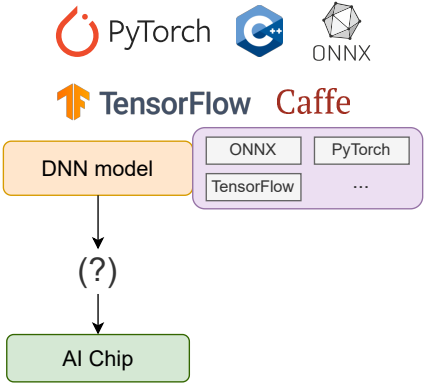
Source: https://basicmi.github.io/Deep-Learning-Processor-List/

DNN Deployment Flow

Deployment Flow: A Naive Approach

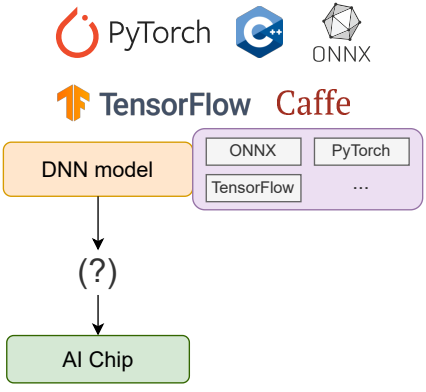


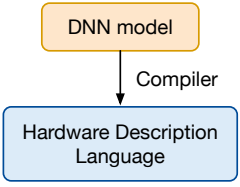
Deployment Flow: A Naive Approach



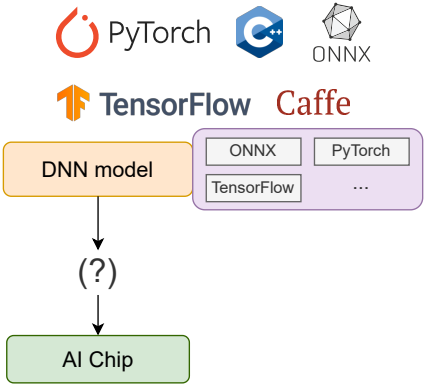
DNN model

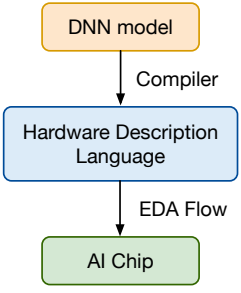
Deployment Flow: A Naive Approach



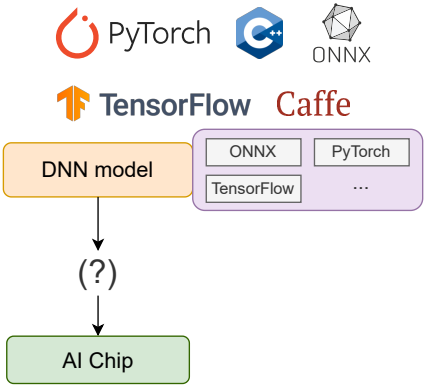


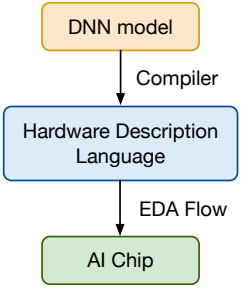
Deployment Flow: A Naive Approach





Deployment Flow: A Naive Approach

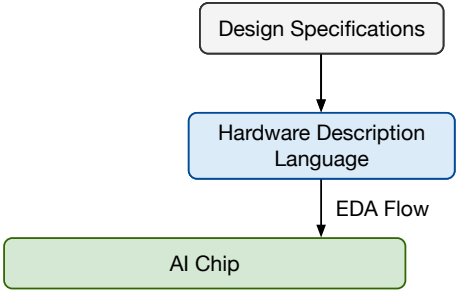


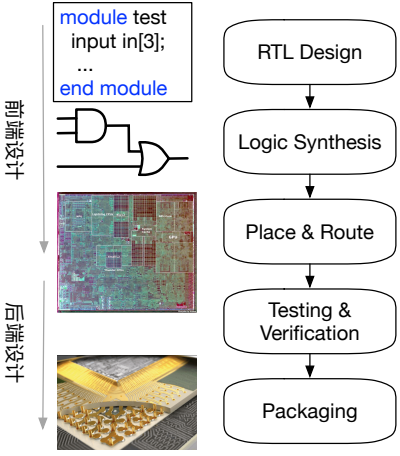


Question:

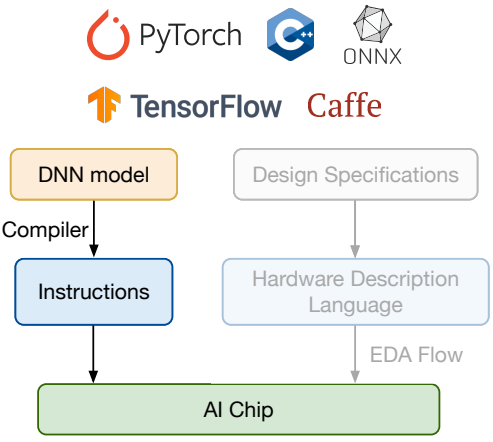
Why NOT using such deployment appoarch?

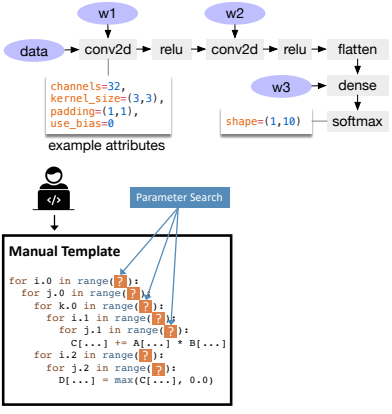
Deployment Flow: AI Chip Generation





Deployment Flow: DNN Model Compilation





Multi-Level Intermediate Representation (MLIR)[4]

LLVM

- Pro: Target-independent representation for optimization
- Con: Operating low-level operations is tricky

MLIR

- A tool for multi-level IR design (MLIR dialects)
- Enable different levels of abstraction

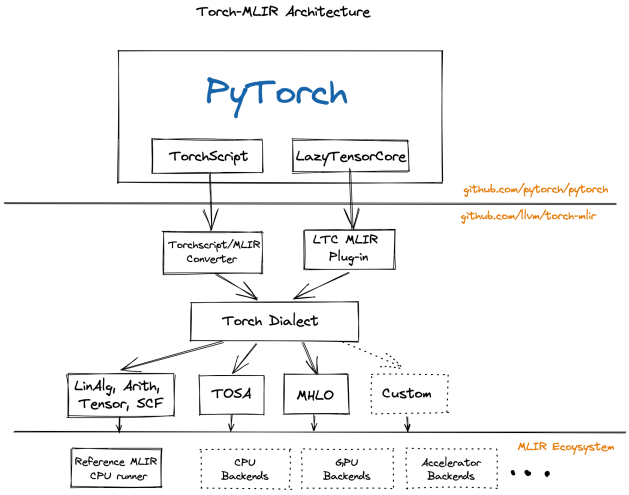
[4] https://mlir.llvm.org/

Torch-MLIR

The Torch-MLIR project aims to provide first class compiler support from the PyTorch ecosystem to the MLIR ecosystem.

