SONY

CMOS SOI 90nm 版 Cell Broadband Engine プロセッサにおける命令枯渇に起因する Synergistic Processor Element の無限ストールの防止について

アプリケーションノート

Version 1.0

2007年 2月 19日

ソニー株式会社
〒108-0075 東京都港区港南 1-7-1

株式会社ソニー・コンピュータエンタテインメント
〒107-0062 東京都港区南青山 2-6-21

ソニーのホームページ　http://www.sony.net
SCEI のホームページ　http://www.scei.co.jp

Cell Broadband Engine のホームページ　http://cell.scei.co.jp

Version 1.0
2007 年 2 月 19 日

目次

概要

ある種の状況下で分岐ヒント命令を使用した場合に、Synergistic Processor Element(SPE)の無限ストールが生じることがある。SPE の命令プリフェッチが妨げられることで、実行すべき命令が尽きてそれ以上進行しなくなることがあり、その際、SPE は実行（RUN）ステートにとどまったままである。

SPE の無限ストールは、特定の分岐ヒント命令がきっかけとなって生じる。これらでヒント付けされた分岐命令が実行される際、ローカルストレージが、DMA ユニットやプログラムのロード/ストアによるリード/ライトのサービスで非常にビジーな場合に、命令フェッチ障害が生じる。この命令フェッチ障害により、脱出不能な枯渇状態に至り、SPE の無限ストールを生じる。
無限ストールした SPE は、そのプログラム・ステートやその他のシステム・リソースを破壊することは無い。ストール状態の SPE も、非同期割り込みには応答し、通常の実行に復帰する。90nm の Cell Broadband Engine™ (CBE)プロセッサよりも後の全バージョンの SPE は修正されており、これらのコード・シーケンスを正常に実行する。

ここで重要なこととして注意していただきたいのは、通常に実行されるプログラムにおいても、枯渇状態はときどき生じており、それ自体が無限ストール状態を招くわけではないという点である。実際、SPE が無限ストールする直前に枯渇状態を招いているのは、上記命令フェッチ障害である。

２つのステップを取ることによって、この無限ストール状況を緩和する必要がある。

- 90nm の CBE プロセッサを組み込んだシステムを更新し、ファームウェアあるいは OS ソフトウェアにデバイスドライバを含めて、それにより SPE のストール状況を検出してストールした SPE を再開させるようにする。再開すれば、SPE は正常に実行することになる。ストールした SPE は、システムソフトウェアが SPE 無限ストールに反応して実行を再開できるまでに数千サイクルの間ストール状態にとどまるかもしれない。この復旧時間はシステムソフトウェアの実装に依存する。
- 90nm CBE プロセッサ上で動作するソフトウェアを、無限ストールが生じる既知のシナリオが無いようにコーディングすることが可能である。ただし、本問題の解析は難解であるので、90nm ベースの全システムはシステムソフトウェアの更新を行なうべきである。

本アプリケーションノートでは、無限枯渇ストールから復旧できるシステムの構成方法、および、現在既知の無限ストールの発生原因から免れるソフトウェアの構成方法を論じる。

システムソフトウェアでの復旧サポート

OS コンポーネントやシステムファームウェアなどのシステムソフトウェアにて、パフォーマンスモニタをプログラムしてストールした SPE を検出することができる。そうすれば、システムはストールした SPE を復旧できる。復旧には 2 通りの実施方法がある。

- SPU 実行制御レジスタ(SPU_RunCntl) を用いて、まず SPE を停止させ、アイドル状態になったら、再開させる。SPE が再開すれば、再び進行することになる。この復旧は、メモリフロー・コントローラ(MFC) のステートレジスタ 1 (MFC_SR1) の S ビットでは行なえないことに注意のこと。SPE の動作開始は同期イベントの一種であり、現在有効となっているヒント情報が実行開始の前にクリアされる。
- システムソフトウェアはまた、サーマルスロットル・コントローラをプログラムして高温イベントを模擬し、ストールした SPE を短時間休止させることもできる。休止が完了すると、SPE は現在有効となっているヒント情報をクリアし、進行を開始する。

90nm CBE プロセッサベースのシステムには、再コンパイルや再コーディングが不可能な既存ソフトウェアが含まれているかもしれず、ゆえに、それらはシステムソフトウェアでの復旧サポートに頼ることになる。これらのワークアラウンドを使用するシステムでは、パフォーマンスモニタやサーマルスロットル・コントローラの他の用途での使用は許されないかもしれない。システムソフトウェアは、命令枯渇無限ストールが生じた SPE 命令のアドレスを記録するようなデバッグモードを提供することもできる。

無限ストールを免れるコードの生成を可能にするコンパイラオプション

ソフトウェアを、SPE 無限ストールの確率を低減するように構成することが可能である。SPU コンパイラは、安全性と性能の度合いが異なる複数のコンパイルモードを提供することができる。無限ストール防止のために用意されているかもしれないコンパイラオプションについての情報は、SPU コンパイラのリファレンスドキュメントを参照のこと。

