# tinyML: the mWatts world

**Marco Fariselli**
Embedded ML Engineer
marco.fariselli@greenwaves-technologies.com

GREENWAVES TECHNOLOGIES

# GreenWaves Technologies

- Fabless semiconductor startup founded in 2014.
- We design and sell **extreme performance** processors for **energy constrained devices**
- 45 people, HQ in **Grenoble,** France
- Offices in **Bologna**, Italy, **Shanghai**, China, **Copenhagen**, Denmark. Global sales footprint.

embeddedaward2023
winner: Hardware

Best hardware product
Embedded World 2023

Assises de l'Embarqué

Embedded France
L'EMBARQUÉ MADE IN FRANCE

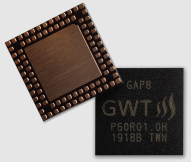
Embedded Technologies Award 2023
Les Assises de l'Embarqué

Gartner
COOL VENDOR 2019

Cool Vendors in AI Semiconductors, Alan Priestley, Saniye Alaybeyi, April 29, 2019.

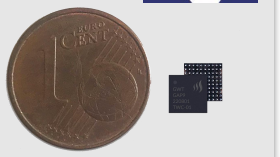
## GAP8

In production since 2020 one of the very first commercially available RISC-V processor and AI microcontroller




## GAP9

Second generation ultra-low power AI and DSP enabled Microcontroller

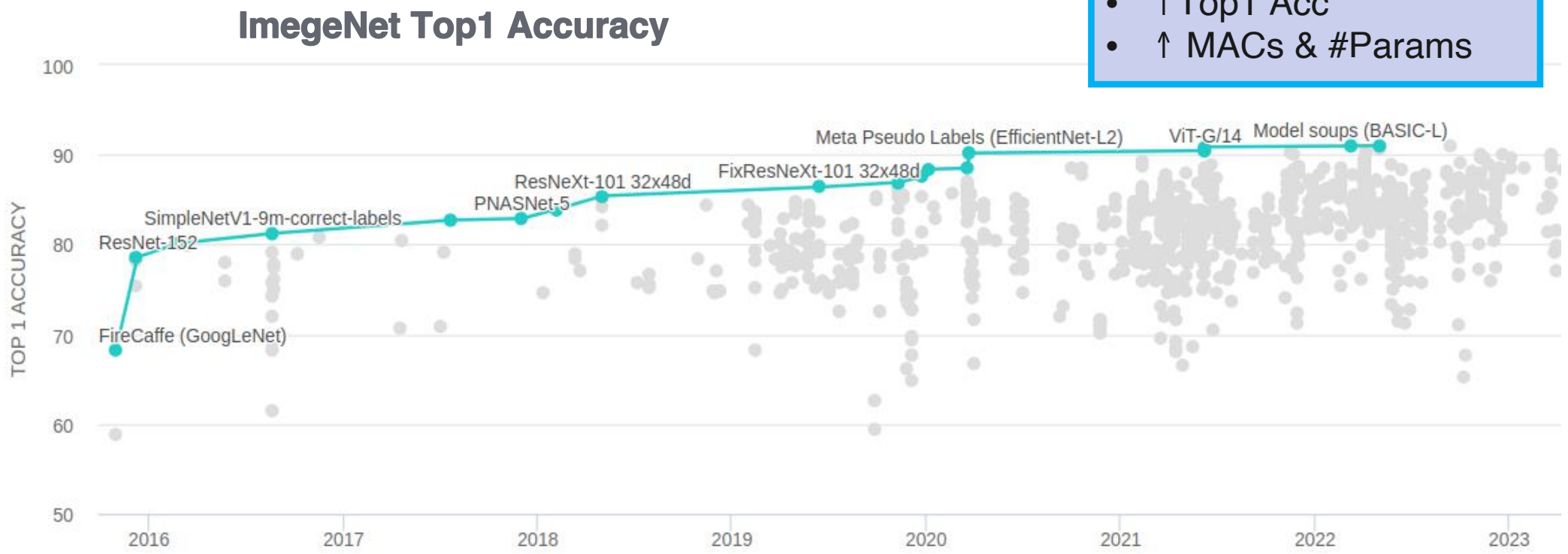


GREENWAVES TECHNOLOGIES

# What is a “tinyML” workload?

***“Getting Better” trend***

- ↑Top1 Acc
- ↑ MACs & #Params



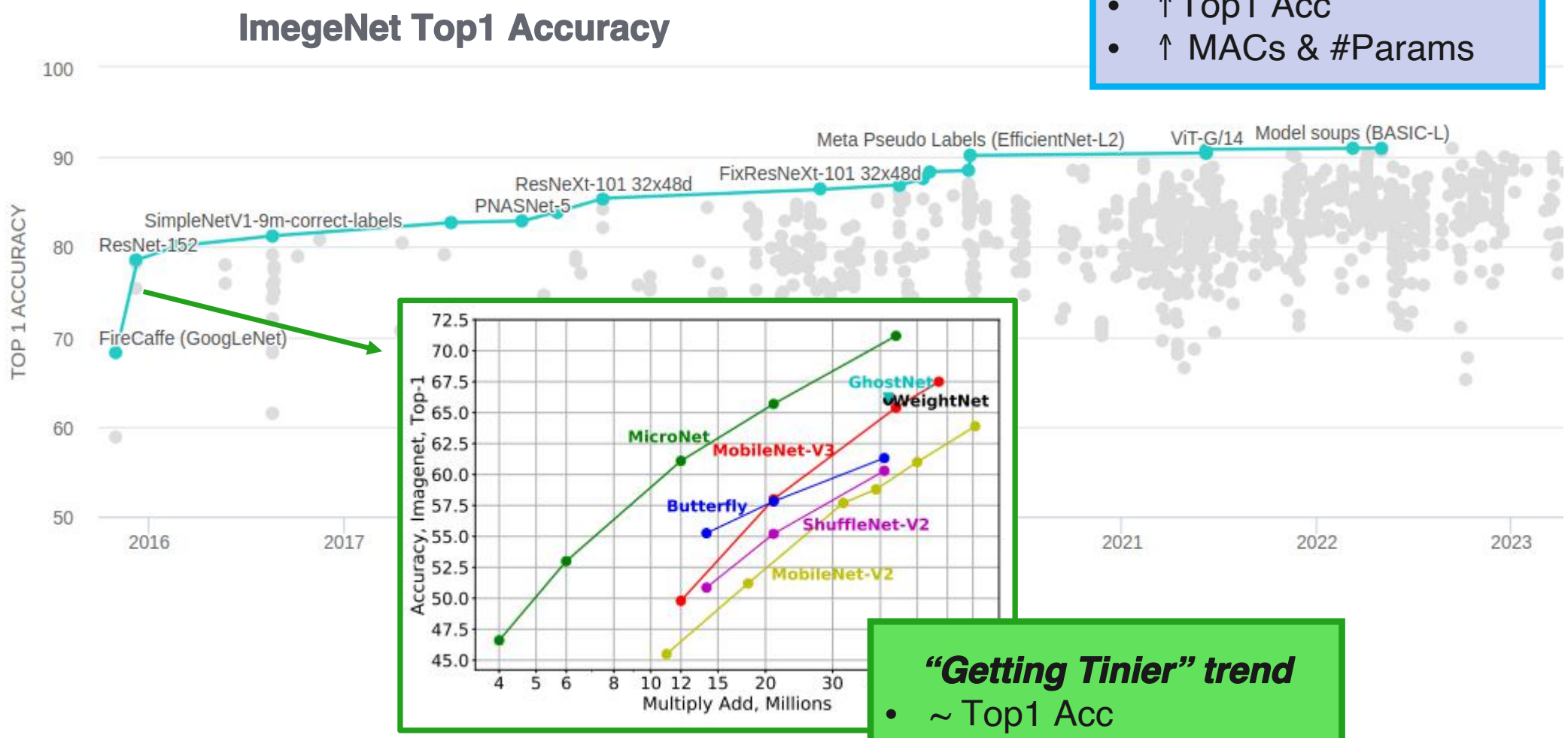





GREENWAVES TECHNOLOGIES

# What is a “tinyML” workload?

**“Getting Better” trend**
- ↑Top1 Acc
- ↑ MACs & #Params




**“Getting Tinier” trend**
- ~ Top1 Acc
- ↓MACs & #Params



GREENWAVES TECHNOLOGIES

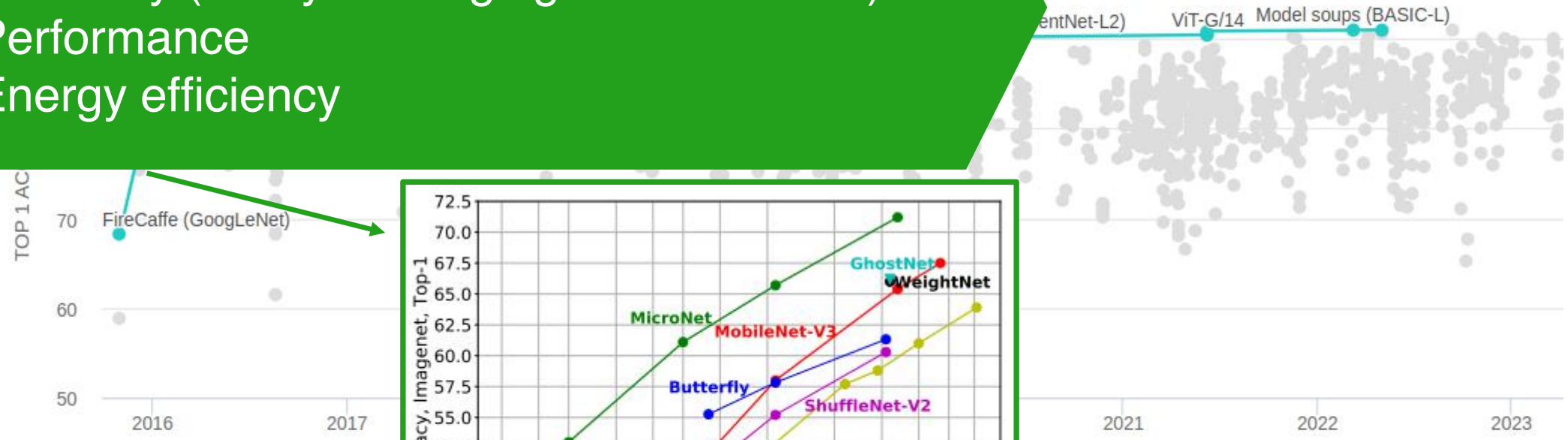
# What is a “tinyML” workload?

