C-002

CDFGからの高速な複合演算抽出手法
Combine operation pattern extraction from CDFG for DSP generation

加藤 俊之† 三宅 貴章† 宮内 崇旭† 西門 秀人† 山内 寛紀† 小林 士朗††
Kato Toshiyuki† Miyake Takaaki† Miyauchi Tkaaki† Nishikado Hideto† Yamauchi Hironori† Kobayashi shirou††

1. はじめに

現在，携帯電話をはじめとする組み込みシステムの発展が著しい．各製品の多機能化が進み開発サイクルも短くなっている．そのためLSIの開発負担の増加が問題となり,抽象度の高いLSI開発手法の研究が盛んとなってきている．現在では，C言語の動作記述から回路を自動生成する動作合成が開発現場で用いられる様になってきている．しかし，既存の手法[1]ではHW処理部とSW処理部とを人が指定しなければならないためSoCやDSP(Digital Signal Processor)等のLSIの開発に用いる事は出来ない．現在，本研究室では特定用途向けであるプロセッサ,DSPを自動合成のターゲットとしてHW自動合成システムの研究を進めている．

本研究室の提案するDSP自動合成システムをはじめ，HW自動生成手法にはCDFGを用いる手法が多く存在する[2][3]．しかし，特殊演算器等のHW処理は人が指定しなければならない．そこで今回CDFGから頻出パターンを発見し頻出演算パターンを求める手法を提案する．この手法を用いる事で自動的に複合演算を求める事ができHW/SW分割手法に用いる事ができる．2.で本研究室の提案するDSP自動合成システムの概要を述べる．3.でCDFGについての説明を行い，本手法で用いる最右拡張について4.で紹介する．そして５.にて本手法の説明を行い，実装結果を6.で紹介する．

2. DSP自動合成システム

提案するDSP自動合成システムはDSPハードウェアの設計期間の短縮を目的としたものである．設計者は所望する機能が記述されたC言語記述を当システムに与える．そうすれば，本システムが記述されたアルゴリズムに最適なDSPのハードウェアの構成と実行コードを出力する．

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，入力されたC言語記述を解析し、構文解析木を作る。次に，この構文解析木の依存解析を行う．その結果からCDFGを生成する。CDFG最適化部では、CDFGから性能向上が多く見込める演算パターンを発見し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ではCDFGと複合演算の情報を元にハードウェア構成を決定する．出力はHDLとする。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。

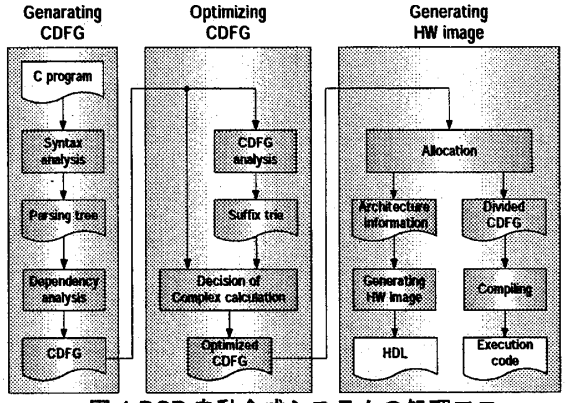


図１DSP自動合成システムの処理フロー

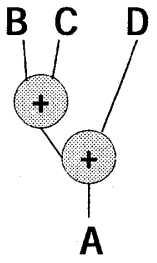
3. CDFG

3.1. CDFG

DFG(Data Flow Graph)とはデータの流れを表したグラフの事をいう．

DFGの例を図２に示す．円で囲まれた演算子は演算処理を表し，各円を繋ぐ線はデータの流れを表している．このDFGはA=(B+C)+Dを表している．CDFGとは，このDFGに,分岐や繰り返し等の制御情報を加えたグラフの事をいう．一般的なCDFGを図３に示す．ここでの三角形は分岐処理を表しており，制御変数'x'の値によって異なる演算処理が行われる．このCDFGは'x'が０ならa=(b-d)+c; 'x'が１ならa=b+d; の処理が行われる事を示している．CDFGには特定の書式は決まっておらず多くの種類のCDFGが提案されている．本研究室でもDSP自動合成に適した独自のCDFGを提案しており，次項に本研究室の提案するCDFGの説明を行う．今回の複合演算抽出手法では，このCDFGを用いているが，本稿の提案手法はCDFG全てに用いることが可能である．

