tinyML® Talks

Enabling Ultra-low Power Machine Learning at the Edge

"CFU Playground: Customize Your ML Processor for Your Specific TinyML Model"

Tim Callahan - Google

January 11, 2022

TINY ML

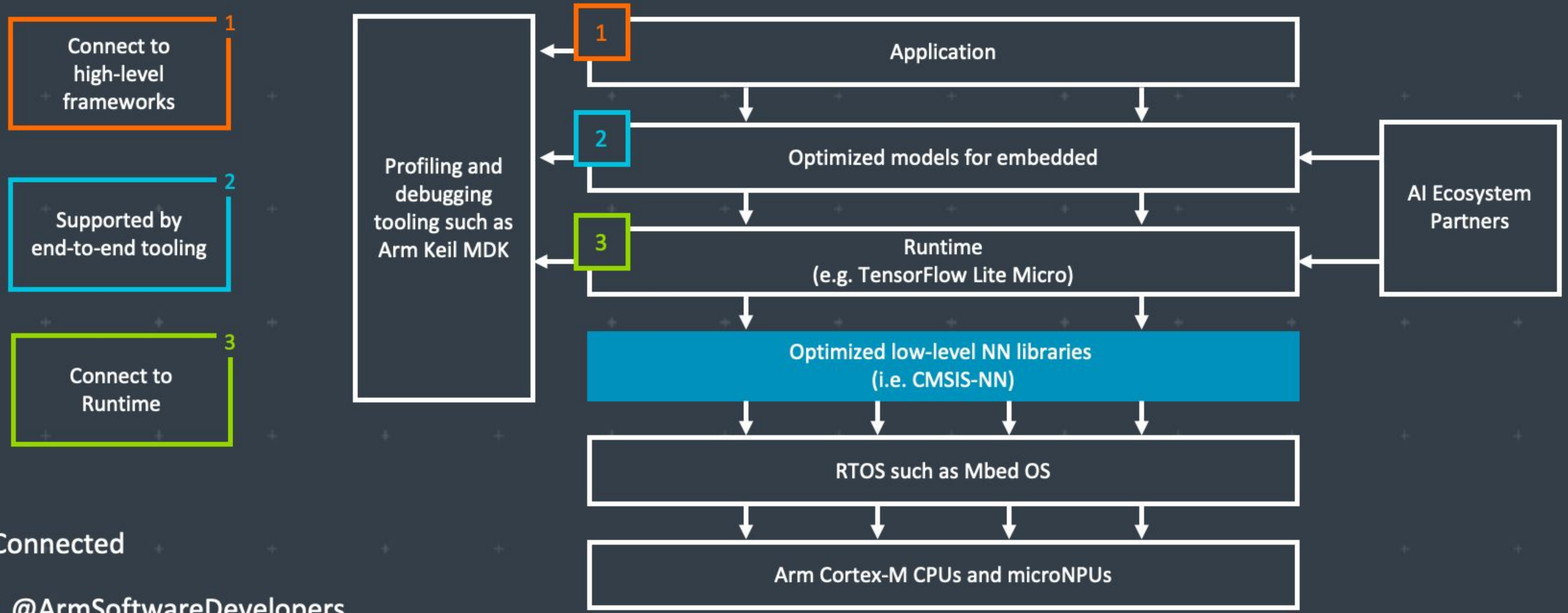
www.tinyML.org

tinyML Talks Strategic Partners

Additional Sponsorships available – contact Olga@tinyML.org for info

Executive Strategic Partners

Arm: The Software and Hardware Foundation for tinyML



Stay Connected

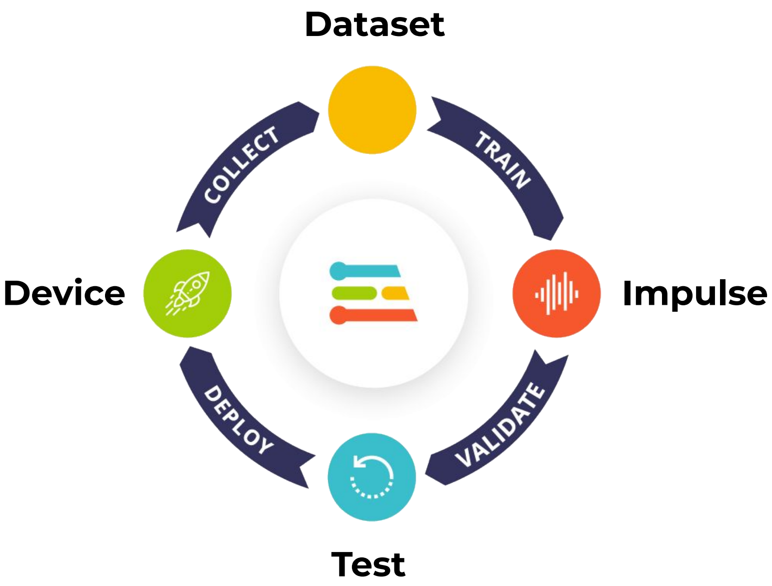
@ArmSoftwareDevelopers

@ArmSoftwareDev

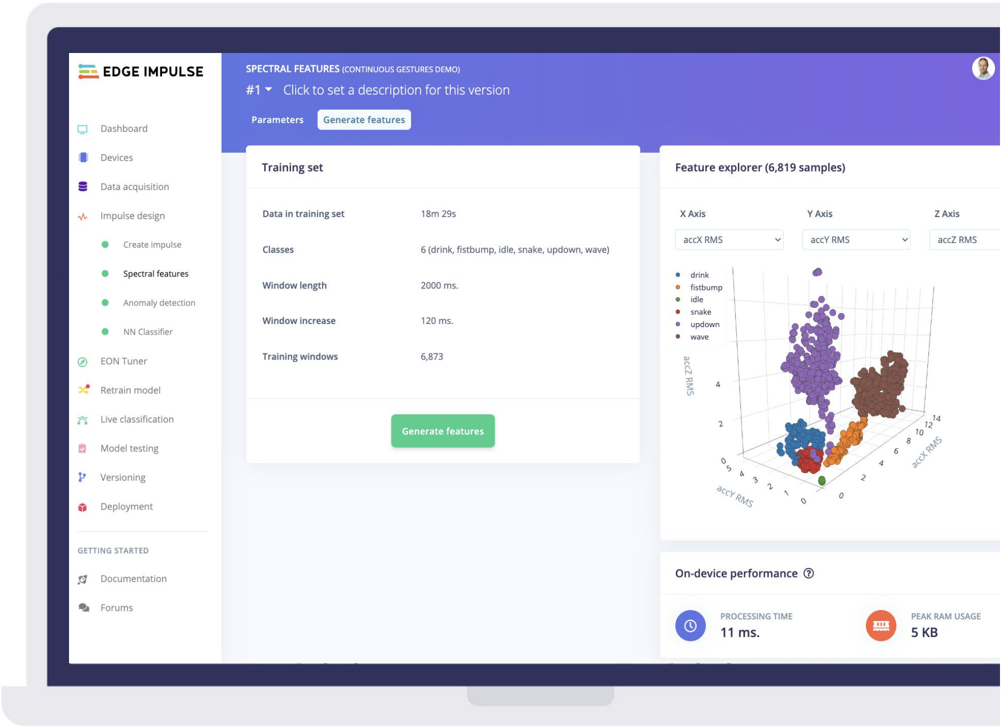
Resources: developer.arm.com/solutions/machine-learning-on-arm

arm

EDGE IMPULSE The leading edge ML platform



www.edgeimpulse.com



Qualcomm AI research

Advancing AI research to make efficient AI ubiquitous

Power efficiency

Model design, compression, quantization, algorithms, efficient hardware, software tool

Personalization

Continuous learning, contextual, always-on, privacy-preserved, distributed learning

Efficient learning

Robust learning through minimal data, unsupervised learning, on-device learning

A platform to scale AI across the industry

Perception

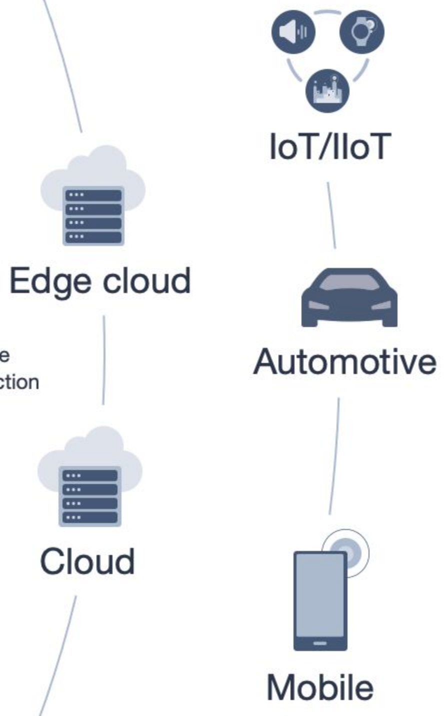
Object detection, speech recognition, contextual fusion

Reasoning

Scene understanding, language understanding, behavior prediction

Action

Reinforcement learning for decision making



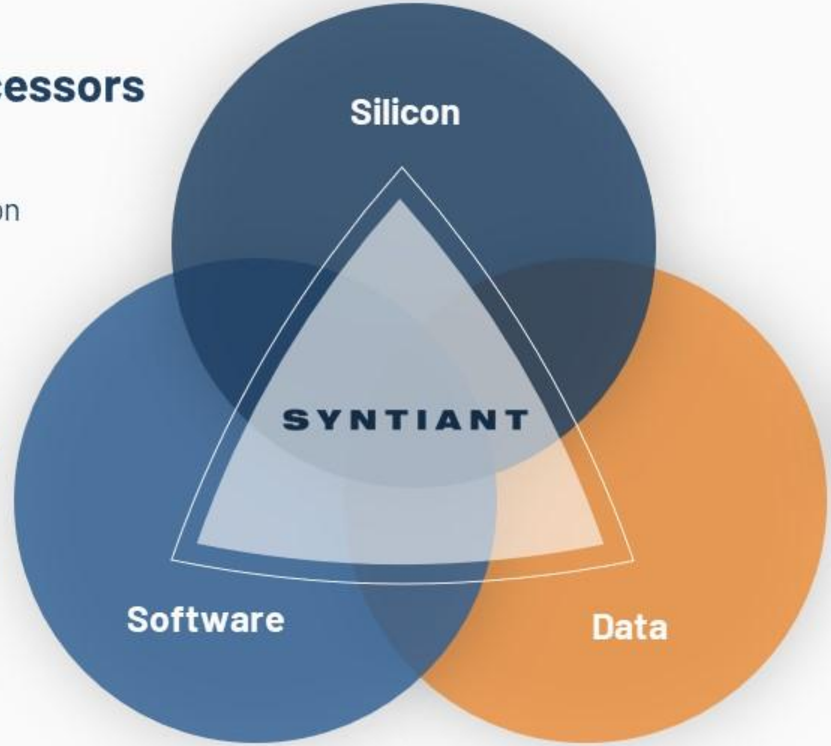
SYNTIANT

Neural Decision Processors

- At-Memory Compute
- Sustained High MAC Utilization
- Native Neural Network Processing

