Multi-ISA Firmware Compatibility
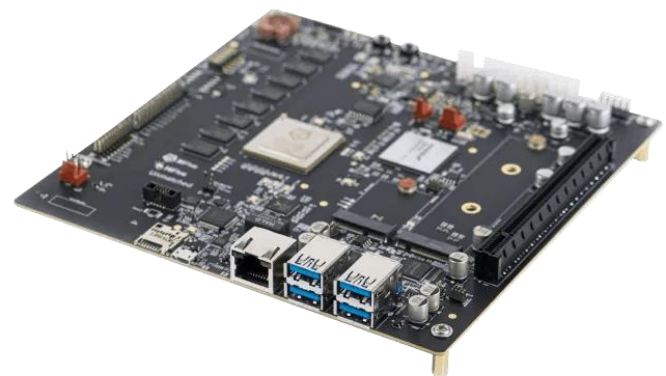## Bringing RISC-V and IHV Ecosystems Together

Andrei Warkentin

June 2023

intel.

Background

Typical RISC-V
SBC

Off-the-shelf PCIe adapter
(NIC/IPU, GPU, RAID)

Nothing

Background

❖ **Today** ➔ Non-standard RISC-V platforms, that don't lend to building to horizontal market segments where interoperability is key.

   ❖ Single-vendor.

   ❖ Embedded hardware with little extensibility.

   ❖ Embedded firmware and OS with all the drivers baked in.

   ❖ "I plugged a PCIe NIC in, but couldn't figure out how to network boot my OS"

❖ **Tomorrow** ➔ An interoperable RISC-V ecosystem that allows building servers and PCs.

   ❖ Many vendors come together for a solution.

   ❖ PCIe/CXL connectivity for off-the-shelf devices.

   ❖ Rich UEFI + ACPI firmware experience.

   ❖ Same OS image can boot across different platforms, SoC, IP implementations.
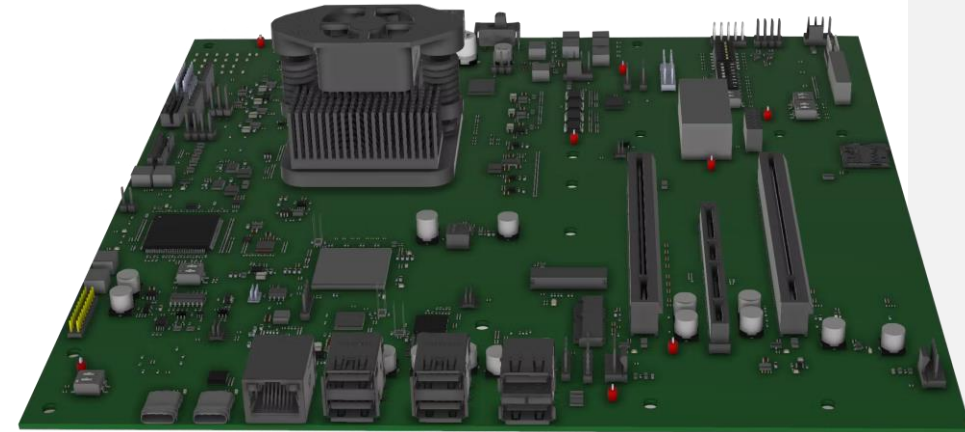
How to make existing PCIe devices work?

