Accelerated ML at the edge with mainline

Background

- Binary drivers strongly discourage people from using mainline
- Companies not using mainline are much less motivated to engage with the community and contribute back
- It used to be that the lack of FOSS GPU drivers drove companies to vendor BSPs
- Nowadays we are seeing the same happen with NPU drivers
- Working with vendor BSPs sucks!

The path to mainline

- NPUs and GPUs have a lot in common at the kernel level:
  - Hardware abstraction
  - Job scheduling
  - Memory management
  - Power management
- The compute-only kernel drivers tend to be relatively small and there is most of the time an out-of-tree GPL driver from the vendor
- But for acceptance into the DRM subsystem, the driver needs to be testable, which implies open userspace

NPU’s userspace problem

- GPU drivers are used via an open standard: OpenGL, Vulkan, OpenCL, etc
- You implement one or more of these and your kernel driver can be tested and thus could be merged into mainline
- But no such thing exists for NPUs as of 2023
- Vendors often ship binary-only forks of TensorFlow, Pytorch, ONNX, etc
- We need to start somewhere

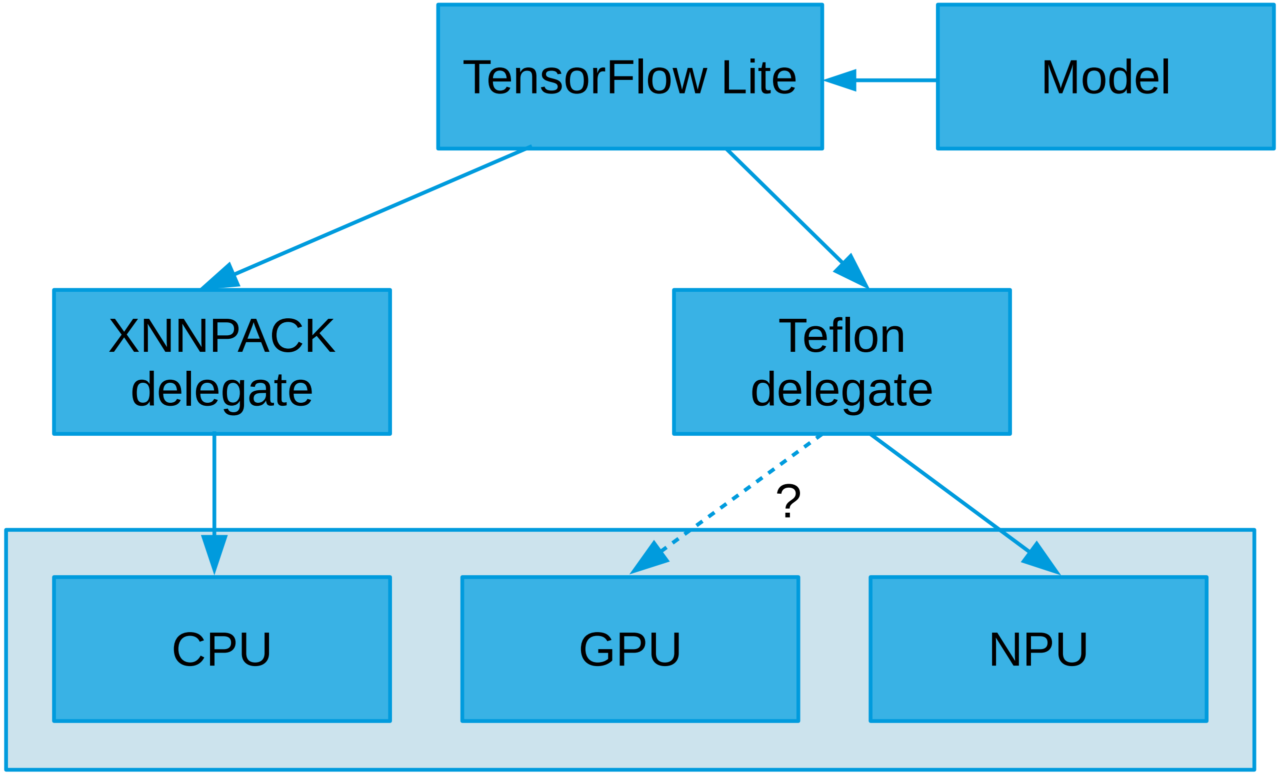
My proposal

- Choose one ML framework to start with, using this criteria:
  - Widely used
  - Optimized for embedded
  - Well-abstracted backends
- Use an abstraction layer so different ML frameworks can use the same HW-specific driver code
- When the different ML frameworks agree to use a common userspace API, move to it

