



Technische Übersicht

Speichercontroller-Technologie
Designkriterien für einen
Hochleistungs-Speichercontroller



Inhaltsverzeichnis

Designkriterien für einen Hochleistungs-Speichercontroller	1
Überblick	1
Speichercontroller-Design.....	1
DualDDR2 Speicherarchitektur	2
Unterstützung für DDR2-Hochleistungsspeicher	4
Effiziente Busnutzung.....	6
NVIDIA DASP 3.0 (dynamischer adaptiv-spekulativer Präprozessor).....	8
NVIDIA QuickSync Technologie	10
Zusammenfassung.....	12

Designkriterien für einen Hochleistungs-Speichercontroller

Überblick

Bei modernen Betriebssystemen werden Programmcode und Daten von laufenden Anwendungen im so genannten Haupt- oder Arbeitsspeicher des Systems abgelegt. Die Gesamtleistung des Systems hängt daher zu einem guten Teil davon ab, wie schnell der Hauptprozessor auf diese Daten und Handlungsanweisungen zugreifen kann. Vermittler zwischen Hauptprozessor und Arbeitsspeicher ist der Speichercontroller, der in Systemen mit einem Intel x86-basierten Prozessor in aller Regel im Systemchipsatz integriert ist.

Die Leistung des Arbeitsspeichers wird im Wesentlichen durch zwei Faktoren bestimmt: die Latenz und die Bandbreite (die den maximalen Datendurchsatz festlegt). Die *Speicherlatenz* gibt an, wie lange der Hauptprozessor nach der Anforderung bestimmter Daten aus dem Arbeitsspeicher auf diese Daten warten muss. Die *Speicherbandbreite* bestimmt das Datenvolumen, das innerhalb einer bestimmten Zeiteinheit übertragen werden kann. Betrachten wir beispielsweise einen *Cache Miss*, also einen Fall, in dem der Hauptprozessor auf Daten aus dem Arbeitsspeicher warten muss, weil sie sich nicht im Prozessorcaché befinden. Die Speicherlatenz wäre hier die Zeitdauer bis zum Eintreffen des ersten angeforderten Datenworts aus dem Speicher; die Speicherbandbreite bestimmt hingegen, wie lange es dauert, im Anschluss daran den Rest der angeforderten Cachezeile aus dem Speicher zu übertragen. Welcher dieser beiden Faktoren letztlich wichtiger für die Leistung ist, hängt von der jeweiligen Anwendung ab; ganz allgemein haben jedoch beide Faktoren großen Einfluss auf die Gesamtleistung des Systems.

Dieser Technikartikel soll einen Überblick über das Design des Speichercontrollers im NVIDIA nForce™4 SLI™ Plattformprozessor geben, die dabei eingesetzten Konzepte kurz erläutern und die positiven Auswirkungen des Designkonzepts auf die Systemleistung verdeutlichen.

Speichercontroller-Design

Der NVIDIA nForce4 SLI (Intel Edition) Plattformprozessor ist NVIDIAs erster nForce Plattformprozessor für Systeme mit einem Intel Hauptprozessor. Er ist eindeutig auf besonders leistungsbewusste Desktop-Anwender ausgerichtet, denen es primär auf die Performance ihres PCs ankommt. Vor dem Hintergrund dieser Vorgaben implementierten die NVIDIA Entwickler im neuen nForce Speichercontroller eine ganze Reihe innovativer Technologiemerkmale:

- ❑ DualDDR2 Speicherarchitektur
- ❑ Unterstützung für DDR2-Hochleistungsspeicher
- ❑ Effiziente Busnutzung
- ❑ NVIDIA DASP 3.0 (dynamischer adaptiv-spekulativer Präprozessor)
- ❑ NVIDIA QuickSync™ Technologie

