ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ICECS 2023
2023 30th IEEE International Conference on Electronics, Circuits and Systems (ICECS)
4-7 December 2023 - Hilton Maslak İstanbul / Türkiye
TECHNOSAPIENS FOR SAVING HUMANITY

APROPOS

MARIE CURIE ACTIONS

Optimizing Self-Organizing Maps for Bacterial Genome Identification on Parallel Ultra-Low-Power Platforms

EEES Laboratory (University of Bologna)

Seyed Ahmad Mirsalari seyedahmad.mirsalar2@unibo.it

@pulp_platform

pulp-platform.org

youtube.com/pulp_platform

PULP Platform

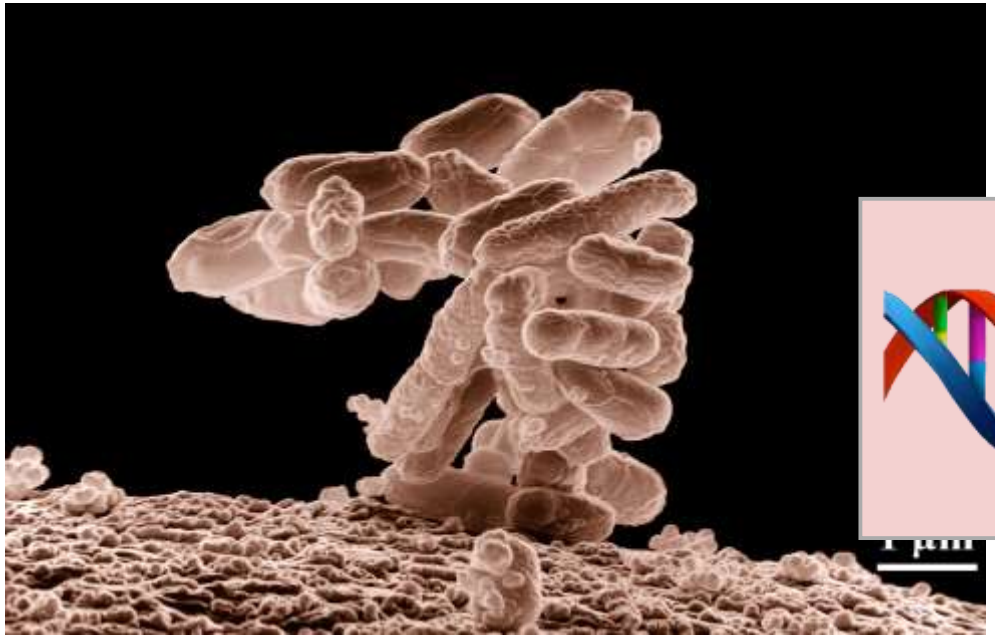
Open Source Hardware, the way it should be!

Outline

- Introduction
- State-of-the-art
- Background
- Methodology
- Experimental Results
- Conclusion

Whole-genome sequence (WGS)

- Comprehensive method for analyzing entire genomes.



Input: DNA sample extracted from the organism

Output: Comprehensive genomic dataset

Bacteria Identification

Use cases

- Diagnosing diseases.
- Improving the safety of drug treatments.

Challenges

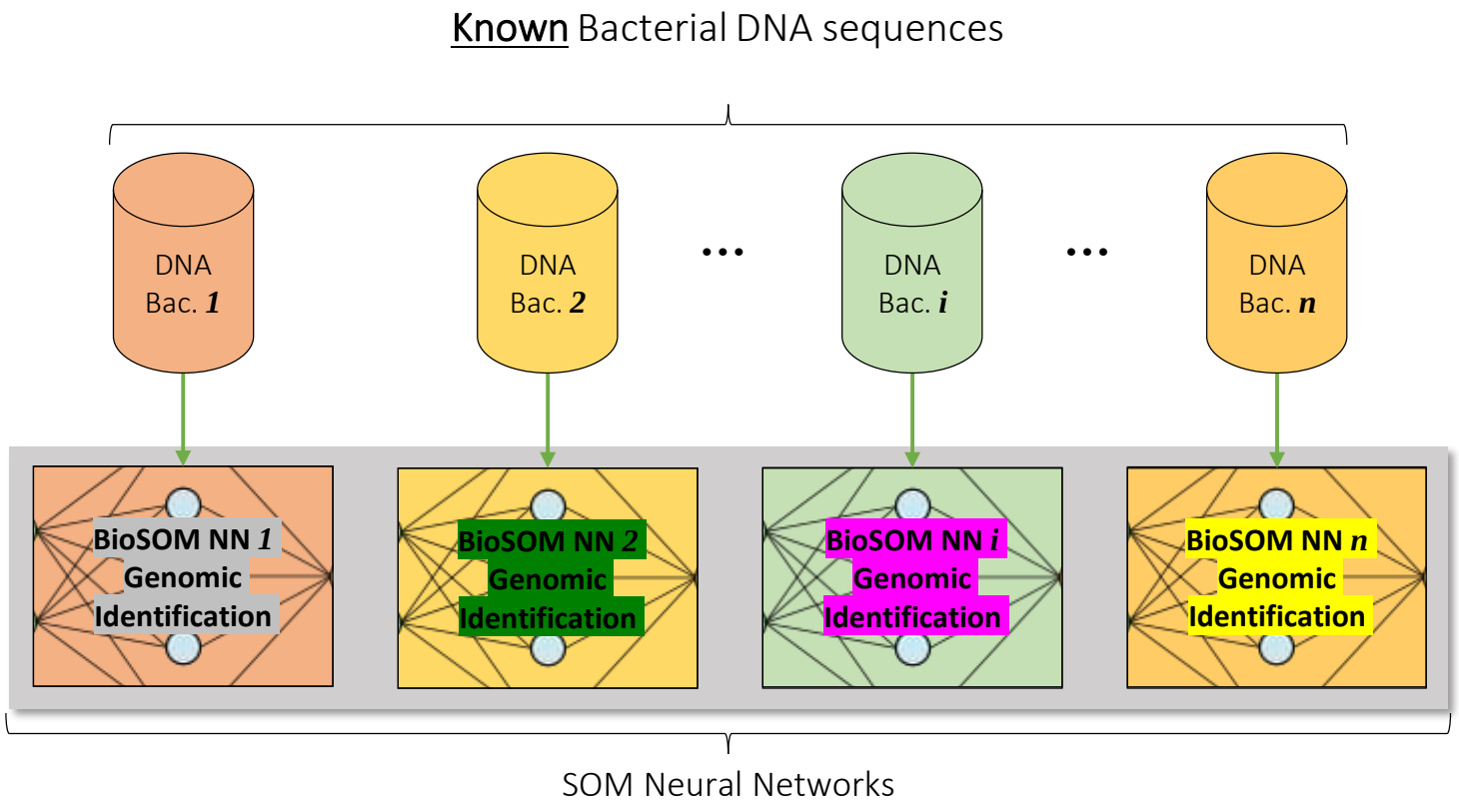
- Time consuming
- Compute intensive

Goal

- Proposing an efficient and low-cost genomics computations for pathogen identification on embedded systems.

RiBoSOM: rapid bacterial genome identification using SOM

- Self-Organizing-Map for bacterial genome identification



Y. Yang et al., "RiBoSOM: rapid bacterial genome identification using self-organizing map implemented on the synchoros SiLago platform," Proc. of the 18th Int. Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, 2018

RiBoSOM: rapid bacterial genome identification using SOM

Algorithm 1: SOM learning and inference for genome identification

Algorithm Part 1: SOM training for 1 bacterial genome;

Input N: Number of neurons;
Input $I = [i_1, i_2, ..., i_M]$: Input-Vector;
Input $IS = I_1, I_2, ..., I_S$: Sequence of Input-Vectors, Each IS represents one bacterial genome;
Input $W_{i,j} = M \times N$: Weight Matrix for 1 bacteria;
$\beta_{min} = 0.01$; $decay\_factor = 0.99$; $\beta = 1.0$;
for $I_k \in IS$ **do**
  $dist_{min} = \min_{j=1...N}(\sum_{i=1}^{M} |I_{k,i} - W_{i,j}|)$;
  $j_{min} = j$ where $dist_j = dist_{min}$;
  for $j \in 1...N$ **do**
    $dist = \frac{N}{2} - ||j - j_{min}| - \frac{N}{2}|$; /* toroid distance */
    $W_j = W_j - \frac{\beta}{2^{dist}}(W_j - I_k)$;
  end
  $\beta = \max(\beta * decay\_factor, \beta_{min})$; /* decay β */
end

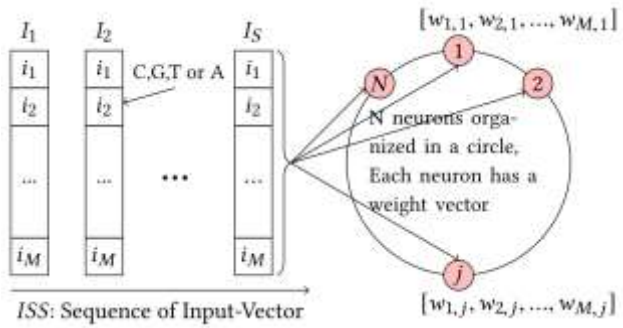
Algorithm Part 2: SOM inference;
Input $TIS = I_1, I_2, ..., I_S$: Test Input Sequence of Input-Vectors for which the bacteria is to be identified;
Input $W_{r,j,i} = R \times N \times M$: Weights for R bacteria;
$Inferred\_r$ is r with $score = \min_{r=1...R}[\sum_{k=1}^{S} \min_{j=1...N}(\sum_{i=1}^{M} |I_{k,j} - W_{r,i,j}|)]$;

