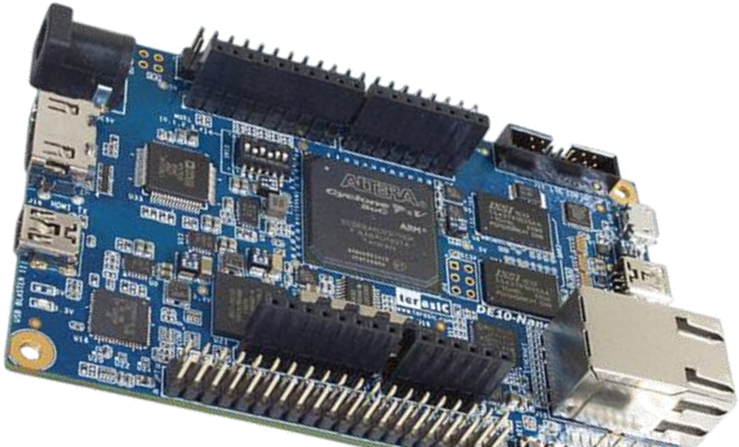
SC-DNN

Deep Neural Network using Stochastic Computing

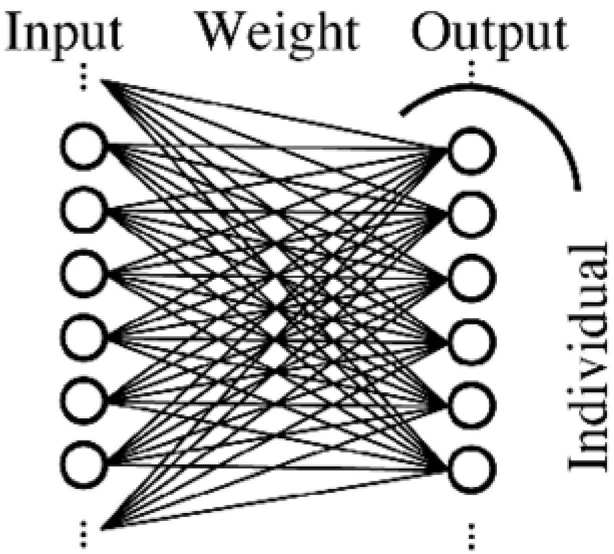
PR144
B04901112 錢柏均
B04901113 吳翊玄
B04901122 何榮晟

