Hardware-Aware, Scalable, Combinatorial Optimization on a Boltzmann Machine

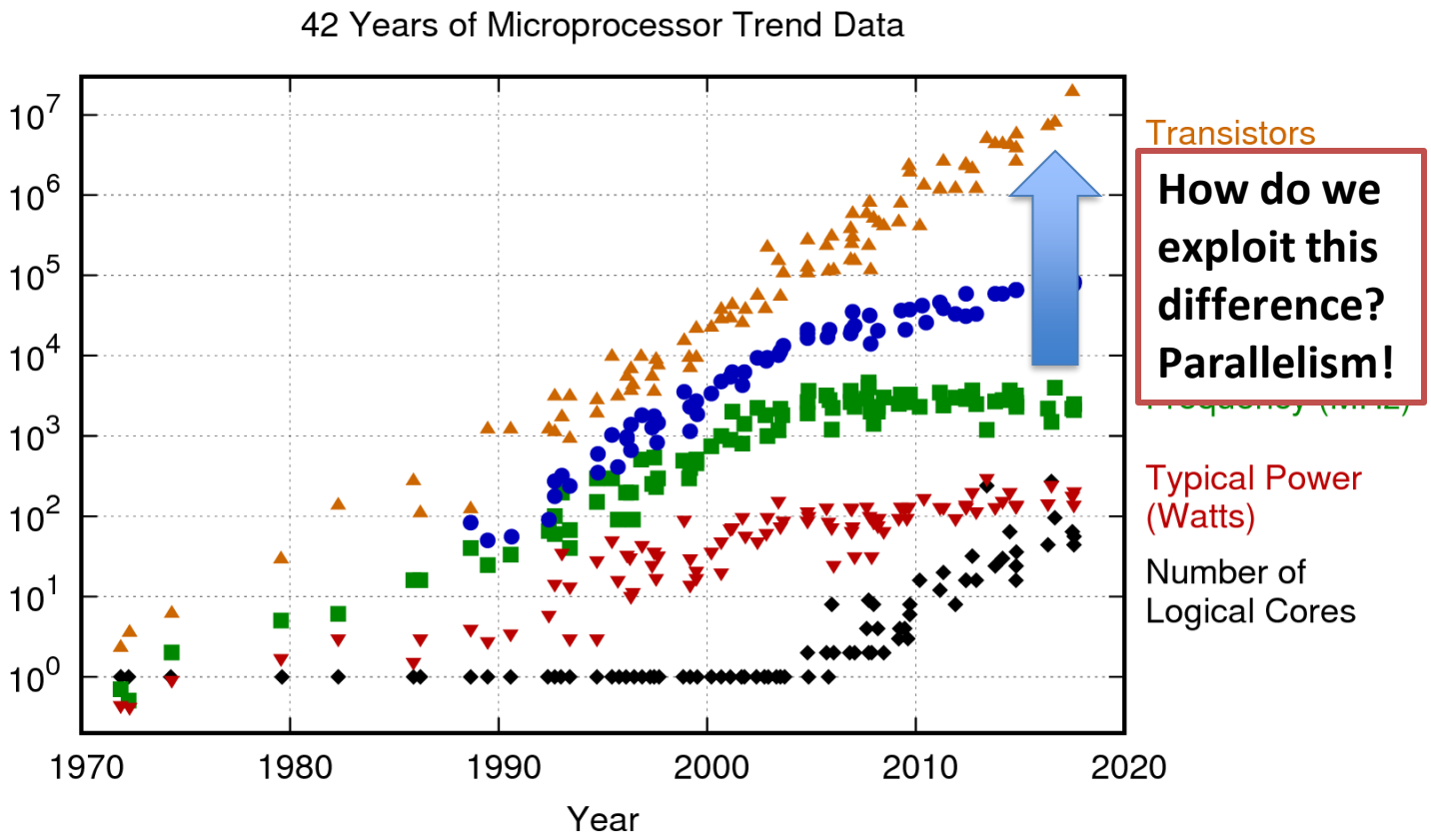
Saavan Patel and Prof. Sayeef Salahuddin
Department of Electrical Engineering and Computer Science
University of California, Berkeley

Berkeley | EECS
Electrical Engineering and Computer Sciences

Outline

1. The Future of Parallelism
2. Ising Model and Optimization Problems
2. Boltzmann Machines
2. Software Approach
3. Hardware Approach

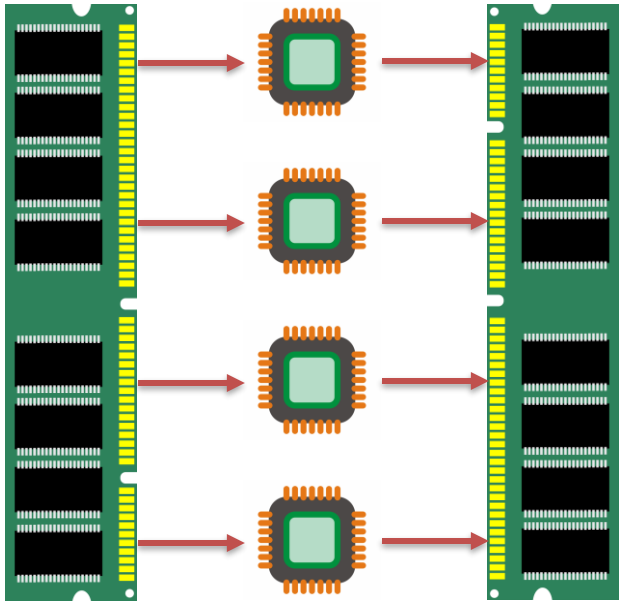
40+ years of Moore's Law



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

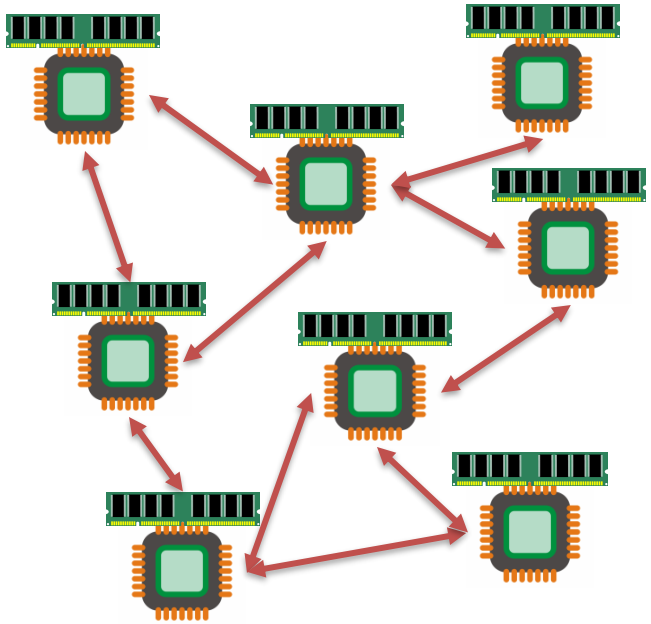
Standard vs. Intrinsic Parallelism

Standard Parallelism



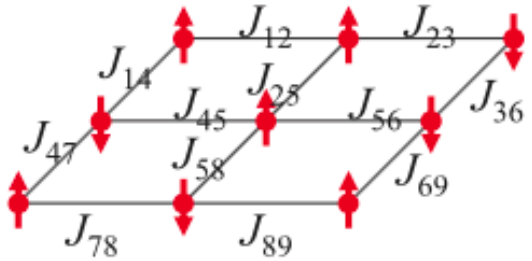
Centralized
Clock Driven
Minimally Scalable

