Packed SIMD型命令を持つプロセッサを対象としたハードウェア/ソフトウェア協調合成システムのためのハードウェアユニット生成手法

宮岡 祐一郎[†1] 戸川 望[,†2,†3]
柳澤 政生[†4] 大附辰夫[†4]

本稿では，Packed SIMD型命令を持つプロセッサを対象としたハードウェア/ソフトウェア協調合成システムのためのハードウェアユニット生成手法を提案する．ハードウェアユニットを複数の部分機能を実現するハードウェアの組合せで構成することにより，与えられた命令集合を実行できるハードウェアユニットの構成を，高速に複数列挙し，ハードウェアユニットの面積と遅延を見積もることができ，Packed SIMD型命令を持つプロセッサコアを協調合成システムにより得ることができるようになる．計算機実験により本手法の有効性を評価した．

A Hardware Unit Generation Algorithm for a Hardware/Software Cosynthesis System of Digital Signal Processor Cores with Packed SIMD Type Instructions

Yuichiro Miyaoka,[†1] Nozomu Togawa,[†2,†3] Masao Yanagisawa[†4] and Tatsuo Ohtsuki[†4]

This paper proposes a hardware unit generation algorithm for a hardware/software cosynthesis system of digital signal processors with packed SIMD type instructions. Given a set of instructions, the proposed algorithm extracts a set of subfunctions to be required by the hardware unit and generates more than one architecture candidates for hardware units. The algorithm also outputs the estimated area and delay of each of the generated hardware units. The execution time of the proposed algorithm is very short and thus it can be easily incorporated into the processor core synthesis system. Experimental results demonstrate effectiveness and efficiency of the alogorithm.

1. まえがき

bビットの演算ユニットを改良すれば，1つのbビット演算ユニットを用いてn個のb/nビット演算を実行させることができる．この演算を，Packed SIMD型演算と呼ぶ．Packed SIMD型演算を用いると，画像処理アプリケーションを画素並列に実行することができるので，Packed SIMD型演算を用いたプロセッサは，画像処理アプリケーションを高速に実行することが可能である[2)~4),10),12),15)]．Packed SIMD型演算はn個のデータを並列に扱うので，個々のデータの演算結果ごとに特定の処理をすることはできない．そこで，Packed SIMD型演算ののちに演算結果のあふれを飽和させる処理や，$n/2$個の演算結果のみをレジスタに書き込むことによって演算の精度を保つ処理を一連の流れとして扱う命令を用いるのが有効である．この命令をPacked SIMD型命令と呼ぶ．Packed SIMD型命令は同種の演算（加算・乗算など）であっても，nの値や飽和処理の有無などによって多数の命令が存在する．しかし特定のアプリケーションを実行する場合，ごく一部のPacked SIMD型命令のみが必要となるので，Packed SIMD型命令を持つプロセッサの設

†1 早稲田大学大学院理工学研究科電子・情報通信学専攻
Department of Electroics, Information and Communication Engineering, Waseda University
†2 北九州市立大学国際環境工学部情報メディア工学科
Department of Information and Media Sciences, The University of Kitakyushu
†3 早稲田大学理工学総合研究センター
Advanced Research Institute for Science and Engineering, Waseda University
†4 早稲田大学理工学部電子・情報通信学科
Department of Electroics, Information and Communication Engineering, Waseda University

計にはアプリケーションに応じて命令セットを変更するハードウェア/ソフトウェア協調設計が有効であると考えられる．

我々はディジタル信号処理向けプロセッサコアのハードウェア/ソフトウェア協調合成システムを提案している[14]．システムの核となるハードウェア/ソフトウェア分割では，プロセッサが持つ命令を最適化し，合成されるプロセッサコアの面積とアプリケーションの実行時間を見積もって，要求にあった面積と実行時間を持つプロセッサコアの構成を得る．プロセッサの持つ命令に応じて，プロセッサコアに付加される演算ユニット，アドレッシングユニットやハードウェアループユニットが決定される．演算ユニットに代表される，プロセッサの命令を実行するための機構をハードウェアユニットと呼ぶ．プロセッサコアの面積とアプリケーションの実行時間の見積りには，ハードウェアユニットの面積と遅延の見積りが不可欠である．Packed SIMD 型演算ユニットのように，実行できる命令が複数あるハードウェアユニットでは，とりうるすべての構成に対して面積と遅延の見積り値を得る必要がある．要求にあった面積と実行時間を持つプロセッサコアの構成を得るためには，プロセッサが持つ命令を再構成し，多数のプロセッサ構成に対してプロセッサコアの面積およびアプリケーションの実行時間を見積もる必要がある．そのため，ハードウェアユニットの面積と遅延の見積りは高速に実行されなければならない．

これまでの我々のシステム[14]や，その他のプロセッサのハードウェア/ソフトウェア協調設計およびプロセッサ記述の合成に関する研究例[1),13),16)]では，必要とするハードウェアユニットを人手で設計して用意している．しかし，Packed SIMD 型演算ユニットはとりうる構成が多数あるため，そのすべてを用意するのが困難である．文献 5), 6) などでは，ハードウェアユニットの管理システムとして FHM-DBMS[9)] を用いているが，FHM-DBMS はハードウェアユニットのアーキテクチャや演算アルゴリズムを指定して，面積や遅延の見積りを出力するため，面積や遅延の値を考慮しながら高速に複数の候補を列挙するのには向いていない．

以上の背景から，Packed SIMD 型命令を持つプロセッサのハードウェア/ソフトウェア協調合成システムのためのハードウェアユニット生成系を提案する．1 つのハードウェアユニットで実行される命令の集合と，生成されるハードウェアユニットの面積と遅延の制約を入力とし，ハードウェアユニットの構成を複数列挙し面積と遅延の見積り値を出力する．複数の解候補を列挙することで，同じ命令集合に対して，何度もハードウェアユニットの面積や遅延の見積り値を呼び出すことなくハードウェア構成を選択することが可能である．ハードウェアユニット生成系は，入力された命令から必要な機能を抽出する部分機能抽出と，部分機能抽出で決定されたアーキテクチャテンプレートに部分機能ユニットを割り当てるアーキテクチャ構成により実現される．提案するアーキテクチャ構成アルゴリズムにより，すべてのハードウェアユニットで実現される命令集合の組合せに対して，その構成を高速に列挙することを可能とする．

