Sparsity & Quantization in DNN HW Architectures

Intelligent Architectures: 5LIL0

March 28, 2025

Dr. Federico Corradi, Assistant Professor, Electronic System Group

Department of Electrical Engineering, Electronic Systems Group

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Outline

Recap

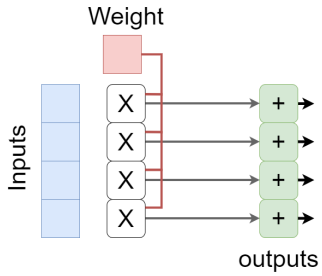
Sparsity
- Diving into state-of-the-art accelerators

Quantization in HW
- Diving into state-of-the-art accelerators

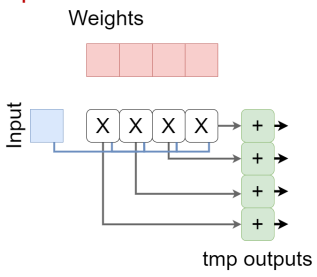
Bonus: IBM Hybrid training / inference in 7nm

Spatial data reuse (spatial unrolling): summary

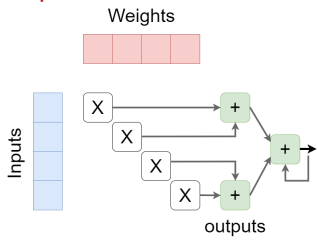
weight reuse



input reuse



output reuse

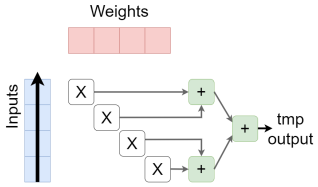


Which one is the best?

- Accelerators use a combination of those, we will see some practical examples
- OS is a good choice, because the outputs has typically more bits than the inputs!

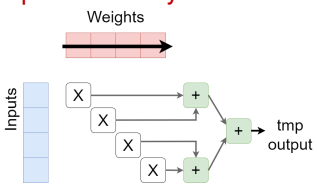
Impact on temporal data reuse T (output reuse baseline)

weight stationary



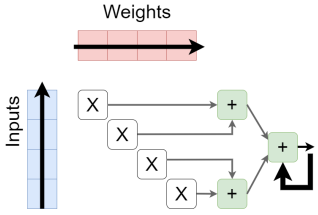
- weight BW = S/T
- input BW = S
- output BW = 1
- $A_I(SRAM)$ = S / (S/T+S+1)

input stationary



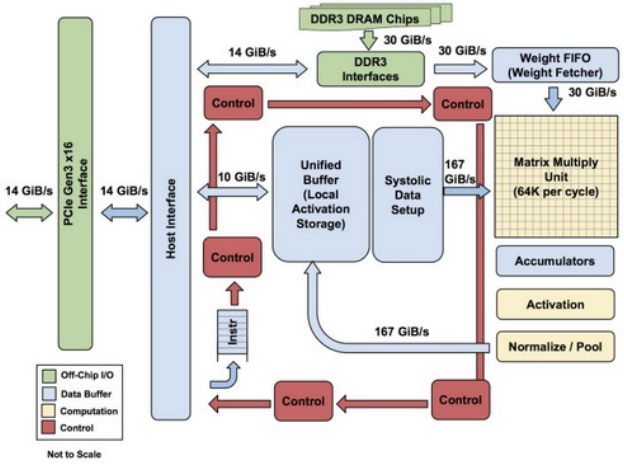
- weight BW = S
- input BW = S/T
- output BW = 1
- $A_I(SRAM)$ = S / (S+S/T+1)

output stationary



- weight BW = S
- input BW = S
- output BW = 1/T
- $A_I(SRAM)$ = S / (2S+1/T)

Google TPU architecture



[Jou, Micro 2018]

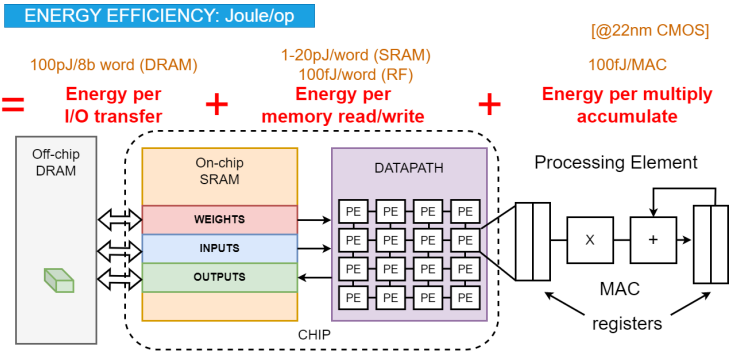
Executing CNN

```
for(b=0 to B-1); //for each image in the batch
 for(k=0 to K-1); //for each output channel
  for(y=0 to Y-1); //for each in/out row
   for(x=0 to X-1); //for each in/out column
    for(c=0 to C-1); //for each input channel
     for(fx=0 to FX-1); //for each kernel row
      for(fy=0 to FY-1); //for each kernel column
       o[b][k][x][y] += i[b][c][x+fx][y+fy]*w[k][c][fx][fy]
```

What?

- Whatever memory optimization, or scheduling we do, we always need to execute N_{MAC}! [$N_{MAC} = K * X * Y * C * FX * FY * B$]

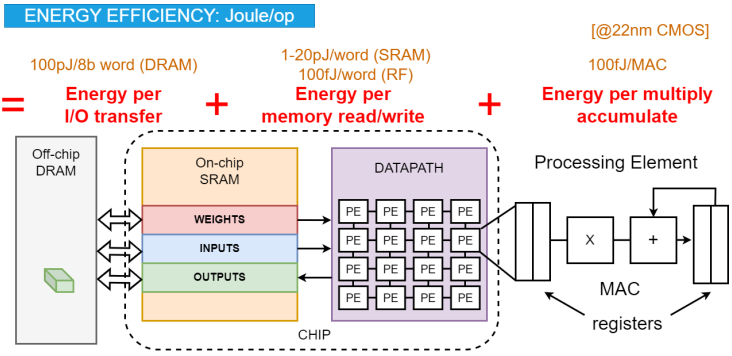
Energy Efficient DNN processors, where are we?



What have we seen so far?

- how to minimize memory transfer (fetching time and energy) ?

Energy Efficient DNN processors, where are we?



What have we seen so far?

- how to minimize memory transfer (fetching time and energy) ?
- how to minimize the number of operations ?

Outline

Recap

Sparsity
- Diving into state-of-the-art accelerators

Quantization in HW
- Diving into state-of-the-art accelerators

Bonus: IBM Hybrid training / inference in 7nm

Lesson Plan

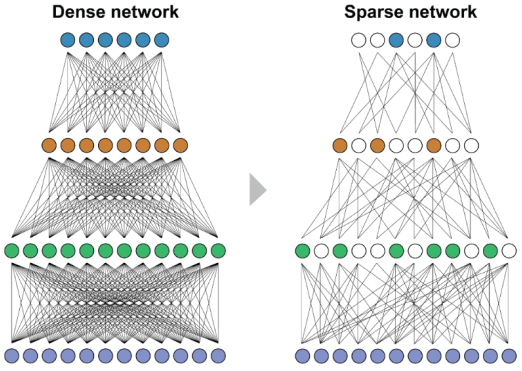
GeMM & CNN processors enhancements in ASIC, CPU, GPU

- Sparsity
  - Sparse CNN engine
  - Efficient Inference Engine
  - NVIDIA Ampere GPU

Sparsity

Sparsity

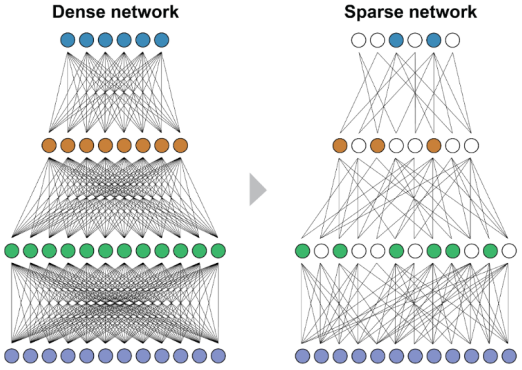
- Sparsity, i.e., '0' values in weights and layers inputs (ReLu!)
- Deep learning nets have large (yet variable) sparsity



Sparsity

Sparsity

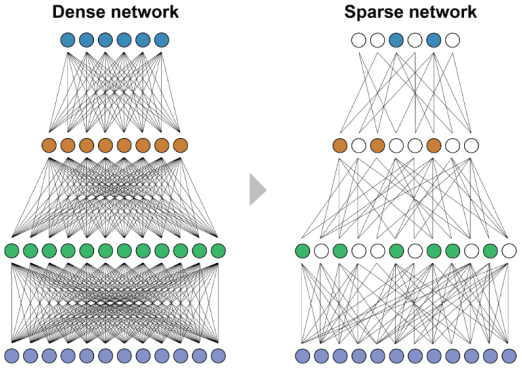
- Sparsity, i.e., '0' values in weights and layers inputs (ReLu!)
- Deep learning nets have large (yet variable) sparsity
- Additional sparsity if often enforced:



Sparsity

Sparsity

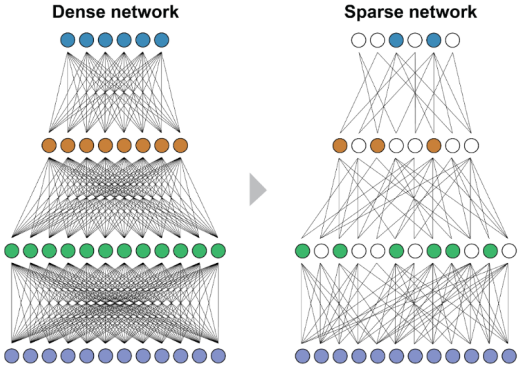
- Sparsity, i.e., '0' values in weights and layers inputs (ReLu!)
- Deep learning nets have large (yet variable) sparsity
- Additional sparsity if often enforced:
  - Quantization



Sparsity

Sparsity

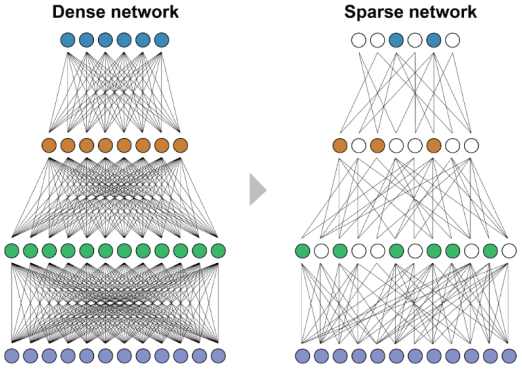
- Sparsity, i.e., '0' values in weights and layers inputs (ReLu!)
- Deep learning nets have large (yet variable) sparsity
- Additional sparsity if often enforced:
  - Quantization
  - Network pruning or compression



Sparsity

Sparsity

- Sparsity, i.e., '0' values in weights and layers inputs (ReLu!)
- Deep learning nets have large (yet variable) sparsity
- Additional sparsity if often enforced:
  - Quantization
  - Network pruning or compression
  - Regularization



Algorithm (DNN Model) & Hardware Co-Design

Approaches

- For hardware accelerators, we **exploit sparsity** for reducing:

Algorithm (DNN Model) & Hardware Co-Design

Approaches

- For hardware accelerators, we **exploit sparsity** for reducing:
  - I/O traffic optimization E_{mem}

Algorithm (DNN Model) & Hardware Co-Design

Approaches

- For hardware accelerators, we **exploit sparsity** for reducing:
  - I/O traffic optimization E_{mem}
  - Sparse processing optimization E_{mac}, E_{RF}

Sparse data streams are compressible

Exploit sparsity for compression

- Data streams of I,O and W have many zeros, $\rightarrow$ let's compress!

Sparse data streams are compressible

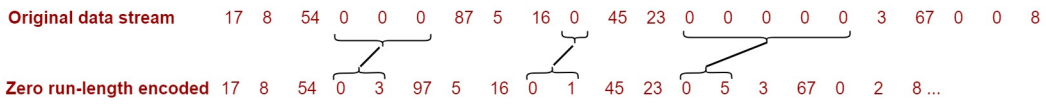
Exploit sparsity for compression

- Data streams of I,O and W have many zeros, $\rightarrow$ let's compress!
- Data streams can be stored / communicated in compressed format, e.g.,

Sparse data streams are compressible

Exploit sparsity for compression

- Data streams of I,O and W have many zeros, $\rightarrow$ let's compress!
- Data streams can be stored / communicated in compressed format, e.g.,
  1. Linear compression on sparse features
     - 1.1 Zero-runlength encoding (e.g., 3x VGG16) [Rutishauser et al 2020]



Sparse data streams are compressible

Exploit sparsity for compression

- Data streams of I,O and W have many zeros, $\rightarrow$ let's compress!
- Data streams can be stored / communicated in compressed format, e.g.,
  1. Linear compression on sparse features
     - 1.1 Zero-runlength encoding (e.g., 3x VGG16) [Rutishauser et al 2020]
     - 1.2 Huffman-encoding (e.g., 5x VGG16) [Moons et al, 2017]

Huffman Encoding

We want to minimize:

- small bit-length for most frequent elements
- easy decoding

[Huffman, 1952] [Youtube Tutorial, 4Min!]

Original data stream

$A\ A\ B\ A\ C\ C\ D$ **7x8bits = 56bits**

Huffman Encoding

We want to minimize:

- small bit-length for most frequent elements
- easy decoding

[Huffman, 1952] [Youtube Tutorial, 4Min!]

Original data stream

$A\ A\ B\ A\ C\ C\ D$ **7x8bits = 56bits**

A	3
B	1
C	2
D	1

$A_3 \quad C_2 \quad B_1 \quad D_1$

Huffman Encoding

We want to minimize:

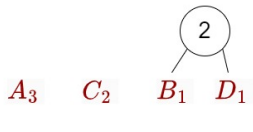
- small bit-length for most frequent elements
- easy decoding

[Huffman, 1952] [Youtube Tutorial, 4Min!]

Original data stream

$A\ A\ B\ A\ C\ C\ D$ **7x8bits = 56bits**

A	3
B	1
C	2
D	1



Huffman Encoding

We want to minimize:

- small bit-length for most frequent elements
- easy decoding

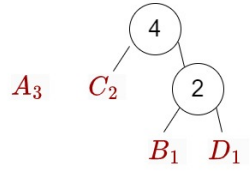
[Huffman, 1952] [Youtube Tutorial, 4Min!]

Original data stream

$A\ A\ B\ A\ C\ C\ D$

7x8bits = 56bits

A	3
B	1
C	2
D	1



Huffman Encoding

We want to minimize:

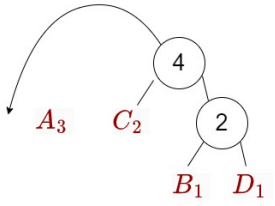
- small bit-length for most frequent elements
- easy decoding

[Huffman, 1952] [Youtube Tutorial, 4Min!]

Original data stream

$A\ A\ B\ A\ C\ C\ D$ **7x8bits = 56bits**

A	3
B	1
C	2
D	1



Huffman Encoding

Original data stream

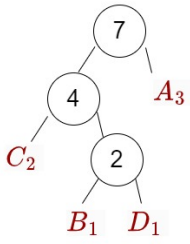
$A\ A\ B\ A\ C\ C\ D$ **7x8bits = 56bits**

We want to minimize:

- small bit-length for most frequent elements
- easy decoding

[Huffman, 1952] [Youtube Tutorial, 4Min!]

A	3
B	1
C	2
D	1



Huffman Encoding

Original data stream

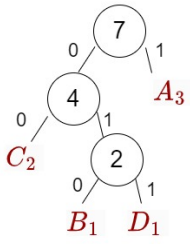
$A\ A\ B\ A\ C\ C\ D$ **7x8bits = 56bits**

We want to minimize:

- small bit-length for most frequent elements
- easy decoding

[Huffman, 1952] [Youtube Tutorial, 4Min!]

A	3
B	1
C	2
D	1



Huffman Encoding

We want to minimize:

- small bit-length for most frequent elements
- easy decoding

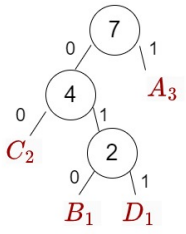
[Huffman, 1952] [Youtube Tutorial, 4Min!]

