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

2007－ＳＬＤＭ－132（29）
2007／11／22

専用演算器の使用を考慮した効率的な動作合成手法

貞方　毅†　松永　裕介†

† 九州大学大学院システム情報科学府　〒819–0395 福岡市西区元岡 744 番地
E-mail: †{sadakata,matsunaga}@c.csce.kyushu-u.ac.jp

あらまし　本論文では，動作合成において専用演算器を活用することで合成結果の性能改善を効率的に図る手法を提案する．専用演算器とは，特定の演算パターンに特化して遅延や面積が小さくなるよう設計された専用の演算器である．提案手法は，モジュール選択/スケジューリング/演算器アロケーションにおいて専用演算器を活用することで，演算器の総面積制約及びクロックサイクル周期制約下で効率的にサイクル数削減を実現するヒューリスティックな手法である．実験では，実用的な計算時間でサイクル数削減を最大 35%，平均で 14%達成した．
キーワード　動作合成，モジュール選択，演算器アロケーション，スケジューリング，専用演算器

An Efficient Behavioral Synthesis Method Considering Specialized Functional Units

Tsuyoshi SADAKATA† and Yusuke MATSUNAGA†

† Graduate School of Information Science and Electrical Engineering, Kyushu University　Motooka 744, Nishi-ku, Fukuoka City, 819–0395 Japan
E-mail: †{sadakata,matsunaga}@c.csce.kyushu-u.ac.jp

Abstract　This paper proposes a novel Behavioral Synthesis method that improves a performance of synthesized circuits utilizing specialized functional units efficiently. Specialized functional units are the units designed for specific operation patterns to achieve shorter delay and/or smaller area than cascaded basic functional units. The proposed method makes it possible to solve module selection, scheduling, and functional unit allocation problems utilizing specialized functional units in practical time with some heuristics, and the number of clock cycles can be reduced under total area and clock cycle time constraints with the proposed method. Experimental results show that the proposed method has achieved up to 35 % and on average 14 % reduction of the number of cycles with specialized functional units in practical time.
Key words　Behavioral Synthesis, Module Selection, Functional Unit Allocation, Scheduling, Specialized Functional Unit

1. はじめに

動作合成において，積和演算器のような専用演算器を導入することは，合成結果の改善に効果的である．しかしながら，資源制約下において専用演算器の使用と資源共有を同時に考慮するモジュール選択や演算アロケーションは複雑な問題である．専用演算器は，特定の演算パターンに対し，基本的な演算器（以降，基本演算器）の組み合わせで実現される構成よりも最大遅延時間や面積を削減することを目的に設計される専用の演算器である．例えば，整数の加算を基にする演算パターンに対して，桁上げ保存加算器を活用した専用演算器の構成方法 [1] などがある．よって，専用演算器を適切に使用することで，基本演算器を使用する場合に比べ，同一の資源制約下でも性能が良い合成結果を得られる可能性がある．しかしながら，専用演算器は特定の演算パターンに特化して設計されたものであるため，専用演算器上で異なる演算パターンを実行することは効率的でない場合がある．ゆえに，資源共有を考慮した合成においては，専用演算器を使用することによる性能の改善が得られなかったり，使用が適切でない場合は逆に性能を悪化させてしまう可能性もある．

図 1 は，カスケード接続された基本演算器と専用演算器に対して資源共有を行う例を示している．演算パターン 1 は，整数乗算 1 個と整数加算 1 個がデータ依存関係を持つ演算パターンを表している．演算パターン 2 は，整数乗算 1 個と整数加算 1

個がデータ依存関係を持たない演算パターンを表している．カスケード接続された基本演算器は，乗算器1個と加算器1個で構成され，2つの演算器間に2入力のセレクタが挿入されている．基本演算器のカスケード接続による構成では，セレクタを適切に制御することで柔軟に資源共有を行うことが可能であり，演算パターン1と演算パターン2を両方とも1クロックサイクル（以降，サイクル）で実行可能である．一方，専用演算器である積和演算器は演算パターン2を1サイクルで実行することができない．なぜならば，専用演算器では中間の演算結果を得ることができないからである．もし，両方の演算パターンが1回ずつ合成対象に出現し，各演算器構成が与えられる資源制約とサイクル周期を満たすと仮定すると，基本演算器のカスケード接続による構成を用いた方が少ないサイクル数で実行可能である．

