A High-Throughput and Energy-Efficient RRAM-based Convolutional Neural Network using Data Encoding and Dynamic Quantization

Xizi Chen, Jingbo Jiang, Jingyang Zhu and Chi-Ying Tsui

The Hong Kong University of Science & Technology, Hong Kong

ASP-DAC 2018

Outline

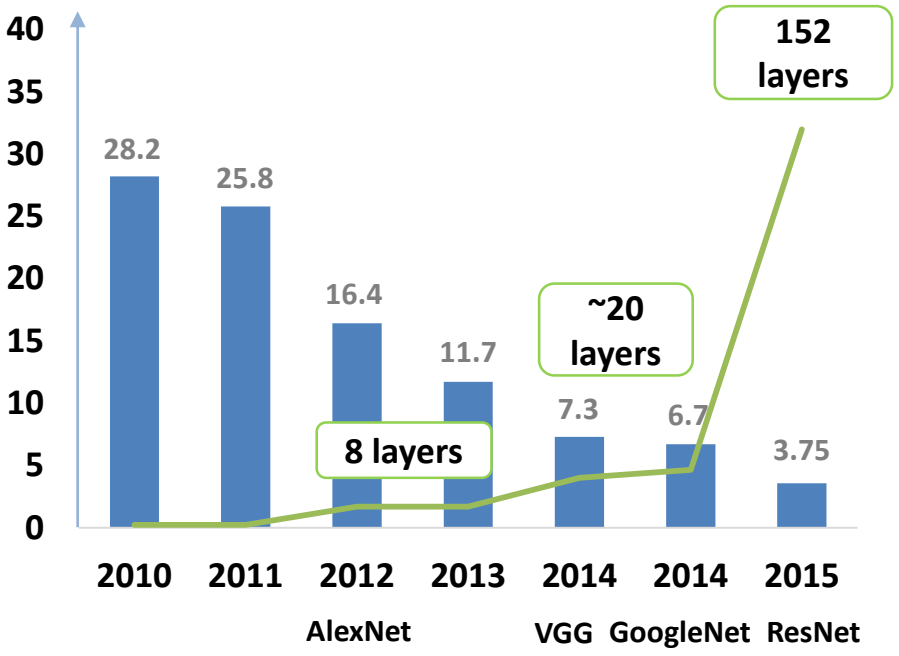
- Introduction
- Hardware architecture and related works
- Segmented compression encoding
- Optimizing the bit-resolution of the A/D interface
- MAC rescheduling based on the dynamic quantization
- Experimental result
- Conclusion

Outline

- Introduction
- Hardware architecture and related works
- Segmented compression encoding
- Optimizing the bit-resolution of the A/D interface
- MAC rescheduling based on the dynamic quantization
- Experimental result
- Conclusion

Introduction

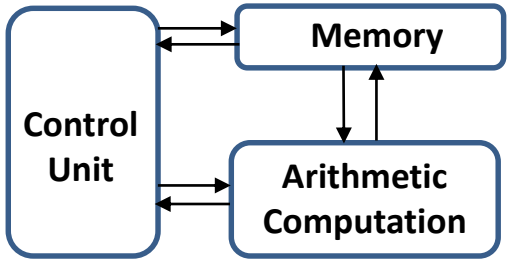
Top-5 Error Rate (%) *vs* Depth (ILSVRC)



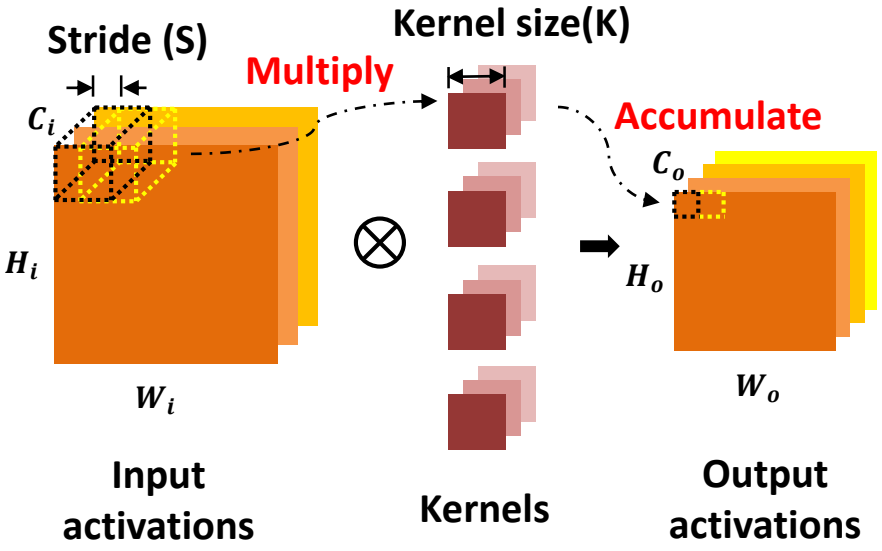
Large memory requirement & intensive computations

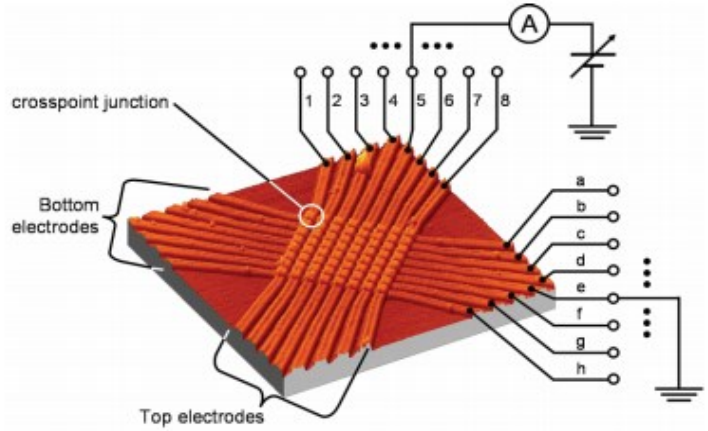
# of layers	# of parameters	# of multiply-accumulate computations per image
152	60M	11G

