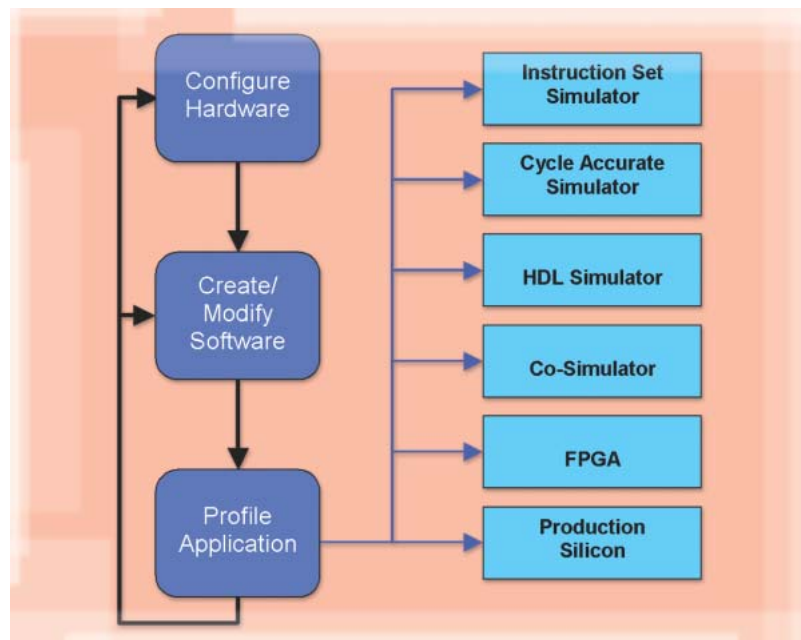


# Konfigurierbare Prozessoren – Revolution oder nur ein Gimmick?

Eine Abwägung des Aufwands lohnt sich

Bei fest konfigurierten Prozessoren bleibt dem Entwickler zur Erhöhung der Leistungsfähigkeit eines Systems nur der Umstieg auf einen anderen Prozessor oder die Erhöhung der Taktfrequenz. Konfigurierbare Prozessoren bieten dem Entwickler dagegen mehr Optionen, denn Prozessorleistung oder Systemkonfiguration können durch Modifizieren des IP genau auf die jeweilige Applikation zugeschnitten werden. Zweifellos ist eine Abwägung nötig, was mehr Aufwand macht: das Beheben der Probleme, die sich aus dem fest konfigurierten Prozessorkern ergeben, oder das individuelle Konfigurieren eines Prozessors. STEFANO ZAMMATTIO



Stefano Zammattio ist als Product Marketing Manager für ARC International tätig.

Die Zahl der Anbieter konfigurierbarer Prozessoren ist in den vergangenen Jahren erheblich gewachsen. Die Vorteile der individuellen Konfiguration von IP-Modulen (Intellectual Property) sind klar, doch gilt dies auch für das Konfigurieren des Prozessors, der das Herzstück eines Embedded-Systems darstellt? Bei fest konfigurierten Prozessoren bleibt dem Entwickler, der bestimmte Performance- und Kosten-Vorgaben erreichen muss und hierzu die Leistungsfähigkeit des Systems anheben möchte, nur der Umstieg auf einen anderen Prozessor oder die Erhöhung der Taktfrequenz. Beide Lösungen sind jedoch unbefriedigend, da sie in der Regel mit einer deutlichen Zunahme der Leistungsaufnahme und der Kosten einhergehen und die Halbleiterimplementierung erschweren. Konfigurierbare Prozessoren bieten dem Entwickler dagegen mehr Optionen, denn Prozessorleistung oder Systemkonfiguration können durch Modifizieren des IP genau auf

die jeweilige Applikation zugeschnitten werden. Aufgrund verbesserter Entwicklungswerkzeuge kann heutzutage auch die System- und Prozessor-Performance eingehender analysiert werden. Dies wiederum ermöglicht, die Ursachen von Performance-Problemen zu ermitteln. Ein konfigurierbarer Prozessor erlaubt es dem Designer, spezielle Performanceprobleme direkt anzugehen, während er bei fest konfigurierten Prozessoren nur um diese Probleme herum designen kann. Daher setzen viele Entwickler diese neue Technologie ein, und erkennen, dass damit echte Leistungszuwächse möglich sind.

