CONSOLE HACKING 2016

PS4: PC MASTER RACE

@marcan42

IN MEMORY OF BEN ‘BUSHING’ BYER

It's a bit different from previous consoles

✓ x86 ✓ FreeBSD ✓ WebKit

✗ Hypervisor

But not completely different

✓ Security processor (that you can just ignore)

HOW TO PWN A PS4

Step 1: Write a WebKit exploit

Step 2: Write a FreeBSD exploit

BORING
BORING EVERYWHERE!
quickmeme.com

HOW TO PWN A PS4

Step 0. Dump the code

Step 1. Write a WebKit exploit

Step 2. Write a FreeBSD exploit

Step 3. ?

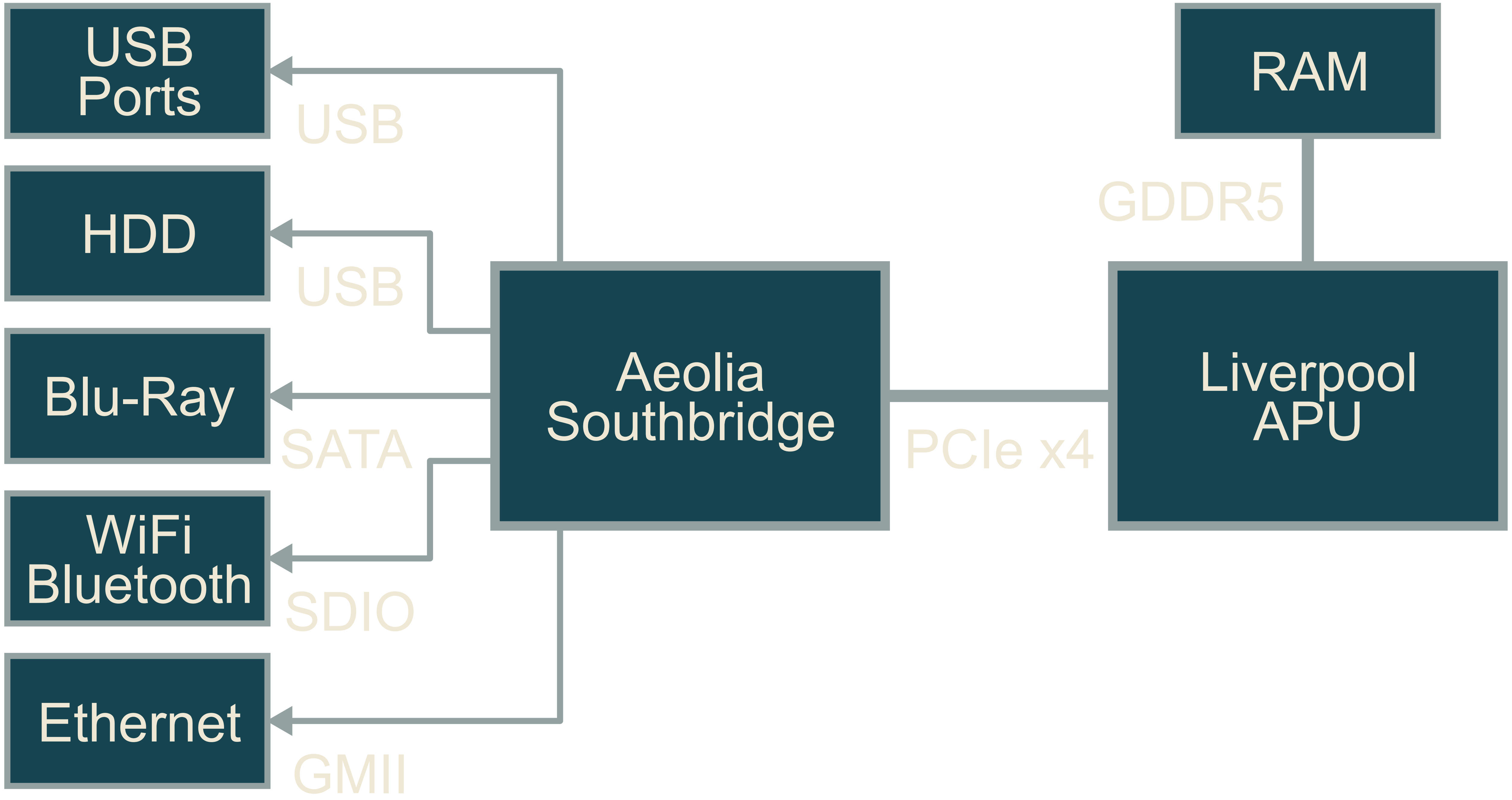
Step 4. PROFIT

BLACK-BOX CODE EXTRACTION

THE FUN WAY

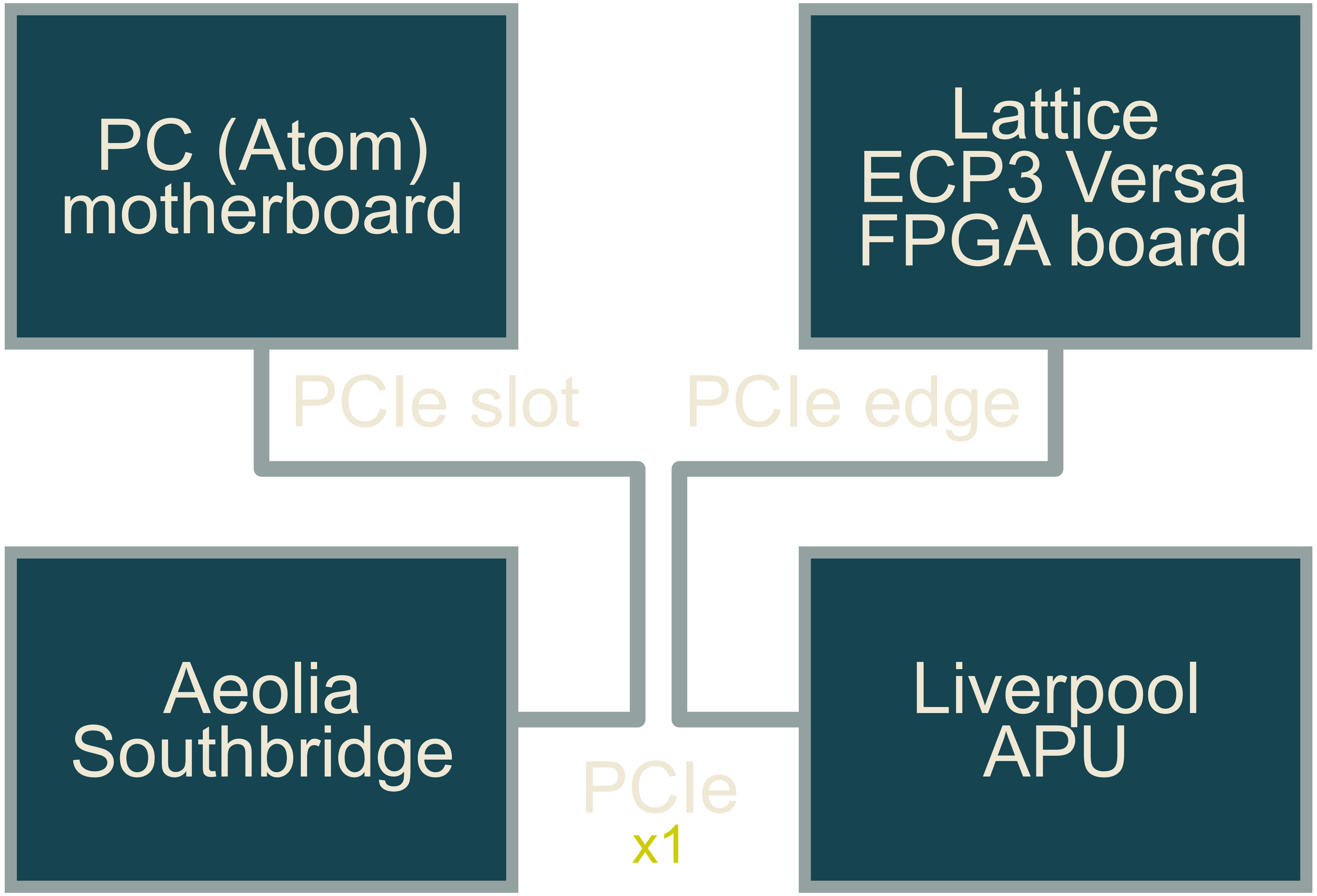
A long time ago in a hackerspace far, far away….

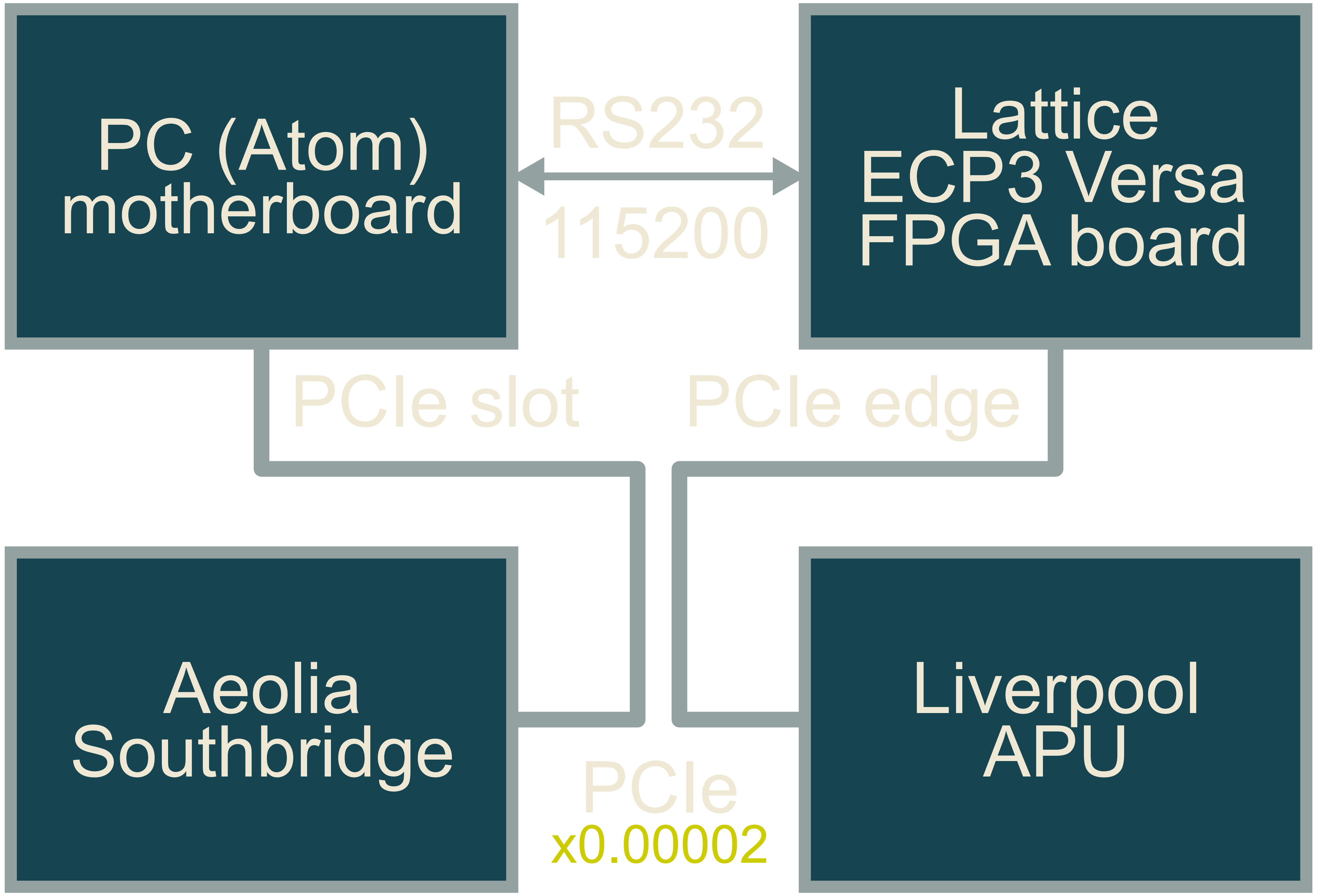
(fail0verflow got together after 31c3)


