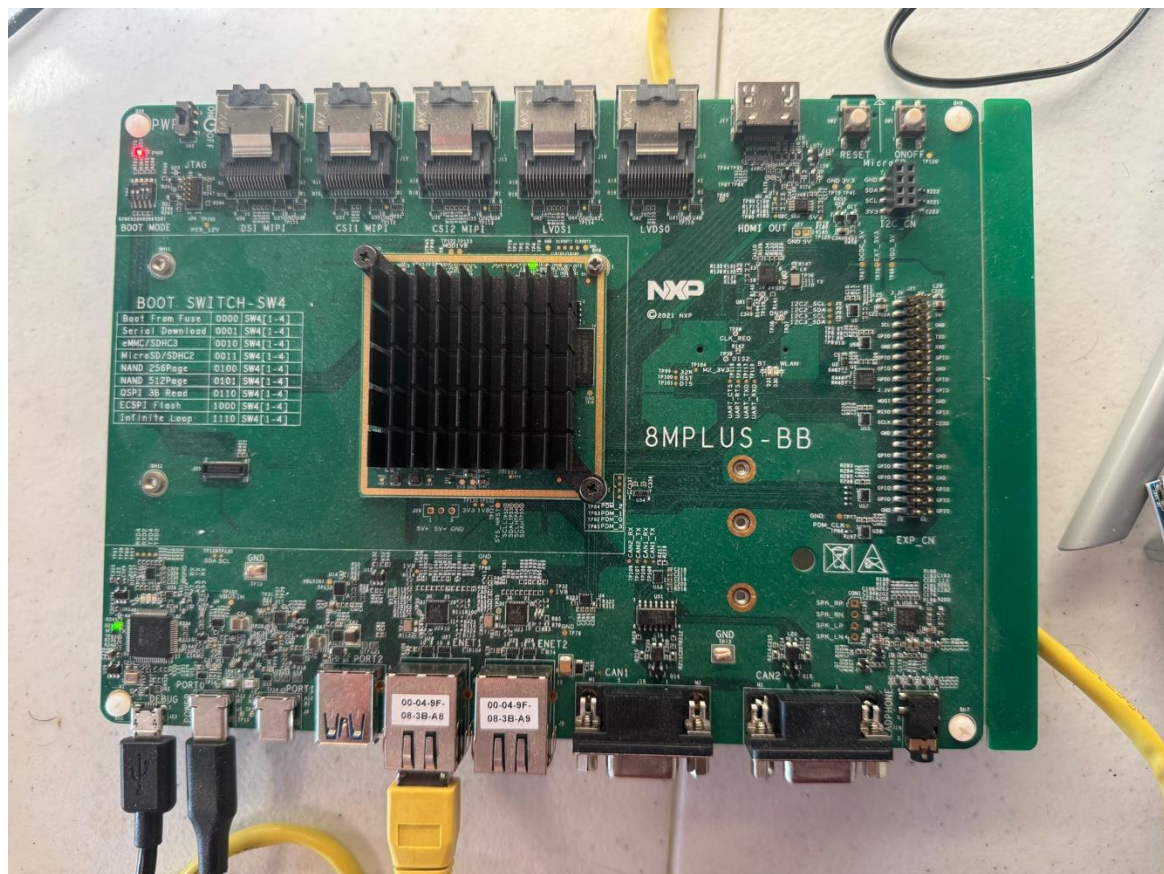


Speed Kills: Exploring Security Aspects of Edge AI Accelerators

Presenter: Srihari Danduri



- What are Edge AI Accelerators?
- Why inference speed is crucial for these devices?
- What are NPU's (Neural Processing Units) ?
- In the Image, below big square heatsink is SOC (System on chip).

What is a confused deputy problem?



In information security, a confused deputy is **a computer program that is tricked by another program (with fewer privileges or less rights) into misusing its authority on the system.** It is a specific type of privilege escalation.



Wikipedia

[https://en.wikipedia.org › wiki › Confused_deputy_pro...](https://en.wikipedia.org/wiki/Confused_deputy_pro...)