Intrinsic Parallelism

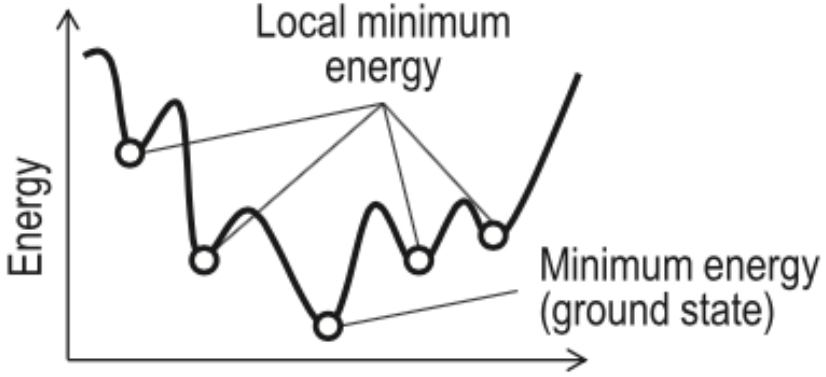


Decentralized
Event Driven
Highly Scalable

Ising Model Computation



$$H = -\sum_{\langle i,j \rangle} J_{ij}\sigma_i\sigma_j - \sum_j h_j\sigma_j$$



Minimizing Ising Model Energy

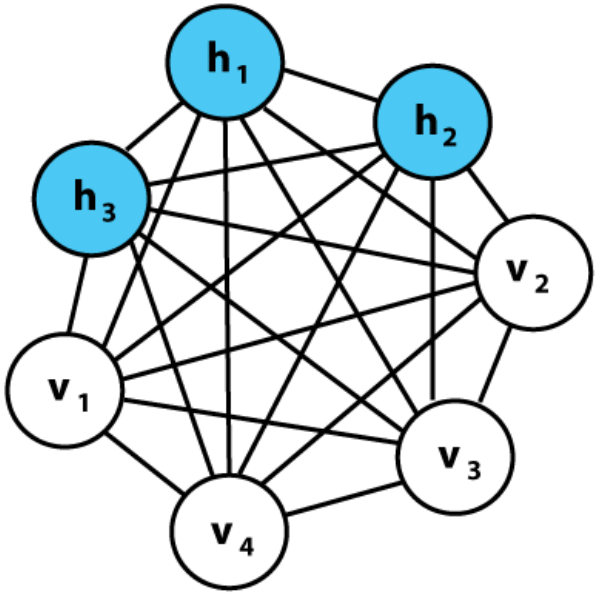
NP-Hard Combinatorial Optimization

Many Hard Problems

- Integer Factorization
- Travelling Salesman
- Boolean SAT
- MAX-CUT
- many more

Yamaoka, et al. *IEEE JSSC* 2016

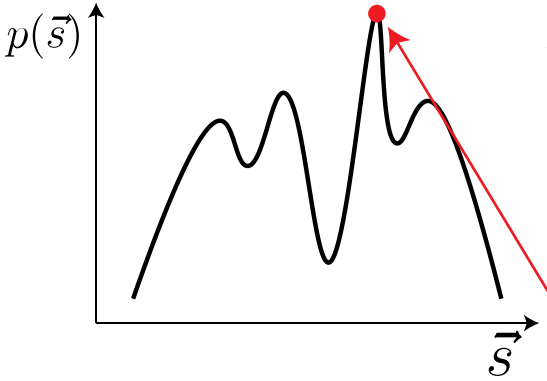
Probabilistic Ising Model (Boltzmann Machines)



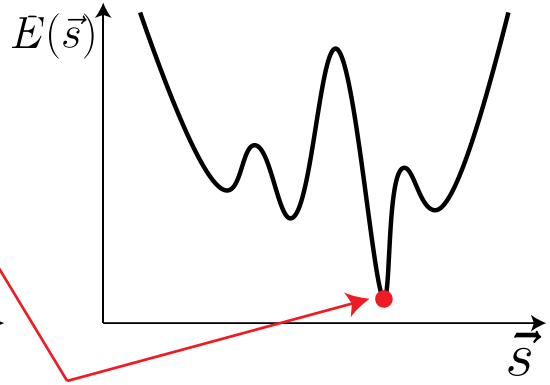
Boltzmann Probability Law

$$p(\vec{s}) = \frac{1}{Z} e^{-E(\vec{s})} \qquad E(\vec{s}) = -\vec{s}^T W \vec{s} + \vec{b}^T \vec{s}$$

Boltzmann Probabilities



Ising Model Energies



Ground State

Low Energy (correct) states correspond to high probability states

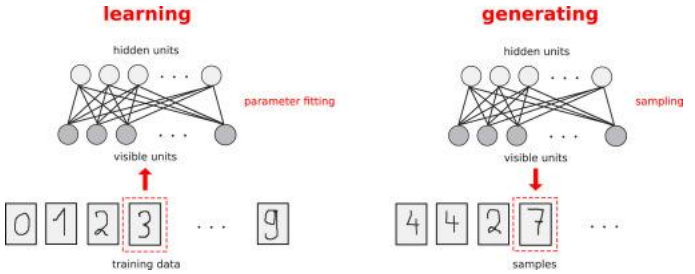
Restricted Boltzmann Machine (RBM)

Individual Neuron Probability

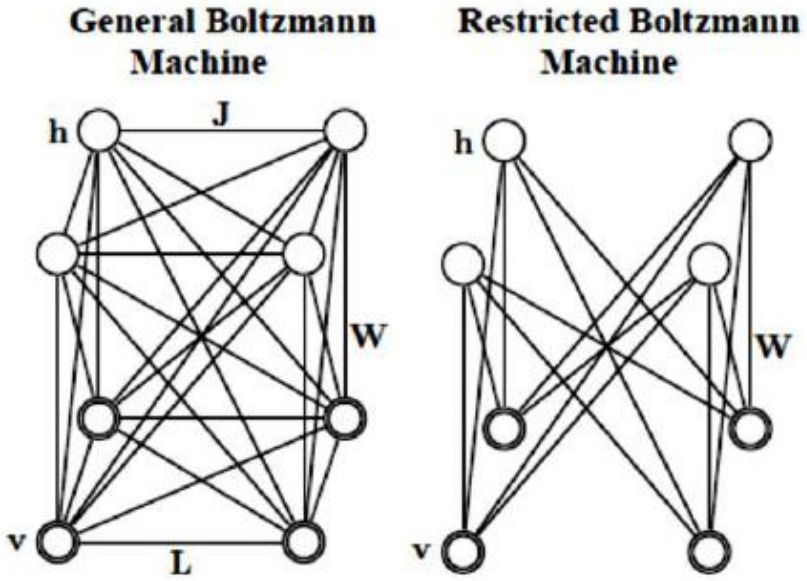
$$p(v_j = 1|\vec{h}) = \sigma(W^T\vec{h} + b_j)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Machine Learning Approach