USB Ports
USB
RAM
GDDR5
HDD
USB
Aeolia Southbridge
Liverpool APU
Blu-Ray
SATA
PCIe x4
WiFi Bluetooth
SDIO
Ethernet
GMII

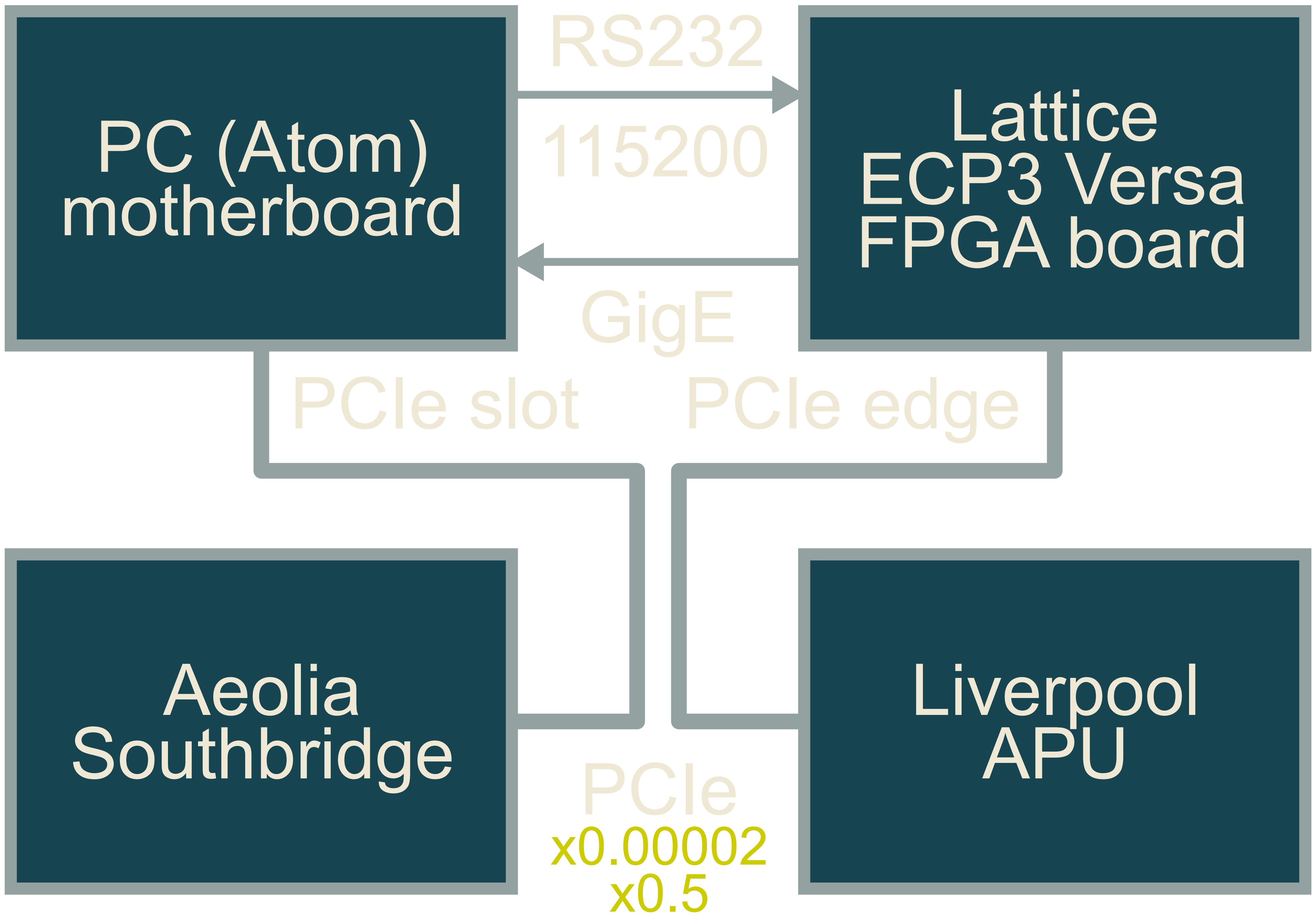
PCIE: FUN FOR THE WHOLE FAMILY

- Bus mastering
- Complicated
- Robust
- Delay tolerant
- Drivers full of fail


PC (Atom) motherboard
Lattice ECP3 Versa FPGA board
PCIe slot
PCIe edge
Aeolia Southbridge
Liverpool APU
PCIe
x1



PC (Atom) motherboard
RS232
115200
Lattice ECP3 Versa FPGA board
PCIe slot
PCIe edge
Aeolia Southbridge
PCIe
x0.00002
Liverpool APU


PC (Atom) motherboard
RS232
115200
Lattice ECP3 Versa FPGA board
GigE
PCIe slot
PCIe edge
Aeolia Southbridge
Liverpool APU
PCIe
x0.00002
x0.5

PCIE 101

PCIe is a reliable switched packet network

Transaction Layer Packets (TLPs):

- Memory reads/writes
- IO reads/writes
- Configuration reads/writes
- Message signaled interrupts (MSI) (writes)
- Legacy interrupts
- Completions

GO WILD WITH DMA!

Except there's an IOMMU...

```
void load_some_stuff(void)
{
    char buf[32];

    plz2read_from_flash(SOME_ADDRESS, buf, 32);
}

void plz2read_from_flash(uint32_t addr, void *buf, size_t size)
{
    iommu_map(buf, size);
    flash_send_read_command(addr, buf, size);
    iommu_unmap(buf, size);
}
```

✓ Code execution

✓ FreeBSD kernel dump

✓ WebKit and OS libs
dump

HOW TO PWN A PS4

✓ Step 0. Dump the code

✓ Step 1. Write a WebKit exploit

✓ Step 2. Write a FreeBSD exploit

Step 3. ?ps4-kexec

Step 4. PROFIT (Linux)

FROM FREEBSD TO LINUX

PS4-KEXEC

jmp linux

Not so fast... we need to:

- Load Linux into contiguous physical RAM
- Set up Linux boot parameters
- Shut down FreeBSD cleanly
- Halt secondary CPUs
- Make new pagetables and GDT
- Disable the IOMMU
- Relocate various things in memory
- And more...

OKAY, NOW `jmp linux`, RIGHT?

Sure, Linux will *technically* run

For a little bit anyway

And then it stops

No video, no serial output, nothing

LET'S TALK ABOUT HARDWARE

WHAT IS x86?

A mediocre instruction set architecture

The PS4 is x86 (x86-64)

WHAT IS A PC?

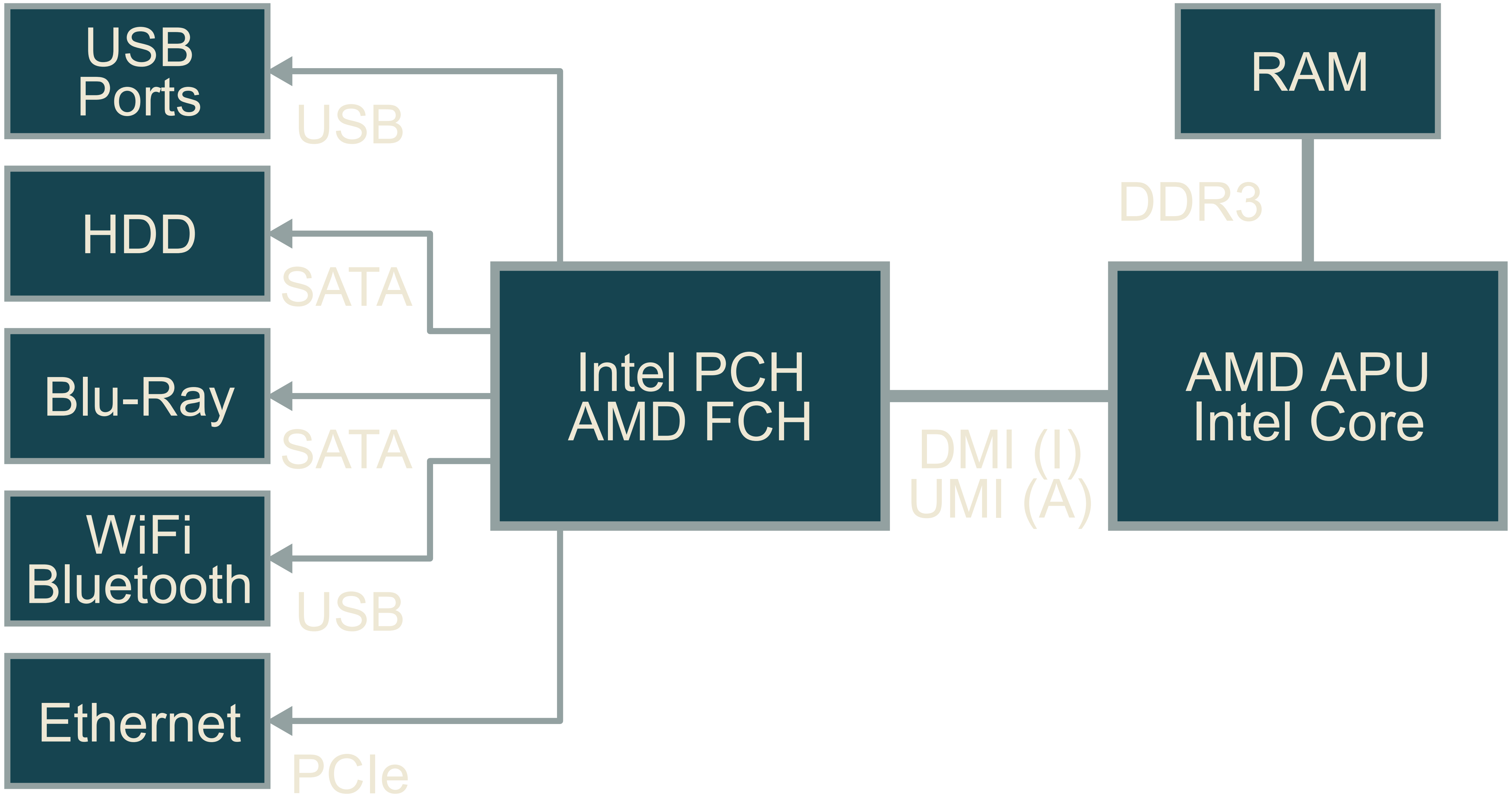
A horrible, horrible thing built upon piles and piles of legacy nonsense dating back to 1981