**Need:**

1. Flexibility (always-changing NN structures)
2. Performance
3. Energy efficiency

***“Getting Better” trend***

- ↑Top1 Acc
- ↑ MACs & #Params







***“Getting Tinier” trend***

- ~ Top1 Acc
- ↓MACs & #Params

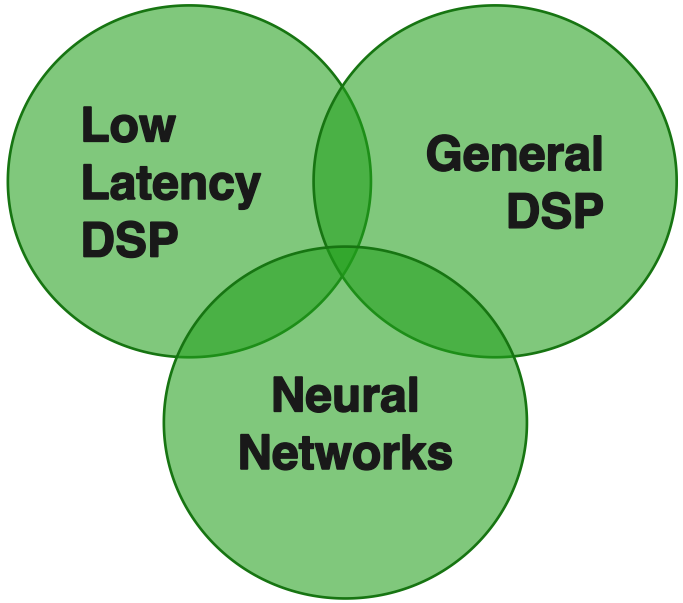




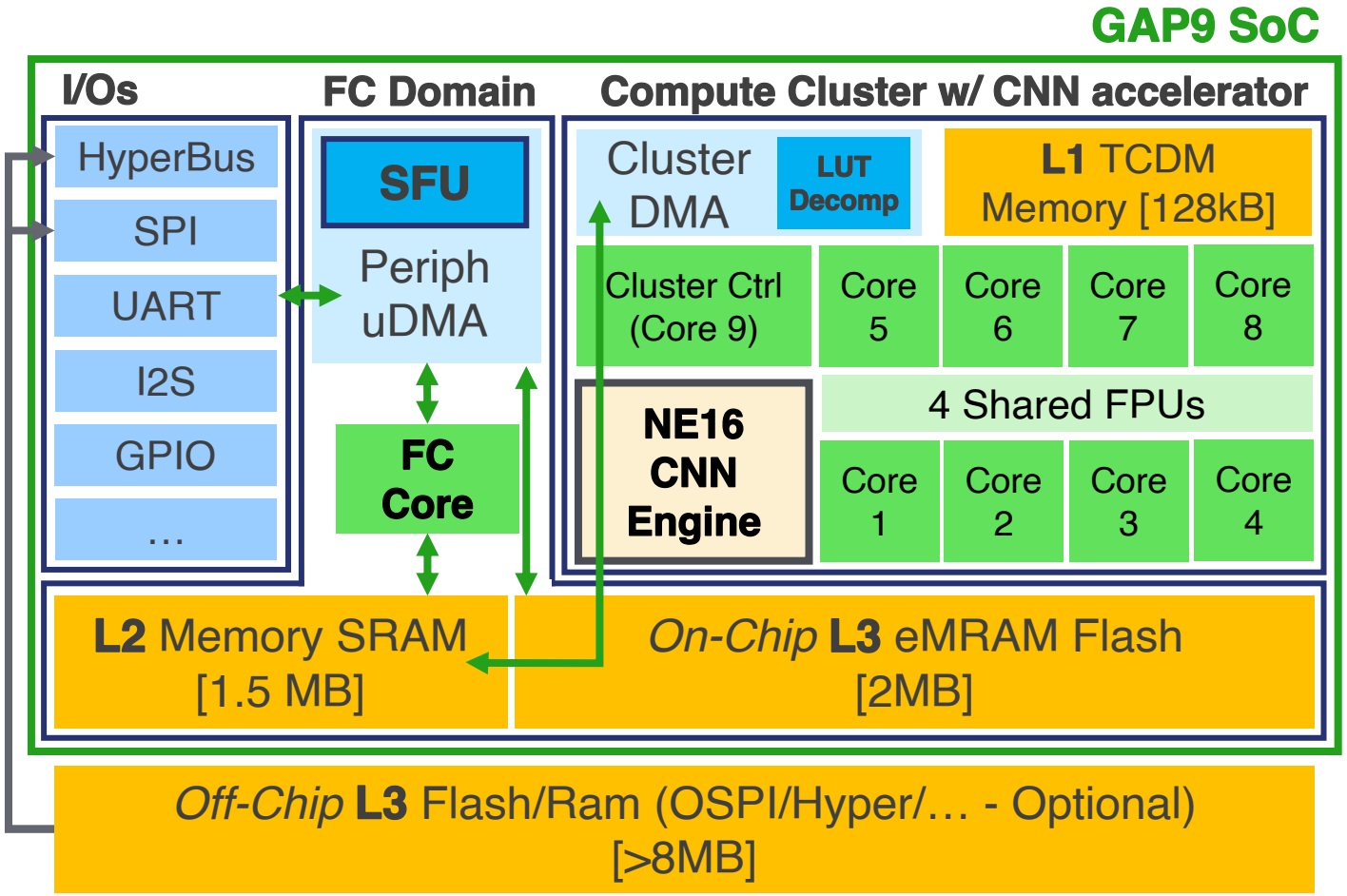
GREENWAVES TECHNOLOGIES

# Foundations of GAP

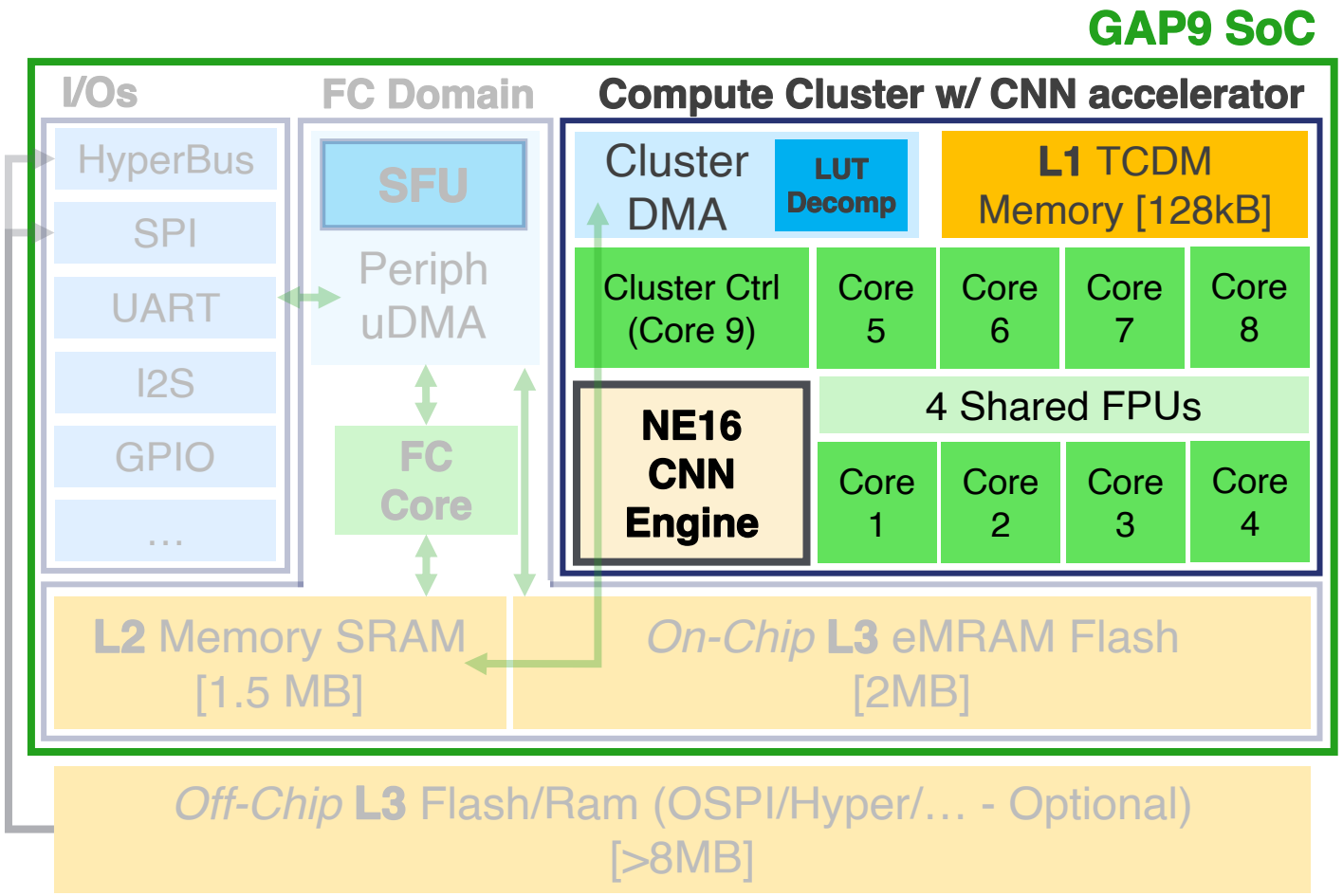
- Born from research groups:
  - **PULP** (Unibo & ETH Zurich)
  - **RISC-V**
- Push **Edge AI** to the limits
  - Specialized HW
  - Optimized SW
  - 10-100 GOPS @ <10mW always-on
- **Programmability**
  - RISC-V open source ISA
  - High level tools



