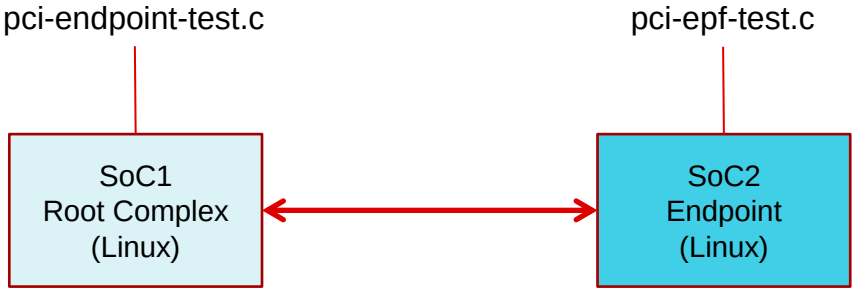
Virtio based communication between RC<->EP and between HOSTS connected to NTB

Kishon Vijay Abraham I

Texas Instruments

System 1: Root Complex <-> Endpoint

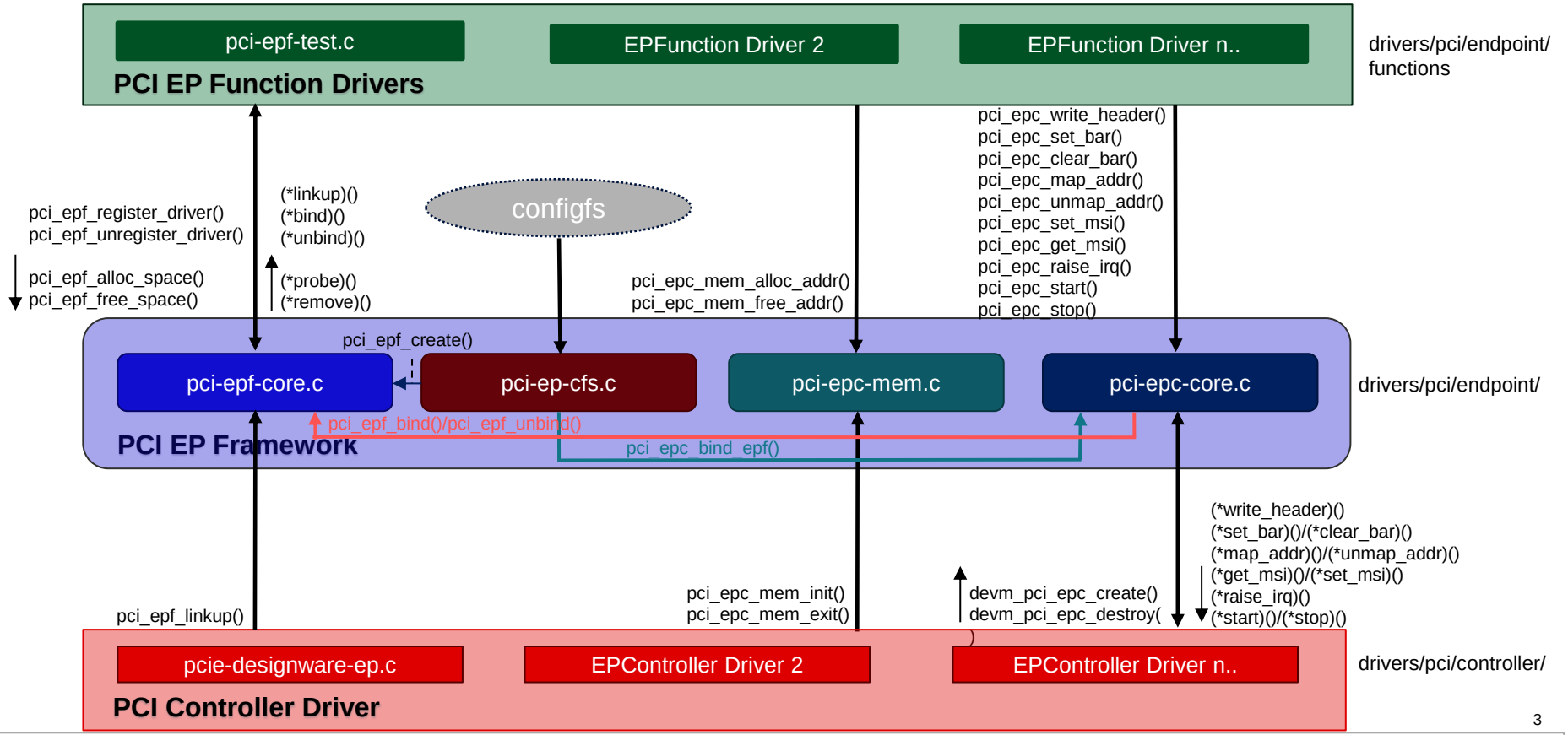
```
root@j7200-evm:~# pcitest -h
usage: pcitest [options]
Options:
        -D <dev>                PCI endpoint test device {default:
/dev/pci-endpoint-test.0}
        -b <bar num>            BAR test (bar number between 0..5)
        -m <msi num>            MSI test (msi number between 1..32)
        -x <msix num>           MSI-X test (msix number between 1..2048)
        -i <irq type>           Set IRQ type (0 - Legacy, 1 - MSI, 2 - MSI-
X)
        -e                      Clear IRQ
        -I                      Get current IRQ type configured
        -d                      Use DMA
        -l                      Legacy IRQ test
        -r                      Read buffer test
        -w                      Write buffer test
        -c                      Copy buffer test
        -s <size>               Size of buffer {default: 100KB}
        -h                      Print this help message
```