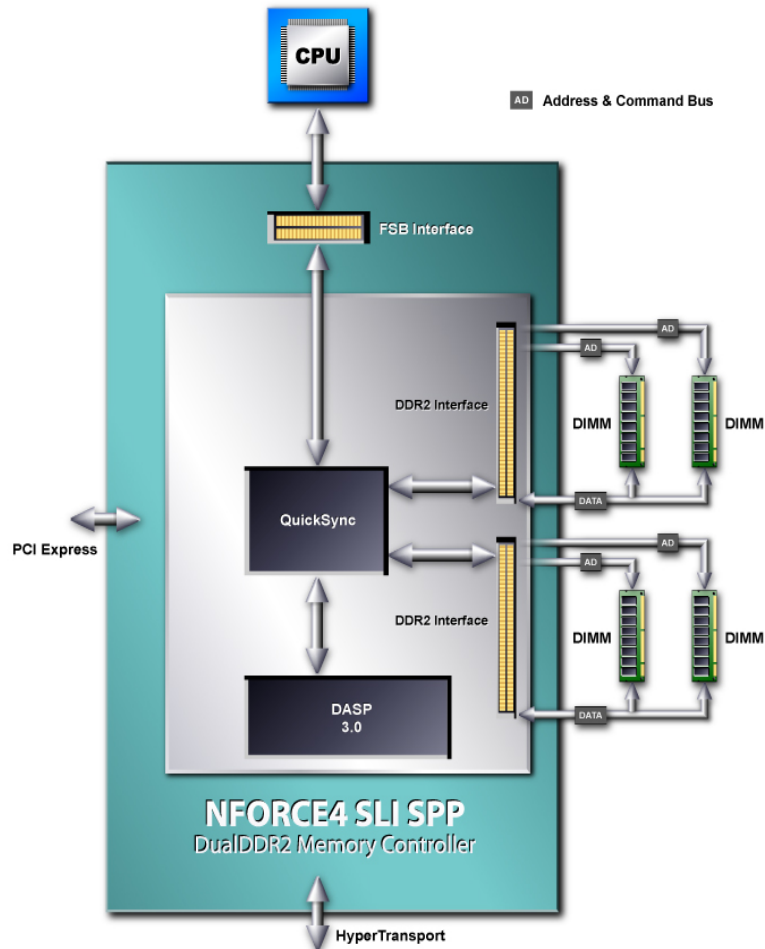


Abb. 1: NVIDIA nForce4 SLI (Intel Edition) Speichercontroller

DualDDR2 Speicherarchitektur

Der Vorgänger: DualDDR

Das Konzept, zwei parallele DDR-Speicherkanäle einzusetzen, kam bei NVIDIA bereits in früheren Modellen der NVIDIA nForce Plattformprozessoren zum Einsatz. Ziel war es, die verfügbare Speicherbandbreite zu maximieren, um auf diese Weise optimale System- und Grafikperformance zu erzielen.

DualDDR wie diese erste Generation der Dual-Channel-Technologie genannt wurde, war allerdings für eine ältere Prozessorgeneration konzipiert, die niedrigere Anforderungen an das Speichersystem stellte. So war der erste NVIDIA nForce Systemplattformprozessor mit integriertem Grafikprozessor (der SPP/IGP) auf den Betrieb mit AMD Athlon XP Prozessoren ausgelegt. Hierzu verfügte er über eine zweikanalige DDR-266-Speicherschnittstelle. Der AMD Athlon XP erreicht mit seinem mit 133 MHz getakteten 64-Bit-FSB (Frontside-Bus) eine effektive Datenrate von 266 MHz und eine theoretische FSB-Spitzenbandbreite von 2,1 GB/s. Die theoretische Speicherbandbreite einer 128 Bit breiten DDR-266-Speicherschnittstelle liegt jedoch mit maximal 4,2 GB/s noch einmal deutlich höher. Der Hauptprozessor beanspruchte also selbst bei Volllast nur die Hälfte der verfügbaren Speicherbandbreite, sodass noch ausreichende Kapazitäten für den Grafikprozessor und andere E/A-Vorgänge verblieben. Folglich bestand der erste DualDDR Speichercontroller aus zwei separaten 64-Bit-DDR-266-Speichercontrollern und einem Crossbar, der es dem Hauptprozessor und dem Grafikprozessor ermöglichte, gleichzeitig auf die beiden 64-Bit-Speicherkanäle zuzugreifen.

Die Gegenwart: DualDDR2

Moderne Prozessoren wie der Intel Pentium 4 (P4) können mit ihrer hohen FSB-Bandbreite durchaus mühelos die gesamte Speicherbandbreite in Anspruch nehmen. Um ein optimales Zusammenspiel des NVIDIA nForce4 SLI (Intel Edition) mit diesen Prozessoren zu garantieren, waren daher einige Veränderungen an der Architektur des DualDDR Speichercontrollers nötig. Ein P4-Prozessor mit einem 1066-MHz-FSB erreicht eine Spitzenbandbreite von 8,6 GB/s – was der Spitzenbandbreite einer Dual-Channel-DDR2-667 Speicherlösung von 10,6 GB/s schon sehr nahe kommt. Um die Spitzenbandbreite des Prozessors effektiv nutzen zu können, muss der Speicherzugriff folglich in jedem Fall auf beiden Speicherkanälen parallel erfolgen.

Der DualDDR2 Speichercontroller im NVIDIA nForce4 SLI (Intel Edition) verwendet hierfür ein Low-Order-Interleaving-Verfahren, über das die Speicheranforderungen des Hauptprozessors simultan an beide Kanäle geleitet werden. Das tatsächliche Ausmaß des Interleaving hängt davon ab, ob die beiden Kanäle symmetrisch oder asymmetrisch beschickt werden. Bei symmetrischer Beschickung verwendet der DualDDR2 Speichercontroller feineres Interleaving, bei asymmetrischer Beschickung muss hingegen eine gröbere Granularität verwendet werden.

Diese Interleaving-Granularität wiederum wirkt sich auf die Systemperformance aus; eine identische Beschickung der beiden Kanäle mit feinem Interleaving ergibt die beste Performance. Eine asymmetrische Beschickung der Kanäle hingegen wirkt sich aufgrund des damit einhergehenden größeren Interleaving negativ auf die Leistung aus. Zu einem gewissen Grad kann der Speichercontroller des NVIDIA nForce4 SLI (Intel Edition) dies allerdings wieder wettmachen, da er unabhängig von der Beschickung der beiden Kanäle stets im 128-Bit-Modus arbeitet. Verglichen mit anderen P4-Systemchipsatz-Lösungen, die bei asymmetrischer Beschickung der Kanäle den Speichercontroller von 128 Bit auf 64-Bit-Betrieb drosseln, ergibt sich dadurch immer noch ein deutlich besseres Leistungsverhalten.