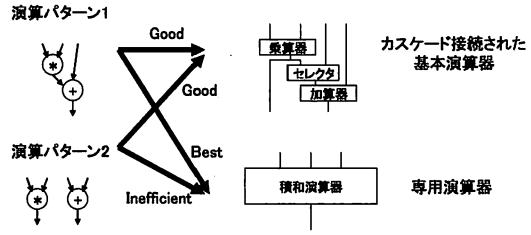

図1　カスケード接続された基本演算器と専用演算器に対する資源共有の実施例

動作合成において専用演算器の使用を考慮する関連研究はいくつか存在する．しかしながら，資源制約下で性能の最大化を目的にモジュール選択/演算器アロケーション/スケジューリングを実用的な時間で解く手法は存在しない．Landwehr と Mardwedel らは，資源制約とサイクル周期制約下で実行サイクル数の最小化を目的として，モジュール選択/スケジューリング/演算器アロケーション/バインディングを同時に解く整数線形計画法に基づく手法を提案した[2][3]．この手法では専用演算器の使用が考慮されている．しかし，整数線形計画法を用いるため，サイズが大きい問題に対しては現実的な時間で解を得ることが困難な場合がある．Corazao らは，サイクル数とサイクル周期の積の最小化を目的として，スケジューリングと演算器アロケーションを解くテンプレートマッピングに基づくヒューリスティックな手法を提案した[4]．この手法では，専用演算器をテンプレートとして取り扱うことが可能である．しかしながら，資源制約を考慮していないため資源の増加を制御することができない．Bringmann らは，性能の最大化を目的として，モジュール選択/演算器アロケーション/スケジューリングを解く山登り法的アプローチのヒューリスティックな手法を提案した[5]．この手法でも，資源制約は考慮されていない．

本論文では，演算器の総面積制約とサイクル周期制約下でサイクル数の最小化を目的とした，モジュール選択/スケジューリング/演算器アロケーションにおいて専用演算器の使用を考慮するヒューリスティックな手法を提案する．提案手法の主な特徴は，モジュール選択において新しいヒューリスティックなモジュールセット限定方法を導入することで，実用的な時間で専用演算器の使用によるサイクル削減効果を得ることが可能な点である．

本論文は以下のように構成されている．2章で本論文で用いる仮定，記号，定義について説明する．3章で提案手法の詳細について説明する．4章で実験結果について述べ，5章で本稿をまとめる．

2. 準　　備

合成対象として，ディスジョイントな Control/Data-Flow Graph[6]（以降，CDFG）を仮定する．ディスジョイントな CDFG は，1つの Control-Flow Graph（以降，CFG）と CFG の各ノードに対応する Data-Flow Graph（以降，DFG）から構成される．CFG は条件分岐を表す有向グラフである．DFG は演算間のデータ依存関係を表す非循環有向グラフである．CFG のノード数を $L \in N$（N は自然数の集合）で表し，各ノードに対応する DFG は $DFG_l(V_l, E_l), l = 1, 2, ..., L$（$V_l$ はノードの集合，E_l はエッジの集合）で表す．各 DFG_l において，ノード $v \in V_l$ に隣接する先行ノードの集合を $PARENTS(v)$ で表す．ノード $v \in V_l$ のすべての先行ノードの集合を $PRED(v)$ で表す．

演算器のライブラリは与えられるものとし，ライブラリには演算器の情報（面積，最大遅延時間，実行可能な演算パターン等）が含まれるものとする．演算器の種類の集合を F とする．演算器の面積は，関数 $f_{area}(fu) : F \mapsto R^+$（$R^+$ は正数の集合）で表される．演算器での演算実行に要するサイクル数は，関数 $f_{cycle}(fu) : F \mapsto N$ で表される．組み合わせ回路の演算器では，$f_{cycle}(fu) = 0$ とする．演算器の最大遅延時間は，関数 $f_{maxdelay}(fu) : F \mapsto R^+$ で，演算器の入力から初段のフリップフロップまでの最大遅延時間は関数 $f_{maxdelay\_in}(fu) : F \mapsto R^+$ で，最終段のフリップフロップから演算器の出力までの最大遅延時間は関数 $f_{maxdelay\_out}(fu) : F \mapsto R^+$ で表される．組み合わせ回路の演算器では，$f_{maxdelay}(fu) = f_{maxdelay\_in}(fu) = f_{maxdelay\_out}(fu)$ とする．

