SAR 画像再生処理の高速化 ——キャッシュアクセスを考慮したコーナーターンの 1 改善法

和泉秀幸[†,☆] 中島克人[††] 佐藤裕幸[†]

全天候性や高解像度といった特徴から，リモートセンシング・システムにおいて SAR（Synthetic Aperture Radar）画像の利用が進んでいる．このようなシステムを実現するうえで，SAR 画像再生処理の高速化が重要な課題となってきている．我々はこの課題を達成するため，SMP（Symmetric Multi-Processor）計算機上で並列処理を利用して，SAR 画像再生処理の効率の良い実行法の開発を進めている．ここでは，SAR 画像再生処理の 1 部分処理である “コーナーターン” に焦点をあてて高速化を行った．これは，コーナーターンを単純に並列実行すると，キャッシュミスにより性能が低下し，効率の良い並列化が困難と予測したためである．我々は，キャッシュアクセスの効率を改善するコーナーターンの並列実行方法 “画像ブロックコーナーターン法（SBCT: Square Block Corner-turn Technique）” を提案する．SBCT では，キャッシュのラインサイズ，キャッシュラインの衝突，処理対象の SAR 画像のサイズを考慮して，コーナーターンの実行を制御することでキャッシュミスを低減する．実機上での性能評価結果から，SBCT は，8 プロセッサの環境で，コーナーターン処理を約 25 倍高速化した．これにより，SAR 画像再生処理全体の性能を約 20%改善した．

An Efficient Technique for Corner-turn in SAR Image Reconstruction by Improving Cache Access

Hideyuki Izumi,[†,☆] Tuyoshi Nakajima[††] and Hiroyuki Sato[†]

As the importance of SAR (Synthetic Aperture Radar) image and its application increases, performance improvements in SAR image reconstruction are the key issues in developing practical SAR image processing systems. In order to achieve this goal, we are working to develop an efficient algorithm to process SAR image reconstruction on SMP (Symmetric Multi-processor) by applying multi-thread programming. In our study, we are focusing on “corner-turn”, a subprocess of SAR image reconstruction, which becomes a bottleneck in conventional parallel algorithms because of intensive cache miss. We proposed an efficient technique “SBCT (Square Block Corner-turn Technique)” for parallelizing “corner-turn”, considering cache line size, collisions in cache line, and the size of SAR image data. This dramatically reduces the cache miss. The evaluation result for our new scheme shows about 25 times speed-up in the parallel “corner-turn” on 8 processors, which contributes to a total performance improvement in SAR image reconstruction by about 20%.

1. はじめに

SAR（Synthetic Aperture Radar：合成開口レーダ）は，日中，夜間，雲霧などの天候を問わずに，地表を高解像度で撮像可能な画像レーダである[1),2)]．SAR では，センサが収集した信号データから，人間が理解可能な画像データを生成する “SAR 画像再生処理” が必要になる．この処理は画素あたりの演算量が多く，かつ画像サイズが大きい．SAR を利用するシステムでは，SAR 画像再生処理の性能改善が重要な課題となっている．このため，我々は，SMP（Symmteric Multiprocessor）でマルチスレッドプログラムによる処理の並列化（高速化）を進めている．

コーナーターンは，メモリアクセスを中心とした SAR 画像再生処理の 1 部分処理である．コーナーターンでは，キャッシュのヒット率を考慮せずに，単純に実行するとキャッシュミスを起こしやすく，性能低下の原因となる．特に，SMP 上の並列プログラムでは，共有バスへのアクセスを増加させ，台数効果までも低

† 三菱電機株式会社情報技術総合研究所
Information Technology R&D Center, Mitsubishi Electric Corporation
†† 三菱電機株式会社本社
Mitsubishi Electric Corporation
☆ 現在，三菱電機株式会社鎌倉製作所
Presently with Kamakura Works, Mitsubishi Electric Corporation

下させる原因となる．

市販の COTS（Commercial Off-The-Shelf）計算機では，プロセッサの性能向上と比較して，メモリアクセスの性能向上は緩やかである．このため，キャッシュのヒット率を改善することで，メモリアクセス時間を削減するプログラムの高速化方法が提案されている[3)~5)]．“ブロッキング” は，このような高速化技法の 1 つであり，処理対象の領域を部分領域である “ブロック” に分割して実行する．ここでは，キャッシュのラインサイズ，ラインの数（総容量），制御のメカニズム（set associative など）や階層を考慮して，効率の良いブロックを定義することが重要である．

コーナーターンは，2 次元画像データのメモリ上の配置を変更（転置）する処理である．我々はコーナーターンを解析し，ブロッキングを適用することで，行列の演算と同様に，キャッシュのヒット率改善が可能だと考えた．このような 2 次元データ（配列）の計算へブロッキングを適用する方法として，対象の計算でのデータへのアクセスパターンを定式化してキャッシュミスの回数を算出（推定）し，ブロックのサイズを決定する方法[6)]，複数の配列計算を 1 つにまとめる（contract）方式と組み合わせてブロックのサイズを決定する方法[7)]，キャッシュのサイズと配列のサイズからブロックのサイズとメモリ上での配置を決定する TSS（Tile Size Selection）法[8)]などがある．

これらの文献は，汎用的な手法であり，コーナーターンへのブロッキングの適用においても十分参考となる．ただし，我々の最終的な目的は，SAR 画像再生処理の高速化である．ここでは，コーナーターンに絞ってブロッキングのサイズを決定できればよい．このため，汎用的な手法よりも簡易に，対象の計算機のキャッシュサイズや制御メカニズムから適切なブロックのサイズを決定することを考えた．また，コーナーターンは SAR 画像再生処理において，他の部分処理を高速化するためのオーバヘッドの位置付けである．このため，コーナーターンだけのために，TSS のようにメモリ上の配置を変更するようなことは避け，他の部分処理ではソースコードを変更せずに，画像再生処理に容易に組み込めるような方法を考えることにした．

SAR 画像再生処理は，人間が判読可能な画像を生成するほかに，画像から情報を判読するアプリケーションの前処理でも使用される．このため，複数プロセッサを占有して 1 枚の画像をできるだけ短い時間で生成する，一定時間内に規定の枚数を生成する，他の アプリケーションと混在した環境で（プロセッサを共有して）効率良く画像を生成するなど，様々な状況下で実行される．一般にブロッキングでは，各プロセッサに割り当てた部分領域が，その処理が完了するまで同じプロセッサで実行されることを前提としている．ただし，他のアプリケーションが混在したシステムでは，部分領域の処理を実行中（完了前）に，プロセッサの割当てが変更される可能性がある．我々は，コーナーターンの高速化でブロッキングを適用するに際し，同じソースコードで様々な状況に対応できることも，重要な課題の 1 つと考えた．

効率良くコーナーターンを実行し，同じソースコードで様々な状況に対応し，かつ，実行中のプロセッサ割当て変更の影響を抑えるには，キャッシュのヒット率が悪くならない範囲で，最小の画像領域を “ブロック” として算出できることが重要と考えた．そこで，我々は，コーナーターンの並列実行方法 “画像ブロックコーナーターン法（SBCT: Square Block Corner-turn Technique）” を提案する．SBCT では，キャッシュのラインサイズを基準に正方形の部分画像領域を “画像ブロック” として算出し，処理対象の領域を “画像ブロック” ごとに分割してコーナーターンを並列実行する．