Original data stream

$A\ A\ B\ A\ C\ C\ D$

7x8bits = 56bits
7x2bits = 14bits

A	3	1
B	1	010
C	2	00
D	1	011

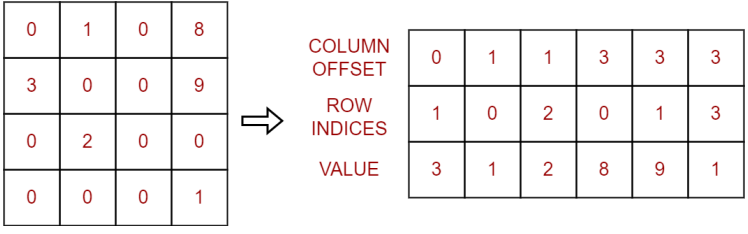


Huffman encoded **1101010000011** **13 bits!**

Sparse data streams are compressible

Exploit sparsity for compression

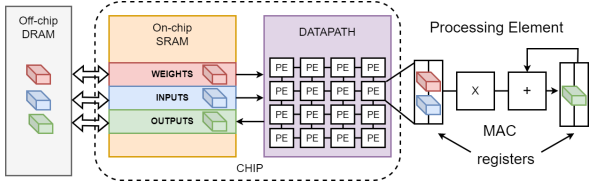
- Data streams of I,O and W have many zeros, $\rightarrow$ let's compress!
- Data streams can be stored / communicated in compressed format, e.g.,
  1. Linear compression on sparse features
     - 1.1 Zero-runlength encoding (e.g., 3x VGG16) [Rutishauser et al 2020]
     - 1.2 Huffman-encoding (e.g., 5x VGG16) [Moons et al, 2017]
     - 1.3 And much more complex schemes, e.g. compressed sparse column format (CSC), can be used for I/O communication (e.g., 10x lower rate) or for on-chip format.



Where to do compression and decompression?

I and W data can be stored compressed or uncompressed on chip

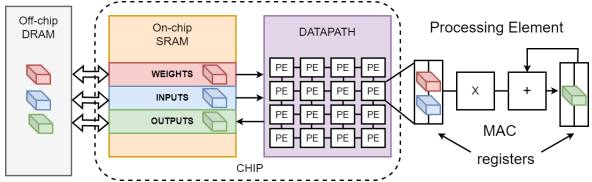
- Either decompressed in DRAM-to-chip interface



Where to do compression and decompression?

I and W data can be stored compressed or uncompressed on chip

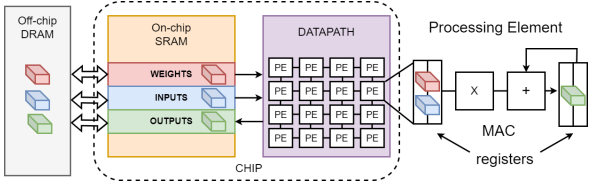
- Either decompressed in DRAM-to-chip interface
  - benefits: simple, datapath regularity maintained (simple scheduling)



Where to do compression and decompression?

I and W data can be stored compressed or uncompressed on chip

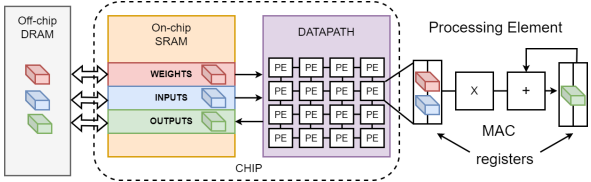
- Either decompressed in DRAM-to-chip interface
  - benefits: simple, datapath regularity maintained (simple scheduling)
  - downsides: does not improve on-chip memory usage



Where to do compression and decompression?

I and W data can be stored compressed or uncompressed on chip

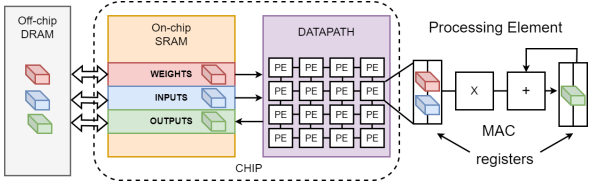
- Either decompressed in DRAM-to-chip interface
  - benefits: simple, datapath regularity maintained (simple scheduling)
  - downsides: does not improve on-chip memory usage
- Or in SRAM-to-RF interface



Where to do compression and decompression?

I and W data can be stored compressed or uncompressed on chip

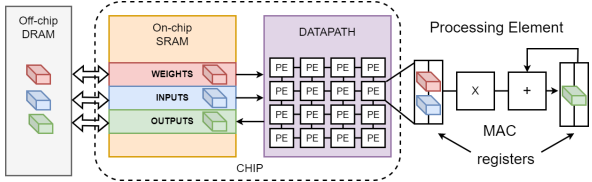
- Either decompressed in DRAM-to-chip interface
  - benefits: simple, datapath regularity maintained (simple scheduling)
  - downsides: does not improve on-chip memory usage
- Or in SRAM-to-RF interface
  - benefits: on-chicp memory reduction $\rightarrow$, now larger part of weights and activations data fits on chip (SRAM data reuse)



Where to do compression and decompression?

I and W data can be stored compressed or uncompressed on chip

- Either decompressed in DRAM-to-chip interface
  - benefits: simple, datapath regularity maintained (simple scheduling)
  - downsides: does not improve on-chip memory usage
- Or in SRAM-to-RF interface
  - benefits: on-chicp memory reduction $\rightarrow$, now larger part of weights and activations data fits on chip (SRAM data reuse)
  - downsides: we need to do decompression while keeping the datapath active, i.e, it is complex keep parallelization high



Algorithm (DNN Model) & Hardware Co-Design

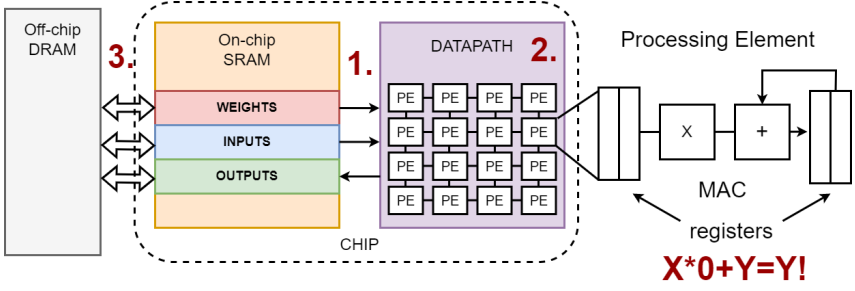
Approaches

- For hardware accelerators, we **exploit sparsity** for reducing:
  - I/O traffic optimization E_{mem}
  - Sparse processing optimization E_{mac}, E_{RF}

Algorithm (DNN Model) & Hardware Co-Design

Exploiting sparsity in DNN accelerators

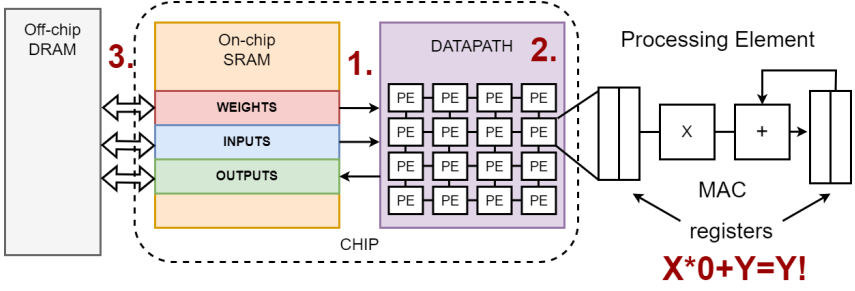
- Do not waste energy in '0'-valued data
  1. Do not fetch zeros



Algorithm (DNN Model) & Hardware Co-Design

Exploiting sparsity in DNN accelerators

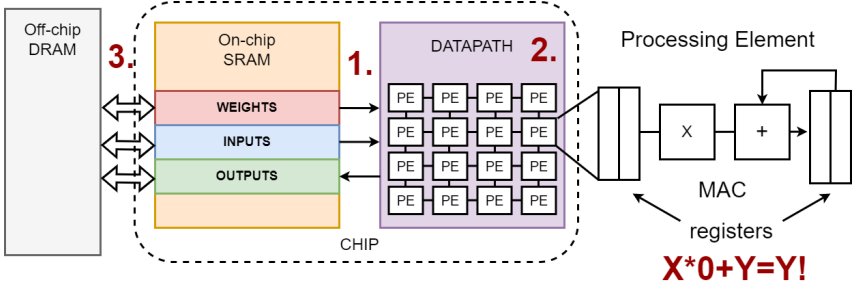
- Do not waste energy in '0'-valued data
  1. Do not fetch zeros
  2. Guard MAC operations with zero (zeros skipping)



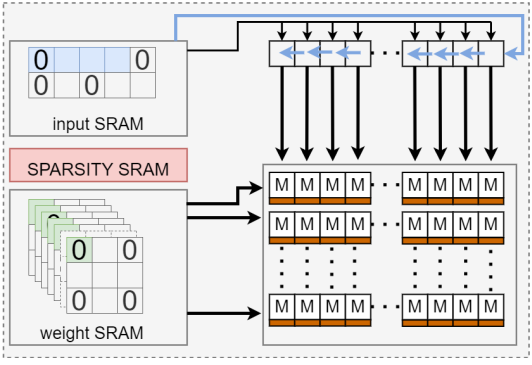
Algorithm (DNN Model) & Hardware Co-Design

Exploiting sparsity in DNN accelerators

- Do not waste energy in '0'-valued data
  1. Do not fetch zeros
  2. Guard MAC operations with zero (zeros skipping)
  3. Compress off-chip data stream



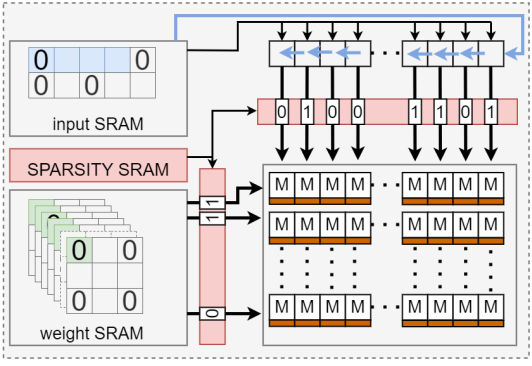
Sparsity in Envision (KU) Leuven



[Moons et al 2017]

SRAM stores sparsity flags

Sparsity in Envision (KU) Leuven

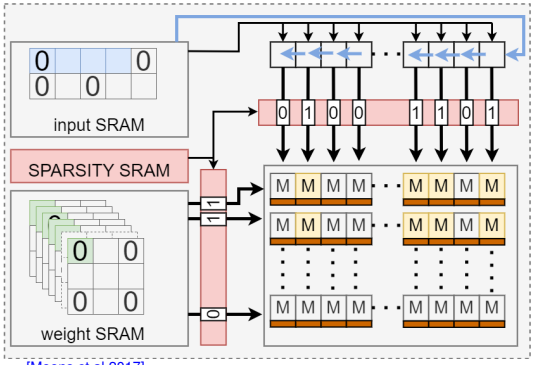


[Moons et al 2017]

SRAM stores sparsity flags

- sparsity flag guard mem read, save mem transfer

Sparsity in Envision (KU) Leuven

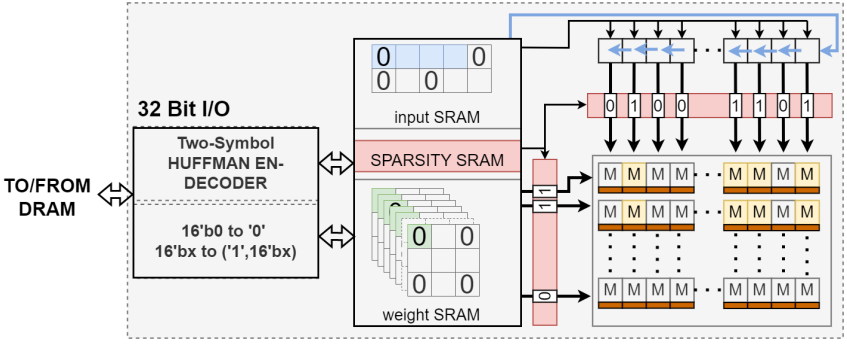


[Moons et al 2017]

SRAM stores sparsity flags

- sparsity flag guard mem read, save mem transfer
- use same flag to guard MAC execution (row/column), save compute

Sparsity in Envision (KU) Leuven



SRAM stores sparsity flags

- sparsity flag guard mem read, save mem transfer
- use same flag to guard MAC execution (row/column), save compute
- Huffman compression on DRAM I/O save mem BW [Moons et al 2017]

Sparsity gain in Envision (KU) Leuven

Measured

- Off-chip BW scales with sparsity

Sparsity gain in Envision (KU) Leuven

Measured

- Off-chip BW scales with sparsity
- Less compute energy

Sparsity gain in Envision (KU) Leuven

Measured

- Off-chip BW scales with sparsity
- Less compute energy
- Disadvantages? Why this is not very useful?

Sparsity gain in Envision (KU) Leuven

Measured

- Off-chip BW scales with sparsity
- Less compute energy
- Disadvantages? Why this is not very useful?
  - Because we have **unused MAC units**! no throughput benefits!

Sparsity gain in Envision (KU) Leuven

Measured

- Off-chip BW scales with sparsity
- Less compute energy
- Disadvantages? Why this is not very useful?
  - Because we have **unused MAC units**! no throughput benefits!
  - This is also true in GPUs! Sparse BLAS matrix (GeMM) library needs huge sparsity to be beneficial.[Wei et al, 2016]

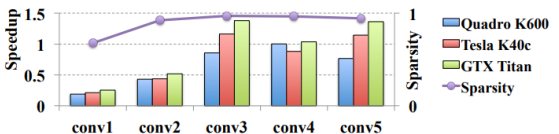


Figure 1: Evaluation speedups of AlexNet on GPU platforms and the sparsity. conv1 refers to convolutional layer 1, and so forth. Baseline is profiled by GEMM of cuBLAS. The sparse matrixes are stored in the format of Compressed Sparse Row (CSR) and accelerated by cuSPARSE.

Sparsity gain in Envision (KU) Leuven

Measured

- Off-chip BW scales with sparsity
- Less compute energy
- Disadvantages? Why this is not very useful?
  - Because we have **unused MAC units**! no throughput benefits!
  - This is also true in GPUs! Sparse BLAS matrix (GeMM) library needs huge sparsity to be beneficial.[Wei et al, 2016]
  - Custom hardware to rescue?

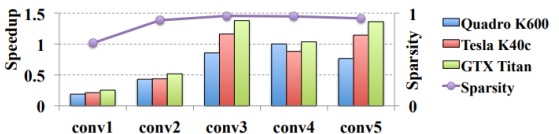
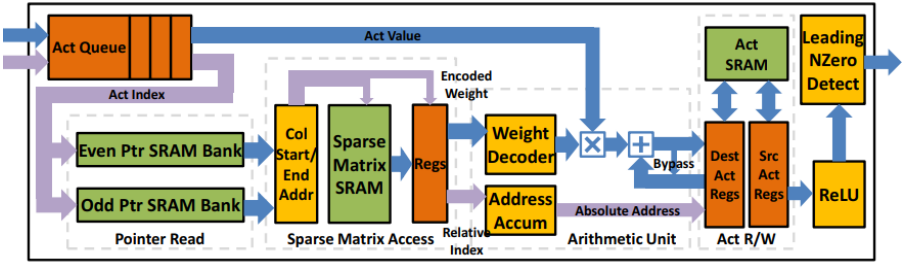


Figure 1: Evaluation speedups of AlexNet on GPU platforms and the sparsity. conv1 refers to convolutional layer 1, and so forth. Baseline is profiled by GEMM of cuBLAS. The sparse matrixes are stored in the format of Compressed Sparse Row (CSR) and accelerated by cuSPARSE.

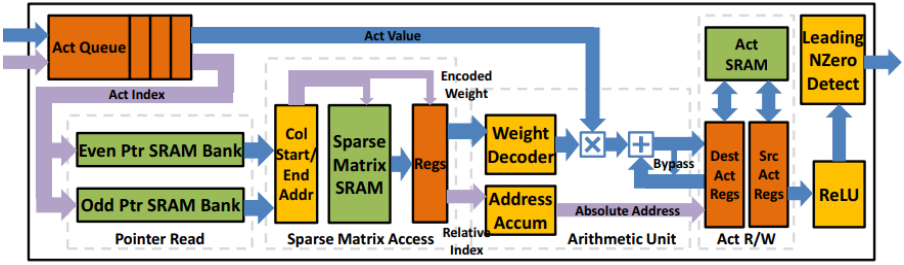
Sparse processor: Efficient Inference Engine (EIE) Stanford



Efficient Inference Engine [Han et al, 2016]

- Stores weights in compressed format, **no zeros!** (compressed sparse column fmt.)