ある安全性の度合いに対して性能をどれほどトレードオフするかの判断は、ソフトウェアのリアルタイム制約と平均性能要件との対比を考慮して行なうべきである。厳しいリアルタイム制約を伴うソフトウェアでは、復旧時間が容認できないかもしれず、SPE 無限ストールのリスクを最小にして既知の性能を得るようなコーディングを必要とするかもしれない。非常に高い性能が必要なソフトウェアでは、性能優先のコーディングと、システムソフトウェアによるレアケースの無限ストールの復旧が許容されるかもしれない。

アセンブラ使用者とコンパイラ実装者向けの情報

無限ストールの回避が重要である場合は、ヒント付けされた分岐命令までのインライン命令列、および、その分岐先のコーディング方法に対して、ある種の制約を課さなければならない。これらのルールに従えば、無限ストールに至るような既知のシナリオは存在しない。これらのルールを守ることにより、SPE ソフトウェアの命令数の増加、SPE ソフトウェアの平均実行時間の増加、あるいはその両方、があるかもしれない。

ほとんどの場合、マイナスの影響は小さく、かつ、それは 8 サイクル以上連続してロード/ストアを行なうことのないコードにおいてである。ロード/ストアの長い連続を伴うものでは、多くの場合、ソフトウェアのワーストケース性能は改善される。（無限ストールが実際には生じない場合においても。）

分岐ヒント命令の多重使用の禁止

第 1 のルールは、ヒント付けされた分岐命令までのインライン命令列を規定するものである。このインライン命令列には、他の分岐命令に対するヒントが含まれていてはいけない。もし、他の分岐ヒントがあると、たとえ分岐の飛び先側が、以下のセクションで述べるように注意深くコーディングされていたとしても、SPE ソフトウェアは無限ストールする可能性がある。

ヒント付き分岐先ストリームのコーディングルール

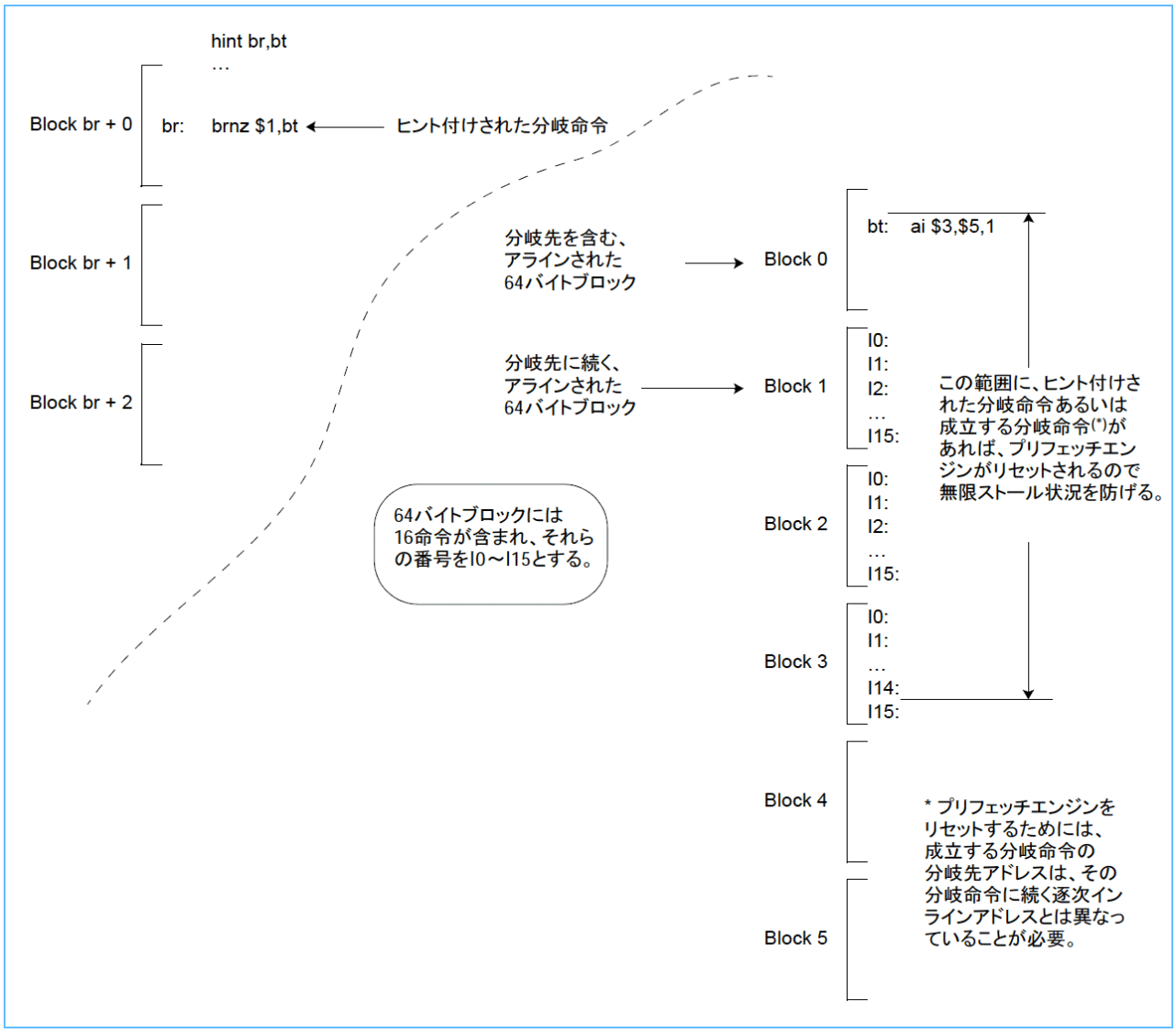
残りのルールは、ヒント付けされた分岐命令の分岐先以後の命令列を規定するものである。分岐先では以下の条件の少なくとも一つが成り立つようにコーディングしなければならない。

