科学研究費助成事業　　研究成果報告書

平成　２６　年　　６　月　１８　日現在

機関番号：　３２６７８
研究種目：　若手研究(B)
研究期間：　2012～2013
課題番号：　２４７０００５４

研究課題名（和文）マルチプロセッサＳｏＣ向けカスタム命令自動生成技術

研究課題名（英文）Custom Instruction Generation for Multiprocessor SoCs

研究代表者
瀬戸　謙修（Kenshu, Seto）

東京都市大学・工学部・講師

研究者番号：１０４２０２４１

交付決定額（研究期間全体）：（直接経費）　　2,600,000円、（間接経費）　　780,000円

研究成果の概要（和文）：高位合成におけるループパイプライン化では開始間隔の削減が性能向上の鍵となるが、従来技術では、発生するかどうか実行前に不明なRAW依存関係を常に発生すると想定するため、開始間隔が増大する課題があった。本研究では、そのような依存関係において、メモリに書き込む値をレジスタにも書き込み、依存関係の発生を実行時にチェックし、発生時にはレジスタをアクセスすることで、開始間隔を短縮する技術を開発し、最新のループパイプライン化技術と比べて実行時間を平均４０％削減できた。

研究成果の概要（英文）：In the loop pipelining of high-level synthesis, the reduction of initiation intervals (IIs) is the key for high-performance. Traditional loop pipelining techniques, however, assume that the RAW dependences whose occurrences are unknown before execution always occur, resulting in increased IIs. In this research, we developed a technique that reduces IIs. In this technique, data written to memories in such dependences are also written to registers and the occurrences of the dependences are checked at runtime and the registers are accessed in case the dependences occur. We demonstrated that our technique reduces the numbers of execution cycles by 40% on average compared to the state-of-the-art loop pipelining technique.

研究分野：　総合領域

科研費の分科・細目：　情報学・計算機システム・ネットワーク

キーワード：　SoC　高位合成　ループパイプライン化

１．研究開始当初の背景

最先端 SoC ではプロセッサを多数搭載することで、高速化や省エネ化を実現している。そのような多数のプロセッサを含むシステムオンチップ(SoC)である MPSoC において、主要部品の一つがアプリケーションに特化した処理を効率良く実行するカスタムプロセッサであり、SoC の高機能化によりカスタムプロセッサの搭載個数は今後ますます増大することが見込まれるため、高性能で省エネなカスタムプロセッサを素早く設計することが求められている。カスタムプロセッサはループ処理の高速化を担うことが多く、ループ処理のデータ依存グラフが循環グラフとなる場合が少なくないが、そのような場合に従来技術によってカスタムプロセッサを自動生成した場合に性能向上が不十分である問題があった。

２．研究の目的

本研究の目的は、ループ処理のデータ依存グラフが循環グラフの場合でも十分なループ高速化が可能な、新しいカスタムプロセッサ自動生成技術を開発することである。本研究におけるカスタムプロセッサの自動生成では、C言語からハードウェアを自動生成可能な高位合成、特にループパイプライン化機能を組合せて使用する。

３．研究の方法

ループ処理に対するカスタムプロセッサ自動生成において、ループ処理のデータ依存グラフが循環グラフとなる場合に、開始間隔削減に必要な人手による最適化技術を検討した。高位合成ツールの内部コードに手を加えて、最適化アルゴリズムを実装することは難しいため、ループ処理に対してC記述レベルでの最適化技術を適用し、最適化されたC記述に対して高位合成のループパイプライン化機能を適用することでカスタムプロセッサを自動生成する方法をとった。サイクル数の評価についてはサイクル精度の RTL シミュレーションを使用し、面積やクリティカルパス遅延の評価については、0.45um のテクノロジライブラリを使用しクロック制約 250MHz のもとで高位合成によって出力された RTL 記述を論理合成することにより実施した。開発した人手最適化を、ツールによって自動適用できるようにするため、アルゴリズムの検討と実装を行った。開発したツールをいくつかのループに適用し、ツールによって自動最適化されたC記述を高位合成、論理合成を用いてハードウェア化し、開始間隔、サイクル数、面積、クリティカルパス遅延を、従来技術によって生成したハードウェアのものと比較することで、本研究の効果を確認した。さらに、依存関係の発生が実行前に決定できない場合のループパイプライン化技術として、パイプラインをストールさせることで開始間隔の増大を防ぐ技術(参考文献[1])が本研究実施期間中に発表されたため、その技術と本研究との比較も行い、本研究の優位性を確認した。

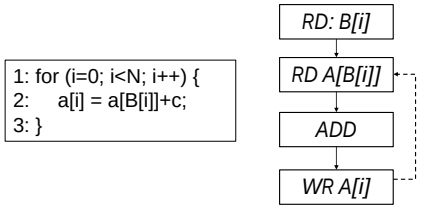

図１: 発生の有無を実行前に判定不可能な RAW 依存関係を持つループ(右)とそのデータ依存グラフ(循環グラフ)(左)

