Experiences with device tree support development for ARM based SoC's

Thomas Abraham <thomas.abraham@linaro.org>

Linaro Kernel Working Group

February 15th, 2012

SAMSUNG

Linaro

Why device tree on ARM?

- ARM platforms rely on static list of platform devices for all non-discoverable devices
  - Too many board files
- Device tree is a simple tree like data structure that can describe a non-discoverable hardware configuration to the kernel
  - Platform devices are created at run-time by the kernel by parsing the device tree nodes
  - Device nodes can carry configuration / platform data for the devices
  - Allows kernel code and platform data to be decoupled

Benefits of Device Tree for ARM platforms

- Decouples kernel code and SoC data
  - Step towards realization of single kernel binary images for ARM based platforms
- Easier to add support for newer platforms
- Reduces amount of board specific code
  - Platform device and platform data are not statically defined
  - Usually ends up with one board file per SoC.
- Faster board ports
  - For new board support for dt-enabled SoC, write a dts file with minimal board specific fix-ups

Current Status of Device Tree Support for ARM Platforms

- Core device tree support for ARM upstreamed by Grant Likely starting from Linux 3.0
  - Platform matching and selection
  - Runtime Device creation (platform and AMBA)
- Device tree support completed for PL310 (L2CC), PL330 (DMAC), PL390 (GIC), PL192 (VIC) ARM peripherals
- DT support for multiple ARM based SoC's
  - SoC specific drivers modified to include DT support

Typical Sequence of adding device tree support

- Add device tree support to board files
  - Start with a new dt-enabled board file
  - Enable DT support in existing board files
- Create a SoC specific and board specific device tree source files (dtsi and dts)
- Enable DT support for system peripherals
  - Interrupt Controller, GPIO, DMA
- Enable DT support for peripheral drivers
  - UART, I2C, SD/MMC, SPI, etc.

Minimal Device Tree Enabled board file

```
static char const *exynos4210_dt_compat[] __initdata = {
        "samsung,exynos4210",
        NULL
};

DT_MACHINE_START(EXYNOS4210_DT, "Samsung Exynos4 (Flattened Device Tree)")
        .init_irq       = exynos4_init_irq,
        .map_io         = exynos4210_dt_map_io,
        .handle_irq     = gic_handle_irq,
        .init_machine   = exynos4210_dt_machine_init,
        .timer          = &exynos4_timer,
        .dt_compat      = exynos4210_dt_compat,
        .restart        = exynos4_restart,
MACHINE_END
```

Linaro

Minimal Device Tree Source file

```
/dts-v1/;
/ {
        model = "Insignal Origen evaluation board based on Exynos4210";
        compatible = "insignal,origen", "samsung,exynos4210";

        memory {
                reg = <0x40000000 0x40000000>;
        };

        chosen {
                bootargs ="root=/dev/ram0 rw ramdisk=8192 console=ttySAC2,115200";
        };
};
```

Note: dtsi and dts files are located at arch/arm/boot/dts

Compiling and Testing - 1

- Enable Device Tree Support
  - menuconfig → boot options → flattened device tree
  - Or use 'select USE_OF' in Kconfig entry of the device tree enabled board file
- Build the kernel image
  - make <defconfig>
  - make menuconfig
  - make uImage
  - Builds the dtc compiler as well
    - scripts/dtc
- Build the device tree blob
  - make <dts filename>

Compiling and Testing - 2

- Two options for passing dtb blob to kernel
  - Use bootm command of u-boot
  - Append dtb blob to the kernel image
- Option 1: Using the bootm command
  - Build u-boot with CONFIG_OF_LIBFDT enabled
  - bootm <kernel base> <initrd base> <dtb base>
    - Example: bootm 40007000 - 40004000
- Option 2: Appending dtb blob to kernel
  - menuconfig → boot options
    - select “Use appended device tree blob”
  - Used with legacy u-boot

