Neues Debug-Konzept für Symmetrisches Multiprocessing (SMP)

Lauterbach, der weltweit führende Hersteller hochwertiger Debugger und Echtzeit-Trace-Tools, stellt auf der Embedded World 2008 sein neues Debug-Konzept für SMP-Systeme vor. SMP-Systeme setzen ein Betriebssystem ein, um Prozesse dynamisch auf mehrere Cores bzw. Hardware Threads zu verteilen. Als Live-Demonstration wird auf der Messe SMP-Linux auf einem MIPS 34K präsentiert.

Hardwareseitige Parallelisierung

Um eine höhere Rechenleistung für komplexe Aufgaben zu erreichen und Strom zu sparen, wird heute zunehmend auf die Parallelisierung von Abläufen gesetzt. Am weitesten verbreitet ist der Ansatz, mehrere identische Ausführungseinheiten bereitzustellen, die alle die gleichen Aufgaben bearbeiten können.

Bei dem Begriff „identische Ausführungseinheiten“ denken die meisten Leser sicher zunächst an symmetrische Multi-Core Prozessoren. Man kann sich leicht vorstellen, dass der Kernel eines Betriebssystems fest auf einem Core läuft und dafür sorgt, dass die Anwendungsprozesse gleichmäßig auf alle Cores verteilt werden (siehe Bild 1).

Für die Parallelisierung von Abläufen benötigt man jedoch nicht zwangsläufig einen Multi-Core Prozessor. Hardwareseitiges Multithreading beispielsweise ist ein Ansatz, der eine Parallelisierung auch für Single-Core Prozessoren ermöglicht. Hierbei wird

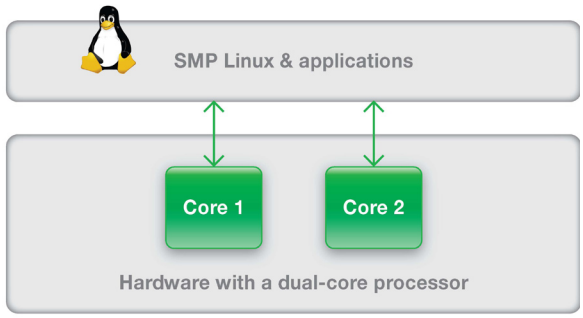

Bild 1: : SMP-Linux auf einem Dual-Core Prozessor.

bei einem Grundproblem von Cores mit Pipeline-Architektur angesetzt: *Cache Misses* oder Datenabhängigkeiten zwischen den einzelnen Befehlen führen hier nämlich immer wieder dazu, dass die Pipeline-Verarbeitung der Befehle angehalten werden muss, um auf die Verfügbarkeit der passenden Daten zu warten. Je größer die Differenz zwischen Befehlsausführzeit und Speicherzugriffszeit ist, umso mehr Rechenleistung geht verloren.

Hardwareseitiges Multithreading versucht dieses Problem dadurch zu umgehen, dass der Core mehrere, voneinander unabhängige Aufgaben quasi gleichzeitig bearbeitet. Im Falle eines Betriebssystems sind mit Aufgaben Prozesse gemeint.

Im Prinzip funktioniert dies wie folgt: Sobald ein Prozess nicht mehr weiter bearbeitet werden kann, weil auf die passenden Daten gewartet werden muss, wird einfach die Bearbeitung eines anderen Prozesses fortgesetzt.

Bild 2 zeigt zunächst die Abarbeitung dreier Prozesse auf 3 Cores mit einfacher Pipeline-Architektur und stellt dieser dann die Ausführung auf einem multithreaded Core gegenüber.