SBCT の効果を検証するため，我々はプラットホームである CC-NUMA（cache coherent non-uniform memory access）[9)]アーキテクチャの SGI ORIGIN 2000 で性能評価を行った．ここでは，“画像ブロック” の大きさなどを変えて，コーナーターンの実行性能を比較した．また，SBCT による SAR 画像再生処理全体の性能改善についても評価を行った．

本稿では，まず，2 章でコーナーターンでのキャッシュアクセスが不利になる課題を示す．次に，3 章で我々が提案する SBCT を説明し，4 章で SGI ORIGIN 上で実施した性能評価の結果を示す．最後に 5 章でまとめる．

2. コーナーターンの課題

我々は，種々の SAR 画像再生アルゴリズムのうち，レンジ・ドップラ・アルゴリズムを対象に，処理の高速化を検討している．この画像再生処理では，レンジ（センサが電波を照射する）方向とアジマス（センサを搭載したプラットホームが移動する）方向で，FFT や IFFT といった一連の処理でデータを次々に加工していく（**図 1**）．このため，部分処理やサブルーチン単位での並列化戦略をとりにくい（効果が小さい）．一方，データサイズが大きく，データ間の依存関係も小さいため，各部分処理（FFT や IFFT）で，データ並列による並列化を行えばよいことが分かっている．

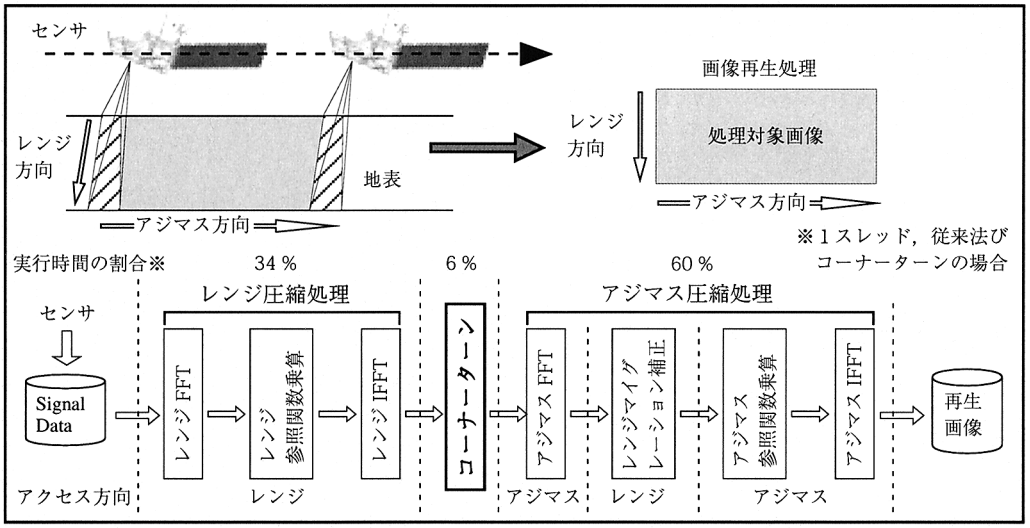


図 1 SAR 画像再生処理
Fig. 1 SAR image reconstruction.

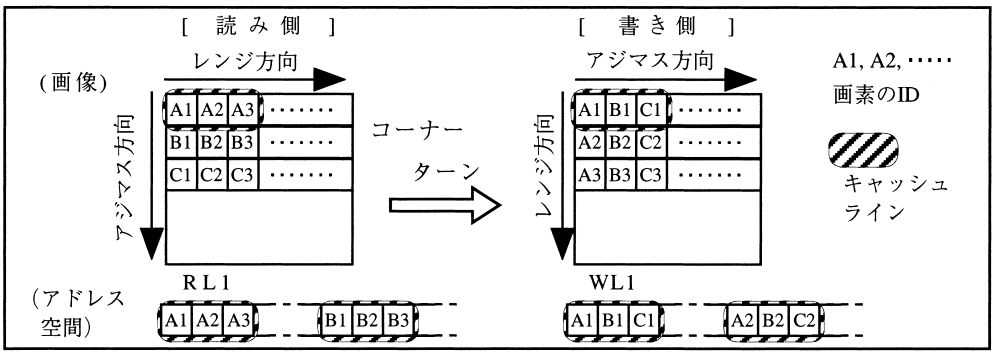


図 2 コーナーターンの問題
Fig. 2 Corner-turn problem.

ただし，各部分処理内で，画像領域でのアクセス方向（レンジ方向とアジマス方向）が異なる（図 1）．そこで各アクセス方向での処理時にキャッシュヒット率を向上させるために，2 次元画像データのメモリ上の配置を転置するコーナーターンを行う．一般に，コーナーターン実施による性能改善量とオーバヘッドとのトレードオフを考慮して，画像再生処理中で処理負荷が高い FFT や IFFT を効率良く実行するために，レンジ圧縮処理とアジマス圧縮処理の間でコーナーターンを 1 度実施する．なお，コーナーターンを実施しない場合は，従来法（画素ごとに単純に逐次実行するコーナーターン法．4.4 節参照）で 1 カ所のコーナーターンを実行する場合と比較し，8 プロセッサ使用時で，画像再生処理全体の実行性能が約 5 分の 1 に低下することを確認している．

次に，コーナーターンでのメモリアクセスの課題を説明する．**図 2** は，レンジ方向からアジマス方向へのコーナーターンの 1 例である．

ここで，コーナーターンを読み側の画像を基準に行うと，読み側では連続したアドレスにアクセスするため，キャッシュのヒット率が高い．一方で，書き側では離散したアドレスにアクセスする必要がある．SAR 画像のサイズが大きいため，書き側では，キャッシュラインに一度入ったデータへ再度書き込みを試みるときに，すでにデータがキャッシュから追い出されてしまっている（図 2 では，画素 “A1” に書き込んだ後，キャッシュライン “WL1” に入っていたデータが，画素 “B1” を書き込む前に追い出されてしまう）．逆に，コーナーターンを書き側の画像を基準に行うと，読み側でデータがキャッシュから追い出されるという問題

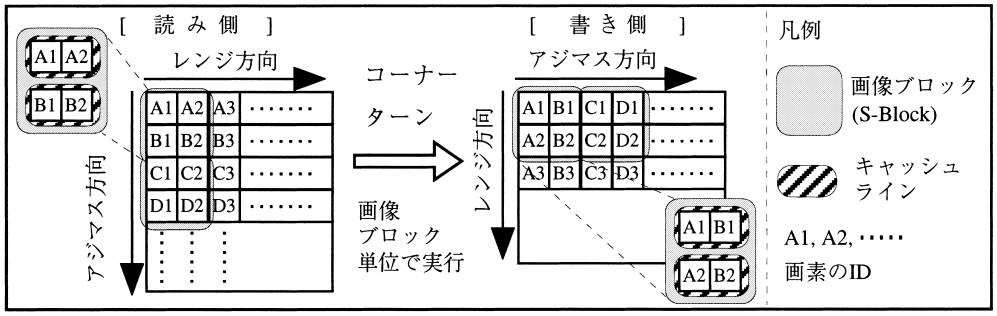


図 3 画像ブロックコーナーターン法
Fig. 3 Square Block Corner-turn Technique (SBCT).

