“GAP8: A Parallel, Ultra-low-power and flexible RISC-V based IoT Application Processor for the TinyML ecosystem”

Manuele Rusci – Greenwaves Technologies

October 27, 2020
tinyML Talks Sponsors

Additional Sponsorships available – contact Bette@tinyML.org for info
WE USE AI TO MAKE OTHER AI FASTER, SMALLER AND MORE POWER EFFICIENT

Automatically compress SOTA models like MobileNet to <200KB with little to no drop in accuracy for inference on resource-limited MCUs

Reduce model optimization trial & error from weeks to days using Deeplite's design space exploration

Deploy more models to your device without sacrificing performance or battery life with our easy-to-use software

BECOME BETA USER bit.ly/testdeeplite
TinyML for all developers

Acquire valuable training data securely

Enrich data and train ML algorithms

Edge Device
Real sensors in real time
Open source SDK

Embedded and edge compute deployment options

Test

Test impulse with real-time device data flows

Dataset

Get your free account at http://edgeimpulse.com
Health sensors measure PPG and ECG signals critical to understanding vital signs. Signal chain products enable measuring even the most sensitive signals.

The biggest (3MB flash and 1MB SRAM) and the smallest (256KB flash and 96KB SRAM) Cortex M4 microcontrollers enable algorithms and neural networks to run at wearable power levels.

The new MAX78000 implements AI inferences at over 100x lower energy than other embedded options. Now the edge can see and hear like never before.
Qeexo AutoML for Embedded AI
Automated Machine Learning Platform that builds tinyML solutions for the Edge using sensor data

Key Features

- Wide range of ML methods: GBM, XGBoost, Random Forest, Logistic Regression, Decision Tree, SVM, CNN, RNN, CRNN, ANN, Local Outlier Factor, and Isolation Forest
- Easy-to-use interface for labeling, recording, validating, and visualizing time-series sensor data
- On-device inference optimized for low latency, low power consumption, and a small memory footprint
- Supports Arm® Cortex™- M0 to M4 class MCUs
- Automates complex and labor-intensive processes of a typical ML workflow – no coding or ML expertise required!

Target Markets/Applications

- Industrial Predictive Maintenance
- Automotive
- Smart Home
- Mobile
- Wearables
- IoT

For a limited time, sign up to use Qeexo AutoML at automl.qeexo.com for FREE to bring intelligence to your devices!
is for building products

Reality AI Tools® software
- Automated Feature Exploration and Model Generation
- Bill-of-Materials Optimization
- Automated Data Assessment
- Edge AI / TinyML code for the smallest MCUs

Reality AI solutions
- Automotive sound recognition & localization
- Indoor/outdoor sound event recognition
- RealityCheck™ voice anti-spoofing

https://reality.ai  info@reality.ai  @SensorAI  Reality AI
SynSense (formerly known as aiCTX) builds ultra-low-power (sub-mW) sensing and inference hardware for embedded, mobile and edge devices. We design systems for real-time always-on smart sensing, for audio, vision, bio-signals and more.

https://SynSense.ai
tinyML Strategic Partners

arm

Additional Sponsorships available – contact Bette@tinyML.org for info
Free event courtesy of our sponsors – register today!

Call for Video ‘Posters’
5 minute presentations
Abstracts due October 31

Full program at www.tinyml.org/asia2020
Live online program daily at 9 am China time

Premier Sponsor

Additional sponsorships available - contact Bette@tinyml.org for info
### Next tinyML Talks

<table>
<thead>
<tr>
<th>Date</th>
<th>Presenter</th>
<th>Topic / Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, November 10</td>
<td><strong>Ehsan Saboori</strong> Co-founder and CTO, Deeplite</td>
<td>Networks within Networks: Novel CNN design space exploration for resource limited devices</td>
</tr>
<tr>
<td></td>
<td><strong>Alexander Samuelsson</strong> CTO and co-founder, Imagimob</td>
<td>How to build advanced hand-gestures using radar and tinyML</td>
</tr>
</tbody>
</table>