- Testing RC to EP communication
  - BAR Tests
  - Read/Write/Copy Tests (Single Buffer)
  - Use DMA
  - Interrupt Tests (Legacy/MSI/MSI-X)
- Not useful for actual use cases
  - Does not support multiple-buffers
  - No standard protocols (additional application development effort)

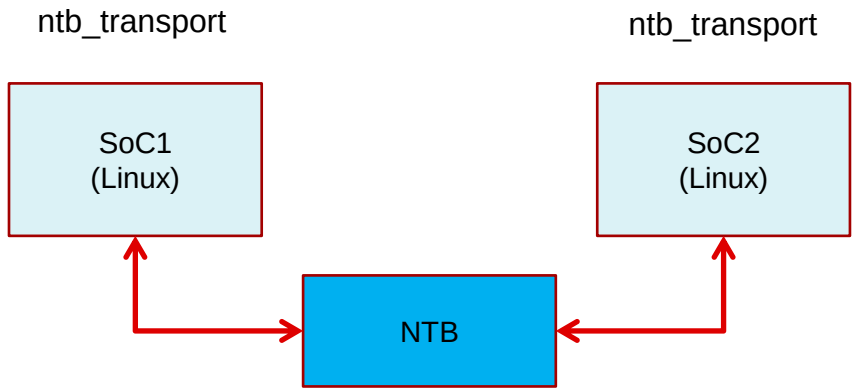
Texas Instruments

Recap: EP Framework



Texas Instruments

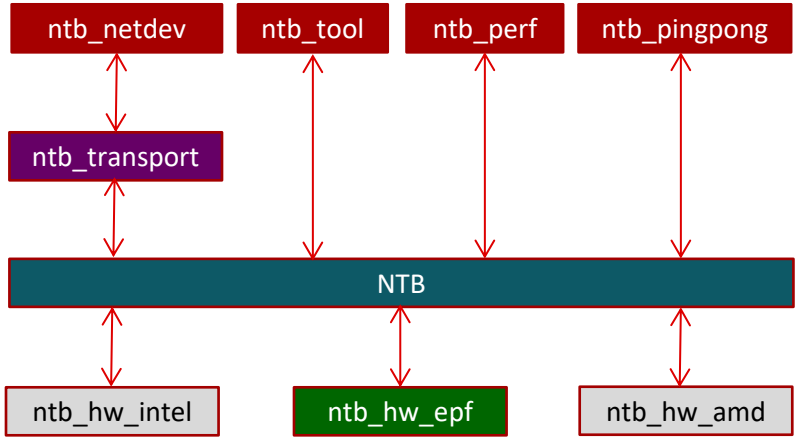
System 2: HOST <-> NTB <->HOST

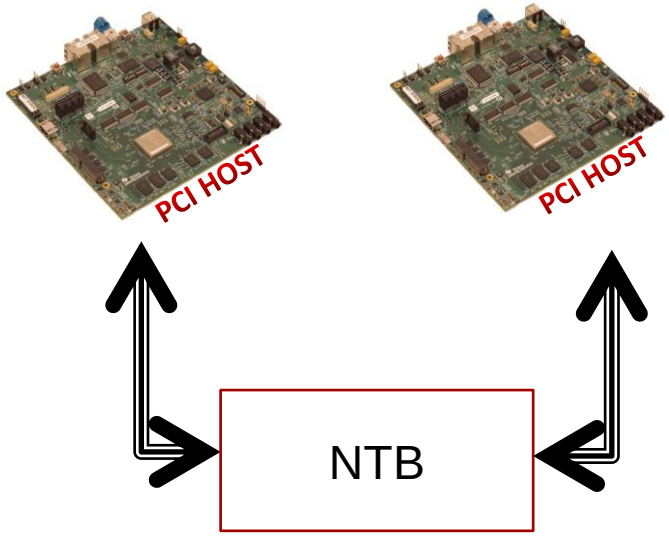


- Single NTB application using NTB transport client (netdev)
- Network based applications can be used
- Cannot leverage development in other places of kernel (read virtio)