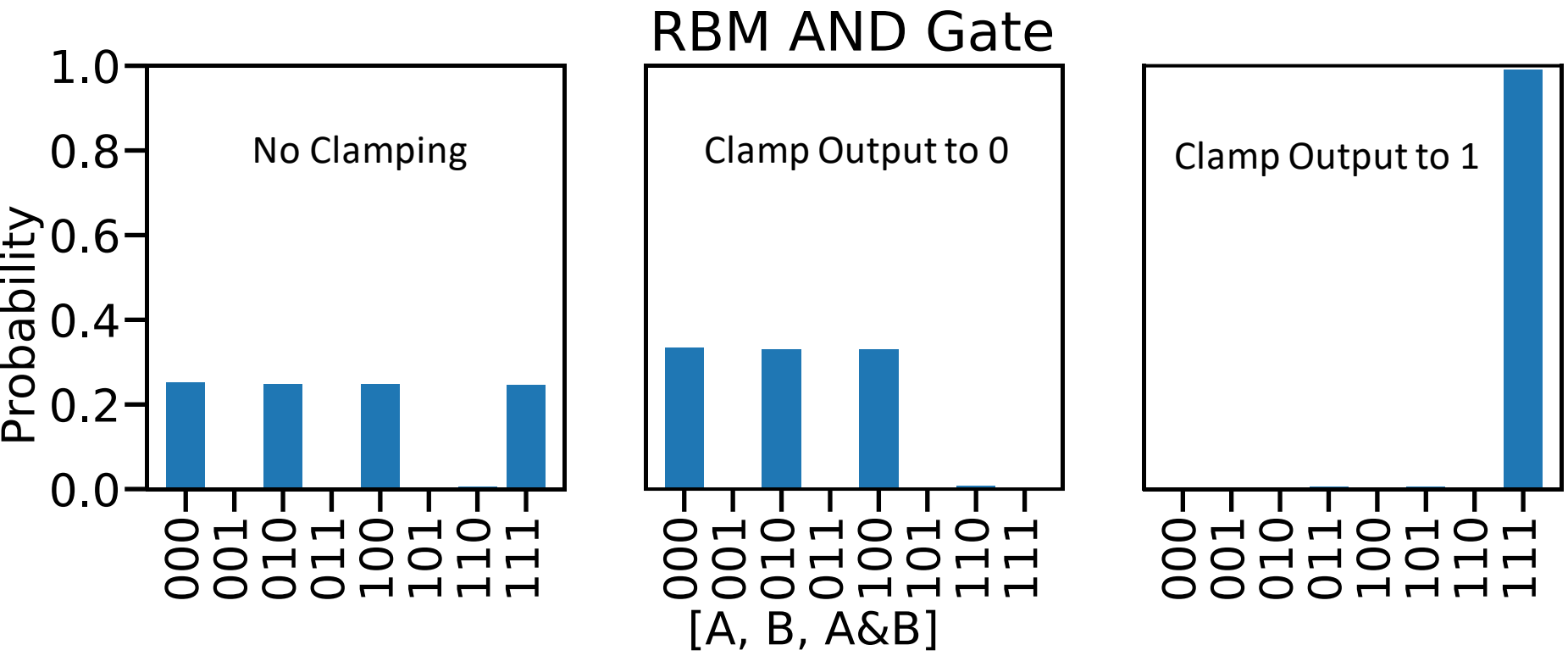


Network can be trained to model an arbitrary distribution

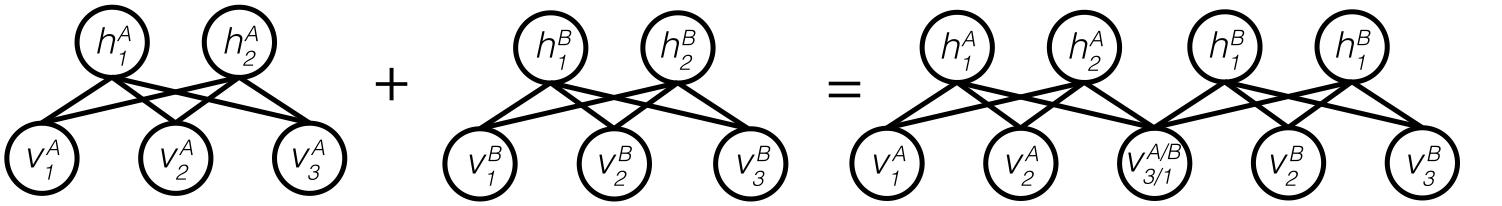


Decreasing number of connections decreases hardware cost without reducing representational power

AND Gate Example



Merging RBMs



$W^A \in \mathbb{R}^{3\times 2}$

$p(v^A, h^A) \propto e^{-E(v^A, h^A)}$

$W^B \in \mathbb{R}^{3\times 2}$

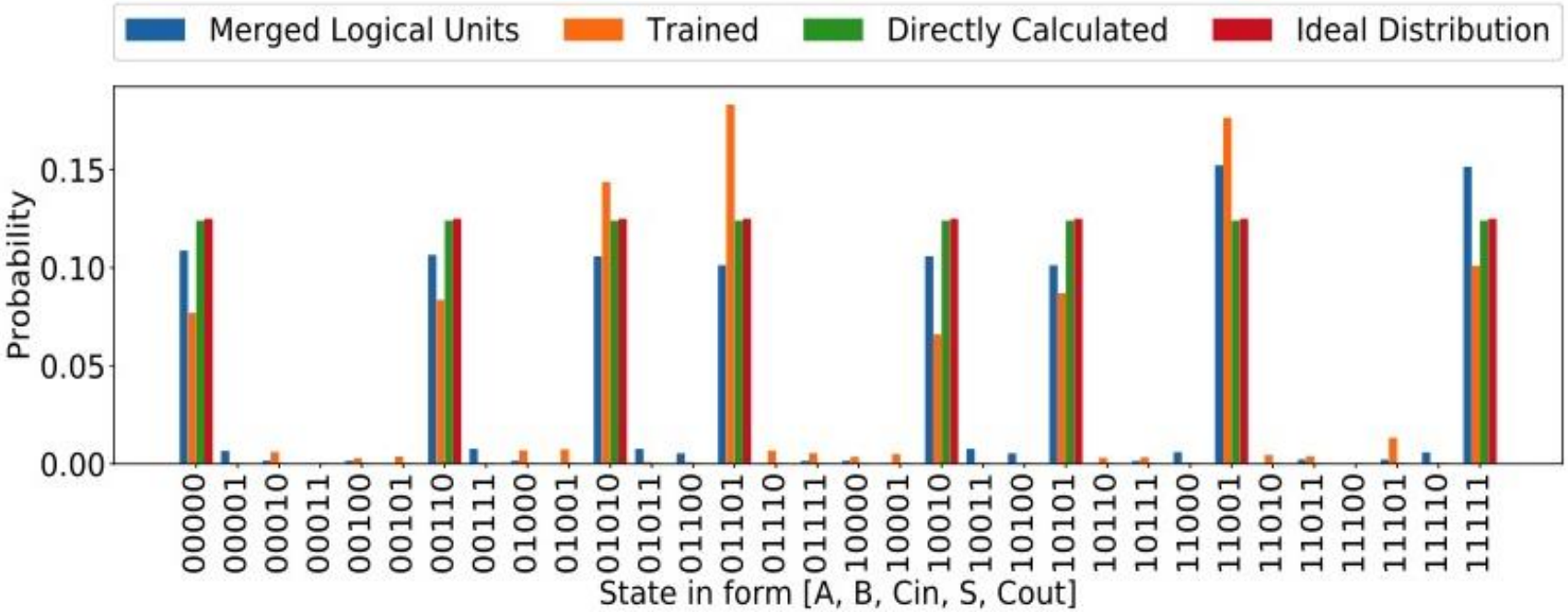
$p(v^B, h^B) \propto e^{-E(v^B, h^B)}$

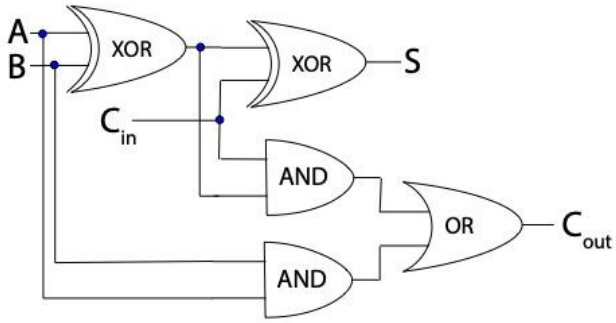
$W^{A+B} \in \mathbb{R}^{5\times 4}$

$p(v^{A+B}, h^{A+B}) \propto e^{-(E(v^A, h^A)+E(v^B, h^b)} \propto p(v^A, h^A)p(v^B, h^B)$