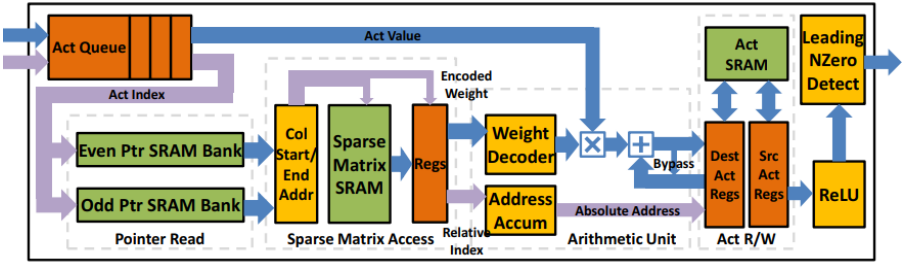
Sparse processor: Efficient Inference Engine (EIE) Stanford



Efficient Inference Engine [Han et al, 2016]

- Stores weights in compressed format, **no zeros!** (compressed sparse column fmt.)
- Dynamically scheduled compute array

Sparse processor: Efficient Inference Engine (EIE) Stanford



Efficient Inference Engine [Han et al, 2016]

- Stores weights in compressed format, **no zeros!** (compressed sparse column fmt.)
- Dynamically scheduled compute array
- Sends only non-zero inputs and non-zero weights to MACs

Sparse processor: Efficient Inference Engine (EIE) Stanford

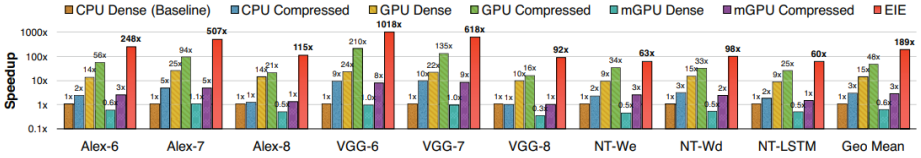
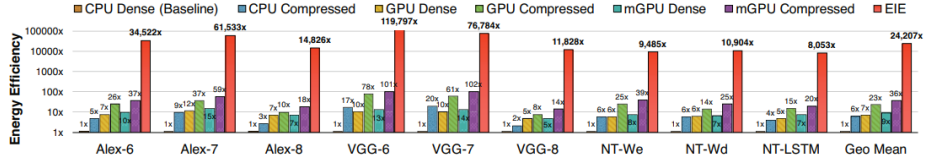


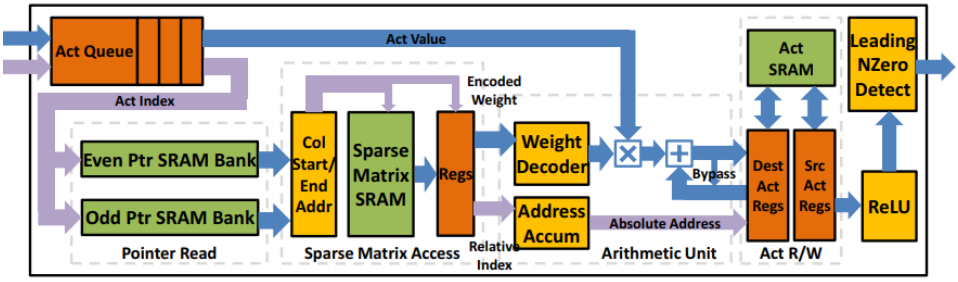
Figure 6. Speedups of GPU, mobile GPU and EIE compared with CPU running uncompressed DNN model. There is no batching in all cases.



Efficient Inference Engine: [Han et al, 2016]

- Benefits: speedup and energy benefits from sparsity

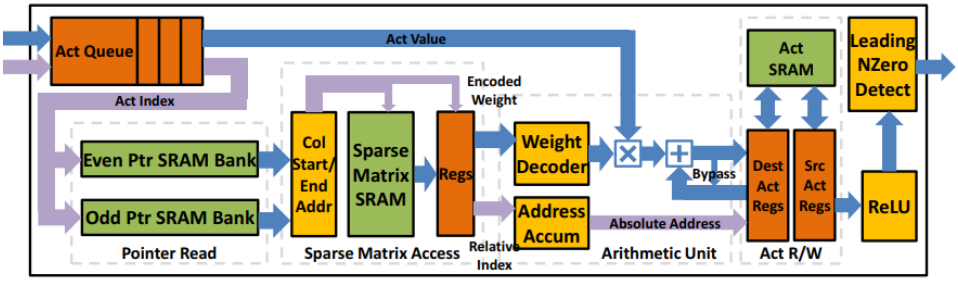
Sparse processor: Efficient Inference Engine (EIE) Stanford



Efficient Inference Engine: [Han et al, 2016]

- Downsides:

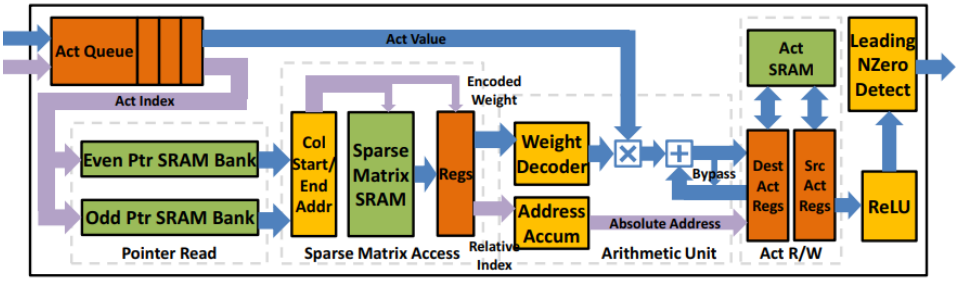
Sparse processor: Efficient Inference Engine (EIE) Stanford



Efficient Inference Engine: [Han et al, 2016]

- Downsides:
  - **complex address generation and loop control**. It needs complex index handling to know which weight must be multiplied with which inputs and where to store the results and what multiplication to do!

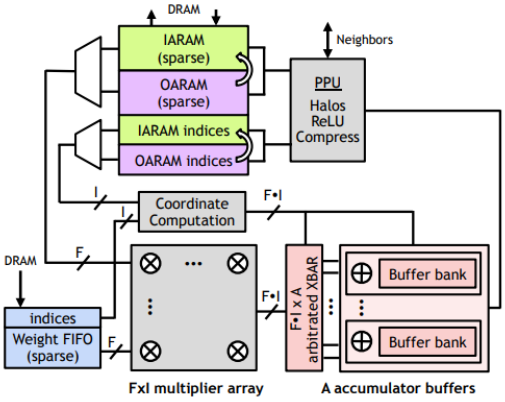
Sparse processor: Efficient Inference Engine (EIE) Stanford



Efficient Inference Engine: [Han et al, 2016]

- Downsides:
  - **complex address generation and loop control**. It needs complex index handling to know which weight must be multiplied with which inputs and where to store the results and what multiplication to do!
  - it was only for weight compression

Sparse processor: Sparse CNN engine (SCNN)

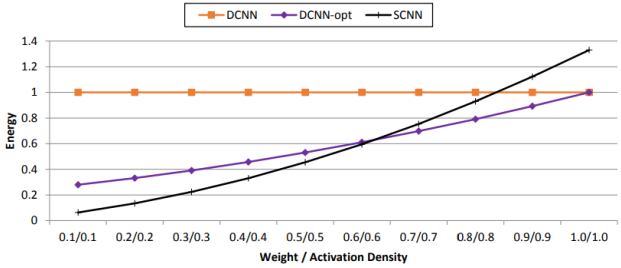


[Parashar et al, 2017]

SCNN

- stores both inputs and weights compressed
- F = num. of non-zeros weights
- I = num. of non-zeros inputs

Sparse processor: Sparse CNN engine (SCNN)



Analytical computation of energy benefits in terms of sparsity [Parashar et al, 2017]

- DCNN: is a dense CNN processor (does not exploit sparsity)
- DCNN-opt a processor like Envision (MAC clock-gating & DRAM compression) no throughput benefits (energy savings are due to clock-gating)
- SCNN sparse processor $\sim$2X gains if sufficient sparsity, plus area saving!

Sparse processor: Sparse CNN engine (SCNN)

SCNN: Is it worth the effort?

- MobileNet, EfficientNet, and other more recent networks (e.g. transformers) are not that sparse!

Sparse processor: Sparse CNN engine (SCNN)

SCNN: Is it worth the effort?

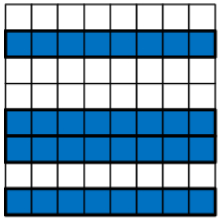
- MobileNet, EfficientNet, and other more recent networks (e.g. transformers) are not that sparse!
- It depends on the workload and application.

Sparse processor: Sparse CNN engine (SCNN)

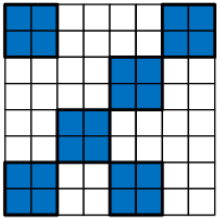
SCNN: Is it worth the effort?

- MobileNet, EfficientNet, and other more recent networks (e.g. transformers) are not that sparse!
- It depends on the workload and application.
- There is also an intermediate scheme... structured sparsity.

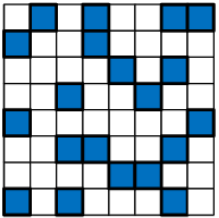
Structured sparsity



Structured (R^8)



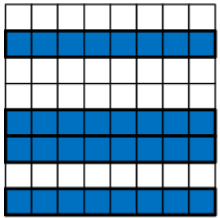
Block-based (R^{4x4})

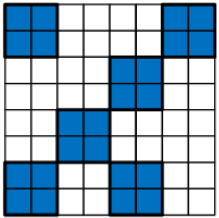


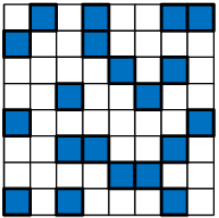
Unstructured (R^{8x8})

structured sparsity

Structured sparsity


Structured (R^8)

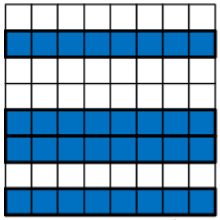

Block-based (R^{4x4})

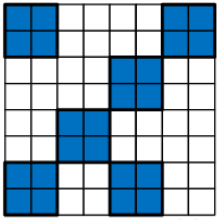

Unstructured (R^{8x8})

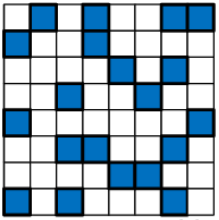
structured sparsity

- we only use one bit to indicate if a complete row, or block is zero

Structured sparsity


Structured (R^{8})

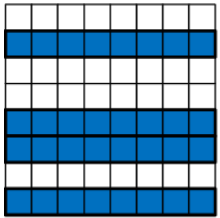

Block-based (R^{4x4})

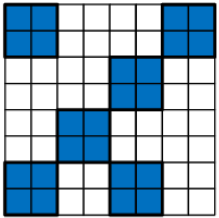

Unstructured (R^{8x8})

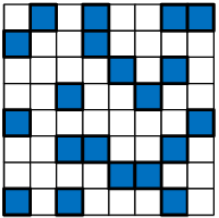
structured sparsity

- we only use one bit to indicate if a complete row, or block is zero
- we still keep some parallelization degrees (within one block we can still spatially unroll)

Structured sparsity


Structured (R^8)


Block-based (R^{4x4})


Unstructured (R^{8x8})

structured sparsity

- we only use one bit to indicate if a complete row, or block is zero
- we still keep some parallelization degrees (within one block we can still spatially unroll)
- but... we need to train DNN to enforce structural sparsity

How to enforce structural sparsity?

$$L(W) = L_D(W) \tag{1}$$

- $L_D(W)$ normal loss function

How to enforce structural sparsity?

$$L(W) = L_D(W) + \lambda \cdot R(W) \tag{1}$$

- $L_D(W)$ normal loss function
- $\lambda \cdot R(W)$ enforce sparse weights (L1 or L2 norm for all weights)

How to enforce structural sparsity?

$$L(W) = L_D(W) + \lambda \cdot R(W) + \lambda_g \cdot \sum_{l=1}^{L} R_g(W^{(l)}) \tag{1}$$

- $L_D(W)$ normal loss function
- $\lambda \cdot R(W)$ enforce sparse weights (L1 or L2 norm for all weights)
- $\lambda_g \sum_{l=1}^{L} R_g(W^{(l)})$ enforce sparse weight "groups" (L1 or L2 norm of weights in a group)

[Wen et al, 2016]

Structural sparsity on GPUs

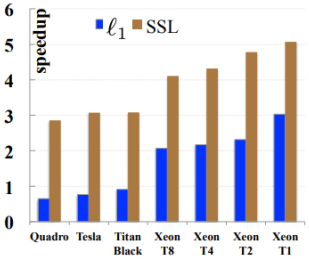


Table 4: Sparsity and speedup of *AlexNet* on ILSVRC 2012

#	Method	Top1 err.	Statistics	conv1	conv2	conv3	conv4	conv5
1	ℓ_1	44.67%	sparsity	67.6%	92.4%	97.2%	96.6%	94.3%
			CPU ×	0.80	2.91	4.84	3.83	2.76
			GPU ×	0.25	0.52	1.38	1.04	1.36
2	SSL	44.66%	column sparsity	0.0%	63.2%	76.9%	84.7%	80.7%
			row sparsity	9.4%	12.9%	40.6%	46.9%	0.0%
			CPU ×	1.05	3.37	6.27	9.73	4.93
			GPU ×	1.00	2.37	4.94	4.03	3.05

Results based on AlexNet in CPU and GPU

Structural sparsity on GPUs

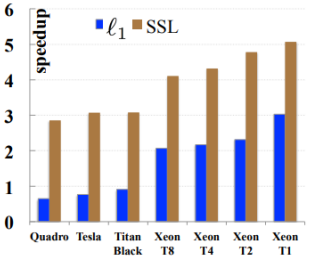


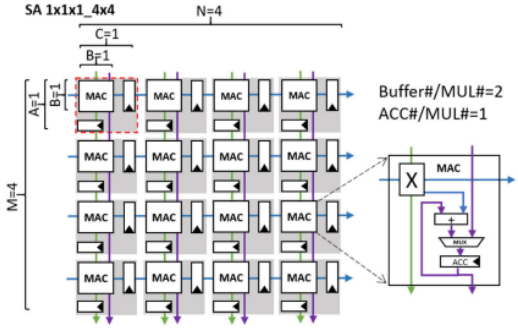
Table 4: Sparsity and speedup of *AlexNet* on ILSVRC 2012

#	Method	Top1 err.	Statistics	conv1	conv2	conv3	conv4	conv5
1	ℓ_1	44.67%	sparsity	67.6%	92.4%	97.2%	96.6%	94.3%
			CPU ×	0.80	2.91	4.84	3.83	2.76
			GPU ×	0.25	0.52	1.38	1.04	1.36
2	SSL	44.66%	column sparsity	0.0%	63.2%	76.9%	84.7%	80.7%
			row sparsity	9.4%	12.9%	40.6%	46.9%	0.0%
			CPU ×	1.05	3.37	6.27	9.73	4.93
			GPU ×	1.00	2.37	4.94	4.03	3.05

Results based on AlexNet in CPU and GPU

- Can we do even better with custom processors, optimized for structured sparsity?
  [Wen et al, 2016]

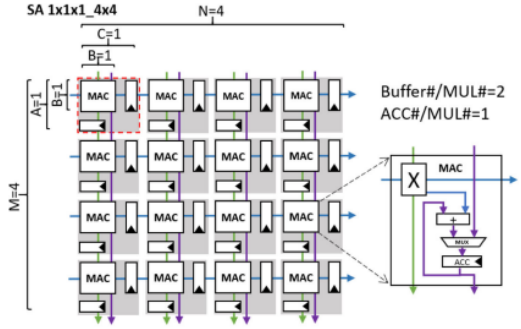
Processor supporting structured sparsity



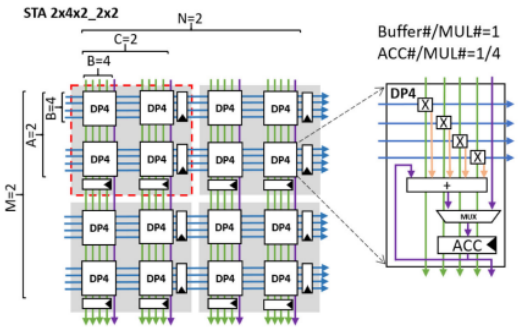
(a) Conventional Systolic Array (**SA**)

- Traditional systolic arrays of MACs

Processor supporting structured sparsity



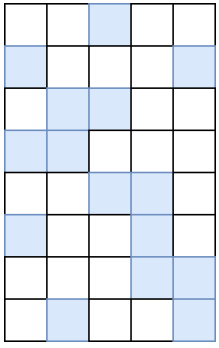
(a) Conventional Systolic Array (**SA**)

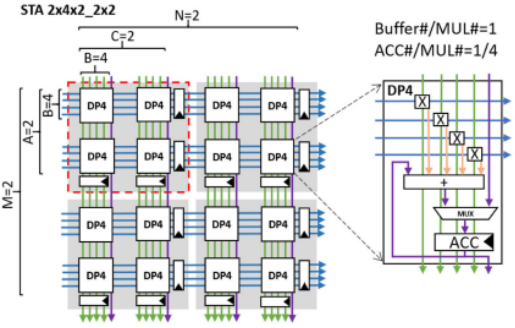


(b) Systolic Tensor Array (**STA**)

- Traditional systolic arrays of MACs
- Extension of systolic arrays with a multidimensional arrays of MACs (Systolic Tensor Array). They have multiple multipliers in a single DP4 units, and you can introduce granular structured sparsity.