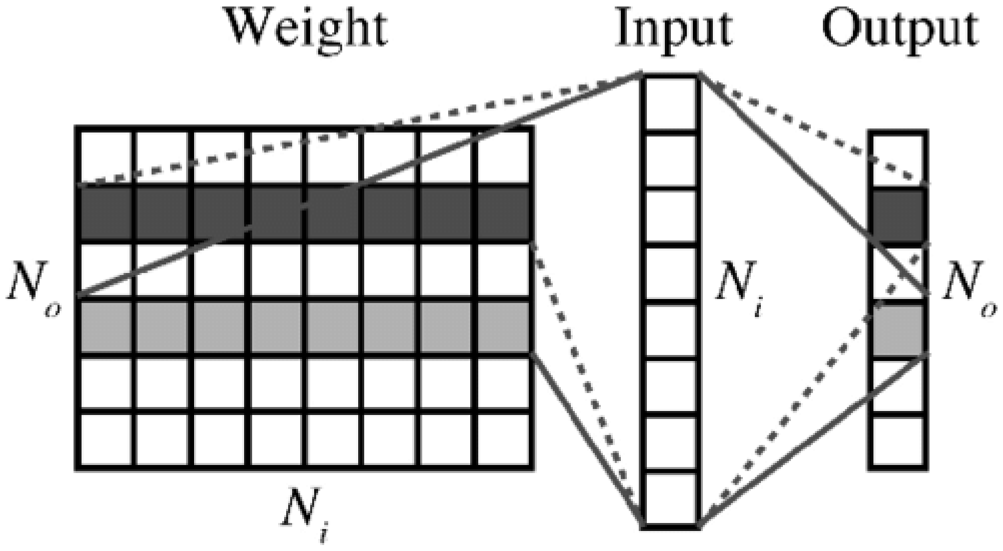
Deep Learning

- High computation effort



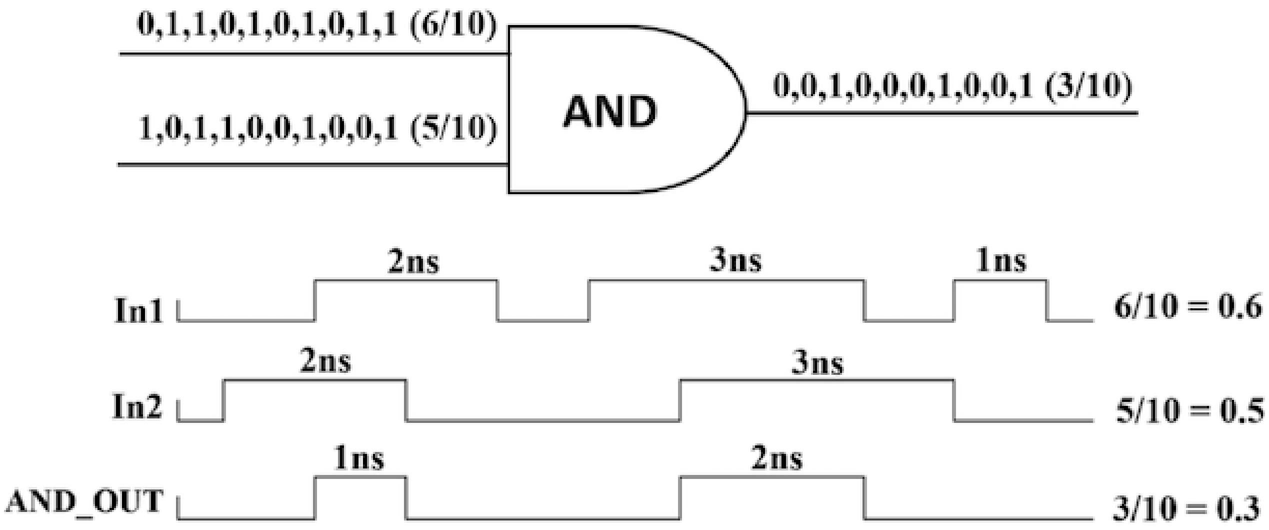
Mat-Vec Multiply





Stochastic Computation

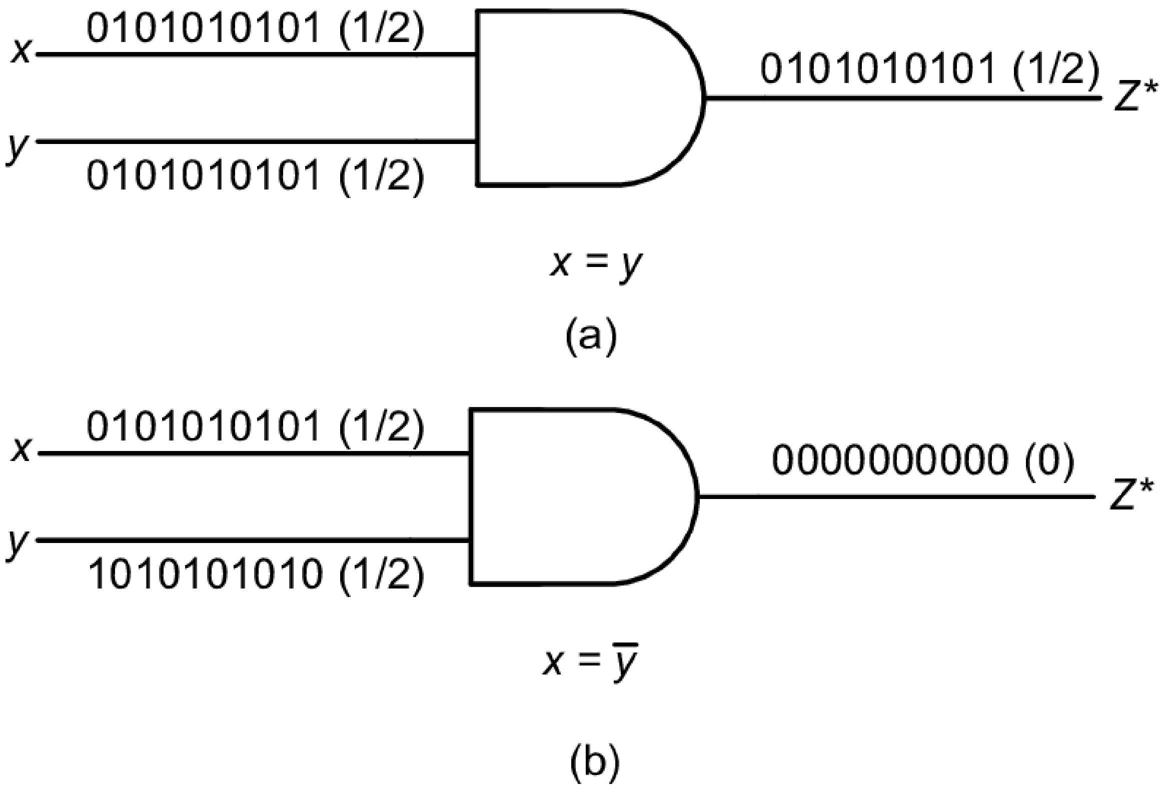
- Unipolar



Multiplier → a single AND gate!!

Bit-stream Correlation

- Non-ideal characteristic



$x = y$

(a)

$x = \overline{y}$

(b)

Pros and Cons

👍

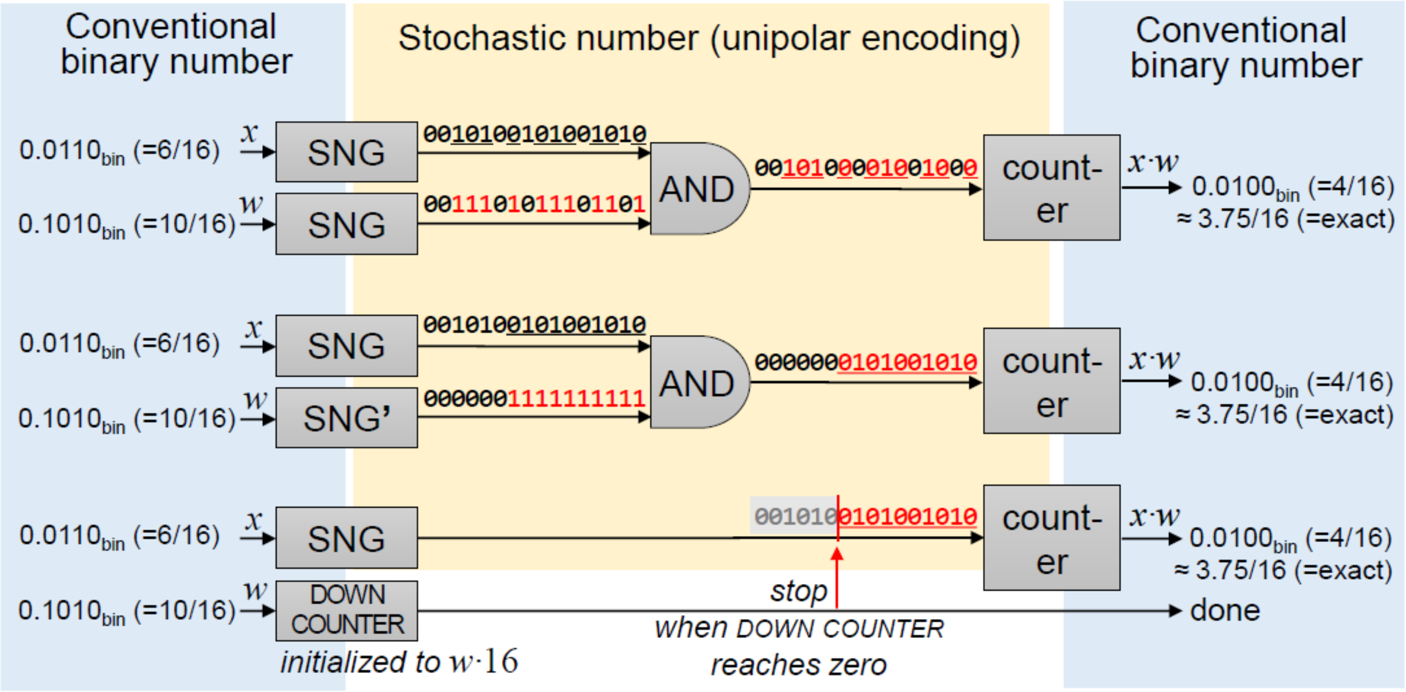
- Smaller LE usage
- Low power consumption
- Higher error (bit-flip) tolerance

👎

- Longer latency
- Not accurate
- Need conversion

SC-Multiplier

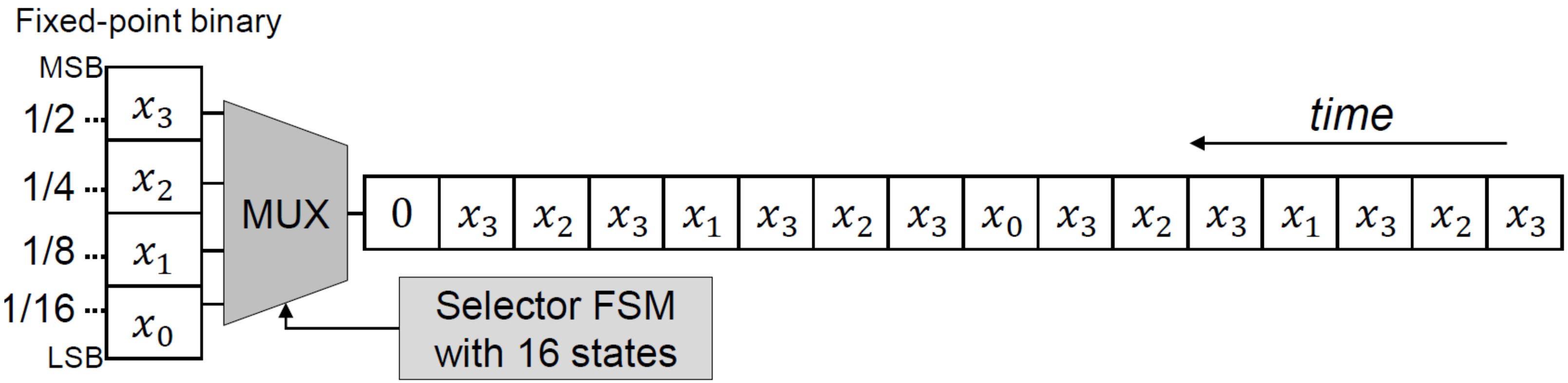
- DAC 2017



Reference:
Sim, Hyeonuk, and Jongeun Lee. "A New Stochastic Computing Multiplier with Application to Deep Convolutional Neural Networks." *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017.