## Leistungsfähigkeit: Ein mehrstufiger Ansatz

Man kann sich die Performance eines Embedded-Systems durch einen mehrstufigen Ansatz veranschaulichen. Jede Stufe kann der

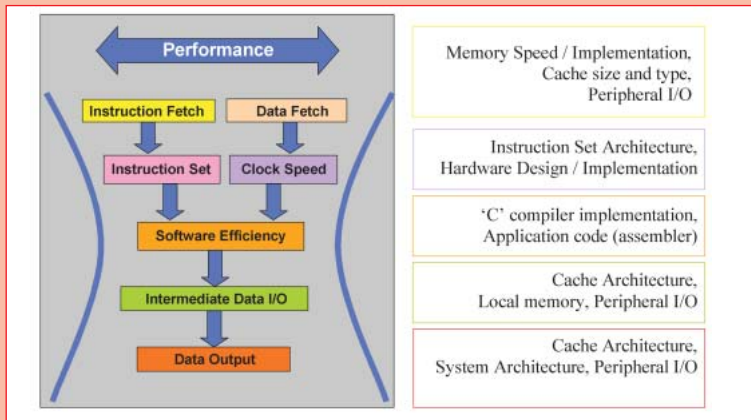


Abb. 1: Die Performance eines Embedded-Systems kann durch einen mehrstufigen Ansatz veranschaulicht werden. Jede Stufe kann dabei der Applikation Restriktionen auferlegen, die zu unerwarteten Einschränkungen der System-Performance führen können.

Applikation Restriktionen auferlegen, die zu unerwarteten Einschränkungen der Systemleistung führen können (Abb. 1). Zum Beispiel können die Speicherzugriffszeit, die Art und Größe des Cache-Speichers sowie die Implementierung der peripheren I/Os die Leistungsfähigkeit des Prozessors und damit des ganzen Systems deutlich begrenzen.

Der Entwickler nimmt normalerweise eine grobe Schätzung der Performance-Anforderung für den System- und den Speicherbus vor, indem er zu den Bandbreitenanforderungen der vom Prozessor ausgehenden Daten- und Befehlszugriffe die Bandbreite für die Peripherietransaktionen hinzuaddiert. Dies erscheint auf den ersten Blick als eine unkomplizierte Rechnung. In Wirklichkeit jedoch ist die erforderliche Prozessorbandbreite nicht einfach zu bestimmen, denn das Verhalten des Cache-Speichers lässt sich nur schwierig vorhersagen. Besonders problematisch gestaltet sich eine solche Vorhersage, wenn ein Großteil des Applikations-Codes mit Hilfe eines Compilers entwickelt wurde. Zusätzlich verschärft wird das Problem, wenn der Code intensiven Gebrauch von Funktionen und Datentransfers z.B. für Stack-Operationen oder das Ablegen globaler Variablen und Zwischenergebnisse im Speicher macht. Gleichzeitig führt die Kombination aus niedriger Taktfrequenz und einem Befehlssatz, der – bezogen auf die Applikation – relativ ineffektiv ist, zu einem System, dessen Performance durch die Verarbeitungsleistung seiner CPU begrenzt wird.

Die Softwareeffizienz eines Systems kann auch durch einen ineffektiven Compiler begrenzt werden. Dies ist nicht immer offensichtlich, da die Effizienz in hohem Maße von der Applikation, dem Programmierstil und der Prozessorarchitektur abhängt. Eine einfache Problemlösung wäre die Assembler-Programmierung. Hier jedoch hängt die Effizienz von der Fähigkeit des Programmierers ab, im verfügbaren Zeitrahmen schnellen und betriebssicheren Code zu entwickeln.

Abgesehen von den eben genannten Faktoren kann es vorkommen, dass das System in seinen I/O- und Verarbeitungs-Eigenschaften Echtzeitfähigkeit bieten muss. Unabhängig davon, ob dies mit Hilfe eines Echtzeitbetriebssystems erzielt wird, bleibt stets ein gewisser Aufwand zum Anhalten und Neustarten des Systems, damit die Echtzeitanforderungen erfüllt werden können.

## Leistungssteigerungen durch konfigurierbare Prozessoren

Der wichtigste Vorteil konfigurierbarer Prozessoren ist, dass die Leistungsfähigkeit einer jeden Stufe variiert werden kann. Die hierdurch erzielte Flexibilität erlaubt es dem Entwickler, sein System gezielt abzustimmen, um Engpässen zu beseitigen und wesentlich mehr Performance zu erreichen, als sie normalerweise mit einem fest konfigurierten Prozessor erzielbar wäre.