ML Training Pipeline

- Enables Production Quality Deep Learning Deployments



End-to-End Deep Learning Solutions

for

TinyML & Edge AI

Data Platform

- Reduces Data Collection Time and Cost
- Increases Model Performance

partners@syntiant.com

www.syntiant.com

Platinum Strategic Partners

Deeplite

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**

APPLY NOW

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

BECOME BETA USER **bit.ly/testdeeplite**

mobilityXlab arm AI 100 CBINSIGHTS

TINY ML TALKS webcast

KLIKA·TECH

GLOBAL IOT SOLUTIONS

BROAD AND SCALABLE EDGE COMPUTING PORTFOLIO

Microcontrollers & Microprocessors

Arm® Core

- **RENESAS RA** — Arm® Cortex®-M 32-bit MCUs
  Arm ecosystem, Advanced security, Intelligent IoT
- **RENESAS RZ** — Arm®-based High-end 32 & 64-bit MPUs
  High-resolution HMI, Industrial network & real-time control
- **RENESAS RE** — Arm® Cortex®-M0+ Ultra-low Power 32-bit MCUs
  Innovative process tech (SOTB), Energy harvesting
- **Renesas Synergy™** — Arm®-based 32-bit MCUs for Qualified Platform
  Qualified software and tools

Renesas Core

- **RENESAS RL78** — Ultra-low Energy 8 & 16-bit MCUs
  Bluetooth® Low Energy, SubGHz, LoRa®-based Solutions
- **RENESAS RX** — High Power Efficiently 32-bit MCUs
  Motor control, Capacitive touch, Functional safety, GUI
- **RENESAS RH850** — 40nm/28nm process Automotive 32-bit MCUs
  Rich functional safety and embedded security features

Core technologies

AI

A broad set of high-power and energy-efficient embedded processors

Digital & Analog & Power Solution

Winning Combinations that combine our complementary product portfolios

Security & Safety

Comprehensive technology and support that meet the industry's stringent standards

Cloud Native

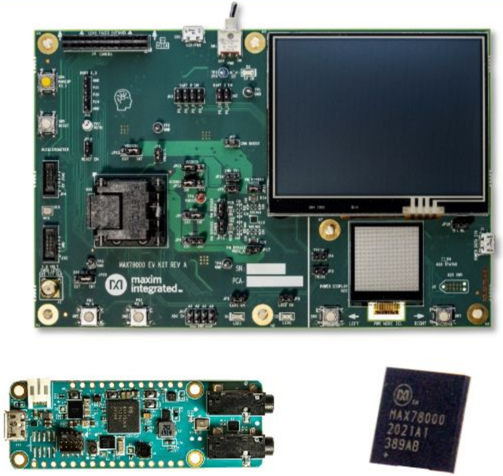
Cross-platforms working with partners in different verticals and organizations

Gold Strategic Partners

maxim integrated

Maxim Integrated: Enabling Edge Intelligence

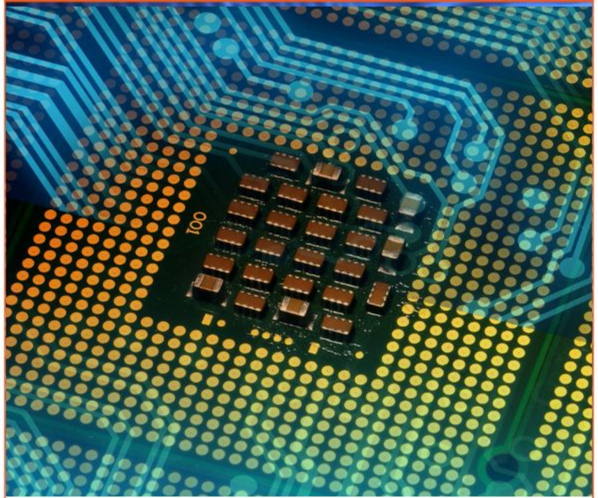
Advanced AI Acceleration IC



The new MAX78000 implements AI inferences at low energy levels, enabling complex audio and video inferencing to run on small batteries. Now the edge can see and hear like never before.

www.maximintegrated.com/MAX78000

Low Power Cortex M4 Micros



Large (3MB flash + 1MB SRAM) and small (256KB flash + 96KB SRAM, 1.6mm x 1.6mm) Cortex M4 microcontrollers enable algorithms and neural networks to run at wearable power levels.

www.maximintegrated.com/microcontrollers

Sensors and Signal Conditioning

Health sensors measure PPG and ECG signals critical to understanding vital signs. Signal chain products enable measuring even the most sensitive signals.

www.maximintegrated.com/sensors

LatentAI

Adaptive AI for the Intelligent Edge

Latentai.com

Micro.ai

seeed studio

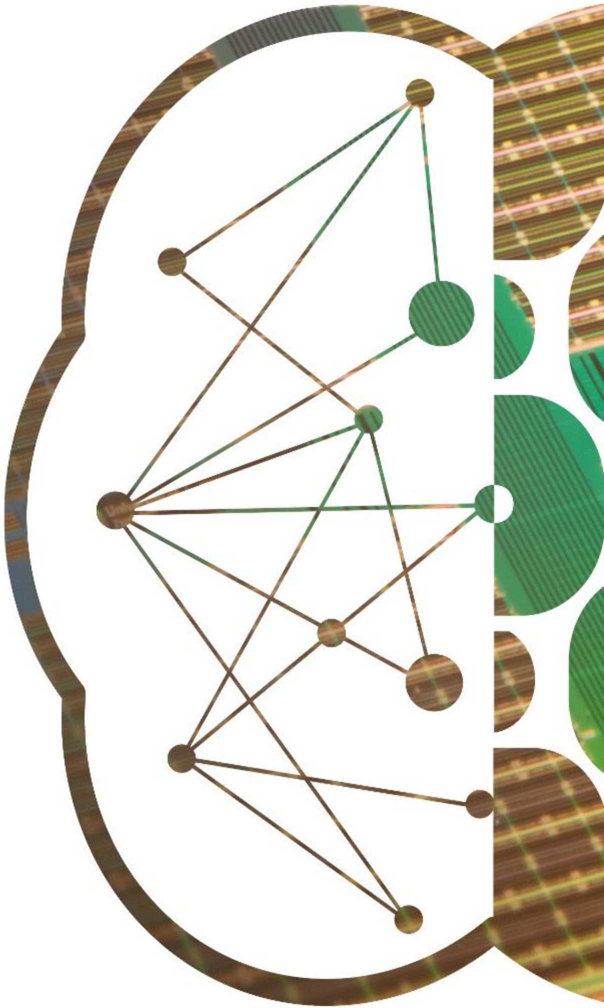
The IoT Hardware Enabler

life.augmented

SynSense

SynSense builds **sensing and inference** hardware for **ultra-low-power** (sub-mW) **embedded, mobile and edge** devices. We design systems for **real-time always-on smart sensing**, for audio, vision, IMUs, bio-signals and more.

https://SynSense.ai



Silver Strategic Partners

tinyML Summit 2022

Miniature dreams can come true…

March 28-30, 2022

Hyatt Regency San Francisco Airport

https://www.tinyml.org/event/summit-2022/

*The Best Product of the Year and the Best Innovation of the Year awards are open for nominations between **November 15 and February 28.***

tinyML Research Symposium 2022

March 28, 2022

https://www.tinyml.org/event/research-symposium-2022

Join Growing tinyML Communities:

7.7k members in
40 Groups in 33 Countries

tinyML - Enabling ultra-low Power ML at the Edge

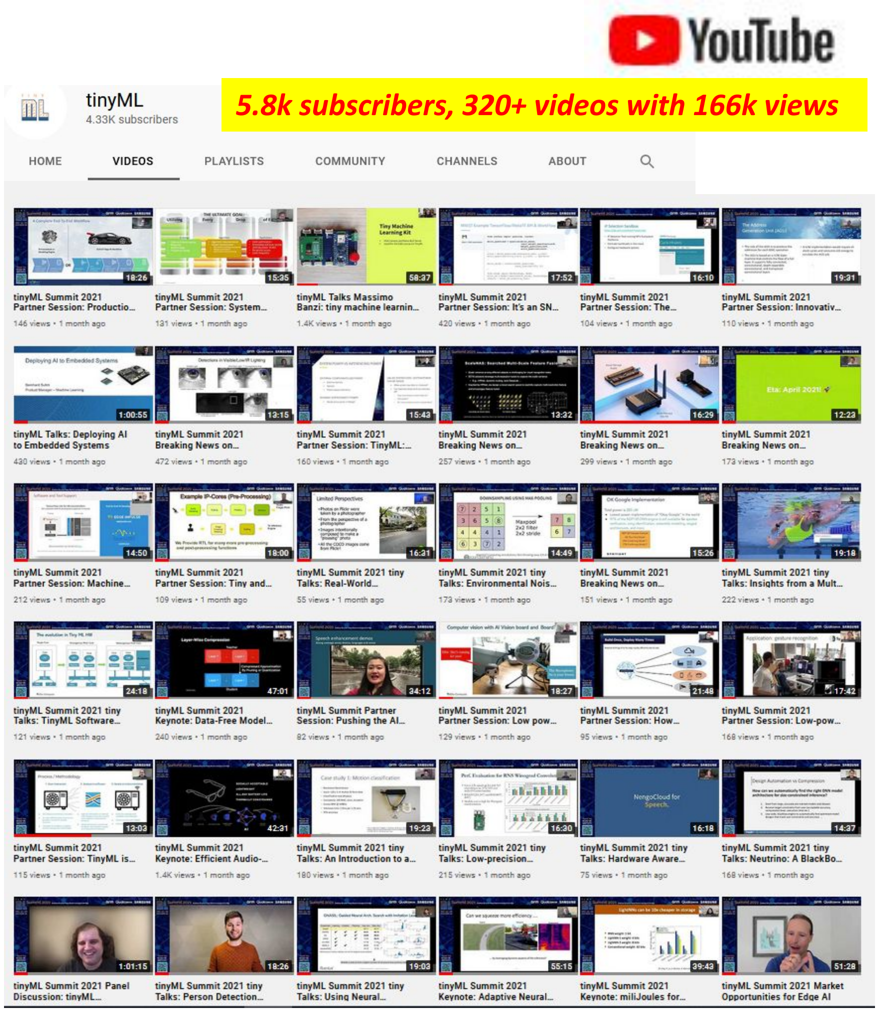
https://www.meetup.com/tinyML-Enabling-ultra-low-Power-ML-at-the-Edge/

2.5k members
&
4.3k followers

The tinyML Community

https://www.linkedin.com/groups/13694488/

5.8k subscribers, 320+ videos with 166k views



Subscribe to
tinyML YouTube Channel
for updates and notifications
(including this video)

www.youtube.com/tinyML

Next tinyML Talks

Date	Presenter	Topic / Title
Tuesday, January 18	Ashutosh Pandey, Infineon Technologies	Exploring techniques to build efficient and robust TinyML deployments

Webcast start time is 8:00 am Pacific time

Please contact talks@tinyml.org if you are interested in presenting

Reminders

Slides & Videos will be posted tomorrow

tinyml.org/forums youtube.com/tinyml

Please use the Q&A window for your questions

Tim Callahan

Tim Callahan works at Google with the open source FPGA toolchain (SymbiFlow) team. His work is to help make FPGA development more accessible, fun, and rewarding. His research interests include anything that involves optimizing the hardware/software boundary. He has degrees from UC Berkeley, Cambridge University, and the University of Minnesota.

CFU Playground

Custom
Function
Unit

Customize Your ML Processor for
Your Specific TinyML Model

TinyML.org Talk
January 11, 2022
Tim Callahan, tcal@google.com presenting the work of many

Disclaimer: NOT an official Google project

It's on GitHub: Fork it, fix it, send a PR!

Acknowledgements

Googlers: Alan Green, David Lattimore, Dan Callaghan,
Tim Ansell,
Interns Rachel Sugrono & Joey Bushagour

TFLM (Pete Warden and team)

Antmicro

Harvard: Prof Vijay Reddi, Shvetank Prakash, Colby Banbury

Open source: VexRiscv , LiteX, SymbiFlow, Yosys, Nextpnr,
VTR, Migen, nMigen, Renode, Verilator, OpenOCD, ..

