ACPI and FreeBSD

Nate Lawson
nate@root.org

Bay Area FreeBSD Users' Group
May 3, 2006

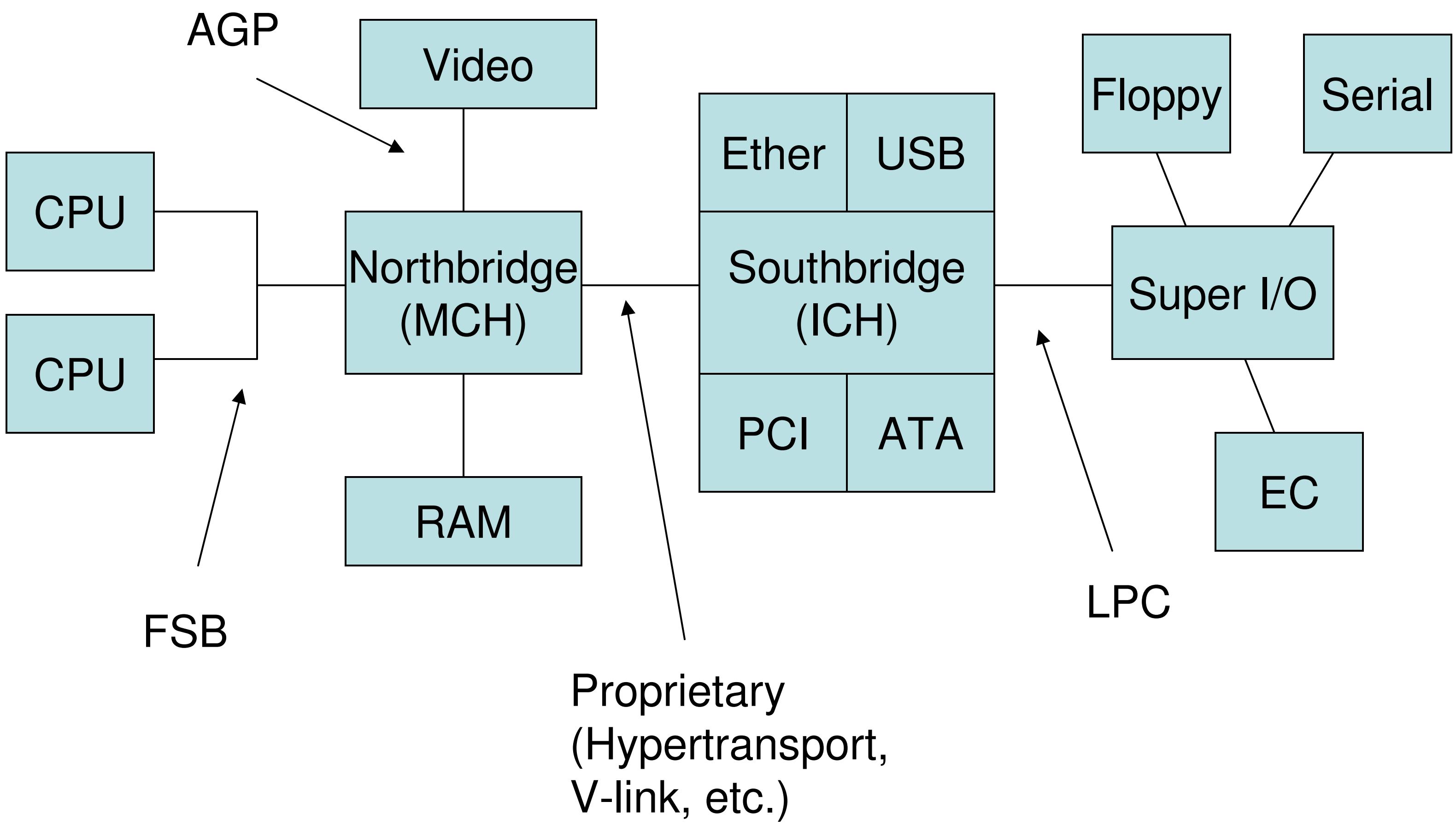
Overview

- Introduction
  - PC platform and architecture
  - ACPI
- FreeBSD ACPI support
- Challenges and issues
- How you can help

