Vol. 36 No. 7　　情　報　処　理　　July 1995

解　説

ハードウェア/ソフトウェア・コデザイン

2. ハードウェアの見積りと生成†

今　井　正　治††

1. はじめに

ハードウェア/ソフトウェア・コデザイン (Hardware/Software Codesign) は，ハードウェアとソフトウェアのトレードオフを考慮して最適なシステムを設計する手法である．以下では，「ハードウェア/ソフトウェア・コデザイン」を．単に「コデザイン」と呼ぶ．

さて，コデザインの対象となっている“システム”という言葉は，しばしば曖昧さを持って使われている．本稿では，“システム”を“ハードウェア（電子回路）およびソフトウェアから構成される複雑な装置”と定義する．“システム”の例として，**図-1** に示すような電子装置を考える．同図のアーキテクチャで実現できるシステムの例としては，ロボティックス制御や通信制御などで用いられる，いわゆる組込み型システム（embedded system）がある[1)~4)]．

図-1 に示すシステムの最適設計を自動的に行う一般的な手法は，現在のところ明らかにされていない．これまでは，システム・アーキテクトが自らの経験と直観に基づいてシステム設計を行ってきたと言っても言い過ぎではないであろう．コデザインは，このようなシステムの最適設計を効率よく支援する手法として期待されている．

本稿では，コデザイン手法の中で，ハードウェアの見積りと生成技術の現状と今後の課題について解説する．本稿の構成は次のとおりである．まず **2.** では，コデザイン環境を構築するために必要な要素技術および最適化問題を分類し，理想的なコデザイン環境を提案する．次に **3.** では，専用ハードウェアのコストと性能の見積り方法，および生成方法について解説する．**4.** では，CPU コアのコストと性能の見積り方法，および生成方法について解説する．**5.** では，特定用途向き集積化プロセッサコアのコデザインシステム PEAS-I を紹介する．最後に **6.** で今後の課題について述べる．

2. コデザイン手法の概要

2.1 理想的なコデザイン環境

図-1 のシステムは次のような構成要素を含んでいる．

(1) CPU コア
(2) 専用周辺回路
(3) メ　モ　リ
(4) 入出力ポート

本稿では，これらの構成要素のうち(1)および(2)を中心に解説する．このようなシステムのコデザインを行うためには，次の要素技術を確立することが必要である．

- ハードウェアとソフトウェアを含むシステム全体をすばやくモデル化し，機能の検証を行う方法（rapid prototyping, cosimulation）
- ハードウェア・コスト，性能，消費電力を精確に見積る方法（accurate estimation）
- ハードウェアとソフトウェアのトレードオフを考慮した最適な機能分割を行う方法（optimal partitioning）
- ハードウェアおよびソフトウェアの記述を自動生成する方法（cosynthesis）

ここでのハードウェアの生成とは，VHDL や Verlig HDL などのハードウェア記述言語（HDL： Hardware Description Language）による記述の生成を意味する．

図-1 に示したシステムの最適設計を行うための“理想的な”コデザイン環境としては，たとえ

† Estimation and Synthesis of Hardware by Masaharu IMAI (Department of Information and Computer Sciences, Toyohashi University of Technology).
†† 豊橋技術科学大学情報工学系