Open Source Showcase!

ML library	TensorFlow Lite	-- *open source*
CPU ISA	RISC-V	-- *open*
CPU design	VexRiscv	-- *open source*
FPGA SoC/IP	LiteX	-- *open source*
FPGA synth/PnR	SymbiFlow,Yosys, Nextpnr, VPR	-- *open source*

- FPGA synth/PnR
    - FPGA vendor tools can be used if you wish

Python HW gen	Migen, nMigen	-- *open source*
Simulation	Renode, Verilator	-- *open source*

- The only proprietary component is the FPGA itself

Benefits of Open Source

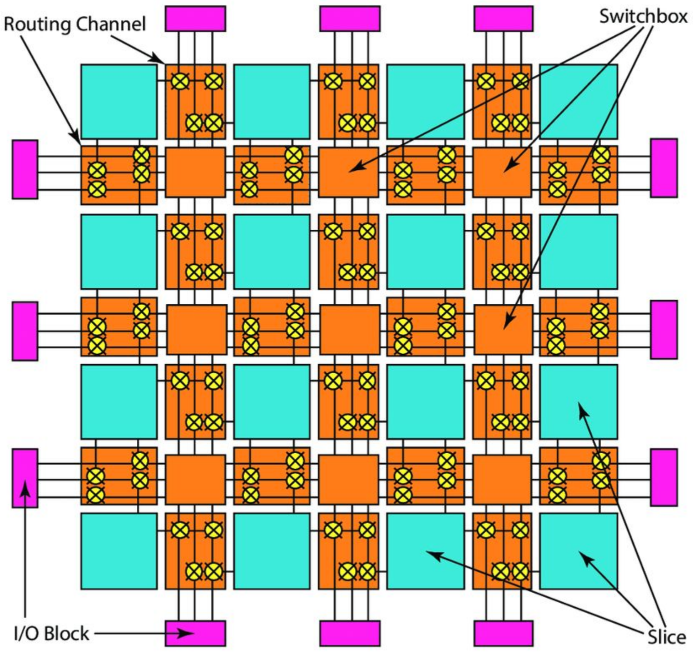
- No licensing fees
- No license headaches in your build system
- You can fix bugs and address shortcomings
- No vendor lock-in
- Transparency -- open for inspection, both sw & hw
  - Do you trust a third-party blob of sw or hw integrated with your product?
    → see betrusted.io
    "At its core lies not a CPU, but an FPGA, so that users are empowered to build their processors from scratch and know there are no flaws or backdoors in the architecture that could lead to security compromises."

CFU Playground Key Ideas

- **Specialize** the entire stack for your particular model – both the ML kernels AND the processor

- **Make it easy to get started; allow quick iterations → rapid iterative design space exploration**

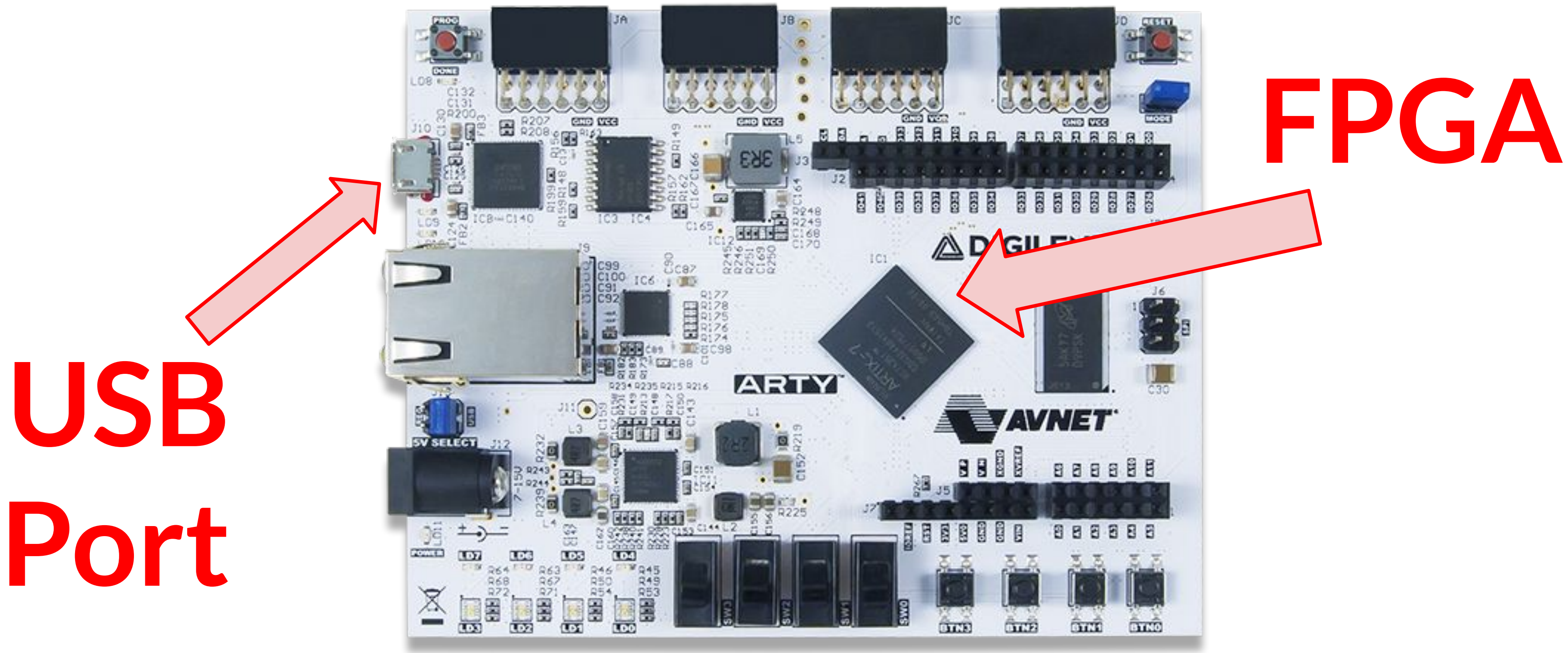
FPGA – programmable hardware



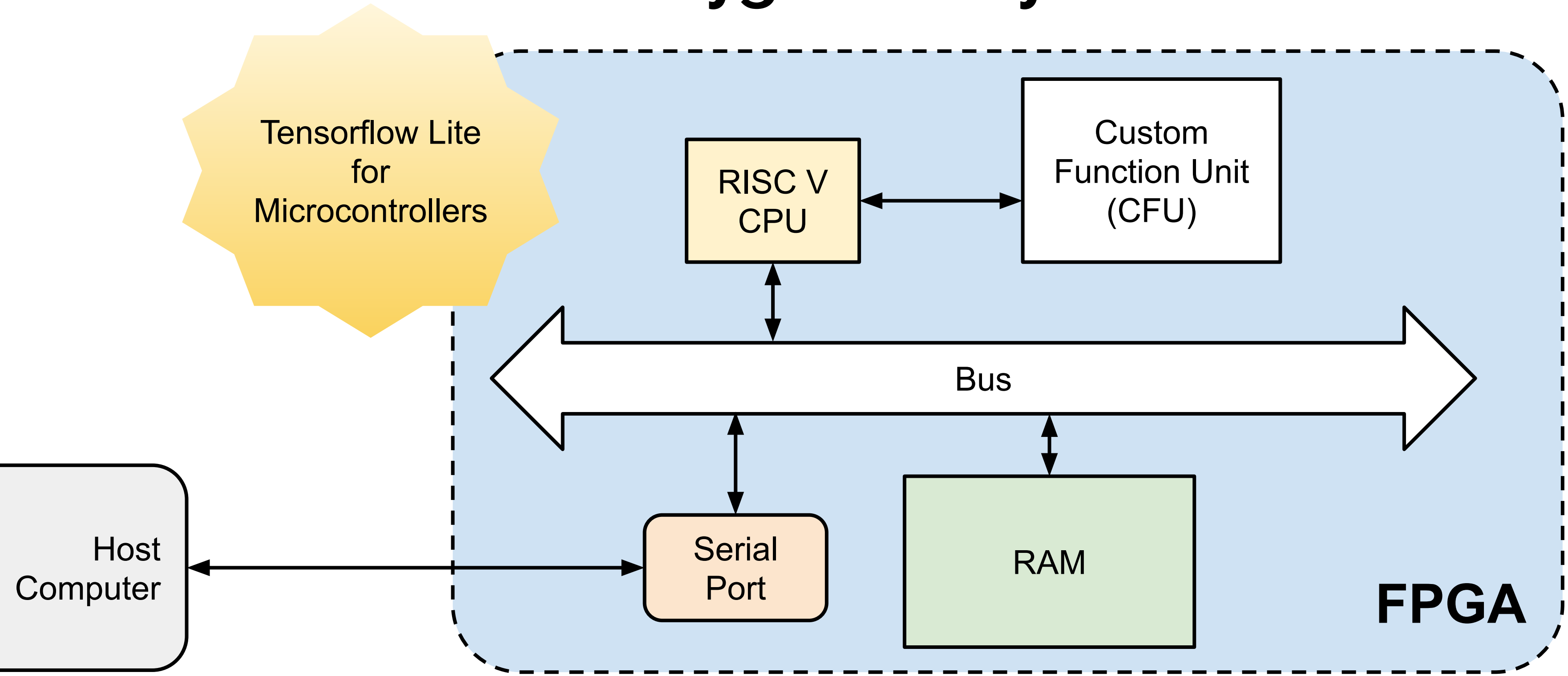
A "bitstream" or "configuration"

- determines the function of each configurable logic block
- and sets switches in the interconnect to connect them to each other and to I/O

