VLSI向きデータ駆動型プロセッサ

技術解説

寺田浩詔*

A VLSI-Oriented Data-Driven Processor

Key Words : Data-Flow, Data-Driven, Processor Architecture, VLSI-Oriented, Reliable Software, Program Specificastion, Self-Timed Logic, Flow-Through Processing

1. はじめに

情報処理技術は，急速な進歩を遂げつつある現代の諸技術の中でも，もっとも進歩の速い領域であると広く信じられているようである．確かに，情報処理技術の応用は非常に急速に拡大しており，通信技術の発展とあいまって，社会の重要なインフラストラクチャと認められるようになっている．しかし，情報処理の基本技術の一つである，プロセッサ・アーキテクチャの側面からみると，本質的な進歩が見られないまま，推移してきた．その結果，ますます増大する新規の情報処理需要への適切な対応と既存システムの保守ならびに安定な運用とが基本的に脅かされるというゆゆしい事態が生じ，ソフトウェア危機などと言われる，困難な状況に立ち至っているのが実情である．

本稿では，いわゆるソフトウェア危機をもたらした原因を指摘し，その本質的な解決策としての，データ駆動型プロセッサとVLSI向きの実現法の考え方を紹介する．

2. 諸悪の根源

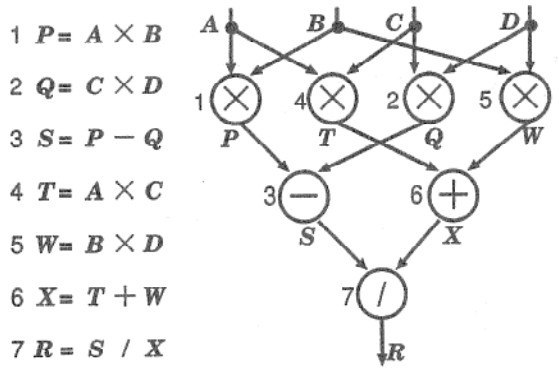
*Hiroaki TERADA
1933年7月23日生
1961年大阪大学大学院工学研究科通信工学専攻博士課程修了
現在，大阪大学工学部情報システム工学科，教授，工学博士，情報システム工学
TEL 06−879−7800

現在商用化されているほとんどのプロセッサは，1940年代に，高速数値計算機械として考案された基本的ハードウェア構成法を非常に忠実に継承している．この計算機械は幸いにも，一般的な情報処理の基本機能である，抽象的な情報加工機能と汎用的な情報蓄積能力とを持っていたために，数値計算以外の広範な情報処理にもそのまま転用されてきた．しかし，この機械の基本構成は，あくまで1940年代の手計算アルゴリズムを，当時の技術で実現可能な，最小のハードウェアで実現するために考案されたものであり，したがって，今日ソフトウェアと呼ばれる概念はまったく考慮されていなかったことが今日深刻な問題を生じるに至った最大の原因となっている．

もちろん，たとえ数値計算だけを目的とする機械であっても，そのアルゴリズムを指示する手段としての，なんらかの記述が必要となる．したがって，その誕生の直後から，この機械の物理的な性質をそのまま反映した，アセンブリ言語が考案されさらに，個々の機械の物理的性質からは独立な記述を可能にするために，いわゆる高位言語が生み出された．後者の代表格が，今日なお数値計算の世界では強い影響力を残している，FORTRANである．しかし，この種の言語にも，アセンブリ言語の基本的な性質が残念ながらそのまま保存されている．

2.1 現在のプログラム言語の基本的性質

この種の言語の基本的性質は，〔1〕ロケーショ



(ア). 逐次代入型記述　(イ). 計算グラフ型記述

図1　プログラム表記法の比較

ン(プログラム)・カウンタの指示による文の逐次実行ならびに,〔2〕代入による変数の値の間接的な受渡し,に要約される.前者の性質は,この種の言語を実行するプロセッサが『プログラム・カウンタを持つプロセッサ』と総称されるように非常に基本的な特徴であり,文章型に書き並べられたプログラムの特定の一つの文が,ロケーション・カウンタの指示にしたがって,文の並びの前後関係とはまったく無関係に,無条件に実行されることを意味している.第二の性質もまたこの種の機械の物理的な機構を反映したもので,記憶装置のある特定の場所の内容を書き換える動作と,このように書き換えられた変数の値を,複数の別の文から,無制限に参照できることを反映している.

