Algorithms and Hardware for AI Accelerated Discoveries

Deming Chen, Song Han, Philip Harris, Scott Hauck, **Pan Li**, Dylan Rankin

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Massachusetts Institute of Technology

UNIVERSITY of WASHINGTON

PURDUE UNIVERSITY

Roadmap

Overview (16 min)

Pan: Graph NNs (8 min)

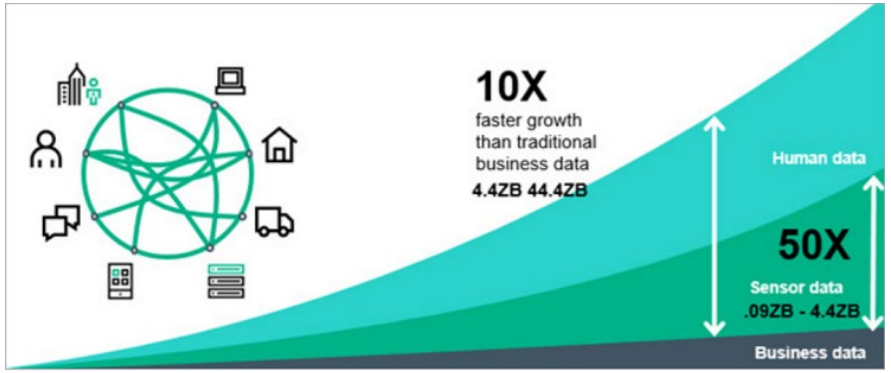
Dylan: HLS4ML (8 min)

Deming: PyLog, ScaleHLS (8 min)

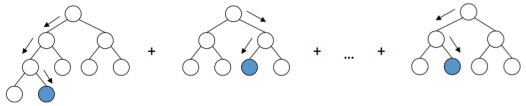
Song: PVCNN, SPVNAS, PointAcc (8 min)

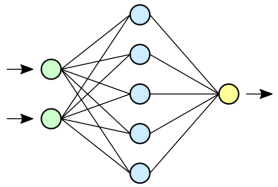
Recent boost of AI techniques

Massive Data



[The Exponential Growth of Data]



