計算機アーキテクチャ　98－21
設　計　自　動　化　65－21
（1993. 1. 22）

パネル討論：アーキテクチャ設計にCADは有効か？

遠藤忠男[1)]，大鶴祥介[2)]，小栗 清[3)]，若林一敏[4)]
司会：安浦寛人[5)]

1)物産システムテクノロジー　〒９６３郡山市朝日１－２８－１２グレイスコート朝日
2)(株)ＹＨＰシステム技術研究所　〒８３０久留米市合川町２４３２
3)ＮＴＴコミュニケーション科学研究所　〒６１９－０２京都府相楽郡精華町光台２丁目
4)ＮＥＣ Ｃ＆Ｃシステム研究所　〒２１３川崎市宮前区宮崎４－１－１
5)九州大学大学院総合理工学研究科　春日市春日公園６－１

あらまし

近年のハードウェア記述言語，論理合成技術，レイアウト合成技術などのCAD技術の進歩により，ソフトウェアにおける高級言語によるプログラミング程度のレベルでハードウェア設計を行なって，あとは自動的に最終的な回路を生成できるようになった．このような状況を踏まえて，新たに計算機アーキテクチャレベルの設計を支援するCADシステムやそれらを用いた設計手法の提案が数多くなされている．本パネル討論では，種々のアーキテクチャ設計用ツールの開発者にそのツールの基本思想，機能，利用例などを紹介してもらい，アーキテクチャ設計におけるCADツールの在り方について議論する．

和文キーワード　計算機アーキテクチャ，CADツール，高位合成，論理合成

Panel: Is CAD Useful in Computer Architecture Design?

Tadao Endo[1)], Yoshisuke Ohtsuru[2)],
Kiyoshi Oguri[3)] and Kazutoshi Wakabayashi[4)]
Coodinator: Hiroto Yasuura[5)]

1) Bussan Electronic Systems Technology, Inc. Kooriyama, Fukushima 963
2) YHP Design Systems Laboratory Kurume, Fukuoka 830
3) NTT Communication Science Laboratories, Seika-cho, Kyoto 619-02
4) NEC C&C Systems Research Laboratories, Kawasakai, Kanagawa 213
5) Department of Information Systems, Kasuga, Fukuoka 816

Abstract

Recent progress of CAD technology in hardware design languages, logic synthesis and layout synthesis makes it possible to generate a circuit automatically from a high level design description as a program in programming language for software. Many design methodologies and CAD systems for designning computer architectures using these CAD technologies have been propopsed. In this panel discussion, we will argue new directions of CAD for computer architects based on the proposals from CAD system designers of four advanced CAD tools.

英文 key words　Computer Architecture, CAD Tools, High-Level Synthesis, Logic Synthesis

アーキテクチャ設計とCAD

安浦寛人
九州大学大学院総合理工学研究科

近年のハードウェア記述言語，論理合成技術，レイアウト合成技術などのCAD技術の進歩により，ソフトウェアにおける高級言語によるプログラミング程度のレベルでハードウェア設計を行なって，あとは自動的に最終的な回路を生成できるようになった．このような状況を踏まえて，新たに計算機アーキテクチャレベルの設計を支援するCADシステムやそれらを用いた設計手法の提案が数多くなされている．本パネル討論では，種々のアーキテクチャ設計用ツールの開発者にそのツールの基本思想，機能，利用例などを紹介してもらい，アーキテクチャ設計におけるCADツールの在り方について議論する．本パネル討論は，従来我が国において比較的交流が少なかった計算機アーキテクチャの分野の研究者とCAD分野の研究者の始めての合同研究会のなかで企画されたものであり，今後この両分野の交流が重要になるとの予想の元で，その将来の動向を議論するものである．

近年の論理合成技術の発達によって，CADをアーキテクチャ設計者が直接利用してより複雑なアーキテクチャを比較的容易に設計できる状況が生まれてきた．本パネルでは，CAD研究者の側からアーキテクチャ設計へのCADの利用の可能性を提示して，アーキテクチャ設計者への新しい設計手法の提案を行なうとともに，今後のより高いレベルのCADの在り方を探るのが狙いである．パネラーはすべてCAD側の方々にお願いし，CAD側で構想しているツールとそれを用いた設計手法をお話しいただき，フロアからアーキテクチャ設計者側からの質問や意見さらには今後のCADへの注文をいただこうというもくろみである．

パネラーは，物産システムテクノロジーの遠藤忠男氏，(株)ＹＨＰシステム技術研究所の大鶴祥介所長，ＮＴＴコミュニケーション科学研究所の小栗清氏，ＮＥＣＣ＆Ｃシステム研究所の若林一敏博士にお願いした．

遠藤氏には，米国Scientific and Engineering Software社で開発されたシステム設計用ツールをご紹介いただく．本ツールは，ソフトウェアとハードウェアの一体となったシステムをモデリングして性能を評価するツールである．

大鶴氏には，同社で開発されたApplication Orientedな上流設計ツール「つつじ」を紹介していただく．「つつじ」は，アーキテクチャレベルの設計と性能評価のための強力なグラフィックスヒューマンインタフェース機能をもち，論理合成によって自動的に回路を生成することが可能である．