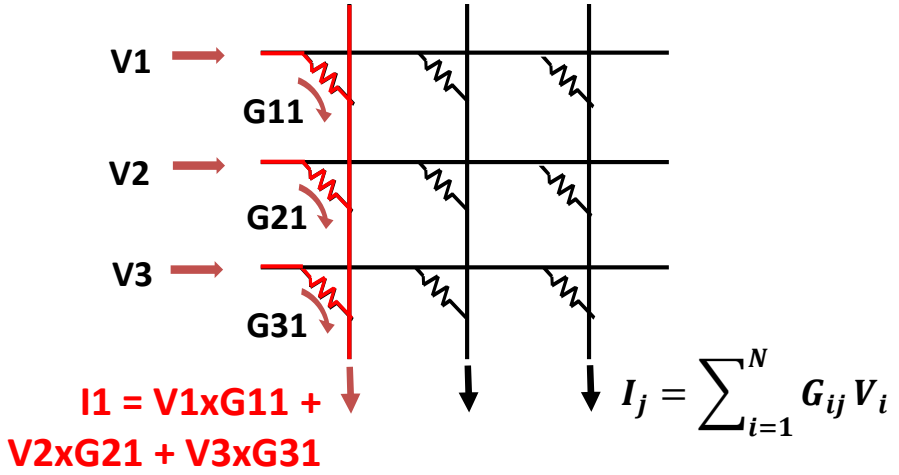
Conventional CMOS-based Architecture



RRAM-based Multiply-Accumulate Computation







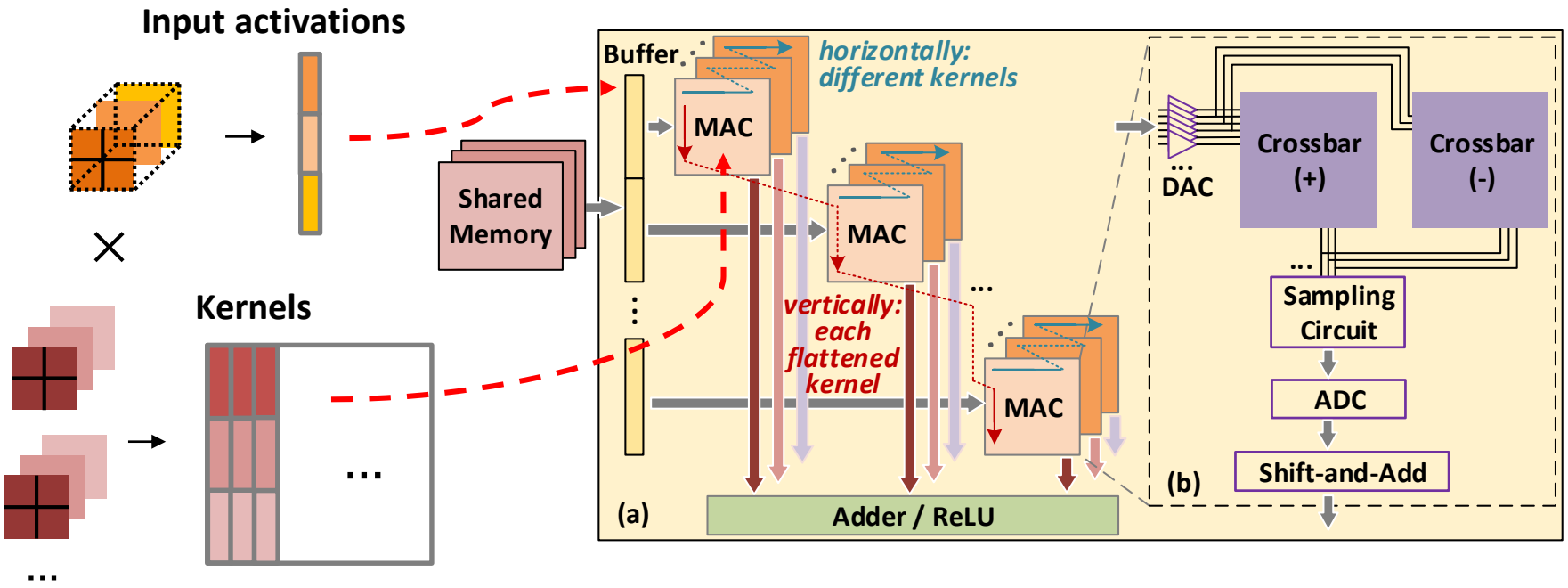
In-situ computation based on RRAM : an effective approach to the memory wall

C. Nauenheim, *et al*. Integration of resistive switching devices in crossbar structures, 2017

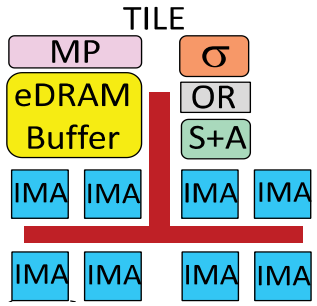
Outline

- Introduction
- Hardware architecture and related works
- Segmented compression encoding
- Optimizing the bit-resolution of the A/D interface
- MAC rescheduling based on the dynamic quantization
- Experimental result
- Conclusion

Hardware Architecture



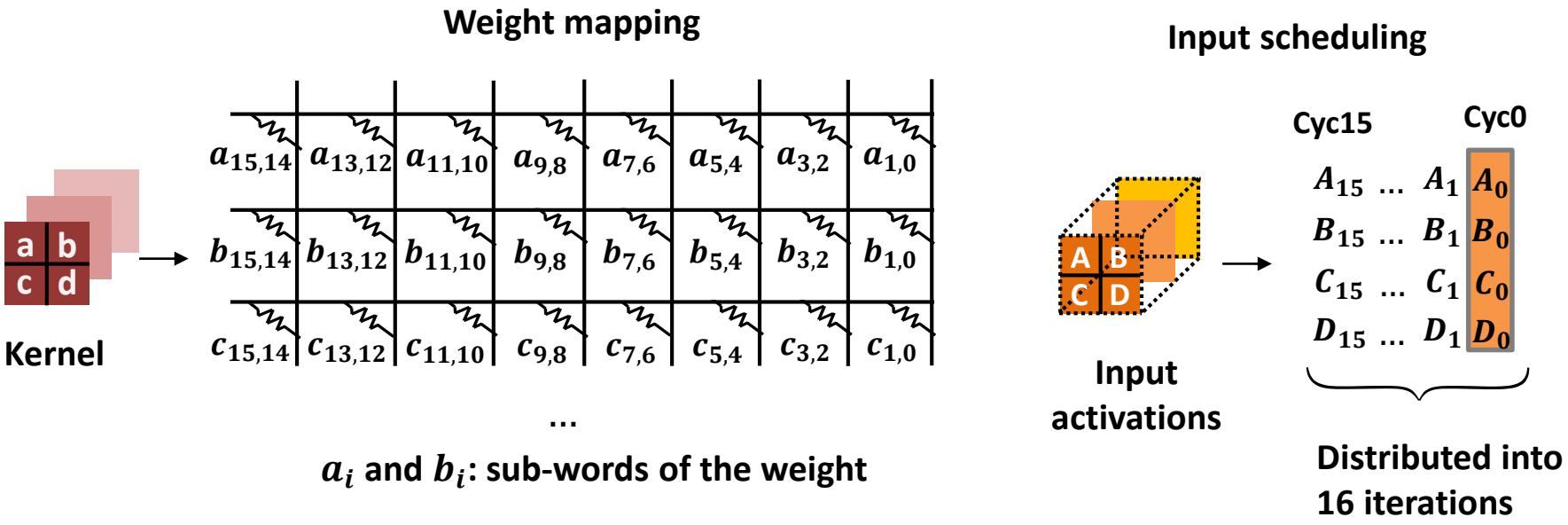
- State-of-the-art architecture: *ISAAC* [1]
  - Hierarchical structure
  - 3-stage pipeline



[1] A. Shafiee, *et al*. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ISCA*, 14-26, 2016.

