計算機アーキテクチャ　103－20
設　計　自　動　化　69－20
(1993. 12. 17)

ASIP向きハードウェア／ソフトウェア・コデザインシステム PEAS-Iにおけるハードウェア生成手法

中田武治† 佐藤淳†† 塩見彰睦† 今井正治† 引地信之†††

†豊橋技術科学大学情報工学系 †† 鶴岡工業高等専門学校 ††† SRA

†〒441 愛知県豊橋市天伯町字雲雀ヶ丘1-1
††〒997 山形県鶴岡市大字井岡字沢田104
†††〒102 東京都千代田区平河町1-1-1

あらまし　PEAS-Iは，応用プログラムから特定分野で必要とされる特徴を抽出し，ASIPのハードウェアとソフトウェア開発環境を同時に生成するシステムである．ハードウェア生成系は，応用プログラムの解析結果をもとに決定された各命令の実現方法を入力として，CPUコアのデザインをハードウェア記述言語HDLの形式で生成する．ハードウェア生成系に与えるアーキテクチャ情報の変更により，それぞれのCPUコアのデザインが容易に得られる事が確認された．また，本システムを用いた実験の結果，レジスタ数と演算器のハードウェア量のトレードオフを考慮して命令セットを決定する方法についての基礎資料が得られた．

和文キーワード　VLSI, ASIP, ハードウェア／ソフトウェア協調設計，高位論理合成，ハードウェア記述言語

Hardware Generation Methodology in PEAS-I : A Hardware/Software Codesign System for ASIP

Takeharu NAKATA† Jun SATO†† Akichika SHIOMI† Masaharu IMAI† Nobuyuki HIKICHI†††

†Toyohashi University of Technology †† Tsuruoka National College of Technology ††† SRA

†1-1 Hibarigaoka, Tenpaku-cho, Toyohashi, Aichi, 441 Japan
††Sawada, Ioka, Tsuruoka, Yamagata, 997 Japan
†††1-1-1 Hirakawa-cho, Chiyoda-ku, Tokyo, 102 Japan

Abstract

In this paper, we present the hardware generation method in PEAS-I, which is a hardware/software codesign system for ASIP development. PEAS-I system analyzes a set of application programs and associated data set and determines optimal instruction set. Hardware generator accepts the determined instruction set and then generates CPU core design in the form of an HDL. The experimental results give the basic information to optimize the ASIP architecture taking the trade-off of computational module and register file into account.

英文 key words　VLSI, ASIP, Hardware/Software Codesign, High-level synthesis, Hardware description language

1 はじめに

近年の半導体の集積度の向上等に伴い，CPU コアと周辺回路を 1 チップに集積した ASIP (Application Specific Integrated Processor) の開発が行われるようになった．図1に ASIP の構成例を示す．ASIP は 1 チップでシステムを実現するため，汎用 CPU に周辺回路を組み合わせたシステムよりも小さく，組み込み用途に多くの需要があると思われる．

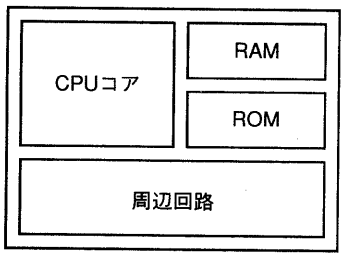

図 1: ASIP の構成例

ASIP システムの構築のためには，ASIP のハードウェア設計以外にもその ASIP のための応用プログラム開発ツール（コンパイラ，シミュレータ等）も開発する必要がある．数万ゲート規模の ASIP の設計を従来のゲートレベルでの設計手法で行うと，長期の設計期間を必要とする．また，今後の半導体技術の進歩を考えると設計期間はさらに長くなることが予測される．一方，従来の設計手法では，ハードウェア・アーキテクチャの決定の後に応用プログラム開発環境の設計が開始されるため，ASIP システム設計の全体の工程が長くなる．設計期間の長期化は，ハードウェア，ソフトウェアの設計結果をフィードバックしてシステムを再設計することを困難にする．

近年，HDL（ハードウェア記述言語）や CAD 技術の発達により，論理合成等の作業が自動化され，簡単な CPU ならば数人日程度で設計が完了するようになってきている．しかし，チップのアーキテクチャは設計者の経験にもとづいて決定され，形式的な方法（数理的最適化手法）はあまり用いられていない．

ASIP の設計期間を短縮し，ハードウェアとソフトウェアのバランスがとれたシステムを設計するためには，ハードウェアとソフトウェアの協調設計が必要である．この協調設計の目的となるのは，ハードウェアとソフトウェアの最適な機能分割と，設計期間の短縮である．著者らは，ASIP 開発のための協調設計システム PEAS (Practical Environment for ASIP Development) を提案し，PEAS-I を試作した [1].

