# Scalable Stochastic-Computing Accelerator for Convolutional Neural Networks

Hyeonuk Sim[1], Dong Nguyen[1], Jongeun Lee[1] and Kiyoung Choi[2]

[1]UNIST, Ulsan, South Korea

[2]Seoul National University, Seoul, South Korea

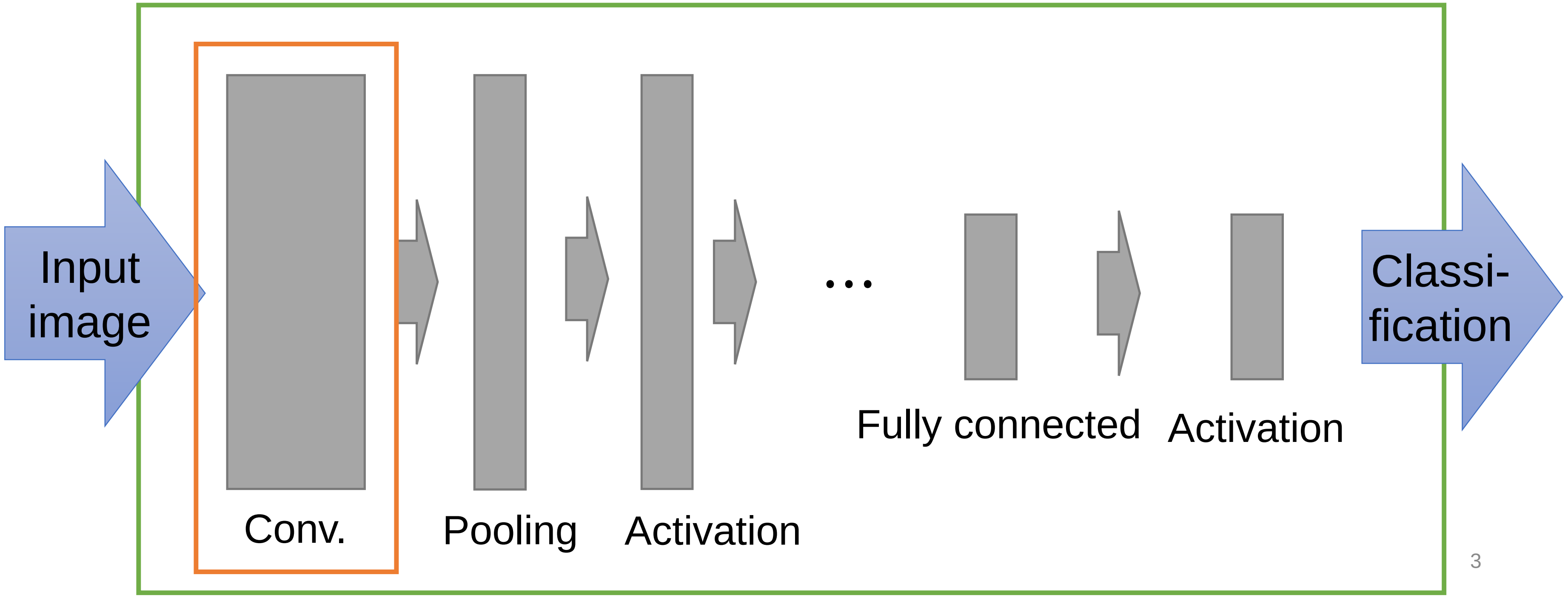
Renew: Reconfigurable and Neuromorphic Computing Lab

## Outline

- Introduction
- Previous Approach
- Problems and Challenges
- Main Contributions/Solutions
- Experimental Results
- Conclusion