が起きる．

このように，コーナーターンを単純に実行すると，キャッシュミスを起こしやすい．キャッシュミスは共有バスへのアクセスを増加させるため，並列処理における台数効果が低下し，性能低下の原因になると判断した．

3. 画像ブロックコーナーターン法

3.1 画像ブロックの決定方法

我々は，コーナーターンでキャッシュのヒット率を改善する “画像ブロックコーナーターン法（SBCT: Square Block Corner-turn Technique）” を提案する．SBCT では，ブロッキングによる高速化技法をコーナーターンへ適用する．ブロッキングでは，処理対象の画像領域を部分領域 “ブロック” に分割し，各ブロックごとに処理を実行する．このブロックは，処理対象のアルゴリズムと，キャッシュメモリの仕様（ラインサイズ，階層，制御メカニズムなど）から決定する．ブロッキングを利用した高速化では，ブロックの形状とサイズが，性能改善量を決めるキーポイントになる．

我々は，まず，キャッシュラインのサイズ分の画素数を 1 辺とする正方形の領域を，コーナーターンでのブロック候補として考えた．このブロックでコーナーターンを行うと，読み側では正方形の領域に含まれるデータが，数個のキャッシュラインへロード される．書き側では，ロード されたデータが転置され，対応する正方形の領域（数個のキャッシュライン）へ保存される．このため，読み側でキャッシュラインへロードされたデータが，キャッシュラインの追い出しを引き起こさずに，書き側のキャッシュラインへ保存される（**図 3**）．

十分な数のキャッシュラインがあれば，コーナーターンでのブロックのサイズを大きくすること（正方形の 1 辺を，キャッシュラインのサイズの定数倍にすること）が可能である．この場合，システムのキャッシュラインを可能なだけ利用して，大きなサイズのブロックごとにコーナーターンを実行することができる．

ブロッキングを利用してキャッシュのヒット率を改善する場合，各ブロックでの処理は，同じプロセッサで実行される（1 つのプロセッサを占有して実行を完了する）ことを前提としている．もし，各ブロックごとの処理を完了する前に，プロセッサの割当てが変更されると，ブロッキングによるキャッシュのヒット率改善の利点が減少する．

我々は，複数のプロセッサを占有して実行する環境だけでなく，他のアプリケーションと混在した環境でも，同じソースコードで，効率の良いコーナーターンの高速化方法を得ることを課題の 1 つと考えている．現在のプラットホームである UNIX を OS とする SMP システムでは，OS により，処理を実行中にプロセッサの割当てが変更される可能性がある．特に，他のアプリケーションが混在して実行される環境では，この割当て変更が起きる可能性が高い．

我々は，小さいサイズのブロックの方が，プロセッサの割当て変更に対して有利だと考えた．このため，SBCT では，十分な数のキャッシュラインがある場合でも，ブロックのサイズをキャッシュのヒット率が低下しない範囲で最小（キャッシュラインのサイズ）とした．

ここまでの検討結果から，SBCT での “画像ブロック（S-Block: Square Block）” を次のように定義する．S-Block は，1 辺の長さが式 (1) の B（pixel）の正方形の画像領域である．式 (1) において，L（byte）はキャッシュラインのサイズ，P（byte）は画素のサイズ，n は B が自然数となる最小の自然数である（SAR 画像再生処理では，一般に，計算機での処理が有利なサイズを画素として選択するため，画素を分割しない方が有利なケースが多いため）．

$$B = n * L/P \quad (1)$$

SBCT は，SAR 画像を S-Block 単位に分割し，並列計算機上で，コーナーターン処理を S-Block ごとに実行する．

3.2 キャッシュの階層とアクセスパターン

階層構成のキャッシュの場合には，プロセッサに最も近い最上位のキャッシュを基準にした最小の S-Block から，最下位のキャッシュを基準にした最大の S-Block まで，各階層別のキャッシュラインを基準とした S-Block を考えることができる．この S-Block のサイズには，次のトレードオフがある．

各階層でのキャッシュミス時のペナルティの大きさを考えると，最下位のキャッシュを基準に大きなサイズの S-Block を選択するのが有利である．一方で，この大きなサイズの S-Block は，上位キャッシュにとっては，自分の階層で算出される S-Block の数倍という大きな値となる．この S-Block に単純にアクセスすると，上位キャッシュでは，ブロック内のコーナーターンで必要なキャッシュラインが増加し，キャッシュに集中アクセスしてヒット率を高めるブロッキングの恩恵を受けにくい課題がある．上位階層でキャッシュのヒット率を向上させるには，小さなサイズの S-Block を選択するのが有利である．

SBCT では，キャッシュミス時のペナルティを抑えるため，最下位のキャッシュを基準に最終的な S-Block の大きさを算出する．ただし，上位キャッシュについても効率の良いアクセスを実現するため，下位キャッシュを基準とした大きな S-Block の内部を，上位キャッシュを基準とした小さな S-Block に分割したうえで，大きな S-Block のコーナーターンを，小さな S-Block のコーナーターンを繰り返す形式で実行することにした．ここでは，最上位のキャッシュから最下位のキャッシュまで，各階層ごとに順次 S-Block を決定していく．

ここまでの検討結果から，階層構成のキャッシュの場合，SBCT では，次の手順で S-Block を決定していく．まず，プロセッサに最も近い最上位のキャッシュに対する S-Block を，式 (1) を使って（1 次キャッシュのラインサイズと画素のデータサイズから）決定する．次に，1 つ下位のキャッシュに対する S-Block を決定する．ここで，この下位のキャッシュ（算出対象の階層）に対する S-BlocK は，次の式 (2) と式 (3) を使って決定する，1 辺の長さ BT（pixel）の正方形の画像領域である．

$$K = n * LT/(BH * P) \quad (2)$$

$$BT = K * BH \quad (3)$$

式 (2) と式 (3) で，BT（pixel）は算出対象の階層での S-Block の 1 辺の長さ，LT（byte）は算出対象の階層でのキャッシュのラインサイズ，BH（pixel）は 1 つ上位の階層での S-Block の 1 辺の長さである．K は，算出対象の階層の S-Block に納まる，1 つ上位の階層の S-Block の個数である．P（byte）は画素サイズ，n は BT が自然数となる最小の自然数である．

算出対象の階層で S-Block を決定した後，さらに下位の階層があれば以下同様に，式 (2) と式 (3) を使って，最下層のキャッシュまで順次 S-Block を決定していく．なお，下位の階層の S-Block 内のコーナーターンは，1 つ上位の階層の S-Block 単位のコーナーターンを繰り返す形式で実行する．