Compiling and Testing - 3

```
Uncompressing Linux... done, booting the kernel.
Booting Linux on physical CPU 0
Linux version 3.3.0-rc1-00045-g5ab5d35 (thomas@Thomas) (gcc version 4.4.1 (Sourcery
CPU: ARMv7 Processor [412fc091] revision 1 (ARMv7), cr=10c5387d
CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine: Samsung Exynos4 (Flattened Device Tree), model: Insignal Origen evaluation
Ignoring RAM at 80000000-8fffffff (vmalloc region overlap).
Ignoring RAM at 90000000-9fffffff (vmalloc region overlap).
Memory policy: ECC disabled, Data cache writealloc
CPU EXYNOS4210 (id 0x43210010)
S3C24XX Clocks, Copyright 2004 Simtec Electronics
s3c_register_clksrc: clock armclk has no registers set
EXYNOS4: PLL settings, A=1000000000, M=800000000, E=96000000 V=108000000
EXYNOS4: ARMCLK=1000000000, DMC=400000000, ACLK200=200000000
ACLK100=100000000, ACLK160=160000000, ACLK133=133333333
```

Linaro

Instantiating platform devices from device tree - 1

- Non-DT platforms relied on a static list of platform devices for all non-discoverable devices
- For DT platforms, infrastructure exists to create platform devices at runtime from device tree
  - of_platform_populate() call walks through the nodes in device tree and creates platform devices from it
    - Call of_platform_populate during machine_init
  - Nodes should have a compatible property
  - For creating platform devices for sub-nodes, provide a list of all root nodes (second parameter)

Instantiating platform devices from device tree - 2

```
static void __init exynos4210_dt_machine_init(void)
{
        of_platform_populate(NULL, of_default_bus_match_table,
                                exynos4210_auxdata_lookup, NULL);
}
```

```
watchdog@10060000 {
        compatible = "samsung,s3c2410-wdt";
        reg = <0x10060000 0x100>;
        interrupts = <0 43 0>;
};
```

```
struct platform_device dt_watchdog = {
        ...
        ...
};
```

```
rtc@10070000 {
        compatible = "samsung,s3c6410-rtc";
        reg = <0x10070000 0x100>;
        interrupts = <0 44 0>, <0 45 0>;
};
```

```
struct platform_device dt_rtc = {
        ...
        ...
};
```

```
keypad@100A0000 {
        compatible = "samsung,s5pv210-keypad";
        reg = <0x100A0000 0x100>;
        interrupts = <0 109 0>;
};
```

```
struct platform_device dt_keypad = {
        ...
        ...
};
```

How to add minimal DT support for Device Drivers - 1

```
/dts-v1/;
/include/ "exynos4210.dtsi"

/ {
        model = "Insignal Origen evaluation board based on Exynos4210";
        compatible = "insignal,origen", "samsung,exynos4210";

        memory {
                reg = <0x40000000 0x40000000>;
        };

        chosen {
                bootargs ="root=/dev/ram0 rw ramdisk=8192 console=ttySAC2,115200";
        };

        watchdog@10060000 {
                compatible = "samsung,s3c2410-wdt";
                reg = <0x10060000 0x100>;
                interrupts = <0 43 0>;
        };
};
```

How to add minimal DT support for Device Drivers - 2

```
#ifdef CONFIG_OF
static const struct of_device_id s3c2410_wdt_match[] = {
        { .compatible = "samsung,s3c2410-wdt" },
        {},
};
MODULE_DEVICE_TABLE(of, s3c2410_wdt_match);
#endif

static struct platform_driver s3c2410wdt_driver = {
        .driver         = {
                .owner  = THIS_MODULE,
                .name   = "s3c2410-wdt",
                .of_match_table = of_match_ptr(s3c2410_wdt_match),
        },
        .probe          = s3c2410wdt_probe,
        .remove         = __devexit_p(s3c2410wdt_remove),
};
```

```
watchdog@10060000 {
        compatible = "samsung,s3c2410-wdt";
        reg = <0x10060000 0x100>;
        interrupts = <0 43 0>;
};
```

Linaro

Retrieving driver configuration data from device node - 1

- Design the device tree node for the device that the driver will instantiate
  - Compatible string
  - Register base and memory region length
  - IRQ numbers, if any
  - Bindings for supplying platform / configuration data to the driver
  - List of gpios if any
  - Document the bindings in Documentation/devicetree/bindings/