DFG において，1つの演算器上で同時に実行可能なデータ依存関係があるノードの集合を演算ブロックと定義する．演算ブロックは，専用演算器を取り扱うために導入される概念である．演算ブロックは，スケジューリングや演算器アロケーションにおいて，演算単体に代わる処理単位として用いられる．

[定義1]　演算ブロックは，$DFG_l(V_l, E_l)(l = 1, 2, ..., L)$ において，以下の条件をすべて満たすノードの集合 $b = V_l' \subseteq V_l$ である．

（1）　$|V_l'| \geq 1$

（2）　$|V_l'| > 1 \Rightarrow \forall v_1' \in V_l', \exists v_2' \in V_l', (v_1', v_2') \in E_l \vee (v_2', v_1') \in E_l$

（3）　$\forall v' \in V_l', \forall v'' \in \mathrm{PARENTS}(v') - (\mathrm{PARENTS}(v') \cap V_l'), |\mathrm{PRED}(v'') \cap V_l'| = 0$

（4）　$\exists v' \in V_l', \forall v'' \in V_l' (v'' \neq v')$, v'' からへの v' パスが存在する

2番目の条件は，V_l' による誘導部分グラフが連結であることを表している．3番目の条件は，コンベキシティを満たすための条件である．4番目の条件により，演算ブロックの出力に対応するノードが1つに制限されている．DFG_l 内のすべての演算ブロックの集合を B_l で表すものとする．演算ブロック間のデータ依存関係は，DFG のノード間のデータ依存関係から導かれる．

モジュール選択問題は，モジュールセットベクタ（module set vector: 以降，MSV）を決定する問題である．MSV は，$msv \in N^{|F|}$ と定義され，使用する演算器の種類と数を表す．$msv[fu]$ $(fu \in F)$ を演算器 fu の使用数を表す表記として用いる．MSV に含まれる演算器の総面積は，$\sum_{fu \in F} f_{area}(fu) \cdot msv[fu]$ で表される．

スケジューリング問題は，各 DFG の演算を実行するサイクルを決定する問題である．DFG_l でのスケジューリング結果は，関数 $schedule_l(b) : B_l \mapsto N$ で表される．

演算器アロケーション問題は，各 DFG の演算を実行する演算器の種類を決定する問題である．DFG_l 演算器アロケーションの結果は，関数 $allocation_l(b) : B_l \mapsto F$ で表される．

各 DFG には，関数 $weight(l) : N \mapsto N(l = 1, 2, ..., L)$ で表される重みが与えられるものとする．重みは，ループなどで繰り返し実行される DFG の実行回数を表すために用いられる．

本論文では，演算器の総面積制約 $\alpha \in R^+$，及び，サイクル周期制約 $\beta \in R^+$ の下において，各 DFG のスケジューリング後のサイクル数と重みの積の総和 $\sum_{l=1}^{L} weight(l) \cdot max_{b \in B_l}(schedule_l(b) + f_{cycle}(allocation_l(b))$ の最小化を目的とする問題に取り組む．演算器のみでの理想的な性能を解析するために，現在のところはメモリやインターコネクトのコストは無視されている．

3. 提案手法

提案手法では，全体の合成処理をモジュール選択とそれ以外の処理に分け，モジュール選択でいくつかの MSV を列挙し，列挙された MSV ごとに残りの処理を行うというアプローチをとっている．重要となるのは，最終的にサイクル数が少ない結果が得られるような MSV をモジュール選択で列挙することである．MSV は，多次元のベクトル空間内の整数値の座標点に対応する．よって，演算器の種類の総数や総面積制約の増大に対して，MSV の総数は指数的に増大する可能性があり，すべての MSV を探索し列挙することは現実的ではない．そこで提案手法では，始めに総面積制約に基づいて列挙の対象を制約の境界に対応する MSV に限定する．さらに，それらの MSV を使用する演算器の種類がすべて同じものごとにグループ化する．そして，演算器の種類の組み合わせごとに，それから導かれる MSV によって達成される最小サイクル数をヒューリスティックな方法で見積り，見積り値が小さい組み合わせに対応する MSV に限定して列挙を行うことで，実用的な時間での列挙を実現している．