Will future PCIe devices ship with RISC-V firmware drivers?
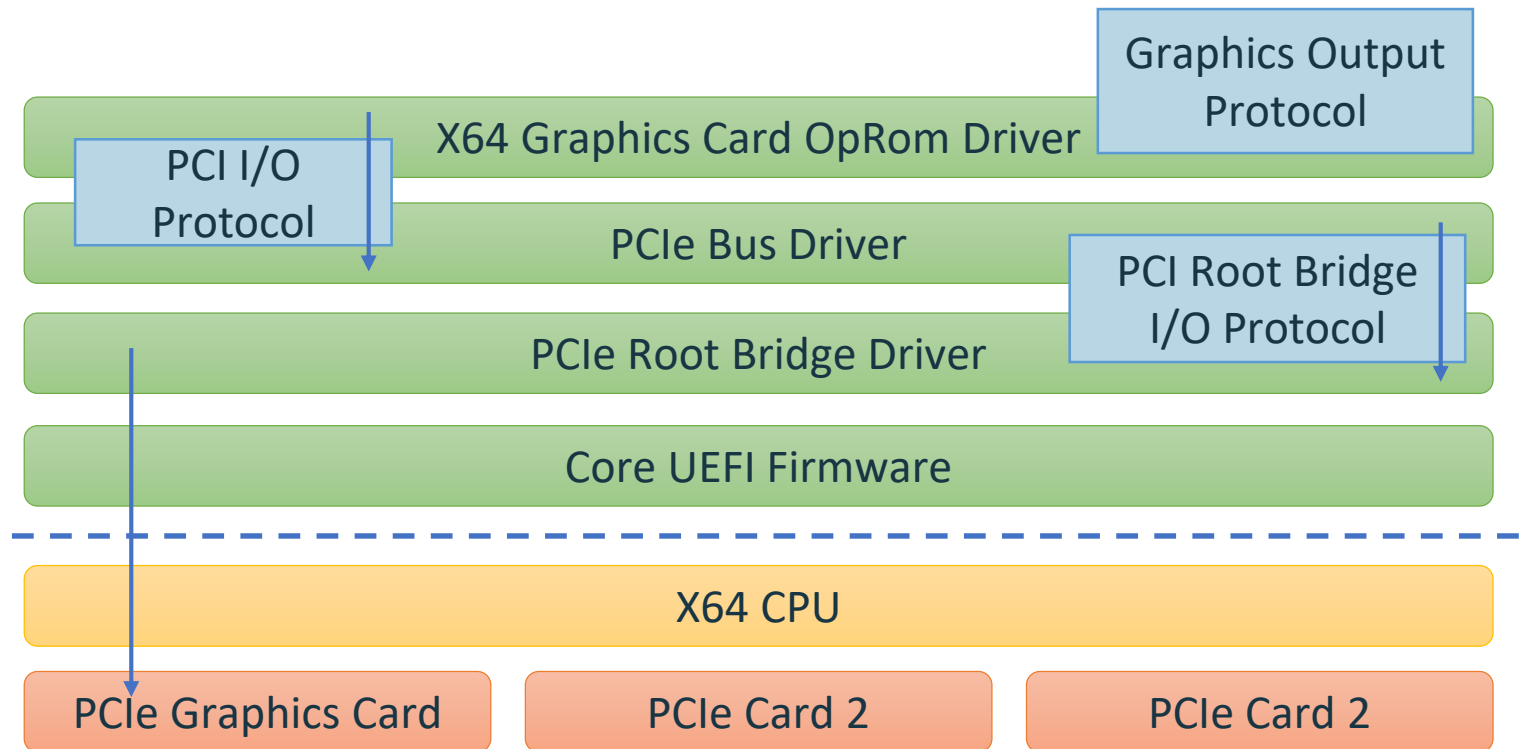
PCIe Firmware Drivers

Device ID		Vendor ID	00
Status		Command	04
Class Code		Revision ID	08
BIST	Header Type	Master Latency Timer	Cache Line Size	0C
Base Address Registers			10
			14
			18
			1C
			20
			24
Cardbus CIS Pointer			28
Subsystem ID		Subsystem Vendor ID	2C
Expansion ROM Base Address			30
Reserved		Capabilities Pointer	34
Reserved			38
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	3C

31 ... 0

PCI Expansion ROM Image 0

PCI Expansion ROM Image 1

.
.
.
.
.

