Gemstone2是dual-core
in dts

```
    cpus {
        #address-cells = <0x1>;
        #size-cells = <0x0>;

        cpu@0 {
            device_type = "cpu";
            compatible = "arm,cortex-a53";
            reg = <0xffff00>;
        };

        cpu@1 {
            device_type = "cpu";
            compatible = "arm,cortex-a53";
            reg = <0xffff01>;
            enable-method = "marvell,pegmatite-apmu-boot";
        };
    };
```

in arch/arm/kernel/setup.c

```
void __init setup_arch(char **cmdline_p)
{
......

    unflatten_device_tree();

    arm_dt_init_cpu_maps();
......
}
```

展开device tree后的第一步就是config core。

in arch/arm/kernel/devtree.c

```c
/*
 * arm_dt_init_cpu_maps - Function retrieves cpu nodes from the device tree
 * and builds the cpu logical map array containing MPIDR values related to
 * logical cpus
 *
 * Updates the cpu possible mask with the number of parsed cpu nodes
 */
void __init arm_dt_init_cpu_maps(void)
{
    /*
     * Temp logical map is initialized with UINT_MAX values that are
     * considered invalid logical map entries since the logical map must
     * contain a list of MPIDR[23:0] values where MPIDR[31:24] must
     * read as 0.
     */
    struct device_node *cpu, *cpus;
    int found_method = 0;
    u32 i, j, cpuidx = 1;
    u32 mpidr = is_smp() ? read_cpuid_mpidr() & MPIDR_HWID_BITMASK : 0;

    u32 tmp_map[NR_CPUS] = { [0 ... NR_CPUS-1] = MPIDR_INVALID };
    bool bootcpu_valid = false;
    cpus = of_find_node_by_path("/cpus");        ①

    if (!cpus)
        return;

    for_each_child_of_node(cpus, cpu) {
        u32 hwid;

        if (of_node_cmp(cpu->type, "cpu"))        ②
            continue;

        pr_debug(" * %s...\n", cpu->full_name);
        /*
         * A device tree containing CPU nodes with missing "reg"
         * properties is considered invalid to build the
         * cpu_logical_map.
         */
        if (of_property_read_u32(cpu, "reg", &hwid)) {     ③
            pr_debug(" * %s missing reg property\n",
                     cpu->full_name);
            return;
        }

        /*
         * 8 MSBs must be set to 0 in the DT since the reg property
         * defines the MPIDR[23:0].
         */
        if (hwid & ~MPIDR_HWID_BITMASK)                ④
            return;

        /*
```

```
         * Duplicate MPIDRs are a recipe for disaster.
         * Scan all initialized entries and check for
         * duplicates. If any is found just bail out.
         * temp values were initialized to UINT_MAX
         * to avoid matching valid MPIDR[23:0] values.
         */
        for (j = 0; j < cpuidx; j++)
            if (WARN(tmp_map[j] == hwid, "Duplicate /cpu reg "
                             "properties in the DT\n"))
                return;

        /*
         * Build a stashed array of MPIDR values. Numbering scheme
         * requires that if detected the boot CPU must be assigned
         * logical id 0. Other CPUs get sequential indexes starting
         * from 1. If a CPU node with a reg property matching the
         * boot CPU MPIDR is detected, this is recorded so that the
         * logical map built from DT is validated and can be used
         * to override the map created in smp_setup_processor_id().
         */
        if (hwid == mpidr) {                    ⑤
            i = 0;
            bootcpu_valid = true;
        } else {
            i = cpuidx++;
        }

        if (WARN(cpuidx > nr_cpu_ids, "DT /cpu %u nodes greater than "
                           "max cores %u, capping them\n",
                           cpuidx, nr_cpu_ids)) {
            cpuidx = nr_cpu_ids;
            break;
        }

        tmp_map[i] = hwid;

        if (!found_method)                                    ⑥
            found_method = set_smp_ops_by_method(cpu);
    }

    /*
     * Fallback to an enable-method in the cpus node if nothing found in
     * a cpu node.
     */
    if (!found_method)
        set_smp_ops_by_method(cpus);                        ⑦

    if (!bootcpu_valid) {
        pr_warn("DT missing boot CPU MPIDR[23:0], fall back to default cpu_logical_map\n");
        return;
    }

    /*
```

```
	 * Since the boot CPU node contains proper data, and all nodes have
	 * a reg property, the DT CPU list can be considered valid and the
	 * logical map created in smp_setup_processor_id() can be overridden
	 */
	for (i = 0; i < cpuidx; i++) {            ⑧
		set_cpu_possible(i, true);
		cpu_logical_map(i) = tmp_map[i];
		pr_debug("cpu logical map 0x%x\n", cpu_logical_map(i));
	}
}
```

①
serach `cpus` node
②
每个cpu sub-node必须有

> device_type = "cpu";

③
每个cpu sub-node必须有 `reg` property

> reg = <0xffff00>;

hwid of core 0 = 0xffff00

hwid of core 1 = 0xffff01
这个格式好像必须与MPIDR register中的format匹配。

④

```
#define MPIDR_HWID_BITMASK 0xFFFFFF
```

~MPIDR_HWID_BITMASK = 0xFF0000000

0xffff00 & ~MPIDR_HWID_BITMASK = 0
0xffff01 & ~MPIDR_HWID_BITMASK = 0
即
reg = <24-bit>;
只能设置低24 bits(bit 0 to bit 23) (是否意味着当前ARM kernel最多支持24 cores ?)
⑤
MPIDR register中是boot core,boot core被指定为index 0.
⑥
对boot core,并没有设置 `enable-method` ，所以set_smp_ops_by_method() return 0
对core 1

```
static int __init set_smp_ops_by_method(struct device_node *node)
{
    const char *method;
    struct of_cpu_method *m = __cpu_method_of_table;

    if (of_property_read_string(node, "enable-method", &method))
        return 0;

    for (; m->method; m++)
        if (!strcmp(m->method, method)) {
            smp_set_ops(m->ops);
            return 1;
        }

    return 0;
}
```

method = "marvell,pegmatite-apmu-boot"
in drivers/platform/pegmatite/smp/platsmp.c

```
struct smp_operations pegmatite_smp_ops __initdata = {
    .smp_prepare_cpus    = pegmatite_smp_prepare_cpus,
    .smp_boot_secondary = pegmatite_boot_secondary,
#ifdef CONFIG_HOTPLUG_CPU
    .cpu_die          = pegmatite_cpu_die,
    .cpu_kill                 = pegmatite_cpu_kill,
#endif
};
```

```
CPU_METHOD_OF_DECLARE(pegmatite_smp, "marvell,pegmatite-apmu-boot", &pegmati
te_smp_ops);
```

global variable in arch/arm/kernel/smp.c

> static struct smp_operations smp_ops;

smp_ops = pegmatite_smp_ops

pegmatite_smp_ops中是针对specific CPU core的bootup, 断电等callback.
⑦
这里是针对如下情况

```
    cpus {
        #address-cells = <0x1>;
        #size-cells = <0x0>;

        cpu@0 {
            device_type = "cpu";
            compatible = "arm,cortex-a53";
            reg = <0xffff00>;
        };

        cpu@1 {
            device_type = "cpu";
            compatible = "arm,cortex-a53";
            reg = <0xffff01>;
        };
        enable-method = "marvell,pegmatite-apmu-boot";
    };
```

即 `enable-method` property并不是在特定core中指定，而是对整个 `cpus` 而言的。

⑧

设置"possible cpu"

即 `possible cpu` 是在device tree中指定的cores。