Repeat for all input-vectors

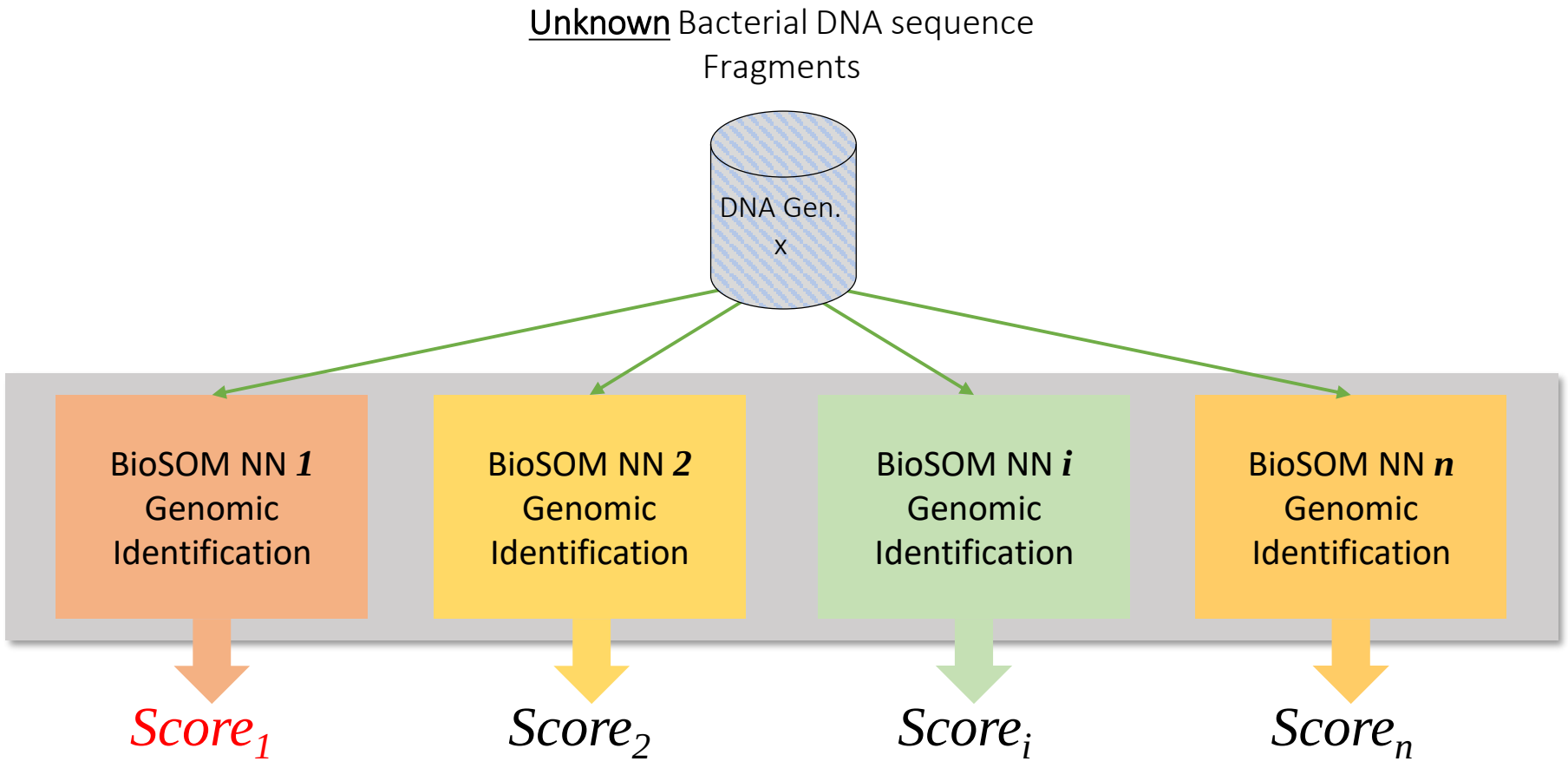
Calculate the difference of each neuron from the input

Find the "Best" neuron

Update the weights of each neuron depending on its distance from the "Best"

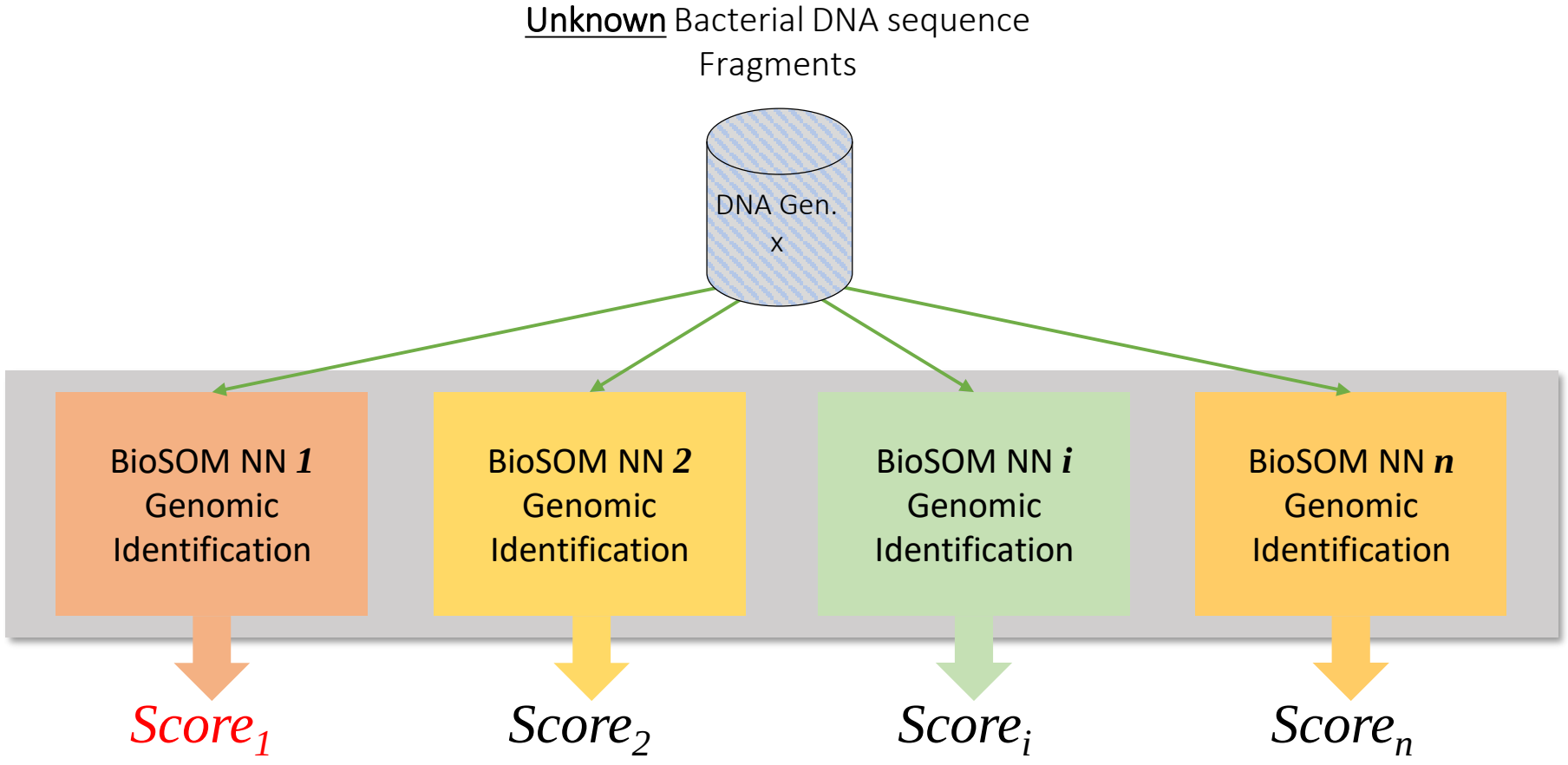


Inference for genomic identification of unknown bacteria



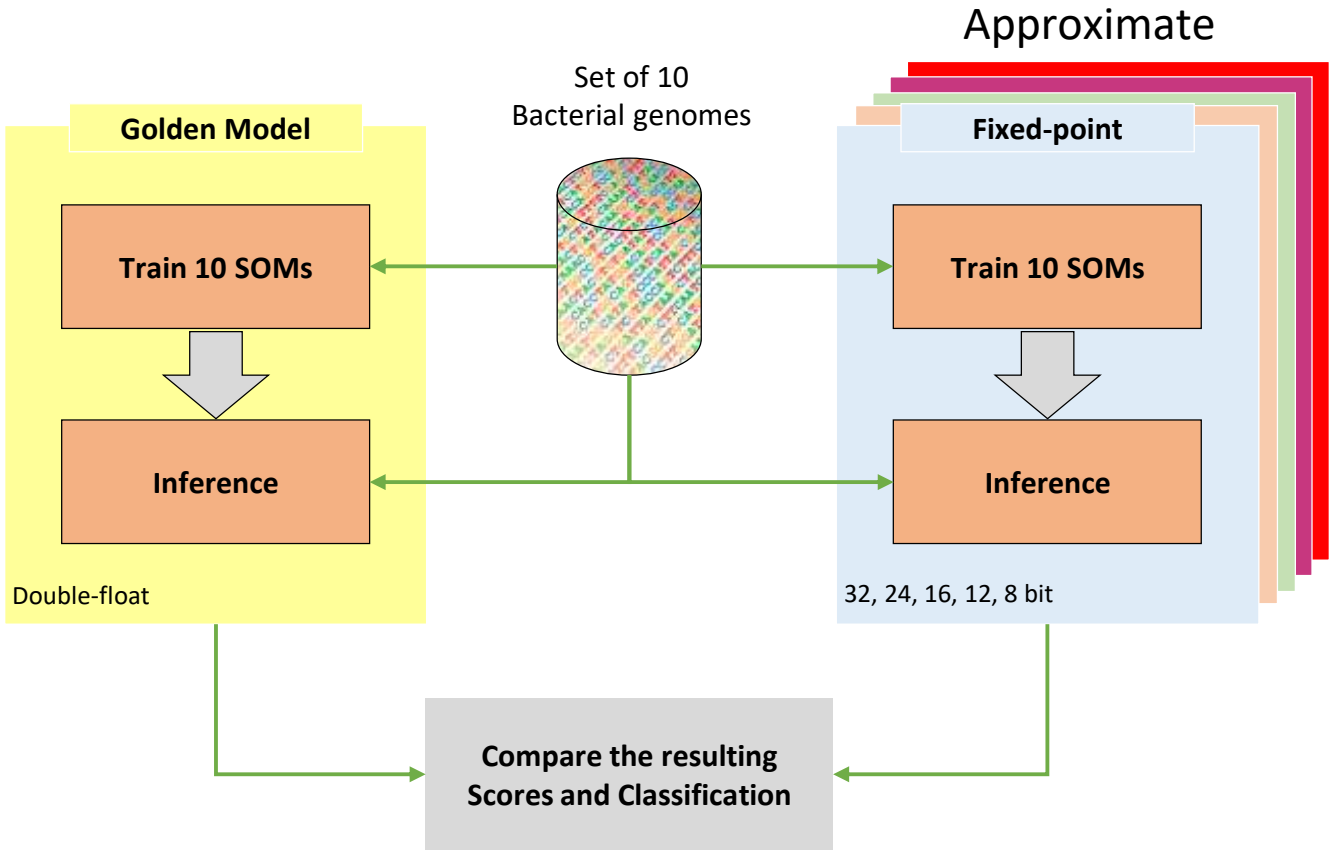
Smallest, or best, or closest score identifies the wining SOM and the matching bacterial genome

Inference for genomic identification of unknown bacteria



How much can we quantize the network?