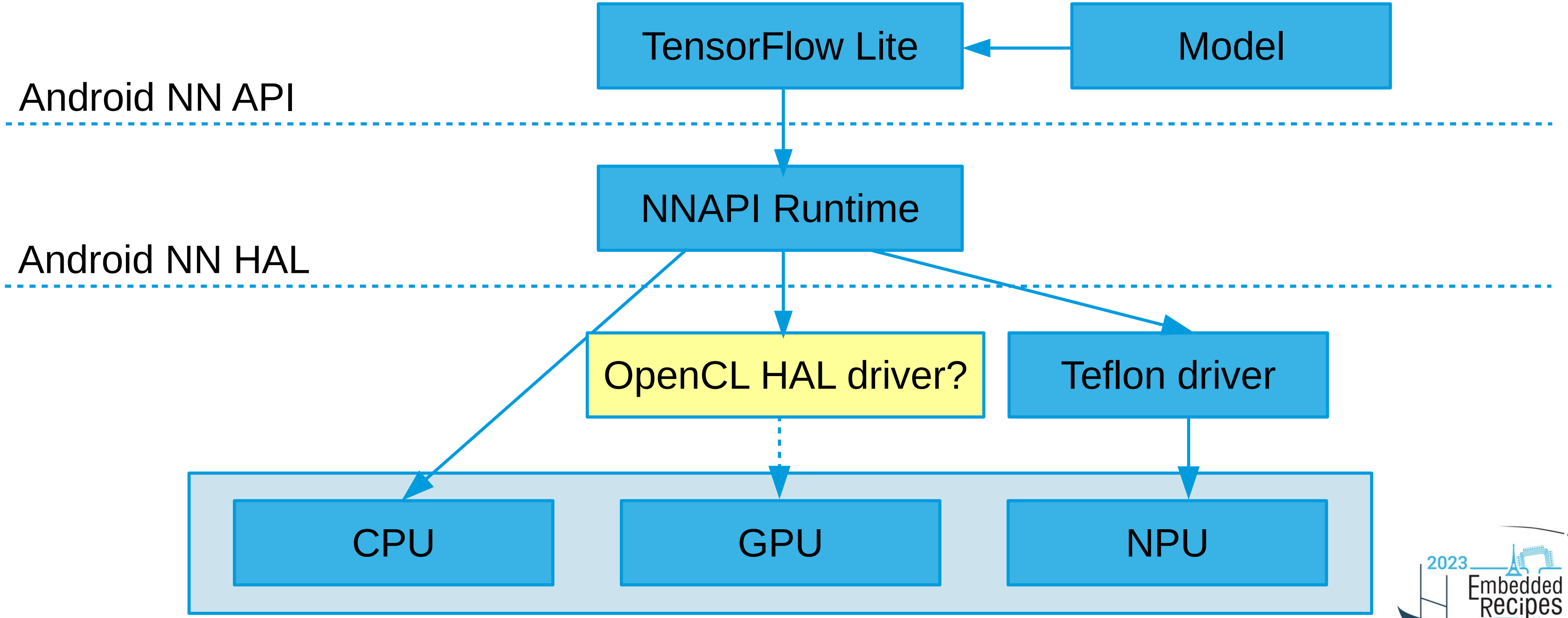
The plan to date

- Implement a TensorFlow Lite delegate in Mesa
- Use Gallium to abstract the HW-specific parts
- Start with VeriSilicon’s Vivante NPU
  - Used in several SoCs (Amlogic, Rockchip, NXP, and more)
  - Widely available in popular SBCs
  - Allows reuse of Etnaviv kernel driver and reverse-engineering tools
- Implement enough to run popular models at least 3x faster than on the CPUs on the A311D, starting with MobileNetV1

Multi-delegate cooperation



Multi-delegate cooperation 2



What could come later

- Support programmable cores in the NPUs with OpenCL (GPGPUs, DSPs, FPGAs, …)
- Optimally run parts of the model on the GPU that is in the SoC
- Add other drivers:
  - Mediatek/Cadence’s, Rockchip’s, Amlogic’s, Arm’s, …
- Add frontends for other ML frameworks:
  - NNAPI, Arm NN, PyTorch, ONNX, XLA backend, ...

