```
        cdma {
                compatible = "mrvl,mv61_cdma";
                max_owned = <0x4>;
                max_shared = <0x3>;
                max_cyclic = <0x1>;
                max_memops = <0x0>;
                reg = <0x0 0xf9060000 0x0 0x9000>;
                interrupts = <0x0 0xbd 0x4>;
                clocks = <0x32>;
        };

        vdma-owned {
                compatible = "mrvl,mv61_vdma";
                id = <0x0>;
        };

        vdma-shared {
                compatible = "mrvl,mv61_vdma";
                id = <0x1>;
        };

        vdma-cyclic {
                compatible = "mrvl,mv61_vdma";
                id = <0x2>;
        };

        vdma-memops {
                compatible = "mrvl,mv61_vdma";
                id = <0x3>;
        };
        max_owned = <0x4>;

        max_shared = <0x3>;

        max_cyclic = <0x1>;

        max_memops = <0x0>;
```

这些参数用于指定对virtual channel的分配情况。

即MV61_VDMA_OWNED type virtual channel为4个

MV61_VDMA_SHARED type virtual channel为3个

MV61_VDMA_CYCLIC type virtual channel为1个

MV61_VDMA_MEMOPS type virtual channel为0个

有8个physical channel，但为什么virtual channel也是8个呢？

mv61_init_channels()记录了分配情况

```c
/**
 * mv61_init_channels - initialize physical channels and channel allocation
 * @mv61p: top physical dma control
 * @pdata: platform data of physical device
 *
 * Still single-threaded when this is called, but lock to be consistent.
 *
 * Physical channels are assigned in contiguous blocks.
 * All virtual channels that are not SHARED that are directly mapped to
 * corresponding physical channels.
 */
static int __init mv61_init_channels(struct mv61_dma *mv61p,
                        struct mv61_dma_platform_data *pdata)
{
        int             i;
        int             chans_avail = mv61p->pchannels;            ①
        int             ret = 0;
        unsigned long   biglockflags;

        if(chans_avail > MV61_DMA_MAX_NR_PCHANNELS) {
                ret = -EINVAL;
                goto done;
        }

        spin_lock_irqsave(&mv61p->biglock, biglockflags);

        for (i = 0; i < chans_avail; i++) {                    ②
                struct mv61_pdma_chan   *mv61pc = &mv61p->chan[i];
                mv61pc->index = i;
                mv61pc->mv61p = mv61p;

                mv61pc->ch_regs = mv61p->ch_regs[i];
                mv61pc->vtype = MV61_VDMA_UNASSIGNED;

                mv61_clear_pchannel(mv61pc);
        }

        pdata->nr_pool_chans[MV61_VDMA_OWNED] = (max_owned > chans_avail) ?
  ③
                                                        chans_avail :
                                                        max_owned;
        chans_avail -= pdata->nr_pool_chans[MV61_VDMA_OWNED];
                ④
        pdata->nr_virt_chans[MV61_VDMA_OWNED] = pdata->nr_pool_chans[MV61_VDMA_OW
NED];

        if(chans_avail) {                                  ⑤
                pdata->nr_pool_chans[MV61_VDMA_SHARED] = max_shared > chans_avail
 ?
                                                                chans_avail :
                                                                max_shared;
                chans_avail -= pdata->nr_pool_chans[MV61_VDMA_SHARED];
                pdata->nr_virt_chans[MV61_VDMA_SHARED] = max_vshared;
        } else {
                pdata->nr_pool_chans[MV61_VDMA_SHARED] = 0;
                pdata->nr_virt_chans[MV61_VDMA_SHARED] = 0;
        }

        if(chans_avail) {                                   ⑥
                pdata->nr_pool_chans[MV61_VDMA_CYCLIC] = max_cyclic > chans_avail
 ?
                                                                chans_avail :
                                                                max_cyclic;
                chans_avail -= pdata->nr_pool_chans[MV61_VDMA_CYCLIC];
                pdata->nr_virt_chans[MV61_VDMA_CYCLIC] =
                                        pdata->nr_pool_chans[MV61_VDMA_CYCLIC];
        } else {
                pdata->nr_pool_chans[MV61_VDMA_CYCLIC] = 0;
                pdata->nr_virt_chans[MV61_VDMA_CYCLIC] = 0;
        }

        if(chans_avail) {                                   ⑦
                pdata->nr_pool_chans[MV61_VDMA_MEMOPS] = max_memops > chans_avail
 ?
                                                                chans_avail :
                                                                max_memops;
                chans_avail -= pdata->nr_pool_chans[MV61_VDMA_MEMOPS];
                pdata->nr_virt_chans[MV61_VDMA_MEMOPS] =
                                        pdata->nr_pool_chans[MV61_VDMA_MEMOPS];
        } else {
                pdata->nr_pool_chans[MV61_VDMA_MEMOPS] = 0;
                pdata->nr_virt_chans[MV61_VDMA_MEMOPS] = 0;
        }

        if(pdata->nr_virt_chans[MV61_VDMA_SHARED] > MV61_DMA_MAX_NR_VCHANNELS)
                ret = -EINVAL;
        else
                ret = 0;

        spin_unlock_irqrestore(&mv61p->biglock, biglockflags);

done:
        return ret;
}
```