Other MLIR Based DL Compiler:

1. OpenXLA
2. StableHLO
3. Triton
4. OneFlow



The overall architecture of Torch-MLIR

[5] https://github.com/llvm/torch-mlir

TPU-MLIR[6]

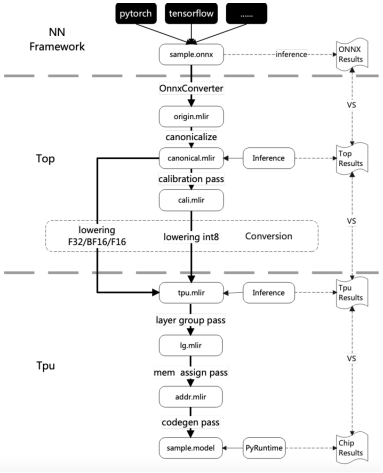
TPU-MLIR is an open-source machine-learning compiler based on MLIR for TPU.

Top Dialect:

- graph optimization
- quantization and inference
- ...

TPU Dialect:

- weight reordering
- operator slicing
- address assignment
- ...

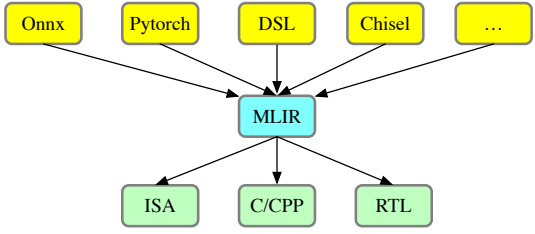


[6]Pengchao Hu et al. (2022). "TPU-MLIR: A Compiler For TPU Using MLIR". In: *arXiv preprint arXiv:2210.15016*.

Why MLIR?

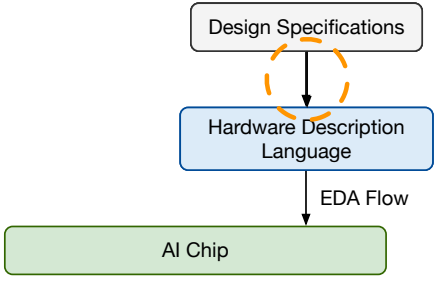
- Stable LLVM community
- Easy to implement
- Efficient optimization in multi-level abstract

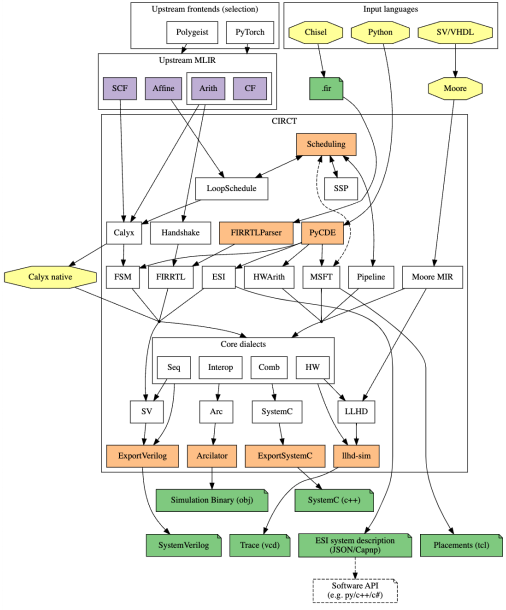
⋆: You can simply combine other MLIR projects by dialect conversion.



MLIR For Efficient Chip Design – CIRCT

- Circuit IR Compilers and Tools (CIRCT)
- apply MLIR and the LLVM development methodology
- to the domain of hardware design tools.





Calyx[7]

Calyx is a compiler infrastructure for languages that target hardware accelerators.

Firrtl [8]

Firrtl is an intermediate representation (IR) for digital circuits designed as a platform for writing circuit-level transformations.

Calyx and Firrtl has been integrated with the LLVM CIRCT infrastructure and is available as a dialect within it.

[7] Rachit Nigam et al. (2021). "A compiler infrastructure for accelerator generators". In: *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 804–817.

[8] https://github.com/chipsalliance/firrtl

[9] https://circt.llvm.org/

Update on Chisel[10]

Chisel v3.6.0:

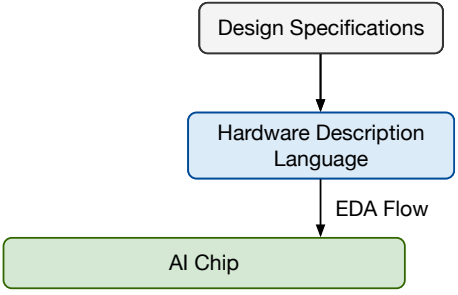
- The primary change in Chisel v3.6.0 is the transition from the Scala FIRRTL Compiler to the new MLIR FIRRTL Compiler(CIRCT/FIRTOOL).
- "Meaningless intermediate variable" problem is solved.

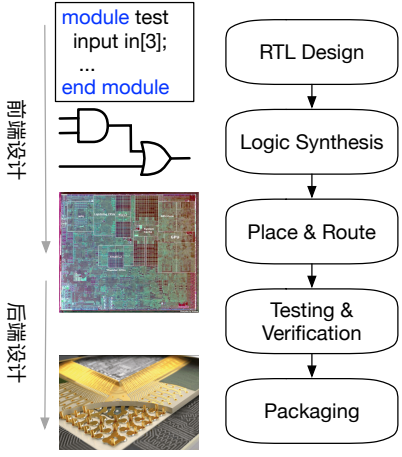
Chisel v5.0.0:

- Scala FIRRTL Compiler (SFC) is deprecated.

[10] Jonathan Bachrach et al. (2012). "Chisel: constructing hardware in a scala embedded language". In: *Proceedings of the 49th Annual Design Automation Conference*, pp. 1216–1225.