- 無限ストールが生じる前に SPE のプリフェッチエンジンがリセットされるようになっている。
- 命令プリフェッチがローカルストレージをタイムリーにアクセスすることが許されようになっている。
  本書で提案する方策は、プリフェッチヒント(HBR.P)命令を、分岐先に続く命令列中のキー・ポイントに、注意深く挿入するというものである。

プリフェッチブロック

本書で示すルールは、プリフェッチエンジンでフェッチされるコードブロックに関するものである。命令は、アラインされた 64 バイトブロック内での位置に従って処理される。
図 1 は、ヒント付けされた分岐命令と分岐先のコードが 64 バイトブロック（ブロック当たり 16 命令）に収まっている様子を示したものである。記述を容易にするために、ブロックには名前が付けられている。
ブロック br+0 にはヒント付けされた分岐命令が入っている。
ブロック br+1 は、ローカルストレージ中でブロック br+0 に続く 64 バイトブロックである。
ブロック br+2 は、ローカルストレージ中でブロック br+1 に続いている。
ブロック 0 には、ヒント付けされた分岐命令の分岐先が入っている。
後述するような特定の特殊ケースの分岐命令がない限り、ブロック 1 からブロック 4 まではローカルストレージ中でブロック 0 の後にリニアに続いている。ここで注意すべきなのは、分岐命令側のインライン命令ストリーム (br+0, br+1, br+2) と、分岐先の命令ストリーム(ブロック 0, 1, 2, ...) とは、任意に重なっていることもあるという点である。

図1 ヒント付けされた分岐命令と分岐先（無限ストールの可能性有り）



無限ストール発生前のプリフェッチエンジンのリセット

分岐先の命令ストリームが無限ストールを誘発するおそれがあるのは、SPE がブロック 3 の最後の命令までシーケンシャルにインラインで実行できる場合である。本件の命令フェッチ障害が起こると、ブロック 4 および以降の命令は決してフェッチされないかもしれない。そうすると、SPE は必要な命令が到着するのを待ちつづけて無限にストールすることになる。ただし、シーケンシャルな実行が終わってプリフェッチエンジンがリセットされる場合は、その限りでない。
シーケンシャルなインライン実行には、以下の 2 通りの終わり方がある。