①

chans_avail是physical channel number，在88PA6270上是8

这里显然认为virtual channel == physical channel

②

这是对cdma device的所有physical channel进行初始化。

③④⑤⑥⑦

对struct mv61_dma_platform_data pdata初始化

```
/**
 * struct mv61_dma_platform_data - Controller configuration parameters
 * @cdma_type: CDMA_MV61X0 = 1, CDMA_PEGMATITE = 2
 * @nr_channels: Number of channels supported by hardware (max 12)
 * @nr_pool_chans: Number of hardware channels per virtual controller
 * @nr_virt_chans: Number of virtual channels per virtual controller
 * @__mv61_dma: address of struct mv61_dma (not always in scope here)
 */
struct mv61_dma_platform_data {
        unsigned int    cdma_type;
        unsigned int    nr_channels;
unsigned int    nr_pool_chans[MV61_NR_VDMA_CONTROLLERS];
        unsigned int    nr_virt_chans[MV61_NR_VDMA_CONTROLLERS];
};
```

```
pdata->nr_channels = readl(base + CDMAPR_OFFSET);
```

nr_channels是cdma controller的physical channel number

unsigned int   nr_pool_chans[MV61_NR_VDMA_CONTROLLERS];

unsigned int   nr_pool_chans[4];

{

  nr_pool_chans[MV61_VDMA_OWNED],

  nr_pool_chans[MV61_VDMA_SHARED],

  nr_pool_chans[MV61_VDMA_CYCLIC],

  nr_pool_chans[MV61_VDMA_MEMOPS],

}

nr_pool_chans[4]表示每种类型的physical channel被分配了所少个。

比如按dts中设置

nr_pool_chans[MV61_VDMA_OWNED] = 4个

nr_pool_chans[MV61_VDMA_SHARED] = 3个

nr_pool_chans[MV61_VDMA_CYCLIC] = 1个

nr_pool_chans[MV61_VDMA_MEMOPS] = 0个

由于总的physical channel为8，所以总和自然不能超过8个。

目前各种类型的virtual channel被分配的个数与physical channel是一样的。

struct mv61_dma_platform_data记录了dts中指定的对physical / virtual channel的分配状况。

```
          cdma {
                  compatible = "mrvl,mv61_cdma";
                  max_owned = <4>;
                  max_shared = <3>;
                  max_cyclic = <1>;
                  max_memops = <0>;
                  reg = <0 0xf9060000 0 0x9000>;
                  interrupts = <0 189 4>;
                  clocks = <&ipsbus_cdma_clkgate>;
          };
```

==>

struct mv61_dma_platform_data

```
{
   nr_channels = 8,


   nr_pool_chans[MV61_VDMA_OWNED] = nr_pool_chans[0] = 4

   nr_pool_chans[MV61_VDMA_SHARED] = nr_pool_chans[1] = 3

   nr_pool_chans[MV61_VDMA_MEMOPS] = nr_pool_chans[2] = 1

   nr_pool_chans[MV61_VDMA_MEMOPS] = nr_pool_chans[3] = 0


   nr_virt_chans[MV61_VDMA_OWNED] = nr_virt_chans[0] = 4

   nr_virt_chans[MV61_VDMA_SHARED] = nr_virt_chans[1] = 3

   nr_virt_chans[MV61_VDMA_MEMOPS] = nr_virt_chans[2] = 1

   nr_virt_chans[MV61_VDMA_MEMOPS] = nr_virt_chans[3] = 0
}
```

这是静态指定的。

```
            vdma-owned {
                    compatible = "mrvl,mv61_vdma";
                    id = <0>;
            };

            vdma-shared {
                    compatible = "mrvl,mv61_vdma";
                    id = <1>;
            };

            vdma-cyclic {
                    compatible = "mrvl,mv61_vdma";
                    id = <2>;
            };

            vdma-memops {
                    compatible = "mrvl,mv61_vdma";
                    id = <3>;
            };
```

这里id = <?>的作用?