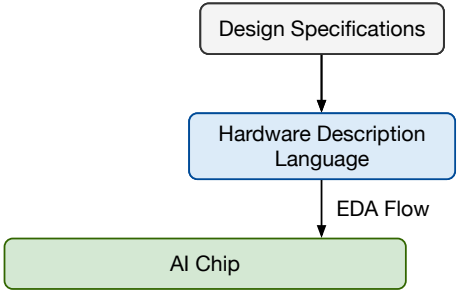
Today We Focus on EAD Part

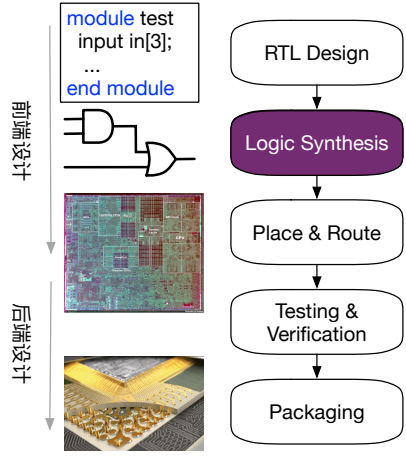




Outline

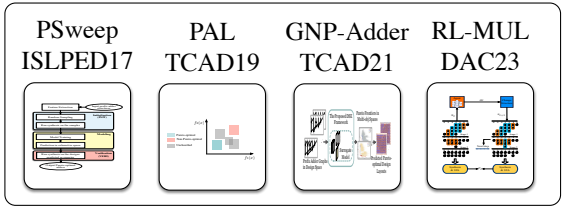
Arithmetic Unit Synthesis


Design Specifications
Hardware Description Language
EDA Flow
AI Chip

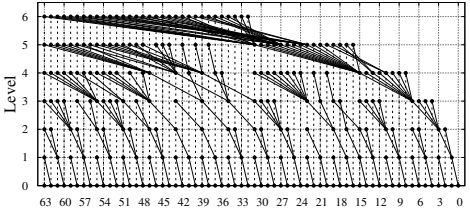

module test
input in[3];
...
end module
前端设计
后端设计
RTL Design
Logic Synthesis
Place & Route
Testing & Verification
Packaging

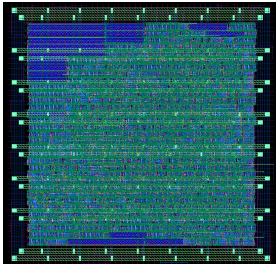
Arithmetic Unit Synthesis

Arithmetic Unit Synthesis



Logic synthesis v.s. Physical synthesis





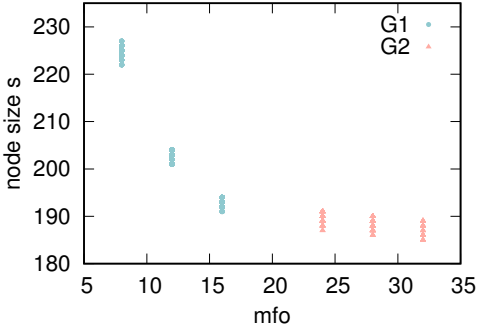
Constraints mapping between two synthesis stages is difficult.

Question:

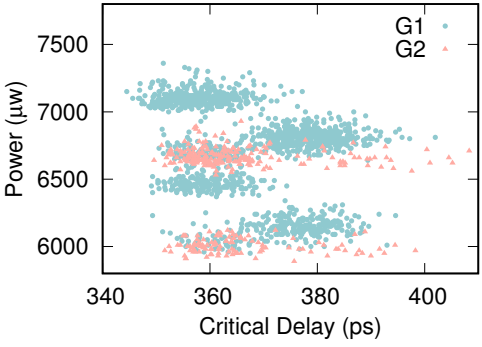
Why We Need to Optimize Arithmetic Circuits (Adder & Multiplier)?

Key Challenges

Front-End Team Perspective:



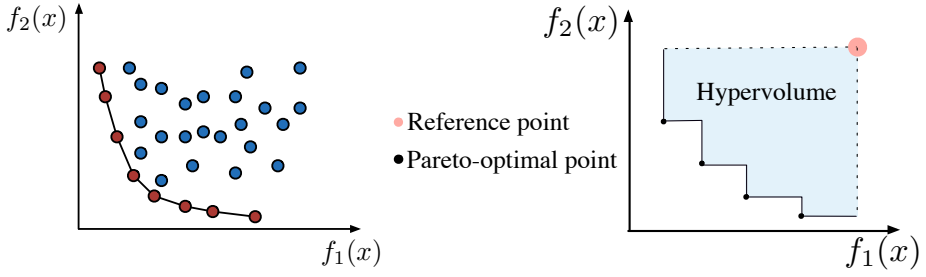
Back-End Team Perspective:



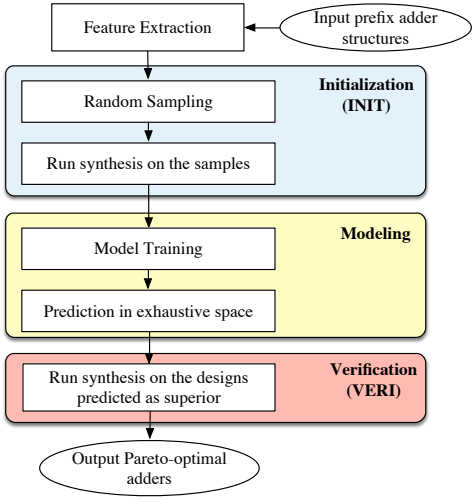
- Run design tools with all solutions is time-consuming. => **Enumeration is not feasible!**
- For 3K solutions, running time is $3000 \times 5 = 15$K mins.
- What we care: Pareto Frontier Curve

Pareto Frontier

- All the points are not dominated by any other point.
- Evaluation: Hyper-volume.
  - Size of the region bounded by the Pareto frontier and reference point.
  - Each dimension of the reference point is the maximum value on that dimension.



A Learning Flow for DSE in Adder Synthesis[ISLPED'17][4]



α-sweep learning

- General flow;
- Use a joint output Power-Delay function (PD) as the regression output rather than using any single output;
- Select different features in different applications.

[4]S. Roy, Y. Ma, J. Miao and B. Yu, "A learning bridge from architectural synthesis to physical design for exploring power efficient high-performance adders", ISLPED'17.

Can we do more?

Directions

- Can we improve the quality of the Pareto-Frontier?
- We use 3K samples to cover the solution space. Can we use less labelled data for training ?

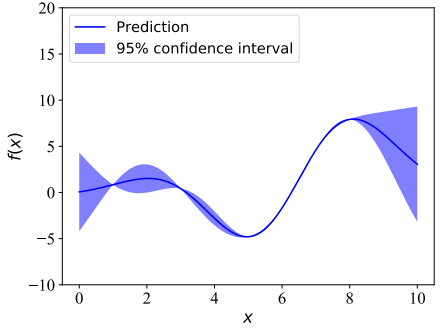
Solution: Active Learning

- Select representative samples to learn the property of the Pareto-Frontier;
- Fewer samples are needed, and the cost for labeling data is reduced.

Active Learning Flow [TCAD'19][2]

Regression

- Gaussian process model;
- A prediction consists of a mean and a variance;
- Off-the-shelf library for implementation.



[2]Y. Ma, S. Roy, J. Miao, J. Chen and B. Yu, "Cross-Layer Optimization for High Speed Adders: A Pareto Driven Machine Learning Approach", TCAD'19.