Nachfolgend werden zwei Beispiele beschrieben. Zum einen geht es um die Leistungssteigerung einer Anwendung mit festen Randbedingungen, nämlich eines MP3-Decoders, durch Variieren der Cache-Parameter. Zum anderen handelt es sich um die Performance-Optimierung einer Applikation, die sich weitgehend auf einen einzigen Algorithmus (Data Encryption Standard; DES) stützt. Hier wurde ein kundenspezifischer Befehl definiert.

### Beispiel 1: MP3-Decoder

Die Zielsetzung im ersten Projekt bestand darin, zur Senkung der Systemkosten die Cache-Kapazität zu minimieren, ohne die Leistungsfähigkeit negativ zu beeinflussen. Zu diesem Zweck wurde untersucht, welche System-Performance sich mit einer Reihe verschiedener Cache-Konfigurationen einstellt.

Der hauptsächlich in Assembler geschriebene Code war durch relativ unkomplizierte Daten-I/O-Anforderungen gekennzeichnet. Hieraus

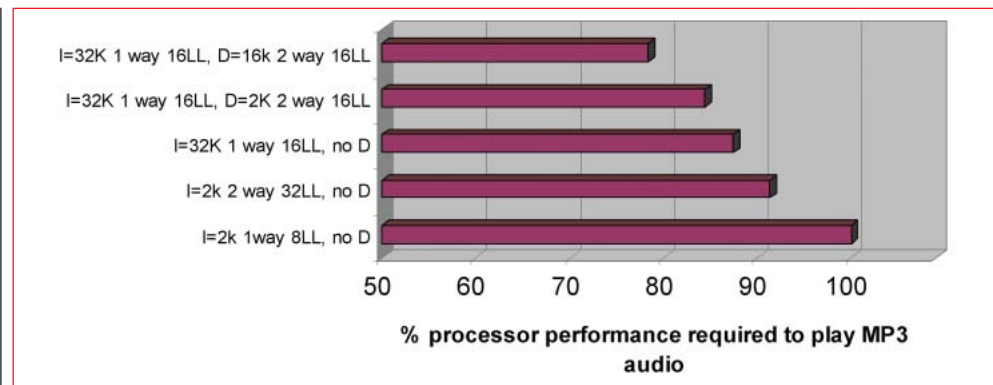


Abb. 2: In der MP3-Beispielanwendung wurde ein in Assembler geschriebene Code mit relativ unkomplizierten Daten-I/O-Anforderungen eingesetzt. Die Cache-Konfiguration kann selbst in diesem scheinbar simplen Fall einen über 20%igen Einfluss auf die Performance haben.

könnte man normalerweise den Schluss ziehen, die Cache-Konfiguration hätte nur geringfügige Auswirkungen, solange nur die Speicherkapazität des Cache ausreichend ist. Wie aber Abb. 2 verdeutlicht, kann die Cache-Konfiguration selbst in diesem scheinbar simplen Fall einen über 20%igen Einfluss auf die Performance haben. Da das Profiling eines MP3-Decoders viele Prozessorzyklen benötigt, wurde zum Generieren der Daten ein zyklusgenaues Modell des Prozessors verwendet. Um die Ergebnisse zu verifizieren, wurde dieselbe Applikation auf einem FPGA implementiert.

Ist ein System in strukturiertem C programmiert, woraus sich wesentlich mehr Stack-Transaktionen und globale Operationen ergeben, gestaltet es sich sehr schwierig, die optimale Cache-Konfiguration zu prognostizieren. Bei einem fest konfigurierten Prozessor sind die Alternativen des Entwicklers bezüglich der Cache-Größen eingeschränkt. Anders ist es bei einem konfigurierbaren Prozessor wie dem ARCTangent, der eine breite Palette von Lösungsmöglichkeiten bietet, darunter auch eine Variante ganz ohne Cache. Der Entwickler erhält hier somit die vollständige Kontrolle über die Cache-Konfiguration, ergänzt durch weitere Alternativen zum Modifizieren der I/O-Charakteristika des Prozessors. So verfügt der ARCTangent beispielsweise über einen zweiten, separat adressierbaren Bereich, der als ‚Auxiliary Space‘ bezeichnet wird und den Bedarf an Bandbreite senken kann, indem er den Datenfluss auf dem Systembus reduziert. In einem MP3-Player etwa lässt sich die Datenausgabe an diese Hilfsregister umleiten, sodass der Systembus nicht die volle Bandbreite von 44 kHz unterstützen muss.