本稿では，ハードウェア生成系の現状とともに，現在の命令セットのみの最適化から，レジスタ個数を含む最適化のための基礎的な実験結果を報告する．

2 PEAS-I システム

2.1 概要

PEAS-I システムは，C 言語で記述された応用プログラムとデータを入力し，与えられた制約条件のもとで最大の性能を持つ CPU コアを自動設計すると共に，その CPU コアのための C コンパイラ，シミュレータ等の応用プログラム開発ツールを自動生成するシステムである．

PEAS-I は，以下の特徴を持つ．

- 命令セットレベルで最適な CPU コアを生成する
- CPU コアを HDL の形で生成する
- クロス開発環境を自動生成する
- スループットが短い

2.2 構成

PEAS-I システムの構成を図2に示す．PEAS-I システムは以下の 4 つのサブシステムから構成されている．

(1) 応用プログラム解析系
C 言語で記述された応用プログラムとそのプログラムへの入力データの動的解析を行い，命令の実行頻度等を求める．

(2) アーキテクチャ情報生成系
(1) で得られた解析結果をもとに，チップ面積等の制約条件のもとで性能を最大にする命令セット等のアーキテクチャ情報を決定する．

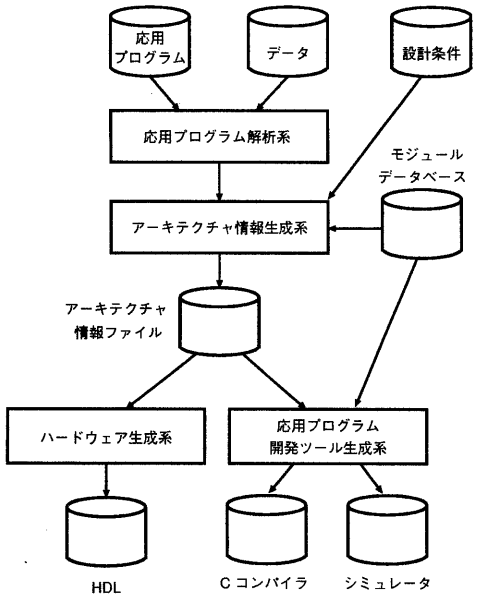

図 2: PEAS-I システムの構成

(3) ハードウェア生成系
(2) で得られたアーキテクチャ情報にもとづいて，CPU コアを HDL の形で生成する．

(4) 応用プログラム開発ツール生成系
(2) で得られたアーキテクチャ情報にもとづいて，C コンパイラ，シミュレータ等の応用プログラム開発ツールを生成する．

現在のバージョンでは HDL として NTT で開発された高位合成システム PARTHENON の SFL (Structured Function description Language) を用いている．PARTHENON は動作シミュレータ，シンセサイザ，オプティマイザからなる LSI 設計システムである [2]．ハードウェア生成系で生成された CPU コアの HDL 記述は，PARTHENON を用いて論理合成され，ネットリストの形式で出力される．また，C コンパイラは，GNU C コンパイラ [3] を利用して自動生成される．

2.3 命令セット最適化手法

PEAS-I システムはハードウェアとソフトウェアの機能分割を命令セット・アーキテクチャレベルで行う．命令セットは GNU C コンパイラの中間コードである RTL (Register Transfer Language) に対応している．この命令セットを表 1 に示すように次の 3 つに分類し，命令の実現方法を変化させる事により，特定分野の応用プログラムに適した CPU コアの生成を行う．

表 1: 命令セットの分類

種類	項目	命令
PRTL	演算	add sub and ior xor one_cmpl neg ashl1 ashr1 lshl1 lshr1
	転送	mov set
	制御	jmp nop beq bne bgt blt bltu bge bgeu bleu bgtu call return
BRTL	演算	mul umul div mod udiv umod trunc extend zero_extend rotate ashr ashl lshr lshl
	転送	movpc
XRTL		user-defined functions abs sqrt etc.