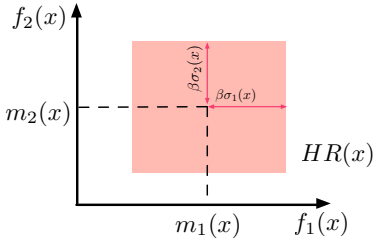
Uncertainty

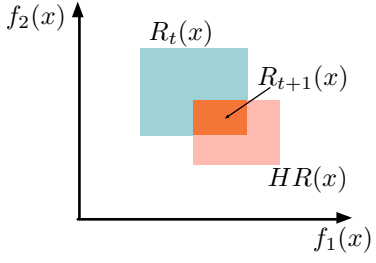
- Given the prediction (m, σ), a hyper-rectangle is defined as

$$HR(x) = \{\mathbf{y} : m_i(x) - \beta\sigma_i(x) \leq y_i \leq m_i(x) + \beta\sigma_i(x)\}$$

- The uncertainty region is defined as:

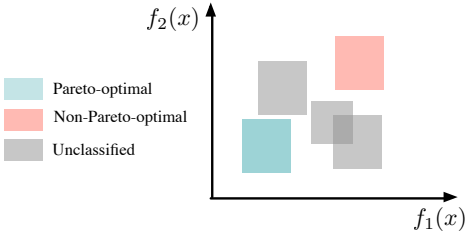
$$R_{t+1}(x) = R_t(x) \cap HR(x)$$





Classification

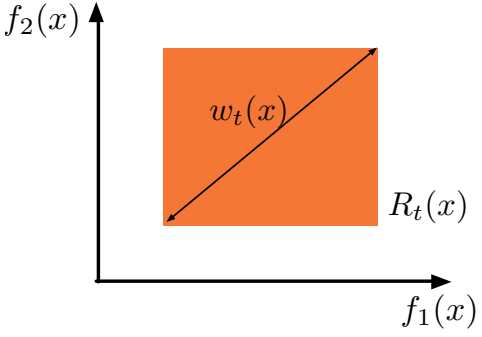
$$x \in \begin{cases} P, \text{ if } \max(R_t(x)) \leq \min(R_t(x')) + \delta, \\ N, \text{ if } \max(R_t(x')) \leq \min(R_t(x)) + \delta, \\ U, \text{ otherwise.} \end{cases}$$



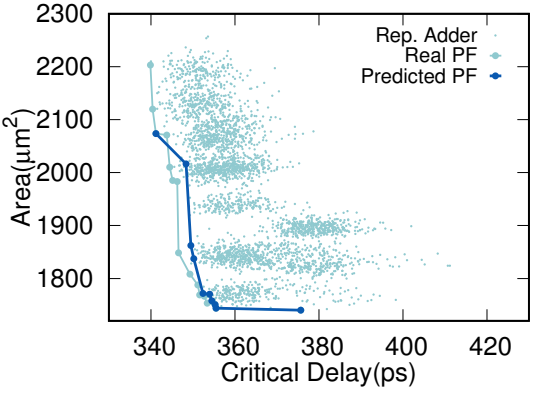
Sampling

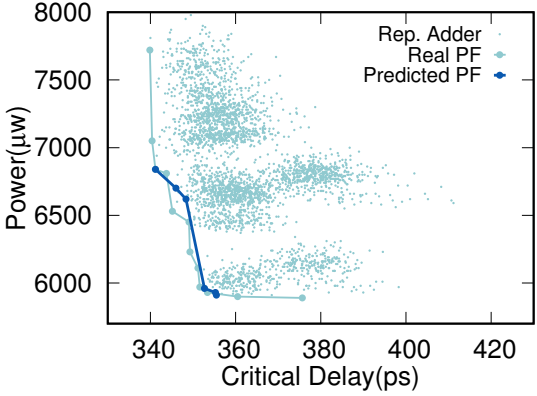
- Pick the one with largest uncertainty among the Pareto-optimal designs and unclassified designs.

$$w_t(x) = \max_{\mathbf{y},\mathbf{y}' \in R_t(x)} ||\mathbf{y} - \mathbf{y}'||_2.$$

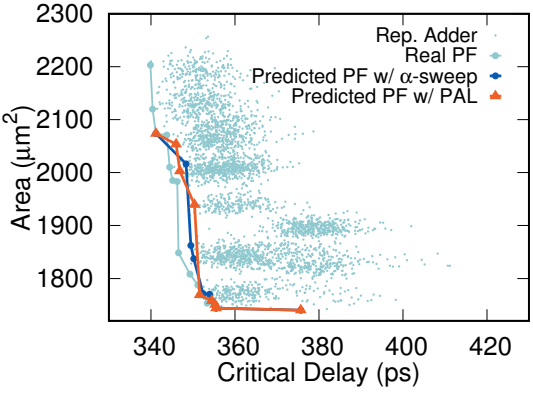


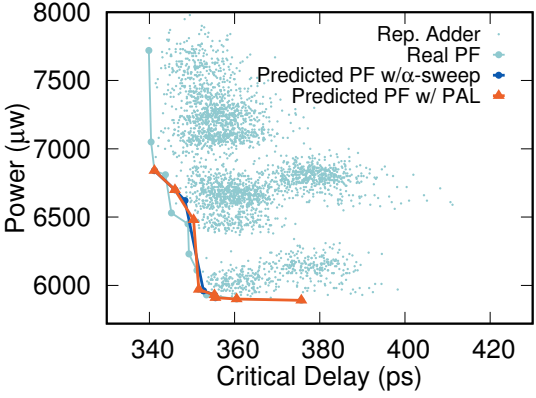
Pareto-Frontier Results



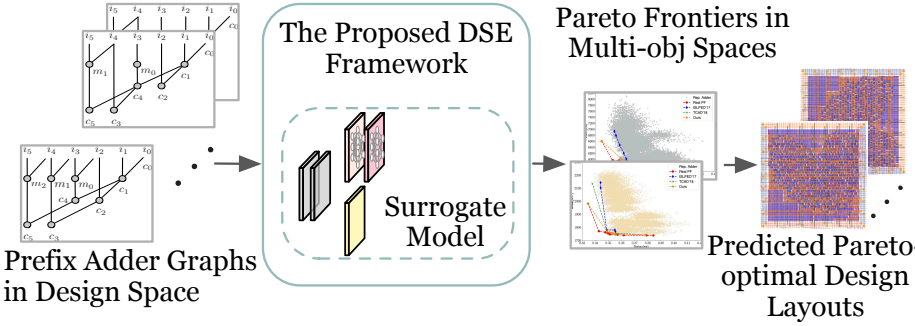


Pareto-Frontier Results





Graph Learning for Adder Structure [TCAD'21][3]

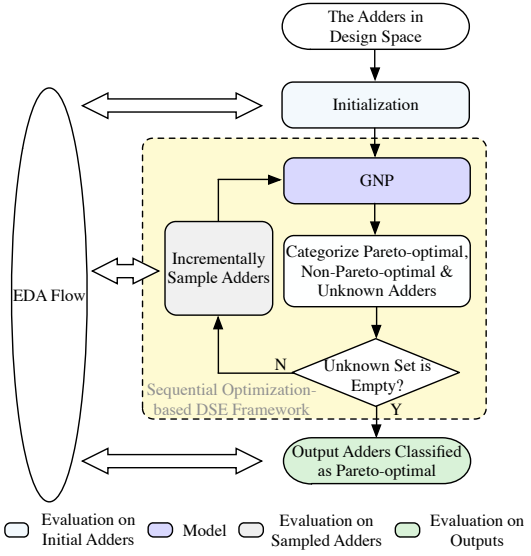


Graph Neural Process in the adder design space exploration.

