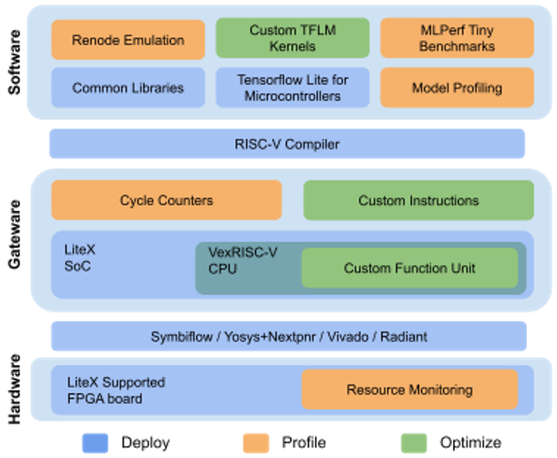
CFU Playground: Full-Stack Open-Source Framework for TinyML Acceleration on FPGAs

Shvetank Prakash* Tim Callahan† Joseph Bushagour§ Colby Banbury*
Alan V. Green† Pete Warden† Tim Ansell† Vijay Janapa Reddi*

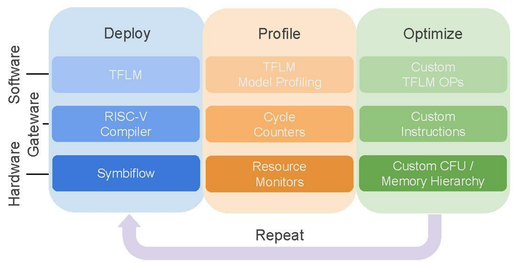
SCAN ME

The Playground



We present CFU Playground, a full-stack open-source framework that enables rapid and iterative design of machine learning (ML) accelerators for tinyML systems. Our toolchain tightly integrates open-source software, RTL generators, and FPGA tools for synthesis, place, and route. This full-stack development framework gives engineers access to explore bespoke architectures that are customized and co-optimized for tinyML. The rapid, deploy-profile-optimization feedback loop lets ML hardware and software developers achieve significant returns out of a relatively small investment in customization. Using CFU Playground's design loop, we show substantial speedups (55×-75×) and design space exploration between the CPU and accelerator.

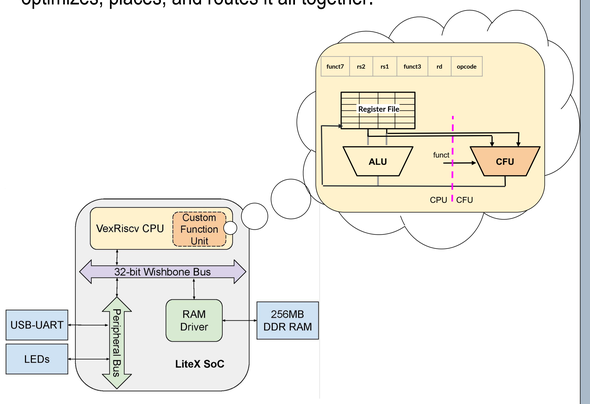
Methodology



CFU Playground enables a deploy→profile→optimize loop that enables iterative, guided optimization of resource-constrained ML systems. The toolchain allows a developer to quickly focus their design effort at any layer of the stack, measure its performance at a fine granularity, implement custom optimizations, and repeat. The figure above highlights components of CFU Playground that fulfill a role in the design cycle across the deployment stack: Software, Gateware, and Hardware. An FPGA platform is assumed to be the final deployment, although this process could also be considered to be a prototyping step for ASIC deployment.

Gateware

The custom function unit (CFU) is a small piece of custom logic added in hardware to extend the CPU's datapath to accelerate a discrete function determined by the developer. It follows the RISC-V R-format in which it receives two operands from the register file and writes one result back. A CFU can support state, multiple custom instructions, and pipelining. A notable feature of this architecture is that the CFU does not have direct memory access. It relies on the CPU to move data back and forth. The merits of a direct CFU-memory connection are being considered and may be added in the future. The boundary between CPU and CFU is strictly logical. The implementation flattens the design and optimizes, places, and routes it all together.



CFU Playground runs a complete System-on-Chip (SoC) on an FPGA to capture the full-stack system effects of accelerating ML models. LiteX provides a convenient and efficient infrastructure to create FPGA soft cores and SoCs. The soft core used in CFU Playground is VexRiscv, an implementation of a RISC-V CPU in SpinalHDL. The design of the VexRiscv is highly configurable, providing the ability to easily plugin or remove many different features for performance and functionality such as pipelining stages, caches, and floating point units. This customization ability lends itself well to enabling the design space exploration of CPU vs. CFU.

Hardware

The gateware for CFU Playground is adaptable to a wide range of FPGA hardware platforms. It can fit on a board as small as Fomu, which is 1 cm^2 and fits entirely within a USB port, which enables rapid prototyping for tinyML.
But when more resources are available, a larger more powerful soft CPU and CFU can be built, and with more memory, larger models can be run.

CFU Playground currently support the Xilinx 7-Series as well as the Lattice iCE40, ECP5, and CrossLink FPGAs. The Fomu with the iCE40UP5k FPGA is close to the smallest usable board; it features 5280 logic cells, 128kB on-chip large RAM, 30 512-byte block RAMs, and 8 16b x 16b DSP/multiplier blocks and is small enough to slot into a USB.