Vivante NPU driver

- The target is the VIPNano-QI as found in the Amlogic A311D SoC
- The project started as an attempt at an OpenCL driver
- That proved futile once the first model was run on it, as the single programmable core (GPGPU) on the NPU is really slow
- The focus switched to the fixed-function convolution and tensor manipulation units
- The OpenCL effort wasn’t wasted though as Christian Gmeiner of Igalia is taking the work and upstreaming it

Hardware

- VIPNano-QI (GC8000):
  - 8 NN (convolution) units, supporting INT8 and INT16
  - 4 TP (tensor manipulation) units
  - 1 programmable core
  - On-chip SRAM: 512 KB
  - External SRAM: 1024 KB

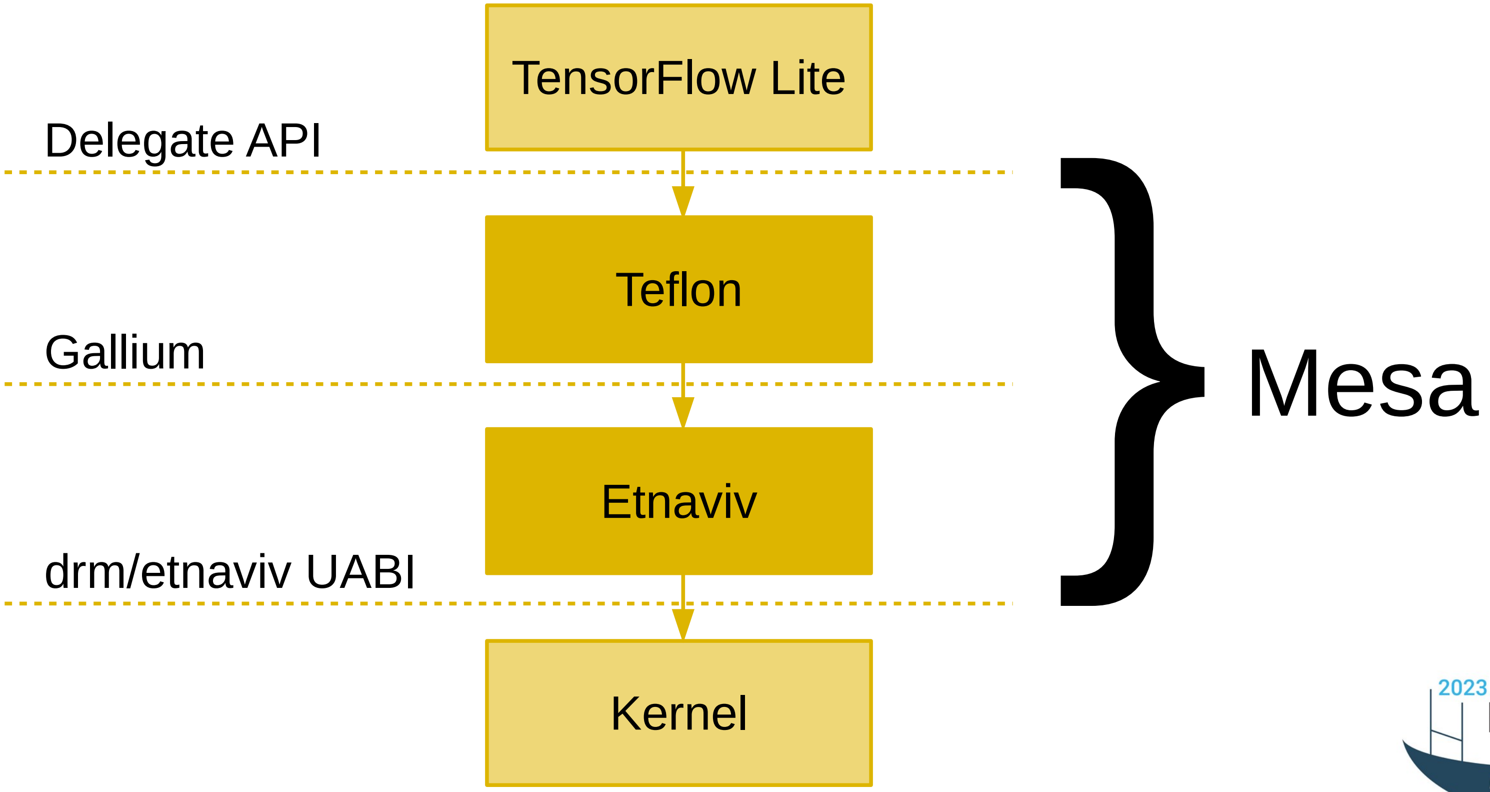
Kernel driver: drm/etnaviv

- The changes in the kernel side were minimal, they mostly involved adding stuff to the power domains for the SoC, to its clocks and to the device tree
- Most are in mainline already, but the DT node is disabled by default for now, and there are small stability and performance fixes in the pipeline

