static struct comip\_pcm\_dma\_params comip\_i2s\_pcm\_stereo[2] = {//DMA的硬件描述

.name = "I2S PCM Stereo out", .name = "I2S PCM Stereo in",

};

2:

获取dma资源:

static struct resource comip\_resource\_i2s0[] = {

[2] = { /\* RX \*/

.start = TOP\_DMAS\_CH8,

.end = TOP\_DMAS\_CH8,

.flags = IORESOURCE\_DMA,

},

[3] = { /\* TX \*/

.start = TOP\_DMAS\_CH0,

.end = TOP\_DMAS\_CH0,

.flags = IORESOURCE\_DMA,

},

};

mmres = platform\_get\_resource(pdev, IORESOURCE\_MEM, 0);

ramres = platform\_get\_resource(pdev, IORESOURCE\_MEM, 1);

dmarxres = platform\_get\_resource(pdev, IORESOURCE\_DMA, 0);

dmatxres = platform\_get\_resource(pdev, IORESOURCE\_DMA, 1);

3:comip\_pcm\_dma\_params通道和物理地址

comip\_i2s\_pcm\_stereo[0].channel = dmatxres->start;

comip\_i2s\_pcm\_stereo[0].daddr = mmres->start + I2S\_TRAN\_FIFO;

comip\_i2s\_pcm\_stereo[1].channel = dmarxres->start;

comip\_i2s\_pcm\_stereo[1].saddr = mmres->start + I2S\_REC\_FIFO;

4：i2s平台驱动设置comip\_pcm\_dma\_param

comip\_i2s\_hw\_params

--->cpu\_dai->playback\_dma\_data = &comip\_i2s\_pcm\_stereo[0];

--->cpu\_dai->capture\_dma\_data = &comip\_i2s\_pcm\_stereo[1];

5：i2s平台驱动获取comip\_pcm\_dma\_param 根据DMA通道名字申请

comip\_pcm\_hw\_params

--->comip\_pcm\_dma\_params \*dma = snd\_soc\_dai\_get\_dma\_data(rtd->cpu\_dai, substream);

--->prtd->params = dma;

--->comip\_dmas\_request(prtd->params->name, dma->channel);

--->module = DMAS\_MODULE(ch);

--->dmas = DMAS\_CHANNEL(ch);

--->dmas->name = name;

--->dmas->flags |= DMAS\_CH\_REQUESTED;//表明这个DMA已经申请了

--->prtd->dma\_ch = dma->channel;

--->snd\_pcm\_set\_runtime\_buffer(substream, &substream->dma\_buffer);

运行时的dma buffer

--->runtime->dma\_buffer\_p = bufp;

--->runtime->dma\_area = bufp->area;

--->runtime->dma\_addr = bufp->addr;

--->runtime->dma\_bytes = bufp->bytes;

6:dma 工作config

comip\_pcm\_prepare

--->config = &pcm\_dmas\_config[1];

--->config->block\_size = runtime->period\_size\*(runtime->frame\_bits / 8);

--->config->src\_addr = (u32)runtime->dma\_addr;

--->config->dst\_addr = dma->daddr;

--->config->bus\_width = DMAS\_DEV\_WIDTH\_16BIT;

--->config->priority = DMAS\_CH\_PRI\_DEFAULT;

--->config->flags = DMAS\_CFG\_ALL;

--->config->irq\_handler = comip\_pcm\_dma\_irq;//dma中断

--->comip\_dmas\_config(dma->channel,config);

--->module = DMAS\_MODULE(ch);

--->dmas = DMAS\_CHANNEL(ch);

--->\_\_comip\_dmas\_config(ch, cfg);

--->writel(block\_size, DMAS\_CH\_CTL0(dmas));

--->writel(cfg->src\_addr, DMAS\_CH\_SAR(dmas));

--->writel(cfg->dst\_addr, DMAS\_CH\_DAR(dmas));

--->DMAS\_BIT\_VALUE(reg\_ctl1, 0xf, 8, cfg->priority);

--->DMAS\_BIT\_VALUE(reg\_ctl1, 0x1, 3, cfg->rx\_trans\_type);

--->writel(int\_en, DMAS\_REG(module, module->int\_en0\_reg));//中断使能

--->dmas->irq\_handler = cfg->irq\_handler;//中断处理

--->dmas->irq\_data = cfg->irq\_data;//中断数据

--->dmas->flags |= DMAS\_CH\_CONFIGURED;//dma已经使用

7:pcm 启动

comip\_pcm\_trigger

--->comip\_dmas\_start(prtd->dma\_ch);

