Erledigt El Capitan Clover Ruhezustand Shutdown Probleme

Beitrag von „clairon“ vom 7. Oktober 2016, 22:30

Was eine kühle CPU betrifft, bin ich grade von diesem System total begeistert. Mit ein paar kleinen Handgriffen bringt man seine CPU in wirklich kühle Regionen auch unter Vollast:

Ein einfacher Patch der **AppleLPC.kext** wäre: im Paketinhalt der Kext-datei die Info.plist mit z.B. PlistEditPro aufzurufen und dann unter **IONameMatch** den Eintrag **pci8086,3a16** hinzuzufügen. (3a16 steht für die Nummer aller CPU wie z.B. Bloomfield, Nehalem, Westmere u.s.w. in unseren Rechnern). Ein uneleganter Weg, da mit jedem Update und anderen Systemversionen immer wieder der Patch neu gemacht werden müsste.

Besser mit DSDT:
In meinen DSDT-Dateien ist unter „**Device LPCB**“, was wiederum im „**Device PCI0**“ liegt, eine DSM-Methode (direkt der erste Eintrag unter **LPCB**), die der **AppleLPC.kext** eine native CPU vermittelt. Apple erkennt nämlich nur die Version 0x18, 0x3A als nativ, wenn man aber in der DSDT-Datei den Hinweis zur Kompatibilität gibt, schluckt die AppleLPC.kext auch die 0x16, 0x3A als nativ und bedankt sich mit der vollen Unterstützung des Powermanagament der CPU ohne vorher direkt gepatcht werden zu müssen. Das ganze sieht dann in der DSDT so aus:

```
Device (LPCB)
{
Name (_ADR, 0x001F0000)
Method (_DSM, 4, NotSerialized)
{
Store (Package (0x08)
{
"device-id",
Buffer (0x04)
{
0x18, 0x3A, 0x00, 0x00
},
"compatible",
Buffer (0x0D)
{
```

```
"pci8086,3a18"
},
"IOName",
Buffer (0x0D)
{
"pci8086,3a18"
},
"name",
Buffer (0x0D)
{
"pci8086,3a18"
}
}, Local0)
DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
Return (Local0)
}
```

Das kann jeder mit einem 1366-Sockel-Board getrost in seine DSDT reinkopieren.
Mit weiteren sub-Informationen wäre ich vorsichtig, denn die können sich unterscheiden.
Bei dir liegt diese DSM-Methode am Ende von Device LPCB ohne „compatible“

Mit dieser Methode braucht man nicht einmal mehr eine Speedstep-Definition für die CPU um PM zu fahren.
Schöner ist ein Speedstep-Eintrag dennoch, da es beim booten vorkommen kann, dass nicht alles sauber eingelesen wird.
Daher sollte man hierzu noch die Eintäge für die CPU einfügen oder eine **SSDT.aml** anfertigen, die man zur **DSDT** dazulegt.

Hiefür gibt es zwei einfache Lösungen:
Man bootet mit einer DSDT-Datei **ohne Speedstep-Einträge** (mit meinen DSDT-ohne-Speedstep-Dateien wäre dies für unsere x58a-ud3r Boards möglich)
Auch sonst werden keine P-States u.a. generiert (wie mit Clover oder Chameleon möglich)
Dann nehme man **MaciASL**.app und läßt sich eine **SSDT** unter **Tools** generieren (**Generate SSDT**) und trägt die Daten zur eigenen CPU ein (wird beim generieren in MaciASL.app angezeigt: **TDP** (meist 130W bei i7); **Max Turbo Freq**; **Logical CPU**´s = 8 für i7 4-core; **CPU Frequency** (die default geschwindigkeit))
Diese Datei noch unter Tools compilern und entweder als **SSDT.aml** (zur direkten Verwendung) oder zur weiteren Verwendung als **SSDT-Vorlage.aml** sichern.

Native SSDT-Datei erstellen:
Mit MaciASL.app unter File -> New from ACPI -> **SSDT (PPM RCM)** ziehen.
Es sind die gleichen Werte, die man sich auch mit Clover ziehen kann. Nun sind erstmal tonnenweise Werte da. Uns interessieren aber nur die Werte unter **PSS** „Method (_PSS, 0, NotSerialized)“ -> die **P-States**. Der Vorteil hierbei ist, das dies die nativen Werte der CPU sind und nicht generiert. Diese Werte kann man sich nun einfach in die davor generierte **SSDT-Vorlage.aml** Datei einkopieren. Bei der generierten SSDT genügt es dies auch nur einmal für eine **CPU** zu machen, da die anderen CPU´s mit Alias informationen gefüttert werden. Wenn alle **P-States** einkopiert sind Datei nochmal compilern lassen und als fertige **SSDT.aml** abspeichern. Diese Datei kann man nun wahlweise als seperate Datei zur DSDT legen oder man fügt diese Werte in den CPU-Kopf der DSDT-Datei -> fertig!