Host Memory, Shared between co-processor and Application Processor

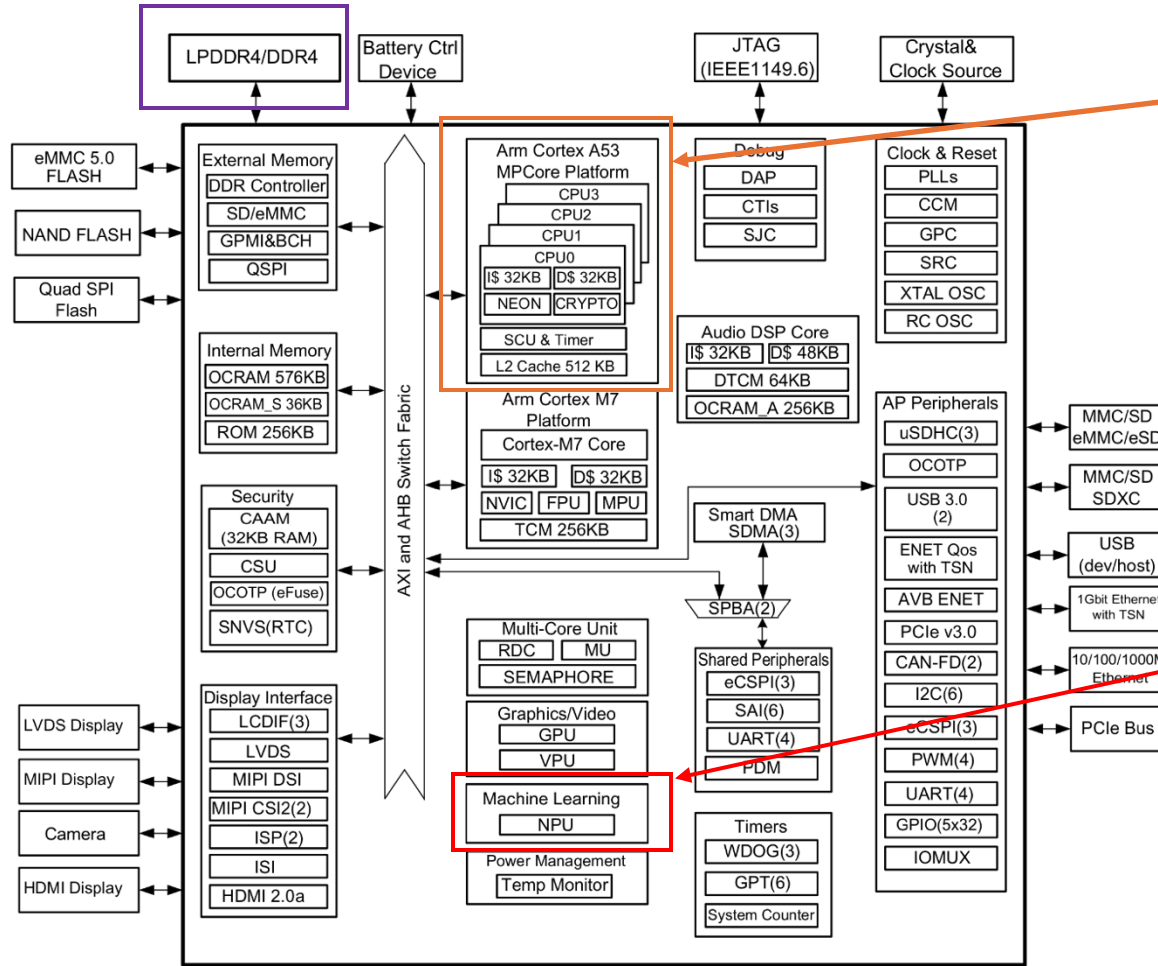


Figure 1-1. Block Diagram

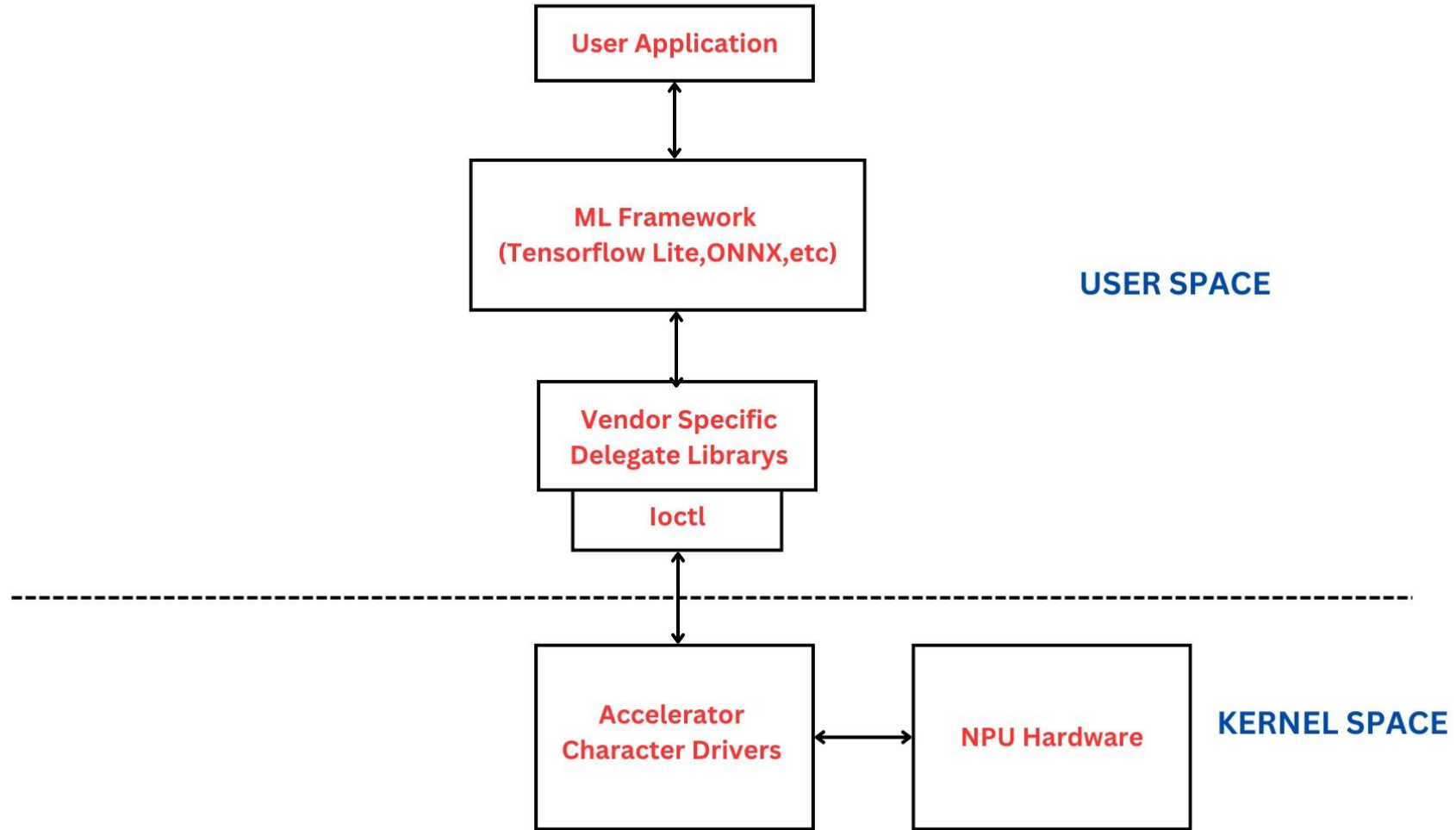
This is where your OS(Linux) and Applications Run

