coreboot porting for ARM

Contents

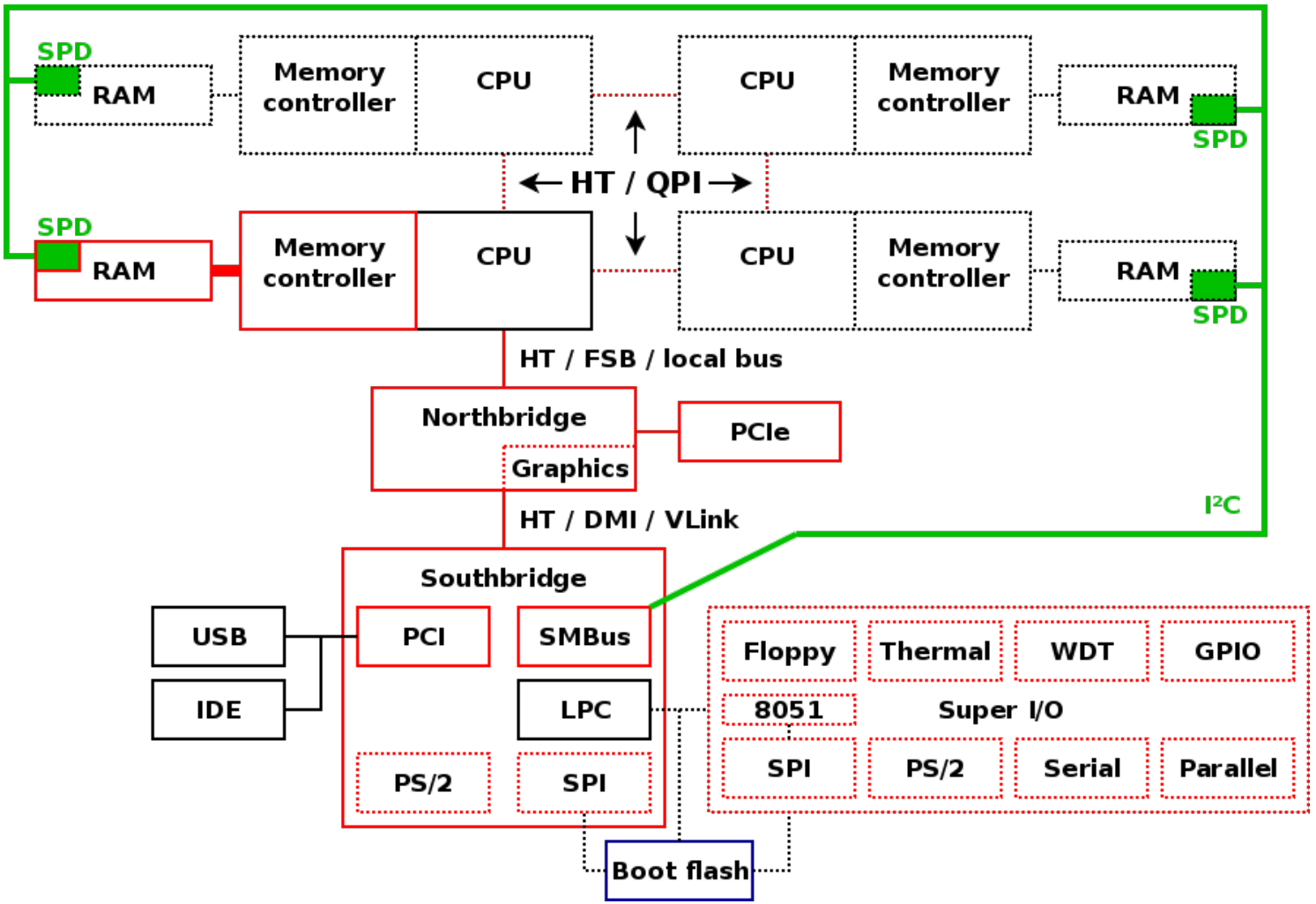
- Current Status
- Difference between x86 & ARM
- Early Boot flow
- SRAM
- Device configuration
- Payloads / libpayload