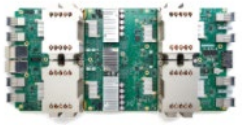


AI Algorithms

Models

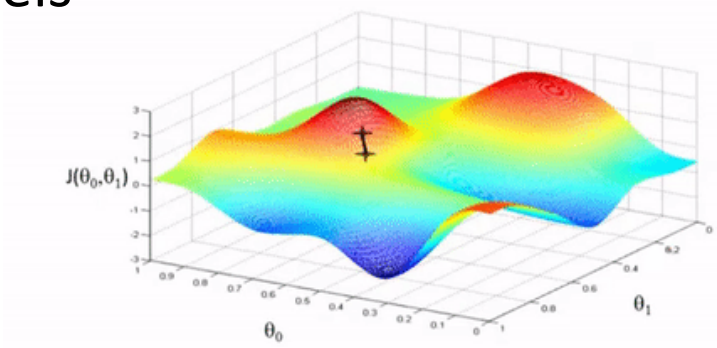
Computing hardware

FPGA



TPU

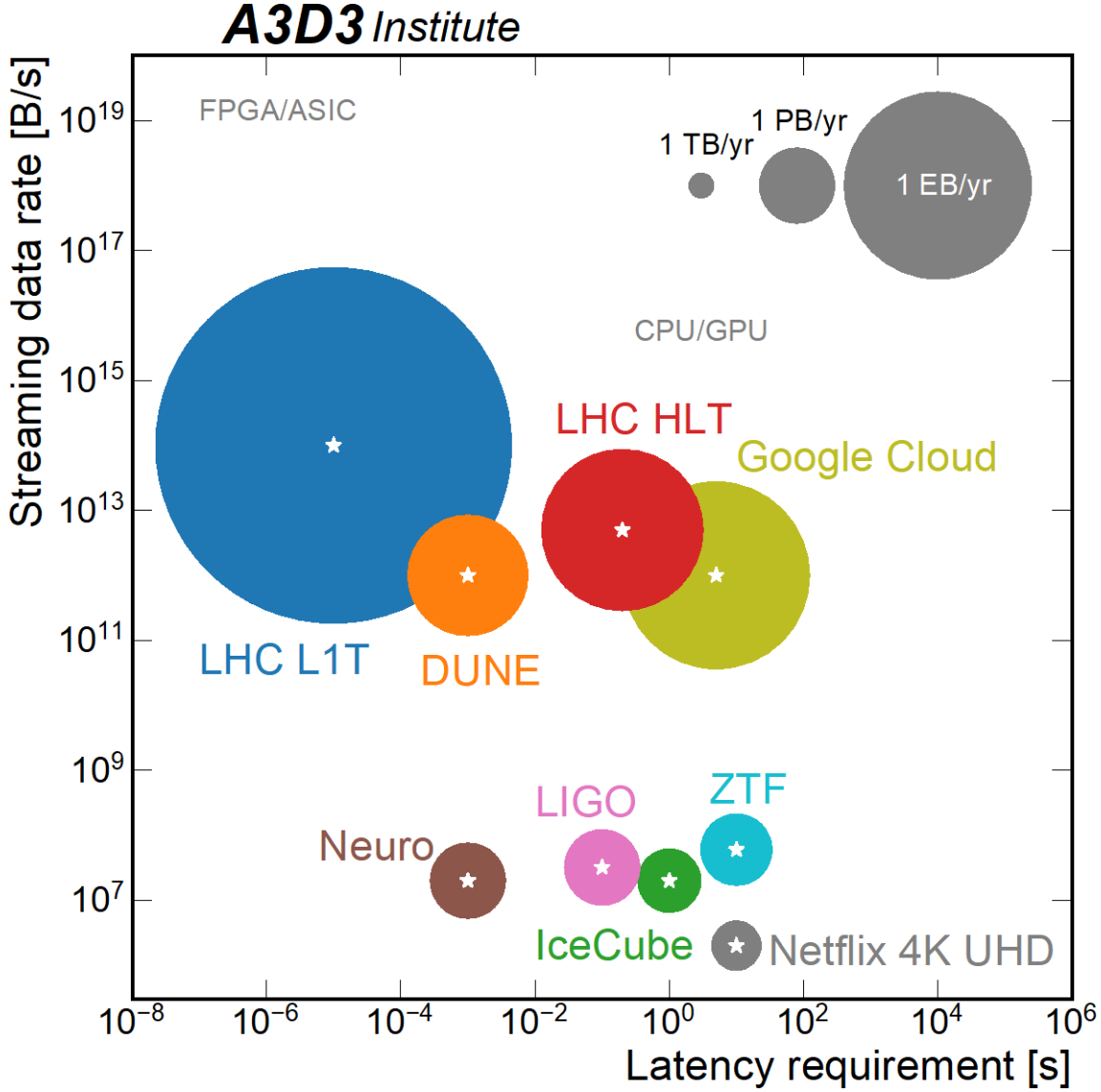
GPU



Optimization

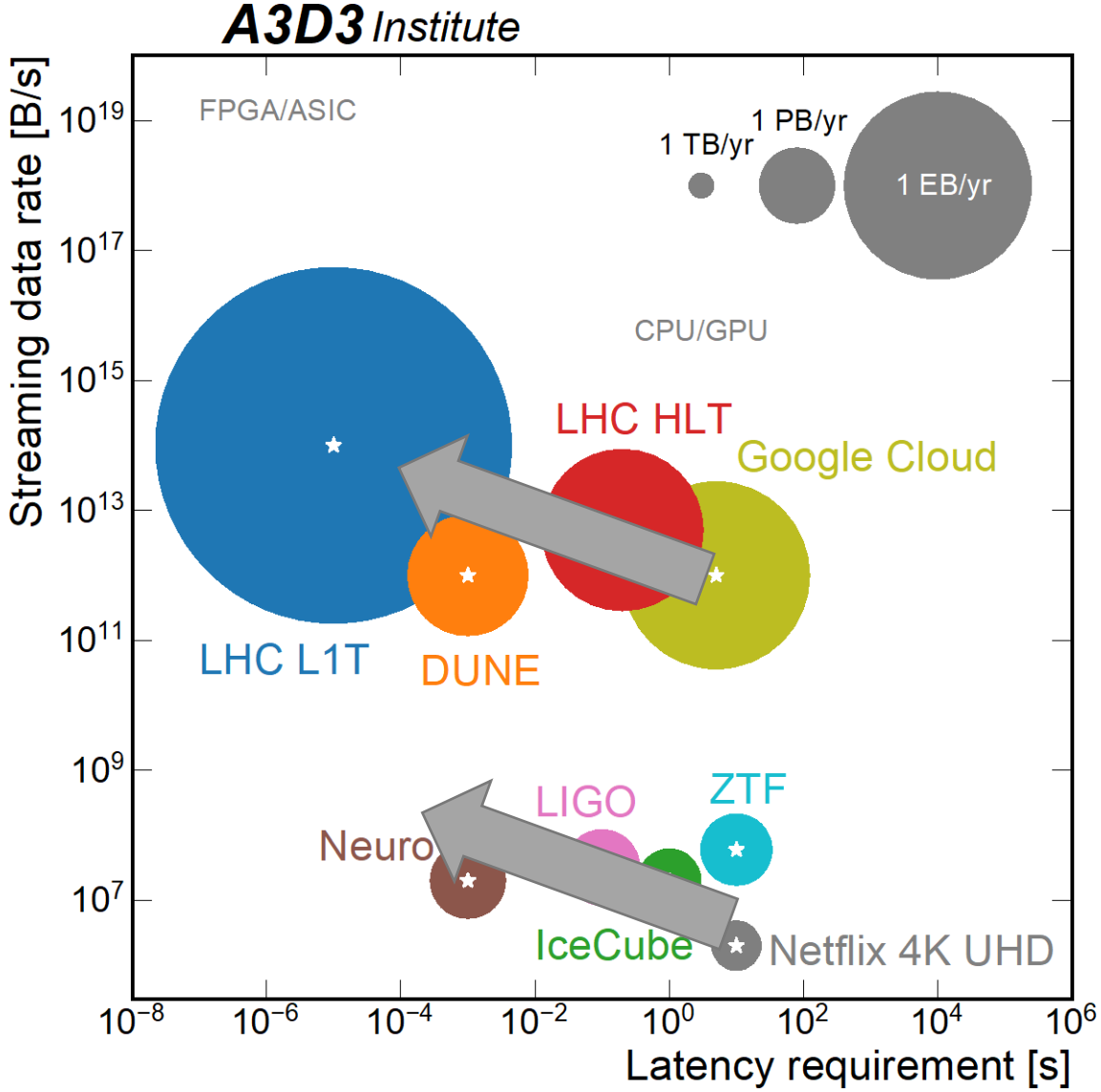
For scientific applications…

- Large-scale scientific data introduce *new challenges*.

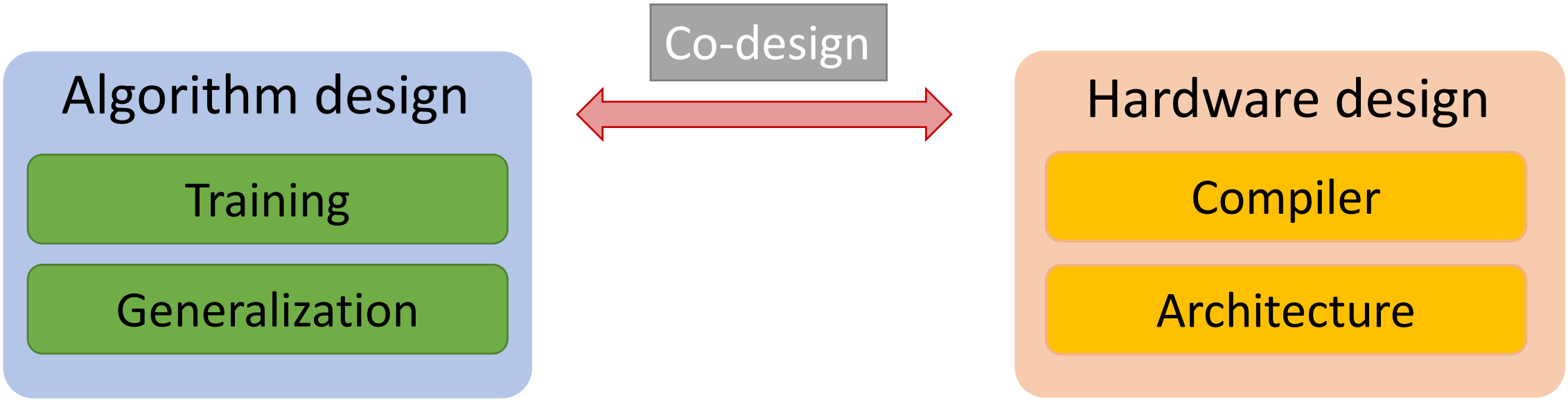


For scientific applications…

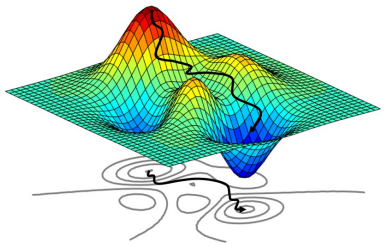
- Large-scale scientific data introduce *new challenges*.
- Our goal: Prototype new *algorithms and hardware* for domain scientists to deal with such new challenges.