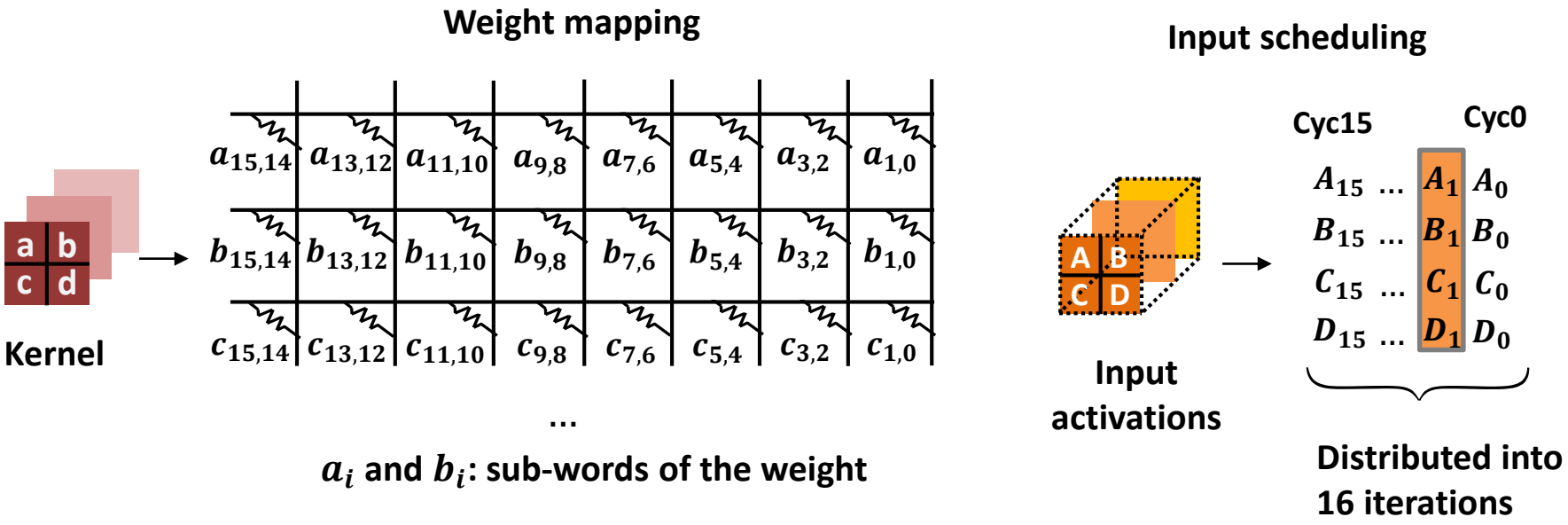
Weight Mapping & Input Scheduling

- Splitting the weights and input activations into sub-words
  - Limited precision of RRAM cells
  - High resolution DAC and ADC are costly in terms of power and area



Weight Mapping & Input Scheduling

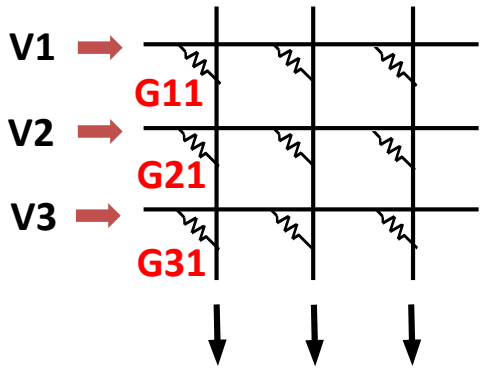
- Splitting the weights and input activations into sub-words
  - Limited precision of RRAM cells
  - High resolution DAC and ADC are costly in terms of power and area



Outline

- Introduction
- Hardware architecture and related works
- Segmented compression encoding
- Optimizing the bit-resolution of the A/D interface
- MAC rescheduling based on the dynamic quantization
- Experimental result
- Conclusion

Segmented Compression Encoding (SCE)



$$I(j) = \sum_{i=1}^{N} G(i,j) \times V(i)$$

- RRAM crossbar consumes ~20% energy
- Reducing the conductance of the memristors is a natural way for power saving
- Directly reducing the range of conductance
  - Not a good idea
  - Margin becomes smaller, making the device variance and the noise of the analog computation less tolerable…

Segmented Compression Encoding (SCE)

- Reducing the energy while keeping the margin of the conductance levels unchanged

Weight: 0010_1110_1001_1100 b

Original sub-words:

sw0 = 1100 b

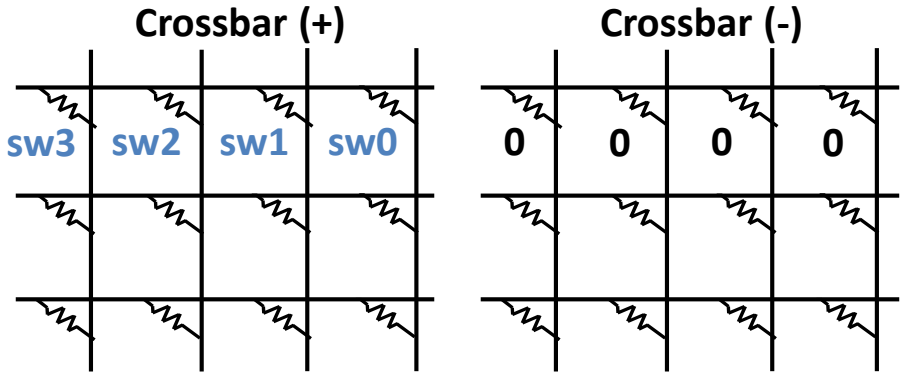
sw1 = 1001 b

sw2 = 1110 b

sw3 = 0010 b

>1000b

the middle value of 0000b~1111b



Segmented Compression Encoding (SCE)

- Reducing the energy while keeping the margin of the conductance levels unchanged

Weight: 0010_1110_1001_1100 b

sw0 = 1100 b > 1000 b

→ = 10000 b – (10000 b – 1100 b)

= 10000 b – 0100 b

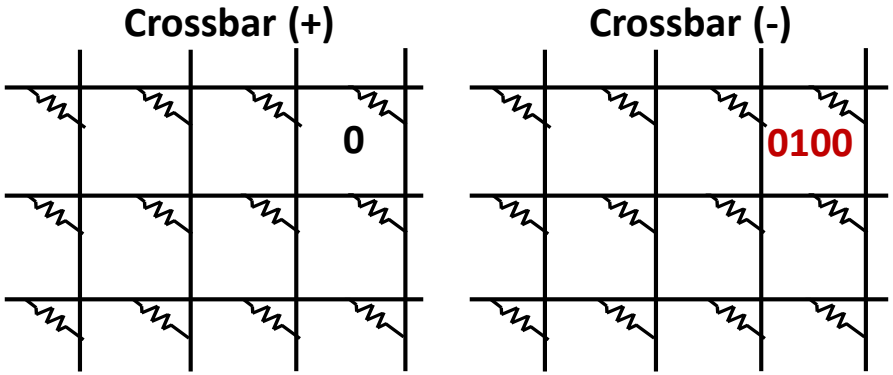
forward to Crossbar(-)

send as a carry to sw1

sw1 = 1001 + 1 = 1010 b > 1000 b

sw2 = 1110 b

sw3 = 0010 b

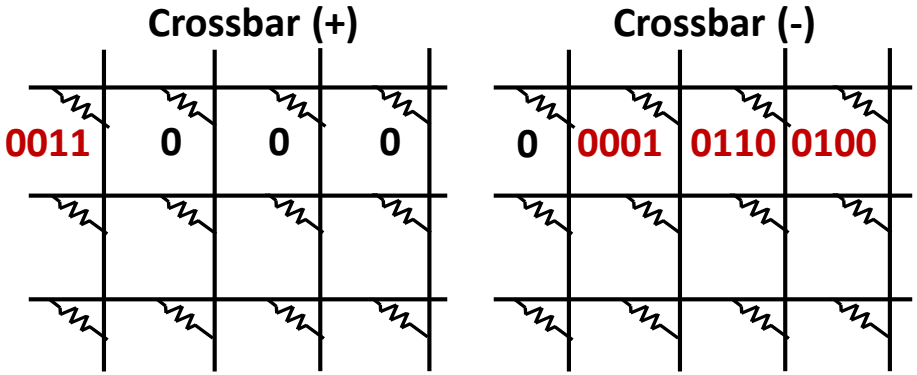


Segmented Compression Encoding (SCE)