小栗氏には，同氏のグループで長年開発を進められているパルテノンの経験をベースに今後のアーキテクチャ設計用のCADの動向をお話しいただく．パルテノンは，すでにNTTや大学などで広く利用されておりそれを用いた設計例も豊富である．これらの経験を元に，今後のアーキテクチャ設計用CADに対する提案が行なわれると思う．

若林氏には，計算機メーカーの社内に居られるCAD研究者の立場から，開発されたCyberシステムの経験を踏まえて，アーキテクチャ設計用のCADに必要な機能を整理して，今後の高位合成，ハードウェア記述言語，コンパイラ技術などの在り方を示唆していただく．

パネラーの方々にはパネルに先だって以下のような質問をお送りした．

1）開発されたCADツールの概要
2）そのツールを利用した設計手法（特にアーキテクチャ設計にかかわる部分）
3）設計事例
4）今後のツールの在り方と開発方針

アーキテクチャ設計用のCADの在り方，実用性などはまだまだ固まった議論ではない．本パネルを通して，アーキテクチャ設計者とCAD設計者の間の理解が深まり，今後の両分野の発展につながることを期待する．

「システム設計用ツールSES/workbench」

遠藤忠男（物産システムテクノロジー）

1. SES/workbenchの紹介

コンピュータなどのシステムを設計する場合、ハードウェアの設計またはソフトウェアの設計を行う前に、アーキテクチャ・レベルの設計／評価（システム設計）を行わなければならない。システムが複雑であればあるほど、それを効率的に行うツールが必要となる。アーキテクチャ・レベルの設計／評価のツールには、ハードウェアとソフトウェアを同時にモデリングし、性能の評価を行い、さまざまな選択肢の中から最良の方法を決定できる機能が不可欠である。

ここでは、これらのモデリングとシミュレーションに使用するシステム設計用ツールSES/workbenchを紹介する。

2. SES/workbenchの概要

2.1 SES/workbenchの設計手法

システムを設計する場合、アプリケーション（要求される処理／機能）だけでなく、システムに要求されるワークロード（処理の数量）、実行環境（ハードウェアの種類）、性能（処理速度）といった、システム設計に必要なすべての要因を考慮しなければならない。

SES/workbenchでは、システム設計をこれらの４つの観点からとらえ、システムのアーキテクチャ・レベルに注目して設計仕様の記述、モデリング、シミュレーション、および性能の評価を行う。

設計と評価は、トップダウンまたはボトムアップどちらでもアプローチできる。関心のあるレベルから開始し、そのサブモデルの設計と評価が目的を達成したら次の（上または下）のレベルの設計と評価を行う。これを繰り返してシステム全体の設計を完了させる。

2.2 モデリング

設計作業は、SES/workbenchモデルを作成することから開始する。SES/workbenchモデルは、24個のアイコンと、アイコンを結ぶアークを用いて作成する拡張有向グラフの階層で表現する。アイコンは、リソースの操作、トランザクションの生成と消滅、モデルの階層を表す。アークは、各ノードのトポロジを表し、これを介してトランザクションがシミュレーション時に移動する。

ここで、SES/workbenchのグラフの基本要素であるノード、リソース、アーク、およびトランザクションについて簡単に説明する。

ノードは、物理的または論理的なリソースの操作（メモリの割り当て／解放など）、またはトランザクション生存期間中の処理ステップを表現できる。たとえばコンピュータのモデルでは、プロセッサのスケジュールやディスク・ドライブなどを表現できる。ソフトウェアのモデルでは、サブプログラムやプロセス制御動作を表現できる。

リソースは、トランザクションが取り合う物理的または論理的な構成要素を表現する。プロセッサ、バス、ソフトウェア・スケジューラ、RAM、またはデータベース・ロック機構などがその例である。

アークは２つのノードを接続し、一方のノードから他方のノードへ向かう。これは、トランザクションが、流れる経路を示す。

トランザクションは、実行されるプロセス、処理または転送されるデータ、処理に必要な制御信号を表す。SES/workbenchモデルには、通常、並行に実行される多くのトランザクションが存在する。トランザクションはノードからノードへと流れ、データを運び、リソースを操作し、ステートメントを実行する。

各グラフの要素、すなわちノード、アーク、およびリソースには、それぞれの特性を規定するフォーム（書式）が付いている。これらのフォームはデフォルト値を持っているので、通常は、標準と異なる値についてだけ入力すればよい。SES/workbenchモデルのほとんどの部分で、C言語の宣言、式、ステートメント、および関数を自由に記述することができるので、SES/workbenchモデルの表現を自由に拡張することができる。

モデル作成の初期段階では、設計要素のハードウェア／ソフトウェアの区別を意識しなくてもよい。

2.3 シミュレーション

作成したグラフおよび仕様を実行可能なシミ

ュレーション・モデルに変換し、シミュレーションを行う。

SES/workbenchモデルは、多重処理を実行する並行プログラムとして考えることができる。モデル内の各トランザクションは、個々の実行スレッドを表す。これらの多数のトランザクション（実行スレッド）が、同時に並行して実行される。

