Building Energy Efficient Computers

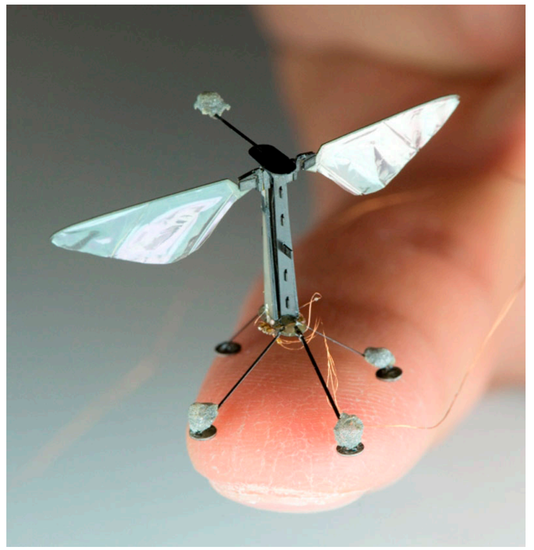
CSHL Seminar

Kyle Daruwalla (9 Feb. 2021)

Talk summary

- Most existing computer performance trends are dominated by transistor technology scaling
  - Slowing down and expected to end in the near term

Talk summary

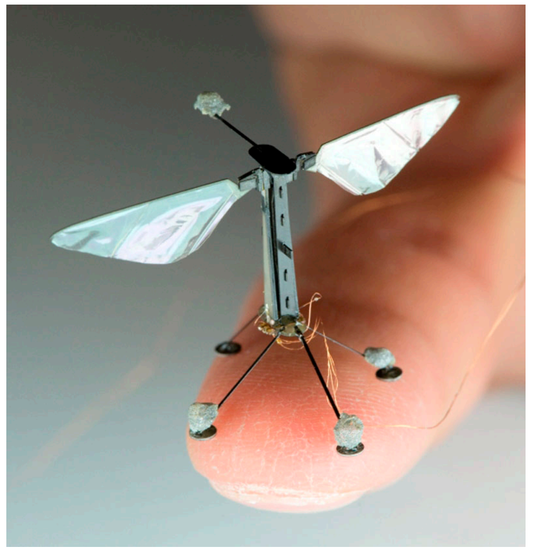


[1]

- Most existing computer performance trends are dominated by transistor technology scaling
  - Slowing down and expected to end in the near term
- We need to build computers that do not rely on technology scaling
  - Continue to scale performance in the future
  - Create energy-efficient computers today

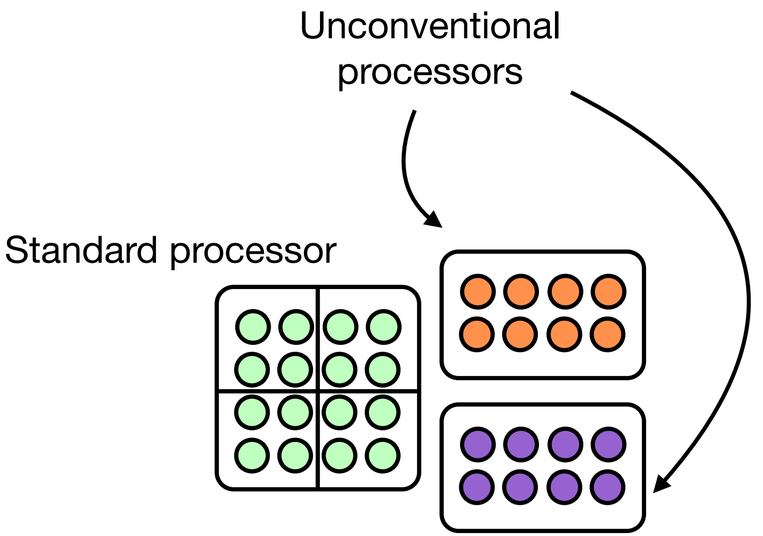
[1] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood, “Controlled Flight of a Biologically Inspired, Insect-Scale Robot,” *Science*, vol. 340, no. 6132, pp. 603–607, May 2013, doi: 10.1126/science.1231806.

Talk summary



[1]

- Most existing computer performance trends are dominated by transistor technology scaling
  - Slowing down and expected to end in the near term
- We need to build computers that do not rely on technology scaling
  - Continue to scale performance in the future
  - Create energy-efficient computers today
- My work focuses on two unconventional computing paradigms
  (*bitstream computing* and *neuromorphic computing*)

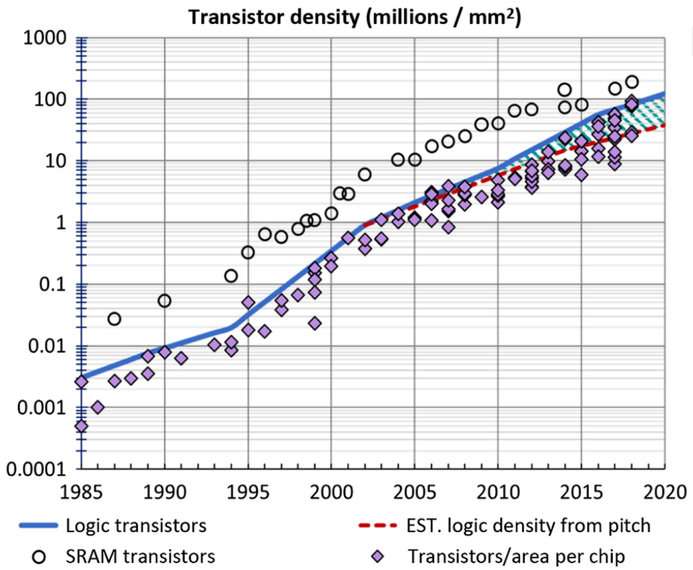


[1] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood, “Controlled Flight of a Biologically Inspired, Insect-Scale Robot,” *Science*, vol. 340, no. 6132, pp. 603–607, May 2013, doi: 10.1126/science.1231806.

Motivation

Quick history of computing

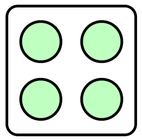
Quick history of computing



[1]

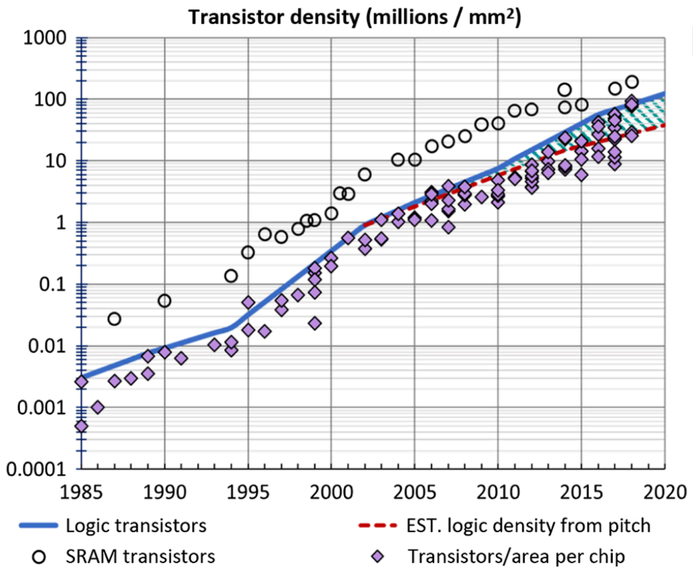
Moore's Law promises 2x transistors each generation!

1960s



[1] M. L. Rieger, "Retrospective on VLSI value scaling and lithography," *J. Micro/Nanolith. MEMS MOEMS*, vol. 18, no. 04, p. 1, Nov. 2019, doi: 10.1117/1.JMM.18.4.040902.
[2] A. M. Rahmani, P. Liljeberg, A. Hemani, A. Jantsch, and H. Tenhunen, Eds., *The Dark Side of Silicon*. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-31596-6.

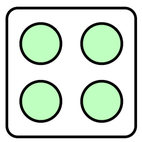
Quick history of computing

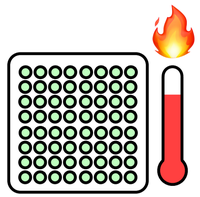


[1]

1960s

Moore's Law promises 2x transistors each generation!

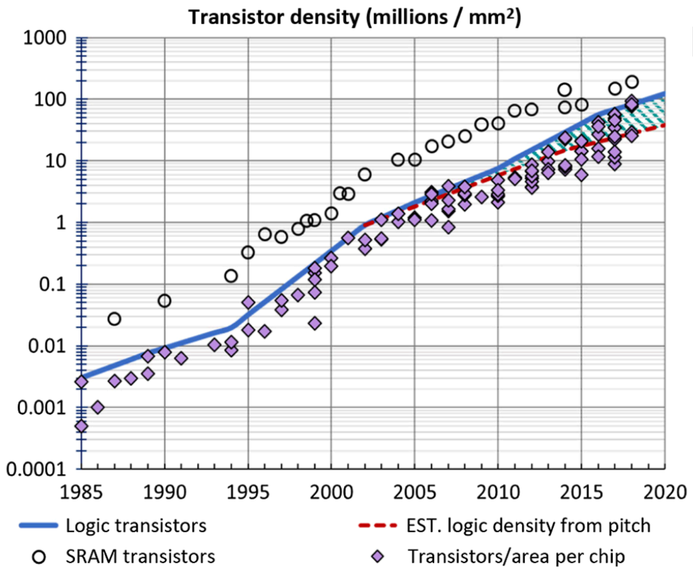




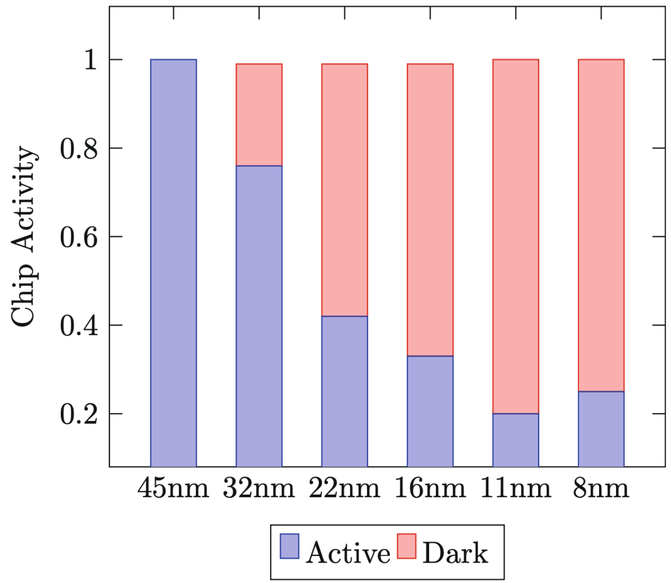
[1] M. L. Rieger, "Retrospective on VLSI value scaling and lithography," *J. Micro/Nanolith. MEMS MOEMS*, vol. 18, no. 04, p. 1, Nov. 2019, doi: 10.1117/1.JMM.18.4.040902.

[2] A. M. Rahmani, P. Liljeberg, A. Hemani, A. Jantsch, and H. Tenhunen, Eds., *The Dark Side of Silicon*. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-31596-6.

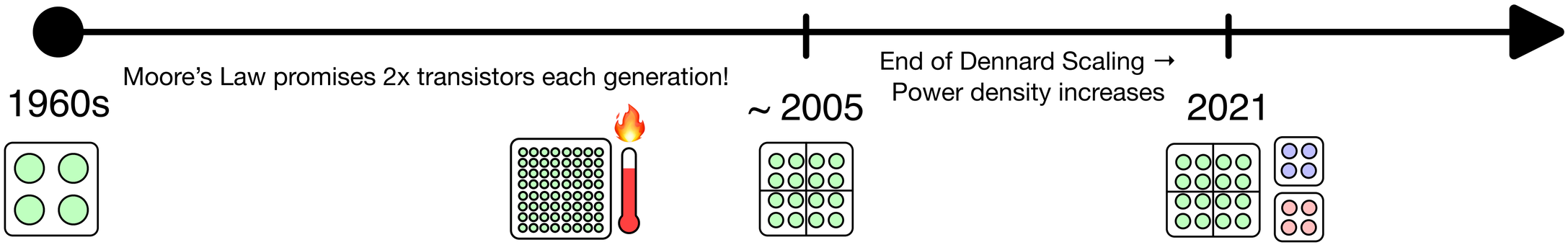
Quick history of computing



[1]



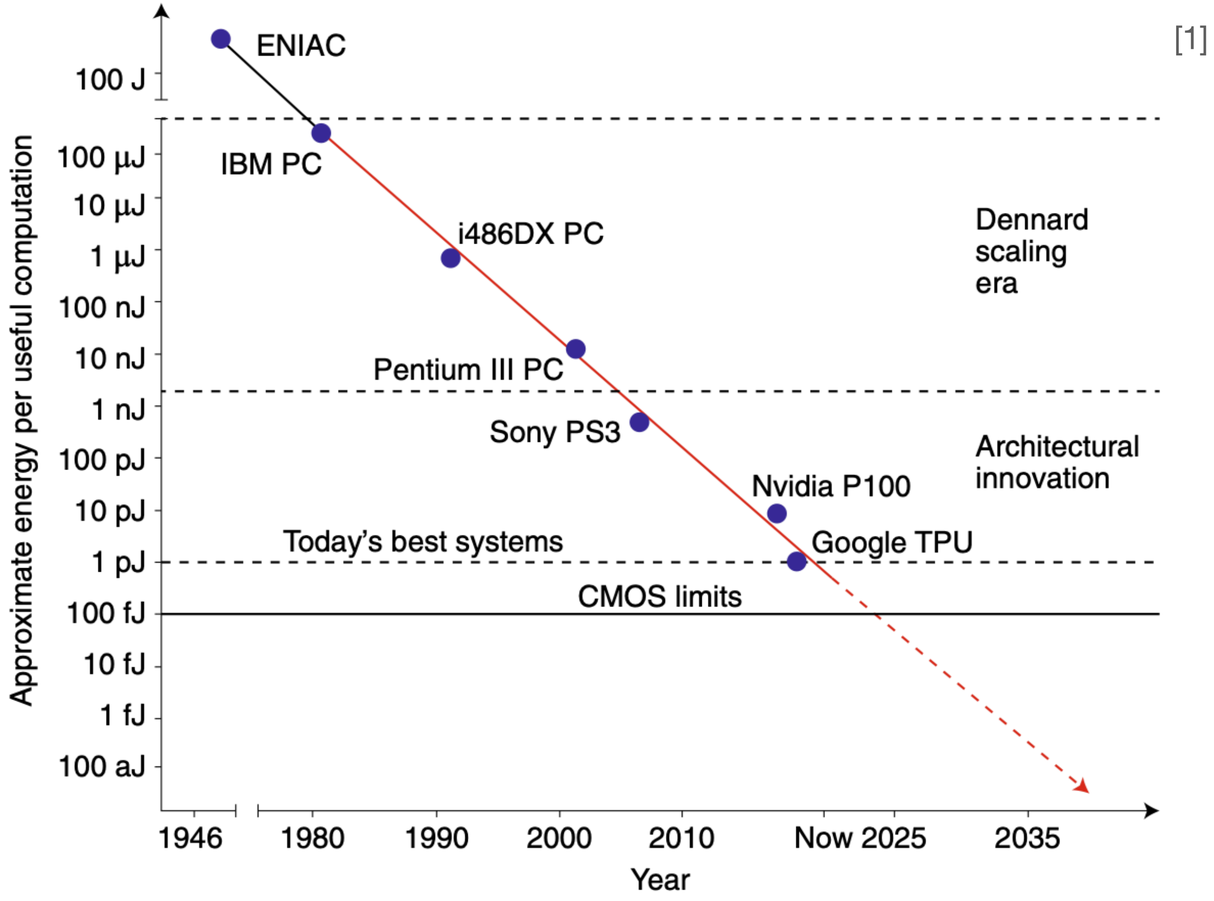
[2]



[1] M. L. Rieger, "Retrospective on VLSI value scaling and lithography," *J. Micro/Nanolith. MEMS MOEMS*, vol. 18, no. 04, p. 1, Nov. 2019, doi: 10.1117/1.JMM.18.4.040902.

[2] A. M. Rahmani, P. Liljeberg, A. Hemani, A. Jantsch, and H. Tenhunen, Eds., *The Dark Side of Silicon*. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-31596-6.

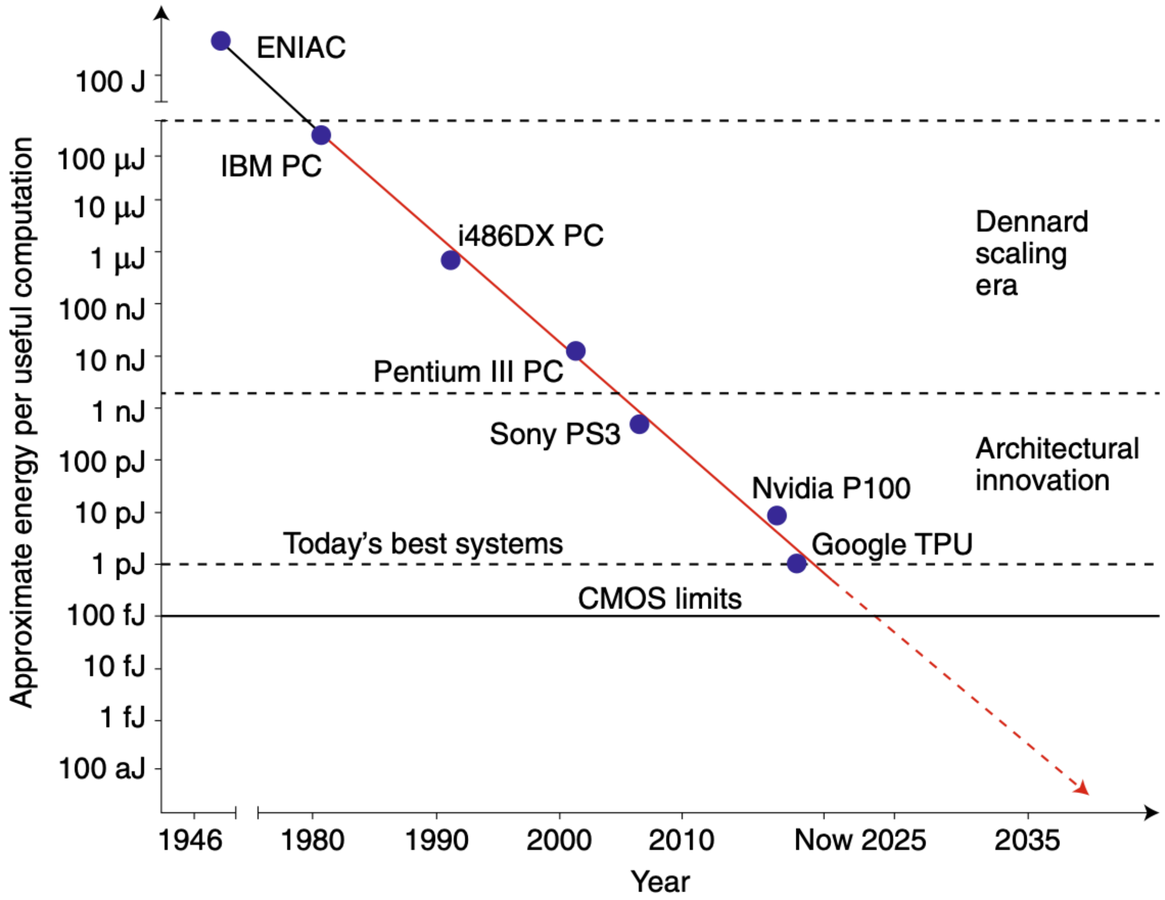
End of energy efficiency scaling



[1]

[1] M. J. Marinella and S. Agarwal, "Efficient reservoir computing with memristors," *Nat Electron*, vol. 2, no. 10, pp. 437–438, Oct. 2019, doi: 10.1038/s41928-019-0318-y.