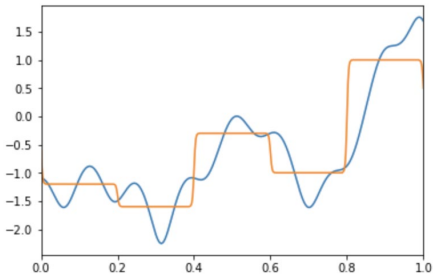


AI Design Modules



AI Design Modules





f: Target func.

f_θ: The model

The key question in training:

How to learn a model f_θ that can *well and efficiently* approximate the target function f?

Co-design

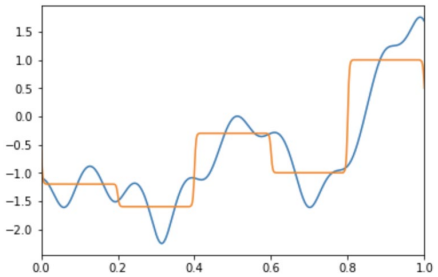
Algorithm design

- Training
- Generalization

Hardware design

- Compiler
- Architecture

Capacity



f: Target func.

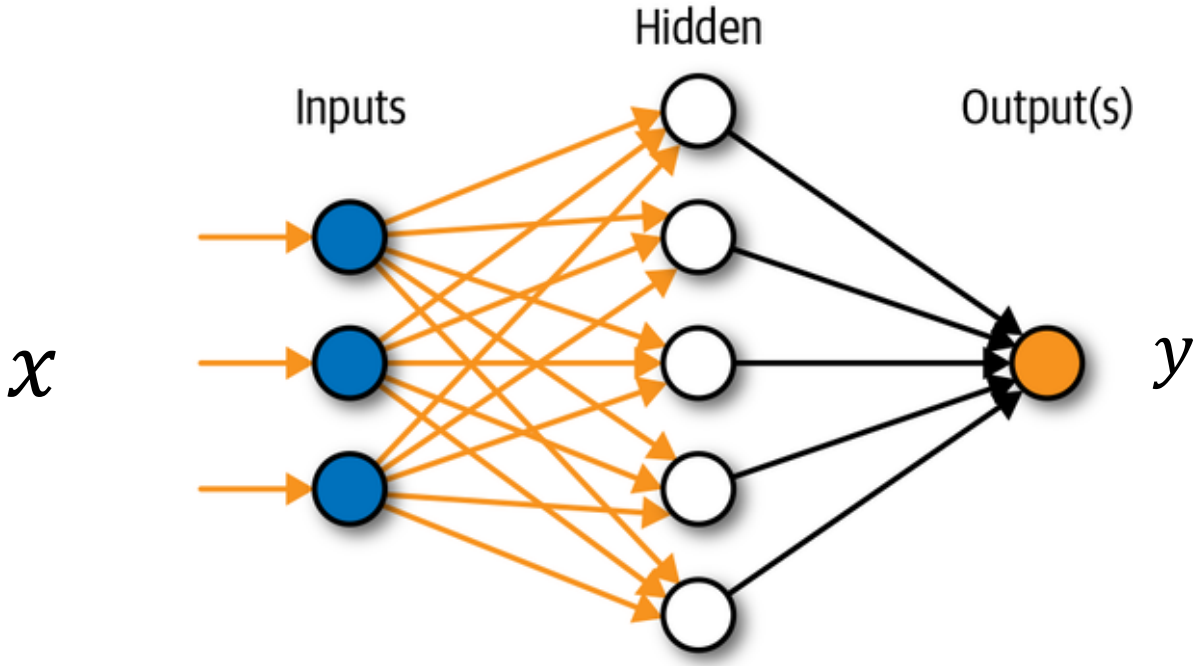
f_θ: The model

The key question in training:
How to learn a model that can *well and efficiently* approximate the target function f?

- Can f be well approximated by f_θ with a given number of parameters θ?
- The model f_θ with a small number of parameters θ indicates good generalization, low memory cost and maybe low latency as well.

Capacity

Neural Networks have extremely large capacity!



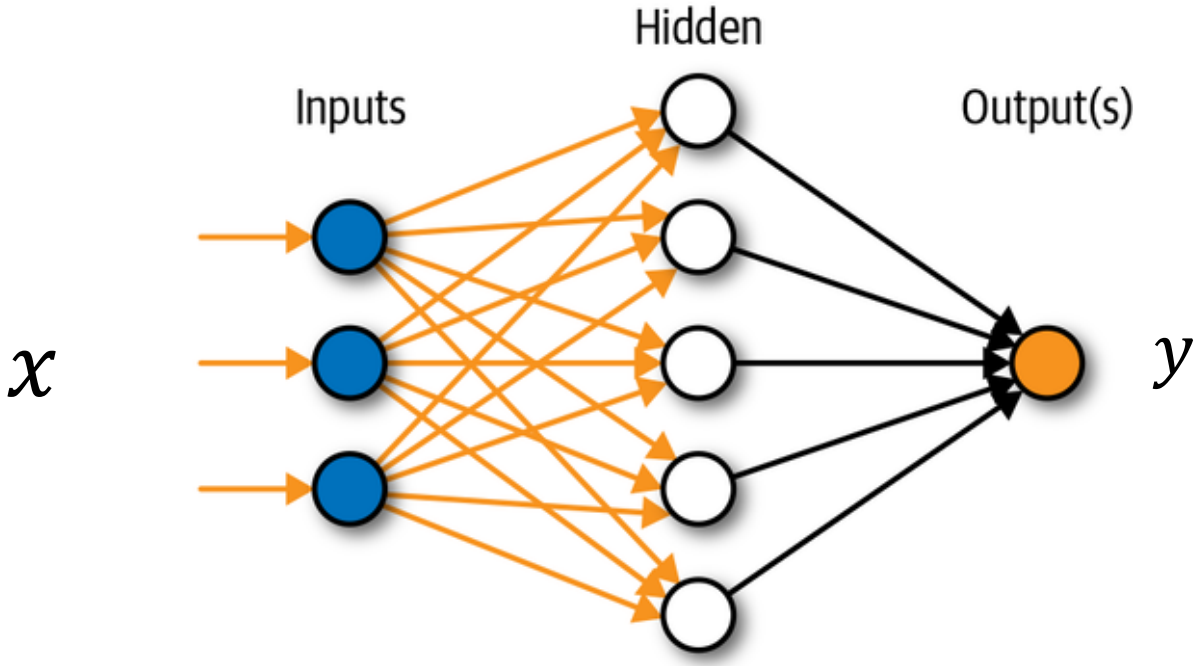
- Shallow NNs approximate continuous functions [Cybenko (1989), Hornik et al (1989)].
- Deep NNs better approximate highly-differentiable functions [Yarotsky (2017)].

$$x \longrightarrow Wx + b \longrightarrow \sigma(Wx + b) \longrightarrow W'\sigma(Wx + b)$$

$\sigma(\cdot)$ is a entry-wise nonlinear function, e.g. $\sigma(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$

Capacity

Neural Networks have extremely large capacity!