# GAP9 Architecture



# GAP9 Architecture




**Hierarchical Compute paradigm**

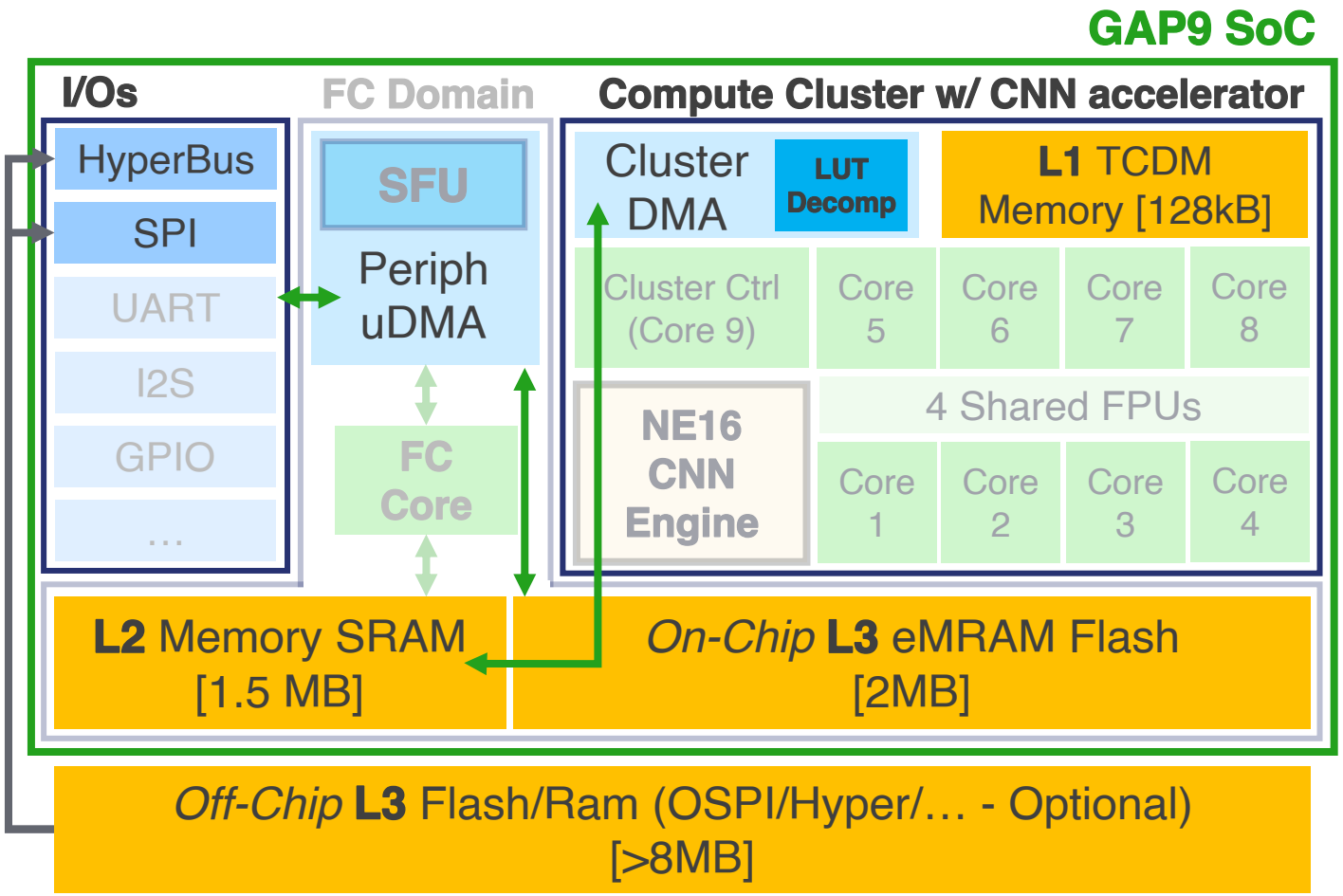
- **4** independent frequency domains: **FC - I/Os - Cluster - SFU**
- *"Turn-on when you need"*





GREENWAVES TECHNOLOGIES

# GAP9 Architecture



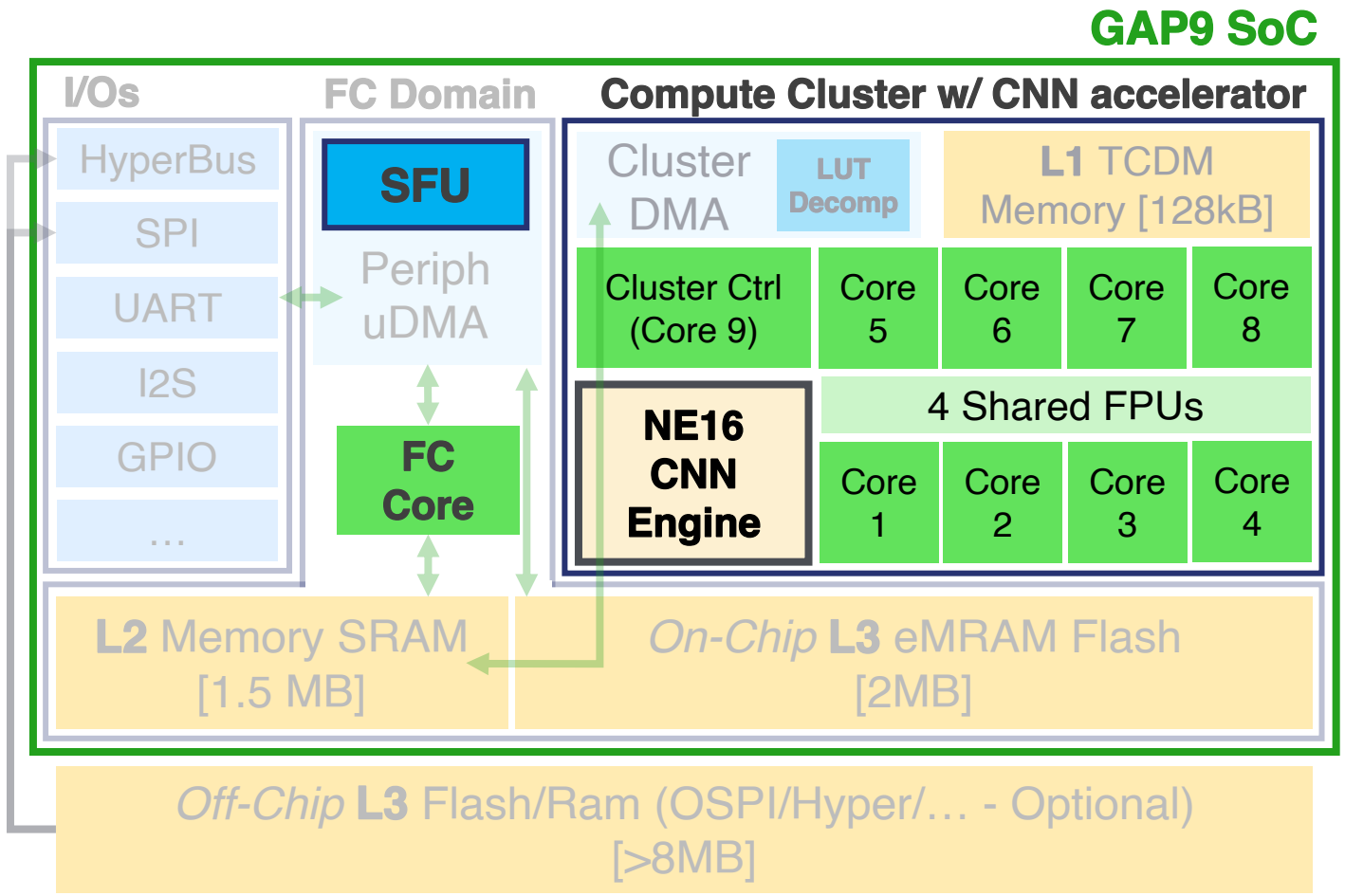

**Hierarchical Compute paradigm**

- **4** independent frequency domains: **FC - I/Os - Cluster - SFU**
- *"Turn-on when you need"*

**Hierarchical Memory Architecture (w/o D-Cache !!!)**

- L1: 128kB – 1 Cyc/Access
- L2: 1.5MB – 10-100 Cyc/Access
- L3: >2MB – 100-1000 Cyc/Access
- DMA and uDMA for background copies+decompression

# GAP9 Architecture




## Hierarchical Compute paradigm

- **4** independent frequency domains: **FC - I/Os - Cluster - SFU**
- *"Turn-on when you need"*

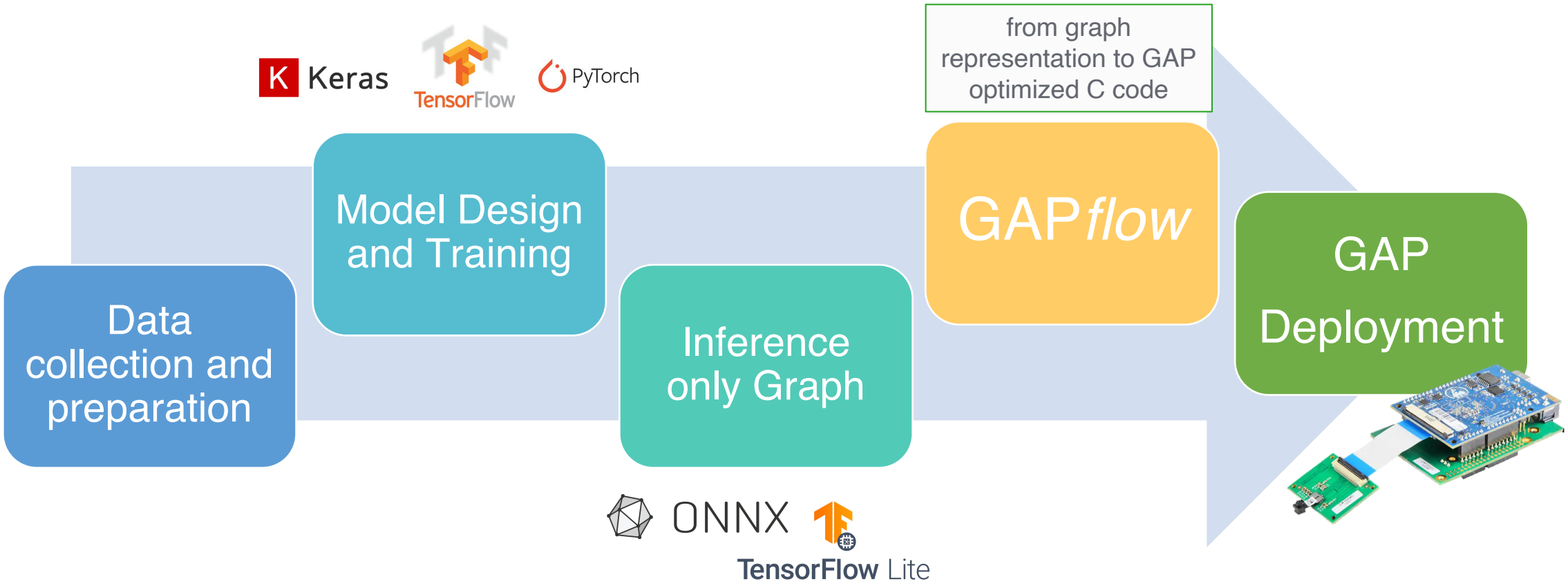
## Hierarchical Memory Architecture (w/o D-Cache !!!)

