

Pentium[®] III プロセッサの実装条件

Jagannath Keshava -- インテル・コーポレーション、マイクロプロセッサ・プロダクツ・グループ
Vladimir Pentkovski -- インテル・コーポレーション、マイクロプロセッサ・プロダクツ・グループ

摘要

この記事では、互いにトレードオフの関係にある Pentium[®] III プロセッサのさまざまな実装条件について解説します。Pentium III プロセッサでは、インターネット・ストリーミング SIMD (Single-Instruction、Multiple-Data) 拡張命令(インターネット SSE)と呼ばれる IA-32 の新しい拡張命令が実装されています。Pentium III プロセッサの設計は、Pentium[®] Pro マイクロアーキテクチャに基づくものです。

Pentium III プロセッサの当初の開発目標は、パフォーマンスとコストと動作周波数のバランスを実現することにあります。この記事では、SIMD 浮動小数点(FP)アーキテクチャとメモリ・ストリーミング・アーキテクチャの要点について解説し、3D パイプのバランスの観点から、これらのアーキテクチャが、メモリ・アクセスと演算実行を分離して効率的な処理を実現する上でどのような役割を果たしているかを説明します。また、実装の主要部については、具体的にどのような選択肢があるのかを示します。新しい浮動小数点ユニットなど、インターネット SSE の実行ユニットの詳細と、既存の 64 ビット・データパスを使用した 128 ビット命令の実装方法についても説明します。さらに、メモリ・ストリーミングの実装についても詳しく解説します。

Pentium III プロセッサには、550MHz 版をはじめ、動作周波数の異なるいくつかの製品が用意されています。Pentium III プロセッサに実装されたインターネット SSE の新しい命令群は、ダイ・サイズに関して僅か 10%程度のコスト増で、迫力ある豊かなビジュアル・コンピューティングを実現しました。

はじめに

インターネット SSE の開発目標は、ビジュアル・コンピューティングの質をさらに向上させること、および、リアルタイム・ビデオ・エンコードや音声認識などの新しいアプ

リケーションの実現することにあります(参考資料[7])。Pentium III プロセッサは、インターネット SSE を初めて実装した製品です。Pentium III プロセッサは P6 マイクロアーキテクチャを基盤としているため、ダイ・サイズや作業量の点で効率的な実装が可能になっています。開発目標の主眼は、ダイ・サイズの拡大を Pentium[®] II プロセッサの 10%増程度に抑えながら、インターネット SSE を実装し、同時に、少なくとも 1 ランク上の高速動作を実現することでした。

インターネット SSE がターゲットとする新しいアプリケーションには、コンピュータ・システム設計上の課題となる 2 つの特徴があります。第 1 に、これらのアプリケーションの基盤をなすアルゴリズムは、一連の同じ処理を同時に複数のデータ要素に適用できるという意味において、本質的な並行性を持っています。インターネット SSE を使用すればこの並行性は明示的に表現できますが、その並行性をパフォーマンスの向上につなげるにはハードウェアがその機能を備えていることが必要です。P6 のスーパースケラ・アウトオブオーダー・マイクロアーキテクチャは、抽出された暗黙の並行性と同様に明示的な並行性を活用する能力も備えています。しかし、P6 よりも高速な演算スループトをサポートするハードウェアを使用すれば、これらのアルゴリズムのパフォーマンスがさらに向上することは言うまでもありません。Pentium III プロセッサの開発では、そのようなハードウェアの開発とその使用効率の向上が中心的な課題となりました。第 2 に、並行処理するデータを遅延なく供給するためには、システムで広帯域のメモリ・アクセスを実現して、メモリのレイテンシを隠蔽する必要があります。

この記事の「実装」のセクションでは、ダイ・サイズと動作周波数の厳しい条件を満たしながら、演算とメモリ・アクセスの高速なスループトを実現する方法について、その一部を紹介します。ただし、この記事の主目的は、あく

まで Pentium III プロセッサで使用されている主要な実装技術とそれらの開発理由を解説することにあります。

アーキテクチャ

Pentium® III プロセッサは、インターネット SSE を初めて実装した製品です。インターネット SSE には 70 の新しい命令と 1 つの新しいアーキテクチャ・ステートが含まれています。インターネット SSE の導入は、80386 以来のインテル・アーキテクチャの歴史において、MMX®テクノロジーの導入に次ぐ 2 度目の大幅な命令セットの拡張ですが、新たなアーキテクチャ・ステートが追加されたのは今回が初めてです。MMX®テクノロジーによる拡張の際は、アーキテクチャ・ステートはまったく追加されませんでした。インターネット SSE の新しい命令は、以下のようなカテゴリに分類できます。

- 単精度の浮動小数点値を同時に 4 つ処理できる SIMD FP 命令
- スカラ FP 命令
- キャッシュ階層の特定レベルへのプリフェッチなどを実行する、キャッシュビリティ命令
- 制御命令
- データ変換命令
- エンコードを高速化する PSAD やデコードを高速化する PAVG などを含む、新しいメディア拡張命令
- エンコードを高速化する PSAD やデコードを高速化する PAVG などを含む、新しいメディア拡張命令

新しいステートが追加されたことにより、実装の複雑性が抑えられてプログラミング・モデルの問題が軽減された

だけでなく、SIMD-FP 命令を MMX テクノロジー命令または X87 命令と並行して使用することもできるようになっています。また、新しいステートの導入により、ソフトウェア・メカや OS メーカーから出されていた要望の多くが実現しました。インターネット SSE のステートはすべて x87-FP ステートとは分離され、数値例外の処理用に専用の割り込みベクタが用意されたほか、新たに制御/ステータス・レジスタの MXCSR も追加されています。MXCSR レジスタは、数値例外処理のマスク/アンマスク、丸めモードの設定、flush-to-zero モードの設定、およびステータス・フラグの確認に使用されます。アプリケーションによっては、スカラ演算とパックド・データ演算の両方が必要になるものがありますが、この問題への対処方法として、新しい SIMD-FP モードでは明示的なスカラ命令が定義されました。このスカラ命令は、Pentium III プロセッサでは単一のマイクロ命令しか実行しません。浮動小数点演算では、厳密な単精度計算とポータビリティを必要とするアプリケーションに適した IEEE モードと、高性能リアルタイム・アプリケーション向けの Flush-To-Zero (FTZ) モードの 2 つのモードがサポートされています。

(命令セットについては、この号のインテル・テクノロジー・ジャーナルの別の記事で詳しく解説されています。)