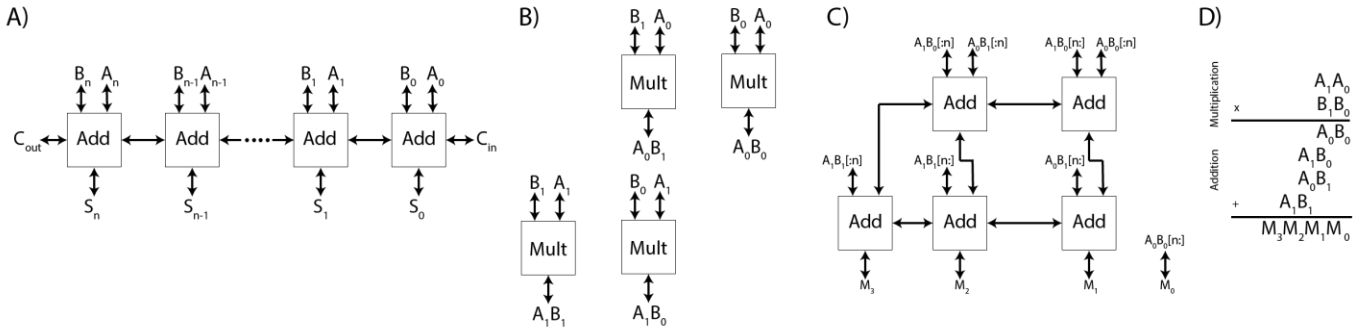
$$p_{A+B}(v,h) = \frac{1}{Z_{A+B}} e^{-E_{A+B}(v,h)} \propto p_A(v_A, h_A) p_B(v_B, h_B)$$