Texas Instruments

Recap: NTB SW Architecture

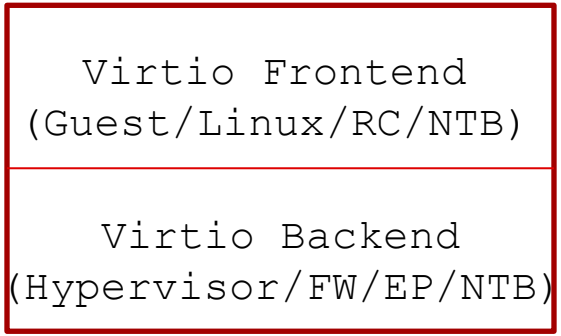




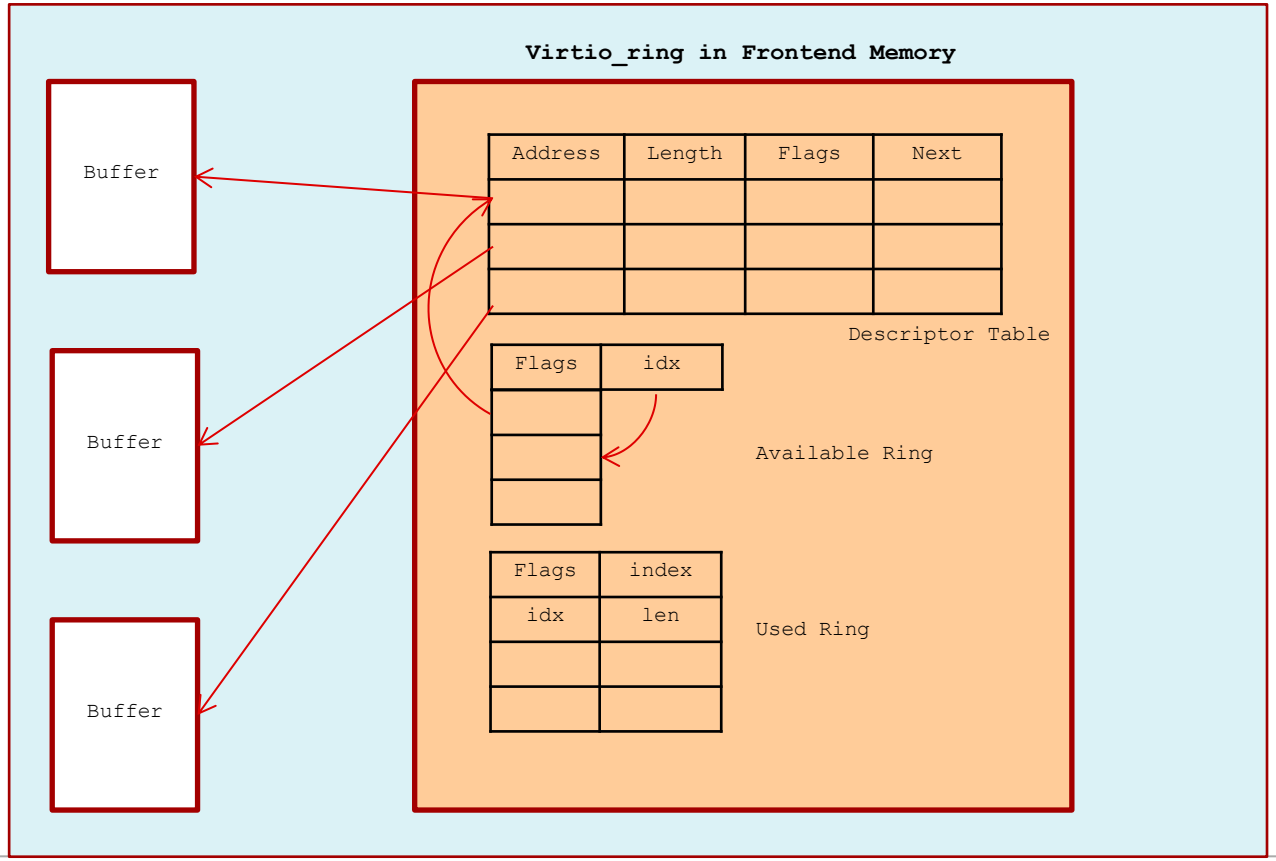
Texas Instruments

Virtio/Vhost Introduction

- Used in 2 places in kernel
  - Communication between HOST (Hypervisor) and GUEST systems in Virtualization context
  - Communication between different cores in an SoC
- Virtio-> Frontend (Guest or alternate cores); Vhost-> Backend (Hypervisor or Linux core)
- Provides standard producer-consumer implementation
- Supports multiple buffers
- Provides notification mechanism
- Multiple applications already exist (net, rpmsg, scsi..)



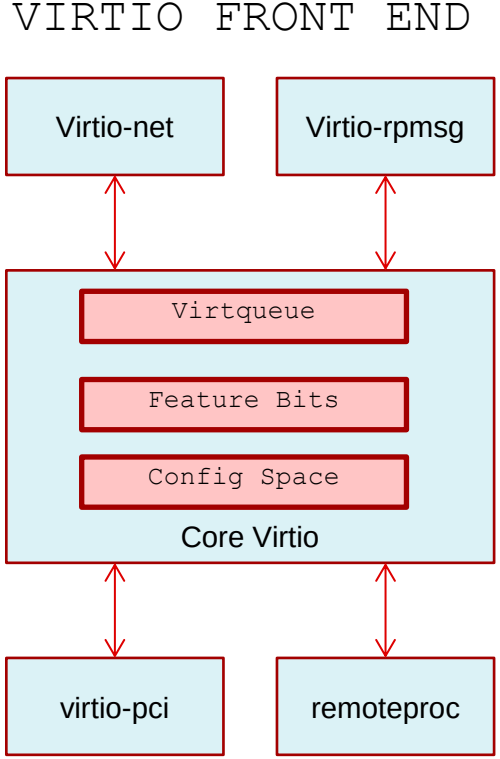
Virtio_ring



VHOST

- Access Virtio Memory
  - Access virtio ring
  - Access Buffers
- Userspace Access
- Kernel Space Access
- MMIO Access