実装

このセクションでは、Pentium® III プロセッサの浮動小数点演算とメモリ・アクセスのスループットを高めるために開発された主要な機能について解説します。また、ダイ上のスペースを効率的に活用するために開発された 2 つのテクニックについても取り上げます。図 1 は、P6 パイプラインの機能をブロック・ダイヤグラムで示したものです。

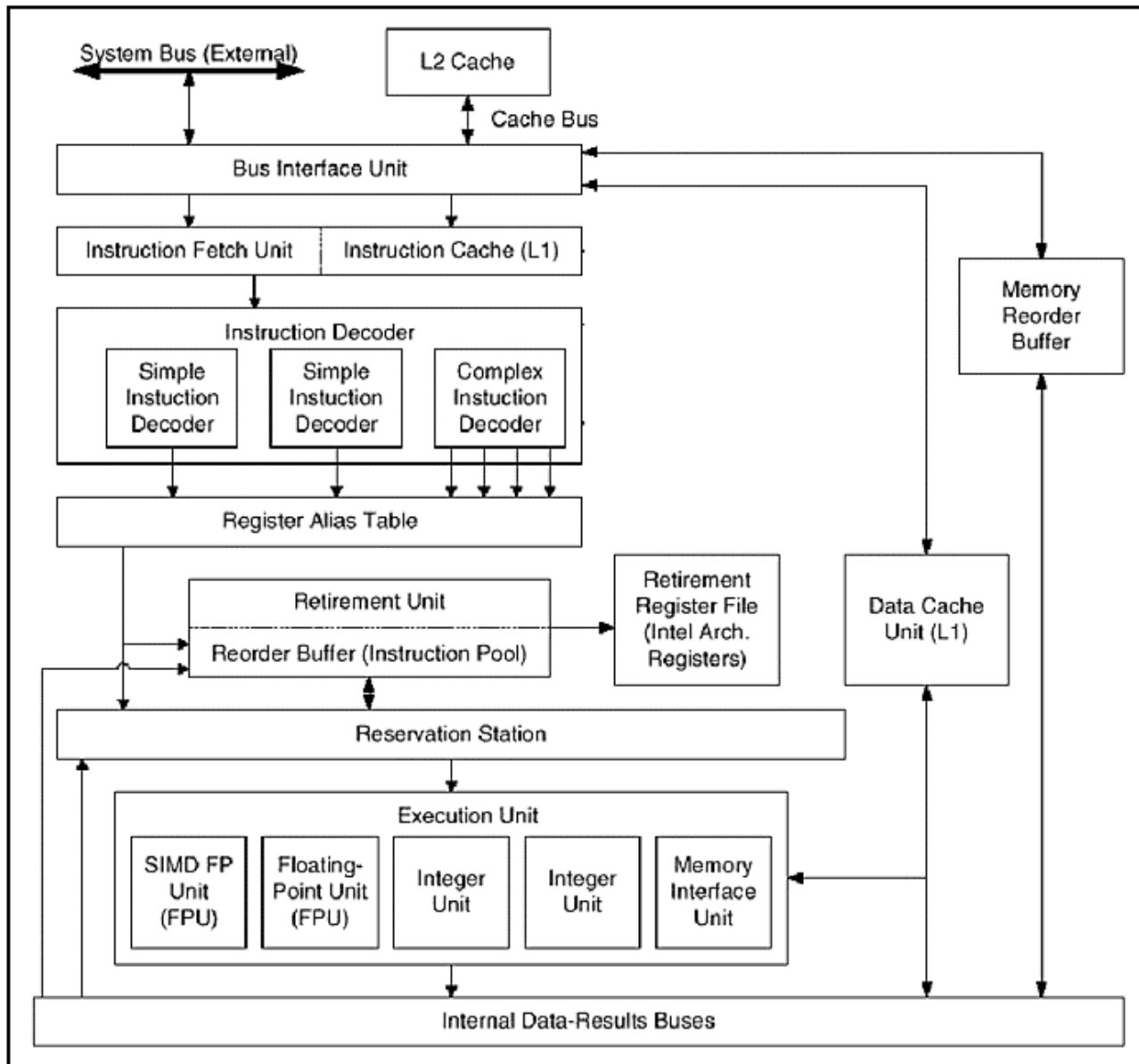


図1: P6 ファミリ・プロセッサ・アーキテクチャの機能ブロック・ダイアグラム

図2に、Pentium® II プロセッサのディスパッチ/実行ユニットのブロック・ダイアグラムを示します。P6のアーキテクチャおよびマイクロアーキテクチャの概要については、参

考資料の[5]と[6]を参照してください。これらの資料には図1と図2にブロックで示された各ユニットの説明も含まれています。

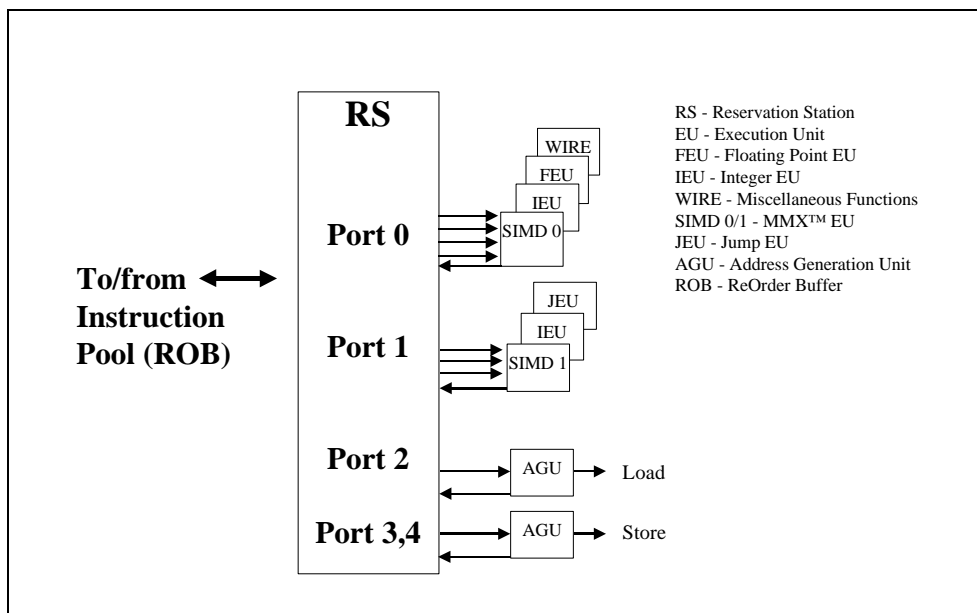


図2: Pentium® II プロセッサのディスパッチ実行ユニット

インターネットSSE 実行ユニットの実装

インターネット SSE は以下のような方法で実装されています。4ワイド(128ビット)インターネット SSE 命令は、命令デコーダにより、2ワイド(64ビット)内部マイクロオペレーションのペアに変換されます。この2ワイド・マイクロオペレーションの実行は、2ワイド実行ユニットでサポートされます。FP 実行ユニットの一部は、既存の P6 FP ユニットの機能を拡張する方法で開発されました。2ワイド実行ユニットは、Pentium II プロセッサの2倍に相当するパフォーマンスを実現します。しかも、64ビット・データバスを使用して128ビット命令を実装しているため、デコーダの仕様や、既存および新規の実行ユニットの使用方法を