End of energy efficiency scaling

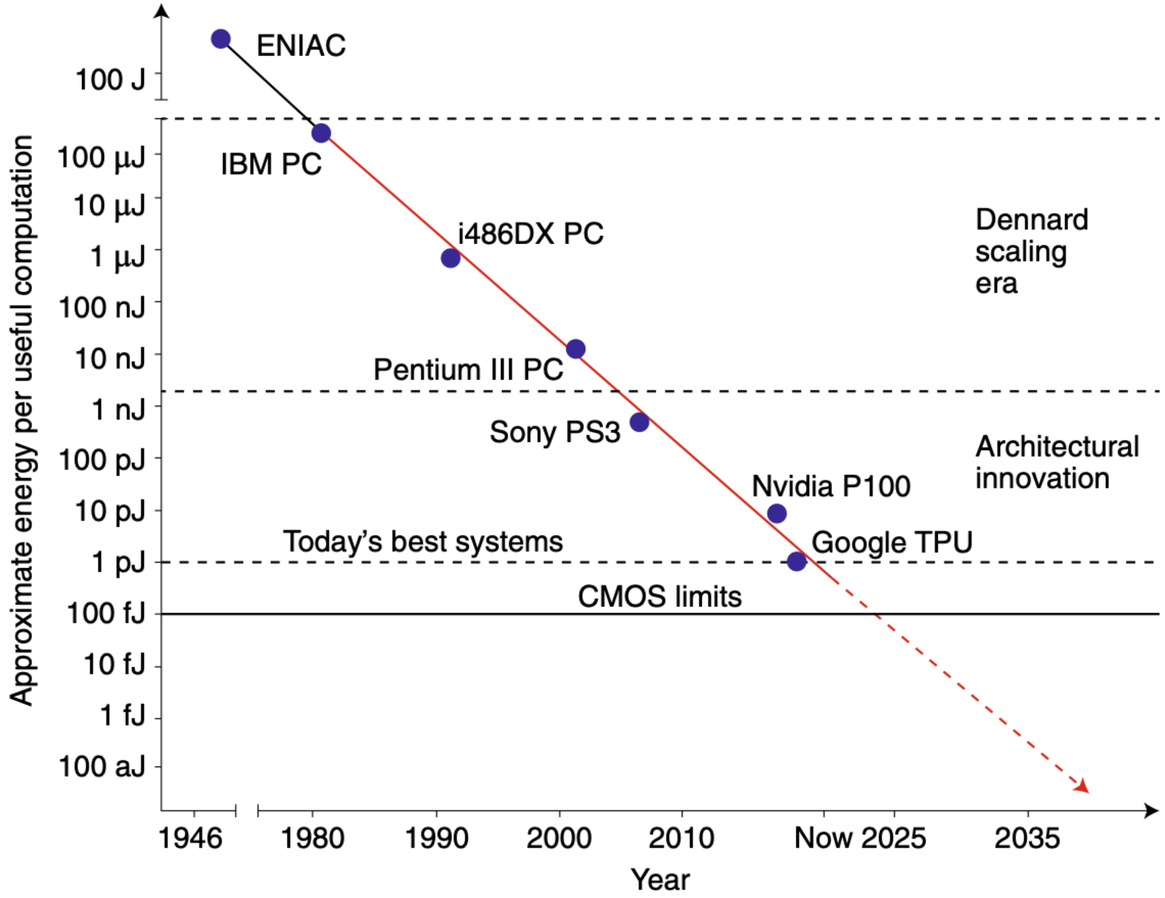


[1]

$$\text{Energy} = \frac{\text{Power}}{\text{Transistor}} \times \frac{\text{Transistors}}{\text{Computation}} \times \text{Latency}$$

[1] M. J. Marinella and S. Agarwal, "Efficient reservoir computing with memristors," *Nat Electron*, vol. 2, no. 10, pp. 437–438, Oct. 2019, doi: 10.1038/s41928-019-0318-y.

End of energy efficiency scaling



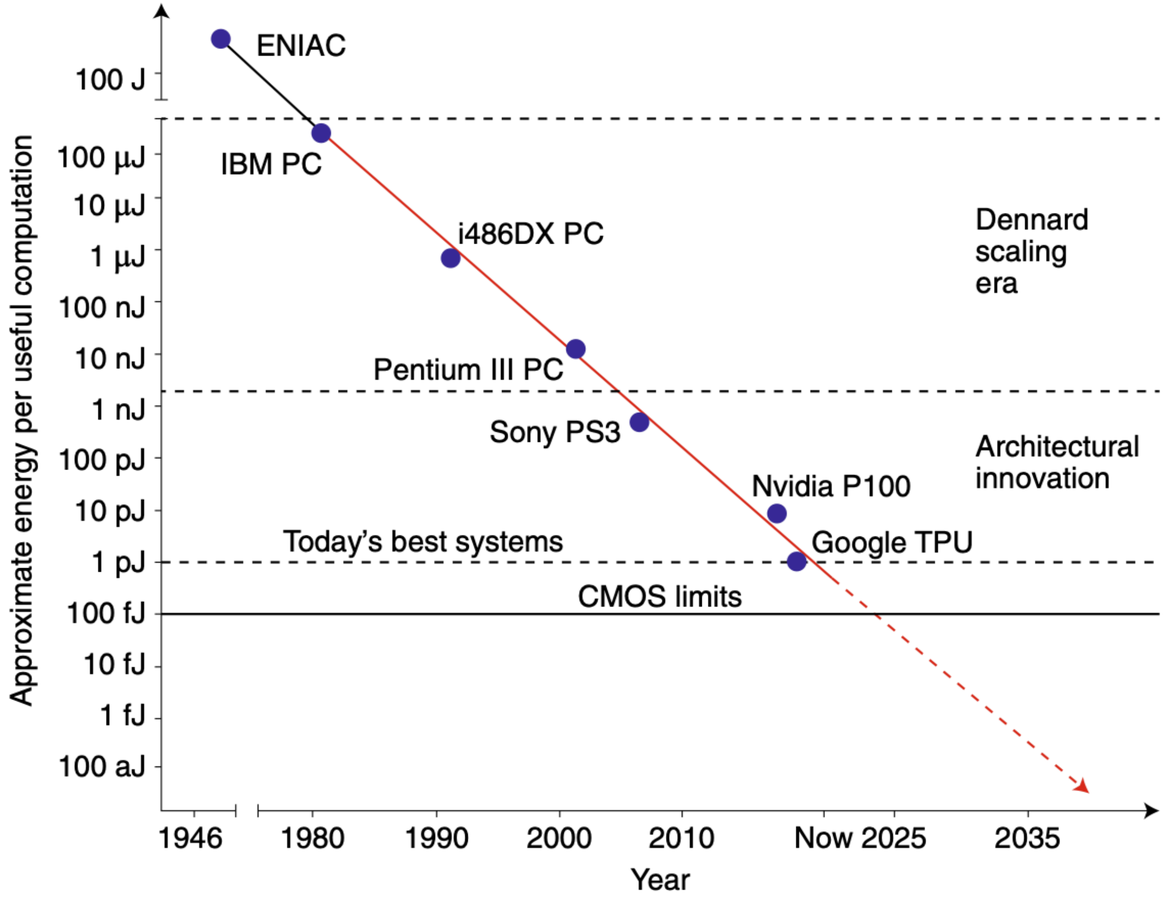
[1]

Dennard scaling

$$\text{Energy} = \frac{\text{Power}}{\text{Transistor}} \times \frac{\text{Transistors}}{\text{Computation}} \times \text{Latency}$$

[1] M. J. Marinella and S. Agarwal, "Efficient reservoir computing with memristors," *Nat Electron*, vol. 2, no. 10, pp. 437–438, Oct. 2019, doi: 10.1038/s41928-019-0318-y.

End of energy efficiency scaling



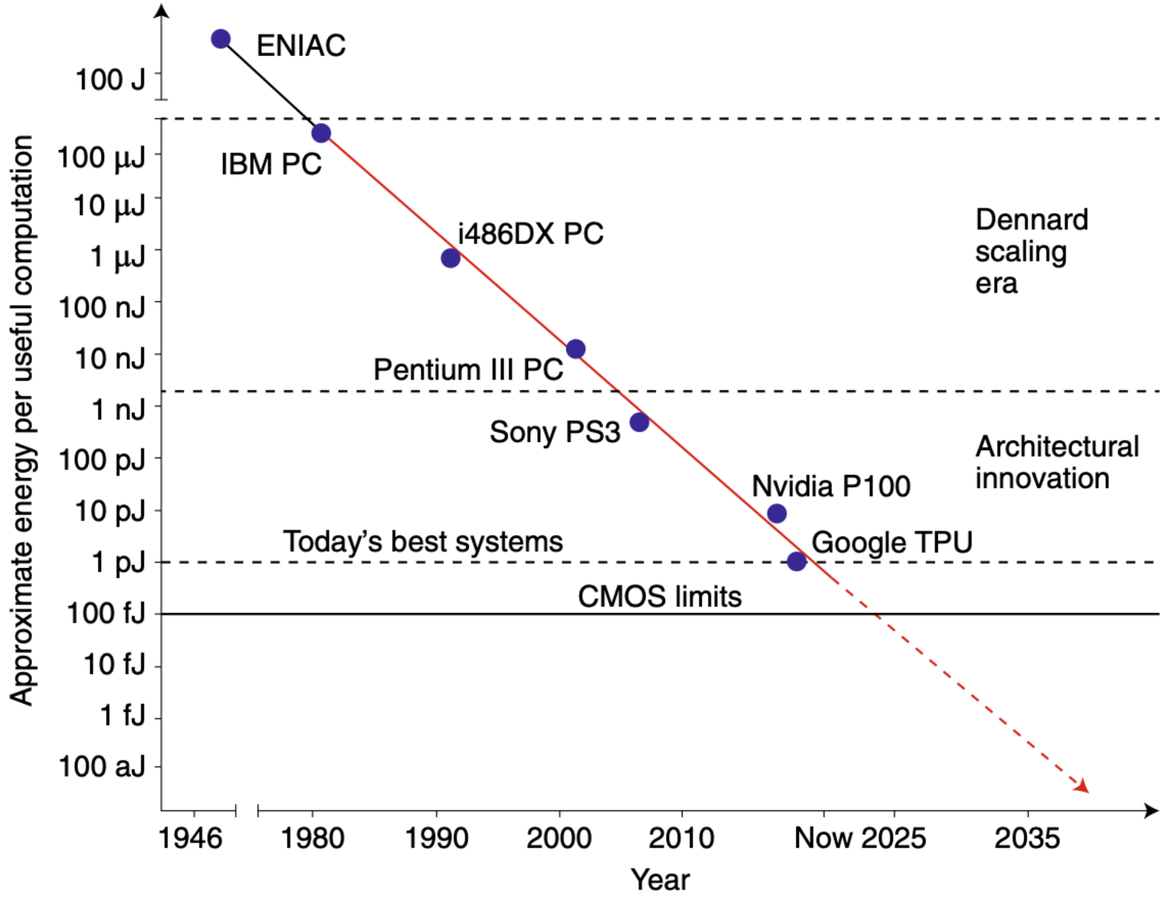
[1]

Dennard scaling — Computing paradigm

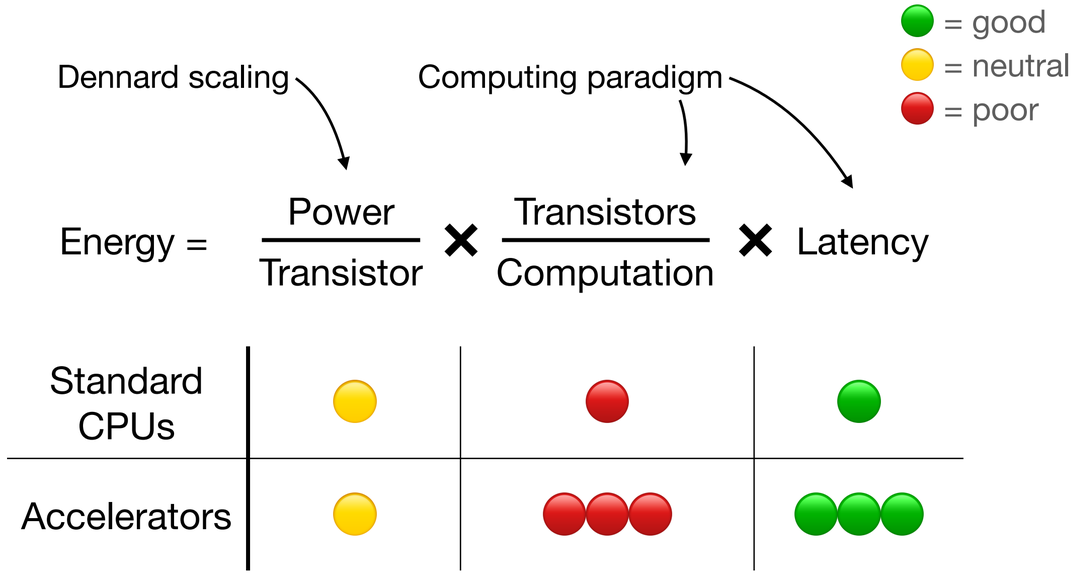
$$\text{Energy} = \frac{\text{Power}}{\text{Transistor}} \times \frac{\text{Transistors}}{\text{Computation}} \times \text{Latency}$$

[1] M. J. Marinella and S. Agarwal, "Efficient reservoir computing with memristors," *Nat Electron*, vol. 2, no. 10, pp. 437–438, Oct. 2019, doi: 10.1038/s41928-019-0318-y.

End of energy efficiency scaling

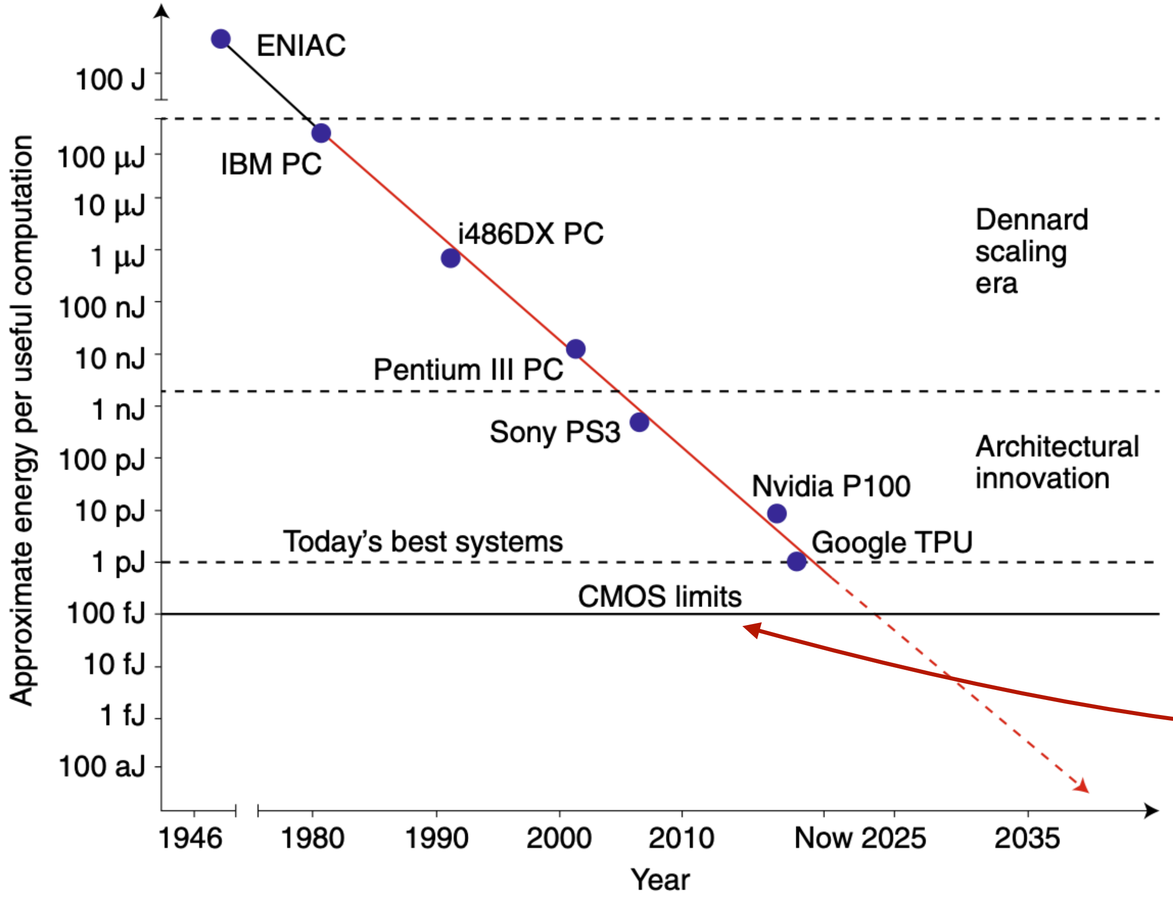


[1]

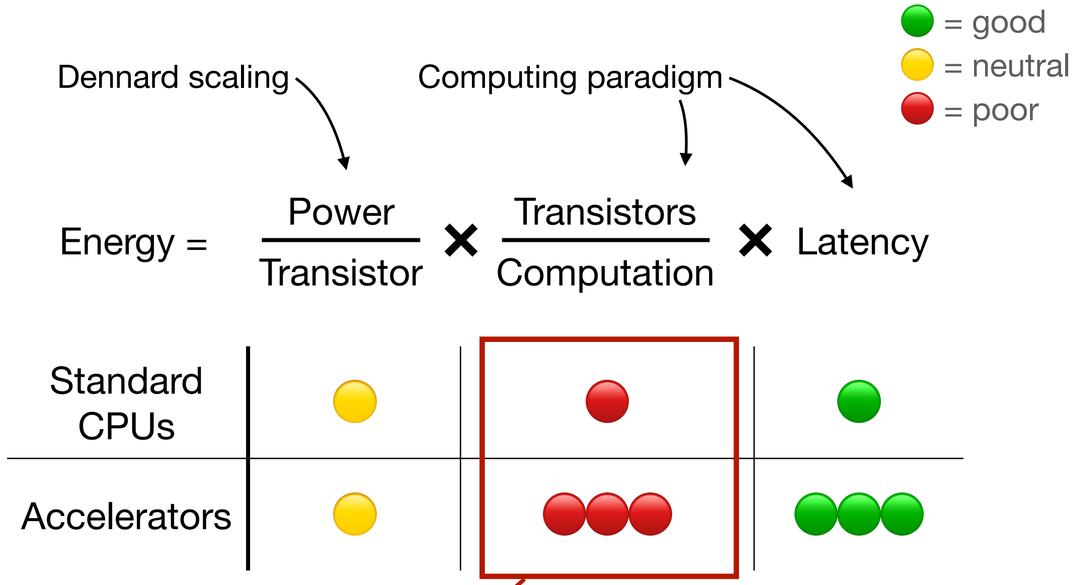


[1] M. J. Marinella and S. Agarwal, "Efficient reservoir computing with memristors," *Nat Electron*, vol. 2, no. 10, pp. 437–438, Oct. 2019, doi: 10.1038/s41928-019-0318-y.