S-Block 内の画素および複数の S-Block 間でのアクセス順序については，読み側アクセス優先や書き側アクセス優先など，様々なパターンを考えることができる．キャッシュのデータ読み出しとデータ書き込み速度や，キャッシュ制御アルゴリズムなどの情報から，最適なアクセス順序を決定することも可能である．しかし，我々は，すべての情報を入手し，最適なアクセス順序を得ることは，性能改善量と比較してコストが大きいと判断した．このため，SBCT では，複数のパターンから性能評価の結果に基づいてアクセス順序を決定することにした．ただし，S-Block のコーナーターンを実行するのに十分な数のキャッシュラインがない場合には，キャッシュ制御アルゴリズムから，キャッシュラインの追い出し回数が最小になるアクセス順序を選択する．

3.3 キャッシュ制御メカニズム

キャッシュラインへのデータの読み込み方式として，“direct mapping” や “set associative” 方式を採用したキャッシュでは，一定のアドレス間隔ごとに同じキャッシュライン（の組）を利用することになる．SBCT のように，正方形の部分画像領域単位でコーナーターンを実行すると，処理対象の画像のサイズによっては，部分画像領域（S-Block）内の画素データの読み出しどうしや書き込みどうしで，一部のキャッシュラインを集中的に利用し，キャッシュラインの衝突が発生する．なお，SBCT では，各階層のキャッシュはプロセッサ固定（プロセッサ間で共有されていないこと）を前提としている．

SBCT では，コーナーターン実行時のキャッシュラインの衝突発生の有無を，式 (4) と式 (5) で決定する．ここで，B（pixel）は S-Block の 1 辺の長さ，S（pixel）は連続したアドレス方向での処理対象画像のサイズ，L（byte）はキャッシュラインのサイズ，N はキャッシュラインの総数，P（byte）は画素のサイズ，

W は set associative のキャッシュでの way 数（1つの組に属するキャッシュラインの数．direct mapping では $W=1$），m は $0<m<B$ の条件を満たす自然数，n は任意の自然数である．

$$-L < m*S*P - n*N*L/W < 2*L \quad (4)$$

$$2*B/(m+1) > W \quad (5)$$

式 (4) の $m*S*P$ は，S-Block の開始位置から $m+1$ 行目（開始位置を 1 行目と数えて）の先頭の画素までのサイズ（byte），$n*N*L/W$ はキャッシュラインの衝突が発生するサイズ（byte）である．式 (4) は，S-Block の $m+1$ 行目の先頭の画素までのサイズが，衝突が発生する範囲にあるときに成立する．

SAR 画像再生処理では，与えられた領域の中から指定範囲だけを実行することもある．このため，処理領域（コーナーターン実行領域）の先頭が，キャッシュラインの先頭と一致しないこともある．そこで，式 (4) では衝突発生の範囲にマージンを持たせて判定する．式 (4) は，ブロック内およびブロック間のアクセス順序にかかわらず，S-Block 単位で実行するときに，衝突発生の可能性があれば成立する．

式 (5) の $B/(m+1)$ は，S-Block 内でキャッシュラインの衝突が起きる回数である．読み側と書き側で同じキャッシュラインを使う場合を考慮して，左辺は $2*B/(m+1)$ としている．ここでは，式 (4) が成立する最少の m で評価する．式 (5) は，衝突が起きる回数が way 数を上回るときに成立する．

SBCT では，連続したアドレスとなる方向（レンジまたはアジマス）で，式 (4) と式 (5) の両方が満たされたとき，キャッシュラインの衝突が発生すると判断する．この場合，衝突を回避するための “オフセット領域” を，処理対象の SAR 画像の端に追加する（**図 4**）．

この “オフセット領域” ではコーナーターンを行わない．サイズは，式 (6) で算出する（式 (6) で X（byte）はオフセット領域のサイズ，S，L，N，P，W，m，n は，式 (4) および式 (5) と同じ値である．ただし，X は式 (6) を満たす最小の自然数とする）．ここでも，キャッシュラインの先頭と一致しないことを考慮して，マージンを持たせてオフセット領域のサイズを決定する．また，オフセット領域追加の有効性は，プラットホームマシン（後述の SGI 社の ORIGIN）上で，実測により別途確認している．

$$X \geq 2*L/m + n*N*L/W/m - S*P \quad (6)$$

この改善方法を選択した理由は次のとおりである．

- SAR 画像再生処理全体への影響が小さいため
  コーナーターンは SAR 画像再生処理の 1 部分であり，画像再生処理全体への影響が小さい実装が望ましい．SAR 画像再生処理は与えられた領域中の指定範囲を実行できる設計のため，オフセット領域の追加が全体へ与える影響は小さいため．
- 実装が容易で開発コストが小さいため
  オフセット領域を追加する方法のほかに，中間処理用のバッファを確保し，このバッファ経由でコーナーターンを実行することで衝突回避を行う実装も考えた．ここでは，オフセット領域追加を除く SBCT の本体部分が先に完成していたため，オフセット領域の追加法が，プログラムの修正が少なく，実装が容易で開発コストが小さいと判断したため．

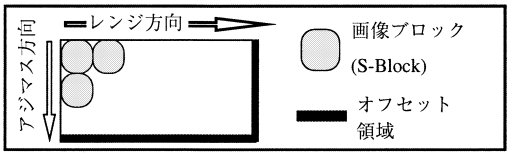


図 4 キャッシュライン衝突の改善
Fig. 4 Improve cache collision.

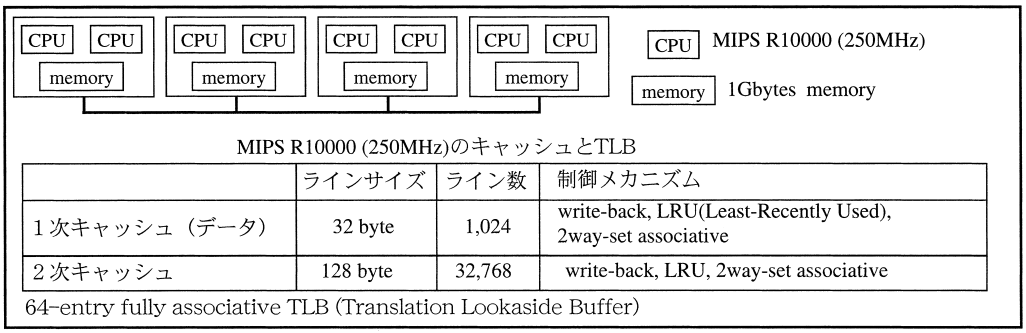
4. 性能評価

この章では，コーナーターンの性能評価を示す．ここでは，ブロッキングのサイズ，キャッシュラインの衝突の有無，アクセスパターンが，コーナーターンの性能に与える影響を評価する．また，SBCT を SAR 画像再生処理に適用した効果も評価する．

4.1 評価環境

性能評価で不変の条件（パラメータ）を示す．なお，プラットホーム，画素データ，スケジューリング方式と評価実行環境は，コーナーターンと画像再生全体の評価で共通の条件である．

- プラットホーム
  8 プロセッサの SGI ORIGIN 2000（**図 5** 参照）．各プロセッサボード上に 2 つのプロセッサがあり，システム全体は 4 ボードで構成される．OS は IRIX 6.5.
- 画素データ
  複素数の構造体．実数部と虚数部に対応した，2 つの浮動小数点データで構成する（各 4 byte で合計 8 byte）．
- スケジューリング方式
  プロセッサへの処理領域（各ブロックまたは SAR 画像再生の領域）の割当ては，Block スケジューリング（分割した処理領域を，均等な個数になるように，連続したひとかたまりの処理領域にして各プロセッサへ割り当てる）とする．



MIPS R10000 (250MHz)のキャッシュとTLB

	ラインサイズ	ライン数	制御メカニズム
1 次キャッシュ（データ）	32 byte	1,024	write-back, LRU(Least-Recently Used), 2way-set associative
2 次キャッシュ	128 byte	32,768	write-back, LRU, 2way-set associative

64-entry fully associative TLB (Translation Lookaside Buffer)

図 5 性能評価実行システム
Fig. 5 Target system.