従来のものから大幅に変更する必要がありません。このほかに、FP ハードウェアの使用効率を高めるために、以下のような機能が実装されています。

1. 加算器と乗算器は別々のポートに配置されているため、2ワイド加算演算と2ワイド乗算演算を同時にディスパッチできます。その結果、ピーク時のパフォーマンスは Pentium II プロセッサに比べて倍になり、550MHz 動作では 2.2GFLOP/秒のピーク値が達成されました。図3では、Pentium III プロセッサのために開発された新しいユニットと変更が加えられた P6 ユニットの色付きのブロックで示しています。

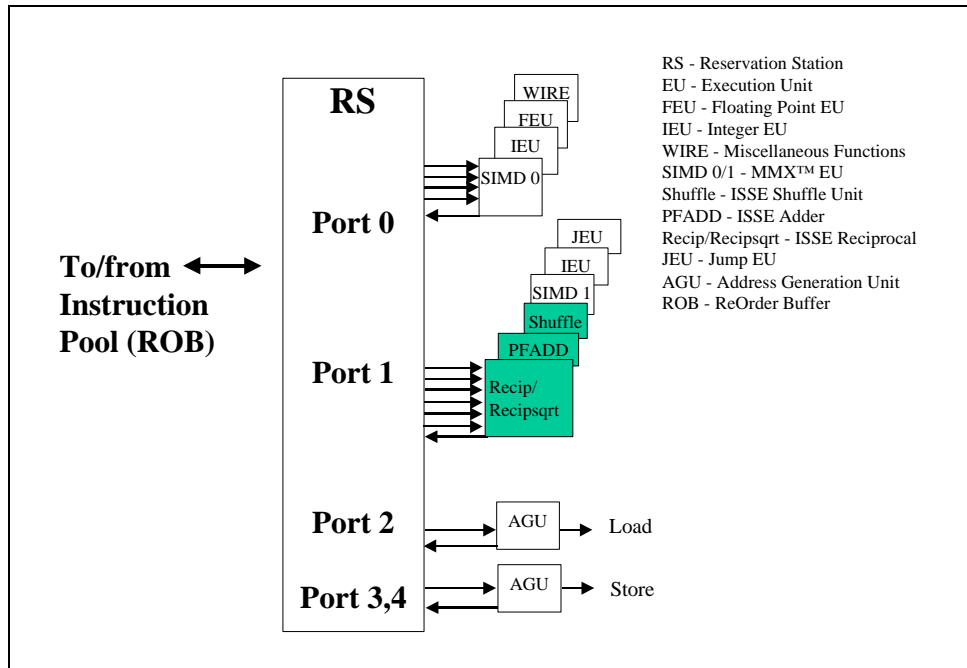


図 3: Pentium® III プロセッサのディスパッチ実行ユニット

新開発のユニットはすべてポート 1 に追加されました。ポート 0 上の新しい演算は、既存ユニットの変更によって実現されています。乗算には、従来の FP 乗算器を改良したポート 0 上のユニットが使用されます。新たにサポートされたパックド SP 乗算器は、1 クロック・サイクルごとに 2 つの SP 値を処理できます。レイテンシは 4 クロックです。(従来の X87 SP プロセッサのスルーputでは、1 つの SP 値の処理に 2 クロックが必要で、レイテンシも 5 クロックになります。)パックド・データ用の新しい SP 加算器はポート 1 上に実装されています。この加算器は 1 クロックで同時に 2 つの SP 値を処理できます。レイテンシは 3 クロックです。比較命令、減算命令、最小/最大命令、および変換命令も、すべてこの加算器で実行されます。この実装で最も重要なのは、ピーク時に完全に 4 ワイドの実行効率を実現できるように、種類の異なるユニット(種類の異なる命令)がポート 0 とポート 1 に振り分けられていることです。

2. データの再構成はハードウェアでサポートされています。SIMD 演算を効率的に実行するには、データを SIMD 演算に適した構成に組み換える必要があります。たとえば、3D データは通常、頂点の 3 次元の座標値 x 、 y 、 z と透視補正因子 w を 1 組にした (x, y, z, w) のフォーマットで表現されます。しかし、一部の SIMD 演算では、各頂点の同種の座標を成分とするベクトル

$(x_1, x_2, \dots), (y_1, y_2, \dots), (z_1, z_2, \dots), (w_1, w_2, \dots)$ の形式でデータを表した方が演算の実行効率が高くなります。インターネット SSE には、これらのデータ表現タイプの相互変換をサポートするため、データ操作命令のセットが含まれています。データ操作命令の活用により、データの再構成に費やされる実行時間が短縮されるため、開発過程では、FP ユニットの使用効率を高めるための有効な手段として、これらの命令をハードウェアによって効率的にサポートすることが重視されました。そして、このサポートを実現するために、新たにシャッフル/論理ユニットが実装されました。シャッフル/論理ユニットはポート 1 を他のユニットと共有し、上位アンパック、下位アンパック、移動、および論理演算のマイクロオペレーションを実行します。128 ビットのシャッフル・オペレーションは、(1)コピー・テンポラリ、(2)下位シャッフル、(3)上位シャッフル、の 3 つのマイクロオペレーションを使用して実行されます。さらに、シャッフル・ユニットは、FP シャッフル・ユニットの共有により、パックド整数シャッフル命令の PINSRW および PEXTRW も実行します。

3. データのコピー操作が高速化されました。IA-32 命令は、その命令のオペランドの 1 つを上書きします。こ

の仕様により、コード内での MOVE 命令の使用回数は必然的に多くなります。例えば、次のような処理を想定してください。

メモリ・オペランド A を XMM0 レジスタにロードする

XMM0 の値をメモリ・オペランド B で乗算する

2 番目の命令が実行されると、XMM0 レジスタの内容は上書きされます。したがって、後続のコードが同じメモリ・オペランド A の値を使用する場合は、そのデータをメモリから再ロードするか(この操作は、マルチメディア・アルゴリズムで多用されるロード・ポートの負荷をさらに増加させる原因になります)、あるいは、後で再利用するために、XMM0 の内容をあらかじめ別のレジスタにコピーしておく必要があります。この後者の方法を効率化するため、Pentium III プロセッサには 2 つの移動ポートが実装されています。