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

Google

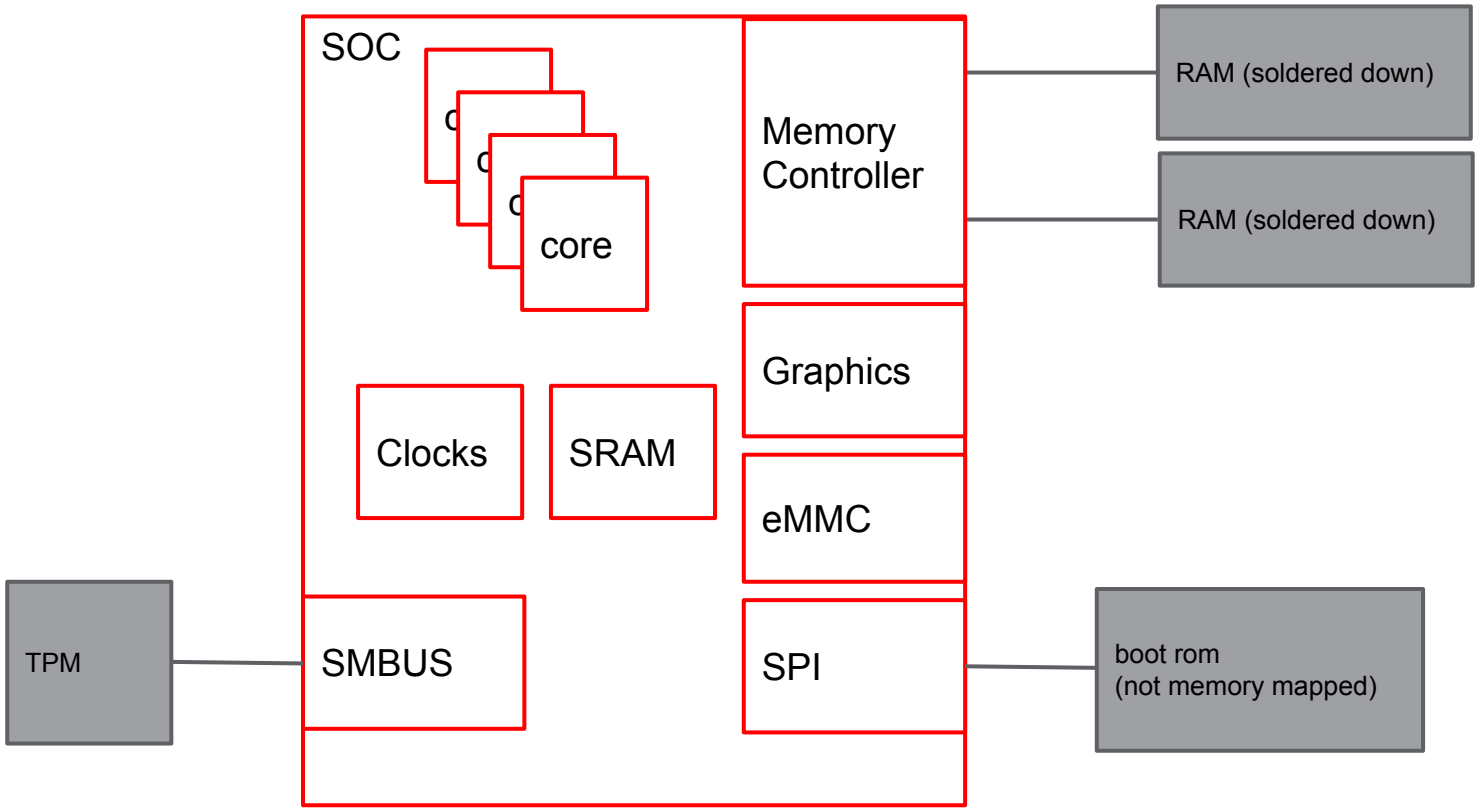
Typical x86 system (“2010 PC”)