End of energy efficiency scaling

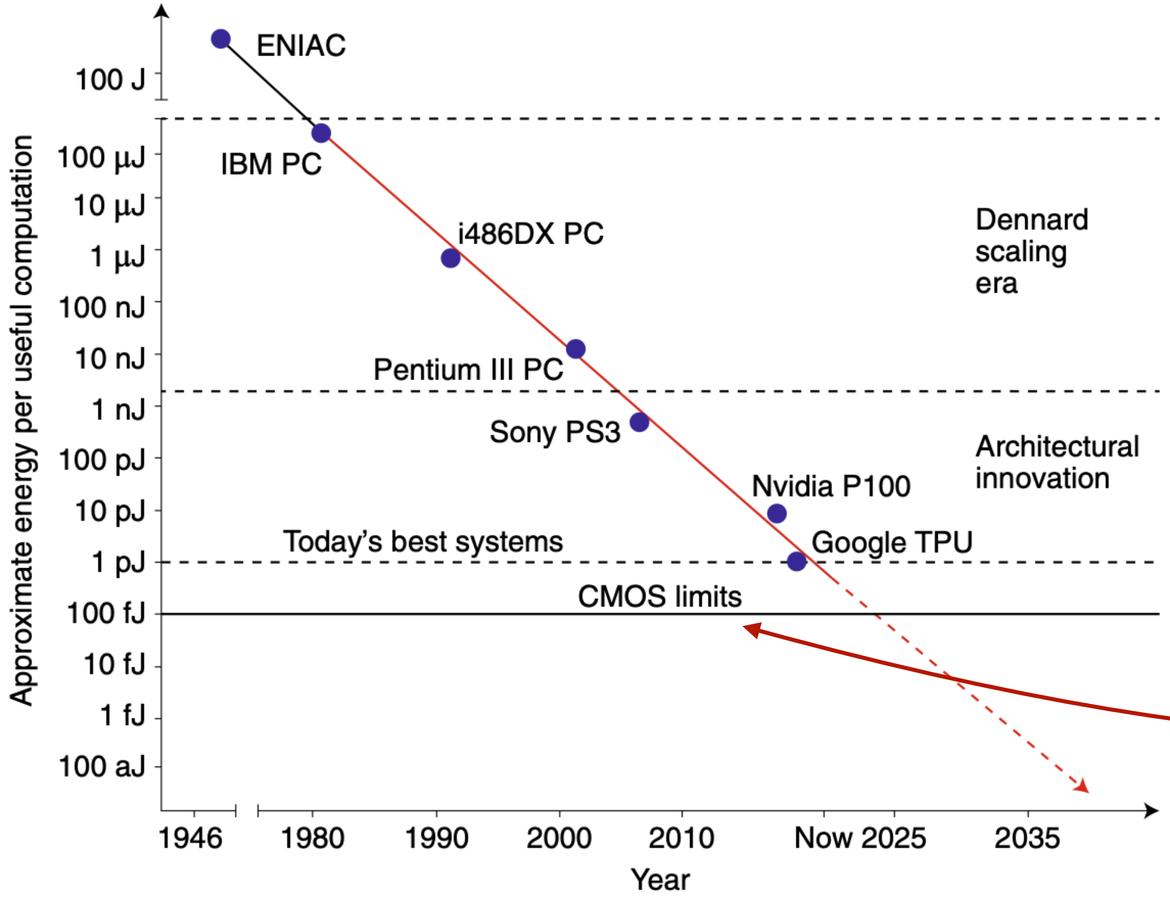


[1]

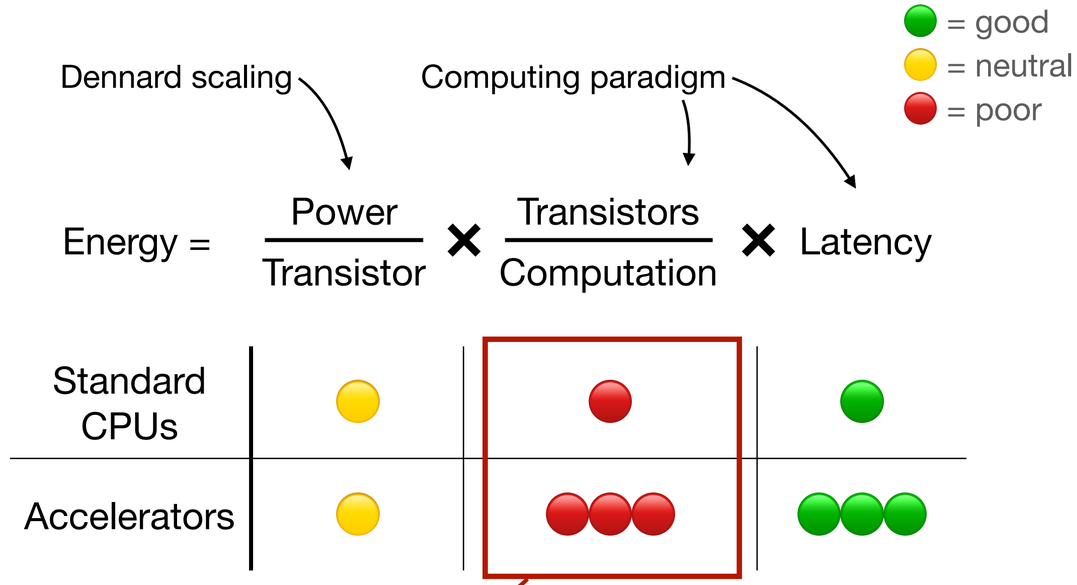


[1] M. J. Marinella and S. Agarwal, "Efficient reservoir computing with memristors," *Nat Electron*, vol. 2, no. 10, pp. 437–438, Oct. 2019, doi: 10.1038/s41928-019-0318-y.

End of energy efficiency scaling



[1]

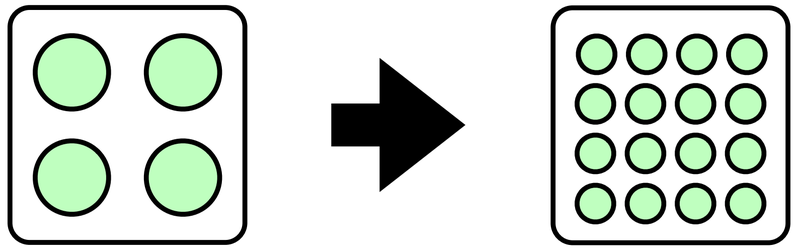


we need to build flexible computers without more transistors

[1] M. J. Marinella and S. Agarwal, "Efficient reservoir computing with memristors," *Nat Electron*, vol. 2, no. 10, pp. 437–438, Oct. 2019, doi: 10.1038/s41928-019-0318-y.

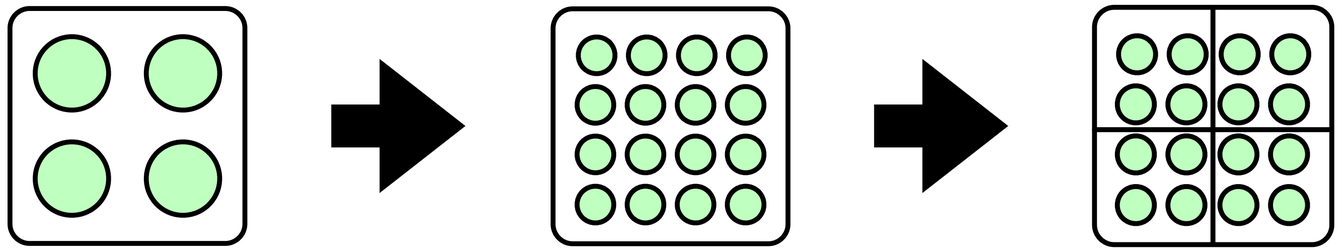
How we get there

How we get there



Increase performance by
building larger computers

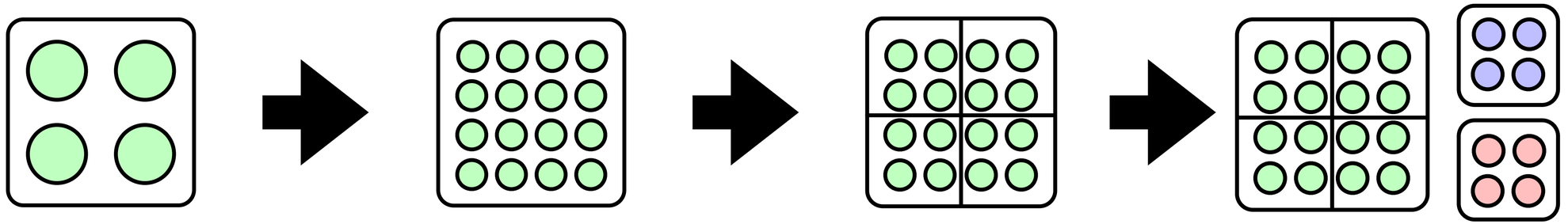
How we get there



Increase performance by building larger computers

Increase performance by exploiting parallelism

How we get there

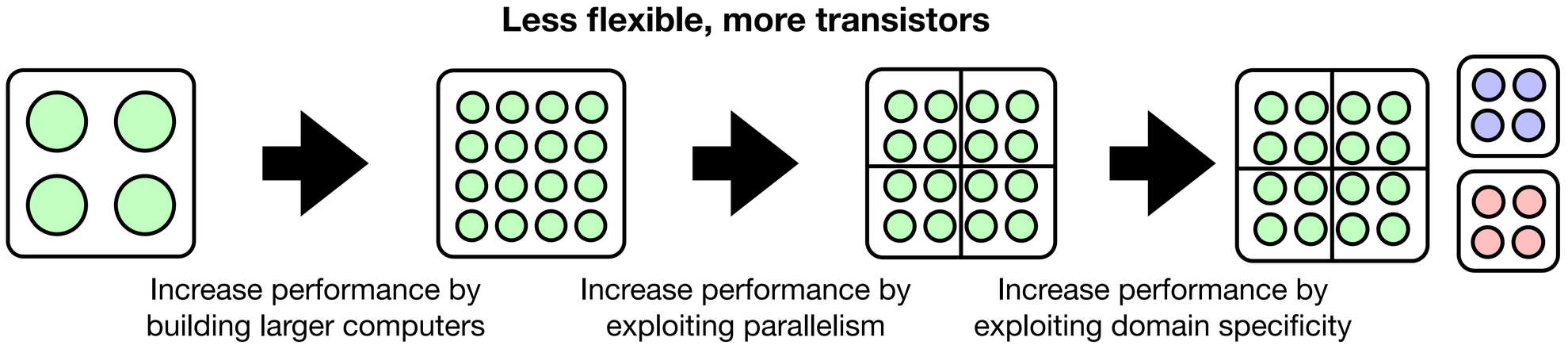


Increase performance by building larger computers

Increase performance by exploiting parallelism

Increase performance by exploiting domain specificity

How we get there



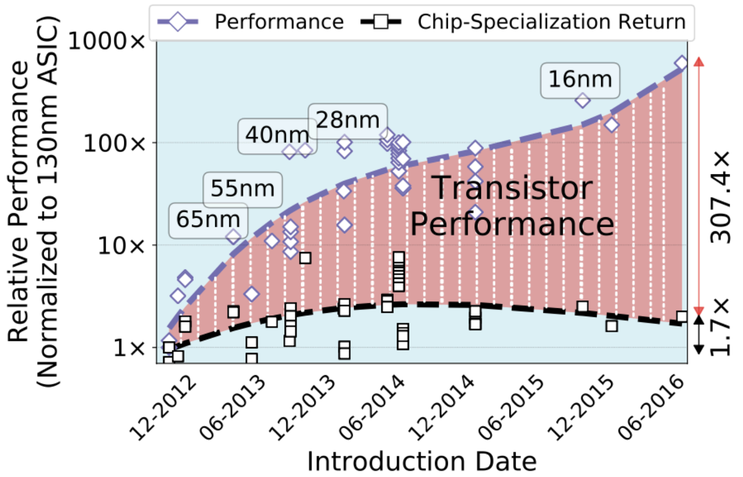
How we get there

Less flexible, more transistors

Increase performance by building larger computers

Increase performance by exploiting parallelism

Increase performance by exploiting domain specificity

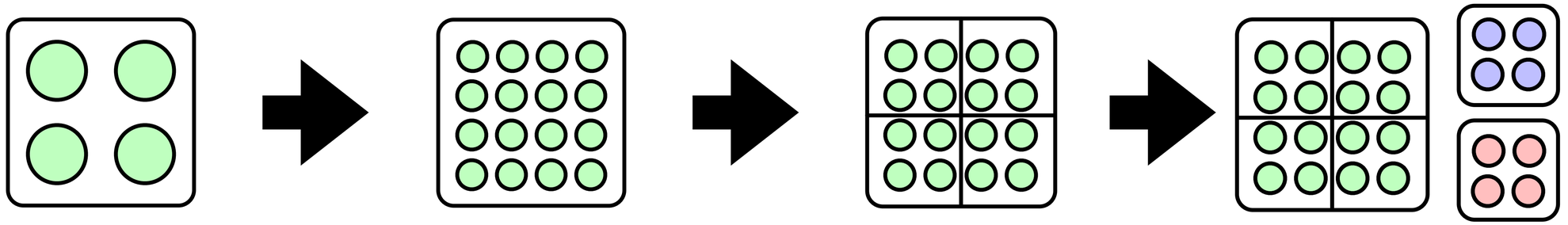


[1]

[1] A. Fuchs and D. Wentzlaff, “The Accelerator Wall: Limits of Chip Specialization,” in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Washington, DC, USA, Feb. 2019, pp. 1–14. doi: 10.1109/HPCA.2019.00023.

How we get there

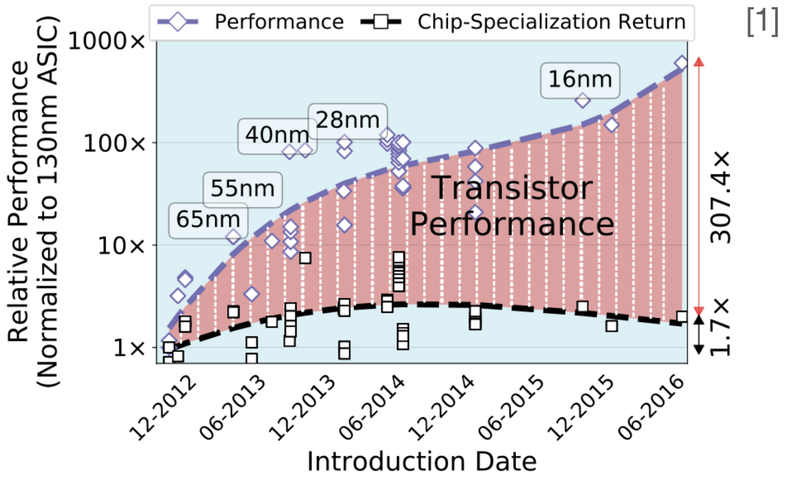
Less flexible, more transistors



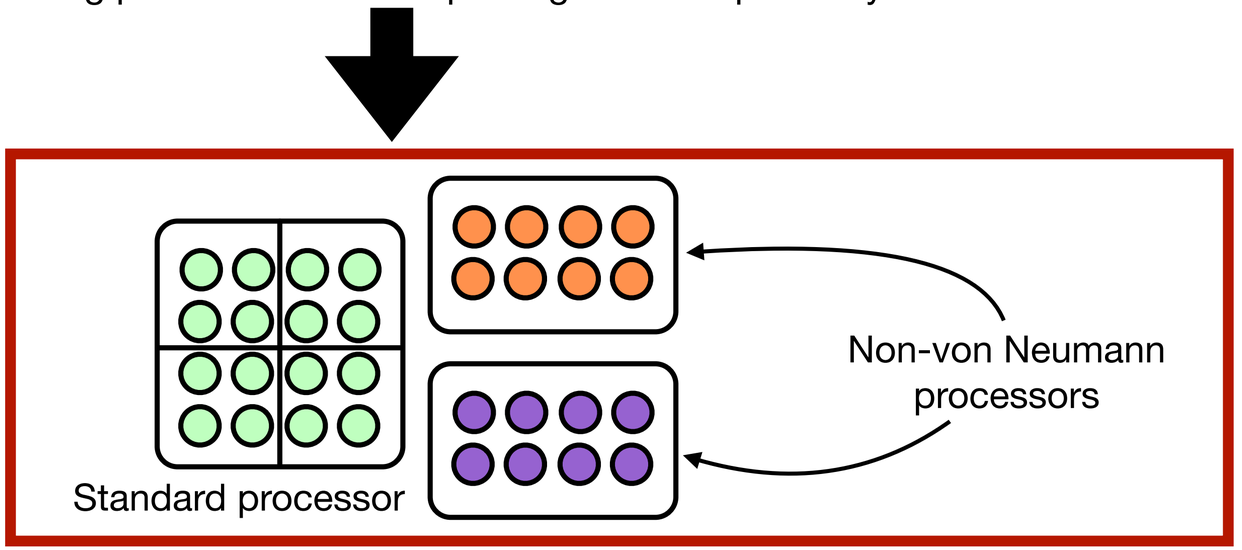
Increase performance by building larger computers

Increase performance by exploiting parallelism

Increase performance by exploiting domain specificity



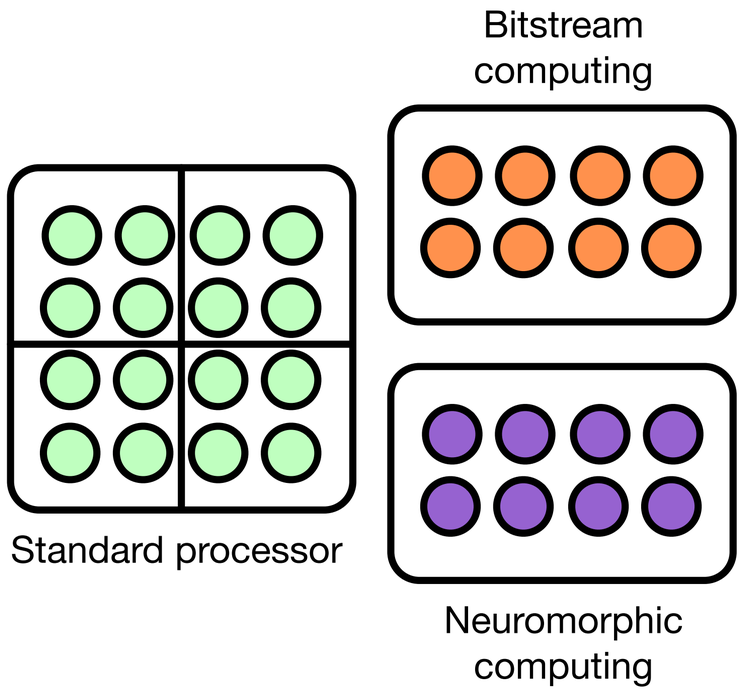
[1]



[1] A. Fuchs and D. Wentzlaff, "The Accelerator Wall: Limits of Chip Specialization," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Washington, DC, USA, Feb. 2019, pp. 1–14. doi: 10.1109/HPCA.2019.00023.