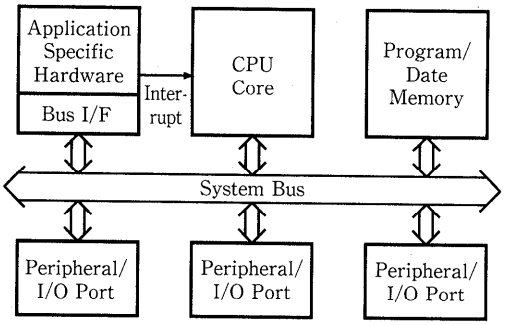

図-1 “システム”のモデル

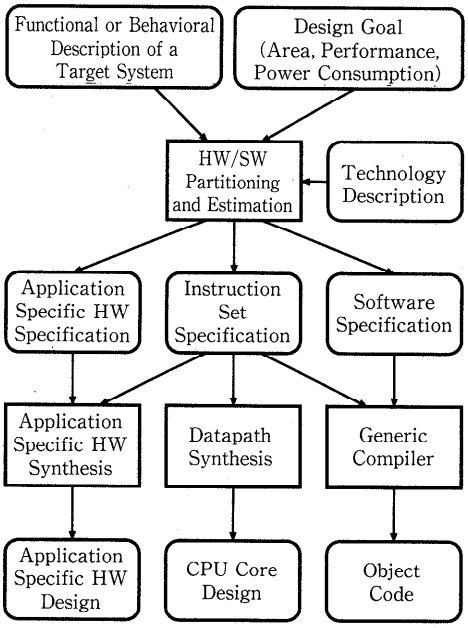

図-2 “理想的な”コデザイン環境

ば図-2のような構成が考えられる．同図の設計環境への入力は次の3種類の情報である．

(1) ターゲット・システムの記述（動作または機能レベル）
(2) 設計目標（性能，コスト，消費電力）
(3) 実装に使用する製造技術の記述（各基本セルおよび基本モジュールの性能，コストなど）

また，この設計環境からの出力は次の3種類の記述である．

(1) CPUコアのHDL記述
(2) 専用周辺回路のHDL記述
(3) CPUコア上で実行される応用プログラムのオブジェクト・コード

図-2の環境は，次のような機能を持っている．

(1) ターゲット・システムの記述中に現れるそれぞれの機能をハードウェアまたはソフトウェアで実現する場合の性能，コスト，消費電力など見積りを行う．
(2) (1)の見積りに基づいてそれぞれの機能の実現方法（HWまたはSW）の組合せの中で最適な実現方法を選択する．
(3) (2)で得られた実現方法の組合せに基づいて機能分割を行い，HWおよびSWの記述を生成する．

表-1 システム最適化問題の分類

タイプ	設計目標	制約条件	
HP	性能	HWコスト	消費電力
LC	HWコスト	性能	消費電力
LP	消費電力	性能	HWコスト

2.2 システムの最適化

設計対象となるシステムの最適化を行う場合の主要な評価項目としては，**性能**，**ハードウェア・コスト**，**消費電力**の3種類がある．

これらの項目のうち，**性能**は，応用プログラム（サンプルプログラム）の実行時間，すなわち，「基本クロック・レート(ns)×実行サイクル数」で評価される．また，**ハードウェア・コスト**はシステムの実装方法に応じて，チップ面積，ゲート数，基本セル数，などによって評価される．システムの最適化問題は，これらの評価項目の組合せによって，**表-1**に示すように3種類に分類できる．

(1) **タイプHP—性能の最大化**

ハードウェア・コストおよび消費電力の制約条件のもとで，性能が最大になるようなアーキテクチャを決定する．

(2) **タイプLC—ハードウェア・コストの最小化**

性能および消費電力の制約条件のもとで，ハードウェア・コストが最小になるようなアーキテクチャを決定する．

(3) **タイプLP—消費電力の最小化**

ハードウェア・コストおよび性能の制約条件のもとで，消費電力が最小になるようなアーキテクチャを決定する．

従来のシステムの最適化設計では，主として

性能最大化問題（タイプHP）が取り扱われてきた．しかし，実時間制御システムなどでは，ハードウェア・コスト最小化問題（タイプLC）を取り扱う必要がある．さらに，移動体通信などへの応用では，消費電力最小化問題（タイプLP）を取り扱う必要がある．

2.3 コデザイン手法の要素技術

これまでのコデザインの研究は次の二つの方向から検討されてきた．

(1) 専用周辺回路のコデザイン[1)~6)]

(2) 専用CPUコアのコデザイン[7)~11)]

ただし，(1)の研究と(2)の研究は設計対象が異なるだけであり，システム全体すなわち周辺回路やメモリを含めたシステムの最適設計を行うためには相補的な関係にある．したがって，理想的なコデザイン・システムを実現するためには，両者のアプローチを統合化した，新しい手法が必要である．

3. 専用周辺回路の見積りと生成

専用周辺回路のコストおよび性能の見積りを行うためには，**論理合成**および**高位合成**[12)]の研究成果が利用できる．たとえば，Thomasらが提案しているコデザイン・システム[5)]では，以下のようにしてハードウェアとソフトウェアへの機能分割を行っている．このシステムへの入力は，HDLおよびソフトウェア・プログラミング言語（SPL）で記述された，システムの動作記述である．

(1) ハードウェアで実現すべき機能をHDLで記述し，ハードウェアもしくはソフトウェアで実現される機能はSPLで記述して，システムに入力する．

(2) 次に，SPLで記述されたタスク（ソフトウェア）とHDLで記述されたハードウェアのコシミュレーション（cosimulation）を行い，各タスクの実行時間および並列実行可能性を調べる．