本稿は以下のように構成される．2 章ではハードウェア/ソフトウェア協調合成システムが対象とするプロセッサのモデルと Packed SIMD 型命令セットを定義する．3 章ではハードウェアユニット生成系を提案する．4 章では高速にハードウェアユニットの面積・遅延を見積もるためのアーキテクチャ構成アルゴリズムを提案する．5 章では提案したハードウェアユニット生成手法を計算機上に実装し，Packed SIMD 型演算ユニットなどのハードウェアユニットに適用した結果を報告する．

2. プロセッサアーキテクチャモデル

ハードウェア/ソフトウェア協調合成システムが対象とするプロセッサのアーキテクチャは，文献 14) で対象とするアーキテクチャに Packed SIMD 型命令を追加したものである．命令メモリ，データメモリ用のバス，レジスタファイル，バレルシフタおよび ALU を含むプロセッサカーネルに，ハードウェアユニットを付加したものをプロセッサコアと呼ぶ．プロセッサカーネルは，VLIW 型のアーキテクチャをとり，パイプライン段数として，3 段（`IF`，`ID`，`EXE`）あるいは 5 段（`IF`，`ID`，`EXE`，`MEM`，`WB`）のどちらかを選択することができる．プロセッサは**表 1** に示す命令セットを持ち，加えて**表 2** の命令を Packed SIMD 型命令[11)]とする．Packed SIMD 型命令は，b ビットの演算ユニットを用いて，同時に n 個の b/n ビット演算をする命令である．ここで n を梱包数と呼び，$k = b/n$ とする．表 1 および表 2 に示す命令を複数個並列に発行する命令を複合命令と呼ぶ．アプリケーションに応じて必要な命令の組合せのみを複合命令とする．

2.1 Packed SIMD 型命令

Packed SIMD 型命令[11)]は以下に定義される命令である．

算術演算命令 算術演算命令は次の 2 種類に分けることができる．

表 1 基本命令
Table 1 Basic instructions.

Arithmetic and logic operation	ADD, SUB, SRA, SRL, SLL, AND, OR, XOR, MUL, DIV, SLT, SEQ, SNE, COM2, MAC, INC, DEC, ADDI, SUBI, SRAI, SRLI, SLLI, ANDI, ORI, XORI, MULI, DIVI
Load and store	LDX, LDY, STX, STY, LDRX, LDRY, STRX, STRY, LDXI, LDYI, STXI, STYI, LDIX, LDIY, STIX, STIY, MV, IMM
Jump	BEQ, BNE, BZ, BNZ, JP, LOOP, RPT, CALL, RET, NOP, HLT
Parallel load and store	LDPX, STPX

表 2 Packed SIMD 型命令
Table 2 Packed SIMD type instructions.

Arithmetic operation	ADD, SUB, MUL, MAC
Shift operation	SRA, SLA, SLL
Bit extend/extract operation	EXTD, EXTR
Others	EXCH

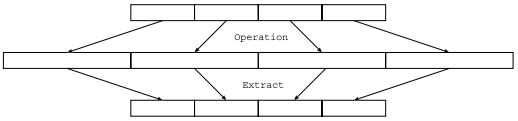

図 1 n 個の k ビット演算
Fig. 1 n operations for k-bits data.

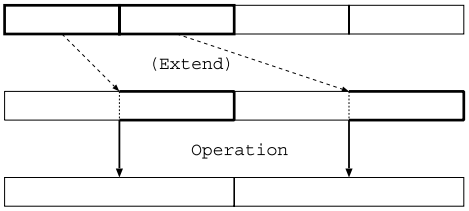

図 2 $n/2$ 個のビット拡張 k ビット演算
Fig. 2 $n/2$ extended operations for k-bits data.

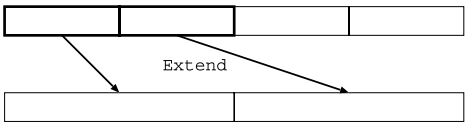

図 3 ビット拡張命令
Fig. 3 Bits extended operation.

n 個の k ビット演算命令（図 1） n 個の k ビットデータが 2 組与えられて，n 個の k ビット演算が同時に実行される．演算に対して，(i) 符号の有無，(ii) 演算結果のシフトの有無およびシフト量・シフト方向，および (iii) 飽和処理の有無を選択できる．この命令を {#1}_{#2}_{#3}{#4}{#5} という形式で表す．#1 は演算の種類を表し，#2 は梱包数を表す．#3 は符号の有無を s および u で表し，#4 は演算結果のシフトをするとき，その方向を r と l，シフト量を数字で表す．#5 は飽和処理の有無を s および w で表す．たとえば，梱包数が 4，符号あり，2 ビット右シフト，飽和ありの乗算命令を MUL_4_sr2s と表す．

$n/2$ 個のビット拡張 k ビット演算（図 2） n 個の k ビットデータが 2 組与えられたとき，上位あるいは下位を構成する $n/2$ 個の k ビットデータに対して演算が同時に実行され，結果を $2\times k$ ビットとして得る．演算に対して，(i) 上位と下位のどちらに演算を実行するか，(ii) 符号の有無，(iii) 演算結果のシフトの有無およびシフト量・シフト方向を選択できる．この命令を {#1}_{#2}{#3}_{#4}{#5} という形式で表す．#1 は演算の種類，#2 は梱包数を表す．#3 は上位か下位かを h と l で表し，#4 は符号の有無，#5 はシフト方向と量を表す．たとえば，梱包数 4 のデータの上位 2 つのデータに対して，符号ありで演算結果を 7 ビット左にシフトする加算命令を ADD_4h_sl7 と表す．

シフト演算命令 シフト演算も算術演算と同様に，n 個の k ビット演算命令と $n/2$ 個のビット拡張 k ビット演算に分けることができる．2 個のデータに対して算術右シフトをする命令を SRA_2，4 個のデータの下位 2 個のデータを左シフトする命令を SLL_4l と表す．

ビット拡張，縮小命令 ビット拡張命令 EXTD は，n 個のデータの上位，あるいは下位を構成する $n/2$ 個のデータに対して，上位ビットを拡張して $2\times k$ ビットのデータを構成するものである（**図 3**）．ビット拡張は，(i) 符号の有無，(ii) 拡張結果のシフトの有無を指定できる．

ビット縮小命令 EXTR は，n 個の $2\times k$ ビットデータが与えられて，それぞれのデータの下位 k ビットを抽出することで n 個の k ビットデータを構成する命令である（**図 4**）．ビット縮小命令は，(i) 符号の有無，(ii) 与えられた $2\times k$ ビットデータのシフトの有無および，(iii) 飽和処理の有無を指定できる．

交換命令 n 個の k ビットデータが 2 組与えられたとき，k ビットデータの位置を交換し新たな n 個の k ビットデータを 2 組構成する命令を EXCH 命令

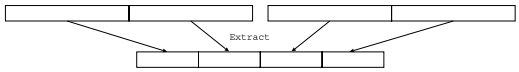

図 4 ビット縮小命令
Fig. 4 Bits extracted operation.