My work

- Bitstream computing
  - Standard CMOS substrate
  - Some connections to neural circuits
- Neuromorphic computing
  - Heavily brain-inspired
  - Many potential substrates
  - Can draw insight from deep learning



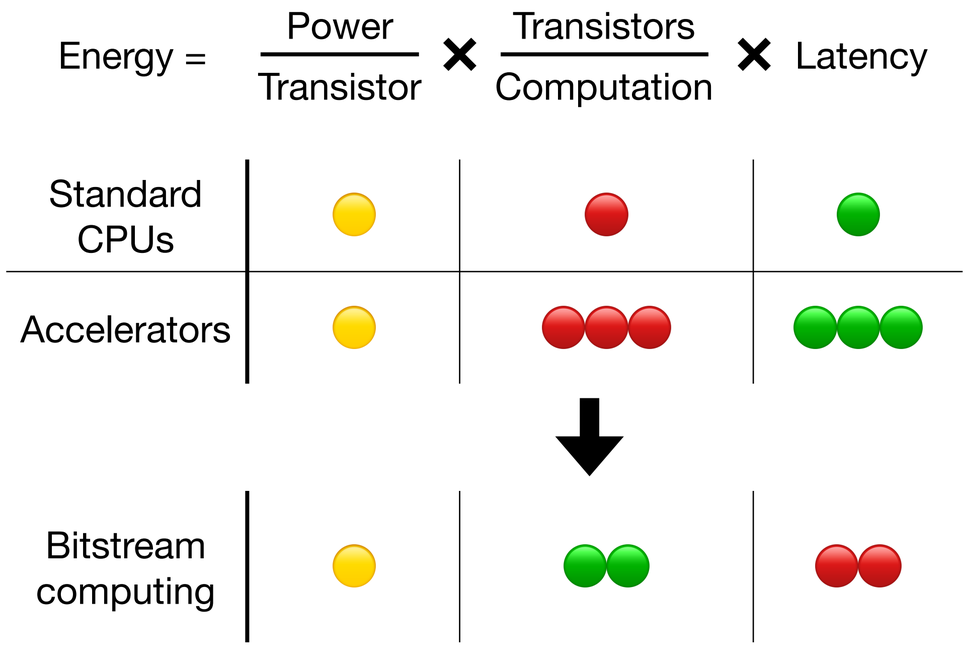
Bitstream computing

Why bitstream computing?

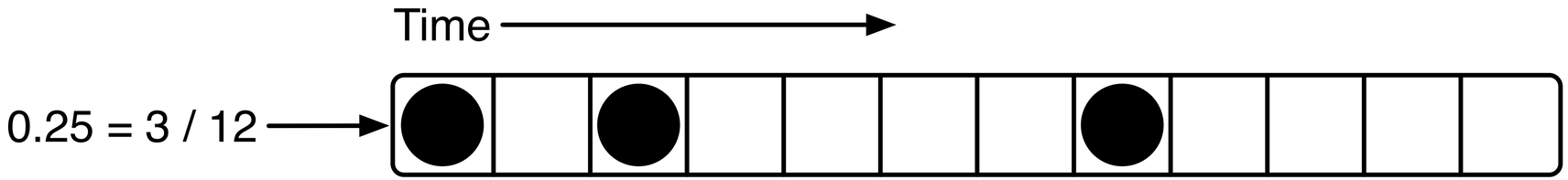
- Greater energy efficiency per computation
- Uses existing CMOS technology
- Long history of research

Why bitstream computing?

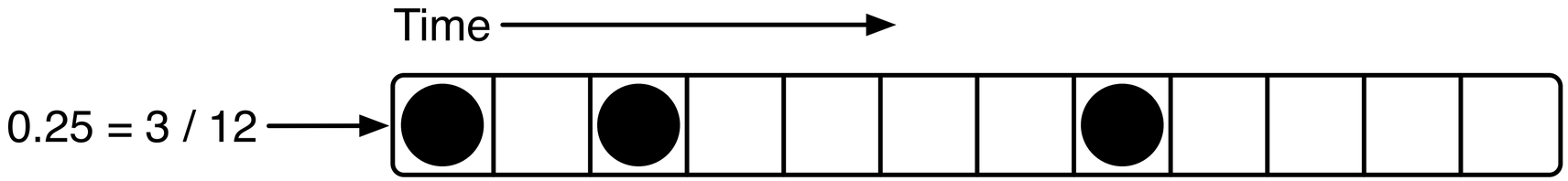
- Greater energy efficiency per computation
- Uses existing CMOS technology
- Long history of research

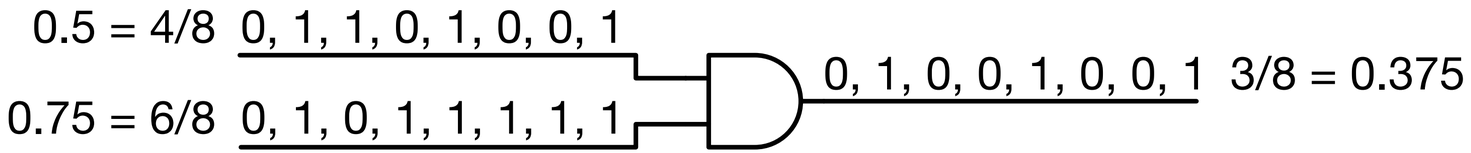


Basics of stochastic bitstreams

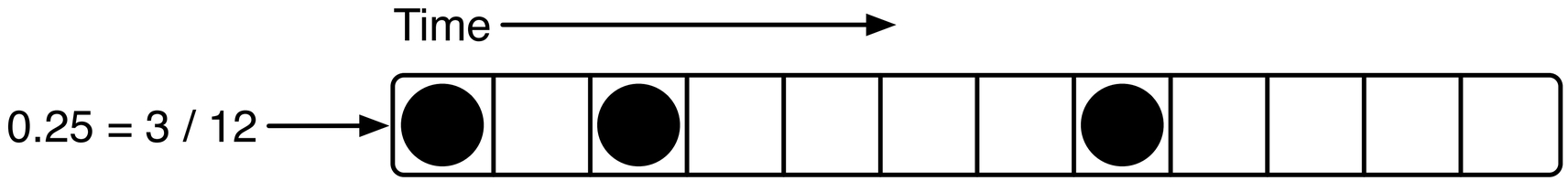


Basics of stochastic bitstreams





Basics of stochastic bitstreams



$\mathbb{E}[S_1] =$ 0.5 = 4/8 0, 1, 1, 0, 1, 0, 0, 1

$\mathbb{E}[S_2] =$ 0.75 = 6/8 0, 1, 0, 1, 1, 1, 1, 1

0, 1, 0, 0, 1, 0, 0, 1

$\mathbb{E}[S_1 S_2] =$

$\mathbb{E}[S_1]\mathbb{E}[S_2] =$

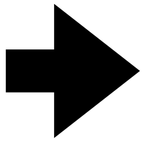
3/8 = 0.375

General purpose bitstream computing

- More operators than just multiplication:
  - Addition, subtraction, division
  - Non-linear functions: square root, tanh
  - Matrix multiplication
  - Vector norms
  - Matrix pseudo-inverse
  - Singular-value decomposition

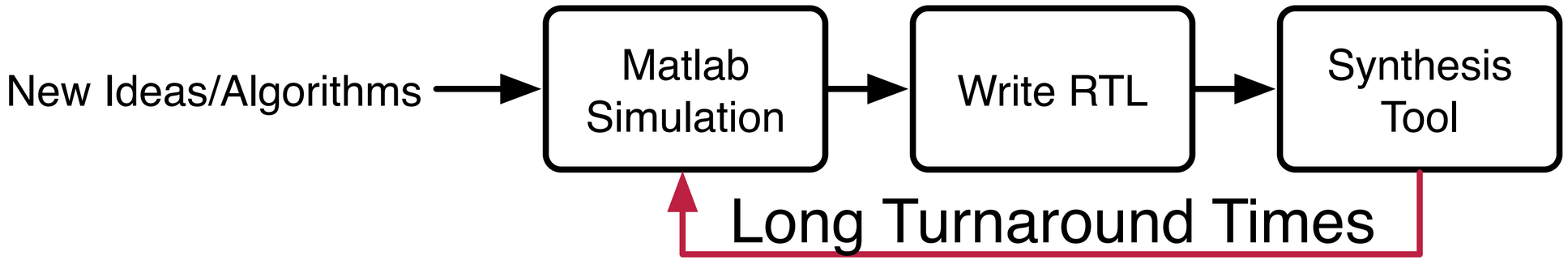
General purpose bitstream computing

- More operators than just multiplication:
  - Addition, subtraction, division
  - Non-linear functions: square root, tanh
  - Matrix multiplication
  - Vector norms
  - Matrix pseudo-inverse
  - Singular-value decomposition



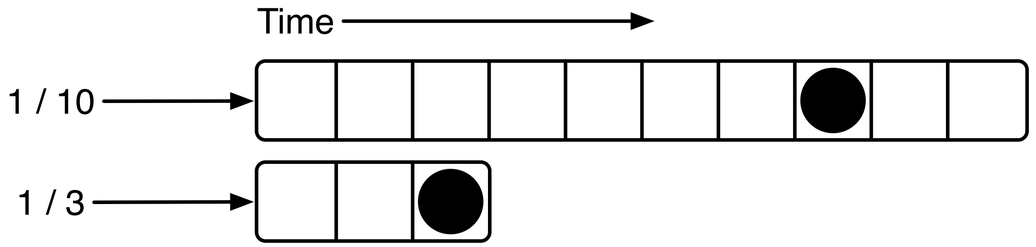
Just build a bitstream computer?

