Low-power Implementation of Mitchell's Approximate Logarithmic Multiplication for Convolutional Neural Networks

Min Soo Kim[1], Alberto A. Del Barrio[2], Román Hermida[2], Nader Bagherzadeh[1]

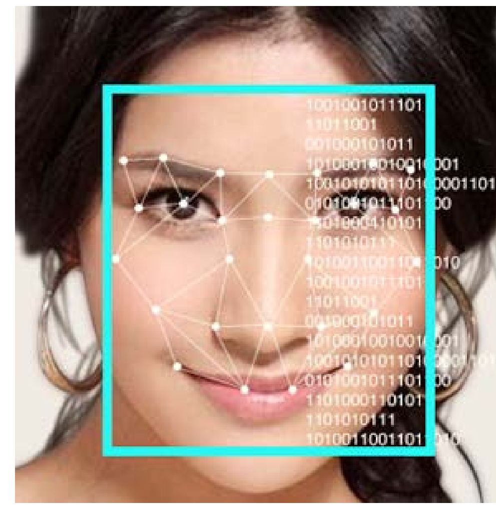
[1]University of California, Irvine, USA
[2]Univ. Complutense de Madrid, Spain

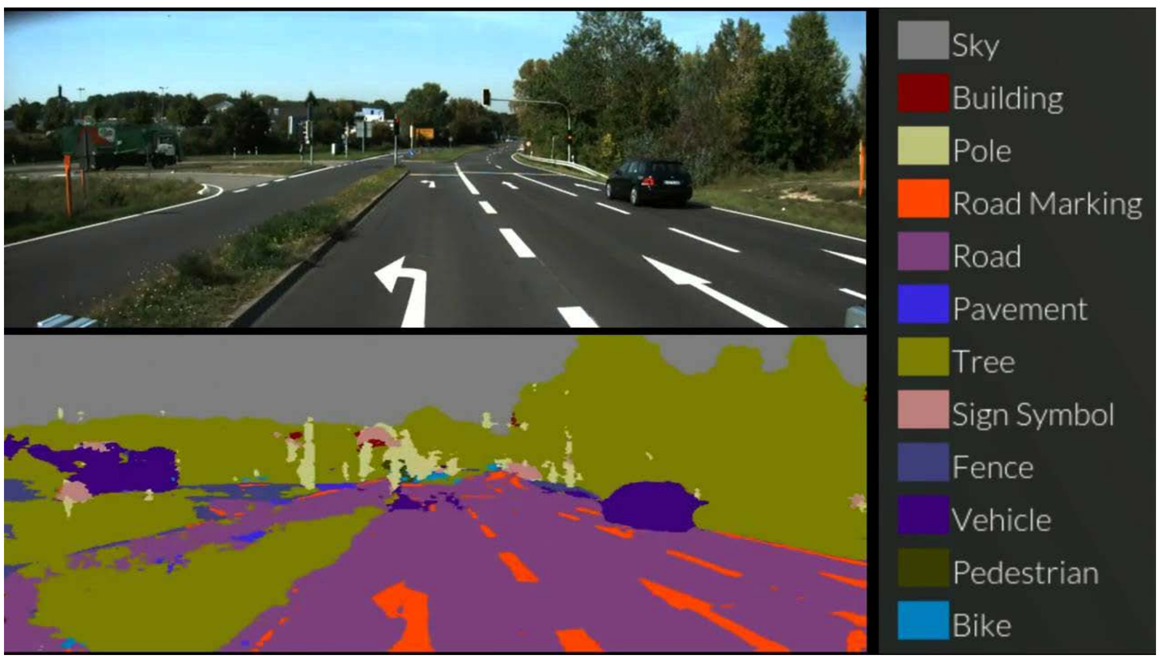
Jan 2018

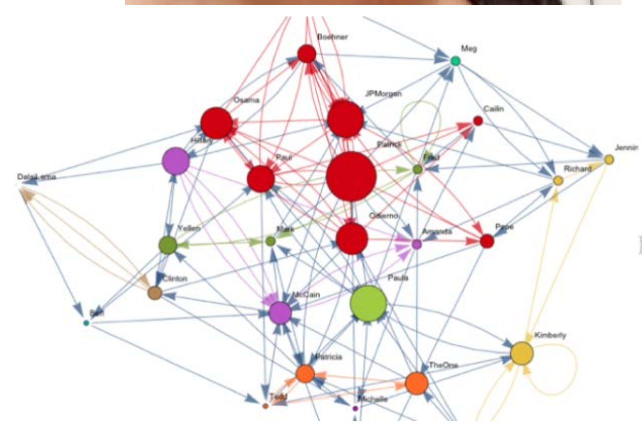
23rd Asia and South Pacific Design Automation Conference

Computational Challenge in Machine Learning

- **Machine Learning growing in diverse applications**
  - Autonomous Driving, Face Recognition, Social Analysis…
- **Large amount of data and/or time constraint**
  - Computationally costly and challenging!

