計算機アーキテクチャ　103－22
設　計　自　動　化　69－22
(1993. 12. 17)

# ハードウェア／ソフトウェア同時協調設計のための Soft-Core Processor

中村秀一　　安浦 寛人

九州大学 大学院総合理工学研究科 情報システム学専攻

〒816 春日市春日公園 6-1

*E-mail: {nakamura, yasuura}@is.kyushu-u.ac.jp*

あらまし

Soft-Core Processor を用いた新しい ASIC 設計手法を提案する．本手法は，コアプロセッサと基本ソフトウェアをベースとしたシステム仕様の決定，ハードウェアとソフトウェアとの境界の変更，システム性能の評価手法，ハードウェアの再設計およびそれに伴う基本ソフトウェアの自動変更，開発環境の整備といったシステム全体の設計手法を体系的に提案するもので，統括的かつ実際的な手法である．本報告では Soft-Core Processor のコンセプトを紹介し，設計例を通して本手法に要求される要素技術とについて考察する．

和文キーワード：Soft-Core Processor，VLSI，ASIC，ハードウェア/ソフトウェア協調設計，高位論理合成，

# A Soft-Core Processor for Hardware/Software Co-design

Shuichi Nakamura　Hiroto Yasuura

Department of Information Systems
Interdisciplinary Graduate School of Engineering Sciences
Kyushu University
Kasuga-shi, Fukuoka 816 Japan

*E-mail: {nakamura, yasuura}@is.kyushu-u.ac.jp*

**Abstract**

We proposed a new method for designing an ASIC with Soft-Core Processor. In our method, it is able to decide system specification based on a core processor and basic software, and then to change boundary between hardware and software, to estimate system performance, to modify basic software according to redesign of hardware, and finally to get an optimal system solution. In this paper, we introduce the concept of Soft-Core Processor, and consider a tools required to this method through an example.

英文 **key word**: Soft-Core Processor，VLSI，ASIC，Hardware/Software Co-Design，High-level synthesis

# 1 はじめに

性能に対する要求から，専用回路やASICが使用されている．近年のCAD(コンピュータ支援設計)技術やプロセス技術の発達により，開発にかかるコストは引き下げられ，今後ますます専用回路やASICの使用は普及するとみられている．

従来の設計手法では，ハードウェアのレイアウト・ライブラリとユーザ側で設計した高機能周辺回路部のレイアウトを組み合わせてASICを設計する．しかし，選択したレイアウト・ライブラリのモジュールコアに不必要なハードウェア機能を実装している場合もある．このような不必要なハードウェア機能を持つことは製造コストの観点からは望ましくない．

一方，高位合成技術の進歩により，ハードウェア記述言語(HDL)で記述されたハードウェアは論理合成，レイアウト合成され，最終的なレイアウト・パターンまで自動的に生成される．したがって，高機能周辺回路部だけでなく，他の周辺回路やコアプロセッサ部もHDLで記述すれば，それらを自動合成してVLSIのレイアウト・パターンを生成することができる．HDLで記述されたハードウェアは比較的容易に変更が可能であり，したがって，従来のASIC設計で生じたような不必要なハードウェア機能は持つことはなく，製造コストの無駄を省くことができる．(図1)

しかし，このようにして開発されたハードウェアを組み込むためのシステムは，開発される度に開発環境を整えなくてはならず，システムの設計に要する期間は長くならざるを得ない．しかも，こうした開発環境の再利用性を考慮していない場合，特定用途向けのシステムの開発環境を他のシステムの開発環境に再利用することは非常に困難である．

本稿で提案するSoft-Core Processorを利用した新しいASIC設計手法は，コアプロセッサと基本ソフトウェアをベースとしたシステム仕様の決定，ソフトウェアとハードウェアの境界の変更，システム性能の評価手法，ハードウェアの再設計およびそれに伴う基本ソフトウェアの自動変更といったシステム全体の設計手法を体系的に提案するもので，統括的かつ実際的な手法である．

本稿では，Soft-Core Processor単体と再設計環境となるCADに焦点を絞って議論する．2章でハードウェア／ソフトウェア協調設計に関連する研究の中での本研究の位置付けを明確にし，3章でSoft-Core Processorの概念について説明し，4章でSoft-Core Processorを用いた新しい設計手法の実験例を示し，5章で実験の結果について考察する．