--->writel(dmas->index, DMAS\_REG(module, DMAS\_EN));

soc\_driver ---> “soc-audio”

soc\_probe

--->snd\_soc\_register\_card(card)

--->snd\_soc\_instantiate\_card(card);

--->soc\_probe\_link\_dais(card, i, order);

--->soc\_new\_pcm(rtd, num);

--->platform->driver->pcm\_new(rtd)

--->.pcm\_new = comip\_dma\_new

--->card->dev->dma\_mask = &comip\_pcm\_dmamask; =COMIP\_DMA\_BIT\_MASK=DMA\_BIT\_MASK(64)//dma的寻址能力64

--->card->dev->coherent\_dma\_mask = COMIP\_DMA\_BIT\_MASK;=DMA\_BIT\_MASK(64)//DMA一致性映射能力

--->comip\_pcm\_preallocate\_dma\_buffer(pcm, SNDRV\_PCM\_STREAM\_PLAYBACK);//创建播放通道

--->snd\_pcm\_substream \*substream = pcm->streams[stream].substream;

--->snd\_dma\_buffer \*buf = &substream->dma\_buffer;

--->size = comip\_pcm\_hardware.buffer\_bytes\_max;

--->buf->area = dma\_alloc\_writecombine(pcm->card->dev, size,

&buf->addr, GFP\_KERNEL);

--->dma\_set\_attr(DMA\_ATTR\_WRITE\_COMBINE, &attrs);

--->dma\_alloc\_attrs(dev, size, dma\_handle, flag, &attrs);

--->cpu\_addr = ops->alloc(dev, size, dma\_handle, flag, attrs);

--->.alloc            = arm\_dma\_alloc

---> dma\_alloc\_from\_coherent(dev, size, handle, &memory)

--->\_\_dma\_alloc(dev, size, handle, gfp, prot, false,\_\_builtin\_return\_address(0));

--->debug\_dma\_alloc\_coherent(dev, size, \*dma\_handle, cpu\_addr);

--->buf->bytes = size;

enum {

DMAS\_CH0 = 0,

DMAS\_CH1,

DMAS\_CH2,

DMAS\_CH3,

DMAS\_CH4,

DMAS\_CH5,

DMAS\_CH6,

DMAS\_CH7,

DMAS\_CH8,

DMAS\_CH9,

DMAS\_CH10,

DMAS\_CH11,

DMAS\_CH12,

DMAS\_CH13,

DMAS\_CH14,

DMAS\_CH15,

#if defined(AUDIO\_DMAS\_BASE)

AUDIO\_DMAS\_CH0,

AUDIO\_DMAS\_CH1,

AUDIO\_DMAS\_CH2,

AUDIO\_DMAS\_CH3,

AUDIO\_DMAS\_CH4,

AUDIO\_DMAS\_CH5,

AUDIO\_DMAS\_CH6,

AUDIO\_DMAS\_CH7,

AUDIO\_DMAS\_CH8,

AUDIO\_DMAS\_CH9,

AUDIO\_DMAS\_CH10,

AUDIO\_DMAS\_CH11,

AUDIO\_DMAS\_CH12,

AUDIO\_DMAS\_CH13,

AUDIO\_DMAS\_CH14,

AUDIO\_DMAS\_CH15,

#endif

#if defined(TOP\_DMAS\_BASE)

TOP\_DMAS\_CH0,

TOP\_DMAS\_CH1,

TOP\_DMAS\_CH2,

TOP\_DMAS\_CH3,

TOP\_DMAS\_CH4,

TOP\_DMAS\_CH5,

TOP\_DMAS\_CH6,

TOP\_DMAS\_CH7,

TOP\_DMAS\_CH8,

TOP\_DMAS\_CH9,

TOP\_DMAS\_CH10,

TOP\_DMAS\_CH11,

TOP\_DMAS\_CH12,

TOP\_DMAS\_CH13,

TOP\_DMAS\_CH14,

TOP\_DMAS\_CH15,

#endif

DMAS\_CH\_MAX

};

static struct comip\_dmas\_module comip\_dmas[] = {

{

.name = "ap\_dmas",

#if defined(CONFIG\_ARCH\_LC181X)

.clk\_name = "bus\_mclk",

#elif defined(CONFIG\_ARCH\_LC186X)