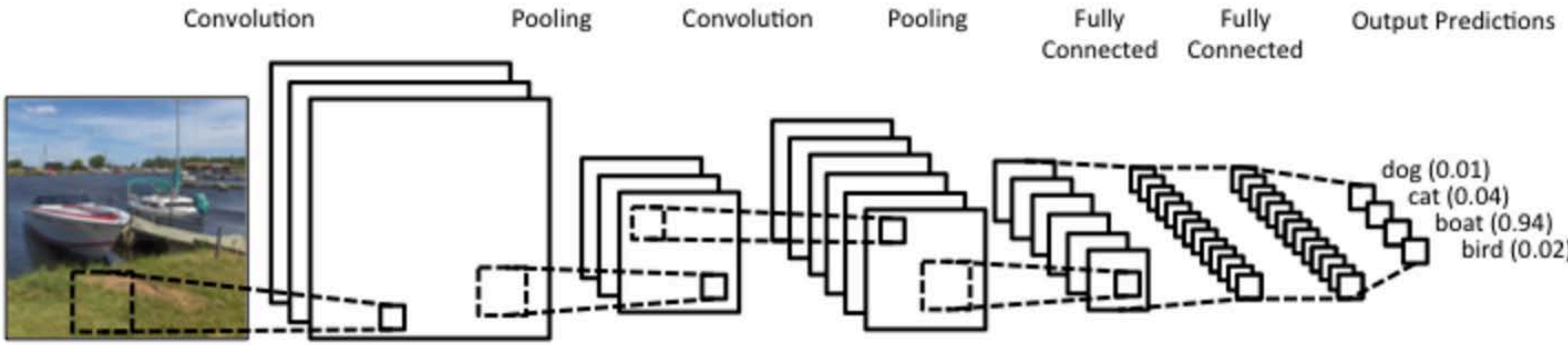


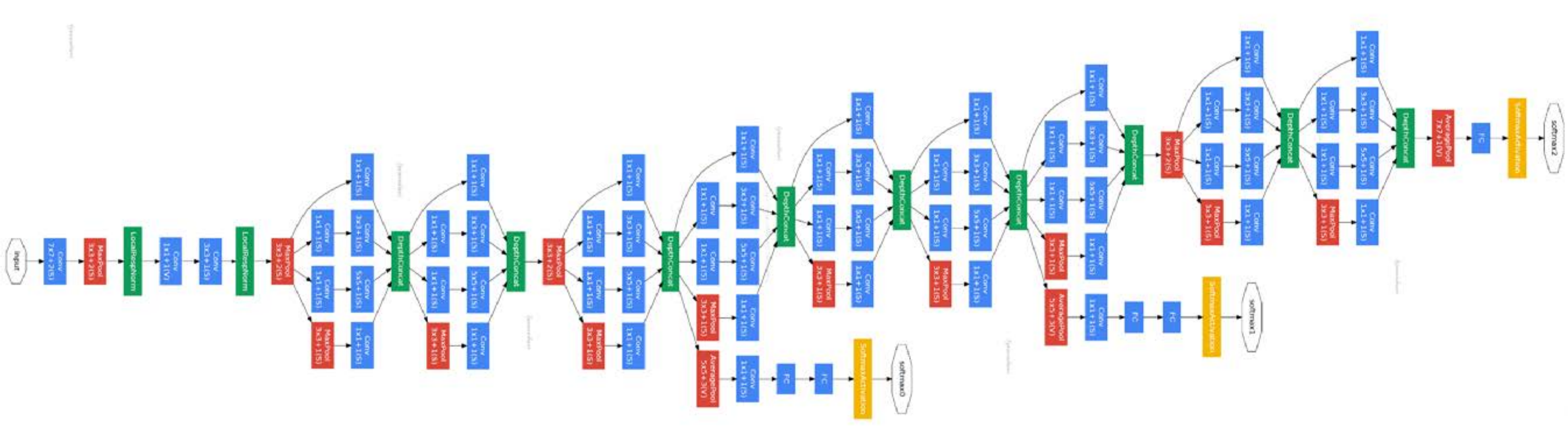




Convolutional Neural Network (CNN)

- Popular model for Visual and Speech Recognition
- Large amount of multiply-accumulate(MAC)



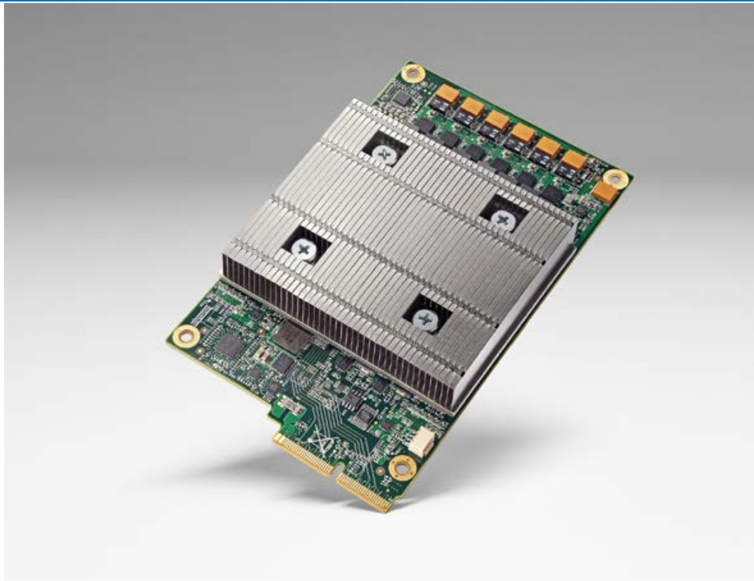


Opportunities for Power Savings

- **Perfect for hardware acceleration**
  - A lot of MAC operations
  - Parallel and regular structure

- **Suitable for Approximate Computing**
  - Inherent error in machine learning
  - Applications can tolerate small errors

- **Approximate multiplier for the CNN accelerator can reduce power consumption for datacenters and embedded systems**



Google TPU Accelerator [1]

Page Ranking

Translate

Visual Recognition

AlphaGo

Services that use TPU

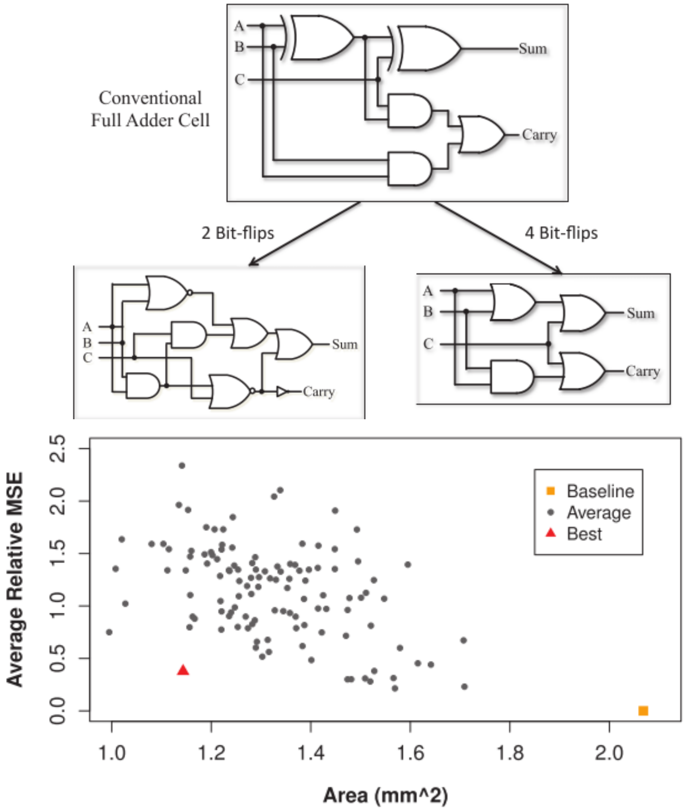
[1] Jouppi, Norman P., et al. "In-datacenter performance analysis of a tensor processing unit." Proceedings of the 44th Annual International Symposium on Computer Architecture. ACM, 2017.