(1) *Primitive RTL (PRTL)*
応用プログラムの実行のために絶対に必要とされ，必ずハードウェアで実現される RTL

(2) *Basic RTL (BRTL)*
ハードウェアで直接実現できるが，PRTL の組み合わせによっても実現できる RTL

(3) *Extended RTL (XRTL)*
ライブラリ関数，ユーザ定義関数に対応する RTL

現在のバージョンでは XRTL はサポートしておらず，BRTL の実現方法だけを変化させている．

生成される CPU コアの性能を最大にするためには，命令セットのそれぞれの実現方法の組み合わせを調べればよい．PEAS-I では，チップ面積等の制約条件のもとで性能を最大にする問題を，組み合わ

せ問題として定式化し，この問題を解くことにより最適な命令セットを決定する [4].

また，命令セットを決定した時点でチップ面積，応用プログラムの実行時間を見積もる事ができ，レジスタファイルのレジスタ数やチップ面積等のパラメータの組み合わせを変えて数種類のアーキテクチャ構成について性能評価を繰り返すことによって，最適なアーキテクチャが選択できる.

3 ハードウェア生成系

3.1 アーキテクチャモデル

ハードウェア生成系で生成される CPU コアは以下の特徴を持つ.

- 32 ビットのハーバードアーキテクチャ
- 命令は 3 アドレスの汎用レジスタ方式
- ロード/ストア・マシンである
- 命令セットは GCC の中間言語 RTL に対応している
- パイプライン制御を行う

CPU コアの構成を図 3に示す. 2.3 節で述べた PRTL を実現するためのハードウェアである ALU や 1 ビットシフタ等は，PEAS-I が生成する CPU コアの最小構成であり，これを kernel と呼ぶ. kernel によってハードウェアで実現されている命令群，すなわち PRTL を使用すれば，この命令を組み合わせることにより BRTL，XRTL に相当する動作を構成することができ，任意の C プログラムの実行が可能である. しかし，BRTL，XRTL を PRTL だけで実現した場合には，プログラムの実行サイクル数は，BRTL や XRTL をハードウェアで実現した場合に比べて増大する.

応用プログラムの高速な実行のためには，応用プログラムでの実行頻度が高い命令をハードウェアで実現すれば良い. 乗算命令等の BRTL をハードウェアで実現する場合には，kernel に命令を実行する演算器をバスで結合させた構成となる.

ハードウェアでの命令の実現方法は，どの演算器を用いて命令を実行するかにより決定される.

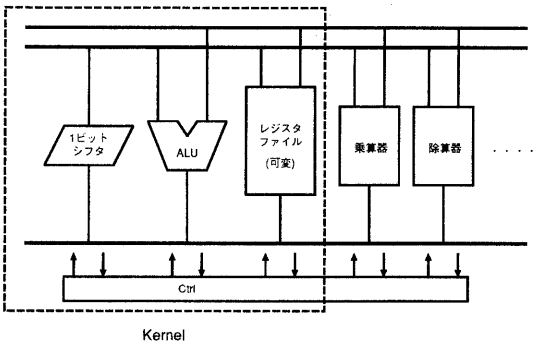

図 3: CPU コアの構成

乗/除算命令をハードウェアで実現する場合，異なるアルゴリズムを実現した数種類の演算器の中からどれか 1 つを選択する. これらの乗/除算器はそれぞれゲート数，実行速度等が異なる.

3.2 パイプラインの構成

パイプラインは次に示す 4 つのステージより成る.

IF	命令フェッチ，デコードおよびレジスタフェッチ
EX	実効アドレスの計算および演算
MEM	メモリアクセス
WR	レジスタファイルへの書き込み

現在生成される CPU コアは，パイプライン・インターロックを伴う制御を行う. また，図 4に示すように、ステージ EX の演算結果を後続の命令に対して，レジスタバイパスを行う. ただし，ロード命令と後続命令 1 のリソースが重なった場合，ロードされる値の決定はステージ MEM で行われるため，図 5に示すようにレジスタバイパスが行われない. このためロード命令と後続命令 1 との間に遅延スロットが必要である. 遅延スロット以降の 2 命令に対しては，レジスタバイパスが行われる.

マルチサイクル命令に対するパイプラインの制御は図 6のようにして行う.