(3) 上記の結果に基づいて，実行時間が長く性能面でのボトルネックになっているタスクをいくつか選択する．次にこれらのタスクをHDLで書き直し論理合成を行い，ハードウェアで実現した場合の回路規模と実行時間を調べる．

(4) 次に，設計条件を考慮してハードウェアで実現すべきタスクの候補を選択する（これはデザイナが行う）．

(5) 最後に，ハードウェア部分およびソフトウェア部分それぞれの生成（cosynthesis）を行う．

このシステムで，SPLで記述されていたタスクをハードウェアで実装する場合には，生成されるハードウェア・モジュールにバスインタフェース回路を追加し，ハードウェア・モジュールの動作が終了した場合に，CPUに割込みをかけることによって，タスク間の同期を実現している．

上記の論文[5)]では，SPLで記述された機能をハードウェアで実現する具体的な方法については述べられていない．ハードウェアでの最適な実装方法を選択するためには，同一の機能の複数の実装方法の間のトレードオフを考える必要がある．これは，高位合成で研究されてきた，**リソース制約スケジューリング問題**のアルゴリズム[12)]および**時間制約スケジューリング問題**のアルゴリズム[12)]を適用することによってある程度解決できると考えられる．

4. 専用CPUコアの見積りと生成

特定の応用プログラムまたは特定の応用分野で用いられるプログラムの実行に適したCPUコアを自動生成するためには，以下の要素技術を確立する必要がある．

(1) **アーキテクチャの最適化**

与えられた応用プログラムとデータを解析し，その応用に対して最適なアーキテクチャを決定する技術が必要である．そのためには，与えられた応用プログラムとデータに適したCPUコアをすばやくモデル化し，そのCPUの性能，ハードウェア・コスト，消費電力などをできるだけ精確に見積もる技術が必要である．

(2) **ハードウェア記述の自動生成**

決定されたアーキテクチャのハードウェア記述を自動生成する技術が必要である．CPUの基本モジュールをあらかじめ系統的に用意しておけば，データパスの自動生成はさほど困難ではない．残る問題は制御パスの自動生成であるが，この問題は，これまでの論理合成や高位合成技術を拡張することにより解決可能であると期待される．また，専用周辺回路を含むシステ

ムの記述を生成するためには，インタフェース回路の自動生成が必要であるが，この部分は標準化が可能であろう．

(3) **システム・ソフトウェアの自動生成**

与えられたアーキテクチャに対して，対応するコンパイラ，シミュレータ，ハードウェアとのインタフェース（デバイス・ドライバ）などのソフトウェアを自動生成する技術が必要である．

5. 事例紹介： PEAS-Iシステム

CPUコアの最適化問題に対する一般的な解決方法は今後の研究成果を待つ必要があるが，CPUアーキテクチャのモデルに制約を加えれば，比較的容易に最適化が行えるという報告がある[7),10)]．この章では，特定用途向き集積化プロセッサ（ASIP）のコデザイン・システムである，PEAS-I[7)]を紹介する．

5.1 PEAS-Iシステムの概要

PEAS-Iシステムは，与えられた応用プログラムの解析結果に基づいて最適な命令セットを持つCPUコアのHDL記述を生成する．また，これと同時に，コンパイラおよび命令レベルシミュレータなどの応用プログラム開発ツールも同時に生成する．

何種類かのサンプル・プログラムを用いて，PEAS-Iシステムで生成されるCPUコアの評価が行われた．その結果，PEAS-Iシステムで生成されるCPUコアのハードウェア・コスト（ゲート数）の見積り誤差は，最大で数%程度であり，性能（実行サイクル数）の見積り誤差も最大で数%程度であったと報告されている[13)]．また，生成されたCPUコアを商用の汎用マイクロ・プロセッサと比較した結果，PEAS-Iシステムで生成されるCPUコアは，これらの汎用マイクロプロセッサよりもかなり少ないゲート数（ほぼ数万ゲート）で実装され，しかも命令パイプライン方式を採用している商用の汎用マイクロ・プロセッサを上回る性能を持つと報告されている[14)]．