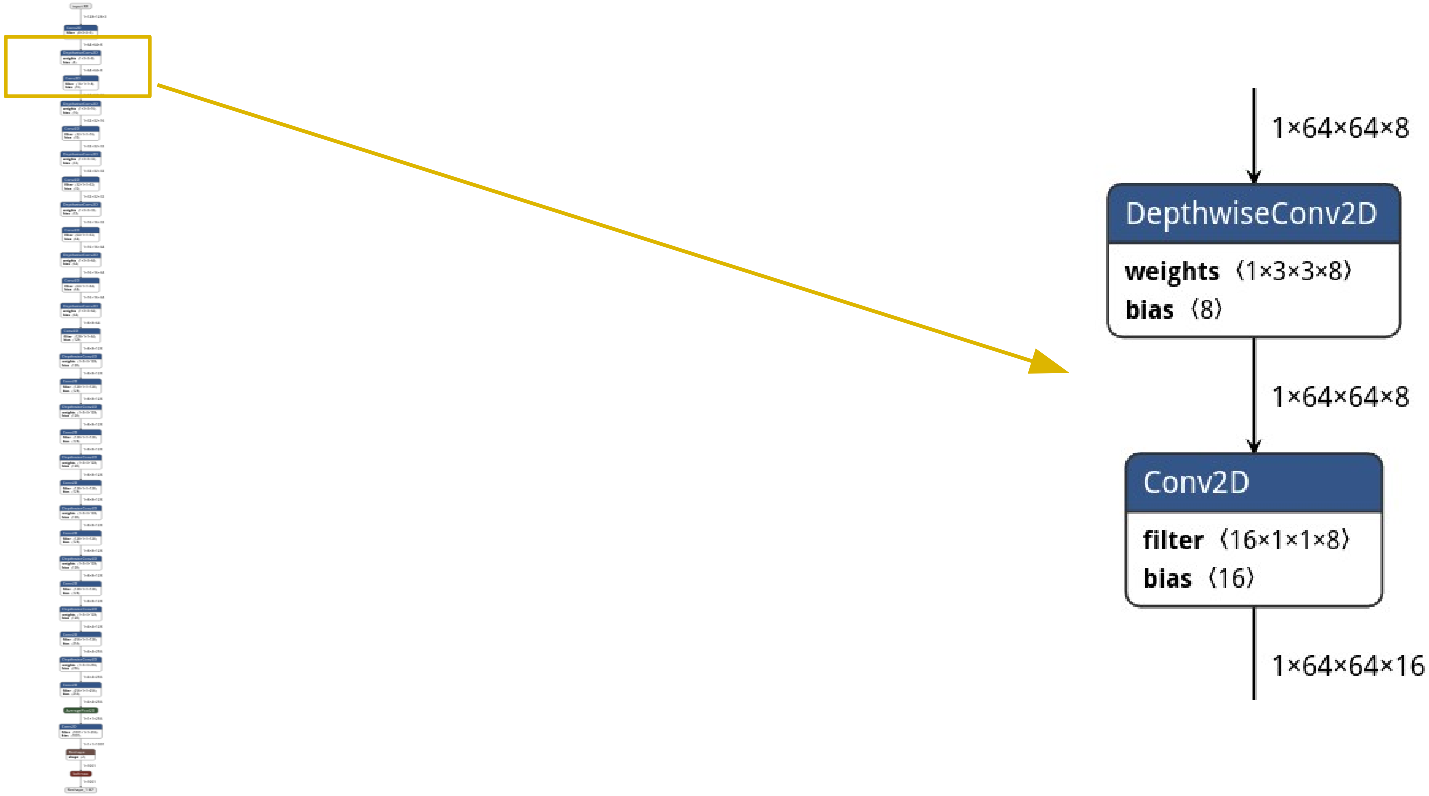
Teflon

- A Gallium state tracker that implements the TensorFlow Lite delegate API in terms of Gallium
    - https://docs.mesa3d.org/gallium/index.html
- HW-independent
- Currently it is at the proof of concept stage, it will see a rewrite soon to get it on the path to production readiness