† 立命館大学大学院 Ritsumeikan University
‡ 旭化成株式会社研究開発本部 Aahikasei Corporation



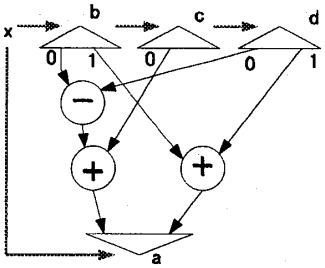


図 2 DFG　図 3 CDFG

3.2. 提案する CDFG

本研究室で提案する CDFG の例を図 4 に示す．図 4 における四角の枠は階層を表している．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．（いくらでもネストが可能）そこで，この CDFG では制御を階層構造を用いて表している．これにより CDFG と C 言語記述との対応関係が理解しやすいだけでなくマクロからミクロの可変の大きさで並列性を抽出することができる．各階層の右上に四角い枠に囲まれた表記がある．各階層には処理の種類によってラベルを付けている．関数全体は FB (function block)．分岐処理は SLB(selection block)．ループ処理部は LB（loop block）及びループ毎の処理を ITB (iteration block)．演算部を SB (statement block)としている．LB の右上にはラベル名と同時にループ条件を表記する．演算処理を表す SB では，演算子を表すノードに円と五角形が存在している．これはアドレス演算部とデータ演算部の CDFG を区別するためである．五角形のノードで表現された部分はアドレス演算部となっている．丸いノードで表現された演算はデータ演算である．アドレス演算部とデータ演算部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も最適な情報を得ることが可能となっている．

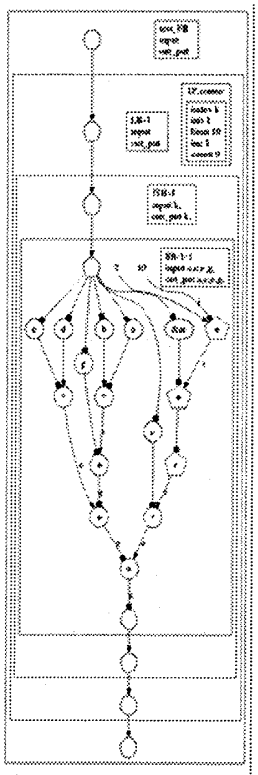


図 4 提案 CDFG

本研究室ではこの CDFG の自動描画システムの開発を行っている．そのため C 言語を入力として図 4 の様な CDFG を自動的に描画することができる．

4. 半構造データマイニング問題

データマイニング(Data Mining)とは，データベースに蓄積された大量のデータから，自明でない規則性やパターンを半自動的にとりだす方法についての科学的研究である．データマイニングは，現在，ビジネス分野や科学技術分野などのさまざまな対象分野で，その適用が盛んに行われている．

一方，高速なネットワークと安価な大容量記憶装置の発達によって，ウェブページやXML文書に代表される半構造データマイニング がネットワーク上に蓄積されており，大規模半構造データを対象としたデータマイニング（半構造データマイニング）に対する要求が高まっている。この要求に対し、浅井らは半構造データマイニング問題を，与えられた半構造データの集積から出現頻度の高い部分構造を発見する問題と定式化し，Bayard[8]の手法に基づき，この問題を効率よく解決する、最右拡張を用いたアルゴリズムFREQTを提案した[6,7]．我々は、最右拡張を用いてCDFG上のすべての複合演算パターンおよびその頻出回数を求める手法を提案する。

5. 最右拡張

最右拡張とは効率のよい順序木枚挙法である。ここで，順序木Sの最右拡張とは、Sの最右枝(根節点から最右葉に至る経路)に節点kを追加して得られる順序木Tである．ただし、kはT の兄弟関係において末弟であるように追加する。特に，Sの最右葉(最も右に位置する葉節点)rml(S)=k-1のp代前の親π(k-1)に、ラベル1をもつ節点kを追加して得られたSの最右拡張を、(p,1)拡張と呼び（図5），具体例を図6に示す。また、サイズ0のパターン空木を仮定し、任意の1-パターンは空木の最右拡張とする。

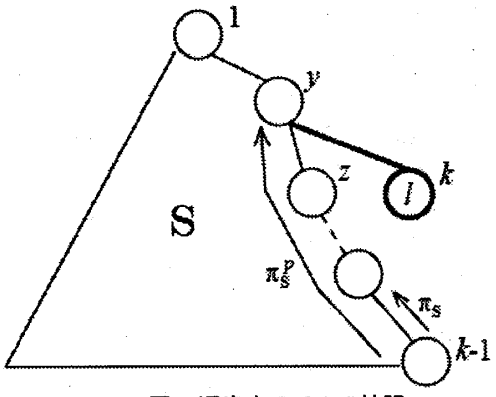


図５順序木Ｓの(p,l)拡張

最右拡張を繰り返すことにより，すべてのパターンを重複なく生成することができる．この順序木枚挙手法は，Bayard による集合枚挙木に基づくアイテム集合枚挙法を，順序木に拡張したものである．

