vfio-platform: Live and let die ?

Eric Auger
KVM Forum 2024

Context

- ~10y anniversary!
- Significant initial upstream effort (2014-2015)

and then?

- Last integrated device in 2017
- Last functional kernel contribution in 2018
- Known to be used but without upstream contributions
- Maintenance issue due to e-waste

redhat.

Goal

- What is it used for?
- How to support a new device?
- Challenges & right Candidates
- Open discussion on its future

redhat.

What is it used for?

- Give user-space control over a physical platform device
  - VFIO → IOMMU (DMA capable device)
  - direct access to registers
  - interrupt handling
  - DMA buffer mapping to user VA
- Use cases
  - userspace drivers
  - assignment to a guest
- HW platforms
  - ARM embedded, automotive, edge

redhat.

How to support new devices?

- 3 devices are officially supported at kernel & QEMU level:
  - calxeda,hb-xgmac (2015)
  - amd,xgbe-seattle-v1a (2015)
  - brcm,iproc-flexrm-mbox (2017)

- Integrating a new device requires adaptation
  - kernel: to support device reset
  - vmm: to expose the device to the guest

redhat.

Device Reset

- Need
  - stop DMA transfers
  - disable interrupts
  - [get/put dynamic resources if any]

- Problem
  - Platform device reset does not follow any specification
  - vfio-platform driver is device agnostic

- Implementation
  - ACPI boot: _RST method (FW)
  - DT boot: custom reset module (kernel)

redhat.

Kernel Reset Module - Basics

- Module Infrastructure already in place
  - Automatically loaded on device attach using the device compatibility string
  - Reset method called on device open/close and VFIO_DEVICE_RESET ioctl

- Examples in drivers/vfio/platform/reset/*

- Challenges
  - Requires driver and/or device knowledge
  - Dynamic resources (resets, clocks)

redhat.

Kernel Reset Module - Tegra234 mgbe

- A more complex example
  - [RFC PATCH v2 0/6] vfio: platform: reset: Introduce tegra234 mgbe reset module

- Dependency on dynamic resources
  - clocks
  - Resets

- Requested to enhance the too simplistic single reset function
  - init
  - release

redhat.

Expose the device to a guest

redhat.

QEMU Magic

- “-device vfio-platform,host=6810000.ethernet” `/sys/bus/platform/devices/6810000.ethernet`
- Dynamic sysbus Instantiation
  - create a platform bus
  - create & connect the vfio-platform device (mmio, interrupts)
- dt node generation in hw/core/sysbus-fdt.c

```
static const BindingEntry bindings[] = {
#ifdef CONFIG_LINUX
    TYPE_BINDING(TYPE_VFIO_CALXEDA_XGMAC, add_calxeda_midway_xgmac_fdt_node),
    TYPE_BINDING(TYPE_VFIO_AMD_XGBE, add_amd_xgbe_fdt_node),
    VFIO_PLATFORM_BINDING("amd,xgbe-seattle-v1a", add_amd_xgbe_fdt_node),
    VFIO_PLATFORM_BINDING("nvidia,tegra234-mgbe", add_nvidia_tegra234_mgbe_fdt_node),
#endif
#ifdef CONFIG_TPM
    TYPE_BINDING(TYPE_TPM_TIS_SYSBUS, add_tpm_tis_fdt_node),
#endif
../..
};
```


redhat.

calxeda,hb-xgmac example 😄

```
 ethernet@fff50000 {
    dma-coherent;
    reg = <0xfff50000 0x1000>;
    interrupts = <0 77 4>, <0 78 4>, <0 79 4>;
    compatible = "calxeda,hb-xgmac";
};
```

platform_bus_get_mmio_addr()

memory_region_size(vdev->regions[i]->mem)

platform_bus_get_irqn()

redhat.

amd, xgbe-seattle-v1a 😑

```
xgbe@e0700000 {
   reg = <0 0xe0700000 0 0x80000>,
        <0 0xe0780000 0 0x80000>,
        <0 0xe1240800 0 0x00400>,
        <0 0xe1250000 0 0x00060>,
        <0 0xe1250080 0 0x00004>;
   interrupt-parent = <&gic>;
   interrupts = <0 325 4>, <0 326 1>, <0 327 1>,
                <0 328 1>, <0 329 1>, <0 323 4>;
   compatible = "amd,xgbe-seattle-v1a";
   amd,per-channel-interrupt;
   phy-mode = "xgmii";
   mac-address = [ 02 a1 a2 a3 a4 a5 ];
   amd,speed-set = <0>;
   amd,serdes-blwc = <1>, <1>, <0>;
   amd,serdes-cdr-rate = <2>, <2>, <7>;
   amd,serdes-pq-skew = <10>, <10>, <30>;
   amd,serdes-tx-amp = <15>, <15>, <10>;
   amd,serdes-dfe-tap-config = <3>, <3>, <1>;
   amd,serdes-dfe-tap-enable = <0>, <0>, <127>;
   clock-names = "dma_clk", "ptp_clk";
   clocks = <&xgbe_dma_clk>, <&xgbe_ptp_clk>;
};
```

copy_properties_from_host()
amd_xgbe_copied_properties[]

```
xgbe_dma_clk {
        #clock-cells = <0x00>;
        clock-frequency = <0x78>;
        compatible = "fixed-clock";
        phandle = <0x800e>;
};
```

fdt_build_clock_node() builds dummy clk nodes

redhat.