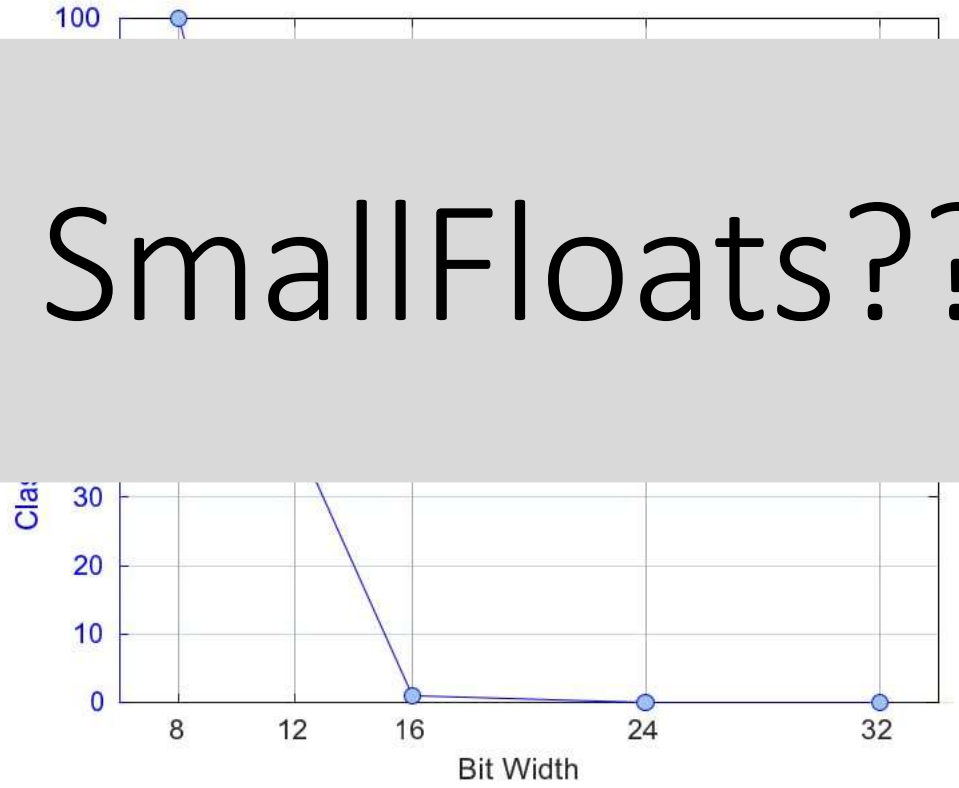
Approximate Computing Applied to Bacterial Genome Identification



D. Stathis *et al.*, "Approximate Computing Applied to Bacterial Genome Identification using Self-Organizing Maps," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 560–567.

Approximate Computing Applied to Bacterial Genome Identification

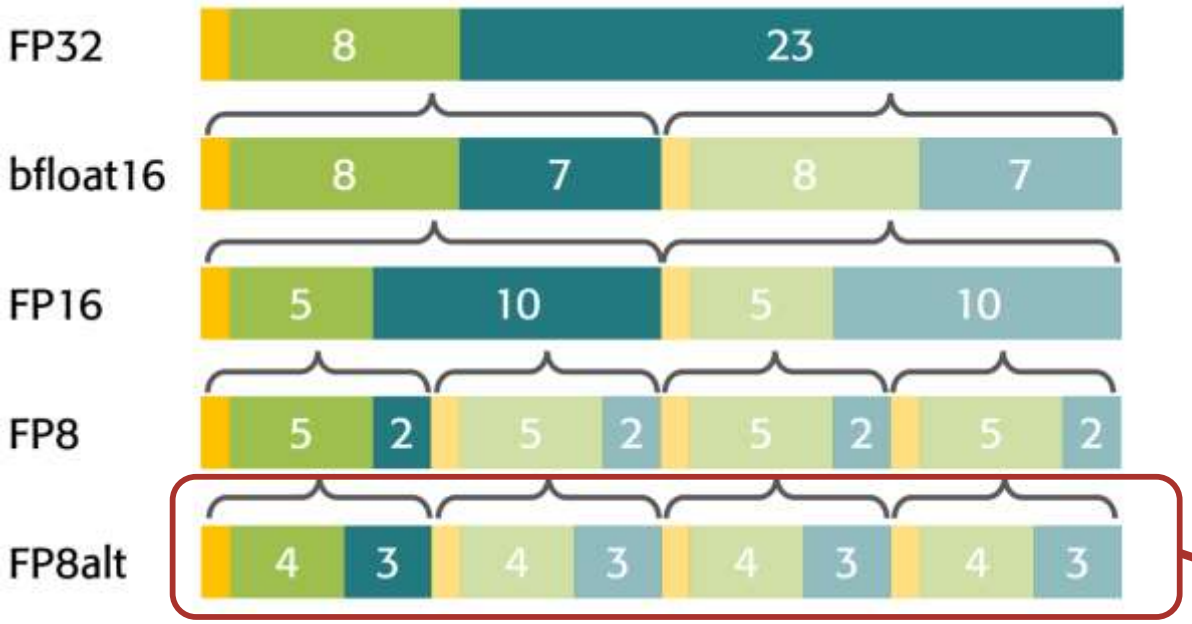
- Classification accuracy using 10 DNAs



SmallFloats???

Classification fails in 8 to 12-bit resolution region

SmallFloats



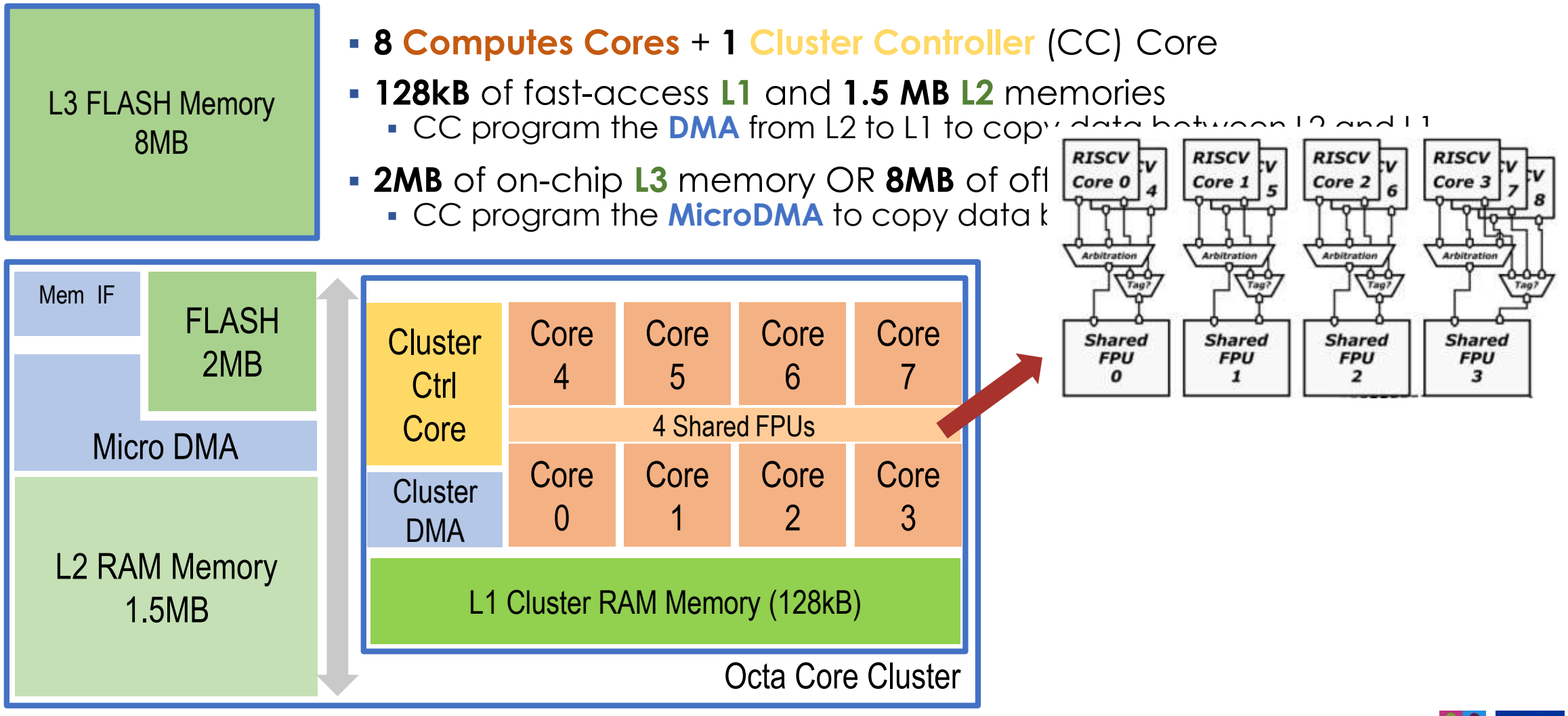
Format	# Representable values	Maximum Value
FP32	4.29×10^9	$\approx 3.40 \times 10^{38}$
Bfloat16	65536	$\approx 3.40 \times 10^{38}$
FP16	65536	≈ 65504
FP8	256	≈ 57344
FP8ALT	256	≈ 488