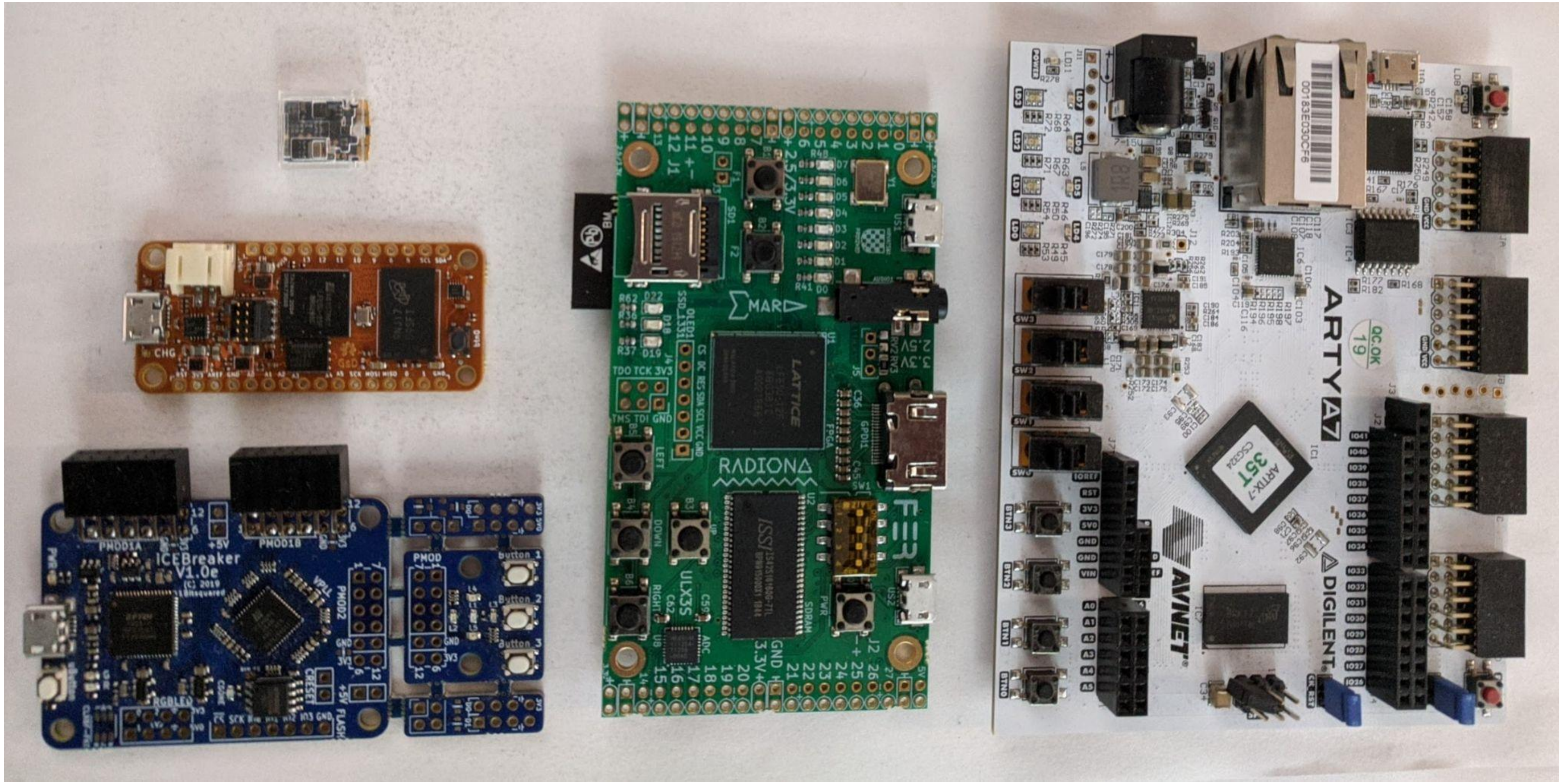
Image source: "Parallel Programming for FPGAs", Ryan Kastner, Janarbek Matai, Stephen Neuendorffer (CC 4.0)


USB
Port
FPGA
ARTY
AVNET

CFU Playground System Architecture



CFU Playground -- some of the tested boards



Summary

Jobs

- setup-matrix
- test-projects (proj_template, ...
- test-projects (proj_template, ...
- test-projects (proj_template, ...
- test-projects (proj_template, ...
- test-projects (proj_template, ...
- test-projects (proj_template, ...
- test-projects (proj_template, ...
- test-projects (proj_template, ...
- test-projects (proj_template, ...
- test-projects (proj_template, ...
- test-projects (proj_template, ...
- test-projects (proj_template, ...

Name	Size
test-results-example_cfu-common_soc-1bitsquared_icebreaker	511 KB
test-results-example_cfu-common_soc-antmicro_lpddr4_test_board	511 KB
test-results-example_cfu-common_soc-camlink_4k	512 KB
test-results-example_cfu-common_soc-colorlight_i5	511 KB
test-results-example_cfu-common_soc-decklink_mini_4k	511 KB
test-results-example_cfu-common_soc-decklink_quad_hdmi_recorder	511 KB
test-results-example_cfu-common_soc-digilent_arty	511 KB
test-results-example_cfu-common_soc-digilent_arty_s7	511 KB
test-results-example_cfu-common_soc-digilent_genesys2	511 KB

CFU Playground Prerequisites:

- Linux host
- You will use:
  - git
  - C++ (mostly C)
  - Hardware design – Verilog or ???
    - unless you just want to experiment with CPU configuration

Chisel

nMigen (Python)

XLS

SystemVerilog

- A supported FPGA board
  - or you could just simulate with Renode or Verilator

CFU Playground Uses

- **Deploy** a soft CPU+CFU on FPGA for tinyML
  - Firmware updates include a new ML model, new software, *and* a new CPU+CFU
- **Prototype** a custom RISC-V-based ASIC
  - Yes you! efabless.com MPW will fab your open-source chip
- **Learn** about ML software, hardware, and performance
- **Research** new ML approaches
  - while co-designing the hardware to support it

The Hardware Lottery

- Sara Hooker's observation that the success of new ML approaches depends on their compatibility with downstream software and hardware

- Here you can "make your own luck"!

RISC-V and CFU Basics

The RISCV Add Instruction

```
00720C33    add x24,x4,x7    # x24 = x4 + x7
```

The RISCV Add Instruction

```
00720C33        add x24,x4,x7     # x24 = x4 + x7
```

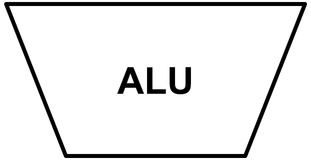
0000000	00111	00100	000	11000	0110011

The RISCV Add Instruction

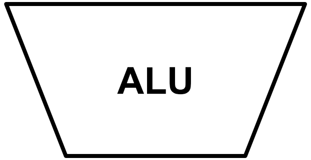
```
00720C33      add x24,x4,x7     # x24 = x4 + x7
```

0000000	00111	00100	000	11000	0110011
					opcode
					ALU

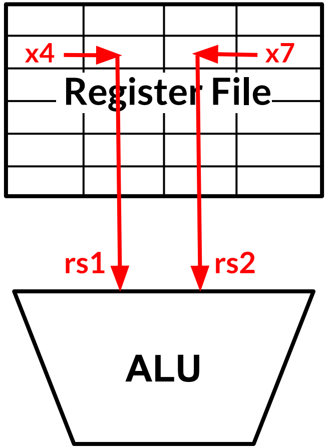
0000000	00111	00100	000	11000	0110011
					opcode
					ALU



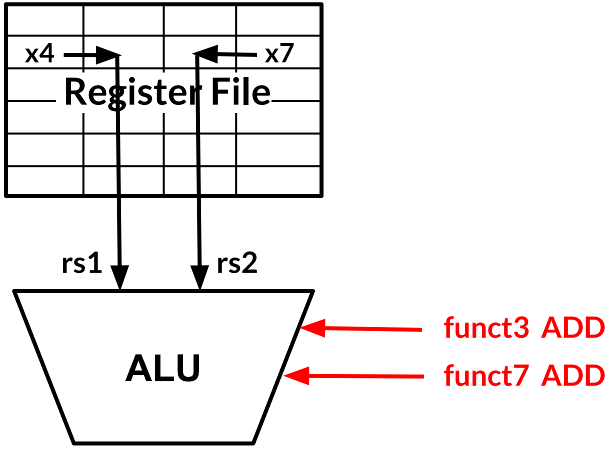
0000000	00111	00100	000	11000	0110011
funct7	**rs2**	**rs1**	**funct3**	**rd**	**opcode**
ADD	x7	x4	**ADD**	x24	**ALU**



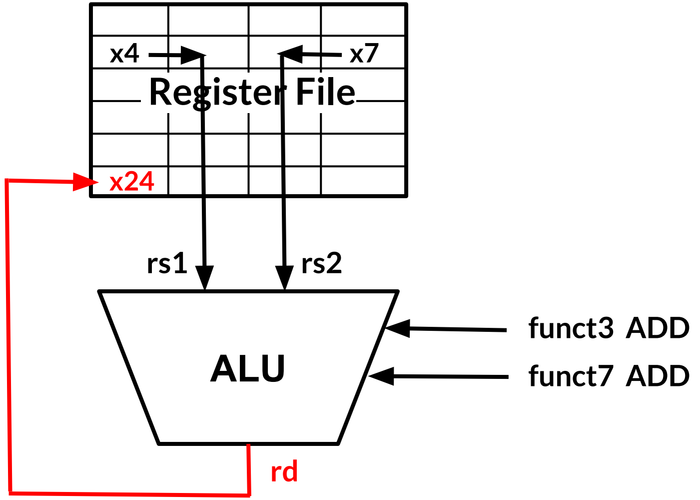
0000000	00111	00100	000	11000	0110011
funct7	rs2	rs1	funct3	rd	opcode
ADD	x7	x4	ADD	x24	ALU



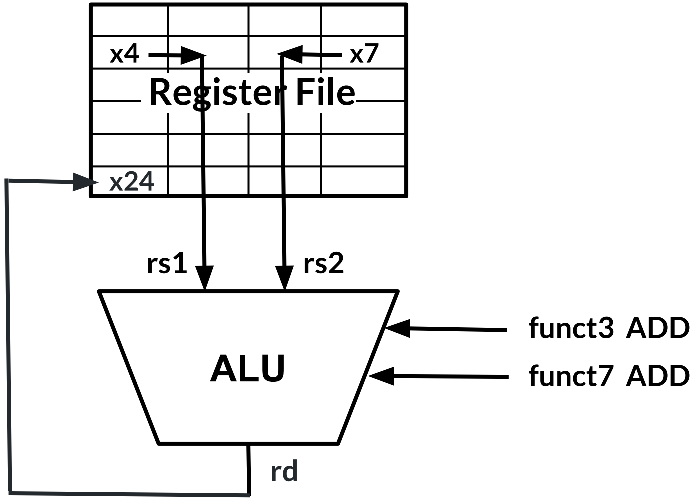
0000000	00111	00100	000	11000	0110011
funct7	rs2	rs1	funct3	rd	opcode
ADD	x7	x4	ADD	x24	ALU



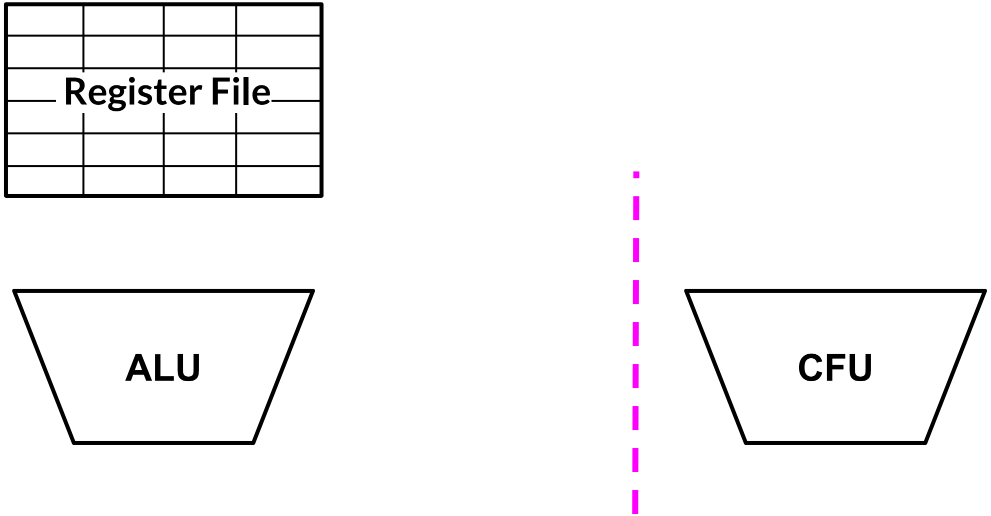
0000000	00111	00100	000	11000	0110011
funct7	rs2	rs1	funct3	rd	opcode
ADD	x7	x4	ADD	x24	ALU