- Reducing the energy while keeping the margin of the conductance levels unchanged

Weight: 0010_1110_1001_1100 b

sw0 = 1100 b
→ = 10000 b – (10000 b – 1100 b)
= 10000 b – 0100 b



sw1 = 1001 + 1 = 10000 b – 0110 b

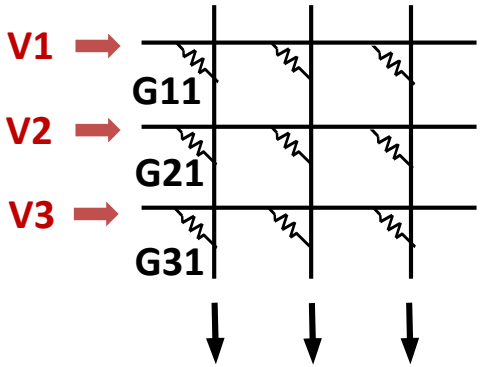
sw2 = 1110 b + 1 = 10000 b – 0001 b

sw3 = 0010 b + 1 = 0011 b

- Originally: 3 of the 4 sub-words > 1000b
- After SCE: **all the sub-words ≤ 1000b**

Segmented Compression Encoding (SCE)

- Encoding is also applicable for input activations
  - Since DAC is 1-bit, the canonic signed digit (CSD) encoding is used



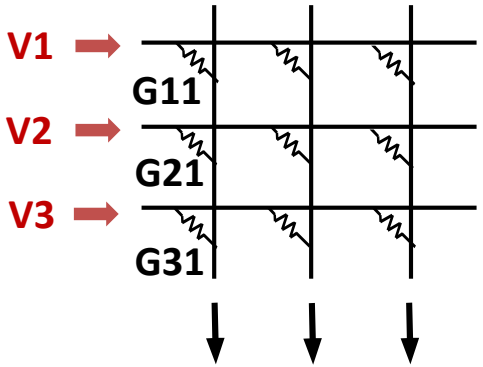
Input activation : 0010_1110_1001_1100 b

# of non-zero bits = 8

$$I(j) = \sum_{i=1}^{N} G(i,j) \times V(i)$$

Segmented Compression Encoding (SCE)

- Encoding is also applicable for input activations
  - Since DAC is 1-bit, the canonic signed digit (CSD) encoding is used



$$I(j) = \sum_{i=1}^{N} G(i,j) \times V(i)$$

Input activation : 0010_1110_1001_1100 b

# of non-zero bits = 8

eliminate the consecutive '1's from LSB

0010_1110_10$1\bar{0}$_$0\bar{1}$00 b

00$11$_00$\bar{1}$0_$10$10_$0\bar{1}$00 b

$01\bar{0}1$_00$\bar{1}$0_$10$10_$0\bar{1}$00 b

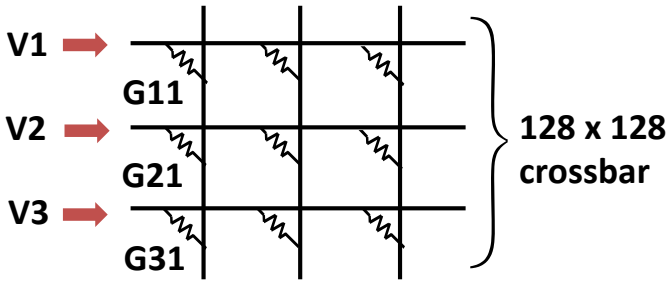
# of non-zero bits = 6

Outline

- Introduction
- Hardware architecture and related works
- Segmented compression encoding
- Optimizing the bit-resolution of the A/D interface
- MAC rescheduling based on the dynamic quantization
- Experimental result
- Conclusion

Analog-to-Digital Interface Optimization

- The high-precision A/D interface consumes ~50% energy & ~30% area



$$I(j) = \sum_{i=1}^{N} G(i,j) \times V(i)$$

- Splitting the weights and activations into sub-words
  - A 1-bit by 2-bit multiplication is done at each memristor cell

- Bit-resolution of ADC required in the worst case

$$R = log_2 128(rows) + 2(at\ each\ cell) + 1(sign)$$
$$= 10\ bit$$

Power and area increase exponentially with ADC resolution!

Analog-to-Digital Interface Optimization

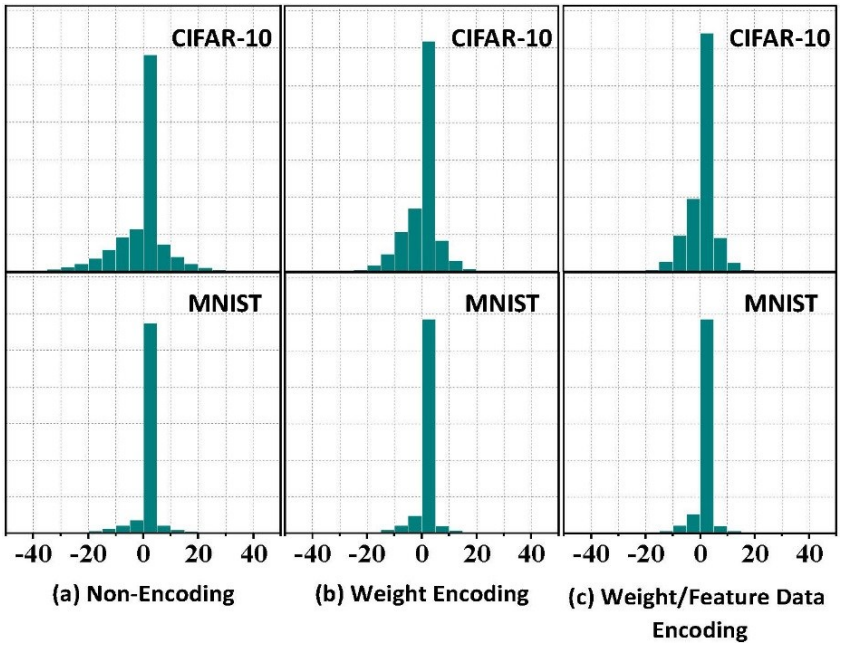
- The worst-case bit-resolution is unnecessary
  - High sparsity of activations and weights
- A 6-bit ADC is enough to capture the correct computation most of the time
  - Range provided by the 10-bit ADC

$$[-2^9, 2^9 - 1]$$

  - Around 99% of the bitline outputs

$$[-2^5, 2^5 - 1]$$

- Using SCE can further reduce the resolution by 1-bit



Distribution of the Bitline Output
(x axis: digital output y axis: intensity)