# 2 ハードウェア／ソフトウェア協調設計

ハードウェアとソフトウェアを組み合わせて処理を特定の処理を行うものをシステムと呼ぶことにする．従来のシステム設計手法では，ハードウェアとソフトウェアの間のインターフェイスを固定し，そのインターフェイスを遵守してハードウェアやソフトウェアをそれぞれに設計する．しかし，ASIC等によりソフトウェアで行っていた処理をハードウェアに取り込むといったことが頻繁に起こるようになり，ハードウェアとソフトウェアの間のインターフェイスはもはや固定ではなくなってきた．そこで，従来分離されていたハードウェアとソフトウェアの設計を同時に協調して行う新しいシステム設計手法が提案され始めている．

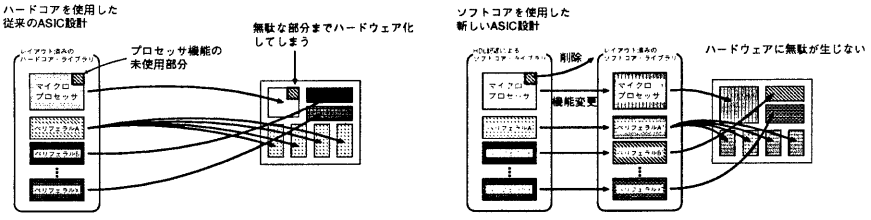


図 1: ASIC 設計

新しいシステム設計では，システム全体としてのインターフェイスは固定し，ハードウェアとソフトウェアの間のインターフェイスを変更し，目的にあったシステムを構築するという手法がとられる．(図 2)

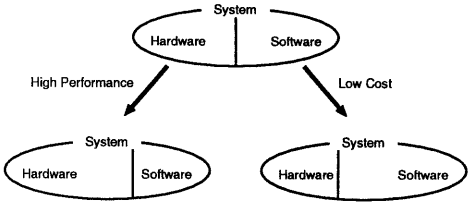


図 2: ハードウェア／ソフトウェア協調設計

VULCAN-II[2] は Stanford 大で開発されている協調設計手法である．ハードウェア C 言語によりシステムの統一的な記述を行い，システム全体を専用ハードウェアでインプリメントした出発点から既存のプロセッサの命令シーケンスで専用ハードウェアの一部の機能を置換し，ハードウェアコストを下げることを目的とする．

ADAS[3] は Southern California 大で開発されている協調設計手法である．手法の特徴は，命令セットアーキテクチャを設計者が指定し，パイプライン動作のステージ分割を任意に行ってパイプラインプロセッサを設計する点である．

PEAS[4] は豊橋技科大で開発されている協調設計手法である．手法の特徴は，ハードウェア・モジュールのライブラリを与え，ハードウェア・モジュールのハードウェア量とソフトウェアの実行時間の概算から制約条件に従って最適なモジュールの組合せを選択する点である．

これらの設計手法は大きく 2 つに分類することができる．1 つは，従来の ASIC 設計手法にしたがってコアプロセッサを固定して周辺を設計する手法で，もう 1 つは，コアプロセッサを最初から設計する手法である．前者では，既存のコアプロセッサのための開発環境や，設計ライブラリが整っているため，設計に要する期間が短いことが利点である．欠点としては，コアプロセッサが一種のブラックボックスであり，その中身が変更されないため，システム構成が最適でない場合が生ずる．また，他のコアプロセッサを使用する場合はコアプロセッサに応じた開発環境が別に必要になる．後者では，命令セットアーキテクチャや演算器をうまく選択すればシステム構成を最適にすることができることが利点であるが，反面，はじめから設計をするため，ベースとなるハードウェアのライブラリやソフトウェア設計や検証のための環境やツールが整っていないので，これらの開発環境を新しく作成することのオーバーヘッドにより，システムの設計に要する期間が長くなる．

今回提案する Soft-Core Processor を用いたシステム設計手法はこれら 2 つの手法の利点を兼ね備えた特徴を持つ．すなわち，システムの開発環境や設計ライブラリが整っていて設計に要する期間が短く，システム構成を最適にするためにコアプロセッサの中身まで変更することができる．また，一つの Soft-Core Processor アーキテクチャにより開発ができるため，開発するシステムごとに開発環境を作りかえる必要がない．(表 1)