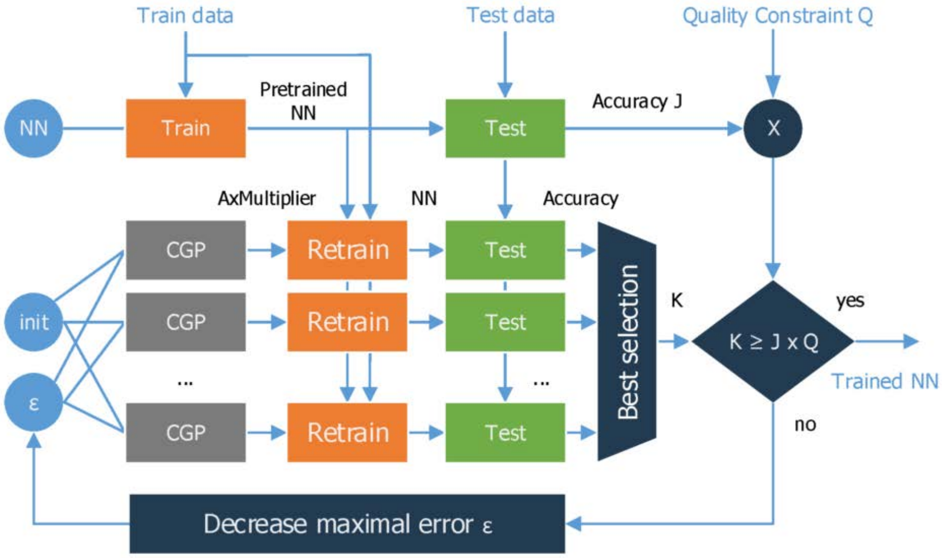
Previous Approaches

- Introduction
- **Previous Approaches**
- Proposed Multiplier
- Experimental Results
- Conclusion

Previous Approaches

- **Approximations based on logic bit flips demonstrated significant resource reduction, but not scalable [1,2]**

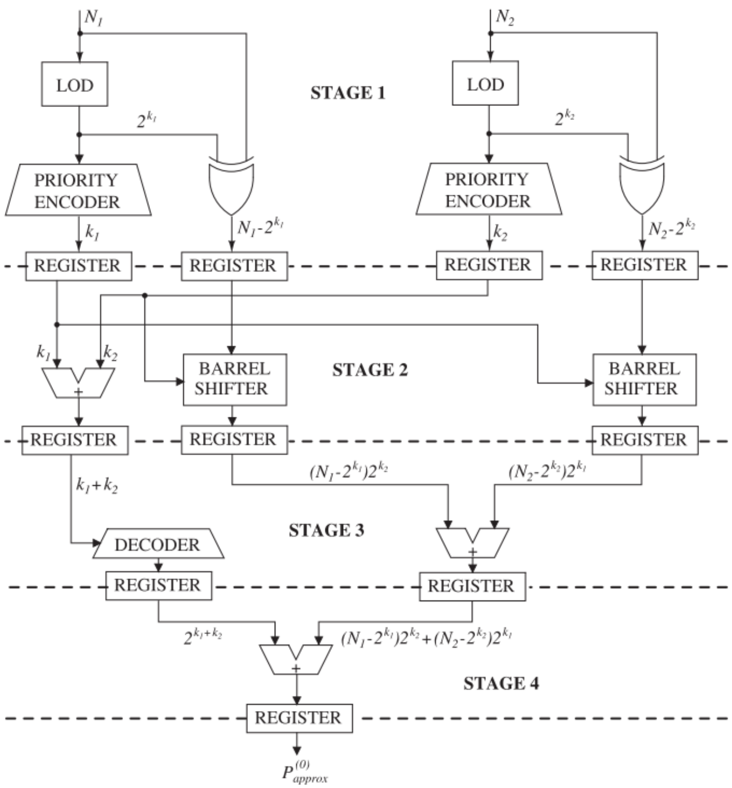


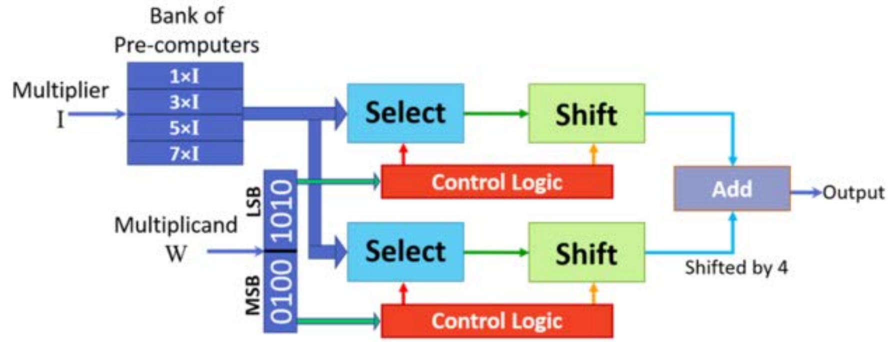


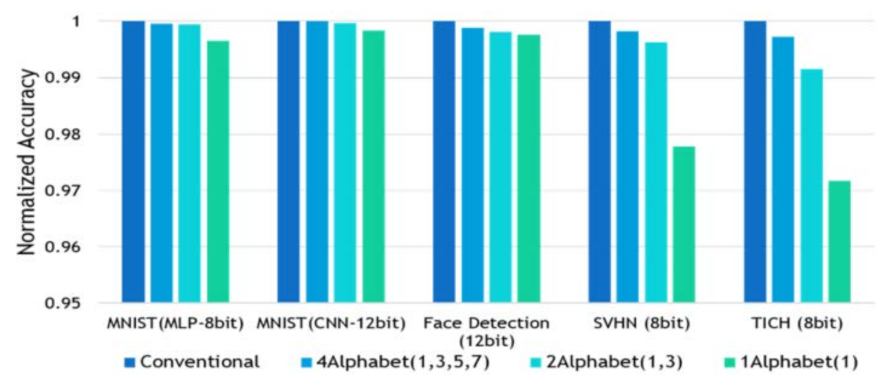
[1] Du, Z., Palem, K., Lingamneni, A., Temam, O., Chen, Y., & Wu, C. (2014). Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators. *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, 201–206.
[2] Mrazek, V., Sarwar, S. S., Sekanina, L., Vasicek, Z., & Roy, K. (2016). Design of power-efficient approximate multipliers for approximate artificial neural networks. Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD '16

Previous Approaches

- **Approximations based on algorithms are scalable, but had shown inefficiency or CNN performance degradation [1,2]**