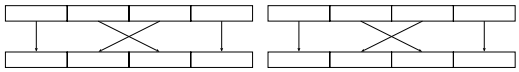

図 5 交換命令
Fig. 5 Exchange operation.

と呼ぶ．EXCH 命令では，データが交換される場所は梱包数ごとにあらかじめ決定されている．**図 5** に梱包数 4 のときの動作を示す（EXCH_4）．

2.2 ハードウェアユニット

プロセッサカーネルに付加されるハードウェアユニットは以下の 3 つである．

演算ユニット 演算ユニットは，ALU，バレルシフタ，乗算器，乗加算器と，Packed SIMD 型命令を実現する Packed SIMD 型加算ユニット，乗算ユニット，乗加算ユニット，シフタ，ビット拡張ユニット，ビット縮小ユニットおよび交換演算ユニットである．演算ユニットは，EXE ステージに付加される．付加される演算ユニットの種類が増えると，演算ユニットの面積に加えて EXE ステージでの制御部の面積が増加する☆．

アドレッシングユニット アドレッシングユニットは，内部にメモリアドレスを保持するためのレジスタを持ち，レジスタの内容にあるアドレスにメモリ参照し，同時にレジスタの値を更新して次参照アドレスを計算することで，メモリ内の連続した箇所への参照や，ある一定の規則での参照を高速に実行することを可能とするハードウェアユニットである．アドレッシングユニットとして，(i) no operation，(ii) post increment，(iii) post decrement，(iv) index add，(v) modulo add，(vi) bit reverse のアドレス演算を実現できるものを考える[7),8)]．アドレッシングユニットは，パイプライン段数が 3 段の場合にのみ付加される．アドレッシングユニットは実行できる演算の種類（(i)-(vi)），アドレスレジスタの数，インデックスレジスタの数を可変とし，アプリケーションに応じて構成を決定する．

ハードウェアループユニット ハードウェアループユニットは，定数回実行されるループを，パイプラインの流れを中断することなく実行させるためのハードウェアユニットであり，LOOP 命令で設定されたループ回数とループの開始および終了アドレスを保持して，次実行命令アドレスを決定し出力する[7),8)]．ハードウェアループユニットはパイプライン段数が 3 段の場合にのみ付加される．

3. Packed SIMD 型命令を持つプロセッサのハードウェア/ソフトウェア協調合成システムのためのハードウェアユニット生成

本章では，まず Packed SIMD 型命令を持つプロセッサのハードウェア/ソフトウェア協調合成システムの中で，特にハードウェア/ソフトウェア分割とハードウェアユニットの面積・遅延の見積りとの関係について検討し，その後ハードウェアユニット生成系を提案する．

3.1 ハードウェア/ソフトウェア協調合成システムとハードウェアユニット

ディジタル信号処理向けプロセッサのハードウェア/ソフトウェア協調合成システム[14)]は，C 言語で書かれたアプリケーションプログラム，アプリケーションデータおよびアプリケーションの実行時間制約を入力として，プロセッサコアのハードウェア記述，オブジェクトコードおよび生成されたプロセッサコア専用の C コンパイラ，アセンブラ，命令セットシミュレータを出力する．システムは主にコンパイラ，ハードウェア/ソフトウェア分割，およびハードウェア/ソフトウェア生成から構成される．

ハードウェア/ソフトウェア分割では，コンパイラの出力したアセンブリコードから決まるプロセッサアーキテクチャを再構成し，プロセッサコアの面積やアプリケーション実行時間をもとに最適なハードウェア構成を決定する．再構成の対象となるハードウェアユニットの中心は，演算ユニットなどに代表されるハードウェアユニットである．そのため，ハードウェア/ソフトウェア分割にはハードウェアユニットの面積や遅延の見積りが必要となる（**図 6**）．たとえば，ハードウェアユニットを 1 つ削減すればプロセッサコアの面積が削減され，そのハードウェアユニットがクリティカルパス上に存在してプロセッサコアのクロック周期を決定していれば，クロック周期が短くなる．Packed

☆ Packed SIMD 型演算ユニットは，梱包数や飽和処理の有無などが異なる複数の命令を 1 つの演算ユニットで実行することができるので，実行すべき命令動作を制御する入力を持つ．したがって，加算器，乗算器が付加されているプロセッサコアに対して，付加されている加算器か乗算器，あるいはその両方が Packed SIMD 型加算ユニットおよび乗算ユニットとなった場合，演算ユニットの面積の増減以外に Packed SIMD 型演算ユニットの制御入力を生成する回路が必要となる．

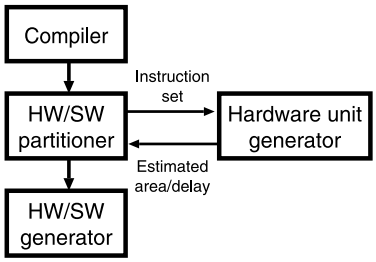

図 6 ハードウェア/ソフトウェア協調合成システムとハードウェアユニット生成
Fig. 6 A hardware/software cosynthesis system and hardware unit generation.