提案手法は3つの処理部からなる: 1) MSV の列挙，2) 演算ブロックの選択，3) 資源制約型スケジューリング/演算器アロケーション．各々の処理に対し，ヒューリスティックなアルゴリズムを導入している．処理全体の流れは次のようになる．始めに，ヒューリティックな指標に基づき，いくつかの MSV を列挙する．そして，各々の MSV に対し演算ブロックの選択とスケジューリング/演算器アロケーションが反復して行われる．最終的に，最も良い結果を選択する．

3.1 Module Set Vector の列挙

列挙すべき MSV を **feasible** MSV（以降，FMSV）とする．FMSV とは，総面積制約とサイクル周期制約を満足する MSV である．提案手法では，列挙する FMSV を資源制約に基づいて **maximal** FMSV に限定する．maximal FMSV とは，どれか1個でも演算器を追加すると総面積制約を違反してしまうような FMSV である．スケジューリングと演算器アロケーションの厳密解が得られるとすれば，maximal でない FMSV により得られる解のサイクル数は，その FMSV を包含する maximal FMSV による解のサイクル数以下であることは明らかである．よって，提案手法では maximal FMSV のみを列挙対象として取り扱う．

maximal FMSV の総数は，$|F|$ と演算器の総面積制約に対して指数的である可能性がある．よって，常にすべての maximal FMSV を列挙することは現実的ではない．提案手法では，列挙にかかる計算時間の増大を避けるため，列挙する maximal FMSV を **unit** FMSV を基に限定する．unit FMSV とは，すべての要素が1以下，つまり，$\forall fu \in F, msv[fu] \leq 1$ となる FMSV である．各 maximal FMSV $msv_{maximal}$ に対し，次の条件を満たす unit FMSV msv_{unit} を考える．

$$\forall fu \in F,\ msv_{unit}[fu] = \begin{cases} 0 & (\mathrm{msv_{maximal}[fu]} = 0) \\ 1 & (\mathrm{msv_{maximal}[fu]} \geqq 1) \end{cases}$$

maximal FMSV に対し，上記の条件を満たす unit FMSV がユニークに存在することは明らかである．そこで提案手法では，すべての unit FMSV を列挙し，unit FMSV をヒューリスティックな指標に基づいて評価する．そして，良い評価値が得られた unit FMSV から順番に，maximal FMSV を一定数に達するまで列挙するというアプローチをとる．

unit FMSV に対する評価指標には，unit FMSV から導かれる maximal FMSV によって達成可能な最小サイクル数の見積り値を用いる．見積り値は，unit FMSV を資源制約とみなして合成した結果と，unit FMSV に含まれる演算器の種類のみ資源制約を無視して合成した結果から，線形補間することで求める．図2は線形補間の方法を表している．図2の横軸は演算器の総面積を表し，縦軸は合成結果のサイクル数を表す．始めに，unit FMSV を資源制約とみなして合成し，unit FMSV の総面積と結果のサイクル数に対応する点を得る．次に，unit FMSV に含まれる演算器の種類のみいくつでも使用できると仮定して資源制約を無視して合成し，実際に使用された演算器の総面積とサイクル数に対応する点を得る．これら2つの点の間を線形的に補間し，その直線上の点を与えられた面積制約に対する最小サイクル数の見積り値とする．

すべての unit FMSV に対して見積りを行った後，最も小さ

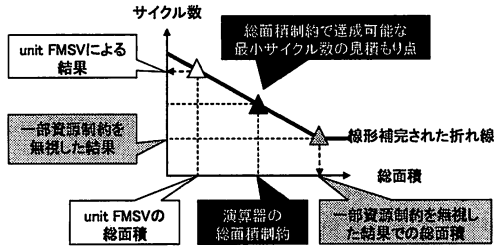

図 2 unit FMSV から導かれる maximal FMSV で達成可能な最小サイクル数の見積り方法

い見積り値を得た unit FMSV から maximal FMSV を列挙する．列挙する maximal FMSV の数は，与えられる定数 $\epsilon \in N$ を上限とする．

unit FMSV の総数は，依然として $|F|$ に対して指数的である可能性がある．もし，面積制約を違反する unit FMSV が存在しなければ，すべての unit FMSV が列挙される．そのような場合，実用的には $|F|$ の上限は 16 ぐらいである．列挙される maximal FMSV の総数は，ϵ により制限される．