以上の基本的な性質を具体的な例で示すために図1(ア)のプログラムの断片を用いる.たとえば,ロケーション・カウンタが文3の実行を指示すれば,前述の性質〔1〕にしたがって,この文が無条件に実行される.この文はいわゆる代入文と呼ばれ,性質〔2〕にしたがって,その右辺にある二つの変数がその時点で保持している値が,指示された演算(乗算)を受けて,左辺の指示する変数の値を書き換える.ロケーション・カウンタの値は,一つの文の実行と同時に,原則として次の文を指示するよう変更されるので,一連の文が逐次実行される.

2.2 現在のプログラム言語の問題点

上述の実行規則の最大の問題は,ある文の右辺に書かれた変数の値が,そのプログラムの文脈の中で,正しい値であることの保証がまったく無いことにある.すなわち,このプログラムの文脈は,文3の実行時には,その右辺には,文1,2によって書き換えられた,変数の値が正しく記憶されていなければならない.しかし,文3の実行時に,これらの処置が行われたかどうかはまったく確認の方法がないので,非常に危険な処理が行われることになる.

このプログラムの断片は,非常に簡単な直線的な部分を取り出したものであり,一連の処理が文1から実行されることさえ保証できれば,プログラムの意味論的な正しさが検証できる.しかし,一般のプログラムは,多数の分岐を含むので,実行時に遭遇する可能性のあるすべての実行経路について,その正しさの完全な証明は,組合せ論的な爆発の問題に直面し,一般に実行不可能となる.

3. データ駆動原理

前述のように,現在のプロセッサすなわち逐次代入型処理方式の原動力はロケーション(プログラム)・カウンタにある.これに対して,データ駆動型処理はプログラム・カウンタの無いプロセッサとも言われるように,処理実行をプログラム・カウンタのような,人工的な手段に頼らない点に大きな特徴がある.

3.1 データ駆動型処理の原理

図1(イ)は,前述のプログラムの断片を,計算グラフ形式に書いたものである.もし,このグラフを,ある節(ノード)が処理を,そのノードに入る有向直線(アーク)がその処理に必要なデータの経路をそれぞれ示すと解釈すれば,これら二つの表現はまったく等価であることは容易に理解されよう.

データ駆動型の解釈ではさらに,あるノードの処理実行に必要なすべてのデータの組がすべての入力アーク上に到着したとき(そのノードによる)処理実行が開始されるとする.この条件が成立すれば,個々のノードは,入力データの組をノード内に取り込み,処理を施した後に,出力アーク上に結果を送り出す.図中のコピー・

ノードは，データ駆動型の解釈を厳密にするために挿入されたもので，その入力へのデータを単に複製し，複数の出力に与えるノードである．ノードへのデータの到着を抽象的に表現するために，（データの）トークンという概念を導入し，すべての入力にトークンが与えられたときに，そのノードが処理可能になり，発火するという言うことである．

この簡単で自然な実行原理にしたがえば，同図（ア）とは非常に異なった，実行が可能になる．たとえば，〔1〕発火条件は複数のノードで同時に生起できるので，もし処理の実行に必要なハードウェアが複数準備されていれば，複数の処理が同時並行に実行される，〔2〕もし入力データの組が複数与えられれば，それぞれのデータの組について，それぞれ独立に処理が進行する，という重要な性質が導かれる．前者の性質を同時並行処理，後者の性質をパイプライン並列処理と呼んでいる．これらの性質は，前述の逐次代入型処理にはまったく見られない性質である．以下，簡単に二つのプログラムの比較を試みる．