Retrieving driver configuration data from device node - 2

- Modify the driver to obtain the data from the device node.
  - Maintain a local copy of the platform data instead of referencing pdev->dev.pdata for pdata values
  - Add a runtime check to determine if a device node is available.
    - If node is available, parse all properties which the driver requires and populate the copy of local platform data
  - Avoid parsing device node for properties after probe
    - Keep a copy of the property value in private data

Retrieving driver configuration data from device node - 3

- Non-DT ARM platforms will continue to exist in few more kernel releases.
  - Hence all DT support related additions should maintain compatibility to non-DT platforms.
- Runtime determination of availability of a device tree node can be determined by checking of_node pointer

```
if (pdev->dev->of_node) {
        /* DT based instantiation */
} else {
        /* Non-DT based instantiation */
}
```

Retrieving driver configuration data from device node - 4

```
i2c@13860000 {
        compatible = "samsung,s3c2440-i2c";
        reg = <0x13860000 0x100>;
        interrupts = <0 58 0>;
        samsung,i2c-sda-delay = <100>;
        samsung,i2c-max-bus-freq = <20000>;
        gpios = <&gpd1 0 2 3 0>,
                <&gpd1 1 2 3 0>;
};
```

```
static void
s3c24xx_i2c_parse_dt(struct device_node *np, struct s3c24xx_i2c *i2c)
{
        struct s3c2410_platform_i2c *pdata = i2c->pdata;

        if (!np)
                return;

        pdata->bus_num = -1; /* i2c bus number is dynamically assigned */
        of_property_read_u32(np, "samsung,i2c-sda-delay", &pdata->sda_delay);
        of_property_read_u32(np, "samsung,i2c-slave-addr", &pdata->slave_addr);
        of_property_read_u32(np, "samsung,i2c-max-bus-freq",
                             (u32 *)&pdata->frequency);
}
```

Linaro

Setting up device names and platform data - 1

- Platform devices instantiated from device tree are not assigned a device name
  - Driver's looking up clocks would need device names
- Device names can be assigned by
  - Preparing a 'struct of_dev_auxdata' lookup table
  - Passing that table to of_platform_populate()
- Use the same 'struct of_dev_auxdata' lookup table to supply platform data, if required
- Note: 'struct of_dev_auxdata' lookup table is a temporary solution

Setting up device names and platform data - 2

```
serial@13800000 {
        compatible = "samsung,exynos4210-uart";
        reg = <0x13800000 0x100>;
        interrupts = <0 52 0>;
};

serial@13810000 {
        compatible = "samsung,exynos4210-uart";
        reg = <0x13810000 0x100>;
        interrupts = <0 53 0>;
};
```

```
static const struct of_dev_auxdata exynos4210_auxdata_lookup[] __initconst = {
        OF_DEV_AUXDATA("samsung,exynos4210-uart", 0x13800000, "exynos4210-uart.0", NULL),
        OF_DEV_AUXDATA("samsung,exynos4210-uart", 0x13810000, "exynos4210-uart.1", NULL),
        {},
};


static void __init exynos4210_dt_machine_init(void)
{
        of_platform_populate(NULL, of_default_bus_match_table,
                             exynos4210_auxdata_lookup, NULL);
}
```

Callback functions in platform data of a driver

- Determine if the callback functions can be dropped from platform data
  - Implement callback functions in the driver
  - Redesign the driver with no dependency on callbacks
- In inevitable case, use the 'struct of_dev_auxdata' to pass the callback function pointers
  - Populate only the callback function pointers in pdata
  - Driver parses other pdata elements from DT
  - But, this is just a temporary workaround since auxdata would be dropped eventually

Guidelines for designing bindings

- Should be OS agnostic
  - Bindings should be reusable across all operating systems (and u-boot as well)
  - If linux specific behavior needs encoding, use the 'linux' prefix for the binding.
- Should be generic information which the driver can decode and program the hardware or setup the operating system
- Should not be used to hard-code register values
  - Acceptable in some cases for one-time writes
- Should not be a used to encode read/modify/write cycles with information on delays.

Conclusion

- Device tree for ARM helps to
  - Reduce bloat in arm-linux
  - Reduce the churn in each kernel release
- Use DT for all new SoC platform and board code intended to be merged in linux mainline

www.linaro.org