3.2 演算ブロックの選択

演算ブロックの選択は，各 DFG に対して山登り法的に実行される．始めに，ノード数が 1 個だけの演算ブロックの集合をカレントブロック集合として用意する．次に，ノード数が 2 以上の演算ブロックを 1 個選びカレントブロック集合に追加し，追加した演算ブロックとノードが重複するものを除いた新たなブロック集合に対し，スケジューリング/演算器アロケーションを行い結果を保存する．これらを，すべてのノード数が 2 以上の演算ブロックに対して行い，最も良い結果を得たブロック集合を新たなカレントブロック集合とする．以上の処理を，結果の改善が行われるまで繰り返し行う．スケジューリング/演算器アロケーションの最大の実行回数は，ノード数が 2 以上の演算ブロックの集合を B_{multi} とすると，$1+|B_{multi}|(|B_{multi}|-1)/2$ である．

3.3 スケジューリングと演算器アロケーション

スケジューリングと演算器アロケーションのアルゴリズムは，静的なリストスケジューリングアルゴリズム [6] を基にしている．各サイクルにおけるスケジューリング可能な演算ブロックのリストでは，資源制約とサイクルの境界を無視した場合に演算ブロックから DFG の終端のブロックまでに必要となる最短処理時間の小さいブロックに高い優先度を与える．この値は，あらかじめ演算ブロックをトポロジカルソートし，As Soon As Possible スケジューリングアルゴリズムを用いて $O(|B'|)$ で計算可能である（B' は演算ブロックの選択結果）．リストのソートは $O(|B'| \cdot log_2(|B'|))$ で可能である．リストをソートした後，先頭の演算ブロックから，利用可能な演算器の中で最も高速な演算器を割り当てていく．各サイクルでの，スケジューリングと演算器のアロケーションの計算時間は $O(C_l \cdot |B'| \cdot |F|)$ となる（C_l は DFG_l のサイクル数の上限値）．よって，DFG 全体に対する計算時間は $O(|B'| \cdot (log_2(|B'|) + C_l \cdot |F|))$ となる．最終的に，演算ブロックの選択まで含めた計算時間は，$O(|B_{multi}|^2 \cdot |B'| \cdot (log_2(|B'|) + C_l \cdot |F|))$ となる．

4. 実　　験

提案手法を C++言語で実装し，ベンチマークに対して評価実験を行った．ベンチマークには，HLSynth'92 [7] の微分器（diffeq），MediaBench [8] の JPEG に含まれる DCT 関数（jpeg_fdct_islow，以降 fdct）と MPEG2 エンコーダの動き予測で用いられる関数（bdist2）を用いた．各ベンチマークを手作業で CDFG へ変換した．含まれる演算ノードの数は，diffeq で 11 個，bdits2 で 43 個，fdct で 138 個である．bdist2 には 8 ビット演算と 32 ビット演算が含まれており，diffeq と fdct には 32 ビット演算のみが含まれている．データ型はすべて符号無し整数である．ノード数 2 以上の演算ブロックとして，桁上げ保存加算器を基にした専用演算器構成手法 [1] に条件が合致するものをすべて列挙した．列挙されたすべての演算ブロックに対し，VDEC で提供されている日立社の 0.18μmCMOS プロセス向けに京都大学で設計されたセルライブラリと Synopsys 社の Module Compiler を用いて，専用演算器をタイミング制約 3 ns と 6 ns の 2 通りで合成した．表 1 は，使用した演算器の機能と合成結果を表している．表の上から 9 行は基本演算器の結果を表し，残りは専用演算器の結果を表す．最も左の列が演算器の機能を表している．“8 bit” と書かれた演算器はデータ幅が 8 ビットのものを表し，“8 bit” と書かれていない演算器はデータ幅が 32 ビットのものを表す．“DC” は Module Compiler に与えたタイミング制約を表している．“D” の列は合成結果から得た最大遅延時間を表す．“A” の列は合成結果から得られたセルの総面積を表す．“使用” の列は，各ベンチマークで演算器が使用されているかどうかを表している．“*” は演算器が使用されていることを表す．

