インテル® Parallel Composer 入門ガイド

本ガイドは、インテル® Parallel Composer を使用して、スレッドの導入、コンパイル、デバッグを行う方法を説明します。簡単なコードサンプルを使用して、以下の作業を行います。

- インテル® Parallel Composer を起動する。
- アプリケーションをスレッド化する。
- インテルのライブラリーをプロジェクトで使用する。
- インテル® Parallel Debugger Extension を使用してコードのデバッグを開始する。

本ガイドは、開発アプリケーションのパフォーマンス向上目標に最も適したスレッド化手法の選択に役立つ一般的なシナリオを紹介します。また、より多くのリファレンス情報を入手するための手段も提供します。

注: インテル® Parallel Composer のビデオデモ (Show Me) をご利用いただけます。ビデオデモ (Show Me) を表示するには、Adobe* Flash* Player が必要です。

目次

1 サンプルを開く

インテル® Parallel Composer では、以下の場所にサンプル・アプリケーションが用意されています。

<install-dir>\Samples\<locale>\C++\

C:\Program Files\Intel\Parallel Studio\Composer\Samples\<locale>\C++*<solution name>*.zip

1. *<solution name>*.zip を任意のディレクトリーに解凍します。
2. .sln ファイルをダブルクリックするか、または Microsoft* Visual Studio* 開発環境からソリューション・ファイルを開きます。

この付属のサンプルを使用して、下記の手順に従ってください。

2 インテル® Parallel Composer の起動

インテル® Parallel Composer は以下の環境に統合されます。

- Microsoft Visual Studio 2008
- Microsoft Visual Studio 2005

インテル® Parallel Composer を起動するには、次の手順を行います。

1. Microsoft Visual Studio を起動します。
2. **[ソリューション エクスプローラ]** ペインで、付属の .sln ファイル (例: **par-OpenMP.sln**) を開きます。

注: 下記の手順で示されているインテル® Parallel Composer のメニューコマンドは、ツールバーからも利用できます。

3. Microsoft Visual Studio のメインメニューから、**[プロジェクト] > [Intel Parallel Composer (インテル(R) Parallel Composer)] > [Use Intel C++ (インテル® C++ を使用)]** を選択します。

4. **[Confirmation (確認)]** ダイアログボックスで、**[Yes (はい)]** をクリックします。これにより、ソリューションでインテル® C++ コンパイラーが使用されるようになります。
5. Visual Studio の **[ビルド]** メニューから **[ソリューションのリビルド]** を選択します。

コンパイル結果が **[出力]** ウィンドウに表示されます。

アプリケーションのビルドについての詳細は、インテル® Parallel Composer オンラインヘルプを参照してください。オンラインヘルプの表示方法については、「ユーザー・リファレンス・ドキュメント」を参照してください。

3 アプリケーションのスレッド化

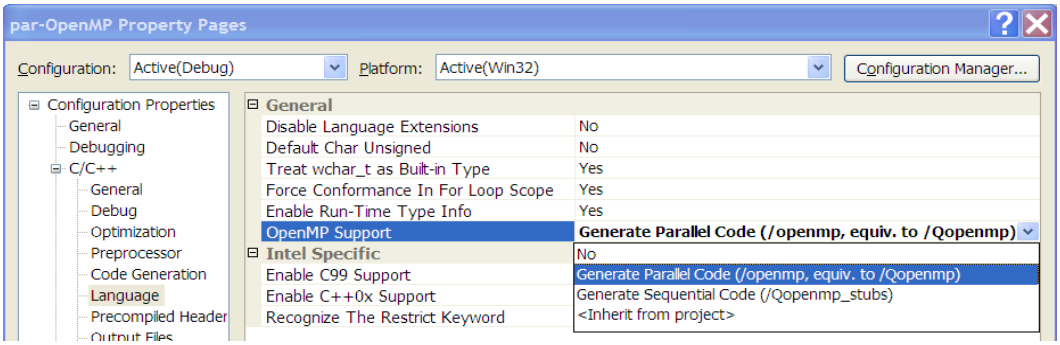
このセクションでは、インテル® C++ コンパイラーとインテルのライブラリーを使用して、アプリケーションを並列化する方法を説明します。簡単なコードサンプルで、各種の並列化手法を実装する方法を示します。