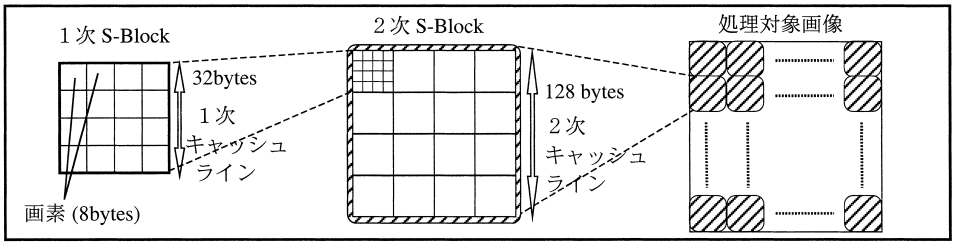


図 6 対象システムの S-Block
Fig. 6 S-Block for target system.

- 評価実行環境
  性能評価で利用するすべてのプログラムは，SPL（Simple Parallel Library）上に実装している．SPL は，我々が標準の Pthread ライブラリ上に実現したループ並列化用の簡易スレッドライブラリである．SPL では，実行開始時に指定のスレッド数を確保し，ループの処理を並列実行したうえで，バリア同期で待ち合わせる．
- コーナーターンの評価で不変の条件
  レンジ方向からアジマス方向へ（読み側ではレンジ方向で連続アドレス，書き側ではアジマス方向で連続アドレス）1 回コーナーターンを実施．画像サイズは 8,192×8,192 画素．
- 画像再生全体での評価で不変の条件
  処理対象の画像サイズは 2,048（レンジ）× 16,384（アジマス）画素．

4.2 対象システムの S-Block

S-Block は，対象システムのキャッシュから決定する（3 章参照）．今回の評価対象である SGI ORIGIN 2000 では，次のようになる（**図 6** 参照）．

まず，1 次キャッシュに対しては式 (1) を使って（$L = 32$，$P = 8$，$n = 1$ で $B = 4$ となる），4×4 画素の画像領域を S-Block として定義する（以降本稿では，1 次 S-Block とする）．次に，2 次キャッシュに対しては式 (2) と式 (3) を使って（$LT = 128$，$P = 8$，$BH = 4$，$n = 1$，$K = 4$ で $BT = 16$ となる），16×16 画素の画像領域を S-Block として定義する（以降本稿では，2 次 S-Block とする）．

今回の計測で対象とする 8,192×8,192 画素の画像では，1 次 S-Block 単位のコーナーターン実行時にキャッシュラインの衝突が発生する（式 (4) と式 (5) で，$B = 4$，$S = 8192$，$L = 32$，$N = 1024$，$P = 8$，$W = 2$，$m = 1$，$n = 4$ となり，衝突発生条件を満たす）．このため，SBCT では，8 画素（64 byte．式 (6) から $X \geq 64$）のオフセット領域を処理対象の画像領域に加える．

この S-Block でのコーナーターンの妥当性を検証するため，次節ではブロックのサイズなどを変更して，コーナーターンの評価を行う．

4.3 コーナーターンの評価

この節では，コーナーターン処理単独で評価を進める．まず，ブロックのサイズを変更してコーナーターンの実行時間を計測する．ここでは，N×N 画素の正方形の領域をコーナーターンを実行するブロックのサイズとし，N の値を変えて計測する．正方形の領域を採用した理由は，コーナーターン対象の書き側と読み側の画像で，同じ画素数のデータを利用するためである．

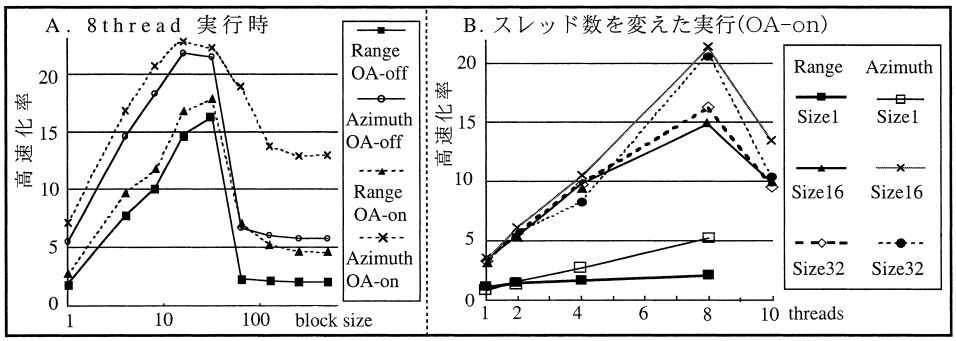


図 7 コーナーターンの計測結果
Fig. 7 Result for Corner-turn.

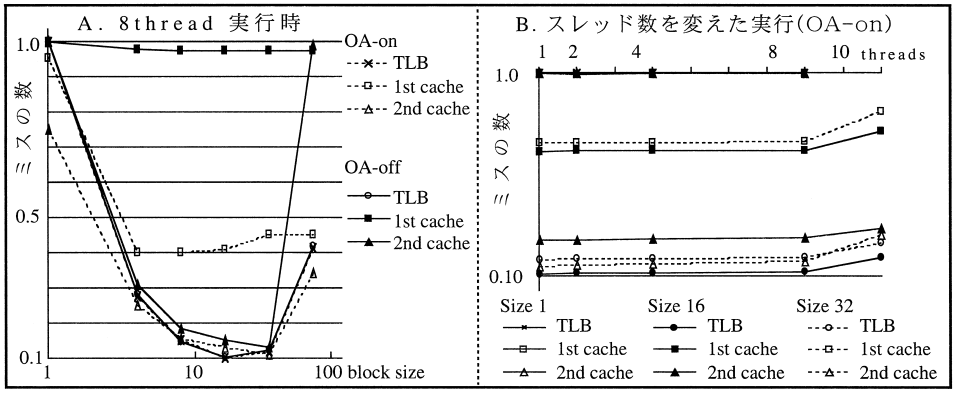


図 8 コーナーターンでのキャッシュミスの数
Fig. 8 Cache miss for Corner-turn.