Processor supporting structured sparsity: ARM



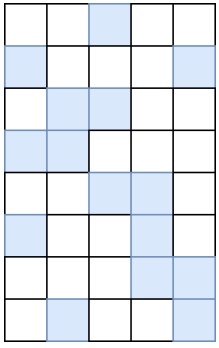


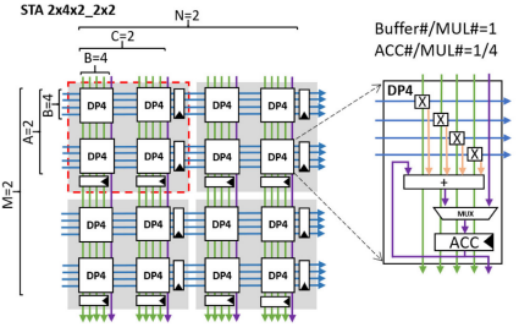
(b) Systolic Tensor Array (**STA**)

- max X/Y elements non-zeros (no need of fully structured sparsity)

[Liu et at, 2020]

Processor supporting structured sparsity: ARM



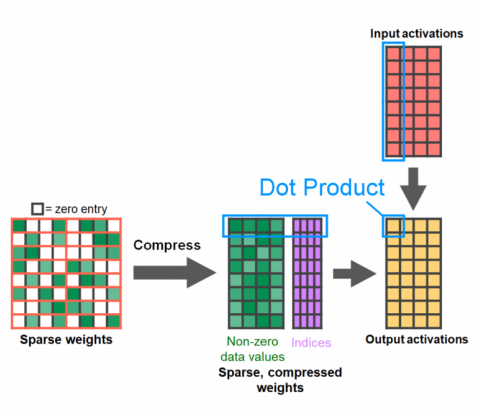


(b) Systolic Tensor Array (**STA**)

- max X/Y elements non-zeros (no need of fully structured sparsity)
- For example, every column is only allowed to have 3/8 non-zero elements (density block bounded)

[Liu et at, 2020]

Processor supporting structured sparsity: GPU Nvidia Ampere

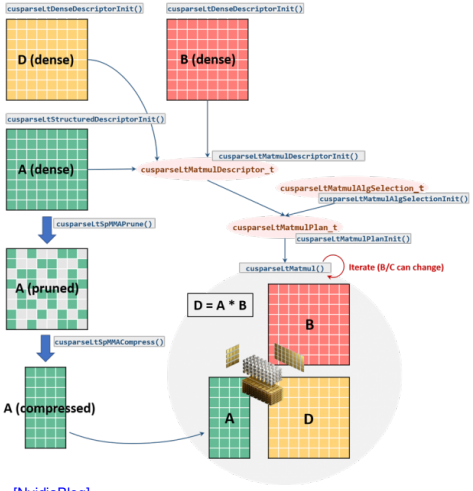


[NvidiaBlog]

CuSPARSELt

- 50% elements non-zeros
- weights only

Processor supporting structured sparsity: GPU Nvidia Ampere

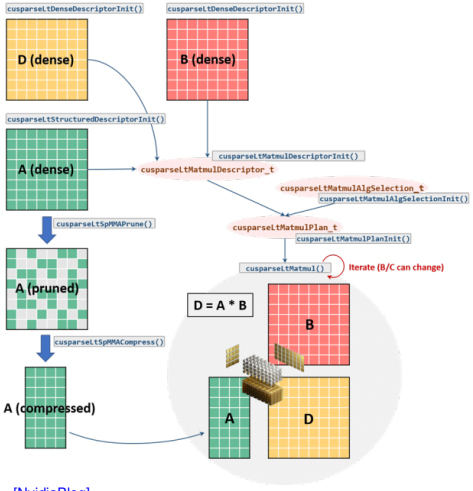


[NvidiaBlog]

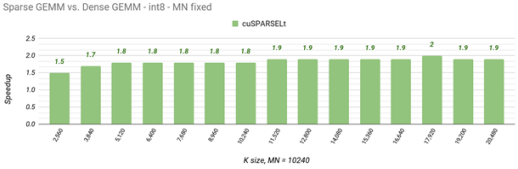
CuSPARSELt

- 50% elements non-zeros
- weights only
- it has a density block bound (1 out of 2), see A (pruned matrix)

Processor supporting structured sparsity: GPU Nvidia Ampere



[NvidiaBlog]



CuSPARSELt

- 50% elements non-zeros
- weights only
- it has a density block bound (1 out of 2), see A (pruned matrix)
- near perfect speedup (if $\sim$50 sparse)

Sparsity conclusions

Sparsity in HW accelerators

- clear need to consider HW and SW at the same time!

Sparsity conclusions

Sparsity in HW accelerators

- clear need to consider HW and SW at the same time!
- decision on sparsity techniques strongly impacted by neural network

Sparsity conclusions

Sparsity in HW accelerators

- clear need to consider HW and SW at the same time!
- decision on sparsity techniques strongly impacted by neural network
- decision on sparsity techniques strongly impacted by hardware architectures

Sparsity conclusions

Sparsity in HW accelerators

- clear need to consider HW and SW at the same time!
- decision on sparsity techniques strongly impacted by neural network
- decision on sparsity techniques strongly impacted by hardware architectures
  - dense = regular, easy to accelerate efficiently

Sparsity conclusions

Sparsity in HW accelerators

- clear need to consider HW and SW at the same time!
- decision on sparsity techniques strongly impacted by neural network
- decision on sparsity techniques strongly impacted by hardware architectures
  - dense = regular, easy to accelerate efficiently
  - sparse = irregular, control overheads

Sparsity conclusions

Sparsity in HW accelerators

- clear need to consider HW and SW at the same time!
- decision on sparsity techniques strongly impacted by neural network
- decision on sparsity techniques strongly impacted by hardware architectures
  - dense = regular, easy to accelerate efficiently
  - sparse = irregular, control overheads
  - structured sparse: sparsity takes HW support into account

Sparsity conclusions

Sparsity in HW accelerators

- clear need to consider HW and SW at the same time!
- decision on sparsity techniques strongly impacted by neural network
- decision on sparsity techniques strongly impacted by hardware architectures
  - dense = regular, easy to accelerate efficiently
  - sparse = irregular, control overheads
  - structured sparse: sparsity takes HW support into account
  - strong impact on the data flow

Sparsity conclusions

Sparsity in HW accelerators

- clear need to consider HW and SW at the same time!
- decision on sparsity techniques strongly impacted by neural network
- decision on sparsity techniques strongly impacted by hardware architectures
  - dense = regular, easy to accelerate efficiently
  - sparse = irregular, control overheads
  - structured sparse: sparsity takes HW support into account
  - strong impact on the data flow
- many studies and gain have been shown, yet in ad-hoc fashion (require fine-tuning, and retraining)

Sparsity conclusions

Sparsity in HW accelerators

- clear need to consider HW and SW at the same time!
- decision on sparsity techniques strongly impacted by neural network
- decision on sparsity techniques strongly impacted by hardware architectures
  - dense = regular, easy to accelerate efficiently
  - sparse = irregular, control overheads
  - structured sparse: sparsity takes HW support into account
  - strong impact on the data flow
- many studies and gain have been shown, yet in ad-hoc fashion (require fine-tuning, and retraining)
- for the weights: structural sparsity seems to be established. For the inputs: not yet.

Lesson Plan

GeMM & CNN processors enhancements in ASIC, CPU, GPU

- Quantization in hardware
  - Envision (KU) Leuven
  - Bonus: IBM Hybrid training/inference in 7nm

Lesson Plan

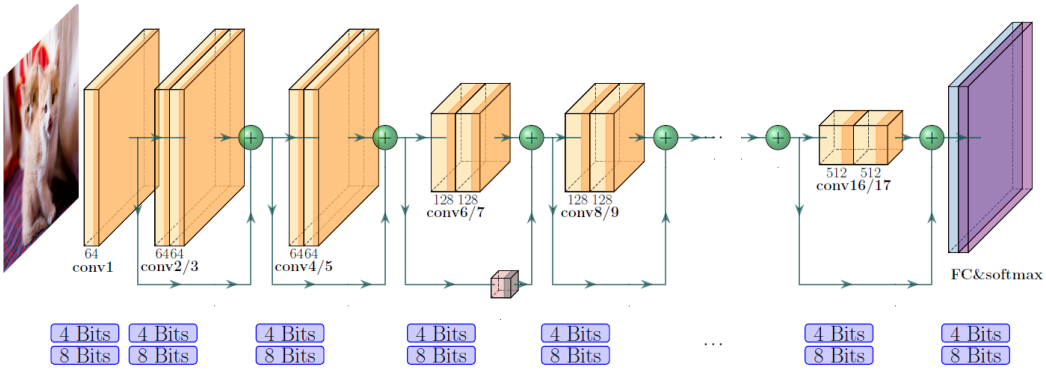
GeMM & CNN processors enhancements in ASIC, CPU, GPU

- Quantization in hardware
  - Envision (KU) Leuven
  - Bonus: IBM Hybrid training/inference in 7nm

Learning objective of this class

- you will be able to describe multi-precision hardware approaches and their trade-offs in terms of energy, throughput, and circuit overhead.
- you will be able to estimate energy trade-offs of multi-precision data paths.

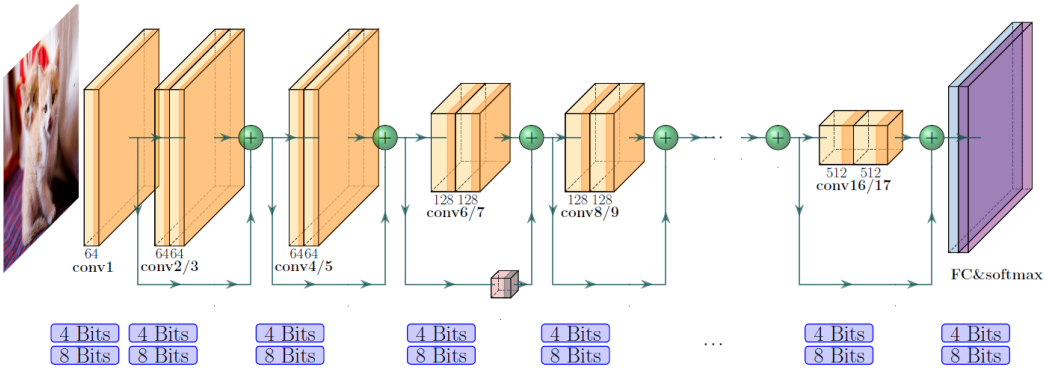
Recap: Mixed Precision Quantization



- Each layer has selected quantization (mix-precision quantization).

(Lecture 6) [Gholami et al, 2021]

Recap: Mixed Precision Quantization



- Each layer has selected quantization (mix-precision quantization).
- **Huge design space**, how can we support different quantized layers and networks?

(Lecture 6) [Gholami et al, 2021]

Outline

Recap

Sparsity
- Diving into state-of-the-art accelerators

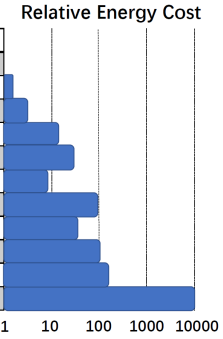
Quantization in HW
- Diving into state-of-the-art accelerators

Bonus: IBM Hybrid training / inference in 7nm

Quantization in HW many approaches!

Approaches

Operation:	Energy(pJ):
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8kb)	5.0
32b DRAM Read	640

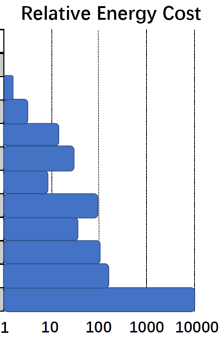


Quantization in HW many approaches!

Approaches

- **Reduce size of operands** for storage/compute (quantization)

Operation:	Energy(pJ):
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8kb)	5.0
32b DRAM Read	640

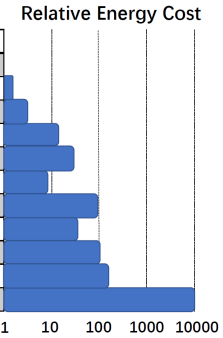


Quantization in HW many approaches!

Approaches

- **Reduce size of operands** for storage/compute (quantization)
  - Floating point $\Rightarrow$ Fixed point

Operation:	Energy(pJ):
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8kb)	5.0
32b DRAM Read	640

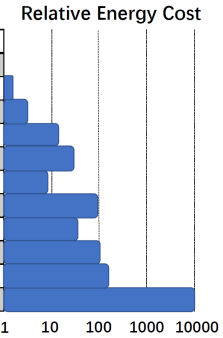


Quantization in HW many approaches!

Approaches

- **Reduce size of operands** for storage/compute (quantization)
  - Floating point $\Rightarrow$ Fixed point
  - Uniform quantization (bit-width reduction)

Operation:	Energy(pJ):
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8kb)	5.0
32b DRAM Read	640

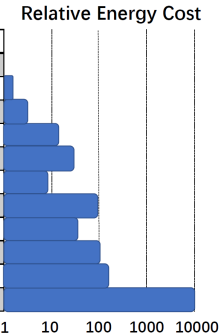


Quantization in HW many approaches!

Approaches

- **Reduce size of operands** for storage/compute (quantization)
  - Floating point $\Rightarrow$ Fixed point
  - Uniform quantization (bit-width reduction)
  - Non-uniform quantization

Operation:	Energy(pJ):
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8kb)	5.0
32b DRAM Read	640

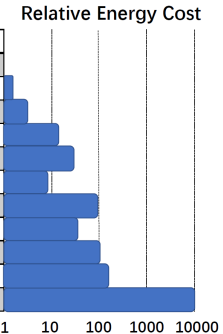


Quantization in HW many approaches!

Approaches

- **Reduce size of operands** for storage/compute (quantization)
  - Floating point $\Rightarrow$ Fixed point
  - Uniform quantization (bit-width reduction)
  - Non-uniform quantization
- **Reduce number of operations** for storage/compute (sparsity)

Operation:	Energy(pJ):
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8kb)	5.0
32b DRAM Read	640

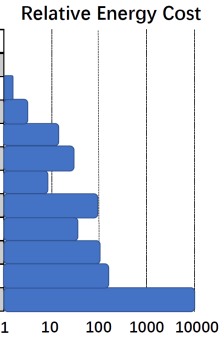


Quantization in HW many approaches!

Approaches

- **Reduce size of operands** for storage/compute (quantization)
  - Floating point $\Rightarrow$ Fixed point
  - Uniform quantization (bit-width reduction)
  - Non-uniform quantization
- **Reduce number of operations** for storage/compute (sparsity)
  - Network Pruning (unstructured/structured sparsity)

Operation:	Energy(pJ):
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8kb)	5.0
32b DRAM Read	640

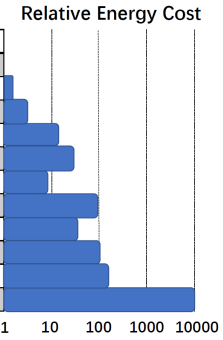


Quantization in HW many approaches!

Approaches

- **Reduce size of operands** for storage/compute (quantization)
  - Floating point $\Rightarrow$ Fixed point
  - Uniform quantization (bit-width reduction)
  - Non-uniform quantization
- **Reduce number of operations** for storage/compute (sparsity)
  - Network Pruning (unstructured/structured sparsity)
  - Exploit Activation Statistics (Compression)

Operation:	Energy(pJ):
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8kb)	5.0
32b DRAM Read	640

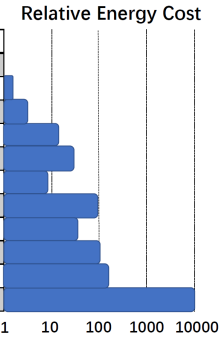


Quantization in HW many approaches!