OpenMP* によるループの並列化

OpenMP 宣言子、またはインテル® C++ コンパイラー言語拡張機能を使用してアプリケーションをスレッド化することができます。

操作:

1. Microsoft Visual Studio で、**par-OpenMP.sln** または **par-LangExtensions.sln** を開きます。
2. メインメニューから、**[プロジェクト] > [プロパティ]** を選択します。
3. **[プロパティ ページ]** ウィンドウで、**[C/C++] > [Language (言語)] > [OpenMP Support (OpenMP サポート)]** を選択します。
4. ドロップダウン・メニューから **[Generate Parallel Code (/openmp, equiv. to /Qopenmp) (並列コードの生成 (/openmp, equiv. to /Qopenmp))]** を選択します。



5. **[適用]** をクリックします。

OpenMP 診断を有効にする

1. **[プロパティ ページ]** の **[Command Line (コマンドライン)] > [Additional Options (追加のオプション)]** で `/Qdiag-enable:openmp` を追加します。
2. **[OK]** をクリックします。
3. ソリューションをリビルドします。

結果:

ループが正常に並列化されると、出力ウィンドウに「OpenMP DEFINED LOOP WAS PARALLELIZED (OpenMP 定義のループが並列化されました)」というメッセージが表示されます。

サンプルコードの説明:

サンプルコードを確認すると、`omp parallel for` 宣言子が次の `for` ループを複数のスレッドで並列実行するよう指定していることがわかります。

```
void f sum ( size t length, const float *a, const float *b,
float *c ) {
#pragma omp parallel for
        for (int i=0; i<length; i++)
                c[i] = a[i] + b[i];
}
```

C++ 言語キーワードの使用

`__par` キーワードを文のプリフィックスとして `for` 文の直前に使用し、上記コードの `for` ループを並列化することができます。`__par` キーワードは、OpenMP 宣言子の `#pragma omp parallel for` に相当します。

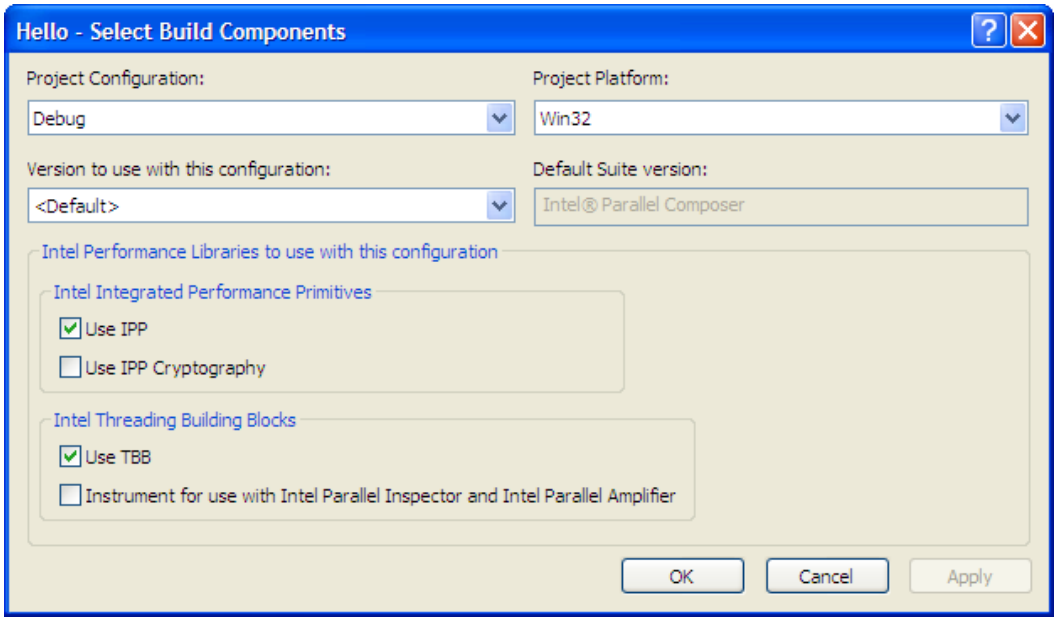
```
void f sum ( size t length, const float *a, const float *b,
float *c ) {
        __par for (int i=0; i<length; i++)
                c[i] = a[i] + b[i];
}
```

