Vol.2011-SLDM-149 No.35
Vol.2011-EMB-20 No.35
2011/3/19

ベクトルデータのキャッシュ置き換え制御機構の設計と実装

利長勇児[†1]　村田裕介[†2]　山崎信行[†2]

マルチメディア処理では，ベクトル演算やSIMD演算を利用することで大量のデータを高速に処理することができる．しかしながら，マルチメディア処理が扱う大量のデータはキャッシュ中のデータブロックを次々と置き換えていく．そのため，ベクトル演算とスカラ演算を同時実行した場合，スカラデータがキャッシュ上から追い出されシステム全体のスループットが低下してしまう．本研究では，ベクトルロード命令の命令アドレスからアクセス頻度の低いデータをロードする命令を見分け，キャッシュ中へのデータの置き換えを抑制することでキャッシュの使用効率を向上する手法を提案する．Responsive Multithreaded Processor上に本研究で提案した機構を実装し評価を行った．その結果，不要なデータのキャッシュ入れ替えを削減し，システム全体のスループットを向上することができた．

Design and Implementation of Cache Replacement Control Mechanism for Vector Data

YUJI TOSHINAGA,[†1] YUSUKE MURATA[†2]
and NOBUYUKI YAMASAKI[†2]

In multimedia processing, it is possible to deal with large amounts of data quickly by using vector operation or SIMD operation. However, the data handled multimedia processing replace cache blocks one after another. When scalar and vector operations execute simultaneously, scalar data is evicted from a cache and throughput of whole system is degraded. In this paper, we propose a method to improve cache utilization by identifying less frequently accessed data from vector load instruction address and suppressing the data replacement into the cache. The mechanisms were implemented and evaluated on Responsive Multithreaded Processor. As a result, reducing unnecessary data cache replacement, we were able to improve throughput of whole system.

1. はじめに

マルチメディア処理では，データの並列性を抽出しベクトル演算やSIMD演算によって同時に多くのデータを処理する．その際，キャッシュ階層を持つプロセッサではデータキャッシュ中の大量のブロックを次々と置き換えていく．一般的にマルチメディア処理におけるデータは一度アクセスしてから再びアクセスする可能性が低いデータが多く，データをキャッシュ中に取り込むことによる効果は低いといえる．しかしながら，マルチメディア処理と非マルチメディア処理を同時に行うタスクにおいては，マルチメディア処理で使用される大量のデータがキャッシュ内を埋め尽くしてしまう．これによって，キャッシュ使用率の高い非マルチメディア処理のスカラデータがキャッシュ上から追い出されてしまい，システム全体のスループットに悪影響をもたらす．

そこで本研究では，マルチメディア処理のベクトルロード命令の命令アドレスに着目し，アクセス頻度の低いベクトルデータをロードする命令と，アクセス頻度の高いベクトルデータをロードする命令を見分けキャッシュの入れ替えを制御する手法を提案し，Responsive Multithreaded Processor上に設計及び実装を行う．この機構によりマルチメディア処理におけるデータがキャッシュを占有することを抑え，一方で，スカラデータのキャッシュ使用率が高くなりシステム全体のスループットを向上する．

本論文の構成は次の通りである．第2章ではマルチメディア処理の特徴とそのキャッシュの扱い及び関連研究について述べる．第3章では本研究の実装対象とするResponsive Multithreaded Processorについて述べる．第4章では本研究で提案するキャッシュ置き換え制御機構について述べる．第5章ではキャッシュ置き換え制御機構を評価し，第6章で本論文をまとめる．

2. マルチメディア処理

マルチメディア処理には，以下の特徴がある．

- 同じ処理を繰り返す

†1 慶應義塾大学理工学部情報工学科
Department of Information and Computer Science, Faculty of Science and Technology, Keio University
†2 慶應義塾大学大学院理工学研究科開放環境科学専攻
Department of Computer Science, Graduate School of Science and Technology, Keio University

Vol.2011-SLDM-149 No.35
Vol.2011-EMB-20 No.35
2011/3/19

- 大量のデータを処理する
- 一度使用したデータに再びアクセスする可能性が低い

データの並列性を抽出しベクトル演算器や SIMD 演算器を使用することで高速にマルチメディア処理を行うことができるが，大量のメモリアクセスが発生する．特に，ストリーミングアプリケーションでは一度アクセスしたデータは二度とアクセスされない．このようなデータを non-temporal data[1] と呼ぶ．non-temporal data は必ずキャッシュミスを起こし，また，そのデータは将来使用することがないためキャッシュ中へデータの置き換えを行っても無駄といえる．