シミュレーションに必要なイベントの発生は、トランザクションの注入によって行う。非同期なイベントの発生や確率事象も、乱数発生器と各種確率分布関数によってサポートされる。シミュレーションにはアニメーション機能がある。システムの動きをビジュアルに観察できるので、複雑なシステムの設計グループのメンバの直感的な「意志の疎通」、あるいはそのシステムの依頼主に対するプレゼンテーションに役立つ。

モデルのシミュレーションを通して、考案したアーキテクチャが目標の機能／性能を達成するかどうかを検証する。また、いろいろな設計のトレードオフが最終設計、つまり、実際のシステムにどのように影響を与えるかを観察する。

2.4 性能の評価

性能の評価は、シミュレーション時にレスポンスや利用度などの統計を収集することによって行う。統計の指定もアイコンやフォームを用いて行う。設計の正確さの評価は、各設計要素に表明を行うことによってシミュレーション時に判断する。

このようにして、実際にソフトウェアのコーディングや試作機を製造をする前に、問題点を明らかにして改善を行うことができる。モデル表現とデバッグ時の画面の例を下図に示す。

3. 設計事例

●ハードウェア：SPARC、MIPS、x86。バス・アービトレーション、キャッシュ、パイプライン、並列コンピューティング・プロセッサ。

●ソフトウェア：DBMS、NFS。

●通信：X.25、TCP/IP、Ethernet、FDDI。バス・コンテンション、通信プロトコル。

●システム：ディスク・サブシステム、ワークステーション、クライアント・サーバ情報システム

4. 今後の開発方針

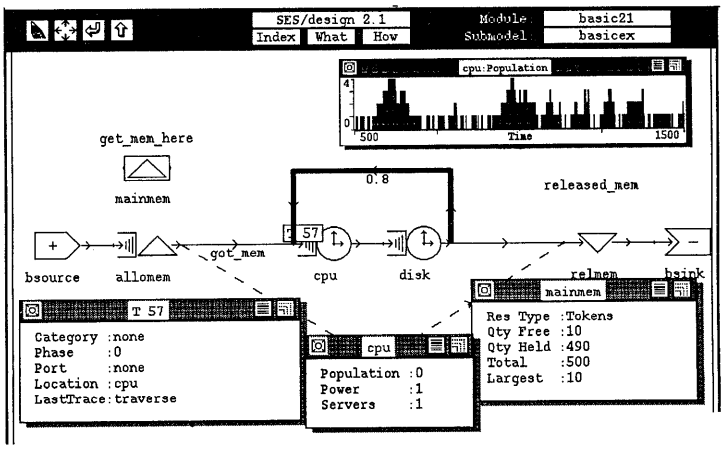
既存のCAEツールとのブリッジを充実させ、インプリメンテーションを支援する。

●VHDLコード・ジェネレータ

●コシミュレーション：Verilog、VHDL(Mentor)

●CASEツールとのインタフェース：StP、Teamwork。Yourdon/Demarcoのデータフロー・ダイアグラムにHatley/Pirhbaiのリアルタイム拡張。

●OOAツール：Shlaer-Mellerオブジェクト指向分析／設計手法のサポート。



CADシステムの上流設計への応用

大鶴祥介
株式会社YHPシステム技術研究所

近年、VHDL等の普及により、LSIおよびH/Wの設計記述の上位化が進んで来ているが、まだ、これらの用途はアルゴリズム等の研究開発後の実際の製品設計の分野がほとんどであり、この分野での生産性向上にしか結びついていない。
ここで、CADシステムが本当の意味の研究開発の効率化に利用されるためには、今後CADシステムがどの様なコンセプトを基盤とし、またどの様な機能を有しなければならないかを、当社で開発したCADシステム“つつじ”を通して述べる。

1．ツールの基本コンセプト

“つつじ”の基本コンセプトは「Application Oriented」である。より一般的な表現では、「目的指向」と言い替えることができる。

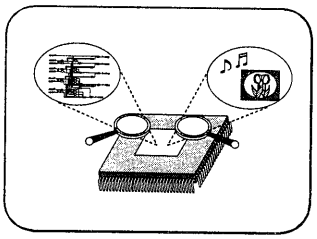


図1

“つつじ”が生成する最終的なアウトプット、は、デジタルLSIであるが、そのLSIの用途は様々である。ここで、そのデジタルLSIの中身を「論理ゲートの集積」と捉えるか、「Application」と捉えるかで設計のアプローチは異なる。（図1）

上流設計の意図するところは後者であり、最終生成物が同じでも上流設計でのアプローチはそれぞれのApplicationによって千差万別である。そのアプローチを標準化や画一化することは困難である。（図2）

つまり、設計が上流に行けば行くほどそれぞれのApplicationに特化した設計・評価環境を準備する必要がある。この設計アプローチが、「Application Oriented」のコンセプトである。

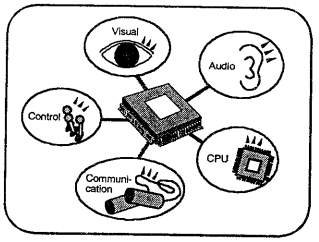


図2

2．“つつじ”の概要

先に筆者は、「設計が上流に行けば行くほどそれぞれのApplicationに特化した設計・評価環境を準備する必要がある。」と述べた。これは、「CADシステムには、Application毎に柔軟に設計・評価環境を構築可能な能力が必要である。」とも言える。この意味から単にツールとしての“つつじ”は、半完成システムである。この“つつじ”に実際のApplicationの環境を取り込むことにより、上流設計を支援するCADシステムとして機能する。