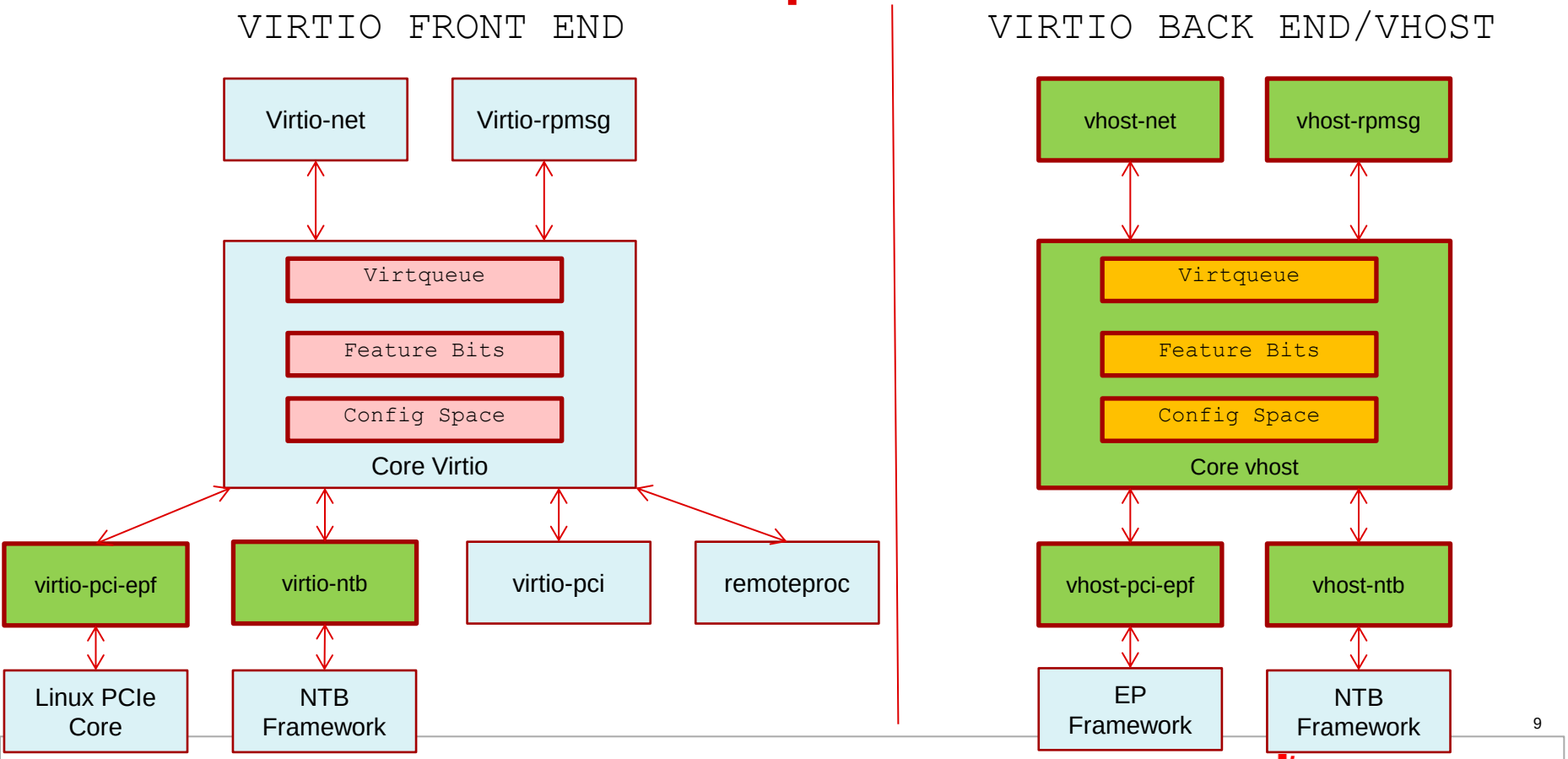
Virtio Front End SW Layering




```
struct virtio_config_ops {
        u8 (*get_status)(struct virtio_device *vdev);
        void (*set_status)(struct virtio_device *vdev, u8 status);
        void (*reset)(struct virtio_device *vdev);
        int (*find_vqs)(struct virtio_device *, unsigned nvqs,
                        struct virtqueue *vqs[], vq_callback_t *callbacks[],
                        const char * const names[], const bool *ctx,
                        struct irq_affinity *desc);
        void (*del_vqs)(struct virtio_device *);
        u64 (*get_features)(struct virtio_device *vdev);
        int (*finalize_features)(struct virtio_device *vdev);
};

int virtqueue_add_outbuf(struct virtqueue *vq,
                         struct scatterlist sg[], unsigned int num,
                         void *data,
                         gfp_t gfp);

int virtqueue_add_inbuf(struct virtqueue *vq,
                        struct scatterlist sg[], unsigned int num,
                        void *data,
                        gfp_t gfp);
bool virtqueue_kick(struct virtqueue *vq);
void *virtqueue_get_buf(struct virtqueue *vq, unsigned int *len);
```