モジュール選択を除く演算ブロックの選択とスケジューリング/演算器アロケーションのアルゴリズムを評価するために，最も小さいベンチマークである diffeq を合成対象として，Landwehr らの手法 [2] を本論文が対象とする問題に拡張した整数線形計画法に基づく手法での合成と，maximal FMSV を全列挙して提案手法を適用する合成を行い結果を比較した．整数線形計画法に基づく手法による解は，商用のソルバである ILOG 社の CPLEX を用いて得た．実験は，CPU が AMD Opteron 275 でメモリが 8GB の PC 上行った．サイクル周期の制約は 6 ns とし，演算器の総面積制約は 110,000〜190,000 μm^2 の間で 10,000 μm^2 刻みで変化させた．表 2 は，各制約下での合成結果のサイクル数表している．最も左の列は，面積制約を表している．“ILP” は整数線形計画法に基づく手法による結果を表し，“ALL” は提案手法で maximal FMSV を全列挙した場合の結果を表す．“w/o” の列は，専用演算器を使用しない場合の結果を表し，“w” の列は，専用演算器を使用した場合の結果を表す．結果から，専用演算器の使用により小さいベンチマークでもサイクル数が 1〜3 削減されていることが確認された．また，整数線形計画法に基づく手法による結果と同等の結果が提案手法により得られている．計算時間に関しては，整数線形計

画法に基づく手法では最悪 2,000,000 秒以上かかった．一方，提案手法はすべての場合において 2 秒以内であった．これらの結果から，専用演算器の使用は資源制約下でサイクル数を削減するのに効果的であり，提案手法の演算ブロックの選択とスケジューリング/演算器アロケーションのアルゴリズムは実用的であると考えられる．

提案手法のモジュール選択アルゴリズムを評価するために，dist2 と fdct を合成対象として，提案するヒューリスティックなアルゴリズムを用いた合成と maximal FMSV を全列挙した合成を行い結果を比較した．サイクル周期制約はどちらのベンチマークに対しても 6 ns と 9 ns の 2 通りで行った．提案手法での maximal FMSV の列挙数上限値 ϵ は 1,000 とした．表 3 と表 6 は各ベンチマークで列挙された unit FMSV と maximal FMSV の総数を表している．表中の "UNIT" は unit FMSV の総数を表し，"MAXIMAL" は maximal FMSV の総数を表す．表 4 と表 7 は各ベンチマークの合成結果のサイクル数を表している．表中の "CCT = 6 ns" はサイクル周期制約が 6 ns の場合の結果を表し，"CCT = 9 ns" はサイクル周期制約が 9 ns の場合の結果を表す．"ALL" は maximal FMSV を全列挙した場合の結果を表し，"OUR" はヒューリスティックなモジュール選択を適用した結果を表す．表 5 と表 8 はサイクル周期制約が 6 ns での各ベンチマークでの計算時間を表している．実験は，CPU が Intel Xeon 5140 でメモリが 8GB の PC 上で行った．サイクル周期制約が 9 ns での計算時間の結果はスペースの都合により省略されている．しかし，結果はサイクル周期制約 6 ns の場合とほぼ同じであった．始めに，専用演算器の有無による結果 "w/o" と "w" を比較すると，ほぼすべての条件においてサイクル数の削減が確認された．特に，bdist2 の面積制約が 120,000 μm^2 かつ "CCT = 6" において，専用演算器によりサイクル数が 35%削減されている．平均では，bdit2 で 15%，fdct で 13%，両方あわせて 14%削減された．さらに，bdist2 の面積制約が 110,000 μm^2 の場合と，fdct の面積制約が 120,000 μm^2 の場合では，専用演算器を用いた場合のみ実現可能解が得られている．次に，モジュール選択で全列挙する場合と提案するヒューリスティックなアルゴリズムを用いた場合とでサイクル数を比較すると，提案手法による結果は全列挙による結果に対してほぼ同等の結果が得られている．一方，計算時間に関しては，bdist2 では面積制約が 130,000 μm^2 以下で fdct では面積制約が 160,000 μm^2 以下で，提案手法が全列挙の場合より長くなっている．この理由は，提案手法では 1 つの unit FMSV に対し評価指標の値を求めるために 2 回合成を行うため，unit FMSV の総数が maximal FMSV の総数の半分より大きい場合により多くの計算時間を必要とするためである．しかしながら，面積制約が大きくなると提案手法の方が計算時間が短くなる．最も差が見られる場合では，bdist2 で 1/50，fdct で 1/10 の計算時間で提案手法は解が得られている．これらの結果から，提案手法により maximal FMSV を全列挙する場合と同等のサイクル数を実現する解を実用的な時間で得られると考えられる．