SIMD Vectorization

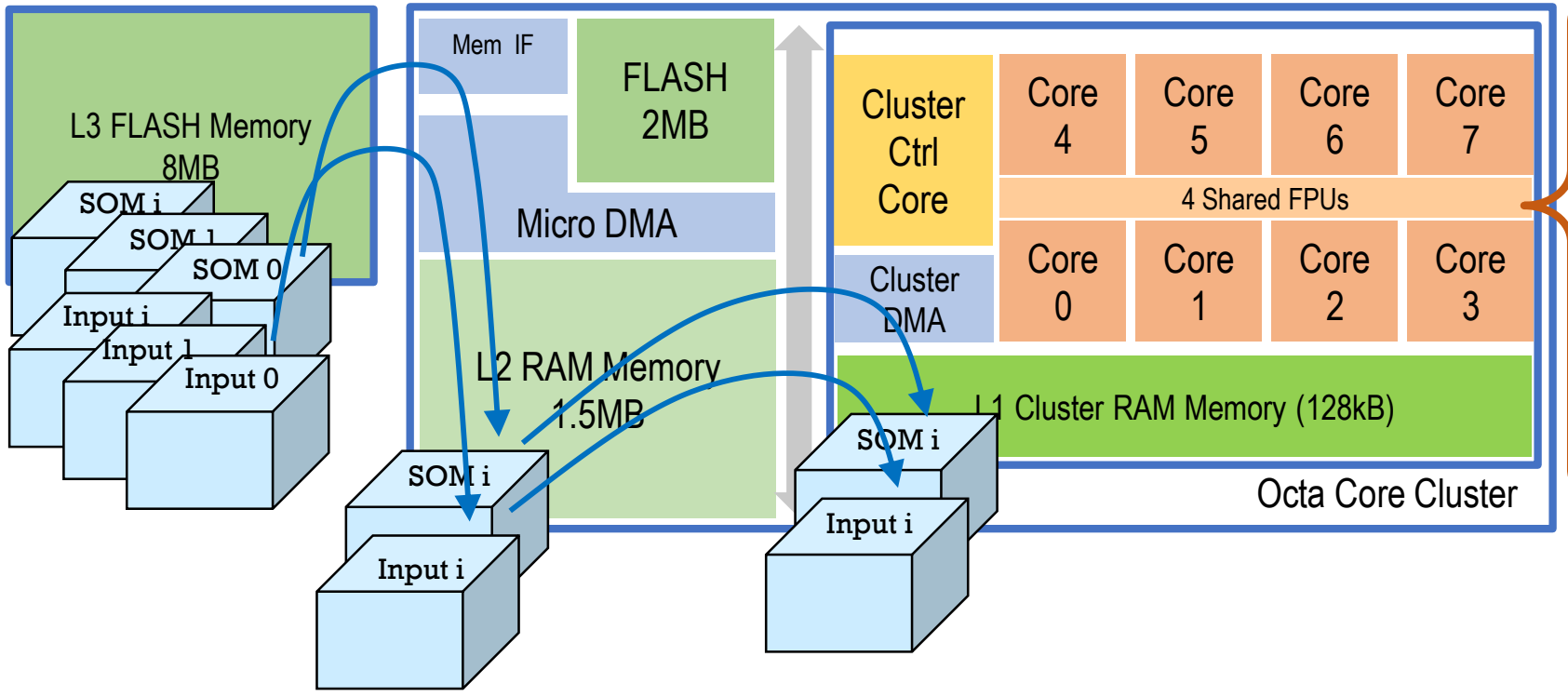
- Higher performance
- Higher energy efficiency
- Lower memory footprint
- Lower data movement energy

However, narrower formats produce lower-accuracy results

GreenWaves' GAP9 ultra-low power



Vertical Computation Mapping



Algorithm 1: SOM learning and inference for genome identification

Algorithm Part 1: SOM training for 1 bacterial genome;

Input N: Number of neurons;
Input $I = [i_1, i_2, ..., i_M]$: Input-Vector;
Input $IS = I_1, I_2, ..., I_S$: Sequence of Input-Vectors, Each IS represents one bacterial genome;
Input $W_{i,j} = M \times N$: Weight Matrix for 1 bacteria;
$\beta_{min} = 0.01$; $decay\_factor = 0.99$; $\beta = 1.0$;
for $I_k \in IS$ **do**
  $dist_{min} = \min_{j=1...N}(\sum_{i=1}^{M} |I_{k,i} - W_{i,j}|)$;
  $j_{min} = j$ where $dist_j = dist_{min}$;
  for $j \in 1...N$ **do**
    $dist = \frac{N}{2} - ||j - j_{min}| - \frac{N}{2}|$; /* toroid distance */
    $W_j = W_j - \frac{\beta}{2^{dist}}(W_j - I_k)$;
  end
  $\beta = \max(\beta * decay\_factor, \beta_{min})$; /* decay β */
end

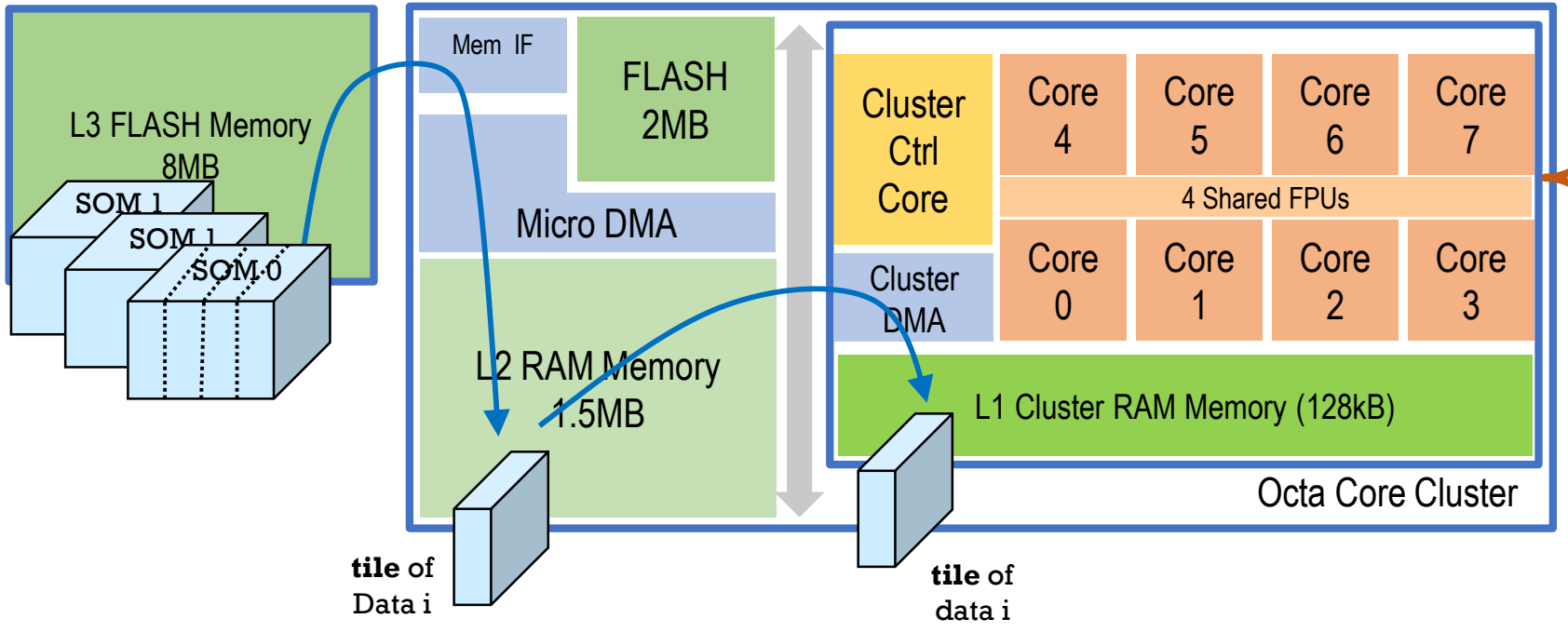
Algorithm Part 2: SOM inference;

Input $TIS = I_1, I_2, ..., I_S$: Test Input Sequence of Input-Vectors for which the bacteria is to be identified;
Input $W_{r,j,i} = R \times N \times M$: Weights for R bacteria;
$Inferred\_r$ is r with $score = \min_{r=1...R}[\sum_{k=1}^{S} \min_{j=1...N}(\sum_{i=1}^{M} |I_{k,j} - W_{r,i,j}|)]$;

- Model Coefficient stored in non-volatile FLASH memory (on-chip, if fitting)
- Execution SOM-by-SOM
  - Data copied from L3 to L1
  - Parallel Execution on 8 cores

Problem: data may not fit L1 memory

Tiling



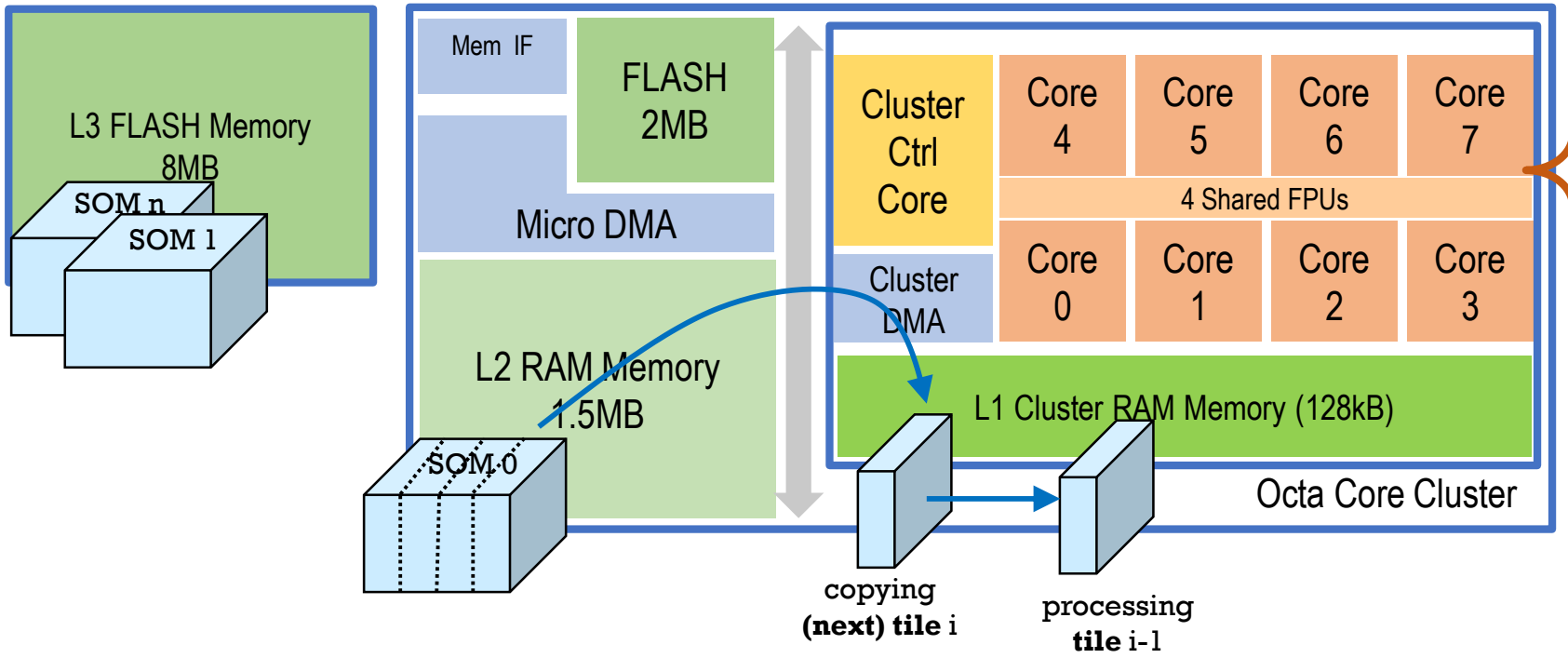
Algorithm 1: SOM learning and inference for genome identification