3.3 CPU コアの生成方法

図 7に CPU コアの HDL 記述の生成方法を示す. ハードウェア生成系の入力は，アーキテクチャ情報

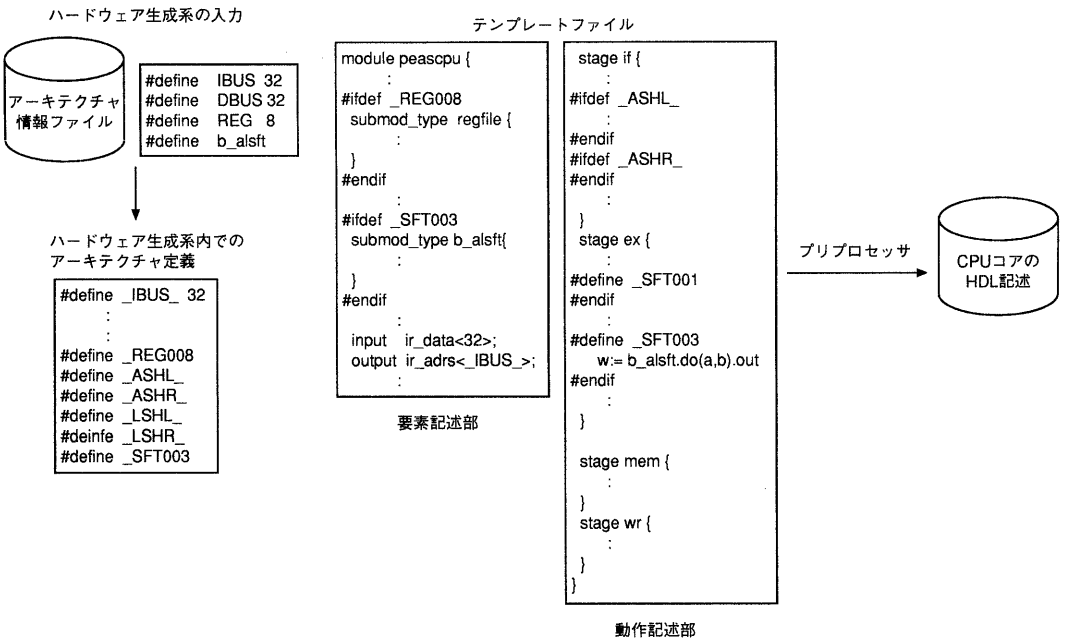

図 7: HDL 記述の生成方法

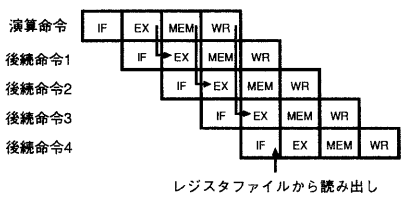

図 4: レジスタバイパス

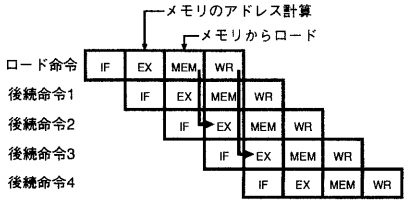

図 5: ロード命令による遅延スロット

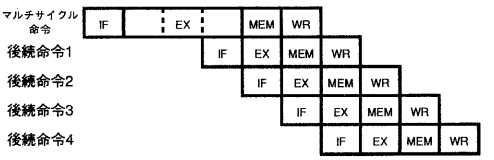

図 6: マルチサイクル命令の制御

ファイルである．このファイルには，

(1) 命令メモリ，データメモリのアドレス空間

(2) レジスタファイルのレジスタの個数

(3) BRTL をハードウェアで実現するための演算器の名前

が記述されている．(1)，(2) は，ユーザが決定し，システムの入力として与えられる．(3) は，アーキテクチャ情報生成系によって決定される．演算器名からハードウェアで実現する命令等を決定し，ハードウェア生成系内のアーキテクチャ定義を行う．

この内部でのアーキテクチャ定義と，ハードウェア生成系が持つテンプレートをもとにCPUコアの生成を行う．テンプレートにはBRTLを含む実際のCPUの動作が記述されている．この記述の中で，ステージIFのBRTLのデコード部分と，ステージEXのBRTLを実行する付加的な演算器の動作部分は，命令をハードウェアで実現しない場合，プリプロセッサにより削除される．ある命令をハードウェアで実現する場合は，ステージIF，EXの動作部分が選択され，使用する演算器自身の記述が取り込まれる．