The PS4 is *NOT* a PC

PC 101

- 8259 Programmable Interrupt Controller (PIC)
- 8253 Programmable Interval Timer (PIT)
- 8250 UART at I/O 3f8h
- 8042 PS/2 Keyboard Controller
- MC146818 RTC/CMOS
- ISA bus
- VGA

The PS4 has none of these



USB Ports

USB

HDD

SATA

Blu-Ray

SATA

WiFi Bluetooth

USB

Ethernet

PCIe

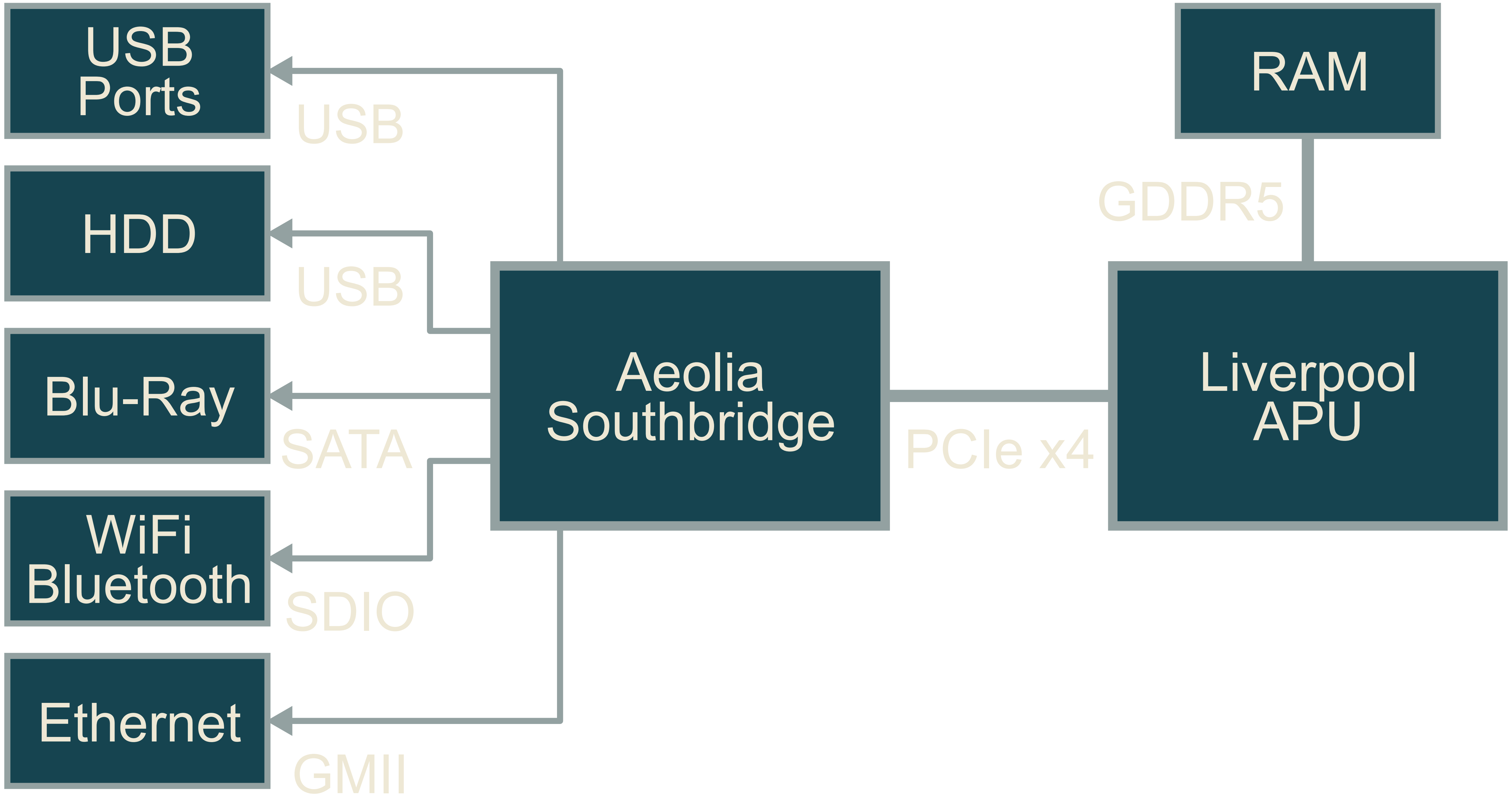
Intel PCH
AMD FCH

DMI (I)
UMI (A)

AMD APU
Intel Core

DDR3

RAM



USB Ports
USB
HDD
USB
Blu-Ray
SATA
WiFi Bluetooth
SDIO
Ethernet
GMII
Aeolia Southbridge
PCIe x4
RAM
GDDR5
Liverpool APU

AMD FCH SOUTHBRIDGE

Implements Intel legacy (1981)

MARVELL AEOLIA SOUTHBRIDGE

Implements Intel legacy (2002)

????

THAT'S NO SOUTHBRIDGE

THAT'S A MARVELL ARMADA SOC

- Descendant from Intel StrongARM/XScale
- ARM SoC with a bunch of peripherals
- They stuck a PCIe bridge on it
- Exposes ARM peripherals to the x86 side
- Some extra stuff (e.g. HPET, ACPI stuff)
- 256MB DDR RAM
- *Also* runs FreeBSD in standby mode
- Batshit insane

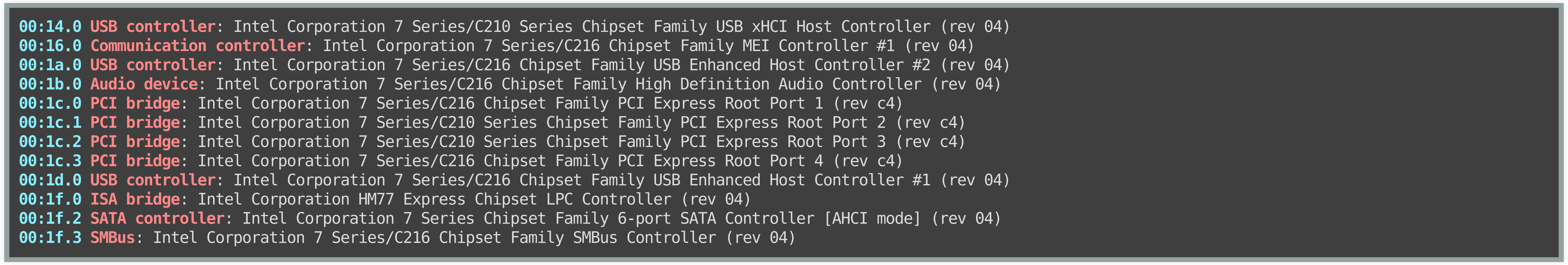
00:01.2

00: bus number (8 bits)

01: device number (5 bits)

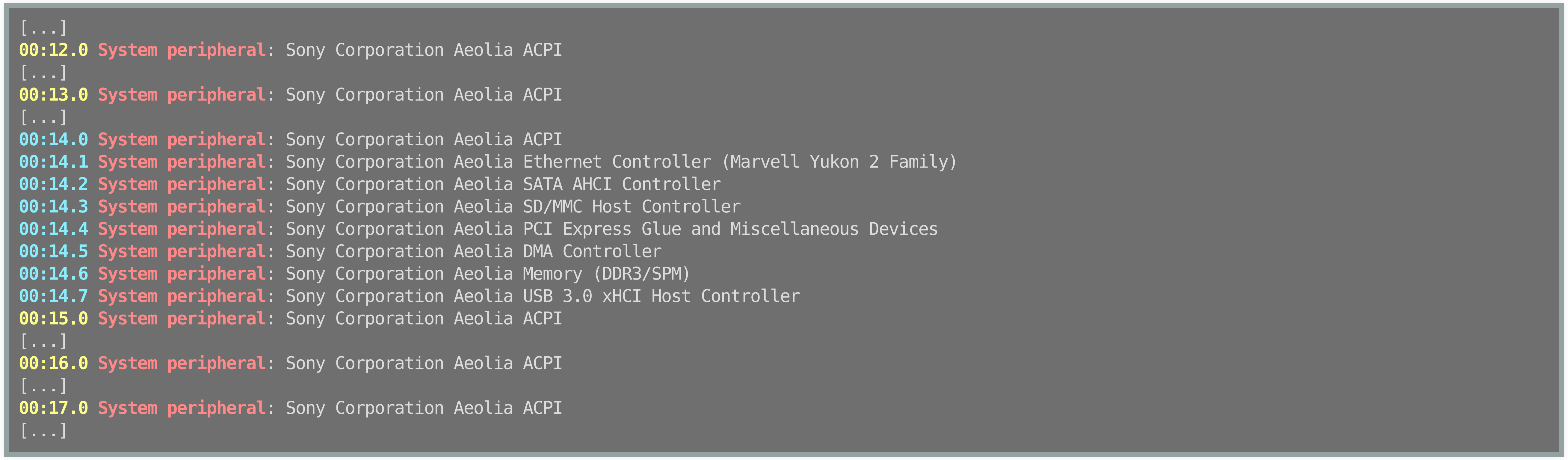
2: function number (3 bits)

INTEL “PANTHER POINT” PCH



```
00:14.0 USB controller: Intel Corporation 7 Series/C210 Series Chipset Family USB xHCI Host Controller (rev 04)
00:16.0 Communication controller: Intel Corporation 7 Series/C216 Chipset Family MEI Controller #1 (rev 04)
00:1a.0 USB controller: Intel Corporation 7 Series/C216 Chipset Family USB Enhanced Host Controller #2 (rev 04)
00:1b.0 Audio device: Intel Corporation 7 Series/C216 Chipset Family High Definition Audio Controller (rev 04)
00:1c.0 PCI bridge: Intel Corporation 7 Series/C216 Chipset Family PCI Express Root Port 1 (rev c4)
00:1c.1 PCI bridge: Intel Corporation 7 Series/C210 Series Chipset Family PCI Express Root Port 2 (rev c4)
00:1c.2 PCI bridge: Intel Corporation 7 Series/C210 Series Chipset Family PCI Express Root Port 3 (rev c4)
00:1c.3 PCI bridge: Intel Corporation 7 Series/C216 Chipset Family PCI Express Root Port 4 (rev c4)
00:1d.0 USB controller: Intel Corporation 7 Series/C216 Chipset Family USB Enhanced Host Controller #1 (rev 04)
00:1f.0 ISA bridge: Intel Corporation HM77 Express Chipset LPC Controller (rev 04)
00:1f.2 SATA controller: Intel Corporation 7 Series Chipset Family 6-port SATA Controller [AHCI mode] (rev 04)
00:1f.3 SMBus: Intel Corporation 7 Series/C216 Chipset Family SMBus Controller (rev 04)
```