4.3.1 評価条件

以下に，コーナーターンの評価で変更する計測条件を示す．なお，ここでは，サイズ 4 とは 4×4 画素のブロック，サイズ 1 とは普通に画素単位で実行した場合である．

- ブロックのサイズ
  サイズ 1〜サイズ 512.
- スレッドの数
  1〜10．ただし，スレッド数 1 のときは，逐次実行し，並列化のオーバヘッドを含まない．
- アクセス方向とスケジューリング
  ブロック内およびブロック間でのアクセス方向は，読み側で有利なレンジ方向優先アクセスまたは，書き側で有利なアジマス方向優先アクセス．
- オフセット領域
  オフセット領域あり（OA-on）とオフセット領域なし（OA-off）．なお，オフセット領域のサイズは 8 画素（64 byte，4.2 節参照）．

8 スレッドで実行の条件で，ブロックサイズを変えたときの実行性能を**図 7** A に示す．また，スレッド数を変えたときの実行性能を図 7 B に示す．なお，この節では，計測結果を $H = SC/TC$ の高速化率 H で示す（H は高速化率，TC は評価対象のコーナーターンの実行時間，SC は 1 スレッド，サイズ 1，レンジ方向アクセス，OA-off の条件での実行時間である）．

また，実行性能に影響を与えた原因を解析するため，perfex コマンドを使って，1 次と 2 次のキャッシュと TLB（Translation Lookaside Buffer）のミス数を**図 8** に示す．perfex は，プログラム実行前後でハードウェアのカウンタを参照して，キャッシュミスの回数などを採取するコマンドである．なお，計測の結果，レンジとアジマスの方向を変えても，ミス数は大きく変動しなかったため，アジマスでの結果のみを示す，

また，この節では，ミスの数を $M = TD/SD$ の M で示す（TD は評価対象時のミス数，SD は 1 スレッド，サイズ 1，レンジ方向アクセス，OA-off の基準条件でのミス数，M は基準条件のミス数に対する評価対象時のミス数の割合である）．ミスの数 M が 0.5 とは，基準条件に比べてミスの数が半分になったことを示す．

4.3.2 ブロックサイズ

まず，ブロックサイズについて考察する．図 7 A から，2 次キャッシュのラインサイズと（正方形の 1 辺の長さが）同じサイズ 16 で，最も性能が良い．また，サイズ 32 でも同じくらい性能が良い．図 8 A から，OA-off の条件では 1 次キャッシュが性能に影響を与えるとは考えにくい．ブロックサイズによる性能差は，2 次キャッシュと TLB のアクセス効率に起因すると分析している．サイズ 16 とサイズ 32 では，サイズ 16 の方が TLB のミスが少なく，サイズ 32 の方が 2 次キャッシュのミスが少い．前述のようにレンジとアジマスの方向を変えても，2 次キャッシュと TLB のミスの回数は変わらない．ただし，レンジではライトミスが，アジマスではリードミスが多く発生していると予測している．ここから，ライトミスでは 2 次キャッシュのペナルティが TLB のペナルティよりも相対的に大きいため，レンジではサイズ 32 がサイズ 16 よりも性能が良くなり，一方で，リードミスでは 2 次キャッシュと TLB のペナルティの差が縮小するために，アジマスではサイズ 16 がサイズ 32 より性能が良くなったと考えている．なお，1 次キャッシュのミス数が OA-off でつねに多いことと，サイズ 64 以上で性能が低下する原因は後述する．

図 7 B から，ブロックサイズが台数効果にも影響を与えることが分かる．サイズ 1（単純な画素単位の実行）では，プロセッサが 8 台あっても，台数効果がレンジ方向優先アクセスでは 2 倍程度，アジマス方向優先アクセスでも 5 倍程度に抑えられる．一方で，サイズ 16 やサイズ 32 では，レンジ方向で 4 倍程度まで，アジマス方向では 7 倍程度まで台数効果が改善する．図 8 B から，1 次 2 次キャッシュと TLB のミス数は，8 台まではプロセッサ数を変えても大きく変動していない．また，レンジとアジマスの方向を変えても，このミス数は変わらないことを確認している．このため，書き側で有利なアジマス方向で，読み側で有利なレンジ方向よりも台数効果が良い理由は，キャッシュと TLB のライトミスがリードミスに比べて台数効果に不利なためと考える．

スレッド数がプロセッサ数より多い条件で実行した場合，ブロック単位でのコーナーターン実行中（完了前）に，プロセッサとスレッド（処理）の割当て変更が起きる可能性が高くなると考えている．図 7 B から，2 次キャッシュを効率良くアクセスできるサイズ 16 とサイズ 32 は，スレッド数が 8 以下ではほぼ同じ性能を示している．しかし，割当て変更が起きやすいと考えるスレッド数が 8 を超えた条件では，サイズ 32 はサイズ 16 と比べて実行性能の低下が大きいことが分かる．

図 8 B から，サイズ 32 はサイズ 16 と比べて 1 次 2 次キャッシュと TLB のミス数の増加が大きいことが確認できる．ここから，サイズ 16 とサイズ 32 の性能低下の差には，キャッシュと TLB のミス数の増加が影響していると判断した．このため，我々は，小さいサイズのブロックの方が，プロセッサとスレッドの割当て変更による性能低下の影響を小さく抑えることが可能だと考える．そこで，本環境でのコーナーターンのブロックサイズとしては，2 次キャッシュのアクセス効率が悪くならない範囲で，かつ，ブロックの大きさが最小の，サイズ 16 が有利だと判断した．

4.3.3 オフセット領域

次に，オフセット領域の効果について考察する．図 7 A から，すべてのブロックサイズで，オフセット領域を加えた OA-on の性能が，OA-off に比べて改善していることが分かる．今回の計測対象の画像サイズで OA-off の条件では，すべてのブロックサイズで 1 次キャッシュラインの衝突が発生する．図 8 A で，OA-off の条件で 1 次キャッシュのミス数がつねに多いのは，この衝突が原因だと考えている．OA-on の条件では，1 次キャッシュラインの衝突が回避され，ミス数が減少している．ただし，実行性能との比較から，1 次キャッシュでの改善が全体の性能に与える影響は小さいと判断した．

図 8 A から，OA-on の条件では，すべてのサイズで 2 次キャッシュのミス数が減少しているのが分かる．これは，計測対象の 8,192×8,192 画素の画像では，2 次キャッシュラインの衝突が 32 行ごとに発生するが，オフセット領域の追加により，この衝突が改善されたためと考える．OA-on の条件では，TLB のミス数は減少していないことと，これまでの評価結果から，オフセット領域の追加による性能改善は，この 2 次キャッシュラインの衝突回避が主要因だと分析している．

サイズ 64 では，サイズ 32 と比べて著しく性能が低下している．図 8 A から，この原因は，2 次キャッシュラインの衝突と TLB の不足だと考えている．まず，TLB については，画像の 1 行が 64 Kbyte（8 byte×8,192 画素）あるため，行をまたがって処理を行うと，少なくとも 1 行に 1 個の TLB が使われると考えられる．ORIGIN2000 には TLB が 64 エントリしかなく，コーナーターンでは読み側と書き側の両方でメモリ領域にアクセスするため，サイズ 64 では TLB が不足し，この結果 TLB のミス数が大きく増加したと考える．

