```
struct device {
    …
    const struct dma_map_ops *dma_ops;
    …
    struct iommu_group      *iommu_group;
    struct dev_iommu        *iommu;
    …
};
```

```
static const struct dma_map_ops iommu_dma_ops = {
        .alloc                  = iommu_dma_alloc,
        .free                   = iommu_dma_free,
        .alloc_pages            = dma_common_alloc_pages,
        .free_pages             = dma_common_free_pages,
        .alloc_noncoherent      = iommu_dma_alloc_noncoherent,
        .free_noncoherent       = iommu_dma_free_noncoherent,
        .mmap                   = iommu_dma_mmap,
        .get_sgtable            = iommu_dma_get_sgtable,
        .map_page               = iommu_dma_map_page,
        .unmap_page             = iommu_dma_unmap_page,
.map_sg                             = iommu_dma_map_sg,
        .unmap_sg               = iommu_dma_unmap_sg,
        .sync_single_for_cpu    = iommu_dma_sync_single_for_cpu,
        .sync_single_for_device = iommu_dma_sync_single_for_device,
        .sync_sg_for_cpu        = iommu_dma_sync_sg_for_cpu,
        .sync_sg_for_device     = iommu_dma_sync_sg_for_device,
        .map_resource           = iommu_dma_map_resource,
        .unmap_resource                 = iommu_dma_unmap_resource,
        .get_merge_boundary     = iommu_dma_get_merge_boundary,
};
```

由really_probe->platform_dma_configure->of_dma_configure->of_dma_configure_id->arch_setup_dma_ops->iommu_setup_dma_ops注册

```
/**
* struct dev_iommu - Collection of per-device IOMMU data
 *
* @fault_param: IOMMU detected device fault reporting data
* @fwspec:       IOMMU fwspec data
* @iommu_dev:            IOMMU device this device is linked to
* @priv:         IOMMU Driver private data
 *
* TODO: migrate other per device data pointers under
iommu_dev_data, e.g.
*         struct iommu_group      *iommu_group;
 */
struct dev_iommu {
        struct mutex lock;
        struct iommu_fault_param        *fault_param;
        struct iommu_fwspec             *fwspec;
        struct iommu_device             *iommu_dev;
        void                            *priv;
};
```

```
/**
* struct iommu_device - IOMMU core representation of one IOMMU hardware
*                         instance
* @list: Used by the iommu-core to keep a list of registered iommus
* @ops: iommu-ops for talking to this iommu
* @dev: struct device for sysfs handling
 */
struct iommu_device {
     struct list_head list;
     const struct iommu_ops *ops;
     struct fwnode_handle *fwnode;
     struct device *dev;
};
```

由platform_driver->rk_iommu_probe分配结构体，用iommu_device_register加入全局iommu_device_list链表

```
static const struct iommu_ops rk_iommu_ops = {
     .domain_alloc = rk_iommu_domain_alloc,
     .domain_free = rk_iommu_domain_free,
     .attach_dev = rk_iommu_attach_device,
     .detach_dev = rk_iommu_detach_device,
     .map = rk_iommu_map,
     .unmap = rk_iommu_unmap,
     .probe_device = rk_iommu_probe_device,
     .release_device = rk_iommu_release_device,
     .iova_to_phys = rk_iommu_iova_to_phys,
     .device_group = rk_iommu_device_group,
     .pgsize_bitmap = RK_IOMMU_PGSIZE_BITMAP,
     .of_xlate = rk_iommu_of_xlate,
};
```

```
struct iommu_group {
      struct kobject kobj;
      struct kobject *devices_kobj;
      struct list_head devices;
      struct mutex mutex;
      struct blocking_notifier_head
notifier;
      void *iommu_data;
      void (*iommu_data_release)
(void *iommu_data);
      char *name;
      int id;
      struct iommu_domain
*default_domain;
      struct iommu_domain *domain;
      struct list_head entry;
};
```

iommu_group代表iommu的最小粒度。由platform_driver->rk_iommu_probe->iommu_group_alloc分配

```
struct iommu_domain {
     unsigned type;
     const struct iommu_ops *ops;
     unsigned long pgsize_bitmap;    /* Bitmap of page sizes in use */
     iommu_fault_handler_t handler;
     void *handler_token;
     struct iommu_domain_geometry geometry;
     void *iova_cookie;
};
```

```
struct iommu_dma_cookie {
     enum iommu_dma_cookie_type type;
     union {
         /* Full allocator for IOMMU_DMA_IOVA_COOKIE */
          struct iova_domain       iovad;
         /* Trivial linear page allocator for IOMMU_DMA_MSI_COOKIE */
          dma_addr_t                 msi_iova;
     };
     struct list_head            msi_page_list;

    /* Domain for flush queue callback; NULL if flush queue not in use */
     struct iommu_domain                *fq_domain;
};
```

由really_probe->platform_dma_configure->of_dma_configure->of_dma_configure_id->of_iommu_configure->iommu_probe_device->iommu_alloc_default_domain->iommu_group_alloc_default_domain->rk_iommu_domain_alloc->iommu_get_dma_cookie->cookie_alloc分配iova_cookie，类型iommu_dma_cookie

```
/* holds all the iova translations for a domain */
struct iova_domain {
    spinlock_t        iova_rbtree_lock; /* Lock to protect update of rbtree */
     struct rb_root    rbroot;             /* iova domain rbtree root */
     struct rb_node   *cached_node;   /* Save last alloced node */
    struct rb_node   *cached32_node; /* Save last 32-bit alloced node */
     unsigned long   granule;/* pfn granularity for this domain */
     unsigned long   start_pfn;          /* Lower limit for this domain */
     unsigned long   dma_32bit_pfn;
    unsigned long   max32_alloc_size; /* Size of last failed allocation */
     struct iova_fq __percpu *fq;        /* Flush Queue */

     atomic64_t       fq_flush_start_cnt;         /* Number of TLB flushes that
                            have been started */

     atomic64_t       fq_flush_finish_cnt;        /* Number of TLB flushes that
                            have been finished */

     struct iova        anchor;             /* rbtree lookup anchor */
     struct iova_rcache rcaches[IOVA_RANGE_CACHE_MAX_SIZE];      /* IOVA range caches */

     iova_flush_cb     flush_cb;           /* Call-Back function to flush IOMMU
                         TLBs */

     iova_entry_dtor entry_dtor;        /* IOMMU driver specific destructor for
                         iova entry */

struct timer_list fq_timer;                        /* Timer to regularily empty the
                            flush-queues */
     atomic_t fq_timer_on;                          /* 1 when timer is active, 0
                            when not */
};
```

iova域，跟着device结构体走的，里面有rb_node来代表红黑树当前位置