MARVELL “AEOLIA”



```
[...]
00:12.0 System peripheral: Sony Corporation Aeolia ACPI
[...]
00:13.0 System peripheral: Sony Corporation Aeolia ACPI
[...]
00:14.0 System peripheral: Sony Corporation Aeolia ACPI
00:14.1 System peripheral: Sony Corporation Aeolia Ethernet Controller (Marvell Yukon 2 Family)
00:14.2 System peripheral: Sony Corporation Aeolia SATA AHCI Controller
00:14.3 System peripheral: Sony Corporation Aeolia SD/MMC Host Controller
00:14.4 System peripheral: Sony Corporation Aeolia PCI Express Glue and Miscellaneous Devices
00:14.5 System peripheral: Sony Corporation Aeolia DMA Controller
00:14.6 System peripheral: Sony Corporation Aeolia Memory (DDR3/SPM)
00:14.7 System peripheral: Sony Corporation Aeolia USB 3.0 xHCI Host Controller
00:15.0 System peripheral: Sony Corporation Aeolia ACPI
[...]
00:16.0 System peripheral: Sony Corporation Aeolia ACPI
[...]
00:17.0 System peripheral: Sony Corporation Aeolia ACPI
[...]
```

It clones itself across all PCI device numbers

8 FUNCTIONS AIN'T ENOUGH FOR EVERYBODY

00:14.4 “PCI Express Glue”

- PCIe bridge config
- MSI interrupt controller
- ICC
- HPET
- Flash controller
- RTC
- Timers
- 2 serial ports
- I^{2}C

LINUX MINIMUM SYSTEM REQUIREMENTS

- A timer (PIT)
- Interrupts (PIC)
- Some kind of console

PS4: no PIT, no PIC, no standard serial

Board has testpoints for an 8250-derived serial port

DMESG PLZ

Linux earlycon: early console for debugging

No IRQs required

```
console=uart8250,mmio32,0xd0340000,3200n8
```

Clock is different… 3200 means 115200

This gets us a boot log

TIME STAMP COUNTER (TSC)

Newfangled timer, in-CPU

PS4 Liverpool APU supports proper TSC

Linux tries to calibrate it...

... against PIC or PMTIMER

Fail

AGAIN, IT REALLY ISN'T A PC

```
 enum {
         X86_SUBARCH_PC = 0,
         X86_SUBARCH_LGUEST,
         X86_SUBARCH_XEN,
         X86_SUBARCH_INTEL_MID,
         X86_SUBARCH_CE4100,
+        X86_SUBARCH_PS4,
         X86_NR_SUBARCHS,
 };
```

Subarch specified by bootloader (ps4-kexec)

Enables custom TSC calibration code

Disables legacy PIC and RTC

ACPI

NOT JUST “POWER”

Needed for proper PCI config, IOMMU, CPU frequency scaling…

PS4 has broken ACPI tables…

Fix them in ps4-kexec

PCI MSI 101

MESSAGE SIGNALED INTERRUPTS

- Device configuration registers for address and value
- To fire an interrupt, devices write a value to an address
- CPU IRQ controller (LAPIC) receives and fires interrupt vector
- The message value directly defines the CPU IRQ vector

AEOLIA MSI 101

- Device MSI configuration registers ignored
- Function 4 ("glue") implements custom MSI controller
- Each function gets shared addr and top 27 bits of data
- Each "sub-function" only gets separate bottom 5 bits
- All MSIs originate from Function 4

(□□□□)□□┻━┻

DRIVER HELL

- Sibling devices are inter-dependent
- Linux IRQ vector allocation not sequential
- Need to modify all drivers to use custom IRQ code

AEOLIA ON LINUX

- Core driver implements IRQ controller interface
- Linux probe-defer mechanism to fix ordering issue
- Some drivers (SDHCI, GigE) modified to request Aeolia IRQs
- Some drivers (serial, USB) instantiated from wrappers
- Each function uses a single shared IRQ :(

IOMMU TO THE RESCUE

- Allows interrupt remapping
- Consecutive message numbering
- Can use unique IRQs per sub-function :)
- Falls back to shared IRQs if IOMMU off
- The ACPI table for the IOMMU is missing :(

CHECKLIST

✓ IRQs (apcie)

✓ Timer (TSC)

✓ Early serial

✓ Late serial with IRQs (apcie-uart)

✓ Initramfs userspace

✗ Serial I/O hangs sometimes :(

MORE CLEANUP NEEDED

FreeBSD masks some IRQ vectors on CPU#0 with nonstandard AMD LAPIC features

Clean them up in ps4-kexec

✓ Serial is stable

This took *ages* to debug

JUST ADD DRIVERS

✓ USB xHCI (3 USB controllers in one function…)

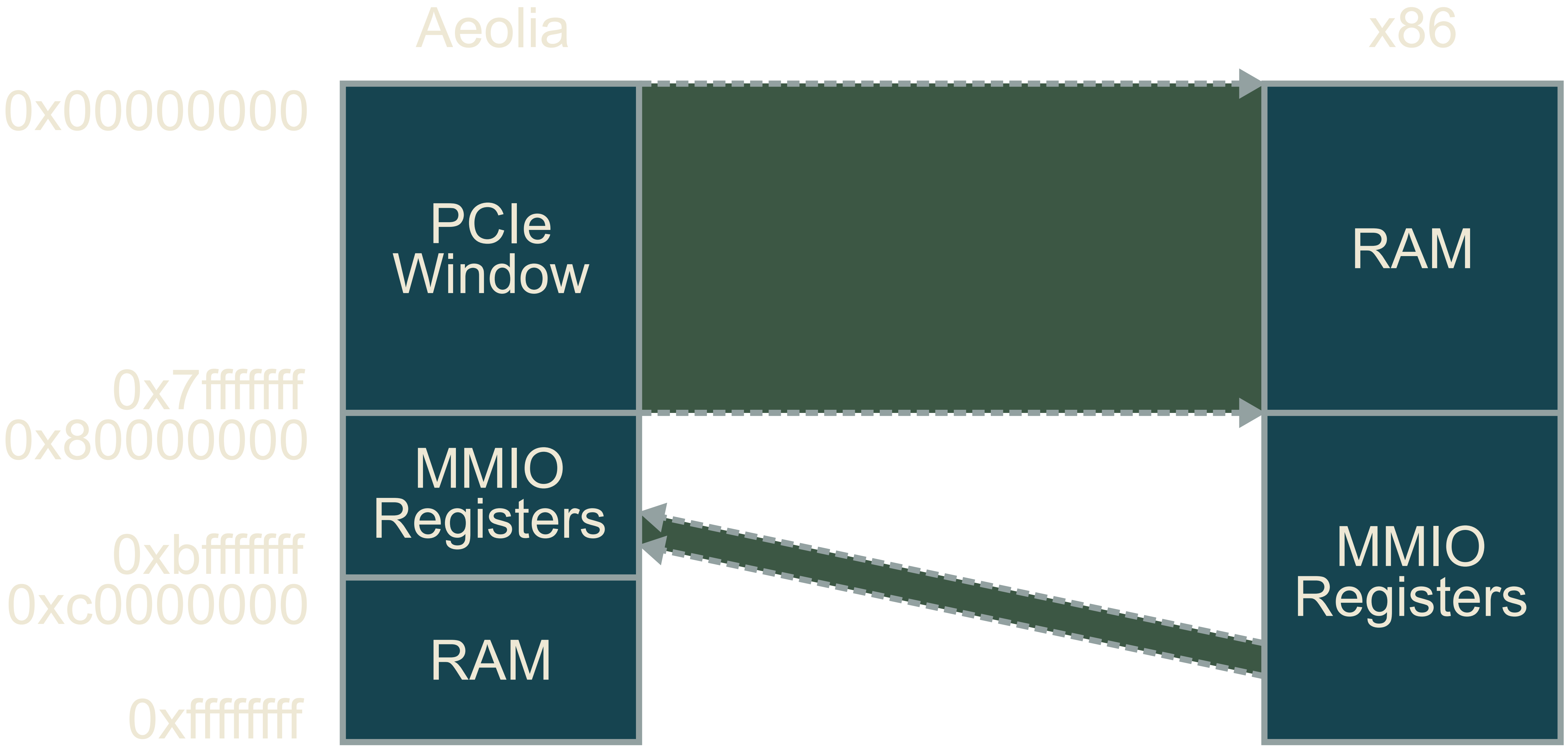
✓ SDHCI (Nonstandard PCI config, needs quirks…)