SIMD 型命令を持つプロセッサは，Packed SIMD 型演算ユニットをハードウェアユニットに持つ．Packed SIMD 型演算ユニットは 1 つの演算ユニットで複数の命令を実行するため，演算ユニットで実行する命令の組合せによってハードウェアユニットの面積や遅延の値が異なる．命令の再構成は，プロセッサコアの面積の削減やあるいはアプリケーションの実行時間の削減を目的とするので，命令の再構成によって得られる新たな Packed SIMD 型演算ユニットは，命令を再構成する前の Packed SIMD 型演算ユニットより面積やクリティカルパス遅延の値が小さいものでなければならない．ハードウェア/ソフトウェア分割では，命令を再構成するごとにそのときのプロセッサコアの面積やアプリケーションの実行時間を見積もってプロセッサアーキテクチャを決定するので，高速にハードウェアユニットを構成し，面積と遅延を見積もる必要がある．

3.2 ハードウェアユニット生成系

3.1 節のハードウェア/ソフトウェア分割で必要となるシステムの議論をもとに，ハードウェアユニット生成系を提案する．

3.2.1 アーキテクチャテンプレート

ハードウェアユニットは，部分的な機能を実現するハードウェア（部分機能ユニットと呼ぶ）を複数組み合わせて実現される．部分機能ユニットが実行する処理を部分機能と呼ぶ．ハードウェアユニットを実現するのに必要な部分機能の集合とその接続関係を表したものをアーキテクチャテンプレートと呼ぶ．ハードウェアユニットの面積は，アーキテクチャテンプレートの部分機能を実現する部分機能ユニットの面積の総和に，アーキテクチャテンプレートにより定まる制御部の面積を加えることで見積もられる．ハードウェアユニットの遅延は，アーキテクチャテンプレートにより定まるクリティカルパス上にある部分機能ユニットのクリティカルパス遅延と制御部の遅延から見積もることができる．

Packed SIMD 型演算ユニットは，精度拡張，演算，演算結果のシフト，飽和処理を一連の流れによって処理するハードウェアユニットであり，加算，乗算などの演算の種類の違いを除いてその手順は一定である[11]．したがって，命令形式に対応して，精度拡張部，演算部，固定シフト部，飽和処理部の 4 つの部分機能に分割し，それを順に接続したものを Packed SIMD 型演算ユニット共通のアーキテクチャテンプレートとすることができる．部分機能ユニットは，たとえば精度拡張部に対しては，(1) 精度拡張をしない，(2) 上位をビット拡張する，および (3) 下位をビット拡張するの 3 種類の動作の組合せ（精度拡張をしないもの，精度拡張をしない場合と上位をビット拡張をする場合とを選択できるもの，精度拡張をしない場合，上位をビット拡張する，下位をビット拡張することのどれもができるもの……）の 7 種類の部分機能に対する部分機能ユニットを用意する．演算部，固定シフト部，飽和処理部でも 7 種類の部分機能に対して部分機能ユニットを用意すると仮定すれば，$7 \times 4 = 28$ 種類の部分機能ユニットを用意することで，$7^4 = 2401$ 種類の Packed SIMD 型演算ユニットを構成することができる．このように，1 つのアーキテクチャテンプレートによって容易に多種類の Packed SIMD 型演算ユニットを構成することができる．

Packed SIMD 型演算ユニットのアーキテクチャテンプレートを**図 7** (a) に示す．たとえば，`mul_4_sr3s` と `mul_4h_sl5` を実行するハードウェアユニットは，図 7 (a) のアーキテクチャテンプレートで示されるように，2 つの入力はそれぞれ精度拡張部へと接続され，精度拡張部からの出力が演算部での 2 つの入力となる．演算部によって得られた出力は，シフト部でシフトされ，飽和部で飽和処理をされたあとで演算ユニットの出力となる．精度拡張部は上位 2 つのビット拡張，演算部は梱包数 4 の符号つき Packed SIMD 型乗算および梱包数 4 のビット拡張 Packed SIMD 型乗算，シフト部は右 3 ビットシフトおよび左 5 ビットシフト，飽和処理部は符号つき飽和演算を実現する部分機能ユニットを持つ．図 7 (b) は，アドレッシングユニットのアーキテクチャテンプレートである．アドレッシングユニットは，アドレスレジスタ *dp* と，インデックスレジスタ *dn* および，*dp* 値，*dn* 値およびアドレスモードの指定をもとにアドレス計算をするための演算部から構成される．アドレッシングユニットの入力は *dp* および *dn* に設定する値と，アドレスモードの指定の 3 つである．アドレス計算のための演算部により参

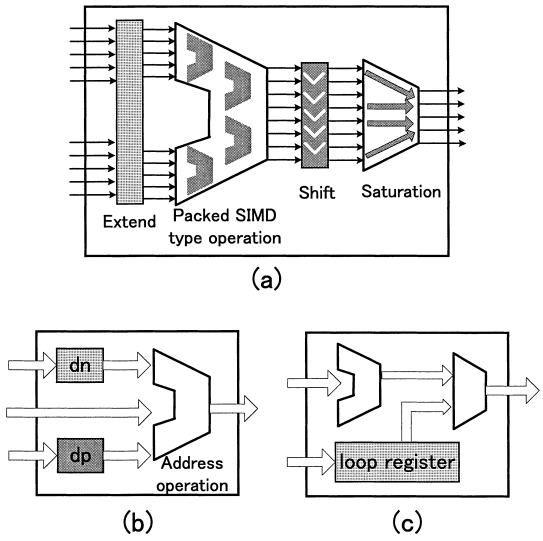

図 7 アーキテクチャテンプレート．(a)Packed SIMD 型演算ユニットのアーキテクチャテンプレート．(b) アドレッシングユニットのアーキテクチャテンプレート．(c) ハードウェアループユニットのアーキテクチャテンプレート

Fig. 7 Architecture templates. (a) The template for a packed SIMD type functional unit. (b) The template for an addressing unit. (c) The template for a hardware loop unit.