Approaches

- **Reduce size of operands** for storage/compute (quantization)
  - Floating point $\Rightarrow$ Fixed point
  - Uniform quantization (bit-width reduction)
  - Non-uniform quantization
- **Reduce number of operations** for storage/compute (sparsity)
  - Network Pruning (unstructured/structured sparsity)
  - Exploit Activation Statistics (Compression)
  - Compact Network Architectures

Operation:	Energy(pJ):
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8kb)	5.0
32b DRAM Read	640

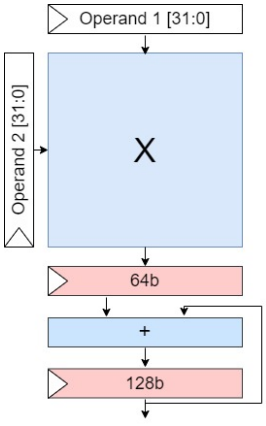


Number of bits for MAC operation

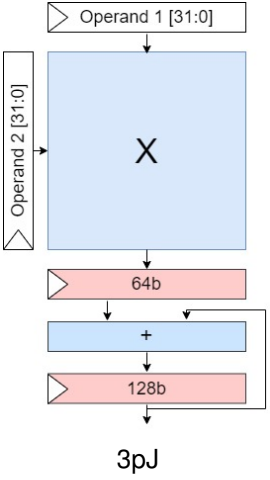
We have 2^m-bits and 2^n-bits numbers (MAC):

$$(2^n - 1) \cdot (2^m - 1) = 2^{n+m} - 2^n - 2^m + 1 \quad (2)$$

we need 2^{m+n} bits to store the result, plus some more bits for providing enough space for accumulating results.

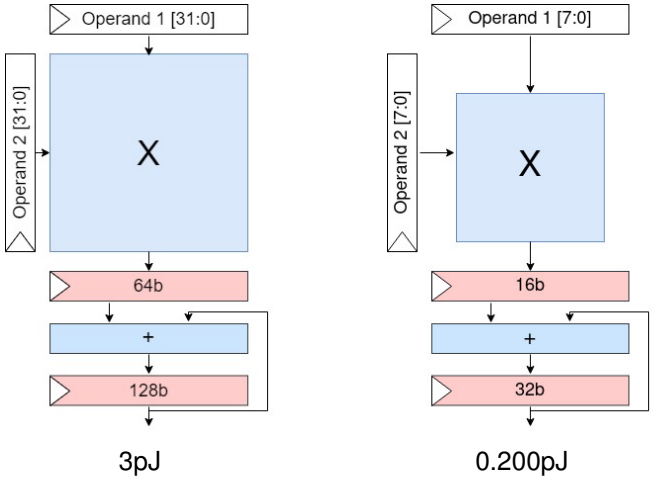


Quantized number computation



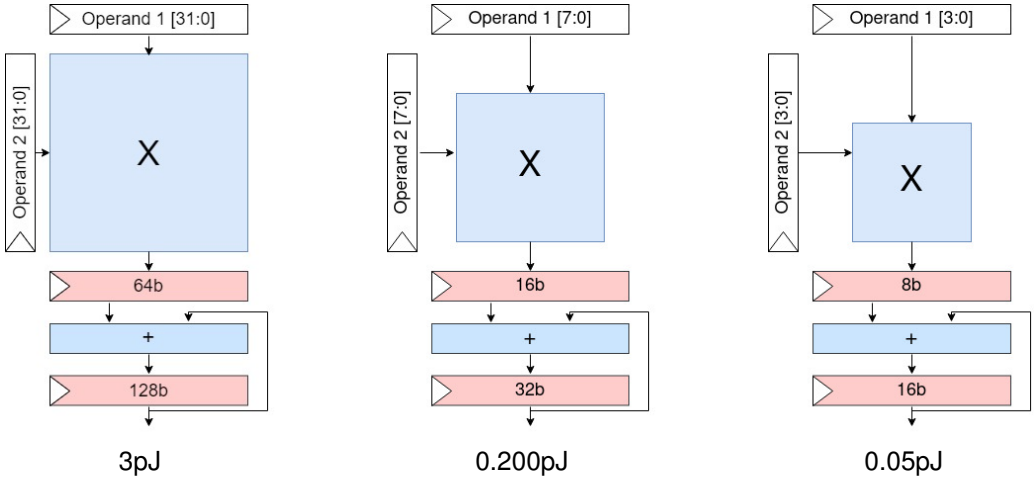
Energy consumption in CMOS 22nm

Quantized number computation



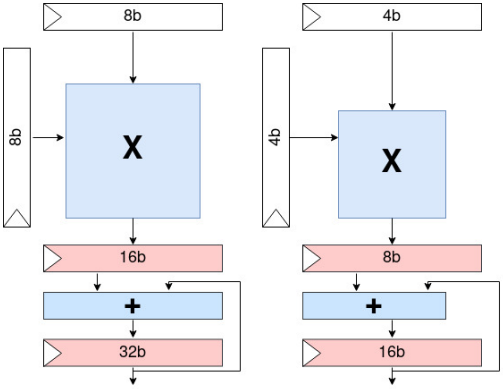
Energy consumption in CMOS 22nm

Quantized number computation



Energy consumption in CMOS 22nm

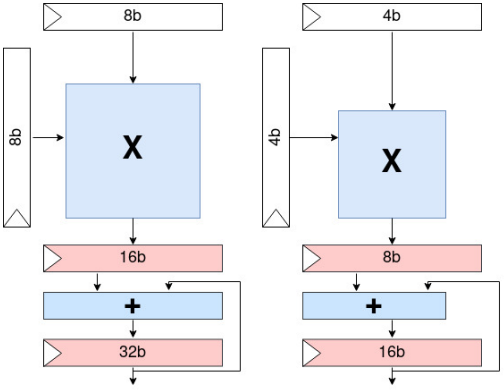
Quantization terminology



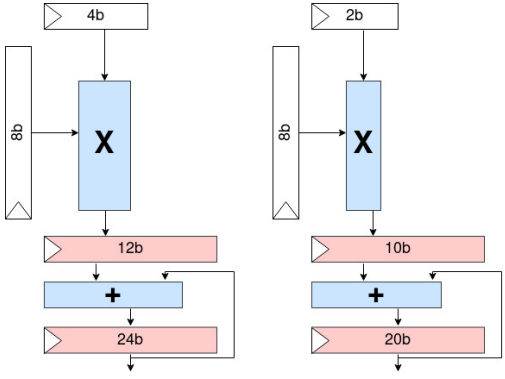
Symmetric precision

[Camus et al, 2019]

Quantization terminology



Symmetric precision



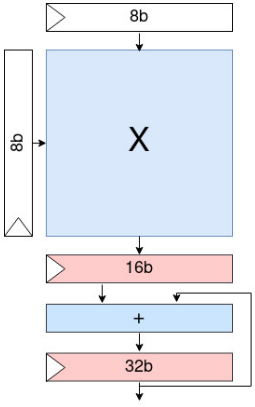
Asymmetric precision

Typically, fewer bits for weights

[Camus et al, 2019]

Variable precision MAC units

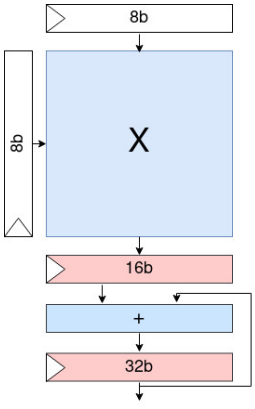
Data gating approach



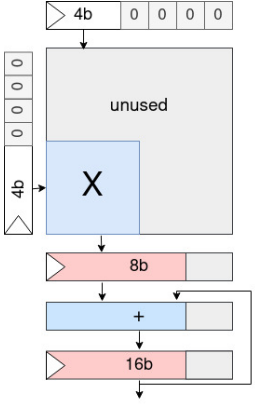
HW MAC unit

Variable precision MAC units

Data gating approach



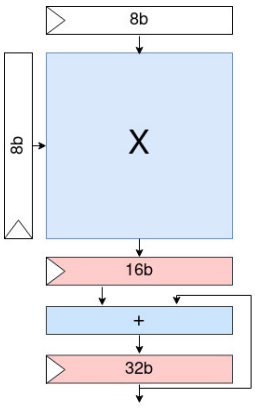
HW MAC unit



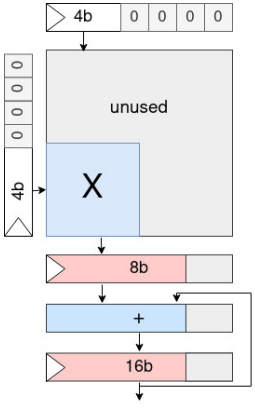
symmetric scaling

Variable precision MAC units

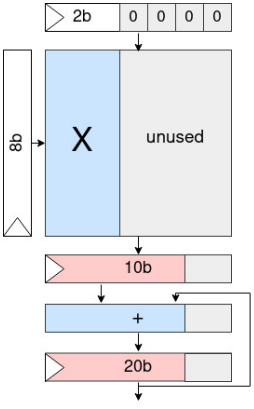
Data gating approach



HW MAC unit



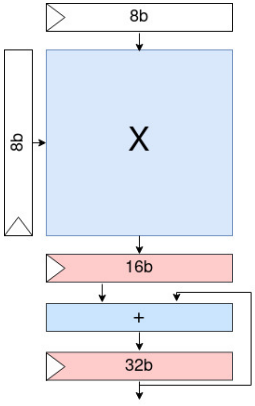
symmetric scaling



asymmetric scaling

Variable precision MAC units

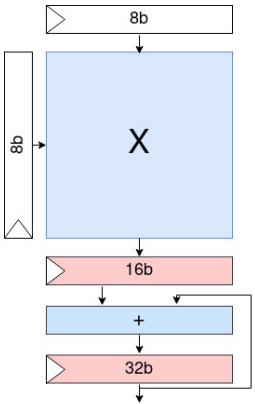
Multiplier-gating approach, i.e., gating of the multiplier [Moons et al 2017]



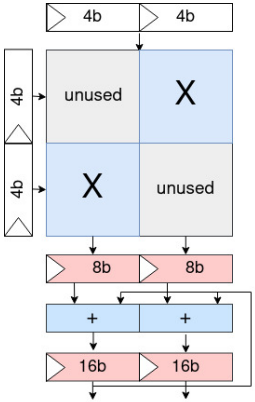
HW MAC unit

Variable precision MAC units

Multiplier-gating approach, i.e., gating of the multiplier [Moons et al 2017]



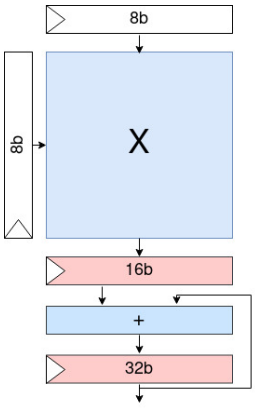
HW MAC unit



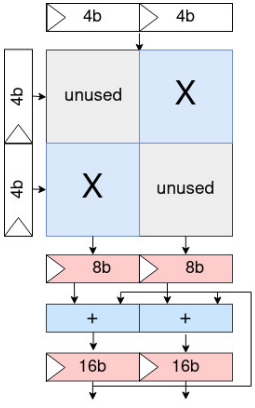
symmetric scaling

Variable precision MAC units

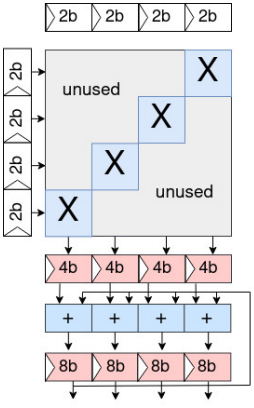
Multiplier-gating approach, i.e., gating of the multiplier [Moons et al 2017]



HW MAC unit



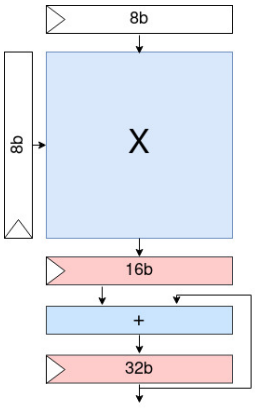
symmetric scaling



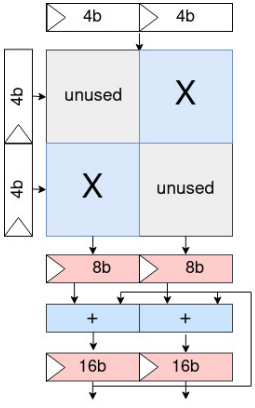
symmetric scaling

Variable precision MAC units

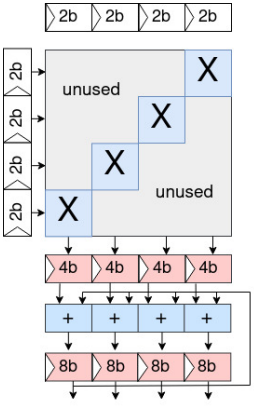
Multiplier-gating approach, i.e., gating of the multiplier [Moons et al 2017]



HW MAC unit



symmetric scaling

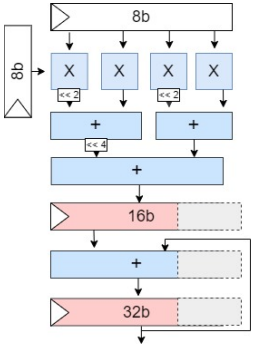


symmetric scaling

- Only for symmetric scaling!

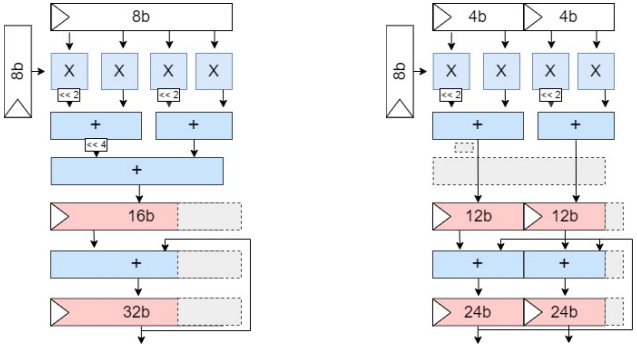
Variable precision MAC units

Add/shift-gating, i.e., divide-and-conquer (D&C) [Shin et al, 2017]



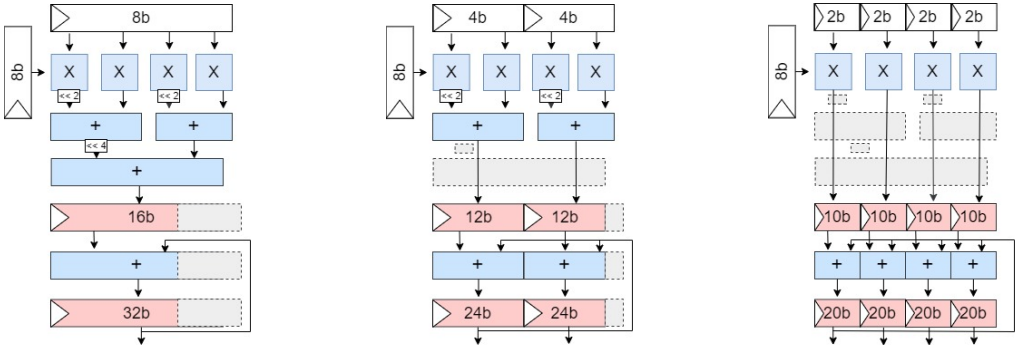
Variable precision MAC units

Add/shift-gating, i.e., divide-and-conquer (D&C) [Shin et al, 2017]



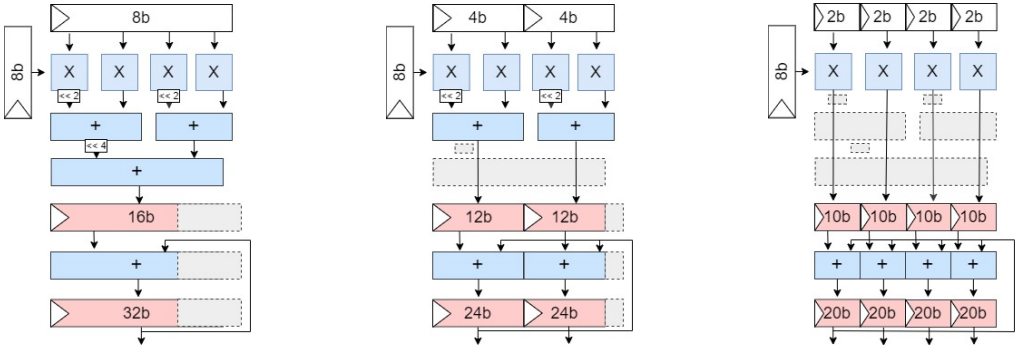
Variable precision MAC units

Add/shift-gating, i.e., divide-and-conquer (D&C) [Shin et al, 2017]