Unterstützung für DDR2-Hochleistungsspeicher

Der NVIDIA nForce4 SLI (Intel Edition) unterstützt DDR2-Speicher mit Datentransferraten von bis zu 667 MHz. Die ursprünglichen DDR-Spezifikationen des JEDEC (Joint Electron Devices Engineering Council, einem Standardisierungsgremium der Elektronik- und Speicherhersteller) waren lediglich auf Taktraten von maximal 400 MHz ausgelegt. Als absehbar war, dass dies für moderne Prozessoren nicht mehr ausreichte, wurde vom JEDEC ein neuer Speicherstandard namens DDR2 verabschiedet. Dieser unterstützt Datenraten von 400 bis 800 MHz, verwendet eine niedrigere E/A-Spannung von 1,8 V und nutzt dank einer Reihe von Verbesserungen an der Architektur den Speicherbus effizienter.

Aus Gründen der Abwärtskompatibilität entschieden sich einige Hersteller, in ihren P4-Chipsätzen sowohl DDR als auch DDR2 zu unterstützen. Da dies jedoch aufgrund der unweigerlich anfallenden Designkompromisse negative Auswirkungen auf die Systemleistung hat, kam diese Strategie für NVIDIA nicht infrage. Folglich ist der NVIDIA nForce4 SLI (Intel Edition) ausschließlich auf DDR2-Speicher mit einer Taktrate von bis zu 667 MHz ausgelegt.

Um einen zuverlässigen Betrieb der Speicherschnittstelle bei derart hohen Taktraten von 667 MHz und mehr zu garantieren, hat NVIDIA eine Reihe von Verbesserungen in den NVIDIA nForce4 SLI (Intel Edition) integriert. So verfügt etwa jeder DIMM-Baustein über seinen eigenen Adress- und Befehlsbus (oder kurz *Adressbus*). Da die Lastkapazität des Adressbus eine der wichtigsten Beschränkungsfaktoren hinsichtlich der Speichergeschwindigkeit darstellt, lassen sich auf diese Weise deutliche Leistungsvorteile erzielen. Bei einem normalen ungepufferten DIMM kommen auf 2 Datenladungen auf dem Datenbus immerhin 8 oder 16 Ladungen auf dem Adressbus. Bei vielen Lösungen müssen sich mehrere DIMM-Bausteine diesen Adressbus teilen – bei der NVIDIA Lösung hingegen hat jedes DIMM seinen eigenen Adressbus. So erreicht der Speichercontroller einen konstant hohen Durchsatz und kann den Speicher zudem im schnellen 1T-Timing adressieren, was die Speicherlatenz senkt.

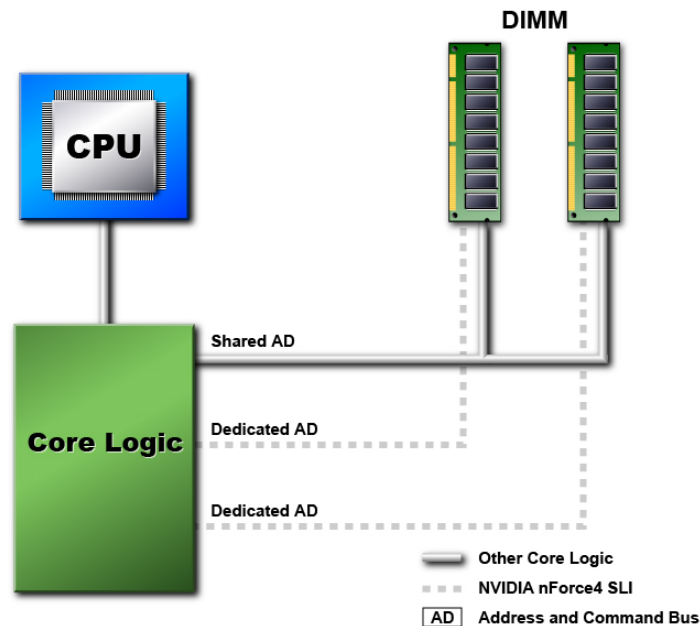


Abb. 2: Separater vs. gemeinsam genutzter Adressbus

Das Adresstiming legt fest, wie schnell Adress- und Befehlsinformationen auf den Adressbus ausgegeben werden, nachdem die für die jeweilige Anforderung „zuständige“ DRAM-Bank ausgewählt wurde. Beim 1T-Timing geschieht dies komplett in einem Taktzyklus. Beim 2T-Timing erfolgen diese Vorgänge jedoch über zwei Taktzyklen verteilt. 2T-Timing wird verwendet, wenn bei 1T-Timing nicht genügend Zeit für eine zuverlässige Vorbereitung und Übermittlung der Anforderung zur Verfügung stehen würde. Eine solche Situation tritt insbesondere bei einer hohen Last auf dem Adressbus auf – und logischerweise ist dies gerade dann mit hoher Wahrscheinlichkeit der Fall, wenn sich mehrere DIMMs den Adressbus teilen müssen.

Da sich die CAS-Latenz beim 2T-Timing um einen ganzen Taktzyklus verlängert (siehe Abb. 3), ist die Systemperformance im Vergleich zu 1T-Timing zwangsläufig schlechter. Besonders deutlich zeigt sich dies bei Anwendungen mit einem wenig linearen Speicherzugriffsmuster, da die höhere Latenz jedes Mal zuschlägt, wenn eine neue Seite geöffnet werden muss.