Texas Instruments

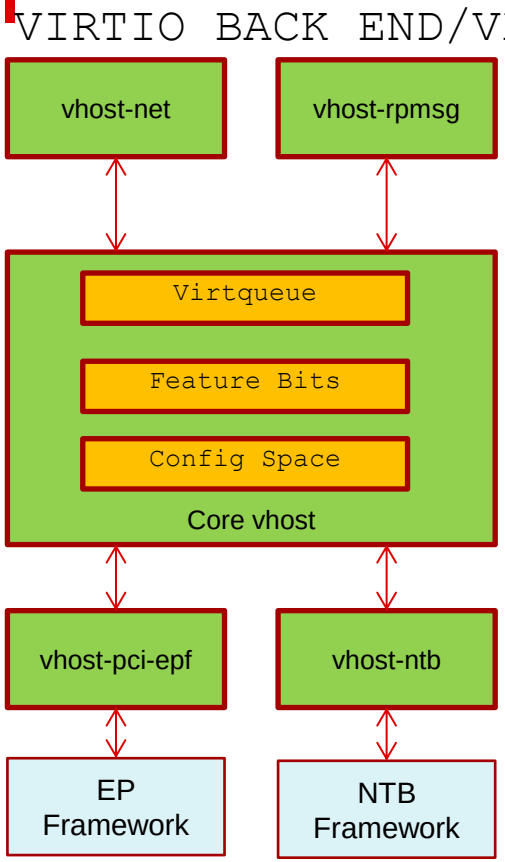
Virtio Backend/Vhost Proposal



Texas Instruments

Vhost Standard Linux Driver Model

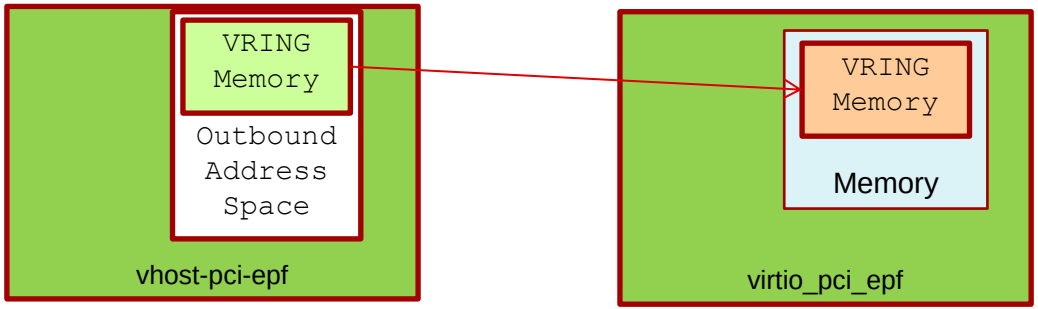
```
struct vhost_driver {
        struct device_driver driver;
        struct vhost_device_id *id_table;
        struct config_group *group;
        int (*probe)(struct vhost_dev *dev);
        int (*remove)(struct vhost_dev *dev);
};

int vhost_register_driver(struct vhost_driver *driver);
void vhost_unregister_driver(struct vhost_driver *driver);
int vhost_register_device(struct vhost_dev *vdev);
void vhost_unregister_device(struct vhost_dev *vdev);
```