PCI Expansion ROM Image N

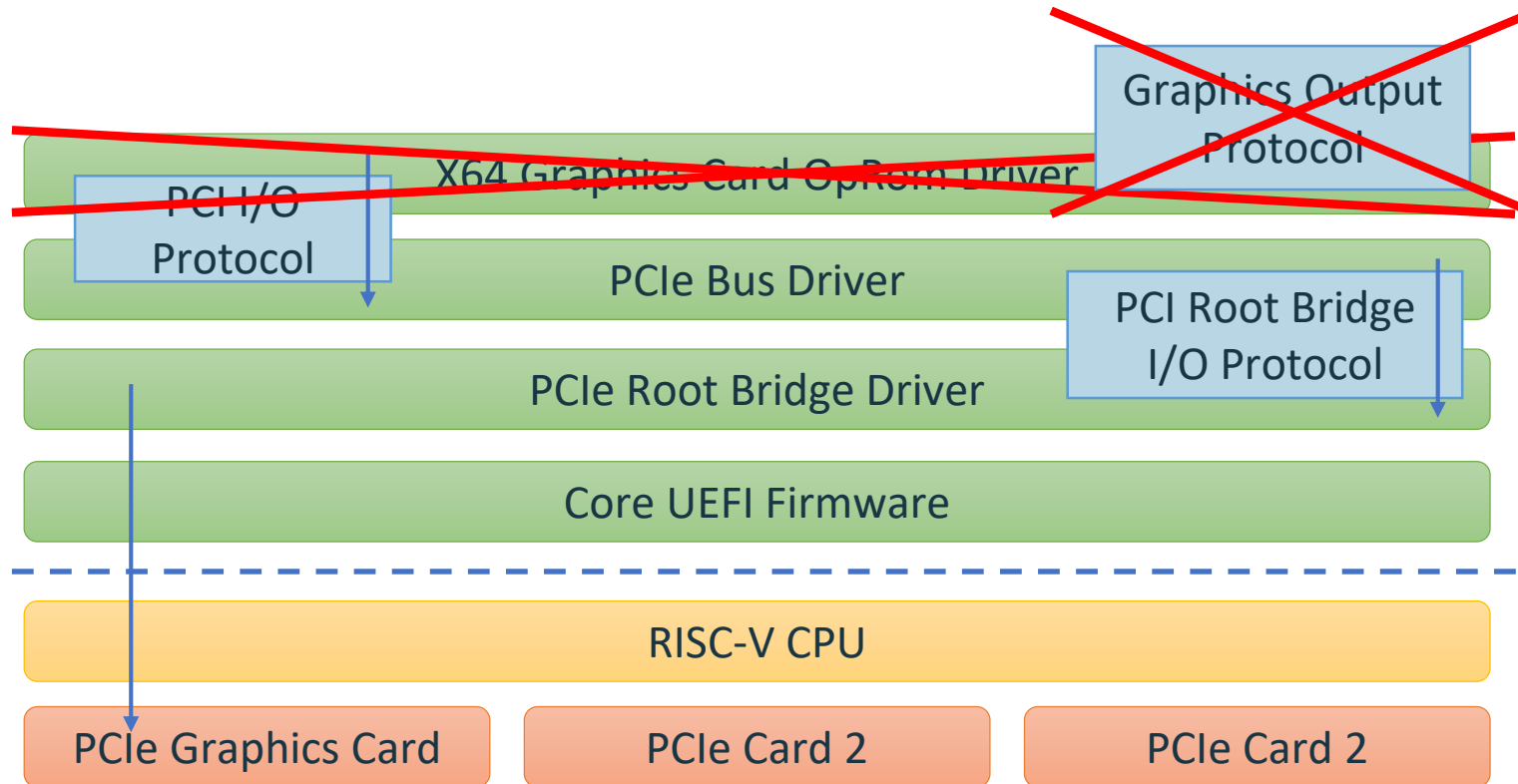Legacy PC-AT BIOS ROM

X64 UEFI Driver

Life of PCIe in UEFI

Life of PCIe in UEFI on RISC-V

Hasn't this been solved before?