Webcast start time is 8 am Pacific time
Each presentation is approximately 30 minutes in length

Please contact talks@tinylm.org if you are interested in presenting
Dr. Manuele Rusci works as Embedded Machine Learning Engineer at GreenWaves Technologies. He obtained the PhD in 2018 from the University of Bologna, where he also works as a research assistant. His main research interests include low-power embedded systems and AI-powered smart sensors.
GAP8

A Parallel, Ultra-low-power and flexible RISC-V based IoT Application Processor for the TinyML ecosystem

Presented by:
Manuele Rusci
GreenWaves Technologies
From Open-Source HW to GAP8

Company Foundation
Grenoble France
2014

Launch First Product
GAP8
February 2018

PULP project started
2013

Credits: https://www.pulp-platform.org/

Company Proprietary
GAP8 - Architected to scale energy proportional to demand

Two independent clock and voltage domains, from 0-133MHz/1V up to 0-250MHz/1.2V

An integrated, hierarchical architecture

MCU Function
- Extended RISC-V core
- Extensive I/O set
- Micro DMA
- Embedded DC/DC converters

Computation engine function
- 8 extended RISC-V cores
- Fully programmable
- Efficient parallelization
- Shared instruction cache
- HW synchronization
- HW convolution Engine (3 * 3x3)
Scaling proportional to demand

**Duty Cycling**
- Ultra fast switching time from one mode to another
- Ultra fast voltage and frequency change time

**Coarse Grain Classification**
- Low quiescent LDO
- Real Time Clock 32KHz only
- L2 Memory partially retentive

**Full Blown Analysis**
- Embedded DC/DC, high current
- Voltage can dynamically change
- One clock gen active, frequency can dynamically change
- Systematic clock gating

**MCU sleep mode**
- 1 to 50 uW

**MCU active mode**
- 0.5 to 10 mW
- Embedded DC/DC, high current
- Voltage can dynamically change
- One clock gen active, frequency can dynamically change
- Systematic clock gating

**MCU + Parallel processor active mode**
- 10 to 100 mW
- Embedded DC/DC, high current
- Voltage can dynamically change
- Two clock gen active, frequencies can dynamically change
- Systematic Clock Gating

Ultra fast switching time from one mode to another
Ultra fast voltage and frequency change time

Highly optimized system level power consumption

Company Proprietary
Deep Learning based Image Processing on GAP8

$ git clone git@github.com:GreenWaves-Technologies/image_classification_networks.git
$ make clean all run platform=gvsoc

Credits: https://ai.googleblog.com/2017/06/mobilenets-open-source-models-for.html

Supported Models

Below the models tested with the GAP flow. More will come shortly...

<table>
<thead>
<tr>
<th>MODEL ID</th>
<th>Quantized TFLite Graph</th>
<th>MACs (M)</th>
<th>Parameters (M)</th>
<th>Top 1 Accuracy</th>
<th>FPS</th>
<th>Energy (mJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MobileNet V1 224 1.0</td>
<td>569</td>
<td>4.24</td>
<td>70.1</td>
<td>2.1</td>
<td>49.7</td>
</tr>
<tr>
<td>1</td>
<td>MobileNet V1 192 1.0</td>
<td>418</td>
<td>4.24</td>
<td>69.2</td>
<td>3.1</td>
<td>35.2</td>
</tr>
<tr>
<td>2</td>
<td>MobileNet V1 160 1.0</td>
<td>291</td>
<td>4.24</td>
<td>67.2</td>
<td>4.4</td>
<td>25.1</td>
</tr>
<tr>
<td>3</td>
<td>MobileNet V1 128 1.0</td>
<td>186</td>
<td>4.24</td>
<td>63.4</td>
<td>7.2</td>
<td>15.2</td>
</tr>
<tr>
<td>4</td>
<td>MobileNet V1 224 0.75</td>
<td>317</td>
<td>2.59</td>
<td>66.8</td>
<td>4.0</td>
<td>27.6</td>
</tr>
<tr>
<td>5</td>
<td>MobileNet V1 192 0.75</td>
<td>233</td>
<td>2.59</td>
<td>66.1</td>
<td>5.3</td>
<td>20.9</td>
</tr>
<tr>
<td>6</td>
<td>MobileNet V1 160 0.75</td>
<td>162</td>
<td>2.59</td>
<td>62.3</td>
<td>8.5</td>
<td>13.3</td>
</tr>
<tr>
<td>7</td>
<td>MobileNet V1 128 0.75</td>
<td>104</td>
<td>2.59</td>
<td>55.8</td>
<td>12.9</td>
<td>8.5</td>
</tr>
</tbody>
</table>