インテル® IPP とインテル® TBB ライブラリーの使用

インテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP) またはインテル® スレッディング・ビルディング・ブロック (インテル® TBB) を使用してアプリケーションをスレッド化できます。

1. **[ソリューション エクスプローラ]** ペインでプロジェクトを開きます。
2. メインメニューから **[プロジェクト]** > **[Intel Parallel Composer (インテル(R) Parallel Composer)]** > **[Select Build Components (ビルド・コンポーネントの選択)]** を選択します。

[Select Build Components (ビルド・コンポーネントの選択)] ダイアログボックスが開きます。



ダイアログボックスで、インテルのライブラリーを 1 つまたは両方を選択することができます。

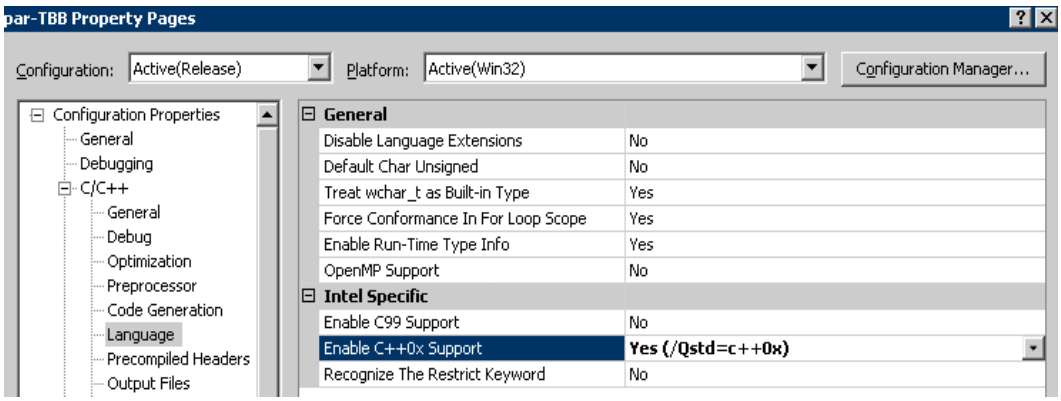
操作	結果
[Use IPP (IPP を使用)] チェックボックスをオンにします。	インテル® IPP の一般的なライブラリーのみが有効になります。
[Use IPP Cryptography (IPP 暗号化を使用)] チェックボックスをオンにします。	インテル® IPP の暗号化ライブラリーとインテル® IPP の一般的なライブラリーの両方が有効になります。

[Use IPP (IPP を使用)] チェックボックスをオフにします。	インテル® IPP ライブラリーが無効になります。
[Use TBB (TBB を使用)] チェックボックスをオンにします。	インテル® TBB ライブラリーが有効になります。
[Use TBB (TBB を使用)] チェックボックスをオフにします。	インテル® TBB ライブラリーが無効になります。
[Instrument for Use with Intel Parallel Inspector and Intel Parallel Amplifier (インテル(R) Parallel Inspector とインテル(R) Parallel Amplifier 用にインストルメント)] チェックボックスをオンにします。	プリプロセッサー定義 TBB_USE_THREADING_TOOLS をプロジェクトのプロパティー (**[Preprocessor (プリプロセッサー)] > [Preprocessor Definitions (プリプロセッサー定義)]**) に追加します。 ***注:*** **[Instrument for Use with Intel Parallel Inspector and Intel Parallel Amplifier (インテル(R) Parallel Inspector とインテル(R) Parallel Amplifier 用にインストルメント)]** は、プロジェクトのプロパティーで **[Code Generation (コード生成)] > [Runtime Library (ランタイム・ライブラリー)]** が **[Multi-threaded DLL (/MD) (マルチスレッド DLL (/MD))]** に設定されている場合のみ表示されます。