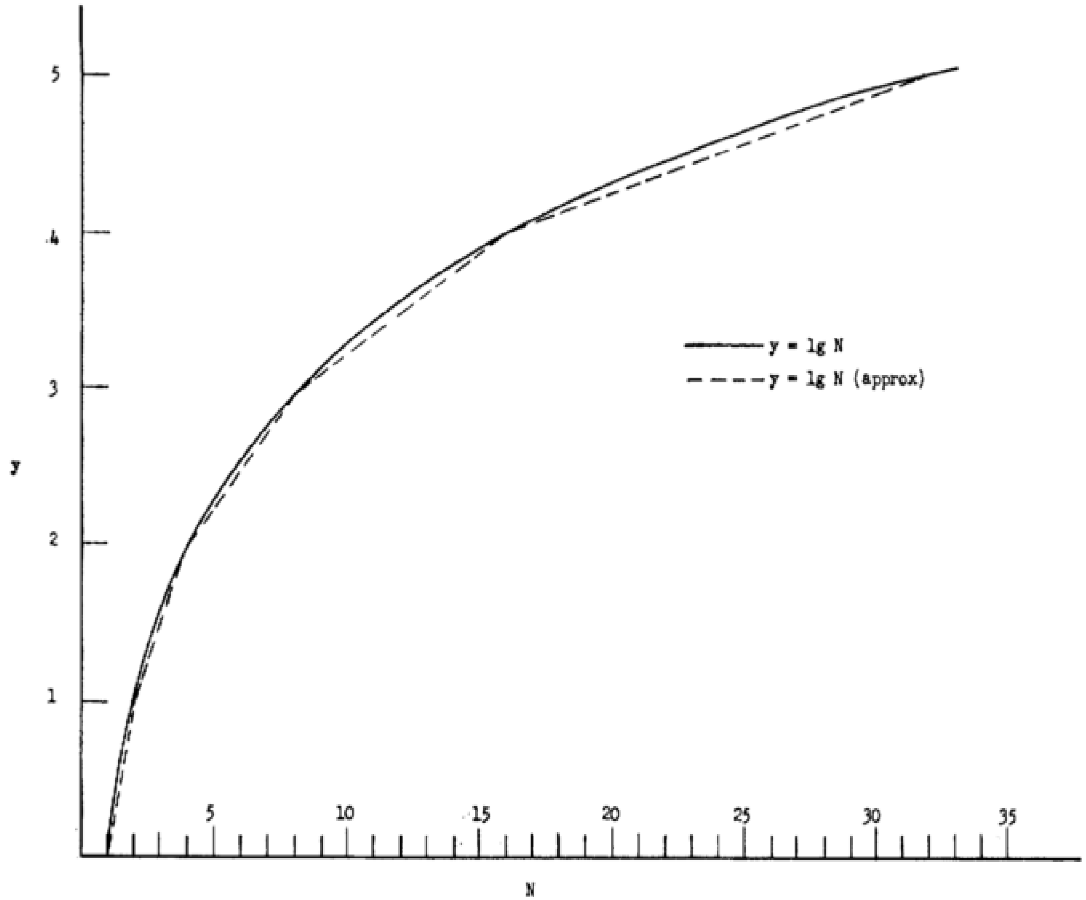
[1] Lotrič, U., & Bulić, P. (2012). Applicability of approximate multipliers in hardware neural networks. Neurocomputing, 96, 57–65
[2] Sarwar, S. S., Venkataramani, S., Raghunathan, A., & Roy, K. (2016). Multiplier-less Artificial Neurons Exploiting Error Resiliency for Energy-Efficient Neural Computing. Date 16, 0–5. Retrieved from http://arxiv.org/abs/1602.08557

Proposed Multiplier

Approximate Log Multiplication

- Based on the approximate logarithm
- Reduces logarithm to LOD and Shifter operations

N (binary)	Approx. log(N) (binary)
00001	000.0000
00010	001.0000
00011	001.1000
00100	010.0000
00101	010.0100
00110	010.1000
00111	010.1100
01000	011.0000
…	…
10000	100.0000



Mitchell, J. N. (1962). Computer Multiplication and Division Using Binary Logarithms. *Electronic Computers, IRE Transactions on, EC-11*(4), 512–517. http://doi.org/10.1109/TEC.1962.5219391

Approximate Log Multiplication

- **Multiplication → Addition in Log Domain**
- **Worst case relative error = 11.1%**

	7	6	5	4	3	2	1	0	
A	0	0	0	0	1	1	1	1	15
B	0	0	0	0	0	0	1	1	3

C	0	1	1	1	1	1	0	0	0	0
D	0	0	1	1	0	0	0	0	0	0

C+D = E	0	1	0	1	0	1	1	0	0	0	0

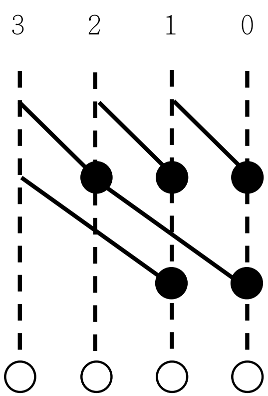
F

									7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	44

2% error

Mitchell Log Multiplier

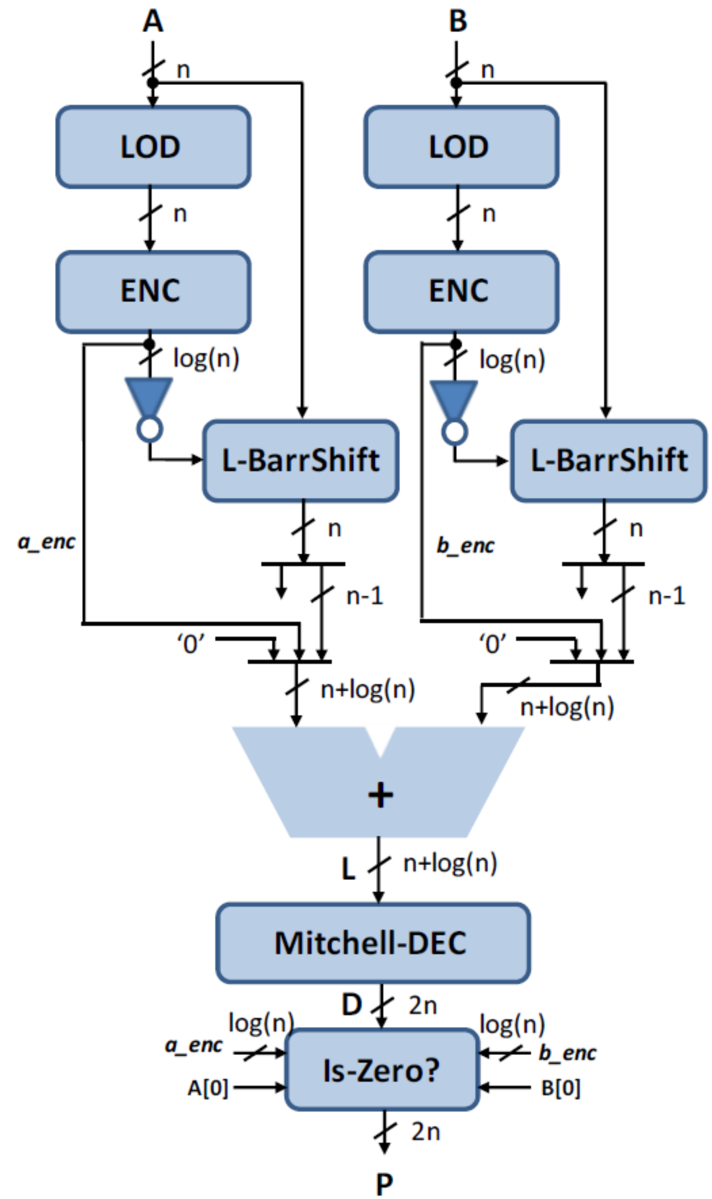
- **Logic optimization of LOD and ENC**
  - Fast and efficient fully parallel LOD
  - OR-Tree encoder
- **Shift amount calculation**
  - (n-k-1) = not(k) when n is a power of 2