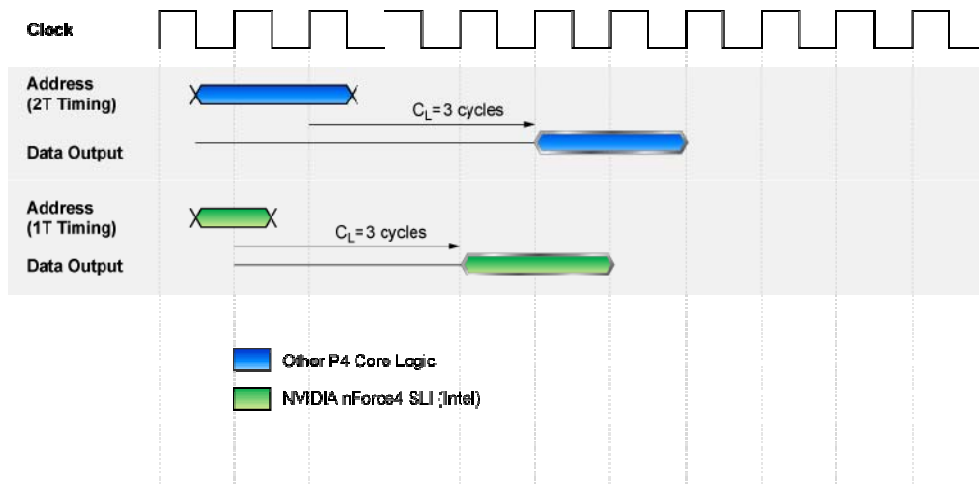


Abb. 3: 2T-Timing und 1T-Timing im Vergleich

Effiziente Busnutzung

DRAM ist unter anderem deshalb so beliebt, weil er verglichen mit alternativen Speichertechnologien so kostengünstig ist. Der niedrige Preis wiederum wird jedoch nur durch Kompromisse bei der Betriebsgeschwindigkeit, der Latenz und der Zyklendauer möglich. Während sich etwa die Rechenleistung von Hauptprozessoren in den letzten Jahren vervierfacht hat, ist die Zyklendauer von DRAM-Bausteinen im gleichen Zeitraum pro Jahr nur um etwa 7% gesunken. Die Zyklendauer von DRAM kann mit der Geschwindigkeitsentwicklung der Prozessoren also bei weitem nicht mithalten. Um diese Kluft zumindest ansatzweise zu überbrücken, versucht man, die interne Unterteilung der DRAM-Bausteine in *Bänke* möglichst geschickt zu optimieren.

Um die Systemleistung zu steigern, werden bei modernen Speichercontroller die verschiedenen Befehle an die DRAM-Bänke (Lesen/Schreiben, Activate, Precharge, Refresh) zeitlich überlappend gelegt. Hierdurch lassen sich die mit den einzelnen Befehlen verbundenen Wartezeiten teilweise eliminieren, was die Gesamtlatenz des Speichers senkt. Wie effizient sich der Adressbus auf diese Weise nutzen lässt, hängt jedoch davon ab, wie intelligent die Befehle vom Controller angeordnet werden. Bei einem ineffizienten Verfahren ergeben sich Lücken in den Befehlsketten – Adressbuskapazität geht ungenutzt verloren.

Die Effizienz dieser Befehlssteuerung hängt im Wesentlichen von der Architektur des Controllers und damit von grundlegenden Designentscheidungen der Entwickler ab. Seitenmanagement, die Anzahl der unterstützten Bänke, Adresszuordnung und Interleaving, die Burst-Blocklänge und die Prefetch-Richtlinien spielen dabei allesamt eine Rolle. Um die Problematik zu verdeutlichen, werden wir im Folgenden näher betrachten, wie sich das Adresstiming und die Burst-Blocklänge auf die Effizienz der Busnutzung auswirken.

DDR2-DRAM unterstützt Burst-Blocklängen von 4 oder 8. Die *Burst-Blocklänge* gibt an, wie viel Daten bei einem Lese- oder Schreibbefehl aus dem Speicher gelesen bzw. darin geschrieben werden. Eine Burst-Blocklänge von 4 bedeutet bei einer

64-Bit-Speicherschnittstelle etwa, dass bei einem Lese- oder Schreibvorgang 32 Byte Daten übertragen werden; bei einer Blocklänge von 8 wären es dementsprechend 64 Byte.

Der NVIDIA nForce4 SLI (Intel Edition) verwendet 1T-Adressierung mit einer Burst-Blocklänge von 4. Andere P4-Chipsätze verwenden dagegen 2T-Adressierung mit einer Blocklänge von 8. Wie sich dieser Unterschied auswirkt, wenn der Hauptprozessor beispielsweise unmittelbar nacheinander zwei Cache-Zeilen aus dem Speicher anfordert, ist in Abbildung 4 dargestellt. Der Einfachheit halber wird eine CAS-Latenz von einem Taktzyklus angenommen.