3.2 処理方式の基本的相違点

i．プログラムの了解性：図1（ア）のプログラムでは，意味を変えること無く，幾つかの文の実行順序を入れ換えることができることは容易に確かめられる．これに対して，同図（イ）のプログラムでは，処理の間には，位相幾何学的に不変な，本質的な処理の構造が維持されている．すなわち，ある処理の結果を必要とする処理は，不特定多数ではなく，アークで結ばれた処理に限定されている．これは，ある処理に先行するべき処理（先行性制約）が厳密に指示されていることを同時に意味している．また，見方を変えれば，たとえば処理1,2,4および5は，どの順序で行われても，処理1,2が処理3に，処理4,5が処理6にそれぞれ先行して行われる限り，このプログラムの意味は変わらないことが明瞭に示されている．

逐次代入型のプログラムでは，本来は順序関係のない処理であっても，文章型に記述するために，前後関係のある記述を見かけ上強制されることになる．すなわち，同じ意味のプログラムでも，プログラマの恣意によって，様々な書き方が許される．アルゴリズムは同じであるが，異なった書き方の二つのプログラムの等価性の確認は，小規模なプログラムでは可能な場合もあるが，実用的なプログラムでは，組合せ論的な困難から，実際上不可能である．また，この性質が，逐次代入型プログラムの了解性を悪化させる大きな要因になっている．

ii．部品化の容易性：ハードウェア・システムと同じように，ソフトウェアを部品から作るあるいは既存ソフトウェアの再利用は長い間の夢であった．

しかし，逐次代入型のプログラムでは，前述の例にも見られるように，あるプログラムに受け渡されるのは，その入口の指定すなわち制御であって，処理にとって本質的な，データ（の組）ではないために，接続関係の確認が極めて複雑になる．したがって，部品化は，言うべくして行うことができない，夢に止まらざるを得なかった．

これに対して，データ駆動型プログラムでは，ハードウェアの部品と同じようにデータの出入り口が規定されかつ，その入力に必要なデータの組が与えられない限り，そのプログラム・モジュール（部品）は起動されることが無いという原理的に優れた特質があるために，接続関係が非常に明確であり，部品化が格段に容易になることが理解されよう．

4．ソフトウェア仕用記述の問題

ハードウェアよりソフトウェアが問題であるという主張は，必ずしもコンピュータの専門家ではない人々の間でも，常識化しているようである．しかし，この表現には十分な注意が必要である．

4.1 ソフトウェアの難しさとは？

コンピュータ・ソフトウェアを作る場合にまず必要になるものは，そのコンピュータ・ソフトウェアが，あるハードウェア環境において，どのような機能を要求されるかを定義した，要求仕様である．たとえば，あるハードウェア環

境のもとで，そのゲーム・ソフトウェアを作るという場合，そのゲームの内容（WHAT）からシステム要求を決定し，これを満たすソフトウェアの仕様（HOW）を決めるという手順を経るのが順当である．ソフトウェアが難しいと言われるのは，知的な創造活動の領域に属する，前者の領域を指しているのであろうと考えられる．現状では少なくとも，要求仕様定義とそれに対するソフトウェア仕様定義とが明確に区分されないことが多いので，要求仕様の明確化を含めて，ソフトウェアあるいはその生産と呼ぶのが慣行化しているように見受けられる．

しかし，単に要求定義をソフトウェア仕様定義として記述しさらにこれを実現することがソフトウェア生産の課題であると問題を単純化してもなお様々な問題が残されているのが現状である．

4.2 上流工程と下流工程

いわゆるソフトウェア工学の領域では，要求定義に見合うソフトウェア仕様定義をする作業を上流工程，これをソフトウェア構造に変換する作業を下流工程と呼ぶ習慣がある．

このような分割があること自体が不思議と言うべきであるが，この呼び名が成立したのは，両者の間に深い溝があったからである．すなわち，上流工程ではあくまで，要求仕様に合った，システム設計が課題であるために，ハードウェア・システムを含めて，システム設計に用いられてきた記述手法が援用される．たとえばシステムの機能構成図に始まり，そのサブシステムへの分解，サブシステム間でのデータの授受の内容と時間的な順序関係などが，主として図的な手法を用いて，記述され，これが要求仕様と合致するかどうかが検討される．