[OK] または **[適用]** をクリックすると、選択されたアーキテクチャーに適切なインテル® IPP およびインテル® TBB の `include` パスと `lib` パスがプロジェクトのプロパティー設定に追加されます。

インテル® TBB によるループの並列化

1. **par-TBB.sln** ソリューションを開きます。
2. インテル® TBB を有効にします (前のセクションの説明を参照してください)。
3. メインメニューから、**[プロジェクト] > [プロパティ]** を選択します。
4. **[プロパティ ページ]** ウィンドウで、**[C/C++]> [Language (言語)] > [Intel Specific (インテル固有)] > [Enable C++0x Support (C++0x サポートを有効にする)]** を選択します。
5. ドロップダウン・メニューから [Yes **(Qstd=c++0x) (はい (Qstd=c++0x))]** を選択します。



サンプルコードの説明:

以下のコードは、プログラム実行時間のタスク・スケジューラーを初期化します。

```
tbb::task_scheduler_init init;
```

反復空間は `size_t` 型で、`0` から `length-1` までです。テンプレート関数 `tbb::parallel_for` は、反復空間を分割します。サンプルで使用されている `parallel_for` の署名は、次のようになります。

```
template<typename Range, typename Body>
parallel_for( const Range& range, const Body& body,
              const auto_partitioner&);
```

このループを並列化するには、まずループ本文を body クラス (関数オブジェクト、または範囲全体を通してチャンク上で動作する `functor`) に変換します。

```
[&](const tbb::blocked_range<size_t> &r){
              for (size_t i=r.begin(); i<r.end(); i++)
```

```
c[i] = a[i] + b[i];
```

[&] はラムダを導入します。& 内は、ラムダの外のローカル変数がローカルコピーの作成によってではなく、ラムダ内の参照によってアクセスされるよう指定します。ラムダ式に関する情報は、「ユーザー・リファレンス・ドキュメント」を参照してください。

スレッド化手法の選択

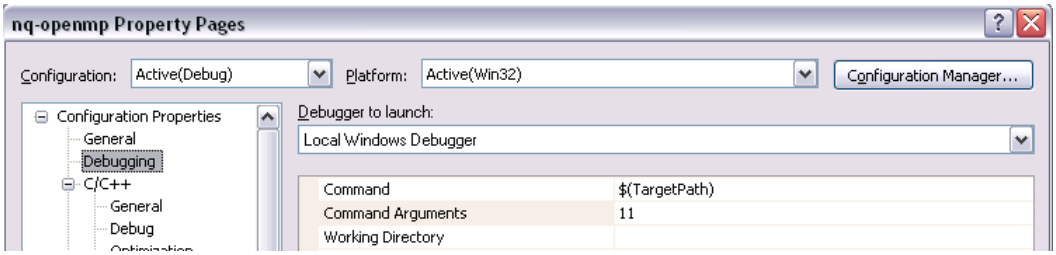
並列化手法	利点	考慮事項
インテル® IPP	高いパフォーマンス、特定のマルチメディア・アプリケーション・ドメインでスケーラブル、コードの並列性を公開しない。	暗号化、画像、音声、ビデオ、圧縮など。マルチメディアに最適です。
自動並列化	自動の並列化。コードの明示的な並列構文を公開しない。	適格なループに限定されます。
新しい並列拡張	ラピッド・プロトタイピングに最適。簡単な構文、OpenMP 3.0 ランタイムにマップ。	OpenMP やインテル® TBB のような shared/private 変数の制御はありません。
OpenMP 3.0	良く知られた仕様。効率的なランタイムで、スケーラブル。task 構造をサポートする OpenMP 3.0 により、関数レベルの並列化を支援。インテルの OpenMP サポートは Microsoft 実装との互換性を確保。	一部の C++ 開発者にとってはプラグマの使用は一般的でありません。C++ オブジェクト/データに対しては難しい可能性があります。Microsoft Visual Studio 2008 は、OpenMP 2.5 をサポートしています。
インテル® TBB	自然な C++ ソリューション、効率的、スケーラブルなランタイム。	C++ オブジェクト指向のプログラミング (テンプレートとコンテナー) の理解が必要。