“つつじ”の開発に際して、以下の要件定義を行った。これらは全て設計者の立場から定義されたものである。

◇設計者が機能を記述しやすい手法の提供
- ◆アルゴリズムの記述手法として、C言語をサポート
- ◆LSI化を前提とした記述手法としてブロック図、HDL（VHDLを含む）をサポート

◇簡潔で使い易い操作体系
- ◆CADの経験があれば、１日以内の操作法の体得

◇設計と評価・検証の動的なリンケージによる思考の連続性を支援
- ◆設計環境から評価・検証環境へのスムーズな移行
- ◆超高速シミュレーションの実現

◇柔軟な評価・検証環境の構築
- ◆仮想計測器による評価・検証環境の提供
- ◆Applicationに特化した評価・検証ツールとの柔軟なインターフェイス
- ◆ユーザ定義モデル（C言語、ブロック図、HDL（VHDLを含む）等で記述）との柔軟なインターフェイス

なお、ブロック図、HDLを使用して機能を記述した場合、モジュール合成、論理合成のテクノロジによりそのままLSI化することが可能である（ゲート・サイズ、動作速度の評価は別途必要）。

3．“つつじ”を用いた上流設計の手法

筆者は、“つつじ”を用いた設計手法として、「Rapid Prototyping」を提唱している。このRapid Prototypingとは、開発の初期段階における仕様決定まで含めた設計手法である。今後、設計の上流化が進むと、ツールの供給者は単に設計記述の手法だけを提供するだけでは不十分である。設計しようとしている、または設計したシステムの評価・検証の環境を併せて提供することが必要である。

ここで言う評価・検証の環境とは、単なる論理シミュレーションでは無意味である。当然ながらここにもApplication Orientedの評価・検証の手法が必要となる。

“つつじ”による評価・検証環境のサンプルを以下に示す。（図3）

◇Rapid Prototypingの目的
- ◆システムとしての全体の整合性の評価
- ◆アルゴリズム、アーキテクチャの迅速な評価
- ◆複数のPrototypeを評価・検討し、最も満足できるアーキテクチャの選択

◇“つつじ”のRapid Prototyping機能
- ◆C言語、ブロック図、HDL（VHDLを含む）によるシステム全体のモデル化
- ◆システム全体のシミュレーション機能
- ◆Application Domainによる直感的な評価
- ◆LSI化した時のゲート・サイズおよび動作速度の予測

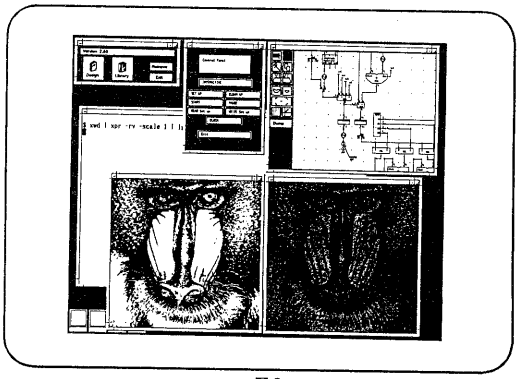


図3

4．設計事例

ここでは、“つつじ”による「誤差拡散法による画像の2値化」[1]を行なう回路の設計例を紹介する。

■誤差拡散法による画像２値化アルゴリズム

$$f'_{mn} = f_{mn} + \frac{1}{\sum \alpha_{kl}} \sum \alpha_{kl} e_{m-k,n-1}$$

$$g_{mn} = \begin{cases} 1 & f'_{mn} \geq T \\ 0 & f'_{mn} < T \end{cases}$$

$$e_{mn} = f'_{mn} - g_{mn}$$

f_{mn} 原データ
f'_{mn} 補正値
g_{mn} 2値化データ
α_{kl} 重み付け係数
e_{mn} 2値化誤差
T 閾値

■Rapid Prototypingによる2値化アルゴリズムの評価・検証

数式をC言語で記述する。サンプルを以下に示す。（図４）

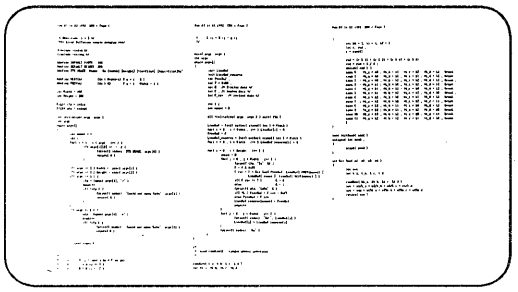
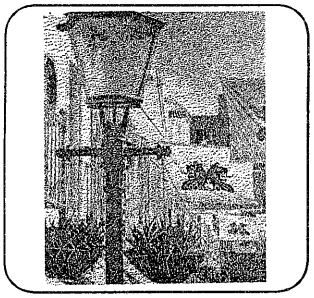


図４

そのCプログラムを“つつじ”に取り込み、シミュレーションを実行すると、そのアルゴリズムを直接目で見て評価することが可能である。シミュレーション結果を以下に示す。（図５）



