6J-05

シングルポートメモリによる循環バッファを用いたスカラリプレイス

平尾　誠治　　瀬戸 謙修
東京都市大学工学部電気電子工学科

1　はじめに

高位合成向けメモリアクセス最適化手法の一つであるスカラリプレイス [1] は、並列処理をする上でボトルネックとなるメモリアクセスの競合を少なくする技術である。スカラリプレイスは一度メモリから呼び出されたデータをレジスタと循環バッファに保持し、再利用することで同じメモリからの同時アクセスを減らし、ボトルネックの解消に貢献する仕組みとなっている。しかし既存のスカラリプレイス技術を適用した結果、回路性能は向上する反面、デュアルポートメモリを用いた循環バッファを使用するため、メモリ面積の増加が問題となる。この問題を解決するために、本稿ではスカラリプレイスを適用する上で必要な循環バッファを、シングルポートメモリで構築し、面積増加を抑える方法を提案する。

2　デュアルポートメモリによる循環バッファを用いた既存のスカラリプレイス

```
1  for(i=0; i<16; i++){
2    for(j=0; j<8; j++){
3      if(i>=1 && j>=1) A[i][j] = B[i][j] + B[i-1][j-1];
4    }
5  }
```

(a) スカラリプレイス適用前

```
1  for(i=0; i<16; i++){
2    for(j=0; j<8; j++){
3      b0 = B[i][j];
4      if(j>=1&&i>=1) A[i][j] = b0 + b9;
5      b9 = B_mem[mem];
6      B_mem[mem] = b0;
7      mem++;
8      if(mem == 8)mem=0;
9    }
10 }
```

(b) スカラリプレイス後

図 1: スカラリプレイス適用前後のコード

ここでは既存のスカラリプレイス [1] の手順を説明する。図 1(a) のソースコードに対し、スカラリプレイスを適用した結果が図 1(b) のソースコードとなる。スカラリプレイスはコード上の複数回参照のある配列を一時変数に置き換える。まず、配列依存を考え、データの再利用を発見することから始まる。図 1(a) のソースコード中で、任意のループで B[i][j] がアクセスするアドレスに対して、外側ループ 1 回、内側ループ 1 回実行されたときに B[i-1][j-1] がアクセスし、保持されている値が変化しないため再利用が可能である。このとき B[i][j] から B[i-1][j-1] の依存ベクトル $< d1, d2 >$ を $< 1, 1 >$ と表記する。再利用距離は内側ループ回数 N と依存ベクトル $< d1, d2 >$ から以下の式で求められる。[2]

$$d1 \times N + d2 \quad (1)$$

式より B[i][j] から B[i-1][j-1] の再利用距離は 9 と求められ、配列 B[i][j] は一時変数 b0 に、B[i-1][j-1] は b9 に置き換える。

次に循環バッファを構築する。循環バッファは、依存元の配列に入力された値を保持し、再利用距離に合った適切なタイミングで依存先へ値を渡す機構である。b0 に入力された値は、ループが 8 回実行される間循環バッファに保持され、9 回目のループで b9 に渡される。その動作を行う循環バッファを構築すると図 2 のようになる。

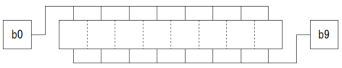

図 2: 循環バッファ

循環バッファには再利用距離-1 個分要素を持つメモリを用意する。b0 に入力された値はメモリ内の要素に渡され、ポインタを回し順次値を埋めていく。ループが 8 回実行され、ポインタが最初の要素に戻されたとき、保持されている値が b9 に渡される構造となっている。

以上よりスカラリプレイスは適用可能となる。スカラリプレイス適用後のコードは図 1(b) となる。5 行目で B 配列の値を b0 に入力し、5 行目～8 行目に記述された循環バッファで保持し、b9 で再利用する。図 1(b) のソースコードより循環バッファには read、write 参照が同時に存在し、メモリアクセスの競合が発生するため開始間隔 1 でループパイプライン化するためにはデュアルポートメモリを使用する必要がある。