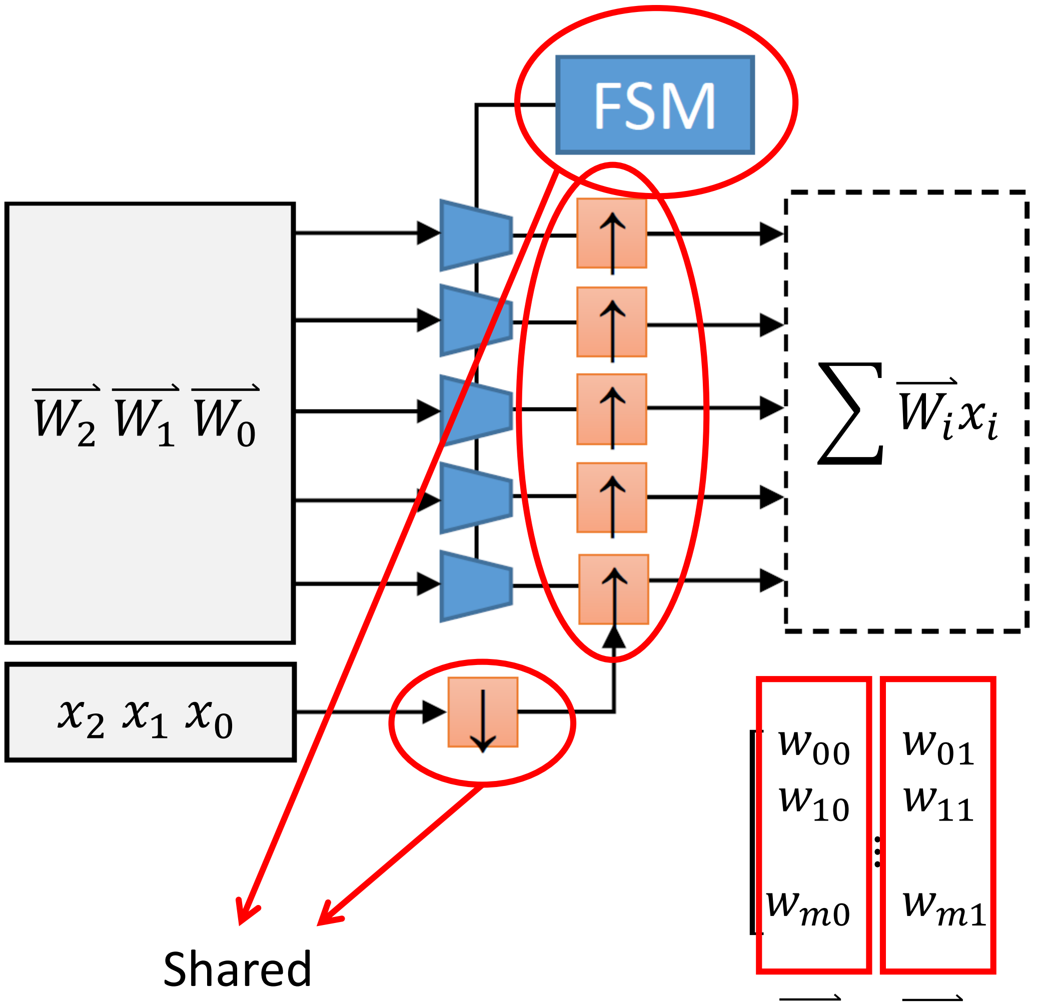
Serial MUX

A deterministic way to generate bit stream:



x_{N-i} first appears at cycle 2^{i-1}, and thereafter in every 2^i cycles, yields a theoretical maximum error: $N/2^{N+1}$ for wx

SC-Multiplier


FSM
$\overrightarrow{W_2}\,\overrightarrow{W_1}\,\overrightarrow{W_0}$
$x_2\ x_1\ x_0$
$\sum \overrightarrow{W_i} x_i$