.clk\_name = "bus\_mclk1",

#else

#error unknown arch

#endif

.base = AP\_DMAS\_BASE,

.start = DMAS\_CH0,

.end = DMAS\_CH15,

.int\_en0\_reg = DMAS\_INT\_EN0,

.int0\_reg = DMAS\_INT0,

.int\_en1\_reg = DMAS\_INT\_EN1,

.int1\_reg = DMAS\_INT1,

.irq = INT\_AP\_DMAS,

},

#if defined(AUDIO\_DMAS\_BASE)

{

.name = "audio\_dmas",

.clk\_name = "bus\_mclk",

.base = AUDIO\_DMAS\_BASE,

.start = AUDIO\_DMAS\_CH0,

.end = AUDIO\_DMAS\_CH15,

.int\_en0\_reg = DMAS\_INT\_EN0,

.int0\_reg = DMAS\_INT0,

.int\_en1\_reg = DMAS\_INT\_EN1,

.int1\_reg = DMAS\_INT1,

.irq = INT\_AUDIO\_DMAS,

},

#endif

#if defined(TOP\_DMAS\_BASE)

{

.name = "top\_dmas",

.clk\_name = "top\_bus\_clk",

.base = TOP\_DMAS\_BASE,

.start = TOP\_DMAS\_CH0,

.end = TOP\_DMAS\_CH15,

.int\_en0\_reg = DMAS\_INT\_EN0\_TOP,

.int0\_reg = DMAS\_INT0\_TOP,

.int\_en1\_reg = DMAS\_INT\_EN1\_TOP,

.int1\_reg = DMAS\_INT1\_TOP,

.irq = INT\_TOP\_DMAS,

.channel = {

[0] = {

.flags = DMAS\_CH\_BLK\_SIZE\_UNIT\_WORD,

},

}

},

#endif

};

comip\_pcm\_dma\_irq

--->snd\_pcm\_period\_elapsed(substream);

--->snd\_pcm\_update\_hw\_ptr0(substream, 1)

---> substream->ops->pointer(substream);

--->.pointer = comip\_pcm\_pointer

--->comip\_dmas\_get(prtd->dma\_ch, &ptr);

--->dma\_ptr = (dma\_addr\_t)ptr & ~((u32)0);

--->x = bytes\_to\_frames(runtime, dma\_ptr - runtime->dma\_addr);

--->snd\_timer\_interrupt(substream->timer, 1)

snd\_pcm\_ops trigger

Lc1881 comip\_snd\_pcm\_trigger

--->case SNDRV\_PCM\_TRIGGER\_START

--->comip\_dmas\_start(prtd->dma\_data->channel);

--->SNDRV\_PCM\_TRIGGER\_STOP:

--->comip\_dmas\_stop(prtd->dma\_data->channel);

Lc1860 comip\_pcm\_trigger

--->case SNDRV\_PCM\_TRIGGER\_START:

--->comip\_dmas\_start(prtd->dma\_ch);

--->case SNDRV\_PCM\_TRIGGER\_STOP:

--->comip\_dmas\_stop(prtd->dma\_ch);

--->comip\_dmas\_intr\_disable(prtd->dma\_ch, DMAS\_INT\_DONE);

snd\_pcm\_ops pointer

Lc1881 comip\_snd\_pcm\_pointer

--->comip\_dmas\_get(prtd->dma\_data->channel, &addr);

--->dma\_ptr = (dma\_addr\_t)addr & ~((u32)0);

--->bytes\_to\_frames(runtime, dma\_ptr - runtime->dma\_addr);

Lc1860 comip\_pcm\_pointer

--->comip\_dmas\_get(prtd->dma\_ch, &ptr);

--->dma\_ptr = (dma\_addr\_t)ptr & ~((u32)0);

--->x = bytes\_to\_frames(runtime, dma\_ptr - runtime->dma\_addr);

--->if (x == runtime->buffer\_size)

--->x = 0;

snd\_pcm\_ops mmap

Lc1881 comip\_snd\_pcm\_mmap

--->ret = dma\_mmap\_writecombine(substream->pcm->card->dev, vma,

runtime->dma\_area,

runtime->dma\_addr,

runtime->dma\_bytes);

Lc1860 comip\_pcm\_mmap

--->dma\_mmap\_writecombine(substream->pcm->card->dev, vma,

runtime->dma\_area,

runtime->dma\_addr,

runtime->dma\_bytes);

.mmap = comip\_pcm\_mmap

--->snd\_pcm\_runtime \*runtime = substream->runtime;