評価すべき2値化アルゴリズムが複数ある場合どのアルゴリズムが最良か、また、どの様な重み付け係数が最適か、等の評価、検証をこのレベルでの試行錯誤により行なうことが可能である。

図５

■LSI化を前提としたアーキテクチャの設計

Rapid Prototypingで記述したCプログラムを、“つつじ”がサポートするブロック図で記述し直す。このレベルが、実際のアーキテクチャ設計である。有効な桁数、バス幅などここで決定する。ブロック図を以下に示す。（図６）

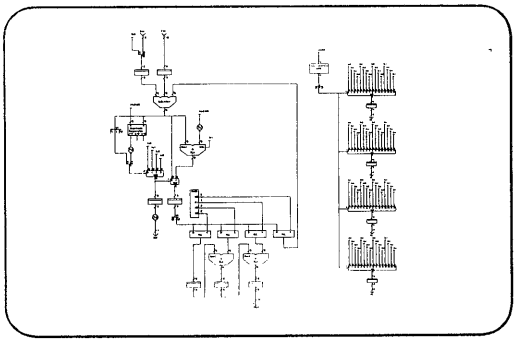


図６

複数のアーキテクチャを開発し、評価・検証することにより、そのApplicationに最適なそれを決定することができる。シミュレーションを実行すると、デジタルLSI化した場合の結果を直接目で見て評価することが可能である。シミュレーション結果を以下に示す。（図７）

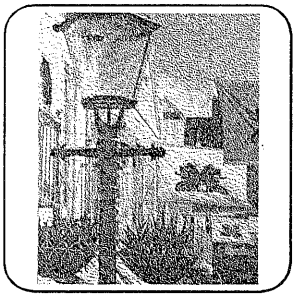


図７

ここでポイントとなるのは、シミュレーション時のテスト・データは全てRapid Prototypingで使用したそれと同一である、ということである。つまり、Rapid Prototyping シミュレーションで得た結果と全く同一の結果が得られれば、機能的に全く同一であることを保証できるのである。但し、この例では、重み付け係数の発生に乱数を使用するため、シミュレーション結果に若干の差が生じている。

このブロック図をデジタルLSI化した場合、ゲート数＝約1,600ベーシック・セル、動作速度＝約20MHz（TYPICAL、0.8μmCMOSプロセスを想定）となることを申し添えておく。

5．今後の開発方針

一般的に、上流設計の成果物はテクノロジとは独立である。また、一般の民生企業ではその成果物を応用して、製品を開発、販売する必要がある。従って、如何にその成果物が優秀でも、実際の製品開発にスムーズに反映されなければ、その価値も半減してしまう。

そこで、当社ではこの問題に対し、以下の方法で対応を検討している。

◇パラレル・アーキテクチャ＝シリアル・アーキテクチャ自動変換

Hard Wired Logicに組み込まれているシーケンサの大半が、ゲート規模の縮小などのリソースの効率化を目的としている。一方、アーキテクチャ的に見ると、例えば行列演算回路ではパラレルに記述すれば非常にすっきりと纏めることが可能だ。

設計者がApplicationに最適なアーキテクチャ（パラレル・アーキテクチャ）を創造し、実際にインプリメンテーション可能なアーキテクチャ（シリアル・アーキテクチャ）への書き換えはシステムが行なう。即ち、パラメトリックなアーキテクチャ・コンパイラを準備したい。

◇ステート・マシンの最適化

シーケンスを有するApplicationを記述する時、ステート・マシンを適用する場合が多い。例えば、上流設計ではC言語によりシーケンスを有するApplicationを記述する方がより自然である。従って、記述されたC言語から直接LSI化可能な状態に変換可能であれば、非常に有効と言える。

直接LSI化可能な状態として、以下を検討している。これらの選択は、パラメータ化して柔軟に対応可能としたい。

◆全て、Hard Wired Logicに変換。今までの論理合成の考え方である。

◆Hard Wired LogicとMicro Codeの組み合わせ。そのApplicationに最適なアーキテクチャを合成する。Micro Codeは必要に応じて組み替え可能とし、より柔軟なアーキテクチャを実現する。

これらを実現することにより、上流設計を担当する設計者は、本来の研究開発に専念することが可能となる。

また、その実現が、「Application Oriented」コンセプトによる上流設計を支援するCADシステムの使命と考えている。

【参考文献】

(1) 丸山・黒沢・中里・春日：中間調画像プロセッサ
電子情報通信学会 技術研究報告 ICD89-1
1989. 4. 20

アーキテクチャ設計にＣＡＤは有効か？

これからのＣＡＤ開発はアーキテクト主導で

小栗　清

日本電信電話株式会社（ＮＴＴ）コミュニケーション科学研究所

〒619-02　京都府相楽郡精華町光台２丁目

あらまし　設計製造行程の下位から上位に向かって発展してきたＣＡＤ技術の先端は論理設計からアーキテクチャ設計へ及ぼうとしている．この様な状況で，アーキテクチャＣＡＤの与える影響は他の設計に与えたものよりも大きいと予想されること，また，アーキテクチャＣＡＤの発展の方向も多岐に渡ること，そして，アーキテクトがその様なＣＡＤを開発すべきであることを述べた．

和文キーワード　ＣＡＤ，コンピュータアーキテクチャ，フリーアーキテクチャ，ＦＰＧＡ，同期回路，非同期回路

Is a CAD system useful for computer architecture design?