このような，キャッシュヒット率の高いデータがキャッシュヒット率の低いデータによって追い出されキャッシュの使用効率が低下することを，キャッシュポリューションと呼ぶ．特に，ストリーミング処理ではキャッシュポリューションが発生し易く，キャッシュの効果を低下させる原因となっている．

2.1 SSE4 Streaming Load

Intel の SSE4（Streaming SIMD Extentions 4）[2] は，マルチメディア処理を高速に行うための SIMD 命令セットである．基本的な整数および単精度，倍精度浮動小数点の計算や，マルチメディア処理の特殊な命令セットを備えている．SSE4 には Streaming Load[3] のための MOVNTDQA と呼ばれる命令が含まれている．この命令では，メモリからキャッシュを介さずに直接データをロードすることによってスループットを改善する．ストリーミングデータのような一度しか使用しないデータに対してこの命令を使用することでソフトウェアによってキャッシュポリューションの発生を防ぐ．

3. Responsive Multithreaded Processor

本研究は，タスクとしてベクトル演算によるマルチメディア処理と，スカラ演算による非マルチメディア処理を複数スレッドで行い，マルチスレッドプロセッサのシステム全体のスループットを向上することを目的とする．実装は，Responsive Multithreaded Processor（RMT Processor）[4] 上に行う．RMT Processor は Simultaneous Multithreading（SMT）[5] に基づくマルチスレッド処理を行い，それらに優先度を付加することが可能である．スレッド間で起こる演算器やキャッシュシステム等の資源の競合を優先度に基づいて調停し，優先度の高いスレッドを優先的に実行する．また，RMT Processor はソフトリアルタイム処理で要求される高い演算性能を得るために，ベクトル演算機構[6] を有している．

ベクトル演算機構は高速に演算を行うために Vector Integer Unit 及び Vector Floating-Point Unit を持つ．それらは，Vector Register Unit を含む Vector Memory Unit を共有している．Vector Memory Unit は 32KB の 1 次データキャッシュと接続されており，256bit の高いバンド幅でデータが供給される．しかし，マルチメディア処理等の大量のデータを扱う処理では，32KB のキャッシュ容量では不十分である．特に複数スレッドを同時実行した場合などは，ベクトル演算によって供給されるデータがキャッシュを占有してしまい，他スレッドの性能を低下させてしまうことが考えられる．

4. キャッシュ置き換え制御機構の設計及び実装

4.1 設 計 方 針

本研究では，ベクトルデータのキャッシュ置き換えを制御するハードウェア機構の設計及び実装を行う．ストリーミングデータのキャッシングを抑制し，一方で繰り返し使われるデータを予測してキャッシングを行う．

マルチメディア処理では大量のデータに対して同じ処理を繰り返すため，プログラムはループ構造を持つと考えられる．ここでループ内のベクトルロード命令の命令列に着目すると，ベクトルロード命令はループ毎に新しいデータをロードする命令と，同じデータをロードする命令に分けられる．**図 1** は，ループ中でベクトルロード命令を行うプログラムのアセンブラ例である．全てのデータに対してキャッシュへの置き換えを行うと，r1 及び r2 が示すアドレスをロードするベクトルロード命令はキャッシュヒットを起こすが，r3 が示すアドレスをロードするベクトルロード命令は必ずキャッシュミスしループ毎に新しいデータをロードする．

そこで，ループ中のベクトルロード命令を命令アドレスによって区別しテーブルにどの命令アドレスによるベクトルロード命令がキャッシュヒットを起こすのか，またはキャッシュミスを起こすのかを記録していく．データのキャッシュ中への置き換えするか否かの判断はそのテーブルの情報をもとに行う．キャッシュヒットの予測情報を記録するために予測テーブルを実装し，その情報をもとにキャッシングするか否かを制御するためにキャッシュ置き換え制御機構をキャッシュコントローラ内に実装する．**図 2** に予測テーブル及びキャッシュ置き換え制御機構を含めたキャッシュコントローラのブロック図を示す．

4.2 予測テーブル

予測テーブルは小容量のメモリである．キャッシュへのデータの置き換え判断時に命令アドレスによって高速に予測テーブルを参照できるように命令アドレスの下位ビットをインデックスとする．予測テーブルの構成を**図 3** に示す．