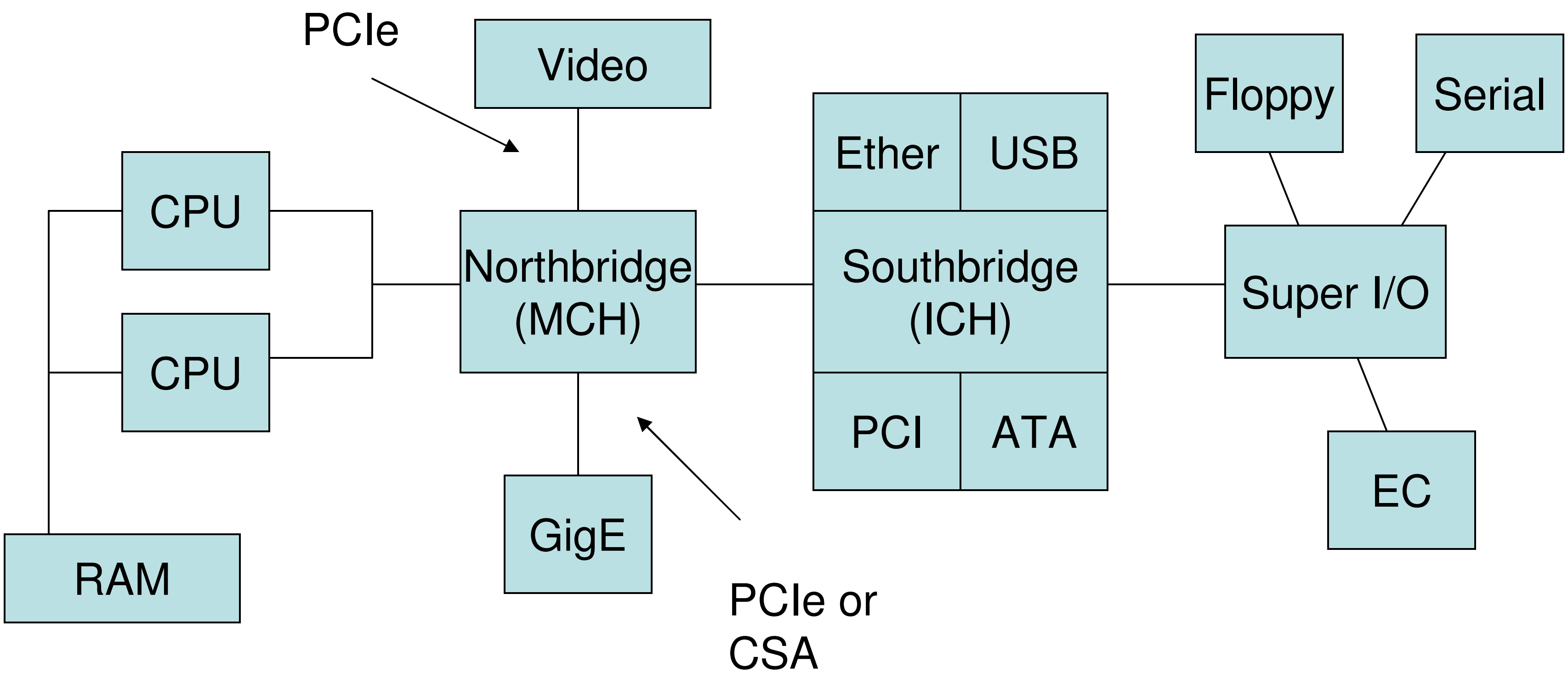
How I got here

- Background in security and cryptography
- Worked on storage security in my day job
  - Built a Fibre Channel encryptor
  - Built a parallel SCSI encryptor
  - Tired of starting anew each time, wrote and committed a SCSI target driver framework
- But my laptop wasn't working so well
- Began working on ACPI in my spare time
  - 4 years later, here I am

PC platform (classic)



PC platform (AMD/PCI Express)



Legacy boot process

- CPU RESET pin triggered, jumps to boot vector
  - Real mode, low memory, etc., just like old DOS days
- BIOS decompressed from flash, executed
  - Self-tests
  - Code copied into SMRAM and SMM enabled
  - Initializes built-in devices and cards in slots
  - Devices set to initial power states
  - Finds other CPUs
  - Sets up RAM tables for OS (e.g., MPtable)
- Loads boot sector and jumps to it

Power management

- Enumerate devices, including hotplug events
  - Location
  - Resources
  - State (on/off/missing)
- Suspend system
  - RAM
  - Disk
  - Power-off
- Power down/up devices based on system activity
  - CPU
  - Internal chipset devices
  - External devices on a bus
- Thermal management
  - Fans
  - Passive cooling

Legacy power management (APM)

- BIOS handles all PM, began with the 386SL
- System management interrupt (SMI) is regularly triggered by device activity
- BIOS code running from SMM performs power activity
  - Powers down idle devices
  - Implements suspend/resume
  - Controls device state
- Problems
  - No OS visibility of what BIOS is doing ("but I don't want it powered down now")
  - Duplicated effort in maintaining large, platform-specific codebase
  - Buggy, especially 32-bit entry points
  - PC-centric (i386 only)