Shared

▶ Multiplexer

↑ Up-down counter

↓ Down counter

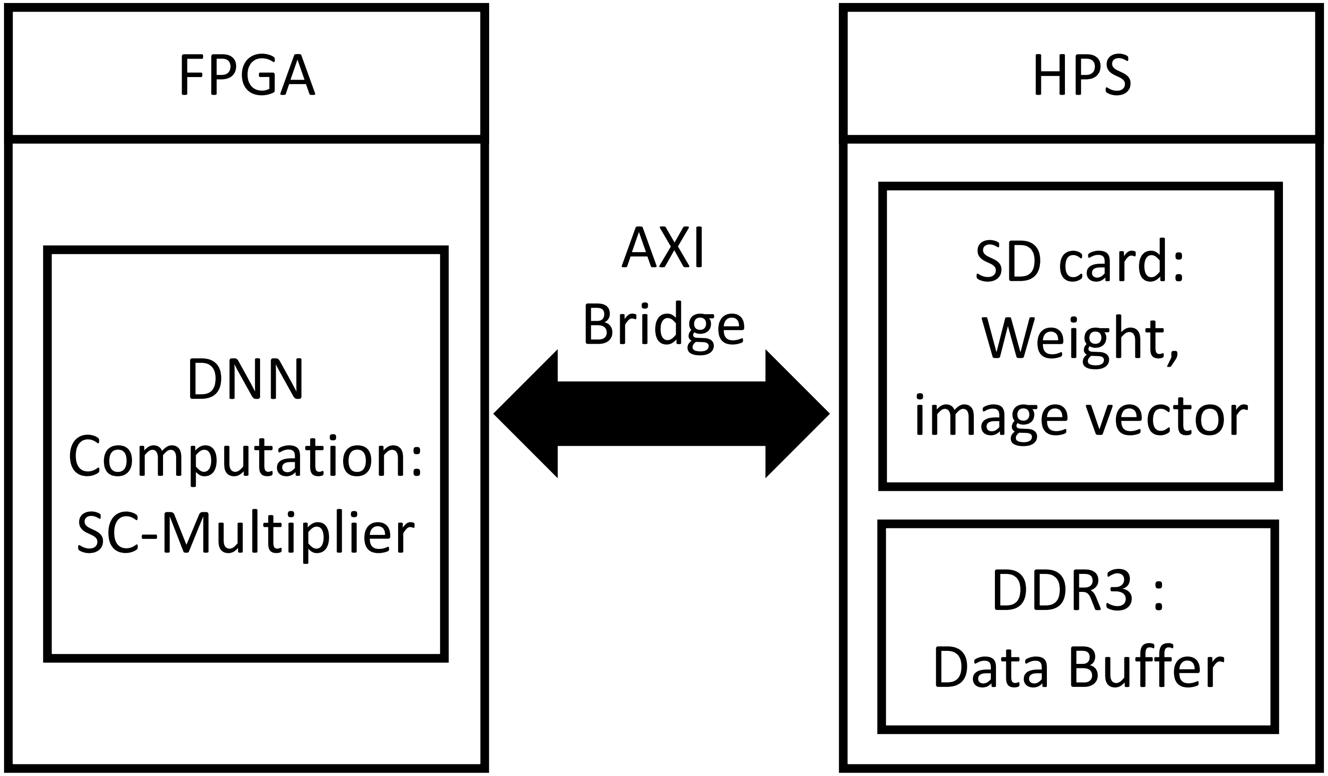
m: hidden dimension

n: input size

$$\begin{bmatrix} w_{00} & w_{01} & \cdots & w_{0n}\\ w_{10} & w_{11} & & w_{1n}\\ \vdots & & \ddots & \vdots\\ w_{m0} & w_{m1} & \cdots & w_{mn}\end{bmatrix}\begin{bmatrix}x_0\\ x_1\\ \vdots\\ x_n\end{bmatrix}=\begin{bmatrix}y_0\\ y_1\\ \vdots\\ y_m\end{bmatrix}$$