Vol.2011-SLDM-149 No.35
Vol.2011-EMB-20 No.35
2011/3/19

```
loop: ...
      VLD   v1, r1
      ...
      VLD   v2, r2
      ...
      VLD   v3, r3
      ...
      VLD   v1, r1
      ...
      ADDI r1, r1, #20
      ADDI r3, r3, #20
      ADDI r4, r4, #-1
      BNZ   r4, loop
```

図 1 アセンブラ例
Fig. 1 Example of assembler

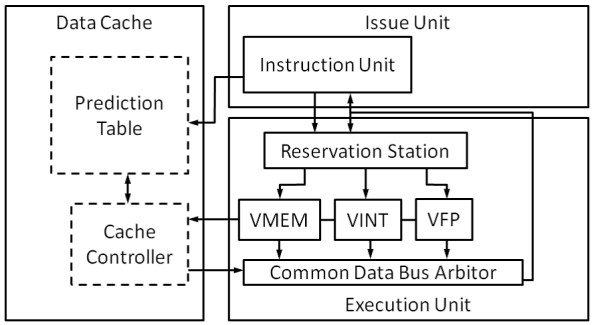

図 2 キャッシュ置き換え制御機構のブロック図
Fig. 2 Block diaglam of cache replacement control mechanism

予測テーブルの各エントリは Data Address フィールドと State フィールドを持つ．

Data Address フィールドにはベクトルロード命令が発行される度に命令アドレスをインデックスとするエントリにデータアドレスを記録していく．キャッシュのデータ置き換えはキャッシュラインサイズ単位に行われるので，データアドレスからキャッシュラインのオフセットを引いたアドレス，すなわち，キャッシュタグアドレスとキャッシュインデックスアドレスを合わせた部分を記録する．

State フィールドは各エントリの状態を保持する．State フィールドは Data Address フィールドに記録されているアドレスとこれからアクセスするアドレスを比較した結果によって以下に示す 3 状態を**図 4** のように遷移をする．3 状態を遷移するため 2bit となる．

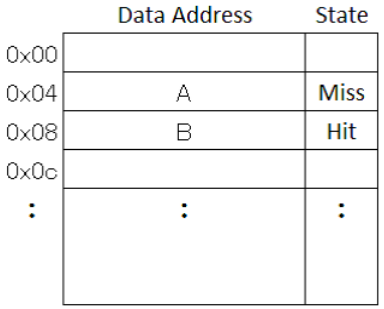

図 3 予測テーブル
Fig. 3 Prediction table

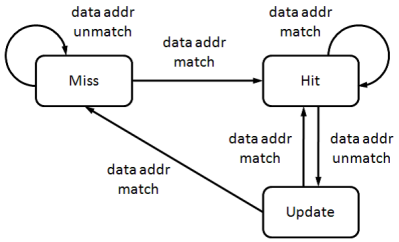

図 4 ステートフィールドの状態遷移図
Fig. 4 State transition diaglam of state field

- Miss
  ループ毎に新しいデータアドレスにアクセスすると毎回キャッシュミスを起こすことになるがそれを示すのがこの状態である．繰り返しロード命令のアクセスするデータアドレスが前回アクセスしたデータアドレスと不一致だった場合に Miss 状態になる．各エントリの初期状態である．
- Hit
  ループ毎に同じデータアドレスにアクセスする場合キャッシュに置き換えることでキャッシュヒットすることになるがそれを示すのがこの状態である．ロード命令のアクセスするデータアドレスが前回アクセスしたデータアドレスと一致した場合に Hit 状態になる．
- Update
  ループ中でヒットするデータをロードする命令がループ毎に新しいデータをロードする場合に対応するのがこの状態である．そのロード命令がアクセスするデータはキャッシュに置き換えることでキャッシュヒットを起こす．該当のエントリが Hit 状態のときにロード命令のアクセスするデータアドレスが前回アクセスしたデータアドレスと不一致だった場合に Update 状態になる．