● = $m_{i-1,j} + m_{i-1,j+2^{i-1}}$

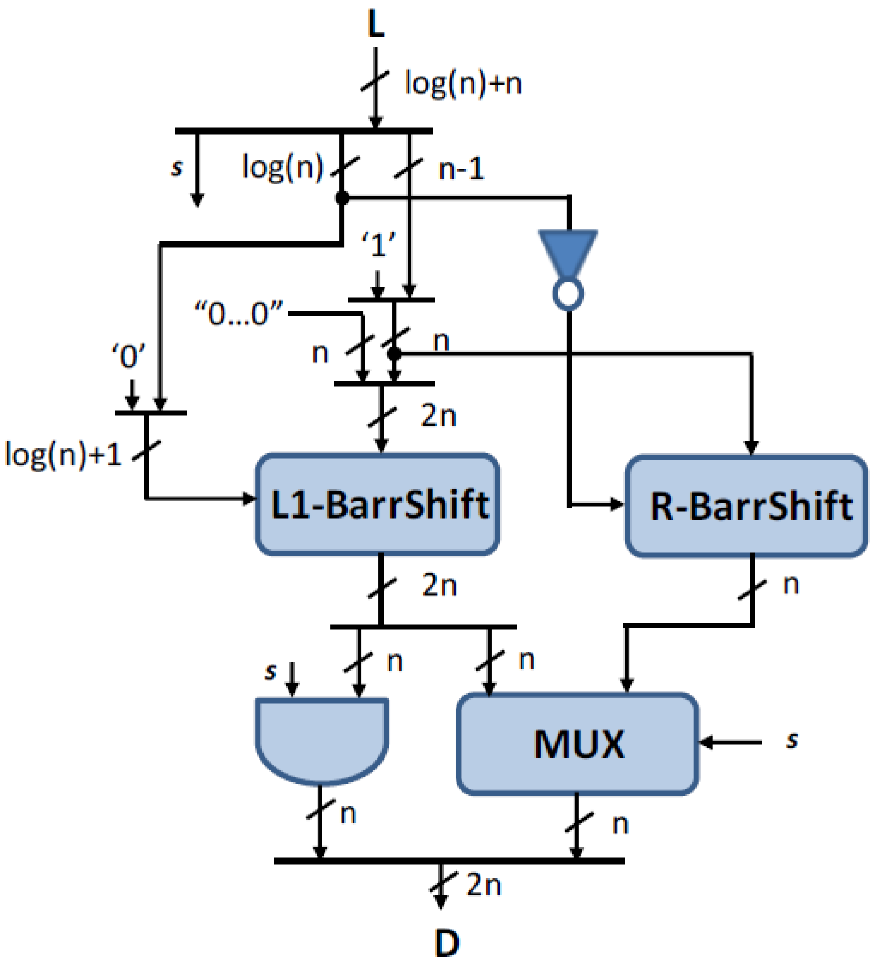
○ = $h_j = \begin{cases} z_j & j = n-1 \\ \overline{m_{\log(n),j+1}} \cdot z_j & j < n-1 \end{cases}$

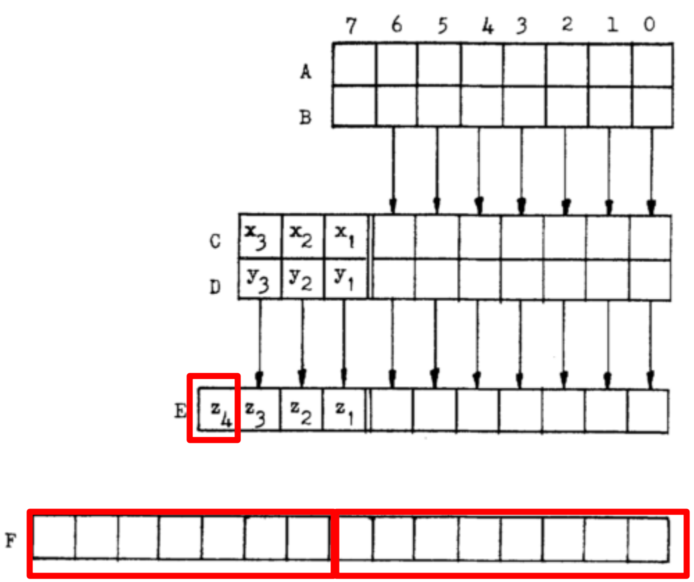
4-bit parallel LOD



Mitchell Decoder

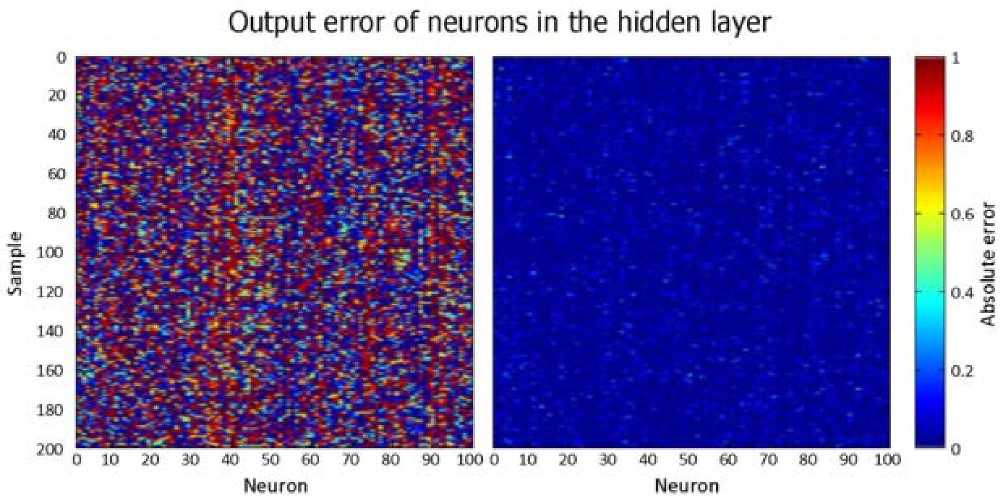
- **Two cases for decoding**
  - Large Characteristic
  - Small Characteristic
- **Only AND needed for MSBs**

