Topics

- Space constraints in SPL
- fdtgrep
- dtoc
- Perfect world

Device Tree in U-Boot SPL

Dealing with space-constraints

Simon Glass
sjg@chromium.org
26 October 2017

Introduction

- SPL is Secondary Program Loader
  - Small program which loads U-Boot 'proper'
- U-Boot uses device tree with driver model (since 2014.04)
- Many benefits to using it in SPL also
  - But there are challenges in making it fit
  - A rough approximation is that we have 32KB SRAM
- Several techniques are used
  - fdtgrep - drops nodes / properties that are not needed
  - dtoc - convert DT to C

fdtgrep

- fdtgrep is a tool for grepping binary .dtb files
  - Sent upstream but rejected so far
- U-Boot uses it to:
  - Remove all nodes not tagged with 'u-boot,dm-spl'
  - Including only matching aliases
  - Drop some properties
  - Compact the string table afterwards

- ```
  fdtgrep --include-node-with-prop u-boot,dm-spl
       --include-root --show-aliases --include-node /chosen -O dtb |
  fdtgrep -r -O dtb - --out $@ --exclude-prop interrupts --exclude-prop dma-names ...
  ```
  ```

Results with fdtgrep

- Typically reduces the .dtb from 40KB to 3KB
- With RK3288 (Thumb2) we have ~4KB of code
- So a total of 7KB of overhead
  - libfdt ~4KB
  - .dtb ~3KB
- This is generally acceptable with 32KB SRAM
  - Firefly-RK3288 works OK with this
  - But MMC stack can put pressure on this - total size about 30KB

dtoc

- Emits C code from a .dtb file:
  - C structures in a header file
  - C data in a C file
- Instantiates U-Boot devices
- Does not require libfdt at run-time
- Does require special code for each C structure in each driver
- Results in close to zero overhead to use device tree in SPL
- Structs contain a superset of all compatible nodes
-
- ```
  dtoc -d xxx.dtb -o $@ platdata
  dtoc -d xxx.dtb -o $@ struct
  ```
  ```

```
    sdmmc: dwmmc@ff0c0000 {
        compatible = "rockchip,rk3288-dw-mshc";
        max-frequency = <150000000>;
        clocks = <&cru HCLK_SDMMC>, <&cru SCLK_SDMMC>,
              <&cru SCLK_SDMMC_DRV>, <&cru SCLK_SDMMC_SAMPLE>;
        clock-names = "biu", "ciu", "ciu_drv", "ciu_sample";
        fifo-depth = <0x100>;
        interrupts = <GIC_SPI 32 IRQ_TYPE_LEVEL_HIGH>;
        reg = <0xff0c0000 0x4000>;
        status = "disabled";
    };
&sdmmc {
    bus-width = <4>;
    cap-mmc-highspeed;
    cap-sd-highspeed;
    card-detect-delay = <200>;
    disable-wp;
    num-slots = <1>;
    pinctrl-names = "default";
    pinctrl-0 = <&sdmmc_clk>, <&sdmmc_cmd>, <&sdmmc_cd>, <&sdmmc_bus4>;
    vmmc-supply = <&vcc_sd>;
    status = "okay";
```

```
static struct dtd_rockchip_rk3288_cru dtv_clock_controller_at_ff760000 = {
        .rockchip_grf           = 0x3e,
        .reg                    = {0xff760000, 0x1000},
};

static struct dtd_rockchip_rk3288_dw_mshc dtv_dwmmc_at_ff0c0000 = {
        .fifo_depth             = 0x100,
        .cap_sd_highspeed       = true,
        .sd_uhs_sdr25           = true,
        .sd_uhs_sdr104          = true,
        .vqmmc_supply           = 0x9,
        .max_frequency          = 0x8f0d180,
        .card_detect_delay      = 0xc8,
        .cd_gpios               = {0x7, 0x5, 0x1},
        .reg                    = {0xff0c0000, 0x4000},
        .num_slots              = 0x1,
        .vmmc_supply            = 0x8,
        .clocks                 = {
                        {&dtv_clock_controller_at_ff760000, {456}},
                        {&dtv_clock_controller_at_ff760000, {68}},
                        {&dtv_clock_controller_at_ff760000, {114}},
                        {&dtv_clock_controller_at_ff760000, {118}},},
        .cap_mmc_highspeed      = true,
        .disable_wp             = true,
        .bus_width              = 0x4,
        .sd_uhs_sdr12           = true,
        .interrupts             = {0x0, 0x20, 0x4},
        .sd_uhs_sdr50           = true,
};
```

