1. http://stackoverflow.com/questions/29461518/interrupt-handling-for-assigned-device-through-vfio

An interrupt from the device is received by the host kernel and routed to an interrupt handler registered by the vfio bus driver, vfio-pci or vfio-platform. That interrupt handler simply relays the interrupt to an eventfd that the user (QEMU) has configured via ioctl. When KVM is used, the user is able to connect the interrupt signalling eventfd from vfio directly to an interrupt injecting irqfd in KVM. This avoids bouncing the interrupt out to QEMU userspace for injection into the guest, though that path is an option if KVM irqfd support is not available.

For a level-triggered interrupt, we must also mask the interrupt interrupt in the host to prevent the device from continuing to interrupt the host while the interrupt is serviced by the guest. We therefore mask the interrupt prior to signaling the eventfd and use a slightly different KVM irqfd called a resampling irqfd, that registers a second eventfd-irqfd pair for signaling the unmask from KVM to vfio.

Various hardware technologies augment this for better efficiency. Intel APICv allows interrupts to be injected directly into the guest without a vmexit in some circumstances. This is handled entirely within KVM. Intel Posted Interrupts will allow interrupts to bypass the host completely when the correct vCPU is running on the processor receiving the hardware interrupt. ARM IRQ Forwarding allows the guest to manage the unmasking of interrupts avoiding the resampling irqfd overhead.

> Just follow up questions from Source point of view, I can see that for vfio-platform "eventfd_signal(irq_ctx->trigger, 1); " would signal the interrupt to KVM guest but I could not find out where in QEMU source eventfd call is made and configured via IOCTL's ? Also I didn't get that point when KVM is involved QEMU is able to connect the Interrupt signaling eventfd from VFIO directly to an interrupt injecting irqfd in KVM. – Amit Singh Tomar Apr 20 '15 at 12:27
>
> In Eric's v12 branch, the eventfd connection is made in platform.c:vfio_set_trigger_irqfd(). This enable the kernel module to notify the eventfd setup in the intp->interrupt event notifier. The irqfd is setup via vfio_start_irqfd_injection() in that same file. Eventfds are a mechanism for the kernel to send a signal to a file descriptor, generally for userspace signaling. IRQfds are a mechanism for userspace to signal the kernel via a file descriptor for the purpose of injecting an interrupt into a VM. If we use the same file descriptor for both, we can transparently connect vfio to kvm. – Alex W

2.http://www.linuxplumbersconf.org/2012/wp-content/uploads/2012/09/2012-lpc-virt-vfio-williamson.pdf

3. https://www.youtube.com/watch?v=XhN2HU_BL3w

4. http://nairobi-embedded.org/001_qemu_pci_device_essentials.html

5.https://www.youtube.com/watch?v=WFkdTFTOTpA

6.http://people.cs.nctu.edu.tw/~chenwj/slide/QEMU/qemu-part3.htm

7.https://www.youtube.com/watch?v=BGGbgufgP_Y
(**ARM® Interrupt Virtualization - The Linux Foundation)**

MSI Interrupt Setup by Guest Driver:

Vfio_pci_write_config
|

-->vfio_msi_enable-->qemu_set_fd_handler(vfio_msi_interrupt)-->vfio_add_kvm_msi_virq-->kvm_irqchip_add_msi_route-->ioctl(KVM_SET_GSI_ROUTING)
|
kvm_irqchip_add_irqfd_notifier_gsi-->kvm_irqchip_assign_irqfd()-->ioctl(KVM_IRQFD)
|
vfio_enable_vectors-->ioctl(VFIO_DEVICE_SET_IRQS)

In Kernel:
ioctl(KVM_SET_GSI_ROUTING)-->kvm_set_irq_routing-->setup_routing_entry-->kvm_set_routing_entry

ioctl(VFIO_DEVICE_SET_IRQS)-->vfio_pci_set_irqs_ioctl-->vfio_pci_set_msi_trigger-->

ioctl(KVM_IRQFD)-->kvm_irqfd--> kvm_irqfd_assign
|
INIT_WORK(&irqfd->inject, irqfd_inject);
INIT_WORK(&irqfd->shutdown, irqfd_shutdown);

MSI IOMMU
---

http://vfio.blogspot.co.uk/2014/08/iommu-groups-inside-and-out.html

Init MSI memory
---
create_pci()-->memory_region_init_reserved_iova()
vfio_listener_region_add()-->vfio_register_msi_iova()-->ioctl(VFIO_IOMMU_MAP_DMA)