5.2 システムの機能

PEAS-Iシステムへの入力は，次の4つである．

(1) C言語で記述された応用プログラム（サンプル・プログラム）

(2) (1)の応用プログラムで使用されるデータ

(3) 設計条件

(4) システムの実装に用いるライブラリ（モジュール・データベース）

現在のPEAS-Iシステムで取り扱える最適化問題は，性能最大化問題（タイプHP）およびハードウェア・コスト最小化問題（タイプLC）の2種類である．消費電力最小化問題に関しても現在研究が進められている．

PEAS-Iシステムが生成するCPUコアのアーキテクチャは，GNU Cコンパイラ（GCC）の抽象マシンモデル[15)]に基づいている．このモデルを採用することによりコンパイラや命令レベルシミュレータの自動生成が容易に行える．PEAS-Iで生成されるCPUコアは，以下の特徴（制約）を持っている．

(1) 内部バス幅は32ビットである．

(2) 汎用レジスタ方式（レジスタ数は可変）である．

(3) 4段の命令パイプラインを持つ．

(4) メモリへのアクセスはLoad/Store命令だけで行う．

(5) 演算器構成が可変である．

PEAS-Iシステムで生成されるCPUコアに内蔵できる演算器は，乗算器，除算器，バレルシフタ，データ型変換器（符号拡張など）である．これらの演算器は，あらかじめHDLで記述してあり，CPUの実装に用いる基本セルライブラリごとに，論理合成結果（ゲート数，実行サイクル数，消費電力）がモジュール・データベースの中に格納されている．多くの演算器には，演算の種類ごとにハードウェア・アルゴリズムが異なる複数の種類のモジュールが用意されている．これらのモジュールは，同一の機能を持つがゲート数，実行サイクル数などが異なっている．命令セットアーキテクチャの最適化は，各命令の実現方法の組合せを変更することで行われる．

5.3 システムの構成

PEAS-Iシステムは，**図-3**に示すように以下の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（APA）

与えられた応用プログラムを静的および動的に解析し，各基本演算の実行頻度などを解析

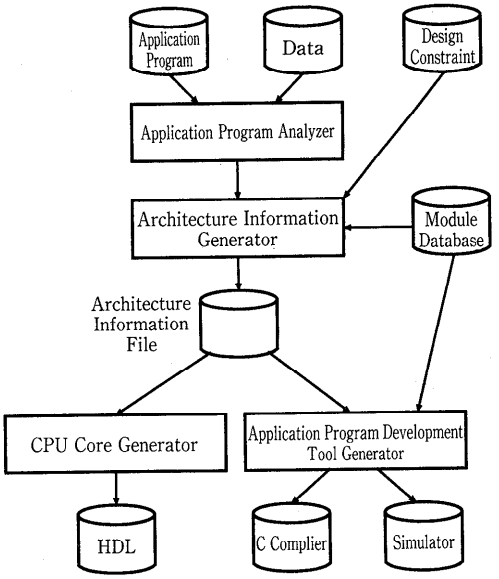

図-3 PEAS-Ⅰシステムの構成

する．

(2) アーキテクチャ情報生成系（AIG）

応用プログラム解析系で得られた情報に基づいて，与えられた応用プログラムの実行に最適な命令セットを持つCPUアーキテクチャを選択する．

(3) CPUコア記述生成系（CCG）

AIGで決定されたアーキテクチャ情報に基づいて，CPUコアのHDL記述を自動生成する．

(4) 応用プログラム開発ツール生成系（DTG）

AIGで決定されたアーキテクチャ情報に基づいて，コンパイラ，アセンブラ，命令レベルシミュレータのC言語での記述を自動生成する．

6. おわりに

本稿では，ハードウェア/ソフトウェア・コデザインの現状について述べた．これまでの研究では，専用周辺回路のコデザインと専用CPUコアのコデザインのそれぞれが比較的独立して進められてきた．しかし，応用分野の性質は各種各様であり，どちらか一方の構成要素の最適化だけではシステム全体の最適化を達成するのは困難であると考えられる．したがって今後は，これら両者の構成要素を同時に最適化する新しい手法が必要になると思われる．本稿では触れなかったが，消費電力最小化問題を解くためには，応用プログラムの実行に必要な消費電力の精密な見積り方法を確立する必要がある．これらはいずれも今後の課題である．