# Introduction

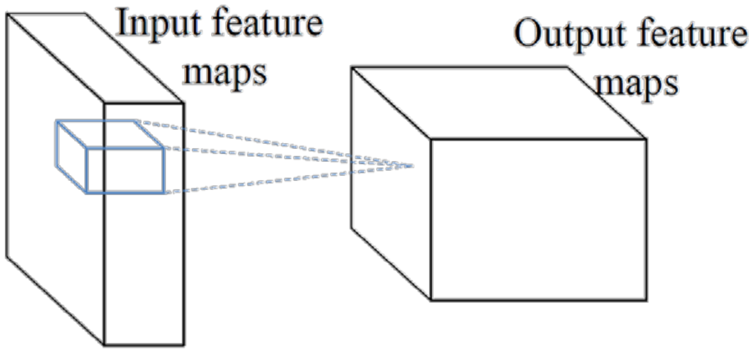
- **Deep Learning**
  - **Deep Convolutional Neural Networks (ConvNets)**
    - Powerful image recognition
    - Computation-hungry
    - For energy efficiency, various accelerators are researched. Ex) GPU, FPGA, TrueNorth from IBM



## Introduction

- Computation of **convolutional layer**
  - **3 dimensional** neurons
  - Lots of **multiplication** and **addition**

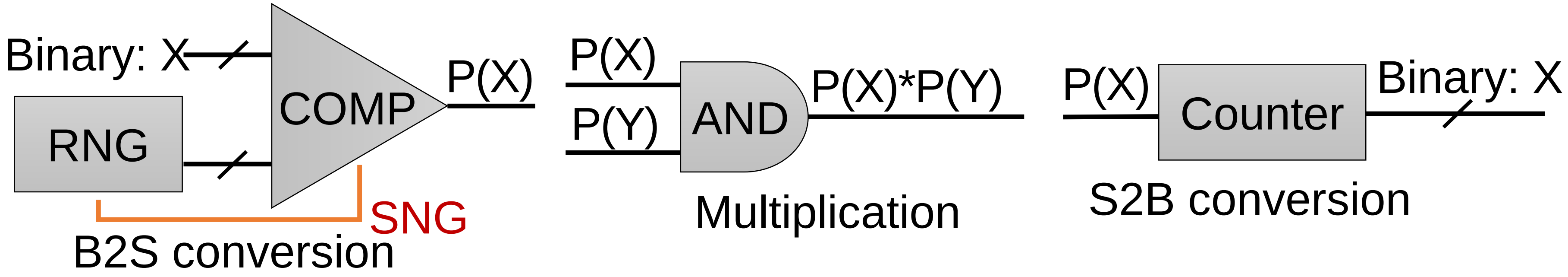
$$Neuron[R, C, M] = \sum_{Z} \sum_{K_{row}} \sum_{K_{col}} in[z, R, C, k_{row}, k_{col}] \times wght[z, k_{row}, k_{col}, M]$$



$R$: $row\ of\ output\ neuron$
$C$: $column\ of\ output\ neuron$
$M$: $feature\ map\ of\ output\ neuron$
$Z$: $feature\ map\ of\ input\ neuron$
$K_{row}$: $row\ of\ convolution\ kernel$
$K_{col}$: $column\ of\ convolution\ kernel$

# Introduction

- **Stochastic Computing (SC)**
  - Area and power efficient computing method using **frequency of 1's** in stream (3/8→00101010)
  - Compact **multiplication**
  - **Conversion** overheads
  - Extend range to [-1,1] using bipolar format: 2P(X)-1

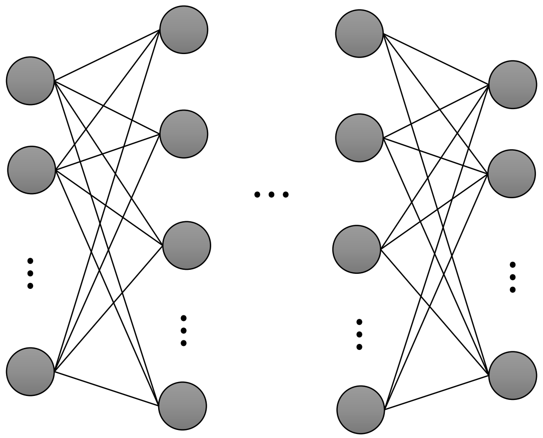


RNG: Random Number Generator
SNG: Stochastic Number Generator
B2S: Binary to Stochastic
S2B: Stochastic to Binary

# Previous Approach

- Fully Connected Networks [1]-[7]
  - $Neuron[M] = AF(\sum_{Z} in[z] * wght[z, M])$
  - Relatively smaller than ConvNets

