coreboot porting for ARM
Contents

● Current Status
● Difference between x86 & ARM
● Early Boot flow
● SRAM
● Device configuration
● Payloads / libpayload
Current Status

- **Supported ARM SOCs**
  - Samsung Exynos 5250
  - Samsung Exynos 5420
  - Nvidia Tegra 124
  - Texas Instruments AM335x
  - Allwinner A10

- **Emulators**
  - Qemu / ARMv7
  - ARM64 Foundation Model
Typical x86 system (“2010 PC”)
Typical (ChromeOS) ARM system

- SOC
- Memory Controller
- Graphics
- eMMC
- SPI
- SRAM
- COclocks
- SMBUS
- TPM
- RAM (soldered down)
- RAM (soldered down)
- boot rom (not memory mapped)
Difference between (typical) ARM and x86 Systems

● x86: Extensible
  ○ SPD for RAM
  ○ PCI devices
  ○ LPC
  ○ cache as ram
  ○ ACPI, SMM

● ARM: Simple Design, Low Power
  ○ In-SOC 1st level boot firmware
  ○ No memory mapped system firmware
  ○ All peripherals memory mapped or attached to SPI / SMBUS
  ○ SRAM
  ○ No ACPI (so far), no SMM
Early Boot Flow

**Bootblock**
- set up stack
- load romstage from cbfs to SRAM
- needs SPI/eMMC driver to access system firmware
- set up MMU for caching

**Romstage**
- set up PMIC, GPIOs, DRAM
- executed from SRAM
- load ramstage to DRAM, execute ramstage
SRAM (Example: Exynos 5420)

<table>
<thead>
<tr>
<th>iSRAM</th>
<th>Address Range</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>vendor-provided BL1</td>
<td>0x2020000 - 0x2024400 (17K)</td>
<td></td>
</tr>
<tr>
<td>variable length bootblock checksum header</td>
<td>0x2024400 - 0x2024410 (16B)</td>
<td></td>
</tr>
<tr>
<td>bootblock (32k max)</td>
<td>0x2024410 - 0x2030000 (~47K)</td>
<td></td>
</tr>
<tr>
<td>romstage (128k max)</td>
<td>0x2030000 - 0x2058000 (160K)</td>
<td></td>
</tr>
<tr>
<td>TTB buffer</td>
<td>0x2058000 - 0x205c000 (16K)</td>
<td></td>
</tr>
<tr>
<td>cache for CBFS data</td>
<td>0x205c000 - 0x206f000 (76K)</td>
<td></td>
</tr>
<tr>
<td>stack</td>
<td>0x206f000 - 0x2073000 (16K)</td>
<td></td>
</tr>
<tr>
<td>kernel shared page</td>
<td>0x2073000 - 0x2074000 (4K)</td>
<td></td>
</tr>
</tbody>
</table>
Device Configuration

- Some settings mainboard specific
- Chipset can export per-board settings using devicetree
- Not automatic: need chip.h structure and C code to honor setting
- SOC setup mostly static and automatic

```c
struct soc_samsung_exynos5420_config {
    int clkval_f;
    int upper_margin;
    int lower_margin;
    int vsync;
    int left_margin;
    int right_margin;
    int hsync;
    int xres;
    int yres;
    int framebuffer_bits_per_pixel;
    int usb_vbus_gpio;
    int usb_hsic_gpio;
};
```

```
chip soc/samsung/exynos5420
    device cpu_cluster 0 on end
    register "xres" = "1366"
    register "yres" = "768"
    register "framebuffer_per_pixel" = "16"
    register "clkval_f" = "2"
    register "upper_margin" = "14"
    register "lower_margin" = "3"
    register "vsync" = "5"
    register "left_margin" = "80"
    register "right_margin" = "48"
    register "hsync" = "32"
end
```
libpayload

- abstraction for payloads
- separate set of drivers per SOC: UART, storage, timer…

Payloads

- ChromeOS specific: depthcharge
- In development: FILO
- device tree handling
Questions/Comments?

firmware@chromium.org