- GAP8 @ 1.2V, 175MHz Cluster, 250MHz FC, up to 110mW
- from 1.5mJ @66fps to 55mJ@2fps per inference (incl. ext memories)
Automatic License Plate Recognition

SSD Detector

<table>
<thead>
<tr>
<th>Net</th>
<th>Acc.</th>
<th>Sz (MB)</th>
<th>MAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD-MV1</td>
<td>39.6%</td>
<td>22.0</td>
<td>931M</td>
</tr>
<tr>
<td>SSD-MV2</td>
<td>42.0%</td>
<td>18.4</td>
<td>540M</td>
</tr>
<tr>
<td>SSDlite-MBV2</td>
<td>39.8%</td>
<td>12.4</td>
<td>515M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Size</th>
<th>Acc.</th>
<th>Sz (MB)</th>
<th>MAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>512x512</td>
<td>45.4%</td>
<td>18.4</td>
<td>1.76G</td>
</tr>
<tr>
<td>352x352</td>
<td>42.1%</td>
<td>18.4</td>
<td>840M</td>
</tr>
<tr>
<td>320x240</td>
<td>42.0%</td>
<td>18.4</td>
<td>540M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8 bit SSDlite sym</th>
<th>Acc.</th>
<th>Sz (MB)</th>
<th>MAC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>38.9%</td>
<td>3.1</td>
<td>515M</td>
</tr>
</tbody>
</table>

LP Read

<table>
<thead>
<tr>
<th>Optimization for deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline Net.</td>
</tr>
<tr>
<td>Layer replaced</td>
</tr>
<tr>
<td>FC Layer halved</td>
</tr>
<tr>
<td>8 bit</td>
</tr>
</tbody>
</table>

- 1.1 FPS inference @ 175MHz, performing 687M MAC.
- 4.1 MB memory footprint (after 8-bit quantization).
- Accuracy: 39% mAP for LP det. & > 99.13% for LP rec.
- Max recognition distance: 4m for detection and 2m for recognition.
- 117mW power envelope, 108 mJ per inference.
- SoA: 73x less energy w.r.t. previous ALPR system.
Running Inference on sensor data

How to deploy it on a GAP8-based system?
A Low-Power Visual Recognition System

1) Build your CV system
A Low-Power Visual Recognition System

1) Build your CV system
2) Get Image
A Low-Power Visual Recognition System

1) Build your CV system
2) Get Image
3) Turn the cluster ON
A Low-Power Visual Recognition System

1) Build your CV system
2) Get Image
3) Turn the cluster ON
4) Run Inference

HyperBus
SPI
UART
I2C
CPI
PIO

Core FC
L1 Mem FC

L2 Memory 512kB

Octa Core Cluster

L1 Cluster TCDM memory (64kB)
Core 1
Core 2
Core 3
Core 4
Core 5
Core 6
Core 7
Core 8
int main(void) {
  // Allocate L2 Buffer
  uint8_t* input_img = (uint8_t*) pmsis_l2_malloc(CAMERA_SIZE*sizeof(char));

  // Get Camera image
  pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);
  pi_camera_capture(&camera, input_img, CAMERA_SIZE);