表 1: ASIC 設計手法の特徴

| | ハードウェア内部の変更可能性 | 利用可能な開発環境や設計データ |
|---|---|---|
| 従来のコアプロセッサ固定の設計手法 | × | ○ |
| コアプロセッサを変更する設計手法 | ○ | × |
| 本手法 | ○ | ○ |

# 3 Soft-Core Processor

## 3.1 背景

従来の ASIC 設計ではコアプロセッサ部等のモジュールコアの中身には手をつけずに，ユーザがインターフェイス情報を利用して設計した高機能周辺回路を付加して高速化を図っていた．このような設計手法ではモジュールコアの中身を変更できないため，利用しないハードウェア・モジュールのために他の用途に利用可能なシリコン面積を使用している場合も少なくない．

このようなモジュールコアの中身を変更できない設計ライブラリのことを**ハードコア・モジュール・ライブラリ**と呼ぶことにする

利用可能なシリコン面積を有効に使用するためには，利用しないハードウェア・モジュールを削除したり利用頻度の低いハードウェア・モジュールの処理内容をソフトウェアで実現したりする，いわゆるハードウェア／ソフトウェア協調設計を行う必要がある．

ところで，CAD 技術の進歩は HDL からハードウェア・モジュールの中身を自由に設計することを可能にした．すなわち，HDL による設計記述から高位合成技術を使用してレイアウト・パターンまで自動生成することが可能となった．したがって，HDL による設計ライブラリ記述をシステムの構成に適したように書き直せば，シリコン面積を有効に利用したコアプロセッサのレイアウト・パターンを得ることができる．このようなモジュールコアの中身を変更できる設計ライブラリのことをハードコア・モジュール・ライブラリに対して，**ソフトコア・モジュール・ライブラリ**と呼ぶことにする．また，このようにコアプロセッサの中身を変更できるプロセッサを**ソフトコア・プロセッサ (Soft-Core Processor)** と呼ぶことにする．

Soft-Core Processor を用いた ASIC 設計手法は以下の通りである．(図 3)

step 1. Soft-Core Processor と基本ソフトウェア群を用いてアプリケーションソフトウェアを開発し，システムのプロトタイプを行ってシステムの仕様を固定する

step 2. システムの最適化目標にしたがってハードウェアとソフトウェアの境界を変更し，ハードウェアとソフトウェアを再設計する

step 3. 性能評価を行う (性能の制約条件が満足できなければ step 2. に戻る)

Soft-Core Processor を用いた ASIC 設計手法では次の 3 つの要件が満足されなければならない．(図 4)

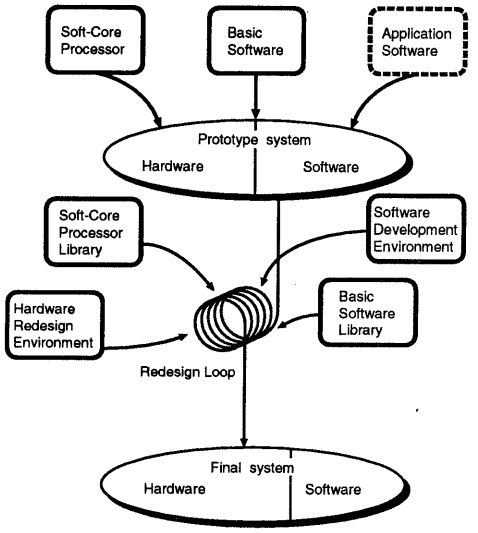


図 3: 本設計手法の流れ

1. Soft-Core Processor 全体ならびに各部分がソフトコア・モジュール・ライブラリとして用意されていること
2. 基本ソフトウェア・ライブラリが充実していること
3. ハードウェア／ソフトウェア協調設計環境が整備されていること