每种类型的virtual dma controller就有一个struct mv61_vdma   *mv61v

这样根据dts就有4个virtual dma controller。

在代表physical cdma controller的struct mv61_dma中的

struct mv61_vdma *mv61v[MV61_NR_VDMA_CONTROLLERS];

就表达这种关系。

```
struct mv61_dma {
        struct device           *dev;
        void __iomem            *ch_regs[MV61_DMA_MAX_NR_PCHANNELS];
        void __iomem            *CDMAInt;
        int                     pchannels;
        u32                     irq_call_cnt;
        struct tasklet_struct   tasklet;

        struct mv61_dma_dispatch *dispatch;
        struct mv61_dma_vpmap   *vpmap;
        struct kmem_cache       *desc_cachep;
        struct kmem_cache       *chain_cachep;
struct mv61_vdma        *mv61v[MV61_NR_VDMA_CONTROLLERS];

        spinlock_t              all_chains_lock;
        struct list_head        all_chains;

        spinlock_t              biglock;
        int reva;
        struct mv61_pdma_chan   chan[0];
};
```

dts中的id = <?>就是表示该vdma在struct mv61_vdma*mv61v[MV61_NR_VDMA_CONTROLLERS]中的index。

比如

```
        vdma-shared {
                compatible = "mrvl,mv61_vdma";
                id = <1>;
        };
```

表示vdma-shared virtual dma device在struct mv61_dma中的位置(index)是1。

在mv61_vdma_probe()中表达了上面的关系联接。

```
        size = sizeof(struct mv61_vdma);
        size += pdata->nr_virt_chans[vid] * sizeof(struct mv61_vdma_chan);

        mv61v = devm_kzalloc(&pdev->dev, size, GFP_KERNEL);
```

```
struct mv61_vdma {
        struct dma_device       dma;
        struct mv61_dma         *mv61p;
        enum mv61_vdma_type     vtype;
        struct mv61_vdma_chan   chan[0];
};
```

mv61_vdma也是个未定size的structure，因为该virtual dma controller所包含的virtual channel是在dts中指定的。

比如

```
                max_owned = <4>;
                max_shared = <3>;
                max_cyclic = <1>;
                max_memops = <0>;
```

也就是

struct mv61_vdma {

struct dma_device dma;

struct mv61_dma   *mv61p;

enum mv61_vdma_type vtype;

struct mv61_vdma_chan   chan[4];

} vdma-owned ;

struct mv61_vdma {

struct dma_device dma;

struct mv61_dma   *mv61p;

enum mv61_vdma_type vtype;

        struct mv61_vdma_chan   chan[3];

} vdma-shared ;

struct mv61_vdma {

        struct dma_device dma;

        struct mv61_dma   *mv61p;

        enum mv61_vdma_type vtype;

        struct mv61_vdma_chan   chan[1];

} vdma-cyclic;

struct mv61_vdma {

        struct dma_device dma;

        struct mv61_dma   *mv61p;

        enum mv61_vdma_type vtype;

        struct mv61_vdma_chan   chan[0];

} vdma-memops;

```
mv61p->mv61v[vid] = mv61v;
```

把生成的struct mv61_vdma赋值给代表cdma controller的mv61v[]

struct mv61_dma {

    struct mv61_vdma   *mv61v[MV61_NR_VDMA_CONTROLLERS];

};

这样当mv61_vdma_probe()运行4次以后(每个virtual dma controller运行一次)

```
struct mv61_dma {
    struct mv61_vdma      *mv61v[0] = vdma-owned;
    struct mv61_vdma      *mv61v[1] = vdma-shared;
    struct mv61_vdma      *mv61v[2] = vdma-cyclic;
    struct mv61_vdma      *mv61v[3] = vdma-memops;
};
```

这里的0,1,2,3就是dts中的

```
        vdma-owned {
                compatible = "mrvl,mv61_vdma";
                id = <0>;
        };

        vdma-shared {
                compatible = "mrvl,mv61_vdma";
                id = <1>;
        };

        vdma-cyclic {
                compatible = "mrvl,mv61_vdma";
                id = <2>;
        };

        vdma-memops {
                compatible = "mrvl,mv61_vdma";
        id = <3>;

    };
```