照するメモリアドレスが計算されアドレスレジスタの出力となる[7),8)]．図 7 (c) にハードウェアループユニットのアーキテクチャテンプレートを示す．ハードウェアループユニットはループ開始，終了アドレスやループ回数を設定し保持するループレジスタ部と，ループ内命令数からループ開始アドレス，終了アドレスを計算する演算部および次命令アドレスを決定する選択部から構成される．ハードウェアループユニットの入力は，ループレジスタに接続されるループ回数，ループ内命令数などの値を示す入力と，現在のプロセッサの PC 値の 2 系統から成る．ハードウェアループユニットの出力は次の PC 値である[7),8)]．

図 8 に，演算部，シフト部および飽和部の部分機能から構成されるアーキテクチャテンプレートに対して，部分機能ユニットを割り当てて Packed SIMD 型演算ユニットを構成する例を示す．図 8 の (a) では，演算部に面積 1000，遅延 20 の部分機能ユニット，シフト部に面積 20，遅延 2 の部分機能ユニット，飽和部に面積 100，遅延 7 の部分機能ユニットを割り当てて，全体で面積 1120，遅延 29 の Packed SIMD 型演算ユニットの構成を得ている．図 8 の (b) では，(a) の構成から，演算部と飽和部の部分機能ユニットを変更することによって，面積 2320，遅延 20 の Packed SIMD 型演算ユニットの構成を得ている．このように，部分機能ユニットの割当てを変更することで複数のハードウェアユニット構成を得ることができる．

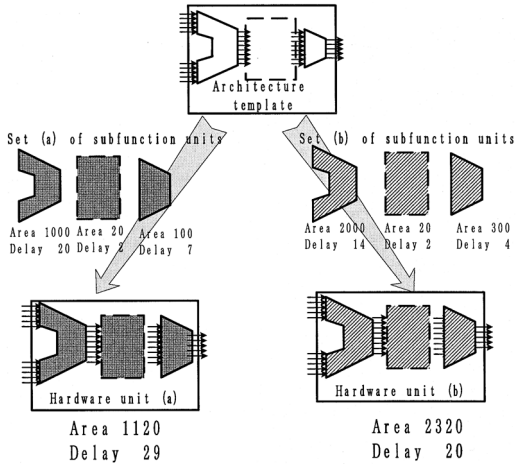

図 8 アーキテクチャテンプレート上での部分機能ユニットの割当て

Fig. 8 Assignment of subfunctional units to subfunctions in an architecture template.

3.2.2 システムの概要

ハードウェアユニット生成系は，1 つのハードウェアユニットで実行される命令の集合と，生成されるハードウェアユニットの面積と遅延の制約を入力とし，ハードウェアユニットの構成を複数列挙し面積と遅延の見積り値を出力する．ハードウェアユニット生成系の概要を**図 9** に示す．ハードウェアユニット生成系は部分機能抽出とアーキテクチャ構成から成る．部分機能抽出では，入力された命令集合をもとにアーキテクチャテンプレートを決定する．アーキテクチャ構成では，アーキテクチャテンプレートの部分機能に，その部分機能を実現する部分機能ユニットを割り当てることによりハードウェアユニットの構成と面積および遅延の見積りを得る．

部分機能ユニットを組み合わせてハードウェアユニットを実現することにより，いくつかの部分機能ユニットを用意するだけで入力された命令集合を実現するハードウェアユニットが生成可能である．これにより，ハードウェア/ソフトウェア分割によって実行する命令が変更されたとき，新たなハードウェアユニットを構成することができ，しかも面積や遅延の制約を与えることにより面積や遅延の削減を目的としたハードウェア/ソフトウェア分割に対して，必要なハードウェアユニットの構成のみを出力することができる．複数の解候補を列挙することで，同じ命令集合に対して何度もハードウェアユニットの面積や遅延の見積り値を呼び出すことなくハードウェア構成を選択することが可

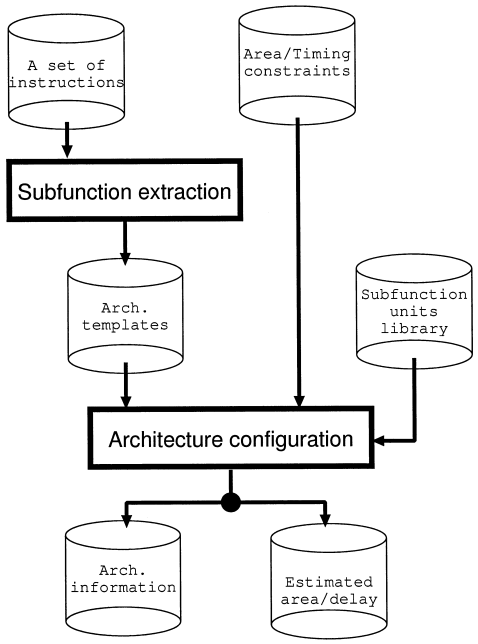

図 9 ハードウェアユニット生成系
Fig. 9 A hardware unit generation system.

能である．したがって，ハードウェア/ソフトウェア分割を高速に実行することができる．

4. アーキテクチャ構成アルゴリズム

本章では，ハードウェアユニット生成系の中心となるアーキテクチャ構成を問題として定義し，高速にハードウェアユニットを構成するアーキテクチャ構成アルゴリズムを提案する．

4.1 アーキテクチャ構成問題

ハードウェアユニット u で実行可能な命令集合 I_u を u の命令集合と呼ぶ．面積制約を満たすとは，ハードウェアユニット u の面積が許容値 a_{max} より小さいことであり，時間制約を満たすとは，ハードウェアユニット u のクリティカルパス遅延が許容値 d_{max} より小さいことである．このときハードウェアユニットのアーキテクチャ構成問題とは，入力として命令集合 I から決定されるアーキテクチャテンプレート，面積制約 a_{max} および時間制約 d_{max} を与えられたときに，面積制約と時間制約を満たし，$I \subseteq I_u$ となるハードウェアユニット u の構成を複数個出力することである．

4.2 アーキテクチャ構成アルゴリズム

アーキテクチャテンプレートの部分機能の集合を $F = \{f_1, f_2, \ldots, f_m\}$ とする．入力されるアーキテクチャテンプレートによって f_i $(i = 1, \ldots, m)$ を実現

Step 1 アーキテクチャテンプレートに基づいて，ハードウェアユニットを構成する．このとき，すべての部分機能に対して，部分機能ユニットの中から遅延が最も小さいものを選ぶ．構成されたハードウェアユニットが時間制約を満たさなければ終了．
Step 2 現在の構成によるハードウェアユニットが面積制約を満たせば，その構成を解の 1 つとして出力する．
Step 3 現在のハードウェアユニットのアーキテクチャから，ある 1 つの部分機能を実現する部分機能ユニットを，現在の部分機能ユニットより面積が小さいユニットの中で最も遅延の小さいユニットに置き換えたときの，ハードウェアユニットの面積とクリティカルパス遅延を調べる．
Step 4 すべての部分機能に対して **Step 3** を実行し，時間制約を満たす中でハードウェアユニットの面積を最小とする構成に変更する．
Step 5 **Step 4** のようなハードウェアユニットが存在する間，**Step 2**–**4** を繰り返す．

図 10 アーキテクチャ構成アルゴリズム
Fig. 10 An algorithm of the architecture configuration problem.