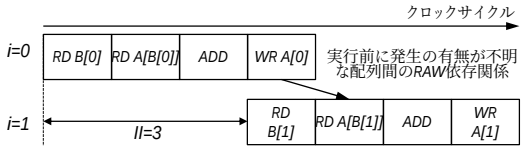

図２: 従来のループパイプライン化技術による開始間隔の増大

４．研究成果

本研究はまず、図１(右)に示すループ(データ依存グラフを図１(左)に示す)を例にとり研究を進めた。従来技術では、発生の有無が実行前に判定不能なループ内の RAW 依存関係を常に発生すると想定するため、開始間隔が増大し、性能向上が不十分である問題があり、本研究では特に、この問題の解決に取り組んだ。図２に、図１のコードを従来のループパイプライン化技術によってスケジュールした様子を示す。図２に示すように、ループ変数 i が i=0 の時の配列 A への書き込み(WR A[0])と、i=1 の時の配列 A からの読み出し(RD A[B[1]])との間の RAW 依存関係が、実行前に発生の有無が不明な配列間の RAW 依存関係であり、従来技術では、B[1]==0 となり依存関係が発生するかもしれないため、RD A[B[1]]を WR A[0]の後にスケジュールする必要が発生し、依存関係が発生しない場合には開始間隔 II=1 でスケジューリングできるにも関わらず、II は常に 3 に増やす必要があった。図２の開始間隔増大の問題を解決する方策として、本研究開始後に、発生の有無が実行前に判定不能な依存関係を実行時にチェックし、依存関係が発生した場合にはパイプラインをストールすることで、開始間隔を短縮する技術が提案された[1]。

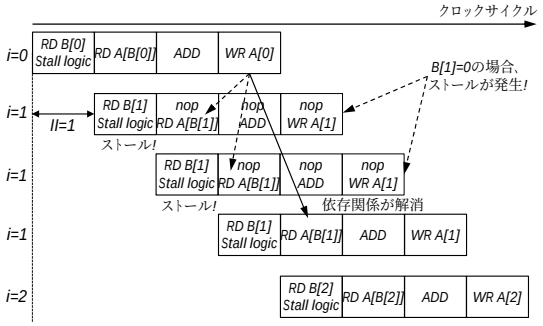

図３: 発生の有無を実行前に判定不可能な依存関係を持つループに対しストールにより開始間隔を短縮する既存技術

```
1: for (i=0; i<N; i++) {
2:    A_raddr = B[i];
3:    if (A_raddr==A_waddr_2)       A_rdata = A_wdata_2;
4:    else if (A_raddr==A_waddr_1) A_rdata = A_wdata_1;
5:    else                                  A_rdata = A[A_raddr];
6:    A_waddr_0 = i;
7:    A_wdata_0 = A_rdata + c;
8:    A[i] = A_wdata_0;
9:    A_waddr_2 = A_waddr_1;
10:   A_waddr_1 = A_waddr_0;
11:   A_wdata_2 = A_wdata_1;
12:   A_wdata_1 = A_wdata_0;
13: }
```

図４: 図１のループに提案する最適化を適用した結果

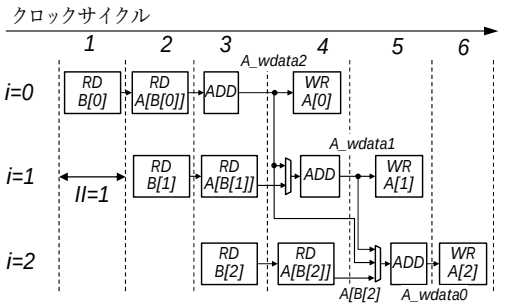

図５：提案手法によるループパイプライン化とフォワーディングユニット生成

この関連技術を調査した結果、図３に示すように開始間隔 II を３から１に削減できるが、ストールの発生が頻繁に起こる場合には、ストールに起因する遅延時間の増大が問題となることが分かった。図３の例で、関連技術[1]がストールを起こす原因は、RAW 依存における配列への書き込み WR A[i]の直後の繰り返しで発生する配列の読出し RD A[B[i]]である。これらの配列アクセスは高位合成において通常同期メモリへのアクセスとして実現されることが多く、結果として２クロックの遅延が発生する。