- L1: 128kB – 1 Cyc/Access
- L2: 1.5MB – 10-100 Cyc/Access
- L3: >2MB – 100-1000 Cyc/Access
- DMA and uDMA for background copies+decompression

## Heterogeneous Compute Units

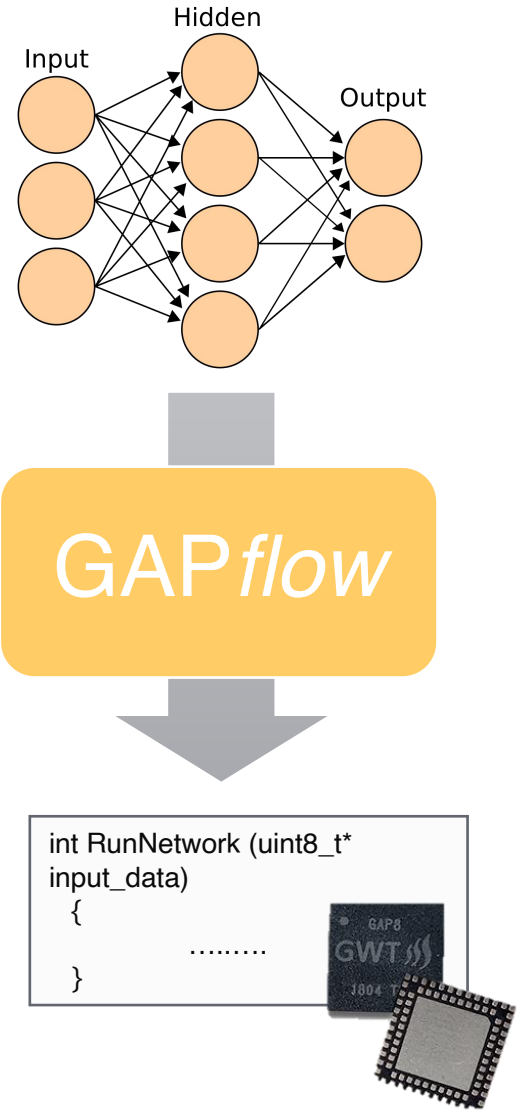
- 10 General Purpose RISC-V Cores
- 4 Shared FPUs (half/single precision)
- Conv/MatMul HW Accelerator (NE16)
- Low-Latency time-domain DSP Accelerator (SFU)

# GAP*flow* enables DNN inference on Parallel-Ultra Low Power GAP MCUs



# What is GAP*flow*?

**MAIN GOAL: turn complex DSP/NN computational graphs into optimized C code for GAP9**



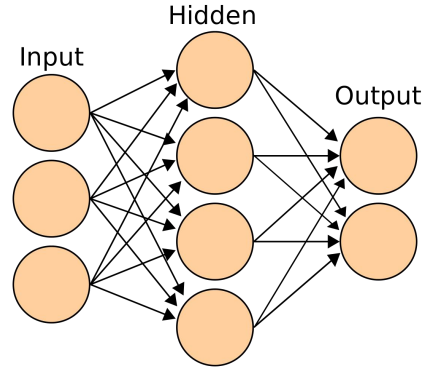





GREENWAVES TECHNOLOGIES

# What is GAP*flow*?

**MAIN GOAL: turn complex DSP/NN computational graphs into optimized C code for GAP9**




GAP*flow*

```
int RunNetwork (uint8_t*
input_data)
  {
        ……..
  }
```


## GAP9 Graph Optimizations

- **Static topology optimization** to minimize number of operations and memory overhead
- **Quantization:** reduce memory usage up to 16x and enable integer only arithmetic when performance is critical

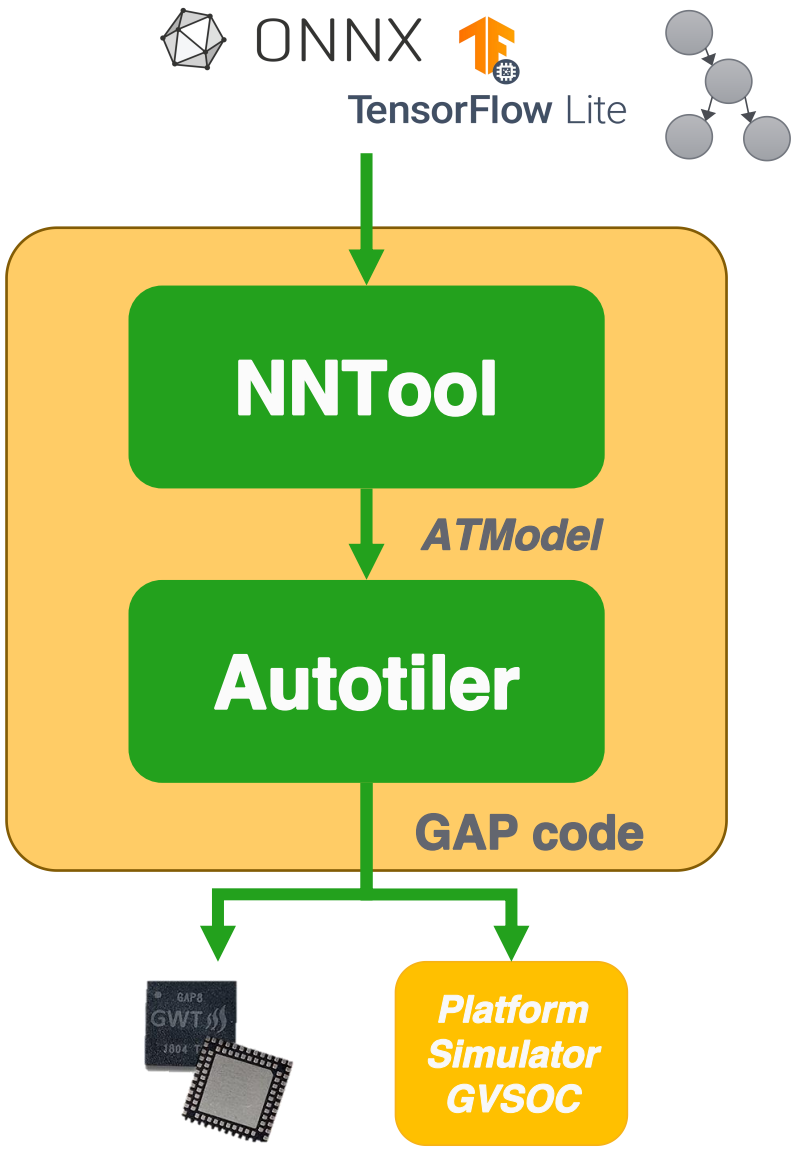
## Validate the Solution

- Validate the numeric precision/accuracy of the deployable model in a user-friendly environment (python)

## Automatic C Code Generation

- Map the graph operations into the **Optimized SW library** of GAP9
- **Optimal Memory Management:** automating memory allocation and data transfers

# GAPflow: Overview




## NNTool

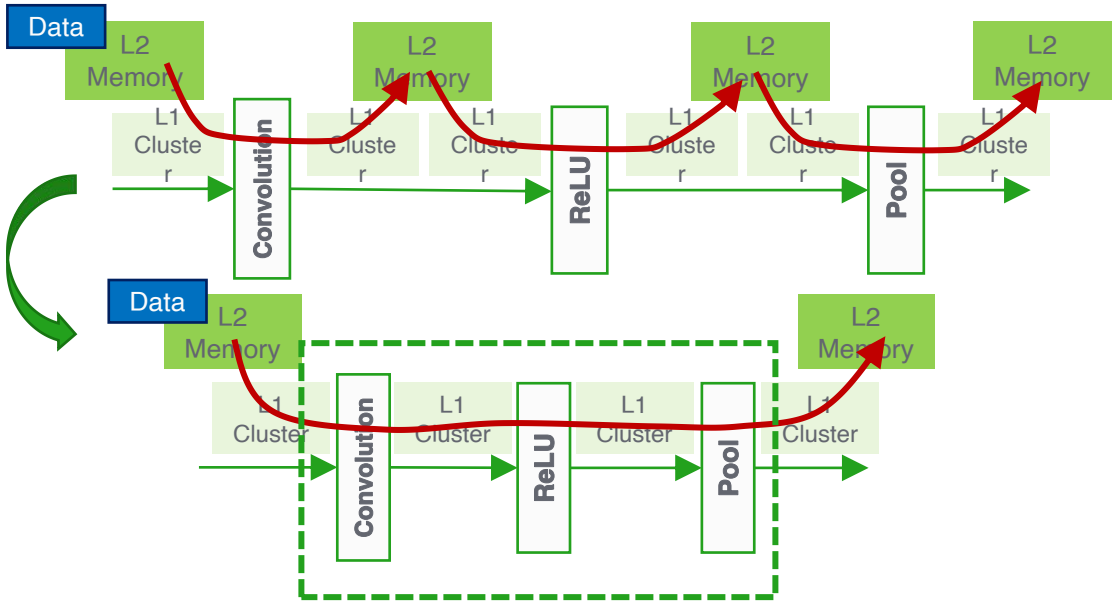
- **Static Topology optimizations** (node fusion)
- **Quantization** w/ calibration dataset (optional)
- **Validate** numerically the deployable solution
- Generates an IR of the graph (***ATModel***)

## Autotiler

- **Optimizes data movement** across the memory hierarchy
- Computes **optimal tiling sizes**
- Generates **GAP code** with double/triple-buffer mechanism using **optimized SW Library primitives**

# Topology Optimizations

- **Remove** useless reshapes/transpose by moving them accross the graph
- **Layer Fusion:** known sequence of nodes merged together thanks to specialized hand-written backend SW
- **Expressions compiler**: fuses an arbitrary sequence of piecewise/broadcastable operations and dynamically generates GAP C code for it

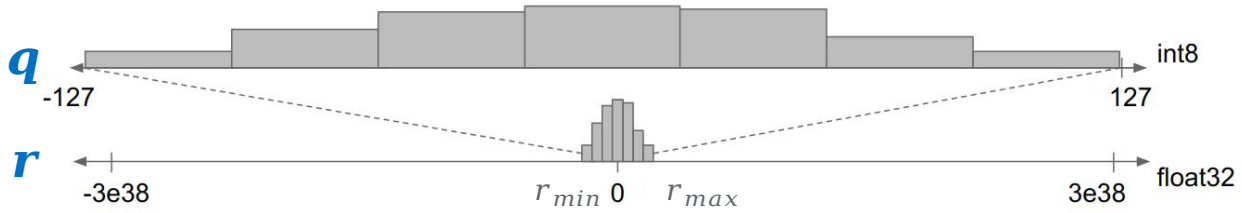


# Quantization: Background

Any real tensor value is mapped into the 8-bit domain (INT8) through an affine transformation:

https://arxiv.org/abs/1712.05877 (TFLite compliant)

$Z = 0$ if symmetric ranges $r_{max} = -r_{min}$




$$r = S\,(q - Z)$$

$$S = \frac{r_{max} - r_{min}}{2^{nbits}}$$

## Convolution Operation

*X* is a quantized value, *x* is a real value

$$y = \sum x \cdot w \longrightarrow S_y Y = \sum S_x X \cdot S_w W \longrightarrow Y = \frac{S_x S_w}{S_y} \sum X \cdot W$$