例外処理

2 つの 64 ビット・マイクロオペレーションによって実装された 128 ビット演算では、独立した 2 つのマイクロオペレーション(μ OP)のどちらか一方で例外が発生した場合の問題を考慮する必要があります。たとえば、第 2 μ OP で例外が発生している間に第 1 μ OP がリタイアすると、アーキテクチャ上は単一のレジスタとして扱われる 128 ビット・レジスタが部分的にしか更新されず、システムのアーキテクチャ・ステートに矛盾が生じます。リタイアとは、演算結果をアーキテクチャ・ステートにコミットする操作のことです。Pentium III プロセッサでは、「厳密な例外処理」を実現するため、第 2 μ OP での例外発生時に第 1 μ OP がリタイアすることを防ぐ CNU (Check Next Micro-Operation)メカニズムが実装されました。このメカニズムは次のように動作します。まず、 μ OP のペアのうち、アトミックな処理、またはアトミックなデータ・タイプ、またはその両方として扱う必要のある第 1 μ OP に、CNU フロー・マーカ

のマークが付けられます。命令デコーダは、その CNU マーカをデコードし、CNU ビットをアロケータに送ります。アロケータは、この μ OP に割り当てられたリオーダー・バッファ(ROB)エントリの中の CNU ビットをセットするように ROB に通知します。ROB には、プログラムのコードで指定された本来の順序で、複数の μ OP が格納されます。ROB は、第 2 μ OP がリタイア可能であることが確認される時点まで、第 1 μ OP のリタイアを遅らせます。この CNU メカニズムは必然的にリタイアの遅延を伴うので、最適化を図るために、例外がすべてマスクされている場合は各 μ OP の独立したリタイアが許可されるようになっていきます。通常、マルチメディア・ソフトウェアでは例外がマスクされるため、実用上、演算スループットの低下はほとんど問題になりません。

また、高速な演算スループットを維持するために、マスクされた例外を高速処理するハードウェアも実装されています。マルチメディア・アルゴリズムの実行では、オーバーフロー、0 による除算、ゼロフラッシュ(flush-to-zero)アンダーフローなど、マスクされた例外がしばしば発生しますが、これらの例外は、改良された丸め/ライトバック・マルチプレクサのハードウェアによって高速に処理されます。

キャッシュ制御機能の実装

このセクションでは、メモリ実装の主要な変更点について説明します。インターネット SSE 命令セットでは、新たにキャッシュ制御機能が導入されました。また、バイト・マスク書き込み、ストリーミング・ストア、データ・プリフェッチ、マルチプル WC バッファ、およびストア・フェンスの操作もサポートされています。

これらの機能は、Pentium® III プロセッサにおけるプリフェッチの実装の一部として実現されています。図 4 は、Pentium® II のロード命令でキャッシュ・ミスが発生した場合の、回避不能なパイプライン・ストールの影響を示したものです。

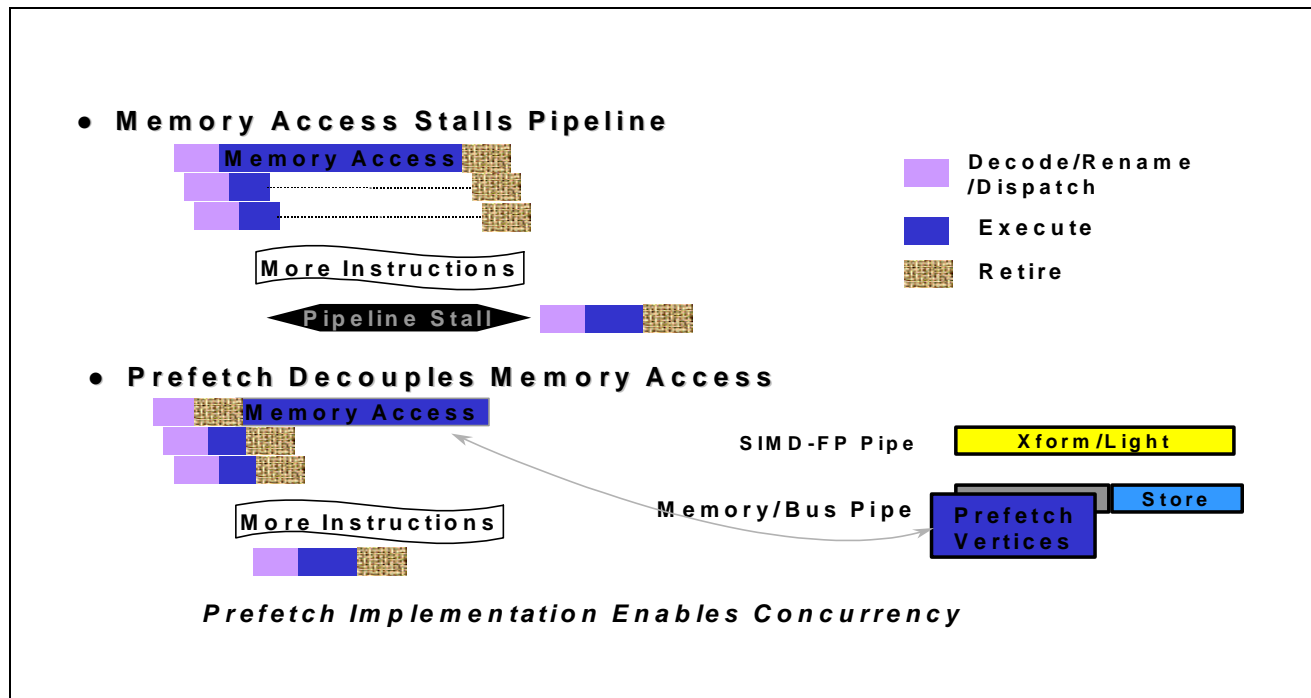


図4: プリフェッチの実装

図 4 の上側に示したメモリ・アクセスによるパイプライン・ストールは、ロード命令のリタイアがその直前の命令よりも大幅に遅れたことが原因で発生しています。ロード命令の後続の命令は、実行はされますが、ロード命令がデータを返すまではユニットからリタイアできません。「Memory Access Stalls Pipeline」の図は、この状態を示しています。メモリ・アクセスに続く命令は、実行後、直ちにはリタイアはできずに、メモリ・アクセスが終了するのを待たなければなりません。したがって、リタイアできないこれらの命令がマシン内に蓄積し、やがて、リタイアしていない命令を管理するための ROB など、プロセッサの重要なリソースの一部が飽和します。そして、この状態では新しい命令の実行に必要なリソースが得られないため、プロセッサのフロント・エンドはすぐにストールしてしまいます。しかし、Pentium III プロセッサでは、プリフェッ