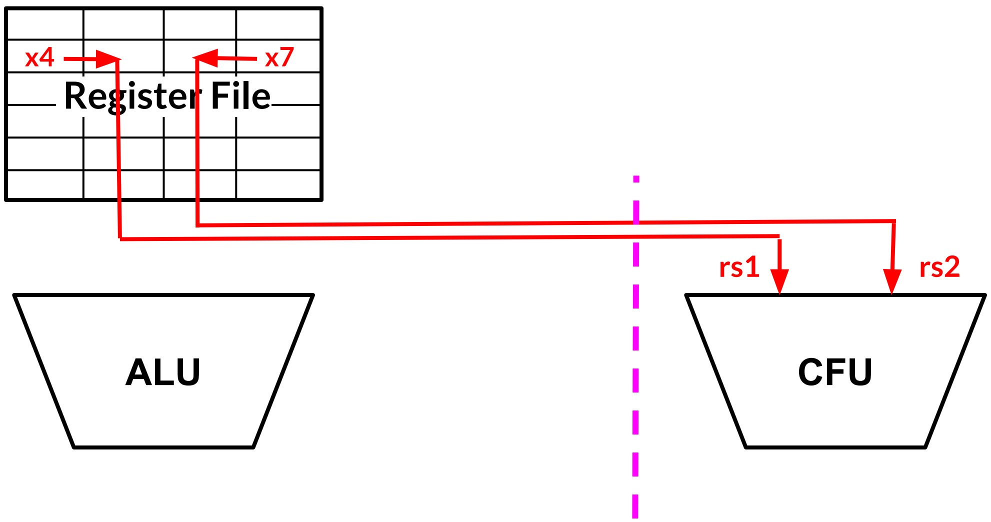
0000000	00111	00100	000	11000	0110011
funct7	**rs2**	**rs1**	**funct3**	**rd**	**opcode**
ADD	x7	x4	**ADD**	x24	**ALU**



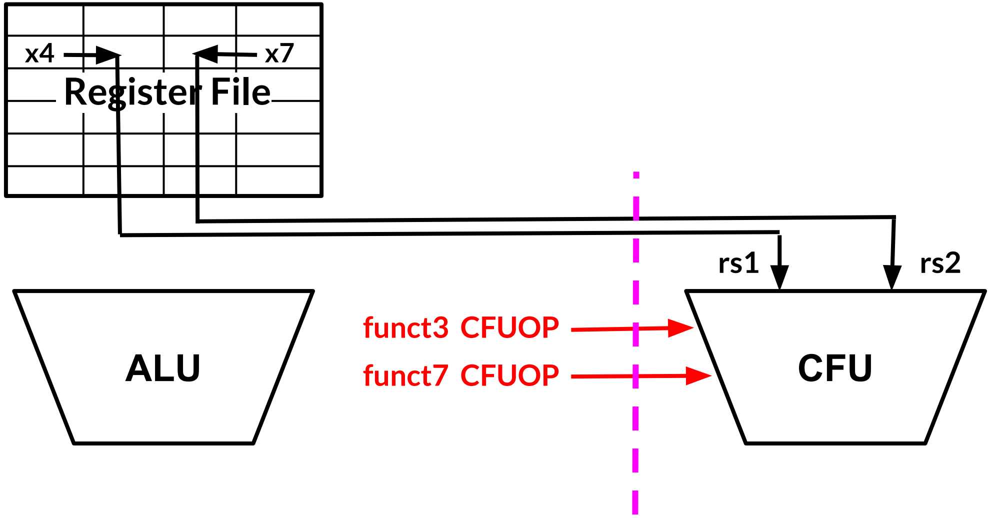
0000000	00111	00100	000	11000	0001011
					opcode
					CUSTOM



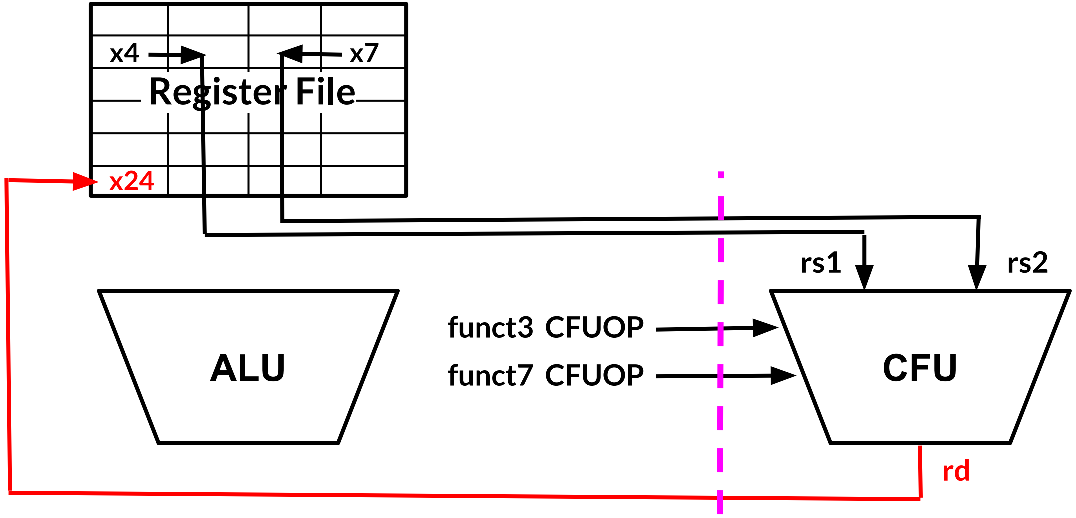
0000000	00111	00100	000	11000	0001011
funct7	rs2	rs1	funct3	rd	opcode
CFUOP	x7	x4	CFUOP	x24	CUSTOM



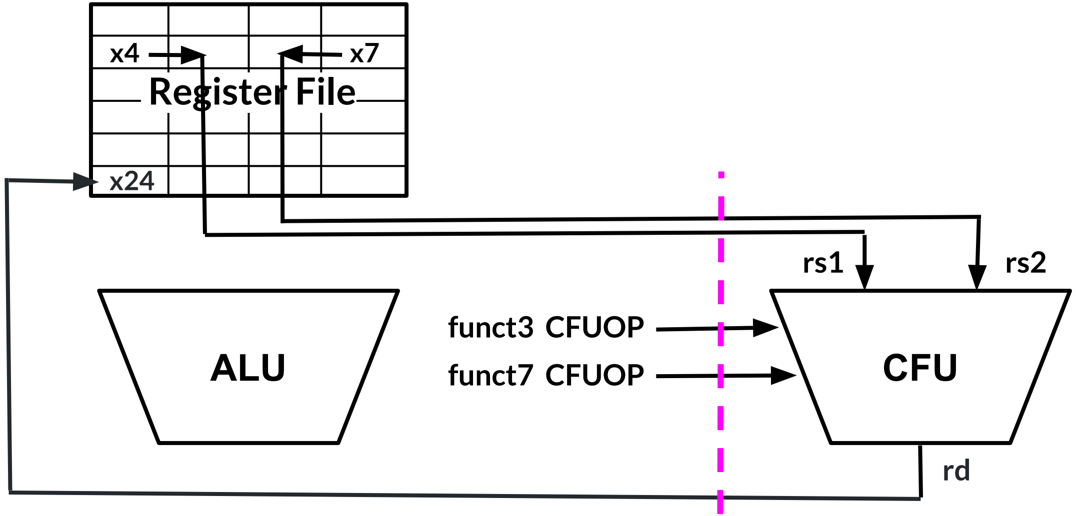
0000000	00111	00100	000	11000	0001011
funct7	rs2	rs1	funct3	rd	opcode
CFUOP	x7	x4	CFUOP	x24	CUSTOM



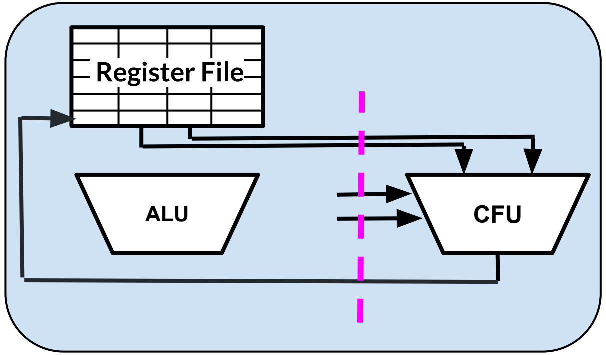
0000000	00111	00100	000	11000	0001011
funct7	rs2	rs1	funct3	rd	opcode
CFUOP	x7	x4	CFUOP	x24	CUSTOM



0000000	00111	00100	000	11000	0001011
funct7	**rs2**	**rs1**	**funct3**	**rd**	**opcode**
CFUOP	x7	x4	**CFUOP**	x24	**CUSTOM**



CFU details



- Two operands from the regfile, one result written back
- Multiple/variable cycles; can refuse new inputs while working
  → pipelined or iterative computation
- NO direct connection between CFU & the memory hierarchy
- CFU cannot otherwise affect CPU state (branch etc.)
- CFU can contain state -- registers and memories that persist
- CFU can contain its own sequencer
- One CFU, multiple instructions that can access the shared state

So how do we use the CFU?

- Do we build a nifty compiler?   NO!
- YOU are the compiler..it’s up to you to insert uses of your new instructions into the code
- After all, you’re the one who just designed the new instruction to speed up your code, so you know exactly where it will be used

How to use the CFU from software

- Access the new instruction using function call syntax:

```
rslt = cfu_op(funct3, funct7, op1, op2);
```

Compile-time constants

C / C++ variables / expressions

How to use the CFU from software

- Access the new instruction using function call syntax:

```
rslt = cfu_op(funct3, funct7, op1, op2);
```

Compile-time constants (funct3, funct7)

C / C++ variables / expressions (op1, op2)

```
rslt = cfu_op(0, 1, op1, op2);
```

How to use the CFU from software

- Access the new instruction using function call syntax:

```
rslt = cfu_op(funct3, funct7, op1, op2);
```

Compile-time constants (funct3, funct7)

C / C++ variables / expressions (op1, op2)

```
#define HAMMING_DISTANCE(x,y) cfu_op(0, 1, (x), (y));

rslt = HAMMING_DISTANCE(op1, op2);
```

How to use the CFU from software

- Custom instruction macros intermix with regular C code:

```
t1 = *x;
t2 = cfu_op(0, 0, t1, b);
t3 = cfu_op(1, 0, t2, b);
*x = t3;
```

Compiled and disassembled:

```
400001a0:    00812783    lw          a5,8(sp)
400001a4:    00d7878b    cfu[0,0]    a5, a5, a3
400001a8:    00d7978b    cfu[0,1]    a5, a5, a3
400001ac:    00f12423    sw          a5,8(sp)
```

Objdump can’t disassemble the custom instructions,
so we wrote a helper script..