Algorithm Part 1: SOM training for 1 bacterial genome;
Input N: Number of neurons;
Input $I = [i_1, i_2, ..., i_M]$: Input-Vector;
Input $IS = I_1, I_2, ..., I_S$: Sequence of Input-Vectors, Each IS represents one bacterial genome;
Input $W_{i,j} = M \times N$: Weight Matrix for 1 bacteria;
$\beta_{min} = 0.01$; $decay\_factor = 0.99$; $\beta = 1.0$;
for $I_k \in IS$ **do**
  $dist_{min} = \min_{j=1...N}(\sum_{i=1}^{M} |I_{k,i} - W_{i,j}|)$;
  $j_{min} = j$ where $dist_j = dist_{min}$;
  for $j \in 1...N$ **do**
    $dist = \frac{N}{2} - ||j - j_{min}| - \frac{N}{2}|$; /* toroid distance */
    $W_j = W_j - \frac{\beta}{2^{dist}}(W_j - I_k)$;
  end
  $\beta = \max(\beta * decay\_factor, \beta_{min})$; /* decay β */
end

Algorithm Part 2: SOM inference;
Input $TIS = I_1, I_2, ..., I_S$: Test Input Sequence of Input-Vectors for which the bacteria is to be identified;
Input $W_{r,j,i} = R \times N \times M$: Weights for R bacteria;
$Inferred\_r$ is r with $score = \min_{r=1...R}[\sum_{k=1}^{S} \min_{j=1...N}(\sum_{i=1}^{M} |I_{k,j} - W_{r,i,j}|)]$;

Need to split a weight tensor into N_{tile} sub-tensors: **tiles**

Double buffering



Algorithm 1: SOM learning and inference for genome identification

Algorithm Part 1: SOM training for 1 bacterial genome;

Input N: Number of neurons;
Input $I = [i_1, i_2, ..., i_M]$: Input-Vector;
Input $IS = I_1, I_2, ..., I_S$: Sequence of Input-Vectors, Each IS represents one bacterial genome;
Input $W_{i,j} = M \times N$: Weight Matrix for 1 bacteria;
$\beta_{min} = 0.01$; $decay\_factor = 0.99$; $\beta = 1.0$;
for $I_k \in IS$ **do**
  $dist_{min} = \min_{j=1...N}(\sum_{i=1}^{M} |I_{k,i} - W_{i,j}|)$;
  $j_{min} = j$ where $dist_j = dist_{min}$;
  for $j \in 1...N$ **do**
    $dist = \frac{N}{2} - ||j - j_{min}| - \frac{N}{2}|$; /* toroid distance */
    $W_j = W_j - \frac{\beta}{2^{dist}}(W_j - I_k)$;
  end
  $\beta = \max(\beta * decay\_factor, \beta_{min})$; /* decay β */
end

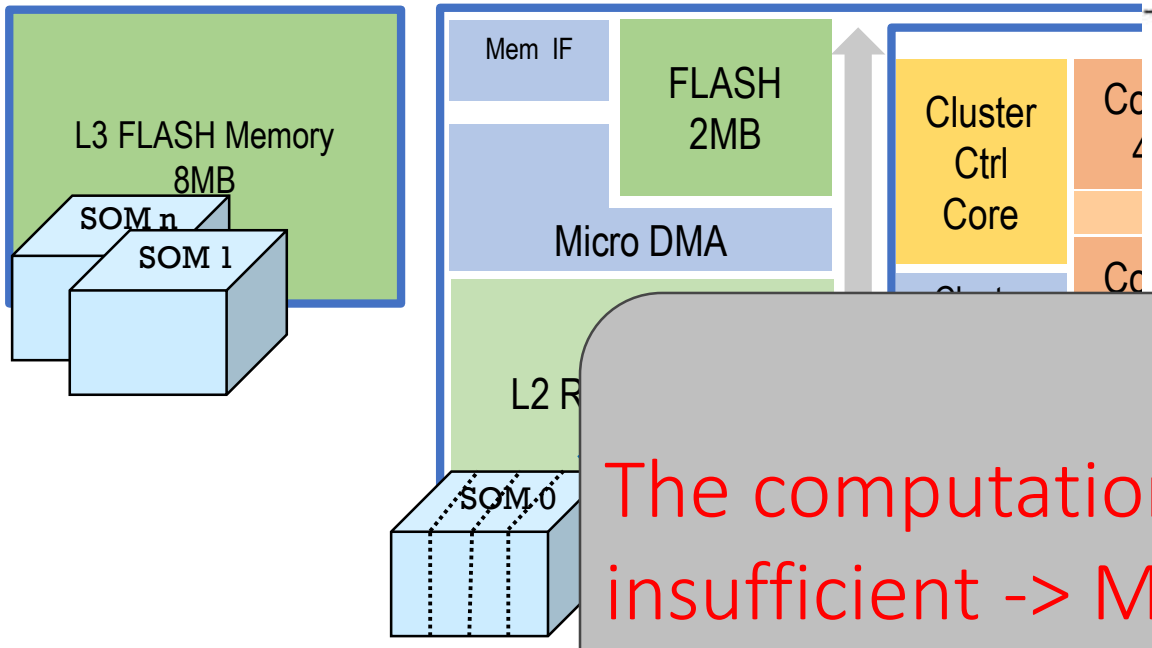
Algorithm Part 2: SOM inference;

Input $TIS = I_1, I_2, ..., I_S$: Test Input Sequence of Input-Vectors for which the bacteria is to be identified;
Input $W_{r,j,i} = R \times N \times M$: Weights for R bacteria;
$Inferred\_r$ is r with $score = \min_{r=1...R}[\sum_{k=1}^{S} \min_{j=1...N}(\sum_{i=1}^{M} |I_{k,j} - W_{r,i,j}|)]$;

Cluster Controller (CC) core interleaves memory transfers and compute tasks

Memory copies and computation happens **concurrently**

Vertical Computation Mapping



Algorithm 1: SOM learning and inference for genome identification

Algorithm Part 1: SOM training for 1 bacterial genome;

1 **Input** N: Number of neurons;
2 **Input** $I = [i_1, i_2, ..., i_M]$: Input-Vector;
3 **Input** $IS = I_1, I_2, ..., I_S$: Sequence of Input-Vectors, Each IS represents one bacterial genome;
4 **Input** $W_{i,j} = M \times N$: Weight Matrix for 1 bacteria;
5 $\beta_{min} = 0.01$; $decay\_factor = 0.99$; $\beta = 1.0$;

Parallelize(N)

Parallelize(N)

uals to 0 if dist > threshold
(e.g., dist>19 if dt=float32)
19+1+19

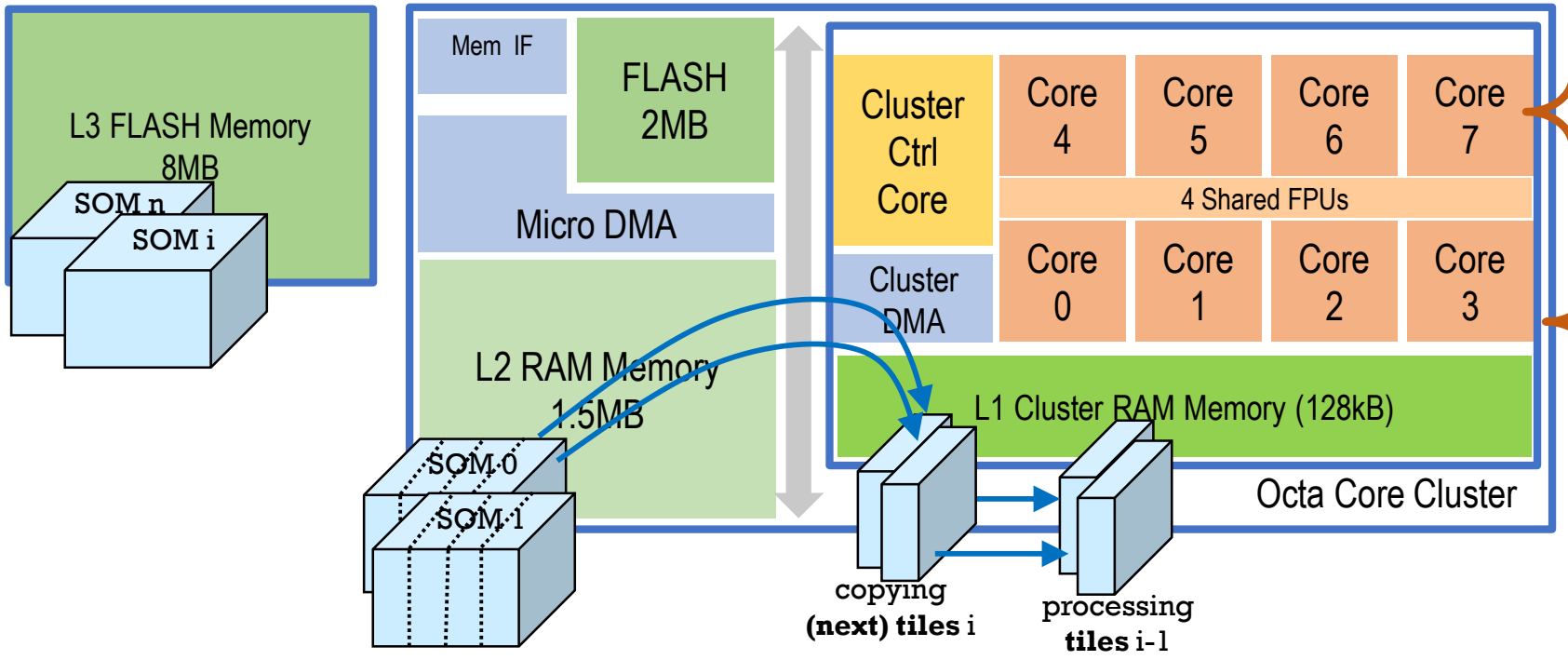
17 **Input** $TIS = I_1, I_2, ..., I_S$: Test Input Sequence of Input-Vectors for which the bacteria is to be identified;
18 **Input** $W_{r,j,i} = R \times N \times M$: Weights for R bacteria;
19 $Inferred\_r$ is r with $score = \min_{r=1...R}[\sum_{k=1}^{S} \min_{j=1...N}(\sum_{i=1}^{M} |I_{k,j} - W_{r,i,j}|)]$;