次に，2 次キャッシュの影響を考える．サイズ 64 の

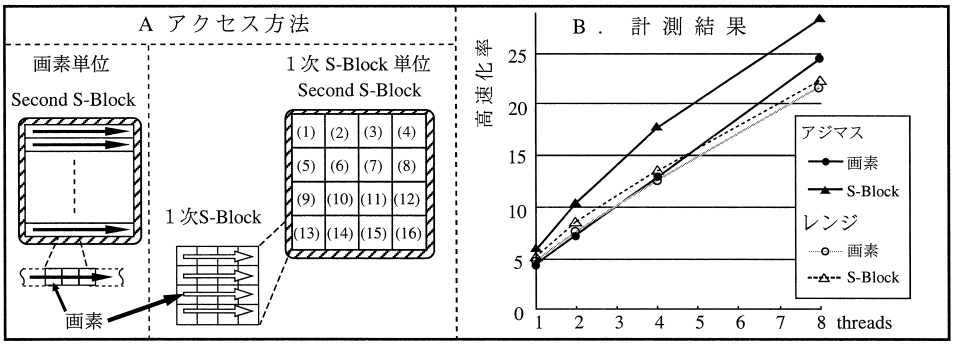


図 9 アクセスパターン
Fig. 9 Access pattern.

OA-on が OA-off よりも性能が改善した理由は，他のサイズと同様に，衝突の回避により，2 次キャッシュのミスの数が減少したためと考える．ここでは，他のサイズよりもミス数の改善が大きいために改善量も大きかったと考える．また，アジマス方向での改善が大きい理由は次のように考える．OA-off では，2 次キャッシュラインの衝突が読み側と書き側の両方で起きる．アジマス方向では，書き側で有利なアクセス順序のため，本来キャッシュのライトミスが少ない．一方で，キャッシュラインの衝突ではリードミスとライトミスが同じ確率で発生する．影響の大きい 2 次キャッシュのライトミスのペナルティが多く発生したために，サイズ 64 のアジマスの OA-off は性能が大きく低下し，OA-on でのオフセット領域の追加の効果が大きくなったと考える．

4.3.4 画像領域内のアクセス方法

上記の評価結果から，SBCT の 2 次 S-Block と同じ大きさのサイズ 16 で，オフセット領域を加えたコーナーターン方法が適していることが確認できた．次に，2 次 S-Block（サイズ 16）内の画像領域のアクセス方法を評価する．ここでは，2 次 S-Block 内のコーナーターンを，1 次 S-Block 単位でのコーナーターンを繰り返して実行する方法と，(サイズ 16 と同様に）画素単位で順次実行する方法の性能を比較する（**図 9** A）．なお，SBCT では，キャッシュの仕様を入力すると，最内側（ここでは 1 次 S-Block 単位）のループを展開して最適化したコードを自動的に生成する．この評価でも，1 次 S-Block 単位での実行で，SBCT が生成したコードを利用している．

図 9 B の計測結果から，1 次 S-Block 単位のコーナーターンを繰り返す方法が，画素単位で順次実行する方法よりも性能が良いことが分かる．この条件の最良のケースでは，単純なサイズ 1 での逐次実行と比較して，約 25 倍の高速化を達成できた．

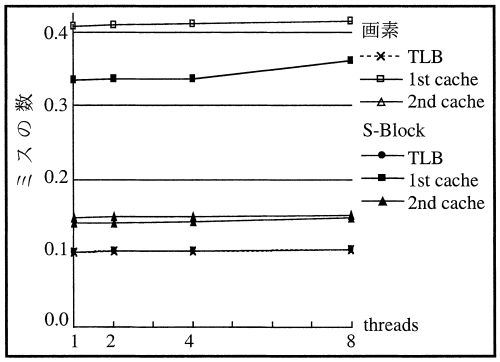


図 10 アクセスパターンでのキャッシュミスの数
Fig. 10 Cache miss for access pattern.

ここでも，perfex コマンドを使って計測した 1 次と 2 次のキャッシュと TLB のミス数を**図 10** に示す．図 10 から，1 次キャッシュのミス数が改善していることが確認できる．これは，読み側と書き側の両方で，1 次キャッシュラインのデータを集中的にアクセスした効果だと考える．図 10 から，1 次キャッシュでの改善が 2 次キャッシュや TLB と比較して大きく，ここでの性能改善の主要因の 1 つと判断した．4.3.2 項と 4.3.3 項の評価では，全体の実行時間が大きく，1 次キャッシュ改善の効果は埋もれていたが，ここでは，全体の実行時間が減少したことで，効果が現れるようになったと考えている．また，最内側ループを展開して最適化したコードを利用することで，サイズの小さいループの実行による実行命令数増加のオーバヘッドを解消していたことも，性能改善に貢献したと考えている．

ここでの評価結果から，2 次 S-Block 内のアクセスについては，1 次 S-Block 単位のコーナーターンを繰り返す実行方法（SBCT で提案した方法）が，適していると判断する．

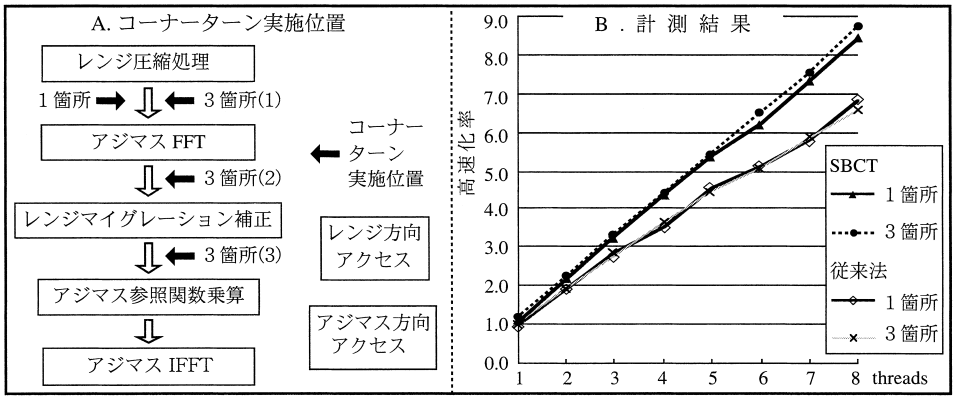


図 11 SAR 画像再生処理での評価
Fig. 11 Evaluation for SAR image reconstruction.