チ命令のリタイアをパイプの前段へ大きく移動させることにより、このプリフェッチ命令のボトルネックが解消されました。この効果を示したのが、図 4 下側の「Prefetch Decouples Memory Access」の図です。この図からわかるように、メモリ・アクセス(この図では Prefetch)の後続の命令は、メモリ・アクセス自体の実行が完了する前でもリタイアできるようになっています。このプリフェッチの実装では、たとえキャッシュ・ミスが発生した場合でもプリフェッチ命令は即座にリタイアするため、メモリ・アクセスが原因でリタイア待ちやリソースの飽和が起こることはありません。その結果、演算処理とメモリ・アクセスの並行実行性が大幅に向上しました。メモリ・アクセスによるストールを回避するための上記のような概念はシニア・ロードと呼ばれます。図 5 に、シニア・ロードの概要を示します。

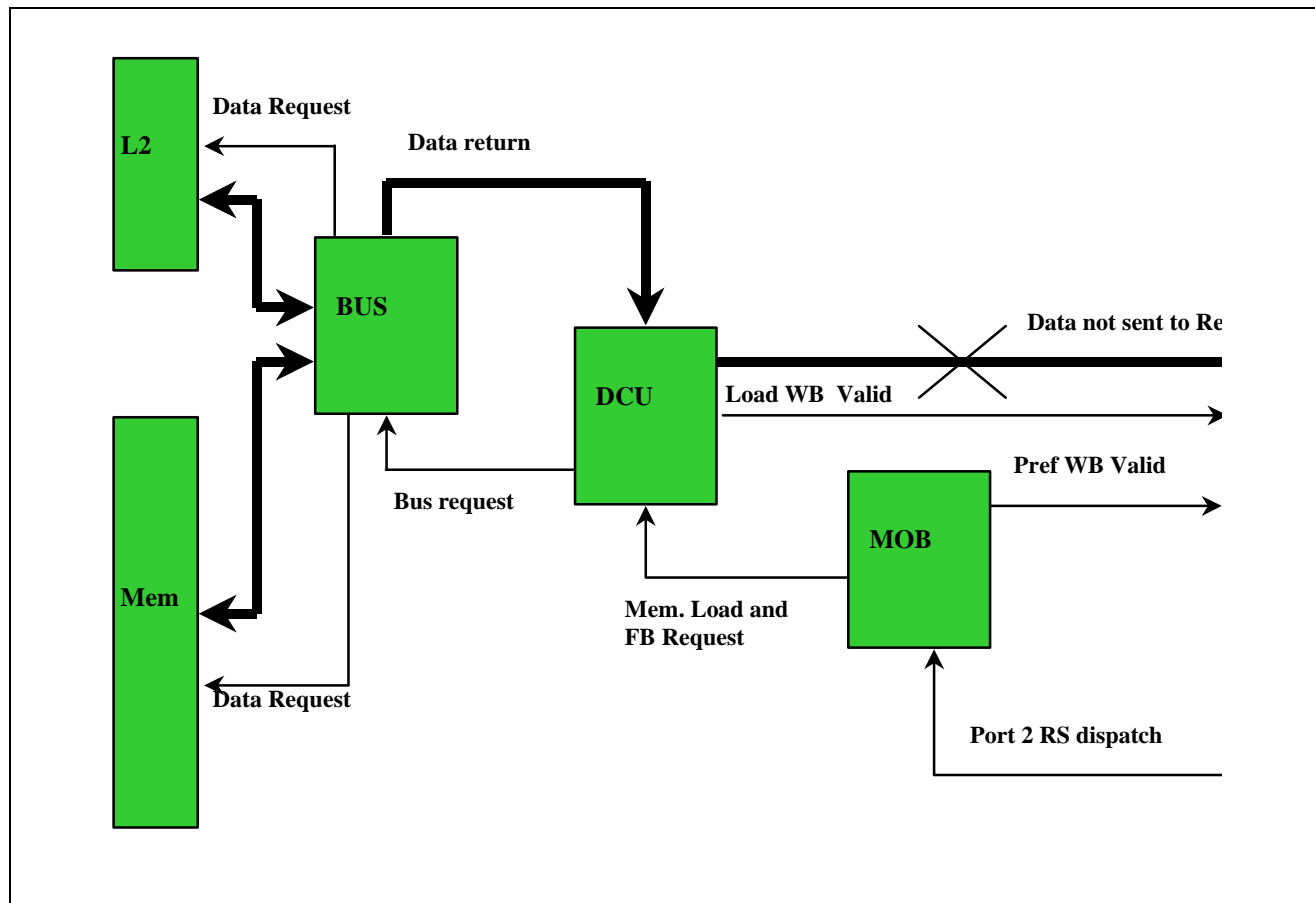


図5: シニア・ロードの実装

図5を見ると、ロード命令とプリフェッチ命令では、リタイア可能な状態を示す信号の伝達方法がどのように異なるのかがわかります。ロードの場合、命令は、リザーベーション・ステーション(RS)からのディスパッチ後、メモリ・パイプヘディスパッチされます。データ・キャッシュ・ユニット(DCU)に目的のデータが存在しない場合、命令はさらにBUSユニットによってディスパッチされます。L2またはバスからデータが返されると、ロードが完了したという信号(Load WB 有効)が送られ、そのロード命令とすでに実行が完了した後続の命令のリタイアが許可されます。一方、プリフェッチの場合は、命令が MOB にアロケートされた直後に、完了を示す信号(Pref WB 有効)が送られます。すなわち、命令は、データがメモリ・サブシステムから実際にフェッチされるのを待たずに完了します。このように完了の信号を直ちに送ることにより、命令の迅速なリタイアが可能になり、ロード命令の場合よりも早い時点で後続の命令が実行できるようになります。そのため、メモリ・アクセスのレイテンシが原因でリソー

ス・ストールが発生することもなくなります。プリフェッチ命令は、キャッシュ階層のさまざまなレベルにデータをフェッチできます。また、ストリーミング・ストア命令もサポートされています。アプリケーションでは、ストリーム・データをメモリから連続的に読み込み、それを一切変更することなく1度だけ使用して破棄する場合も少なくありません。その場合は、通常のキャッシュ・モデルを使用すると、キャッシュ内の有用なデータが上書きされてしまう危険性が高くなります。キャッシュ管理によってパフォーマンスを向上させる方法については、「プログラミング・モデル」のセクションで詳しく解説します。