Next generation of CAD system will be developped by computer architects.

Kiyoshi OGURI

NTT Comunication Science Laboratories

2-Chome, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Abstract　**Rapid progress is being made in LSI design automation as well as in manufacture automation. From the lower to higher parts of the design process, the human designer is being replaced by computer programs. This was once considered impossible, but now the quality of design by computers matches that by human designers.**

The front of automations will cover architecture design in a few years. A architecture design automation will affect enormously. It may remove a architecture design field itself. Also, there will be many ways that the architecture design automation will progress.

英文 key words　**CAD,Computer Architecture,Free Architecture,FPGA,Synchronous Circuit,Asynchronous Circuit**

1．はじめに

コンピュータアーキテクチャはソフトウェアとハードウェアをつなぐ接点であり，さまざまなものが考え得る．そして，システムの性能や規模はアーキテクチャに大きく依存する．従って，アーキテクトには多くの経験と広く深い知識が要求され，たいていの組織で高い権威を与えられている．

良いアーキテクチャを設計するために，コンピュータを利用したい．設計者が面倒だと感じる部分，すなはち簡単な作業からコンピュータによる自動化が進められた．アーキテクチャの設計は最も難しい作業であるため自動化は進んでいない．アーキテクチャ設計の自動化のためには，アーキテクチャ設計以外の全ての設計の自動化が必要だったと言っても良い．

しかし今や，完璧ではないにしろ，アーキテクチャ設計以外の自動化が進み，アーキテクチャ設計の自動化を開始できる時期となった．

2．ＣＡＤ発達の経緯

従来から，サービス（機能）をどの様なハードウェアで処理するかを検討するアーキテクチャ設計者とハードウェアをどの様に構成するかを検討する電子回路設計者，さらにサービス（機能）を表現するソフトウェア設計者は，各々異なる組織に属し，その視点も異なることが多い．

ハードウェアのＣＡＤ技術は，設計製造行程の下位から上位に向かって発展してきており，その先端は論理設計からアーキテクチャ設計へ及ぼうとしている．発達が下位から始まったために，ＣＡＤ設計者の視点はどちらかというと電子回路設計者の視点に近い．

一方，ソフトウェアの設計は，高級言語やそのコンパイラの開発の様に，ソフトウェア設計者によって自動化が進められて来た．

アーキテクチャ設計のためのＣＡＤも他のＣＡＤと同じ様に，ＣＡＤの利用者の視点，すなはちアーキテクトの視点で開発される必要があろう．

3．ハードウェア記述言語ＳＦＬと処理系パルテノン

ハードウェア記述言語ＳＦＬとその処理系パルテノンは，アーキテクチャ設計部門で開発された，ハードウェアの動作設計用ＣＡＤシステムである．ここで動作設計とは，個々のアーキテクチャのある一例を設計する過程である．たくさんの動作設計の結果からアーキテクチャが設計される．

従来，設計者の経験に基づく勘によって行われていた動作設計の評価，見積りが，自動設計により正確に行える様になった．これにより，より定量的なアプローチが可能となった．

アーキテクチャ設計のためのＣＡＤは個々の動作設計のみならず，たくさんの可能性を組織的に検討する作業を支援する必要がある．パルテノンはその第一歩であると言える．

4．超並列プロセッサとＣＡＤ

サービス（機能）毎にハードウェアを設計，製造する訳には行かないので，従来，ソフトウェア，ハードウェアという役割分担があり，そのインタフェースがアーキテクチャであった．

ところが，ハードウェア設計が自動化され，またＦＰＧＡ（Field Programmable Gate Array）等の出現によって，サービス（機能）を直接ハードウェアに行わせることが可能となって来ている．

また，ＣＩＳＣの命令をコンパイラが有効に使えない様に，一般に，ある機能を粒度の大きい要素機能に分割するのは，粒度の小さい要素機能に分割するよりも難しい．

従って，将来非常に高性能なシステムを実現しようという時に，複雑かつ高性能なアーキテクチャを定義して，そこへサービス（機能）をマッピングするよりは，直接論理回路レベルへマッピングする方が良い結果を得られる可能性がある．

この様な場合には，アーキテクトの仕事は，サービス（機能）を一気に回路に変換するシステムに置き換えられてしまう．

5．非同期回路によるアーキテクチャとＣＡＤ

論理システムはその制御性の良さから同期回路による構成が主流となっている．しかし，全ての設計を自動化する場合には，非同期回路であっても制御可能である．クロックやマシンサイクルという概念を持たないスーパパイプラインを実現できる可能性もある．