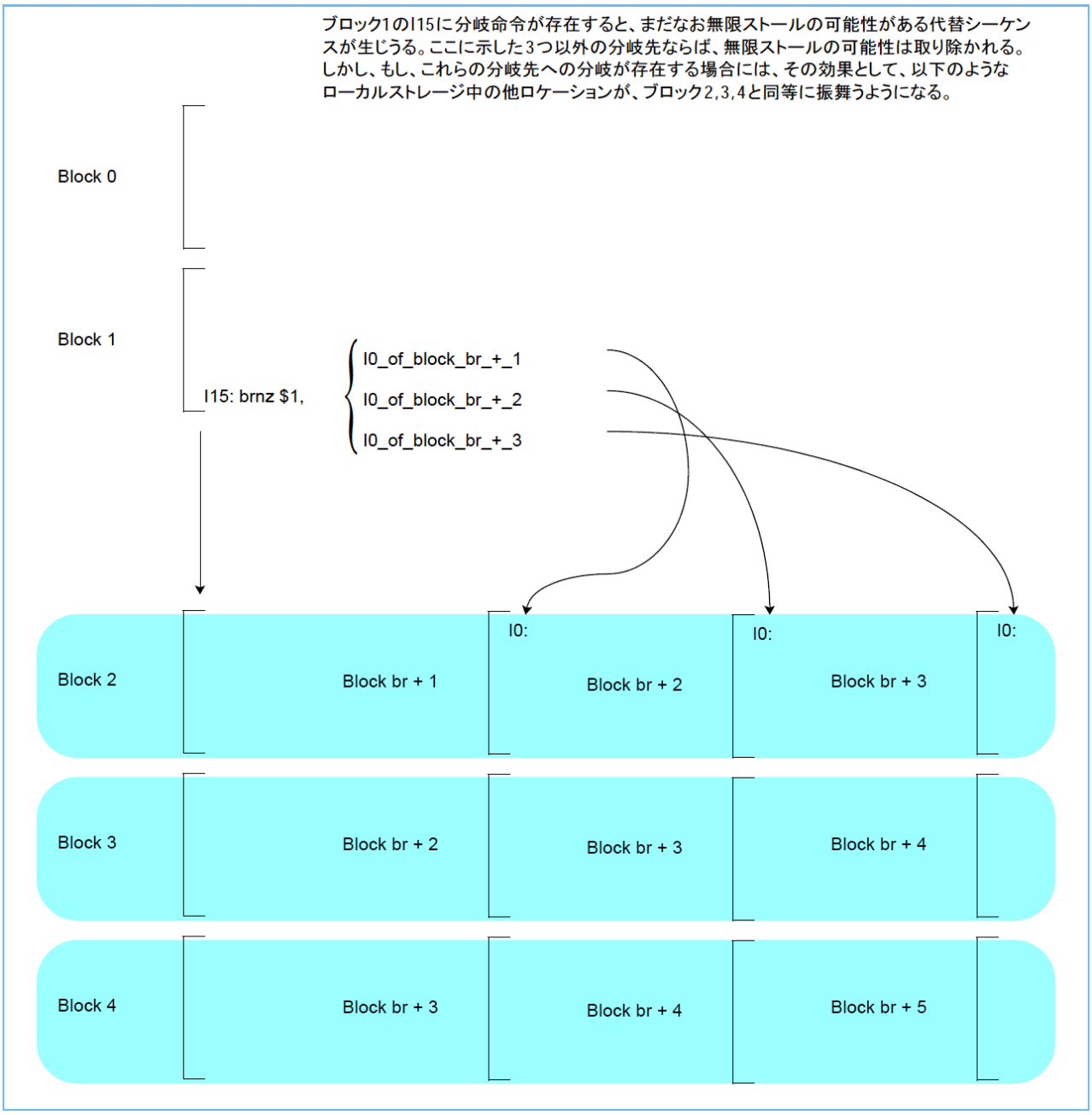
- 予測が外れた分岐命令。ヒントが付いていない分岐命令はインライン側に進むと予測投機される。したがって、ヒント無しの分岐命令が分岐してインライン経路から外れると、予測失敗であり、パイプラインのフラッシュが生じ、プリフェッチエンジンがリセットされる。
- ヒント付きの分岐命令は、インライン経路ではなくヒント付けされた分岐先から実行を行なうので、シーケンシャルなインライン実行が終了する。もちろん、そのヒント付きの分岐先は無限ストールの新たな機会となりうる。

このように、ブロック 3 の I15 より前に、ヒント付き分岐命令あるいは成立する分岐命令が存在することで、この分岐先ストリームにおいて SPE が無限ストールしないことが保証される。
注意事項：ヒント無しの成立分岐命令で、自身の直後の命令に飛ぶものは、no-op 命令と同様に扱われ、通常はパイプラインフラッシュを生じることがなく、そのため、この分岐先ストリームにおいて無限ストールの防止を保証する働きはしない。

特殊ケースの分岐命令

ヒント無しだが成立する特定の分岐命令（特殊ケースの分岐命令と称する。）があると、ブロック 2, 3, 4 の定義が複雑になる。これらの分岐命令により、これらのブロックの定義されたロケーションが、他のローカルストレージアドレスに変更されうる。
図 2 では、ブロック 1 の I15 に分岐命令がある場合を示している。もし、この分岐命令がたまたま、元のヒント付き分岐命令のインライン経路中の 3 つのアドレスのいずれかに飛ぶものであると、SPE はそのアドレスに予測投機してパイプラインのフラッシュを回避することがありうる。パイプラインフラッシュが回避されうる、その 3 つの分岐先アドレスとは、ブロック br+1, br+2, br+3 の I0 である。命令がこれらのアドレスから順に実行される際には、それらがブロック 2 の代わりとなり、SPE には、ブロック 4 に相当する個所に到達すると無限ストールする可能性がまだなお存在する。

図 2 ブロック 1 の I15 での特殊ケースの分岐命令



ブロック 3 にも、特殊ケースの分岐命令が最後に置かれる場合がありうる。この場合もまた、ブロック 3 の特殊ケースの分岐命令はパイプラインフラッシュ無しで分岐先に飛びうるので、無限ストールする可能性がまだなお残るような SPE の進行を許してしまうことある。ブロック 3 の特殊ケースの分岐命令は、ブロック br+1, br+2, br+3 の I0 に飛ぶものである。
ブロック 3 と同様に、ブロック 5 にも特殊ケースの分岐命令が最後に来る場合がありうる。しかし、ブロック 5 の特殊ケースの分岐命令は、ブロック 2 の I0 に飛ぶものだけである。
ブロック 5 の特殊ケースの分岐命令は、後述する方法 2 を用いて無限ストールを防止する際には、あってはならない。また、ブロック 1 の特殊ケースの分岐命令の存在は、ブロック 3 やブロック 5 の特殊ケースの分岐命令の