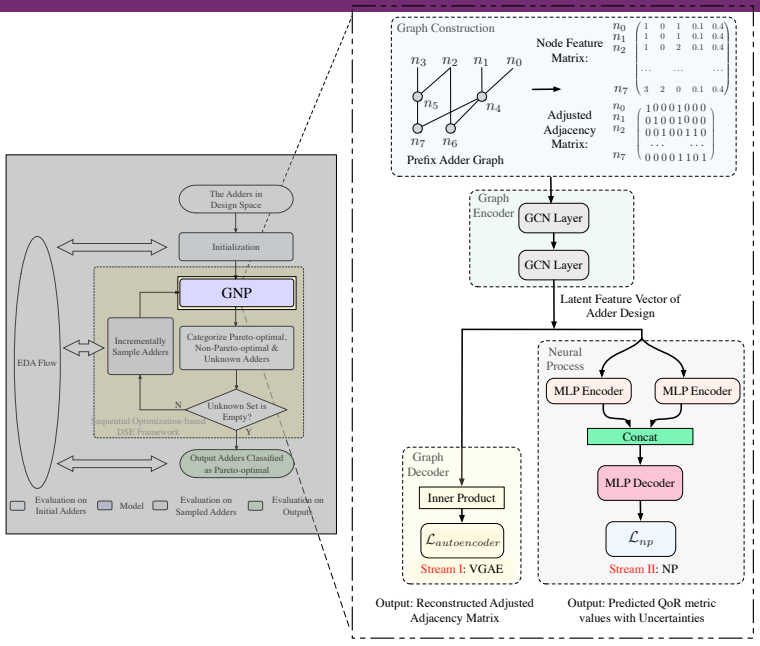
- A variational graph autoencoder is built to extract features from prefix adder structures automatically;
- A neural process is exploited to reduce computational complexity.

[3]H. Geng, Y. Ma, Q. Xu, J. Miao, S. Roy and B. Yu, "High-Speed Adder Design Space Exploration via Graph Neural Processes," TCAD'21.

Our solution: Overview (I) [TCAD'21]

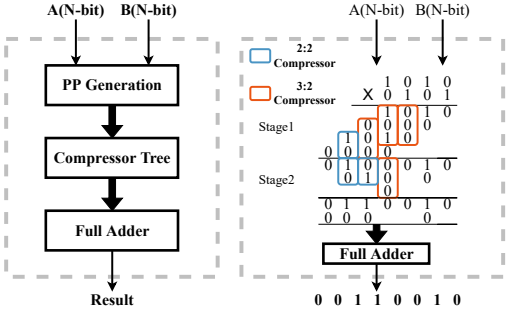


Our solution: Overview (II) [TCAD'21]



Multiplier Synthesis

- A partial product generator (PPG)
- A compressor tree (CT), which is the most critical part.
- A carry propagation adder

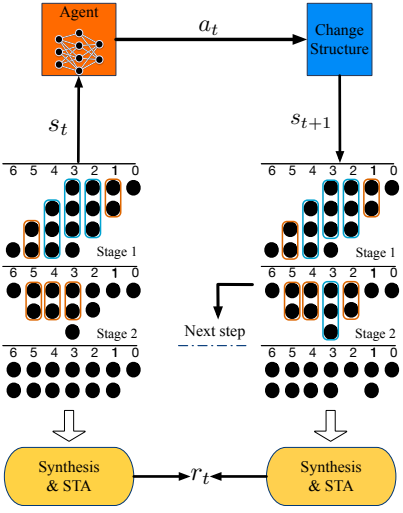


Multiplier architecture

RL in Multiplier Synthesis Optimization[DAC'23][4]

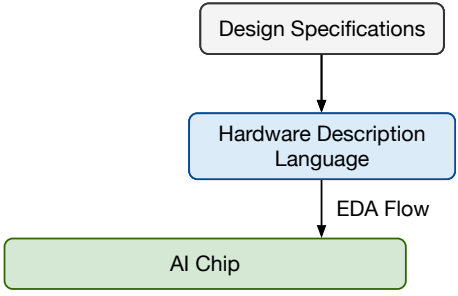
Deep Q-learning(DQN) based framework

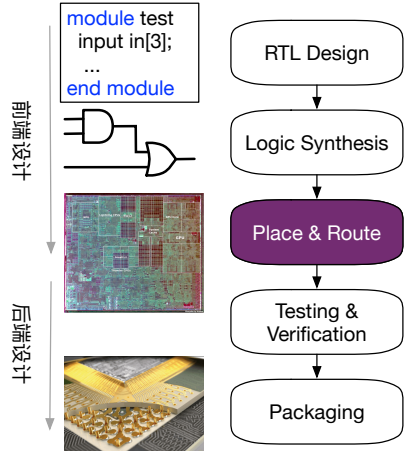
- ResNet-18 as the agent
- A state *s* refers to a structure.
- An action *a* refers to modification on current structure *s*
- Pareto-driven Reward



[4]D. Zuo, Y. Ouyang, Y. Ma "RL-MUL: Multiplier Design Optimization with Deep Reinforcement Learning," DAC'23.

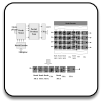
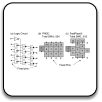
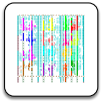
Datapath Driven Placement

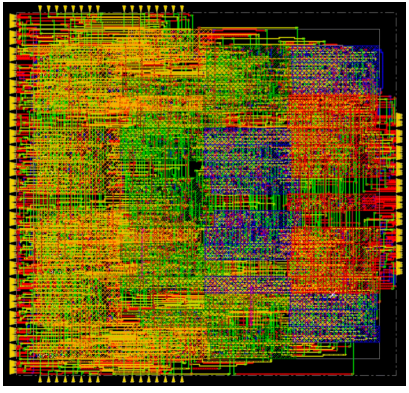

Design Specifications
Hardware Description Language
EDA Flow
AI Chip


module test
input in[3];
...
end module
前端设计
后端设计
RTL Design
Logic Synthesis
Place & Route
Testing & Verification
Packaging

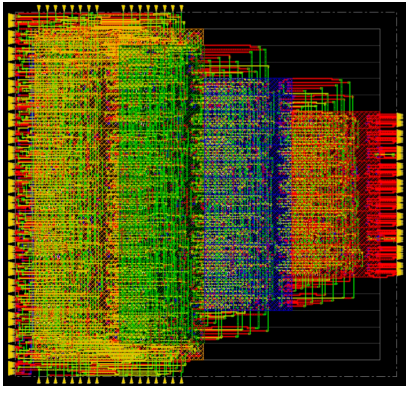
Dataflow Optimization and Physical Synthesis

Datapath Placement

APM ICCAD00	SAPT ISPD12	PADE DAC12	MFI ISPD13	FPGA PEs ISCAS19	ASIC PEs ISPD22
					



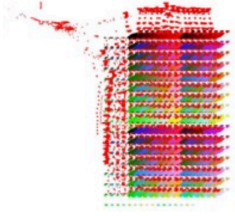
(a)



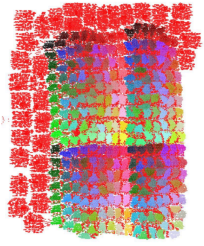
(b)

Dataflow Optimization and Physical Synthesis

- Dataflow optimization becomes essential for AI chips
  - Schedules operation by data availability
  - Exposes opportunity for parallelism and data reuse
- Datapath regularity gives rise to new physical synthesis approaches
  - Directly determines system performance!