Any Illegal operation done by Application is restricted by OS

Application Machine learning workloads are offloaded to NPU for processing

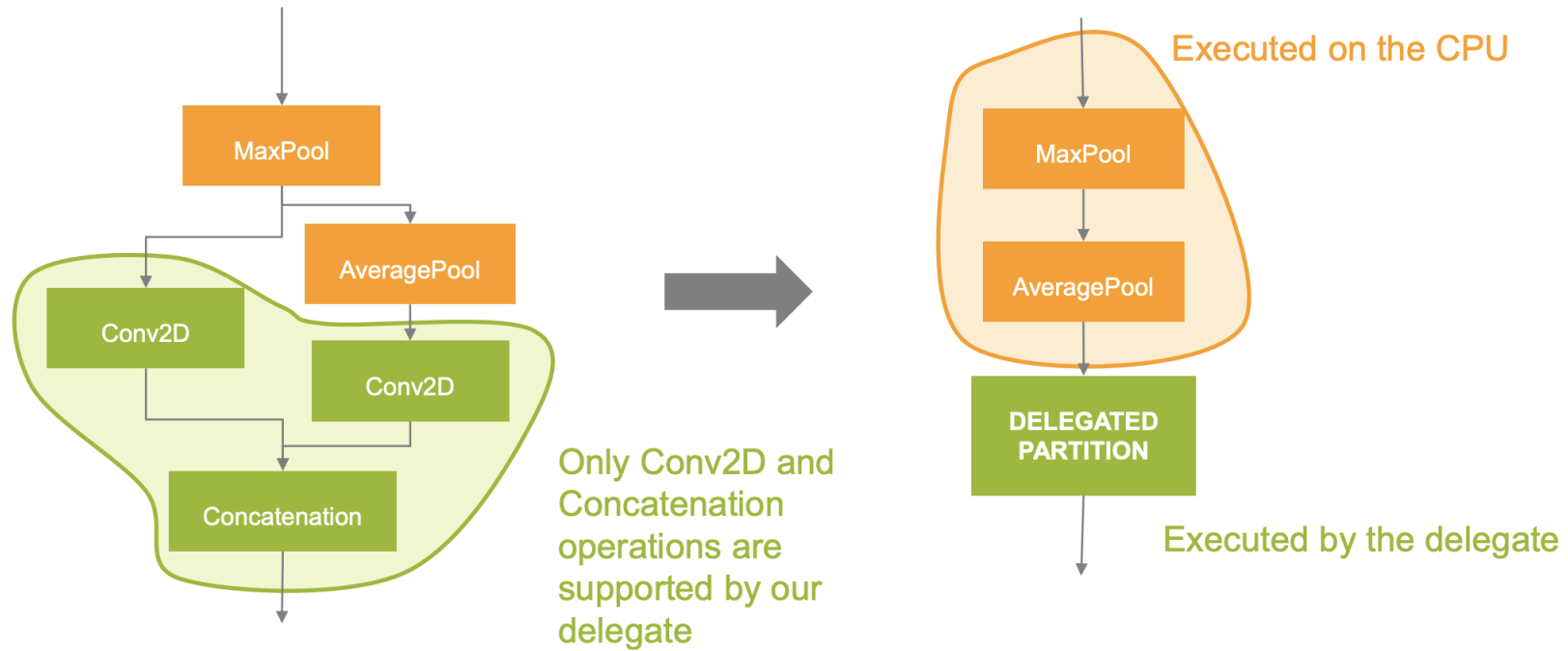
No OS protections when executing Application Machine learning workloads