Merging a Full Adder

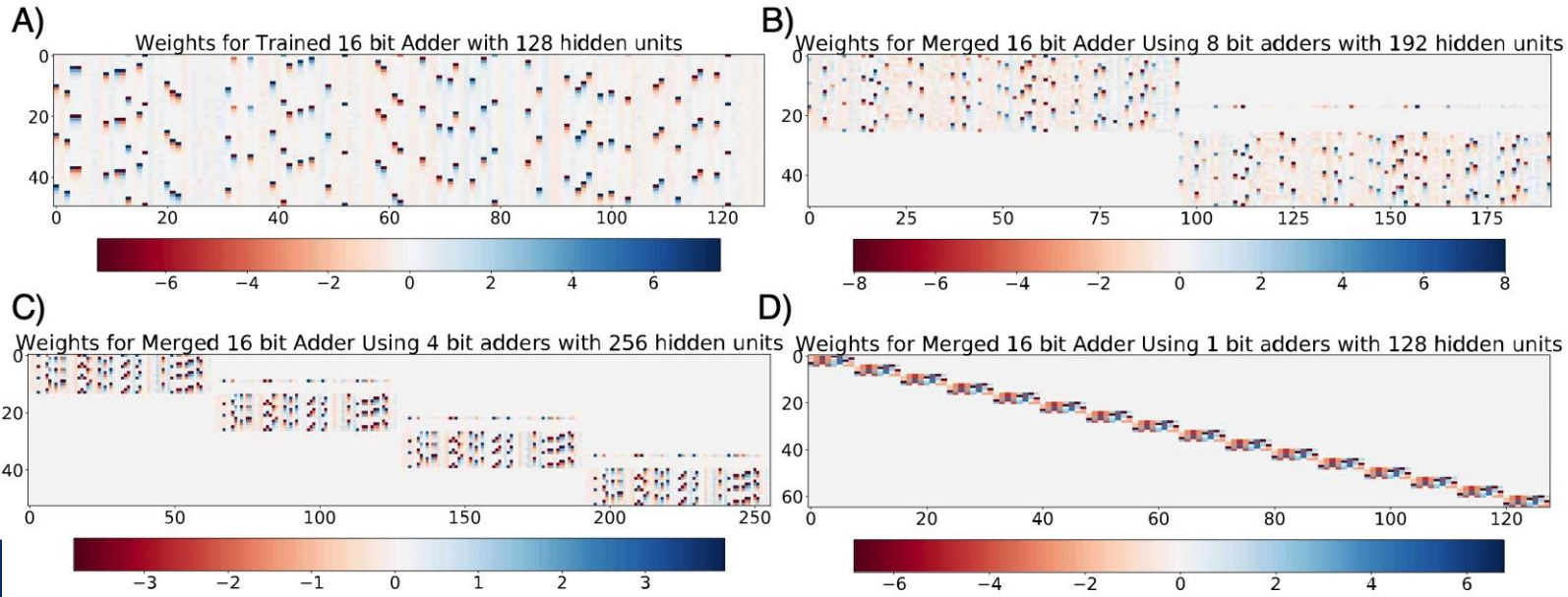




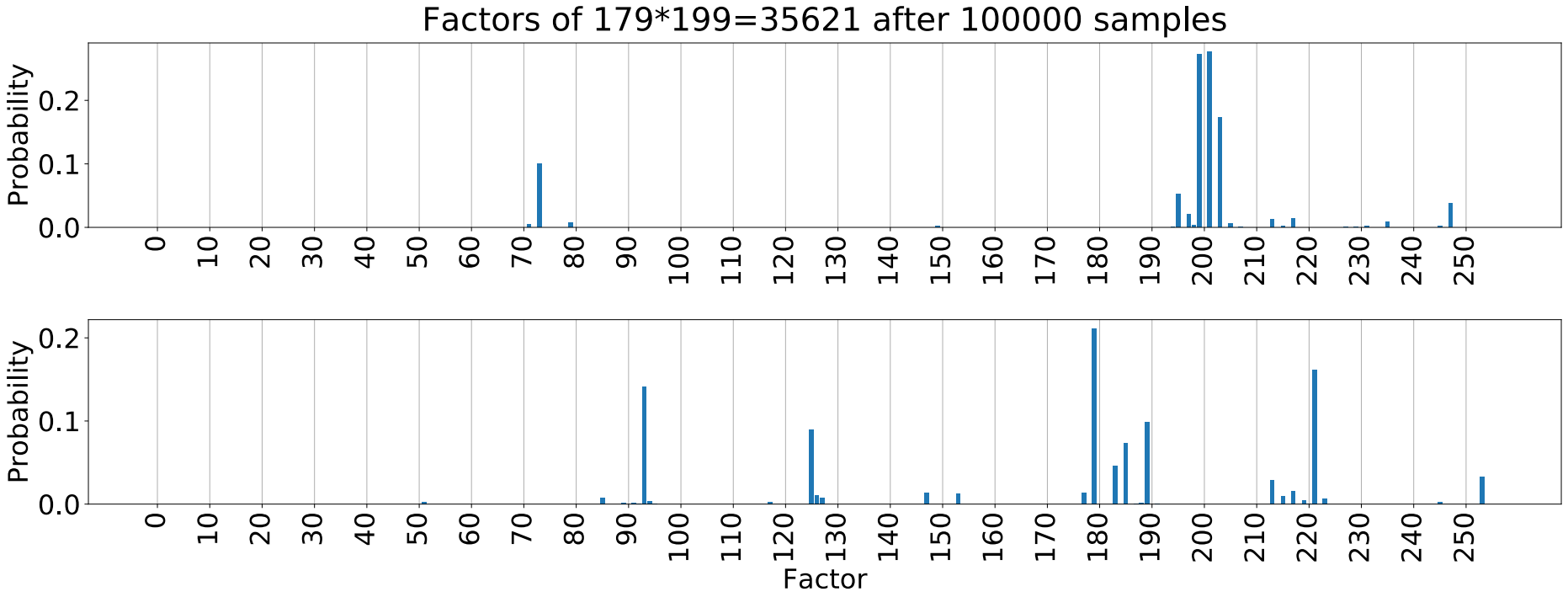
Merging creates sparsity



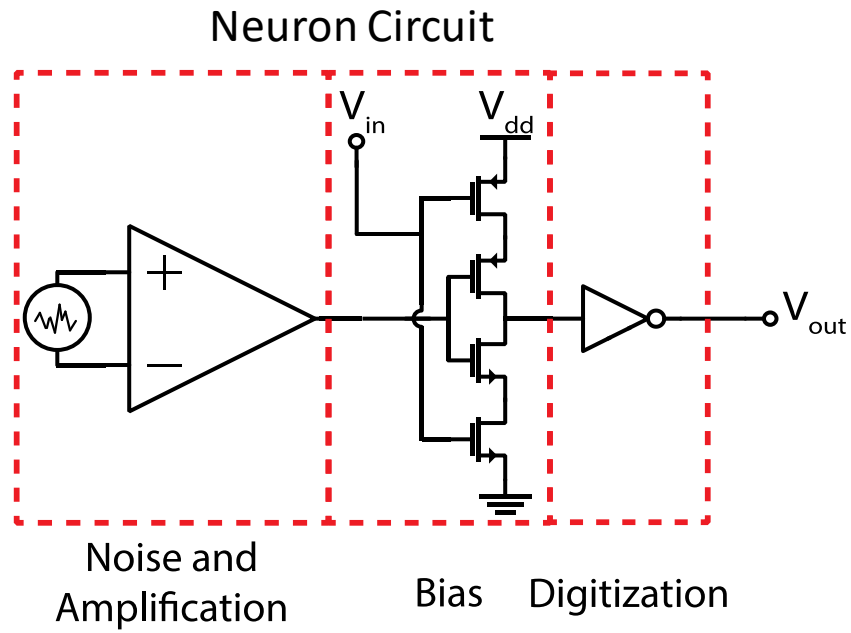
By merging a 16 bit adder together using 1 bit units, sparsity increases by 90%