Variable precision MAC units

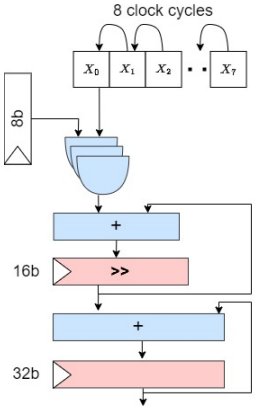
Add/shift-gating, i.e., divide-and-conquer (D&C) [Shin et al, 2017]



- It is inefficient for symmetric scaling, but it is efficient for asymmetric scaling

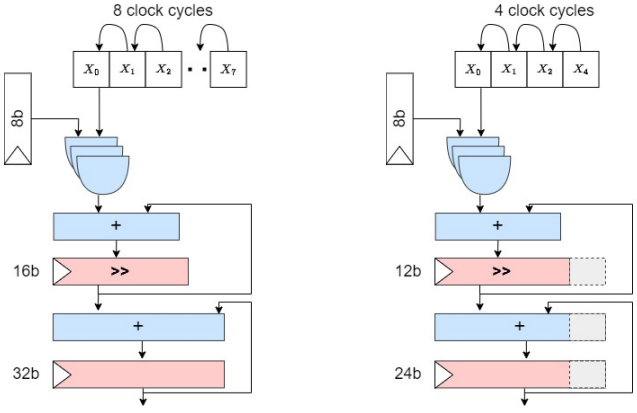
Variable precision MAC units

Bit-serial approach (asymmetric) 1D



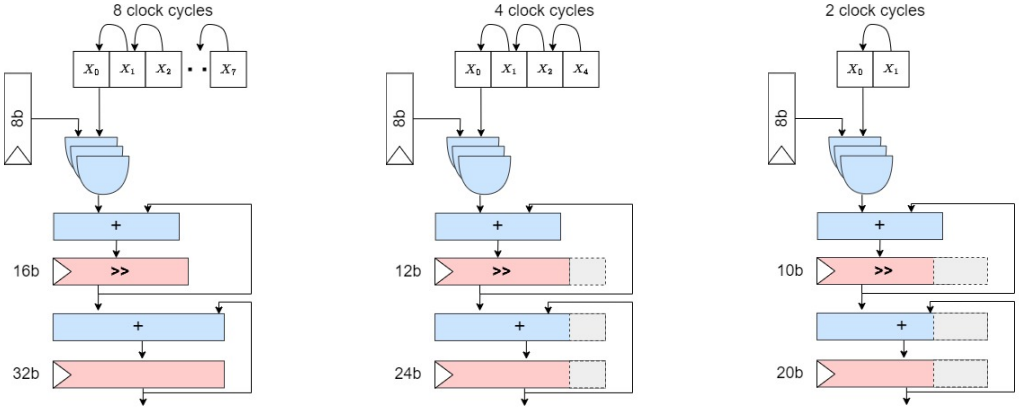
Variable precision MAC units

Bit-serial approach (asymmetric) 1D



Variable precision MAC units

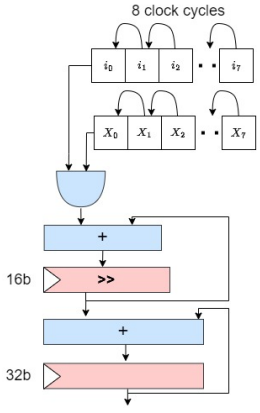
Bit-serial approach (asymmetric) 1D



- Easy to make precision scalable (weights only).

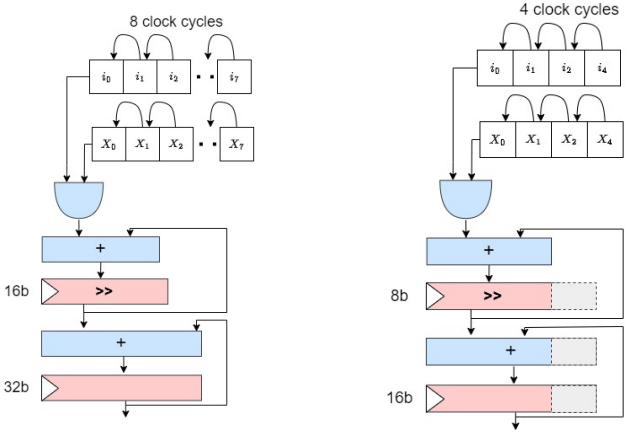
Variable precision MAC units

Bit-serial approach (symmetric) 2D



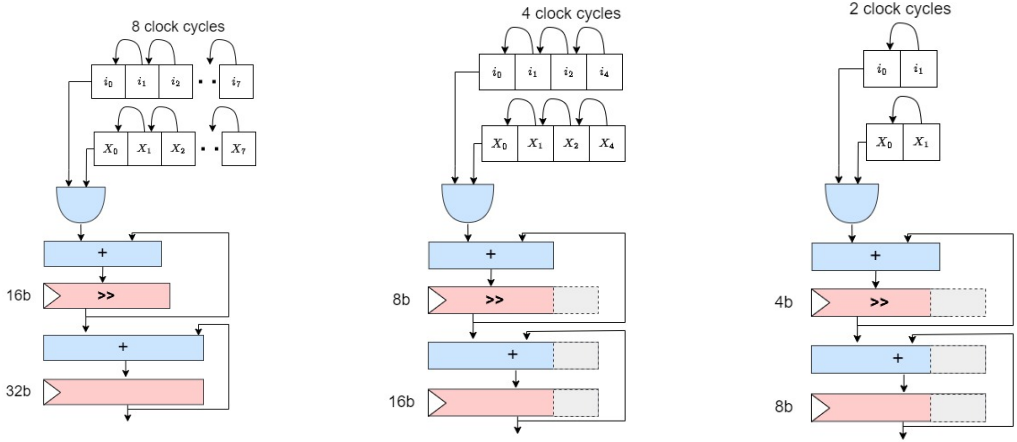
Variable precision MAC units

Bit-serial approach (symmetric) 2D



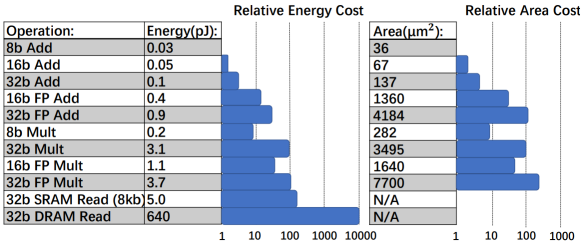
Variable precision MAC units

Bit-serial approach (symmetric) 2D



- Easy to make precision scalable.

Variable precision MAC units: comparisons

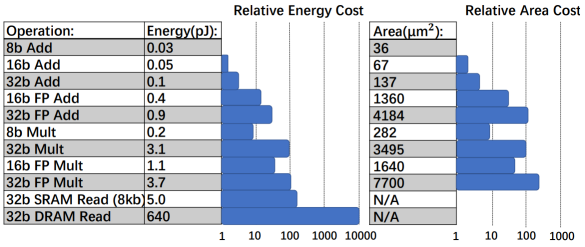


Operation:	Energy(pJ):	Area(μm²):
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8kb)	5.0	N/A
32b DRAM Read	640	N/A

how to compare?

- Some architectures are small but fast, or slow but efficient. Which one is the optimal? What metric matters?

Variable precision MAC units: comparisons

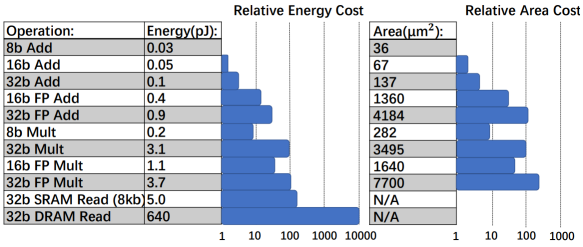


Operation:	Energy(pJ):	Area(μm²):
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8kb)	5.0	N/A
32b DRAM Read	640	N/A

how to compare?

- Some architectures are small but fast, or slow but efficient. Which one is the optimal? What metric matters?
  - Throughput per unit area

Variable precision MAC units: comparisons

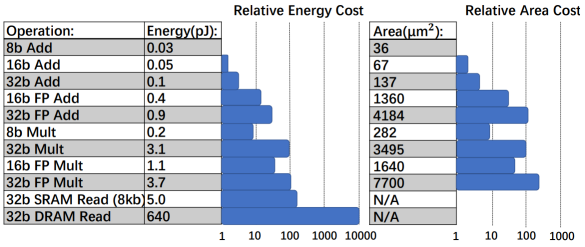


Operation:	Energy(pJ):	Area(μm^2):
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8kb)	5.0	N/A
32b DRAM Read	640	N/A

how to compare?

- Some architectures are small but fast, or slow but efficient. Which one is the optimal? What metric matters?
  - Throughput per unit area
  - Energy per MAC operation performed

Variable precision MAC units: comparisons

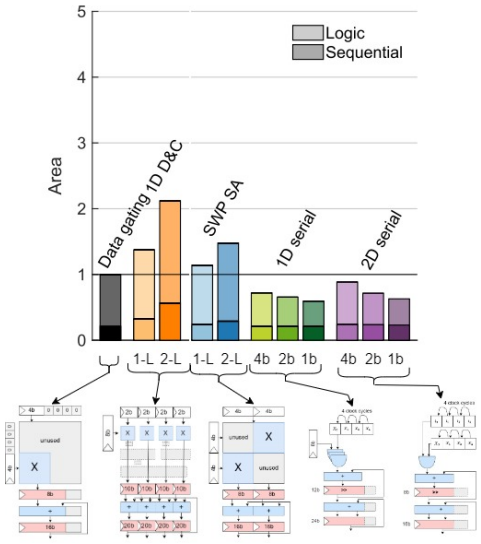


Operation:	Energy(pJ):	Area(μm²):
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8kb)	5.0	N/A
32b DRAM Read	640	N/A

how to compare?

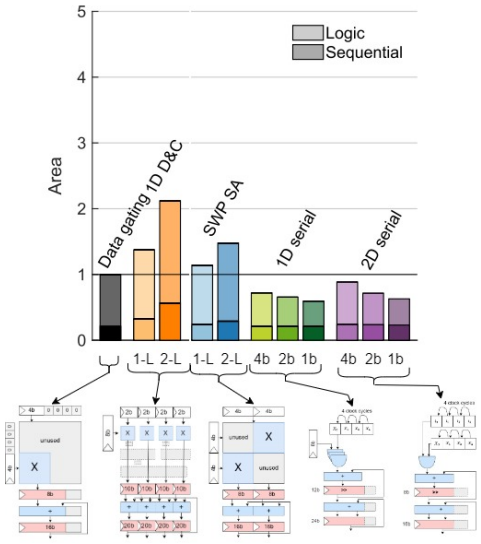
- Some architectures are small but fast, or slow but efficient. Which one is the optimal? What metric matters?
  - Throughput per unit area
  - Energy per MAC operation performed
- A comprehensive comparison has been done! [Camus et al 2019]

Variable precision MAC units: comparisons



Area (normalized)

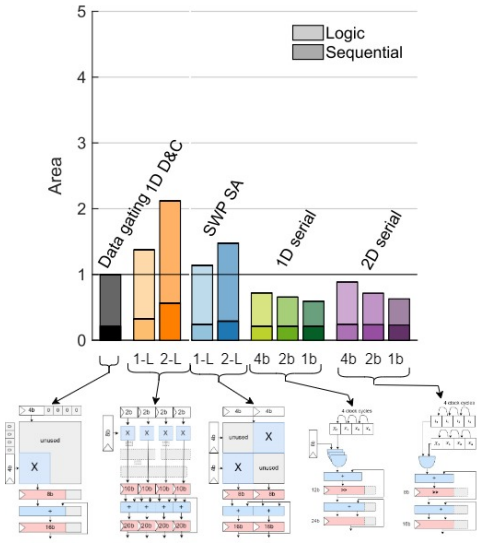
Variable precision MAC units: comparisons



Area (normalized)

- bit-serial approaches are smaller

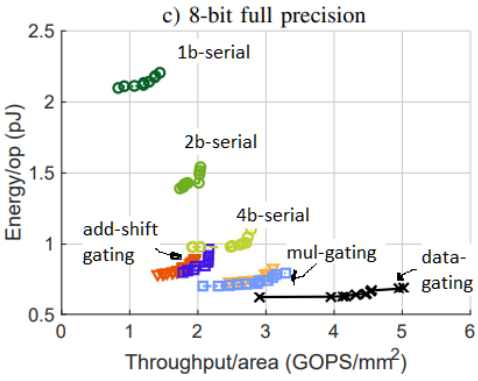
Variable precision MAC units: comparisons

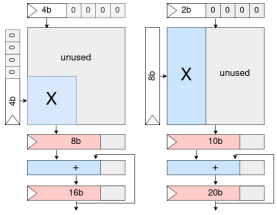


Area (normalized)

- bit-serial approaches are smaller
- add-shift and multiplier-gating approaches require more area

Variable precision MAC units: comparisons



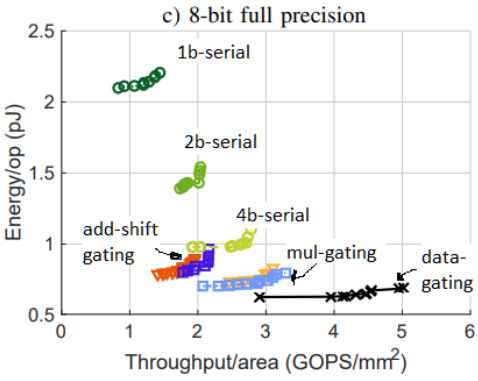


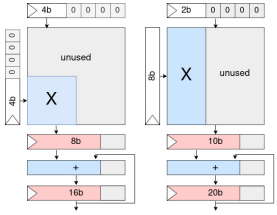
Energy vs Throughput/area

- data gating is the fastest and most efficient... why?

[Camus et al 2017]

Variable precision MAC units: comparisons



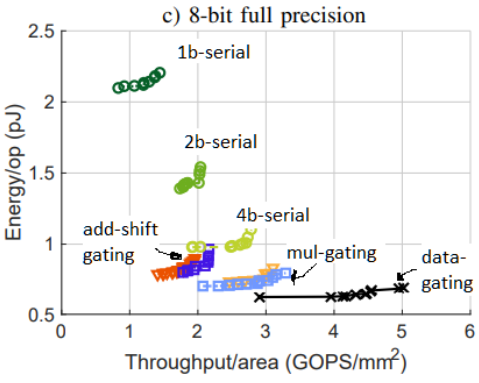


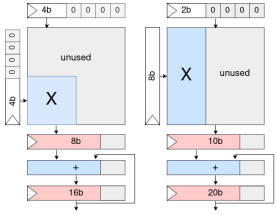
Energy vs Throughput/area

- data gating is the fastest and most efficient... why?
- data-gating has the lowest amount of overhead

[Camus et al 2017]

Variable precision MAC units: comparisons





Energy vs Throughput/area

- data gating is the fastest and most efficient... why?
- data-gating has the lowest amount of overhead
- others have 20% (Mul & Add/shift gating) to 200% (serial) overhead

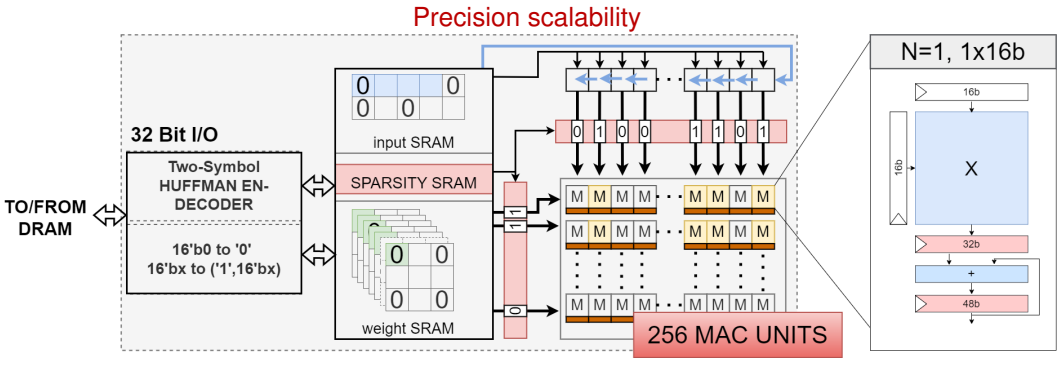
[Camus et al 2017]