Outline

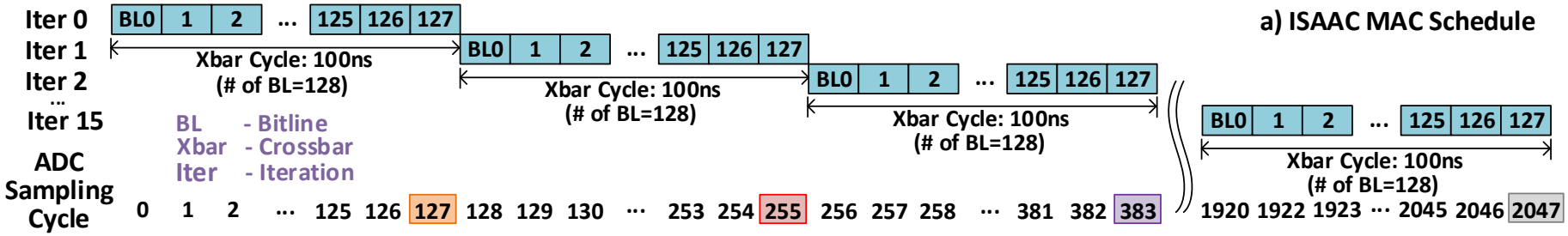
- Introduction
- Hardware architecture and related works
- Segmented compression encoding
- Optimizing the bit-resolution of the A/D interface
- MAC rescheduling based on the dynamic quantization
- Experimental result
- Conclusion

MAC Rescheduling based on the Dynamic Quantization

- MAC is the most time consuming part
  - It takes 16 iterations to serially shift in the activation

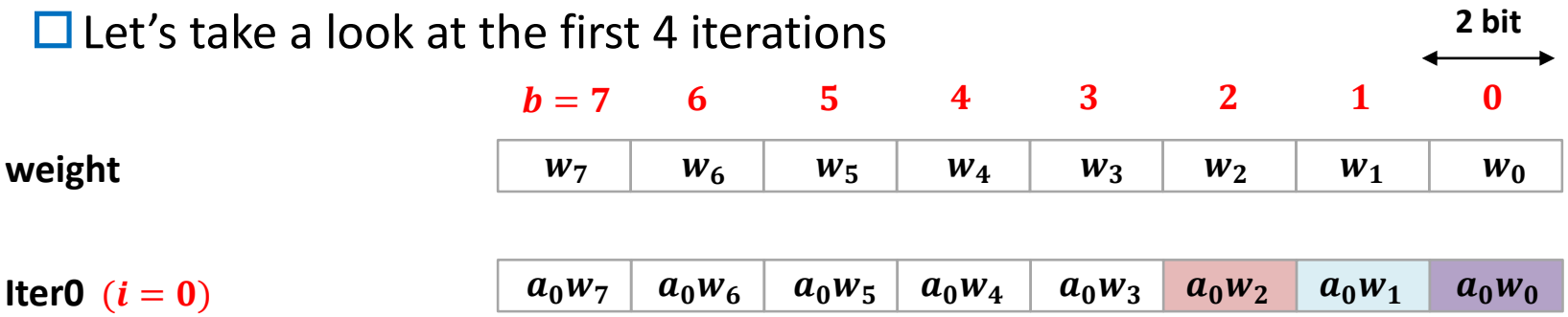
- # of A/D conversions for calculating the outputs on MAC

$$16(\#\ of\ iterations) \times 128(\#\ of\ bitlines) = 2048$$



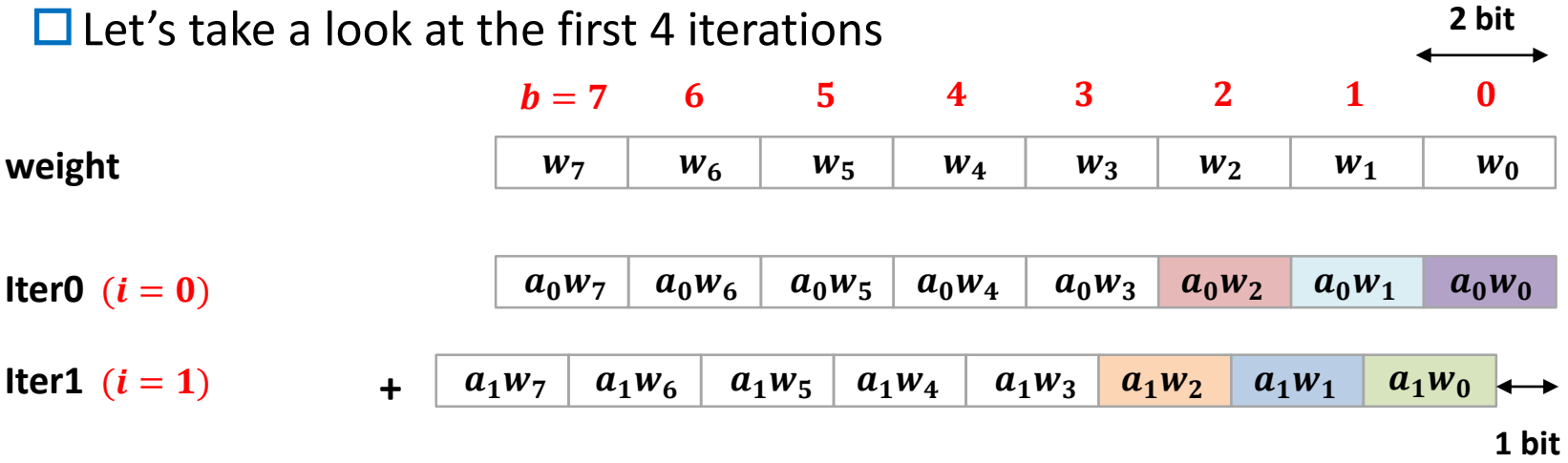
MAC Rescheduling based on the Dynamic Quantization