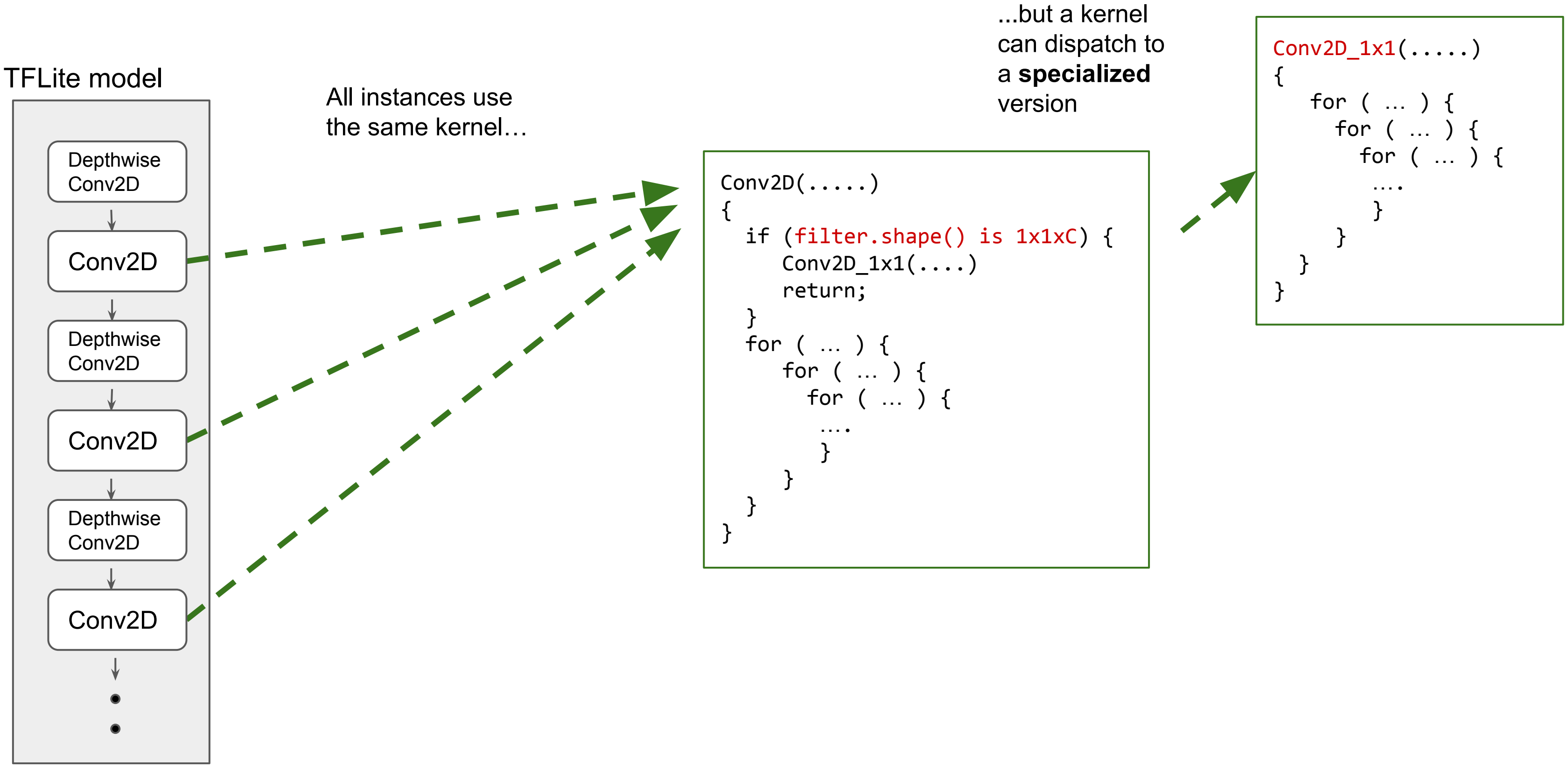
No overhead!

Why use a CFU approach for tinyML?

- With ML inference, the hot spots are small & very hot
- A SMALL amount of custom hardware to exploit the bit-level flexibility of FPGA can accelerate a LARGE fraction of the runtime
- Leave complexity, setup, outer loops in SW
- Iterative approach: easy to take first step, then next step, …
- In our experience, the CFU will incrementally grow to become *almost* a full-blown "accelerator"

TensorFlow Lite for Microcontrollers (TFLM)

TensorFlow Lite for Microcontrollers



CFU Playground

CFU Playground Build

```
$ cd proj/proj_template_v/          # establish project
$ make prog           3.5min        # build gateware using ./cfu.v
                                      and put the bitstream on the FPGA
$ make -j8 software   <1min         # build software w/ local overrides
$ make load                         # load and execute the software
```

...then interact with the software running on the board

```
--============== Liftoff! ================--
Hello, World!

CFU Playground
==============
 1: TfLM Models menu
 2: Functional CFU Tests
 3: Project menu
 4: Performance Counter Tests
 5: TFLite Unit Tests
 6: Benchmarks
 7: Util Tests
main> 1

Running TfLM Models menu

TfLM Models
===========
 1: Person Detection int8 model
 2: Micro Speech
 3: MLCommons Tiny V0.1 Keyword Spotting
 x: eXit to previous menu
models> 3
```

```
Tests for kws model
===================
 0: Run with "down" input
 1: Run with "go" input
 2: Run with "left" input
 g: Run golden tests (check for expected outputs)
 x: eXit to previous menu
kws> g

Running Run golden tests (check for expected outputs)
...

Copied 490 bytes at 0x400dd590
Running kws
.............
"Event","Tag","Ticks"
0,CONV_2D,19841
1,DEPTHWISE_CONV_2D,4306
2,CONV_2D,11690
3,DEPTHWISE_CONV_2D,4226
4,CONV_2D,11662
5,DEPTHWISE_CONV_2D,4230
6,CONV_2D,11050
7,DEPTHWISE_CONV_2D,4757
8,CONV_2D,11905
9,AVERAGE_POOL_2D,237
10,RESHAPE,3
11,FULLY_CONNECTED,16
12,SOFTMAX,26
Perf counters not enabled.
    86M (    85996663) cycles total
OK   Golden tests passed
```

CFU Playground Development

- Start a new project by copying a template:
  - `cp -r  proj/proj_template  proj/my_first_project`
- Choose or bring the TFLite model
- Use built-in profiling to identify time-consuming TF ops
- Look at tensor shapes and other parameters,
  find opportunities to specialize/simplify the kernel(s)
- Then, look for CFU acceleration opportunities
  - Design CFU, alter TFLM kernels to use the custom instructions
- Measure speedup, repeat!

Examples

Simple Example

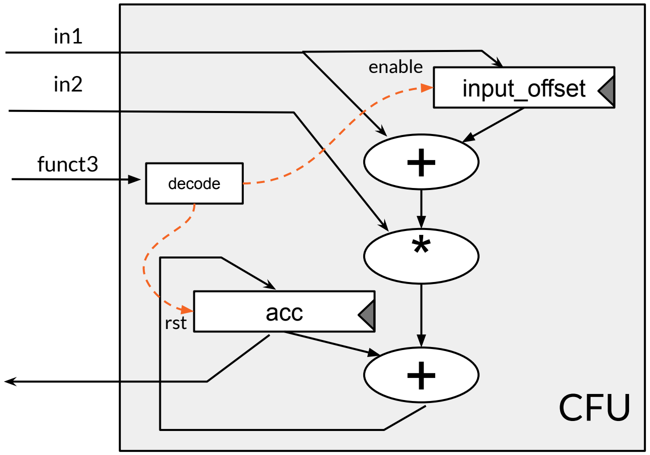
This Conv2D kernel consumes 76% of the execution time with model “pdti8”, so go after the computation in the innermost loop.

```
const int32_t input_offset = params.input_offset;  // r = s(q - Z)

for (int batch = 0; batch < batches; ++batch) {
  for (int out_y = 0; out_y < output_height; ++out_y) {
    const int in_y_origin = (out_y * stride_height) - pad_height;
    for (int out_x = 0; out_x < output_width; ++out_x) {
      const int in_x_origin = (out_x * stride_width) - pad_width;
      for (int out_channel = 0; out_channel < output_depth; ++out_channel) {
        int32_t acc = 0;
        for (int filter_y = 0; filter_y < filter_height; ++filter_y) {
          const int in_y = in_y_origin + dilation_height_factor * filter_y;
          for (int filter_x = 0; filter_x < filter_width; ++filter_x) {
            const int in_x = in_x_origin + dilation_width_factor * filter_x;

            // Zero padding by omitting the areas outside the image.
            const bool is_point_inside_image =
                (in_x >= 0) && (in_x < input_width) && (in_y >= 0) &&
                (in_y < input_height);

            if (!is_point_inside_image) {
              continue;
            }

            for (int in_channel = 0; in_channel < input_depth; ++in_channel) {
              int32_t input_val = input_data[Offset(input_shape, batch, in_y,
                                                    in_x, in_channel)];
              int32_t filter_val = filter_data[Offset(
                  filter_shape, out_channel, filter_y, filter_x, in_channel)];

              acc += filter_val * (input_val + input_offset);
            }
          }
        }
        (use acc)
```

Simple Example

Loop invariant

acc += filter_val * (input_val + input_offset)



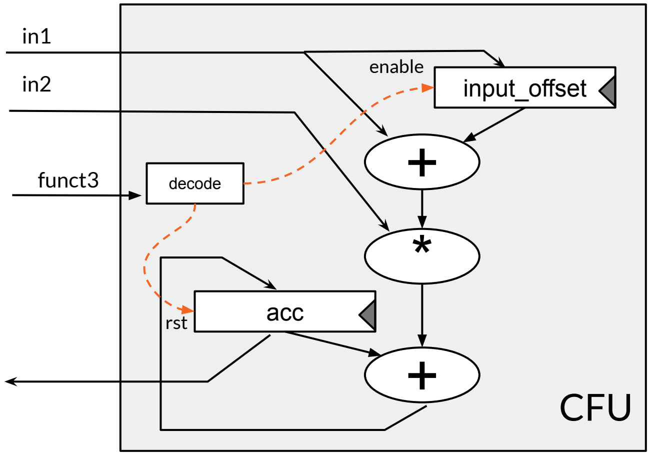
Simple Example

```
//
// The CFU computation
//
reg signed [31:0] input_offset;      // state
reg signed [31:0] acc;               // state

wire signed [31:0] filt_val = in1;
wire signed [31:0] input_val = in2;

wire signed [31:0] newacc = acc + (filt_val * (input_val + input_offset) );
assign out = acc;

always @(posedge clk) begin
  if (cmd_valid) begin
    if (funct3 == 3'b000) begin
      input_offset <= in1;
    end else if (funct3 == 3'b001) begin
      acc <= in1;
    end else if (funct3 == 3'b010) begin
      acc <= newacc;
    end
  end
end
```

Verilog!

Loop invariant

acc += filter_val * (input_val + input_offset)



Simple Example

31% cycle reduction for CONV_2D
24% cycle reduction overall
(model “pdti8”)

```
const int32_t input_offset = params.input_offset;  // r = s(q - Z)
// CFU: copy input_offset into the CFU
cfu_op(0, 0, input_offset, 0);

for (int batch = 0; batch < batches; ++batch) {
  for (int out_y = 0; out_y < output_height; ++out_y) {
    const int in_y_origin = (out_y * stride_height) - pad_height;
    for (int out_x = 0; out_x < output_width; ++out_x) {
      const int in_x_origin = (out_x * stride_width) - pad_width;
      for (int out_channel = 0; out_channel < output_depth; ++out_channel) {

        //int32_t acc = 0;
        // CFU: set the CFU internal acc to ZERO
        cfu_op(1, 0, 0, 0);

        for (int filter_y = 0; filter_y < filter_height; ++filter_y) {
          const int in_y = in_y_origin + dilation_height_factor * filter_y;
          for (int filter_x = 0; filter_x < filter_width; ++filter_x) {
            const int in_x = in_x_origin + dilation_width_factor * filter_x;

            ...

            for (int in_channel = 0; in_channel < input_depth; ++in_channel) {
              int32_t input_val = input_data[Offset(input_shape, batch, in_y,
                                                   in_x, in_channel)];
              int32_t filter_val = filter_data[Offset(
                  filter_shape, out_channel, filter_y, filter_x, in_channel)];

              // acc += filter_val * (input_val + input_offset);
              // CFU: add-multiply-accumulate in the CFU
              cfu_op(2, 0, filter_val, input_val);
            }
          }
        }

        // CFU: retrieve final acc value from the CFU
        int32_t acc = cfu_op(3, 0, 0, 0);
```