nvidia,tegra234-mgbe

```
ethernet@680000 {
        ../..
        resets = <&bpmp TEGRA234_RESET_MGBE0_MAC>, <&bpmp TEGRA234_RESET_MGBE0_PCS>;
        reset-names = "mac", "pcs";
        interconnects = <&mc TEGRA234_MEMORY_CLIENT_MGBEARD &emc>, <&mc TEGRA234_MEMORY_CLIENT_MGBEAWR &emc>;
        interconnect-names = "dma-mem", "write";
        power-domains = <&bpmp TEGRA234_POWER_DOMAIN_MGBEA>;
        phy-handle = <&mgbe0_phy>;
        mdio {
            #address-cells = <1>;
            #size-cells = <0>;
            mgbe0_phy: phy@0 {
                compatible = "ethernet-phy-ieee802.3-c45";
                reg = <0x0>;
                #phy-cells = <0>;
            };
        };
}
```

- Need to generate a reset controller dt node
  - nvidia,tegra186-bpmp is not an option
    - depends on nvidia,tegra186-hsp mailbox that requires emulation
  - hisilicon,hi3660-reset works (with a syscon dependency)

redhat.

Guest Tegra234 view

```
810000.ethernet@0 {
        nvidia,rx_frames = <0x40>;
        nvidia,max-platform-mtu = <0x3fff>;
        phy-mode = "10gbase-r";
        dma-coherent;
        nvidia,tx_frames = <0x10>;
        nvidia,rx_riwt = <0x200>;
        nvidia,num-dma-chans = <0x0a>;
        clock-names = "rx-input-m\0rx-pcs-m\0rx-pcs-input\0rx-pcs\0tx\0tx-pcs\0mac-divider\0mac\0
                       eee-pcs\0mgbe\0ptp-ref\0mgbe_macsec\0rx-input";
        reg-names = "mac\0xpcs\0macsec-base\0hypervisor";
        nvidia,promisc_mode = <0x01>;
        nvidia,mtl-queues = <0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09>;
        nvidia,dcs-enable = <0x01>;
        nvidia,dma_tx_ring_sz = <0x1000>;
        nvidia,num-mtl-queues = <0x0a>;
        nvidia,tx-queue-prio = <0x00 0x01 0x02 0x03 0x04 0x05 0xx06 0x07 0x00 0x00>;
        resets = <0x800c 0x00 0x00 0x800c 0x08 0x00>;
        nvidia,dma-chans = <0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09>;
        interrupts = <0x00 0x70 0x04 0x00 0x71 0x04 0x00 0x72 0x04 0x00 0x73 0x04 0x00
                      0x74 0x04 0x00 0x75 0x04 0x00 0x76 0x04 0x00 0x77 0x04>;
        clocks = <0x800e 0x800f 0x8010 0x8011 0x8012 0x8013 0x8014 0x8015 0x8016
                  0x8017 0x8018 0x8019 0x801a>;
        nvidia,rx-queue-prio = <0x01 0x02 0x04 0x08 0x10 0x20 0x40 0x80 0x00 0x00>;
        mac-address = [48 b0 2d a4 84 0a];
        compatible = "nvidia,tegra234-mgbe";
        nvidia,ptp-rx-queue = <0x03>;
        nvidia,tc-mapping = <0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x00 0x01>;
        nvidia,ptp_ref_clock_speed = <0x12a05f20>;
        interrupt-names = "common\0vm0\0vm1\0vm2\0vm3\0vm4\0macsec-ns-irq\0macsec-s-irq";
        reg = <0x00 0x10000 0x10000 0x10000 0x20000 0x10000 0x30000 0x10000>;
        reset-names = "mac\0pcs\0macsec_ns_rst";
};
```

```
hisi-rst-syscon {
        compatible = "syscon";
        reg = <0x00 0x00 0x00 0x1000>;
        phandle = <0x800d>;
};

hisi-rst {
        #reset-cells = <0x02>;
        hisilicon,rst-syscon = <0x800d>;
        compatible = "hisilicon,hi3660-reset";
        phandle = <0x800c>;
};

mac {
        #clock-cells = <0x00>;
        clock-frequency = <0x78>;
        compatible = "fixed-clock";
        phandle = <0x800e>;
};
```


redhat.

Conclusion

- Current Showstoppers
  - no iommu or usage of DMA controllers
  - if guest dt node stubs are not sufficient
  - shared resources
- Troublesome
  - Dynamic resources (resets, clocks, power domain, ...)
  - dt node stubs generated on guest side
- Device architects & native driver maintainers should have the use case in mind
  - reset and dt generation requires knowledge of the device
  - “virtual functions” would simplify things a lot

redhat.

Discussion

- Is there a market segment for vfio platform?
- Please be transparent on your usage and at least contribute reset drivers
- High risk of driver deprecation if we don't have new integrated devices
- Willing to help for new device integration

redhat.

References

- Platform Device VFIO - Alexander Graf, LPC 2013 (https://www.youtube.com/watch?v=qRITZ8OPF98)
- KVM Platform Device Passthrough - Eric Auger, KVM forum 2014 (https://www.linux-kvm.org/images/a/a8/01x04-ARMdevice.pdf)
- An Introduction to PCI Device Assignment with VFIO by Alex Williamson, KVM forum 2016 (https://www.youtube.com/watch?v=WFkdTFTOTpA&t=28s)
- [PATCH v5] vfio: platform: Add generic reset controller support
- Getting To Blinky: Virt Edition, Making device pass-through work on embedded ARM by Geert Uytterhoeven, FOSDEM 2019
- Runtime ownership transfer of platform devices, by QC, Sept 18 2024, LPC 2024

redhat.

THANK YOU


plus.google.com/+RedHat


facebook.com/redhatinc


linkedin.com/company/red-hat


twitter.com/RedHat


youtube.com/user/RedHatVideos