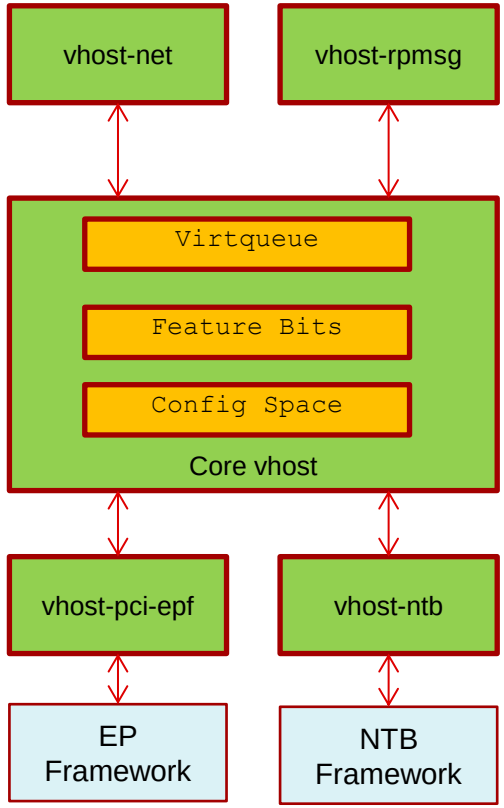


Texas Instruments

Configure vhost device

```
int vhost_create_vqs(struct vhost_dev *vdev, unsigned int nvqs,
                     unsigned int num_bufs, struct
vhost_virtqueue *vqs[],
                     vhost_vq_callback_t *callbacks[],
                     const char * const names[]);
void vhost_del_vqs(struct vhost_dev *vdev);
int vhost_write(struct vhost_dev *vdev, u64 vhost_dst, void
*src, int len);
int vhost_read(struct vhost_dev *vdev, void *dst, u64 vhost_src,
int len);
int vhost_set_features(struct vhost_dev *vdev, u64
device_features);
u64 vhost_get_features(struct vhost_dev *vdev);
int vhost_set_status(struct vhost_dev *vdev, u8 status);
u8 vhost_get_status(struct vhost_dev *vdev);
```