- Let's take a look at the first 4 iterations

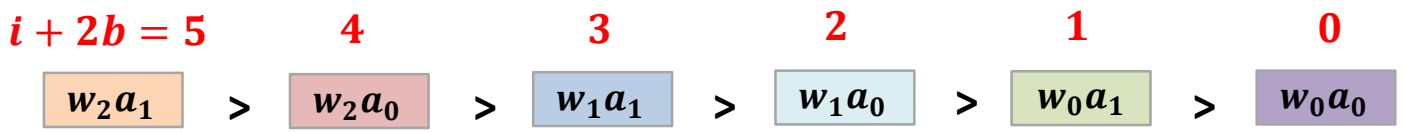


MAC Rescheduling based on the Dynamic Quantization

- Let's take a look at the first 4 iterations



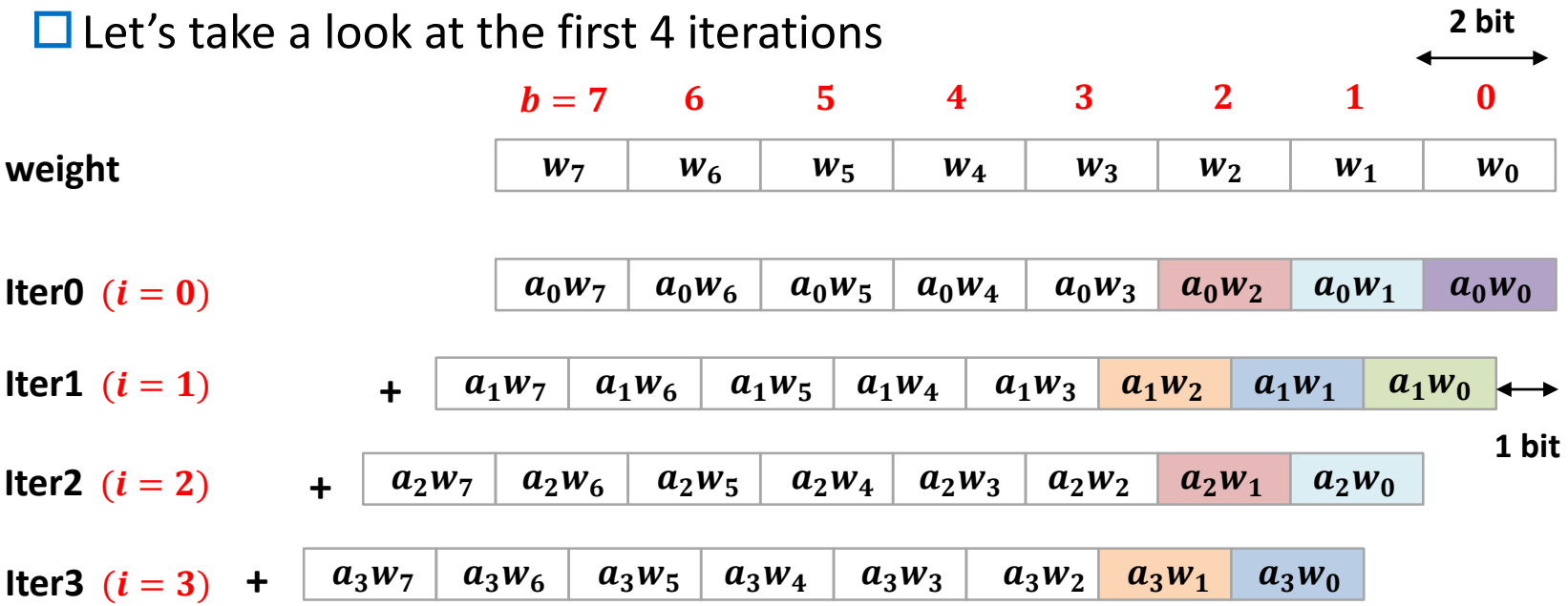
- The significance of the partial product at each cell changes with $(i + 2b)$



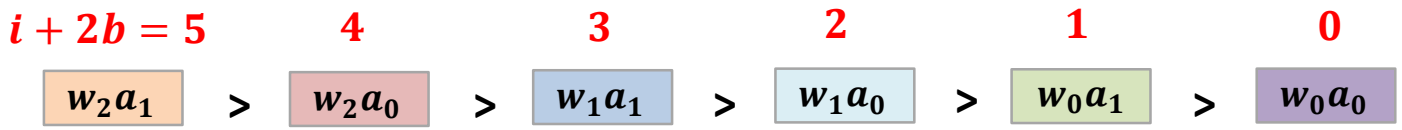
The lower the value, the less impact on the accuracy.

MAC Rescheduling based on the Dynamic Quantization

- Let's take a look at the first 4 iterations



- The significance of the partial product at each cell changes with $(i + 2b)$



The lower the value, the less impact on the accuracy.

MAC Rescheduling based on the Dynamic Quantization

- Dynamic quantization: skipping the A/D conversions of those partial products that have no significant impact on the accuracy

$\boldsymbol{i}$: i^{th} iteration

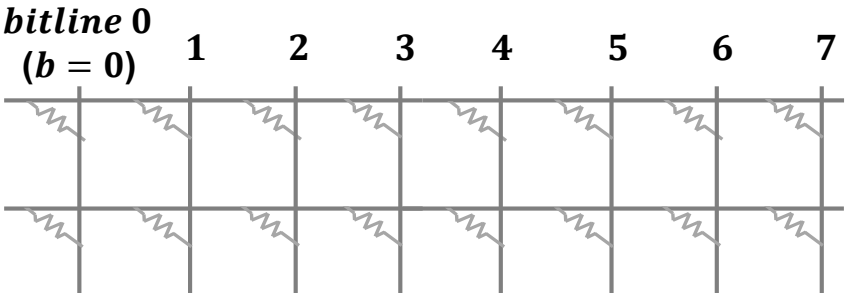
$\boldsymbol{b}$: b^{th} bitline

Value of $i + 2b$

	b = 0	b = 1	b = 2	b = 3	…	b = 7
i = 0	0	2	4	6	…	14
i = 1	1	3	5	7	…	15
i = 2	2	4	6	8	…	16
i = 3	3	5	7	9	…	17
i = 4	4	6	8	10	…	18
i = 5	5	7	9	11	…	19
…	…	…	…	…	…	…
i = 15	15	17	19	21	…	29

Optimal threshold: $i + 2b \leq 14$

$iter0 \ \ (i = 0)$



Skip all bitlines

MAC Rescheduling based on the Dynamic Quantization

- Dynamic quantization: skipping the A/D conversions of those partial products that have no significant impact on the accuracy

$\boldsymbol{i}$: i^{th} iteration

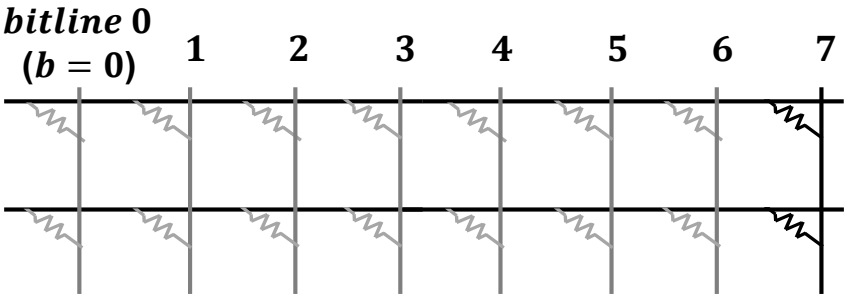
$\boldsymbol{b}$: b^{th} bitline

Value of $i + 2b$

	b = 0	b = 1	b = 2	b = 3	…	b = 7
i = 0	0	2	4	6	…	14
i = 1	1	3	5	7	…	15
i = 2	2	4	6	8	…	16
i = 3	3	5	7	9	…	17
i = 4	4	6	8	10	…	18
i = 5	5	7	9	11	…	19
…	…	…	…	…	…	…
i = 15	15	17	19	21	…	29

Optimal threshold: $i + 2b \leq 14$

$iter1$ & 2 $(i = 1,\ 2)$



Skip all bitlines except 7

MAC Rescheduling based on the Dynamic Quantization

- Dynamic quantization: skipping the A/D conversions of those partial products that have no significant impact on the accuracy

$\boldsymbol{i}$: i^{th} iteration

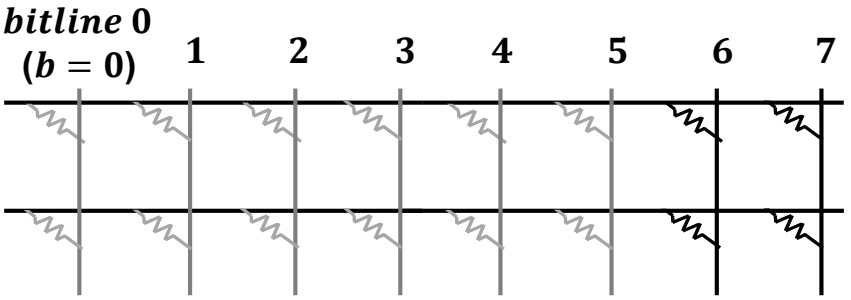
$\boldsymbol{b}$: b^{th} bitline

Value of $i + 2b$

	b = 0	b = 1	b = 2	b = 3	…	b = 7
i = 0	0	2	4	6	…	14
i = 1	1	3	5	7	…	15
i = 2	2	4	6	8	…	16
i = 3	3	5	7	9	…	17
i = 4	4	6	8	10	…	18
i = 5	5	7	9	11	…	19
…	…	…	…	…	…	…
i = 15	15	17	19	21	…	29

