One image to rule them all - Single boot image for SBCs

André Przywara
03/02/2019
apritzel@{Freenode,Github}
Agenda

- SBC boot situation
- Single image theory
- Status

Demo?
Agenda

- SBC boot situation
- Single image theory
- Status

Demo?
Disclaimer: Not an Arm Ltd. story.
**Scope**

- **SBC:** single board computer with ARM core, ”Fruit-Pis”
  - Not servers!
- **SoCs from Allwinner, Rockchip, Amlogic, ...**
  - Others possible, but require good upstream support
- **Storage-less boards**
  - which requires firmware on SD cards
- **Firmware:** board-specific low-level software, including boot loader
  - Ideally mainline, not BSP based.
Current situation
Current situation

ROCK64

Under 'ROCK64 Software and OS Image Download Section' you will find a complete list of currently supported Operating System images that work with the ROCK64 as well as other related software. The list includes OS images and descriptions of:

- Debian (microSD Boot)
- Xenial Mate (eMMC)
- Xenial Mate (microSD Boot)
- Artful Minimal Image (microSD Boot)
- Xenial Minimal Image (microSD Boot)
- Android 7.x (eMMC)
- Android 7.x (microSD Boot)
- Android TV 7.x (eMMC)
- Android TV 7.x (microSD Boot)
Problems

- Separate images for each board
  - Often containing very similar bits
  - Boards might be mistaken
  - Different image might *somehow* work

- Not every board covered

- Different quality for each board
Idea

- One firmware image to cover multiple boards
- Could be centrally maintained
- Could be shipped by distributions
- Relieves people from choosing their board beforehand
Steps toward the single image

- Get something booted
- Detect SoC
- Detect/choose board
- Load rest of firmware
- Hand over to shared UEFI / bootloader / kernel
Boot process
Typical SoC boot process

- Most SoCs contain embedded boot ROM
- Mask programmed in silicon
- Can’t be changed by the user (no updates!)
- But can typically be read (and disassembled)
- Typical size 32-64 KB
- Mission: Find the real boot source, load some boot code and execute
- Code is normally loaded into SRAM (could be L2/L3 cache)
- Boot order could be changeable (pins, fuses)
- Boot code size is typically quite limited (32KB)
Popular SoC boot ROM behaviour

- **Allwinner SoCs**: Load $\leq 32$KB from sector 16 of SD card
  - Tries eMMC, NAND, SPI NOR flash, USB OTG *afterwards*
  - Tries sector 256 on most SoCs as well
- **Rockchip SoCs**: Load from sector 64 of SD card
  - Tries eMMC, NAND, SPI NOR flash *first*, USB OTG later
  - Tries sectors 1088, 2112, 3136, 4160 as well
- **Amlogic SoCs**: Load from sector 1 of SD card
- **Raspberry Pi**: VideoCore loads from 1st FAT partition of SD card
  - Too easy!
<table>
<thead>
<tr>
<th>MBR</th>
<th>GPT</th>
<th>1st partition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>unused</td>
</tr>
</tbody>
</table>

### GPT layout

**1 MB**

- 17 K

### Allwinner layout

**1 MB**

<table>
<thead>
<tr>
<th>MBR</th>
<th>empty</th>
<th>U−Boot SPL</th>
<th>1st partition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>legacy U−Boot image or FIT image:</td>
<td>unused</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U−Boot proper + DTB ( + ATF)</td>
<td></td>
</tr>
</tbody>
</table>

**8 K**

- 40 K

### Amlogic layout

**1 MB**

<table>
<thead>
<tr>
<th>MBR</th>
<th>(Amlogic)</th>
<th>tweaked FIP image:</th>
<th>1st partition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BL1</td>
<td>U−Boot proper (w/ DTB) mgmt firmware, ATF</td>
<td>unused</td>
</tr>
</tbody>
</table>

**48 K**

- ~600 K

### Rockchip layout

**1 MB**

<table>
<thead>
<tr>
<th>MBR</th>
<th>(GPT)</th>
<th>empty</th>
<th>empty</th>
<th>1st partition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IDB / SPL</td>
<td></td>
<td></td>
<td>unused</td>
</tr>
</tbody>
</table>

