```
    sic: interrupt-controller@f9300100 {
        compatible = "arm,cortex-a15-sic", "arm,cortex-a9-sic";
        #interrupt-cells = <1>;
        #address-cells = <1>;
        interrupt-controller;
        interrupt-parent = <&gic>;
        // name conflict between mv6220 & 6270, using actual address
        reg = <0 0xf9300100 0 0x1000>;
        interrupts = <0 212 4>;
        num_irqs = <32>;
        status = "disabled";
    };
```

关键是

```
#interrupt-cells = <1>;
```

即只用一个number就可以指定SIC上的interrupt specifier.

```
    dec-fuser {
        compatible = "dec-fuser";
        id = <0>;
        reg = <0 IPS_DEC_DEC_FUSER_BASE 0 0x100>;
        interrupt-parent = <&sic>;
        interrupts = <26>;
        status = "disabled";
    };
```

dec-fuser device指定的SIC的interrupt如下

> interrupts = <26>;

在3.18.7的irq-sic.c

```
static struct irq_domain_ops sic_irqdomain_ops = {
        .map = sic_irqdomain_map,
};
```

in kernel/irq/irqdomain.c

```
    /* If domain has no translation, then we assume interrupt line */
    if (domain->ops->xlate == NULL)
        hwirq = irq_data->args[0];
    else {
        if (domain->ops->xlate(domain, irq_data->np, irq_data->args,
                    irq_data->args_count, &hwirq, &type))
            return 0;
    }
```

这里sic_irqdomain_ops.xlate没有定义，即NULL，所以domain->ops->xlate为NULL!
所以hwirq = irq_data->args[0] = 26

但在4.2.8的irq-sic.c中

```
1.  static struct irq_domain_ops sic_irqdomain_ops = {
2.      .map = sic_irqdomain_map,
3.      .xlate = sic_irq_domain_xlate,
4.  };
```

定义了sic_irqdomain_ops.xlate。
这样就会运行如下code

```
1.          if (domain->ops->xlate(domain, irq_data->np, irq_data->args,
2.                      irq_data->args_count, &hwirq, &type))
3.              return 0;
```

即运行

```
1.  static int sic_irq_domain_xlate(struct irq_domain *d,
2.                 struct device_node *controller,
3.                 const u32 *intspec, unsigned int intsize,
4.                 unsigned long *out_hwirq, unsigned int *out_type)
5.  {
6.      if (d->of_node != controller)
7.          return -EINVAL;
8.      if (intsize < 3)
9.          return -EINVAL;
10.
11.     *out_hwirq = intspec[1];
12.
13.     *out_type = intspec[2] & IRQ_TYPE_SENSE_MASK;
14.
15.     return 0;
16. }
```