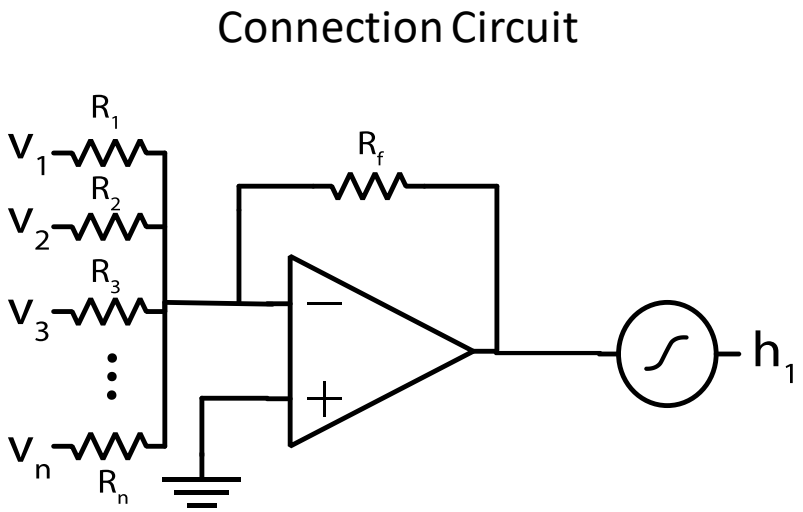


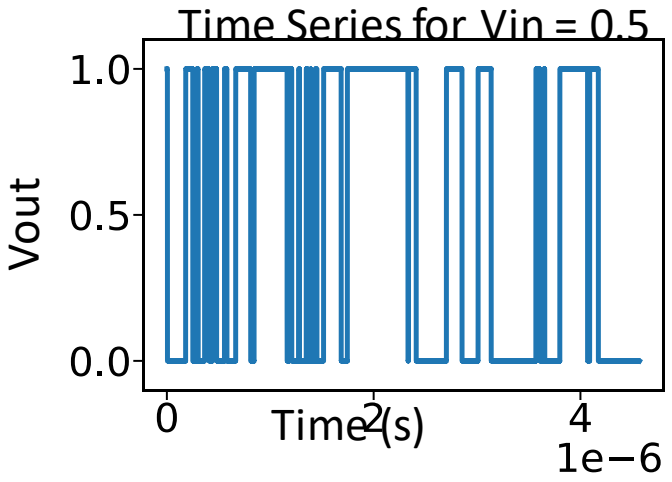
Solving 16 bit factorization Tasks

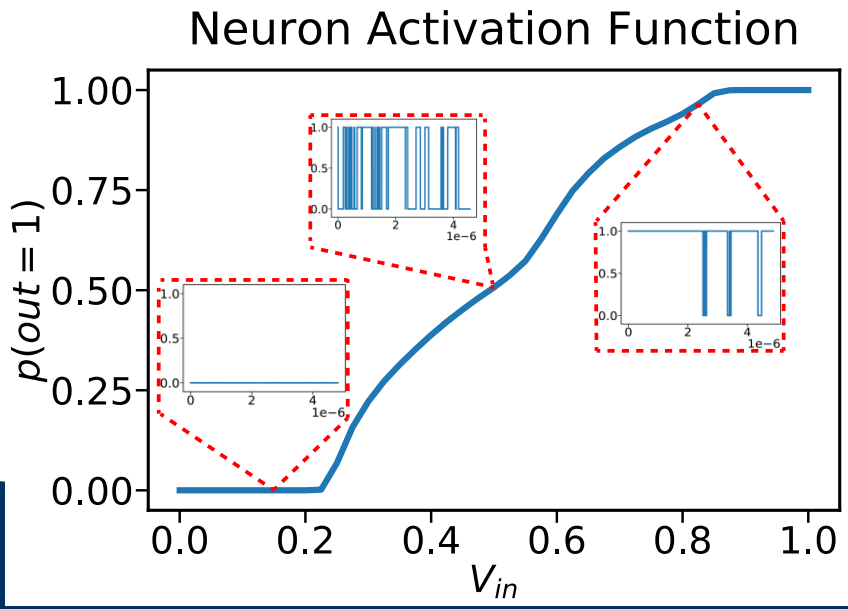


90nm CMOS Neuron

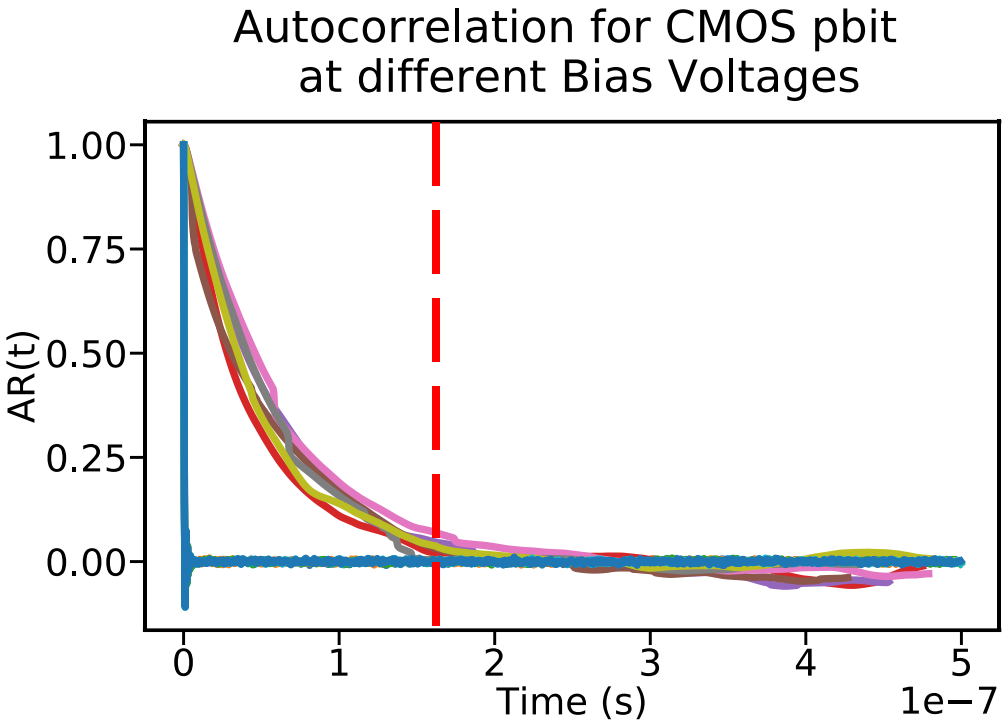








Autocorrelation & Speed



$$AR(k) = \frac{\mathbb{E}[(X_t - \mu_t)(X_{t+k} - \mu)]}{\sigma_X^2}$$

When AR(k) = 0, samples are uncorrelated (independent)

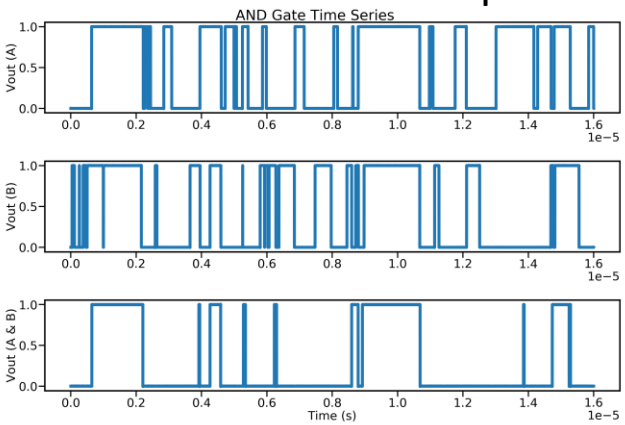
Autocorrelation sets *minimum* sampling time