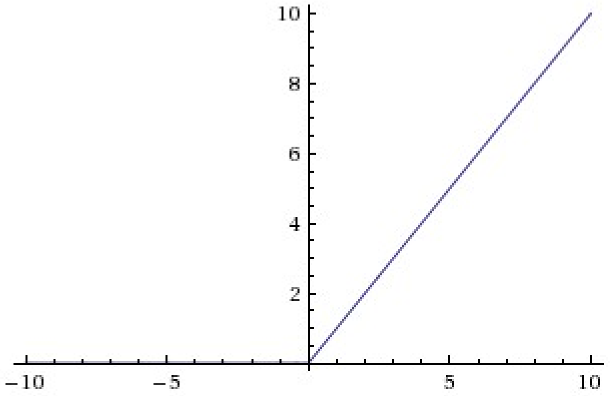


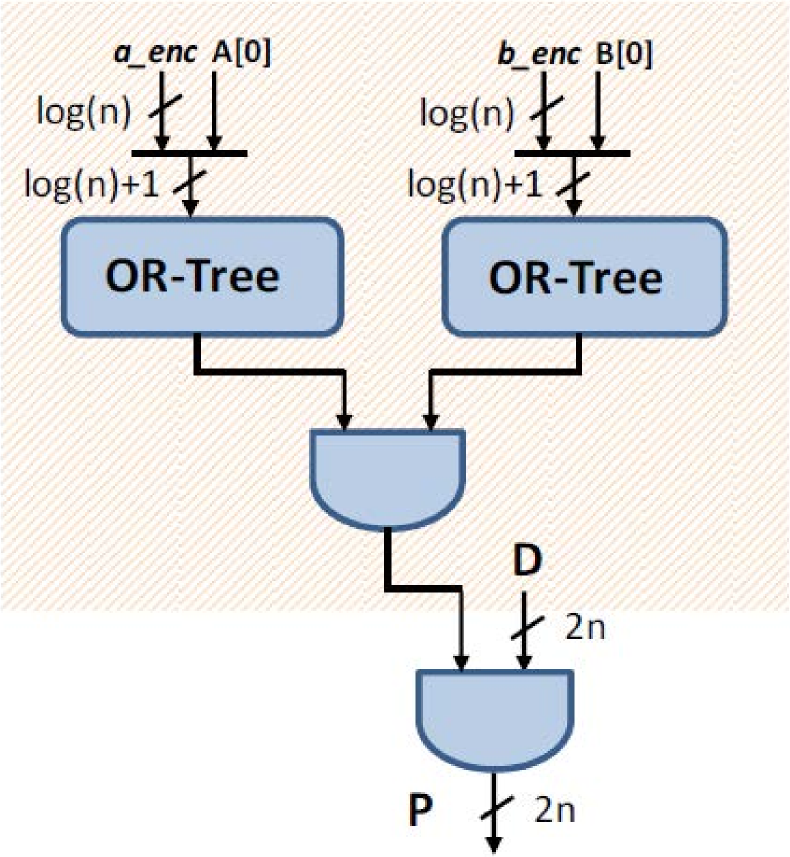


Zero Detection Unit

- Critical to CNN accuracy [1]







[1] Mrazek, V., Sarwar, S. S., Sekanina, L., Vasicek, Z., & Roy, K. (2016). Design of power-efficient approximate multipliers for approximate artificial neural networks. *Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD '16*, 1–7.

Power and Area Savings

- **Synthesis using Synopsys Design Compiler**
  - 32nm tech library from Synopsys
  - 250 MHz Clock
- **Up to 76.6% Power Savings at 32 bits**

	8-bit			16-bit			32-bit		
	Exact	Our Design	Iter. Log	Exact	Our Design	Iter. Log	Exact	Our Design	Iter. Log
Mean Rel. Error	0 %	3.77 %	0.83%	0 %	3.83 %	0.99%	0 %	3.87 %	N/A
Worst Rel. Error	0 %	11.11%	6.25%	0 %	11.11%	6.25%	0 %	11.11%	6.25%
Cell Area (um^2)	403	312	872	1681	909	2189	6409	2161	7220
Critical Path (ns)	1.07	1.13	1.75	2.23	2.31	3.77	3.78	3.70	4.00
Tot.Power (mW)	0.269	0.197	0.544	1.240	0.549	1.310	6.02	1.41	4.64
Power Savings		**26.8%**	-102%		**55.7%**	-5.6 %		**76.6%**	22.9%

Experimental Results

- Introduction
- Previous Approaches
- Proposed Multiplier
- **Experimental Results**
- Conclusion