Software

To invoke the CFU, custom instructions must be added to the CPU's instruction set. However, it is not the compiler's responsibility to find uses for the instructions. It only needs to generate them when requested by the user. Therefore, we can use a stock RISC-V GCC toolchain, coupled with a macro we provide that expands to "asm" directives to generate the encoded custom instructions where necessary. The macro takes 4 arguments, and returns one result:

```
q = cfu_op(funct7, funct3, a, b);
```

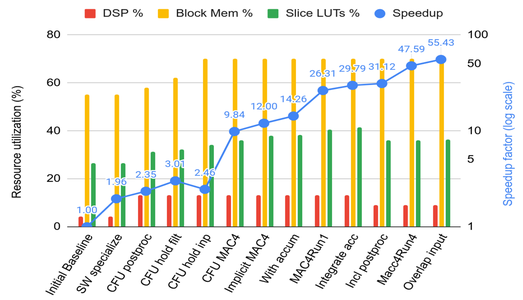
The macro directly generates the encoded 32b value, so not even the assembler needs modification. "funct7" and "funct3" are 7-bit and 3-bit fields respectively that specify the opcode of the custom instruction. They must be compile-time constant expressions. "a" and "b" are the C/C++ 32b integer variables used as operands for the instruction, and a 32b result is returned.

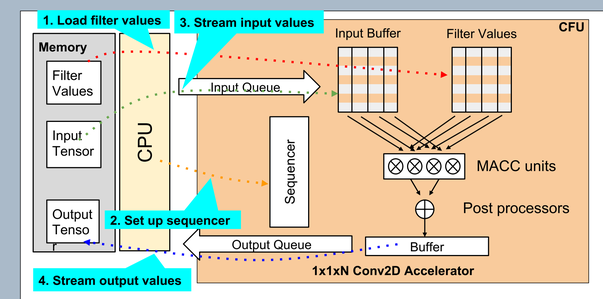
It is the user's responsibility to call the custom operations from their code. The custom instruction macros can be intermixed with regular C code, similar to any other C/C++ operation. TensorFlow Lite for Microcontrollers (TFLite Micro is the inference framework that CFU Playground uses for the deployment of the neural network. The user must provide an optimized kernel that uses the new custom instructions to realize the runtime performance improvements.

Evaluation

Image Classification

Image classification is a common task for low-power, always-on cameras, such as those in a smart doorbell. We present the experience of one of the authors, with limited prior hardware development experience, utilizing the CFU Playground deploy→profile→optimize loop, spending approximately 50% of their time over five weeks. They achieved a speedup of 55× for the most time-consuming TFLite operator in the model, bringing the contribution of that operator down from 5.5 seconds to 0.10 seconds per inference. We targeted a MobileNetV2 (MNV2) model, which is commonly used for efficient image classification, and optimized its performance on an Arty A7-35T board, which has a Xilinx XC7A35T FPGA with 256 MB of external DDR3 memory. The chart below shows the speedup progression as we stepped through each of the optimizations. We start with software optimizations and move on to hardware support using the CFU.

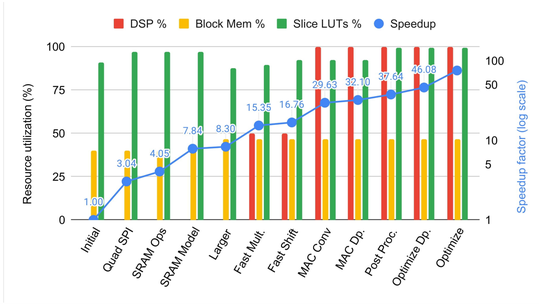




Final MobileNetV2 Accelerator Design

Keyword Spotting

The Keyword Spotting application is ubiquitous and an always-on tinyML use case, making it a perfect candidate for acceleration. This example took under four weeks to implement. We describe how an author (different than in the previous example) utilized CFU Playground to accelerate quantized (int8) inference of the MLPerf Tiny KWS model that included model-specific software optimizations and a custom CFU, by over 75×. We deployed the system to the tiny Fomu FPGA board, which is roughly the size of a penny and fits inside a USB slot. It combines an iCE40UP5k FPGA (with 5280 logic cells and 128 kB of on-chip RAM) with a 2 MB flash memory. This example demonstrates resource allocation optimization among the CPU, memory system, and CFU on a resource-constrained tinyML device.



Existing Frameworks

	Open Source	Tightly Coupled/ Specialized ISA	Full-Stack	Full-SoC	Processor Customization	Discrete ML Ops	Stock Compiler	TinyML Focus
CFU Playground	✓	✓	✓	✓	✓	✓	✓	✓
Gemmini [5]	✓	✗	✓	✓	✓	✗	✓	✗
hls4ml [2]	✓	✗	✗	✓	✗	✗	✗	✓
DNNWeaver [6]	✓	✓	✗	✗	✗	✗	✗	✗
Deepburning [4]	✓	✗	✓	✗	✗	✗	✗	✗
DNNBuilder [3]	✓	✗	✗	✗	✗	✗	✗	✗
FINN [1]	✓	✗	✗	✗	✗	✗	✗	✗

[1] Y. Umuroglu, et al., "Finn: A framework for fast, scalable binarized neural network inference," in International Symposium on FPGA, 2017.

[2] F. Fahim, et al., "hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices," CoRR, 2021.

[3] X. Zhang, et al., "Dnnbuilder: an automated tool for building high-performance dnn hardware accelerators for fpgas," in ICCAD, 2018.

[4] Y. Wang, et al., "Deepburning: Automatic generation of fpga-based learning accelerators for the neural network family," in DAC, 2016.

[5] H. Genc, et al., "Gemmini: Enabling systematic deep- learning architecture evaluation via full-stack integration," in Proceedings of 58th DAC, 2021.

[6] H. Sharma, et al., "Dnnweaver: From high-level deep network models to fpga acceleration," in Workshop on Cognitive Architectures, 2016.

†Google §Purdue University *Harvard University