dt-platdata.c

```
struct dtd_rockchip_rk3288_cru {
        fdt32_t         reg[2];
        fdt32_t         rockchip_grf;
};

struct dtd_rockchip_rk3288_dw_mshc {
        fdt32_t         bus_width;
        bool            cap_mmc_highspeed;
        bool            cap_sd_highspeed;
        fdt32_t         card_detect_delay;
        fdt32_t         cd_gpios[3];
        struct phandle_1_arg clocks[4];
        bool            disable_wp;
        fdt32_t         fifo_depth;
        fdt32_t         interrupts[3];
        fdt32_t         max_frequency;
        fdt32_t         num_slots;
        fdt32_t         reg[2];
        bool            sd_uhs_sdr104;
        bool            sd_uhs_sdr12;
        bool            sd_uhs_sdr25;
        bool            sd_uhs_sdr50;
        fdt32_t         vmmc_supply;
        fdt32_t         vqmmc_supply;
};
#define dtd_rockchip_rk3066_spi dtd_rockchip_rk3288_spi
```

```
struct rockchip_mmc_plat {
#if CONFIG_IS_ENABLED(OF_PLATDATA)
     struct dtd_rockchip_rk3288_dw_mshc dtplat;
#endif
     struct mmc_config cfg;
     struct mmc mmc;
};
…
static int rockchip_dwmmc_probe(struct udevice *dev)
{
#if CONFIG_IS_ENABLED(OF_PLATDATA)
     struct dtd_rockchip_rk3288_dw_mshc *dtplat = &plat->dtplat;

     host->name = dev->name;
     host->ioaddr = map_sysmem(dtplat->reg[0], dtplat->reg[1]);
     host->buswidth = dtplat->bus_width;
     host->dev_index = 0;
     priv->fifo_depth = dtplat->fifo_depth;
     priv->minmax[1] = dtplat->max_frequency;

     ret = clk_get_by_index_platdata(dev, 0, dtplat->clocks, &priv->clk);
     if (ret < 0)
          return ret;
```

Wish list

- Easily drop unwanted stuff at build-time
- Smaller binary format
  - Shared 'property value' table, similar to the string table?
  - Encode a type (int/string), property name, shared value in a single cell?
- Easy conversion to C
  - Without needing any metadata parsing, etc.

```
NodeDesc('models', True, [
    NodeModel([
        PropPhandleTarget(),
        PropPhandle('whitelabel', '/chromeos/models/MODEL', False),
        NodeDesc('firmware', False, [
            PropPhandle('shares', '/chromeos/family/firmware/MODEL',
                        False, {'../whitelabel': False}),
            PropString('key-id', False, '[A-Z][A-Z0-9]+'),
            copy.deepcopy(BUILD_TARGETS_SCHEMA)
            ] + copy.deepcopy(BASE_FIRMWARE_SCHEMA)),
        PropString('brand-code', False, '[A-Z]{4}'),
        PropString('powerd-prefs', conditional_props=NOT_WL),
        PropString('wallpaper', False, '[a-z_]+'),
        NodeDesc('audio', False, [
            NodeAny(r'main', [
                PropPhandle('audio-type', '/chromeos/family/audio/ANY',
                            False),
                PropString('cras-config-dir', True, r'\w+'),
                PropString('ucm-suffix', True, r'\w+'),
                PropString('topology-name', False, r'\w+'),
```