Damit hardwareseitiges Multithreading funktioniert, muss ein schnelles Umschalten zwischen den einzelnen Prozessen möglich sein. Dies lässt sich einfach dadurch realisieren, dass jeder Prozess für seinen Kontext einen eigenen Registersatz erhält. Aus dieser Vorgehensweise kann man leicht ableiten, dass die Anzahl der vom Core zur Verfügung gestellten Registersätze hardwareseitig festlegt, wie viele Prozesse parallel bearbeitet werden können. Beim ⟩⟩

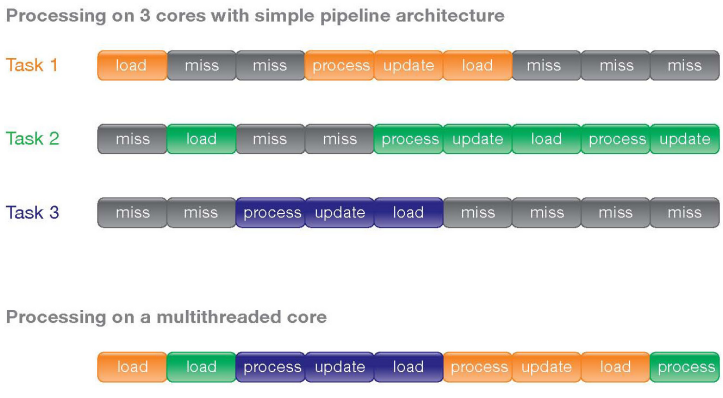

Bild 2: Durch die quasi gleichzeitige Bearbeitung mehrerer Prozesse entstehen keine Wartezeiten mehr in der Pipeline.

MIPS 34K sind dies fünf. Pro Registersatz entsteht so quasi eine „Ausführungseinheit“, ein so genannter *Hardware Thread* (siehe Bild 3).

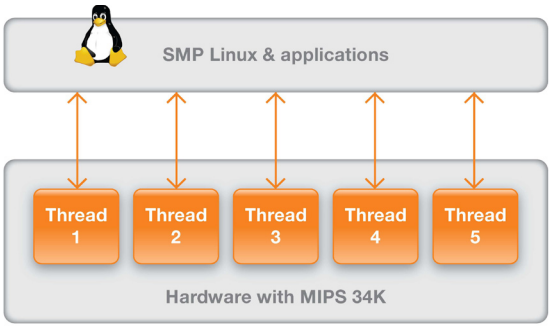

Bild 3: SMP-Linux auf einem multithreaded Core – hier MIPS 34K.

SMP-Betriebssysteme

Auf Hardware-Ebene wird die Parallelisierung von Abläufen durch symmetrische Multi-Core Prozessoren oder multithreaded Cores realisiert. Nun benötigt man noch Software, welche die Parallelisierung organisiert. Für diese Aufgabe wird in der Regel ein Betriebssystem eingesetzt.

Eine Variante, vor allem für Hardware mit identischen Ausführungseinheiten sehr geeignet, sind Betriebssysteme, die ein symmetrisches Multiprocessing – kurz SMP – realisieren. Hauptmerkmal von SMP-Betriebssystemen ist die dynamische Verteilung der Prozesse auf die verfügbaren Cores bzw. *Hardware Threads* zur Programmlaufzeit.

Weitere Merkmale sind:

- Eine Instanz des Betriebssystems betreibt alle Cores bzw. *Hardware Threads*.
- Jeder Anwendungsprozess kann auf jedem Core bzw. *Hardware Thread* laufen. Nur der Kernel ist in der Regel fest einem Core bzw. *Hardware Thread* zugeordnet.
- Alle Cores bzw. *Hardware Threads* können gleichberechtigt Ressourcen anfordern und benutzen (z.B. Speicher, externe Schnittstellen, externe Geräte).
- Das Betriebssystem stellt die Funktionen zur Verteilung der Ressourcen an die Cores bzw. *Hardware Threads* bereit.