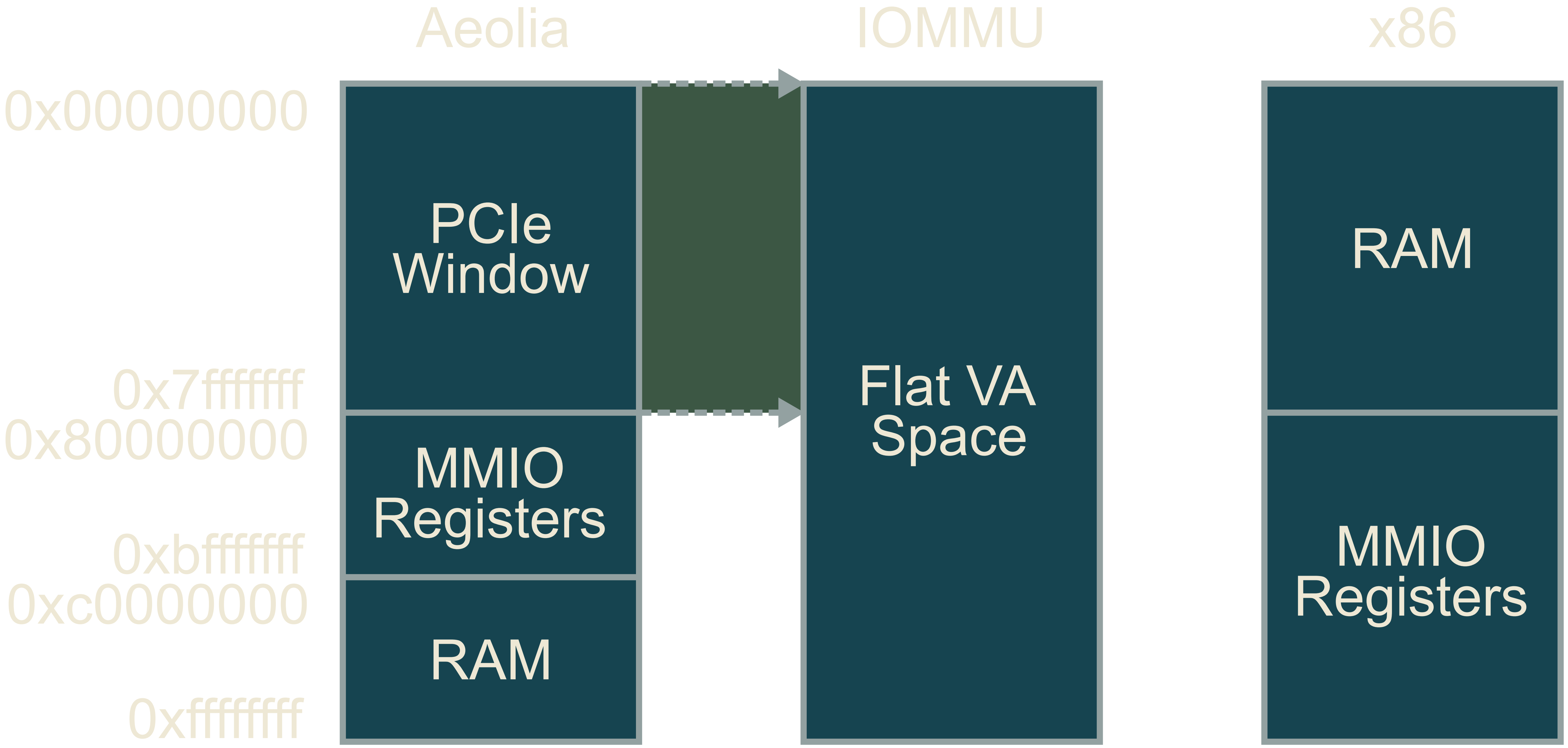
✓ Ethernet (Driver needs hacks; still partially broken…)

Worked fine on Linux 4.4

Failed on 4.9 - DMA broken?

AEOLIA STRIKES BACK


Aeolia
x86
0x00000000
PCIe Window
RAM
0x7fffffff
0x80000000
MMIO Registers
0xbfffffff
0xc0000000
RAM
0xffffffff
MMIO Registers



Aeolia

IOMMU

x86

0x00000000

0x7fffffff

0x80000000

0xbfffffff

0xc0000000

0xffffffff

PCIe Window

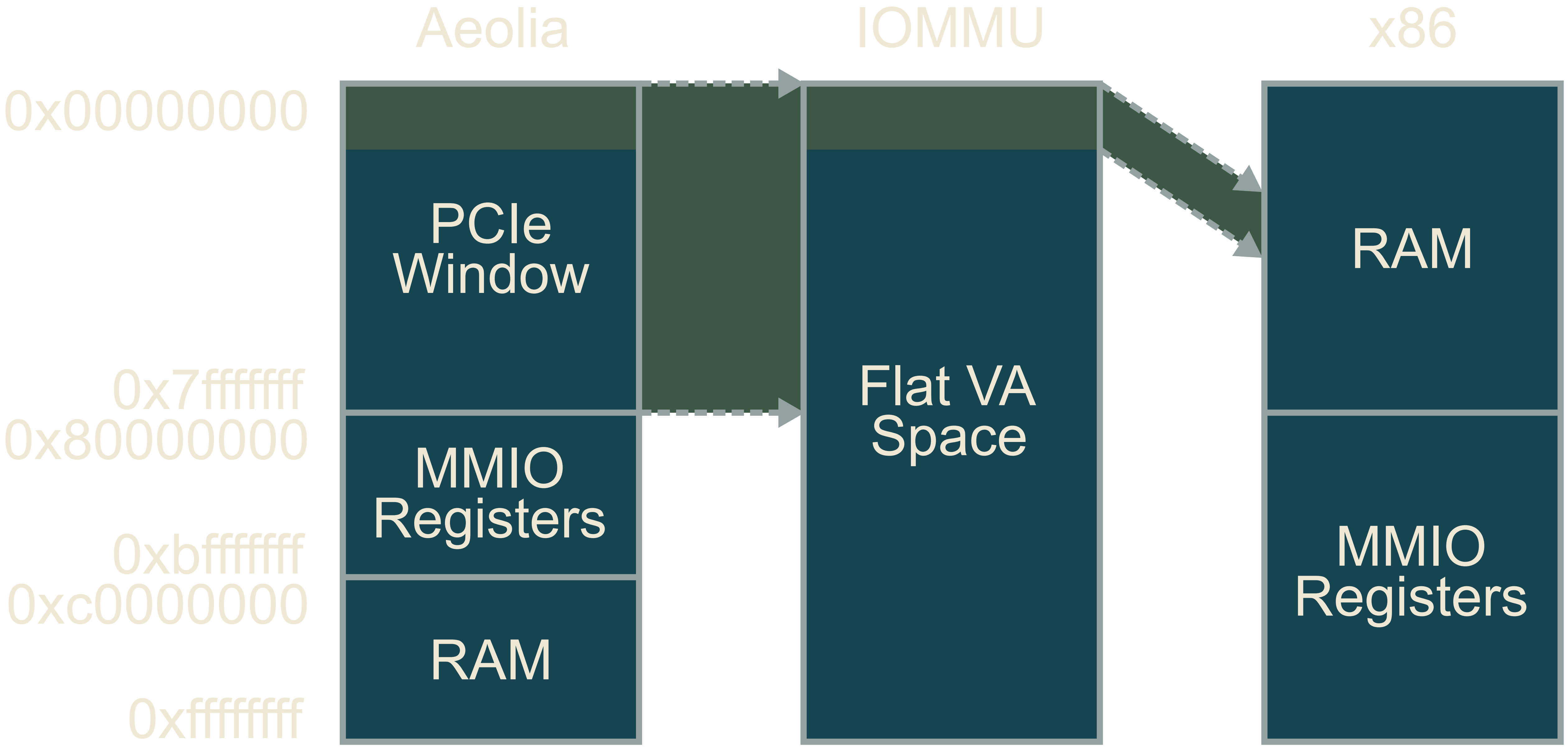
MMIO Registers

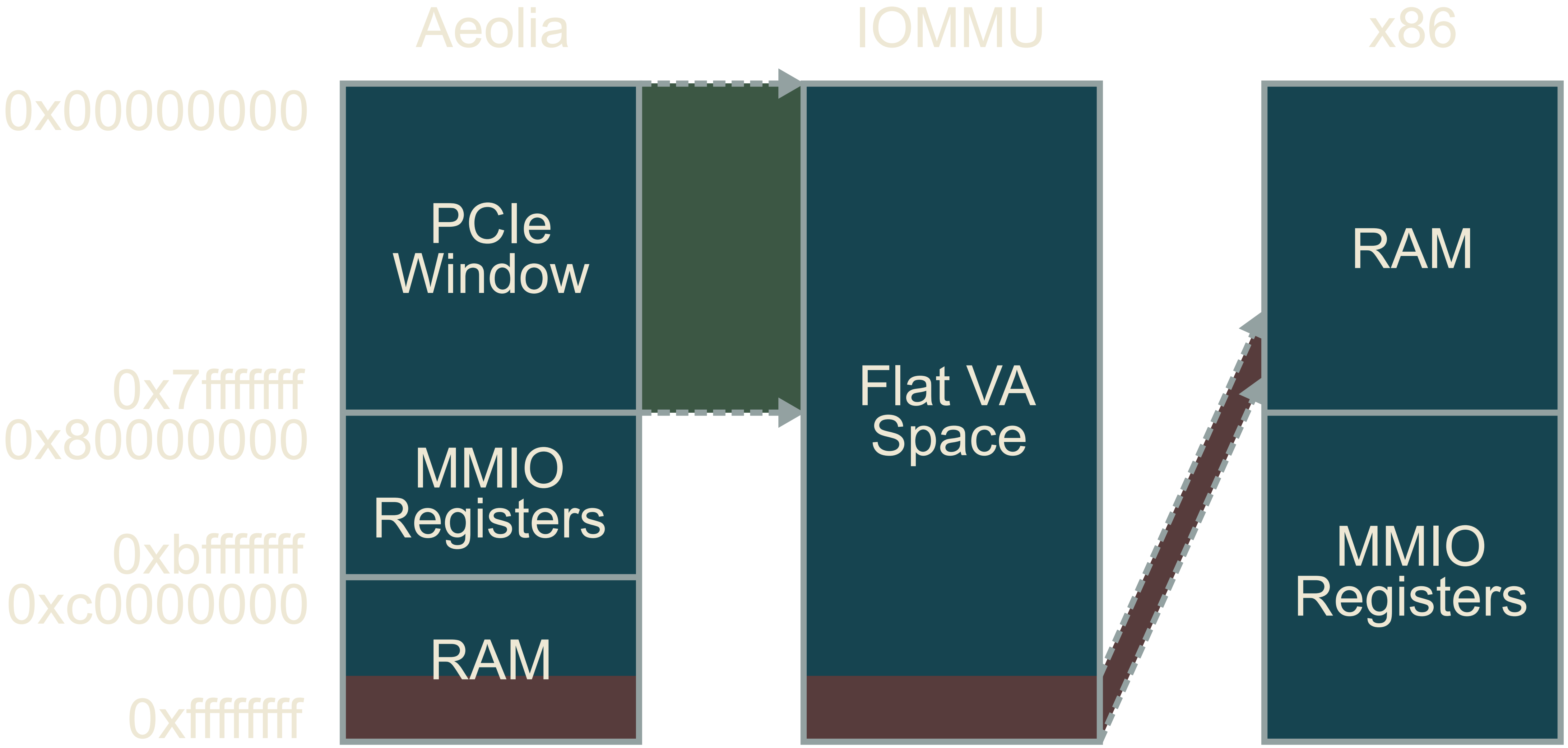
RAM

Flat VA Space

RAM

MMIO Registers


Aeolia
IOMMU
x86
0x00000000
PCIe Window
0x7fffffff
0x80000000
MMIO Registers
0xbfffffff
0xc0000000
RAM
0xffffffff
Flat VA Space
RAM
MMIO Registers


Aeolia
IOMMU
x86
0x00000000
0x7fffffff
0x80000000
0xbfffffff
0xc0000000
0xffffffff
PCIe Window
MMIO Registers
RAM
Flat VA Space
RAM
MMIO Registers