Simple Example

Add aliases for the custom instructions, for readability

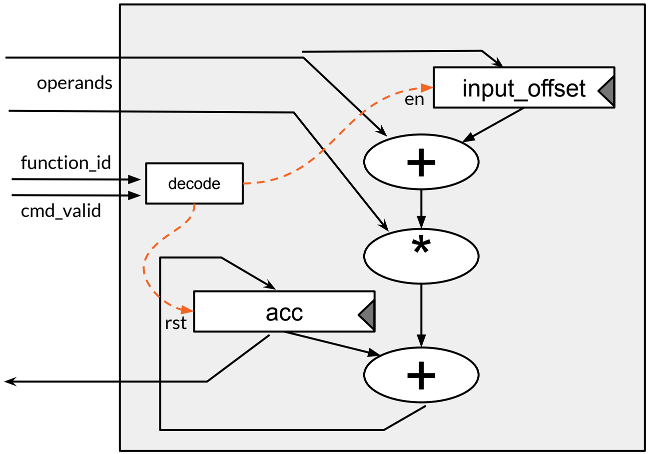
31% cycle reduction for CONV_2D
24% cycle reduction overall
(model “pdti8”)

```
const int32_t input_offset = params.input_offset;  // r = s(q - Z)
// CFU: copy input_offset into the CFU
cfu_init_offset(input_offset);

for (int batch = 0; batch < batches; ++batch) {
  for (int out_y = 0; out_y < output_height; ++out_y) {
    const int in_y_origin = (out_y * stride_height) - pad_height;
    for (int out_x = 0; out_x < output_width; ++out_x) {
      const int in_x_origin = (out_x * stride_width) - pad_width;
      for (int out_channel = 0; out_channel < output_depth; ++out_channel) {

        //int32_t acc = 0;
        // CFU: set the CFU internal acc to ZERO
        cfu_clear_acc();

        for (int filter_y = 0; filter_y < filter_height; ++filter_y) {
          const int in_y = in_y_origin + dilation_height_factor * filter_y;
          for (int filter_x = 0; filter_x < filter_width; ++filter_x) {
            const int in_x = in_x_origin + dilation_width_factor * filter_x;

            ...

            for (int in_channel = 0; in_channel < input_depth; ++in_channel) {
              int32_t input_val = input_data[Offset(input_shape, batch, in_y,
                                                   in_x, in_channel)];
              int32_t filter_val = filter_data[Offset(
                  filter_shape, out_channel, filter_y, filter_x, in_channel)];

              // acc += filter_val * (input_val + input_offset);
              // CFU: add-multiply-accumulate in the CFU
              cfu_macc_with_offset(filter_val, input_val);
            }
          }
        }

        // CFU: retrieve final acc value from the CFU
        int32_t acc = cfu_get_acc();
```

Typical CFU evolution

Loop invariant

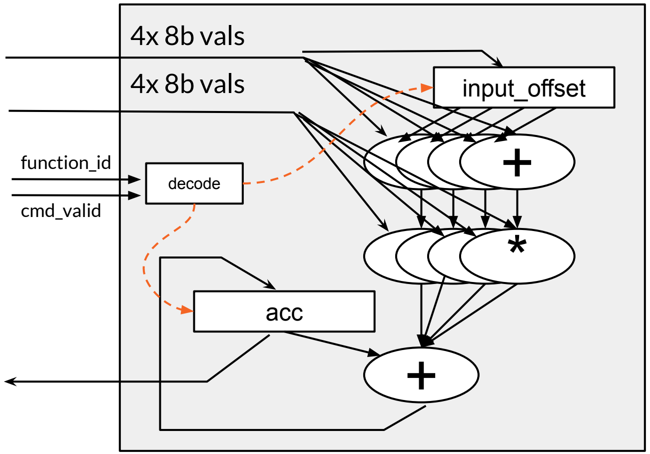
acc += filter_val * (input_val + input_offset)



Typical CFU evolution

Loop invariant

acc += filter_val * (input_val + input_offset)

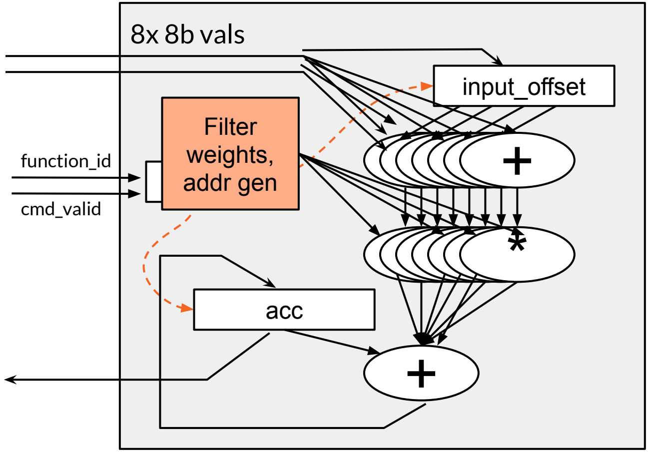


Exploit SIMD

Typical CFU evolution

Loop invariant

acc += filter_val * (input_val + input_offset)



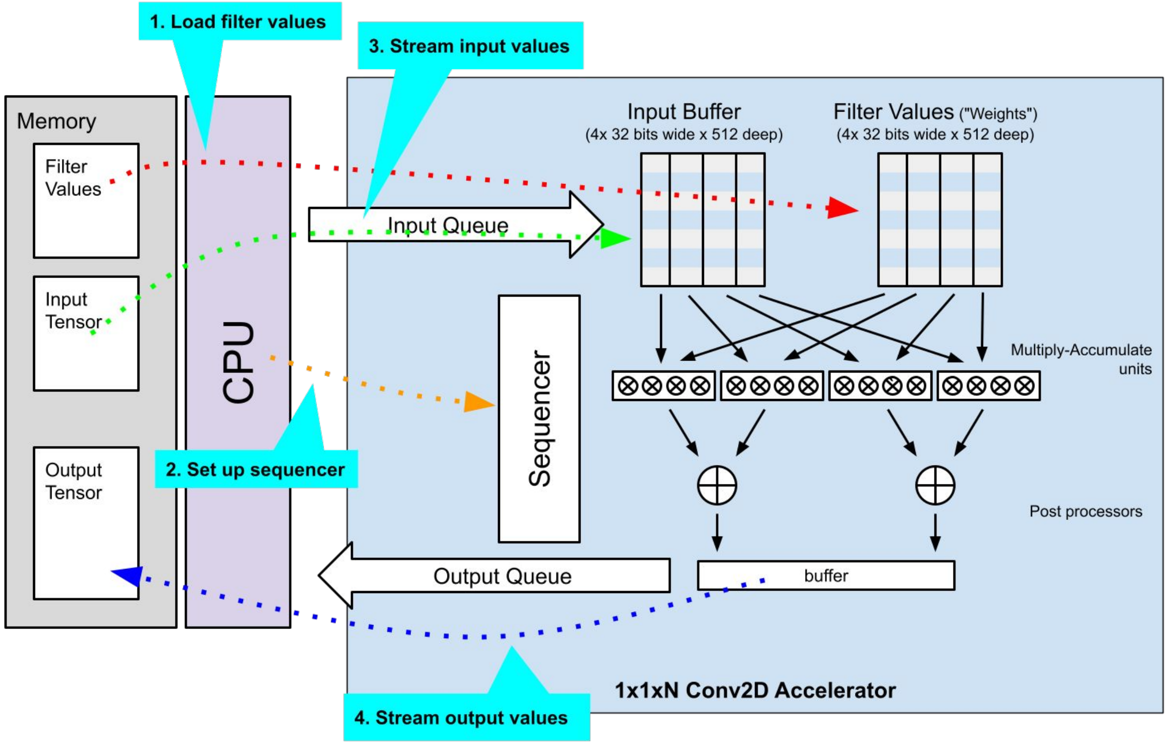
Move re-used data (filter weights) into the CFU

Typical CFU evolution

And after a few more iterations, you've added:

- Local memories for both input_vals and filter_vals
- Increased MACC parallelism
- Sequencer for complete matrix x vector computation
- Activation function & scaling
- Double buffering at input and output buffers to pipeline data transfer and computation

Typical CFU evolution

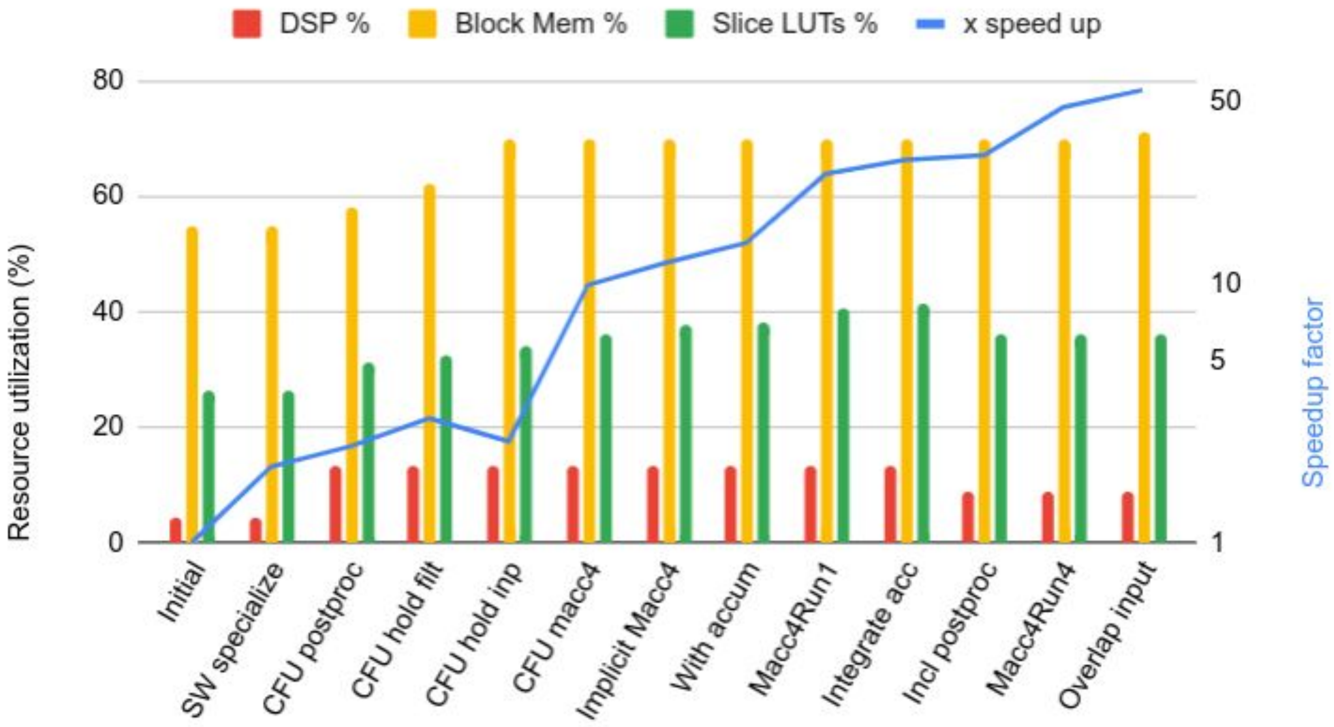


MobileNet v2

Before speeding up conv_2d (1x1):

```
Totals

             SOFTMAX        11510   (0.0M)
             RESHAPE        21887   (0.0M)
     FULLY_CONNECTED        48284   (0.0M)
     AVERAGE_POOL_2D       487497   (0.5M)
                 ADD      4564330   (4.6M)
                 MUL      7236662   (7.2M)
                 SUB     16517877   (16.5M)
          CONV_2D_3x3    95241303   (95.2M)
   DEPTHWISE_CONV_2D    197214007   (197.2M)
          CONV_2D_1x1   559817289   (559.8M)
```

MobileNet v2



55x speedup!
(on just conv2d 1x1)