Optimal threshold: $i + 2b \leq 14$

$iter3$ & 4 $(i = 3,\ 4)$



Skip all bitlines except 6 & 7

MAC Rescheduling based on the Dynamic Quantization

- Dynamic quantization: skipping the A/D conversions of those partial products that have no significant impact on the accuracy

$\boldsymbol{i}$: i^{th} iteration

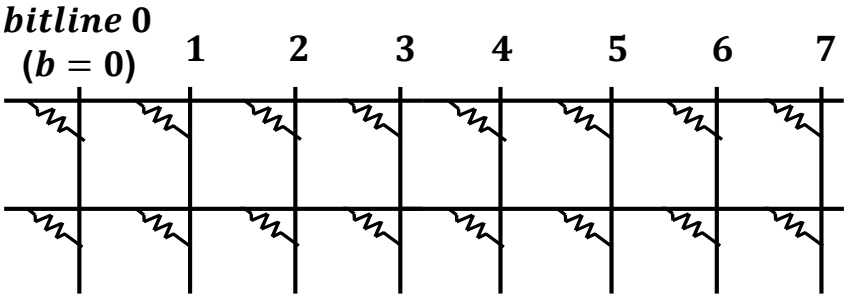
$\boldsymbol{b}$: b^{th} bitline

Value of $i + 2b$

	b = 0	b = 1	b = 2	b = 3	…	b = 7
i = 0	0	2	4	6	…	14
i = 1	1	3	5	7	…	15
i = 2	2	4	6	8	…	16
i = 3	3	5	7	9	…	17
i = 4	4	6	8	10	…	18
i = 5	5	7	9	11	…	19
…	…	…	…	…	…	…
i = 15	15	17	19	21	…	29

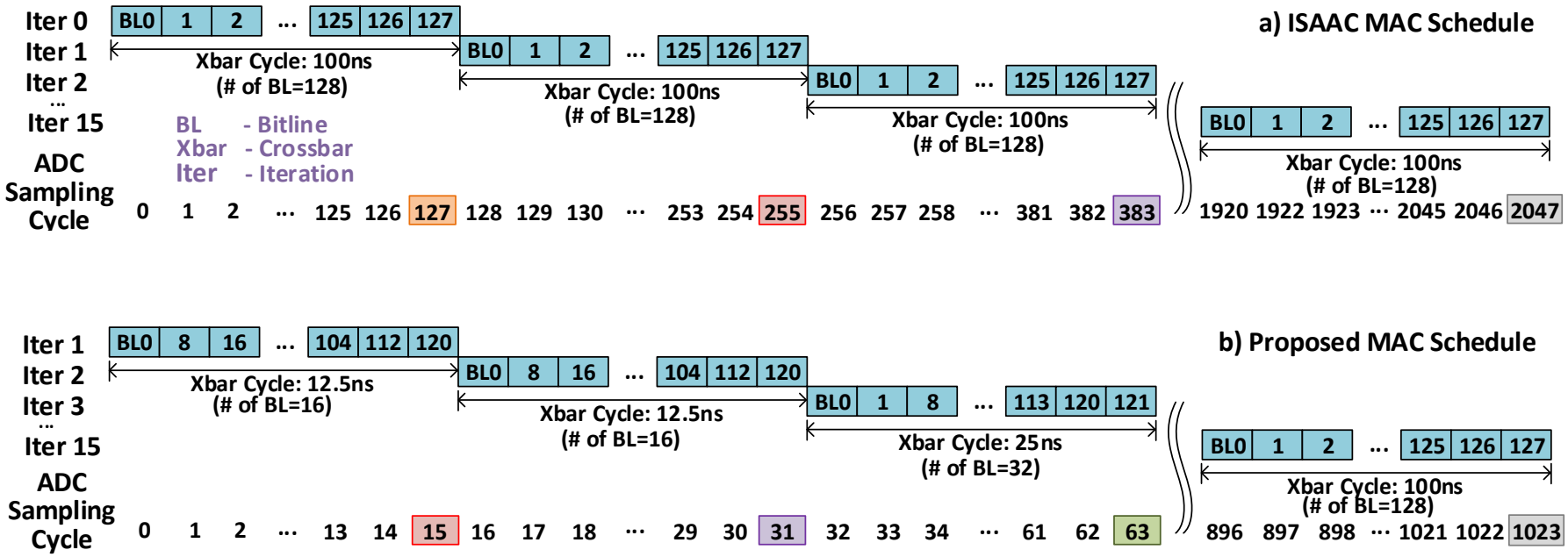
Optimal threshold: $i + 2b \leq 14$

$iter15$ $(i = 15)$



Reserve all the bitlines

MAC Rescheduling based on the Dynamic Quantization



- With a threshold of $i + 2b \leq 14$: 50% of the A/D conversions are skipped.

  # of AD conversions for calculating the outputs on MAC:

  $$16(\#\ of\ iterations) \times 128(\#\ of\ bitlines) \times 50\% = 1024$$

Outline

- Introduction
- Hardware architecture and related works
- Segmented compression encoding
- Optimizing the bit-resolution of the A/D interface
- MAC rescheduling based on the dynamic quantization
- Experimental result
- Conclusion

Experimental Setup

- Benchmarks
  - Caffe model on CIFAR-10[1]
  - LeNet on MNIST[2]
- Throughput evaluation
  - Customized cycle-accurate SystemC simulator
- Power evaluation
  - Memory: CACTI6.5[3]
  - ADC: Kull et al., 2013[4]
  - Peripheral circuits: shift-add, sampling MUX, etc, TSMC 65nm

