社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－ＳＬＤＭ－126（２）
2006／10／26

Ｃ言語からのフラクタル型マルチプロセッサ自動合成システム

西沢 政則† 白井 雄一† 大角 嘉蔵† 西門 秀人†

加藤 俊之†† 山内 寛紀† 小林 士朗†††

†立命館大学理工学部 〒525-0058 滋賀県草津市野路東 1-1-1
††日本ケイデンス・デザイン・システムズ社 〒222-0033 神奈川県横浜市港北区新横浜 3-17-6
†††旭化成株式会社 〒243-0021 神奈川県厚木市岡田 3050

E-mail: †ro007002@se.ritsumei.ac.jp, ††tkv21003@se.ritsumei.ac.jp, †††kobayashi.sb@om.asahi-kasei.co.jp

あらまし Ｃ言語アプリケーションプログラムからマルチ階層型プロセッサを生成する手法を提案する．このプロセッサをフラクタル型プロセッサと名付ける．まず，Ｃ言語アプリケーションプログラムを解析し，そのアルゴリズムの並列度，演算量を階層的に求めCDFGを生成する．このCDFGよりプロセッサのアーキテクチャを決定し，あらかじめ用意しておいたプロセッサIPを組み合わせることでプロセッサを生成する．また，Ｃアプリケーションプログラム独自の頻出演算を専用演算器として生成することによって，高速な演算が可能なプロセッサとする．同時にプロセッサ上で動作するソフトウェアも生成し，メモリとして組み込む．このようにＣ言語プログラムから自動でプロセッサを生成する手法について述べる．

キーワード 高位合成，システムデザイン，マルチプロセッサ，DSP

A multi DSP generation system with fractal structured architecture from C language

Masanori NISHIZAWA† Yuichi SHIRAI† Yoshizo OSUMI† Hideto NISHIKADO†

Toshiyuki KATO†† Hironori YAMAUCHI† and Shiro KOBAYASHI†††

† Department of Science and Engineering, Ritsumeikan Uniberisity 1-1-1 Nojihigashi, Kusatsu, Shiga 525-8577 Japan
†† Cadence Design Systems, Japan 3-17-6 Shinyokohama, Minato-ku, Yokohama, Kanagawa, 243-0021 Japan
††† Information Technology Laboratory, Asahi Kasei Corporation 3050 Okada, Atsugi, Kanagawa 243-0021 Japan

E-mail: †ro007002@se.ritsumei.ac.jp, ††tkv21003@se.ritsumei.ac.jp, †††kobayashi.sb@om.asahi-kasei.co.jp

Abstract In this paper, we propose a method of behavior synthesis from C language to HDL. The method consists of tree parts. First, Control Data Flow Graph (CDFG) is generated, as a result of dependency analysis of the application program written by C language. Secondly, CDFG is optimized to execute the complicated calculations quickly. Finally, hardware image written by HDL is generated. This paper is written about hardware generation in particular.

Keyword high level synthesis, system design, multi processors, DSP

1. はじめに

近年，半導体プロセス技術は目覚しく向上してきている．今までLSIはシステムのほんの一部分を担当するにすぎなかった．しかし，現在ではLSIはシステム全体を含むまでに成長した．しかし，半導体の集積密度の向上率にくらべ，設計の生産性はそれほど向上していない．また，消費者は高機能かつ高性能な製品を望んでいる．よってLSIを短期間で設計することが求められている．

そこで私たちは，システムLSI設計者の設計負担の軽減し，短期間での開発を可能とすることを目的として，Ｃ言語で記述されたアプリケーションからDSPのハードウェアイメージとその DSP 上で実行される実行コードを同時に生成，合成する自動合成システムの研究を行っている．本研究で生成するDSPは階層型マルチプロセッサである．私たちはフラクタル型プロセッサと呼んでいる．DSP自動合成システムは大きく分けてCDFG生成部とCDFG最適化部，DSPハードウェア生成部の３つからなる．DSP自動合成システムの概要と処理フローを２章で説明する．CDFGについては３章で説明する．CDFG最適化については４章で説明する．そして，DSPハードウェア生成については５章