ベクトルロード命令が発行されると**図 5** のように命令アドレスをインデックスとしてエ

Vol.2011-SLDM-149 No.35
Vol.2011-EMB-20 No.35
2011/3/19

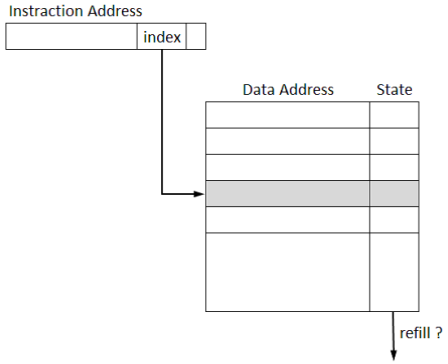

図 5 予測テーブルへのアクセス
Fig. 5 Access to prediction table

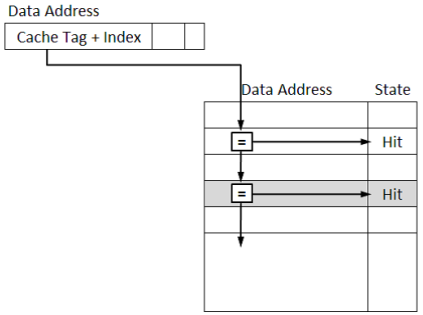

図 6 データアドレスの比較
Fig. 6 Comparison of the data address

ントリにアクセスし，アクセスするデータアドレスと記録されているデータアドレスを比較し上記で示した状態を State フィールドに記録する．その後データアドレスを該当のエントリに記録する．また，ループ中でキャッシュヒットを起こすベクトルロード命令を見分けるために**図 6** に示すように全エントリとデータアドレスを比較し，一致するエントリを Hit 状態へと遷移させる．

なお，命令アドレスの下位ビットが同じ別のベクトルロード命令によって予測テーブルが更新された場合正しい予測を行うことができないが，エントリ数が十分であれば正しい予測を行うことが可能である．ハードウェアコスト削減と高速化のためにタグアドレスを持たない．

4.3 キャッシュ置き換え制御機構

キャッシュ置き換え制御機構は予測テーブルの情報をもとにキャッシュ中へのデータの置き換えを制御する．該当のエントリが Miss 状態ならキャッシュ置き換えを行わず，Hit 状態，及び Update 状態ならば置き換えを行う．これら機構をキャッシュコントローラ内に実装した．

5. 評価及び考察

5.1 評 価 環 境

予測テーブル及びキャッシュ置き換え制御機構は Verilog HDL を用いて実装し，NC-Verilog を用いた RTL シミュレーションによる評価を行った．評価に用いた RMT Processor の命令ユニットのパラメータを**表 1** に示す．また，RMT Processor のキャッシュの概要を**表 2** に示す．

表 1 命令ユニットのパラメータ
Table 1 Parameter of the instruction unit

アクティブスレッド数	8
キャッシュスレッド数	32
命令フェッチ数	8
同時命令発行数	4
同時命令完了数	4
整数演算器	4 + 1(Divider)
浮動小数点演算器	2 + 1(Divider)
整数ベクトル演算器	1 (8IU × 2 line)
浮動小数点ベクトル演算器	1 (4FPU × 2 line)
分岐ユニット	2

表 2 キャッシュユニットの概要
Table 2 The outline of the cache unit

L1 キャッシュ	物理アドレスキャッシュ サイズ 32KB 8 way set-associative 方式 ノンブロッキングキャッシュ ラインサイズ 32byte LRU エントリ入れ替え ライトバック方式
L2 キャッシュ	ビクティムキャッシュ サイズ 512 byte

評価に用いたプログラムを次に示す．

- DCT
  64 ブロックに対して 8 次の 2 次元 DCT を行う
- ブロックマッチング
  64 ブロックに対して 8 × 8 のブロックマッチングを行う
- 3 次元座標変換
  任意の座標 512 点に対し繰り返し 3 次元座標変換を行う

5.2 予測テーブルの性能評価

まず，予測テーブルの性能を確認するために，DCT を 1 スレッドで実行した場合の相対性能及びキャッシュへのデータの置き換え回数を測定した．相対性能は実行時間の逆数を従来手法で正規化した値であり，従来手法とはすべてのデータをキャッシュに置き換える手法である．従来手法を用いた場合，ベクトルロード命令によるデータをキャッシュに置き換えない場合，本提案で予測テーブルのエントリ数を 4，8，16，32，64 にした場合で測定した．結果を，**図 7** に示す．

DCT は，アクセス頻度の高いベクトルデータと一度しかアクセスしないベクトルデータが混在している．本提案の予測テーブルによりそれらを区別し，従来手法と同等の性能を出しながらキャッシュへの置き換え回数を抑えられていることがわかる．無駄なベクトルデータがキャッシュに取り込まれることを防ぐことで，後に示すアクセス頻度の高いデータや他