--->dma\_mmap\_writecombine(substream->pcm->card->dev, vma,

runtime->dma\_area,

runtime->dma\_addr,

runtime->dma\_bytes);

--->dma\_set\_attr(DMA\_ATTR\_WRITE\_COMBINE, &attrs);

--->dma\_mmap\_attrs(dev, vma, cpu\_addr, dma\_addr, size, &attrs);

--->dma\_map\_ops \*ops = get\_dma\_ops(dev);

--->ops->mmap(dev, vma, cpu\_addr, dma\_addr, size, attrs);

--->dma\_common\_mmap(dev, vma, cpu\_addr, dma\_addr, size);

--->vma->vm\_page\_prot = pgprot\_noncached(vma->vm\_page\_prot);

--->dma\_mmap\_from\_coherent(dev, vma, cpu\_addr, size, &ret)

--->remap\_pfn\_range(vma, vma->vm\_start,

pfn + off,

user\_count << PAGE\_SHIFT,

vma->vm\_page\_prot);

pcm\_new

1881 comip\_pcm\_new

--->card->dev->dma\_mask = &comip\_dma\_mask;

--->card->dev->coherent\_dma\_mask = comip\_dma\_mask;

--->buf = &substream->dma\_buffer;

--->buf->dev.type = SNDRV\_DMA\_TYPE\_DEV;

--->buf->dev.dev = pcm->card->dev;

--->buf->private\_data = pdata;

--->size = comip\_pcm\_hardware\_playback.buffer\_bytes\_max;

--->buf->area = dma\_alloc\_coherent(pcm->card->dev, size,&buf->addr, GFP\_KERNEL);

1860 comip\_pcm\_dma\_new

--->card->dev->dma\_mask = &comip\_pcm\_dmamask;

--->card->dev->coherent\_dma\_mask = COMIP\_DMA\_BIT\_MASK;

---> \*buf = &substream->dma\_buffer;

--->size = comip\_pcm\_hardware.buffer\_bytes\_max;

--->buf->dev.type = SNDRV\_DMA\_TYPE\_DEV;

--->buf->dev.dev = pcm->card->dev;

--->buf->area = dma\_alloc\_writecombine(pcm->card->dev, size,&buf->addr, GFP\_KERNEL);

--->buf->bytes = size;

snd\_pcm\_ops open

1881 comip\_snd\_pcm\_open

--->snd\_soc\_set\_runtime\_hwparams(substream, &comip\_pcm\_hardware\_playback);

--->snd\_pcm\_hw\_constraint\_integer(runtime, SNDRV\_PCM\_HW\_PARAM\_PERIODS);

--->prtd->period\_index = 0;

--->runtime->private\_data = prtd;

1860 comip\_pcm\_open

--->snd\_soc\_set\_runtime\_hwparams(substream, &comip\_pcm\_hardware);

--->snd\_pcm\_hw\_constraint\_integer(runtime, SNDRV\_PCM\_HW\_PARAM\_PERIODS);

--->runtime->private\_data = prtd;

snd\_pcm\_ops hw\_params

1881 comip\_snd\_pcm\_hw\_params

--->dma\_data = snd\_soc\_dai\_get\_dma\_data(rtd->cpu\_dai, substream);

--->prtd->dma\_data = dma\_data;

--->comip\_dmas\_request(dma\_data->name, dma\_data->channel);

--->snd\_pcm\_set\_runtime\_buffer(substream, &substream->dma\_buffer);

--->runtime->dma\_bytes = params\_buffer\_bytes(params);

1860 comip\_pcm\_hw\_params

---> \*dma = snd\_soc\_dai\_get\_dma\_data(rtd->cpu\_dai, substream);

--->totsize = params\_buffer\_bytes(params);

--->prtd->params = dma;

--->ret = comip\_dmas\_request(prtd->params->name, dma->channel);

--->prtd->dma\_ch = dma->channel;

--->snd\_pcm\_set\_runtime\_buffer(substream, &substream->dma\_buffer);

--->runtime->dma\_bytes = totsize;

--->prtd->dmas1\_index = 0;

--->prtd->off\_index = 0;

1860 comip\_pcm\_hw\_free --->1881 comip\_snd\_pcm\_hw\_free 一致

snd\_pcm\_ops prepare

1881 comip\_snd\_pcm\_prepare

--->dma\_data = snd\_soc\_dai\_get\_dma\_data(rtd->cpu\_dai, substream);

--->config = &prtd->dmas\_config[0];