## Beispiel 2: DES

Mit dem zweiten Projekt konnte demonstriert werden, wie es durch das Hinzufügen kundenspezifischer Instruktionen zu einem konfigurierbaren Prozessor möglich ist, die applikationsspezifische Leistungsfähigkeit entscheidend zu verbessern. Der DES-Algorithmus, von dem in Netzwerkanwendungen intensiv Gebrauch

gemacht wird, wiederholt sehr oft eine als ‚S-Box‘ bezeichnete Funktion. Diese ist mit dem üblichen Funktionsumfang fest vorgegebener Prozessorbefehlssätze schwierig zu implementieren, eignet sich dafür aber sehr gut für die Hardwarebeschleunigung. Die Implementierung der S-Box-Funktion als kundenspezifischer Befehl ermöglichte es, die Systembelastung bei der Ver- und Entschlüsselung nach dem Internet Security Protocol (IPSec) deutlich zu verringern. Durch eine solche Optimierung werden entweder Ressourcen für andere Aufgaben frei, ohne dass die Taktfrequenz angehoben werden muss, oder die Taktfrequenz des Prozessors kann herabgesetzt werden, um das Systemdesign zu vereinfachen, die Kosten ohne Performance-Einbußen zu senken oder die Leistungsaufnahme zu verringern.

Ein DES-Beschleunigungsbefehl, der die S-Box-Funktion beinhaltet, wurde definiert und in den ARCTangent-Prozessor integriert. Mit IPSec-Protokollcode sowie Tools zur Softwareentwicklung und -Analyse wurde anschließend ein Performance-Vergleich vor und nach der individuellen Konfiguration angestellt. Wie Abbildung 3 zeigt, führte die geringere Zahl der von der S-Box-Funktion benötigten Prozessorzyklen zu einer Performance-Steigerung um einen Faktor von insgesamt 91,8 für eine Triple-DES-Operation.

## Code-Beschleunigung

Die meisten Algorithmen lassen sich durch die Implementierung solch kundenspezifischer Befehle auf ähnliche Weise beschleunigen. Abbildung 3 zeigt weitere Beispiele für Code-Beschleunigungen für verschiedene Anwendungen. Die gezeigten Algorithmen sind Bestandteil des Benchmark-Tests des Embedded Microprocessor Benchmarking Consortium (EEMBC; <http://www.eembc.org>).

Ein weiterer, ebenfalls wichtiger Vorteil einer solchen Optimierung ist die Tatsache, dass der Codeumfang der beschleunigten Funktion häufig geringer ist, da ein Großteil des Codes durch eine einzige Instruktion ersetzt wird. So verrin-

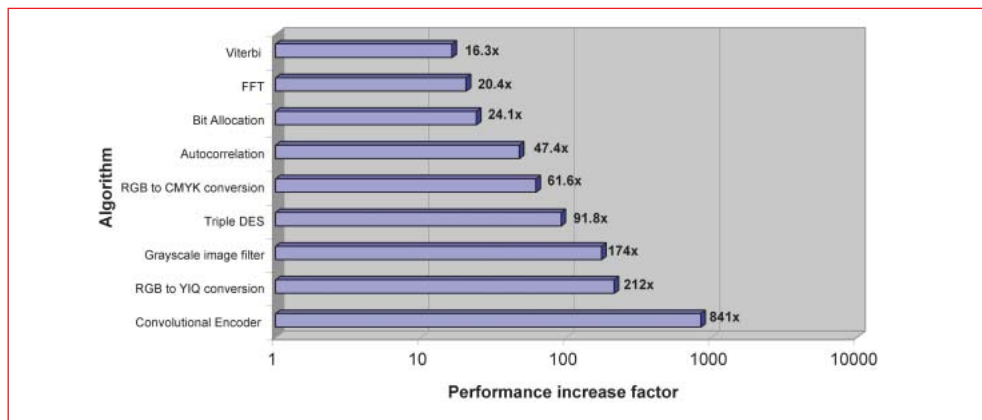


Abb. 3: Die Verwendung eines konfigurierbaren ARCTangent-Prozessor kann für eine beschleunigte Abarbeitung von Algorithmen sorgen. Die gezeigten Algorithmen sind Bestandteil des Benchmark-Tests des Embedded Microprocessor Benchmarking Consortium.

gerte sich der Umfang des Codes für den DES-Algorithmus um 63 %, nämlich von 6,5 K auf 2,5 K. Ein kompakterer Code wiederum kann sich für den Entwickler durch entscheidende Kostenersparnisse auszahlen.