Neues Debug-Konzept

Aktuell ermöglicht das TRACE32-Konzept mit einer Instanz des Debuggers genau einen Core zu debuggen (*Core View*). Bisher funktionierte dieses Konzept auch für Multi-Core Prozessoren reibungslos, da diese Prozessoren als asymmetrische Multiprocessing Systeme – abgekürzt AMP – betrieben wurden. Asymmetrisches Multiprocessing bedeutet: Auf jedem Core läuft eine unabhängige Instanz eines Betriebssystems. Für AMP-Systeme ist dadurch immer statisch festgelegt, auf welchem Core welcher Prozess läuft. Damit können auch die Debug-Informationen eindeutig dem entsprechenden Core zugewiesen werden.

Im Gegensatz dazu findet bei SMP-Systemen die Zuordnung der Prozesse auf die Cores bzw. *Hardware Threads* erst zur Laufzeit statt. Dadurch ist es nicht mehr sinnvoll, für das Debugging eines ausgewählten Cores bzw. *Hardware Threads* gezielt eine Instanz des Debuggers zu starten.

System View

Als Alternative zum *Core View*, der für AMP-Systeme hervorragend funktioniert, gibt es nun für SMP-Systeme den *System View*.

Mit dem *System View* wird für das Debugging aller Cores bzw. aller *Hardware Threads* nur eine Instanz des TRACE32-Debuggers gestartet (siehe Bild 4). Den Ausgangspunkt für die Informationsdarstellung bildet auch hier ein Core bzw. ein *Hardware* ⟩⟩

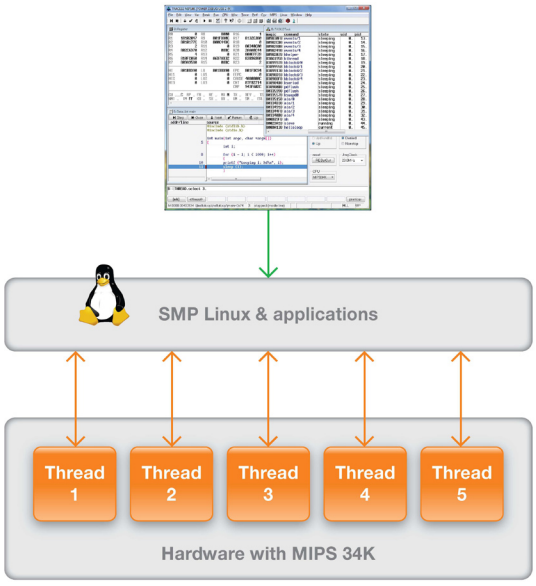

Bild 4: Für das Debugging aller Cores bzw. Hardware Threads wird nur eine Instanz des TRACE32-Debuggers verwendet.

Thread. Neu ist nun aber, dass die Informationsdarstellung per Kommando auf einen anderen Core bzw. *Hardware Thread* umgeschaltet werden kann.

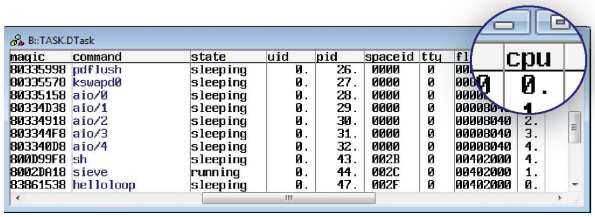

Bild 5: Aus dem TASK.DTASK Fenster lässt sich entnehmen, welchem Core bzw. Hardware Thread SMP-Linux einen Prozess zugewiesen hat.

Das Prozess-Debugging kann nun wie folgt durchgeführt werden:

1. Aus der aktuellen Auflistung aller laufenden Prozesse lässt sich entnehmen, auf welchem Core bzw. *Hardware Thread* ein Prozess läuft (Bild 5). Linux verwendet dabei den Begriff CPU, um vom Core bzw. *Hardware Thread* zu abstrahieren.
2. Durch ein Kommando kann man die Informationsdarstellung im Debugger auf den gewünschten Core bzw. *Hardware Thread* umschalten (Bild 6).