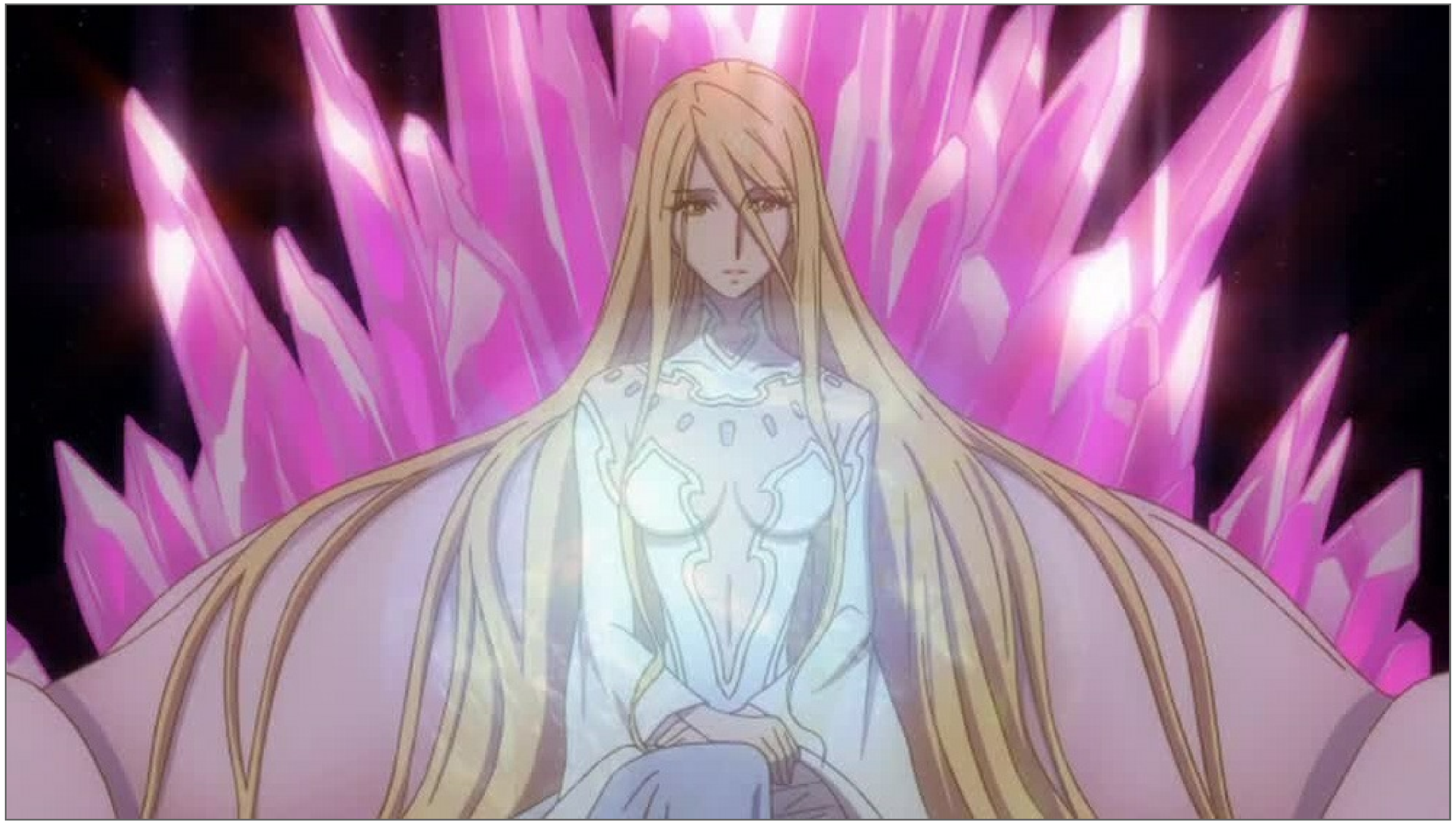
31-BIT DMA

┻━┻ □□(`Д´)ﾉ□ ┻━┻

More Linux driver patching…

AND NOW FOR SOMETHING COMPLETELY DIFFERENT

PS4 RADEON GRAPHICS



“STARSHA”

```
dce_ihdef_get_info_crtc_linea_liverpool
"LVP A0" StarshaAsicStateRegInfo
ThJStarsha AGESAThebeJBDK
```

Nobody (not even Sony/AMD) agrees on the APU codename

We're calling it Liverpool

LIVERPOOL GRAPHICS

- AMD GCN “Sea Islands” (CI) GPU
- Similar to other chips in the generation
- Some quirks, customizations, oddities
- We used Bonaire as a base

HACKING ON AMD DRIVERS

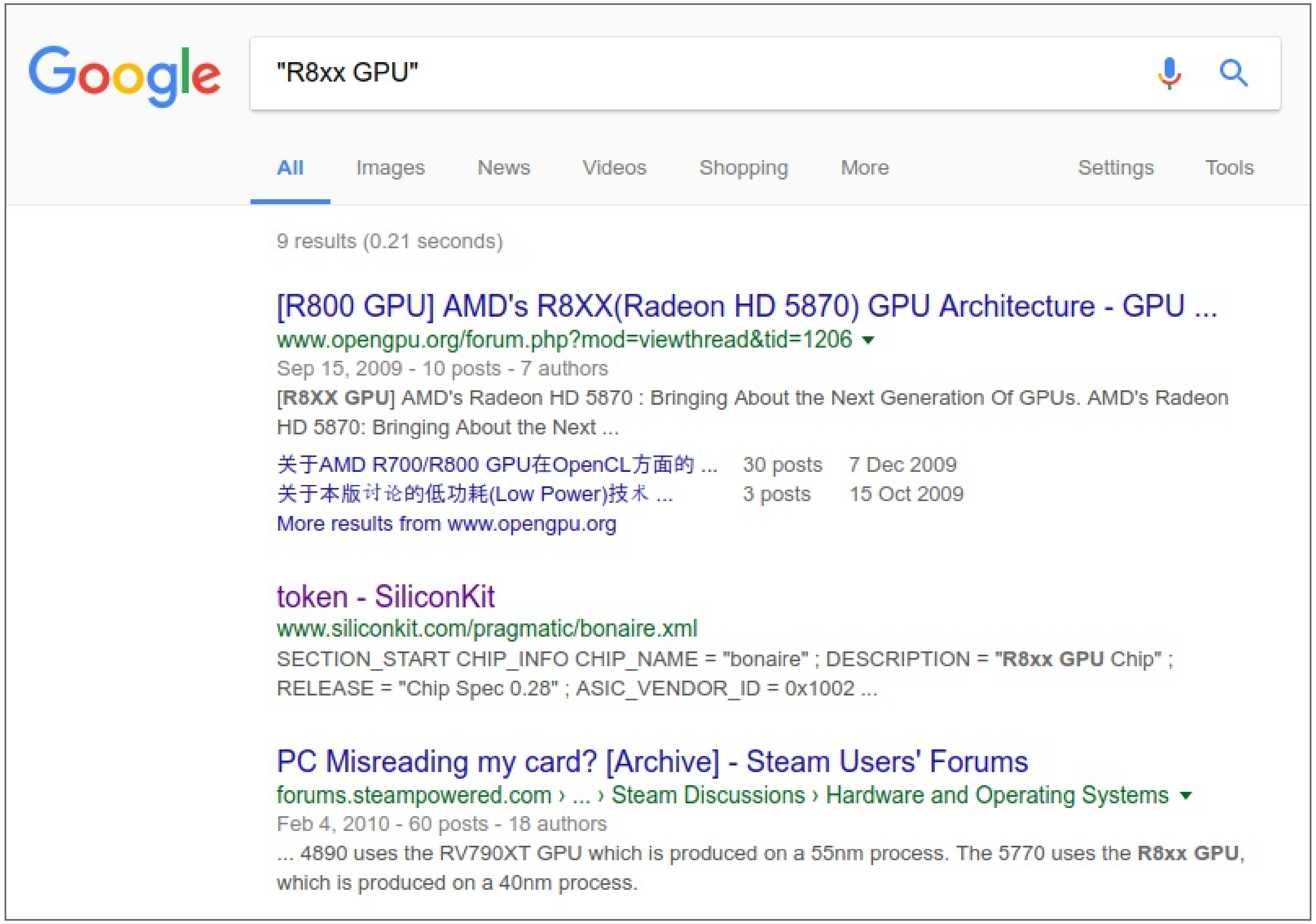
AMD publishes 3D shader and command queue documentation

They do *NOT* publish register docs for recent GPUs

That's what we need to hack on kernel drivers :(

“The code is the documentation” - incomplete, magic numbers

GOOGLE TIME



9 results (0.21 seconds)

[R800 GPU] AMD's R8XX(Radeon HD 5870) GPU Architecture - GPU ...
www.opengpu.org/forum.php?mod=viewthread&tid=1206
Sep 15, 2009 - 10 posts - 7 authors
[**R8XX GPU**] AMD's Radeon HD 5870 : Bringing About the Next Generation Of GPUs. AMD's Radeon HD 5870: Bringing About the Next ...

关于AMD R700/R800 GPU在OpenCL方面的 ...	30 posts	7 Dec 2009
关于本版讨论的低功耗(Low Power)技术 ...	3 posts	15 Oct 2009

More results from www.opengpu.org

token - SiliconKit
www.siliconkit.com/pragmatic/bonaire.xml
SECTION_START CHIP_INFO CHIP_NAME = "bonaire" ; DESCRIPTION = "**R8xx GPU** Chip" ; RELEASE = "Chip Spec 0.28" ; ASIC_VENDOR_ID = 0x1002 ...

PC Misreading my card? [Archive] - Steam Users' Forums
forums.steampowered.com › ... › Steam Discussions › Hardware and Operating Systems
Feb 4, 2010 - 60 posts - 18 authors
... 4890 uses the RV790XT GPU which is produced on a 55nm process. The 5770 uses the **R8xx GPU**, which is produced on a 40nm process.