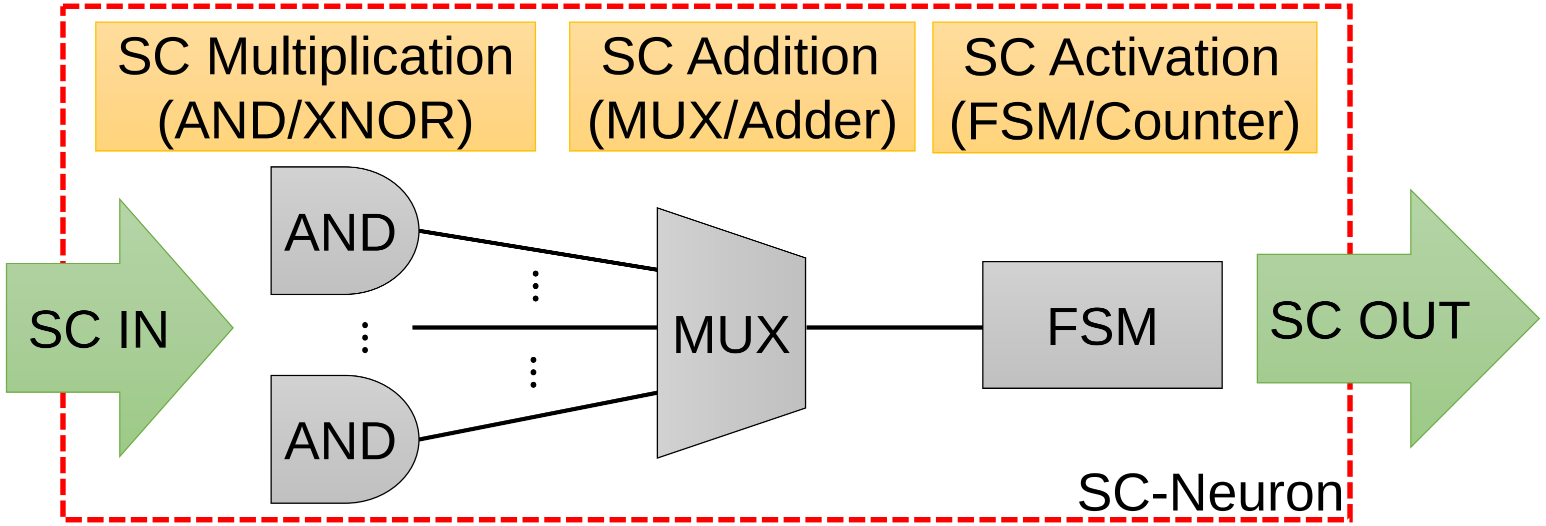
- **Fully Parallel ASIC** [2], [6]-[9]
  - Implement all hardware neurons



[1] FPGA-based stochastic neural networks implementation. 1994;
[2] Stochastic neural computation. I. computational elements. 2001;
[3] A hardware implementation of a radial basis function neural network using stochastic logic. 2015;
[4] FPGA implementation of a Deep Belief Network architecture for character recognition using stochastic computation. 2015;
[5] Using Stochastic Computing to Reduce the Hardware Requirements for a Restricted Boltzmann Machine Classifier. 2016;
[6] VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing. 2016;
[7] Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks. 2016;
[8] Designing reconfigurable large-scale deep learning systems using stochastic computing. 2016;
[9] DSCNN: Hardware-Oriented Optimization for Stochastic Computing Based Deep Convolutional Neural Networks. 2016;

## Previous Approach

- **All Stochastic Computing** [2], [6]-[9]
  - All computations of all layers in stochastic computing



[1] FPGA-based stochastic neural networks implementation. 1994;
[2] Stochastic neural computation. I. computational elements. 2001;
[3] A hardware implementation of a radial basis function neural network using stochastic logic. 2015;
[4] FPGA implementation of a Deep Belief Network architecture for character recognition using stochastic computation. 2015;
[5] Using Stochastic Computing to Reduce the Hardware Requirements for a Restricted Boltzmann Machine Classifier. 2016;
[6] VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing. 2016;
[7] Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks. 2016;
[8] Designing reconfigurable large-scale deep learning systems using stochastic computing. 2016;
[9] DSCNN: Hardware-Oriented Optimization for Stochastic Computing Based Deep Convolutional Neural Networks. 2016;