But Amdahl's law… overall speedup is just 2.8x

MobileNet v2

After speeding up conv_2d (1x1):

```
Totals

          SOFTMAX        11503   (0.0M)
          RESHAPE        21886   (0.0M)
  FULLY_CONNECTED        50183   (0.1M)
  AVERAGE_POOL_2D       494322   (0.5M)
              ADD      4577562   (4.6M)
              MUL      7233115   (7.2M)
       CONV_2D_1x1    10263213   (10.3M)
              SUB     16517493   (16.5M)
       CONV_2D_3x3    76621863   (76.6M)
DEPTHWISE_CONV_2D    199485469   (199.5M)
```

Originally
560M

Example #2

Keyword Spotting on Fomu

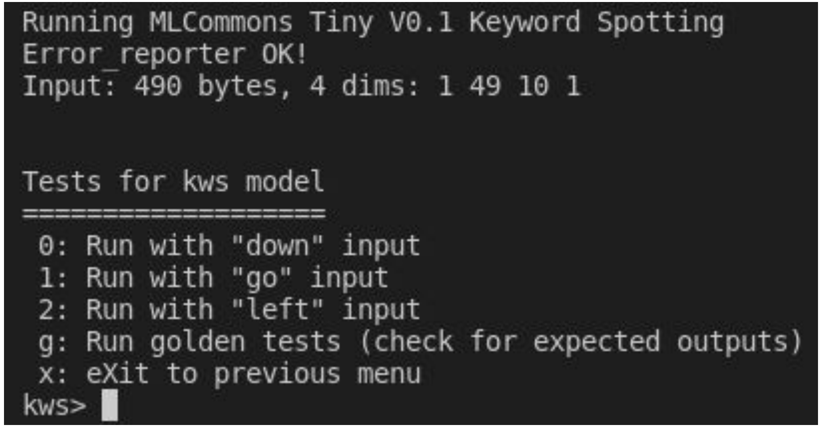
About ~~my~~ Intern Joey Bushagour’s project

- Adding support for tiny FPGAs in CFU-Playground
- Fitting TensorFlow Lite for Microcontrollers on tiny FPGAs
- Making inference on tiny FPGAs faster

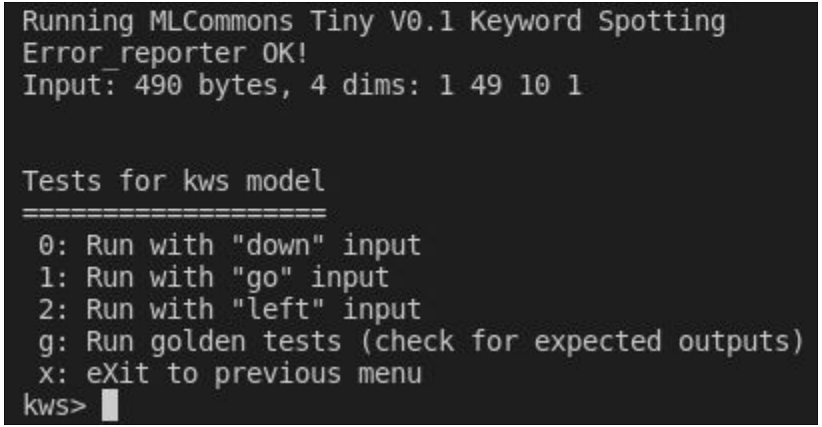
fomu.im

75× speedup on model inference

How it ~~started~~:

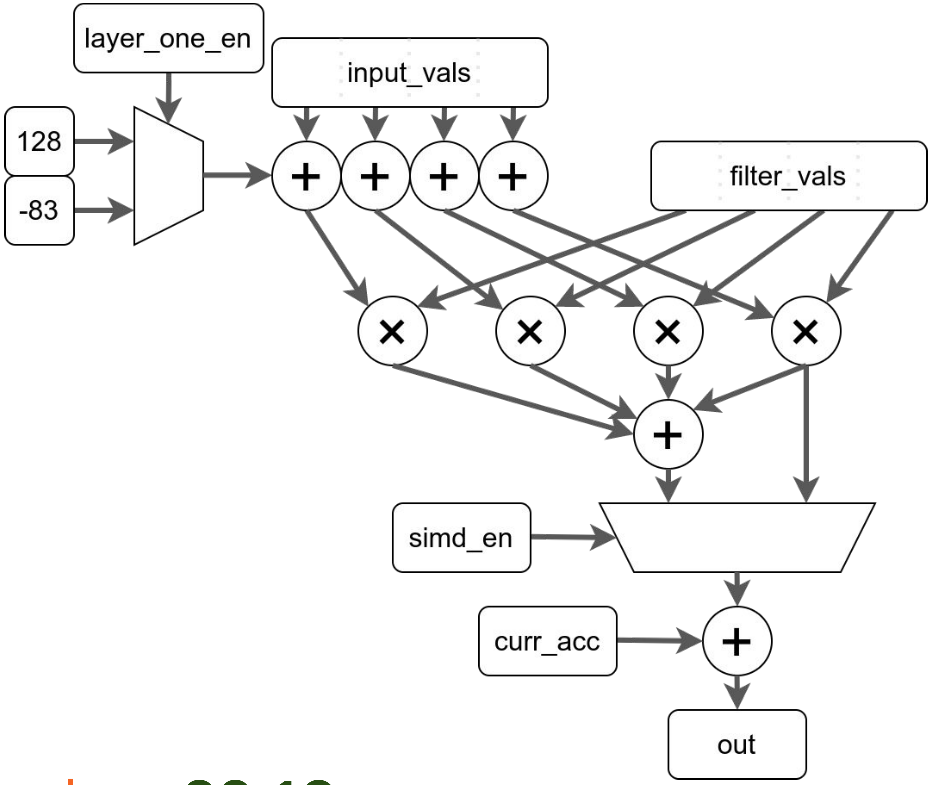


How it's going:

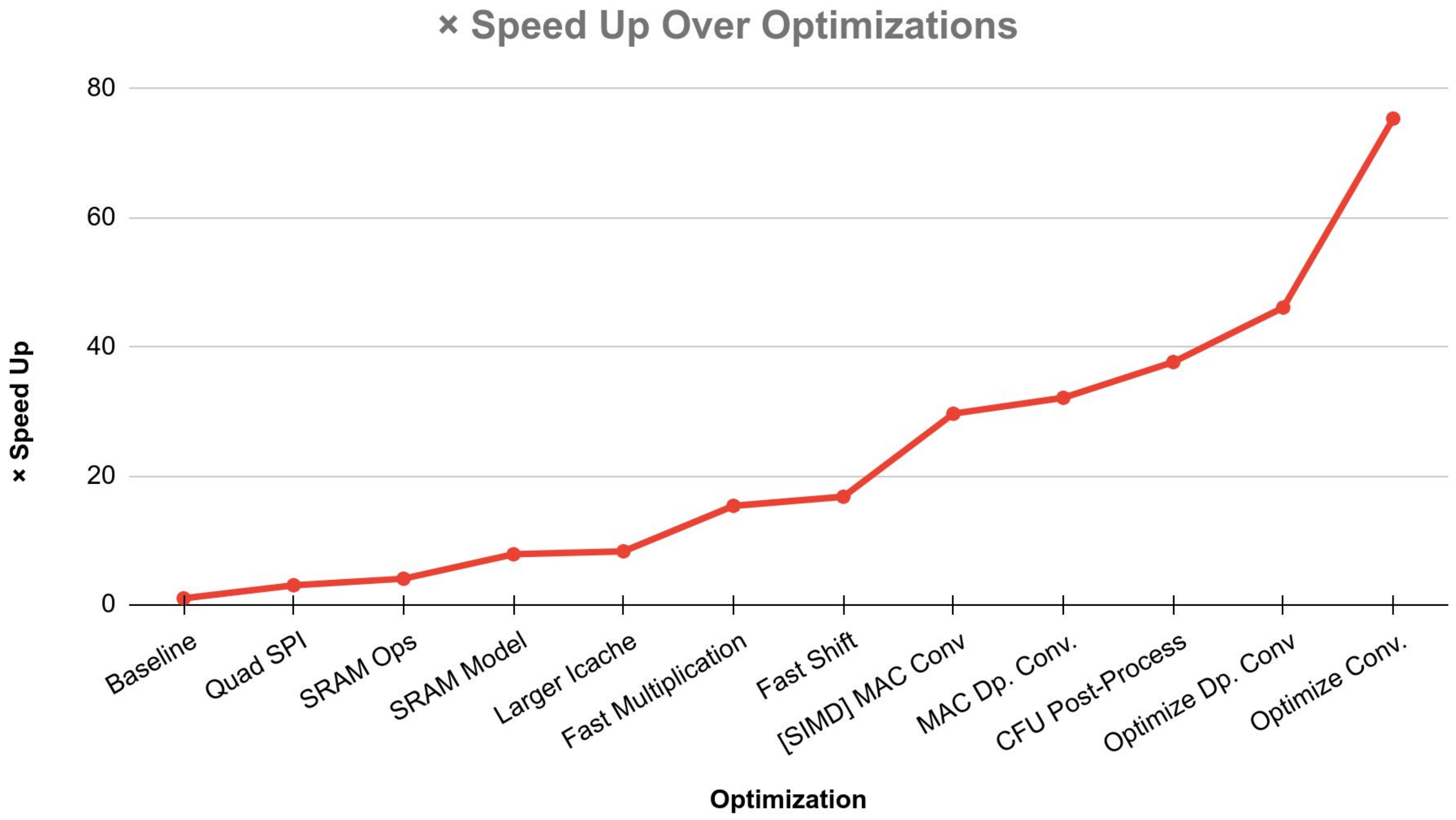


Optimization: Use CFU SIMD MAC in TFLM ops

- Convolution and depthwise convolution are mostly multiply and accumulate
- Sometimes these 8 bit multiply and accumulates are contiguous in memory
- Our registers are 32 bits wide, we can vectorize where possible



Cumulative speedup: **32.10x**


× Speed Up Over Optimizations
80
60
40
20
0
× Speed Up
Baseline
Quad SPI
SRAM Ops
SRAM Model
Larger Icache
Fast Multiplication
Fast Shift
[SIMD] MAC Conv
MAC Dp. Conv.
CFU Post-Process
Optimize Dp. Conv
Optimize Conv.
Optimization

What's left

- We have 75× speedup
- If we could find room to fit a branch predictor and bypass we'd have 102×
- Optimizing depthwise convolution more

```
Running MLCommons Tiny V0.1 Keyword Spotting
Error_reporter OK!
Input: 490 bytes, 4 dims: 1 49 10 1


Tests for kws model
===================
 0: Run with "down" input
 1: Run with "go" input
 2: Run with "left" input
 g: Run golden tests (check for expected outputs)
 x: eXit to previous menu
kws>
```

Wrap up

Generating hardware from Python?

- Yes! We used nMigen (now “Amaranth HDL”) to build our large CFUs.
- It is a Python library for generating HW
- You still need to understand HW
- But you can use Python conveniences (functions, loops, dictionaries, etc)
- Google “pyconline au 2021 cfu” to find Alan’s presentation on YouTube.

Join In The Fun

Clone it: github.com/google/CFU-Playground

- plenty of sample code and models to accelerate
- works with many LiteX supported boards (**check the wiki!**)

Docs: cfu-playground.readthedocs.io

- introductions to nmigen, step-by-step guides, detailed documentation

Contact us:

- raise a github issue
- mail us: tcal@google.com, avg@google.com
- **chat us**: https://gitter.im/CFU-Playground/community

End

Copyright Notice

www.tinyml.org

Copyright Notice