$\overrightarrow{W_0}$ $\overrightarrow{W_1}$

Soft/Hard-ware partition



FPGA

DNN Computation: SC-Multiplier

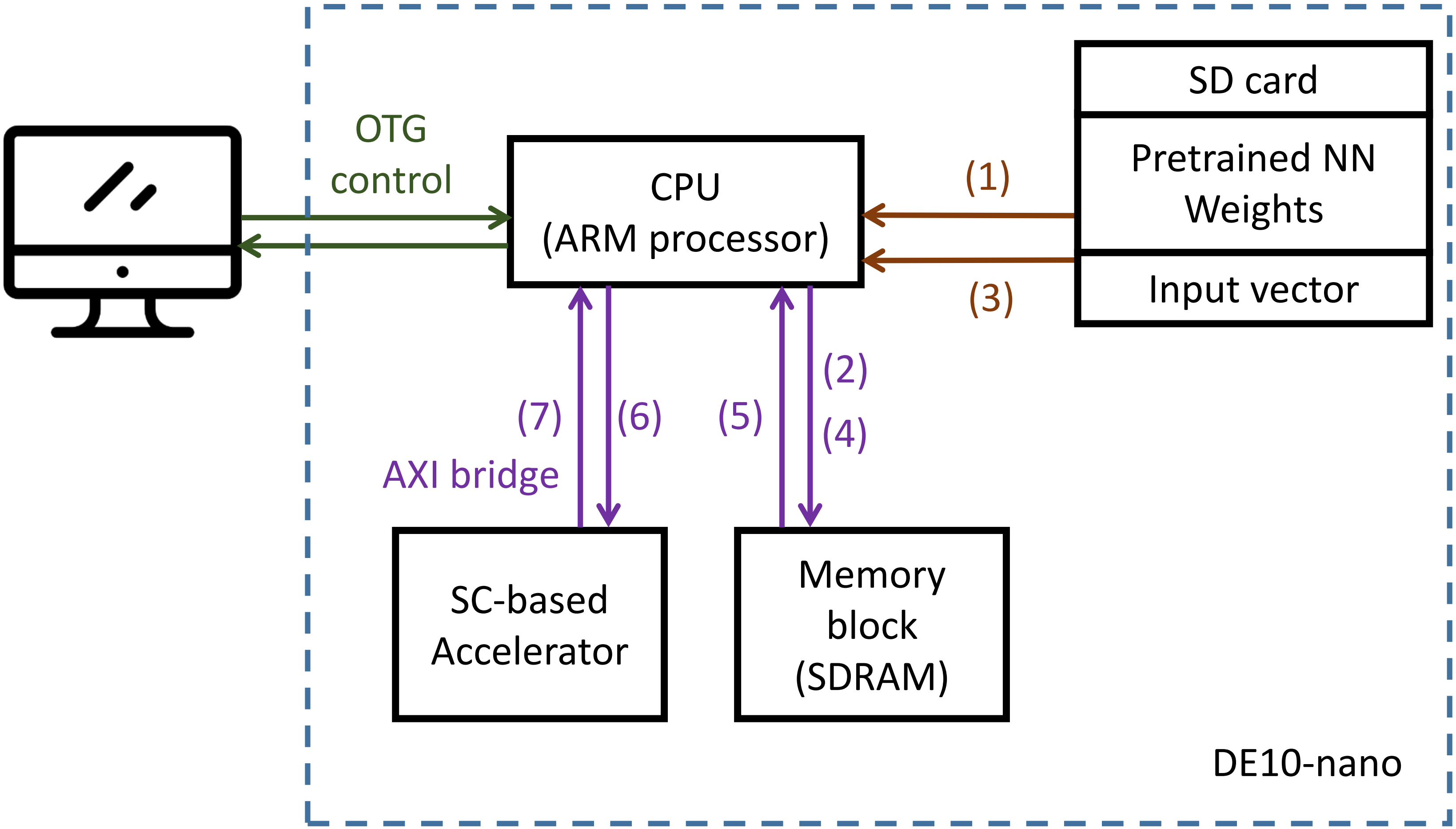
AXI Bridge

HPS

SD card: Weight, image vector

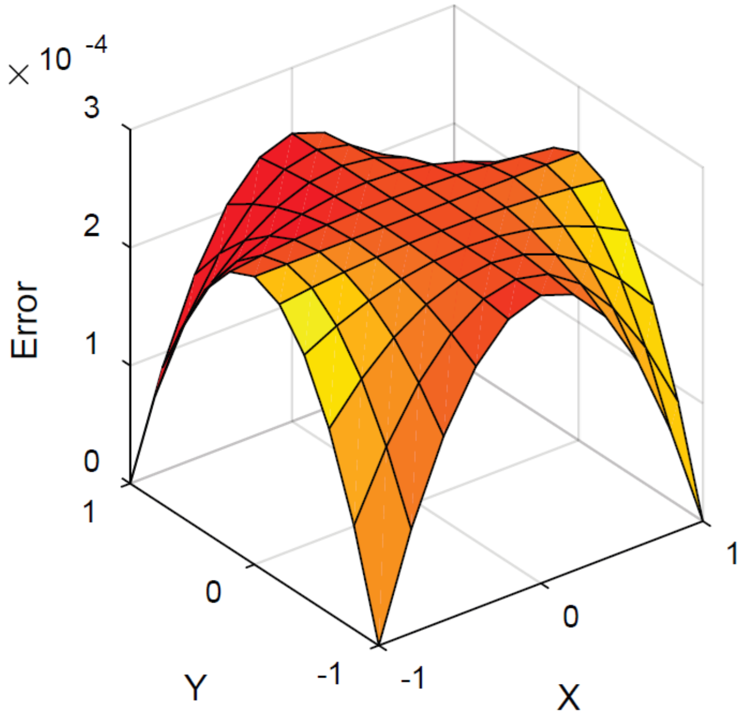
DDR3 : Data Buffer

Workflow

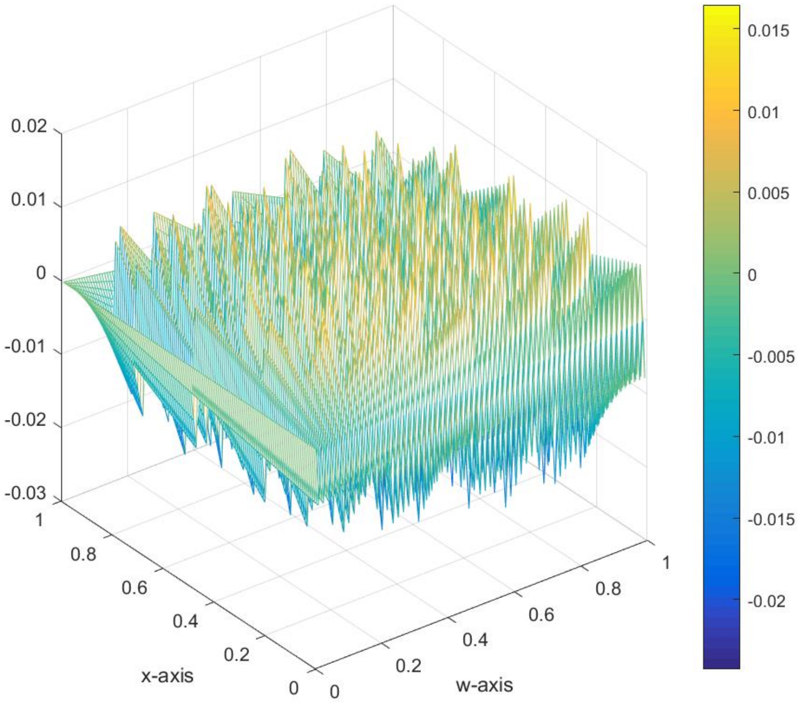

OTG
control
CPU
(ARM processor)
SD card
Pretrained NN
Weights
Input vector
(1)
(3)
(2)
(4)
(5)
(6)
(7)
AXI bridge
SC-based
Accelerator
Memory
block
(SDRAM)
DE10-nano

Error Surface

Convectional SC (1024-bit stream)



DAC 2017 SC (128-bit stream)



Reference:
Kim, Kyounghoon, et al. "Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks." *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016.

Experiment

- mnist DNN model (tensorflow):

784 → 50 → 50 → 50 → 10

- 8-bit fixed point precision, range -1~1

$$\frac{-128}{128}, \frac{-127}{128} \dots 0 \dots \frac{126}{128}, \frac{127}{128}$$

- Overall testing accuracy: 94.41%

Results

Platform	Accuracy
FPGA	9,107/10,000 (91.07%)
software	9,441/10,000 (94.41%)

a significant 3% accuracy drop!!
→ longer bit stream might be able to fix it

Timing Analysis

Theoretical (simplified, other cpu operation not included)

- Transmission (32-bit AXI bridge, 50MHz)

  HPS to FPGA: $(784+50+50+50)\times 13 = 12142\ (\mathrm{c.c.})$

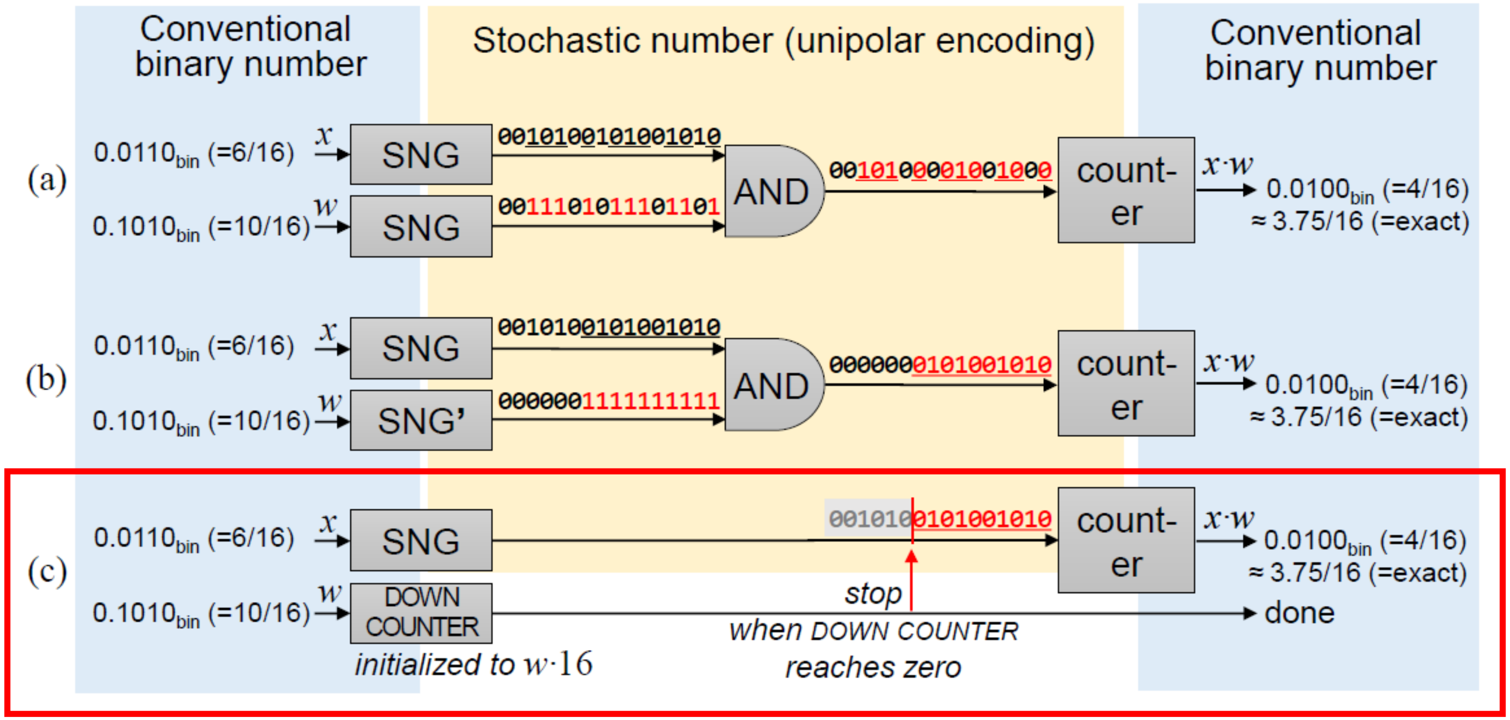
  FPGS to HPS: $(784+50+50+50)\times(50+51) = 94334\ (\mathrm{c.c.})$

  → $106476\ c.c. = \ 2.13ms$

- Calculation

Recall:
SC-Multiplier

- DAC 2017



Reference:
Sim, Hyeonuk, and Jongeun Lee. "A New Stochastic Computing Multiplier with Application to Deep Convolutional Neural Networks." *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017.