6．これからのＣＡＤ開発

この様に，アーキテクチャ設計の自由度が大きい分，有効なＣＡＤ開発の方向も多岐に渡り，またＣＡＤがアーキテクチャ設計に与える影響も大きくなると思われる．

アーキテクチャの設計が最も重要な設計であった様に，アーキテクチャＣＡＤは最も重要なＣＡＤとなるであろう．

「高位合成の実用化へむけて: 必要な機能と応用例」
～機能合成システム Cyber による機能設計支援～

若林一敏　　NEC C&C システム研究所

1　はじめに

「アーキテクチャ設計に CAD は有効か？」と問われて、CAD 研究開発者としてはもちろん Yes と言いたいところだが、使う側からすれば現状では No であるとされる方も多いと考える。そもそも、「アーキテクチャ設計」、「CAD」という用語が意味するところが広範で、携わっている設計対象により意味する所が随分異なると考えられる。そこでまず、「アーキテクチャ設計」をいくつかの分野に分類して CAD の利用のされ方を整理し、今後の上流 CAD のあり方を考察したい。そして、最後に我々が開発中の高位合成システムでの狙いを紹介する。各設計の専門の方から見ると、問題をとり違えているとお叱りを受けそうであるが、パネル討論の種にでもなれば幸いである。

2　アーキテクチャ設計と CAD

2.1　マイクロコンピュータ

マイコンのアーキテクチャ設計では、まずデータパスありきであろう。アーキテクチャ段階からある程度レイアウトまで含めてチップ面積を考慮しながら、回路周波数等を考慮して最適なデータパスの構成を考慮する。CPU コアを利用するようなカスタムチップにおいても同様に考えられる。このような設計では、データパス部は各演算器等をモジュールジェネレータの様なツールで合成し、一方、制御論理は論理合成ツールで合成し、インタラクティブまたは、高性能のフロアープランツールで行なうといった CAD 化が抵抗なく利用できるであろう。また、データパスの性能(パイプラインやキャッシュも含め)を評価するツールがデータパスを設計する上で非常に重要なツールとなる。

この様な設計の場合、データパスを合成するような高位合成が本当の意味で役に立つためには、データパスの最適化ツールが設計者の能力と同等程度にならなければならない。現状のアルゴリズムでも制約条件がはっきりしている場合は、バスや MUX の数の上での最適化等では設計者と同等の性能を出すことはできると考えられるが、詳細な遅延やタイミングを考慮したり、トレードオフ、他の性能との総合的判断等ができない。例えば、メモリ間転送を命令セットに含めるかどうかは、バスの負荷をかなり詳細に見積もって回路シミュレーション等で遅延を調べないと判断できないが、そのような臨機応変な設計システムはまだ存在しない。また、どうしてもメモリ間転送が必要な時には、MUX 数の増加や他の遅延の増加を招いても、バスにつながるレジスタ数等を削減する等して転送速度を向上する方策が必要となる。このように最適化の目的関数自体が設計のフェーズで変更することも多い。

上記のように、マイコンのデータパス系の最適化設計は一般的な評価関数を最適化するようなアルゴリズムでは対処できないことがおおい。よって、きめ細かいさまざまな制約の基で、色々な指標の最適化が行なえるシステムがツールとして必要であると考えられる。

また、フィルタ等、音声や画像の信号処理専用のチップの設計に関してはいくつかのシステムが提案され、実用チップの合成例もかなり発表されている。

2.2　汎用コンピュータ、交換機等のシステム

ボードレベル、システムレベルの高位合成はまだ幼児段階であることから、汎用コンピュータや交換機等のシステム全体のアーキテクチャ設計にアーキテクチャ設計に高位合成を用いることは当然のことながら大変困難である。しかし、各サブシステム内のチップの設計を考えると、次節の ASIC 設計と同様に実用化が考えられる。

また、コンパイラやマイクロコードの最適化技術を基礎に、高位合成のスケジューリングやアロケーション技術が生まれたが、最近では逆に高位合成で提案されたスケジューリングやアロケーション技術をコンパイラに応用することが検討されている。また、DSP のマイクロコードを自動生成するようなツールの研究が盛んに行なわれている。これは、どこまでの機能をソフトウェアで実現するかといったソフトウェア / ハードウェアのトレードオフなどの研究も色々開始されているがまだ、設計者に使える段階にはない。

2.3　ASIC 設計

本稿では ASIC の機能設計(マイクロアーキテクチャ設計)段階をアーキテクチャ設計に含めて考える。というのは、現在、高位合成システムのターゲットとして、DSP の次に盛んになっているものの一つが、ASIC の機能設計の自動化であるからである。

まず、現在の高位合成ツールを成功させるのに、必須なのは ASIC をブロック図のような構造ではなく、プログラムのような動作で設計することである。つまり、伝統的なハードウェア設計スタイルをある意味では忘れ、新たに「HDL を用いた動作による設計」を受け入れる必要がある。ただ、HDL による設計は逆に「構造」の記述に向かないため、本質的に構造を中心に設計すべき回路の設計に用いると、図形的な発想力を低下させる等の問題点が生じる。どうしも、図の助けが必要な部分の設計を如何に統合された環境で支援するかというのは大切な問題である。

現在の論理合成入力用の HDL 記述との最大の相違点はシーケンスを書くかどうかにある。つまり、動作 A をし

た次のクロックで動作Bをするといった複数のクロックにまたがる動作を普通のプログラムを記述するように記述するかどうかという点である。ステートマシン記述は、各ステートではシーケンスは書けないので論理合成用の記述といえる。また、さらに使用する演算としてゲート演算の他、加算器や比較器等のRT部品を使用できる。論理合成は、実用化当初は、ゲート回路の最適化ツールとしての性格が強かったが、現在ではRTレベルのHDLによる入力が主になっている。このRTレベルのHDLはかなり低いレベルのため、より高位のレベルで記述するという設計形式は自然に受け入れられるはずである。よって、HDLの記述レベルの上流化を可能にするための高位合成技術、例えば演算器やレジスタのシェアリング技術等から順に実用化されていくのは間違いない。ASICの高位設計支援については次節のCyberシステムを参照されたい。