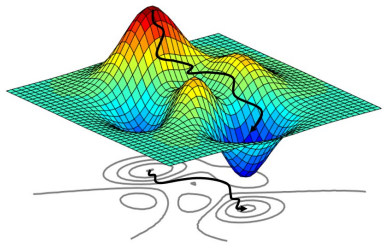


- Shallow NNs approximate continuous functions [Cybenko (1989), Hornik et al (1989)].
- Deep NNs better approximate highly-differentiable functions [Yarotsky (2017)].
- Graph (point cloud) NNs with finite neurons have not been well studied yet…

$$x \longrightarrow Wx + b \longrightarrow \sigma(Wx + b) \longrightarrow W'\sigma(Wx + b)$$

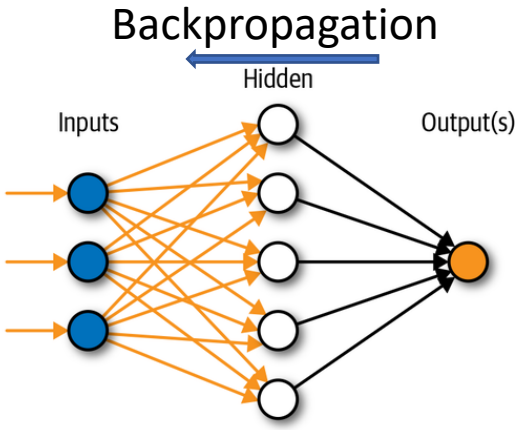
$\sigma(\cdot)$ is a entry-wise nonlinear function, e.g. $\sigma(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$

Optimization



The key question in training:
How to learn a model that can *well and efficiently* fit the data we have observed?

- Can we fast compute the parameter θ such that f_θ well approximate f?



Rumelhart et al [1988]

AI Design Modules

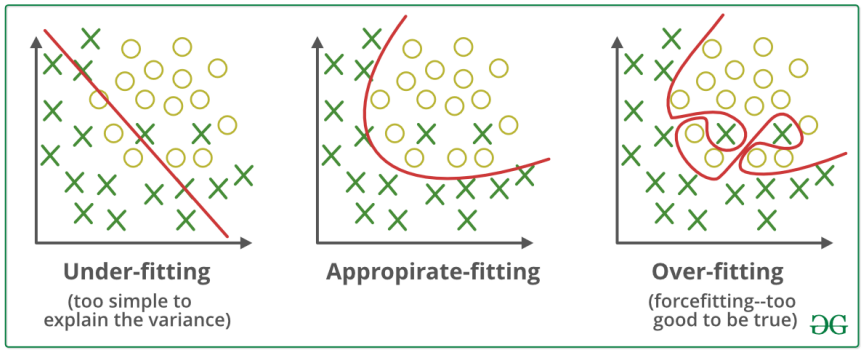
Co-design

Algorithm design

- Training
- Generalization

Hardware design

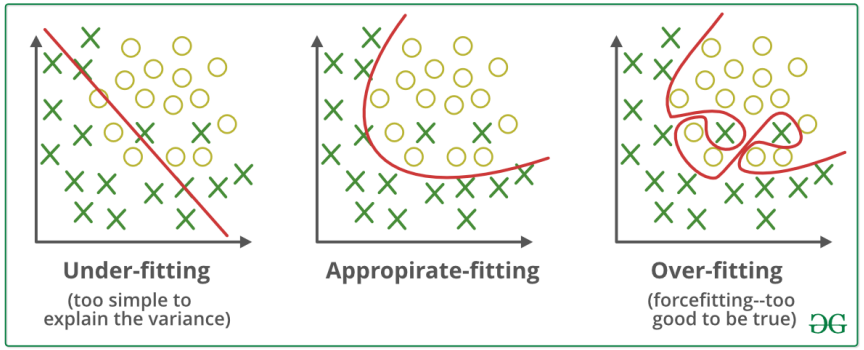
- Compiler
- Architecture



The key question in generalization:

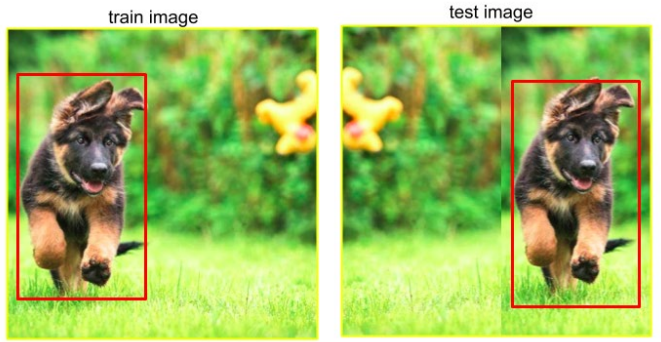
How to guarantee the model *capture the law that can apply to the data that has not been observed*?

Generalization



The key question in generalization:
How to guarantee the model *capture the law that can apply to the data that has not been observed*?

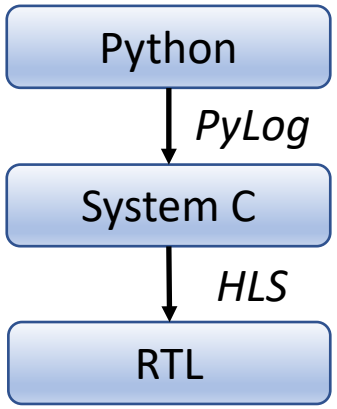
- Data-driven neural network models: Convolutional NNs (CNNs), Graph NNs…
- For A3D3, inject scientific data insights into the model design