次に、書き込みに関する問題を取り上げます。書き込み操作に関する主な問題の1つは、ビデオや3Dグラフィックスなどの多くのアプリケーションにおいて、キャッシュに収まらない大きなワーキング・セットが使用されることです。そのようなアプリケーションでは、むしろキャッシュをバイパスして使わない方がパフォーマンスが向上する可

能性があります。つまり、データをメモリ上に保持して、書き込み操作も直接メモリに対して実行した方が望ましいこととなります。そのような場合には、ストリーミング・ストア命令を使用するのが最適です。

実装によって高速な書き込みを実現することは、広帯域メモリ読み出しの実現と同様に重要です。Pentium III プロセッサの書き込みの実装では、特に、書き込みの広帯域化とライト・コンバインの割り当て/入れ替えの効率化が重視されました。バス書き込みの帯域幅は従来より 20%向上しました。Pentium III プロセッサでは、800MB の書き込みによって 100MHz バスの能力を余すところなく活用できます。このような高速性は、ライト・コンバインによる連続的な書き込みの間の無駄なクロック・サイクルを省くことによって実現されました。

また、この広帯域の書き込みをサポートするために、書き込みバッファを割り当てるメカニズムも改良されました。ただし、Pentium III プロセッサでも Pentium II プロセッサと同じフィル・バッファが採用されているため、Pentium III と Pentium II のバッファの相違については、より厳密に理解する必要があります。両者の違いは、まさに SSE の本質に関わるものです。Pentium III プロセッサよりも前の世代のアーキテクチャは、主としてスカラ・アプリケーションの実行性能を重視したものであり、フィル・バッファの目的は、スカラ・アプリケーションにおけるキャッシュ・ミスのバーストに対応できる瞬間的な高スループットを実現することにあります。したがって、一瞬のピーク時の高速性は重要でしたが、時間平均での帯域幅に対する要求は 100MB/秒程度の比較的低いものでした。一方、SSE が対象とするのはベクトル・アルゴリズムなどを使用したストリーミング・アプリケーションであり、SSE バッファの目的は、平均的に高いスループットを維持することにあります。総合的な(リード+ライトの)スループット性能で見れば、Pentium II プロセッサのフィル・バッファの能力でも Pentium III プロセッサのタイムフレームに十分対応できます。しかし、Pentium III プロセッサで使用するには、そのバッファ性能の使用効率をさらに高める必要がありました。そこで、アロケーション・ポリシーの見直しが行われ、複数の書き込みバッファの同時使用と、バッファの迅速な開放による占有時間の短縮が実現されました。次のパラグラフでは、バッファの開放の迅速化と複数バッファの効率的な活用を実現する方法について説明します。

Pentium III プロセッサのライト・コンバインの実装は、メモリ・クラスタにおいて、すべてのフィル・バッファをライト・コンバイン用のフィル・バッファとして同時に活用できるようにする方法で実現されています。Pentium II プロセッサでは 1 度に 1 つのフィル・バッファしか使用できませんでした。この機能拡張をサポートするため、Pentium III プロセッサでは、従来の Pentium® Pro ライト・コンバイン入れ替えポリシーに加えて、次のような新しいライト・コンバイン入れ替え条件が実装されています。

- フィル・バッファのすべてのバイトが書き込まれると(全てダーティになると)、バッファの内容はメモリへ書き込まれます。「リソース要求」に基づく従来のバッファ入れ替えポリシーでは、DCU が新しいバッファの割り当てを要求するまでバッファは開放されませんでした。
- すべてのフィル・バッファが使用中の場合、ロードやストア、あるいはフィル・バッファを必要とするプリフェッチなどの実行にともなって DCU がフィル・バッファの割り当てを要求すると、WC バッファは、まだフルになっていない場合でも開放されます。

小型のダイで高速動作を実現する効率的な実装

Pentium III プロセッサの開発では、ダイ・サイズに関する厳しい制約のもとで目標の動作周波数を実現するために、さまざまな条件のトレードオフが考慮されました。例えば、次のような実装上の選択も、そのようなトレードオフの結果です。

- Pentium III プロセッサでは、x87 乗算器はパックド FP 乗算器に統合されました。この統合は、ダイ・サイズの小型化に大きく貢献しているだけでなく、ポートのロード負荷を低く抑える意味でも重要です。ライトバック・バス上のロードの問題は、開発における重要な要素の 1 つでした。ライトバック・バスは従来の実装でも速度上、かなりのボトルネックとなっていたため、インターネット SSE の実装による新しいユニットや演算ロジックの追加によって問題がさらに悪化する懸念がありました。そのため、開発のプロジェクトの発足当初から、この問題に対する重点的な取り組みが図られ、乗算器の統合が実現しました。なお、x87 加算器とパックド FP 加算器の統合も検討されましたが、こ

ちらは開発スケジュールの関係で実現には至りませんでした。

- 乗算器の Wallace ツリーを利用して PSAD 命令が実装されました。パックド MMX 値の絶対差の和を計算する PSAD 命令は、差の計算、絶対値の計算、および絶対値の和の計算という 3 つのマイクロオペレーションによって実装されています。従来の実装でも、絶対値の和の計算は、乗算器の Wallace ツリーにおいて、通常は乗算の部分積の和を計算するツリーをバイト・データの加算に利用する方法で実行されてきました。PSAD 命令は、このロジックを再利用する形で実装されているため、実装によるダイ面積や動作速度への影響が非常に小さくなっています。仮に、演算のパフォーマンスを損なわずにこの命令を別の方法で実装しようとするれば、大幅なコストの上昇が避けられないでしょう。

開発の成果

このセクションでは、Pentium III プロセッサの開発における 2 つの主要な成果の概要を示します。その 1 つは、2 つの 2 ワイド演算ストリームの並行ディスパッチによる 4 ワイド命令セット・アーキテクチャ(ISA)の実装、もう 1 つは、演算ストリームとメモリ・アクセス・ストリームの独立実行の実現です。

2 ワイド・データ・パスを使用した 4 ワイド命令セット・アーキテクチャは、ダイ・サイズとパフォーマンスのトレードオフにおいて、両者の条件をともに満たすバランスの良い実装を実現します。パフォーマンスの観点からは、2 ワイド・データパスによって 4 ワイド ISA を実装するこのアプローチと、2 ワイド・データパスによる 2 ワイド ISA の実装との違いが問題になります。両者の違いは、4 ワイド ISA のレジスタ数が 2 ワイド ISA の 2 倍であることに起因します。複数の命令を含むループで 8 つのレジスタが使用される場合を考えてみてください。この場合、4 ワイド ISA でコーディングされたループは、2 ワイド ISA でコーディングされたループを 2 回アンロールしたのちソフトウェアによってパイプライン化したものと同じと考えられます。一般に、ループの明示的なアンロールはパフォーマンスの向上につながります。特にアウトオブオーダー・アーキテクチャでは、アンロールによってコードの並行性がさらに顕著になるため、より一層のパフォーマンスの向上が期待できます。同じループを 2 ワイド ISA で実行した場合、