Affine transformation

Integer only MACs

INT8

# Quantization: Background

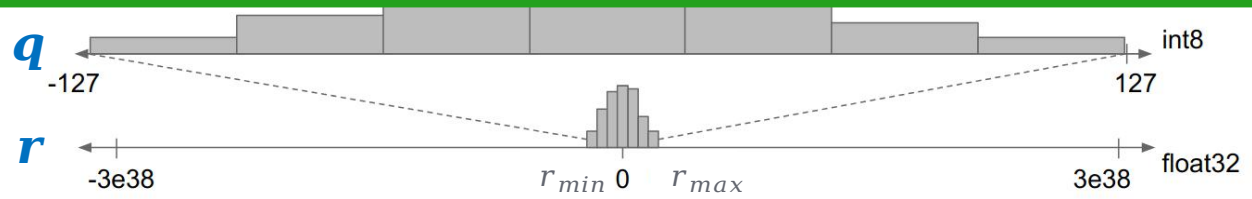
- 4x memory reduction
- >4x speed up thanks to specialized HW
- But accuracy???

through an affine

$Z = 0$ if symmetric ranges $r_{max} = -r_{min}$

$$r = S\,(q - Z)$$

$$\frac{r_{max} - r_{min}}{2^{nbits}}$$




## Convolution Operation

*X* is a quantized value, *x* is a real value

$$y = \sum x \cdot w \longrightarrow S_y Y = \sum S_x X \cdot S_w W \longrightarrow Y = \frac{S_x S_w}{S_y} \sum X \cdot W$$

Affine transformation

Integer only MACs

INT8

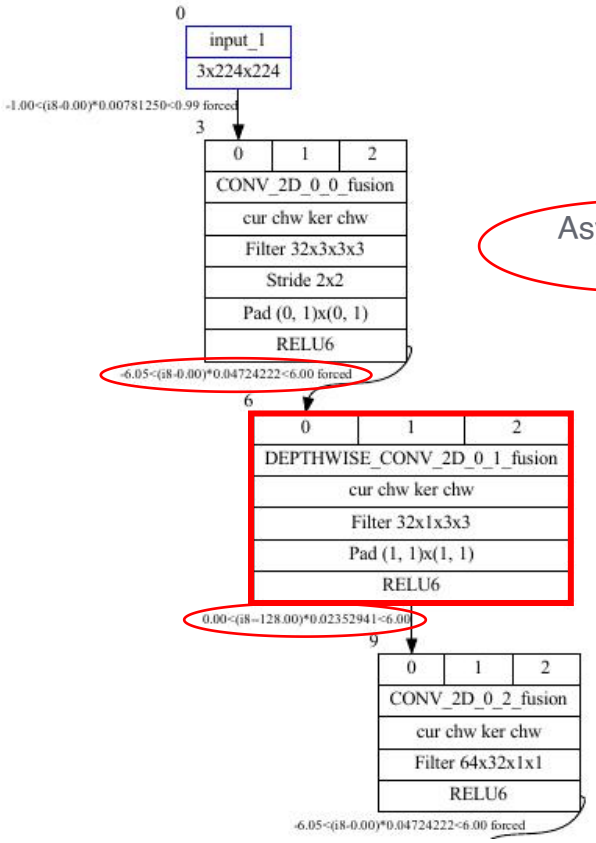




GREENWAVES TECHNOLOGIES

# Mixed-Precision Quantization

GAP NNTool enables layer-wise selection of quantization scheme and number of bits

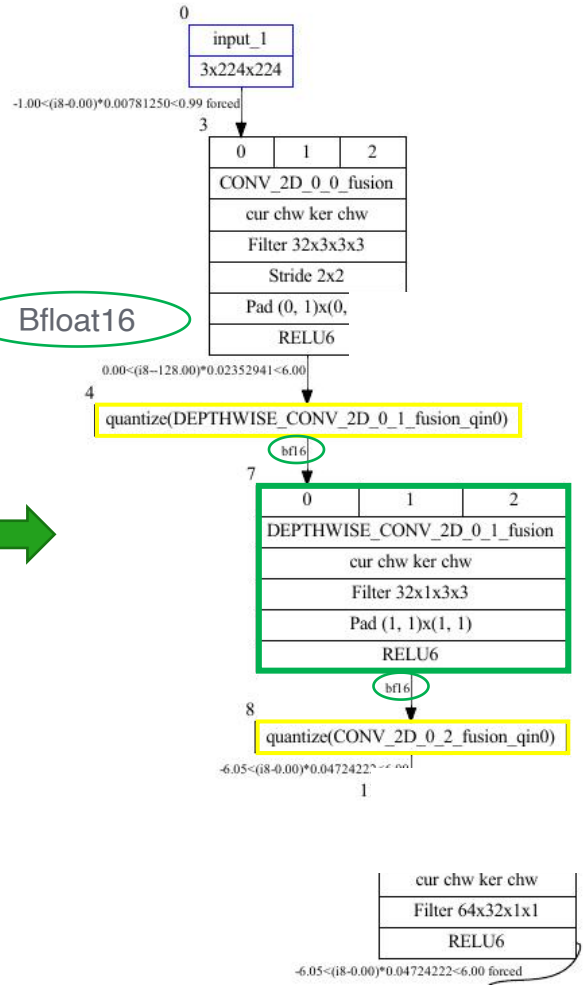



Asymmetric int8

```
G.quantize(
  node_options={
    "DEPTHSWISE_CONV_2D_0_1_fusion":
    {
        "scheme": "FLOAT",
        "float_type"="bfloat16"
    }
  }
)
```

NNTool automatically inserts quantize/dequantize nodes to match quantization range between connected nodes

Bfloat16



# Validation of the solution

- Check deployed model accuracy:

☹ On-device: use directly the final platform to test the accuracy (**very slow**)

☺ NNTool: bit-accurate numpy backend, the user can test accuracy in a python environment without need of device (**fast**)

```
G = NNGraph.load_graph(file_path)
stats = G.collect_statistics(calibration_dataset)
G.quantize(stats, quantization_options)
G.adjust_order()
G.fusions("scaled_match_group")
# Ready for inference
acc1 = 0
for in_data, target in test_dataset:
    outq = G.execute(in_data, quantize=True)
    acc1 += np.argmax(outq[-1][0]) == target
```

***Prepare your model for deployment***
(quantization+graph manipulation)

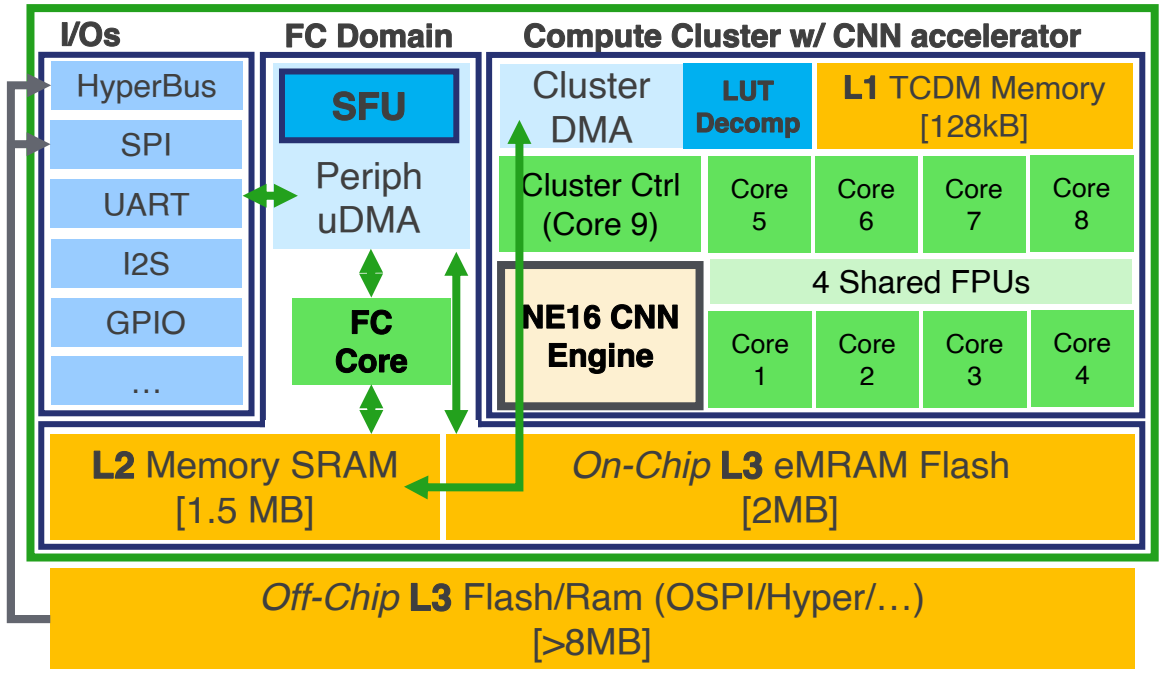
***Run inference and check results***

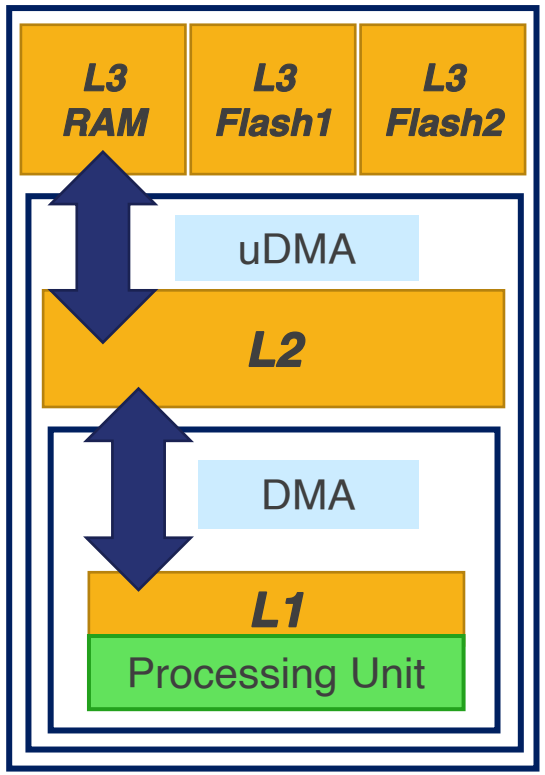
# GAP Autotiler

GAP does not have a data CACHE:

- Silicon Area
- Energy Efficiency
- NN/DSP algo have predictable data traffic

→ Generate C code for all data movement at compile time



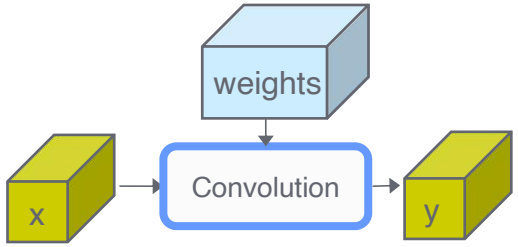


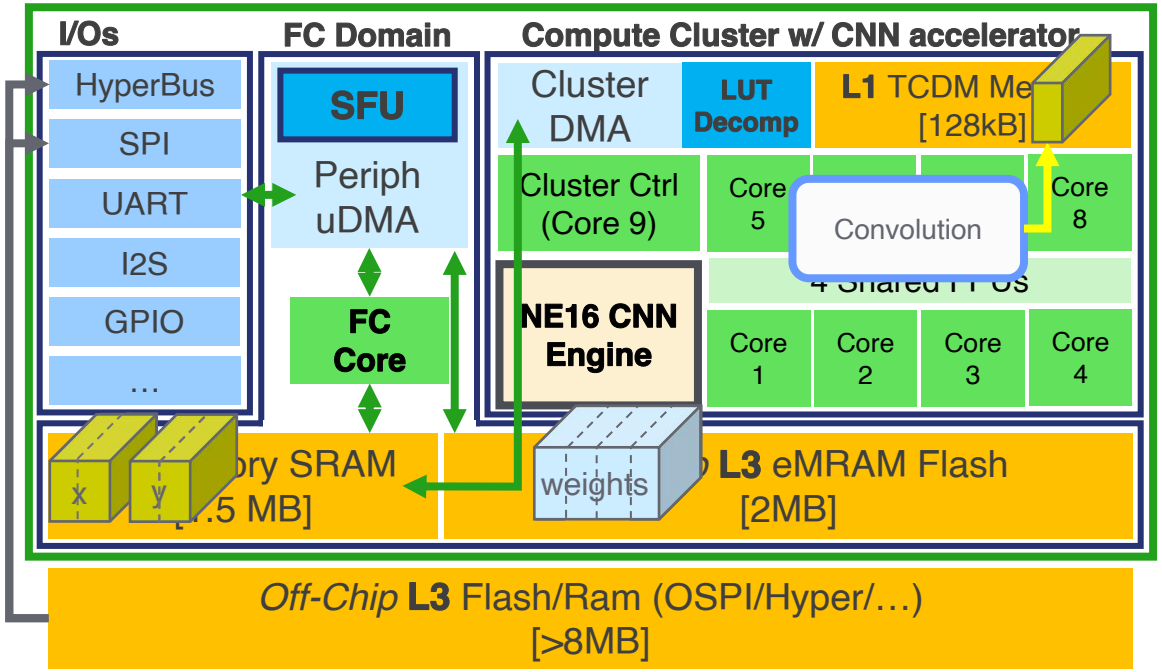



GREENWAVES TECHNOLOGIES

# Autotiler User Kernel: NN Node to the GAP architecture

$$y = \sum x \bullet w$$




## *Computation dataflow*

Ahead of time

Store data (parameters & input vector) in L2 (or L3)

At run time, for any computational node:

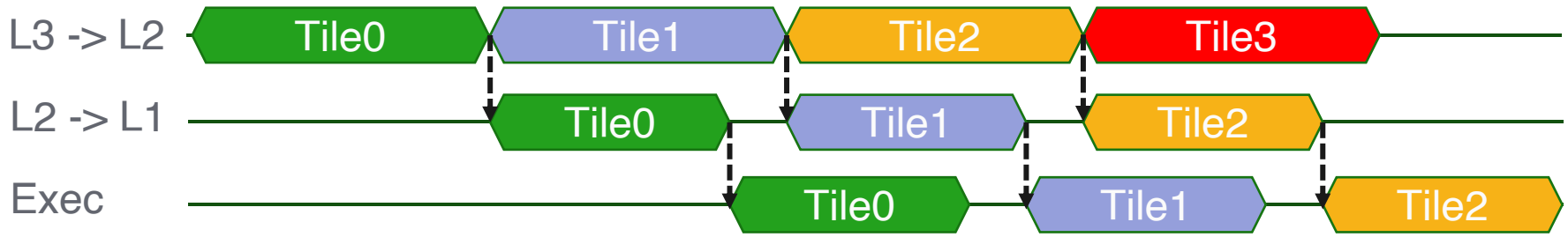
Partition and Load data (parameters & input tensors) to L1

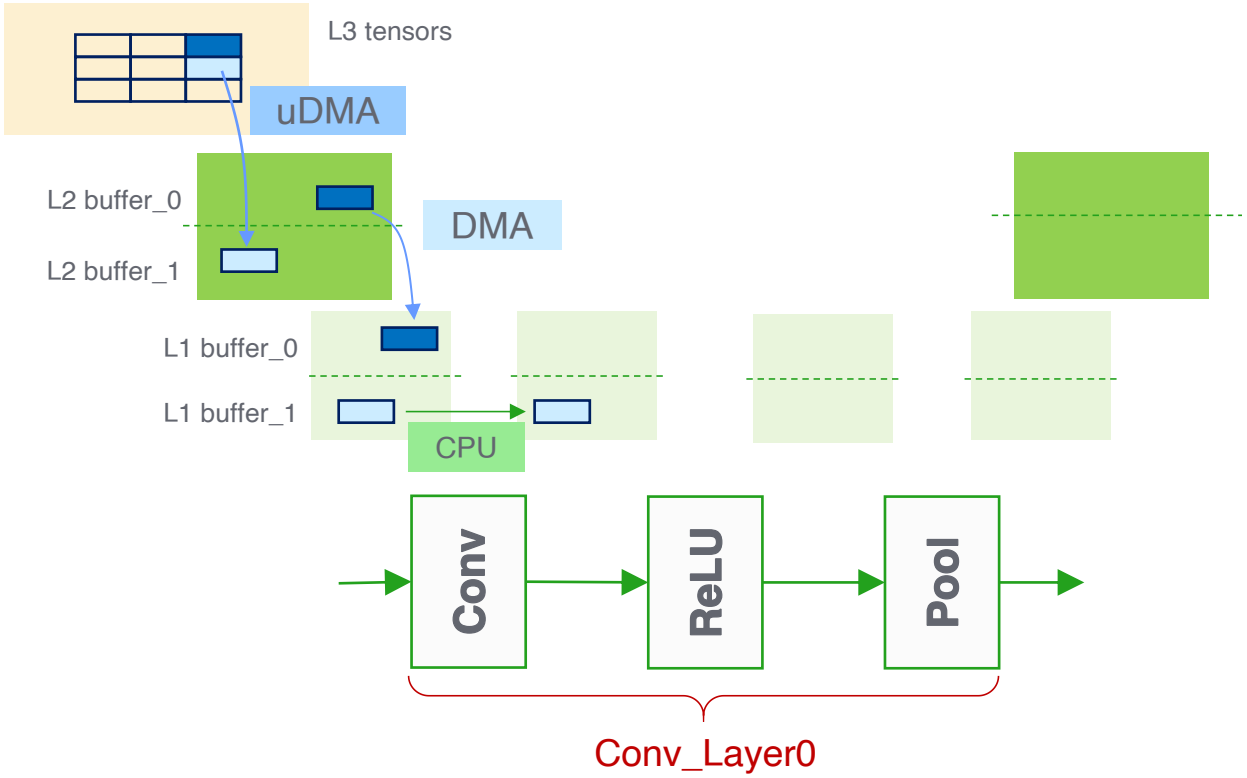
Run data-parallel/CNN Engine computation

Store data (output tensors) back in L2 (or L3)

```
static void Conv_Layer0
(
    signed char * In,       // input L2 vector
    signed char * Weights,  // input L2 vector
    signed char * Bias,     // input L2 vector
    signed char * Out,      // output L2 vector
){

    //tile sizes of In, Weights, Bias computed offline
    //L1 buffer allocated to handle double buffering
    //  two L1 memory buffers for double buffering

   DMA load first tiles to L1 memory buffer

   for any tile of In, Weights, Bias tensors:

      DMA load next tiles to L1 memory buffer

      ParConv() on L1 tile
      ParReLU() on L1 tile
      ParPool() on L1 tile

      DMA write results (Out) to L2

}
```

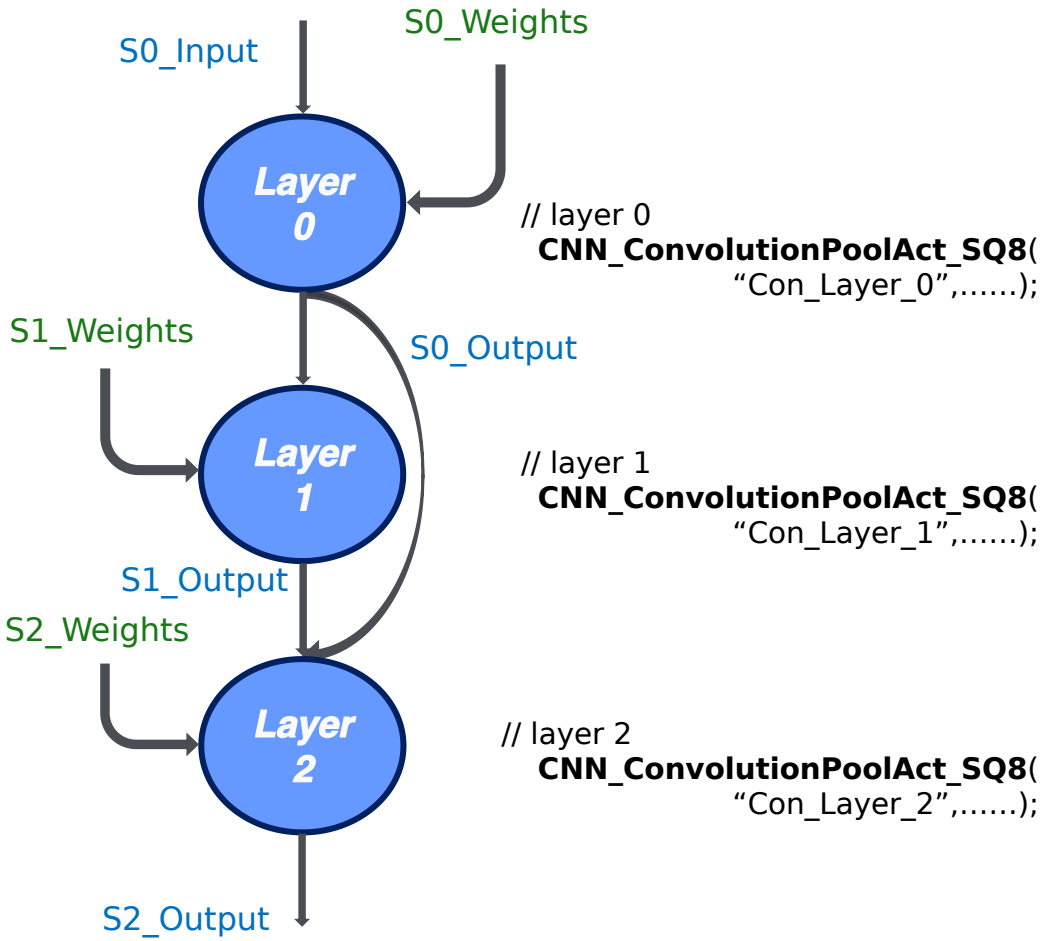
# Mapping a NN Node to the GAP HW/SW architecture




```
static void Conv_Layer0
(
    signed char * In,       // input L3 vector
    signed char * Weights,  // input L3 vector
    signed char * Bias,     // input L3 vector
    signed char * Out,      // output L3 vector
){

    //tile sizes of In, Weights, Bias computed offline
    //L1 buffer allocated to handle double buffering
    //  two L1 memory buffers for double buffering

    uDMA load first tiles to L2 memory buffer

    DMA load first tiles to L1 memory buffer

    for any tile of In, Weights, Bias tensors:

        uDMA load next next tiles to L2 memory buffer

        DMA load next tiles to L1 memory buffer

        ParConv() on L1 tile
        ParReLU() on L1 tile
        ParPool() on L1 tile

        DMA write results (Out) to L2

        uDMA write prev results to L3
}
```