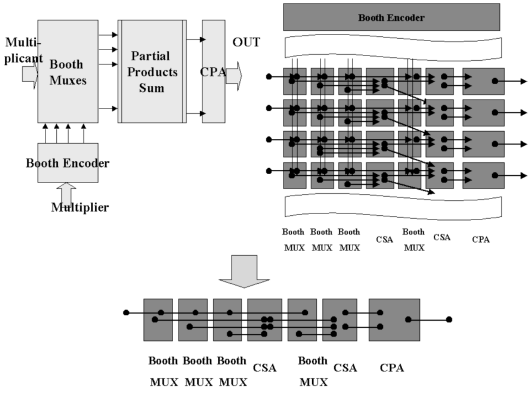


(a)Our flow 200 iteration



(b)Our flow detailed placement

- Datapath driven physical synthesis is not new!
- Placer has little control of exact locations if datapath is generated separately
- Abstract physical model
  - Bit-sliced abstraction
  - Compiled from HDL
  - Blocks are placed abutted
- Two-step heuristic for linear placement
  - quadratic assignment
  - sliding window optimization

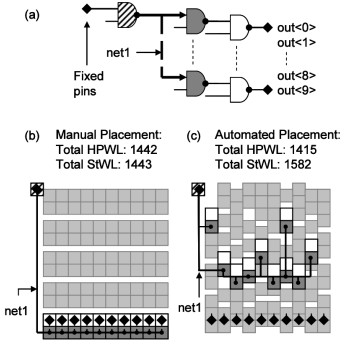


APM of a booth multiplier

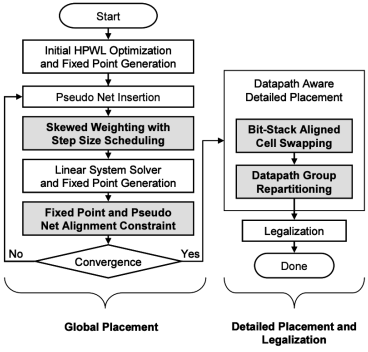
[1] Ye, T. Tao, and Giovanni De Micheli. "Data path placement with regularity", ICCAD'00.

Datapath Driven Placement: SAPT [ISPD'12][2]

- Structure aware placement obtains better steiner wire length → better routed wire length
- Apply bit-slice alignment constraints for force-directed placer
- Regularity driven detailed placement



Regularity matters

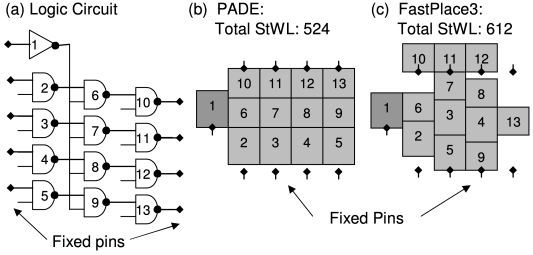


Structure aware placement

[2]Ward, Samuel I., et al. "Keep it straight: Teaching placement how to better handle designs with datapaths", ISPD'12.

ML Techniques for Datapath Driven Placement [DAC'12][3]

- Machine learning techniques are involved
  - Graph-based (e.g., automorphism) and physical features (e.g., cell area) are analyzed and extracted from the netlist
  - Features are fed to SVMs and NNs to classify and evaluate datapath patterns
  - Maximize the evaluation accuracies of datapath and non-datapath pattern
  - Proposed new placer: PADE

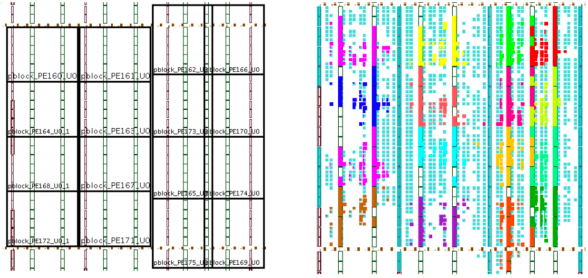


PADE effectively handles datapath in placement.

[3]Ward, Samuel, Duo Ding, and David Z. Pan. "PADE: A high-performance placer with automatic datapath extraction and evaluation through high dimensional data learning", DAC'12.

Datapath Driven Systolic Array Placement [ISCAS'19][4]

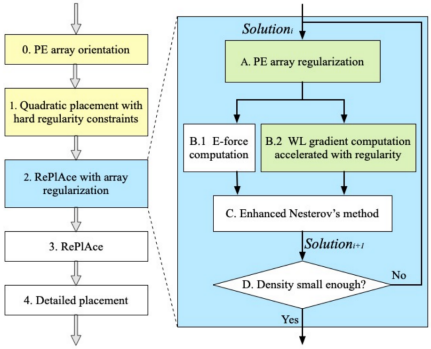
- Systolic arrays are a popular choice to support neural network computations
- Current FPGA CAD tools cannot synthesize them in high quality
- One solution: restrict fixed locations for PEs
  - Sufficient DSPs, close to used I/O banks



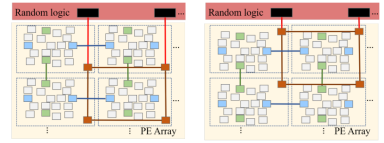
PE placement with floorplan constraints

[4] Zhang, Jiaxi, et al. "Frequency improvement of systolic array-based CNNs on FPGAs", ISCAS'19.

PE Array Placement with 2D Regularity [ISPD'22][5]

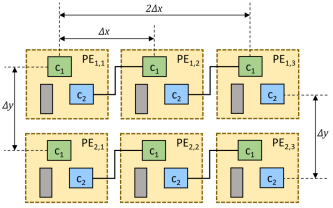


Placement Flow



(a) PE cells are placed in macro generation, wirelength: 264

(b) PE cells are placed along with random logic, wirelength: 244



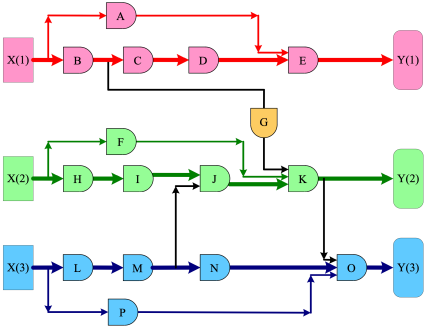
Regularity Constraints

- Simultaneously place the PEs and random logic cells for better solution quality
- Analytical global placement with PE array regularization

[5]Fang, Donghao, et al. "Global placement exploiting soft 2D regularity", ISPD'22.