で説明する．最後にまとめと今後の展望を6章で説明する．

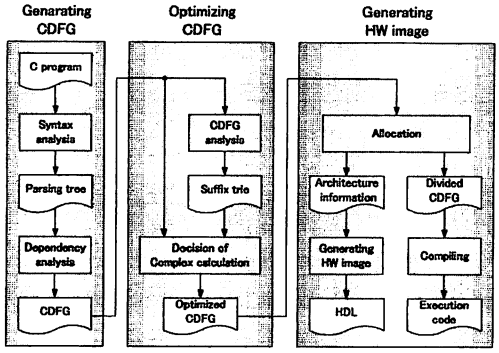

図1 DSP自動合成システムの処理フロー

2. DSP自動合成システム

提案するDSP自動合成システムはディジタル信号処理用ハードウェアの設計期間の短縮およびハードウェア設計者の負担を軽減することを目的としたものである．システム設計者は所望する機能が記述されたCアプリケーションプログラムを当システムに与えることにより，そのプログラムのアルゴリズムに最適な信号処理ハードウェアの構成とそのハードウェア上で動作する実行コードを得ることができる．

図1にDSP自動合成システムの処理フローを示す．当システムは，大きく分けてCDFG生成部とCDFG最適化部，ハードウェア生成部の3つからなる．CDFG生成部では，入力となるCアプリケーションプログラムの構文解析を行い，構文解析木を作る．そして，この構文解析木の依存解析を行ってCDFGというデータフローグラフを生成する．CDFG最適化部では，suffix trieを用いてCDFGの解析を行い，複合演算を抽出する．この複合演算の情報はプロセッサ内の演算器を生成するときに必要となる．そして，ハードウェア生成部は最適化されたCDFGを元にハードウェア構成を決定し，HDLを出力する．また，そのプロセッサ上で実行するための実行コードも同時に生成する．

3. CDFG生成部

まず，CDFGは何かについて説明する．CDFGはData Flow Graph (DFG)を拡張したものである．DFGとはデータの流れを表したグラフである．DFGの例を図2に示す．CDFGは，このDFGに分岐や繰り返し等の制御情報を加えたグラフである．一般的なCDFGを図3に示す．しかしCDFGには特定の書式は決まっていない．例えば[1]-[4]があげられる．本自動合成システムではDSP自動合成に適した独自のCDFGを提案している．本研究室で提案するCDFGの例を図4に示す．このCDFGの特徴は以下の通りである．

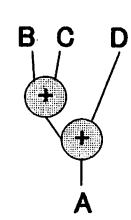

図2 DFGの例

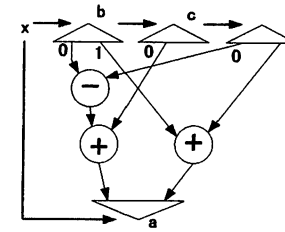

図3 一般的なCDFG

- Cプログラムの階層構造の維持

図4に示した四角の枠は階層を現している．C言語ではif文やfor文などの制御処理は入れ子状に幾つでも重ねる事が可能である．そこで，私たちのCDFGはC言語の階層構造を維持している．これによりCDFGとソフトウェア記述の対応関係が理解しやすい．CDFGからC言語記述へのフィードバックが容易である．

- ラベルを付加する

ラベルを付加する各階層の右上に四角い枠に囲まれた表記がされている．各階層に処理によって名前を付けている．関数全体はFunction Block (FB).分岐処理はSelection Block (SLB).ループ処理部はLoop Block (LB)及びループ毎の処理をIteration Block (ITB).演算部をStatement Block (SB).LBの右上の表記はループ条件を表している．

- 演算解析を行う

演算解析を行う各階層に演算量と並列度を算出する．これによりHWアーキテクチャの決定を手助けすることが出来る．

- アドレス演算部とデータ演算部の識別

アドレス演算部とデータ演算部の識別演算部SBでは演算子を表すノードに円と五角形を用いている．これはアドレス演算部とデータ演算部を区別するためである．五角形のノードはアドレス演算となっている．丸いノードはデータ演算である．アドレス演算とデータ演算を分離しているため処理の流れを理解しやすい．また DSP の様にアドレス計算器を持つアーキテクチャに対して有用な情報を与えることが可能である．

- 演算の重み付け可能

演算の重み付け可能各演算に対して演算量を定義することができる．このため，特殊な演算器（積和演算器等）を持つアーキテクチャにおいても演算量をシミュレート可能である．