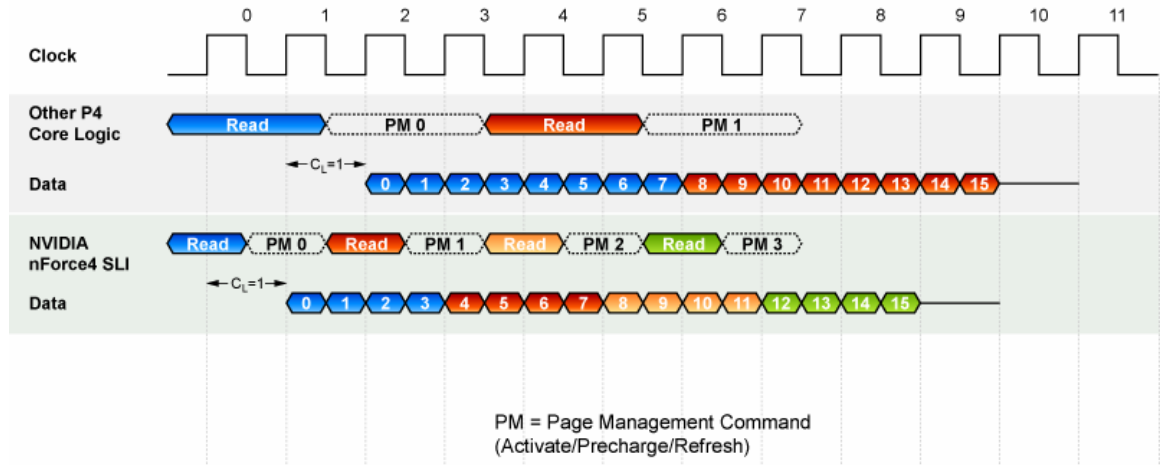


Abb. 4: Busnutzung

Bei 2T-Adressierung und einer Burst-Blocklänge von 8 werden zwei Lesebefehle benötigt, um zwei 64-Byte-Cachezeilen aus dem Speicher zu lesen. Wie in Abbildung 4 gezeigt, können per Interleaving zwei PM-Befehle (Seitenverwaltung, bestehend aus Activate/Precharge/Refresh) mit den beiden Lesebefehlen „verschachtelt“ werden.

Der NVIDIA nForce4 SLI (Intel Edition) hingegen verwendet 1T-Adressierung mit einer niedrigeren Burst-Blocklänge von 4. Um die beiden 64-Byte-Cachezeilen zu lesen, benötigt er daher vier Lesebefehle. Aufgrund der 1T-Adressierung können mit den vier Lesebefehlen jedoch gleichzeitig auch vier PM-Befehle verschachtelt werden.

Oberflächlich betrachtet übertragen sowohl der NVIDIA nForce4 SLI (Intel Edition) als auch der P4-Beispielchipsatz die gleiche Datenmenge in der gleichen Zeit. Allerdings kann der NVIDIA nForce SLI (Intel Edition) in diesem Zeitraum zwei zusätzliche PM-Befehle ausgeben. Da mehr Befehle pro Zeiteinheit bedient werden, ist also per definitionem die effektive Adressbusbandbreite beim NVIDIA nForce4 SLI (Intel Edition) höher. Dies verschafft ihm zusätzlichen Spielraum, um aggressive, komplexe Seitenmanagement-Strategien umzusetzen und auf diese Weise das Speicherlatenzverhalten sowie die Gesamtpformance des Systems zu verbessern.

NVIDIA DASP 3.0 (dynamischer adaptiv-spekulativer Präprozessor)

Um die Kluft zwischen dem schnellen Hauptprozessor und dem vergleichsweise langsamen DRAM (und der sogar noch langsameren Festplatte) zu überbrücken, setzen moderne Rechnerarchitekturen auf eine so genannte *Speicherhierarchie*. Diese erlaubt es ihnen gleichzeitig, das *Prinzip der Lokalität* auszunutzen, das in den meisten Anwendungen zu beobachten ist. Lokalität kann dabei sowohl *in der Zeit* als auch *im Raum* auftreten. *Lokalität in der Zeit* bedeutet, dass in einem bestimmten (kurzen) Zeitraum immer wieder dieselben Befehle und Daten verwendet werden. *Lokalität im Raum* bedeutet, dass die verwendeten Daten im Speicher oft in enger räumlicher Nähe zueinander liegen. Dieses Phänomen lässt sich geschickt ausnutzen: Der Prozessor trifft anhand des bisherigen Verhaltens der Anwendung eine Vorhersage, welche Daten die Anwendung in nächster Zukunft vermutlich anfordern wird, und lädt diese Daten dann schon „vorsorglich“ in einen kleinen, schnellen, prozessornahen Cache-Speicher. Wenn die Vorhersage richtig lag, kann die folgende Datenanforderung der Anwendung direkt aus dem schnellen Cache bedient werden – die Leistung steigt. Die meisten modernen Prozessoren verwenden für diesen Zweck einen sehr schnellen, aber kleinen Level-1-Cache (L1-Cache) und einen langsameren, dafür aber größeren L2-Cache.

Obwohl sich wie gesagt bei den meisten Anwendungen eine gewisse Lokalität im Raum und in der Zeit beobachten lässt, treten natürlich immer auch unerwartete Verzweigungen und Sprünge auf – insbesondere im Multitasking-Betrieb. In einem solchen Fall muss der Cache geleert und der neuen Situation entsprechend mit neuen Daten gefüllt werden. Man spricht hier von einem *Cache Miss*. Da DRAM im Zugriff gut und gerne zehnmal langsamer ist als lokaler Cache-Speicher, liegt es ganz offensichtlich im Interesse der Prozessorentwickler, solche Cache Misses nach Möglichkeit zu vermeiden; daher wird die „Vorhersagelogik“ in der Regel auch noch dahingehend erweitert, dass spekulativ auch Daten geladen werden, die nach einem eventuellen Cache Miss benötigt werden könnten.