Timing Analysis

Theoretical (simplified, other cpu operation not included)

- Transmission (32-bit AXI bridge, 50MHz)

  HPS to FPGA: $(784 + 50 + 50 + 50) \times 13 = 12142\ (\mathrm{c.c.})$

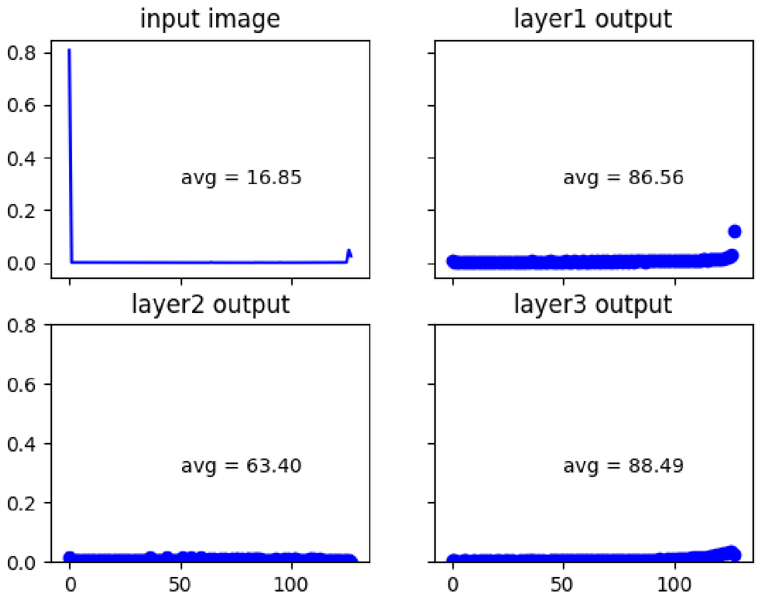
  FPGS to HPS: $(784 + 50 + 50 + 50) \times (50 + 51) = 94334\ (\mathrm{c.c.})$

  $\rightarrow 106476\ c.c. = \boxed{2.13ms}$

- Calculation

  $$784 \times 16.85 + 50 \times (86.56 + 63.40 + 88.49) = 25132.9\ (\mathrm{c.c.}) = \boxed{0.50ms}$$

$$\boxed{latency = 2.63ms}$$



Timing Analysis

Experimental value

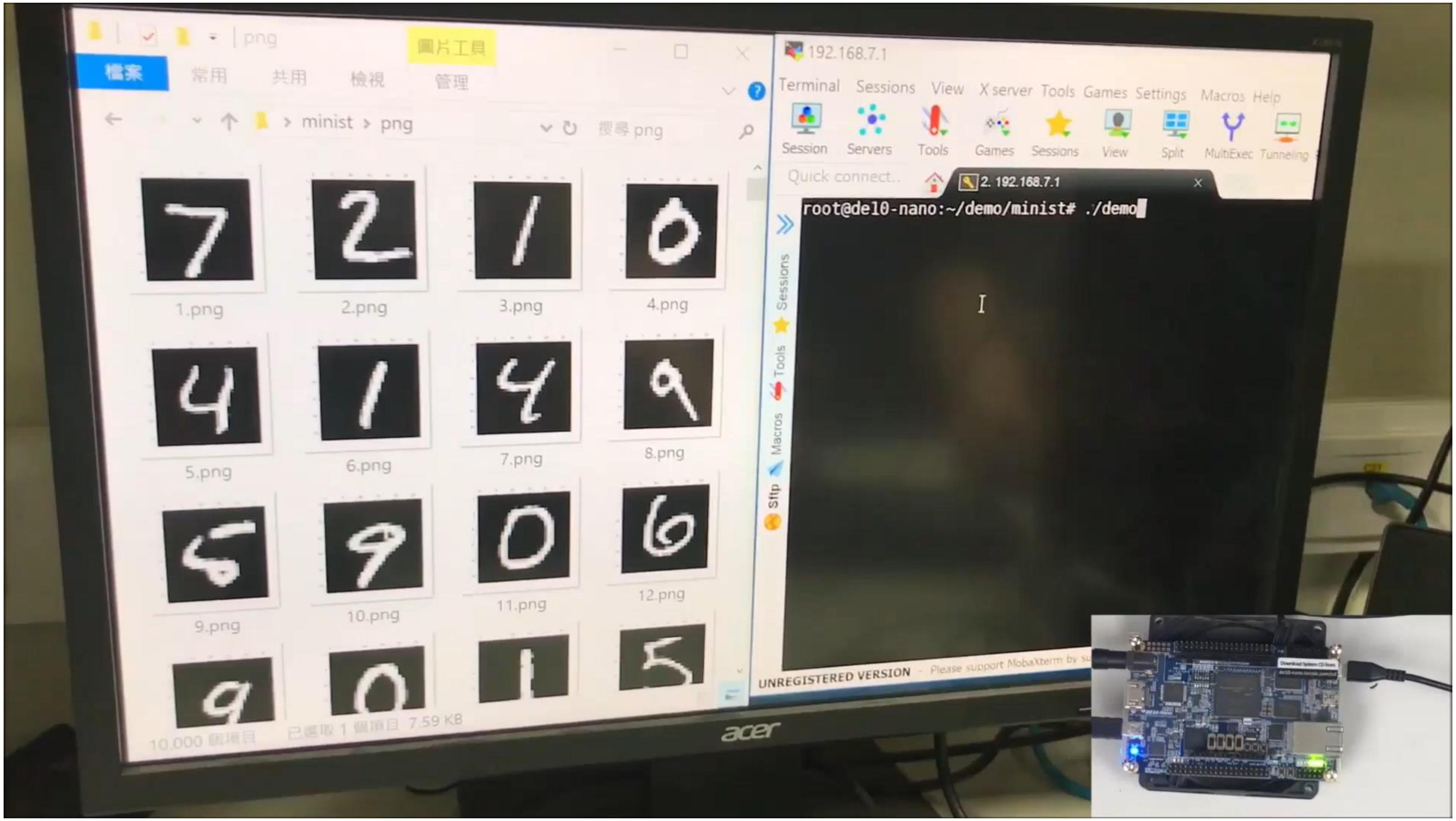
Operation	load	Layer1		Layer2		Layer3		Layer4		other	total
		write	read	write	read	write	read	write	read		
Time (ms)	0.34	12.0	0.056	0.84	0.052	0.81	0.051	0.84	0.051	0.21	15.22
(%)	2.23	78.8	0.37	5.52	0.34	5.32	0.34	5.52	0.34	1.38	100

Performance bottleneck

Possible solutions:

- Wider AXI bridge
- Better data reuse (Conv.)