3　シングルポートメモリによる循環バッファを用いた提案するスカラリプレイス

既存のスカラリプレイスではメモリアクセスの競合を回避するため、循環バッファの構築にデュアルポートメモリを使用していた。しかし、デュアルポートメモリはシングルポートメモリに比べ、約 1.5 倍面積が大きくなる [3]。本論文では工夫によって、性能はそのままにデュアルポートメモリをシングルポートメモリに置き換える方法を提案する。具体的にデュアルポートメモリでの構築に必要なメモリの 2 倍のビット幅、1/2 倍の要素数のシングルポートメモリを使用する。

3.1　メモリの要素数が偶数の場合

図 1(b) のソースコード中の循環バッファ(図 2) をシングルポートメモリに置き換えた場合、図 3 となる。

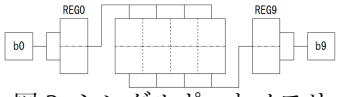

図 3: シングルポートメモリ

図 3 のように、要素数 4、ビット幅 2 倍のメモリと、レジスタ REG0、REG9 を用意する。用意したメモリとレジスタは上位ビットと下位ビットに分け、1 つの要素で 2 つの値が保持できるようにする。b0 に入力された値は、最初のループで REG0 の上位ビット、2 回目のループで下位ビットに保持される。2 回目のループで REG0 の両ビットに値が保持されたと同時にメモリに両データが渡される。つまりメモリへの write はループ回数が偶数回目のとき発生する。また、メモリから REG9 への read はループ回数が奇数回目のとき発生する。REG9 へ出力され、保持された上位ビットの値は直後に b0 へ出力され、次のループで下位ビットに保持された値が b0 へ出力される。b0 から b9 への値の再利用を図示したものが図 4 である。図 4 より、メモリへの入出力は交互に行われ同時に発生しないため、シン

グルポートメモリでの構築が可能となる。図5のソースコードはシングルポートメモリによる循環バッファを記述したものである。flagはメモリへの入出力を交互に行うために用いている。

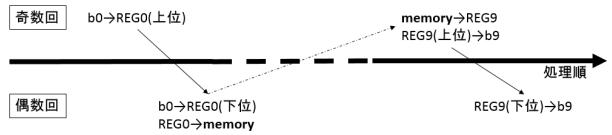

図4: b0からb9への再利用のフロー

```
1  if(flag == 0){   //奇数回
2      REG9 = B_mem[mem];
3      b9 = REG9 &0x0000ffff;
4      REG0 = B0;
5      flag = 1;
6  }else if(flag == 1){   //偶数回
7      b9 = REG9 >> 16;
8      REG0 |= (b0 << 16)
9      mem[mem] = REG0;
10     mem++;
11     flag0 = 0;
12     if(mem == 4)mem = 0;
13 }
```

図5: シングルポートメモリによる循環バッファの記述

3.2 メモリの要素数が奇数の場合

3.1節ではメモリの要素数が偶数の場合で示してきたが、メモリの要素数が奇数だった場合は、上下ビットに分割できない余りのメモリの要素が発生してしまうため工夫が必要となる。これまでの説明ではB配列の再利用距離は9であったが、再利用距離が1大きい10であった場合を新たに例に用いる。シングルポートメモリによる循環バッファの構図と記述をそれぞれ図6と図7のソースコードで示す。図7のソースコード中の2行目には、図5のソースコードのシングルポートメモリによる循環バッファの記述が入る。再利用距離が10の場合、間に必要なメモリの要素は9となり、上下ビットに分割すると1つ余る。余った部分はTMPという新しいレジスタに置き換える。カウンタを設け、カウンタが0〜7まではこれまで通り循環バッファを動作させ、カウンタが8のときだけb0とTMP、TMPとb10で直接値の受け渡しを行う。