  // Turn on the Cluster
  struct pi_device cluster_dev;
  struct pi_cluster_conf conf;
  pi_cluster_conf_init(&conf);
  pi_open_from_conf(&cluster_dev, (void *)&conf);
  pi_cluster_open(&cluster_dev);

  // Dispatch Inference Task on Cluster
  struct pi_cluster_task *task = pmsis_l2_malloc(sizeof(struct pi_cluster_task));
  if(task==NULL) {
    pmsis_exit(-1);
  }
  task->entry = &RunNetwork;
  task->arg = input_img;

  pi_cluster_send_task_to_cl(&cluster_dev, task);
  pi_cluster_close(&cluster_dev);
}

int RunNetwork(uint8_t* input_data) { ... }
The GAPflow

The GAPflow automates the deployment of deep networks on the GAP processors

```c
int RunNetwork (uint8_t* input_data) {
    // .......
}
```
GAPflow

Employing deep learning frameworks (e.g., TensorFlow, Keras, etc.) to perform designing a model, training, evaluation, fine tuning, and prediction.

GAPflow

Convert the complex models into smaller and less complex models (i.e., tflite) to be able to run on small devices with low capacity.

TensorFlow Lite

from graph representation to GAP-optimized C code

Data collection and preparation

Gathering data and the process of cleaning and transforming raw data prior to processing and analysis.

Model Training

Graph Conversion

GAP Deployment
GAPflow: Overview

GAPflow

GAP NNTool

- Import TFLite graph
- Map TF nodes to GAP Computational node (fusion and reshaping operation)
- Quantization w/ calibration dataset (optional)
- Produce an AT Model

GAP AutoTiler

- Fed with a C-coded graph model – AT Model
- Optimize data movement across the memory hierarchy based on the provided graph and the memory constraints (L1, L2, L3 sizes)
  - Compute optimal tiling sizes at compile-time
  - Produce an optimized parallel code with memory movement calls

Tflite Graph

AT Model

C code

Platform Simulator GVSOC

GAP NNTool

GAP AutoTiler

Company Proprietary
Mapping a NN Graph to the GAP HW/SW architecture

Deploy on GAP

Main Challenge
Optimize data transfer from to the cluster parallel engine (no Dcache)

void ParConv (uint8_t* input_L1, uint8_t* weight_L1, uint8_t* output_L1 )
{
    core_id = get_core_id();
    apply data parallel convolution
}

GAP Optimized SW Basic Kernel
Mapping a NN Graph to the GAP HW/SW architecture

void ParConv (uint8_t* input_L1, uint8_t* weight_L1, uint8_t* output_L1 )
{
    core_id = get_core_id();
    apply data parallel convolution
}

GAP Optimized SW Basic Kernel

Computation dataflow
Mapping a NN Graph to the GAP HW/SW architecture

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

```c
void ParConv (uint8_t* input_L1, uint8_t* weight_L1, uint8_t* output_L1 )
{
    core_id = get_core_id();
    apply data parallel convolution
}
```

GAP Optimized SW Basic Kernel
Mapping a NN Graph to the GAP HW/SW architecture

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

```c
void ParConv (uint8_t* input_L1, uint8_t* weight_L1, uint8_t* output_L1)
{
    core_id = get_core_id();
    apply data parallel convolution
}
```

GAP Optimized SW Basic Kernel
Given L1, L2, L3 memory constraints

- **Compute the best tile size for any tensors**
  - Aim at reducing memory latency overhead
  - Based on layer type and memory constraints
    - ML/Signal Processing data traffic is predictable at compile time…

- **Produce an optimized code!**
  - Interleaving memory transfers with computation calls

- **Allocate static and temporary data on the more suitable (L3/L2) memory level**
  - Based on liveness analysis of tensors
How to handle Quantization with the GAPflow

Quantization means (in practice) estimating the quantization range \([r_{\text{min}}, r_{\text{max}}]\) and the number of bits of any network tensor (activations and weights).
Inside NNTool