しかし，現状の言語を用い，これらの機能をプログラム化する過程すなわち下流工程では，このソフトウェア仕様をさらに，現状の言語で記述できる構造に変換する作業が必要になる．この作業は，図１（イ）のような記述を同図（ロ）の記述の形に変換する過程にほぼ相当している．このような簡単な例では理解しがたいが，実際の規模のプログラムでは，この作業は非常な熟練と見通しを要する作業になるので，その専門家としてのシステム・エンジニアの活躍に待たねばならない領域が存在することになる．もちろん，上級のシステム・エンジニアには，単にこの構造変換だけではなく，要求仕様そのものが一般に流動的であるという現実から，システム全体を見直す作業を並行して行うという，超人的な能力が要求されるので，その養成は困難を極める過程になっている．

4.3 一貫的なソフトウェア生産へ

このような過程をより一貫的にし，作業全体を容易にして生産性を向上するとともに，不必要な技能の習得を不要にしたいと考えるのは当然であろう．

すでに示唆したところからも想像されるように，上流工程と下流工程との乖離は，システム記述手法とかけ離れたプログラム手法を利用せざるを得ないことから発生している．したがって，両者を統一的に結合するためには，下流工程のそれを変革することが自然な筋道になる．データ駆動型プロセッサ研究の大きな狙いは，既に述べた並行処理やパイプライン並列処理を活用することはもちろんであるがそれ以上に，ソフトウェア生産を一貫化することにあった．すなわち，すでにハードウェア・システムの記述に用いられて多くの経験が得られている，システム記述手法をほとんどそのまま利用できるソフトウェア生産体系を作ることが最終的な目標である．

この手法の基礎は，階層的な部品化にある．たとえば，多様なハードウェア・システムが標準的なネジやアセンブリから作られるように，標準的なソフトウェア部品からアセンブリを作り，さらにそれらで作られたより大きいシステムを再利用するといった階層の基盤を作り上げることが最終的な目標となっている．

5．VLSI向きデータ駆動型プロセッサ

データ駆動型プログラムはそのまま素直に解釈すると，一般的なシステムの表現と動作に酷似し，特に，ハードウェア・システムの表現としてそのまま解釈できる．

たとえば，図1（ロ）において，ノードの表現する演算をハードウェア的な演算器で置き換え，アークをそれらの間の配線と解釈すれば，同図は特定の演算を行うハードウェア・モジュールの表現そのものになっていることが理解されよう．

したがって，このようなプログラムを，限られた個数の演算器をもつプロセッサで実行するためには，演算器の接続関係を記述する部分とその接続関係にしたがって，実行可能なデータが存在するかどうかを検出する機構とを実現すればよい．すなわち，前者がプログラム記憶機構であり，後者が，データ駆動型プロセッサに固有な，発火検出機構になる．

5.1 VLSI向きデータ駆動型プロセッサ

図2に，このような考え方に基づいて作られた，VLSI向きデータ駆動型プロセッサの原理的構成を示す．

データ駆動原理が非常に簡単であるために，そのハードウェア実現もまた非常に簡単な機構で実現される．すなわち，図2において，入力データは，宛て先ノードとデータとの組からなる，入力パケットとしてプロセッサに入る．発火検出機構MMは，宛て先ノードを鍵とする，一種の連想記憶機構であり，そのノードの演算に必要なデータの組の到着を互いに待ち合わせる．もし，データの組（演算パケット）が成立すれば，これらは演算コードとその演算に必要なデータの組となって演算機構FALUに，宛て先ノードはプログラム記憶PMにそれぞれ送られる．FALUでは演算が実行され，その結果が出力される．一方のPMでは，現在の宛て先ノードを鍵として，次の宛て先ノードが読みだされる．両者は再び入力パケットとして組み合わされることによって，処理が継続する．