謝辞 本稿を執筆するにあたり，資料の整理および図表の作成を手伝っていただいた，本学大学院博士課程の本間啓道君，Nguyen Ngoc Binh君，および修士課程の櫻井涼二君（現在シャープ(株)）に深謝します．また日頃，アーキテクチャおよび設計自動化について討論していただく，(株)SRAの引地信之氏，鶴岡高専の佐藤淳講師，本学の塩見彰睦助手，ならびに本学VLSI設計研究室の学生諸君に感謝します．

参考文献

1) Kalavade, A. and Lee, E. A.: A Hardware Software Codesign Methodology for DSP Application, IEEE, Design & Test of Computers, Vol. 10, No. 3, pp. 16-28 (Sep. 1993).
2) Ernst, R., Henkel, J. and Benner, T.: Hardware-Software Consynthesis for Microcontrollers, IEEE, Design & Test of Computers, Vol. 10, No. 4, pp. 64-75 (Dec. 1993).
3) Smailagic, A. and Siewiorek, D. P.: A Case Study in Embedded - System Design: The VuMan 2 Wearable Computer, IEEE, Design & Test of Computers, Vol. 10, No. 3, pp. 56-67 (Sep. 1993).
4) Gupta, R. K., Coelho, C. N. and De Micheli, G.: Program Implementation Schemes for Hardware-Software Systems, IEEE,Computer, Vol. 27, No. 1, pp. 48-56 (Jan. 1994).
5) Thomas, D. E., Adams, J. K. and Schmit, H.: A Model and Methodology for Hardware-Software Codesign, IEEE, Design & Test of Computers, Vol. 10, No. 3, pp. 6-15 (Sep. 1993).
6) Kumar, S., Aylor, J. H., Johnson, B. W. and Wulf, W. A.: A Framework for Hardware/Software Codesign, IEEE, Computer, Vol. 26, No. 12, pp. 39-45 (Dec. 1994).
7) 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, Trans. IEICE, Vol. E77-A, No. 3, pp. 483-491 (Mar. 1994).
8) Akaboshi, H. and Yasuura, H.: COACH: A Computer Aided Design Tool for Computer Architects, Trans. IEICE, Vol. E76-A, No. 10, pp. 1760-1769 (Oct. 1993).
9) Salinas, A. H., Johnson, B. W. and Aylor, J. H.: Implementation-Independent Model of an

Instruction Set Architecture in VHDL, IEEE, Design & Test of Computers, Vol. 10, No. 3, pp. 42-55 (Sep. 1993).
10) Huang, I-J. and Despain, A. M.: Synthesis of Instruction Sets for Pipelined Microprocessors, Proc. of Design Automation Conference '94, pp. 5-11 (1994).
11) Wilberg, J., Camposano, R. and Rosenstiel, W.: Design Flow for Hardware/Software Cosynthesis of a Video Compression System, Proc. of Codes/CASHE '94 (1994).
12) Gajski, D. D., Dutt, N. D., Wu, A. C-H. and Lin, S. Y-L.: High-Level Synthesis, Introduction to Chip and System Design, Kluwer Academic Publishers (1992).
13) Binh, N. N., Imai, M., Shiomi, A. and Hikichi, N.: A Hardware/Software Codesign Method for Pipelined Instruction Set Processor using Adaptive Database, Submitted to ASP-DAC '95 (Aug. 1995).
14) Sato, J., Hikichi, N., Shiomi, A. and Imai, M.: Effectiveness of a HW/SW Codesign System PEAS- I in the CPU Core Design, Proc. of APCHDL '94, pp. 259-262 (Oct. 1994).
15) Stallman, R.: Using and Porting GNU CC, Version 2.5, Free Software Foundation, Inc. (1993).

(平成7年5月13日受付)

今井 正治(正会員)

1974年名古屋大学工学部電気卒業. 1979年同大学院博士課程(情報工学専攻)修了, 工学博士(情報工学). 1979年より豊橋技科大・情報工学系勤務. 現在, 同教授. 1984年より85年にかけて米国サウスカロライナ大学に文部省在外研究員(客員助教授)として滞在し教鞭をとる. 1991年より日本電子機械工業会(EIAJ)およびIEEEの設計自動化標準化委員会(DASC)において, ハードウェア記述言語VHDLの標準化に従事. これまで, 組合せ最適化問題, 計算機アーキテクチャ, VLSIの設計自動化, ハードウェア/ソフトウェア・コデザインなどの研究に従事. IEEE, ACM, 電子情報通信学会, 人工知能学会, コンピュータ支援画像診断学会各会員.