Lesson Plan

GeMM & CNN processors enhancements in ASIC, CPU, GPU

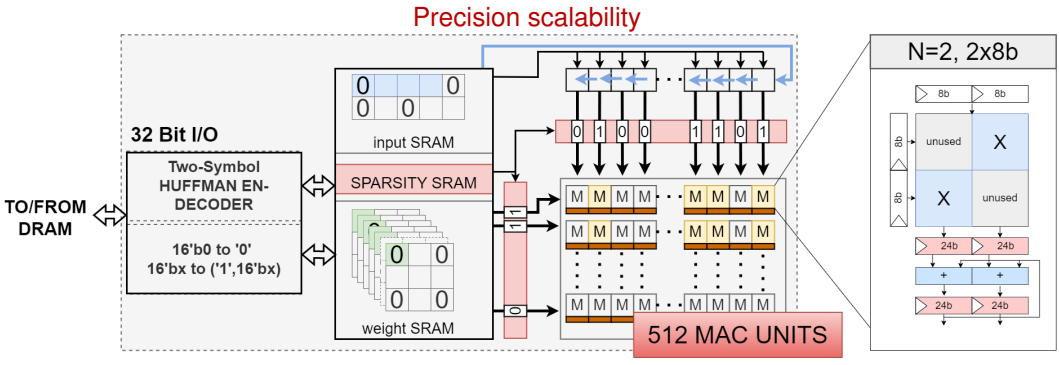
- Quantization
  - Envision (KU) Leuven
  - IBM Hybrid training/inference in 7nm

Variable precision MAC units: Envision (KU) Leuven

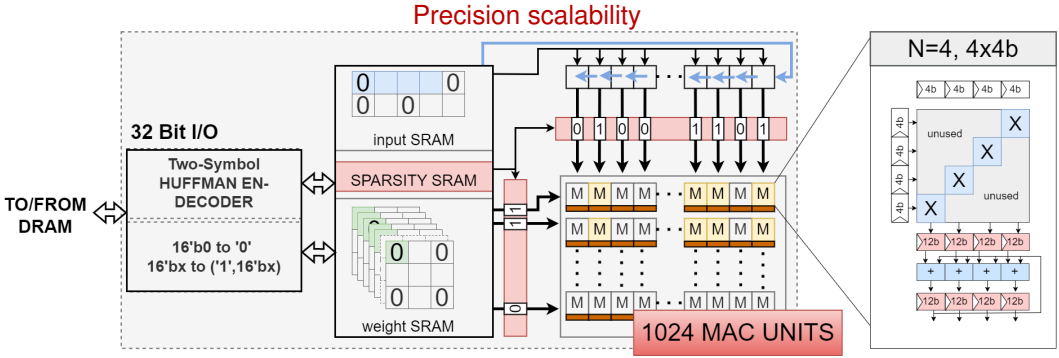


TU/e

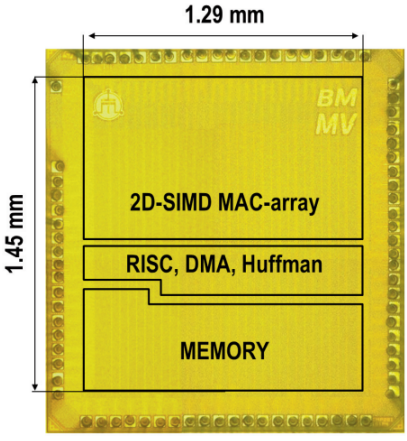
Variable precision MAC units: Envision (KU) Leuven



Variable precision MAC units: Envision (KU) Leuven



Variable precision MAC units: Envision (KU) Leuven



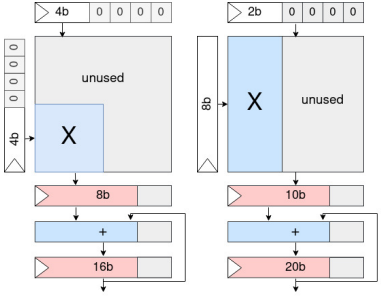
[Moons et al, 2017]

- datapath has different supply domains from the rest of the processor
- 2b,4b,8b modes different shorter/longer critical paths (less/more capacitance, less/more slack), thus voltage can be scaled!

Exploiting dynamic precision for energy benefits

Dynamic-Accuracy-Scaling

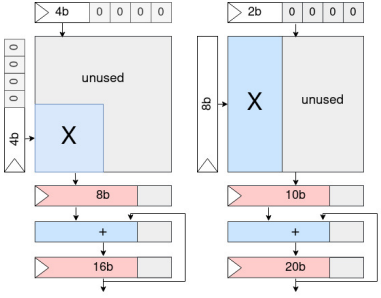
- $f \rightarrow$ the operating frequency



Exploiting dynamic precision for energy benefits

Dynamic-Accuracy-Scaling

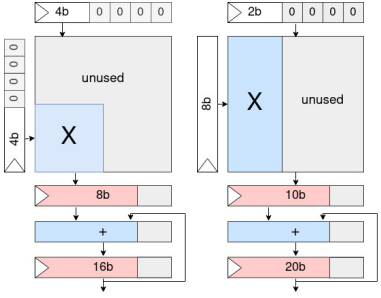
- $f \rightarrow$ the operating frequency
- $C \rightarrow$ switching capacitance



Exploiting dynamic precision for energy benefits

Dynamic-Accuracy-Scaling

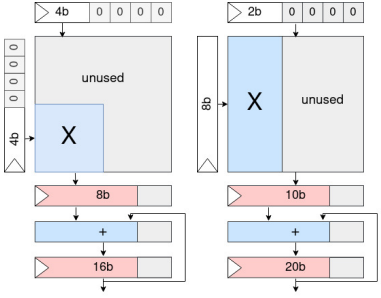
- $f \rightarrow$ the operating frequency
- $C \rightarrow$ switching capacitance
- $V \rightarrow$ supply voltage



Exploiting dynamic precision for energy benefits

Dynamic-Accuracy-Scaling

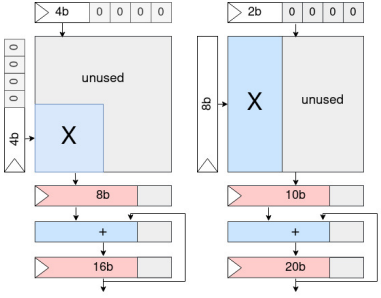
- $f \rightarrow$ the operating frequency
- $C \rightarrow$ switching capacitance
- $V \rightarrow$ supply voltage
- $\alpha \rightarrow$ circuit's switching activity



Exploiting dynamic precision for energy benefits

Dynamic-Accuracy-Scaling

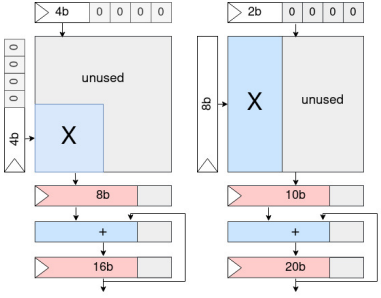
- $f \rightarrow$ the operating frequency
- $C \rightarrow$ switching capacitance
- $V \rightarrow$ supply voltage
- $\alpha \rightarrow$ circuit's switching activity
- $as \rightarrow$ modules with scaled precision



Exploiting dynamic precision for energy benefits

Dynamic-Accuracy-Scaling

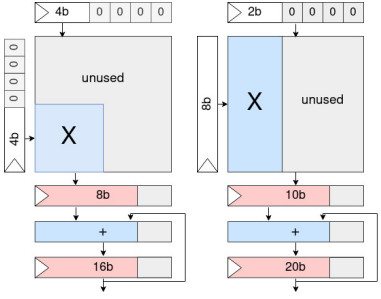
- $f \rightarrow$ the operating frequency
- $C \rightarrow$ switching capacitance
- $V \rightarrow$ supply voltage
- $\alpha \rightarrow$ circuit's switching activity
- *as* $\rightarrow$ modules with scaled precision
- *nas* $\rightarrow$ which does not modulate with scaled precision



Exploiting dynamic precision for energy benefits

Dynamic-Accuracy-Scaling

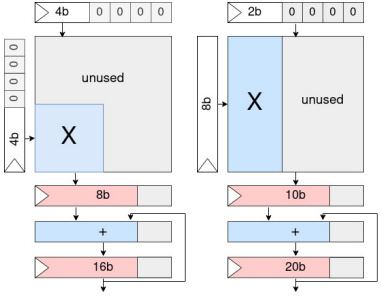
- $f \rightarrow$ the operating frequency
- $C \rightarrow$ switching capacitance
- $V \rightarrow$ supply voltage
- $\alpha \rightarrow$ circuit's switching activity
- *as* $\rightarrow$ modules with scaled precision
- *nas* $\rightarrow$ which does not modulate with scaled precision
- $\kappa_0 \rightarrow$ depends on precision, circuit, architecture dependent



Exploiting dynamic precision for energy benefits

Dynamic-Accuracy-Scaling

- $f \rightarrow$ the operating frequency
- $C \rightarrow$ switching capacitance
- $V \rightarrow$ supply voltage
- $\alpha \rightarrow$ circuit's switching activity
- $as \rightarrow$ modules with scaled precision
- $nas \rightarrow$ which does not modulate with scaled precision
- $\kappa_0 \rightarrow$ depends on precision, circuit, architecture dependent



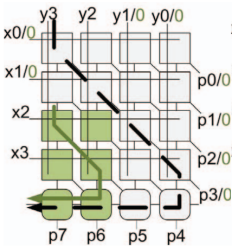
$$P_{DAS} = \frac{\alpha_{as}}{\kappa_0} \cdot C_{as} f V^2 + \alpha_{nas} \cdot C_{nas} f V^2 \qquad (3)$$

leakage power is neglected!

Exploiting dynamic precision for energy benefits

Dynamic-Voltage-Accuracy-Scaling

- $\alpha \rightarrow$ circuit's switching activity
- $f \rightarrow$ the operating frequency
- $C \rightarrow$ switching capacitance
- $V \rightarrow$ supply voltage
- $as \rightarrow$ modules with scaled precision
- $nas \rightarrow$ does not modulate with scaled precision
- κ_0 is a precision, circuit, architecture dependent
- $V_{as} \rightarrow$ accuracy scalable

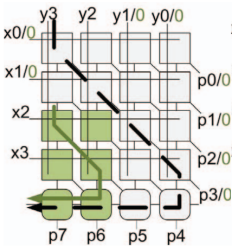


(a) DVAS example

Exploiting dynamic precision for energy benefits

Dynamic-Voltage-Accuracy-Scaling

- $\alpha \rightarrow$ circuit's switching activity
- $f \rightarrow$ the operating frequency
- $C \rightarrow$ switching capacitance
- $V \rightarrow$ supply voltage
- $as \rightarrow$ modules with scaled precision
- $nas \rightarrow$ does not modulate with scaled precision
- κ_0 is a precision, circuit, architecture dependent
- $V_{as} \rightarrow$ accuracy scalable
- $V_{nas} \rightarrow$ non-accuracy-scalable

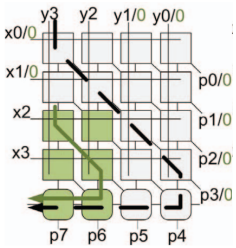


(a) DVAS example

Exploiting dynamic precision for energy benefits

Dynamic-Voltage-Accuracy-Scaling

- $\alpha \rightarrow$ circuit's switching activity
- $f \rightarrow$ the operating frequency
- $C \rightarrow$ switching capacitance
- $V \rightarrow$ supply voltage
- $as \rightarrow$ modules with scaled precision
- $nas \rightarrow$ does not modulate with scaled precision
- κ_0 is a precision, circuit, architecture dependent
- $V_{as} \rightarrow$ accuracy scalable
- $V_{nas} \rightarrow$ non-accuracy-scalable

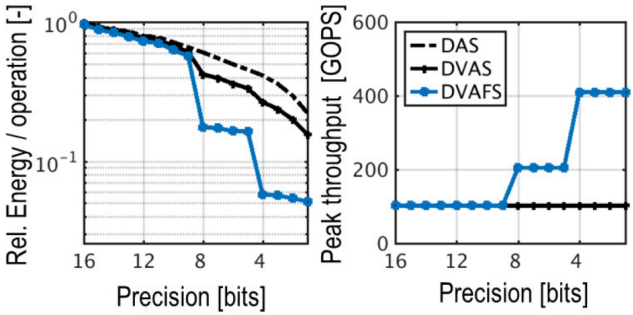


(a) DVAS example

$$P_{DVAS} = \frac{\alpha_{as}}{\kappa_1} \cdot C_{as} f \left(\frac{V_{as}}{\kappa_2} \right)^2 + \alpha_{nas} \cdot C_{nas} f V_{nas}^2 \quad (4)$$

leakage power is neglected! [Moons et al, 2017]

Variable precision MAC units: Envision (KU) Leuven



- Throughput goes up
- Energy efficiency as well ($\sim$20X)
- Dynamic-Voltage-Accuracy Frequency Scaling (DVAF), Dynamic-Accuracy Scaling (DAS), Dynamic-Voltage-Accuracy Scaling (DVAS).

Advanced CMOS technologies FD-SOI

Image: STMicroelectronics

Fully-Depleted Silicon-On-Insulator [analog design]

- Added a buried oxide layer that isolate the channel from the bulk

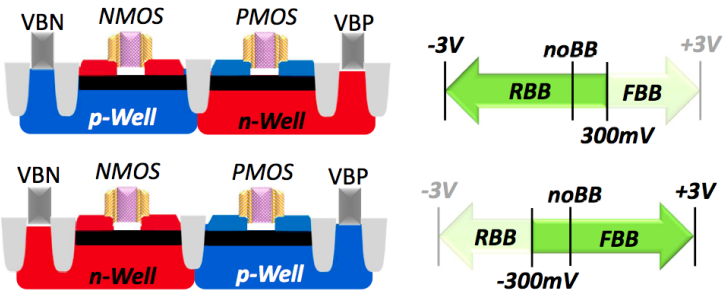
Advanced CMOS technologies FD-SOI

Image: STMicroelectronics

Fully-Depleted Silicon-On-Insulator [analog design]

- Added a buried oxide layer that isolate the channel from the bulk
- Very thin, controllable channel with less leakage

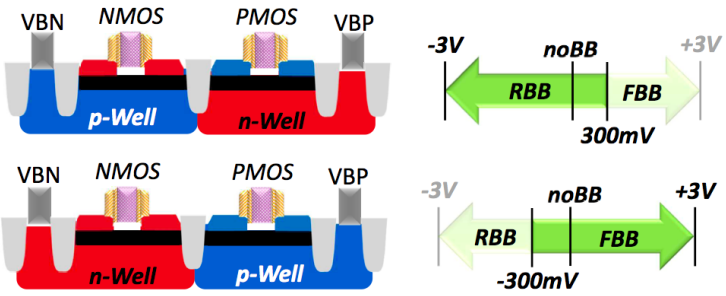
Advanced CMOS technologies FD-SOI



Body Biasing [tech design forum]

- The body of the transistor is biased to a voltage
  - higher than V_{dd} PMOS
  - lower than V_{ss} NMOS

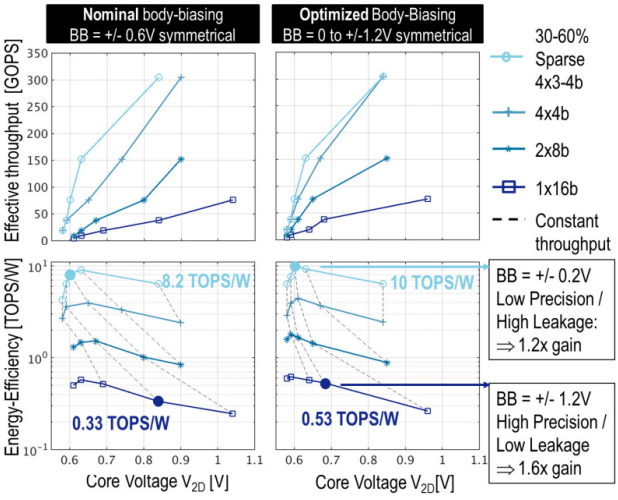
Advanced CMOS technologies FD-SOI



Body Biasing [tech design forum]

- The body of the transistor is biased to a voltage
  - higher than V_{dd} PMOS
  - lower than V_{ss} NMOS
- Technique mainly used to reduce leackage power

Variable precision MAC units: Envision (KU) Leuven



[Moons et al, 2017]

- 28nm FDSOI
- 10TOPs/W @ peak-performance!