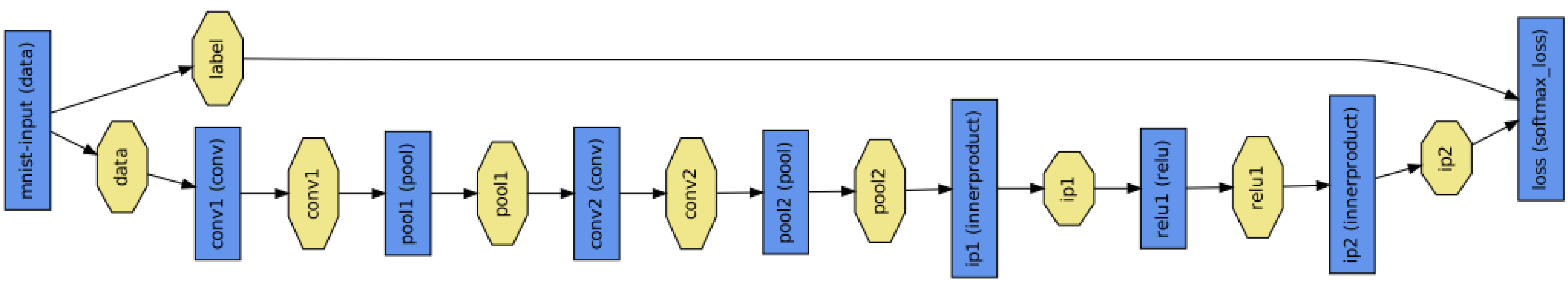
Accuracy Evaluation on CNNs

Caffe

Berkley Vision and Learning Centre

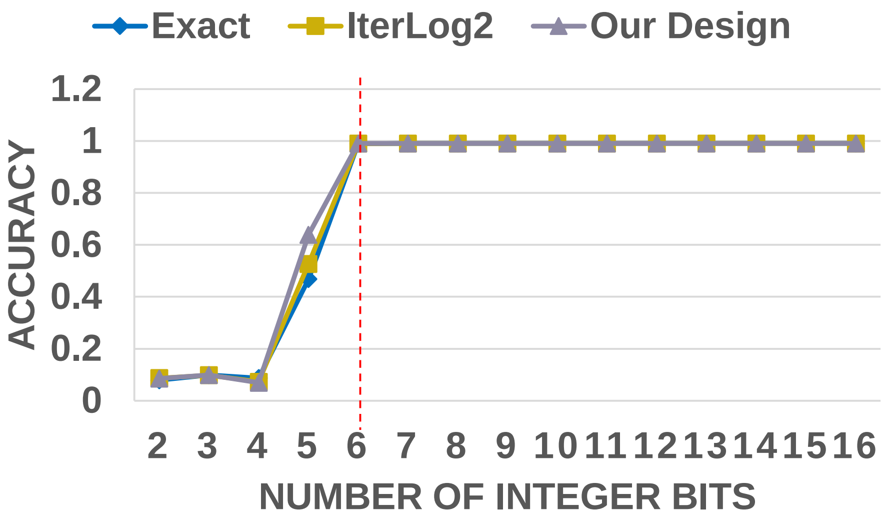
- **Caffe's floating-point matrix multiplication replaced by Fixed-point C++ Subroutines**

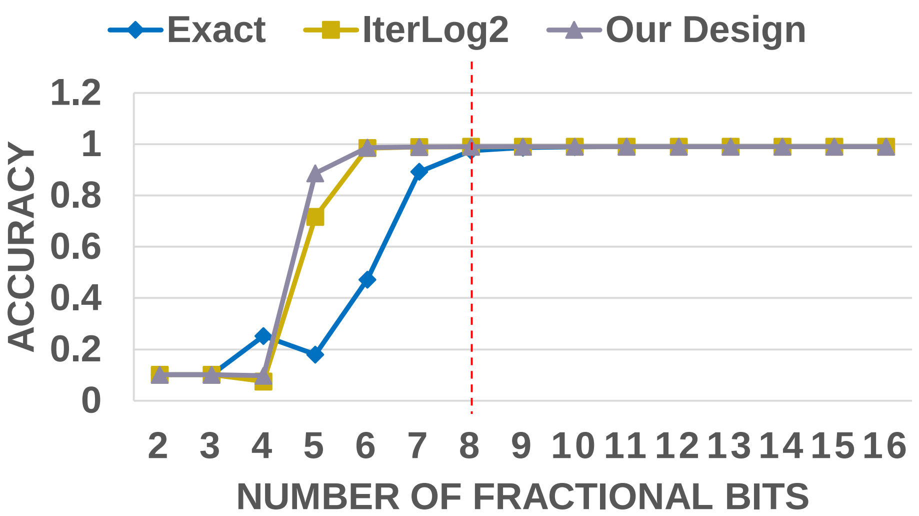
Image Dataset	CNN	Layers
MNIST Handwritten Digit Recognition	LeNet	Convolution → Pooling → Convolution → Pooling → FC → ReLU → FC
CIFAR-10 Object Recognition	Cuda-convnet	Convolution → Pooling → ReLU → LRN → Convolution → ReLU → Pooling → LRN → Convolution → ReLU → Pooling → FC



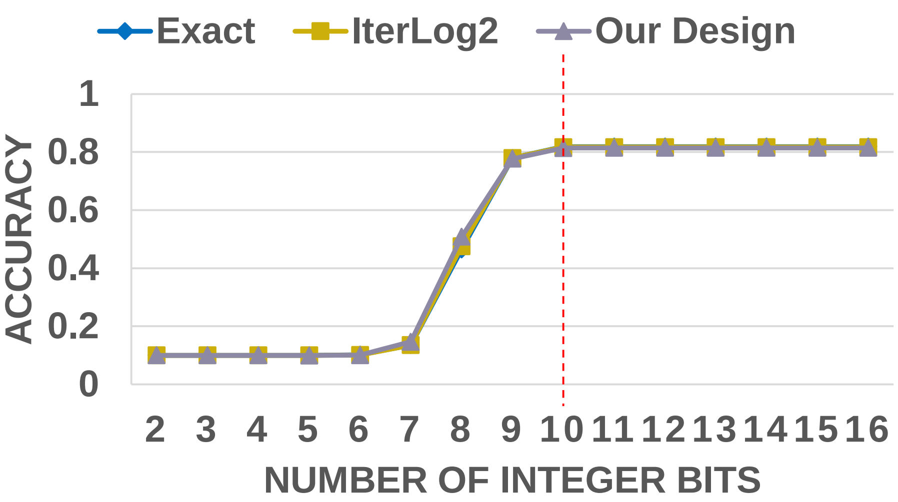
CNN Top-1 Accuracy Comparison

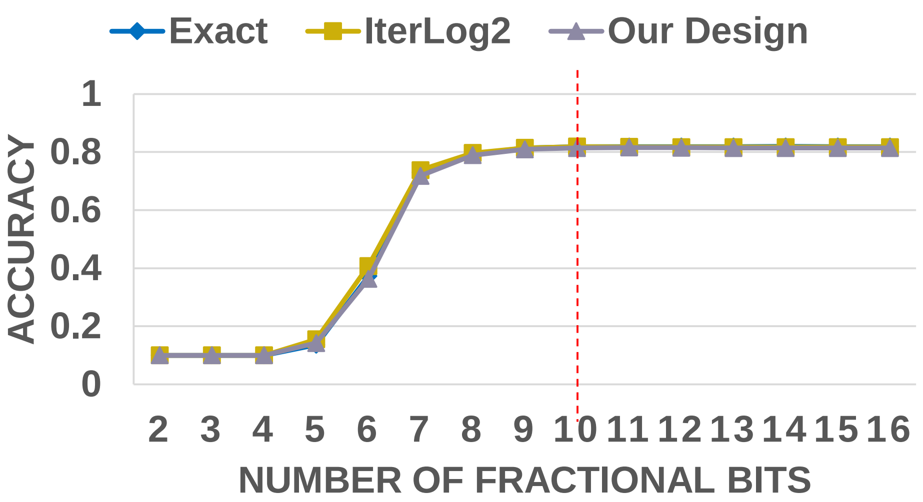
MNIST





CIFAR-10





CNN Top-1 Accuracy Comparison

Top-1 accuracies with 10 integer bits and 22 fractional bits

Dataset	Reference Floating-point	Our Design	Exact Fixed-point	Iterative Logarithm (2 stage)
MNIST	99.02 %	99.02 %	99.02 %	99.02 %
CIFAR-10	81.43 %	81.43 %	81.89 %	81.71 %