Vol.2011-SLDM-149 No.35
Vol.2011-EMB-20 No.35
2011/3/19

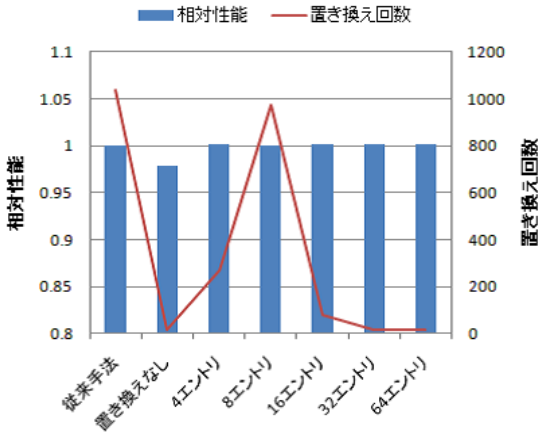

図 7 DCT 実行時の相対性能とキャッシュ置き換え回数
Fig. 7 Normalized performance and the number of cache replacement of DCT

スレッド等がキャッシュを利用することを可能にする．32 エントリの場合でほとんどの無駄なキャッシュ置き換えを削減できている．以上により，本提案では従来手法と同等の性能を出しながら，無駄なキャッシュ置き換えを抑制可能であることがわかった．これによって，アクセス頻度の高いデータや他スレッドがキャッシュを使用可能となり，キャッシュポリューションの発生を軽減させる効果が期待できる．

5.3 4 スレッド実行時の性能評価

DCT，ブロックマッチング，3 次元座標変換を 4 スレッド同時実行した．従来手法で実行した場合と 32 エントリの予測テーブルを用いて実行した場合の評価結果を**図 8** に示す．グラフの縦軸は従来手法を 1 とする相対性能である．

DCT では 6%，ブロックマッチングは 1%程度の性能改善となった．これは，本提案によりアクセス頻度の低いベクトルデータをキャッシュ中に取り込まなくなり他のアクセス頻度の高いベクトルデータやスカラデータがキャッシュを利用可能になったためである．ブロックマッチングの性能改善率が低いのは，扱うデータのほとんどが一度しか使用しないデータであり，キャッシュによる効果がほとんどないからである．また，3 次元座標変換では性能改善しなかったのは扱うベクトルデータの量が少なく，キャッシュをあふれさせなかったためである．

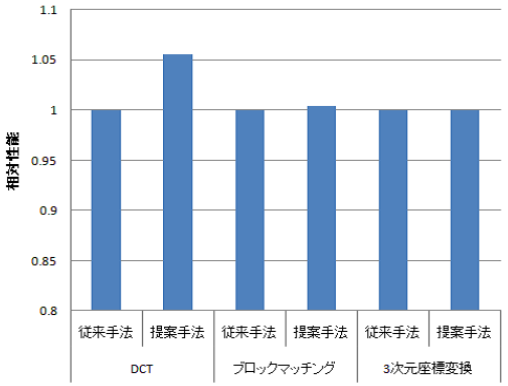

図 8 4 スレッド実行時の相対性能
Fig. 8 Normalized performace for 4 threads

5.4 ソートとの実行時の性能評価

DCT，ブロックマッチング，3 次元座標変換をソートプログラムと同時実行した．ソートは 128 要素をクイックソートにより整列するプログラムである．1 スレッドで各マルチメディア処理を行い，7 スレッドでソート 1-7 を実行した．マルチメディア処理に最も高い優先度を付け，ソートは，ソート 1 からソート 7 まで順に高い優先度を付加した．従来手法と本提案の 32 エントリの場合の性能を**図 9** に示す．縦軸は従来手法の各スレッドの性能を 1 とした相対性能である．

マルチメディア処理との他処理との実行時はアクセス頻度の低い大量のベクトルデータによって他スレッドのスカラデータがキャッシュから追い出されてしまうので，本提案により性能向上が期待できる．DCT とソートとの処理では，DCT 自体の処理が 4%，ソートが最大 11%，平均 3% 性能を向上できた．また，ブロックマッチングとソートとの処理では，ブロックマッチング自体の処理が 18%，ソートが最大 14%，平均 11% 性能を向上した．3 次元座標変換では，性能にほとんど差はなく，これは前述の評価結果と同様で 3 次元座標変換では扱うベクトルデータ量が少なく他処理にほとんど影響を与えないためである．DCT やブロックマッチングなどの，アクセス頻度の低いデータを大量にキャッシュに置き換える処理では，その置き換えを抑制することによってアクセス頻度の高いデータや他スレッドのデータがキャッシュを使用可能になり，性能を向上することができた．