Long design cycles

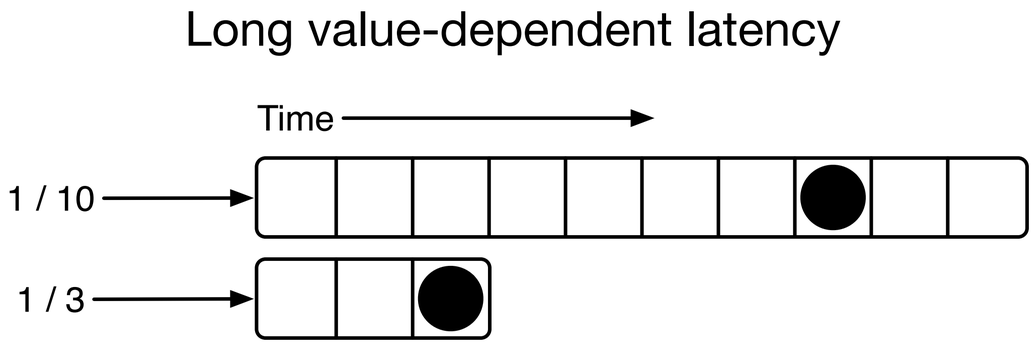


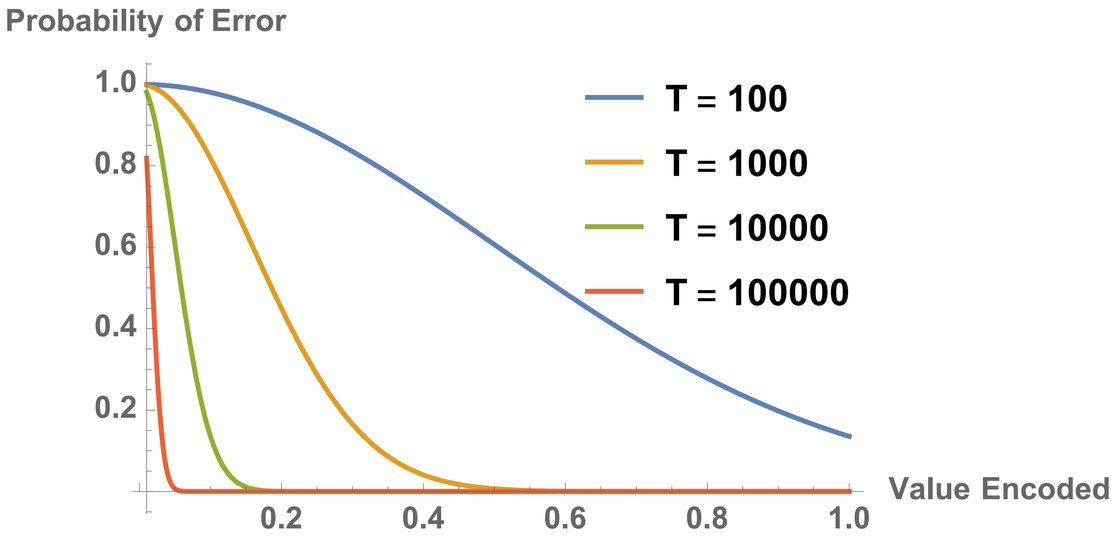
Issues with bitstream computing

Long value-dependent latency



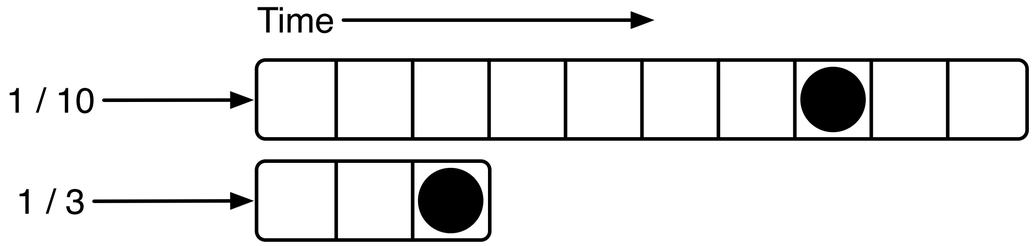
Issues with bitstream computing



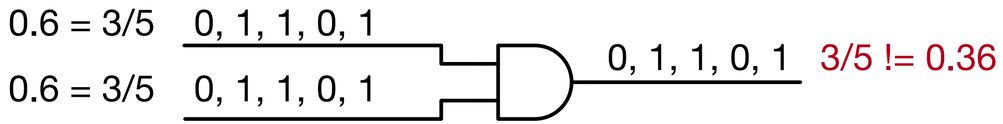


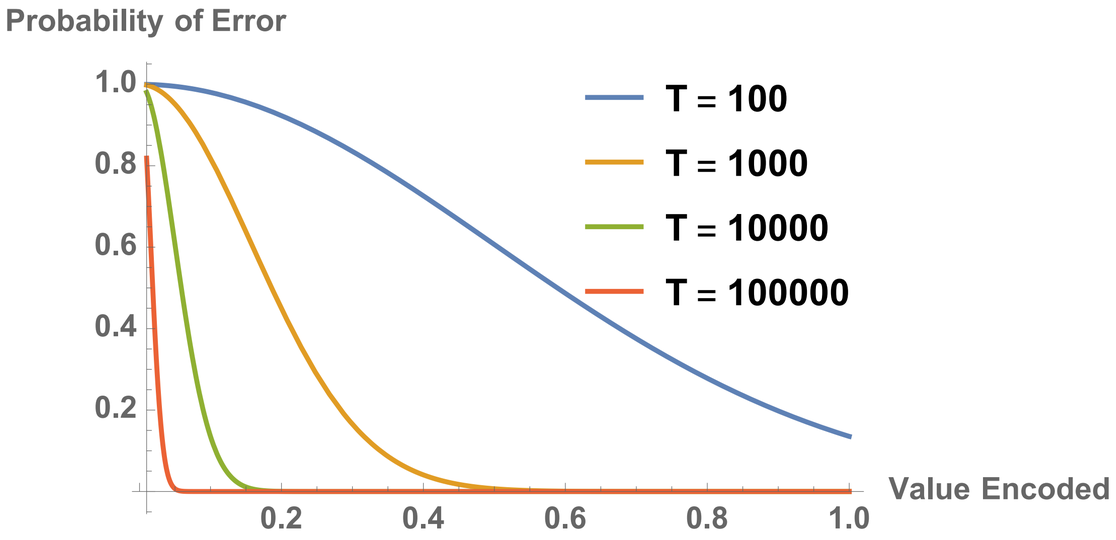
Issues with bitstream computing

Long value-dependent latency



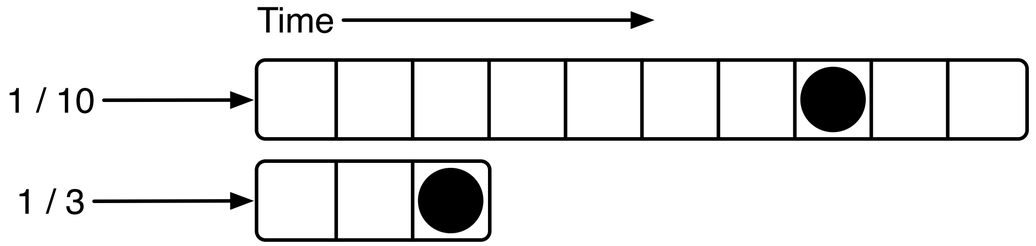
Input correlation error



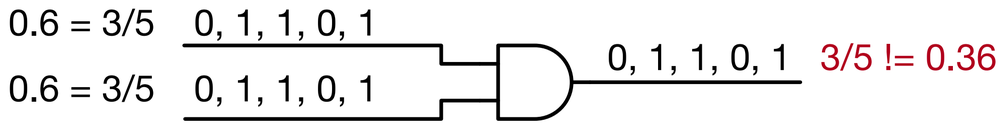


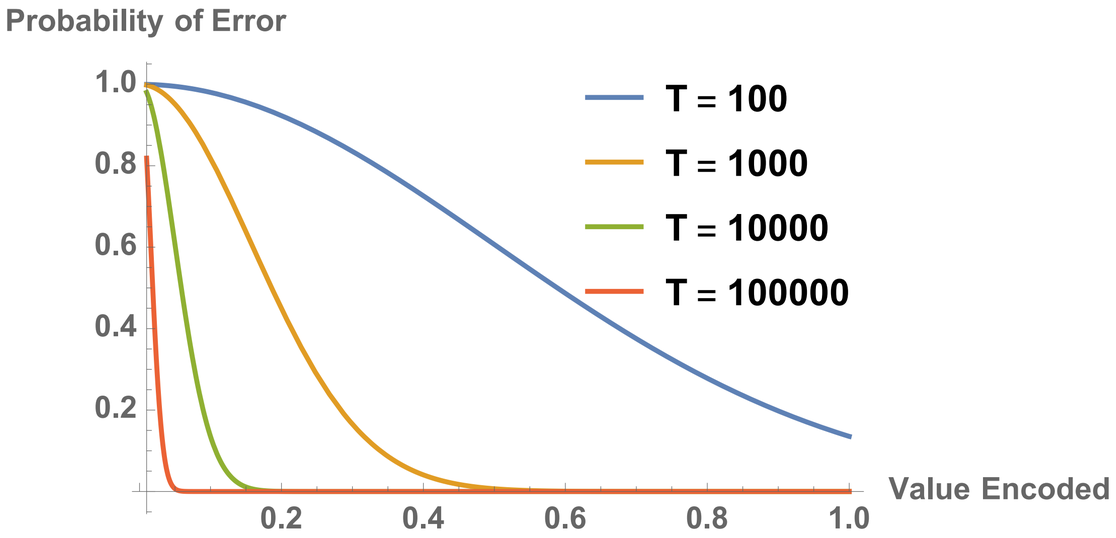
Issues with bitstream computing

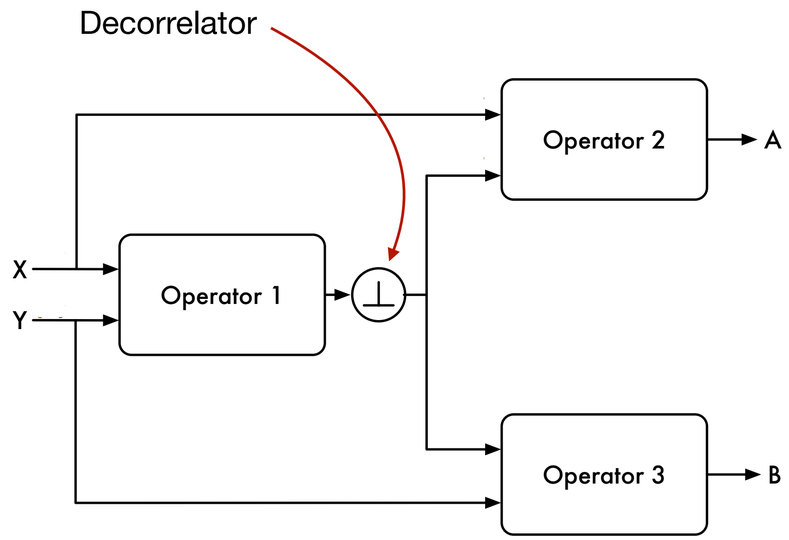
Long value-dependent latency



Input correlation error



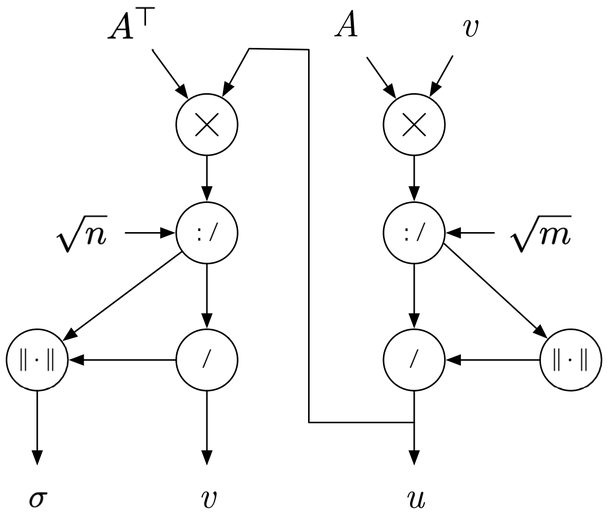




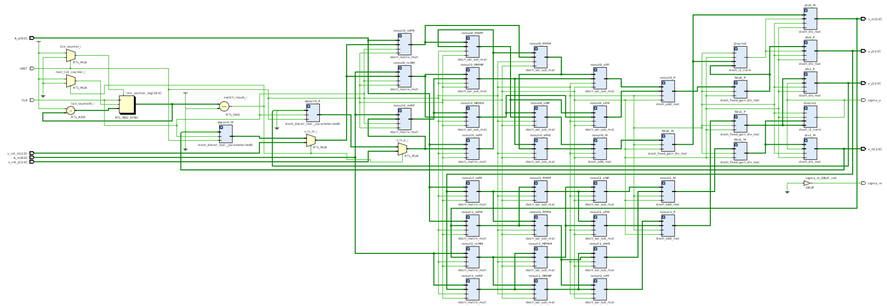
More issues with bitstream computing

- Not just programming a processor

Power iteration SVD algorithm



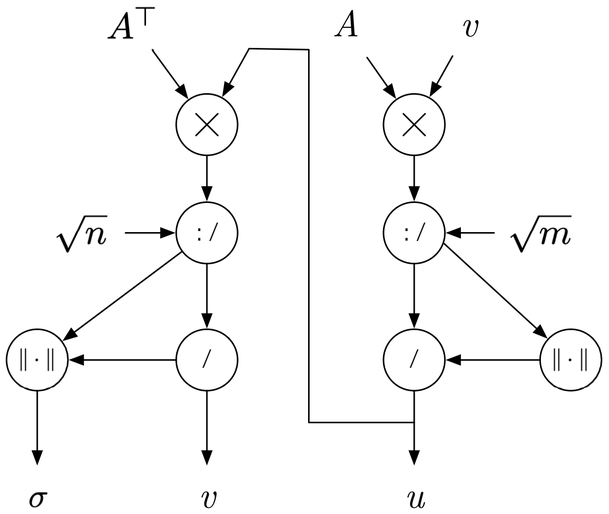
Power iteration SVD circuit



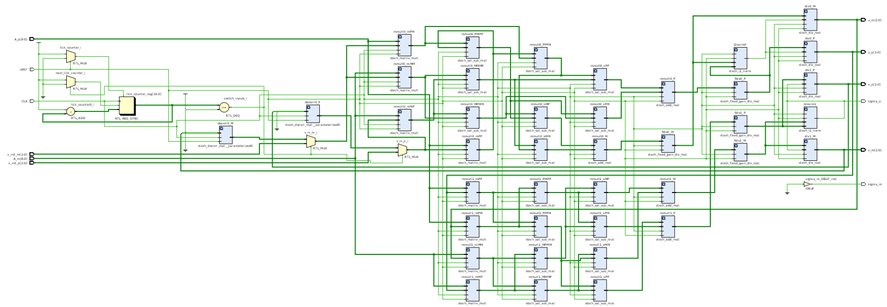
More issues with bitstream computing

- Not just programming a processor
- Building a hardware circuit

Power iteration SVD algorithm

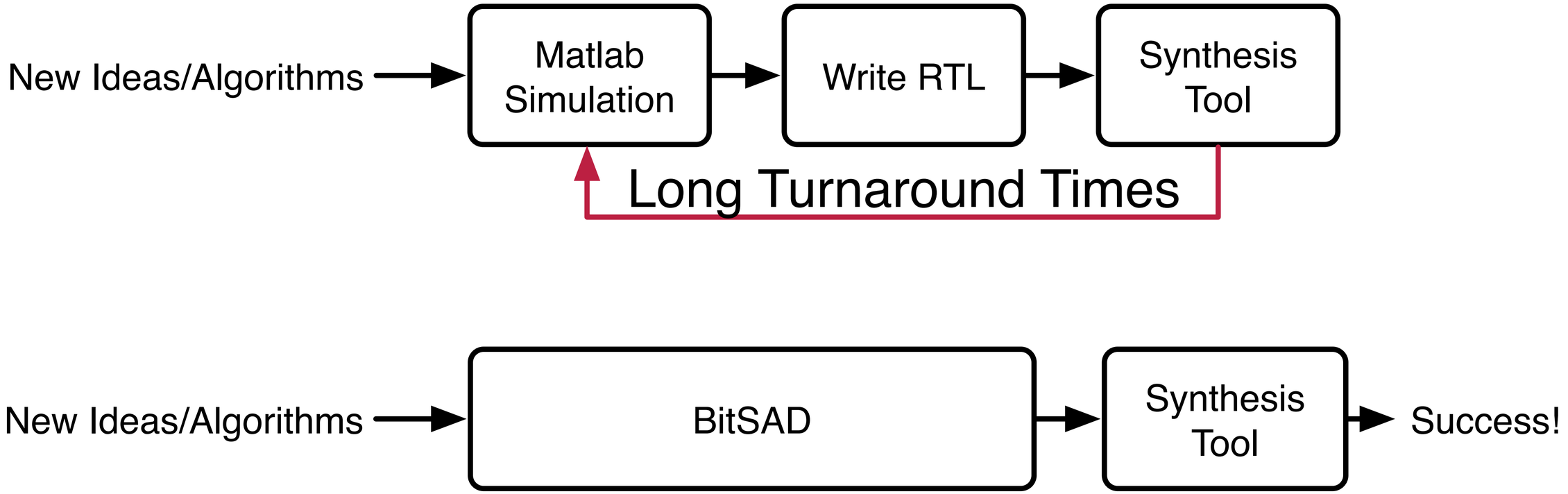


Power iteration SVD circuit

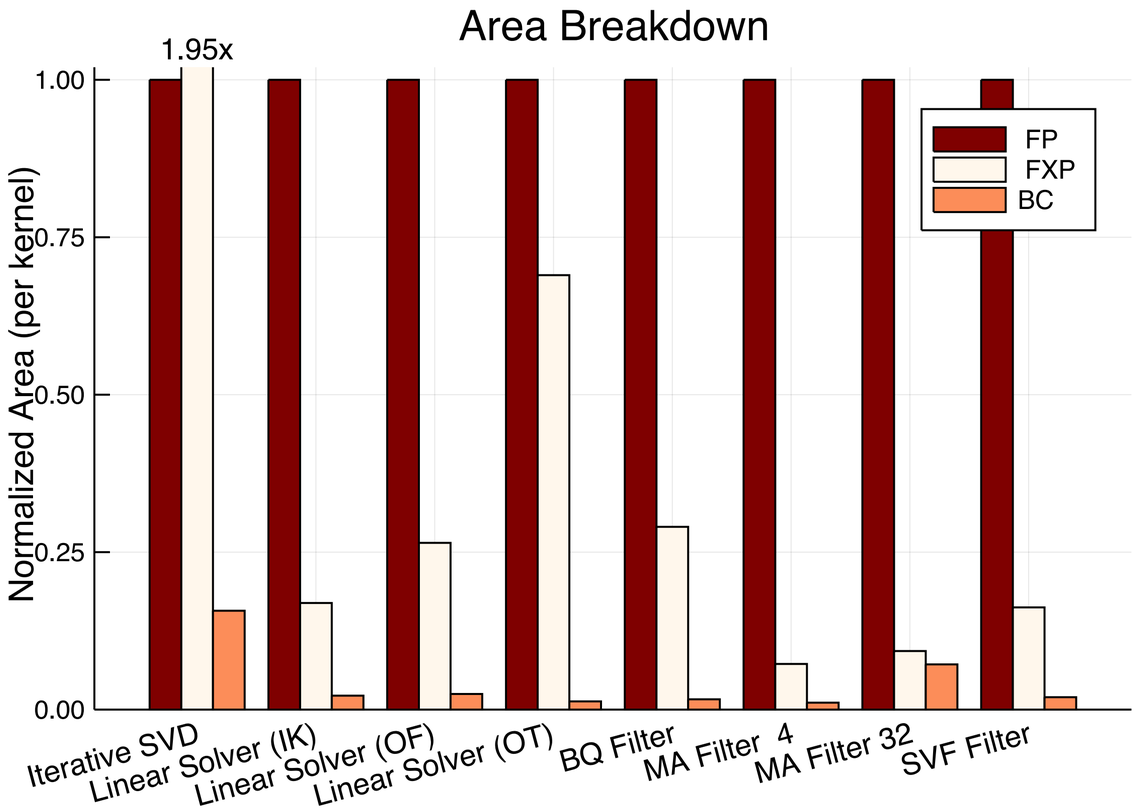


Enter BitSAD

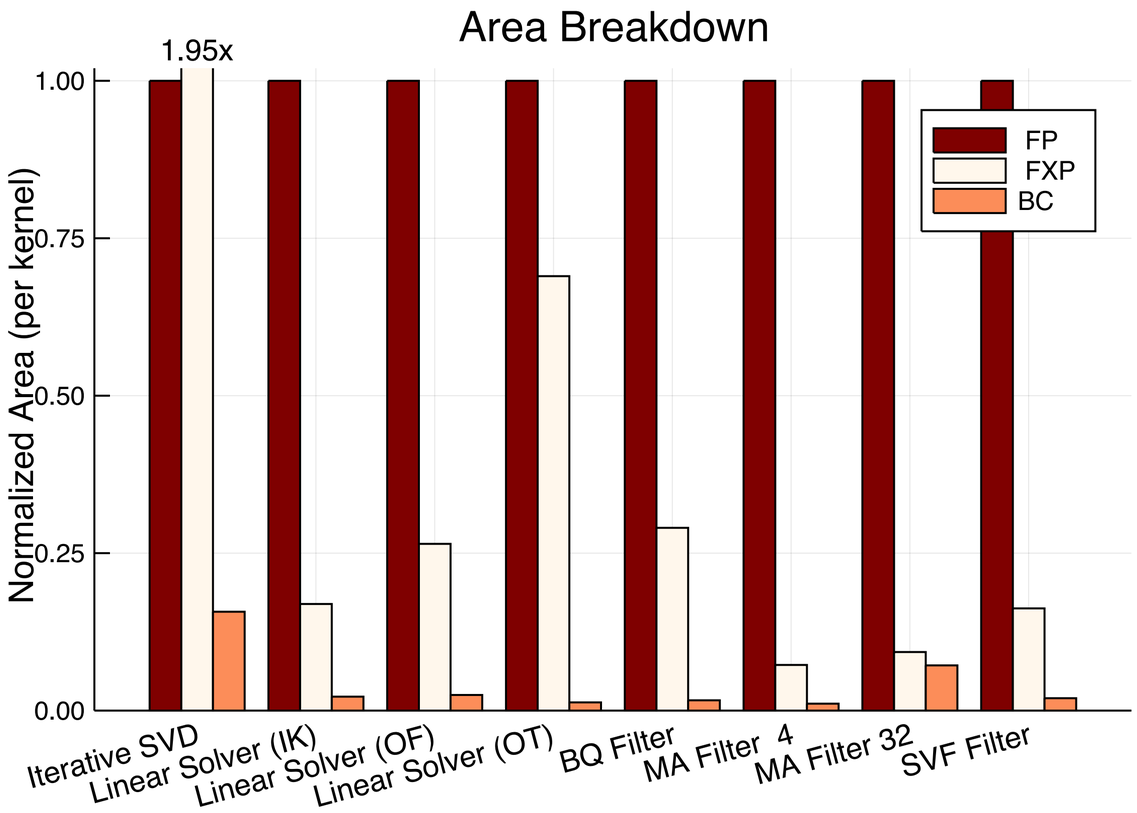
Programming language for bitstream computing