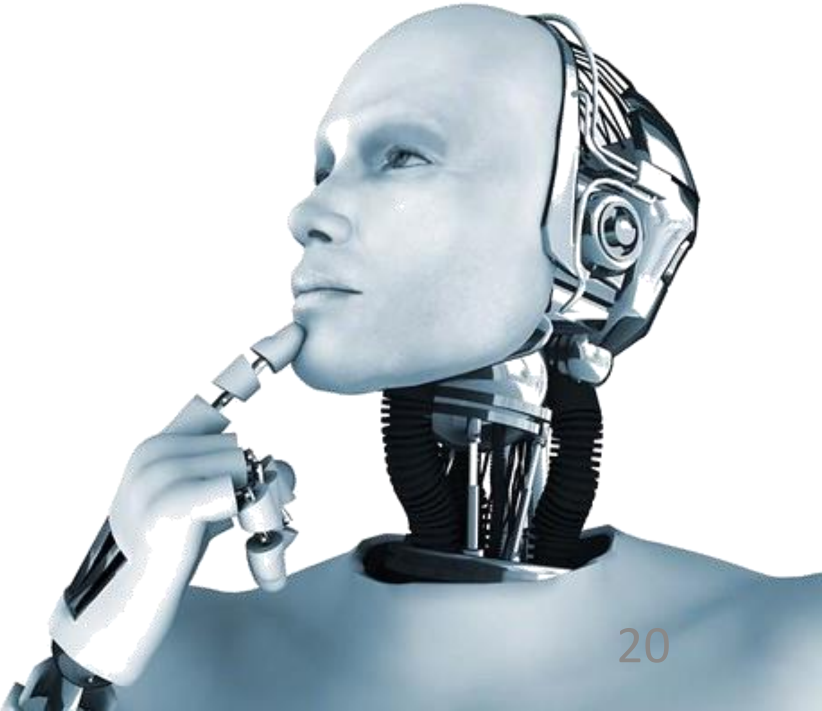
Demo



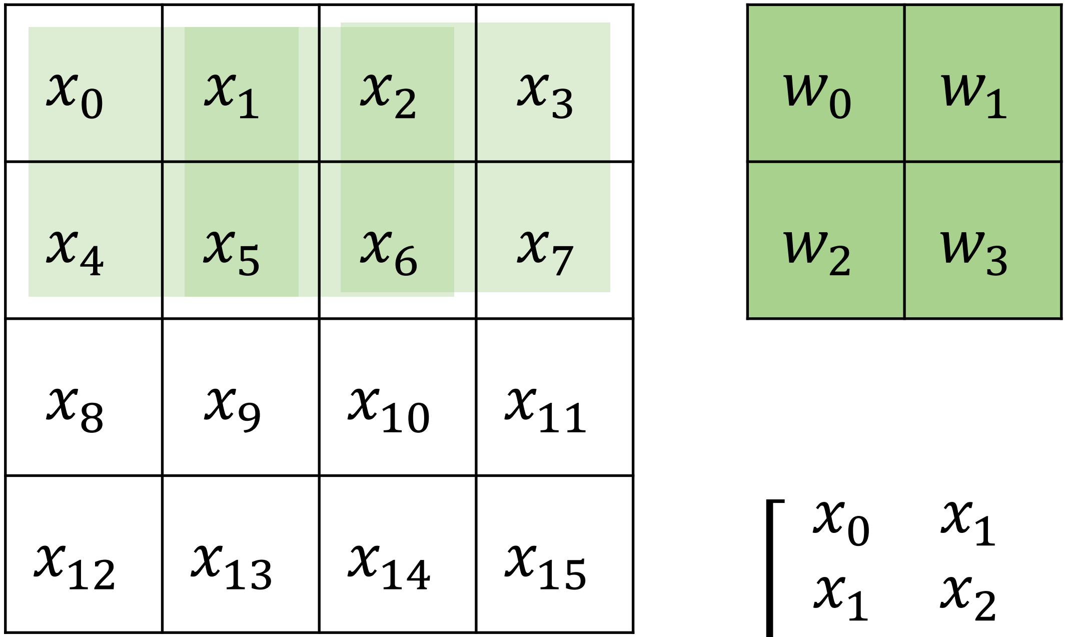
```
root@de10-nano:~/demo/minist# ./demo
```

Future plan

- Convolution layer



Convolution



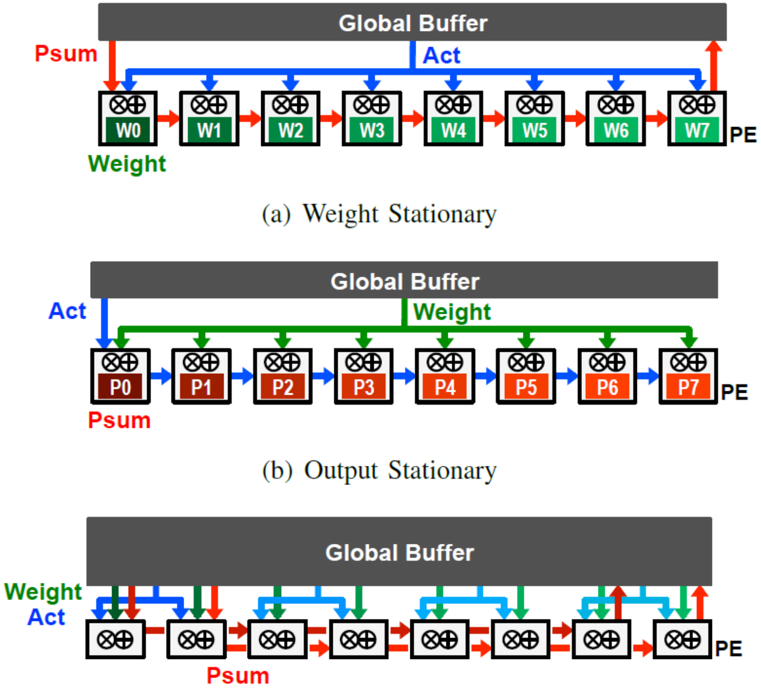
x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}

w_0	w_1
w_2	w_3

$$\begin{bmatrix} x_0 & x_1 & \cdots & x_5 \\ x_1 & x_2 & & x_6 \\ \vdots & & \ddots & \vdots \\ x_{10} & x_{11} & \cdots & x_{15} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_8 \end{bmatrix}$$