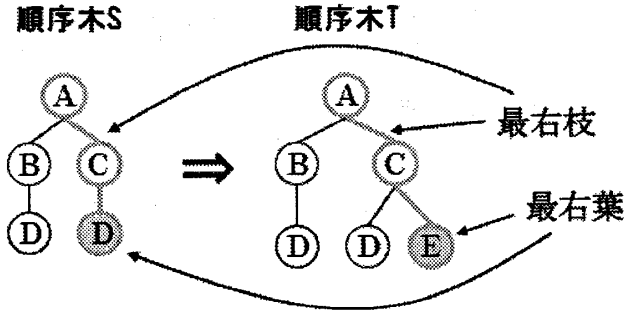


図６最右拡張の具体例

5.1. 頻出パターン抽出手法

頻出演算用に専用演算器を持つためには，頻出演算パターンを知る必要がある．本提案手法では，最右拡張を用いて，頻出パターンを抽出する．前述の通り最右拡張は効率のよい順序木枚挙法であるため,本手法では最右拡張を CDFG 向けに拡張して用いている．CDFG を最右拡張することで CDFG 上に存在する演算のパターン全てが羅列されるのだ．

5.1.1. 演算部(Statement Block)

CDFG は演算部と制御部に分かれるが,CDFG 上の制御部には演算は無い．そのため，まずは演算部（SB）における CDFG を最右拡張し、パターンを生成する．

最右拡張は最初に CDFG のデータ木を巡回しながら、最右拡張の対象となる CDFG のラベルの出現数と出現位置を調べる。CDFG にはデータを読み書きする r,w などの様々なラベルがあるので、今回は＋，－，＊，／，％の 5 つの演算のラベルに限定する。つまり、この 5 つの演算を組み合わせた複合演算を抽出する。次に、抽出したそれぞれの CDFG に対して最右拡張を行い、拡張した CDFG のみを新たな演算パターンとして保存する。これは、パターンとデータ木のマッチング情報を効率よく保存している。あるパターンがもつ全ノードの位置情報を保存するのではなく、パターンの最右葉 k=rml(T)の CDFG 情報だけを保存し、計算している。図 7 に演算部分のラベル＋をもつ CDFG を最右拡張した順序木枚挙グラフを示す。同じようにー，＊，／，％についても同じように拡張し図示できる。ただし、拡張し存在したパターンだけをグラフとして枚挙する。

5.1.2. 制御部

制御部にはループや分岐がある．ループ処理ではループ回数を CDFG が保持しているため，ループ回数を出現回数に乗算する．分岐処理では各分岐の分岐数に応じて出現回数を除算する．

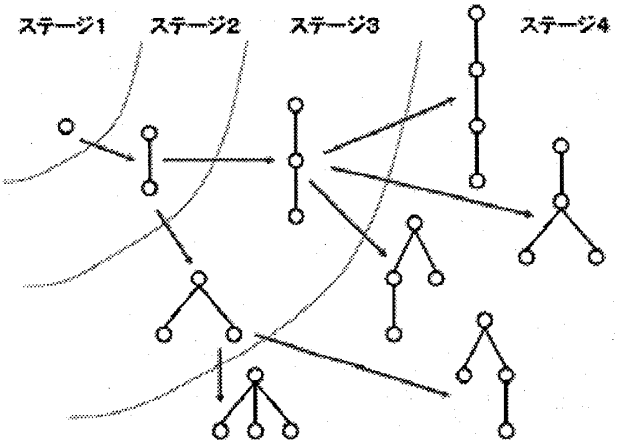


図７CDFG を最右拡張した順序木枚挙グラフ

5.2. 複合演算決定

最右拡張を用いて頻出パターンを知る事ができた．この結果から複合演算を決定する．順序木枚挙グラフには演算パターンと演算パターンの存在回数が出ているので，ここから「百回以上存在するステージ 3 の演算のパターン」や「ステージ 3 演算の最も多いパターン」などの条件を与える事で簡単に求める事ができる．