- 複合演算抽出が容易

複合演算抽出が容易演算子ごとにラベル，上位階層へのポインタ，識別子を付加している．こうする事で

複合演算抽出が行いやすい．4章において複合演算抽出について述べる．

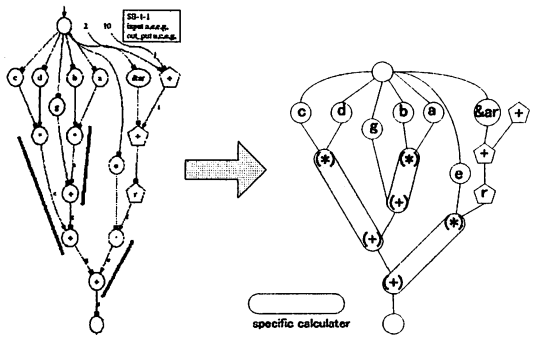

図5 複合演算の決定

4. CDFG 最適化部

DSP は固有の演算器として，積和演算器を持っている．DSP が信号処理を高速に行える理由の一つが，この積和演算器である．つまり，何度も出現する演算用の専用演算器を持つ事で，高速に実行可能な HW を生成することが出来る．本研究では Suffix Trie を用いた頻出演算パターン抽出手法を使用する．頻出演算パターン抽出手法について説明する．

Suffix Trie を用いる事で，頻出パターンを検出することができる．この結果をハードウェア自動生成に反映する．Suffix Trie から専用演算器として搭載する演算パターンを決定する．その演算を複合演算と呼ぶ．その後，基本演算（+,*,- etc）だけの CDFG に複合演算を加える．図5に様子を示す．

Suffix Trie から“* → +”のパターンが3回存在していることを知ることができる．そのパターンを複合演算とする．そこで CDFG 上の”*→+”パターンを複合演算に置き換える．

この後，複合演算が採用された CDFG から HW 生成を行う．そうすれば高速に実行できる HW を生成することが可能となる．

5. ハードウェア生成部

フラクタル型プロセッサは階層型マルチプロセッサである．フラクタル型プロセッサの構成図を図6に示す．ループ繰り越し依存がないループブロックを下位の Processor Element(PE)に割り当てる．プロセッサレベルで並列処理を行う．ループが入り子になっている場合には，さらに下位の PE に処理を分配する．これによって，C 言語ソースのどこが並列処理可能かを知ることができ，またそのときの演算量と並列度をシミュレートすることができる．

HDL 生成部の処理フローは大きく分けて，アロケーション，プロセッサ生成，そして実行コード生成の3つに分けられる．各部分について順番に説明する．

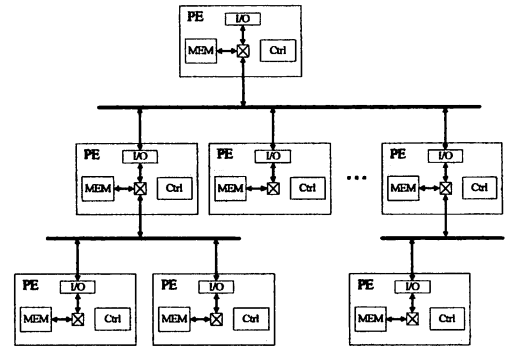

図6 フラクタル型プロセッサ構成図

5.1. アロケーション

アロケーションは CDFG を各 PE に割り当てる処理を行う．まず，CDFG から並列に処理できる部分を見つけ出して分割する．並列に処理できる部分は多数あるが，大きな並列化が見込めるループブロックを対象とする．ただし，ループ繰り越し依存があるループブロックは並列化ができないので除外する．また，ループブロック内の処理が一定の演算量以下の場合は下位のプロセッサへ割り当てない．これは，ループ内の処理が少ないと PE 間の通信のオーバヘッドが増えてしまうためである．ループブロックをいくつの PE で実行するかはユーザーが指定することができる．しかし，最初の状態では適当な数で生成される．

CDFG がどの PE で実行されるかが決まると，PE ごとに CDFG が分割される．そして，PE 間のデータの受け渡し情報が記録される．この時点で通信の情報と同期のタイミングが決定する．

5.2. HDL 生成

ここでの作業は大きく2つある．1つは PE ごとに演算器やレジスタ等を生成することである．もう1つは各 PE を，バスを使って繋ぐことである．

まず PE ごとの演算器等の生成について説明する．ここで生成される演算器は，汎用の演算を行う ALU，アドレス計算器，それと専用演算器である．専用演算器とは CDFG 最適化部で抽出した複合演算器である．これらの1つ又は複数の組み合わせから PE の演算部を生成する．これら演算器は，VLIW 型の形態をとっており，各演算器に対して 1 つの命令を割り当てて並列に実行することができる．1つの PE 内に ALU を複数生成することも可能だが，本研究のフラクタル型プロセッサでは，ALU が1つとアドレス計算器が3つを基本として生成する．アドレス計算器が3つあるのは，2つの