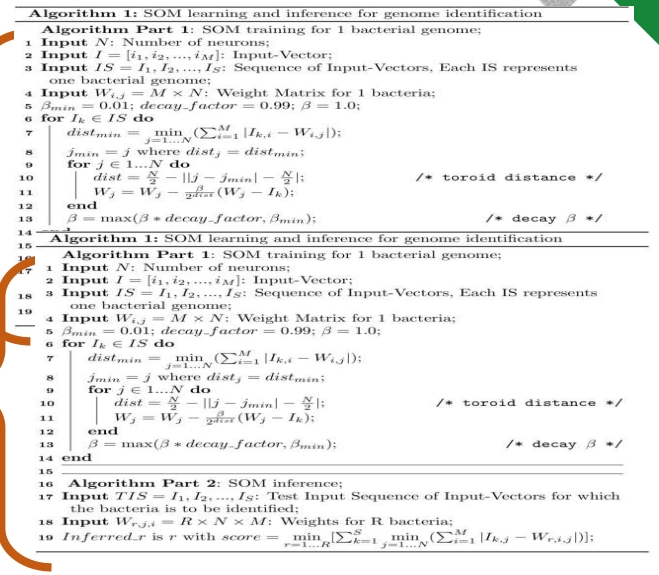
The computation workload is insufficient -> Multi-core

CC core interleaves transfers and compute tasks

Memory copies and computation happens **concurrently**

Horizontal Computation Mapping





Algorithm 1: SOM learning and inference for genome identification

Algorithm Part 1: SOM training for 1 bacterial genome;
Input N: Number of neurons;
Input $I = [i_1, i_2, ..., i_M]$: Input-Vector;
Input $IS = I_1, I_2, ..., I_S$: Sequence of Input-Vectors, Each IS represents one bacterial genome;
Input $W_{i,j} = M \times N$: Weight Matrix for 1 bacteria;
$\beta_{min} = 0.01$; $decay\_factor = 0.99$; $\beta = 1.0$;
for $I_k \in IS$ **do**
  $dist_{min} = \min_{j=1...N}(\sum_{i=1}^{M} |I_{k,i} - W_{i,j}|)$;
  $j_{min} = j$ where $dist_j = dist_{min}$;
  for $j \in 1...N$ **do**
    $dist = \frac{N}{2} - ||j - j_{min}| - \frac{N}{2}|$; /* toroid distance */
    $W_j = W_j - \frac{\beta}{2^{dist}}(W_j - I_k)$;
  end
  $\beta = \max(\beta * decay\_factor, \beta_{min})$; /* decay β */
end

Algorithm 1: SOM learning and inference for genome identification

Algorithm Part 1: SOM training for 1 bacterial genome;
Input N: Number of neurons;
Input $I = [i_1, i_2, ..., i_M]$: Input-Vector;
Input $IS = I_1, I_2, ..., I_S$: Sequence of Input-Vectors, Each IS represents one bacterial genome;
Input $W_{i,j} = M \times N$: Weight Matrix for 1 bacteria;
$\beta_{min} = 0.01$; $decay\_factor = 0.99$; $\beta = 1.0$;
for $I_k \in IS$ **do**
  $dist_{min} = \min_{j=1...N}(\sum_{i=1}^{M} |I_{k,i} - W_{i,j}|)$;
  $j_{min} = j$ where $dist_j = dist_{min}$;
  for $j \in 1...N$ **do**
    $dist = \frac{N}{2} - ||j - j_{min}| - \frac{N}{2}|$; /* toroid distance */
    $W_j = W_j - \frac{\beta}{2^{dist}}(W_j - I_k)$;
  end
  $\beta = \max(\beta * decay\_factor, \beta_{min})$; /* decay β */
end

Algorithm Part 2: SOM inference;
Input $TIS = I_1, I_2, ..., I_S$: Test Input Sequence of Input-Vectors for which the bacteria is to be identified;
Input $W_{r,j,i} = R \times N \times M$: Weights for R bacteria;
$Inferred\_r$ is r with $score = \min_{r=1...R}[\sum_{k=1}^{S} \min_{j=1...N}(\sum_{i=1}^{M} |I_{k,j} - W_{r,i,j}|)]$;

The resources are split among networks

Experimental Setup

- Golden Model in PyTorch
  - Emulate the behavior of hardware
    - MAC flag: Fused Multiply-Add
    - Vec Flag: Supporting **SmallFloats** (e.g., **float16, bfloat16, float8**)
  - Adding FP8|e5m2 to the PyTorch data types(**Open-Source**)
  - Classification error represents the ratio between the false classifications over the total number of tests: $Classification\ error = \frac{C_{false}}{C_{total}} \times 100\%$
- 10 different BioSOM networks were trained with **128**, **256**, **512**, and **1024** neurons with **20K** training vectors for each SOM
- Test the model with 1000 test vectors for all 10 classes

Experimental Setup

- Golden Model in PyTorch
  - Emulate the behavior of hardware
  - Adding FP8|e5m2 to the PyTorch data types
- 10 different BioSOM networks were trained with **128**, **256**, **512**, and **1024** neurons with **20K** training vectors for each SOM
- Test the model with 1000 test vectors for all 10 classes
- For hardware results
  - We trained two different SOMs with 20K training vectors for each SOM
  - Execute on the GAP9 board (supporting float32, float16, and bfloat16)
  - FP8 from cycle-accurate simulations and power numbers from post-physical-synthesis simulations in 22nm FDSOI

Weight Memory and classification error

Neurons	Weights per Neuron	SOMs
128	8	2
Weight Memory (Byte)		
	Weight Memory	Classification e
FP32	8192	0
FP16	4096	0
FP8	2048	<9%
FxP-8bit	2048	fails
FxP-16bit	4096	<1%
FxP-24bit	6144	
FxP-32bit	8192	

The classification error is less than 9% using float8 with 128 Neurons

Neurons	Weights per Neuron	
512	8	
Weight Memory (Byte)		
	Weight Memory	Classification error
FP32	32768	0
FP16	16384	0
FP8	8192	0
FxP-8bit	8192	fails(>96%)
FxP-16bit	16384	<1%
FxP-24bit	24576	0
FxP-32bit	32768	0

neurons

Float8 achieves 2x memory saving compared to 16-bit fixed-point and floating point

Neurons	Weights per Neuron	SOMs
256	8	2
Weight Memory (Byte)		
	Weight Memory	Classification erro
FP32	16384	0
FP16	8192	0
FP8	4096	0
FxP-8bit	4096	fails(>90%)
FxP-16bit	8192	<1%
		0
		0

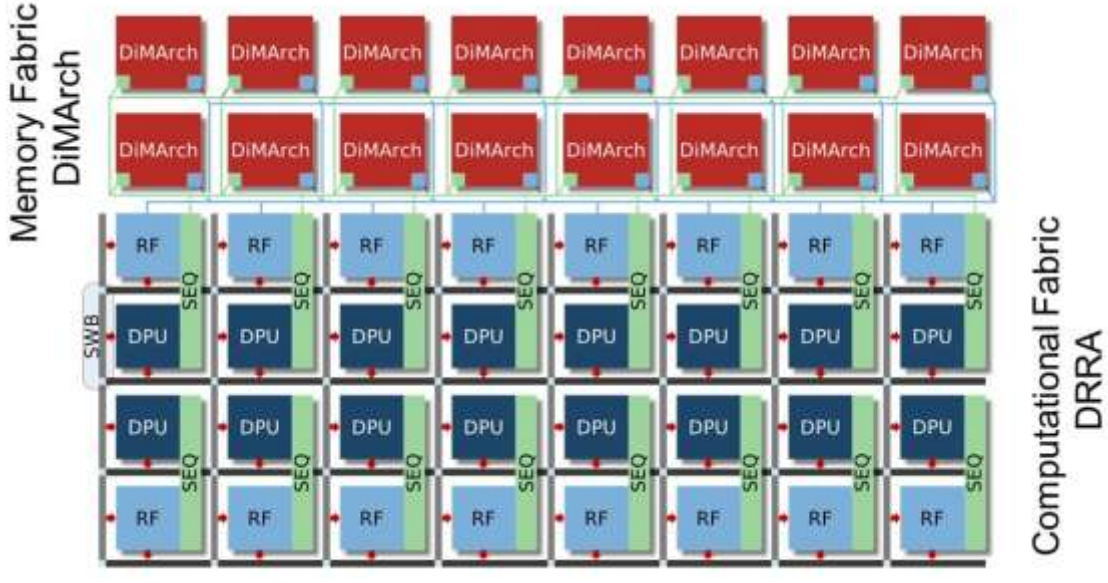
But with 256 Neurons, the classification error is 0%