EFI Byte Code

PCI Expansion ROM Image 0	Legacy PC-AT BIOS ROM
PCI Expansion ROM Image 1	X64 UEFI Driver
. . . .	
PCI Expansion ROM Image N	EBC UEFI Driver

❖ ✔ Specifically made for this scenario!
  ❖ Processor Independence
    ❖ sizeof(VOID*) is a runtime operation.
    ❖ VM takes care of 32 vs 64 vs 128-bit issues.
  ❖ TianoCore comes with an interpreter.
❖ ✘ Not used by the industry!
  ❖ No tooling – the only supported and proprietary C compiler has been retired.
    ❖ Some OSS now exists
      ❖ https://github.com/yabits/ebcvm / ELVM
      ❖ https://github.com/pbatard/fasmg-ebc
  ❖ Different performance profile - interpreted code.
  ❖ Didn't make a come-back when the Arm ecosystem explored this space

How did the Arm ecosystem solve this?

X86EmulatorPkg

* ✅ Supports x64 OpRoms and UEFI applications on AArch64 systems.
    * ❖ Open Source UEFI Boot Service Driver
    * ❖ Targets 64-bit AArch64 systems (servers, workstations)
    * ❖ Developed by Linaro engineers 6 years ago.
    * ❖ Uses Qemu Tiny Code Generator for efficient translation of x64 to AArch64 code.
    * ❖ https://github.com/ardbiesheuvel/X86EmulatorPkg
* ✖ Not trivially portable to RISC-V!
    * ❖ Old TCG code of unknown provenance.
    * ❖ Backporting RISC-V support sounds hard (and time consuming) unless you're a Qemu guru.

MultiArchUefiPkg

Rewrite of X86EmulatorPkg

❖ Portable: Supports AArch64 and 64-bit RISC-V UEFI hosts.

❖ 64-bit x64 and AArch64 UEFI Boot Service emulation.

❖ Clean: Abstracts Qemu/TCG with Unicorn Engine API.

❖ https://github.com/intel/MultiArchUefiPkg

❖ RISE Project in the Firmware WG

❖ Correctness, perf, size.


```
RISC-V EDK2 firmware version 2.7
Press ESCAPE within 10 seconds for boot options
```