ACPI

- OS and BIOS now share power management
  - OS: policy, drivers, and a few hooks
  - BIOS: delivers the SMI (now SCI) to the OS and provides tables that describe what the OS can do
- History
  - Appeared in 1998, not really implemented until 1999
  - Microsoft implementation significantly different from the standard before Windows XP (2001)
  - Spec is updated after major platforms ship with the new features
- Problems
  - Platform-specific ACPI devices (acpi_ibm, acpi_toshiba, …) create duplicated effort
  - Buggy, especially BIOS interface
  - PC-centric (i386, amd64, ia64)

ACPI operation

- BIOS creates tables on boot
  - Table of contents (RSDT/XSDT), pointed to by RSDP
  - DSDT: AML bytecode and device tree
  - MADT: APIC table for SMP and interrupt routing
  - FADT: fixed features, superceded by DSDT in many cases
- OS finds tables in memory and activates ACPI
  - Writes special value to SMM code which enables ACPI mode and the SCI in particular
  - SCI and SMI are shared, BIOS handles SMI transparently
- OS enumerates devices and config
  - Walks device tree from DSDT
  - Powers up any device the BIOS left off
  - Allocates resources and attaches drivers

AML operation

- DSDT consists of bytecode
- Bytecode describes regions (IO ports, memory-mapped devices), objects (containers), methods, and opcodes
- Example:

```
OperationRegion (\SCPP, SystemIO, 0xB2, 0x01)
Field (\SCPP, ByteAcc, NoLock, Preserve)
{
    SMIP,   8
}

Method (\_SB.PCI0._INI, 0, NotSerialized)
{
    If (STRC (\_OS, "Microsoft Windows")) {
        Store (0x56, SMIP)
}
```

- OS AML interpreter runs the requested method by interpreting the code and reading/writing to memory as it directs

ACPI operation (suspend)

- User presses “sleep” button
- Super I/O gets interrupt on GPIO pin
- EC function raises the SMI/SCI interrupt
- OS EC driver queries EC for event type (sleep pressed)
- OS delivers Notify event to the button driver
- Button driver calls OS-specific GoToSleep function
- OS walks device tree, saving state
- OS executes AML bytecode for requested sleep operation (say, _S3)
- Sequence of IO writes causes chipset to enter S3 (STR)

ACPI operation (resume)

- User presses “wake” button
- Super I/O gets interrupt on GPIO pin
- EC raises the SMI/SCI interrupt and signals chipset to wake
- BIOS resumes any devices it manages and jumps to OS wake vector
- OS walks device tree, restoring state
- OS executes AML bytecode for resume (_WAK)
- OS continues execution of processes

ACPI operation (probe)

- Device tree example:

```
Device (PCI0)              Internal PCI
        Device (USB0)      USB ports
        Device (USB1)
        Device (USBE)
        Device (ICHX)      ATA on-board
                Device (PRIM)
                    Device (MAST)
                    Device (SLAV)
                Device (SECN)
                    Device (MAST)
                    Device (SLAV)
        Device (IDE1)      ATA (dock)
                Device (PRIM)
                    Device (DRV0)
                    Device (DRV1)
                Device (SECD)
                    Device (DRV0)
                    Device (DRV1)

Continued →
```

```
Device (PWRB)    ACPI power button
Device (FAN)     ACPI fan
Device (PCI0)
        Device (LNKA)    PCI irq link
        Device (LNKB)
        ...
        Device (PX40)    Super I/O
                Device (SYSR)    IO port resources
                Device (PIC)     Legacy irq control
                Device (RTC)     Real-time clock
                Device (SPKR)    BIOS speaker
                Device (COPR)    FPU
                Device (FDC0)    Floppy
                Device (UAR1)    Serial 1
                Device (UAR2)    Serial 2
                Device (IRDA)    Infrared (serial)
                Device (LPT1)    Parallel
                Device (ECP1)    Parallel (ECP access)
                Device (PS2M)    PS/2 mouse
                Device (PS2K)    PS/2 keyboard
                Device (PSMR)
                Device (PMIO)
```

FreeBSD history

- 1999
  - First implemented by dfr@
- 2000 - 2001
  - Moved to Intel ACPI-CA interpreter
  - Battery, suspend/resume, and core driver brought in (msmith@, iwasaki@, takawata@)
- 2002 - 2003
  - New imports, EC updates, _PxD device power states
  - I stepped too close to the sucking vortex
- 2004
  - rman support
- 2005
  - cpufreq framework implemented
  - CPU-specific drivers for SpeedStep (new, ICH), Powernow, P4TCC, throttling
- 2006
  - acpi_dock (iwasaki@ returns!)

To be continued…