这里的intsize = 1,所以return -EINVAL。
也就是dec的各个driver都获取irq失败！

从4.2.8的code看，新的SIC的irq specifier应该如下
[x hw-irq trigger]
intspec[0]没有使用，所以可以任意值。
intspec[1]就是这里的26 (hardware interrupt number)
intspec[2]是触发interrupt的模式
定义在include/irq.h

```
/*
 * IRQ line status.
 *
 * Bits 0-7 are the same as the IRQF_* bits in linux/interrupt.h
 *
 * IRQ_TYPE_NONE        - default, unspecified type
 * IRQ_TYPE_EDGE_RISING     - rising edge triggered
 * IRQ_TYPE_EDGE_FALLING    - falling edge triggered
 * IRQ_TYPE_EDGE_BOTH       - rising and falling edge triggered
 * IRQ_TYPE_LEVEL_HIGH      - high level triggered
 * IRQ_TYPE_LEVEL_LOW       - low level triggered
 * IRQ_TYPE_LEVEL_MASK      - Mask to filter out the level bits
 * IRQ_TYPE_SENSE_MASK      - Mask for all the above bits
 * IRQ_TYPE_DEFAULT     - For use by some PICs to ask irq_set_type
 *                to setup the HW to a sane default (used
 *                                by irqdomain map() callbacks to synchroniz
e
 *                                the HW state and SW flags for a newly
 *                                allocated descriptor).
 *
 * IRQ_TYPE_PROBE       - Special flag for probing in progress
 *
 * Bits which can be modified via irq_set/clear/modify_status_flags()
 * IRQ_LEVEL            - Interrupt is level type. Will be also
 *                updated in the code when the above trigger
 *                bits are modified via irq_set_irq_type()
 * IRQ_PER_CPU          - Mark an interrupt PER_CPU. Will protect
 *                it from affinity setting
 * IRQ_NOPROBE          - Interrupt cannot be probed by autoprobing
 * IRQ_NOREQUEST        - Interrupt cannot be requested via
 *                request_irq()
 * IRQ_NOTHREAD         - Interrupt cannot be threaded
 * IRQ_NOAUTOEN         - Interrupt is not automatically enabled in
 *                request/setup_irq()
 * IRQ_NO_BALANCING     - Interrupt cannot be balanced (affinity set)
 * IRQ_MOVE_PCNTXT      - Interrupt can be migrated from process context
 * IRQ_NESTED_TRHEAD        - Interrupt nests into another thread
 * IRQ_PER_CPU_DEVID        - Dev_id is a per-cpu variable
 * IRQ_IS_POLLED        - Always polled by another interrupt. Exclude
 *                it from the spurious interrupt detection
 *                mechanism and from core side polling.
 */
enum {
    IRQ_TYPE_NONE       = 0x00000000,
    IRQ_TYPE_EDGE_RISING    = 0x00000001,
    IRQ_TYPE_EDGE_FALLING   = 0x00000002,
    IRQ_TYPE_EDGE_BOTH  = (IRQ_TYPE_EDGE_FALLING | IRQ_TYPE_EDGE_RISING),
    IRQ_TYPE_LEVEL_HIGH = 0x00000004,
    IRQ_TYPE_LEVEL_LOW  = 0x00000008,
    IRQ_TYPE_LEVEL_MASK = (IRQ_TYPE_LEVEL_LOW | IRQ_TYPE_LEVEL_HIGH),
    IRQ_TYPE_SENSE_MASK = 0x0000000f,
    IRQ_TYPE_DEFAULT    = IRQ_TYPE_SENSE_MASK,

```

```
    IRQ_TYPE_PROBE      = 0x00000010,

    IRQ_LEVEL       = (1 <<  8),
    IRQ_PER_CPU     = (1 <<  9),
    IRQ_NOPROBE     = (1 << 10),
    IRQ_NOREQUEST       = (1 << 11),
    IRQ_NOAUTOEN        = (1 << 12),
    IRQ_NO_BALANCING    = (1 << 13),
    IRQ_MOVE_PCNTXT     = (1 << 14),
    IRQ_NESTED_THREAD   = (1 << 15),
    IRQ_NOTHREAD        = (1 << 16),
    IRQ_PER_CPU_DEVID   = (1 << 17),
    IRQ_IS_POLLED       = (1 << 18),
};
```

并且SIC interrupt controller 改为

#interrupt-cells = <3>;

```
    sic: interrupt-controller@f9300100 {
        compatible = "arm,cortex-a15-sic", "arm,cortex-a9-sic";
        #interrupt-cells = <3>;
        #address-cells = <1>;
        interrupt-controller;
        interrupt-parent = <&gic>;
        // name conflict between mv6220 & 6270, using actual address
        reg = <0 0xf9300100 0 0x1000>;
        interrupts = <0 212 4>;
        num_irqs = <32>;
        status = "disabled";
    };
```

---

debugging tips:

如果device获取virtual interrupt失败

> res_data->irq = irq_of_parse_and_map(parent_node, 0);
> 则可以先分别出哪部分失败。
> in drivers/of/irq.c

```
unsigned int irq_of_parse_and_map(struct device_node *dev, int index)
{
    struct of_phandle_args oirq;

    if (of_irq_parse_one(dev, index, &oirq))
        return 0;

    return irq_create_of_mapping(&oirq);
}
```

of_irq_parse_one() parses dtb

irq_create_of_mapping()由interrupt controller的driver来mapping。
先区分上面哪部分出错！