Regularity Extraction: Network Flow Approach [ISPD'13][6]

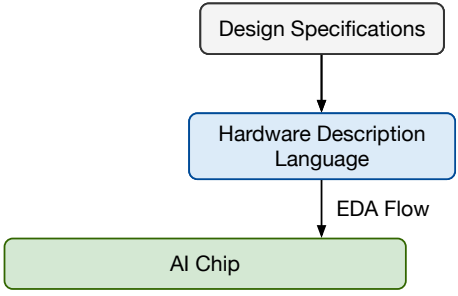
- *Datapath main frame* (DMF)
  - a set of n disjoint paths from input to output
  - maximize the number of datapath gates on these paths
- Can be optimally identified by the min-cost max-flow algorithm

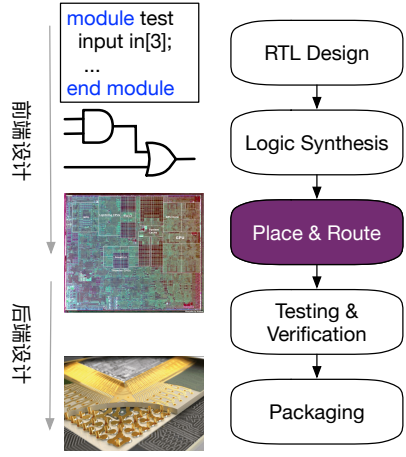


DMF identification can be transformed to a network flow problem

[6]Xiang, Hua, et al. "Network flow based datapath bit slicing", ISPD'13.

Wafer-Scale Floorplan

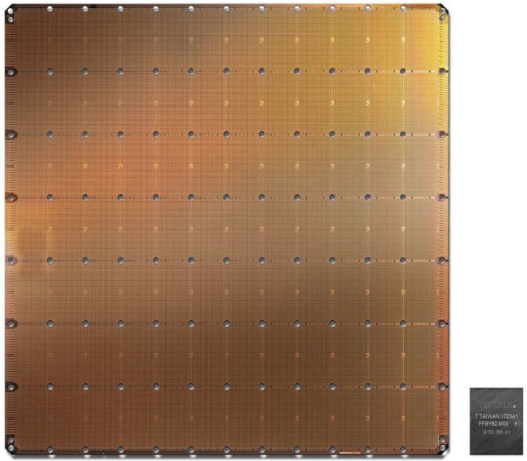

Design Specifications
Hardware Description Language
EDA Flow
AI Chip


module test
input in[3];
...
end module
前端设计
后端设计
RTL Design
Logic Synthesis
Place & Route
Testing & Verification
Packaging

Wafer-Scale Engine (WSE) [SC'19][7]

The Cerebras CS-1's Wafer-Scale Engine (WSE) is the largest and most powerful processor ever built.

- Consisting of 800 by 1060 identical processing elements.
- Keep all memory and all computation together on a single monolithic chip.

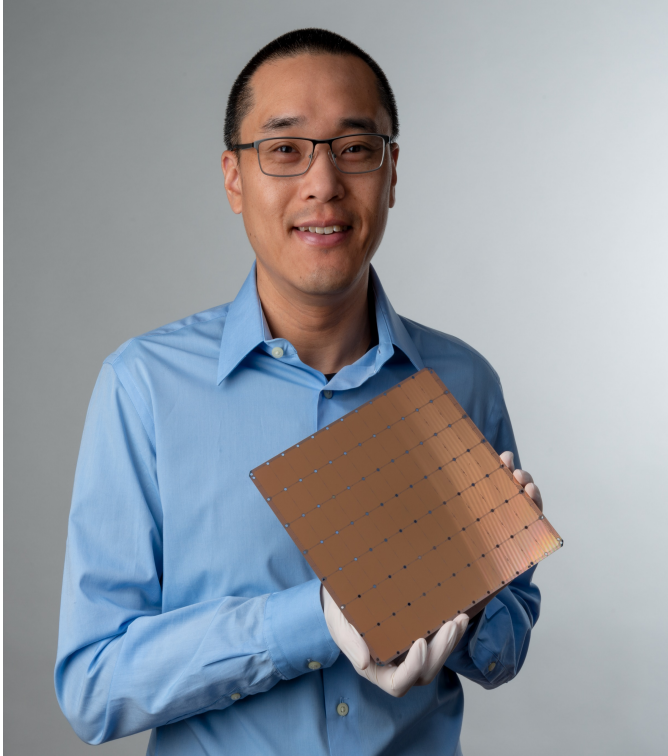


Die photograph of the Cerebras Wafer-Scale Engine (WSE; at left). For comparison, the largest GPU is shown on the right to scale.

[7] J.P. Fricker, A. Hock, "Building a Wafer-Scale Deep Learning System: Lessons Learned", Supercomputing'19.

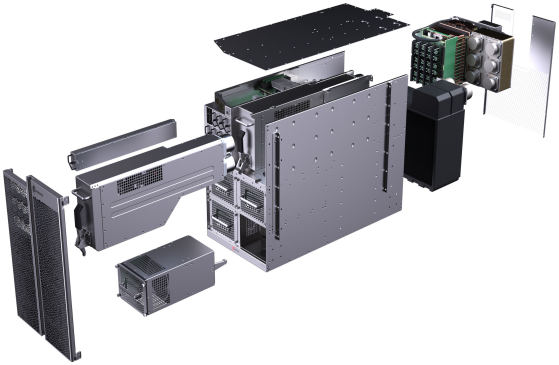
Cerebras: Largest Chip Ever Built

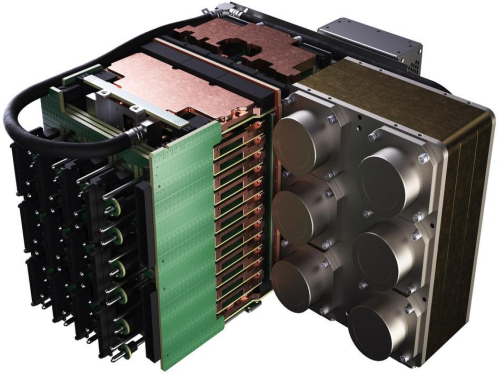
- 46,225 mm^2 silicon
- 2.6 **Trillion** transistors, 7nm TSMC
- 850,000 processor cores
- 40 Gigabytes of On-chip Memory
- 9 PByte/s memory bandwidth
- 100 Pbit/s fabric bandwidth



cerebras

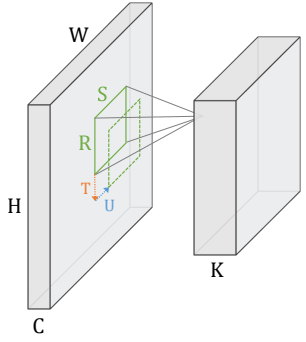
CS-1 Supercomputer Hardware



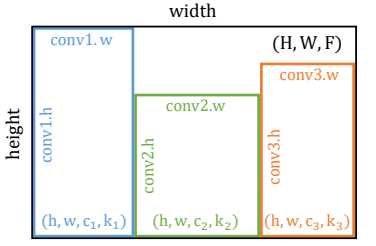


Kernel Definition

- *conv*: basic convolution kernel



(a) Arguments of *conv*



$\text{kernel} = (\text{TP}, \text{H}, \text{W}, \text{F}; \text{h}, \text{w}, \text{c}_1, \text{c}_2, \text{c}_3, \text{k}_1, \text{k}_2, \text{k}_3)$