High-level flow from Application to NPU Hardware



DELEGATE PARTITIONING

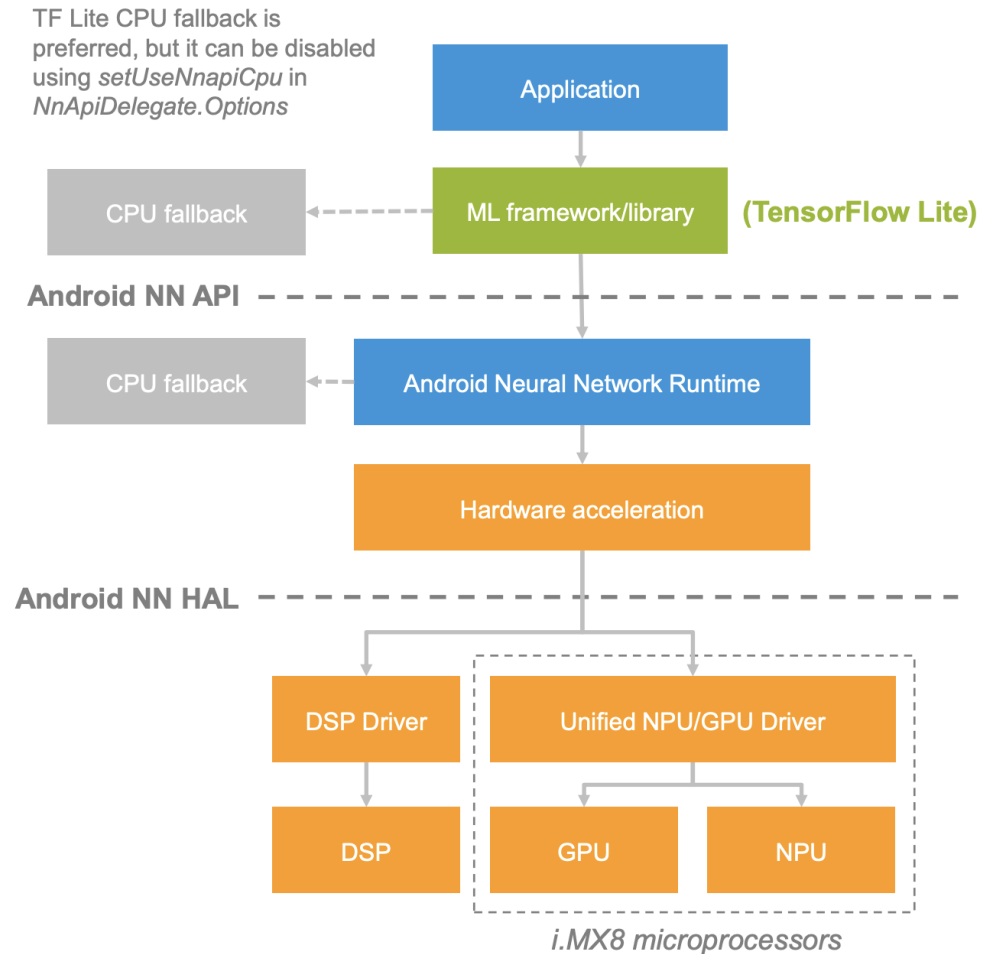
- Graph is partitioned based on op support



Link to official slides from NXP: [slides](#)

NN API DELEGATE

- Android C API designed to run machine learning operations on Android devices
- Limited to *float16*, *float32*, *int8* and *uint8*
- Supports acceleration on a GPU, an NPU or a DSP depending on the target device



Let's Understand System Design (Application all the way to NPU Hardware)

```
tflite::ops::builtin::BuiltinOpResolver resolver;  
tflite::InterpreterBuilder(*model, resolver>(&interpreter);
```

```
auto delegates = delegate_providers.CreateAllDelegates();
```

```
for (auto& delegate : delegates) {
```

```
const auto delegate_name = delegate.provider->GetName();
```

```
if (interpreter->ModifyGraphWithDelegate(std::move(delegate.delegate)) !=
```

```
ktfLiteOk) {
```

```
LOG(ERROR) << "Failed to apply " << delegate_name << " delegate.";
```

```
exit(-1);
```

```
} else {
```

```
LOG(INFO) << "Applied " << delegate_name << " delegate.";
```

```
}
```

```
}
```

```
for (int i = 0; i < settings->loop_count; i++) {
```

```
if (interpreter->Invoke() != ktfLiteOk) {
```

```
LOG(ERROR) << "Failed to invoke tflite!";
```

```
exit(-1);
```

```
}
```

```
}
```

```
Check source code here link
```

```
# Launch Application to use NPU
```

```
export USE_GPU_INFERENCE=0
```

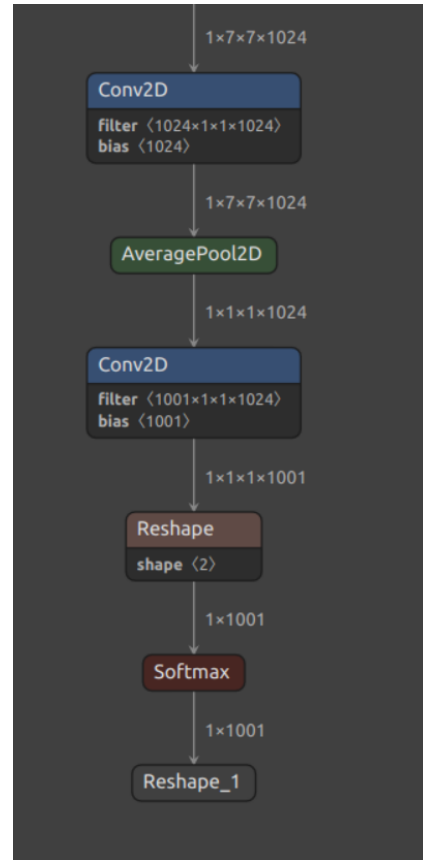
```
gdb --nx --args ./label_image -m mobilenet_v1_1.0_224_quant.tflite -i grace_hopper.bmp -l labels.txt --  
external_delegate_path=/usr/lib/libvx_delegate.so
```