4 インテル® Parallel Debugger Extension によるコードのデバッグ

デバッグ用サンプルコードの準備

1. **NQueens-ParallelStudio.zip** を任意のディレクトリーに解凍します。
2. Microsoft Visual Studio で nq-openmp ソリューションを開き、**nq-openmp.cpp** をダブルクリックします。

3. 84 行目まで移動し、`#pragma omp atomic` 行をコメントアウトして、データ共有違反を作成します。
4. nq-openmp プロジェクト・プロパティーで、コマンド引数の `11` がプログラムに渡されるよう指定されていることを確認します。

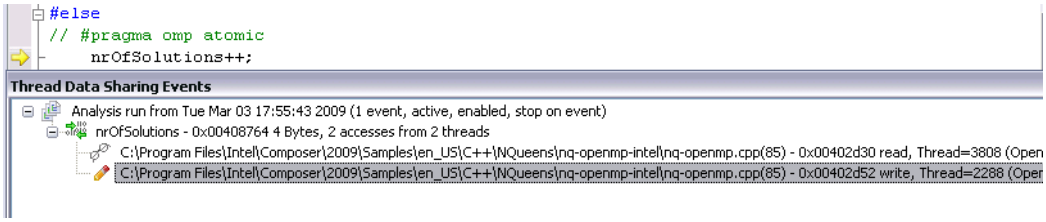


5. コンパイラーのコマンドライン・オプション `/ZI`, `/debug:parallel` と `/Qopenmp` がプロジェクトの **[プロパティ ページ]** で設定されていることを確認します: **[構成プロパティ] > [C/C++] > [General (全般)] > [Debug Information Format (デバッグ情報の形式)]**、**[構成プロパティ] > [C/C++] > [Debug (デバッグ)] > [Enable Parallel Debug Checks (並列デバッグ検証を有効にする)]**、**[構成プロパティ] > [C/C++] > [Language (言語)] > [General (全般)] > [OpenMP Support (OpenMP サポート)]**。
6. リンカーのコマンドライン・オプション、/debug が設定されていることを確認します: **[構成プロパティ] > [Linker (リンカー)] > [Generate Debug Info (デバッグ情報の生成)]**。
7. プロジェクトをリビルドします。

並列コーディングの問題の特定

ランタイムデータの共有違反を引き起こしている可能性のある**スレッドデータ共有イベント**を特定するには、次の操作を行います。

1. **[デバッグ] > [Intel Parallel Debugger Extension (インテル(R) Parallel Debugger Extension)] > [Thread Data Sharing Detection (スレッドデータ共有検出)]** メニュー (または  をクリック) で **[Enable Detection (検出を有効にする)]** を選択します。
2. **[デバッグ]** メニュー (または  をクリック) から **[デバッグ開始]** を選択して、デバッグセッションを開始します。スレッドデータ共有イベントが検出されるとすぐにプログラム実行が停止します。
3. **[Thread Data Sharing Events (スレッドデータ共有イベント)]** ボタン  をクリックして、実行停止のイベントログを参照します。



結果:

[Thread Data Sharing Events (スレッドデータ共有イベント)] ログから、2 つの OpenMP スレッドが **nrOfSolutions** 変数にアクセスしようとしていることがわかります。スレッド 3808 が値を読み取り、スレッド 2288 が値を増分しています。これは、1 つの変数に対して変更、依存する 2 つのスレッドがあるため、N-Queens アルゴリズムで不正な結果を起こすことのある深刻な問題です。**#pragma omp atomic** を使用して、**nrOfSolutions++** をインクリメント・スレッド・セーフにします。

関数の再入可能性イベントを特定するには、次の操作を行います。

1. **[Enable Detection (検出を有効にする)]** ボタン  をクリックして、データ共有イベント検出を無効に設定します。 
2. **[デバッグ] > [Intel Parallel Debugger Extension (インテル(R) Parallel Debugger Extension)] > [Break on Re-Entrant Call (再入可能呼び出しで中断)]** を選択 (または  をクリック) します。
3. 表示されたポップアップ・ウィンドウに「**{,,nq-openmp.exe}setQueen**」と入力します。「**setQueen**」と入力してもかまいません。
4. **[Continue Debug (デバッグを続行)]** をクリックします。
5. 関数の再入可能性を検出すると、**[Re-entrant Call Detected (再入可能呼び出しが検出されました)]** ウィンドウが開き、スレッド ID に関する情報が表示されます。

6. **[OK]** をクリックします。

結果:

再入可能性が検出された関数のエントリーポイントにデバッガー命令ポインターが待機しています。

```
void setQueen(int queens[], int row, int col, int id) {

  for(int i=0; i<row; i++) {
    // vertical attacks
    if (queens[i]==col) {
      return;
```

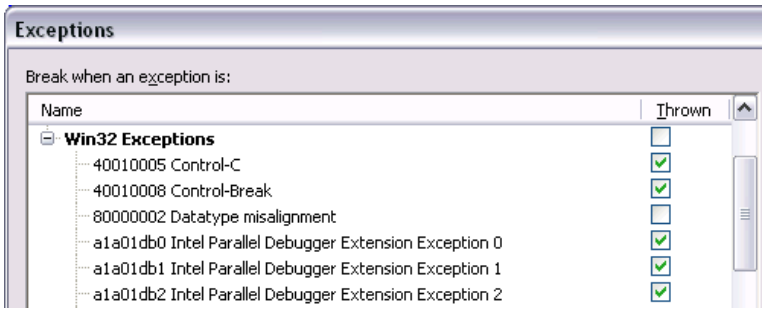
OpenMP タスク 4 と OpenMP タスク 20 の両方とも、`setQueen()` 関数に入っています。両タスクとも値を変更し、この関数内のデータを使用しています。そのため、互いのデータを上書きしないようにしなければなりません。`setQueen()` 関数でランタイム問題が見られるとき、`setQueen()` がスレッドセーフでない場合、このポインターは、スレッドセーフな実装が必要であることを示す非常に重要なポインターとなります。

デバッグに関するその他の考慮事項

Microsoft Visual Studio 2005 を使用している場合、最初にインテル® Parallel Debugger の例外が Microsoft Visual Studio デバッガーで有効に設定されていることを確認してください。

次の操作を行って、例外が有効に設定されていることを確認します。

1. **[デバッグ]** プルダウンメニューから、**[例外...]** を選択します。
2. **[Win32 Exceptions (Win 32 の例外)]** ツリービューで **[Intel Parallel Debugger Extension Exception (インテル(R) Parallel Debugger Extension の例外)]** が有効であることを確認します。



