in arch/arm/kernel/setup.c

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

        unflatten_device_tree();

        arm_dt_init_cpu_maps();
        psci_init();
#ifdef CONFIG_SMP
        if (is_smp()) {
                if (!mdesc->smp_init || !mdesc->smp_init()) {
                        if (psci_smp_available())
                                smp_set_ops(&psci_smp_ops);
                        else if (mdesc->smp)
                                smp_set_ops(mdesc->smp);

                }
                smp_init_cpus();
                smp_build_mpidr_hash();
        }
#endif

        ......
}
```

在G2 LSP中mdesc->smp_init没有定义。

```
DT_MACHINE_START(PEGMATITE_DT, "Marvell Pegmatite (Device Tree)")
        .dt_compat      = pegmatite_dt_compat,
#ifdef CONFIG_ZONE_DMA
        .dma_zone_size  = SZ_256M,
#endif
MACHINE_END
```

CONFIG_ARM_PSCI is not set ==> static inline bool psci_smp_available(void) { return false; }

==> 所以运行的是

smp_set_ops(mdesc->smp);

---

根据dts中cpus device_node中的setting来初始化smp variables

in arch/arm/kernel/setup.c

setup_arch() --> arm_dt_init_cpu_maps()

```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;      ①

        u32 tmp_map[NR_CPUS] = { [0 ... NR_CPUS-1] = MPIDR_INVALID };           ②
        bool bootcpu_valid = false;
        cpus = of_find_node_by_path("/cpus");
                                    ③

        if (!cpus)
                return;

        for_each_child_of_node(cpus, cpu) {
                                        ④
                u32 hwid;

                if (of_node_cmp(cpu->type, "cpu"))
                                        ⑤
                        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)) {
                        ⑥
                        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].
                 */
```

```
50.                 if (hwid & ~MPIDR_HWID_BITMASK)
51.                         return;
52.
53.                 /*
54.                  * Duplicate MPIDRs are a recipe for disaster.
55.                  * Scan all initialized entries and check for
56.                  * duplicates. If any is found just bail out.
57.                  * temp values were initialized to UINT_MAX
58.                  * to avoid matching valid MPIDR[23:0] values.
59.                  */
60.                 for (j = 0; j < cpuidx; j++)
                                        ⑦
61.                         if (WARN(tmp_map[j] == hwid, "Duplicate /cpu reg "
62.                                                      "properties in the DT\n"))
63.                                 return;
64.
65.                 /*
66.                  * Build a stashed array of MPIDR values. Numbering scheme
67.                  * requires that if detected the boot CPU must be assigned
68.                  * logical id 0. Other CPUs get sequential indexes starting
69.                  * from 1. If a CPU node with a reg property matching the
70.                  * boot CPU MPIDR is detected, this is recorded so that the
71.                  * logical map built from DT is validated and can be used
72.                  * to override the map created in smp_setup_processor_id().
73.                  */
74.                 if (hwid == mpidr) {
                                              ⑧
75.                         i = 0;
76.                         bootcpu_valid = true;
77.                 } else {
78.                         i = cpuidx++;
79.                 }
80.
81.                 if (WARN(cpuidx > nr_cpu_ids, "DT /cpu %u nodes greater than "
82.                                               "max cores %u, capping them\n",
83.                                               cpuidx, nr_cpu_ids)) {
84.                         cpuidx = nr_cpu_ids;
85.                         break;
86.                 }
87.
88.                 tmp_map[i] = hwid;
89.
90.                 if (!found_method)
                                              ⑨
91.                         found_method = set_smp_ops_by_method(cpu);
92.         }
93.
94.         /*
95.          * Fallback to an enable-method in the cpus node if nothing found in
96.          * a cpu node.
97.          */
98.         if (!found_method)
99.                 set_smp_ops_by_method(cpus);
100.
```

```
101.          if (!bootcpu_valid) {
102.                  pr_warn("DT missing boot CPU MPIDR[23:0], fall back to default cp
      u_logical_map\n");
103.                  return;
104.          }
105.
106.          /*
107.           * Since the boot CPU node contains proper data, and all nodes have
108.           * a reg property, the DT CPU list can be considered valid and the
109.           * logical map created in smp_setup_processor_id() can be overridden
110.           */
111.          for (i = 0; i < cpuidx; i++) {
                                      ⑩
112.                  set_cpu_possible(i, true);
113.                  cpu_logical_map(i) = tmp_map[i];
114.                  pr_debug("cpu logical map 0x%x\n", cpu_logical_map(i));
115.          }
116.  }
```

①

read cp15's c0's Multiprocessor Affinity Register

```
1.  #define read_cpuid(reg)                                 \
2.          ({                                                                  \
3.                  unsigned int __val;                             \
4.                  asm("mrc        p15, 0, %0, c0, c0, " __stringify(reg)  \
5.                      : "=r" (__val)                              \
6.                      :                                                       \
7.                      : "cc");                                                \
8.                  __val;                                                      \
9.          })
10.
11. static inline unsigned int __attribute_const__ read_cpuid_mpidr(void)
12. {
13.         return read_cpuid(CPUID_MPIDR);
14. }
```

==>

mrc p15, 0, Rx, c0, c0, 5

mpidr = Rx

mpidr的byte 0 = Affinity Level 0, byte 1 = Affinity Level 1, byte 2 = Affinity Level 2, byte 3必须为0

所以这里的mask MPIDR_HWID_BITMASK = 0x00FFFFFF

在gemstone2上 mpidr = 0x80ffff00

bit 31 = 1,Indicates that the processor implements the Multiprocessing Extensions register format.

②

tmp_map[NR_CPUS] = tmp_map[2] = MPIDR_INVALID = 0xFF000000

③

get cpus device_node as follow

```
        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";
                };
        };
```

④

比如gemstone2, cpus device_node有2个childs。对child进行enumeration

⑤

对child的合法性检查，child必须有

device_type = "cpu";

这个property。

⑥

read "reg" property from child, hwid = 0xffff00 或0xffff01

⑦

这里是检查各个child中reg property的值不能有重复的。

⑧

core cpu@0 (reg = <0xffff00>)是boot core,也就是当前运行的core,core cpu@1还没有enable呢

⑨

对core cpu@1而言

```
extern struct of_cpu_method __cpu_method_of_table[];

static const struct of_cpu_method __cpu_method_of_table_sentinel
        __used __section(__cpu_method_of_table_end);

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;
}
```

查询__cpu_method_of_table[].

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", &pegmatite_sm
p_ops);
```

set_smp_ops_by_method()中的

smp_set_ops(m->ops);

m->ops = pegmatite_smp_ops

==>

in arch/arm/kernel/smp.c

```
1.  static struct smp_operations smp_ops;
2.
3.  void __init smp_set_ops(struct smp_operations *ops)
4.  {
5.          if (ops)
6.                  smp_ops = *ops;
7.  };
```

使得global variable smp_ops = pegmatite_smp_ops

初始化smp_ops，在smp initialization II会用到它。

⑩

对gemstone2而言，cpu mask的bit 0, bit 1置位，表示有2 cores.

cpu_logical_map(0) = 0xffff00

cpu_logical_map(1) = 0xffff01