5. おわりに

本論文では，演算器の総面積制約とサイクル周期制約下において，専用演算器の活用によるサイクル数の削減を実用的な時間で達成する手法を提案した．提案手法は，新しい評価指標をモジュール選択に導入することで列挙する maximal FMSV を限定しつつ，良い解を実用的な時間で得ることを可能とする．実験では，専用演算器の活用により最大で 35%，平均で 14%のサイクル数の削減を実用的な時間内で実現することを確認した．今後の課題は，計算時間をより削減するモジュール選択アルゴリズムの検討や，メモリやインターコネクトのコストを含めた評価である．

6. 謝　辞

本研究の一部は日本学術振興会科学研究費補助金特別研究員奨励費（17-6203）の助成を受けたものである．本研究の一部は科学研究費補助金（基盤研究 (A)）（課題番号:19200004）によるものである．本研究の一部は，東京大学大規模集積システム設計教育研究センターを通し，Synopsys 社のツールを用いて行われたものであり，日立製作所株式会社の協力で行われたものである．

文　献

[1] J. Um, and T. Kim, "An optimal allocation of carry-save-adders in arithmetic circuits," IEEE Trans. on Computers, vol.50, no.3, pp.215–233, Mar. 2001.

[2] B. Landwehr, and R.D. P. Marwedel, "Oscar: Optimum simultaneous scheduling, allocation and resource binding based on integer programming," Proc. of the conference on European design automation, pp.90–95, 1994.

[3] P. Marwedel, B. Landwehr, and R. Dömer, "Built-in chaining: Introducing complex components into architectural synthesis," Proc. of the Asia South Pacific Design Automation Conference, pp.599–605, 1997.

[4] M.R. Corazao, M.A. Khalaf, L.M. Guerra, M. Potkonjak, and J.M. Rabaey, "Performance optimization using template mapping for datapath-intensive high-level synthesis," IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems, vol.15, no.8, pp.877-888, Aug. 1996.

[5] O. Bringmann, and W. Rosenstiel, "Cross-level hierarchical high-level synthesis," Proc. of the conference on Design, Automation and Test in Europe, pp.451–456, 1998.

[6] D.D. Gajski, High-Level Synthesis: Introduction to Chip and System Design, Kluwer Academic Pub, 1992.

[7] N. Dutt, "Current status of hlsw benchmarks and guidelines for benchmark submission," HLSynth'92 Benchmark, 1992.

[8] C. Lee, M. Potkonjak, and W.H. Mangione-Smith, "Mediabench: a tool for evaluating and synthesizing multimedia and communications systems," Proc. of the ACM/IEEE international symposium on Microarchitecture, pp.330–335, 1997.

表 1　使用した演算器の情報

機能	DC = 3 ns		DC = 6 ns		使用		
	D (ns)	A (μm^2)	D (ns)	A (μm^2)	diffeq	bdist2	fdct
$a+b$ or $a-b$	2.48	24284	4.60	19384	*	*	*
$a+b$	2.44	10460	4.83	7994	*	*	*
$a+b$ (8 bit)	1.79	1567	1.79	1567		*	
$a-b$	2.22	11927	4.43	9155	*	*	*
$a*b$	3.82	93036	5.58	77821	*	*	*
$a<b$	0.95	4662	0.95	4662	*	*	
$a \geqq b$	1.00	4685	1.00	4685			*
$a>>b$	2.12	10829	2.12	10829		*	*
$a<<b$	1.77	10045	4.43	9155			*
$a-b*c$	4.18	97881	5.69	81116	*		
$a-b-c$	2.46	16397	4.65	14070	*		
$a*b+c$	4.12	93327	5.64	79764	*	*	*
$a-b*c-d$	4.62	98704	5.58	88758	*		
$(a-b)-c*d$	4.37	101268	5.48	86592	*		
$(a-b*c)-d*e$	4.99	169789	6.09	159437	*		
$a+b+c$	2.40	14976	4.87	122296		*	*
$a+b+c$ (8 bit)	1.86	2650	1.86	2650		*	
$a+b+c+d$ (8 bit)	2.11	3802	2.23	3694		*	
$a+b+c+d+e$ (8 bit)	2.21	4631	2.32	4524		*	
$a*b+c+d$	4.48	95578	5.49	84541			*
$a+b+c+d$	2.32	24660	4.84	16827			*
$a*b+c+d+e$	4.82	100877	5.86	90770			*
$a-b+c$	2.16	16818	4.74	13363			*