--->config->block\_size = runtime->period\_size \* (runtime->frame\_bits / 8);

--->config->src\_addr = (u32)runtime->dma\_addr;

--->config->dst\_addr = dma\_data->dst\_addr;

--->config->bus\_width = DMAS\_DEV\_WIDTH\_16BIT;

--->config->priority = DMAS\_CH\_PRI\_DEFAULT;

--->config->flags = DMAS\_CFG\_ALL;

--->config->tx\_trans\_mode = DMAS\_TRANS\_NORMAL;

--->config->tx\_fix\_value = 0;

--->config->tx\_block\_mode = DMAS\_MULTI\_BLOCK;

--->config->rx\_trans\_type = DMAS\_TRANS\_BLOCK;

--->config->rx\_timeout = 0;

--->config->irq\_en = DMAS\_INT\_DONE;

--->config->irq\_handler = comip\_snd\_pcm\_dma\_irq;

--->config->irq\_data = substream;

--->comip\_dmas\_config(dma\_data->channel, config);

1860 comip\_pcm\_prepare

--->\*dma = snd\_soc\_dai\_get\_dma\_data(rtd->cpu\_dai, substream);

--->\*config = &pcm\_dmas\_config[0];

--->config->block\_size = runtime->period\_size\*(runtime->frame\_bits / 8);

--->config->src\_addr = runtime->dma\_addr;

--->config->dst\_addr = dma->daddr;

--->config->bus\_width = DMAS\_DEV\_WIDTH\_16BIT;

--->config->priority = DMAS\_CH\_PRI\_DEFAULT;

--->config->flags = DMAS\_CFG\_ALL;

--->config->tx\_trans\_mode = DMAS\_TRANS\_NORMAL;

--->config->tx\_fix\_value = 0;

--->config->tx\_block\_mode = DMAS\_SINGLE\_BLOCK;

--->config->rx\_trans\_type = DMAS\_TRANS\_BLOCK;

--->config->rx\_timeout = 0;

--->config->irq\_en = DMAS\_INT\_DONE;

--->config->irq\_handler = comip\_pcm\_dma\_irq;

--->config->irq\_data = substream;

--->comip\_dmas\_config(dma->channel,config);

alsa\_sound\_init

--->register\_chrdev(major, "alsa", &snd\_fops)

--->.open = snd\_open,

---> file->f\_op->open(inode, file)

snd\_register\_device\_for\_dev(devtype, pcm->card,

pcm->device,

&snd\_pcm\_f\_ops[cidx],

pcm, str, dev);

--->snd\_pcm\_f\_ops

--->.open = snd\_pcm\_playback\_open

--->snd\_pcm\_open(file, pcm, SNDRV\_PCM\_STREAM\_PLAYBACK);

--->snd\_pcm\_open\_file(file, pcm, stream);

--->snd\_pcm\_open\_substream(pcm, stream, file, &substream);

--->substream->ops->open(substream))

.open = comip\_pcm\_open

--->snd\_soc\_set\_runtime\_hwparams(substream, &comip\_pcm\_hardware);

--->runtime->hw.info = hw->info;

--->runtime->hw.formats = hw->formats;

--->runtime->hw.buffer\_bytes\_max = hw->buffer\_bytes\_max;

--->snd\_pcm\_hw\_constraint\_integer(runtime, SNDRV\_PCM\_HW\_PARAM\_PERIODS);

--->snd\_interval\_setinteger(constrs\_interval(constrs, var));

comip\_pcm\_prepare

--->comip\_dmas\_config(dma->channel,config);

--->module = DMAS\_MODULE(ch);=DMAS\_MODULE(TOP\_DMAS\_CH2)=DMAS\_MODULE(34)

--->(&comip\_dmas[ch / DMAS\_MODULE\_CHANNEL])=&comip\_dmas[34 / 16]=&comip\_dmas[2]

对应top\_dmas

--->dmas = DMAS\_CHANNEL(ch);

--->(&DMAS\_MODULE(ch)->channel[ch % DMAS\_MODULE\_CHANNEL])

--->(&DMAS\_MODULE(34)->channel[34 % 16])=(&DMAS\_MODULE(34)->channel[2])=&comip\_dmas[2]->channel[2]

--->comip\_dmas[2]对应top\_dmas, channel[2]对应pcm发送。

--->\_\_comip\_dmas\_config(ch, cfg);

--->\*module = DMAS\_MODULE(ch);

--->\*dmas = DMAS\_CHANNEL(ch);