NNtool

.tflite file

TensorFlowLite → IR graph description → Code Generation → AT Model
Inside NNTool

- **.tflite file**
  - TensorFlow Lite
- **Dataset**
- **IR graph description**
- **Code Generation**
- **AT Model**
  - Graph Execution
  - GAP-like functional conv kernels
  - Automatic model quantization
  - Validation on a test dataset
  - Model Debug
  - Code Generation
GAP8 – A complete solution for embedded machine learning at the very edge

GAP AutoTiler
NNTool

GAPFlow toolchain for embedded ML development
GAP8 – A complete solution for embedded machine learning at the very edge

- PMSIS API
- RTOS (FreeRTOS, PULPOS, Zephyr)
- SOC Simulator
- RISC-V GCC

GCC Based toolchain
PC SoC Simulator
Variety of different RTOS’s
PMSIS API unifies API across RTOS’s

GAPFlow toolchain for embedded ML development
GAP8 – A complete solution for embedded machine learning at the very edge

- RISC-V 8 + 1 core MCU
- ISA Extensions
- Fine grained parallelism
- Application Boards

- PMSIS API
- RTOS
  - FreeRTOS, PULPOS, Zephyr
- SOC Simulator
- RISC-V GCC

- GAP AutoTiler
- NNTool

- GAPFlow toolchain for embedded ML development

GCC Based toolchain
PC SoC Simulator
Variety of different RTOS’s
PMSIS API unifies API across RTOS’s
Application Reference Platform – Occupancy Management

Complete reference platform for battery operated occupancy management sensor

- GAPPoc B Reference Board including Lynred ThermEye™ IR camera
- People detection neural network
  - 80 x 80 IR Image - LynRED ThermEye
  - Image preprocessing + human detection
  - 62ms ~4.4mW / frame / second
  - 99% accuracy on internally collected training set.
  - A full solution for people counting / occupancy detection on a battery for > 5 years

All provided under permissive Open Source License
Can be used to evaluate technology in real life situations
The Neural Network Menu is a collection of software that implements Neural Networks on Greenwaves Application Processors (GAP). This repository contains common mobile and edge NN architecture examples, NN sample applications and full flagged reference designs.

**ingredients**
- Image Classification Networks (several versions of Mobilenet V1, V2, V3 minimalistic, full V3 to come)
- kws (Google Keyword Spotting)
- Mobilenet V1 from Pytorch Model

**starters**
- Body Detection (SSD w/ custom CNN backbone)
- Face Detection (SSD w/ custom CNN backbone)
- People Spotting (NN from MIT Visual Wakeup Words)
- Vehicle Spotting (Customization and embedding of a deep learning pipeline for visual object spotting)

**main courses**
Full flagged applications (aka reference designs) running on GAPoC series boards.
- ReID (on GAPoC A)
- Occupancy Management (on GAPoC B)
The GAP story has just begun

GAP9 recently announced

- Rich Audio Interfaces
- Innovative development tools
- Sophisticated audio hardware acceleration
- State of the art NN/AI acceleration
- Fully programmable & Flexible

Ready to enable market leading differentiation in audio hearables

Samples Q2 2021 Production Q4 2021
Thank you!

Manuele Rusci

manuele.rusci@greenwaves-technologies.com

https://greenwaves-technologies.com/
Copyright Notice

This presentation in this publication was presented as a tinyML® Talks webcast. The content reflects the opinion of the author(s) and their respective companies. The inclusion of presentations in this publication does not constitute an endorsement by tinyML Foundation or the sponsors.

There is no copyright protection claimed by this publication. However, each presentation is the work of the authors and their respective companies and may contain copyrighted material. As such, it is strongly encouraged that any use reflect proper acknowledgement to the appropriate source. Any questions regarding the use of any materials presented should be directed to the author(s) or their companies.

tinyML is a registered trademark of the tinyML Foundation.

www.tinyML.org