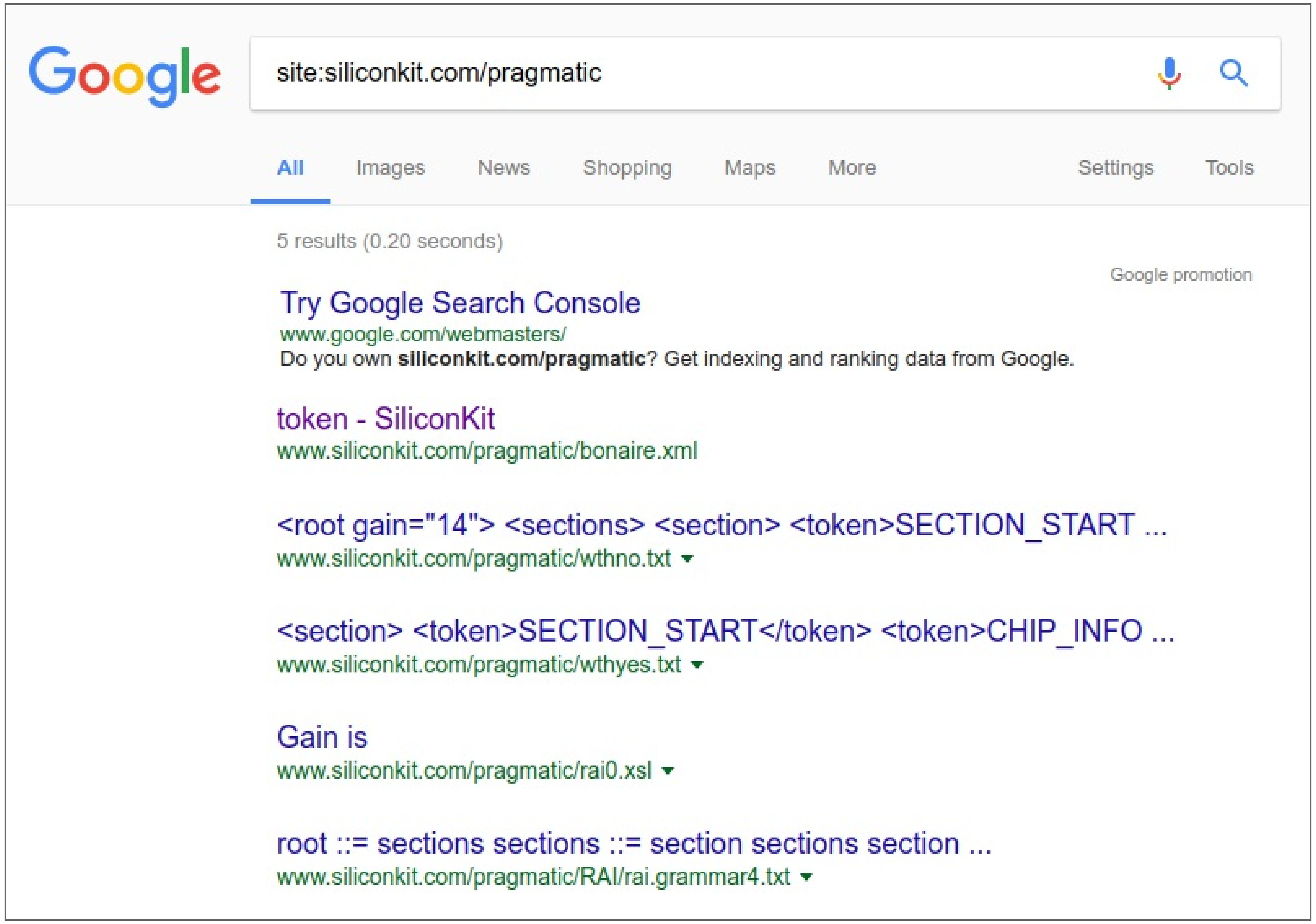
http://www.siliconkit.com/pragmatic/bonaire.xml

XML dump of Bonaire register documentation?

```
    <field>
        <fname>
            <token>P_ALWAYS_USE_FAST_TXCLK</token>
        </fname>
        <frange>
            <token>13:13</token>
        </frange>
        <ftype>
            <token>ALPHA</token>
            <token>{</token>
            <fieldtexts>
                <fieldtext>
                    <quoted>
                        <token>"TXCLK will be either 250MHz, 500MHz, or 1GHz
                               depends on port speeds "</token>
                    </quoted>
```

Broken, incomplete

TELL ME MORE



5 results (0.20 seconds)

Google promotion

Try Google Search Console
www.google.com/webmasters/
Do you own **siliconkit.com/pragmatic**? Get indexing and ranking data from Google.

token - SiliconKit
www.siliconkit.com/pragmatic/bonaire.xml

<root gain="14"> <sections> <section> <token>SECTION_START ...
www.siliconkit.com/pragmatic/wthno.txt ▾

<section> <token>SECTION_START</token> <token>CHIP_INFO ...
www.siliconkit.com/pragmatic/wthyes.txt ▾

Gain is
www.siliconkit.com/pragmatic/rai0.xsl ▾

root ::= sections sections ::= section sections section ...
www.siliconkit.com/pragmatic/RAI/rai.grammar4.txt ▾

WHAT IS RAI

http://www.siliconkit.com/pragmatic/RAI/rai.grammar4.txt

```
root ::= sections
sections ::= section sections
section ::= 'SECTION_START' 'CHIP_INFO' statements 'SECTION_END'
section ::= 'SECTION_START' 'CHIP_SPACES' chipspaces 'SECTION_END'
section ::= 'SECTION_START' 'CHIP_STREAMS' '[a-zA-Z0-9_]*' 'SECTION_END'
section ::= 'SECTION_START' 'CHIP_MEMORIES' '[a-zA-Z0-9_]*' 'SECTION_END'
section ::= 'SECTION_START' 'CHIP_PARAMETERS' '[a-zA-Z0-9_]*' 'SECTION_END'
section ::= 'SECTION_START' 'BLOCK_INFO' statements 'SECTION_END'
section ::= 'SECTION_START' 'BLOCK_REGISTERS' register 'SECTION_END'
register ::= title spaces size rattribute '{' fields '}' ';'
register ::= title spaces size '{' fields '}' ';'
title ::= '[a-zA-Z0-9_]*'
spaces ::= space spaces
[...]
```

AMD internal register description file?

HMMMMM…

http://www.siliconkit.com/pragmatic/bonaire.xml

http://www.siliconkit.com/pragmatic/RAI/rai.grammar4.txt

Maybe…

http://www.siliconkit.com/pragmatic/bonaire.rai

Nope

http://www.siliconkit.com/pragmatic/RAI/bonaire.rai

BINGO

```
//Version 1.0.1.0
//CL# 890079
//Version 1.0.0.0
//CL# 883050

SECTION_START CHIP_INFO

CHIP_NAME = "bonaire";
DESCRIPTION = "R8xx GPU Chip";
RELEASE = "Chip Spec 0.28";
// Edit Vendor ID Here: Default(0xFFFF) means search for all
ASIC_VENDOR_ID = 0x1002;
[...]
```

So I wrote a *working* parser

```
$ python showregname.py HDP_NONSURFACE_INFO
HDP_NONSURFACE_INFO (GpuF0Reg:0x2c08,GpuF1Reg:0x2c08) 32bit:
       0  NONSURF_ADDR_TYPE
          - 0: physical address with no translation.
          - 1: virtual address, requires page table translation.
     4:1  NONSURF_ARRAY_MODE
          - 0: ARRAY_LINEAR_GENERAL: Unaligned linear array
          - 1: ARRAY_LINEAR_ALIGNED: Aligned linear array
[...]
```

Also does annotated register dumps, diffs, #define generation

4000+ registers documented in GpuF0Reg alone

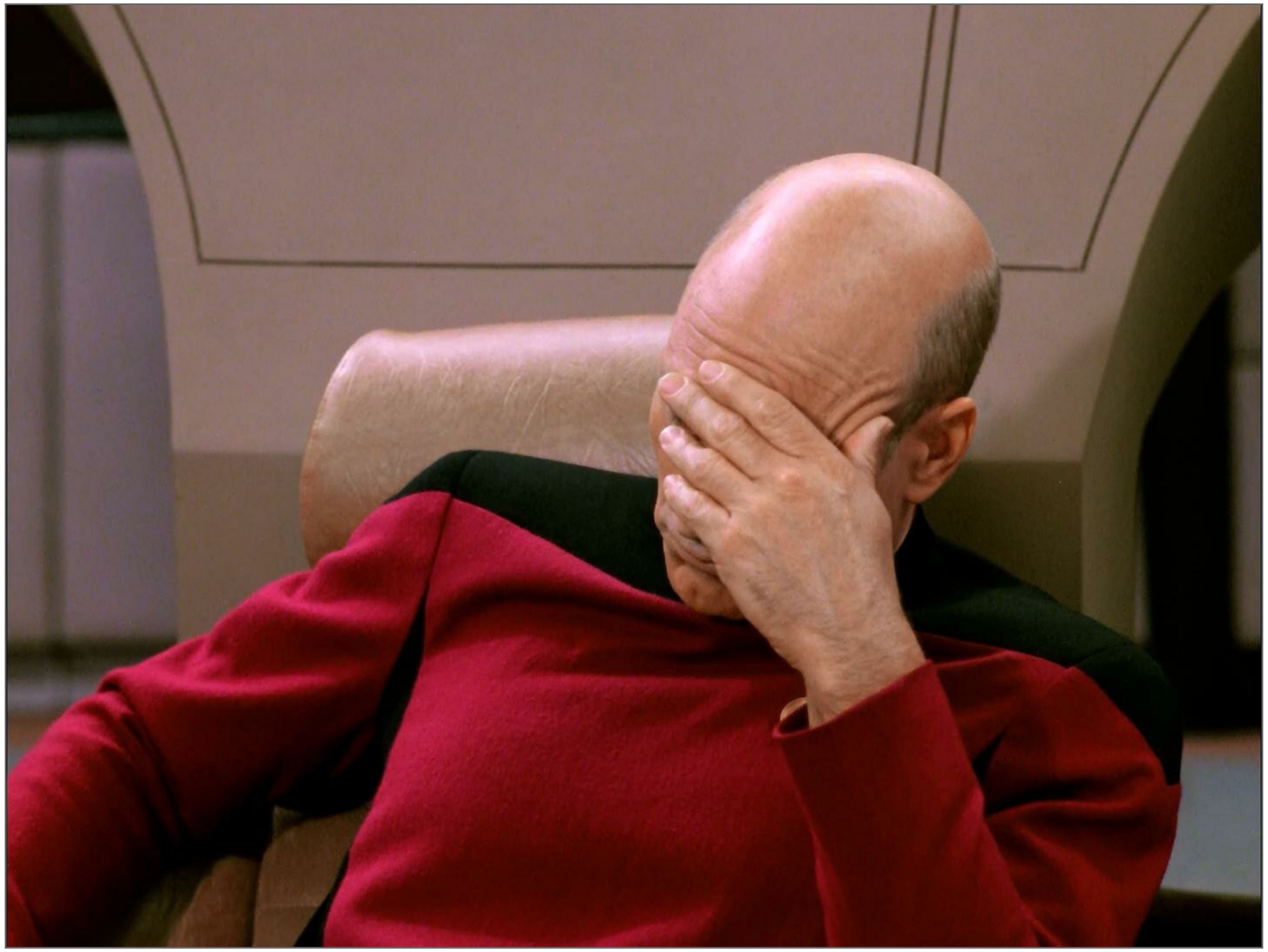
ROAD TO THE FRAMEBUFFER

HDMI IS EASY, RIGHT?

- GPU has HDMI, DisplayPort ports
- HDMI *not* connected; DP connected

????

EXTERNAL HDMI ENCODER



WE MUST GO DEEPER

Panasonic I^{2}C DisplayPort → HDMI bridge

Requires configuration to work

Hooked up to the GPU I^{2}C bus?