- **Our design shows no performance degradation for MNIST and CIFAR-10 datasets**
- **In CNNs, the error associated with approximate multipliers can sometimes help produce correct predictions**
- **Correct zero handling is very important for CNNs**

Conclusion

- Introduction
- Previous Approaches
- Proposed Multiplier
- Experimental Results
- **Conclusion**

Conclusion

- Optimized Mitchell Log Multiplier for CNN Inference
- Significant power reduction expected at little to no degradation in CNN inference performance
- More scalable than the gate-level approximation
- Better power savings or CNN accuracy compared to the state-of-the-art algorithmic approximations

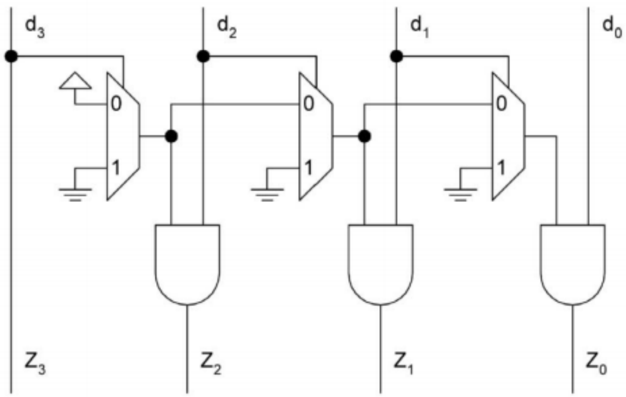
Thank you!

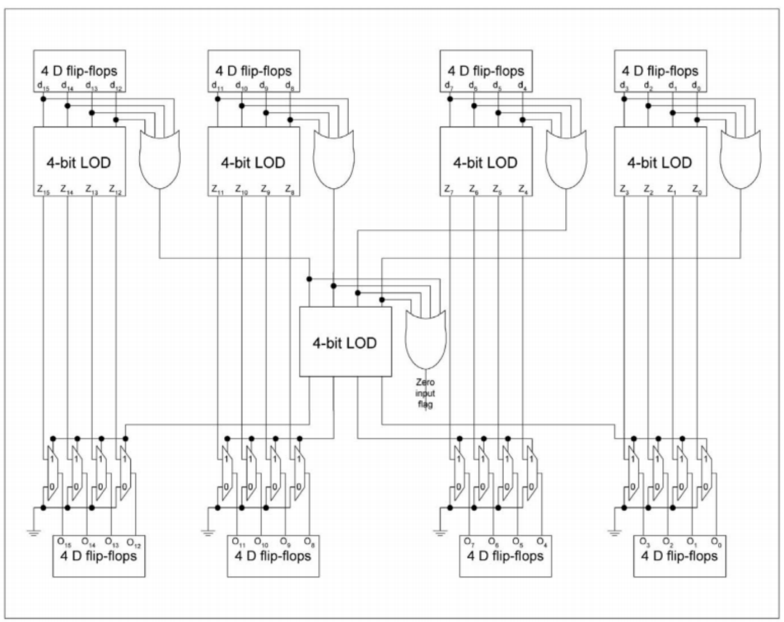
21

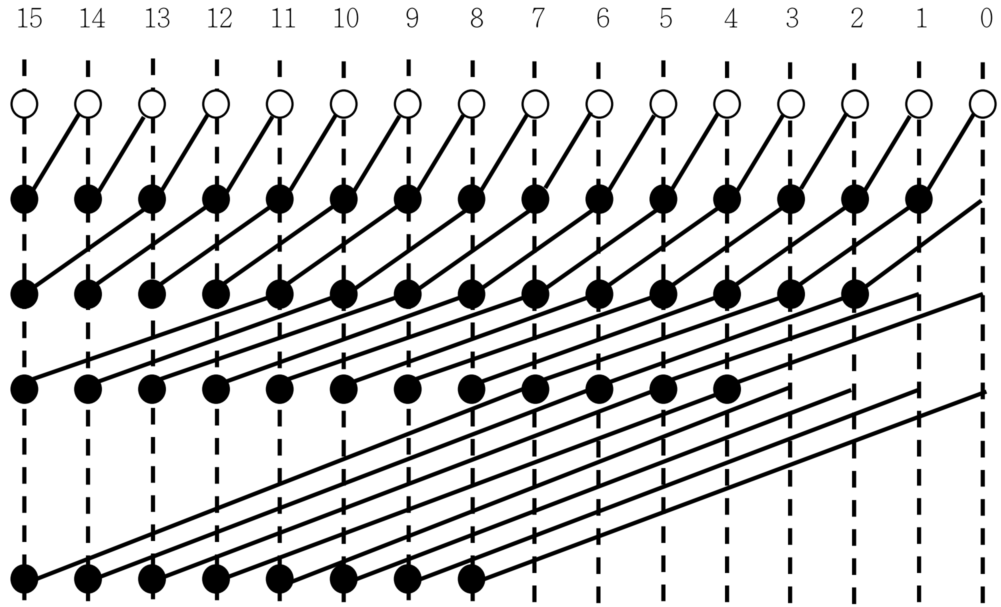
Q & A

Backup: Comparison of LOD

Previous Approach







16-bit Kogge-Stone Adder

$$m_{i,j} = \begin{cases} z_j, & i = 0 \\ m_{i-1,j}, & i > 0, (n-1-j) < 2^{i-1} \\ m_{i-1,j} + m_{i-1,j+2^{i-1}}, & i > 0, (n-1-j) \geq 2^{i-1} \end{cases}$$

$$\forall i, 0 \leq i \leq \log(n), \forall j, 0 \leq j < n$$

$$h_j = \begin{cases} z_j, & j = n-1 \\ \overline{(m_{\log(n),j+1})} \cdot z_j, & j < n-1 \end{cases}$$