Start Addr	End Addr	Size	Offset	Perms	objfile
0xaaaaaaaa000	0xaaaaaaaa000	0x3e000	0x0	r-xp	/home/sri/examples/label_image
0xaaaaaaaa000	0xaaaaaaaaf000	0x2000	0x3e000	r--p	/home/sri/examples/label_image
0xaaaaaaaa1000	0xaaaaaaaa1000	0x4000	0x40000	rw-p	/home/sri/examples/label_image
0xaaaaaaaa1000	0xaaaaaaaa56000	0x475000	0x0	rw-p	[heap]
0xffff00000000	0xffff00000000	0x10000000	0x0	rw-s	/dev/zero (deleted)
0xffff00000000	0xffff0021000	0x21000	0x0	rw-p	
0xffff0021000	0xffff0021000	0x30f000	0x0	---p	
0xffff0400000	0xffff0400000	0x100000	0x0	---p	
0xffff0410000	0xffff0521000	0x800000	0x0	rw-p	
0xffff0521000	0xffff0543000	0x1f3000	0x0	r-xp	/usr/lib/libGAL.so
0xffff0543000	0xffff0541e000	0x10000	0x1f3000	---p	/usr/lib/libGAL.so
0xffff0541e000	0xffff05420000	0x20000	0x1f3000	r--p	/usr/lib/libGAL.so
0xffff05420000	0xffff0543000	0x14000	0x200000	rw-p	/usr/lib/libGAL.so
0xffff0543000	0xffff0544000	0xc000	0x0	rw-p	
0xffff0549000	0xffff0643000	0x1000000	0x0	r-xp	/usr/lib/libVSC.so
0xffff0643000	0xffff0649000	0x6000	0x1000000	---p	/usr/lib/libVSC.so
0xffff0649000	0xffff0648000	0x17000	0x1019000	r--p	/usr/lib/libVSC.so
0xffff0648000	0xffff064ef000	0x6f000	0x1030000	rw-p	/usr/lib/libVSC.so
0xffff064ef000	0xffff0638000	0x348000	0x0	r-xp	/usr/lib/libOpenVX.so.1.3.0
0xffff0638000	0xffff064b000	0x13000	0x348000	---p	/usr/lib/libOpenVX.so.1.3.0
0xffff064b000	0xffff0650000	0x5000	0x34b000	r--p	/usr/lib/libOpenVX.so.1.3.0
0xffff0650000	0xffff0674000	0x24000	0x350000	rw-p	/usr/lib/libOpenVX.so.1.3.0
0xffff0680000	0xffff06c1000	0x641000	0x0	r-xp	/usr/lib/libtin-vx.so
0xffff06c1000	0xffff06c7000	0x6000	0x641000	---p	/usr/lib/libtin-vx.so
0xffff06c7000	0xffff06f0000	0x29000	0x647000	r--p	/usr/lib/libtin-vx.so
0xffff06f0000	0xffff06f0a000	0xa000	0x670000	rw-p	/usr/lib/libtin-vx.so
0xffff06f1a000	0xffff0700000	0xe000	0x0	rw-p	/home/sri/examples/mobilenet_v1_1.0_224_quant.t
0xffff0700000	0xffff07415000	0x415000	0x0	r--s	/usr/lib/libNNArchPerf.so
0xffff0740000	0xffff0740000	0x20000	0x0	rw-p	/usr/lib/libNNArchPerf.so
0xffff0740000	0xffff074b7000	0x17000	0x28000	---p	/usr/lib/libNNArchPerf.so
0xffff074b7000	0xffff074c0000	0x1000	0x2f000	r--p	/usr/lib/libNNArchPerf.so
0xffff074c0000	0xffff074c000	0x4000	0x30000	rw-p	/usr/lib/libNNArchPerf.so
0xffff074c000	0xffff0741a000	0x24000	0x0	r-xp	/usr/lib/libArchModelSw.so
0xffff0741a000	0xffff0750f000	0x15000	0x2a000	---p	/usr/lib/libArchModelSw.so
0xffff0750f000	0xffff07510000	0x1000	0x2f000	r--p	/usr/lib/libArchModelSw.so
0xffff07510000	0xffff07515000	0x5000	0x30000	rw-p	/usr/lib/libArchModelSw.so
0xffff07515000	0xffff07517000	0x2000	0x0	rw-p	
0xffff07520000	0xffff075ba000	0x9a000	0x0	r-xp	/usr/lib/libvx_delegate.so
0xffff075ba000	0xffff075c0000	0xe000	0x9a000	---p	/usr/lib/libvx_delegate.so
0xffff075c0000	0xffff0750000	0x8000	0xa0000	r--p	/usr/lib/libvx_delegate.so
0xffff0750000	0xffff0757000	0x7000	0xb0000	rw-p	/usr/lib/libvx_delegate.so
0xffff0750000	0xffff0776b000	0x18b000	0x0	rw-p	/usr/lib/libc.so.6
0xffff0770000	0xffff0770000	0x12000	0x180000	---p	/usr/lib/libc.so.6
0xffff0770000	0xffff07780000	0x3000	0x180000	r--p	/usr/lib/libc.so.6
0xffff07780000	0xffff0778000	0x2000	0x190000	rw-p	/usr/lib/libc.so.6
0xffff07782000	0xffff0778e000	0xc000	0x0	rw-p	
0xffff07790000	0xffff077a8000	0x18000	0x0	r-xp	/usr/lib/libgcc_s.so.1
0xffff077a8000	0xffff077bf000	0x17000	0x18000	---p	/usr/lib/libgcc_s.so.1
0xffff077bf000	0xffff077c0000	0x1000	0x1f000	r--p	/usr/lib/libgcc_s.so.1
0xffff077c0000	0xffff077c1000	0x1000	0x20000	rw-p	/usr/lib/libgcc_s.so.1
0xffff077d0000	0xffff077a0a000	0x23a000	0x0	r-xp	/usr/lib/libstdc++.so.6.0.32
0xffff077a0a000	0xffff077a13000	0x9000	0x23a000	---p	/usr/lib/libstdc++.so.6.0.32
0xffff077a13000	0xffff077a20000	0xd000	0x243000	r--p	/usr/lib/libstdc++.so.6.0.32
0xffff077a20000	0xffff077a21000	0x1000	0x250000	rw-p	/usr/lib/libstdc++.so.6.0.32
0xffff077a21000	0xffff077a25000	0x4000	0x0	rw-p	
0xffff077a30000	0xffff077ab2000	0x82000	0x0	r-xp	/usr/lib/libm.so.6
0xffff077ab2000	0xffff077ac1000	0x9000	0x82000	---p	/usr/lib/libm.so.6
0xffff077ac1000	0xffff077ab0000	0x1000	0x83000	r--p	/usr/lib/libm.so.6
0xffff077ad0000	0xffff077a1000	0x10000	0x90000	rw-p	/usr/lib/libm.so.6
0xffff077a0000	0xffff07783000	0x4a3000	0x0	r-xp	/usr/lib/libtensorflow-lite.so.2.15.0
0xffff07783000	0xffff07790000	0x15000	0x4a3000	---p	/usr/lib/libtensorflow-lite.so.2.15.0
0xffff07790000	0xffff07790000	0x400000	0x0	rw-p	/usr/lib/libtensorflow-lite.so.2.15.0
0xffff077a0000	0xffff077a6000	0x6000	0x4b0000	rw-p	/usr/lib/libtensorflow-lite.so.2.15.0
0xffff077a000	0xffff077b2000	0xc000	0x0	rw-p	
0xffff077ba000	0xffff077be000	0x4000	0x0	rw-p	
0xffff077be000	0xffff077c5000	0x27000	0x0	r-xp	/usr/lib/ld-linux-aarch64.so.1
0xffff077c0000	0xffff077c0000	0x2000	0x0	rw-p	
0xffff0777000	0xffff077f9000	0x2000	0x0	rw-p	
0xffff077f9000	0xffff077fb000	0x2000	0x0	r--p	[vvar]
0xffff077fb000	0xffff077fc000	0x1000	0x0	r-xp	[vdso]
0xffff077fc000	0xffff077fe000	0x2000	0x2000	r--p	/usr/lib/ld-linux-aarch64.so.1
0xffff077fe000	0xffff08000000	0x2000	0x30000	rw-p	/usr/lib/ld-linux-aarch64.so.1
0xffff08000000	0x1000000000000	0x21000	0x0	rw-p	[stack]