* graphics by Peter Stuge

Google

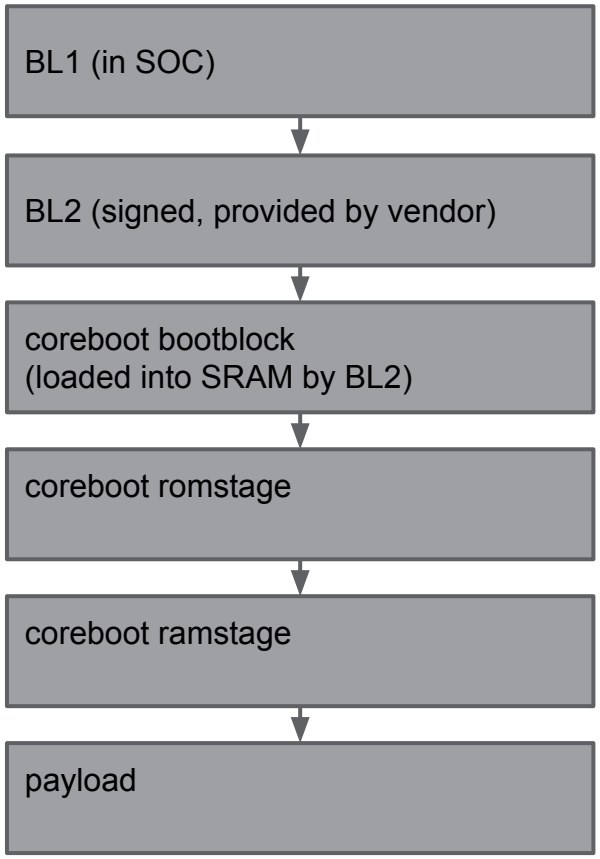
Typical (ChromeOS) ARM system



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

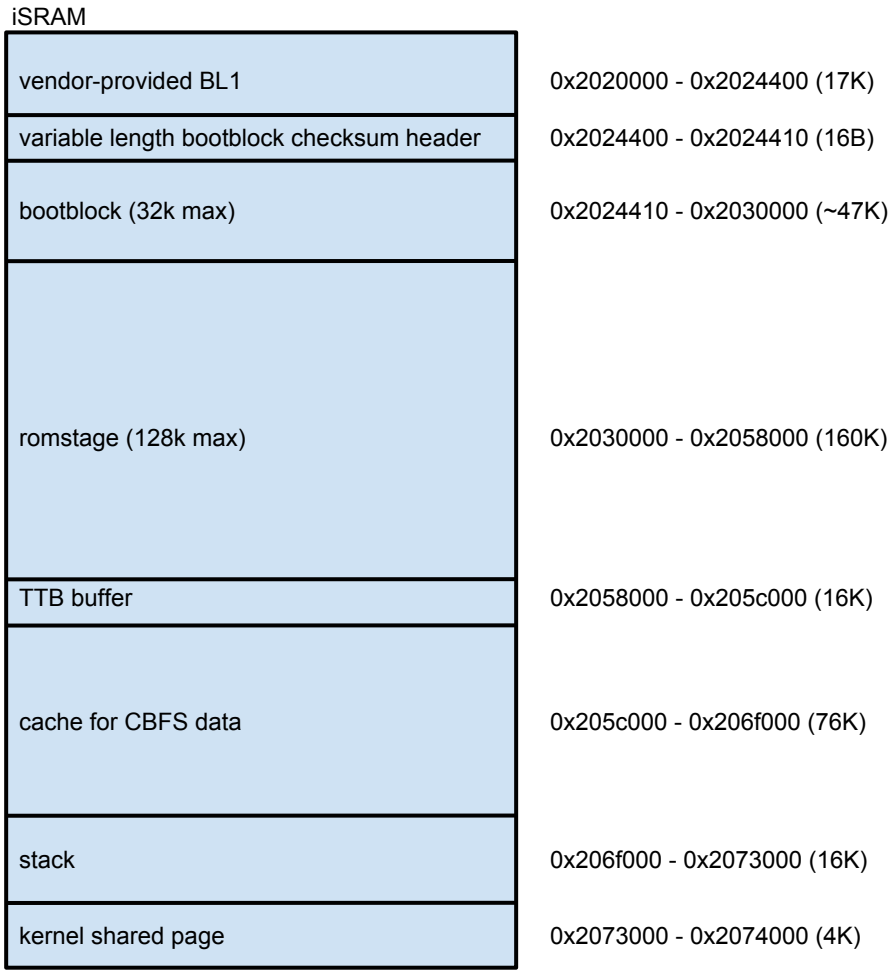
- set up stack
- load romstage from cbfs to SRAM
- needs SPI/eMMC driver to access system firmware
- set up MMU for caching

Romstage

- set up PMIC, GPIOs, DRAM
- executed from SRAM
- load ramstage to DRAM, execute ramstage

Google

SRAM (Example: Exynos 5420)



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

```
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
```

```
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;
 };
```

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