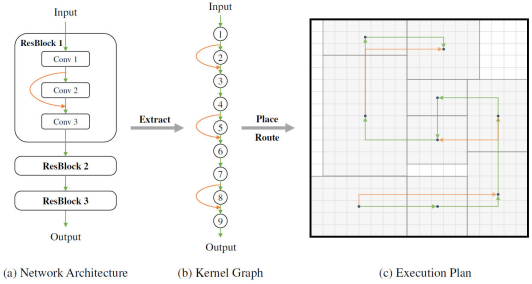
$\text{time} = \max_{1 \le i \le 3} \text{conv}_i.\text{time} \quad \text{mem} = \max_{1 \le i \le 3} \text{conv}_i.\text{mem}$

(b) Performance of a kernel with 3 *convs*

- 8 formal arguments: (H, W, R, S, C, K, T, U) $\Rightarrow$ fixed input parameters.
- 4 execution arguments: (h, w, c, k) $\Rightarrow$ variables to be determined.

Problem Formulation

- Determine the execution parameters and the locations for all kernels. [ISPD'20][8]
  - Blocks are soft: kernel sizing
  - Floorplanning
  - Optimize performance, wirelength, etc.

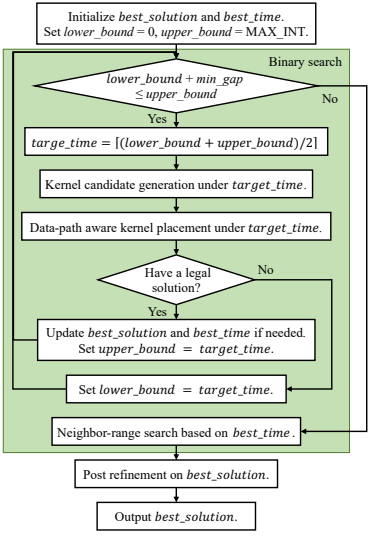


Kernel floorplanning. Figure adopted from [ICCAD'20][9]

[8] Michael James, et.al., "Physical Mapping of Neural Networks on a Wafer-Scale Deep Learning Accelerator", ISPD'20.

[9] Bentian Jiang, et.al., CU.POKer: Placing DNNs onWafer-Scale AI Accelerator with Optimal Kernel Sizing, ICCAD'20.

CU.POKer [ICCAD'20][9]



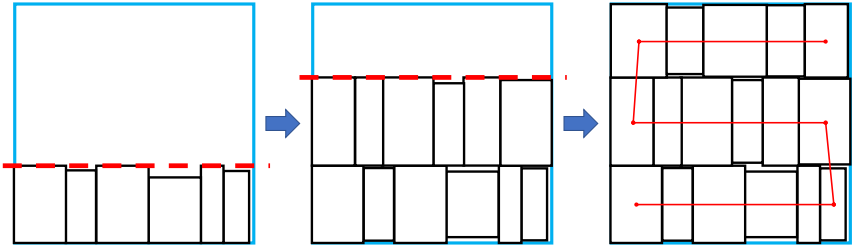
Two-steps Search

- Binary search
- Neighbor-range search
- Post refinement

Searching under Target Time

- Kernel candidates generation with optimal shapes under given target time
- Data-path aware placement

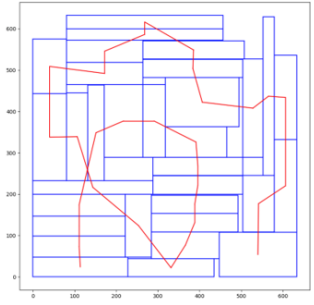
[9]Bentian Jiang, et.al., CU.POKer: Placing DNNs onWafer-Scale AI Accelerator with Optimal Kernel Sizing, ICCAD'20.



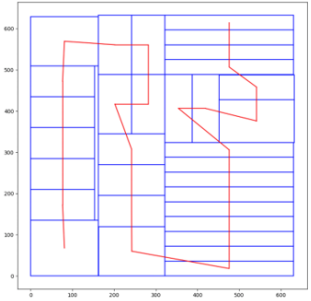
Overall Flow

- Given a target time T, generate all the kernel candidates with optimal shapes and execution times under T.
- According to the connectivity graph, generate the topological order of the kernels for placement.
- Place the kernels compactly row by row in the topological order.

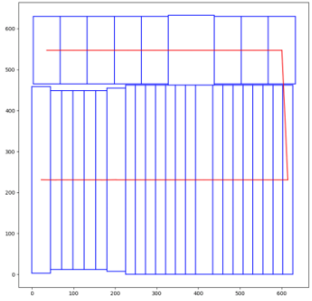
Comparisons with Conventional Floorplanning Heuristics



Simulated Annealing Placer:
Max_time: 76698
Wire_length: 3237
Adapter_cost: 15
Score: 110478



Divide and Conquer Placer:
Max_time: 65106
Wire_length: 2650.5
Adapter_cost: 18
Score: 93321

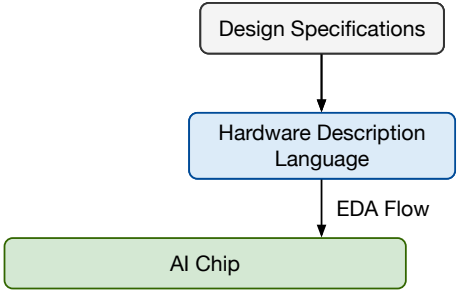


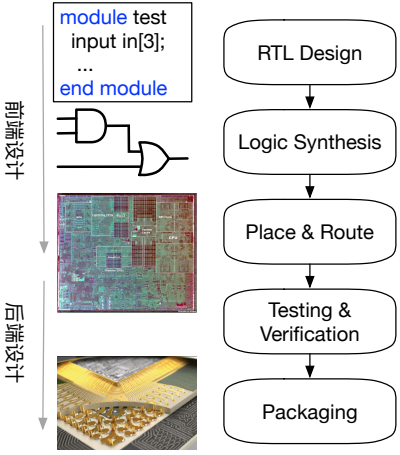
CU.POKer:
Max_time: 65170
Wire_length: 1489.5
Adapter_cost: 12
Score: 81265

Performance comparisons with SA and DC placers on 8 public benchmarks.

Conclusion

Conclusion





- Arithmetic Unit Synthesis
- Datapath Driven Placement
- Wafer-Scale Floorplan

THANK YOU!