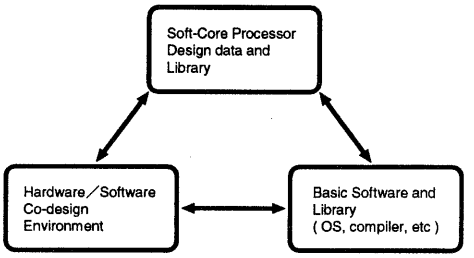


図 4: 本設計手法成立の鍵

以下本稿では 3 つの条件の中の Soft-Core Processor に対する要件とハードウェア／ソフトウェア協調設計環境に対する要件を議論する．

## 3.2 Soft-Core Processor の要件

Soft-Core Processor に関する要件について考察する．本手法による設計では，以下の点が重要である．

1. プロセッサの単体性能が高いこと
2. 広く利用されている言語や OS を利用可能であること
3. 設計変更が容易であること
4. 性能評価によりハードウェアとソフトウェアのバランスをとること

### 3.2.1 単体性能

Soft-Core Processor はハードウェア変更によりハードコア・プロセッサよりも高性能化をねらう．したがって，同機能のハードコア・プロセッサと同程度のクロック周波数，またはチップ面積でなくてはならない．Soft-Core Processor は HDL で記述されており，実際のチップになるには CAD ツールにより論理合成，レイアウト合成が必要となるので，Soft-Core Processor の性能はこれらのツールや設計ライブラリの性能によって左右される．

### 3.2.2 言語や OS のサポート

ソフトウェアの規模や複雑さはますます増大し，オペレーティングシステムや多くの大規模なアプリケーションはその保守だけでも極めて困難な作業となっている．システムの動作仕様を決定する段階でのソフトウェアを作成する場合に広く利用されている言語をサポートすることは，大規模なシステムの場合は非常に重要である．さらにハードウェアの変更にともないコンパイラや OS も柔軟に変更されることが要求される．また，各種関数ライブラリ等ハードウェアの変更に対応できる必要がある．

### 3.2.3 設計変更の容易さ

Soft-Core Processor を用いた設計手法では，ハードウェアやソフトウェアの設計変更が頻繁に起こる．したがって，設計変更を容易にするようなアーキテクチャが望まれる．ハードウェアの設計変更の際に注意しなければならない点を以下に挙げる．

a. ハードウェアは単純であること

b. 性能に対して大きな影響を与える部分から変更すること

c. 変更される部分によって他の部分ができるだけ変更されないこと

d. 全体が正しく動作するように一貫性を保って変更されること

e. ソフトウェアに対して複雑な動作を要求しないこと

f. 変更にともなってハードウェア性能が大きく低下しないこと

a. について，変更によって制御や構造がむやみに複雑になってしまい，次の変更や拡張を行うことが困難になってはならない．これは，検証の容易さや変更の拡張性，ハードウェアとソフトウェアのインターフェイスの複雑さ，CAD ツールの作り易さに影響を与える．

b. について，ヒューリスティックに変更して性能を評価するという手法をとると，性能に対してあまり影響を与えないような変更をする場合がある．したがって，変更する場合は，性能に対して大きな影響を与える部分から変更されなくてはならない．これは，システムの最適化や設計に要する期間に影響を与える．この点に関しては性能評価ツールによるサポートが必要である．

c. について，変更の際に他の部分をできるだけ変更しないで済むような局所的な変更でなければならない．これも設計に要する期間に影響を与える．局所的な変更を可能にするためにはモジュールをパラメタライズ化することも重要である．パラメタライズ化されたモジュールの組合せにより，最適なシステム構成を短期間で発見することができる．また，局所的な変更はハードウェア・モジュールの再利用という点からも必須である．

d. について，変更されるべき箇所が正しく変更されるようでなくてはならない．これは，検証に要する期間に影響を与える．設計変更の履歴を人

手で管理するのは，設計変更が頻繁に起こる場合は非常に困難である．したがって，ある設計変更によって影響を受ける箇所が複数箇所あってもこれらを一貫して変更できるようなCADのサポート(ドキュメンテーションや履歴管理)が必要である．また，これらの設計変更を容易にするような記述法の確立も必要である．

e.について，ソフトウェアで過度に特異な動作を強いることは，ソフトウェア設計者に余計な負担を与えることになる．これは，ソフトウェア(アプリケーションやコンパイラ，OS)の作成効率に影響を与える．