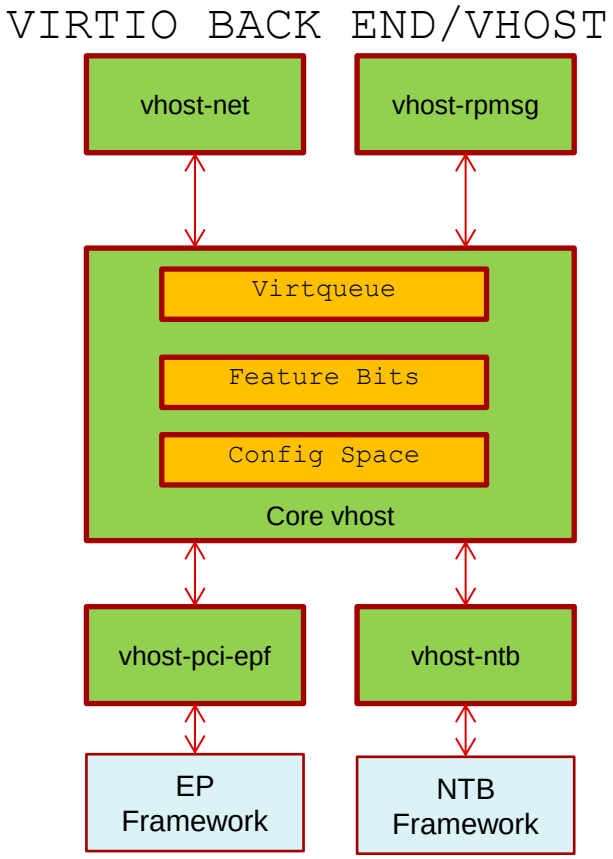




Texas Instruments

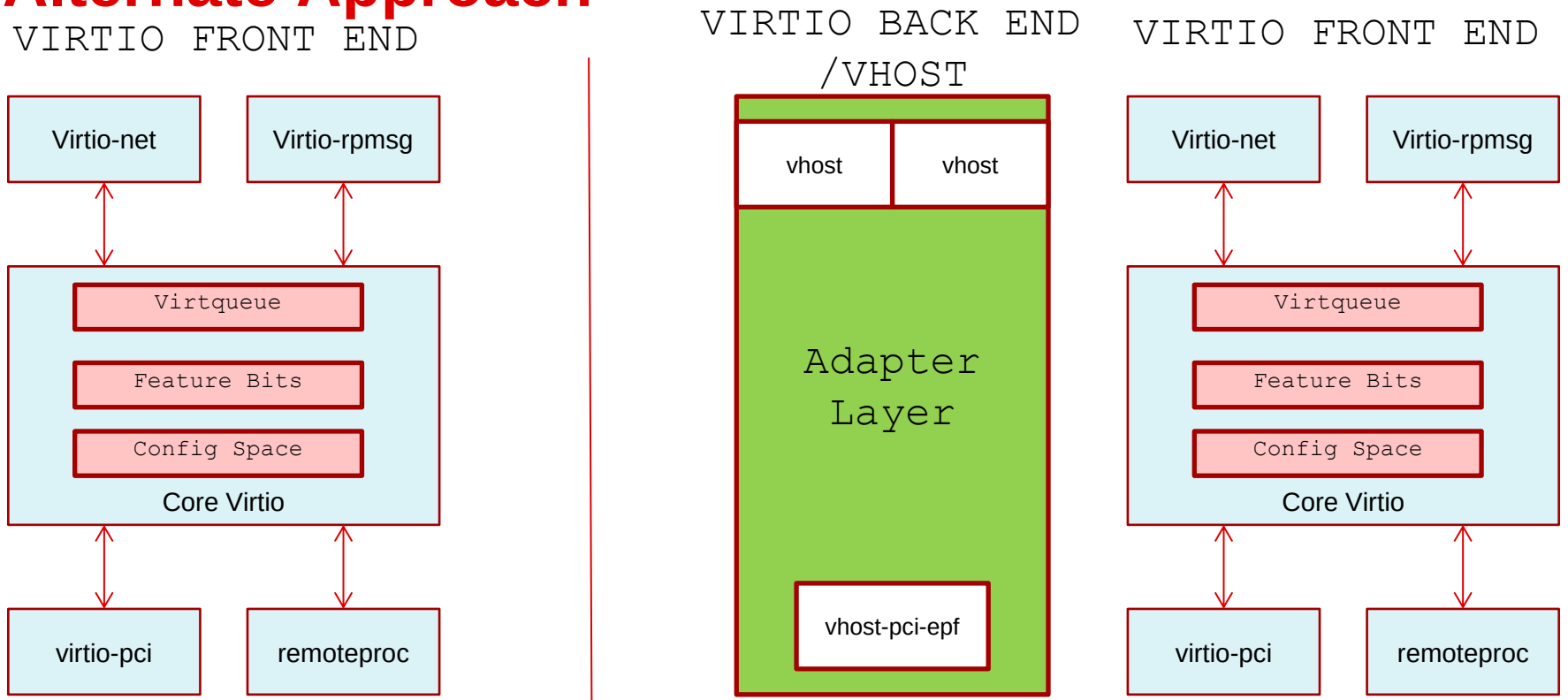
Additional Helpers for MMIO

```
int vringh_init_mmio(struct vringh *vrh, u64 features,
                     unsigned int num, bool weak_barriers,
                     struct vring_desc *desc,
                     struct vring_avail *avail,
                     struct vring_used *used);
int vringh_getdesc_mmio(struct vringh *vrh,
                        struct vringh_mmiov *riov,
                        struct vringh_mmiov *wiov,
                        u16 *head,
                        gfp_t gfp);

int vringh_complete_mmio(struct vringh *vrh, u16 head, u32 len);

bool vringh_notify_enable_mmio(struct vringh *vrh);
void vringh_notify_disable_mmio(struct vringh *vrh);
int vringh_need_notify_mmio(struct vringh *vrh);
```



Texas Instruments

Alternate Approach



Texas Instruments

Status

- RFC Posted
  - https://lore.kernel.org/kvm/20200702082143.25259-1-kishon@ti.com/

Texas Instruments

References

- virtio: Towards a De-Facto Standard For Virtual I/O Devices (http://www.cse.iitd.ernet.in/~sbansal/csl862-virt/readings/p95-russell.pdf)
- Virtual I/O Device (VIRTIO) Version 1.1 https://docs.oasis-open.org/virtio/virtio/v1.1/csprd01/virtio-v1.1-csprd01.html

Happy Hacking!

Texas Instruments