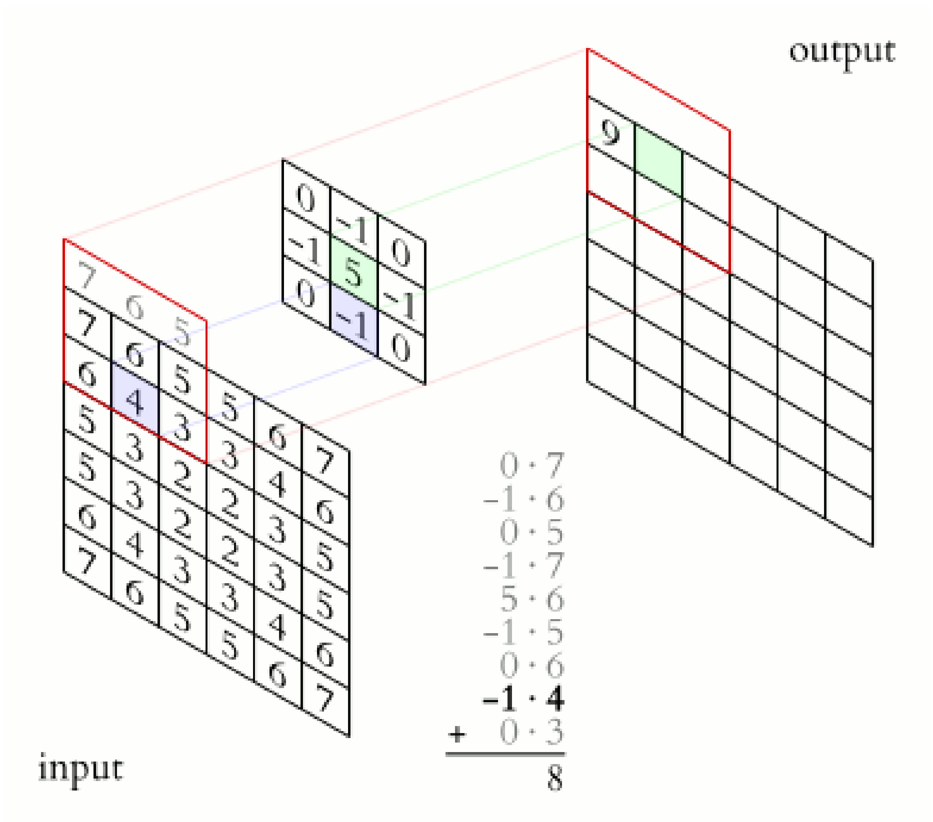
Teflon


TensorFlow Lite
Delegate API
Teflon
Gallium
Etnaviv
drm/etnaviv UABI
Kernel
Mesa

MobileNetV1



Convolution



https://en.wikipedia.org/wiki/File:2D_Convolution_Animation.gif

Reverse-engineering process

1) Intercept communication between userspace and kernel
2) Relate dumps to workload
3) Come up with hypotheses about the parts of the communication that are still unknown
4) Test those hypotheses
5) Improve RE tools with the newly acquired knowledge
6) Change workload and go back to step 1

Workload and its parameters

Conv2D layer

Conv2D class [source]

```
keras_core.layers.Conv2D(
    filters,
    kernel_size,
    strides=(1, 1),
    padding="valid",
    data_format=None,
    dilation_rate=(1, 1),
    groups=1,
    activation=None,
    use_bias=True,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    kernel_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    bias_constraint=None,
    **kwargs
)
```

2D convolution layer.

This layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. If `use_bias` is True, a bias vector is created and added to the outputs. Finally, if `activation` is not `None`, it is applied to the outputs as well.