f.について，ハードウェアは規則性を持つ構造にしておく必要がある．規則性を持たせることにより，パラメタライズ化がしやすくなり，CADツールの作成が容易になる．

以上の点を考慮して，Soft-Core Processorに必要な特性をまとめると次の通りとなる．

- 単純で規則性を持った構造で設計変更が容易
- 性能に大きな影響を及ぼす局所的な設計変更が可能
- ソフトウェアの変更が容易

この特性を満たすものの一つにRISCプロセッサが挙げられる．

## 3.3 CADの要件

Soft-Core Processorを用いたASIC設計手法では再設計のために設計環境をCADでサポートする必要がある．

CADでサポートが必要とされるのは以下の項目についてである．

- 再設計支援
- 性能評価支援
- 検証支援
- ドキュメンテーション合成支援

### 3.3.1 再設計支援

Soft-Core Processorを用いた設計手法では，ハードウェアやソフトウェアの設計変更が頻繁に起こる．したがって，設計変更を容易にするような開発環境の整備が必須である．設計変更をより迅速に行い，かつ設計の質を向上させるために，設計ライブラリを充実させる必要がある．この場合の設計ライブラリとはHDLによって記述されたソフトコア・モジュール・ライブラリだけでなく，設計階層のあらゆる点での設計情報であり，ハードウェアの機能を置換できるソフトウェアのライブラリをも含む．これらの設計ライブラリを効果的に組み合わせることにより，設計の質の向上を図ることができる．また，このような設計ライブラリをシステム設計毎に設計・開発していたのでは設計期間の短縮を図ることができない．したがって，設計ライブラリを自動合成するようなモジュール・ジェネレータが開発される必要がある．また，このような高性能モジュールは簡単に利用できるようにパラメタライズ化し，高性能モジュール・ライブラリとして整備しておく必要がある．

### 3.3.2 性能評価支援

性能評価ツールはシステムの再設計時の指針となる．性能評価の精度が悪くては最適なシステム構成を得ることができない．また，性能評価により性能を低下させているボトルネックの発見ができなくてはならない．この点に関して，コンパイラの自動生成による性能評価支援の研究[5]が行われており近い将来に成果が上がることが期待される．

### 3.3.3 検証支援

設計変更が頻繁に起こる場合，設計変更によりシステムの動作が不正になってはならない．したがって，システムが正しさを保っていることを保証または確認する検証ツールが必要となる．

### 3.3.4 ドキュメンテーション合成支援

ドキュメンテーション系に要求されることは，一貫性の保持である．特に設計変更によるシステムの動作の正しさの保持と，設計変更による設計者間の意思の統一である．

一部の変更により影響を受ける部分が多くある場合，大規模なシステム設計ではその全ての変更点を掌握し，変更することはもはや人手では不可能である．したがって，これらの接続情報や変更情報をCADで管理する必要がある．

また，一般にソフトウェアの分野で言われていることであるが，プログラムが他人にも分かり易いということは，移植性の点で非常に重要なことである．設計のドキュメンテーションを管理し，最新の情報を各設計者に伝達することは，多人数が設計に関わり，かつ設計変更が頻繁に起こる場合には，必要である．

## 4 実験

前述のようなSoft-Core Processorに必要な特性を満たし，かつ設計データが公開されているアーキテクチャにDLXプロセッサ[6]がある．

ここでは，QP-DLX[7]をSoft-Core Processorに用いて本設計手法の実験を行う．今回の実験では，パラメタライズ化できる設計変更を特に意識して，設計変更は以下のことを行った．

- レジスタ数の変更
- 付加演算モジュールの変更

レジスタ数は8,16,32と変化させた．付加演算モジュールは乗算器のインプリメントを変化させた．ハードウェア合成系と実行シミュレーションにはPARTHENON[8]を使用し，ハードウェア量と実行ステップ数を得た．DLX用のコンパイラdlxccのマシン記述ファイル命令用のヘッダファイルを修正してレジスタ数と付加モジュールを変更したコンパイラをそれぞれ作成し，各種プログラムをコンパイルした．