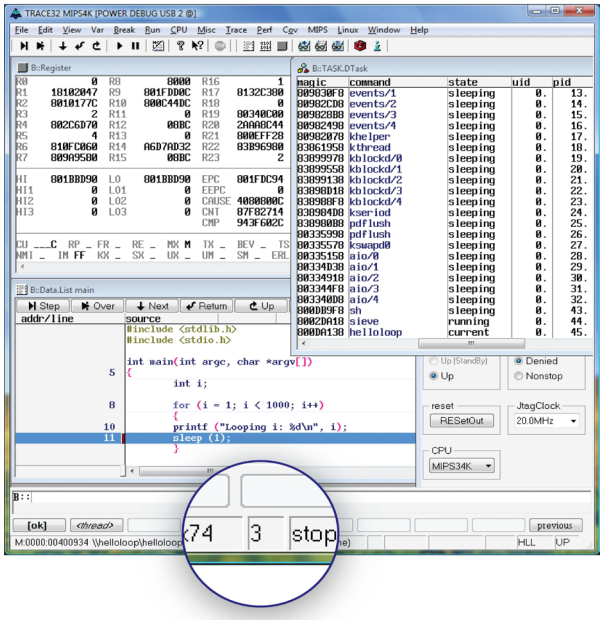

Bild 6: Im TRACE32-Fenster wird immer nur der Kontext eines Cores bzw. Hardware Threads dargestellt. Die Nummer des Cores bzw. Hardware Threads wird in der Statuszeile angezeigt.

Gemeinsame Breakpoints

Die Tatsache, dass ein SMP-Betriebssystem die Anwendungsprozesse erst zur Laufzeit einem Core bzw. *Hardware Thread* zuweist, hat natürlich auch Konsequenzen für das Setzen von *On-Chip Breakpoints*.

Zur Veranschaulichung ein einfaches Beispiel: Der Prozess „sieve“ soll gestoppt werden, sobald er auf die Variable „xyz“ schreibt. Damit der Debugger diese Anforderung umsetzen kann, muss er die Break-Logik aller Cores bzw. *Hardware Threads* für diese Abbruchbedingung programmieren, da ja erst zur Laufzeit festgelegt wird, auf welchem Core bzw. *Hardware Thread* der Anwendungsprozess „sieve“ laufen wird. Das heißt, selbst wenn jeder Core bzw. *Hardware Thread* über eine eigene Break-Logik verfügt, programmiert der Debugger die Breakpoints für den Anwender so, als gäbe es nur eine einzige, allen gemeinsame Break-Logik.

Beim Setzen von *Software Breakpoints*, für deren Realisierung der Originalbefehl im Speicher durch einen Abbruchbefehl temporär überschrieben wird, gibt es keine Änderungen gegenüber den AMP-Systemen. Betriebssystem schirmen die Adressräume der einzelnen Prozesse ja gegeneinander ab. Wird der gleiche Anwendungsprozess jedoch mehrmals gestartet, lädt z.B. Linux den Programmcode nur einmal. Jede Instanz des Anwendungsprozesses sieht so den gleichen Programmspeicher sowie die dort gesetzten *Software Breakpoints*. Um sicher zu stellen, dass die Programmausführung nur im gewünschten Anwendungsprozess gestoppt wird, ermöglicht der TRACE32-Debugger das Setzen von prozess-spezifischen *Software Breakpoints*.

Zusammenfassung

Durch die gezielte Erweiterung des TRACE32-Konzepts bietet Lauterbach seinen Kunden ein komfortables Debugging von embedded Designs, die ein SMP-Betriebssystem zur Steuerung mehrerer Cores bzw. *Hardware Threads* einsetzen. Um dieses neue Konzept für das Debugging von SMP-Betriebssystemen nutzen zu können, genügt ein TRACE32-Debugger, dessen Debug-Kabel neben der Lizenz für die Prozessorarchitektur eine zweite, z.B. eine Multi-Core-Lizenz enthält.

Nach der Unterstützung für den MIPS 34K ist für das Frühjahr 2008 das Debugging von SMP-Systemen auch für die ARM- und die PowerPC-Architekturen geplant.