8-bit fixed-point fails the classification with different number of neurons

uron	SOMs
	2
ory (Byte)	
Weight Memory	Classification error
65536	0
32768	0
16384	0
16384	fails
32768	<1%
49152	0
65536	0

Coarse Grain Reconfigurable Architecture

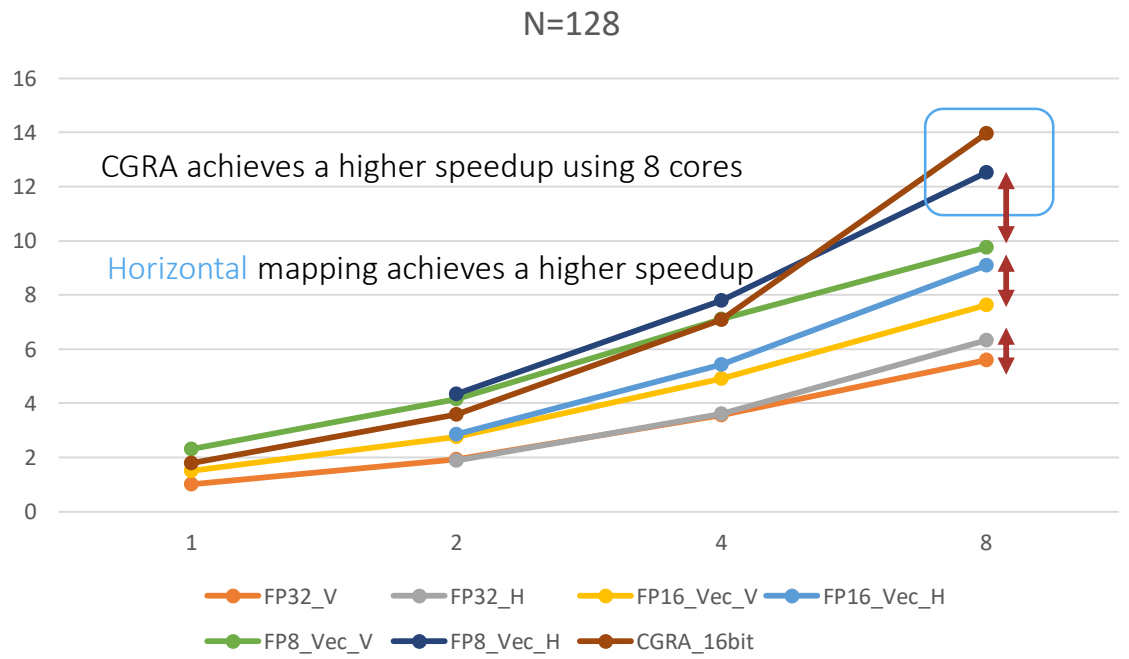
- Comparing our design with a fixed-point implementation on a Coarse Grain Reconfigurable Architecture (CGRA)
- Customized for dense linear algebra
- 8 columns



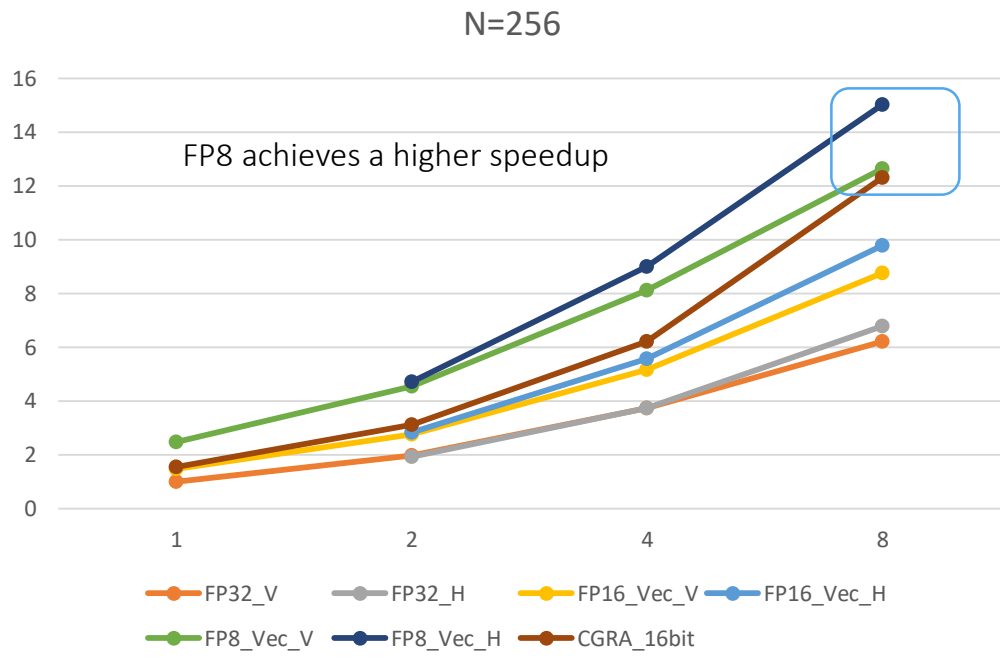
Y. Yang et al., "RiBoSOM: rapid bacterial genome identification using self-organizing map implemented on the synchoros SiLago platform," Proc. of the 18th Int. Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, 2018

Speedup

- This network can fit in SRAM in CGRA



Baseline cycles counts(FP32): 425,180,568

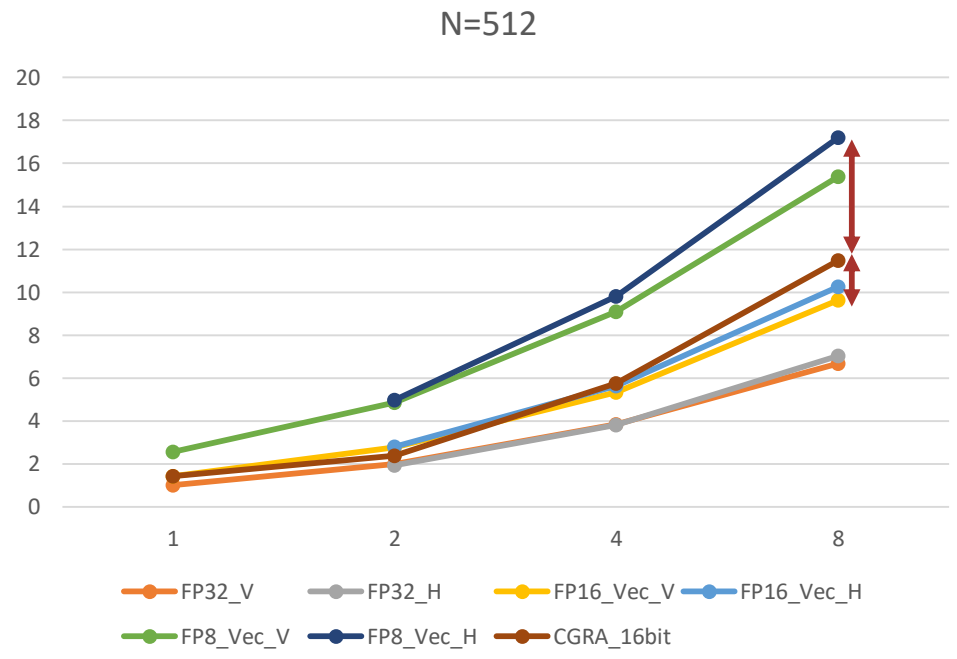


Baseline cycles counts(FP32): 738,577,694

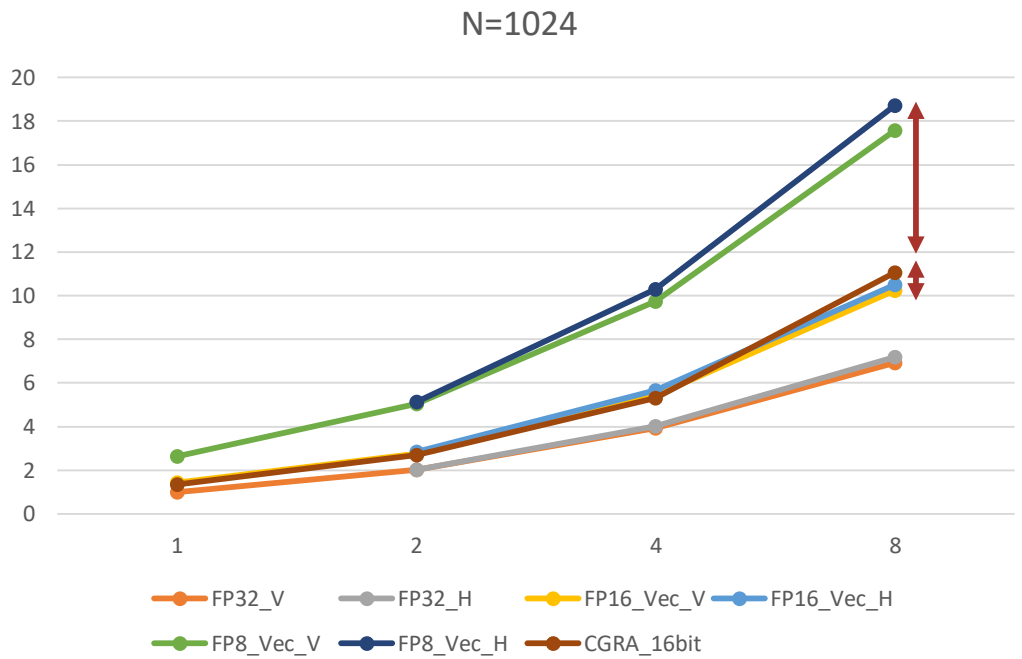
Speedup

- What happens when the network size increased?