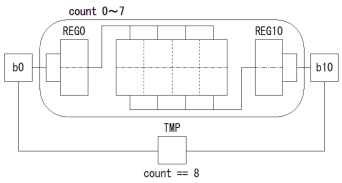

図6: 奇数の場合シングルポートメモリ

```
1  if(count <= 7){
2    シングルポートメモリによる循環バッファの記述
3  }
4  if(count == 8){
5    b10 = TMP;
6    TMP = b0;
7  }
8  count ++;
9  if(count == 9) count = 0;
```

図7: メモリの要素数が奇数だった場合のシングルポートメモリによる循環バッファの記述

4 評価と考察

最適化適用前のオリジナルコード(org)と、既存のスカラリプレイス適用後(2port)の結果を比較対象として、提案手法であるシングルポートメモリによる循環バッファを用いたスカラリプレイスの評価結果を表1に示す。表中の百分率はオリジナルコードを分母としたときの割合である。

表1: 評価結果

例題	jac			seidel-2d		
種類	org	2port	1port	org	2port	1port
サイクル数 (kcycles)	49.0 (100%)	10.0 (21%)	10.1 (21%)	860.2 (100%)	172.0 (20%)	177.4 (21%)
回路 (kgates)	13.9	26.8	31.6	21.0	33.9	39.5
内部メモリ (kgates)	0	97.8	65.2	0	61.5	41.0
合計面積 (kgates)	13.9 (100%)	124.6 (896%)	96.8 (696%)	21.0 (100%)	95.4 (454%)	80.5 (383%)
例題	image			jacobi-2d		
種類	org	2port	1port	org	2port	1port
サイクル数 (kcycles)	52.1 (100%)	5.4 (10%)	5.4 (10%)	49.2 (100%)	15.9 (32%)	15.9 (32%)
回路 (kgates)	63.3	91.3	110.9	24.5	33.3	46.6
内部メモリ (kgates)	5569.2	208.7	139.1	1463.6	153.2	101.5
合計面積 (kgates)	5932.5 (100%)	300.0 (5%)	250.0 (4%)	1488.1 (100%)	186.5 (13%)	148.1 (10%)

スカラリプレイス適用時に、循環バッファにデュアルポートメモリを用いたときと、シングルポートメモリを用いたときではオリジナルに対して同程度の性能向上が見られた。

jacとseidel-2dはスカラリプレイス適用後に発生する循環バッファで初めて内部メモリを持つため、回路面積が増大している。しかしシングルポートメモリを使用した場合はデュアルポートメモリを使用したときに比べて増加率を平均136%削減されている。

imageとjacobi-2dはスカラリプレイス適用によって元々含まれていた内部の一時バッファが削除されたため、適用後のほうが面積が削減されている。imageとjacobi-2dについてはデュアルポートメモリを使用したときの合計面積に比べ、シングルポートメモリを使用したときでは平均19%削減された。

5 まとめ

既存のスカラリプレイス技術は配列アクセスの削減によって速度向上が得られたが、デュアルポートメモリによる循環バッファが必要となり、面積の増大が問題であった。本稿では今まで使用していたデュアルポートメモリを、メモリの要素数が偶数の場合と奇数の場合に分け、シングルポートメモリに置き換える手法を提案した。結果として提案手法のシングルポートメモリによる循環バッファを用いたスカラリプレイス適用により、デュアルポートメモリを使用した既存の手法と比べて性能はそのままに、面積を平均19%削減した。

6 参考文献

参考文献

[1] 竹鼻宏晃, 瀬戸謙修,“配列アクセス実行条件の厳密な解析に基づくスカラリプレイス技術”デザインガイア 2012 -VLSI 設計の新しい大地- VLD2012-60, pp.7-12, 2012,11

[2] Byoungro So and Mary Hall,“Increasing the Applicability of Scalar Replacement,”13th Inter-national Conference on Compiler Construction,CC 2004.

[3] Michael Fingeroff,“High-level Synthesis Blue Book”, Xlibris Corp pp.159-190 2010,9,29