An dieser Stelle noch ein kurzer Einschub zu den verschiedenen denkbaren Situationen beim Speicherzugriff. Die internen Bänke von DRAM-Speicher werden vom Speichercontroller als Seiten verwaltet. Eine Seite wird geöffnet, wenn eine Zeile in einer Bank aktiviert und ihr Inhalt zum Operationsverstärker (Sense Amplifier) übertragen wird. Wenn sich die vom Prozessor angeforderten Daten in der gerade geöffneten Seite befinden, kann der Arbeitsspeicher die Anforderung annehmbar schnell bedienen (wiewohl immer noch deutlich langsamer als ein lokaler Cache). Diese Situation wird als *Page Hit* („Seitentreffer“) bezeichnet. Man könnte sich die Seiten in diesem Zusammenhang also wie einen L3-Cache vorstellen. Wenn sich die angeforderten Daten in einer Bank ohne geöffnete Seite befinden (*Page Miss mit geschlossener Seite*), muss die Seite zunächst geöffnet werden, bevor die Anforderung bedient werden kann. Dies dauert naturgemäß länger als bei einem Seitentreffer. Am längsten ist die Latenz jedoch, wenn sich die angeforderten Daten in einer Bank befinden, in der zu diesem Zeitpunkt die falsche Seite geöffnet ist (*Page Miss mit geöffneter Seite*); hier muss zusätzlich noch die ursprüngliche Seite geschlossen werden, bevor die passende Seite geöffnet werden kann.

Wenn der Prozessor nun Daten aus dem Speicher anfordert, kann dies zwei Gründe haben: Entweder er lädt gemäß seines spekulativen Vorhersagealgorithmus „vorsorglich“ Daten für weitere Anforderungen (sog. *Prefetching*), oder er muss

aufgrund eines Cache Miss seinen Cache neu bestücken. Bei einem intelligenten Prozessordesign können jedoch regelmäßig über 90% der Speicheranforderungen aus dem Cache bedient werden, der Anteil der Cache Misses liegt also unter 10 Prozent. Wenn der Prozessor auf den Speicher zugreift, geschieht dies folglich weitaus öfter zum Zwecke des Prefetching als aufgrund eines Cache Miss.

Um wiederum Page Misses möglichst zu vermeiden und den Anzahl der Seitentreffer zu erhöhen, wurde eine ganze Reihe von teilweise sehr komplexen Verfahren zur Seitenverwaltung entwickelt. Was die Vermeidung von Page Misses angeht, sind die Ergebnisse dieser Ansätze jedoch allesamt eher mager. Erfolgversprechender ist es, im Systemchipsatz eine Prefetching-Logik zu integrieren, die spekulativ die Daten abrufen und bereitstellt, die vom Prozessor als nächstes angefordert werden könnten. Dies ist allerdings leichter gesagt als getan – hauptsächlich aufgrund der ähnlichen Logik, die bereits im Prozessor integriert ist. Um effektiv zu funktionieren, muss die Prefetching-Logik im Chipsatz letztlich absehen können, wie sich ihr Gegenstück, also die Prefetching-Logik im Prozessor, verhalten wird.

Zusätzlich erschwert wird diese Aufgabe durch neue Prozessoren mit Hyper-Threading-Technologie. Hier muss die Prefetching-Logik im Chipsatz überdies intelligent genug sein, um das Verhalten beider Threads gleichzeitig vorherzusagen und die jeweils benötigten Daten bereitzustellen. Sogar noch komplexer wird es bei Dual-Core-Prozessoren, also Prozessoren mit zwei Prozessorkernen, die wiederum jeweils Hyper-Threading-fähig sind.

In der ersten Generation des NVIDIA nForce IGP und SPP wurde ein solcher dynamischer, adaptiv-spekulativer Präprozessor (DASP) erstmals integriert. Der DASP 1.0 des NVIDIA nForce war auf die Zusammenarbeit mit den AMD Prozessorfamilien Athlon XP und Duron ausgelegt und verschaffte NVIDIA nForce-basierten PCs einen spürbaren Leistungsgewinn. Die folgende Generation, der NVIDIA nForce2 IGP und SPP, konnte bereits mit DASP 2.0 und deutlich effizienter arbeitenden Vorhersagealgorithmen aufwarten. Möglich wurde dies einerseits durch die gesammelten Erfahrungen seit der ersten Version, größtenteils jedoch durch Verbesserungen an der Vorhersagelogik der Prozessoren.

Im NVIDIA nForce4 SLI (Intel Edition) wurde diese Technologie ein weiteres Mal verbessert – das Ergebnis heißt demzufolge DASP 3.0. Die Veränderungen gegenüber dem Vorgänger resultieren im Wesentlichen durch besseren Einblick in das Verhalten moderner Anwendungen, Verbesserungen in der Prefetching-Logik des Prozessors sowie herstellereinspezifischen Unterschieden zwischen den Prefetching-Verfahren von AMD bzw. Intel. Daneben wurde der neueste Entwicklungsstand in den Gebieten der Rechner- und Speicherarchitektur, der Programmierung und der Gestaltung von Datenstrukturen berücksichtigt. Mit enormem Aufwand untersuchten die NVIDIA-Entwickler dabei Speicherzugriffsprotokolle von Hunderten von Anwendungen, um den effizientesten Vorhersagemustern auf die Spur zu kommen. Letztlich dürfte DASP 3.0 damit die ausgefeilteste Chipsatz-Prefetching-Logik sein, die momentan auf dem Markt ist.