**17 K**

- 32 K

- 256 K

---

FOSDEM 2019
Integration of several SPLs

- Observation: many clashes between primary boot locations
- Even more so for subsequent images (U-Boot proper, ...)
- But: secondary boot locations help to sort this out:
  - Allwinner: 8K or 128K
  - Rockchip: 32K, 544K, 1056K, ...
- Plan B: Carve out regions for other SoCs to allow overlapping
- Plan C: Have trampoline loaders
- Assumes location of secondary image(s) can be freely chosen
  - Can put in gaps into FIT or FIP image structure
Detect SoCs

- U-Boot typically compiled for one particular board
- Although actually: for one SoC (+ drivers + DTB)
- Many platforms hardcode the SPL
- Bad for single images :-(
- Solution: convert #ifdef into runtime decisions

Detect SoC
  - Use platform specific MMIO register (in fuses)
  - Use heuristic (probe ID registers, for instance in the GIC)

- Use toolchain garbage collection to keep code small
U-Boot today

#if defined(CONFIG_MACH_SUN5I) && CONFIG_CONS_INDEX == 1
    sunxi_gpio_set_cfgpin(SUNXI_GPB(19), SUN5I_GPB_UART0);
sunxi_gpio_set_cfgpin(SUNXI_GPB(20), SUN5I_GPB_UART0);
sunxi_gpio_set_pull(SUNXI_GPB(20), SUNXI_GPIO_PULL_UP);
#endif

#if defined(CONFIG_MACH_SUN6I) && CONFIG_CONS_INDEX == 1
    sunxi_gpio_set_cfgpin(SUNXI_GPH(20), SUN6I_GPH_UART0);
sunxi_gpio_set_cfgpin(SUNXI_GPH(21), SUN6I_GPH_UART0);
sunxi_gpio_set_pull(SUNXI_GPH(21), SUNXI_GPIO_PULL_UP);
#endif

#if defined(CONFIG_MACH_SUN8I_A33) && CONFIG_CONS_INDEX == 1
    ....
#endif
U-Boot SoC runtime

switch (sunxi_get_socid()) {
    case SOCID_A10:
    case SOCID_A20:
    case SOCID_R40:
        mux = SUN4I_GPB_UART0;
        tx_pin = SUNXI_GPB(22);
        rx_pin = SUNXI_GPB(23);
        break;
    case SOCID_A13: // sun5i
        mux = SUN5I_GPB_UART0;
        tx_pin = SUNXI_GPB(19);
        rx_pin = SUNXI_GPB(20);
        break;
    ...
}
DRAM initialisation

- Typically fixed parameters
- Try probing?
- Go with one-size-fits-all, safe values
- DRAM types (LPDDR3, DDR3): try one, if that fail, try other
Detect boards

- Reliable board auto detection is technically impossible
  - and dangerous!
  - But can be achieved for a subset of boards:
    - Uses heuristics: DRAM size, type, I2C/SPI devices, GPIO ...

- Solution: present a list and let the user choose
  - List could be shortened by matching heuristics
  - Each list entry selects one .dtb file
  - FIT image can already hold multiple .dtbs
Status

”Hello World” single image works on:
- All Allwinner SoCs
- Rockchip RK3328 and RK3399 SoCs
- Amlogic S905 (Odroid-C2)

U-Boot (and beyond) works on:
- Boards with Allwinner A64 and H5 SoCs
- Rock64 and Firefly RK3399 board
Open issues

- SPL needs to know load address at link time (breaks Allwinner H6)
  - Make SPL position independent
- No mainline SPL support for RK3328
  - (Old?) patches exist, somewhere
- No SPL support for Amlogic SoCs (signed boot code?)
  - Makes coexistence much more complicated
  - Could be tolerated for at most one SoC
Conclusion

- One image can boot multiple boards
- Proof of concept working, but lot of integration work left
- Upstreaming might get interesting ;-)
- Use cases:
  - Distribution installers
  - Firmware flashers
  - Multi-Distribution installers (NOOBS)
  - Your own (bare metal) application
Thank You!
References

- http://linux-sunxi.org/
- https://github.com/apritzel/pine64
- https://github.com/apritzel/simage
- Freenode: @apritzel