TianoCore
Open Platform Firmware Development

Start boot option


```
7D  000000A  B  - -  1  6 Usb Bus Driver                              UsbBusDxe
7E  0000000A  D  - -  1  - Usb Keyboard Driver                         UsbKbDxe
7F  00000011  D  - -  1  - Usb Mass Storage Driver                     UsbMassStorageDxe
82  00014300  B  - -  1  1 AMD GOP X64 Release Driver Rev.1.67 Offset(0x10000,0x1E5FF)
Shell> _
```
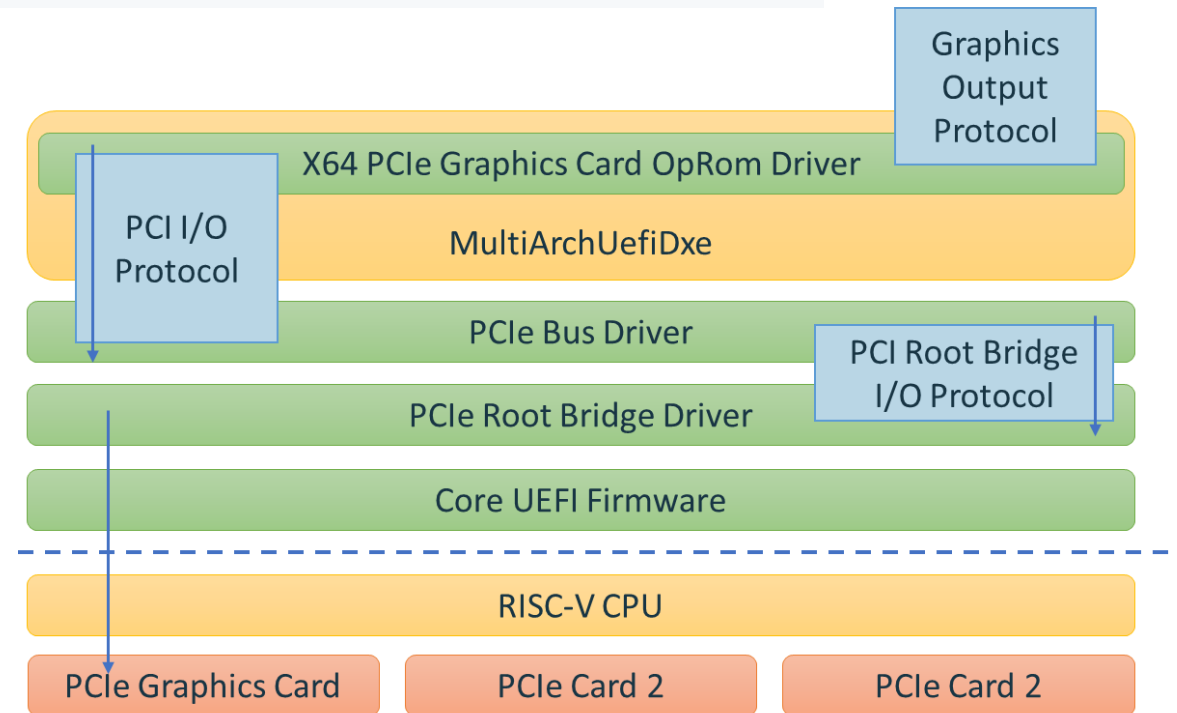
MultiArchUefiPkg
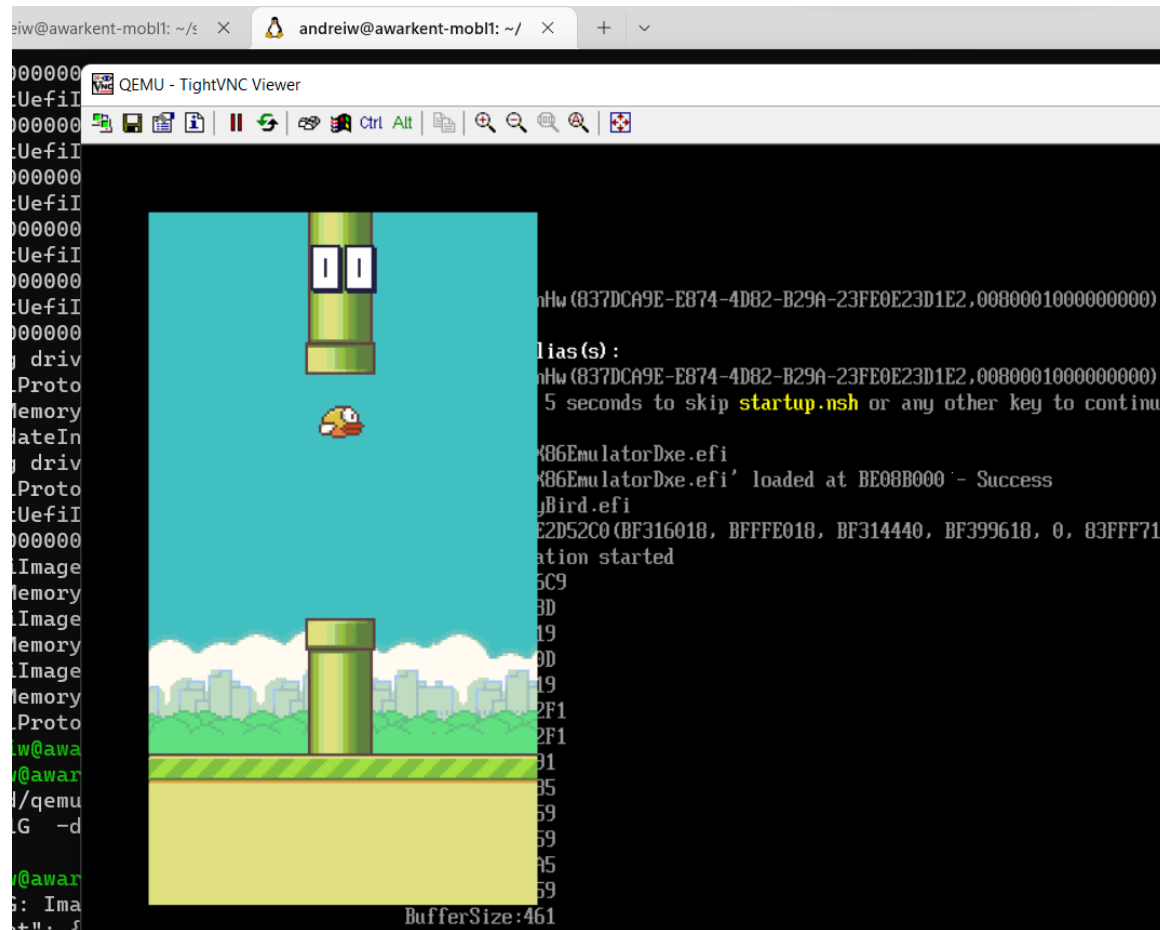
How it works