存在を妨げるものではないことにも注意されたい。
まとめると、3 通りの重要な特殊ケースの分岐命令がある。

1. ブロック 1 の I15 にあるヒント無しの分岐命令で、ブロック br+1,br+2,br+3 の I0 を分岐先とするもの。
2. ブロック 3 の I15 にあるヒント無しの分岐命令で、ブロック br+1,br+2,br+3 の I0 を分岐先とするもの。
3. ブロック 5 の I15 にあるヒント無しの分岐命令で、ブロック 2 の I0 を分岐先とするもの。

プリフェッチヒント(HBR.P) 命令の挿入によるタイムリーなプリフェッチの促進

命令プリフェッチは、ローカルストレージへのアクセスのプライオリティが最低であり、DMA、プログラムのロード/ストア、ヒント命令がスケジュールされていない使用可能スロットがあるまで待機する。もし、プリフェッチが長時間妨げられると、SPE はついに命令が枯渇して枯渇ステートで休止することになりうる。それは、パイプラインが掃けてプリフェッチがローカルストレージにアクセスできるようになるまで続く。
通常、分岐先ストリームは、命令プリフェッチユニットがローカルストレージをタイムリーにアクセスできない場合にストールする。しかし、プリフェッチユニットがローカルストレージの使用を妨害されるような少数のシナリオにおいて、プリフェッチエンジンの障害が誘発され、無限ストールへと至るものがある。

本件の命令フェッチ障害を防ぐには、キューイングされた命令プリフェッチが、特定の命令がイシュー（発行）される前にローカルストレージにアクセスを許されることが重要である。命令プリフェッチは、HBR.P 命令を実行することによって援助できる。HBR.P 命令は、実装固有のプリフェッチタイミングのヒントである。HBR.P 命令は、長く連続するローカルストレージ・アクティビティを有するコードでの命令枯渇に伴う性能ロスを、ソフトウェアで回避することの支援を意図したものである。
90nm の SPE では、HBR.P 命令は、ローカルストレージがプリフェッチリクエストを受付けられるようになるまでストールする。HBR.P 命令が命令プリフェッチユニットに到達すると、最高プライオリティのプリフェッチがローカルストレージに送られる。
DMA のローカルストレージアクセスは、特定のタイミングルールに従うが、本質的には命令実行に対して非同期であり、プログラムの実行の間、ほとんど任意のサイクル期間を使ってしまうかもしれない。SPE は、DMA がローカルストレージを使用しているときは、ローカルストレージを使用しない命令をイシューする。したがって、命令フェッチがローカルストレージを特定の命令がイシューされる前にアクセスできることを保証する、唯一の容易な方法は、HBR.P 命令を挿入することである。HBR 命令とプリフェッチ(P)ビットについての情報は、「Synergistic Processor Unit 命令セット・アーキテクチャ」を参照のこと。

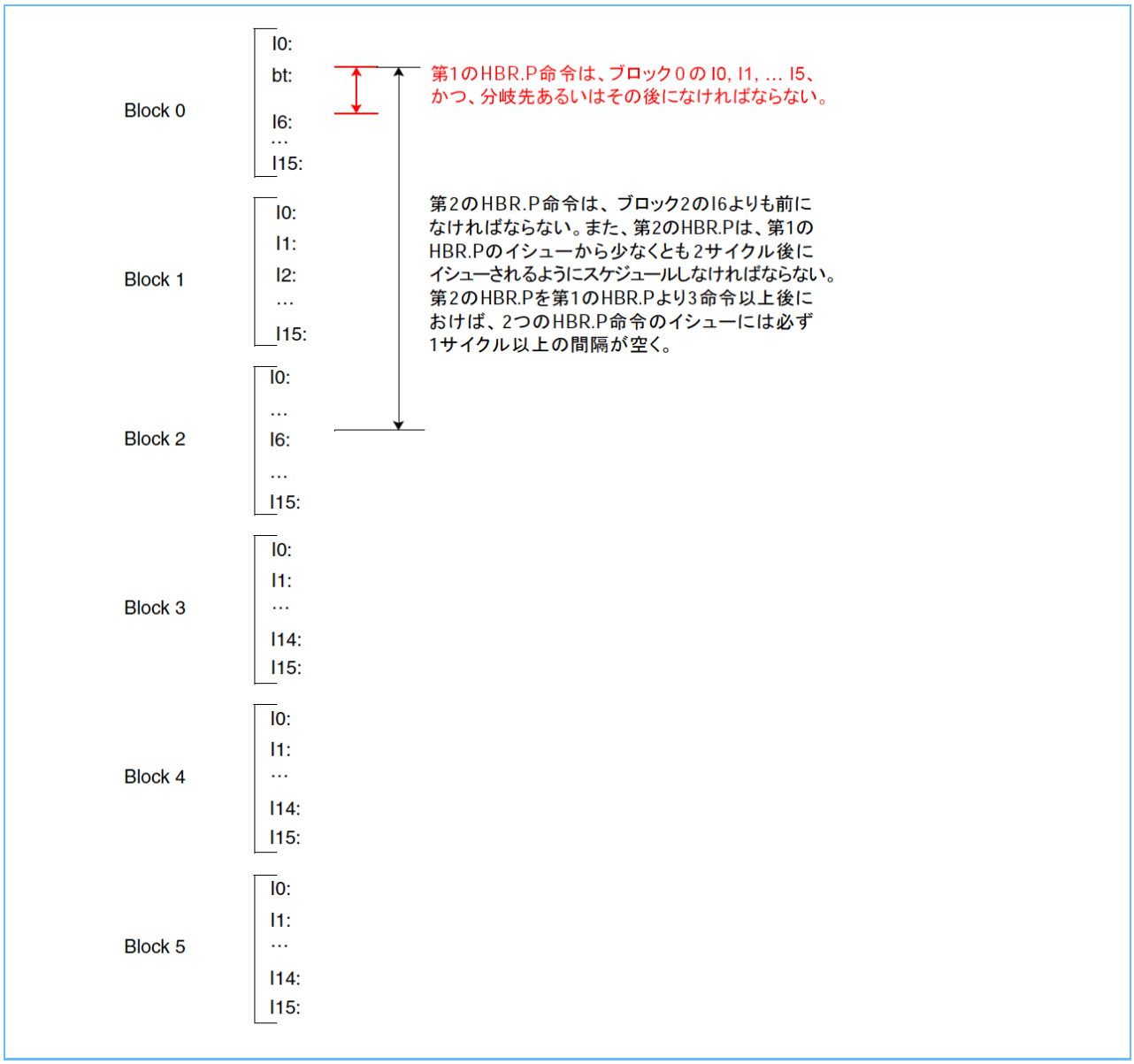
90nm の SPE では、命令プリフェッチには、ローカルストレージの 128 バイトの読み出しパスに最低 2 サイクルの空き時間を必要とする。この要件の意味するところは、HBR.P 命令のスケジュールにおいて、そのイシューサイクルと別の HBR.P 命令や分岐ヒント命令のイシューサイクルとの間に最低 1 サイクルがあるようにして、命令フェッチが発生できるように保証しなければならない、ということである。サイクルの間隔が無いと、命令プリフェッチロジックは HBR.P 命令を無視し、他の機会を待つことになる。HBR.P 命令が無視されると、生じうる無限ストールの回避の助けにはならない。