Vol.2011-SLDM-149 No.35
Vol.2011-EMB-20 No.35
2011/3/19

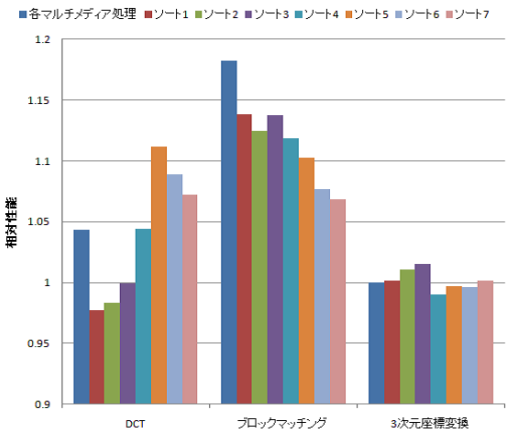

図 9 ソートとの実行時の相対性能
Fig. 9 Normalized performace with sort

表 3 論理合成結果
Table 3 Result of logic synthesis

ユニット		面積 [mm^2]
従来のキャッシュコントローラ		0.16
予測テーブルを含めたキャッシュコントローラ	4 エントリ	0.17
	8 エントリ	0.18
	16 エントリ	0.19
	32 エントリ	0.22
	64 エントリ	0.29
実装前の RMT Processor 全体		54.49

5.5 ハードウェア面積の評価

予測テーブルとキャッシュ置き換え制御機構を加えたキャッシュコントローラを，Synopsys 社の Design Compiler2010.03 を使用して論理合成し，ハードウェアの面積についての評価を行った．テクノロジライブラリには TSMC 社の 130nm のプロセスを使用した．従来のキャッシュコントローラと，予測テーブルのエントリ数がそれぞれ 4，8，16，32, 64 の場合のキャッシュコントローラの論理合成結果を**表 3** に示す．

従来の RMT Processor のチップ全体の面積と比べると，本研究の機構を追加したことによる面積増加は，予測テーブルのエントリ数が 32 エントリの場合で 0.1%ほどのハードウェア面積増加となった．

6. ま と め

本研究では，マルチメディア処理におけるアクセス頻度の低いデータのキャッシュ置き換えを抑制するキャッシュ置き換え制御機構の設計及び実装を行った．これらの機構により，キャッシュへの不要なデータ入れ替えを削減し，マルチメディア処理におけるベクトルデータによってスカラデータが追い出されることを抑制できた．すべてのデータをキャッシュ中へ置き換える従来手法に比べ評価プログラムにおいて 4 スレッド実行時に最大で 5%，複数処理の実行時に最大で 18%性能が向上した．

謝辞 本研究は科学技術振興機構 CREST の支援によるものであることを記し，謝意を表す．また，本研究の一部は文部科学省グローバル COE プログラム「環境共生・安全システムデザインの先導拠点」に依るものであることを記し，謝意を表す．

参 考 文 献

1) AndreasSandberg, D.E. and Hagersten, E.: Reducing Cache Pollution Through Detection and Elimination of Non-Temporal Memory Accesses, *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pp.1–11 (2010).
2) Intel: *Intel SSE4 Programming Reference* (2007).
3) Jha, A. and Yee, D.: Increasing Memory Throughput Width Intel Streaming SIMD Extensions 4 (Intel SSE4) Streaming Load (2007).
4) Yamasaki, N.: Responsive Multithreaded Processor for Distributed Real-Time Processing, *IWIA '06 Proceedings of the International Workshop on Innovative Architecture for Future Generation High Performance Processors and Systems*, pp.44–56 (2006).
5) Tullsen, D.M., Eggers, S.J. and Levy, H.M.: Simultaneous Multithreading: Maximizing On Chip Parallelism, *In Proc. of the 22nd Annual International Symposium on Computer Architecture*, pp.392–403 (1995).
6) Itou, T. and Yamasaki, N.: Design and Implementation of the Multimedia Operation Mechanism for Responsive Multithreaded Processor, *Journal of Robotics and Mechatronics*, Vol.17, No.4, pp.456–462 (2005).