## Problems and Challenges

- **All-SC: Fundamentally unscalable** architecture
  - Implement **all neurons** in hardware? Not scalable for recent networks
  - 665,784,864 multiplications (AND gates) for AlexNet
    - 1um² AND on 45nm tech. → **666mm²** for AND gates only!
- **All-SC: Incompatible with reusing neurons**
  - Due to **very high amount of intermediate SC data**
  - **Exponential** memory and bandwidth consumption

(3/8→$011_{bin}$→00101010)

- Minimize B2S / S2B **conversion overhead**
- How to significantly **reduce SC latency** without compromising on **accuracy**?
  - **Long latency** decreases **energy efficiency**

## Our Solution to Problems

- Unscalable architecture & incompatibility with neuron reuse
  → **Tile-parallel architecture, Binary-Interlaced SC (BISC)**

- Minimize conversion overhead
  → Optimized **bit-parallel SC-MAC** design

- Long latency for accuracy
  → **Hybrid Layer Composition**

# Main Contributions
## - Tile-Parallel BISC Architecture

- **The first tile-parallel SC-architecture**
  - Can support convolution layers **of any size**

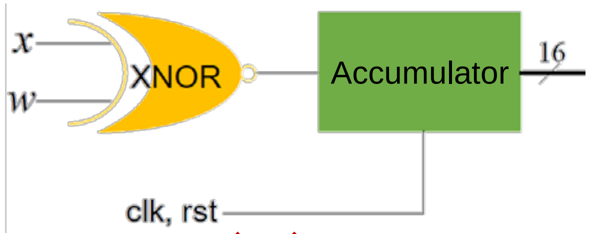
- **Binary-Interlaced SC (BISC)**
  - Conversion before/after multiplication
  - Addition using (binary-)accumulator

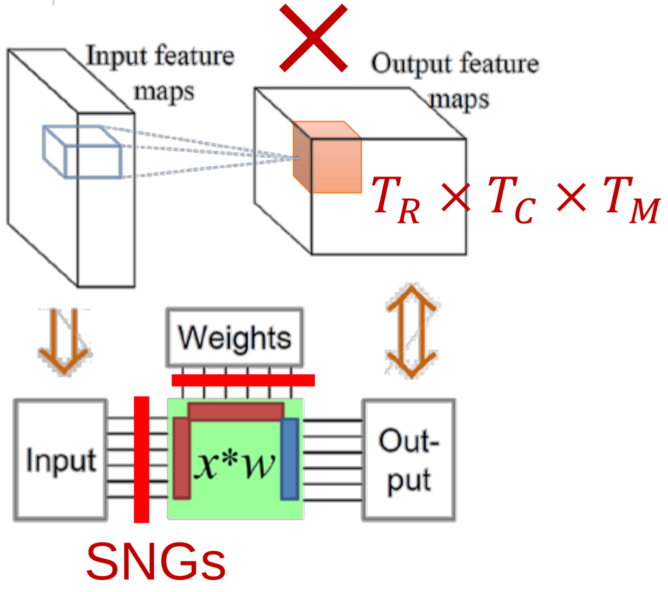
- **Parallelism along all dimensions of output neurons (*R*, *C*, *M*)**
  - Critical path depends on $Z$-$K$-$K$ only
  - Thus, scalable to any $\boldsymbol{T_R \times T_C \times T_M}$ size