Ich habe nun deine DSDT-Datei eher überflogen, aber es könnte bei dir sogar genügen, die P-States Werte vom **SSDT (PPM RCM)** direkt in deine **DSDT** einzukopieren und danach neu compilern.

Hier noch die Downloadlinks für die Tools:

MaciASL

DSDT Editor (Zum besseren DSDT editieren)

Damit wäre der Part der DSDT-Datei erledigt. Kommen wir nun zum **Bios-Part**:
Das Zauberwort im Bios für eine kühle CPU heißt **Vcore**.
Wir wollen die CPU undervolten. Mit meiner **i7-930 CPU** kann ich auf 1.0000V minus zwei weiteren schritten runtergehen und dazu schalte ich den **LLC** (LoadLineCalibration) auf Stufe zwei. Im Bios werden mir ca. 0.9750V angezeigt. Unter MacOS wird mir dann 0.96V angezeigt. Dank **LLC = Stufe 2** erhöht sich das dann bei Vollast bei mir der Wert auf 0.97V. Das scheint bei meiner CPU der Grenzwert zu sein, bei dem meine CPU noch genug Saft bekommt und rock-solide läuft.
Temeraturen liegen bei: **36 C** im idle und max bei **62 C** bei Vollast (bei ca. **23 C** Raumtemperatur) mit einem Boxed Lüfter. Der Verbrauch liegt bei **120W** Gesamtsystem (der komplette Rechner mit fünf HD´s) im idle und **200W** bei Vollast. Bei Spielen ist der Verbrauch höher, da die Grafikkarte ordentlich schluckt bei Vollast-Grafik. ca. 270W - 290W

Jede CPU hat ihre eigenen Werte, den Grenzwert nach unten muss man testen. Wenn es zu

wenig ist, friert der Bildschirm bei Vollast ein, wenn es nicht schon beimm booten einen Kernelpanic gibt. Dann geht man wieder in den Bios und dreht solange am Rad bis es läuft. Wenn man einen Wert gefunden hat, der läuft, stellt man zur Sicherheit den Wert um zwei Schritte höher ein und dann sollte alles absolut stabil laufen. Unter den **CPU Stromsparfunktionen** habe ich alle Optionen auf **enabled** gestellt (wie **EIST, C1, C2** u.s.w.) Zum testen sollte man auch das Tool **Prime95-MacOSX** nehmen und dann unter Options den **Torture test** auswählen. Extremer geht kein anderer Test. Dann hat man aber einen echten Grenzwert. 20-30 min reichen dann auch aus.

Wenn dir das mit der DSDT zu aufwendig ist, kannst du ja trotzdem schon mal den Vcore im Bios runterdrehen wie beschrieben und die Temperaturen mitteilen. Bei mir hat es Wunder bewirkt. Mein Xeon ist nochmals eine ganze Ecke Kühler und Stromsparender. Ihn habe ich aber mit dynamic Vcore runtergeregelt, da er extrem niedrige idle Vorgaben (0.750V) hat und etwas durstiger bei Vollast ist. Beim i7 930 war diese Variante nicht sehr effektiv.

Hossa - ist mal länger geworden als gedacht - dafür aber nur quick and dirty runtergeschrieben.
Ich habe auch grade keine Lust mehr alles nochmal Korrektur zu lesen.
Ich hoffe du kannst dennoch damit etwas anfangen.

PS:
Mir ist noch grade eingefallen: **Zum Testen ob Speedstep auf dem Rechner greift** am besten den
IORegistryExplorer.app in der Vers. 2.1 runterladen und dort unter **ACPI_SMC_Platformplugin**
nachschauen ob zwei Einträge drinstehen:
Der erste Eintrag ganz oben sollte sein:
AICPMVers Number **0x1240105 -** Der zeigt an, ob die P-States aktiv arbeiten

Der zweite Eintrag ganz unten listet die P-States auf:
PerformanceStateArray Array **xxValues** - Zeigt an, wie viele P-States gelesen werden **xx** = die Anzahl der P-States

PPS:
Von meinen Flash-Orgien erzähle ich dann mal zur anderen Zeit...
Ist mir auch nur unter El Capitan passiert!