レジスタはすべてループ自体を実行するのに使用されてしまうため、アンロールは実現できません。したがって、4 ワイド ISA では、内部的なアウトオブオーダー実行とループの明示的なアンロールの 2 つの効果によるパフォーマンスの向上が実現できるのに対して、2 ワイド ISA の場合は、内部的なアウトオブオーダー実行による効果しか期待できないこととなります。

ストリーミング・アーキテクチャの主要な目的は、データ・ストリームの並行処理の実現により、マルチメディア・アプリケーションのパフォーマンス要求を満たすことにあります。実装の観点から言えば、これは演算ストリームとメモリ・アクセス・ストリームの並行実行を実現する必要があることを意味します。

P6 マイクロアーキテクチャは、メモリ・アクセスと演算実行を並行化する能力を備えています。明示的なプリフェッチ命令の実装は、その並行実行性をさらに向上させ、データのフェッチと、後続命令のリタイアを完全に分離することを可能にしました。その結果、各ストリームでは、それぞれのタスクでの理論的な最大値に近いスループットが実現できるようになっています。つまり、あるタスクに関して Pentium® III プロセッサで実現可能なスループットの最大値は、そのタスクの演算スループットの最大値とメモリ・スループットの最大値の、どちらか小さい方の値に等しいと言えます。

このようにメモリの実質的なスループットが向上したため、プロセッサのバッファ・サブシステムのスループットとバススループットのバランスを取ることが必要になりました。しかし、そのために新しいバッファを実装することはせず、既存のバッファの使用効率と外部バスの書き込みのスループットを高める方法を採用しました。その結果、ダイ・サイズにはほとんど影響を与えることなく、メモリ・データパスのパフォーマンスの均衡を図ることができたのです。

プログラミング・モデル

Pentium® III プロセッサの開発と並行して、その実装がもたらす優れたパフォーマンスを実用アプリケーションで活用するためのプログラミング・モデルも開発されました。このセクションでは、3 つのタイプのマルチメディア・アプリケーションを取り上げ、各タイプに適したプログラミング・モデルの概要を説明します。

1. **AC3 オーディオなど、演算処理性能が要求されるアプリケーション** -- このタイプのアプリケーションは、高速な演算スループットを必要としますが、メモリの帯域幅に対する要求はそれほど高くありません。Pentium III プロセッサでは、このタイプのアプリケーションは高速な浮動小数点ユニットによってサポートされます。この浮動小数点ユニットの演算能力を有効に活用するため、実際のプログラミングでは、参考資料[2]で説明されている SSE 最適化ツールの使用が推奨されます。
2. **3D 画像処理など、メモリ・アクセス性能が要求されるアプリケーション** -- このタイプのアプリケーションの顕著な特徴は、ワーキング・セットのサイズがかなり大きいことです。そのため、演算処理性能が要求されるアプリケーションの場合ほどキャッシュが有効に機能せず、データはメモリ上に置かれたままのことが多くなります。したがって、一部のケースでは、キャッシュは使用せずにバイパスし、並行実行によるスループットの高いメモリ・アクセスを活用してメモリ上のデータに直接アクセスした方がパフォーマンスが向上する場合があります。このような高速メモリ・アクセス機能を十分に活かすため、ソフトウェアの開発では、データの入力ストリームと出力ストリームを明確に識別した上で、各ストリームに対する処理をプリフェッチ命令とストリーミング・ストア命令を使用して適切にプログラミングすることが求められます。これらの命令を使用すれば、内部キャッシュによる過剰な処理が発生せず、ストリームのフェッチ/ストアの操作がメモリに対して直接実行されます。このテクニックについては、参考資料[3]のオンライン・ドライバによる 3D ジオメトリ処理に関するセクションでも詳しく説明されています。

データ・ストリームの先頭では、プリフェッチを有効に活用できないことが原因でメモリ・アクセスのパイプライン実行が滞る場合があります。そのため、プリフェッチの操作では、パイプラインが途切れる回数を少しでも減らせるように、データ・ストリームの長さを最適化するテクニックも使用されます。このプログラミング・モデルの詳細については、参考資料[3]のマルチプリミティブ API による描画の説明を参照してください。

参考資料[3]では、3D グラフィックス処理における、このようなプログラミング技法の有効性が詳しく解説されています。実際に、このプログラミング・モデルを実装したソフトウェアでは、アプリケーション・レベルで2倍の高速化が達成されました。

ビデオ・エンコードなど、上記の2つの特徴を併せ持つ中間的なタイプのアプリケーション -- このタイプに含まれるのは、通常、キャッシュが有効な比較的小さいワーキング・セット、およびキャッシュに収まりきらない大きなワーキング・セットを両方使用するアプリケーションです。したがって、実装のサポートやプログラミング・モデルの面でも、上述した2つのタイプに適した技法を組み合わせるのが有効だと言えます。このタイプのアプリケーションでは、何度も再利用されるワーキング・セットと使用頻度の少ないワーキング・セットを分離し、各ワーキングセットの再利用の頻度に基づいたキャッシュの活用方法を確立することが重要です。

たとえば、MPEG-2 エンコーダでは、図6に示したように、フレーム内符号化画像(I-フレーム)の色データと輝度データ、双方向予測用画像(B-フレーム)の色データと輝度データ、およびダウンサンプルされたデータが処理されます(参考資料[4]参照)。

Encoder Tuning

Caching

- Encoder re-uses frequently only portion of data
- Use this feature to control caching strategy

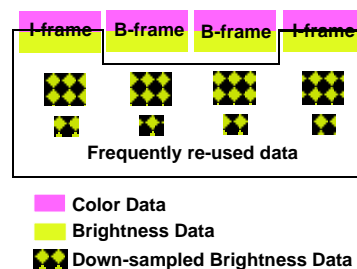


図6: MPEG-2 エンコーダにおけるデータの再利用

MPEG-2 のエンコード・アルゴリズムの仕様上、I-フレームの輝度データとダウンサンプル・データは、処理される回数が I-フレームの色データおよび B-フレームのデータよりも数回多くなっています。したがって、このアプリケーションでは、前者のデータはキャッシュ内に保持し、後者