可能な部分機能ユニットの集合 S_i を選択することができる．部分機能ユニット $s \in S_i$ は面積 $a(s)$ と遅延 $d(s)$ を持つ．

アーキテクチャ構成問題は，複数の解を高速に列挙する必要がある．解として得られる個々のハードウェア構成は，同じ遅延を持つハードウェア構成の中で面積の小さいものであることが望ましい．したがって以下のような手法をとる．

まず，各部分機能 f_i $(i = 1, \ldots, m)$ に対して最も遅延の小さい部分機能ユニットを割り当てることでハードウェアユニットを構成する．得られた構成は解候補の中で最も遅延の小さいハードウェアユニットとなり，時間制約を満たしていることが期待される．

次に，1 つの部分機能 f_i に着目し，現在の構成で用いられている部分機能ユニットを，部分機能ユニット集合 S_i の中で現在の部分機能ユニットより面積が小さい中で最も遅延の小さい部分機能ユニットに置き換え，そのときのハードウェアユニットの面積と遅延を見積もる．すべての部分機能に対してこの処理を実行し，得られたハードウェアユニット構成の中で時間制約を満たし最も面積の小さい構成を解の候補とし，新たな構成とする．この操作で得られる新たなハードウェアユニットの構成は m 回のみの部分機能ユニットの交換で得られる．この操作を時間制約を満たさなくなるまで続けることで，徐々に面積の小さい構成を得ることができ，高速に複数の解候補を列挙できる．

最後に，得られた解候補の中で面積制約を満たすものを解とし列挙する．

以上のアルゴリズムを**図 10** に示す．次に，このアルゴリズムの時間計算量を見積もる．部分機能ユニッ

表 3 入力した命令集合（加算）
Table 3 Given instruction set (add operation).

add	add_1_us
add_1_ur1w	add_1_ul1s
add_2_uw	add_2_us
add_2_ur1w	add_2_ul1s
add_4_uw	add_4_us
add_4_ur1w	add_4_ul1s
add_2h_u	add_2l_u
add_2h_ul8	add_2l_ul8
add_4h_u	add_4l_u
add_4h_ul8	add_4l_ul8

トの総数 $\sum|S_i|$ を n とする．同じ部分機能を実現する部分機能ユニットを，遅延の小さい順にそれぞれ整列しておけば，図 10 の **Step 2** は $O(1)$ で実行可能である．すべての部分機能に対して **Step 3** を実行するので **Step 2–4** は $O(m)$ で求められる．**Step 2–4** は，最大で部分機能ユニットの総数回繰り返されるので，時間計算量は $O(mn + n\log n)$ である．次にこのアルゴリズムの空間計算量を見積もる．n 個の部分機能ユニットを整列して保存する．m 個の部分機能を実現する部分機能ユニットを用意することでハードウェアユニットを構成する．したがって空間計算量は $O(m+n)$ である．

5. 計算機実験結果

提案した手法を C 言語を用いて Sun Ultra SPARC（200 MHz，メモリ 128 MB）上に実装し，Packed SIMD 型加算ユニットおよび Packed SIMD 型乗算ユニットに適用した．使用したコンパイラは gcc（version 2.95.2），最適化レベルを O3 とした．部分機能ユニットはあらかじめ VHDL で記述し，Design Compiler を用いて論理合成して面積と遅延の値を算出した．セルライブラリには VDEC ライブラリ（CMOS0.35 μm テクノロジー）を用いた☆．提案手法との比較のため，すべての部分機能ユニットの組合せを列挙することでハードウェアユニットを構成した．この全列挙手法の時間計算量は，部分機能の数を m，部分機能ユニットの総数を n としたとき $O(m^{\frac{n}{m}})$ である．提案するアーキテクチャ構成アルゴリズムと，全列挙手法に対し，入力として**表 3** に示す命令集合と，面積制約 $100,000\,\mu\mathrm{m}^2$ および時間制約 8.00 ns を与えたときの，実験結果を**図 11** に示す．入力された命令集合より，必要となるアーキテクチャテンプレートは一意に決定さ

☆ VDEC 日立ライブラリ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株式会社日立製作所および大日本印刷株式会社の協力で作成されたものである．

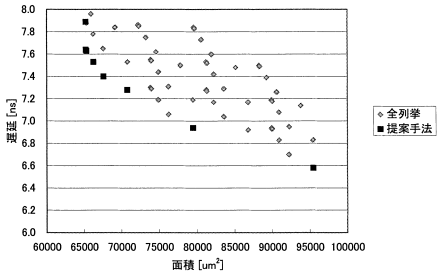

図 11 Packed SIMD 型加算ユニットの実験結果
Fig. 11 Results of a packed SIMD type adding unit.

表 4 表 3 の入力で決まるアーキテクチャテンプレートで使われる部分機能ユニット
Table 4 Subfunctional units used in the architecture template for the instruction set.

精度拡張部の部分機能ユニット	面積 [$\mu\mathrm{m}^2$]	遅延 [ns]
候補 1	21,349	1.05
候補 2	12,682	1.39
加算部の部分機能ユニット	**面積 [$\mu\mathrm{m}^2$]**	**遅延 [ns]**
候補 1	75,699	1.61
候補 2	29,631	3.48
候補 3	29,627	3.73
個補 4	27,981	4.04
候補 5	25,838	9.64
候補 6	25,606	11.34
候補 7	25,534	11.38
候補 8	25,462	11.42
候補 9	25,410	11.63
候補 10	20,577	14.19
候補 11	19,209	14.41
シフト部の部分機能ユニット	**面積 [$\mu\mathrm{m}^2$]**	**遅延 [ns]**
候補 1	26,963	1.11
候補 2	10,982	1.47
飽和部の部分機能ユニット	**面積 [$\mu\mathrm{m}^2$]**	**遅延 [ns]**
候補 1	14,926	0.96
候補 2	11,078	1.08
候補 3	8,821	1.31
候補 4	8,730	1.32