3. **[ファイル]** プルダウンメニューの **[すべてを保存]** を選択します。

5 ユーザー・リファレンス・ドキュメント

本ガイドは、インテル® Parallel Composer の基本機能を紹介しています。より詳細な機能については、次の資料を参照してください。

資料	説明
インテル® Parallel Composer ドキュメント	この HTML ページから、次のインテル® Parallel Composer のドキュメントへリンクされています。 インテル® C++ コンパイラー インテル® スレッディング・ビルディング・ブロック インテル® インテグレーテッド・パフォーマンス・プリミティブ インテル® Parallel Debugger Extension Windows* の [スタート] メニューから、**[Intel Parallel Studio (インテル(R) Parallel Studio)] > [Parallel Studio Documentation (インテル(R) Parallel Studio ドキュメント)] > [Composer Documentation (インテル(R) Parallel Composer ドキュメント)]** をクリックして表示します。
サンプルコード	さまざまなスレッド化手法を学習できるサンプルファイルが <*install_dir*>\ Samples\<*locale*>\C++\ に zip 形式で提供されています。
インテル® IPP スタティック・ライブラリー	ナレッジベース記事 (http://software.intel.com/en-us/articles/intel-ipp-static-libraries/) にインテル® Parallel Composer 用のインテル® IPP スタティック・ライブラリーのダウンロード情報とインストール情報が説明されています
インテル® Parallel Studio の資料	インテル® Parallel Studio は、インテル® Parallel Amplifier、インテル® Parallel Composer、インテル® Parallel Inspector が含まれた並列化のための包括的なツールセットを提供します。 インテル® Parallel Studio の全コンポーネントの概要は、『*インテル® Parallel Studio 入門ガイド*』を参照してください。Windows の [スタート] メニューから、**[Intel Parallel Studio (インテル(R) Parallel Studio)] >[Getting Started (入門ガイド)]>[Parallel Studio Getting Started Guide (インテル(R) Parallel Studio 入門ガイド)]** をクリックして表示します。 インストールされている各インテル® Parallel Studio ツールのドキュメントを開くには、Windows の [スタート] メニューから、**[Intel Parallel Studio (インテル(R) Parallel Studio)] > [Parallel Studio Documentation (インテル(R) Parallel Studio ドキュメント)] > [Composer Documentation (インテル(R) Parallel Composer ドキュメント)]** を選択します。 インテル® Parallel Studio に関するその他の情報は、http://www.intel.com/software/products/ を参照してください。

著作権と商標について

本資料に掲載されている情報は、インテル製品の概要説明を目的としたものです。本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスも許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品性に関する保証、第三者の特許権、著作権、その他知的財産権の侵害への保証を含む) にも一切応じないものとします。
インテルによる書面での合意がない限り、インテル製品は、その欠陥や故障によって人身事故が発生するようなアプリケーションでの使用を想定した設計は行われていません。

インテル製品は、予告なく仕様や説明が変更される場合があります。機能または命令の一覧で「留保」または「未定義」と記されているものがありますが、その「機能が存在しない」あるいは「性質が留保付である」という状態を設計の前提にしないでください。これらの項目は、インテルが将来のために留保しているものです。インテルが将来これらの項目を定義したことにより、衝突が生じたり互換性が失われたりしても、インテルは一切責任を負いません。この情報は予告なく変更されることがあります。この情報だけに基づいて設計を最終的なものとしないでください。
本書で説明されている製品には、エラッタと呼ばれる設計上の不具合が含まれている可能性があり、公表されている仕様とは異なる動作をする場合があります。
最新の仕様をご希望の場合や製品をご注文の場合は、お近くのインテルの営業所または販売代理店にお問い合わせください。
本書で紹介されている注文番号付きのドキュメントや、インテルのその他の資料を入手するには、1-800-548-4725 (アメリカ合衆国) までご連絡いただくか、インテルの Web サイトを参照してください。

インテル® プロセッサー・ナンバーはパフォーマンスの指標ではありません。プロセッサー・ナンバーは同一プロセッサー・ファミリー内の製品の機能を区別します。異なるプロセッサー・ファミリー間の機能の区別には用いません。詳細については、http://www.intel.co.jp/jp/products/processor_number/ を参照してください。

Intel、インテル、Intel ロゴは、アメリカ合衆国およびその他の国における Intel Corporation の商標です。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

Microsoft 製品のスクリーンショットは、Microsoft Corporation の許可を得て使用しています。