Die Präprozessor-Logik von DASP 3.0 kann mehrere Prozessorkerne und Threads gleichzeitig überwachen, das Ausführungsverhalten vorhersagen und die jeweils passenden Prefetching-Daten aus dem Speicher abrufen. Da sie adaptiv konzipiert ist, passt sie außerdem im laufenden Betrieb ihr Verhalten an den jeweiligen Thread

an. So kann sie etwa Feinabstimmungen am Vorhersagealgorithmus vornehmen, auf einen anderen Algorithmus umschalten oder auch aus verschiedenen Algorithmen einen für die aktuelle Situation geeigneten Hybrid-Algorithmus zusammenstellen. Für optimale Koordination sind die verschiedenen Präprozessoren allesamt an einen zentralen Arbiter angebunden, der ihre Prefetch-Anforderungen priorisiert. Dabei überwacht der Arbiter gleichzeitig auch alle Speicherzugriffe des Hauptprozessors, des Grafikprozessors und anderer Geräte im System. Mittels eines ausgeklügelten Fairness-Algorithmus kann der Arbiter auf diese Weise sicherstellen, dass kein Subsystem unangemessen lange auf „seinen“ Speicherzugriff warten muss.

NVIDIA QuickSync Technologie

Sowohl beim FSB als auch bei der Speicherschnittstelle handelt es sich jeweils um voll synchrone Busse, die unabhängig voneinander arbeiten. Die Speicheranforderungen des Hauptprozessors an den Systemchipsatz werden an der FSB-Taktfrequenz synchronisiert; das heißt, sie befinden sich in der *FSB-Taktdomäne*. Die tatsächlichen Speicherzugriffe auf den DRAM hingegen werden mit dem Takt der Speicherschnittstelle synchronisiert, sie liegen also in der *Speichertaktdomäne*. Für die Vermittlung zwischen diesen beiden Bereichen sorgt der Systemchipsatz; er übermittelt die Anforderungen des Prozessors von der FSB- in die Speichertaktdomäne und überträgt die Daten zwischen den beiden Taktdomänen.

Um eine derartige Aufgabe zuverlässig zu bewältigen, werden in der Regel Synchronisierungsschaltungen eingesetzt. Diese bringen jedoch eine gewisse Verzögerung in den Signalweg ein. Wie groß diese Verzögerung ist, hängt von der Phase und der Frequenz der beiden Takte ab. Wenn die beiden Taktdomänen mit derselben Frequenz (aber unterschiedlichen Phasen) arbeiten, lässt sich eine Synchronisierungsschaltung relativ einfach realisieren. Daher empfehlen viele Chipsatzhersteller, nach Möglichkeit Speicherbausteine zu verwenden, deren Takt mit dem FSB-Takt identisch ist. Bei einem Prozessor mit 1066-MHz-FSB (d. h. Taktrate = 266 MHz) sollte man sich nach diesem Verfahren folglich für 533-MHz-Speicher entscheiden, da dieser ebenfalls eine Taktrate von 266 MHz verwendet.

Vom Standpunkt der maximal möglichen Performance betrachtet ist dieser Ansatz jedoch nicht optimal: Aufgrund der Taktbeschränkungen kann der Anwender nicht ohne weiteres immer den Prozessor mit dem schnellsten FSB oder den leistungsfähigsten DDR2-Speicher mit dem höchsten Speichertakt verwenden. Für maximale Performance sollten daher der FSB und die Speicherschnittstelle unabhängig voneinander mit der jeweils höchstmöglichen Geschwindigkeit betrieben werden. Dies macht allerdings die Synchronisierungsschaltung etwas komplexer, da sie nun zwischen zwei zueinander asynchronen Taktdomänen vermitteln muss.

Overclocking (Übertakten des Systems) stellt in dieser Hinsicht eine weitere Schwierigkeit dar. Sobald der FSB- und/oder Speichertakt geändert wird, ändert sich das Timing-Verhältnis zwischen den beiden Takten. In manchen Fällen kann es daher vorkommen, dass ein höherer Takt letztendlich sogar der Performance schadet, da sich die Synchronisierungsschaltung eventuell entscheidet, anstatt der nächsten die übernächste Taktflanke abzuwarten, um die Daten sicher zwischen den beiden Taktdomänen vermitteln zu können (siehe Abb. 5). So entstehen weitere Verzögerungen im Signalweg, und die Performance des Systems sinkt.

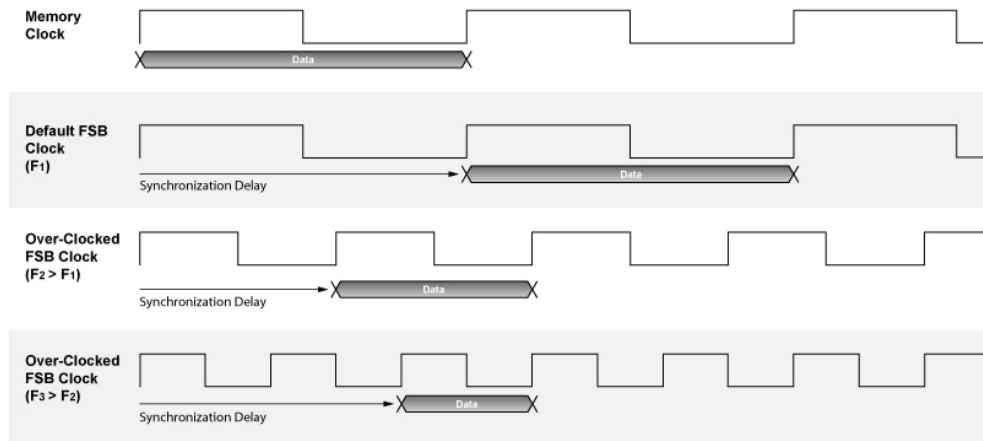


Abb. 5: Verzögerung durch Synchronisierung

Der NVIDIA nForce4 SLI (Intel Edition) ist hingegen so konzipiert, dass er ohne derartige Leistungseinbußen immer den maximal möglichen FSB- und Speichertakt verwenden kann. Möglich wird dies durch seine zum Patent angemeldete NVIDIA QuickSync™ Synchronisierungstechnologie, die Speicheranforderungen in kürzester Zeit zwischen FSB- und Speichertakt domäne vermittelt. Hierzu passt QuickSync die Geschwindigkeit der internen Datenwege zwischen den beiden Domänen automatisch an den FSB- und Speichertakt an; wird einer der Takte erhöht, so läuft auch der Datenverkehr zwischen den beiden Domänen schneller.