れ，精度拡張部は精度拡張しない，上位のビット拡張および下位のビット拡張の 3 種類の部分機能，演算部は梱包数 1，2，4 の 3 種類を実現できる部分機能，シフト部はシフトしない，左 8 ビットシフト，左 1 ビットシフトおよび右 1 ビットシフトの部分機能，飽和部は飽和処理する場合およびしない場合の部分機能を有するアーキテクチャテンプレートを持つ．それぞれの部分機能を実現できる部分機能ユニットとして，**表 4** に示す部分機能ユニットが選択されアーキテクチャ構成に用いられた．入力として**表 5** に示す命令集合と，面積制約 $900,000\,\mu\mathrm{m}^2$ および時間制約 16.0 ns およ

表 5 入力した命令集合（乗算）
Table 5 Given instruction set (mul operation).

mul_2_sw	mul_2_ss
mul_4_sw	mul_4_ss
mul_2_sr7w	mul_2_sr4s
mul_4_sr4w	mul_4_sr4s

表 7 入力した命令集合（乗加算）
Table 7 Given instruction set (mac operation).

mac_1_uw	mac_1_us
mac_2_uw	mac_2_us
mac_2h_u	mac_2l_u
mac_4_uw	mac_4_us

表 6 表 5 の入力で決まるアーキテクチャテンプレートで使われる部分機能ユニット
Table 6 Subfunctional units used in the architecture template for the instruction set.

精度拡張部の部分機能ユニット	面積 [μm^2]	遅延 [ns]
候補 1	0	0
乗算部の部分機能ユニット	**面積 [μm^2]**	**遅延 [ns]**
候補 1	1,054,518	6.02
候補 2	795,504	7.92
候補 3	709,544	9.80
候補 4	315,938	12.71
シフト部の部分機能ユニット	**面積 [μm^2]**	**遅延 [ns]**
候補 1	11,197	1.04
候補 2	6,832	1.11
飽和部の部分機能ユニット	**面積 [μm^2]**	**遅延 [ns]**
候補 1	19,415	2.41
候補 2	16,297	2.53
候補 3	14,853	2.66
候補 4	13,940	2.76
候補 5	13,849	2.77

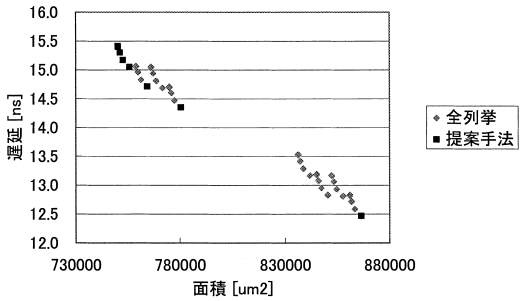

図 12 Packed SIMD 型乗算ユニットの実験結果
Fig. 12 Results of a packed SIMD type multiplying unit.

び**表 6** を与えたときの，実験結果を**図 12** に示す．入力として**表 7** に示す命令集合と，面積制約および時間制約をなしおよび部分機能ユニットとして**表 8** を与えたときの実験結果を**図 13** に示す．

計算機上に実装されたプログラムを 1 万回連続で実行したときの実行時間を**表 9** に示す．図 11，図 12 および図 13 から，提案手法が，複数のハードウェアユニット構成を列挙できることが分かる．全列挙手法に

表 8 表 7 の入力で決まるアーキテクチャテンプレートで使われる部分機能ユニット
Table 8 Subfunctional units used in the architecture template for the instruction set.

精度拡張部の部分機能ユニット	面積 [μm^2]	遅延 [ns]
候補 1	31,123	1.05
候補 2	19,024	1.39
乗算演算部の部分機能ユニット	**面積 [μm^2]**	**遅延 [ns]**
候補 1	1,054,518	6.02
候補 2	795,504	7.92
候補 3	670,944	10.79
候補 4	296,028	13.04
加算演算部の部分機能ユニット	**面積 [μm^2]**	**遅延 [ns]**
候補 1	75,699	1.61
候補 2	29,631	3.48
候補 3	29,627	3.73
個補 4	27,981	4.04
候補 5	25,838	9.64
候補 6	25,606	11.34
候補 7	25,534	11.38
候補 8	25,462	11.42
候補 9	25,410	11.63
候補 10	20,577	14.19
候補 11	19,209	14.41
シフト部の部分機能ユニット	**面積 [μm^2]**	**遅延 [ns]**
候補 1	0	0
飽和部の部分機能ユニット	**面積 [μm^2]**	**遅延 [ns]**
候補 1	14,296	0.96
候補 2	11,078	1.08
候補 3	8,821	1.31
候補 4	8,730	1.32

比べて列挙された構成は Packed SIMD 型加算ユニットで 1 割程度で，Packed SIMD 型乗算ユニットおよび乗加算ユニットで 2 割程度であるが，全列挙で得られる解の中の，同じ遅延を持つ中で最も面積の小さいハードウェアユニットの多くを列挙できた．したがって，ハードウェア/ソフトウェア分割で有用となるハードウェアユニット構成のみを扱うことができるようになる．しかも，表 9 から，その解が全列挙手法に比べておおよそ 1/10 の時間で得られるので，高速にハードウェア/ソフトウェア分割をすることができるようになると考えられる．以上からアルゴリズムの有効性

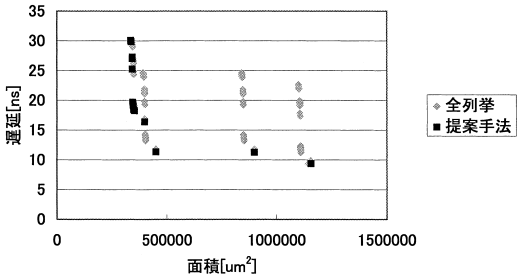

図 13 Packed SIMD 型乗加算ユニットの実験結果
Fig. 13 Results of a packed SIMD type MAC unit.

表 9 1 万回連続実行したときの合計の CPU 時間
Table 9 Execution time of our algorithm repeated 10,000 times.

	提案手法 [s]	全列挙 [s]
加算	6.13	66.3
乗算	1.73	21.0
乗加算	8.69	66.9

を示せたと考える.

6. むすび

Packed SIMD 型命令を持つプロセッサを対象としたハードウェア/ソフトウェア協調合成システムのためのハードウェアユニット生成手法を提案し，高速に，面積・遅延の小さい解を複数列挙することができた.今後は，面積と遅延以外に，消費電力やマルチサイクル演算ユニットを対象としたサイクル数なども考慮したハードウェアユニット生成手法を検討する.