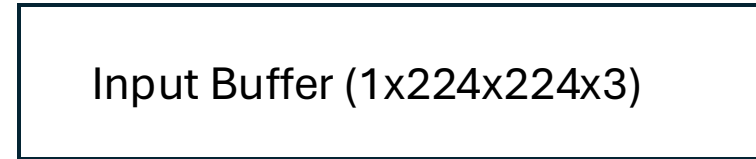
Let's Understand System Design (Application all the way to NPU Hardware)



Mobilenet model



Contiguous in User Virtual address space



Memory Overview

```
[ 0.000000] Booting Linux on physical CPU 0x000000000 [0x410fd034]
[ 0.000000] Linux version 6.6.23-gb586a521770e-dirty (sri@sri-nuc) (aarch64-poky-linux-gcc (GCC) 13.2.0, GNU ld (GNU Binutils) 2.42.0.202402)
[ 0.000000] KASLR disabled due to lack of seed
[ 0.000000] Machine model: NXP i.MX8MPlus EVK board
[ 0.000000] efi: UEFI not found.
[ 0.000000] Reserved memory: created CMA memory pool at 0x00000000c4000000, size 960 MiB
[ 0.000000] OF: reserved mem: initialized node linux,cma, compatible id shared-dma-pool
[ 0.000000] OF: reserved mem: 0x00000000c4000000..0x00000000ffffffff (983040 KiB) map reusable linux,cma
[ 0.000000] OF: reserved mem: 0x0000000009000000..0x000000000096ffff (448 KiB) nomap non-reusable ocram@900000
[ 0.000000] OF: reserved mem: 0x0000000056000000..0x0000000057dfffff (30720 KiB) nomap non-reusable optee_core@56000000
[ 0.000000] OF: reserved mem: 0x0000000057e00000..0x0000000057ffffff (2048 KiB) nomap non-reusable optee_shm@57e00000
[ 0.000000] OF: reserved mem: 0x0000000092400000..0x00000000933fffff (16384 KiB) nomap non-reusable dsp@92400000
[ 0.000000] OF: reserved mem: 0x0000000093400000..0x00000000942effff (15296 KiB) nomap non-reusable dsp_reserved_heap@93400000
[ 0.000000] OF: reserved mem: 0x00000000942f0000..0x00000000942f7fff (32 KiB) nomap non-reusable vdev0vring0@942f0000
[ 0.000000] OF: reserved mem: 0x00000000942f8000..0x00000000942fffff (32 KiB) nomap non-reusable vdev0vring1@942f8000
[ 0.000000] Reserved memory: created DMA memory pool at 0x0000000094300000, size 1 MiB