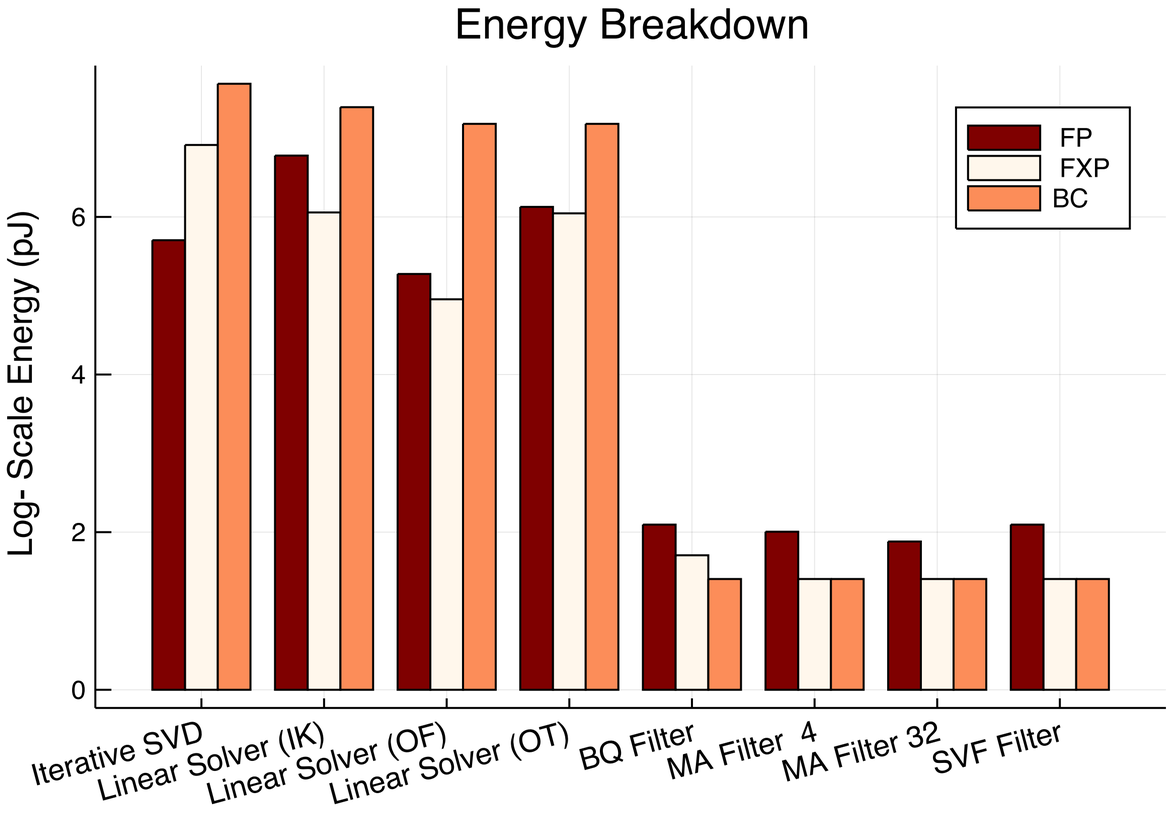


Bitstream computing results

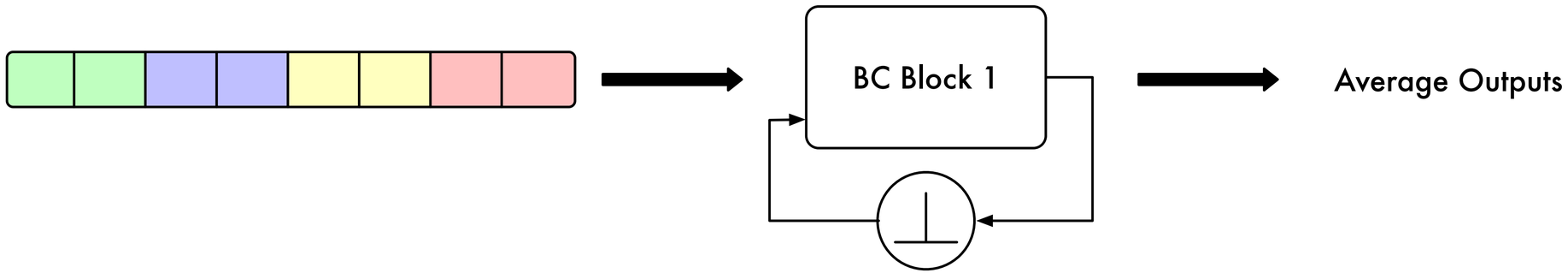


Bitstream computing results

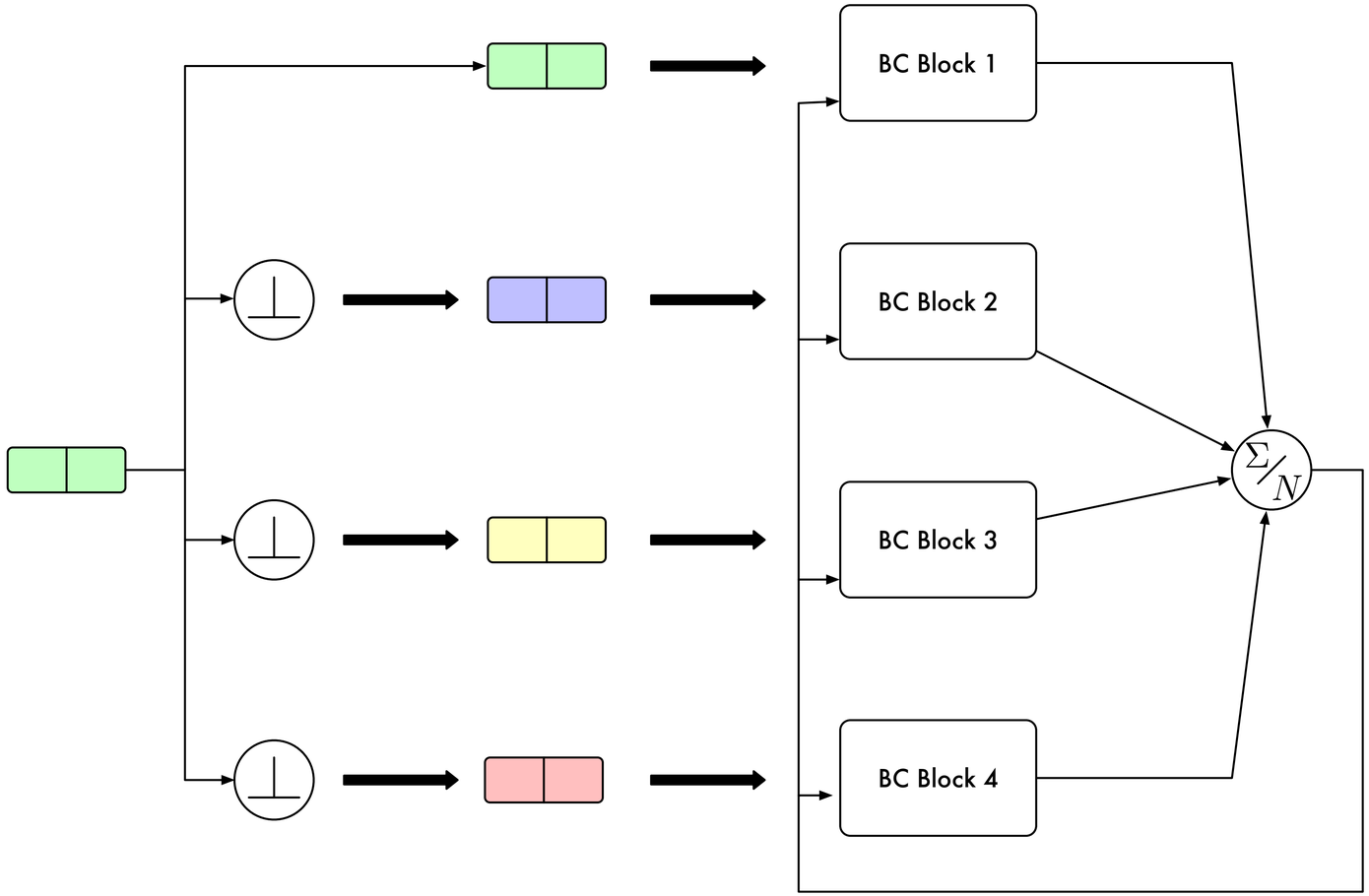




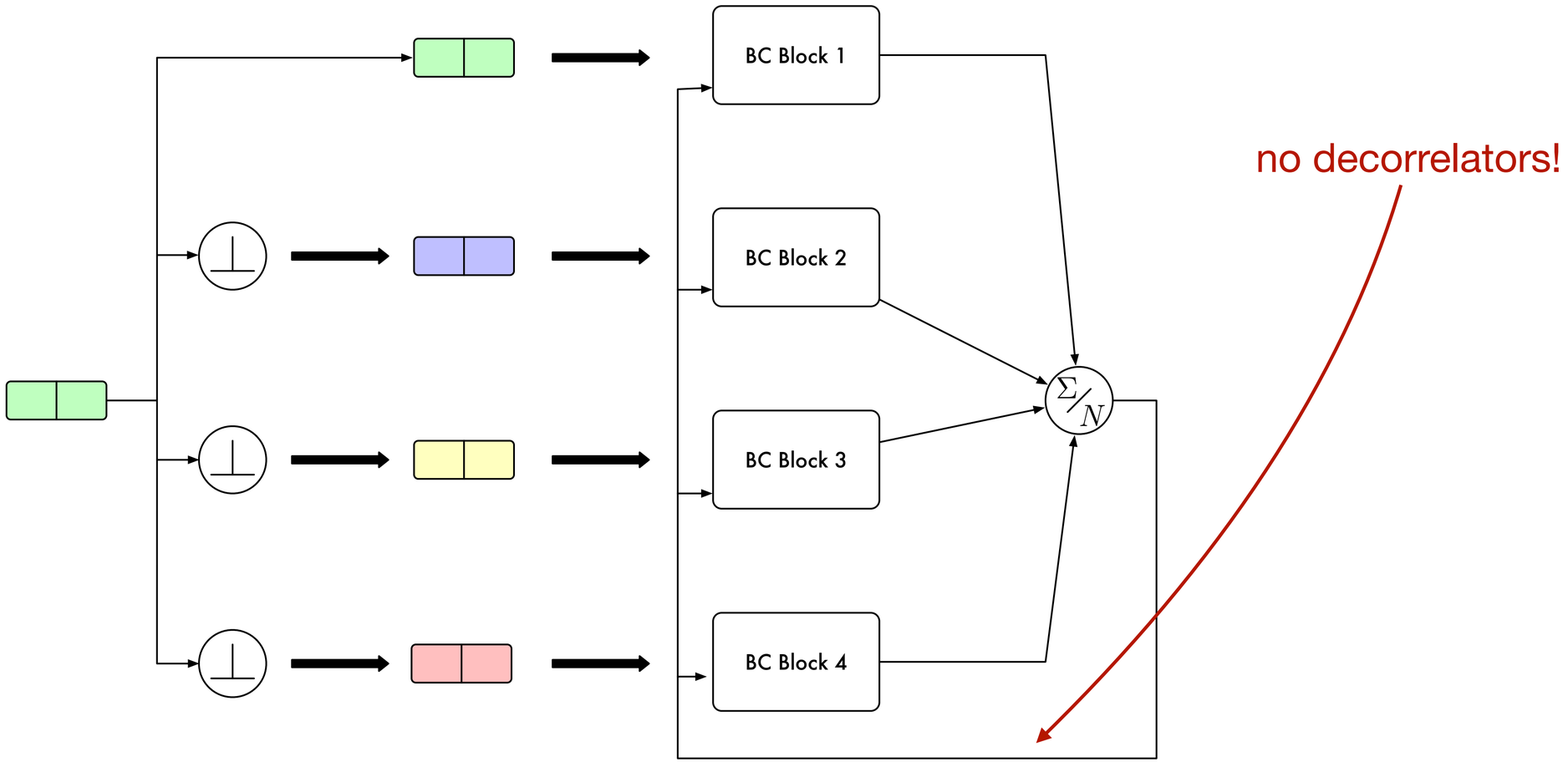
Parallelizing bitstream computing



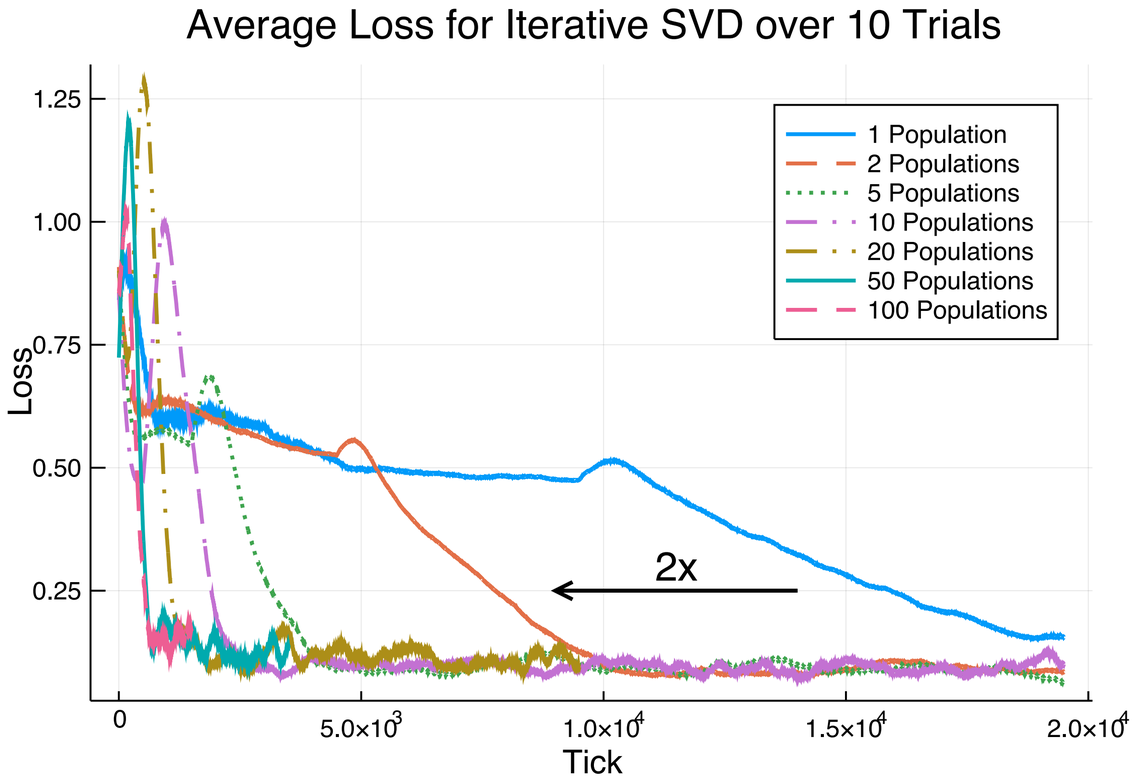
Parallelizing bitstream computing

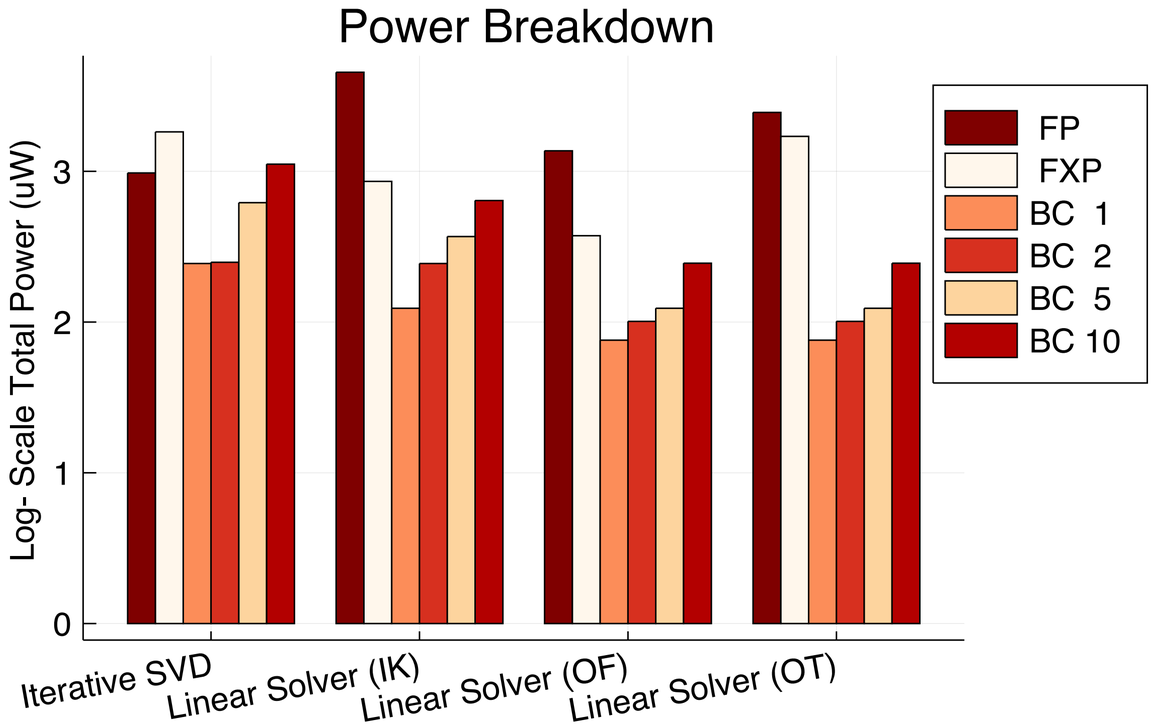


Parallelizing bitstream computing



Population coding is more efficient





Conclusions & future work

- Bitstream computing is a power-efficient computing model for edge devices
- BitSAD allows programmers to easily map high-level algorithms to bitstream computing hardware
- Bitstream computing suffers from long latencies (partially addressed by population coding)
- Future work:
  - Automated optimization of programs using BitSAD compiler
  - Extending population coding theoretical guarantees to a broader class of “reduction” operators

Neuromorphic computing

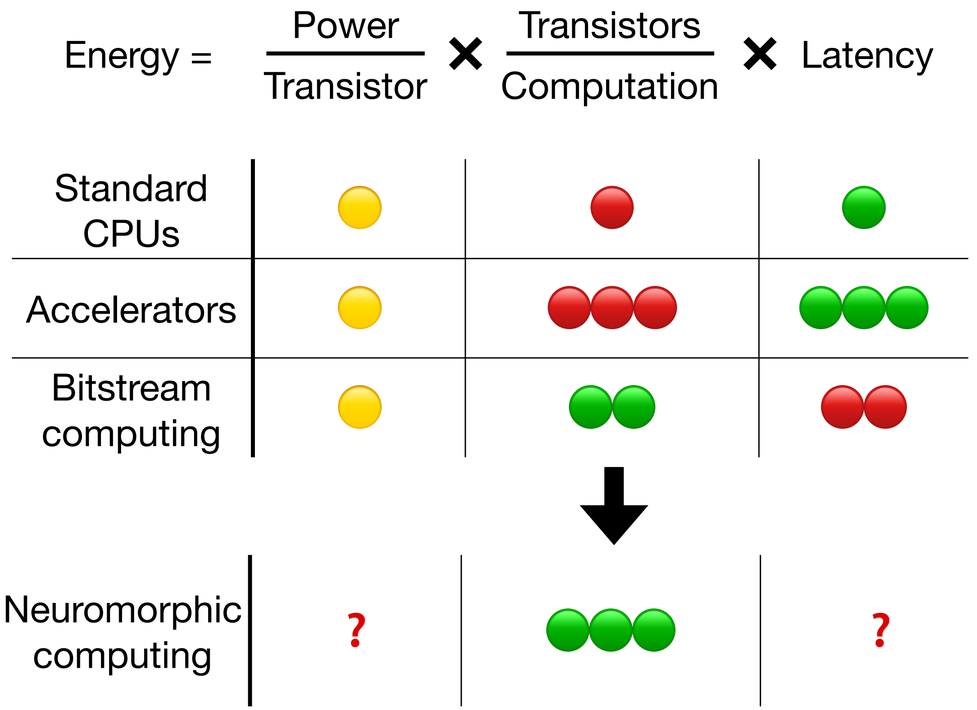
Why neuromorphic computing?

- Deep artificial neural networks are replacement for programs
- Spiking neural networks (SNNs) are an energy efficient alternative
- Opportunity to use new substrates
  - Future is uncertain

- Still waiting to scale up SNNs

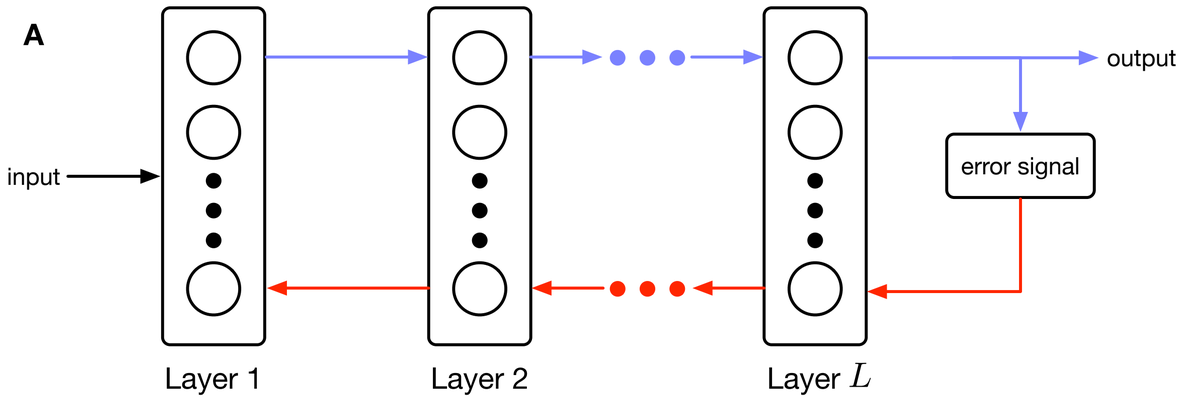
Why neuromorphic computing?

- Deep artificial neural networks are replacement for programs
- Spiking neural networks (SNNs) are an energy efficient alternative
- Opportunity to use new substrates
  - Future is uncertain
- Still waiting to scale up SNNs



Difficulty scaling up SNNs

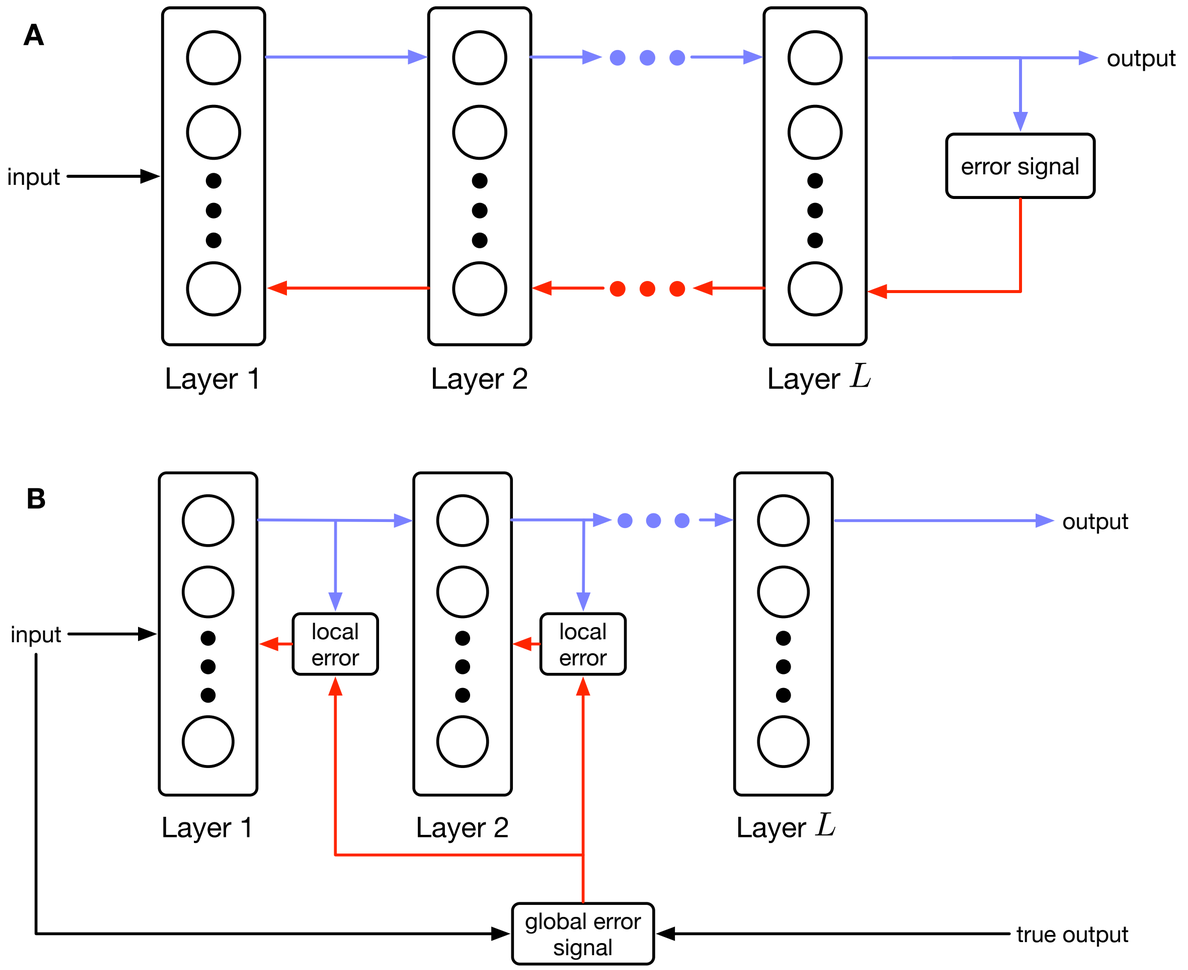
- Closest cousin: back propagation
- Biologically implausible
- Global information overhead