## 5 結果

各種のCプログラムをコンパイルして，使用しているレジスタ数を測定した．表2にその結果を示す．

表2: レジスタ使用数

| プログラム | レジスタ数 | レジスタ使用数 |
|---|---|---|
| dhrystone.c | 32 | 11 |
| | 16 | 11 |
| | 8 | 8 |
| whetstone.c | 32 | 14 |
| | 16 | 14 |
| | 8 | 8 |
| spice | 32 | 8～32 (平均12.1) |
| | 16 | 8～16 (平均10.9) |
| | 8 | 8 |
| tex | 32 | 8～17 (平均10.3) |
| | 16 | 8～15 (平均10.1) |
| | 8 | 8 |
| BMS.c | 32 | 9 |
| | 16 | 9 |
| | 8 | 8 |
| FFT.c | 32 | 22 |
| | 16 | 16 |
| | 8 | 8 |
| Knapsack.c | 32 | 9 |
| | 16 | 9 |
| | 8 | 8 |
| bublesort.c | 32 | 9 |
| | 16 | 9 |
| | 8 | 8 |
| quicksort.c | 32 | 9 |
| | 16 | 9 |
| | 8 | 8 |

上の結果より，プログラムにより使用するレジスタ数が異なることがわかる．ゆえに，特定用途向けに最適化したプロセッサでは，使用するアプリケーションにしたがって，レジスタ数の最適化を行うことができる．例えば，texを主に使用するような場合，レジスタ数は16で性能はほとんど低下しない．

次にレジスタ数と付加ハードウェアを変更したHDL記述をそれぞれ合成し，ハードウェア量の変化を測定した．変更による実行時間を調べるため，乗算ルーチンを呼び出す簡単なプログラムを用意し，乗算命令を使用しないソフトウェアルーチンで実行した場合と乗算命令で実行した場合との実行ステップ数を比較した．表3にその結果を示す．

表3: ハードウェアの変更と実行時間

| ハードウェア構成 | †ゲート数 | ‡実行ステップ |
|---|---|---|
| 32, 乗算器なし | 27430.5 | 933 |
| 16, 乗算器なし | 19895.8 | 936 |
| 8, 乗算器なし | 16073.5 | 933 |
| 32, 乗算器あり | 33697.2 | 83 |
| 16, 乗算器あり | 26212.2 | 86 |
| 8, 乗算器あり | 22395 | 83 |

† ゲート数は2入力NAND換算
‡ 実行ステップはクロックサイクル数

## 6 おわりに

Soft-Core Processorを用いてASICを設計する手法を提案し，簡単な例を用いて実験環境を設定し，実験を行った．

今後，本手法の実現のために開発環境の整備をさらに進め，Soft-Core Processorに必要な機能を熟慮して，本手法による設計環境のプロトタイプを固めていきたい．

## 謝辞

日頃ご討論頂く九州大学 大学院総合理工学研究科村上和彰講師，および安浦研究室の諸氏に感謝致します．

また，ご討論頂いた三菱電気(株)のコデザイングループの皆様に感謝致します．

本研究は一部，京都高度技術研究所の新産学交流事業EAGLの支援による．

## 参考文献

[1] 中村，安浦，“専用システムのためのHW/SW協調再設計の一手法，” 情処研報，ARC98-13, pp.97-104, 1992.

[2] Rajesh K.Gupta, et al. “Synthesis and Simulation of Digital Systems Containing Interacting Hardware and Software Components,” in Proc. of 29th Design Automation Conference, pp.225-230, June 1992.

[3] Iksoo Pyo, et al. “High Level Synthesis of Pipelined Instruction Set Processors and Back-End Compilers,” in Proc. of 29th Design Automation Conference, pp.135-140, June 1992.

[4] 佐藤，他，“ASIP用ハードウェア/ソフトウェア・コデザインシステムPEASの実現とその評価，” 情処研報，DA64-11, pp.79-86, 1992.

[5] 赤星，安浦，“アーキテクチャ評価用ワークベンチ -コンパイラの自動生成-” 情処研報，ARC98-14, pp.113-120, 1992.

[6] John L Hennessy and David A Patterson, “COMPUTER ARCHITECTURE A QUANTITATIVE APPROACH,” Morgan Kaufmann Publishers, Inc., 1990.
富田，村上，新實(訳)，ヘネシー&パターソン コンピュータ・アーキテクチャ— 設計・実現・評価の定量的アプローチ —，日経BP社,1992.

[7] 岩井原，他，“計算機工学一貫教育用マイクロプロセッサQP-DLXの開発，” 情処研報，ARC100-5, pp.35-42, 1993.

[8] “PARTHENON User's Manual，” NTTデータ通信株式会社，1989.

[9] “インサイドGNU Cコンパイラ” フリー・ソフトウェアファンデーション編著，岩谷宏/都田克郎共訳，啓学出版，1991.