You wish

ICC

- RPC protocol used to send commands to system MCU
- Message box / doorbell protocol
- Accessed via Aeolia
- Used for things like power, buttons, LEDs…
- And the HDMI encoder I^{2}C

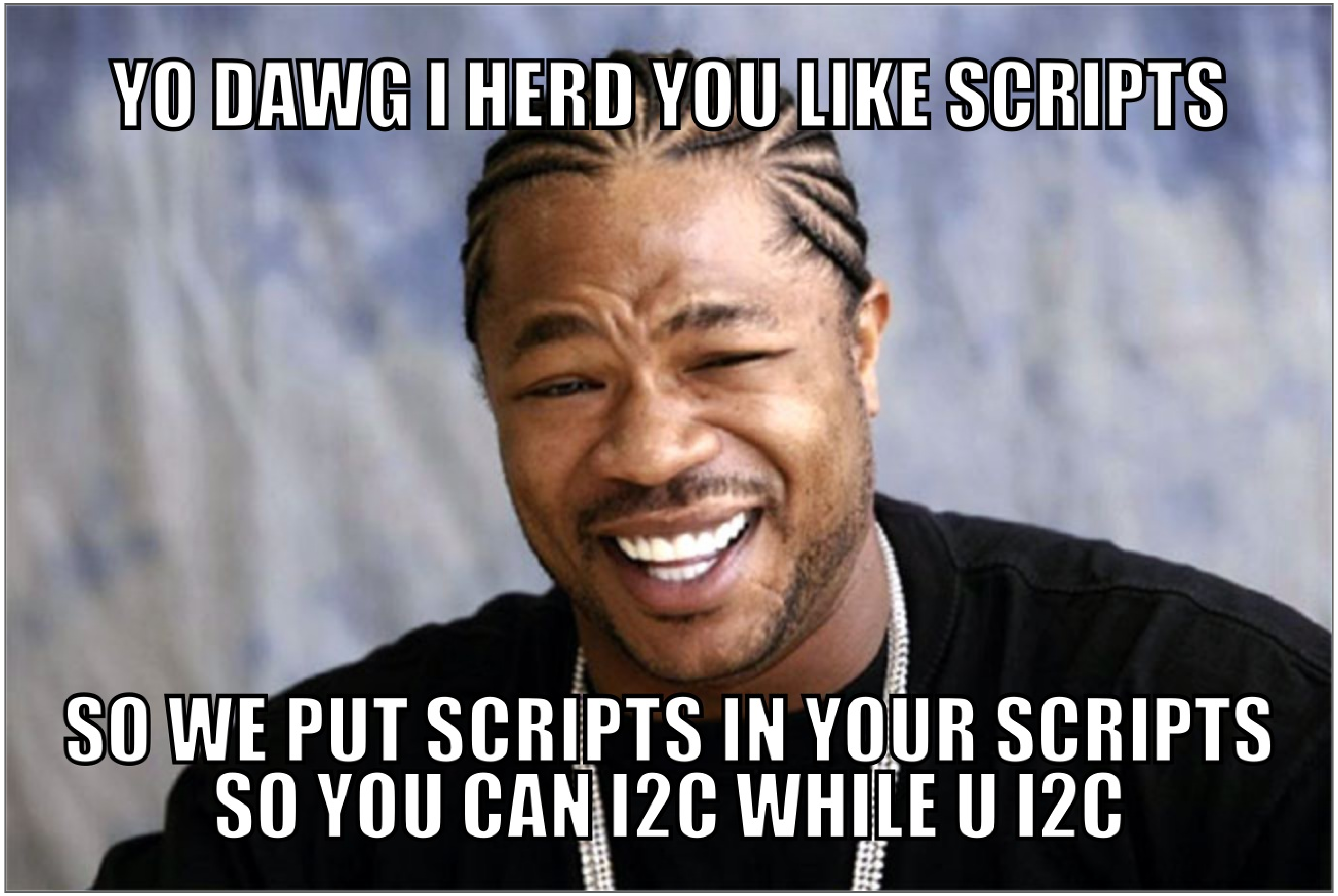
ICC I^{2}C

Let's build a simple I^{2}C interface?

Nah, let's make a bytecode scripting engine to issue I^{2}C commands

WHY?!?

Because ICC is too slow to issue requests one by one


YO DAWG I HERD YOU LIKE SCRIPTS
SO WE PUT SCRIPTS IN YOUR SCRIPTS
SO YOU CAN I2C WHILE U I2C

MORE HACKS...

- HDMI encoder requires all 4 DisplayPort lanes active
- Scanout memory bandwidth calculation is broken
- Mouse cursor size is from previous generation (wat?)

✓ Framebuffer console working

✗ X won't start with radeon driver

A TALE OF TWO MEMORIES

PS4 uses a unified memory architecture

Linux legacy driver expectes a usable amount of "video" memory

PS4 configures emulated VRAM as 16MiB...

Solution: reconfigure memory controller in ps4-kexec to assign 1GiB of RAM as VRAM

✓ X starts

IT'S 3D TIME

RADEON GPU 101

Commands are sent to the GPU by putting them in rings:

- Graphics ring
- Compute rings
- DMA rings

Commands are processed by the GPU Command Processor

It contains multiple sub-units (ME, PFP, CE), each of which is a custom ‘F32’ CPU running microcode firmware

Rings can call out to Indirect Buffers (IBs) with more commands

radeon: ring 0 test failed

The graphics ring isn't working

```
WREG32(scratch, 0xCAFEDEAD);
radeon_ring_lock(rdev, ring, 3);
radeon_ring_write(ring, PACKET3(PACKET3_SET_UCONFIG_REG, 1));
radeon_ring_write(ring, ((scratch - PACKET3_SET_UCONFIG_REG_START) >> 2));
radeon_ring_write(ring, 0xDEADBEEF);
radeon_ring_unlock_commit(rdev, ring, false);
```

The ring test writes to a GPU register from the ring, then checks to see if the write happened

Debug registers (thanks bonaire.rai!) show the CP is stuck...

... waiting for data in the ring...

... after a NOP command?

NOP IS HARD, LET'S GO STALLING

Packet headers have a length field of `size - 2`

2-word packet: size = 0.

They added a 1-word NOP: size = 0x3fff (-1)

Old microcode... interprets it as a huge packet

Hawaii has the same issue on old microcode:

```
if (rdev->family == CHIP_HAWAII) {
    if (rdev->new_fw)
        nop = PACKET3(PACKET3_NOP, 0x3FFF);
    else
        nop = RADEON_CP_PACKET2;
} else {
    nop = PACKET3(PACKET3_NOP, 0x3FFF);
}
```

radeon: ring 3 test failed

That's the SDMA ring

```
radeon_ring_write(ring, SDMA_PACKET(SDMA_OPCODE_WRITE,
                         SDMA_WRITE_SUB_OPCODE_LINEAR, 0));
radeon_ring_write(ring, lower_32_bits(gpu_addr));
radeon_ring_write(ring, upper_32_bits(gpu_addr));
radeon_ring_write(ring, 1); /* number of DWs to follow */
radeon_ring_write(ring, 0xDEADBEEF);
```

Same idea: write a value to memory, check for it

Debugging, the write happens... but it writes zero?

DOUBLE IT UP

So I tried queuing two writes instead:

```
radeon_ring_write(ring, SDMA_PACKET(SDMA_OPCODE_WRITE,
                         SDMA_WRITE_SUB_OPCODE_LINEAR, 0));
radeon_ring_write(ring, lower_32_bits(gpu_addr));
radeon_ring_write(ring, upper_32_bits(gpu_addr));
radeon_ring_write(ring, 1); /* number of DWs to follow */
radeon_ring_write(ring, 0xDEADBEEF);                       <-- What it *should*
radeon_ring_write(ring, SDMA_PACKET(SDMA_OPCODE_WRITE,         write
                         SDMA_WRITE_SUB_OPCODE_LINEAR, 0));
radeon_ring_write(ring, lower_32_bits(gpu_addr2));
radeon_ring_write(ring, upper_32_bits(gpu_addr2));
radeon_ring_write(ring, 1); /* number of DWs to follow */  <-- What it writes
radeon_ring_write(ring, 0x0BADF00D);
```

Now it writes... 1 to the first destination?

SDMA: OFF-BY-FOUR

Linear writes from the ring start 4 words too late in the ring

IBs work fine, only the ring is broken

Workaround: use FILL opcode instead:

```
radeon_ring_write(ring, SDMA_PACKET(SDMA_OPCODE_CONSTANT_FILL, 0,
                         SDMA_CONSTANT_FILL_EXTRA_SIZE(2)));
radeon_ring_write(ring, lower_32_bits(gpu_addr2));
radeon_ring_write(ring, upper_32_bits(gpu_addr2));
radeon_ring_write(ring, 0xDEADBEEF); /* Fill value */
radeon_ring_write(ring, 4); /* number of bytes */
```

STILL NO WORKY

Can't write to pagetable config registers via GPU commands :(

Linux uses this to configure pagetables

Special register firewall in Liverpool? Security?

Workaround by directly writing from CPU, but it sucks

Maybe the register firewall is in the firmware?

SPEAKING OF FIRMWARE

The Command Processor blocks require “microcode”

Thus far undocumented

We pull the firmware blobs from FreeBSD in ps4-kexec and pass them in initramfs (avoids redistribution issues)

Let's dig deeper

REVERSING CPU ARCHITECTURES 101

1. Guess an instruction
2. Try running it
3. See what it did
4. GOTO 1

We can upload custom F32 firmware easily and have it write to scratch regs, then read what it wrote

The basic "write to GPU reg" instruction is easy to find from GPU register offsets, in the microcode blobs

F32DIS

Disassembler for the AMD proprietary ‘F32’ GPU microcode

```
CLEAR_STATE:
  5e  cc800000 | stw r2, [r0, #0x0]
  5f  cc400000 | stw r1, [r0, #0x0]
  60  cc000016 | stw r0, [r0, #0x16]
  61  80000672 | b 0x672

INDEX_BUFFER_SIZE:
  62  cc40002d | stw r1, [r0, #0x2d]
  63  7c408001 | mov r2, r1
  64  88000000 | btab
```

Instruction syntax shamelessly stolen from ARM

Not complete, but disassembles all instructions used in Liverpool and Bonaire firmwares for PFP, ME, CE, MEC, RLC

ALAS

Register blocking not in the firmware

It seems it is blocked in hardware, when issued from GFX block (debug registers show an access violation)

Haven't found how to turn it off yet

3D does work with the CPU write workaround, though!

CURRENT CHECKLIST

✓ IRQs / Timer

✓ Serial port

✓ Shutdown / reboot

✓ Power button

✓ USB

✓ HDD

✓ Blu-Ray

✓ WiFi

✓ Bluetooth

✓ Blinkenlights

✓ Ethernet (mostly)

✓ Framebuffer / KMS

✓ HDMI (basic)

✓ 3D (with ugly hack)

✓ S/PDIF audio

✗ HDMI audio

✗ RTC

CODE

github.com/failoverflow/ps4-kexec

- kexec and hardware reconfiguration / "bootloader" code

github.com/failoverflow/ps4-linux

- Kernel tree

github.com/failoverflow/ps4-radeon-patches

- Userspace library patches

github.com/failoverflow/radeon-tools

- f32dis and RAI tools

http://fail0verflow.com · @fail0verflow

DEMO TIME!

Well…