4.3.5 ま と め

ここまでの評価結果から，次の事項を確認できた．

- SBCT は実行性能の改善に効果があること．さらに，台数効果も改善できること．
- プロセッサとスレッドの割当て変更による性能低下を抑えるには，小さいブロックサイズの方が有利なこと．この結果から，SBCT のブロックサイズの決定方法が適切なこと．
- オフセット領域の追加が，キャッシュラインの衝突を回避し，実行性能の改善に効果があること．
- 2 次 S-Block 内のアクセスは，1 次 S-Block 単位のコーナーターンを繰り返す実行方法が，画素単位で順次実行する方法よりも性能が良いこと．さらに，SBCT が妥当な実行方法であること．
- 1 次 2 次のデータキャッシュだけでなく，TLB へのアクセス効率が性能に大きな影響を与えること．SBCT による小さいブロックサイズの選択は，大きなサイズの画像でコーナーターンを行うときには，TLB の使用数を抑えて TLB ミスを改善できる確率が比較的高いと考える．ただし，TLB の仕様や画像サイズによっては，SBCT で決定したブロックサイズでも TLB の不足を招く可能性がある．この場合には，TLB のミスによるペナルティと，キャッシュのミスによるペナルティのトレードオフを考慮して，別途適切なブロックの形状とサイズを決定する必要があると考える．

4.4 SAR 画像再生処理への適用

この節では，SAR 画像再生処理へ SBCT を適用した効果を評価する．ここでの，計測条件は次のとおりである．

- コーナーターン方法
  SBCT または従来法（サイズ 1 で OA-off）．
- コーナーターン実施位置
  1 カ所または 3 カ所で実施（**図 11** A）．
- スレッド数
  1〜8.

計測結果を，従来法，1 スレッド，コーナーターン 1 カ所での実行時間に対する高速化率で示す（図 11 B）．図 11 B から，コーナーターン実施位置が 1 カ所と 3 カ所のいずれの場合も，SBCT を適用した SAR 画像再生処理の方が，従来法を適用した場合と比べて実行性能が良いことが確認できた．SBCT を適用し，3 カ所でコーナーターンを実施する最適の条件では，8 スレッド実行時に，従来法で 1 カ所の条件と比較して，SAR 画像再生処理全体の実行性能を約 20%改善できた．

SBCT を適用した場合，3 カ所でコーナーターンを実施した方が，1 カ所でのみ実施するよりも実行性能が良い．一方で，従来法では，3 カ所で実施すると 1 カ所よりも性能が低下する．これは，レンジマイグレーション補正での実行性能の改善時間と，コーナーターンを 2 回多く実行するのにかかるオーバヘッド時間がトレードオフの関係になるためである．SBCT では，コーナーターンを高速化できるため，オーバヘッド時間が実行性能の改善時間よりも小さくなっている．しかし，従来法では，コーナーターンによるオーバヘッド時間が，改善時間を上回ってしまう．

コーナーターンの処理時間が SAR 画像再生処理全体の実行時間に占める割合を比較する．従来法で 1 カ所の条件では，1 スレッド（逐次実行）時に 6%のものが，8 スレッド時には 15%に増加する．一方で，SBCT で 3 カ所の条件では，1 スレッド時に 5%，8 スレッド時で 5%のままである．

ここまでの評価結果から，SBCT は SAR 画像再生処理の実行性能を大幅に改善することができ，また，

台数効果の改善にも寄与しているといえる．

5. ま と め

我々は，SAR画像再生処理の1部分処理である“コーナーターン”で，キャッシュのヒット率を改善する“画像ブロックコーナーターン法（SBCT: Square Block Corner-turn Technique）”を提案した．この提案法の効果を検証するため，プラットホームの並列計算機上で性能評価を実施した．

性能評価の結果，コーナーターン処理について，提案したSBCTは，8プロセッサで約25倍の高速化を達成した．また，台数効果の改善にも貢献することを確認した．さらに，SBCTを適用することで，8プロセッサの環境で，SAR画像再生処理全体の実行時間を約20%改善できた．

これらの評価結果から，提案したSBCTは，SAR画像再生処理の実行性能の改善で非常に有効な方法だと考える．

参 考 文 献

1) Curlander, J.C. and Mcdonough, R.N.: *Synthetic Aperture Radar Systems and Signal Processing*, John Wiley and Sons, Inc., New York (1991).
2) Carrara, W.G., Goodman, R.S. and Majewski, R.M.: *Spotlight Synthetic Aperture Radar Signal Processing Algorithms*, Artech House, Boston (1995).
3) Gannon, D., Jalby, W. and Gallivan, K.: Strategies for Cache and Local Memory Management by Global Program Transformation, *Journal of Parallel and Distributed Computing*, Vol.5, No.5, pp.587–616 (1988).
4) Dayde, M.J., Duff, I.S. and Petitet, A.: A Parallel Block Implementation of Level-3 BLAS for MIMD Vector Processors, *ACM Trans. Mathematical Software*, Vol.20, No.2, pp.178–193 (1994).
5) Chatterjee, S.: Compiling Nested Data-Parallel Programs for Shared-Memory Multiprocessors, *ACM Trans. Prog. Lang. Syst.*, Vol.15, No.3, pp.400–462 (1993).
6) Lam, M.S., Rothberg, E.E. and Wolf, M.E.: The Cache Performance and Optimizations of Blocked Algorithms, *ACM SIGPLAN NOTICE*, Vol.26, No.4, pp.63–74 (1991).
7) Lim, A.W., Liao, S.-W. and Lam, M.S.: Blocking and Array Contraction Across Arbitrarily Nested Loops Using Affine Partitioning, *ACM SIGPLAN NOTICE*, Vol.36, No.7, pp.103–112 (2001).
8) Coleman, S. and McKinley, K.S.: Tile Size Selection Using Cache Organization and Data Layout, *ACM SIGPLAN NOTICE*, Vol.30, No.6, pp.279–290 (1995).
9) Bhuyan, L.N., Iyer, R., Wang, H. and Kumar, A.: Impact of CC-NUMA Memory Management Policies on the Application Performance of Multistage Switching Networks, *IEEE Trans. Parallel and Distributed Systems*, Vol.13, No.3, pp.230–246 (2000).

(平成14年10月2日受付)
(平成15年 4 月3日採録)

和泉 秀幸（正会員）

1968年生．1991年名古屋大学工学部情報工学科卒業．同年三菱電機（株）入社．以来，同社情報技術総合研究所にて，並列プログラミング環境および合成開口レーダ画像処理システムやソフトウェア生産技術等の研究開発に従事．2003年4月から同社鎌倉製作所に勤務．電子情報通信学会およびIEEE Computer Society会員．

中島 克人（正会員）

1953年生．1977年京都大学工学部電気第二工学科卒業，1979年同大学院修士課程修了．同年三菱電機（株）に入社し，汎用・専用計算機の開発に従事．1982年より第五世代コンピュータ・プロジェクトに参画し，推論マシンのアーキテクチャ/言語処理系等の研究開発に従事．その後，同社情報技術総合研究所にて並列・分散処理等の研究開発に従事．現在，同社開発本部開発業務部に勤務．工学博士．電子情報通信学会およびIEEE Computer Society会員．

佐藤　裕幸（正会員）

1982 年筑波大学第三学群情報学類卒業．同年三菱電機（株）入社．1985 年 6 月～1989 年 4 月（財）新世代コンピュータ技術開発機構に出向し，逐次型および並列型推論マシンのシステムソフトウェアの研究開発に従事．現在，三菱電機（株）情報技術総合研究所にて，並列処理ソフトウェア，最適化システムの研究開発に従事．