分岐先の命令列は、HBR.P 命令を用いて、（以下の）3 通りのうちの 1 つでコーディングすれば、無限ストールの既知のシナリオを除去することができる。もし、これらのどの方法も不可能な場合でも、当該分岐先に対する分岐命令がどれもヒント付きでなければ、その分岐先の命令列は、まだなお安全である。

方法1

SPE の無限枯渇ストールを防ぐこの方法では、分岐先の後に 2 個の HBR.P 命令を必要とする。これらの HBR.P 命令の第 1 のものはブロック 0 の I6 より前に、また、第 2 の HBR.P はブロック 2 の I6 よりも前になければならない。分岐先がブロック 0 の I6 以降である場合は、この方法は使用できない。
方法 1 においては、ブロック 2 と 3 のプリフェッチは正常に完了するので、ブロック 1 の I15 に特殊ケースの分岐命令があるとパイプラインがフラッシュされることになる点に注意のこと。ブロック 3 と 5 の特殊ケースの分岐命令についても、方法 1 を用いた場合にはパイプラインフラッシュを生じる。
HBR.P 命令の位置の図解は図 3 を参照のこと。

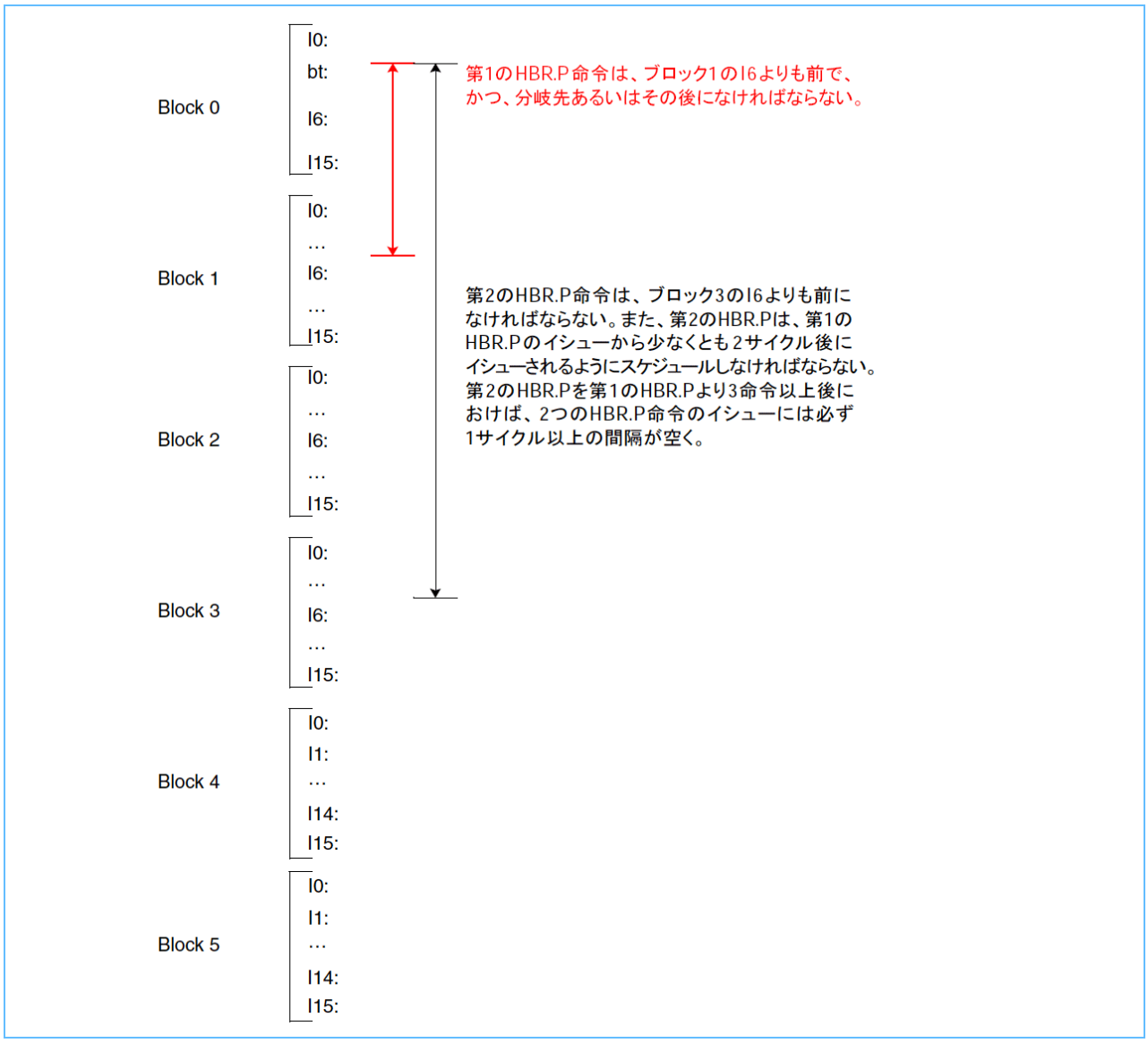
図 3 方法 1



方法2

SPE の無限枯渇ストールを防ぐこの方法では、分岐先の後に 2 個の HBR.P 命令を必要とし、かつ、ブロック 1, 3, 5 に特殊ケースの分岐命令がないことが条件となる。第 1 の HBR.P 命令はブロック 1 の I6 よりも前になければならない。第 2 の HBR.P はブロック 3 の I6 よりも前になければならない。これらの要件の図解は図 4 を参照のこと。