- Well-defined mathematical operations
- HW exposes an interface that is a close match
- Performance considerations bring notable complexity
- Not all combinations are supported by the HW

https://keras.io/keras_core/api/layers/convolution_layers/convolution2d/

Reverse engineering the command stream

```
+ diff -u -U 100 /home/tomeu/mesa.txt /home/tomeu/galcore.txt
--- /home/tomeu/mesa.txt    2023-08-07 18:28:29.939750225 +0200
+++ /home/tomeu/galcore.txt    2023-08-07 18:28:42.116625362 +0200
@@ -1,176 +1,273 @@
 {
-    0x0801028a, /* LOAD_STATE (1) Base: 0x00A28 Size: 1 Fixp: 0 */
-    0x00000011, /*   PA.SYSTEM_MODE := PROVOKING_VERTEX_LAST=1,HALF_PIXEL_CENTER=1 */
-    0x08010e13, /* LOAD_STATE (1) Base: 0x0384C Size: 1 Fixp: 0 */
-    0x00000002, /*   GL.API_MODE := OPENCL */
+    0x00000000, /* UNKNOWN (0) */
+    0x00000000, /*  */
+    0x00000000, /* UNKNOWN (0) */
+    0x00000000, /*  */
+    0x00000000, /* UNKNOWN (0) */
+    0x00000000, /*  */
     0x00000000, /* UNKNOWN (0) */
     0x00000000, /*  */
     0x08010e4f, /* LOAD_STATE (1) Base: 0x0393C Size: 1 Fixp: 0 */
     0x00000000, /*   GL.OCB_REMAP_START := 0x0 */
     0x08010e50, /* LOAD_STATE (1) Base: 0x03940 Size: 1 Fixp: 0 */
     0x00000000, /*   GL.OCB_REMAP_END := 0x0 */
     0x08010e4c, /* LOAD_STATE (1) Base: 0x03930 Size: 1 Fixp: 0 */
     0x00000010, /*   GL.NN_CONFIG := UNK0=0x0,DISABLE_ZDPN=0,DISABLE_SWTILING=0,SMALL_BATCH=1,DDR_BURST_SIZE=0x0,UNK7=0,NN_CORE_COUNT=0x0,UNK12=0 */
     0x08010428, /* LOAD_STATE (1) Base: 0x010A0 Size: 1 Fixp: 0 */
-    0xffff3000, /*   PS.NN_INST_ADDR := *0xffff3000 */
+    0x3348e780, /*   PS.NN_INST_ADDR := *0x3348e780 */
     0x08010429, /* LOAD_STATE (1) Base: 0x010A4 Size: 1 Fixp: 0 */
     0x00000000, /*   0x010A4 */
     0x08010e03, /* LOAD_STATE (1) Base: 0x0380C Size: 1 Fixp: 0 */
     0x00000c23, /*   GL.FLUSH_CACHE := DEPTH=1,COLOR=1,TEXTURE=0,PE2D=0,TEXTUREVS=0,SHADER_L1=1,SHADER_L2=0,UNK10=1,UNK11=1,DESCRIPTOR_UNK12=0,DESCRIPTOR_UNK13=0 */
     0x08010e03, /* LOAD_STATE (1) Base: 0x0380C Size: 1 Fixp: 0 */
     0x00000c23, /*   GL.FLUSH_CACHE := DEPTH=1,COLOR=1,TEXTURE=0,PE2D=0,TEXTUREVS=0,SHADER_L1=1,SHADER_L2=0,UNK10=1,UNK11=1,DESCRIPTOR_UNK12=0,DESCRIPTOR_UNK13=0 */
     0x00000000, /* UNKNOWN (0) */
     0x00000000, /*  */
 }
```