Difficulty scaling up SNNs

- Closest cousin: back propagation
- Biologically implausible
- Global information overhead

- More biologically plausible
- Minimized global information
- Parallelizable



Information bottleneck

- An optimal representation that balances compression and prediction:

$$\min_{P_{T|X}, P_{Y|T}} I(X;T) - \beta I(T;Y)$$

[1] W.-D. K. Ma, J. P. Lewis, and W. B. Kleijn, “The HSIC Bottleneck: Deep Learning without Back-Propagation,” *arXiv:1908.01580 [cs, stat]*, Dec. 2019, Accessed: Jan. 25, 2020. [Online]. Available: http://arxiv.org/abs/1908.01580

Information bottleneck

- An optimal representation that balances compression and prediction:

$$\min_{P_{T|X}, P_{Y|T}} I(X;T) - \beta I(T;Y)$$

- Applied to hidden representation of NN layers [1]:

$$\mathcal{L}_{\text{HSIC}}(X, Y, Z^\ell) = \text{HSIC}(Z^\ell, X) - \lambda \text{HSIC}(Z^\ell, Y) \qquad \forall \ell \in \{1, \ldots, L\}$$

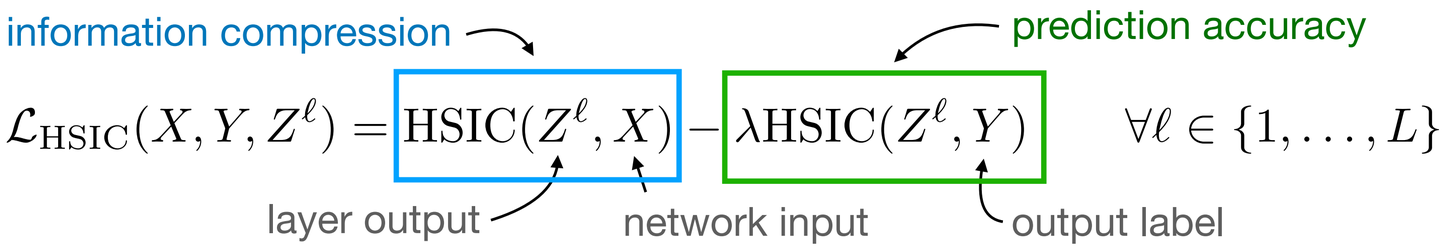
[1] W.-D. K. Ma, J. P. Lewis, and W. B. Kleijn, “The HSIC Bottleneck: Deep Learning without Back-Propagation,” *arXiv:1908.01580 [cs, stat]*, Dec. 2019, Accessed: Jan. 25, 2020. [Online]. Available: http://arxiv.org/abs/1908.01580

Information bottleneck

- An optimal representation that balances compression and prediction:

$$\min_{P_{T|X}, P_{Y|T}} I(X;T) - \beta I(T;Y)$$

- Applied to hidden representation of NN layers [1]:



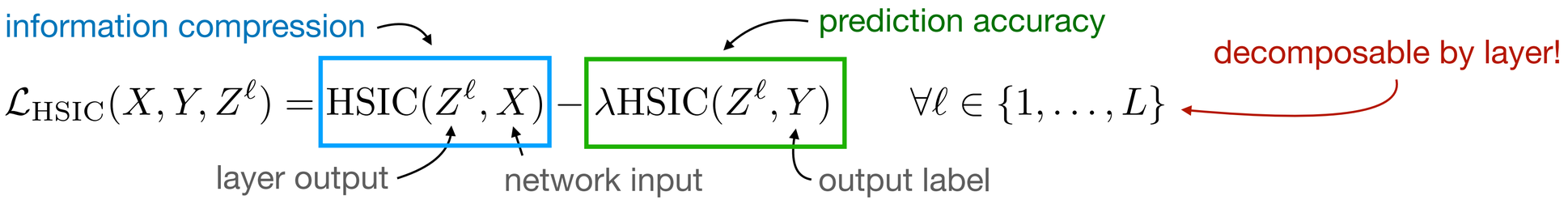
[1] W.-D. K. Ma, J. P. Lewis, and W. B. Kleijn, "The HSIC Bottleneck: Deep Learning without Back-Propagation," *arXiv:1908.01580 [cs, stat]*, Dec. 2019, Accessed: Jan. 25, 2020. [Online]. Available: http://arxiv.org/abs/1908.01580

Information bottleneck

- An optimal representation that balances compression and prediction:

$$\min_{P_{T|X}, P_{Y|T}} I(X;T) - \beta I(T;Y)$$

- Applied to hidden representation of NN layers [1]:



[1] W.-D. K. Ma, J. P. Lewis, and W. B. Kleijn, "The HSIC Bottleneck: Deep Learning without Back-Propagation," *arXiv:1908.01580 [cs, stat]*, Dec. 2019, Accessed: Jan. 25, 2020. [Online]. Available: http://arxiv.org/abs/1908.01580

Information bottleneck

- An optimal representation that balances compression and prediction:

$$\min_{P_{T|X}, P_{Y|T}} I(X;T) - \beta I(T;Y)$$

- Applied to hidden representation of NN layers [1]:

information compression

prediction accuracy

decomposable by layer!

$$\mathcal{L}_{\text{HSIC}}(X, Y, Z^\ell) = \text{HSIC}(Z^\ell, X) - \lambda \text{HSIC}(Z^\ell, Y) \qquad \forall \ell \in \{1, \ldots, L\}$$

layer output

network input

output label

$$\text{HSIC}(X, Y) = (N-1)^{-2} \text{tr}(\mathbf{K_X H K_Y H})$$

$$= \frac{1}{(N-1)^2} \sum_{p=1}^{N} \sum_{q=1}^{N} \bar{k}(\mathbf{x}_p, \mathbf{x}_q) \bar{k}(\mathbf{y}_q, \mathbf{y}_p)$$

$$[\mathbf{K_X}]_{pq} = k(\mathbf{x}_p, \mathbf{x}_q) = \exp\left(-\frac{\|\mathbf{x}_p - \mathbf{x}_q\|^2}{\sigma^2}\right)$$

[1] W.-D. K. Ma, J. P. Lewis, and W. B. Kleijn, "The HSIC Bottleneck: Deep Learning without Back-Propagation," *arXiv:1908.01580 [cs, stat]*, Dec. 2019, Accessed: Jan. 25, 2020. [Online]. Available: http://arxiv.org/abs/1908.01580

Information bottleneck

- An optimal representation that balances compression and prediction:

$$\min_{P_{T|X}, P_{Y|T}} I(X;T) - \beta I(T;Y)$$

- Applied to hidden representation of NN layers [1]:

information compression

prediction accuracy

decomposable by layer!

$$\mathcal{L}_{\text{HSIC}}(X, Y, Z^\ell) = \text{HSIC}(Z^\ell, X) - \lambda \text{HSIC}(Z^\ell, Y) \qquad \forall \ell \in \{1, \ldots, L\}$$

layer output

network input

output label

$$\text{HSIC}(X, Y) = (N-1)^{-2} \text{tr}(\mathbf{K_X H K_Y H})$$

$$= \frac{1}{(N-1)^2} \sum_{p=1}^{N} \sum_{q=1}^{N} \bar{k}(\mathbf{x}_p, \mathbf{x}_q) \bar{k}(\mathbf{y}_q, \mathbf{y}_p)$$

$$[\mathbf{K_X}]_{pq} = k(\mathbf{x}_p, \mathbf{x}_q) = \exp\left(-\frac{\|\mathbf{x}_p - \mathbf{x}_q\|^2}{\sigma^2}\right)$$

[1] W.-D. K. Ma, J. P. Lewis, and W. B. Kleijn, "The HSIC Bottleneck: Deep Learning without Back-Propagation," *arXiv:1908.01580 [cs, stat]*, Dec. 2019, Accessed: Jan. 25, 2020. [Online]. Available: http://arxiv.org/abs/1908.01580

Applying the HSIC bottleneck to SNNs

- Feedforward network of leaky-integrate-and-fire neurons:

$$\tau_{\text{ff}} \frac{\mathrm{d}\mathbf{u}^{\ell}}{\mathrm{d}t} = -\mathbf{u} + \mathbf{W}^{\ell}\mathbf{z}^{\ell-1}$$
$$\mathbf{z}^{\ell} = \tanh(\mathbf{u}^{\ell}) + \zeta$$

- Gradient descent rule on HSIC objective:

$$\Delta[\mathbf{W}^{\ell}]_{ij} \propto \frac{\partial \mathcal{L}_{\text{HSIC}}(X, Y, Z^{\ell})}{\partial \mathbf{W}^{\ell}}$$

Making the weight update plausible

- Assume that the past output does not depend on the current weights

Making the weight update plausible

- Assume that the past output does not depend on the current weights

$$\Delta[W^\ell]_{ij} \propto \beta_{ij}\xi_i$$

local

$$\begin{aligned}\beta_{ij} &= \partial z_0^\ell / \partial [W^\ell]_{ij} \\ &= (1 - ([z_0^\ell]_i)^2)[z_0^{\ell-1}]_j\end{aligned}$$

depends on N samples

global

$$\xi_i = \sum_{p=0}^{-(N-1)} [\bar{k}(x_0, x_p) - \gamma \bar{k}(y_0, y_p)]\bar{\alpha}(z_p^\ell)$$

Making the weight update plausible

- Assume that the past output does not depend on the current weights

$$\Delta[W^\ell]_{ij} \propto \beta_{ij}\xi_i$$

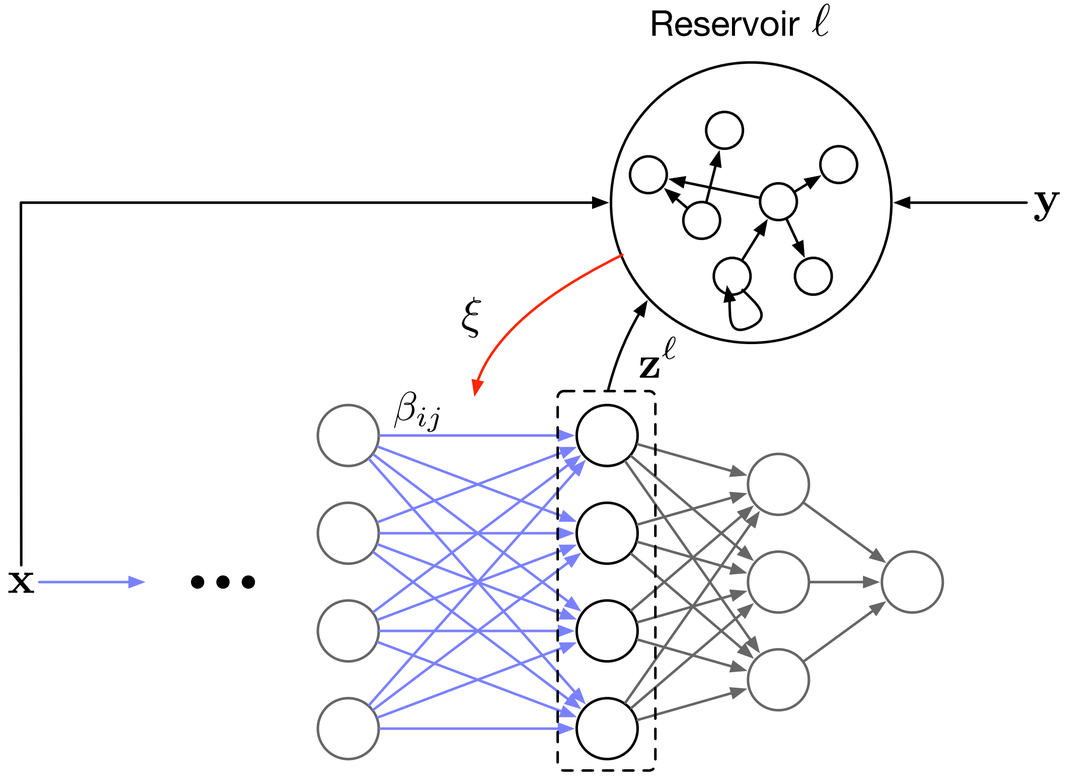
local

$$\begin{aligned}\beta_{ij} &= \partial z_0^\ell / \partial [W^\ell]_{ij} \\ &= (1 - ([z_0^\ell]_i)^2)[z_0^{\ell-1}]_j\end{aligned}$$

depends on N samples

global

$$\xi_i = \sum_{p=0}^{-(N-1)} [\bar{k}(x_0, x_p) - \gamma \bar{k}(y_0, y_p)]\bar{\alpha}(z_p^\ell)$$



Making the weight update plausible

- Assume that the past output does not depend on the current weights

$$\Delta[W^\ell]_{ij} \propto \beta_{ij}\xi_i$$

local

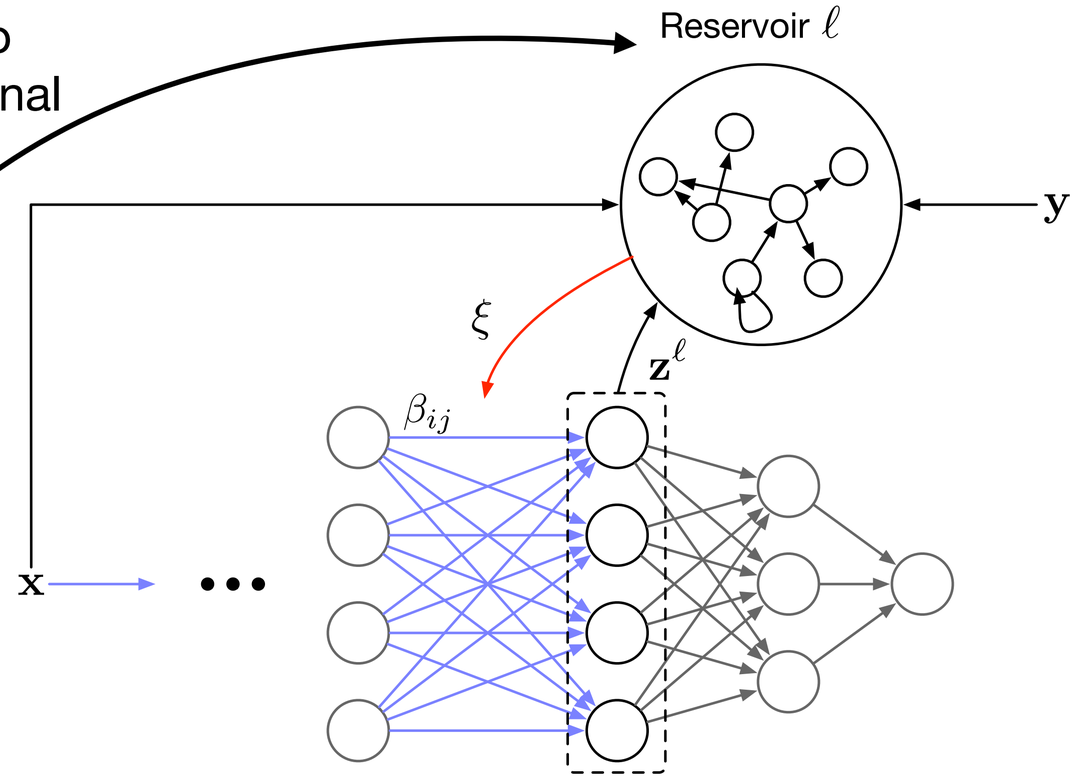
$$\beta_{ij} = \partial z_0^\ell / \partial [W^\ell]_{ij}$$
$$= (1 - ([z_0^\ell]_i)^2)[z_0^{\ell-1}]_j$$

depends on N samples

global

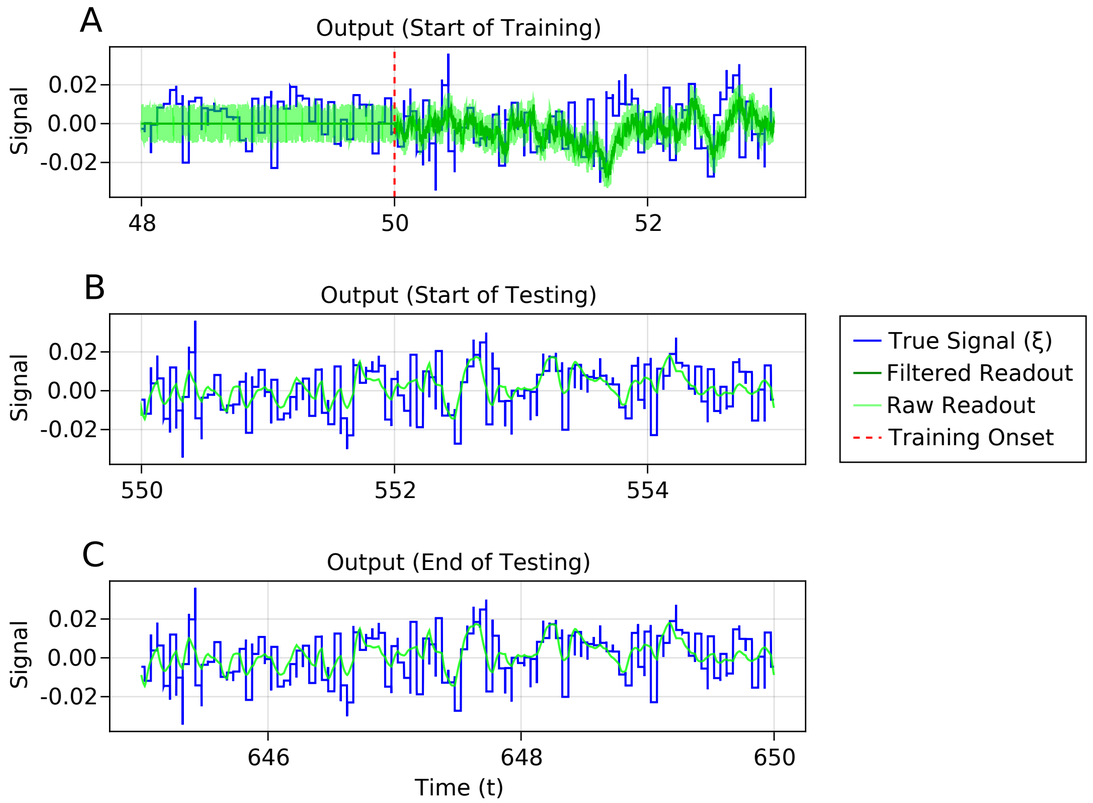
$$\xi_i = \sum_{p=0}^{-(N-1)} [\bar{k}(x_0, x_p) - \gamma \bar{k}(y_0, y_p)]\bar{\alpha}(z_p^\ell)$$