Auf diese Weise kann die QuickSync Technologie im NVIDIA nForce4 SLI (Intel Edition) sicherstellen, dass die Verzögerung zwischen einer Speicheranforderung des Prozessors und deren Übermittlung auf den Speicherbus immer so klein wie möglich bleibt; gleiches gilt für die Gegenrichtung (Speicher zu Prozessor). Dies gilt unabhängig von FSB- und Speichertakt, was maximale Flexibilität und Performance garantiert.

Zusammenfassung

Speicherperformance ist einer der entscheidenden Bestimmungsfaktoren der Gesamtperformance eines Systems. Gleichzeitig ist sie eine sehr komplexe Angelegenheit: Um ein optimales Leistungsverhalten zu erreichen, muss beachtlicher Entwicklungsaufwand getrieben werden. Architekturüberlegungen, Systemsimulationen, Anwendungsverhaltensanalysen, ausgeklügelte Logik und viel Know-how bei der manuellen Optimierung von Schaltungen und Datenwegen gehen hier Hand in Hand.

Beim NVIDIA nForce4 SLI (Intel Edition) hat NVIDIA in dieser Hinsicht keine Mühen gescheut: Ein beträchtlicher Anteil der Chipschaltungen und des Entwicklungsaufwands entfiel auf den Speichercontroller, die in diesem Artikel beschriebenen Technologien und andere proprietäre Leistungsverbesserungen. So kann der NVIDIA nForce4 SLI (Intel Edition) nun mit dem modernsten Speichercontroller der Branche sowie einem exzellenten Leistungsverhalten in den verschiedensten Anwendungen aufwarten.

Bei Systemchipsatz-Lösungen für Highend-PCs mit AMD Prozessoren nehmen die NVIDIA nForce Plattformprozessoren dank ihrer Performance, Funktionalität, Zuverlässigkeit und Qualität längst den höchsten Marktanteil ein. Mit dem NVIDIA nForce4 SLI (Intel Edition) können nun endlich auch Intel Anwender von den Stärken der nForce Plattformprozessoren profitieren.

Hinweis

ALLE NVIDIA-DESIGNSPEZIFIKATIONEN, REFERENZPLATINEN, DATEIEN, ZEICHNUNGEN, DIAGNOSEPROGRAMME, LISTEN UND SONSTIGEN DOKUMENTE (EINZELN ODER IM GANZEN ALS „MATERIALIEN“ BEZEICHNET) WERDEN „AS IS“ („WIE BESEHEN“) BEREITGESTELLT. NVIDIA GIBT HINSICHTLICH DER MATERIALIEN KEINERLEI GARANTIE, UNABHÄNGIG DAVON, OB DIESE AUSDRÜCKLICH, KONKLUDENT, GESETZLICH ODER ANDERWEITIG BEGRÜNDET SIND. INSBESONDERE WERDEN AUSDRÜCKLICH KEINERLEI GARANTIE HINSICHTLICH DER NICHTVERLETZUNG VON URHEBERRECHTEN, DER MARKTGÄNGIGKEIT SOWIE DER EIGNUNG FÜR EINEN BESTIMMTEN ZWECK ÜBERNOMMEN.

Die in diesem Artikel genannten Informationen sind nach bestem Wissen und Gewissen zutreffend und verlässlich. Die NVIDIA Corporation übernimmt jedoch keinerlei Verantwortung für Konsequenzen, die aus der Nutzung dieser Informationen entstehen, bzw. für Patentrechtsverletzungen oder andere Verstöße gegen die Rechte Dritter, die aus einer solchen Nutzung entstehen. Es wird weder konkludent noch anderweitig eine Lizenz im Rahmen eines Patents oder eines Patentanspruchs der NVIDIA Corporation gewährt. Die in diesem Artikel genannten Spezifikationen können sich jederzeit ohne weitere Ankündigung ändern. Dieser Artikel löst alle eventuell vorab bereitgestellten Informationen ab und ersetzt diese. Ohne die ausdrückliche vorherige schriftliche Genehmigung der NVIDIA Corporation dürfen Produkte der NVIDIA Corporation nicht als missionskritische Komponenten in lebenserhaltenden Geräten oder Systemen eingesetzt werden.

Marken

NVIDIA, das NVIDIA Logo, NVIDIA nForce, QuickSync und SLI sind in den USA bzw. international Marken und/oder eingetragene Marken der NVIDIA Corporation. Bei anderen Firmen- und Produktnamen kann es sich um Warenzeichen der jeweils damit verbundenen Unternehmen handeln, die hiermit anerkannt werden.

Copyright

© 2005 NVIDIA Corporation. Alle Rechte vorbehalten.



nVIDIA.

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com