ACPI on ARM/ARM64

Graeme Gregory

18th September 2013

Why ACPI?

- Industry standard for x86
- OEM driven activity
- Already used in Windows RT
- ACPI specification is freely available.
  http://acpi.info/

but is ACPI?

- A horrible standard impossible to implement
- x86 only
- going to cause the end of the world

ARM/ARM64 Status

- Basic Platform support for arm/arm64
- Tables from FDT or UEFI
- Devices can be probed from DSDT
- CPU Topology/Hotplug Plumbing

Development

- armv7 Arndale, armv8 Foundation Model, armv8 RTSM VE Model
- OpenEmbedded and Ubuntu based rootfs
- ACPICA tools available in OpenEmbedded and Linaro PPA
- Kernel - https://git.linaro.org/gitweb?p=arm/acpi/acpi.git
- Example ASL - https://git.linaro.org/gitweb?p=arm/acpi/acpi-asl.git

Testing

- ACPI ASL test suite integrated into LAVA (Linaro Automated Validation Architecture) for arm/arm64
- ACPI API test suite integrated into LAVA for arm/arm64
- FWTS (FirmWare Test Suite) for arm/arm64.

ARM/ARM64 Issues

- How to handle the data currently supplied by FDT?
- Current resources specified in ACPI are insufficient for SoCs.
- CPU power states different from x86.
- No exact NMI equivalent for low latency error reporting.

FDT vs ACPI Resources

DTS

```
reg = <0x12C60000 0x100>;
interrupts = <0 56 0>;
gpios = <&gpb3 0 2 3 0>,
        <&gpb3 1 2 3 0>;
```

ASL

```
Method (_CRS, 0x0, Serialized) {
        Name (RBUF, ResourceTemplate ()
        {
                Memory32Fixed (ReadWrite, 0x12C60000, 0x00000100)
                Interrupt (ResourceConsumer, Level, ActiveLow, Exclusive, , , )
                                {0x58}
                GpioIo (Exclusive, PullDefault, , , , "\\_SB.GPB3")
                                {0x2A, 0x2B}
        })
        Return (RBUF)
}
```

FDT vs ACPI the Rest

DTS

```
samsung,i2c-slave-addr = <0x66>;
samsung,i2c-sda-delay = <100>;
samsung,i2c-max-bus-freq = <100000>;
```

SoCs Extra Resources?

- Clocks
- Voltage/Current Sources
- Other?

CPU Power States

- Power state definitions insufficient for ARM.
- Need both entry and exit latencies for C states
- Latencies vary depending on current P state

NMI equivalent

- Linux does not allow higher priority IRQ to pre-empt
- SEI mechanism may be used on arm64
- Will require firmware support in higher EL

Conclusion

- ACPI for ARM/ARM64 is under ongoing development
- There will need to be extensions to ACPI 5.0 specification
- https://wiki.linaro.org/LEG/Engineering/Kernel/ACPI/


Linaro

UEFI on ARM/ARM64

Leif Lindholm

Linux Plumbers Conference, 18th September 2013

Outline

Why?

- UEFI is becoming the norm for x86 firmware, and is the firmware solution preferred by ARM.
- Linaro's OEM and distribution members want to use the same install/deploy/upgrade mechanisms on arm/arm64 as on x86.
- UEFI provides a standardised way to deal with the greater platform divergence in the ARM hardware echosystem (no more flash-kernel).
- Windows requires UEFI on ARM.

Who?

- UEFI Forum
  - The UEFI specification is maintained/published by the UEFI Forum.
- Tianocore Project
  - The Tianocore project provides an opensource implementation of the UEFI specification, as well as related tools and utilities.
- ARM
  - ARM ltd. provides the fundamental architectural support for the ARM architectures.
- Linaro
  - Linaro maintains platforms trees, which you can build complete firmware images from, and does feature development towards x86 parity.

What?

So what has Linaro been working on so far?

- Platform tree + SCT
- UEFI kernel support
- Grub
- EFI Stub
- Network boot

How?

So what have we actually done?

Platform tree + SCT

We have a platforms tree, based on Tianocore with added support for generating a basic standalone image. This tree also holds features completed but not yet upstreamed. We make monthly(-ish) binary releases for a number of platforms.
git://git.linaro.org/arm/uefi/uefi.git
The Self-Certification Testsuite is available to UEFI Forum members, which includes Linaro. We run this on our platforms tree.

UEFI kernel support

Kernel needs to

- Define an interface to (from?) UEFI
  - preserve (some) UEFI memory regions
  - access EFI System Table
  - scan configuration tables
    - ACPI, SMBIOS, ...
- be able to call back into UEFI
  - SetVirtualAddressMap()
  - EFIVARS

UEFI kernel support

- Preserving memory regions requires early access to the UEFI memory map
  - We define that the loader must add the UEFI memory map to the FDT
- Some configuration tables contain information needed early in the boot process
  - x86 and ia64 contained separate code for doing this - a third separate implementation was not appreciated.
  - x86 and ia64 use early_ioremap(). ARM did not have early_ioremap().

UEFI kernel support

Runtime services

- Before runtime services can be called by the kernel, any preserved UEFI memory regions must be mapped into virtual address space and (linker) relocated via the SetVirtualAddressMap() call.
- This call must be made with a 1:1 physical:virtual address mapping.
  - After the MMU has been enabled.
- This call is only be possible to make once.

Grub

Grub, the project formerly known as Grub2. Port to 32-bit ARM on both U-Boot and UEFI platforms.

- Implement glue layer between firmware and Grub
- Implement dynamic relocations for loadable modules
- Implement a Linux loader
- Implement executable image generation, and adapt install scripts

EFI Stub

zImage as its own UEFI boot loader

- Making zImage a valid ARM PE/COFF executable
  - While still keeping it a valid zImage
- Embed a small loader application, conforming to the same prorocol as previous Grub port
- parts of existing x86 code made generic, and reused

Network boot

- Implement drivers and necessary bits for networking stack (parts done by ARM).
- Utilise UEFI network stack for Grub netbooting.

Next steps

- Making sure everything above gets upstreamed
- Repeating all of the above for ARM64
  - Except for U-Boot port of Grub
- Get UEFI running on QEMU
- Kexec
- Privilege compartmentalisation of UEFI
- Big-endian?

Resources

- https://wiki.linaro.org/LEG/Engineering/Kernel/UEFI/
- http://tianocore.sourceforge.net/
- http://www.uefi.org/


Linaro