Reverse engineering the instruction description

```
map->layer_type = 0x0;  /* (0) */
map->no_z_offset = 0x0;  /* (0) */
map->kernel_xy_size = 0x2;  /* (2) */
map->kernel_z_size = 0x4;  /* (4) */
map->kernels_per_core = 0x1;  /* (1) */
map->pooling = 0x0;  /* (0) */
map->pooling_xy_size = 0x1;  /* (1) */
map->prelu = 0x0;  /* (0) */
map->nn_layer_flush = 0x1;  /* (1) */
map->kernel_data_type = 0x0;  /* (0) */
map->in_image_data_type = 0x0;  /* (0) */
map->out_image_data_type = 0x0;  /* (0) */
map->in_image_x_size = 0x4;  /* (4) */
map->in_image_y_size = 0x4;  /* (4) */
map->in_image_x_offset = 0x0;  /* (0) */
map->in_image_y_offset = 0x0;  /* (0) */
map->unused0 = 0x0;  /* (0) */
map->brick_mode = 0x0;  /* (0) */
map->brick_distance = 0x0;  /* (0) */
map->relu = 0x0;  /* (0) */
map->unused1 = 0x0;  /* (0) */
map->post_multiplier = 0x0;  /* (0) */
map->post_shift = 0x17;  /* (23) */
map->unused2 = 0x0;  /* (0) */
map->no_flush = 0x0;  /* (0) */
map->unused3 = 0x0;  /* (0) */
map->out_image_x_size = 0x3;  /* (3) */
map->out_image_y_size = 0x3;  /* (3) */
map->out_image_z_size = 0x1;  /* (1) */
map->rounding_mode = 0x1;  /* (1) */
map->in_image_x_offset_bit_3 = 0x0;  /* (0) */
map->in_image_y_offset_bit_3 = 0x0;  /* (0) */
map->out_image_tile_x_size = 0x3;  /* (3) */
map->out_image_tile_y_size = 0x3;  /* (3) */
```

```
-map->kernel_address = 0x3fffd00;  /* (67108096) */
+map->kernel_address = 0xcd237f;  /* (13443967) */
 map->kernel_z_size2 = 0x0;  /* (0) */
-map->in_image_address = 0xffff6000;
-map->out_image_address = 0xffff7000;
+map->in_image_address = 0x3348e240;
+map->out_image_address = 0x89ffc500;
 map->image_caching_mode = 0x0;  /* (0) */
 map->kernel_caching_mode = 0x1;  /* (1) */
 map->partial_cache_data_unit = 0x0;  /* (0) */
 map->kernel_pattern_msb = 0x0;  /* (0) */
 map->kernel_y_size = 0x2;  /* (2) */
 map->out_image_y_stride = 0x3;  /* (3) */
 map->kernel_pattern_low = 0x0;  /* (0) */
 map->kernel_pattern_high = 0x0;  /* (0) */
 map->kernel_cache_start_address = 0x800;
 map->kernel_cache_end_address = 0xa00;
 map->image_start_address = 0x0;  /* (0) */
 map->image_end_address = 0x800;  /* (2048) */
 map->in_image_border_mode = 0x0;  /* (0) */
 map->in_image_border_const = 0x7d;  /* (125) */
 map->unused4 = 0x0;  /* (0) */
 map->kernel_data_type_bit_2 = 0x0;  /* (0) */
 map->in_image_data_type_bit_2 = 0x0;  /* (0) */
 map->out_image_data_type_bit_2 = 0x0;  /* (0) */
 map->post_multiplier_1_to_6 = 0x1f;  /* (31) */
 map->post_shift_bit_5_6 = 0x0;  /* (0) */
 map->unused5 = 0x0;  /* (0) */
 map->in_image_x_stride = 0x4;  /* (4) */
 map->in_image_y_stride = 0x4;  /* (4) */
 map->out_image_x_stride = 0x3;  /* (3) */
 map->unused6 = 0x0;  /* (0) */
 map->post_multiplier_7_to_14 = 0x61;  /* (97) */
 map->out_image_circular_buf_size = 0x0;  /* (0) */
```

```
map->unused7 = 0x0;  /* (0) */
map->per_channel_post_mul = 0x0;  /* (0) */
map->out_image_circular_buf_end_addr_plus_1 = 0x3ffffff;
map->unused8 = 0x0;  /* (0) */
map->in_image_circular_buf_size = 0x0;  /* (0) */
map->unused9 = 0x0;  /* (0) */
map->in_image_circular_buf_end_addr_plus_1 = 0x3ffffff;
map->unused10 = 0x0;  /* (0) */
map->coef_zero_point = 0x80;  /* (128) */
map->out_zero_point = 0x77;  /* (119) */
map->kernel_direct_stream_from_VIP_sram = 0x0;
map->depthwise = 0x0;  /* (0) */
map->unused11 = 0x0;  /* (0) */
map->unused12 = 0x0;  /* (0) */
map->unused13 = 0x0;  /* (0) */
map->unused14 = 0x0;  /* (0) */
map->unused15 = 0x0;  /* (0) */
map->unused16 = 0x0;  /* (0) */
map->further1 = 0x0;  /* (0) */
map->further2 = 0x0;  /* (0) */
map->further3 = 0x3ffffff;  /* (67108863) */
map->further4 = 0x7f800000;  /* (2139095040) */
map->further5 = 0xff800000;  /* (4286578688) */
map->further6 = 0x0;  /* (0) */
map->further7 = 0x0;  /* (0) */
map->further8 = 0x0;  /* (0) */
```