3 機能合成システム Cyber

我々が開発中のCyberシステムの当初のターゲットは、前節のようなASICの中で、特に制御が複雑でハードウェア制御するには不向きで、現在はマイクロコードで対応したり、マイコンで制御したりするような回路である。システムは、スケジューリングを行なうか行なわないかで、利用方がかなり異なる。いずれにおいても、RT部品がそのままHDL記述に使用でき、任意ビットの加算器や比較器、カウンタ等は制約にあった構成の回路が自動合成される。

1) スケジューリングを用いない場合。(Cyber I)

動作シーケンスは設計者が決定するが、演算器やMUX、転送路の最適化を行なう。ただし、ステートマシン(FSM)の形式でシーケンスを記述するのでは、

- 制御の流れが見えない。
- 10状態程度でも、状態遷移図の補助が必要。
- 大規模回路では無駄の無い状態遷移図を作成が困難。
- シーケンスの変更が、状態数の増減を伴うため困難。

等という問題点があるため、**ステートマシンと等価な動作を、C言語の制御構造とノーテーションで記述可能なHDL言語(BDL)を用いて設計する。** このHDLから一般のステートマシンへの変換はシステムが自動的に行なう。C言語と同じ制御構造とは、for、while等のループと、if、switch等の分岐、goto文、またbreak、continue文なども記述できる。goto文にはジャンプに関する制約は何も無く、どこからどこにでもジャンプ可能であり、制約の無いgoto文を利用することによって、内部状態を持つようなステートマシンでも制御用のFFを明示することなく動作レベルで記述できる。(制御用のFFは自動合成される。)外部インターフェースとの動作等も通常のHDLと同様に行なえるため、外部とのデータの授受は制約なく自由に行なうことができる。

(回路例)

ISDNのポインタ制御部の回路を用いて設計実験を行なった。設計者は、複数のカウンタを用いて回路の制御部を構成したが、Cyberでは、40状態弱のステートマシンによって合成した。カウンタによる設計や、状態遷移図による設計では、記述に混乱や不注意によるミスが多く、シミュレーションにより最終的に正しい回路を設計するのに2週間程度必要であったが、その動作はプログラムとしては非常に小さなものであるため、Cyber用の動作記述の作成は動作の理解を含め2時間程度で終了し、デバッグも30分程度で終了した。ミスの修正は、通常のプログラムのデバッグと同様に行なえ、状態数の変更などが不要なため非常に容易であった。Cyberと論理合成ツールによる合成回路と人手と論理合成を用いた設計回路を比較した結果、それらは制御系の回路形式が大きく異なったが、Cyberの回路が人手の回路より高速で、小面積な回路が得られた。

2) スケジューリングを用いる場合。(Cyber II)

C言語または、並列記述言語から、演算器の数や種類、クロック周期等を制約として、最小ステップ数で、演算器・レジスタ・マルチプレクサ・転送路数が最適化された回路を合成する。内部では、スケジューリングとデータパスアロケーション、モジュール合成、FSM合成等を行なっている。**演算器数の変更や、加算器をALUにしたり、遅い演算をパイプライン演算に変更する等演算器の種類を変更したり、クロック周期を変更したりすることによる合成結果の変化を実際の合成結果として見ることができる。**合成結果は、人手によるステップ数の最適化を大きく上回ることが多い。

従来のDSP等向けの高位合成では、すべてのクロックでデータが連続的に供給されるとか、はじめのステップでデータが揃っているとかの一般的でない制約があった。一方、Cyberシステムでは、外部との通信ピン数や通信のタイミング制約を考慮してスケジューリング等を実行するため、外部とのデータの授受が不規則な一般的な回路でも合成可能である。

(応用分野)

(1)ASICの動作記述からの合成。現在のRT記述をより上流化することが可能となる。代替データパス構造の探索の支援。

(2)汎用マイコンのASICへの置き換え: 自動車やロボット等のコントローラは、現在マイコンで行なわれているが、C言語などで記述された決まった動作を行なっている。このようなC言語記述を専用ハードウェア化することにより高速動作や1チップ化による信頼性向上とコストダウンが計れる。

(3)計算機内のアプリケーションのハードウェア化: FPGA等の利用により、計算機内に時間のかかるアプリケーションをハードウェアとして持つ。

4 むすび

高位合成の実用化への筋道について所感を述べた。現在のいわゆるアーキテクチャ設計には、現在の高位合成技術は不十分である。より正確にいうならば、現在の高位合成技術は、現在の高位設計の支援としては十分に機能せず、上流設計者にHDLによる動作設計への変革を要求している。この設計方法論の変革により、現在の高位合成技術は実用化していくことが可能と考える。また、今後の高位合成技術の中の性能予測技術の進歩は、現状のデータパス設計等の支援にも大いに役立つであろう。

盛光印刷所