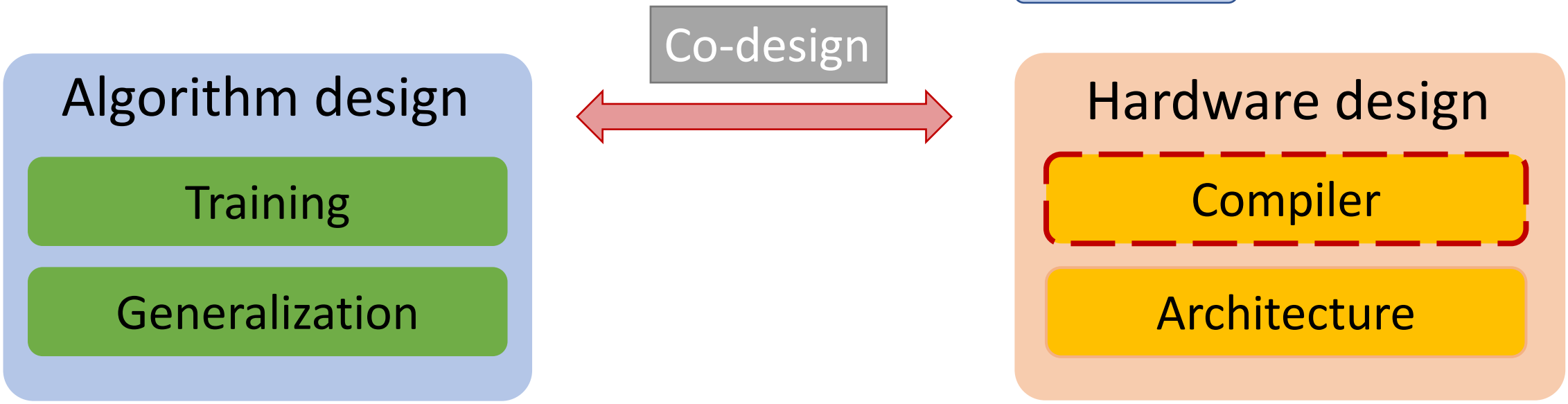


Translation Invariance

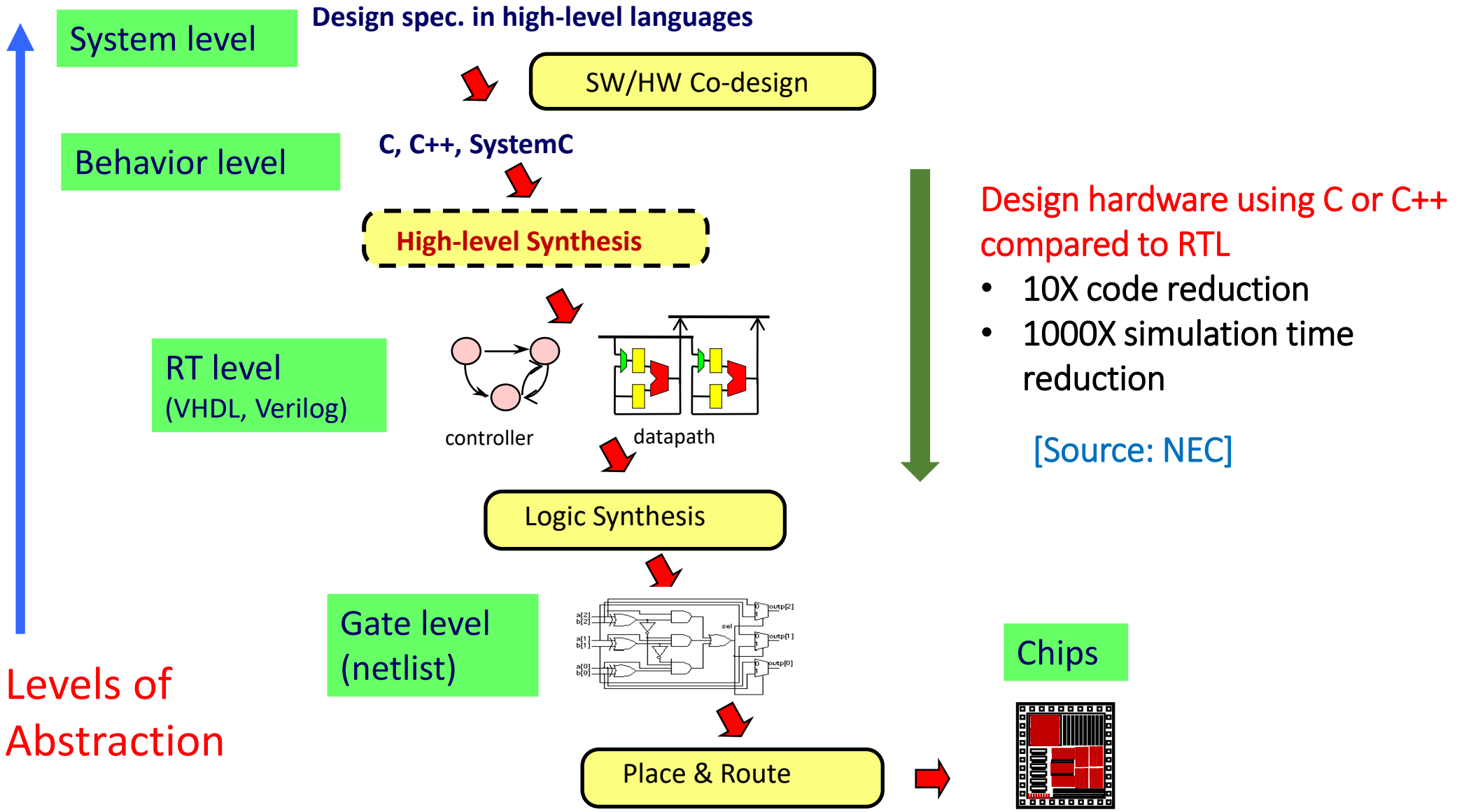
AI Design Modules

The key question in complier design:
How to automatically and efficiently map an algorithm f_θ onto a hardware platform?