$$Neuron(R,C,M) =$$

$$AF(\sum_{Z}\sum_{K_{row}}\sum_{K_{col}}\sum_{SL} XNOR(in_s\,, wght_s))$$

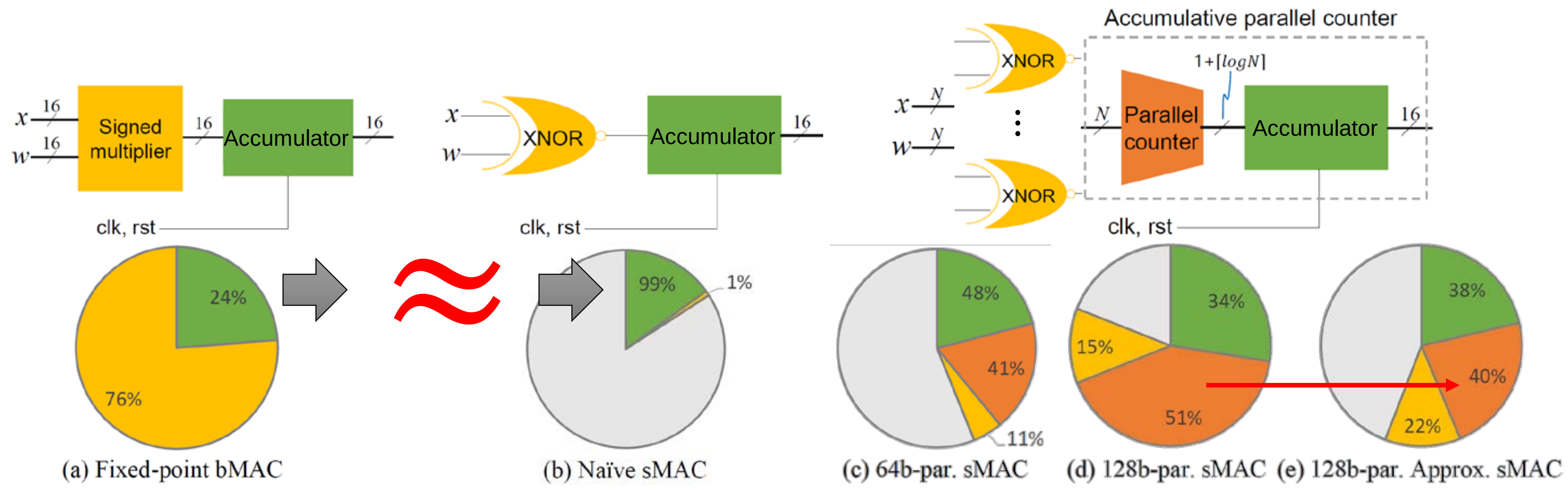
$$SL:\ Stream\ Length$$





# Main Contributions
## - SC-MAC Design Optimization

- Huge **counter (S2B) overhead**
  - Exploit **bit-parallelism** → compute ***all SC-bits together***
    - Requires parallel B2S conversion (i.e., SNGs)
- **Approximate Parallel Counter** nearly **halves** counter area



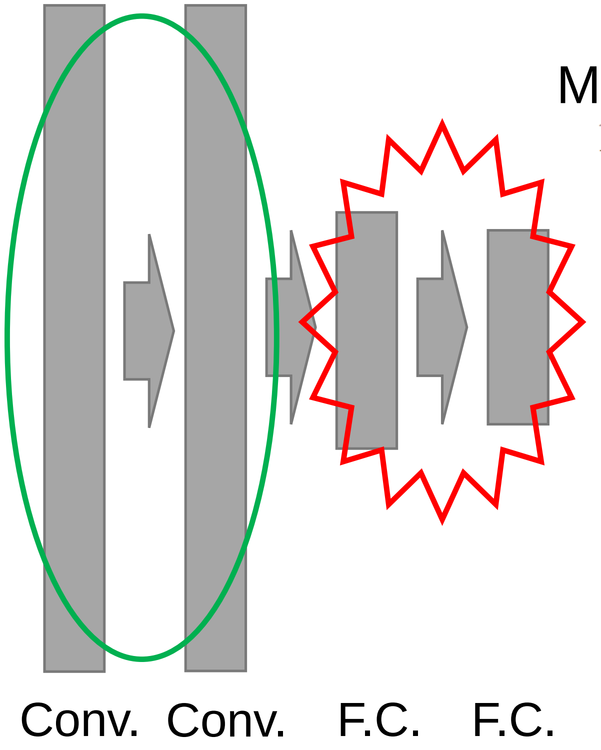
(a) Fixed-point bMAC (b) Naïve sMAC (c) 64b-par. sMAC (d) 128b-par. sMAC (e) 128b-par. Approx. sMAC