TABLE I
: Power and Area Estimation

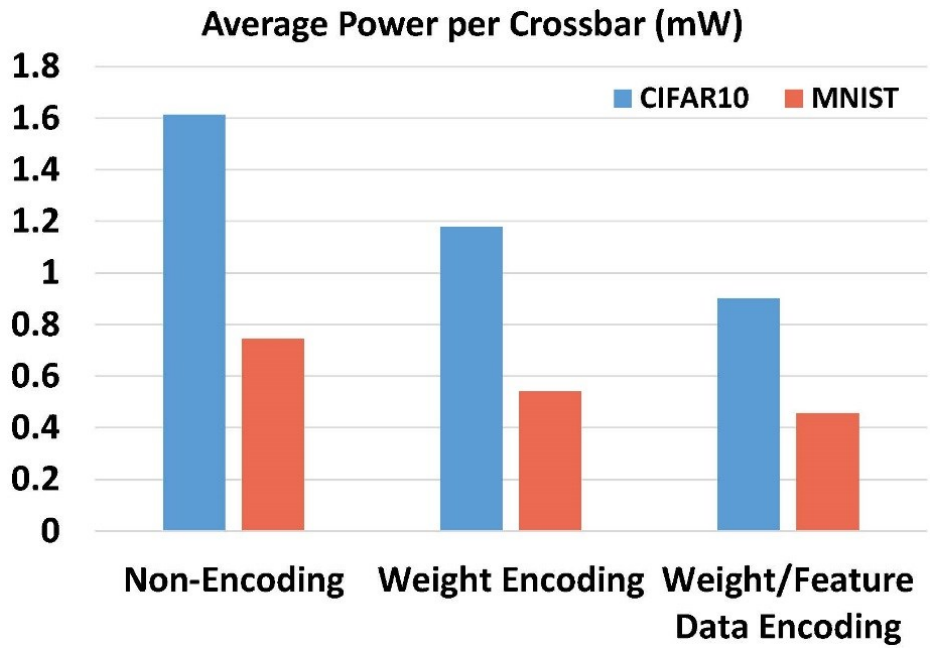
Memory Architecture in One Tile				
Comp	**Params**	**Spec**	**R/W Energy**(nJ)	**Area**(um^2)
eDRAM	size bus width num	32 KB 256 bit 1	0.0271 leakage power: 0.562 mW	38445.2
SRAM Buffer	size bus width num	1KB 128 bit 6	0.0098 leakage power: 1.148 mW	81509.4
One MAC (24 MACs per tile)				
Comp	**Params**	**Spec**	**Power**(mW)	**Area**(um^2)
ADC	resolution num: 1 f_s:1.28G	10 bit 5 bit	4.43 1.84 (58.5% reduced)	4042.5 577.5 (85.7% reduced)
SA+ Output Reg	num	1	1.35	3094.8
DAC	num resolution	256 1 bit	0.002	668.2
CSD	num	128	0.001	323.8
SH	num	128	0.001	5.0
Sampling MUX	num	1	0.07	529.6

[1] Y. Jia, *et al*. Caffe: Convolutional architecture for fast feature embedding. *ACM,* 675–678, 2014.
[2] Y. Lecun, *et al*. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 2278–2324, 1998.
[3] N. Muralimanohar, *et al*. Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0, *MICRO*, 2007.
[4] L. Kull , *et al*. A 3.1mw 8b 1.2gs/s single-channel asynchronous sar adc with alternate comparators for enhanced speed in 32nm digital soi cmos. *ISSCC*, 468-469, 2013.

Result of Segmented Compression Encoding (SCE)

Decreasing of Crossbar Power Consumption after Encoding

	Weight Encoding	Weight + Activation Encoding
CIFAR10	26.9%	44.1%
MNIST	27.4%	38.7%



Result of A/D Interface Optimization

- With 5-bit reduction in resolution

  power of ADC → reduced by 58% area of ADC → reduced by 86%

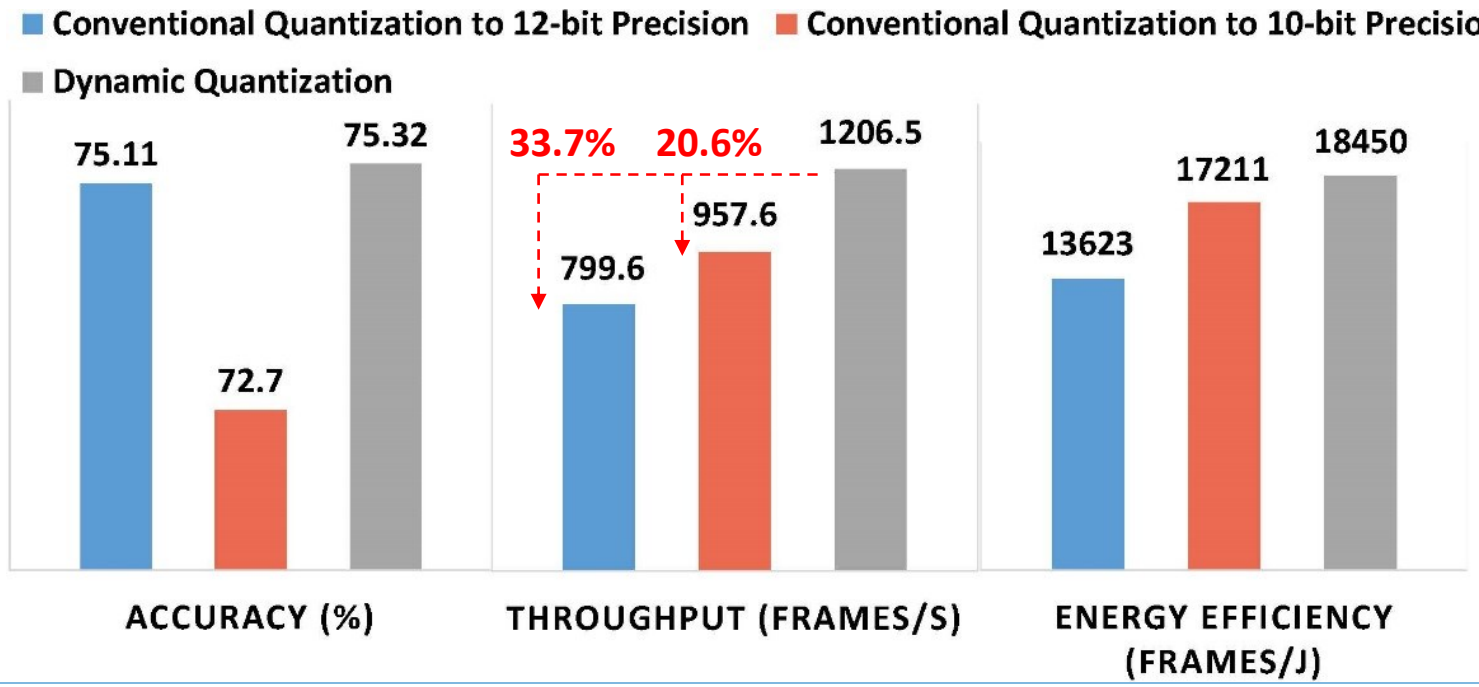
Performance	Conventional Implementation with 10-bit ADC		Modified to 5-bit ADC with Encoding	
	CIFAR-10	MNIST	CIFAR-10	MNIST
Accuracy (%)	75.55	99.09	75.40	99.05
Energy Efficiency $(Frames/J)$	8929	33445	15649	56497
Area Efficiency $(Frames/s/mm^2)$	509	308	677	410

- Overall energy efficiency: 1.7 times larger
- Overall area efficiency: 1.3 times larger
- Accuracy loss: < 0.15%

Result of MAC Rescheduling based on Dynamic Quantization

- The overall improvements after the dynamic quantization
  - Throughput: 2 times larger
  - Energy efficiency: 3.7 times larger
  - Area efficiency: 2.7 times larger
  - Accuracy loss: < 0.32%

- Compared with the 12-bit and 10-bit conventional quantization in CIFAR-10



Outline

- Introduction
- Hardware architecture and related works
- Segmented compression encoding
- Optimizing the bit-resolution of the A/D interface
- MAC rescheduling based on the dynamic quantization
- Experimental result
- Conclusion

Conclusion

- An encoding scheme is proposed for the weights and activations to reduce ~40% energy consumed on the RRAM crossbars and achieve 1-bit reduction in the ADC resolution.
- Based on the distribution analysis of bitline outputs, the bit-resolution of ADC is further reduced by 5 bits, and thus enables 58% & 86% reduction in ADC power and area.
- A dedicated dynamic quantization scheme is proposed for the MAC operation and 50% of the A/D conversions are safely skipped.
- Compared with the state-of-the-art RRAM-based accelerator
  - 2x in throughput
  - 3.7x in energy efficiency
  - 2.7x in area efficiency


Thank you.