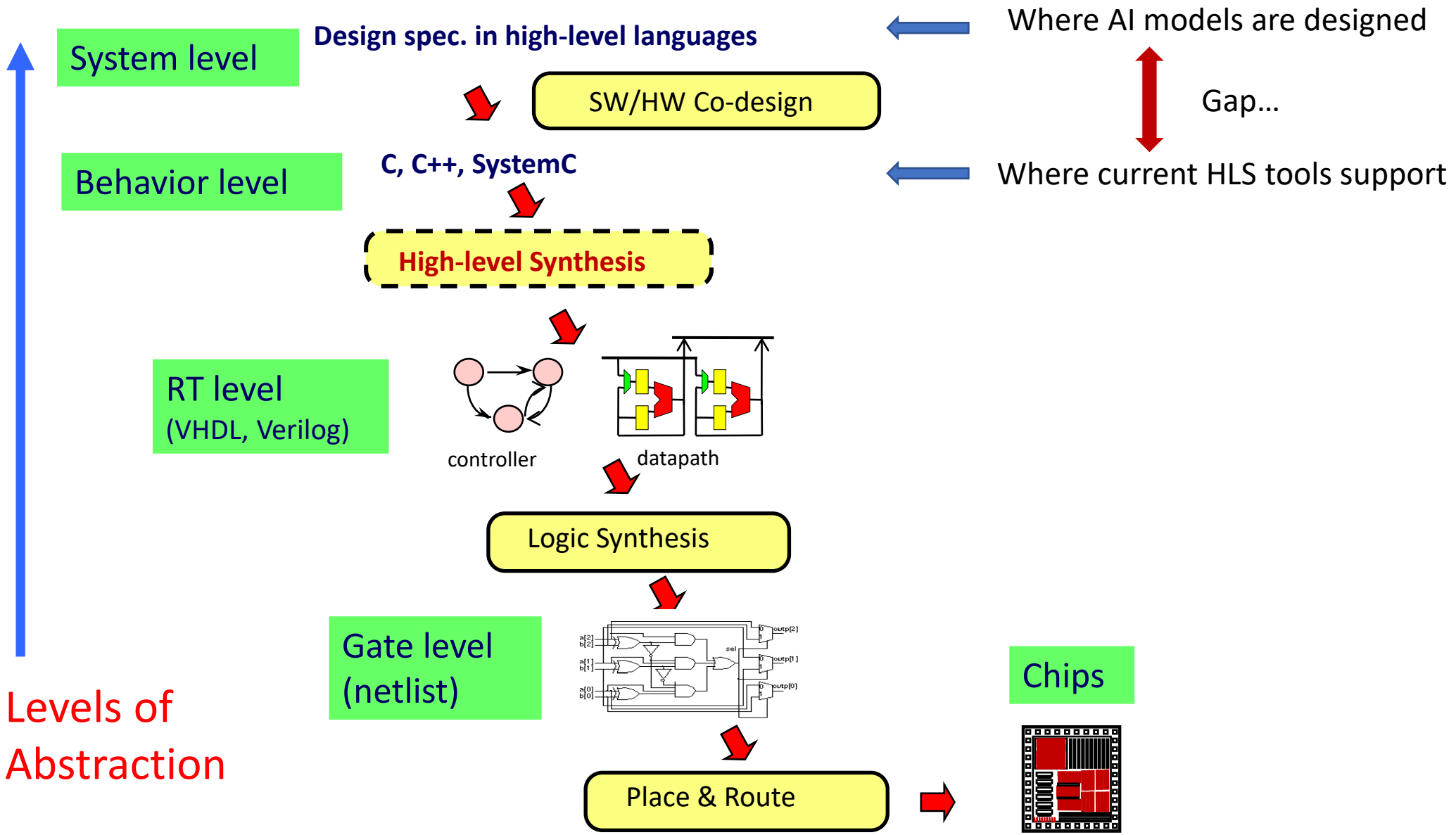




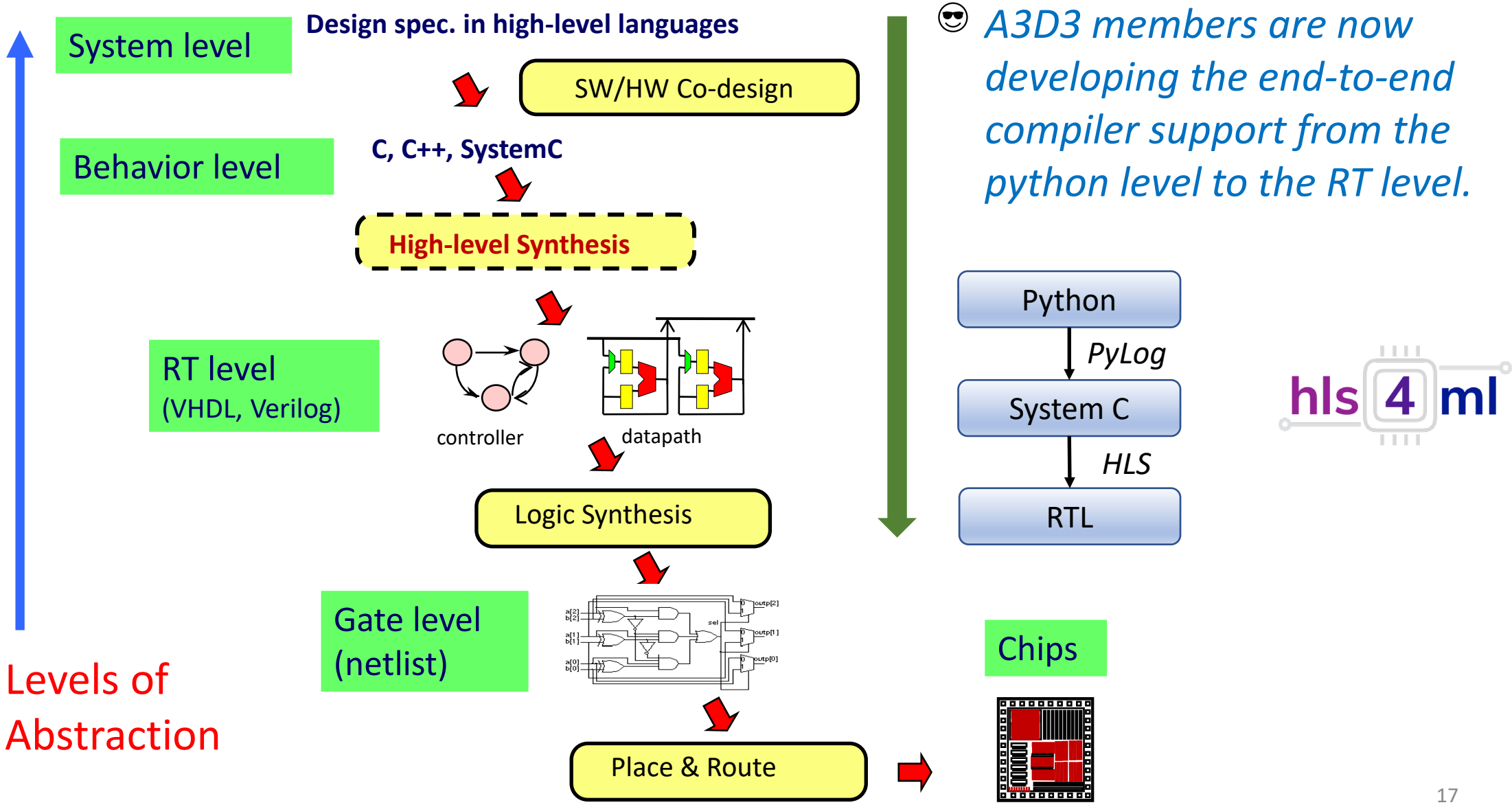
Complier --- High-Level Synthesis (HLS)



Complier --- High-Level Synthesis (HLS)



Complier --- High-Level Synthesis (HLS)