4 CPUコアのアーキテクチャに関する考察

この節ではPEAS-Iで生成するCPUコアのアーキテクチャ，パイプライン段数等に関する考察を行う．

PEAS-Iにおけるハードウェア生成系に対する要求は，命令の実現方法の変化に対応したCPUコアを生成することである．これを実現するために，3.1節で述べた基本的なアーキテクチャは変化させず，バスにつながる演算器構成を変化させて命令のハードウェア化に対応することとした．

このような構成にしたことにより，HDL記述の生成と，生成されるCPUコアのハードウェア量の予測が簡単になった．この実験結果については次節で述べる．

次に，パイプライン段数について考える．以下に示すのは，パイプライン段数に関する一般的な特徴である．

(1) ハザードがなければ，ステージ数が多いほど1命令あたりの実行時間は短くなる．

(2) ハザードがあると，ステージ数が多いほど遅延スロットが多くなる．

(3) ステージ数が多いとハードウェア量が増加する．

このため，ハードウェア量と実行時間のトレードオフを考慮して，適切なステージ数を選択することが重要である．

PEAS-Iで生成されるCPUコアはそれぞれ演算器構成が異なるため，ステージEXを分割してパイプラインピッチを短くすることは難しいと思われた．そのため，他のステージのピッチを考慮して，PEAS-Iでは4ステージのパイプライン制御とした．

5 実験結果

5.1 予測ゲート数と合成結果との比較

表2に示すサンプルプログラムを用いて実験を行った．

表3に示す条件で，PARTHENONの論理合成機能を用いて，生成されたCPUコアの合成を行った．表4に生成されたCPUコアのゲート数の予測と合成結果の比較を示す．サンプルプログラムはNORMであり，モジュールの選択はアーキテクチャ生成系のIMSP (Instruction set implementation Method Selection Problem) によるものである．ここで，b_sftはバレルシフタを表す．また，mul_32, mul_17, mul_3, mul_1は実行時間がそれぞれ32, 17, 3, 1サイクルの乗算器を表す．ゲート数の予測は，kernelと演算器のゲート数の単純和によって求めたものであるが，誤差率は最大でも2.0 %であった．

表2: サンプルプログラム

名前	機能
FPE	浮動小数点数のエミュレータ
TOT	オイラーの関数
GCD	2数の最大公約数
ICR	立方根の整数部分
NORM	ベクトルのユークリッドノルム
FFT	高速フーリエ変換

表3: 設計条件

項目	条件
動作周波数	10 MHz
テクノロジ	1μm CMOS
ライブラリ	VSC370

表 4: 予測ゲート数と合成結果との比較

選択されたモジュール	予測ゲート数	合成結果	誤差率
kernel	15604.0	15604.0	0.0
kernel + b_sft	16337.0	16526.5	1.1
kernel + b_sft + mul_32	18562.5	18923.0	1.9
kernel + b_sft + mul_17	19374.5	19617.0	1.2
kernel + b_sft + mul_3	22107.0	22414.0	1.4
kernel + b_sft + mul_1	24058.5	24545.5	2.0

5.2 レジスタ数と実行サイクル数の関係

図8にレジスタファイルのレジスタ数と応用プログラムの実行サイクル数の関係を示す．実行サイクル数は，レジスタ数6（FPE と FFT は8）の場合の実行サイクル数を100 %としたものである．レジスタ数の増加に伴い，ロード/ストアおよびメモリからのロードによるハザードが減少するため，実行サイクル数が減少する．

なお CPU コアは，全ての BRTL を最も実行サイクル数の短いハードウェアで実現した最大構成である．最大構成でない場合は，BRTL を PRTL の組み合わせに展開するため，ロード/ストアの回数がさらに増加する．

この実行サイクル数の減少は，レジスタの個数と応用プログラムの内部変数（working set）の個数が等しくなるまで続く．また，各プログラムの内部変数の使用状況によって，実行サイクル数の減少傾向は異なる．例えば NORM, GCD についてはレジスタ数7で実行サイクル数の減少が終了しているが，FPE ではレジスタ数14まで減少し，FFT ではレジスタ数16までに減少が終了しない．また，FPE ではレジスタ数を14にしても実行サイクル数は89.1 %に減少するだけであるが，FFT ではレジスタ数を16に増加させると実行サイクル数は49.7 %まで減少する．