図 4 方法 2

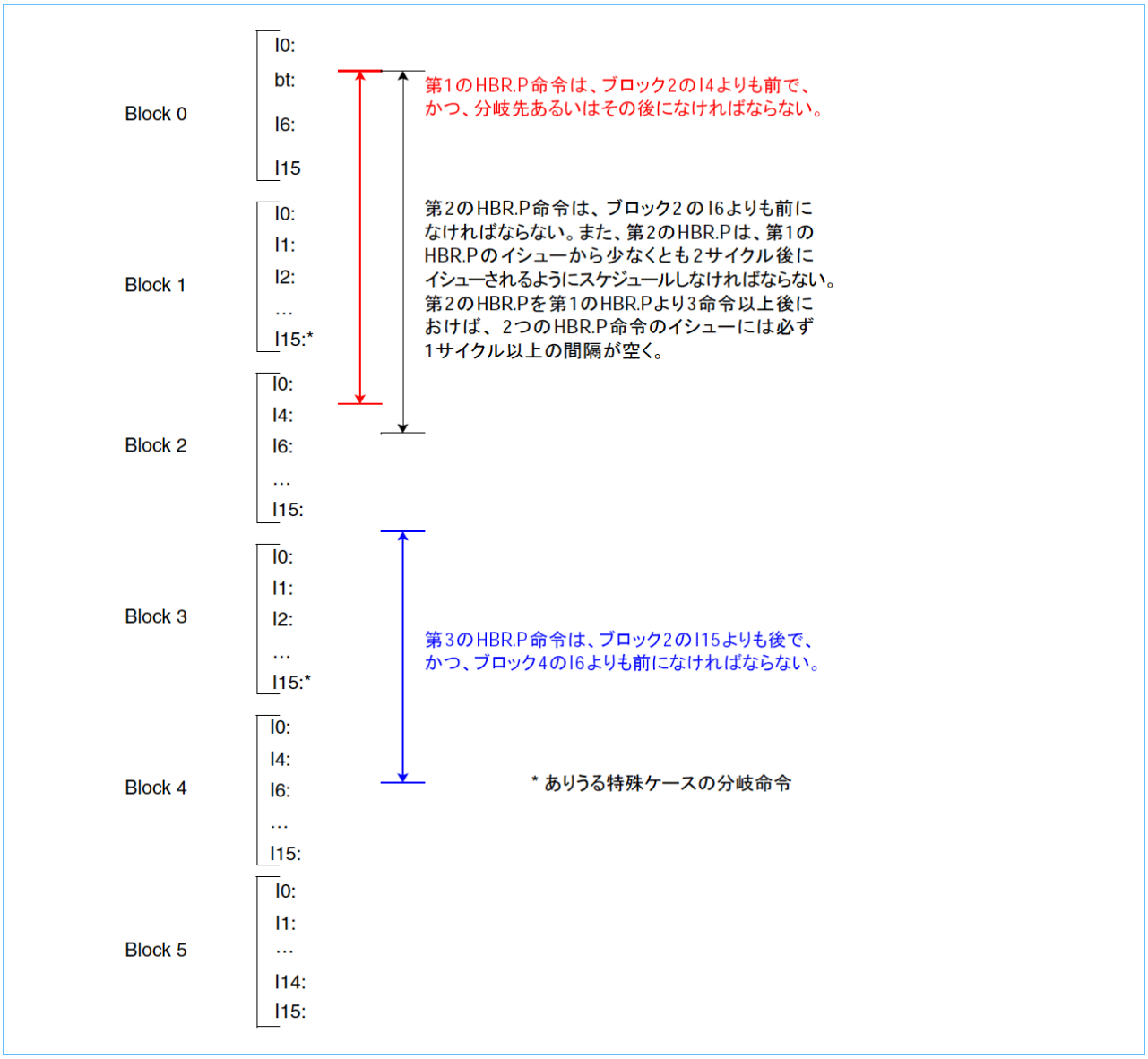


方法3

SPE の無限枯渇ストールを防ぐこの方法では、分岐先の後に 3 個の HBR.P 命令を必要とする。第 1 の HBR.P 命令はブロック 2 の I4 よりも前になければならない。、第 2 の HBR.P 命令はブロック 2 の I6 よりも前になければならない。第 3 の HBR.P 命令は、ブロック 2 の I15 よりも後で、かつ、ブロック 4 の I6 よりも前になければならない。

ブロック 1 と 3 の I15 に重要な特殊ケースの分岐命令が置かれる可能性があり、その場合は HBR.P 命令を挿入すべき場所や個数が変化する。ブロック 5 の最後に置かれる特殊ケースの分岐命令は（方法 3 を用いた場合には）パイプラインフラッシュを生じる。これらの位置要件の図解は図 5 を参照のこと。
例えば、もしそのような分岐命令がブロック 1 の I15 に存在し、かつ、第 1 の HBR.P がブロック 1 の I15 より前に置かれていない場合には、第 1 の HBR.P は、当該特殊ケースの分岐先（図 2 に示すブロック 2 のエイリアス）のブロックの I0 から I3 までに置かなければならない。さらに、この特殊ケースの分岐命令が条件付分岐である場合には、この分岐命令のインライン経路のブロック 2 の I0 から I3 までにも置かなければならない。
同様に、ブロック 3 に特殊ケースの分岐命令があると、ブロック 4 の HBR.P の配置に変更が生じる。

図5 方法3


Block 0
I0:
bt:
I6:
I15
Block 1
I0:
I1:
I2:
…
I15:*
Block 2
I0:
I4:
I6:
…