A3D3 members are now developing the end-to-end compiler support from the python level to the RT level.

hls 4 ml

AI Design Modules

Co-design

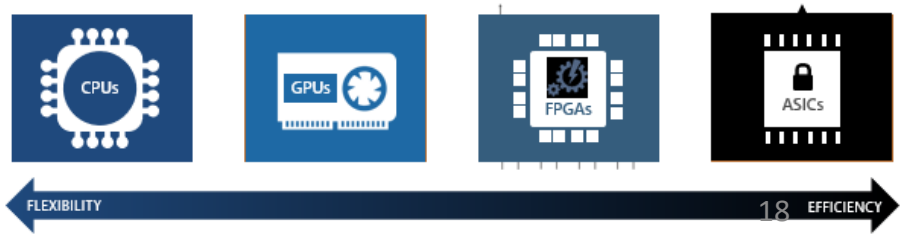
Algorithm design

- Training
- Generalization

Hardware design

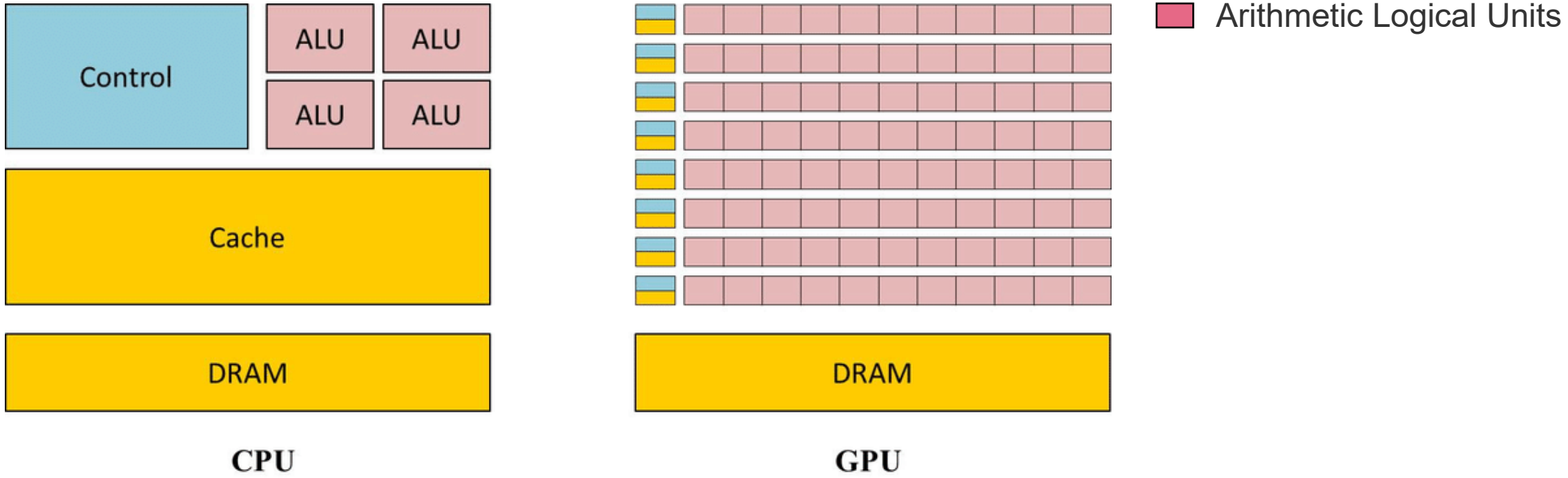
- Compiler
- Architecture

The key question in architecture design:
How to design, allocate and arrange computation and communication components for the algorithm f_θ?



Architecture Design

- GPU outperforms CPU as for the many more ALUs to support parallel computing.

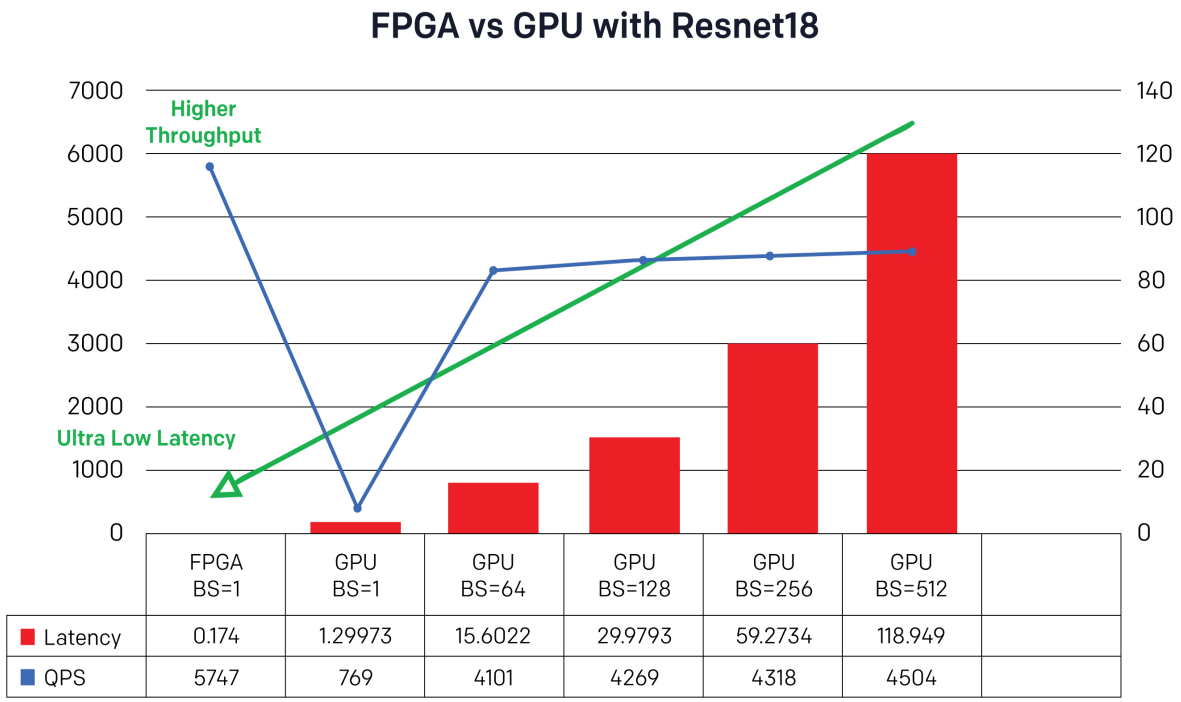


[Cuda C-programming guide 2014]

Architecture Design

- FPGA allows dedicated data flow control and further enhances parallelism.