FP16 gets closer to fxp16, and FP8 achieves a higher speedup



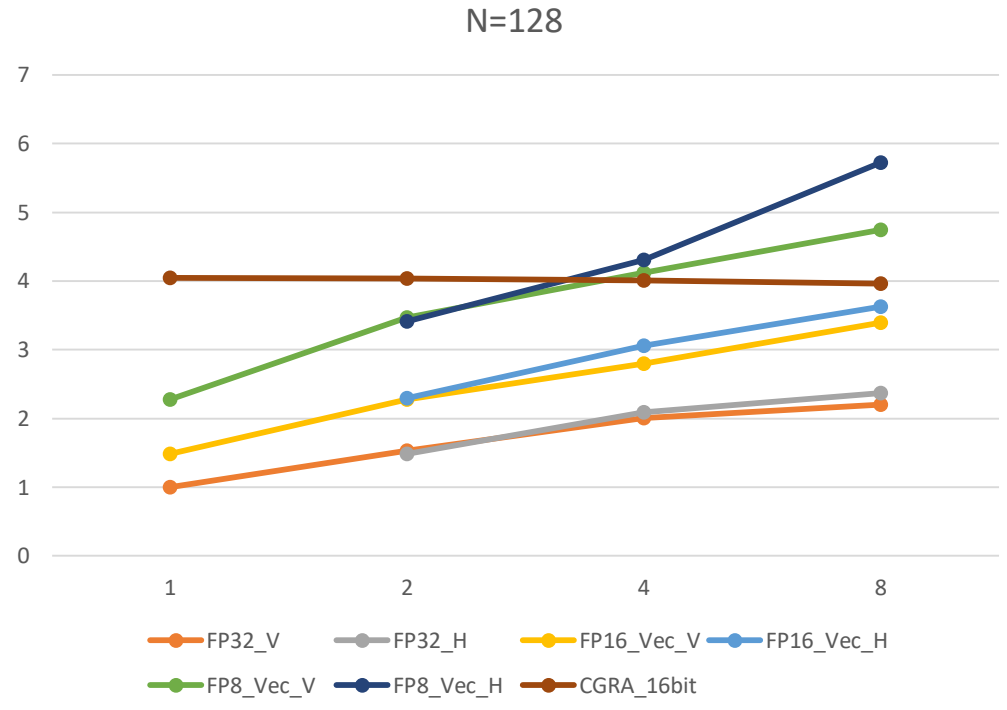
Baseline cycles counts(FP32): 1,364,078,880



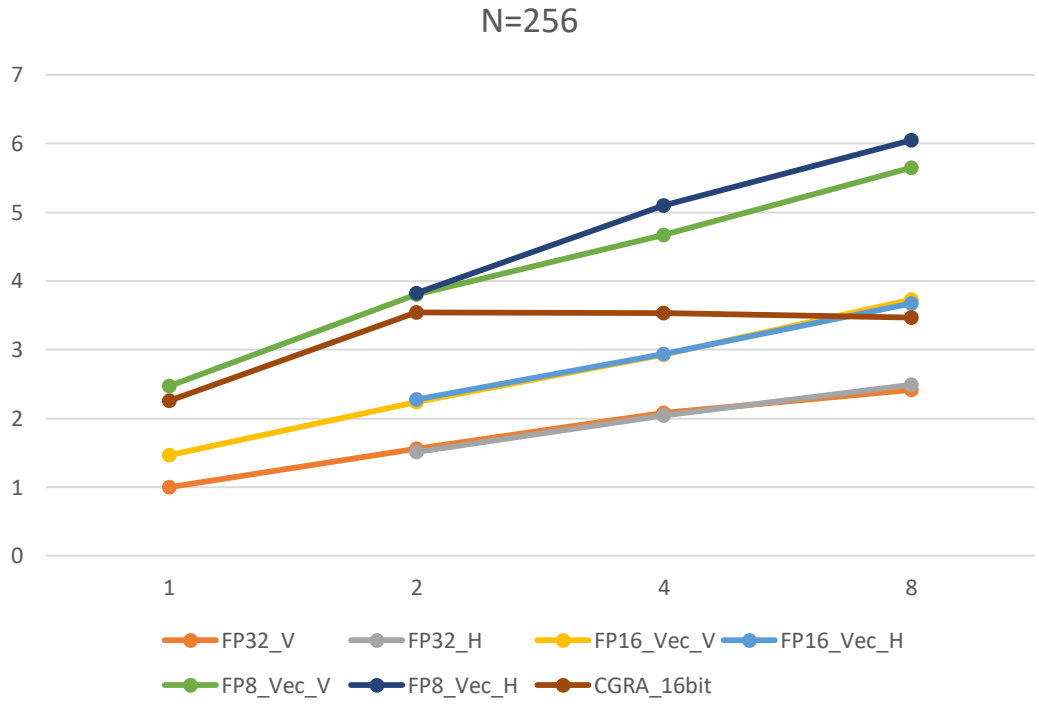
Baseline cycles counts(FP32): 2,613,552,914

Energy Improvement

- The energy efficiency improves by increasing the number of cores on GAP9



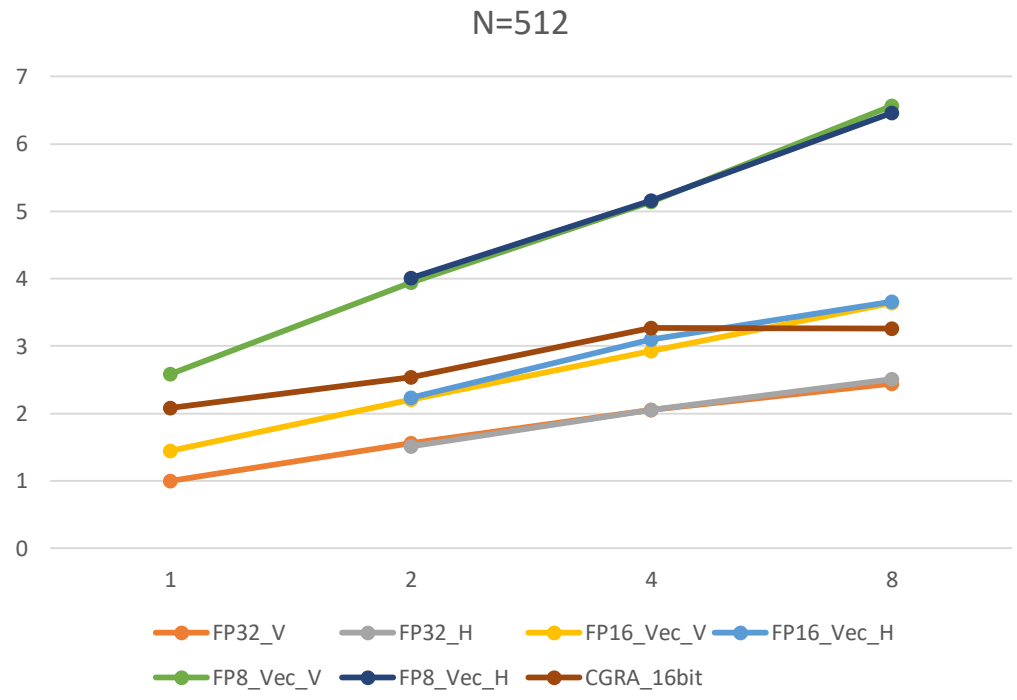
Baseline energy consumption(FP32) : 32.17 mJ



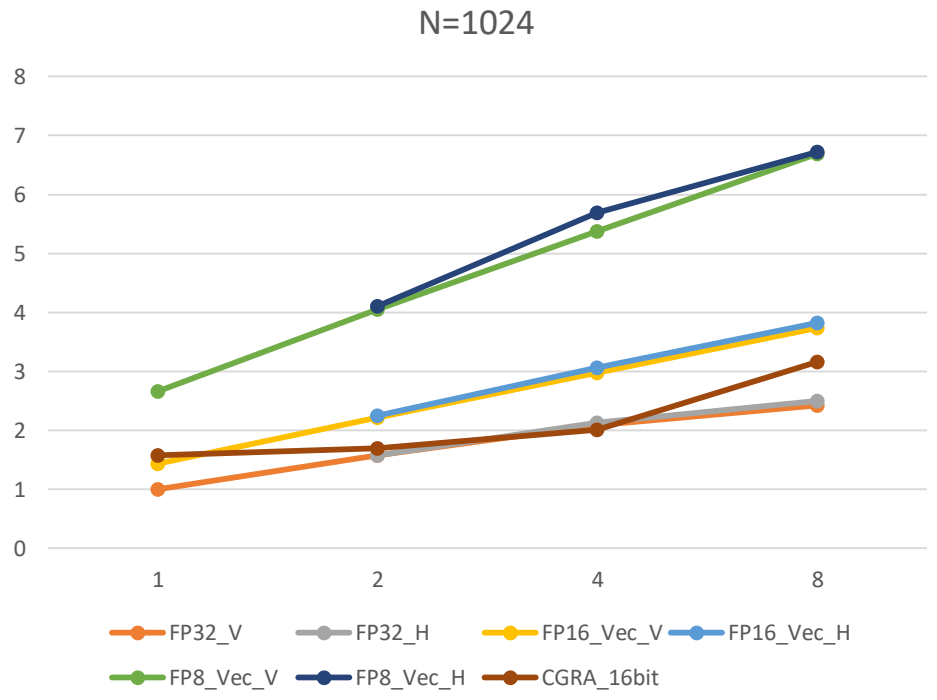
Baseline energy consumption(FP32) : 56.33 mJ

Energy Improvement

- FP16 on GAP9 achieves higher energy efficiency compared to CGRA the network size increased



Baseline energy consumption(FP32) : 104.05 mJ



Baseline energy consumption(FP32) : 200.75 mJ

Conclusions

- We address the challenges associated with performing genomics computations for genome identification on embedded systems with limited computational power
- We investigates the impact of smaller-than-32-bit floating-point formats (smallFloats) on energy savings, and performance
  - smallFloat formats exhibit a 100% classification accuracy
  - The parallel variants achieve a speed-up of 1.98×, 3.79×, and 6.83× on 2, 4, and 8 cores, Respectively.
  - Furthermore, FP8 vectorization increases the average speed-up by 2.5×.
- We compare our design with a fixed-point implementation on a coarse grain reconfigurable architecture:
  - FP8 implementation achieves, on average, 1.42× energy efficiency, 1.51× speedup, and a 50% reduction in memory footprint compared to CGRA.

More information

- Paper's GitHub: https://github.com/ahmad-mirsalari/SOM-on-PULP

SCAN ME

- PULP: https://github.com/ahmad-mirsalari/PULP

SCAN ME

APROPOS

http://apropos-itn.eu

This project has received funding from the European Union's Horizon 2020 (H2020) Marie Sklodowska-Curie Innovative Training Networks H2020-MSCA-ITN-2020 call, under the Grant Agreement no 956090.

Thank you!