Convolution

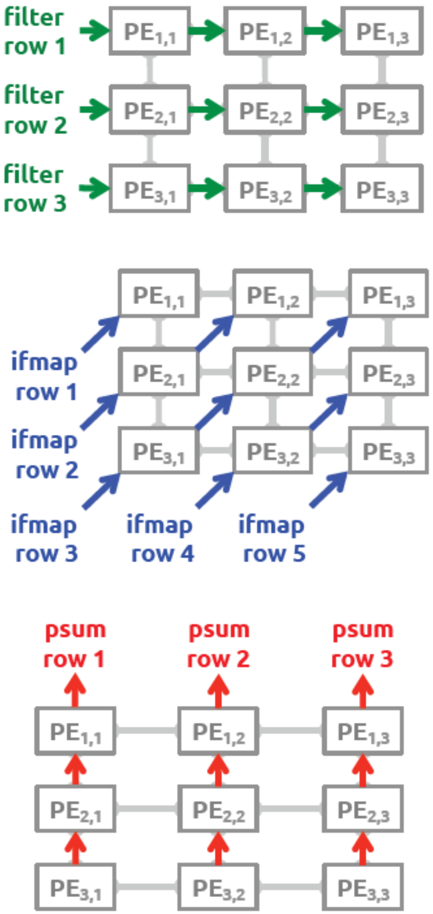
- Data reuse scheme



(a) Weight Stationary

(b) Output Stationary

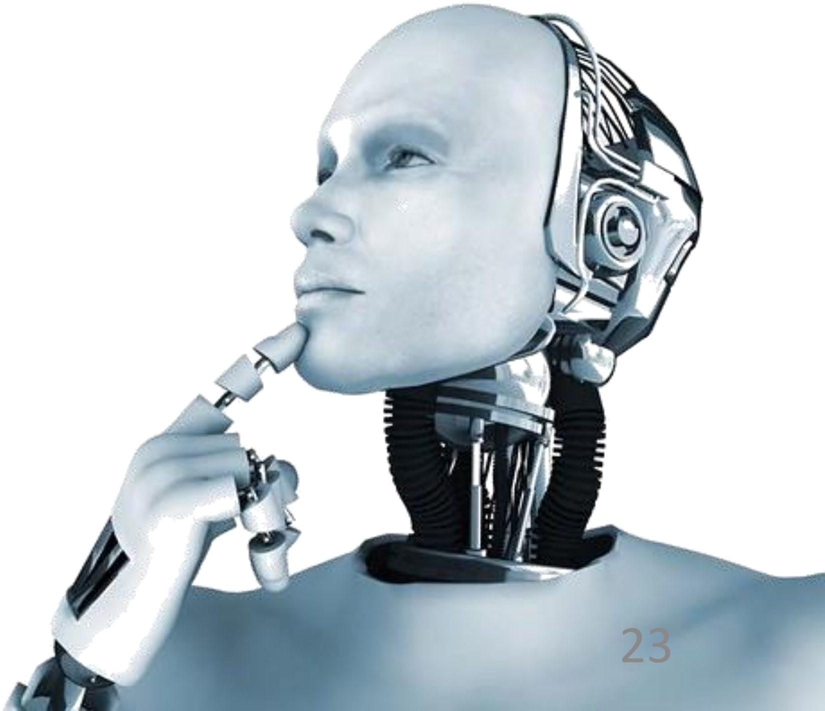
(c) No Local Reuse



Reference:
Y.-H. Chen, J. Emer, V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," International Symposium on Computer Architecture (ISCA), pp. 367-379, June 2016.

Future plan

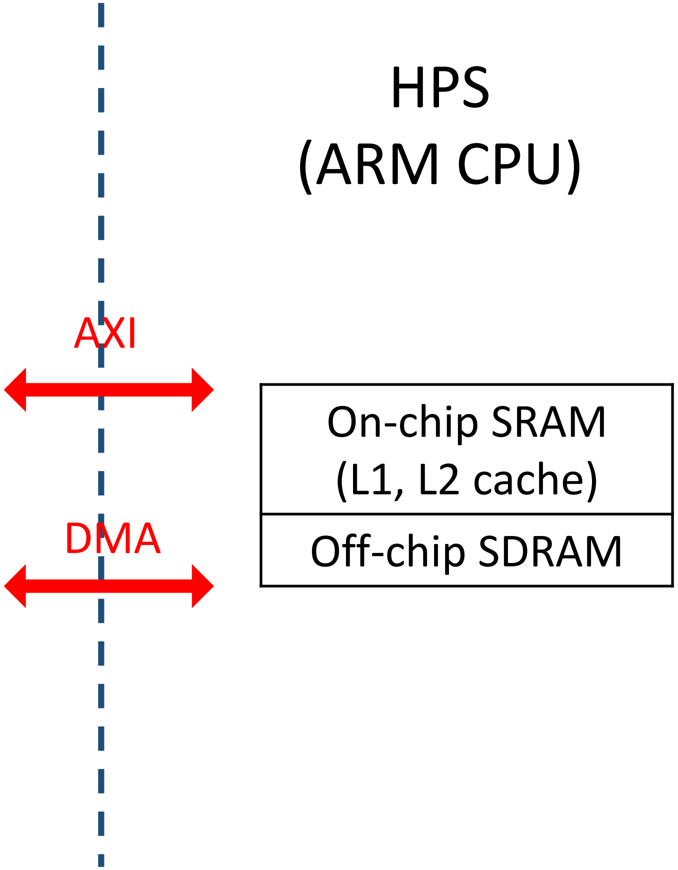
- Convolution layer
- Memory hierarchy



Memory hierarchy

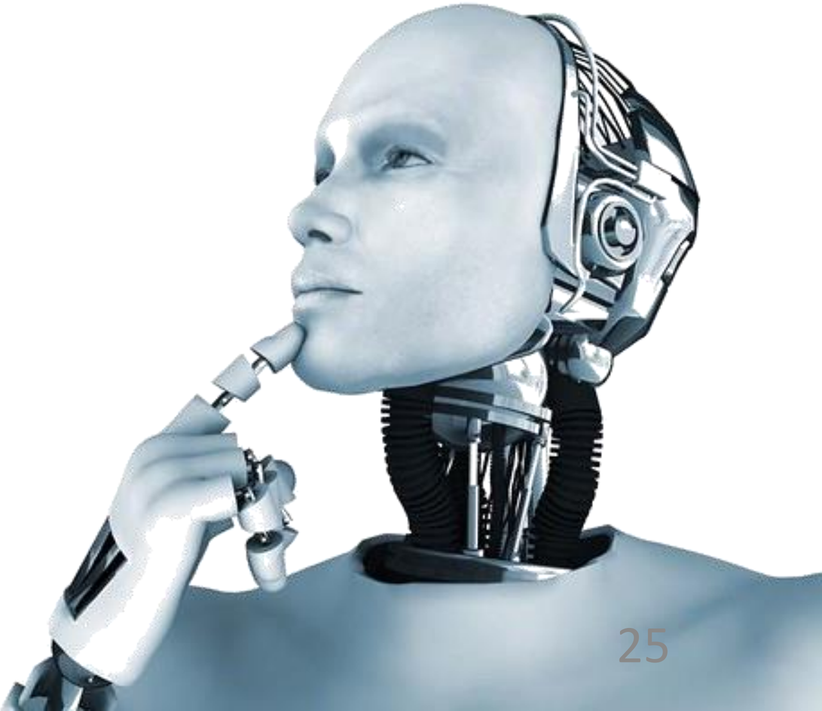
FPGA

size	memory	Eyeriss analogy
~Mb	embedded memory	global buffer
~kb	Distributed ram (composed by LEs)	RF



Future plan

- Convolution layer
- Memory hierarchy
- Integrate embedded multiplier and DSP
- OpenCL HLS



Conclusion

1. SC-based Mat-Vec multiplier

2. Performance:
   - 3% accuracy drop
   - Latency on par with CPU

3. We are new to the field of SoC/FPGA design. There many more possibilities that we’re willing to try.

The End

Thanks for listening.