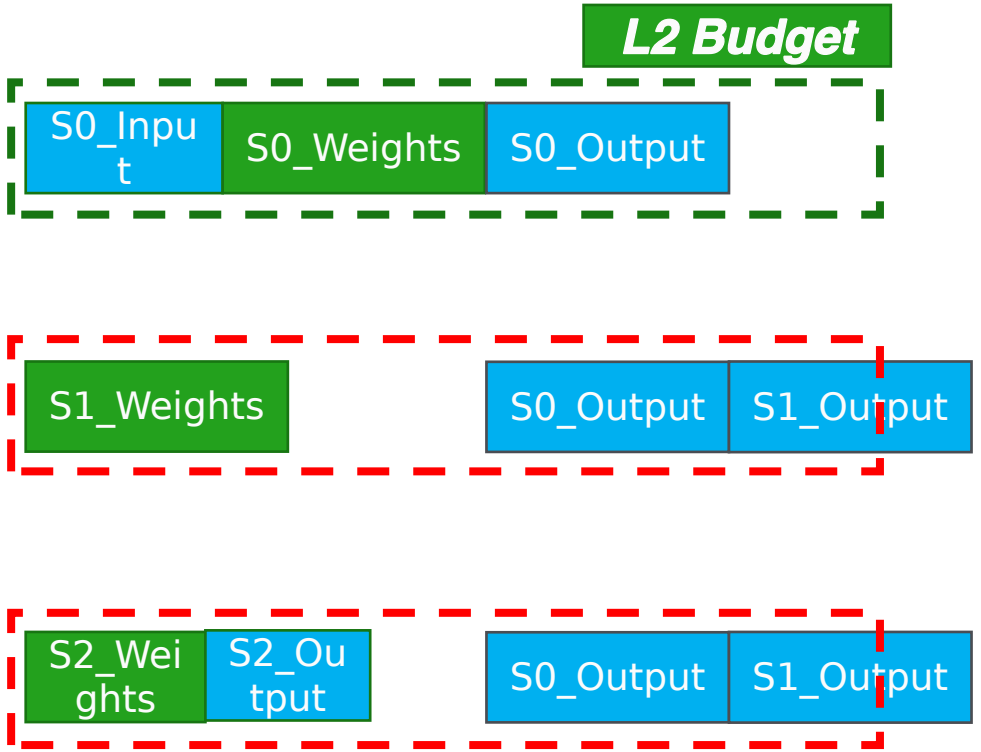
# Autotiler Graph: Static allocation




```
// layer 0
CNN_ConvolutionPoolAct_SQ8(
        "Con_Layer_0",.....);
```

```
// layer 1
CNN_ConvolutionPoolAct_SQ8(
        "Con_Layer_1",.....);
```

```
// layer 2
CNN_ConvolutionPoolAct_SQ8(
        "Con_Layer_2",.....);
```

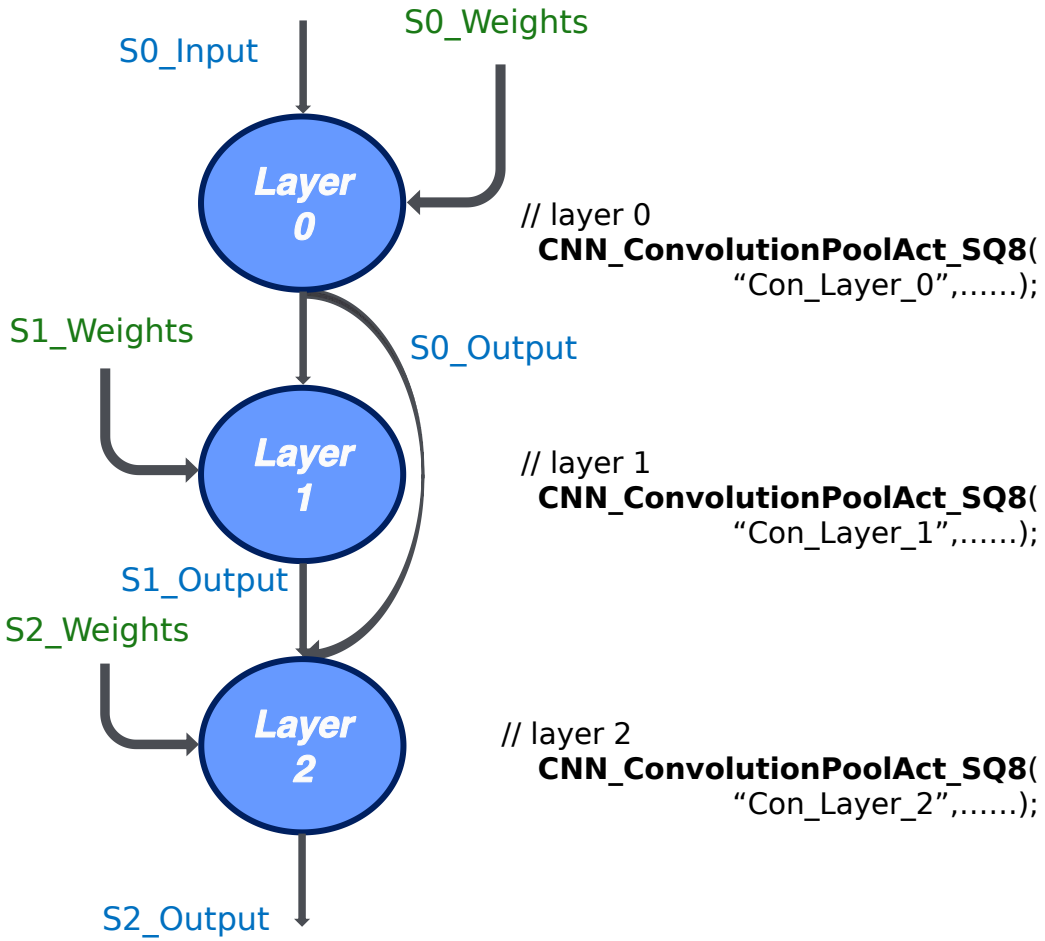




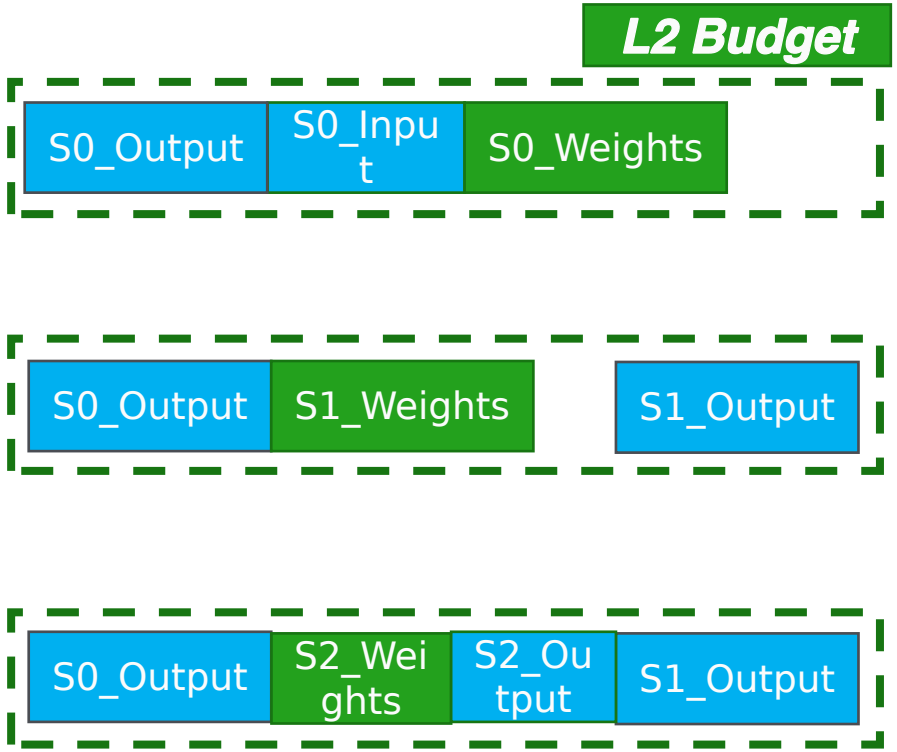



GREENWAVES TECHNOLOGIES

# Autotiler Graph: Static allocation




```
// layer 0
CNN_ConvolutionPoolAct_SQ8(
                "Con_Layer_0",.....);
// layer 1
CNN_ConvolutionPoolAct_SQ8(
                "Con_Layer_1",.....);
// layer 2
CNN_ConvolutionPoolAct_SQ8(
                "Con_Layer_2",.....);
```