All diese Modifikationen verursachen allerdings zusätzlichen Arbeitsaufwand in der Entwicklungs- und Verifikationsphase eines Projekts. Deshalb muss für jedes einzelne Vorhaben genau abgewogen werden zwischen den Restriktionen eines fest konfigurierten Prozessors und dem Arbeitsaufwand, den eine individuelle Konfiguration mit sich bringt. Den Entwicklern wird dabei zunehmend bewusst, dass es nicht allein auf die Leistungsfähigkeit des Prozessors, sondern auf die Performance des Gesamtsystems ankommt. Aus diesem Grund verlangen sie nach leistungsfähigeren Hilfsmitteln und Techniken für das Profiling der von ihnen entworfenen Systeme. Die Anbieter von EDA-Tools und IP tragen diesem Bedarf auf verschiedene Weise Rechnung. Zum Beispiel sind inzwischen zyklusgenaue C-Modelle vieler IP-Module verfügbar, die dem Entwickler eine Simulation der Software seines Systems erlauben. Gleichzeitig gibt die wachsende Leistungsfähigkeit der HDL-Simulatoren dem Entwickler die Möglichkeit, die von ihm entworfene reale Hardware zu simulieren. Diese Simulatoren verfügen zudem über offene Schnittstellen, damit Software-Debugger direkt an die HDL-Systemsimulation angebunden werden können. Diese Schnittstellen werden auch von Co-Simulations-Werkzeugen genutzt, damit HDL-Simulatoren mit anderen Systemen (z.B. mit einem zyklusgenauen C-Modell) interagieren können. Hieraus entsteht ein hybrides Systemmodell, in dem große und komplexe Blöcke (z.B. Prozessoren) in C modelliert werden, während zum Modellieren der entworfenen Hardware und ihrer Systemkonfiguration eine HDL-Simulation herangezogen wird. Vorteil hierbei: Der Entwickler profitiert von der Geschwindigkeit des zyklusgenauen C-Modells, kann aber dennoch die Genauigkeit und Transparenz der HDL-Simulation nutzen. Ohne eine solche Lösung wäre es schlichtweg

unmöglich, umfangreiche Hard- und Software-Systeme mit akzeptablem Zeitaufwand auszutesten. Mittlerweile ist eine ganze Reihe unterschiedlicher Tool-Technologien verfügbar, die dem Entwickler ein Profiling der System-Performance gestatten und ihm die Möglichkeit eröffnen, lange vor der Fertigungsphase wesentlich umfangreichere Informationen über etwaige Performance-Engpässe einzuholen. Jedes Tool übernimmt dabei eine ganz bestimmte Rolle im Entwicklungsprozess. Zum Beispiel wird ganz am Anfang des Designablaufs ein Befehlssatz-Simulator (Instruction Set Simulator) eingesetzt, während in einem späteren Stadium ein Co-Simulations-Tool besser geeignet sein kann. Wenn sich Performance-Restriktionen möglichst früh im Designzyklus aufdecken lassen, kann man dadurch gravierende Probleme vermeiden, die sonst oftmals erst mit den ersten verfügbaren Samples gefunden werden können und dann ein aufwendiges Redesign erfordern. Mit den verfügbaren Informationen kann stattdessen das Design schon während der Produktentwicklung angepasst werden, um zu gewährleisten, dass die gesetzten Performance-Vorgaben eingehalten werden. Bei einem fest konfigurierten Prozessor kann dies sehr schwierig sein, denn er kann selbst nicht verändert werden, sodass möglicherweise eine tiefgreifende Umstrukturierung des übrigen Systems und/oder der Software nötig ist. Einfaches Anheben der Taktfrequenz ist unbefriedigend, denn es erhöht die Leistungsaufnahme und macht auch die Halbleiterimplementierung schwieriger. Der Entwickler hat mehr Möglichkeiten zur Lösung dieser Probleme, wenn die Prozessor-Performance oder die Systemkonfiguration geändert werden können, um den Anforderungen des jeweiligen Produkts gerecht zu werden.

Beitrag als PDF im Internet:

[www.publish-industry.net](http://www.publish-industry.net)   
**more @ click DV13202** 