6. 結果

今回，C 言語記述を入力として，CDFG からシステムの試作を行った．本節ではその結果を紹介する．今回，評価として入力した C 言語記述は DCT（離散コサイン変換）である．DCT は最も有名な信号処理の一つであるために適していると考えた．入力した DCT の記述を以下の図 8 に示す．

```
void main(void)
{

 double cos_table[32]={1.0, 0.9807853, 0.92387953, 0.831469612, 0.70710678,
                       0.555570233, 0.3826834, 0.1950903, 0.0,
                       -0.1950903, -0.3826834, -0.555570233, -0.70710678,
                       -0.831469612, -0.92387953, -0.9807853, -1.0,
                       -0.9807853, -0.92387953, -0.831469612, -0.70710678,
                       -0.555570233, -0.3826834, -0.1950903, 0.0,
                       0.1950903, 0.3826834, 0.555570233, 0.70710678,
                       0.831469612, 0.92387953, 0.9807853};*/

 double cv[7]={0.70710678,1.0,1.0,1.0,1.0,1.0,1.0,1.0};
 double cu[7]={0.70710678,1.0,1.0,1.0,1.0,1.0,1.0,1.0};
 double dct_after[10][10];
 double dct_before[10][10];

  int x,y,u,v;
  double sum;
  for(v=0; v<8; v=v+1){
          for(u=0; u<8; u=u+1){
     sum=0;
     for(y=0; y<8; y=y+1){
                    for(x=0; x<8; x=x+1){
                                  sum =sum + dct_before[y][x] * cos_table[((2*x+1)*u)%32] *
cos_table[((2*y+1)*v)%32];
                    }
     }
          dct_after[v][u] = sum*cu[u]*cv[v]/4;

   }
  }
}
```

図 8 DCT 記述

このプログラムから生成した CDFG データ木から最右拡張により抽出したパターンはテキスト文書でパターンの型は PATTERN NUMBER、頻出回数出力は PATTERN COUNT で示される。

```
PATTERN NUMBER =101 : PATTERN COUNT =28992:+
PATTERN NUMBER =102 : PATTERN COUNT =0    :
PATTERN NUMBER =103 : PATTERN COUNT =24704:*
PATTERN NUMBER =104 : PATTERN COUNT =64   :/
PATTERN NUMBER =105 : PATTERN COUNT =8192 :%
PATTERN NUMBER =206 : PATTERN COUNT =8192 :*+
PATTERN NUMBER =207 : PATTERN COUNT =8192 :+*
PATTERN NUMBER =208 : PATTERN COUNT =4160 :**
PATTERN NUMBER =209 : PATTERN COUNT =64   :*/
PATTERN NUMBER =210 : PATTERN COUNT =8192 :*%
PATTERN NUMBER =211 : PATTERN COUNT =8192 :%+
PATTERN NUMBER =312 : PATTERN COUNT =8192 :*+*
PATTERN NUMBER =313 : PATTERN COUNT =64   :/
PATTERN NUMBER =314 : PATTERN COUNT =8192 :+*%
PATTERN NUMBER =415 : PATTERN COUNT =8192 :*+*%
PATTERN NUMBER =516 : PATTERN COUNT =8192 :*+*%+
```

7. まとめ

本稿では，CDFG からの最右拡張を用いた複合演算抽出手法の提案を行った．C 言語などの高位言語からの LSI 自動設計システムでは，一度高位言語を CDFG に変換して，そこから LSI を設計していく手法が多く提案されている．そのため CDFG の中から頻出演算パターンを自動的に求める手法があれば，専用演算器の指定や，HW/SW コデザインにおける HW/SW 処理の分離に用いる事が可能であると考えている．本手法では XML 文書などにしか使われていなかった最右拡張を CDFG という半構造データの領域に持ち込み，非常に効率的に頻出パターンの抽出を行っている．

文 献

[1] D. Gajski, A. Wu, N. Dutt, and S. Lin, " High-level Synthesis: Introduction to Chip and System Design." , Kluwer Academic Publishers, 1992.

[2] 川田容子,戸川望,佐藤政生,大附辰夫," 動作記述からのデータフローグラフ生成手法," 電子情報通信学会技術研究報告, VLD, vol95(307), pp55-62, Nove.1995.

[3] 浅井達哉, 有村博紀, "半構造データマイニングにおけるパターン発見技法", 電子情報通信学会論文誌, vol.J87-D1, no.2, pp.79-96,2004

[4] 中西恒夫, 中野猛, 福田晃, " 辞書式コード圧縮支援機構の遺伝的アルゴリズムによる最適化, " 電子情報通信学会技術研究報告, ARC, Vol.2001(39), pp.19-24,(2005).

[5] H. Arimura, S. Arikawa, S. Shimozono, Effcient discovery of optimal word-association patterns in large Text databases, New Generation Computing, 18, pp.-49-60, 2000

[6] 西口健一,石浦菜岐佐,西村啓成, 神原弘之, 冨山宏之, 高務祐哲, 小谷学, " ソフトウェア互換ハードウェアを合成する高位合成システム CCAP における変数と関数の扱い," 電子情報通信学会技術研究報告, ICD, Vol.105(446), pp.19-24(2005)

[7] 房延慎二, 浅井達哉, 有村博紀, 宇野毅明, 中野眞一, "半構造データマイニングのための高速な無順序木パターン発見手法", 電子情報通信学会 DE 研究会第 15 回データ工学ワークショップ(DEWS2004), 6-A-03, 2004

[8] http://graphviz.org