I15:
Block 3
I0:
I1:
I2:
…
I15:*
Block 4
I0:
I4:
I6:
…
I15:
Block 5
I0:
I1:
…
I14:
I15:
第1のHBR.P命令は、ブロック2のI4よりも前で、
かつ、分岐先あるいはその後になければならない。
第2のHBR.P命令は、ブロック2のI6よりも前に
なければならない。また、第2のHBR.Pは、第1の
HBR.Pのイシューから少なくとも2サイクル後に
イシューされるようにスケジュールしなければならない。
第2のHBR.Pを第1のHBR.Pより3命令以上後に
おけば、2つのHBR.P命令のイシューには必ず
1サイクル以上の間隔が空く。
第3のHBR.P命令は、ブロック2のI15よりも後で、
かつ、ブロック4のI6よりも前になければならない。
* ありうる特殊ケースの分岐命令

コーディング方法のまとめ

表 1 は、3 通りの分岐先コーディング方法のそれぞれが適用可能なケースをまとめたものである。
注意事項として以下が挙げられる。
方法 1 は、適用可能ならば、方法 2 よりも好ましい。プリフェッチヒントを命令スケジュールのより早い時点に有しているため、命令枯渇による性能ロスの確率が小さくなるからである。
方法 2 は、方法 3 よりもプリフェッチヒントの必要個数が少ない。方法 3 ではまた、代替シーケンスのバリエーションが増えるため、分析がより複雑になる場合がある。

表1 分岐先の方法の適用性

方法	HBR.P の個数	分岐先がブロック 0 の I6 以降のときに使用可能か	特殊ケースの分岐命令があるときに使用可能か
1	2	No	Yes
2	2	Yes	No
3	3	Yes	Yes

更新履歴

改訂日付	バージョン	変更内容
2007 年 2 月 19 日	1.0	初版