ePAPR中对ranges的description

Property: ranges

Value type: <empty> or <prop-encoded-array> encoded as arbitrary number of triplets of

(child-bus-address, parent-bus-address, length).

child-bus-address的size (由多少个WORD组成)由child device的#address-cells指定

parent-bus-address的size (由多少个WORD组成)由parentdevice的#address-cells指定

```
/ {
        #address-cells = <0x2>;
        #size-cells = <0x2>;
        model = "mv6220 TurnOn Card";
        compatible = "marvell,mv6220-toc", "marvell,mv6220", "marvell,pegmatite";
        interrupt-parent = <0x1>;


        squ@d1000000 {
                compatible = "mmio-sram";
                reg = <0x0 0xd1000000 0x0 0x18000>;
                clocks = <0x2>;
                #address-cells = <0x1>;
                #size-cells = <0x1>;
                ranges = <0x0 0x0 0xd1000000 0x18000>;
                linux,phandle = <0x4>;
                phandle = <0x4>;
```

```
            smpboot-sram@0 {
                compatible = "marvell,pegmatite-smpboot-sram";
                reg = <0x0 0x20>;
            };
        };

};
```

squu@d1000000 device的parents的

```
        #address-cells = <0x2>;
        #size-cells = <0x2>;
```

squu@d1000000 device的child device(smpboot-sram@0)的

```
            #address-cells = <0x1>;
            #size-cells = <0x1>;
```

ranges = <0x0 0x0 0xd1000000 0x18000>;

match

(child-bus-address, parent-bus-address, length) triplet

child-bus-address = <**0x0** 0x0 0xd1000000 0x18000>

parent-bus-address = <0x0 **0x0 0xd1000000** 0x18000>

length = <0x0 0x0 0xd1000000 **0x18000**>

the 1st value 0x0 is child device's address-cells (1 size)

the 2nd and 3rd values 0x0 and 0xd1000000 are parents' device address-cells(2 size)

即smpboot-sram@0 device的address 0x0 mapping to parents' device address 0x0 0xd1000000 (这里用了64-bit address,可能是考虑到以后ARM64吧)

The 4th value 0x18000 is the mapping size。The whole squ size is 0x18000.

smpboot-sram@0 device的

reg = <0x0 0x20>;

表示其在SoC的physical address space为<0x0 0xd1000000 0x0 0x20>,即[0xd1000000, 0xd1000020)

==========

drivers/of/address.c中的

u64 of_translate_address(struct device_node *dev, const __be32 *in_addr)

即用于"ranges" property的parsing.

```
int of_translate_one(
        struct device_node *parent,
        struct of_bus *bus,
        struct of_bus *pbus,
        __be32 *addr,
        int na,
        int ns,
        int pna,
        const char *rprop);
```

support "ranges" property的核心函数。

range = <child-addr, parent-addr, size>

这里的na是child-addr的长度，pna是parent-addr的长度，ns就是这里的size。

range包含了 na + pna + ns 个cells.

parent指向当前device node的parent device node.

addr指向当前devcie node的地址信息(reg = <...>)

For example,

```
        squ@d1000000 {
                compatible = "mmio-sram";
                reg = <0x0 0xd1000000 0x0 0x18000>;
                clocks = <0x2>;
                #address-cells = <0x1>;          ( 1 )
                #size-cells = <0x1>;             ( 2 )
                ranges = <0x0 0x0 0xd1000000 0x18000>;
                linux,phandle = <0x4>;
                phandle = <0x4>;

                smpboot-sram@0 {
                        compatible = "marvell,pegmatite-smpboot-sram";
                        reg = <0x0 0x20>;
                };
        };
```

current device node is smpboot-sram@0, parent device node is squ@d1000000。

addr = reg = <0x0 0x20>

na = 0x1,这是（1）处的值

pna = 0x2，squ@d1000000 device的#address-cells是在root中说明的，为0x2

ns = 0x1,这是（2）处的值

rprop指向"ranges"

bus是current device node所在bus

pbus是parent device node所在bus

kernel 3.18.7支持的bus如下

```
/*
 * Array of bus specific translators
 */

static struct of_bus of_busses[] = {
#ifdef CONFIG_OF_ADDRESS_PCI
	/* PCI */
	{
		.name = "pci",
		.addresses = "assigned-addresses",
		.match = of_bus_pci_match,
		.count_cells = of_bus_pci_count_cells,
		.map = of_bus_pci_map,
		.translate = of_bus_pci_translate,
```

```
		.get_flags = of_bus_pci_get_flags,
	},
#endif /* CONFIG_OF_ADDRESS_PCI */
	/* ISA */
	{
		.name = "isa",
		.addresses = "reg",
		.match = of_bus_isa_match,
		.count_cells = of_bus_isa_count_cells,
		.map = of_bus_isa_map,
		.translate = of_bus_isa_translate,
		.get_flags = of_bus_isa_get_flags,
	},
	/* Default */
	{
		.name = "default",
		.addresses = "reg",
		.match = NULL,
		.count_cells = of_bus_default_count_cells,
		.map = of_bus_default_map,
		.translate = of_bus_default_translate,
		.get_flags = of_bus_default_get_flags,
	},
};
```

对smpboot-sram@0 device而言，bus与pbus都是这里的default bus.

==========

About empty "rangs" property

in Gr2 / Gs2 dts

```
        antic-superblock {
                compatible = "ipg2antic", "g2-img-pipe";
                #address-cells = <0x1>;
                #size-cells = <0x1>;
                reg = <0x0 0xf90d0000 0x0 0x20000>;
                interrupts = <0xf3>;
                clock-frequency = <0xbebc200>;
                ranges;         (1)

                antic@0xf90d0000 {
                        compatible = "ipg2antic0";
                        #address-cells = <0x1>;
                        #size-cells = <0x1>;
                        reg = <0x0 0xf90d0000 0x0 0x10000>;
                        ranges;         (2)

                        antic-top@0xf90d0000 {
                                compatibility = "ipg2antic-top";
                                type = "reg-top";
                                reg = <0x0 0xf90d0000 0x0 0x38>;
                                offset = <0x0>;
```

```
            };

            antic-idma-axi@0xf90d1000 {
                compatibility = "ipg2antic-idma-axi";
                type = "idma-axi";
                reg = <0x0 0xf90d1000 0x0 0x34>;
                offset = <0x1000>;
            };

            antic-odma-axi@0xf90d2000 {
                compatibility = "ipg2antic-odma-axi";
                type = "odma-axi";
                reg = <0x0 0xf90d2000 0x0 0x40>;
                offset = <0x2000>;
            };
        };

        antic-reserve@0xf90e0000 {
            compatible = "ipg2antic-reserve";
            type = "reserve";
            reg = <0x0 0xf90e0000 0x0 0x10000>;
            status = "disabled";
        };
    };
```

empty ranges property的作用：

If no the empty ranges property,means we are crossing a non-translatable boundary, and thus the addresses below the current cannot be converted to CPU physical ones.

antic-superblock device的address是

reg = <0x0 0xf90d0000 0x0 0x20000>; 这是SoC的address

而antic@0xf90d0000 device则是antic-superblock device的child,其地址是

reg = <0x0 0xf90d0000 0x0 0x10000>;

即如果没有（1），那么antic@0xf90d0000 device的address <0x0 0xf90d0000 0x0 0x10000>并不是SoC的address。

而有了（1），使得则个<0x0 0xf90d0000 0x0 0x10000>也被解释成SoC的address。

（2）是同样的道理，是对antic@0xf90d0000 device的3个children的地址的“合法化”申明。