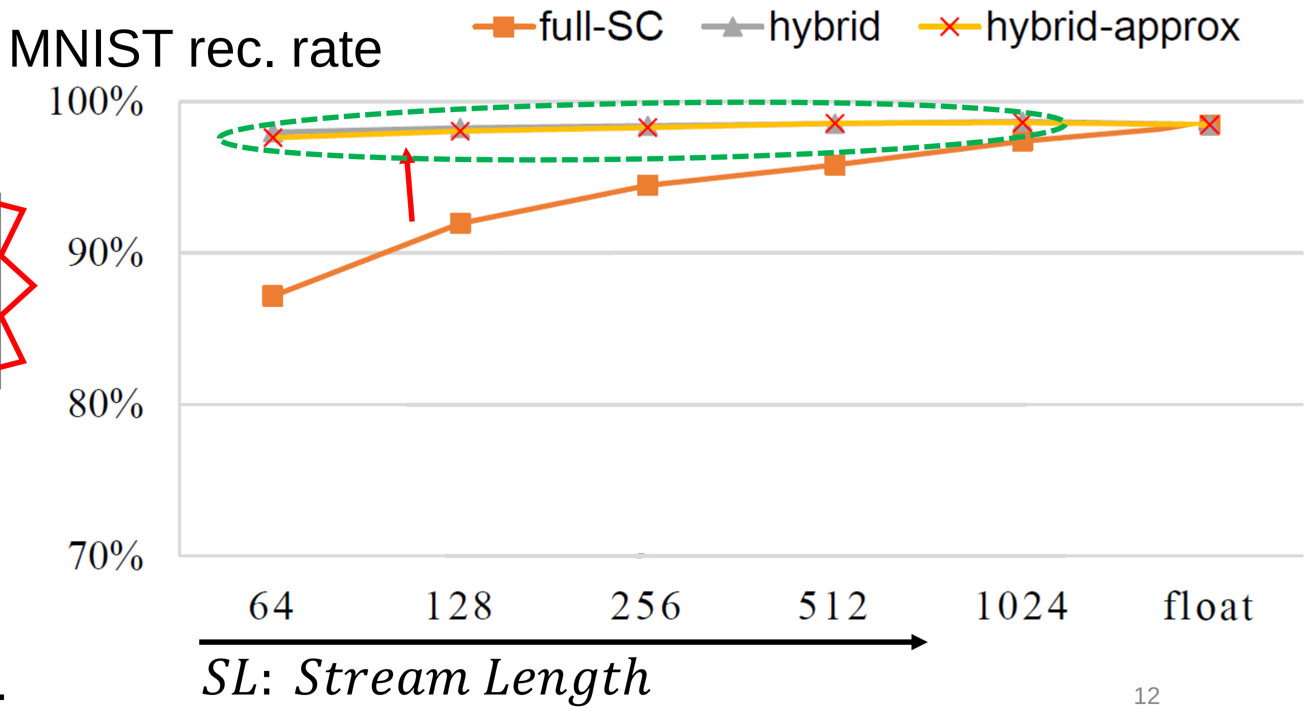
☐ Unused

Relative area composition with color coding

## Main Contributions
### - Hybrid Layer Composition

- Final decision made at **fully connected** (F.C.) **layer**
  - Computation error **directly** affect the recognition rate
- **Hybrid Layer Composition** reduces necessary stream length **by 8X** (1024→128)