表5にレジスタファイルのハードウェア量を，表6に演算器のハードウェア量を示す．regfile の括弧内の数字はレジスタ数を表す．レジスタ数8と12のレジスタファイルのハードウェア量の差は2013ゲートであり，ほぼ mul_32 のハードウェア量に等しい．また，レジスタ数8と16との差は4054ゲートである．レジスタ数を増やすことにより，命令のハードウェア化に使用できるハードウェア量は減少する．しかし，応用プログラムによっては，ハードウェアにより BRTL を実現するよりレジスタファイルを大きくする方が，高速な実行が可能であると考えられる．

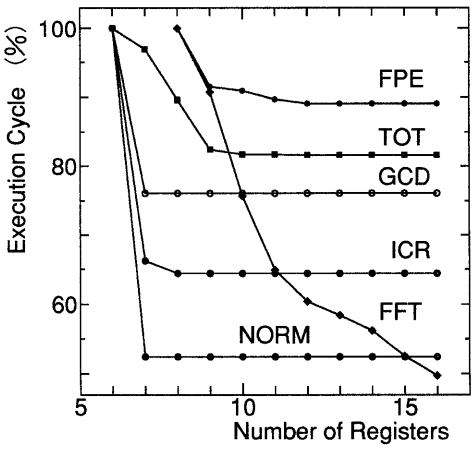

図 8: レジスタ数と実行サイクル数の関係

現在の PEAS-I システムでは，レジスタ数はユーザが与え，システムはそのレジスタ数のもとで最適な命令セットを持つ CPU コアを生成する．レジスタ数を含む命令セットの最適化のためには，レジスタ数と演算器に使用できるハードウェア量のトレードオフを考慮しなければならない．そのためには，現在の命令レベルでの応用プログラム解析でなく，応用プログラムの内部変数の挙動を含む解析を行う

必要がある．

表 5: レジスタファイルのハードウェア量

モジュール名	ゲート数
regfile(8)	3493.0
regfile(12)	5506.0
regfile(16)	7547.0

表 6: 演算器のハードウェア量

演算器名	ゲート数
b_sft	733.0
mul_1	7721.5
mul_3	5770.0
mul_17	3037.5
mul_32	2225.5

6 おわりに

本稿では，PEAS-I システムにおけるハードウェア生成系について，生成される CPU コアのアーキテクチャ，生成方法について述べた．ハードウェア生成系は基本的な命令を実現する kernel に付加的な演算器をバスで結合させる構成によって命令セットの変化に対応している．各命令の実現方法の変化に対してそれぞれの CPU コアの HDL 記述が容易に得られることが確認された．また，生成される CPU コアのゲート数の見積もりも数%以内の誤差で得られることが知られた．

また，レジスタ数と実行時間の関係について調べ，応用プログラムの内部変数の挙動を含む解析と，レジスタ数と演算器のハードウェア量のトレードオフを考慮した命令セットの決定が必要となることについて述べた．

本実験の手法にもとづいて，レジスタファイルと命令セットの間のトレードオフを考慮した ASIP のアーキテクチャの最適化が行えると考えられるが，これは今後の課題である．

謝辞

本研究を進めるにあたりご討論頂いた豊橋技術科学大学 VLSI 設計研究室の本間啓道氏ならびに諸兄に感謝いたします．また，設計ツール及びライブラリ等をご提供して頂いた NTT コミュニケーション科学研究所，VLSI テクノロジー社，ならびに研究をご支援頂く（株）SRA に感謝いたします．

参考文献

[1] Jun Sato, et al: “Current Status of a Hardware/Software Co-design System PEAS-I for ASIP Development,” 第 6 回回路とシステム軽井沢ワークショップ, pp.513-518, 1993.

[2] NTT データ通信: “PARTHENON User's Manual,”1989.

[3] Stallmen, R:“Using and Porting GNU C Compiler,Version 1.40,” Free Software Foundation Inc.,1991.

[4] Aluddin Alomary, et al: “An ASIP Instruction Set Optimization Algorithm with Function Module Sharing Constraint,” IEICE Trans., Vol.E76-A.No.10, pp.1713-1720, 1993.

[5] 中田 武治, 他: “ASIP 向きハードウェア/ソフトウェア・コデザインシステム PEAS-I のハードウェア生成系, ” 情報処理学会 DA シンポジウム’93 論文集, pp17-20, 1993.