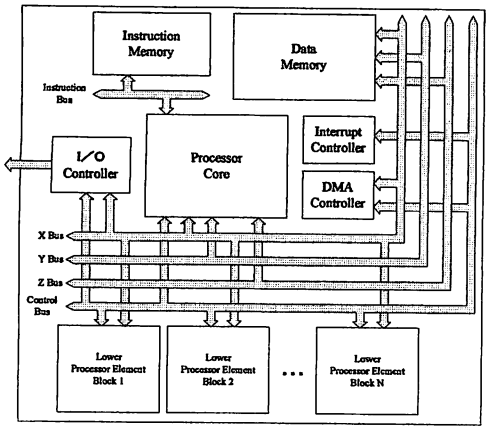

図 7 プロセッサエレメントのブロック図

入力元アドレスと1つの出力先アドレスを同時に計算するためである. PE のブロック図を図 7 に示す. また, プロセッサコアのブロック図を図8に示す. IR_EXE は演算処理の命令を保持するレジスタである. IR_AC_X は X バス用のアドレス計算命令を保持するレジスタである. IR_AC_Y は Y バス用のアドレス計算命令を保持するレジスタである. IRAC_Z は Z バス用のアドレス計算命令を保持するレジスタである. IR_MA はメモリアクセス命令を保持するレジスタである. これらのレジスタには図 9 に示す 80 ビットの命令の対応する部分が書き込まれる. 先頭から演算処理命令 24 ビット, アドレス計算命令 16 ビット×3, メモリアクセス命令 8 ビットである. 各命令の詳細は図 10 に示す. それぞれの命令は, レジスタタイプ, 即値タイプの 2 種類がある.

こうして生成された PE を, バスを用いて1つのプロセッサとして接続することで, フラクタル型プロセッサが生成される.

5.3. ソフトウェア生成

PE ごとに分割された CDFG に対して, まずは演算部とアドレス計算部を分離する. そして, それぞれ並列にできる演算を演算器に割り当てる. このスケジューリング次第で効率のいいコードが生成されることになる. この演算器のスケジューリングは, データの依存関係がはっきりしていて比較的行いやすい. これは CDFG を用いている理由の 1 つである.

こうして PE ごとに, 演算器に割り当てられたコードをコンパイルして実行コードを生成する. また, PE 間通信で必要となる割り込みベクタのコードや, PE 間通信でのデータ転送用のコードを生成する. そして, これらのコードを ROM として生成し, フラクタル型プロセッサの PE へ組み込んで, プロセッサが完成する.

6. まとめと今後の展望

現段階では, ループ部分の PE による並列化と, 複合演算による専用演算器の作成を行っている. しかし, この情報をいかにユーザーにフィードバックするか, および, ユーザーがどのような情報を欲しているのかを十分に吟味する必要がある. それによってユーザーが期待する理想のハードウェアを生成できるようにしなければならない. また, 多入力ポートを持った演算器など, 作成できる専用演算器の種類を増やすことが必要である. そして, もっと複雑な演算にも対応できるプロセッサの生成を目標とする.

また, このフラクタル型プロセッサはあくまで, DSP 自動合成の途中段階である. 実用化するには通信のオーバヘッド等の無駄な部分が多い. よって, 今後より効率のよいプロセッサを生成できるようにする必要がある.

文　献

[1] Y.Miyaoka, N,Togawa, M.Yanagisawa, T.Ohtsuki, "A Hardware/Software Cosynthesis Algorithm for Processors with Heterogeneous Datapaths," IEICE Trans. Fundamentals, vol.E87-A, no.4, pp.830-836, Arp.2004.

[2] N.Ohsawa, O.sakamoto, M.Hariyama, M.Kameyama, "Design of a Field Programmable VLSI Processor Based on Bit-Serial-Pipeline Architectures," IEICE technical report. Image engineering, vol.103(384), pp.53-57, Oct.2003.

[3] T.Ishii, N.Togawa, M.Yanagisawa, T,Ohtsuki, "A Control/Data Flow Graph Transformation Algorithm in High-Level Synthesis for Control-Based Hardwares," Technical report of IEICE. VLD, vol.101(695), pp.41-48, Mar,2002.

[4] N.Ohsawa, M.Hariyama, M.Kameyama, "Architecture of High Performance Field Programmable VLSI Processor," Technical report of IEICE. ICD, vol.101(249), pp.23-30, Jul.2001.