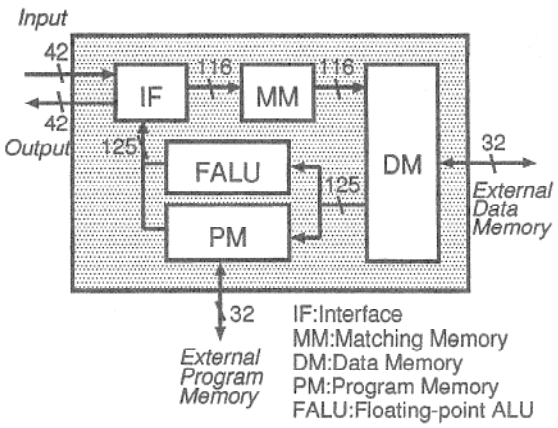


図2　RAPIDの機能構成

5.2 データ駆動型プロセッサの特徴

データ駆動型プログラムの特徴は，その実行原理から，ある処理（ノード）が発火すれば，その処理は他のノードの処理とは独立でありかつ，どの処理も途中で中止する必要がないことにある．したがって，逐次代入型プロセッサでは一種の先取り処理に相当するために複雑な制御を要する，パイプライン並列処理が非常に効果的に行えることがハードウェア実現上の大きな特徴となる．したがって，もし，同時並行処理あるいはパイプライン並列処理によって，あるプログラムのノード群が，一定の割合で発火すれば，個々の処理の遅延とは無関係に，高い処理率が維持される．

図2のプロセッサでは，この事実を極限まで利用するためにあらゆる機構を多段のパイプラインで実現し，さらに段間の転送制御を，いわゆるクロック・スキューが問題となるシステム・クロックを用いた同期型ではなく，自己タイミング型としている．このような方式によって，このプロセッサはVLSIの性能を極限まで引き出すことを意図している．

また，データ駆動型プログラムは，プログラム分割が比較的容易である事実を利用して，非常に多数のプロセッサを結合した，マルチプロセッサを構成できるよう配慮してある．

6．研究・開発の現状

このプロセッサの研究は，当初からVLSI向き実現を想定したために，半導体製造会社4社と我々の研究室との共同研究として，1983年に発足し，1986年にはモックアップが完成し，基本的な動作が確認された．この後，三菱電機（株）ならびに（株）シャープがVLSI化の努力を傾けられ，現在では32ビットおよび12ビット並列処理を行う1チップ版の完成を見た段階にある．

本稿に述べた狙いからも理解されるように，本研究プロジェクトは，ハードウェアからソフトウェアにわたって，新しいパラダイムを創りだそうとする試みであり，非常に幅広い研究者の協力が得られるかどうかが，本研究の成否を握っている．

現在，ハードウェアについては，プロセッサのシミュレータ，エミュレータならびに各種の性能評価環境が稼働している．また，4チップを環状接続した評価用ボードが提供可能な段階にまで達している．ソフトウェアについては，AESOP（Advanced Environment for Software Production）と総称される，仕様記述からデータ駆動型の実行可能なプログラムを直接生成する環境の完成に向けて努力を続けている．

先にも述べたように，本研究には多数の研究者の参加が必須であり，すでに米国ならびに豪州の大学の研究者達と主としてソフトウェア環境に関する共同研究を開始しているが，なお多くの研究機関との共同研究を進めてゆきたいと考えている．関心をお持ちの方々のご協力を切に期待している．

7．むすび

本研究は，計画から10年以上の歳月を費やし，ようやくハードウェア，ソフトウェアともに一応の評価が可能な段階に達した．この間，松下電器産業（株），三洋電機（株）などのご協力を得た，特に（株）シャープならびに三菱電機（株）には，VLSI化について格段のご配慮を戴いていることに厚く感謝の意を表したい．また，我々の研究室の教官，職員，学生諸氏にも多大なご協力を戴いた．個々にお名前を挙げる紙数が許されないが，これら関係各位の熱意なくして，本研究の展開はあり得なかった．厚く感謝申し上げる．

また，研究の過程を通じて，これら各社の研究を支援された通商産業省，我々の研究を科学研究費によって支援していただいた文部省にも厚くお礼を申し上げたい．