[ 0.000000] OF: reserved mem: initialized node vdev0buffer@94300000, compatible id shared-dma-pool
[ 0.000000] OF: reserved mem: 0x0000000094300000..0x00000000943fffff (1024 KiB) nomap non-reusable vdev0buffer@94300000
[ 0.000000] OF: reserved mem: 0x0000000100000000..0x000000010ffffff (262144 KiB) nomap non-reusable gpu_reserved@100000000
[ 0.000000] NUMA: No NUMA configuration found
[ 0.000000] NUMA: Faking a node at [mem 0x0000000040000000-0x00000001bfffffff]
[ 0.000000] NUMA: NODE_DATA [mem 0x1bf4386c0-0x1bf43afff]
[ 0.000000] Zone ranges:
[ 0.000000] DMA [mem 0x0000000040000000-0x00000000ffffffff]
[ 0.000000] DMA32 empty
[ 0.000000] Normal [mem 0x0000000100000000-0x00000001bfffffff]
[ 0.000000] Movable zone start for each node
[ 0.000000] Early memory node ranges
[ 0.000000] node 0: [mem 0x0000000040000000-0x0000000055fffff]
[ 0.000000] node 0: [mem 0x0000000058000000-0x00000000923fffff]
[ 0.000000] node 0: [mem 0x0000000092400000-0x00000000943fffff]
[ 0.000000] node 0: [mem 0x0000000094400000-0x00000000ffffffff]
[ 0.000000] node 0: [mem 0x0000000100000000-0x000000010ffffff]
[ 0.000000] node 0: [mem 0x0000000110000000-0x00000001bfffffff]
[ 0.000000] Initmem setup node 0 [mem 0x0000000040000000-0x00000001bfffffff]
[ 0.000000] On node 0, zone DMA: 8192 pages in unavailable ranges
```

NPU Platform Drivers

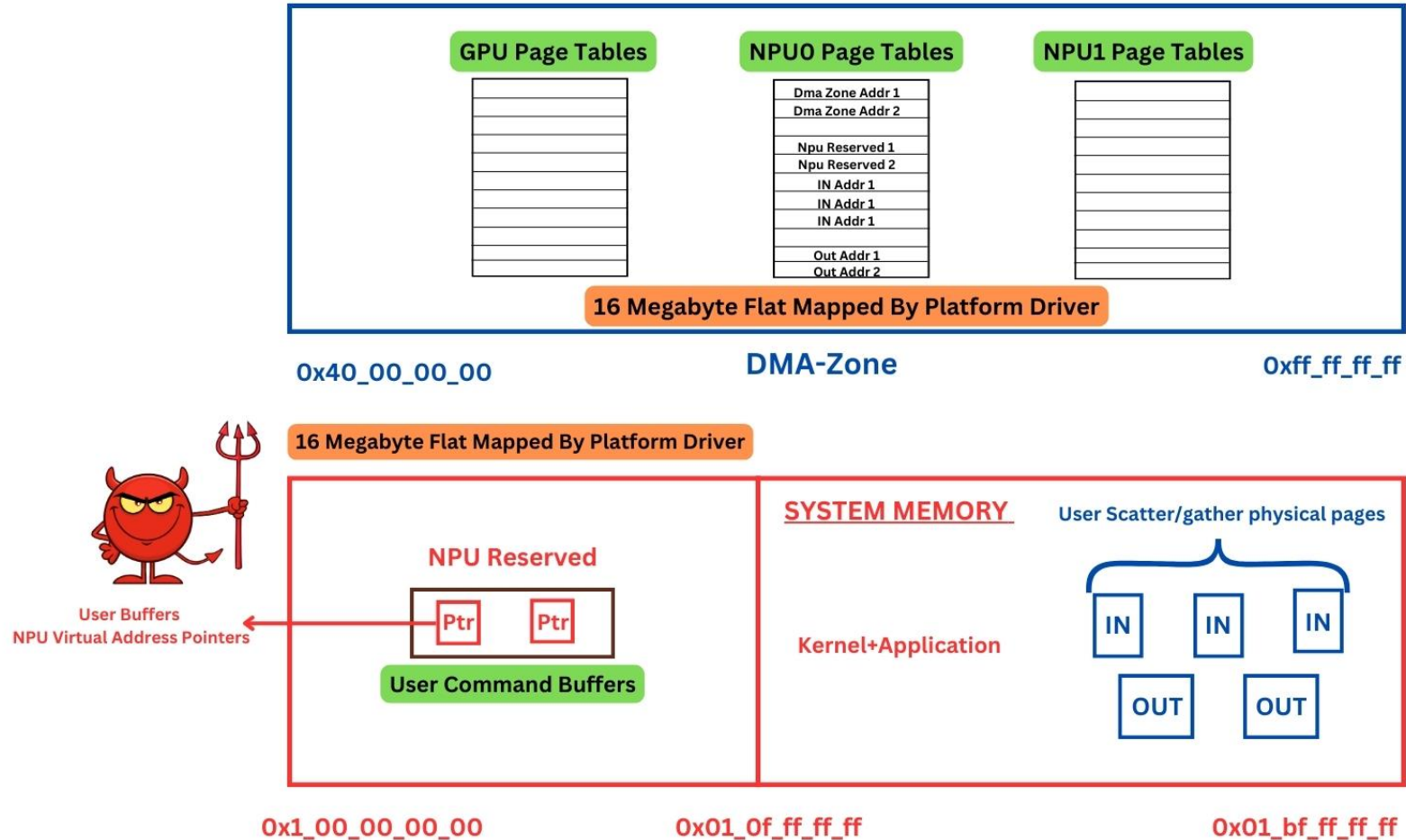
- NPU Platform Drivers probe for supported hardware coming in from the device tree and then allocate and construct page tables.
- These allocated page tables are part of DMA zone.
- The base address of these pagetables are programmed in to MMIO(Memory Mapped regions) of NPU so it know where to do page walk during address translation.