のデータはメモリ上に置くのが賢明なキャッシュの活用法と言えます。図 7 は、2 種類のキャッシュの使用法におけるキャッシュ/メモリ階層へのデータ配置の違いを示したものです。図の左側は、従来の IA-32 のキャッシュ機能による非制御のキャッシング、右側は、Pentium III プロセッサのインターネット SSE に含まれるストリーミング・ストア命令とプリフェッチ命令を使用して実現される、ソフトウ

エア制御によるキャッシングを示しています。ソフトウェア制御によるキャッシングでは、I-フレームの色データと B-フレームのデータはメモリ上に置かれたままですが、メモリに対するプリフェッチ/ストア命令の高速なスループットにより、データ・フェッチのレイテンシを隠蔽できるようになっています。

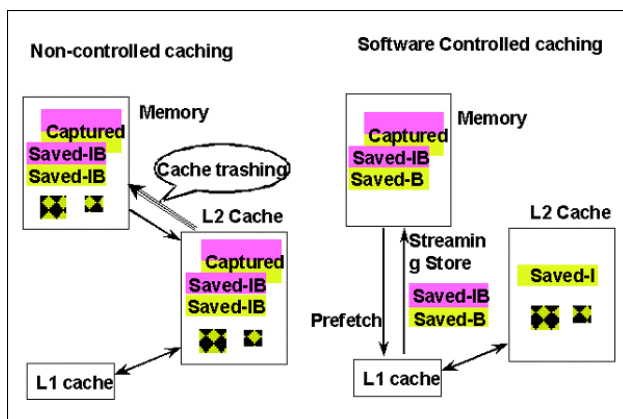


図 7: MPEG-2 エンコーダのソフトウェア制御キャッシュと非制御キャッシュの違い

Pentium III プロセッサは、このキャッシュ・モデルと PSAD 命令(動き評価アルゴリズムの実行効率を高める)の組み合わせにより、ソフトウェアによるリアルタイム MPEG-2 エンコードを実現できるパフォーマンスを達成しています。

まとめ

Pentium[®] III プロセッサには、550MHz 版をはじめ、動作周波数の異なるいくつかの製品が用意されています。Pentium III プロセッサには 70 の新しい命令が追加されましたが、その実装はダイ・サイズに関して僅か 10%程度のコスト増で実現されました。この記事で説明したさまざまな機能により、Pentium III プロセッサでは、マルチメディア・パフォーマンスの水準が格段に向上しています。さらに、4 ワイド・インターネット SSE や、演算ストリームとメモリ・ストリームの並行実行機能が比較的簡明な形で実装されていることも重要な特徴の 1 つです。これにより、SSE アプリケーションでは、将来のさらに高速な動作周波数にも対応できるスケーラビリティが確保されていると言えます。

謝辞

Pentium[®] III プロセッサは、フォルサムのマイクロプロセッサ・プロダクト・グループ(MPG-FM)でアーキテクチャの立案、設計、および検証作業に携わった大勢の開発者たちの努力の成果です。この記事で取り上げた多数の技術は、MPG-FM のアーキテクチャ・グループの手によって実現されました。Pentium[®] III プロセッサのアーキテクチャ定義や実装の実現にあたって指導的な役割を果たしていただいた Srinivas Raman 氏に感謝の意を表します。あわせて、開発中にさまざまな助言をいただいた Dave Papworth、Glenn Hinton、Pete McWilliams、Bob Colwell、Ticky Thakkar、Tom Huff の各氏にも感謝いたします。

参考資料

- [1] Narayanan Iyer, Mohammad Abdallah, S.Maiyuran, Vivek Garg, S.Spangler 著 『Katmai Features POR』、インテル社内ドキュメント
- [2] Joe Wolf 著 『VTune[™]パフォーマンス拡張環境の利用による、Pentium[®] III プロセッサのインターネット・ストリーミング SIMD 拡張命令を活用するためのプログラミング手法』、インテル・テクノロジー・ジャーナル 1999 年第 2 四半期号、http://www.intel.co.jp/jp/developer/technology/itj/q21999/articles/art_6.htm

- [3] Paul Zagacki, Deep Buch, Emile Hsieh, Hsien-Hsin Lee, Daniel Melaku, Vladimir Pentkovski 著『Pentium® III プロセッサのパフォーマンスを最大限に活かす 3D ソフトウェア・スタック・アーキテクチャ』、インテル・テクノロジー・ジャーナル 1999 年第 2 四半期号、
http://www.intel.co.jp/developer/technology/itj/q21999/articles/art_4.htm
- [4] James Abel ほか著『インターネット・ストリーミング SIMD 拡張命令を考慮したアプリケーション・チューニング』、インテル・テクノロジー・ジャーナル 1999 年第 2 四半期号、
http://www.intel.co.jp/developer/technology/itj/q21999/articles/art_5.htm
- [5] David B. Papworth 著『Pentium Pro マイクロアーキテクチャのチューニング』、IEEE Micro 1996
- [6] Intel 編、『Pentium Pro Family Developer's Manual』、インテル・マニュアル
- [7] Ticky Thakkar ほか著『インターネット・ストリーミング SIMD 拡張命令』、インテル・テクノロジー・ジャーナル 1999 年第 2 四半期号、
http://www.intel.co.jp/developer/technology/itj/q21999/articles/art_1.htm

著者紹介

Jagannath Keshava、テキサス大学オースチン校でコンピュータ工学の修士号を取得。1990 年にインテルに入社し、その後、Pentium® II および i960® マイクロプロセッサ・グ

ループにおいて、設計、マイクロアーキテクチャ定義、および検証作業に従事。Pentium® III プロセッサの開発では、定義チームおよびマイクロアーキテクチャ検証チームのリーダーを務めた。現在は、フォルサムにあるマイクロプロセッサ・プロダクツ・グループのアーキテクチャ・グループに所属し、パリュウ PC 向けの統合マイクロプロセッサのアーキテクチャ定義を担当。

電子メール・アドレス: jagannath.keshava@intel.com

Vladimir Pentkovski、フォルサム支社、マイクロプロセッサ・プロダクツ・グループ(MPG)の首席エンジニア。IA-32 アーキテクチャのインターネット・ストリーミング SIMD 拡張命令の定義を手がけた中心チームの一員。Pentium III プロセッサ・アーキテクチャの開発とそのパフォーマンス分析において主導的役割を果たす。以前は、ロシアで Elbrus マルチプロセッサ・コンピュータ用プログラミング言語向けのコンパイラ開発とソフトウェア/ハードウェア・サポートに従事。ロシアでコンピュータ・サイエンスとエンジニアリングの理学博士号および Ph.D. を取得。

電子メール・アドレス: vladimir.m.pentkovski@intel.com