Workload Envision

Layer	Mode	f [MHz]	V [V]	Wght. [b]	In. [b]	Wght. sp. [%]	In. sp. [%]	MMACS/ frame	Power [mW]	Eff. [TOPS/W]
VGG1	$2 \times 8b$	100	0.80	5	4	5	10	87	25	2.1
VGG2-13	$2 \times 8b$	100	0.80	5	6	25-75	30-82	462-1850	19-35	1.5-2.8
Total	–	–	–	–	–	–	–	15346	26	2
AlexNet1	$2 \times 8b$	100	0.80	7	4	21	29	104	37	2.7
AlexNet2	$2 \times 8b$	100	0.80	7	7	19	89	224	20	3.8
AlexNet3	$1 \times 16b$	200	1.03	8	9	11	82	150	52	1
AlexNet4-5	$1 \times 16b$	200	1.03	9	8	4	72	112	58-62	0.8-0.9
Total	–	–	–	–	–	–	–	666	44	1.8
LeNet1	$4 \times 4b$	50	0.65	3	1	35	87	0.3	5.6	13.6
LeNet2	$2 \times 8b$	100	0.80	4	6	26	55	1.6	29	2.6
Total	–	–	–	–	–	–	–	1.9	25	3

[Moons et al, 2017]

Workload Envision

Layer	Mode	f [MHz]	V [V]	Wght. [b]	In. [b]	Wght. sp. [%]	In. sp. [%]	MMACS/ frame	Power [mW]	Eff. [TOPS/W]
VGG1	$2 \times 8b$	100	0.80	5	4	5	10	87	25	2.1
VGG2-13	$2 \times 8b$	100	0.80	5	6	25-75	30-82	462-1850	19-35	1.5-2.8
Total	–	–	–	–	–	–	–	15346	26	2
AlexNet1	$2 \times 8b$	100	0.80	7	4	21	29	104	37	2.7
AlexNet2	$2 \times 8b$	100	0.80	7	7	19	89	224	20	3.8
AlexNet3	$1 \times 16b$	200	1.03	8	9	11	82	150	52	1
AlexNet4-5	$1 \times 16b$	200	1.03	9	8	4	72	112	58-62	0.8-0.9
Total	–	–	–	–	–	–	–	666	44	1.8
LeNet1	$4 \times 4b$	50	0.65	3	1	35	87	0.3	5.6	13.6
LeNet2	$2 \times 8b$	100	0.80	4	6	26	55	1.6	29	2.6
Total	–	–	–	–	–	–	–	1.9	25	3

Summary Envision

- DVAFS is a circuit level technique to trade-off energy for computational accuracy
- 28nm FDSOI - $5.6 - 62mW$ for various CNNs
- 1.8-3 TOPS/W with all the features!

[Moons et al, 2017]

Summary of Quantization in HW

Summary

- **Impact on accuracy**

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training
    - e.g., what quantization scheme does my HW support?

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training
    - e.g., what quantization scheme does my HW support?
- **Does hardware cost exceed benefits?**

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training
    - e.g., what quantization scheme does my HW support?
- **Does hardware cost exceed benefits?**
  - Need extra hardware to support variable precision

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training
    - e.g., what quantization scheme does my HW support?
- **Does hardware cost exceed benefits?**
  - Need extra hardware to support variable precision
    1. drawback: additional shift-and-add logic and registers for variable precision

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training
    - e.g., what quantization scheme does my HW support?
- **Does hardware cost exceed benefits?**
  - Need extra hardware to support variable precision
    1. drawback: additional shift-and-add logic and registers for variable precision
    2. benefits: can apply voltage-accuracy scaling (frequency as well if symmetric)

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training
    - e.g., what quantization scheme does my HW support?
- **Does hardware cost exceed benefits?**
  - Need extra hardware to support variable precision
    1. drawback: additional shift-and-add logic and registers for variable precision
    2. benefits: can apply voltage-accuracy scaling (frequency as well if symmetric)
- **Evision**

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training
    - e.g., what quantization scheme does my HW support?
- **Does hardware cost exceed benefits?**
  - Need extra hardware to support variable precision
    1. drawback: additional shift-and-add logic and registers for variable precision
    2. benefits: can apply voltage-accuracy scaling (frequency as well if symmetric)
- **Evision**
  - Many features in a single chip!

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training
    - e.g., what quantization scheme does my HW support?
- **Does hardware cost exceed benefits?**
  - Need extra hardware to support variable precision
    1. drawback: additional shift-and-add logic and registers for variable precision
    2. benefits: can apply voltage-accuracy scaling (frequency as well if symmetric)
- **Evision**
  - Many features in a single chip!
    - spatial data reuse: weight reuse, input reuse

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training
    - e.g., what quantization scheme does my HW support?
- **Does hardware cost exceed benefits?**
  - Need extra hardware to support variable precision
    1. drawback: additional shift-and-add logic and registers for variable precision
    2. benefits: can apply voltage-accuracy scaling (frequency as well if symmetric)
- **Evision**
  - Many features in a single chip!
    - spatial data reuse: weight reuse, input reuse
    - temporal output reuse

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training
    - e.g., what quantization scheme does my HW support?
- **Does hardware cost exceed benefits?**
  - Need extra hardware to support variable precision
    1. drawback: additional shift-and-add logic and registers for variable precision
    2. benefits: can apply voltage-accuracy scaling (frequency as well if symmetric)
- **Evision**
  - Many features in a single chip!
    - spatial data reuse: weight reuse, input reuse
    - temporal output reuse
    - pseudo input stationarity

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training
    - e.g., what quantization scheme does my HW support?
- **Does hardware cost exceed benefits?**
  - Need extra hardware to support variable precision
    1. drawback: additional shift-and-add logic and registers for variable precision
    2. benefits: can apply voltage-accuracy scaling (frequency as well if symmetric)
- **Evision**
  - Many features in a single chip!
    - spatial data reuse: weight reuse, input reuse
    - temporal output reuse
    - pseudo input stationarity
    - compressed I/O

Summary of Quantization in HW

Summary

- **Impact on accuracy**
  - Must consider quantization during training
    - e.g., what quantization scheme does my HW support?
- **Does hardware cost exceed benefits?**
  - Need extra hardware to support variable precision
    1. drawback: additional shift-and-add logic and registers for variable precision
    2. benefits: can apply voltage-accuracy scaling (frequency as well if symmetric)
- **Evision**
  - Many features in a single chip!
    - spatial data reuse: weight reuse, input reuse
    - temporal output reuse
    - pseudo input stationarity
    - compressed I/O
    - multiplier-gating approach (DVAFS)

Next Time

Next time

- Extreme Quantization:
  - Binary Neural Networks
  - Binareye (KU Leuven) / Stanford

Outline

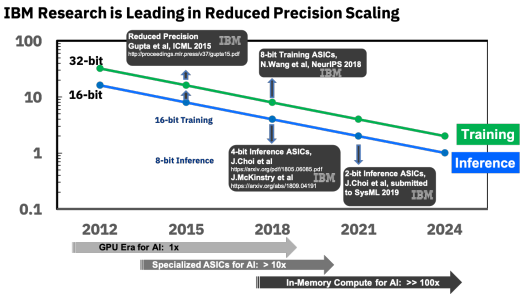
Recap

Sparsity
- Diving into state-of-the-art accelerators

Quantization in HW
- Diving into state-of-the-art accelerators

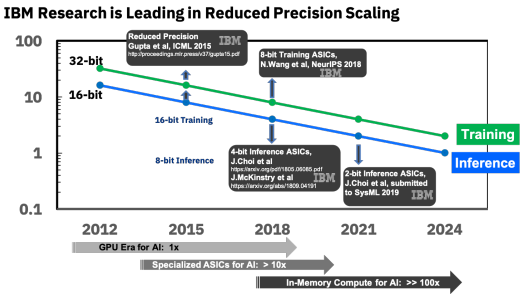
Bonus: IBM Hybrid training / inference in 7nm

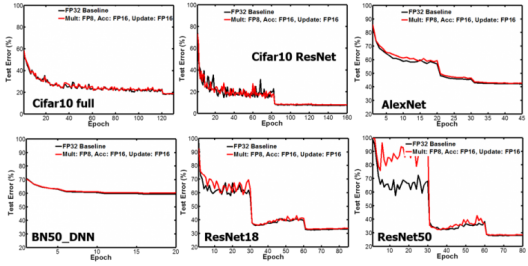
IBM Hybrid training / inference in 7nm



[IBM blog]

IBM Hybrid training / inference in 7nm



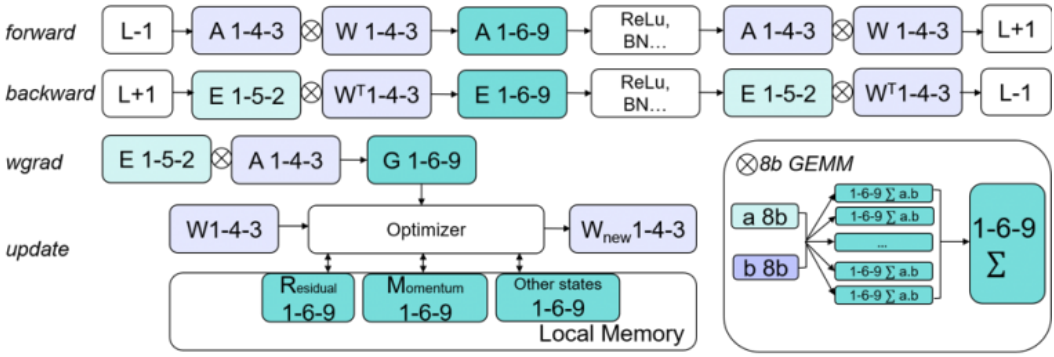


[IBM blog]

IBM conclusions

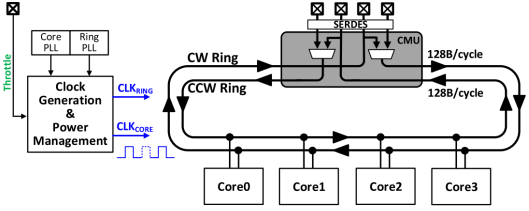
FP8 techniques achieve equivalent model accuracy to FP32 across a spectrum of models and datasets.

IBM Hybrid training / inference in 7nm



- Training requires good dynamic range (higher precision)
- Nevertheless, they use FP8 for training
  - A activation, W weights, E error (loss), G gradient
  - e.g., A 1-4-4 (sign 1 bit, 4 bits exponent, 4 bits mantissa)

IBM Hybrid training / inference in 7nm

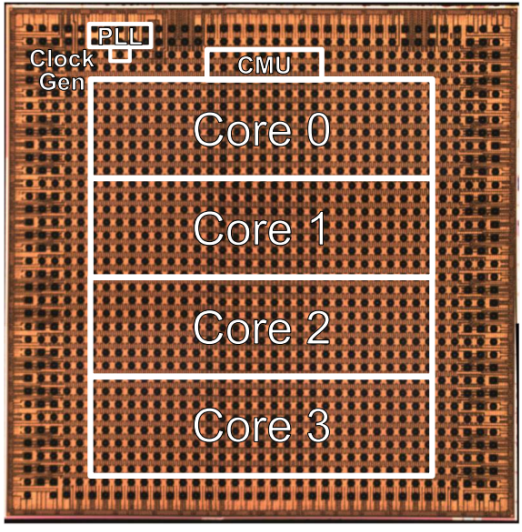


[Agrawal et al, 2022]

Chip in 7 nm

- 4 Core chip with ring interconnect

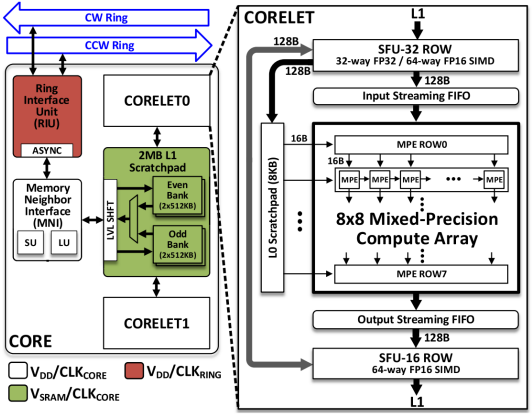
IBM Hybrid training / inference in 7nm



Chip in 7 nm

- 4 Core chip with ring interconnect

IBM Hybrid training / inference in 7nm

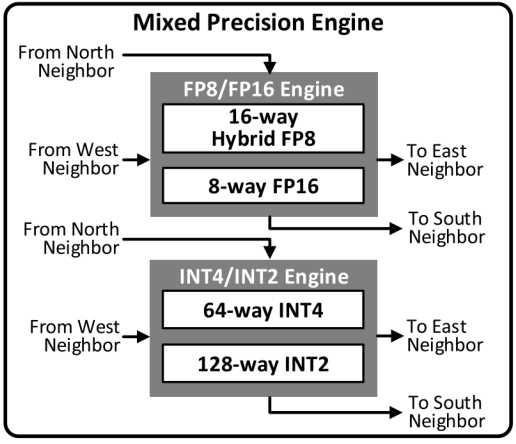


[Agrawal et al, 2022]

Chip in 7 nm

- 4 Core chip with ring interconnect
- The core is a 2D array (8x8 array of processor cores, 64 cores), i.e., it is a vectorized MUL, ADD with output reuse datapath

IBM Hybrid training / inference in 7nm

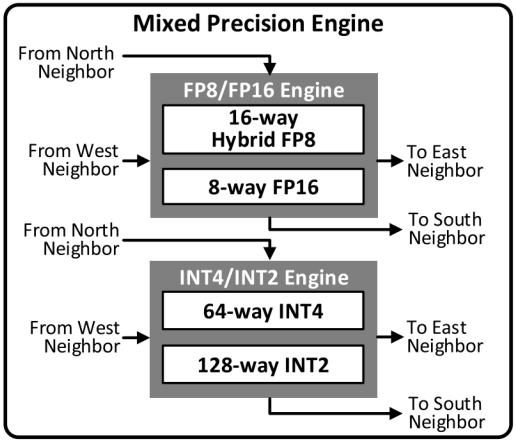


[Agrawal et al, 2022]

Chip in 7 nm

- 4 Core chip with ring interconnect
- The core is a 2D array (8x8 array of processor cores, 64 cores), i.e., it is a vectorized MUL, ADD with output reuse datapath
- inside each core 16-way parallel floating point engines or a 64-way fixed point engine

IBM Hybrid training / inference in 7nm



[Agrawal et al, 2022]

Inside Mixed Processing Engine

- one data path for floating point that can scale to integer or two separate ones?

IBM Hybrid training / inference in 7nm

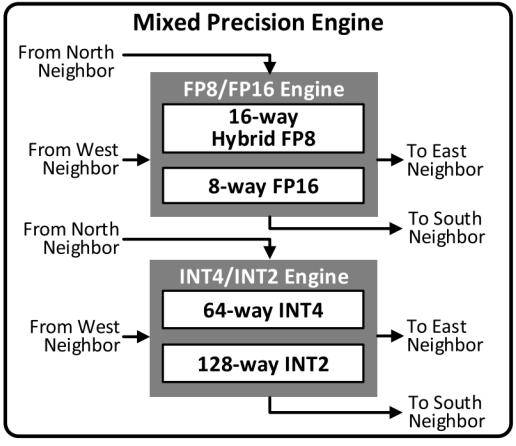
	TOP/W (normalized)			Area
	FP16	FP8	INT4	
Merged FP and INT pipelines	1.0x	2.2x	5.6x	1.0x
Separate FP and INT pipelines	1.1x	2.3x	10.6x	1.16x

[Agrawal et al, 2022]

Inside Mixed Processing Engine

- one data path for floating point that can scale to integer or two separate ones?
- separate FP16/8 and INT data paths to save up to 2x energy for 16% larger area

IBM Hybrid training / inference in 7nm



[Agrawal et al, 2022]

Inside Mixed Processing Engine

- one data path for floating point that can scale to integer or two separate ones?
- separate FP16/8 and INT data paths to save up to 2x energy for 16% larger area
- they have two data paths an FP8/FP16 (8 parallel FP16 or 16 parallel FP16), and an INT4/INT2 datapath (64 or 128 parallel INT4/INT2)

Summary: Design Considerations for Reduced Precision

- **Impact on accuracy**
  - Must consider the difficulty of the dataset, task, and DNN model
    - e.g., Easy to reduce precision for an easy task (e.g., digit classification); does the method work for a more difficult task?
- **Does hardware cost exceed benefits?**
  - Need extra hardware to support variable precision
    1. e.g., additional shift-and-add logic and registers for variable precision
    2. e.g., separate data path increases area but reduces energy (IBM)
  - Granularity impacts hardware overhead as well as accuracy
    1. e.g., More overhead to support (1b, 2b, 3b, 16b) than (2b, 4b, 8b, 16b)
- **Evaluation**
  - Use 8-bit for inference and 16-bit float for training for baseline
  - 32-bit float is a weak baseline