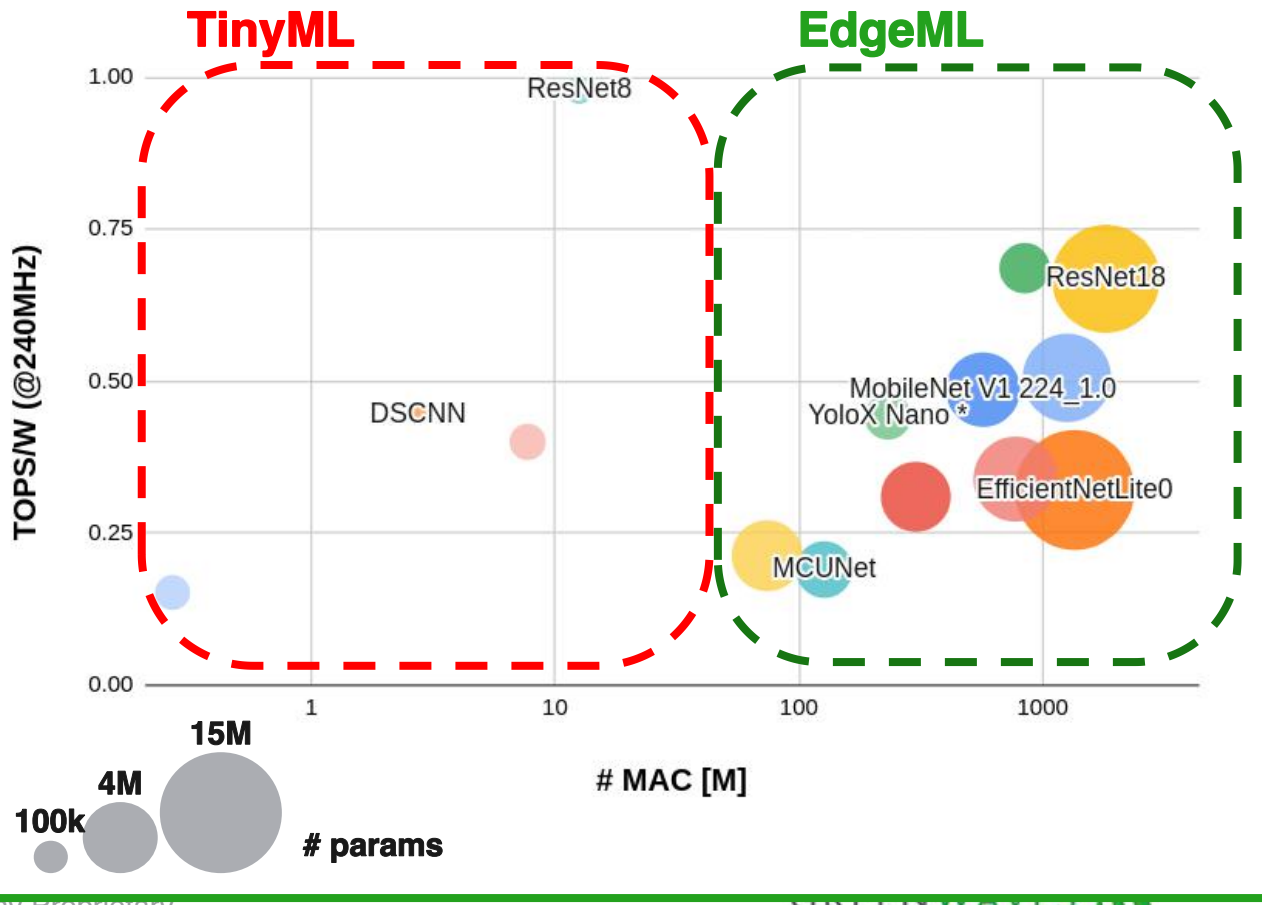
# NN Benchmarks

| | Network | # Params [M] | # MACs [M] | Latency @ 370MHz [ms] | GOPS/W @ 240MHz |
|---|---|---|---|---|---|
| Image and Audio classification and detection | MobileNet V1 224_1.0 | 4.27 | 568.9 | 45.89 | 485.38 |
| | MobileNet V2 224_1.0 | 3.57 | 301.8 | 37.59 | 308.89 |
| | ResNet18 | 11.71 | 1816.7 | 81.54 | 667.98 |
| | SqueezeNet | 1.25 | 843.4 | 46.14 | 685.49 |
| | EfficientNetLite0 | 15.02 | 1352.9 | 182.92 | 320.10 |
| | MCUNet | 1.84 | 127.1 | 29.08 | 189.01 |
| | SSD MobileNet V1 300_1.0 | 6.89 | 1258.1 | 93.59 | 504.79 |
| | SSD MobileNet V2 300_1.0 | 6.17 | 775.8 | 86.11 | 338.02 |
| | YamNet w/ Preprocessing | 3.72 | 74.23 | 13.03 | 211.87 |
| | YoloX Nano * | 0.91 | 231.7 | 22.78 | 441.04 |
| TinyMLPerf: Best in the market | TinyML - KWS: DSCNN | 0.053 | 2.74 | 0.30 | 445.53 |
| | TinyML - IC: ResNet8 | 0.098 | 12.58 | 0.57 | 978.99 |
| | TinyML - AD: DeepAutoEncoder | 0.277 | 0.27 | 0.11 | 151.26 |
| | TinyML - VWW: MobileNet V1 0.25 | 0.333 | 7.72 | 1.05 | 399.28 |

# NN Benchmarks

| | Network | # Params [M] | # MACs [M] | Latency @ 370 |
|---|---|---|---|---|
| Image and Audio classification and detection | MobileNet V1 224_1.0 | 4.27 | 568.9 | |
| | MobileNet V2 224_1.0 | | | |
| | ResNet18 | | | |
| | SqueezeNet | | | |
| | EfficientNetLite0 | | | |
| | MCUNet | | | |
| | SSD MobileNet V1 300_1.0 | | | |
| | SSD MobileNet V2 300_1.0 | | | |
| | YamNet w/ Preprocessing | | | |
| | YoloX Nano * | | | |
| TinyMLPerf: Best in the market | TinyML - KWS: DSCNN | | | |
| | TinyML - IC: ResNet8 | | | |
| | TinyML - AD: DeepAutoEncoder | | | |
| | TinyML - VWW: MobileNet V1 0.25 | | | |

***>100GOPS/W*** accross large range of NNs



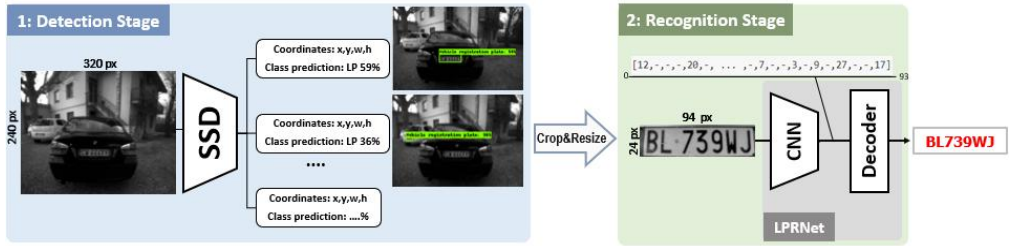





GREENWAVES TECHNOLOGIES

# Enabling complex NN on energy constrained devices

## Multi NN – Licence Plate

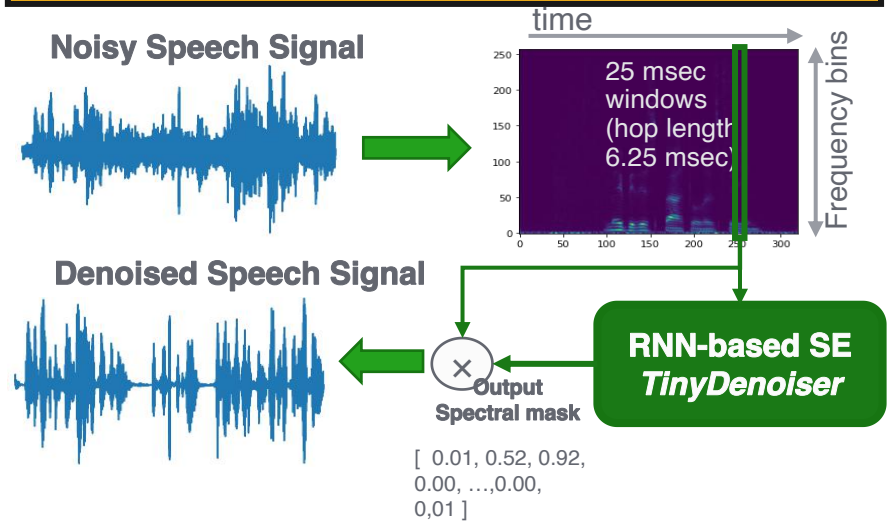



https://ieeexplore.ieee.org/abstract/document/9401730

## Smart glasses - object detection

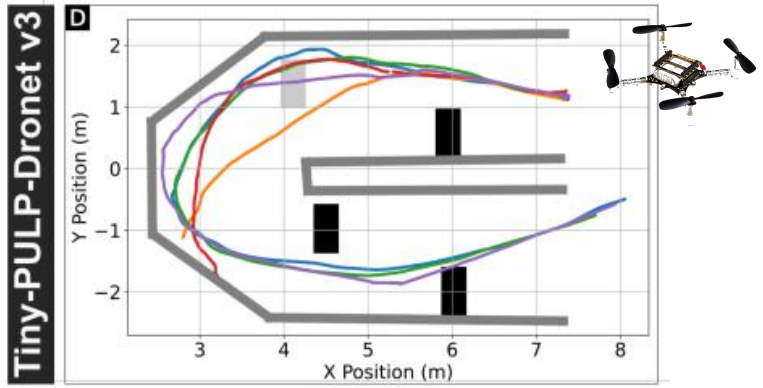

https://arxiv.org/ftp/arxiv/papers/2311/2311.01057.pdf

## DSP/NN Mixed-Precision – Denoiser




https://arxiv.org/pdf/2210.07692

## Nano drones autonomous navigation




https://arxiv.org/pdf/2407.12675



GREENWAVES TECHNOLOGIES

# Try yourself

- Contact us and get our evaluation kit board
- Tools and SW are mostly open-source:
  - https://greenwaves-technologies.com/tools-and-software/
  - https://greenwaves-technologies.com/manuals_gap9/gap9_sdk_doc/html/
- Lot of application and example to start your work
  (write us an email or sign in the website to access our SDK)



GREENWAVES TECHNOLOGIES

# Thank you!

GREENWAVES TECHNOLOGIES

# NN Benchmarks

| | Network | # Params [M] | # MACs [M] | Latency @ 370MHz | GOPS/W |
|---|---|---|---|---|---|
| Image and Audio classification and detection | MobileNet V1 224_1.0 | 4.27 | 568.9 | | |
| | MobileNet V2 224_1.0 | 3.57 | 301.8 | | |
| | ResNet18 | | | | |
| | SqueezeNet | | | | |
| | EfficientNetLite0 | | | | |
| | MCUNet | | | | |
| | SSD MobileNet V1 300_1.0 | | | | |
| | SSD MobileNet V2 300_1.0 | | | | |
| | YamNet w/ Preprocessing | | | | |
| | YoloX Nano * | | | | |
| TinyMLPerf: Best in the market | TinyML - KWS: DSCNN | | | | |
| | TinyML - IC: ResNet8 | | | | |
| | TinyML - AD: DeepAutoEncoder | | | | |
| | TinyML - VWW: MobileNet V1 0.25 | | | | |

***>10GOPS*** accross large range of NNs