Reverse engineering the coefficient buffer

```
0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x2c, 0x99, 0x0e, 0x00, 0x00,
0x40, 0xea, 0x2c, 0xeb, 0x80, 0xaf, 0x80, 0x9b, 0x99, 0x80, 0x80, 0x13,
0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00
```

Other sources of information for RE

- Several very interesting environment variables that enable and disable functionality are published at:
  - https://github.com/boundarydevices/android_device_boundary
- Env. variables such as CNN_PERF and NN_EXT_SHOW_PERF dump quite interesting information
- Vendor kernel driver source code is GPL and contains code to execute trivial jobs to the different execution units
- Marketing material
- Research papers

Research papers

- Take it in your stride: Do we need striding in CNNs?
  - https://arxiv.org/abs/1712.02502
- Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference
  - https://arxiv.org/abs/1712.05877
- Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding
  - https://arxiv.org/abs/1510.00149
- Adaptive Weight Compression for Memory-Efficient Neural Networks
  - https://dl.acm.org/doi/pdf/10.5555/3130379.3130424

2023 Embedded Recipes 5th edition 28 to 29 september

Test suite

- Good test coverage is always a must, but when there is so much uncertainty, it is doubly so
- VeriSilicon’s own TFLite delegate is open-source and I based my test suite on theirs as I add support for more convolution variations:
  - https://github.com/VeriSilicon/tflite-vx-delegate/blob/main/test/python/test_conv2d.py
- No continuous integration yet, but appropriate hardware is already present in Mesa’s CI farm, so it won’t be much work

What already works

- Regular convolutions:

```
@pytest.mark.parametrize("batch_size",  [1])
@pytest.mark.parametrize("input_size",  [4, 112])
@pytest.mark.parametrize("weight_size", [1, 3])
@pytest.mark.parametrize("in_ch",       [32, 128, 256])
@pytest.mark.parametrize("out_ch",      [32, 128, 256])
@pytest.mark.parametrize("stride",      [1, 2])
@pytest.mark.parametrize("padding",     ["valid", "same"])
@pytest.mark.parametrize("signed",      [False])
@pytest.mark.parametrize("seed",        [4, 5])
def test_conv2d(batch_size, input_size, weight_size, in_ch, out_ch, stride, padding, signed, seed):
    if out_ch == 32 and in_ch == 1 and stride == 2 and weight_size == 1:
        pytest.skip("Blob seg faults and it's probably not a useful case")
```

What already works

- Depthwise convolutions:

```
@pytest.mark.parametrize("batch_size",  [1])
@pytest.mark.parametrize("input_size",  [4, 112])
@pytest.mark.parametrize("weight_size", [3])
@pytest.mark.parametrize("channels",    [32, 128, 256])
@pytest.mark.parametrize("stride",      [1, 2])
@pytest.mark.parametrize("padding",     ["valid", "same"])
@pytest.mark.parametrize("signed",      [False])
@pytest.mark.parametrize("seed",        [4, 5])
def test_depthwise(batch_size, input_size, weight_size, channels, stride, padding, signed, seed):
    s = "%r-%s-%r-%r-%r-%r-%r-%r" % (seed, signed, padding, stride, channels, weight_size, input_size, batch_size)
    print(s, file=sys.stderr)
    convolution(batch_size, input_size, weight_size, channels, channels, stride, padding, signed, seed, depthwise=True)
```

What still remains

- Avoid copies in and out of the model partition, by mapping user buffers to the NPU
- Use the TP units for tensor manipulation (transposing, mostly)
- Properly configuring the automatic caching of kernels and images in the internal on-chip SRAM
- Use the external SRAM for intermediate tensor data
- Batch all TP and NN jobs from a model partition in the command stream
- Enable zero-run-length compression in the coefficient buffer
- Tune the tiling parameters for reduced memory bandwidth usage

Start of a community

- For now I aim for this work to happen inside the Mesa project
- Haven't submitted any code for review though
- This has been a bit of a solo journey so far
- Hopefully that will change soon
- I have been posting updates to https://blog.tomeuvizoso.net/
- IRC channel: #ml-mainline at OFTC

Questions time

For those watching this later, you can send any questions you may have to:

tomeu@tomeuvizoso.net