謝辞 本研究に関して，有用な議論，討論をいただいた本学野々垣直浩氏（現東芝）に感謝いたします.

参考文献

1) Akaboshi, H. and Yasuura, H.: COACH: A computer aided design tool for computer architectures, *IEICE Trans. Fundamentals*, Vol.E76-A, No.10, pp.1760–1769 (1993).
2) Basoglu, C., Lee, W. and O'Donnell, J. S.: The MAP 1000A VLIW mediaprocessor, *IEEE Micro*, Vol.20, No.2, pp.48–59 (2000).
3) Diefendorff, K., Dubey, P.K., Hochsprung, R. and Scales, H.: AltiVec extension to PowerPC accelerates media processing, *IEEE Micro*, Vol.20, No.2, pp.85–94 (2000).
4) Ishiwata, S. and Sakurai, T.: Future directions of media processors, *IEICE Trans. Electronics*, Vol.E81-C, No.5, pp.629–635 (1998).
5) 伊藤真紀子，塩見彰睦，佐藤 淳，武内良典，今井正治：パイプライン・ハザードを考慮したプロセッサ生成手法の提案，情報処理学会論文誌，Vol.41, No.4, pp.851–862 (2000).
6) Itoh, M., Higaki, S., Sato, J., Shiomi, A., Takeuchi, Y., Kitajima, A. and Imai, M.: PEAS-III: An ASIP design environment, *Proc. 2000 IEEE International Conference on Computer Design: VLSI in Computers & Processors*, pp.430–436 (2000).
7) Lapsley, P., Bier, J., Shoham, A. and Lee, E.A.: *Processor Fundamentals: Architectures and Features*, Berkeley Design Technology, Inc. (1994–1996).
8) Madisetti, V.K.: *Digital Signal Processors*, IEEE Press (1995).
9) Morifuji, T., Takeuchi, Y., Sato, J. and Imai, M.: Flexible hardware model database management system: Implementation and effectiveness, *Proc. Synthesis and System Integration Mixed Technologies* (*SASIMI'97*), pp.83–89 (1997).
10) 中村泰基，高橋宏政，須賀敦浩，三宅英雄，岡野廣，竹部好正：4Way VLIW 組み込み用途向けマルチメディアプロセッサ，信学技報，CPSY2000-4 (2000).
11) 野々垣直浩，戸川 望，柳澤政生，大附辰夫：画像処理を対象とした Packed SIMD 型命令セットを持つプロセッサのハードウェア/ソフトウェア協調合成システムにおける並列化 C コンパイラ，信学技法，VLD2000-139, ICD2000-215 (2001).
12) Peleg, A. and Weiser, U.: MMX technology extension to the Intel architecture, *IEEE Micro*, Vol.16, No.4, pp.42–50 (1996).
13) Sato, J., Alomary, A.Y., Honma, Y., Nakata, T., Shiomi, A., Hikichi, N. and Imai, M.: PEAS-I: A hardware/software codesign system for ASIP development, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, Vol.E77-A, No.3, pp.483–491 (1994).
14) Togawa, N., Yanagisawa, M. and Ohtsuki, T.: A hardware/software cosynthesis system for digital signal processor cores, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, Vol.E82-A, No.11 (1999).
15) Tremblay, M., O'Connor, J.M., Narayanan, V. and He, L.: VIS speeds new media processing, *IEEE Micro*, Vol.16, No.4, pp.10–20 (1996).
16) Yang, J.-H., Kim, B.-W., Nam, S.-J., Kwon, Y.-S., Lee, D.-H., Lee, J.-Y., Hwang, C.-S., Lee, Y.-H., Hwang, S.-H., Park, I.-C. and Kyung, C.-M.: MetaCore: An application-specific programmable DSP development system, *IEEE Trans. Very Large Scale Integration* (*VLSI*)

systems, Vol.8, No.2, pp.173–183 (2000).

(平成 13 年 9 月 14 日受付)
(平成 14 年 3 月 14 日採録)

宮岡祐一郎（学生会員）

2000 年早稲田大学理工学部電子・情報通信学科卒業．2002 年同大学大学院修士課程修了．現在同大学院博士課程在学．VLSI 設計，特にマイクロプロセッサのハードウェア/ソフトウェア協調設計に関する研究に従事．電子情報通信学会学生会員．

戸川　望（正会員）

1992 年早稲田大学理工学部電子通信学科卒業．1994 年同大学大学院修士課程修了．1997 年同後期課程修了．博士（工学）．現在北九州市立大学国際環境工学部情報メディア工学科助教授および早稲田大学理工学総合研究センター客員助教授．VLSI 設計，計算幾何学，グラフ理論等の研究に従事．1996 年第 9 回安藤博記念学術奨励賞受賞．1997 年度（第 21 回）丹羽記念賞受賞．IEEE，電子情報通信学会各会員．

柳澤　政生（正会員）

1981 年早稲田大学理工学部電子通信学科卒業．1983 年同大学大学院博士前期課程修了．1986 年同後期課程修了．博士（工学）．現在早稲田大学理工学部電子・情報通信学科教授．電子回路の設計自動化，ノイズ解析，計算幾何学，グラフ理論等の研究に従事．1987 年度丹羽記念賞受賞．1990 年安藤博学術奨励賞受賞．IEEE，ACM，電子情報通信学会，プリント回路学会，日本 OR 学会各会員．

大附　辰夫（正会員）

1963 年早稲田大学理工学部電気通信学科卒業．1965 年同大学大学院修士課程修了．同年日本電気（株）入社．1980 年同退社．現在，早稲田大学理工学部電子・情報通信学科教授．博士（工学）．システム LSI およびこれに関連した基礎研究に従事．1969 年度電子情報通信学会論文賞受賞．1994 年度第 32 回電子情報通信学会業績賞受賞．IEEE CAS Society より Guillmin-Cauer Prize Award（1974 年），Meritorious Service Award（1995 年），Golden Jubilee Medal（2000 年）受賞．2000 年 IEEE より 3rd Millennium Medal 受賞．共著「VLSI の設計 I」（岩波書店），編共著「Layout Design and Verification」（North-Holland）．IEEE，電子情報通信学会フェロー，電気学会，プリント回路学会各会員．