teach reservoir to
compute global signal



Training the reservoir

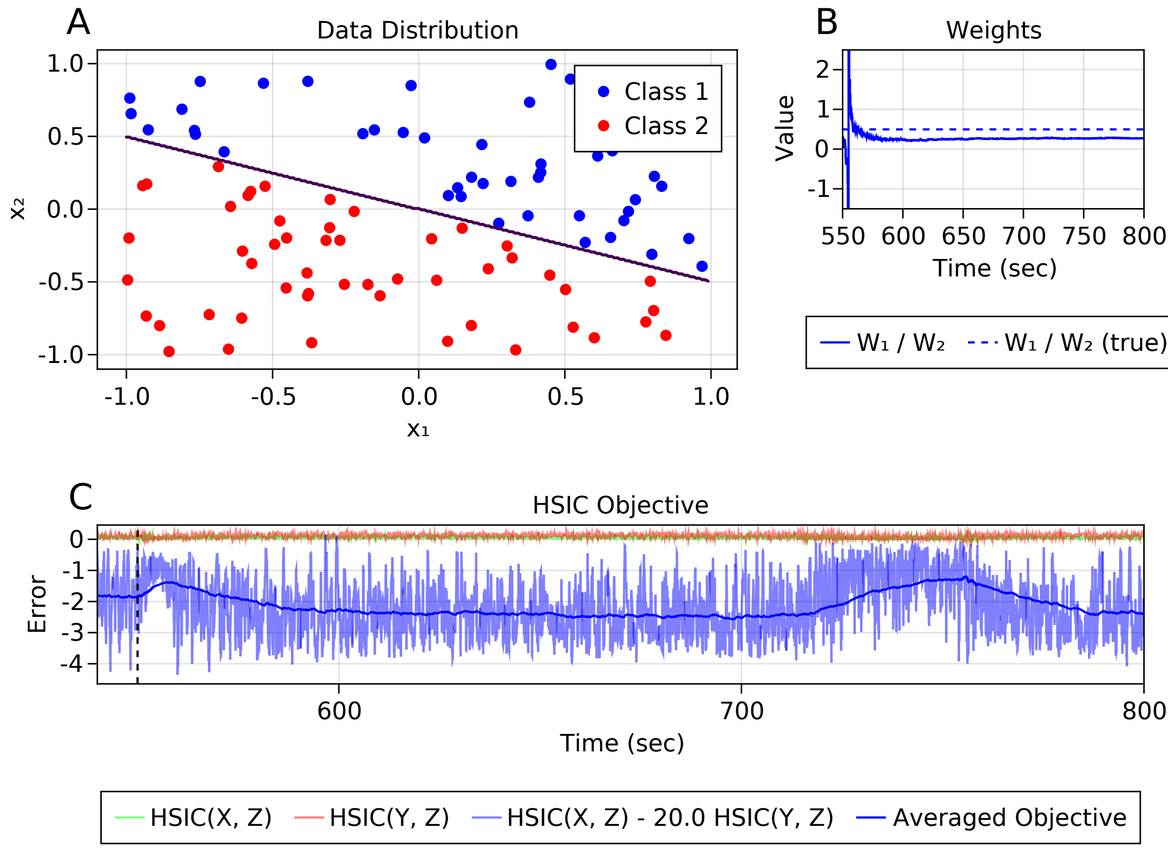
- Use three-factor Hebbian rule from Hoerzer, Legenstein, and Maass [1]
- Trained on random data a priori



[1] G. M. Hoerzer, R. Legenstein, and W. Maass, “Emergence of Complex Computational Structures From Chaotic Neural Networks Through Reward-Modulated Hebbian Learning,” *Cerebral Cortex*, vol. 24, no. 3, pp. 677–690, Mar. 2014, doi: 10.1093/cercor/bhs348.

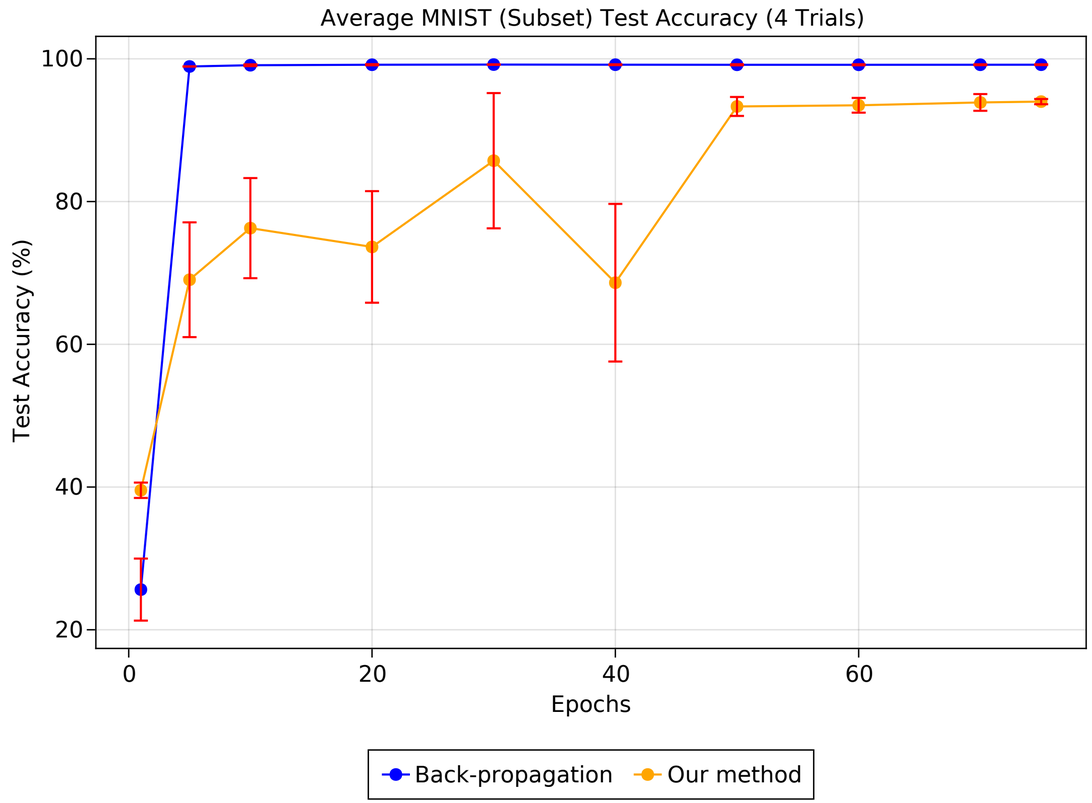
Small scale experiments

- Simple binary classification tasks
- Reservoir trained a priori (then held constant)
- Tested with multiple layers and non-linear decision boundaries



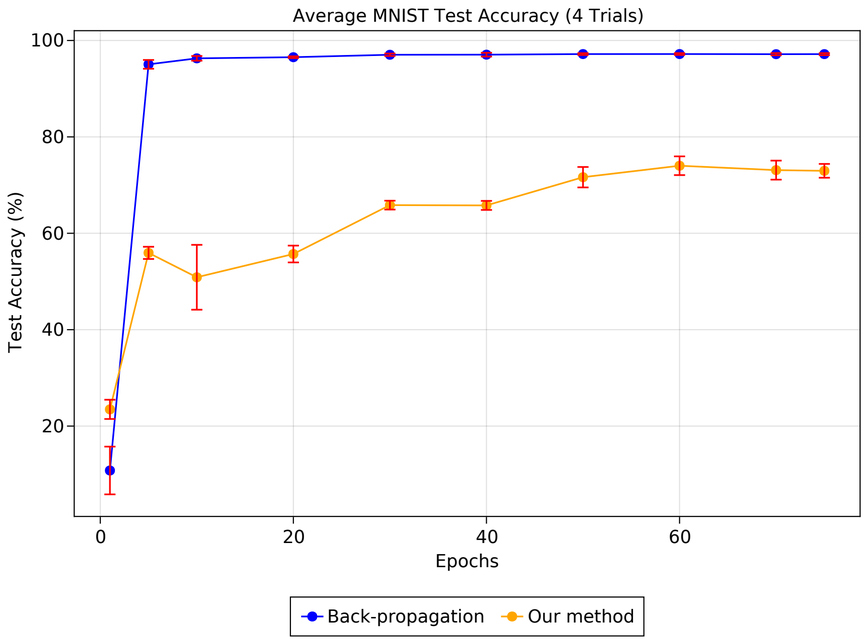
Large scale experiments

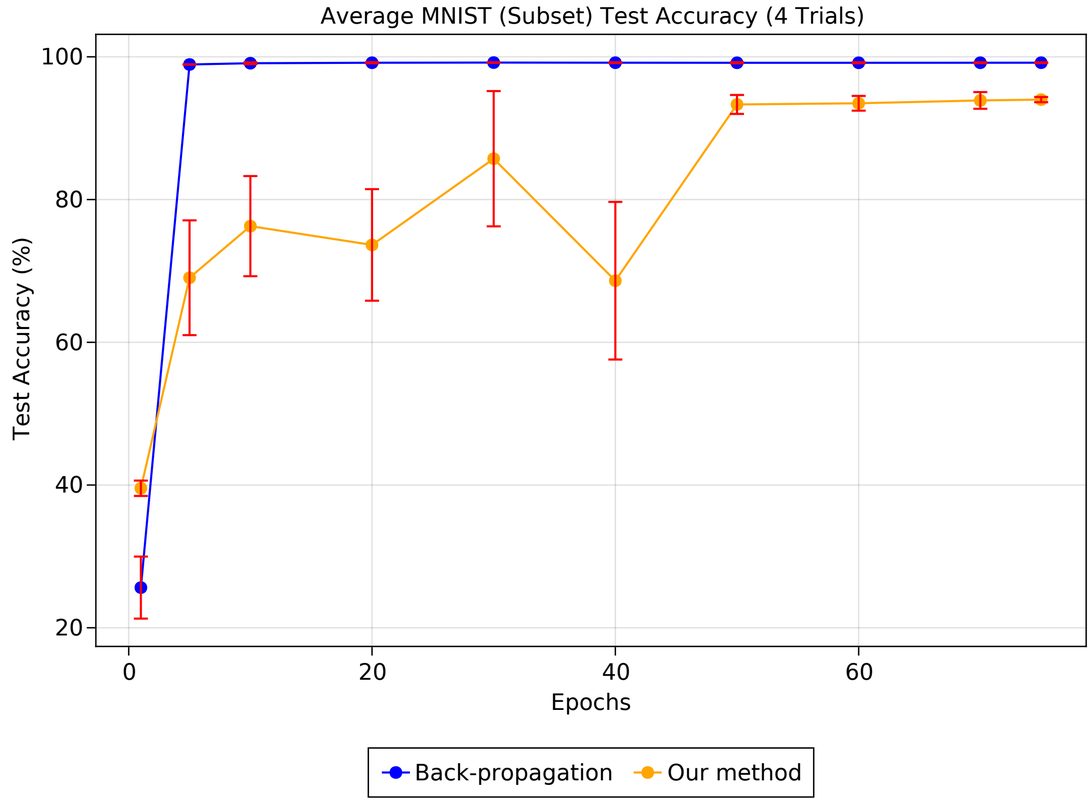
- Tested on subset of MNIST classes
- Simulations are slow due to iterating the dataset one sample at a time



Large scale experiments

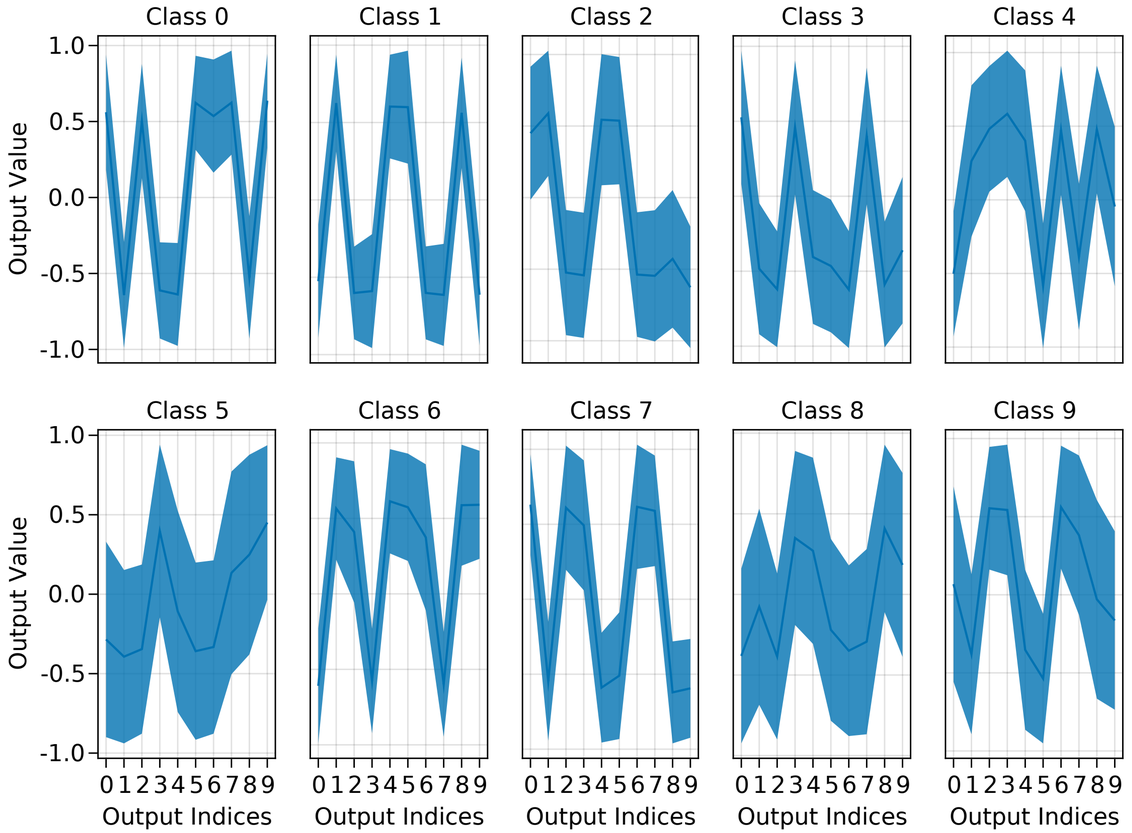
- Tested on subset of MNIST classes
- Simulations are slow due to iterating the dataset one sample at a time





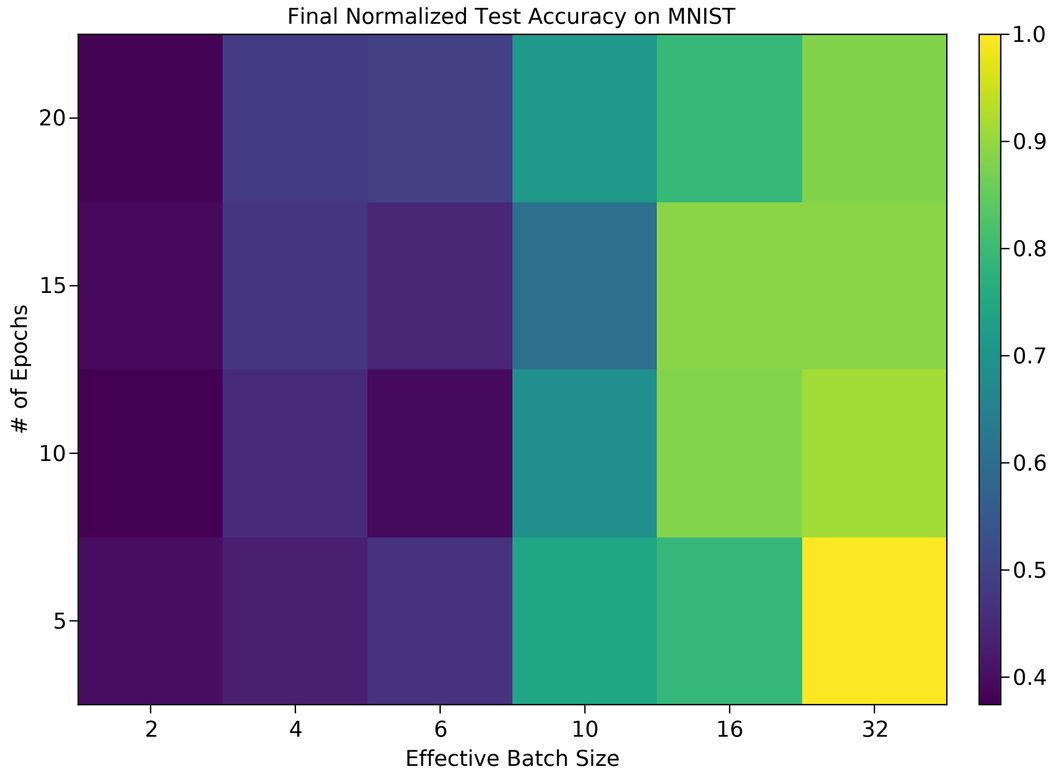
Large scale experiments

- HSIC objective learns unique outputs (not necessarily the same as labels)



Large scale experiments

- Varying memory capacity by adjusting effective batch size



Conclusions & future work

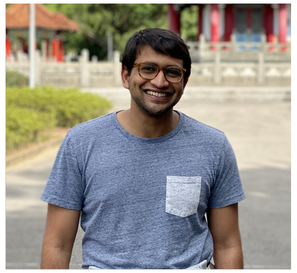
- Neuromorphic computing and SNNs are a promising hardware platform for ML-adjacent applications
- Training SNNs at scale continues to remain a challenge
- Layer-wise objectives are an effective alternative to back propagation
  - Our learning rule based on the information bottleneck
  - Couples synaptic updates with short-term memory
- Future work
  - Scaling our rule up to larger datasets
  - Using alternate memory models to replace the reservoir
  - Hardware implementations on neuromorphic substrates

Conclusions

Conclusions

- End of transistor scaling demands a new approach to computing
- Long term vision: standard and unconventional co-processors
- Bitstream computing is a near-term platform
  - Use compilers to solve programming issues
- Neuromorphic computing is a more promising long-term platform
  - Major issue is scaling up learning
  - Many different hardware platforms

My work


my collaborators

Preprints

R. S. Raju, **K. Daruwalla**, M. Lipasti, *Accelerating Deep Learning with Dynamic Data Pruning*, preprint under review, November, 2021. https://arxiv.org/abs/2111.12621.

K. Daruwalla, M. Lipasti, *Information Bottleneck-Based Hebbian Learning Rule Naturally Ties Working Memory and Synaptic Updates*, preprint, September, 2021. https://arxiv.org/abs/2111.13187.

Conference Publications

K. Daruwalla, H. Zhuo, C. Schulz, M. Lipasti, *BitBench: A Benchmark for Bitstream Computing*, Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '19), June 23, 2019. http://dl.acm.org/citation.cfm?doid=3316482.3326355.

Journal Publications

K. Daruwalla, H. Zhuo, R. Shukla, M. Lipasti, *BitSAD v2: Compiler Optimization and Analysis for Bitstream Computing*, ACM Transcations on Architecture and Code Optimization (TACO), Vol. 16, Iss. 4, No. 43. November 2019. https://dl.acm.org/citation.cfm?id=3364999.

K. Daruwalla, N. Olivero, A. Pluger, S. Rao, D.W. Chang, M. Simoni, *A quantitative analysis of the performance of computing architectures used in neural simulations*, Journal of Neuroscience Methods, Vol. 311. 2019, Pg. 57-66. http://www.sciencedirect.com/science/article/pii/S0165027018303017.

Workshop Publications

K. Daruwalla, H. Zhuo, M. Lipasti, *BitSAD: A Domain-Specific Language for Bitstream Computing*, First ISCA Workshop on Unary Computing, June 2019.

K. Daruwalla, M. Lipasti, *Resource Efficient Navigation Using Bitstream Computing*, First ISCA Workshop on Unary Computing, June 2019.