表 2　diffeq の合成結果（サイクル数）

面積制約	ILP		ALL	
(μm^2)	w/o	w	w/o	w
100,000	9	8	9	8
110,000	9	8	9	8
120,000	9	8	9	8
130,000	9	8	9	8
140,000	9	8	9	8
150,000	9	8	9	8
160,000	9	8	9	8
170,000	9	6	9	6
180,000	7	6	7	6
190,000	7	6	7	6

表 3　bdist2 の合成結果（FMSV の列挙数）

面積制約	UNIT		MAXIMAL	
(μm^2)	w/o	w	w/o	w
110,000	0	11	0	11
120,000	6	224	6	327
130,000	20	1415	31	3773
140,000	41	4699	88	25468
150,000	65	11001	195	122562
160,000	86	20116	360	464922
170,000	100	30802	595	1473367
180,000	109	41124	897	4052582

表 4　bdist2 の合成結果（サイクル数）

面積制約	CCT = 6 ns				CCT = 9 ns			
	ALL		OUR		ALL		OUR	
(μm^2)	w/o	w	w/o	w	w/o	w	w/o	w
110,000	-	29	-	29	-	29	-	29
120,000	28	18	28	18	27	16	27	16
130,000	19	16	19	16	16	13	16	13
140,000	17	14	17	14	13	12	13	12
150,000	16	13	16	14	12	11	12	11
160,000	15	13	15	13	11	10	11	11
170,000	15	12	15	13	11	10	11	11
180,000	14	12	14	12	11	10	11	10

表 5　bist2 の合成結果（計算時間）

面積制約	ALL		OUR	
(μm^2)	w/o	w	w/o	w
110,000	1	1	1	1
120,000	1	1	1	1
130,000	1	7	1	7
140,000	1	48	1	18
150,000	1	219	1	40
160,000	1	848	1	72
170,000	1	2736	1	111
180,000	1	7588	1	149

* CCT = 6 ns

表 6　fdct の合成結果（FMSV の列挙数）

面積制約	UNIT		MAXIMAL	
(μm^2)	w/o	w	w/o	w
120,000	0	5	0	5
130,000	6	47	7	55
140,000	23	206	39	299
150,000	44	628	124	1201
160,000	68	1552	310	3974
170,000	88	3174	659	11292
180,000	102	5712	1251	28975
190,000	110	9080	2234	68458
200,000	113	13193	3763	151268
210,000	114	17603	6051	315858
220,000	119	22032	9356	629034

表 7　fdct の合成結果（サイクル数）

面積制約	CCT = 6 ns				CCT = 9 ns			
	ALL		OUR		ALL		OUR	
(μm^2)	w/o	w	w/o	w	w/o	w	w/o	w
120,000	-	68	-	68	-	66	-	66
130,000	51	46	51	46	50	44	50	44
140,000	44	37	44	37	40	36	40	36
150,000	40	36	40	36	38	34	38	34
160,000	38	36	38	36	38	32	38	32
170,000	38	34	38	34	36	30	36	30
180,000	38	34	38	34	36	30	36	30
190,000	38	34	38	34	36	30	36	30
200,000	38	34	38	34	36	30	36	30
210,000	38	34	38	34	36	30	36	30
220,000	38	34	38	34	36	30	36	30

表 8　fdct の合成結果（計算時間）

面積制約	ALL		OUR	
(μm^2)	w/o	w	w/o	w
120,000	1	1	1	1
130,000	1	1	1	1
140,000	1	3	1	7
150,000	1	13	1	23
160,000	1	45	1	45
170,000	1	133	1	85
180,000	1	357	1	151
190,000	2	869	1	253
200,000	3	1970	1	373
210,000	5	4121	1	677
220,000	8	8218	1	857

* CCT = 6 ns