Minimum sampling frequency ~ 7Mhz

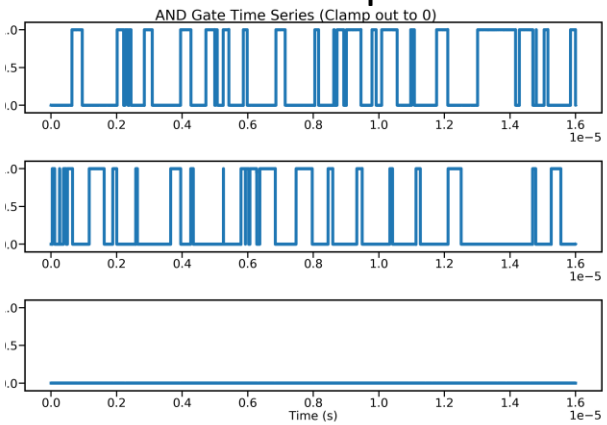
Will improve with smaller process nodes

AND Gate in SPICE

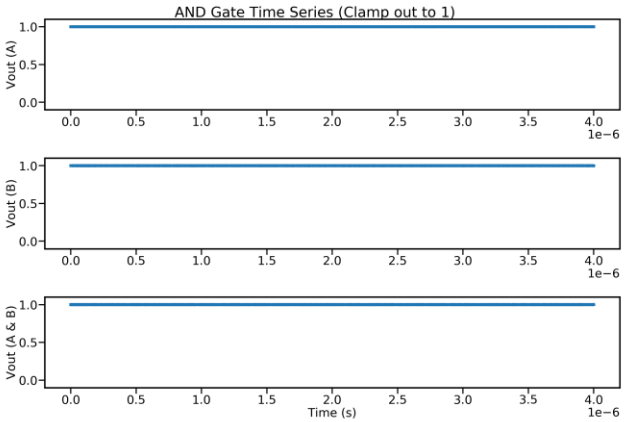
AND Gate No Clamp



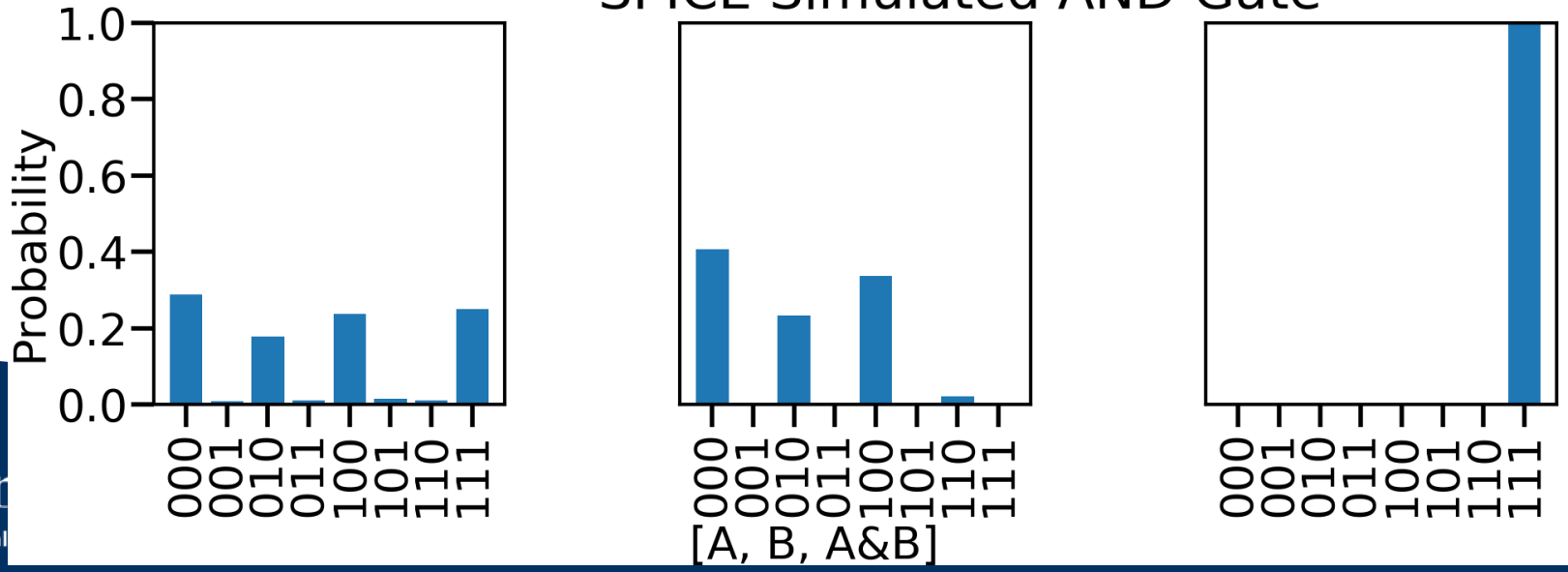
AND Gate Clamp Out to 0



AND Gate Clamp Out to 1

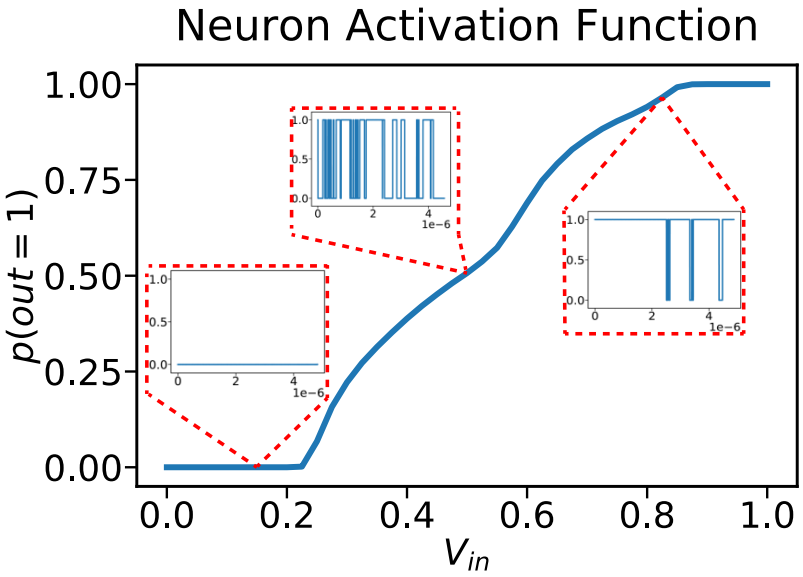


SPICE Simulated AND Gate

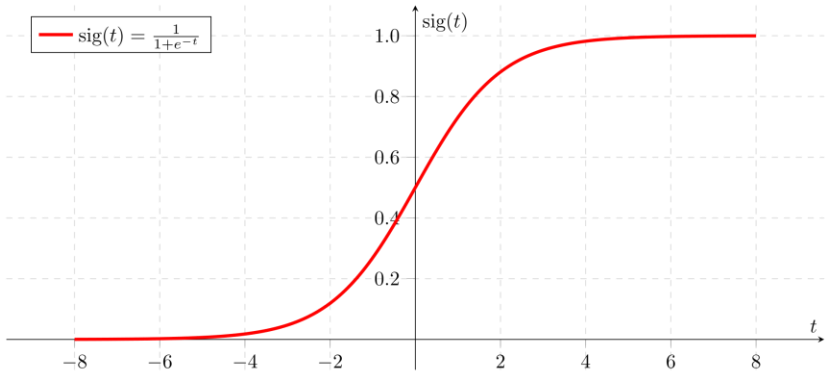


Activation Function

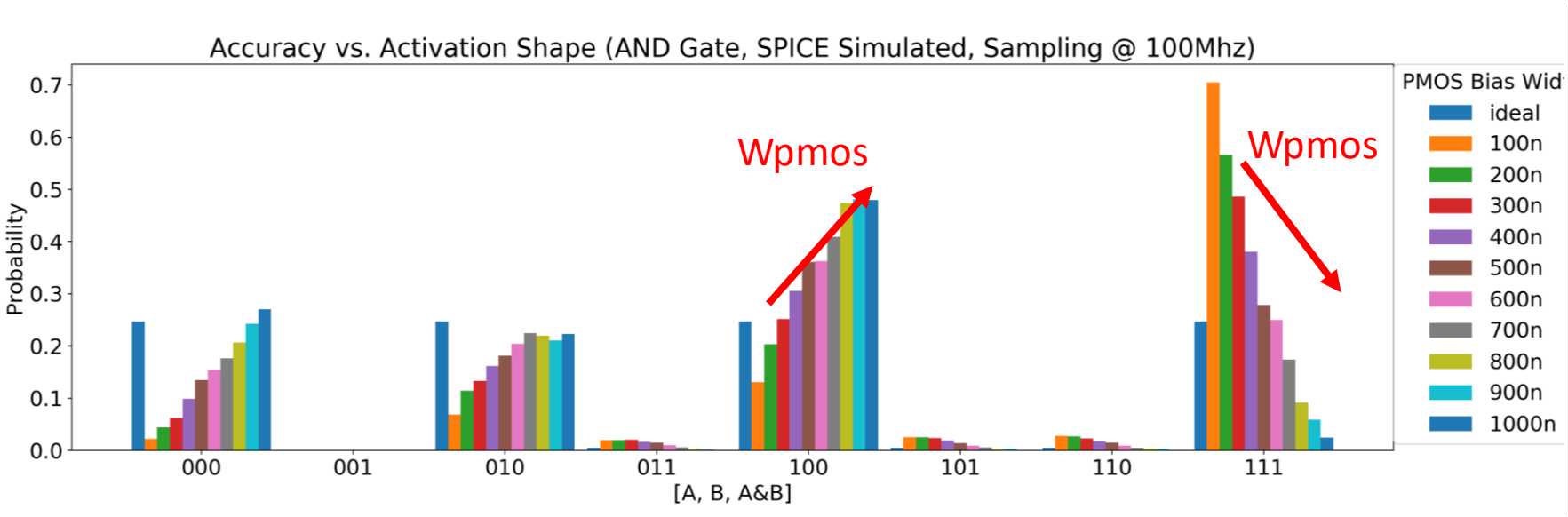
SPICE Extracted Activation Function

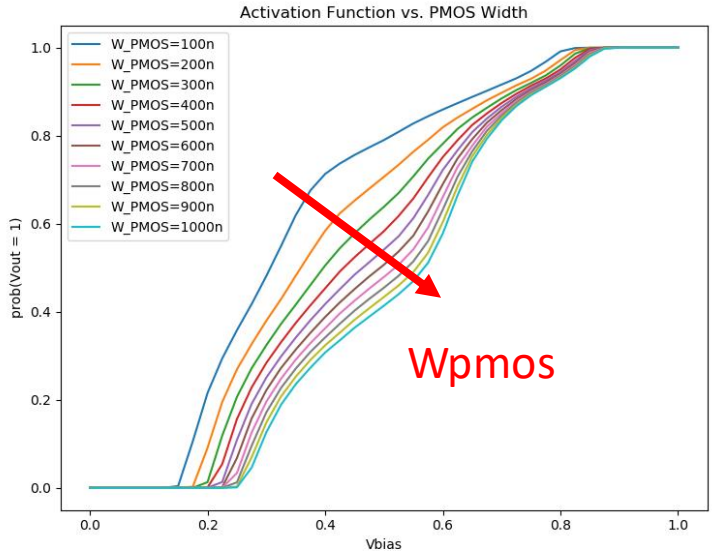


Exact Activation Function

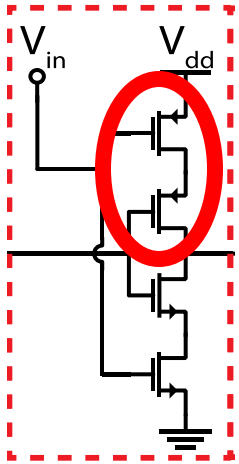