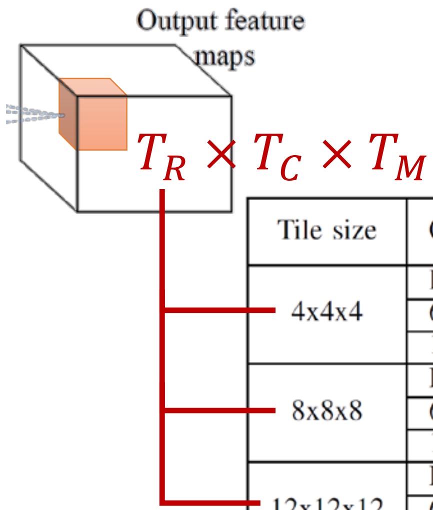
# Experimental Setup

## - Computation Element and Tile Synthesis

- Synthesis and stream length setting
  - HDL implementation
  - Synopsys Design Compiler
  - TSMC 45nm tech.
  - Stream length ($SL$) set to 128
- Compared with **same $\boldsymbol{T_R \times T_C \times T_M}$ architecture**
  - 16 bit precision fixed point binary MAC
  - 64b-par. sMAC, 128b-par. approx. sMAC
    - 3 cases of B2S conversion (SNG)
- Compared with state-of-the-art **neural network accelerators**
  - Without SNG

## Results
### - Comparison with fixed point binary

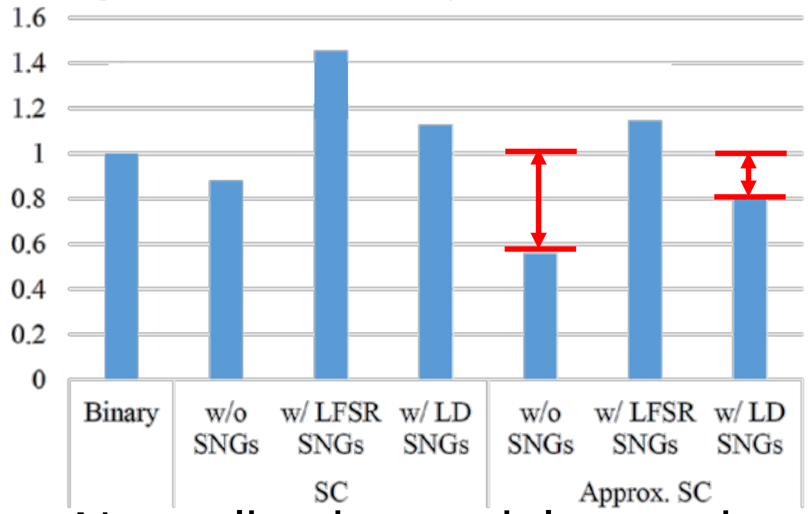
- **Easily scalable** depending on area budget
  - Area is **linear** to #Elements
  - B2S **conversion overheads decrease** with the **large tile size**
- With better SNGs, **20%** better,
- Ideally (w/o SNG), **44%** smaller area-delay product compared to fixed point binary



| Tile size | Case | Area (mm$^2$) SNG | Area (mm$^2$) MAC array |
|---|---|---|---|
| 4x4x4 | Binary | — | 0.094 |
| | 64b-SC | 0.074 | 0.042 |
| | 128b-SC approx. | 0.151 | 0.053 |
| 8x8x8 | Binary | — | 0.765 |
| | 64b-SC | 0.220 | 0.336 |
| | 128b-SC approx. | 0.448 | 0.426 |
| 12x12x12 (est.) | Binary | — | 2.582 |
| | 64b-SC | 0.442 | 1.134 |
| | 128b-SC approx. | 0.904 | 1.438 |