本研究では、従来技術[1]のメモリアクセス遅延を削減するため、配列への書き込み WR A[i]でメモリに書き込むデータを、メモリより遅延の少ないレジスタにも書き込み、依存関係が発生した場合には、メモリではなくレジスタにアクセスすることで、ストールを防止し、実行サイクル数を削減するソースコードレベルの最適化方法を考案した。図４に、提案した最適化結果を示す。提案技術では、図１の例に対し、発生の有無を実行前に判定不可能な RAW 依存関係における配列読み出しアクセス RD A[B[i]]のアドレスと配列書き込みアクセス WR A[i]のアドレスおよびそのアドレスに書き込まれるデータをそれぞれレジスタ変数 A_raddr, A_waddr_0, A_wdata_0 に保持する。また、A_waddr_k, A_wdata_k はそれぞれ k 回前のループで配列書き込みアクセス WR A[i]により書き込まれたアドレスおよびデータを表す。現在の繰り返しにおける配列読み出しのアドレス(A_raddr)と、以前の配列書き込みのアドレス(A_waddr_k)を比較し、比較結果に応じて、メモリではなく適切なレジスタ A_wdata_k を参照することで、コードの機能を等価に保ちつつ、パイプラインストールを発生させることなく開始間隔を１に削減できることが分かった。図５に提案する最適化によるパイプライン実行の様子を示す。メモリへの書き込みと読み出しを介さず、加算器(ADD)の入力マルチプレクサを適切に切り替え、以前の加算結果を格納したレジスタを直接参照することで、性能向上を実現しており、古典的なプロセッサにおけるフォワーディング技術[2]に相当する。このような最適化をはじめて自動で実行するアルゴリズムを開発し(雑誌論文①に掲載)、ツールを実装した。

表１：本研究成果と従来技術との比較

例題	コードの種類	開始間隔	実行サイクル数	クリティカルパス長	回路面積
example	元コード	1.00	1.00	1.00	1.00
	文献[1]	0.33	0.61	1.10	2.10
	提案	0.33	0.34	1.10	2.21
histgram (向日葵)	元コード	1.00	1.00	1.00	1.00
	文献[1]	0.33	0.43	1.00	2.05
	提案	0.33	0.34	1.00	2.83
histgram (夜景)	元コード	-	1.00	-	-
	文献[1]	-	0.78	-	-
	提案	-	0.34	-	-
SMVM	元コード	1.00	1.00	1.00	1.00
	文献[1]	0.29	0.80	1.02	1.15
	提案	0.43	0.53	1.00	1.08
平均	元コード	1.00	1.00	1.00	1.00
	文献[1]	0.32	0.65	1.04	1.77
	提案	0.37	0.39	1.03	2.04

表１に、本研究成果と従来技術(文献[1])の比較結果を示す。表中、「元コード」は、最適化を適用する前の元々のコードをループパイプライン化、論理合成したときの結果を示す。表中の結果はすべて「元コード」の場合に対する比率を表す。example が図１の例題、histgram が画像のヒストグラム算出、SMVM が疎行列ベクトル積のループである。histgram(向日葵)と histgram(夜景)の回路は同一だが、異なる画像を入力した。向日葵の画像は輝度値の変化が大きく、夜景の画像は輝度値の変化が小さいため、夜景の場合にストールが頻発し、実行サイクル数が増大した。提案手法は現在までの最新技術[1]と比べ、ク

リティカルパス長はほぼ同一で、１５%の面積増加に対して、実行サイクル数を４０%削減できたため、本研究成果はカスタムプロセッサの高性能化に効果を発揮する技術であると結論づけられる。

本研究成果によりエンジニアが、様々なアプリケーションを今までと比べて少ない労力で設計できるようになり、またより多くのエンジニアが、今までと比べて高性能なハードウェアを設計できるようになるため、産業上重要な技術を開発できたといえる。

参考文献

[1] M. Alle, A. Morvan, S. Derrien, "Runtime Dependency Analysis for Loop Pipelining in High-Level Synthesis," Design Automation Conference, 2013, pp. 1-10

[2] D. Patterson, J. Hennessy, "Computer Organization and Design: The Hardware/ Software Interface," Fourth Edition, Morgan Kaufmann Publishers, 2008

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計１件）

① S. Kusakabe, K. Seto, "Forwarding Unit Generation for Loop Pipelining in High-Level Synthesis," IPSJ Transactions on System LSI Design Methodology, 査読有, Vol. 8, 2013, 印刷中(ページ番号未定), https://www.jstage.jst.go.jp/browse/ipsjtsldm

〔学会発表〕（計３件）

① 日下部真吾, 瀬戸謙修, "ループパイプライン化における開始間隔短縮のためのデータ依存緩和技術," 2013-SLDM-161(10), 査読無, 2013, pp. 1-6
② S. Kusakabe, K. Seto, "Forwarding Unit Generation with Runtime Dependency Analysis in High-Level Synthesis," Proceedings of SASIMI 2013, 査読有, 2013, pp. 226-230
③ 日下部真吾, 外山知人, 瀬戸謙修, "高位合成のループパイプライン化におけるフォワーディングユニット生成技術," 信学技報, Vol. 113, No. 320, 査読無, 2013, pp. 245-249

〔図書〕（計０件）

〔産業財産権〕
○出願状況（計０件）

名称：
発明者：
権利者：
種類：
番号：
出願年月日：
国内外の別：

○取得状況（計０件）

名称：
発明者：
権利者：
種類：
番号：
取得年月日：
国内外の別：

〔その他〕
ホームページ等

６．研究組織
(1)研究代表者
瀬戸　謙修（SETO KENSHU）
東京都市大学・工学部・講師
研究者番号：１０４２０２４１

(2)研究分担者
（　　）

研究者番号：

(3)連携研究者
（　　）

研究者番号：