Accuracy on AND gate





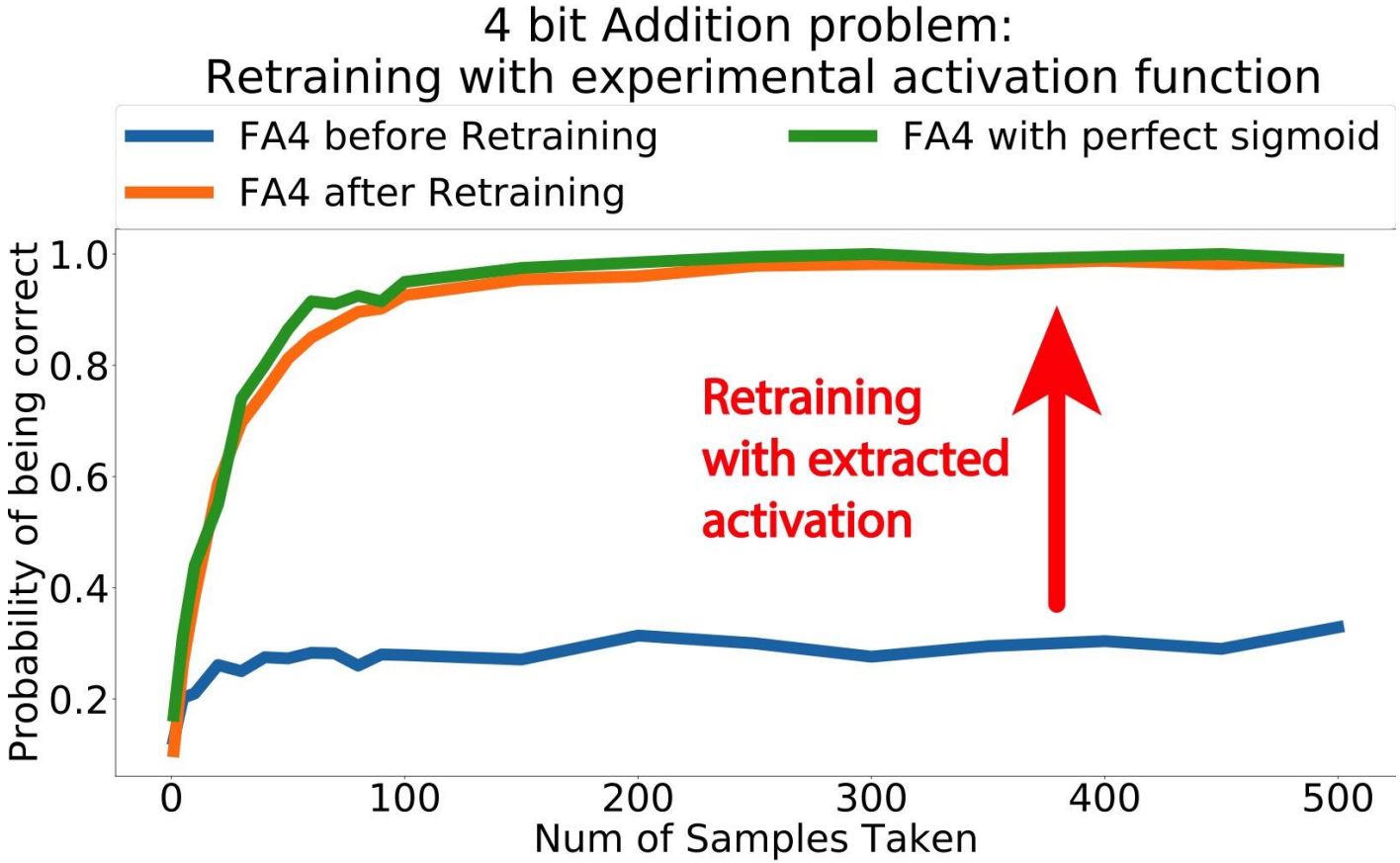
Bias circuit produces non-ideal sigmoid



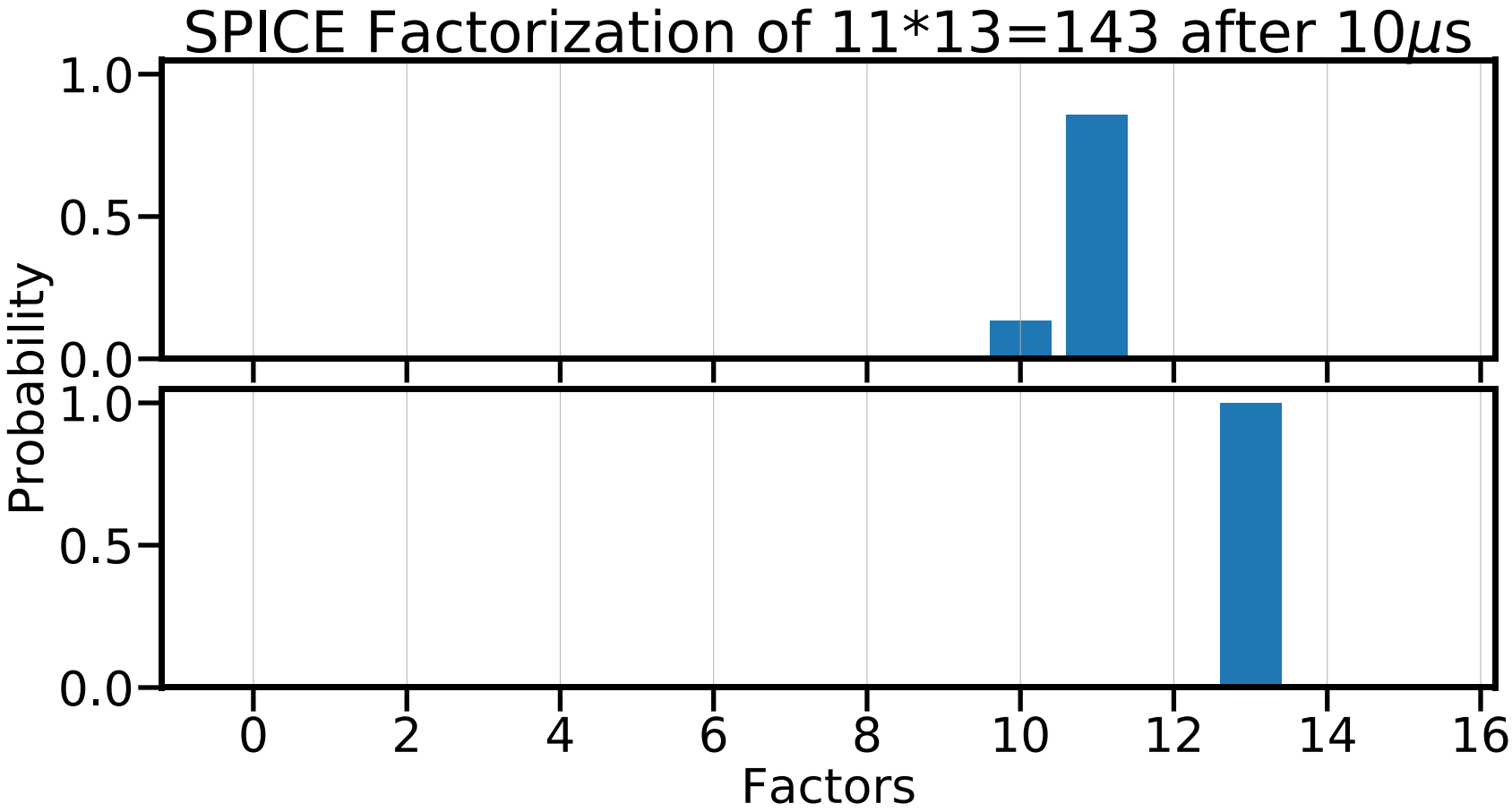
Wpmos

Bias D

Retraining to improve performance



Full SPICE factorization of 8 bit number



Conclusions

- We use RBMs to solve 16 bit factorization tasks in software
- This architecture can be extended to solve other combinatorial optimization tasks
- We provide a possible 90nm CMOS implementation able to solve 8 bit factorization tasks in under 10us
  - Represents ~10^4 improvement over CPU performance

Questions?