8x

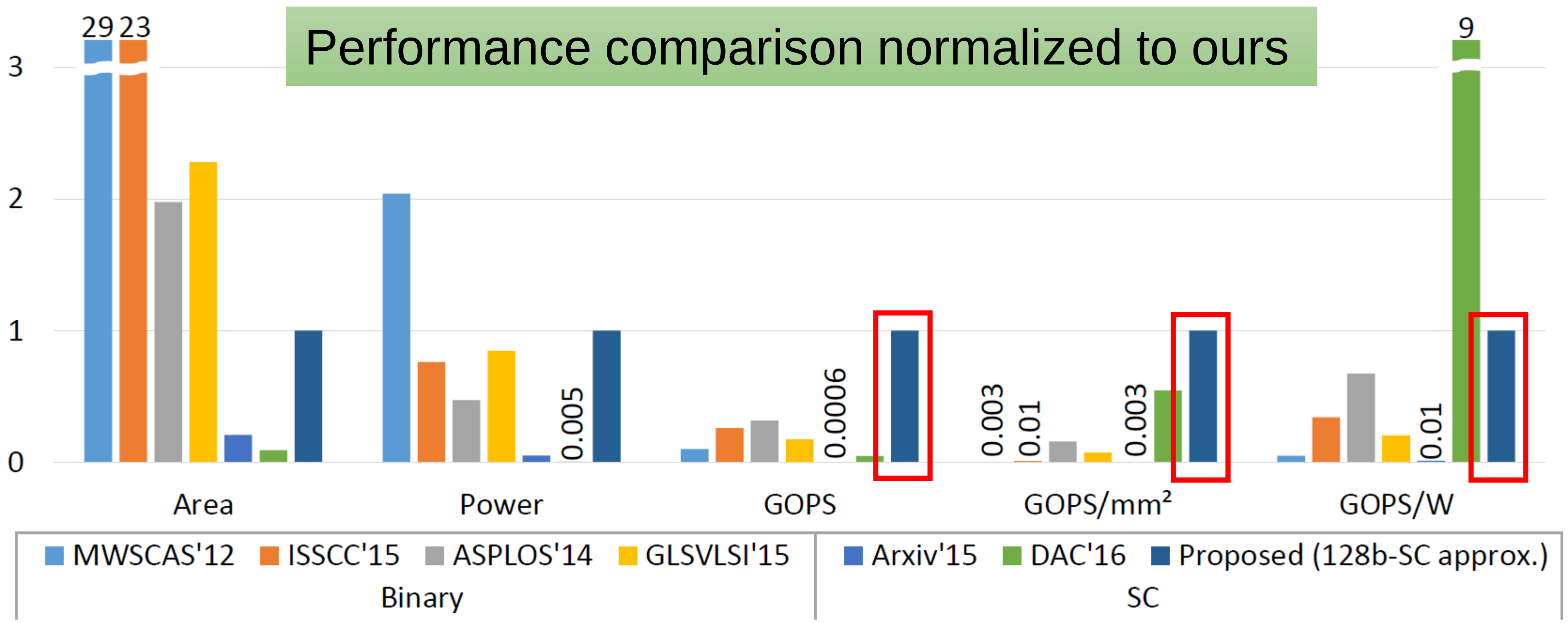
27x



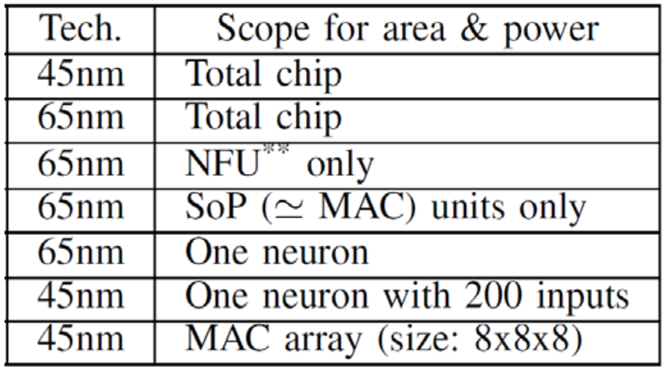
Normalized area-delay product
($T_R \times T_C \times T_M$: 8x8x8)

# Results

## - Comparison with State-of-the-art DNN processors



- Best MAC throughput (per area)
- High MAC throughput per power = Operation per energy



| Tech. | Scope for area & power |
|---|---|
| 45nm | Total chip |
| 65nm | Total chip |
| 65nm | NFU** only |
| 65nm | SoP ($\simeq$ MAC) units only |
| 65nm | One neuron |
| 45nm | One neuron with 200 inputs |
| 45nm | MAC array (size: 8x8x8) |

## Conclusion

- Present the first SC-ConvNet accelerator that is **fully scalable** thanks to **Binary-Interlaced SC.**
- Propose highly optimized **bit-parallel SC-MAC** design, which minimizes conversion overhead.
- Propose **Hybrid Layer Composition**, which improves runtime-accuracy trade-off by 8X. (1024→128 for MNIST)
- Achieve **1.83X** perf./area compared to state-of-the-art (DAC'16) SC deep learning accelerator.

## Thank you for listening my presentation

QnA?