Throughput = Batch-Size (BS) / Latency



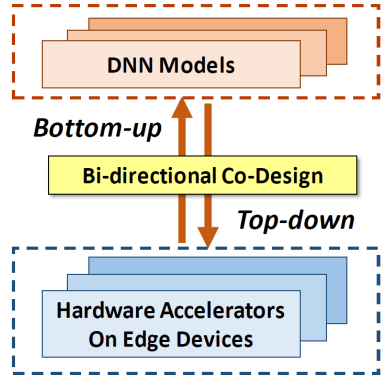
	FPGA BS=1	GPU BS=1	GPU BS=64	GPU BS=128	GPU BS=256	GPU BS=512	
Latency	0.174	1.29973	15.6022	29.9793	59.2734	118.949	
QPS	5747	769	4101	4269	4318	4504	

[Alibaba Clouder]

AI Design Modules

The key question in co-design:

How to jointly design an AI algorithm with its hardware implementation?



[Hao, et al., 2019]

Co-design

Algorithm design

- Training
- Generalization

Hardware design

- Compiler
- Architecture

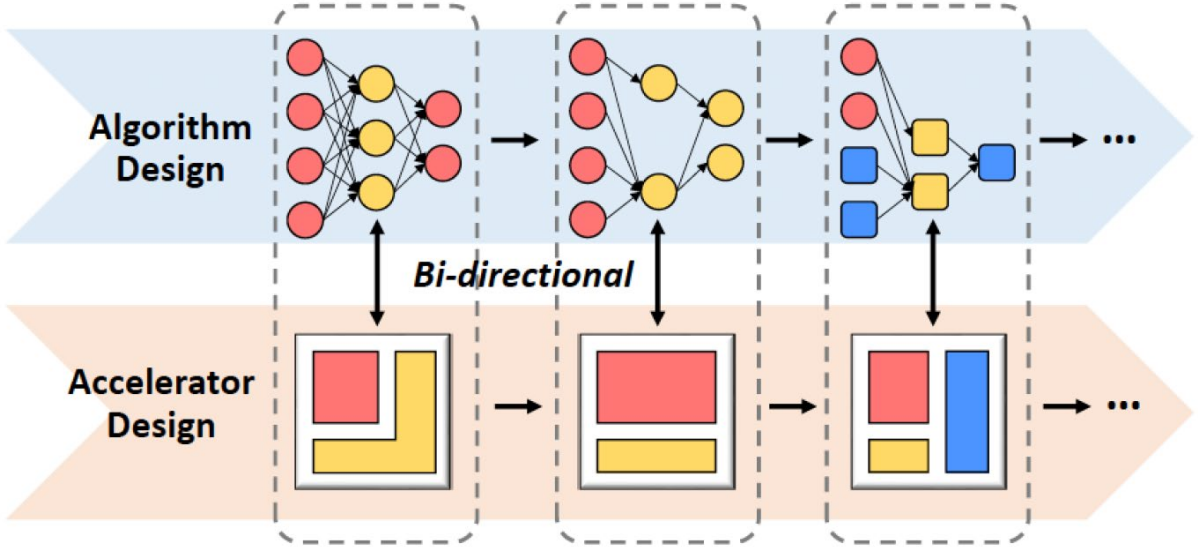
Co-design

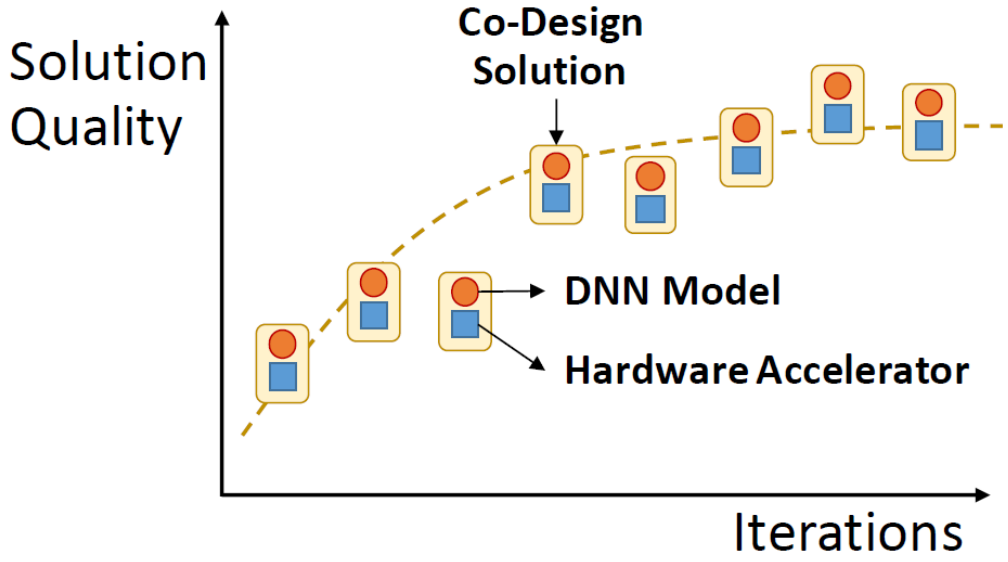
- *Top-down*: evaluate neural networks through architecture template mapping

  --- traditional hardware design for AI models

- *Bottom-up*: architecture template guided neural network model search

  --- Co-design considers both directions





[Hao, et al., 2019]

Co-design

One example of co-design via differentiable neural architecture search:

- One direction: Optimize the model architecture A while keep its hardware implement I fixed.

$$\min_{A} \; Loss_{pred}(A) * Loss_{hw}(I)$$

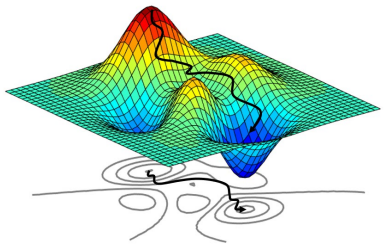
- Bi-direction: Optimize both the model architecture A and its hardware implement I

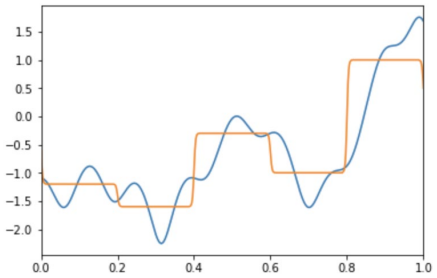
$$\min_{A,I} \; Loss_{pred}(A) * Loss_{hw}(I) + Penalty\ (I)$$

The penalty on the implementation according to its hardware resource consumption

[Hao, et al., 2019] [Cai, et al., 2019]

AI Design Modules





f: Target func.

f_θ: Learnable func.

Python → *PyLog* → System C → *HLS* → RTL

hls 4 ml

Algorithm design

- Training
- Generalization

Co-design

Hardware design

- Compiler
- Architecture