NPU Character Drivers

- ioctl calls are performed to carveout address space for mapping reserved regions in to user process.
- Application runtime uses this mmapped reserved region to construct command buffers.
- Using ioctls these command buffers are submitted to NPU to perform computation.
- ioctl calls are performed to do on-demand page mapping of user buffer pages into NPU pagetables. Also marks those physical pages for DMA capable so DMA controller does the copying.

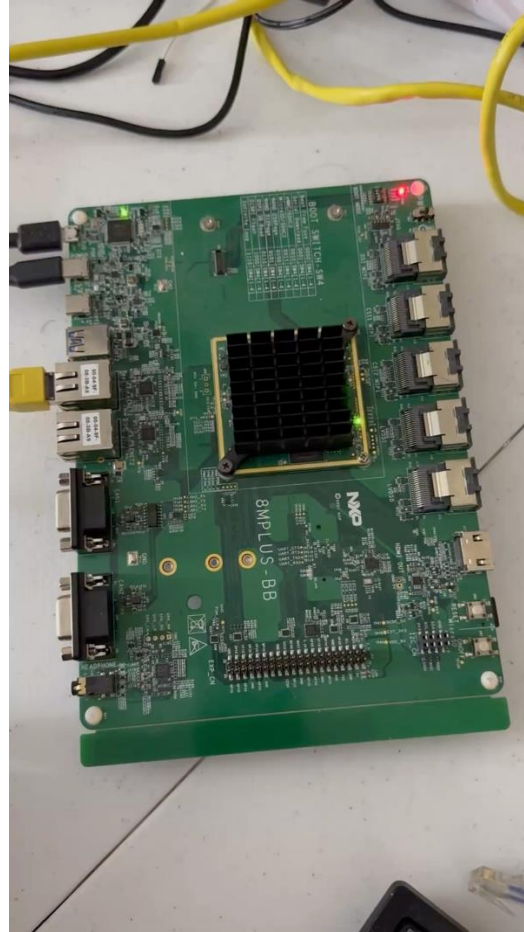
Putting all together



Demo Time

Recorded video

Incase of a live demo disaster



- To run the previous exploit, all you need is 4 bytes of stack corruption.
- Nothing fancy prerequisites to be met to do confused deputy-style attacks using NPU.
- In our case user application can be malicious or it can be a compromised user.

Shortcuts in System design

- To Optimize for Inference speed, vendors used a design with shared memory that can be accessible by the Application, Kernel, and NPU hardware.
- Here input, output buffers and reserved memory act as shared memory for all three.
- Do as much Application \leftrightarrow NPU communication as possible to reduce the overhead of going through Kernel.
- Flat mapping entire regions of memory for speed, this choice only increases the attack surface.

Security Properties Schematic gap between host and NPU:

- On the host each process has its own page tables thereby there is clear separation between processes and the host processor knows what physical memory ranges a process has access to.
- whereas as in NPU there is one set of page tables shared by all processes which kernel driver programs entries, and there is no <process, memory regions it can access information>
- The processing is on a command basis and there is no <processid, command> relationship understood by NPU. Multiple processes can issue commands, and NPU is a shared resource.
- Even if you have protections against kernel memory, there will always be cross-process contamination.

Proposed Solutions

- **Kernel driver Sanitization:** Have the kernel driver sanitize all the command buffers before committing to the co-processor. They might have to open out the closed source code of command buffer construction and all address offsets in it, and implement sanitization in the kernel driver.
- **GPU/NPU Firmware level-fix:** Before reading/writing values to and from physical memory have the co-processor send a request to the host processor to confirm the validity of the regions for a given command.

Thank you