❖ Possible entirely due to narrowly-defined EFI ABI

❖ Models Boot Services environment, with certain services filtered or disabled.

❖ Tiano support for foreign binaries - EDKII_PECOFF_IMAGE_EMULATOR_PROTOCOL

❖ Emulation is only interesting if thunking goes both ways!

  ❖ RISC-V No-Execute handler traps for native → emulated.

  ❖ Unicorn No-Execute handler traps for emulated → native.

```
UINT64
EFIAPI
Fn(UINT64, UINT64, UINT64, UINT64,
    UINT64, UINT64, UINT64, UINT64,
    UINT64, UINT64, UINT64, UINT64,
    UINT64, UINT64, UINT64, UINT64);
```

Graphics Output Protocol

X64 PCIe Graphics Card OpRom Driver

PCI I/O Protocol

MultiArchUefiDxe

PCIe Bus Driver

PCI Root Bridge I/O Protocol

PCIe Root Bridge Driver

Core UEFI Firmware

RISC-V CPU

PCIe Graphics Card   PCIe Card 2   PCIe Card 2

Now what?

- ❖ Fine for short term.
  - ❖ Tons of "correctness" or validation issues https://github.com/intel/MultiArchUefiPkg/issues
  - ❖ More testing on real RISC-V hardware.
  - ❖ UEFI MMU patches from Ventana Micro in soon.
- ❖ What about long term?
  - ❖ Existing ISAs are a moving target.
  - ❖ OpRom environment unconstrained.
- ❖ Multiple ISA support in COTS hardware unlikely.
  - ❖ How many adapters ship with x64 + AArch64 today?
  - ❖ How many will additionally bundle RISCV64, RISCV128 and LOONGARCH support?

> !!! NEED YOUR FEEDBACK !!!

- ❖ Embrace emulation, but how?
  - ❖ Resurrect EBC.
    - ❖ Never been more relevant with 4 64-bit ISAs and 128-bit ISAs following.
    - ❖ Tooling
    - ❖ Performance
  - ❖ Constrain x64 OpRom environment.
    - ❖ Meet IHVs half-way.
    - ❖ Pick a subset of x64– ring3, basic SSE, etc.
    - ❖ Generate code that will guarantee to run via MultiArchUefiPkg.
  - ❖ WASM for EFI.
    - ❖ Great tooling.
    - ❖ Sandboxing?
    - ❖ 128-bit support?