報　告

計算機アーキテクチャ研究会 第100回記念歴代主査特別講演†

相磯　秀夫[1]，　飯塚　肇[2]，　田中　英彦[3]
富田　眞治[4]，　島田　俊夫[5]，　中島　浩[6]
山崎　憲一[7]，　平田　圭二[7]

1. はじめに

さる1994年10月27，28日の両日，奈良先端科学技術大学院大学にて，第100回計算機アーキテクチャ研究会（SIGARC）が開かれた．その100回を記念して同研究会の歴代主査（6名中5名）による記念講演が行われた（**表-1**）．歴代主査の方々は，現在も精力的に研究活動を続けておられ，その研究対象も計算機アーキテクチャだけにとどまらずOS，言語，応用と多岐にわたっている．また現在は全員大学で教鞭をとっておられるが，これまで，大学，公立研究機関，企業など多彩な環境で研究を続けて来られたという貴重な経験もお持ちである．記念講演自体はなごやかな雰囲気のうちに進行したが，その内容は時に技術的であり，時に大所高所からの視点を持ったものであった．学会誌編集委員会では，計算機アーキテクチャに興味を持つ計算機科学者，工学者のみならず，広く情報処理に携わる者全体に対しても，この講演は非常に有益なものであると考え，本稿を企画した．なお，7.“おわりに”では，SIGARC現幹事を代表して中島浩氏にむすびの言葉をお願いした．

2. 相磯秀夫先生

今日のお話の参考文献としましては，情報処理学会歴史委員会(編)の「日本のコンピュータの歴史」と「情報処理学会30年のあゆみ」があり，丁寧に解説されておりますので，皆様ご覧になるとよいかと思います．

過去のアーキテクチャ研究はコンピュータの開発に直結しております．仮想的な設計だけをしておしまいということはありませんでした．最初に1950年代後半の計算機研究について述べたいと思います．この時代は私にとっては楽しい時代でした．基礎理論，論理回路，計算の方法，特に割り算機構については時間を費やして研究しました．また，シーケンシャル制御等にも没頭しました．つまり，どのようにしてコンピュータというか電算機を作るかということです．代表的なものは，申すまでもないと思いますがENIAC等で，その文献はずいぶん参考にさせていただきました．

私は1955年の12月（大学院1年）から，電気試験所のトランジスタ計算機の開発に学生として加わりました．電気試験所はNational Bureau of Standard（NBS）という米国の研究所とタイアップしており，そこのSEAC（Standards Eastern Automatic Computer）というコンピュータの情報を参考にしました．しかし，それほどたくさんの資料があるわけではなく，様々な資料を写真に撮ってガリ版刷りにして勉強しました．

私がそこで一番ショックを受けたのは，ケンブリッジ大学のM. V. Wilkesらが書いたあのEDSACのプログラムの本（The Preparation of Programs for an Electronic Digital Computer, Addison-Wesley (1951)）です．大した説明もない本で最初は全然分からなくて興味も沸かなかったのですが，この本を説明しろと言われまして一生懸命勉強しました．そこで私は初めてstored programの面白さを知りました．そういう意味で最も私の印象に残っている本です．

50年代後半になりますと日本もコンピュータ

† 日時　平成6年10月27日（木），28日（金）
場所　奈良先端科学技術大学院大学
1）慶應大，2）成蹊大，3）東大，4）京大，5）名大，6）京大，7）NTT

表-1 計算機アーキテクチャ研究会歴代主査，幹事の任期(表中敬称略)

	'74	'75	'76	'77	'78	'79	'80	'81	'82	'83	'84	'85	'86	'87	'88	'89	'90	'91	'92	'93	'94	'95
主査	←	相磯秀夫	→	←	石井　治	→	←	飯塚　肇	→	←	田中英彦	→	←	富田眞治	→	←	島田俊夫					
幹事	←	所真理雄	→	←	有澤　博	→	←	大島一純	→	←	喜連川優	→	←	後藤厚宏	→	←	中島　浩					
				←	内田俊一	→	←	坂村　健	→	←	長谷川隆三	→	←	村上和彰	→	←	関口智嗣					
													←	横田　実	→	←	矢野陽	→	←	木村康則		

を作り出すわけです．リレー計算機，ETLMK-II，FACOM 128，真空管計算機 FUJIC，大阪大学の名無しのコンピュータ等がありました．

一方，後藤英一先生，高橋秀俊先生のグループはパラメトロンの研究をなさっていました．それと同時に電電公社の通研が MUSASHINO-1 号というパラメトロン計算機を作りました．その頃は，電気試験所はトランジスタ，通研はパラメトロンという約束があって，それぞれ競って研究開発していたわけです．この MUSASHINO-1 号の責任者が室賀三郎先生です．室賀先生はこれを作る前にイリノイ大学に留学なさっていて，IL-LIAC-I という真空管の非同期式コンピュータ，実は私もイリノイ大学に行ってこれを使ったことがありますが，先生はこの計算機のライブラリが非常にしっかりしているのをご覧になり，このライブラリを使うには同じ命令体系でなければならないと考え，同じ命令体系の計算機を作られたわけです．今でいえば，計算機の互換性をとるということになるわけです．しかし実際には細かい仕様が分からないとそう簡単には互換性はとれません．残念ながらこのライブラリは使えなかったのですが，プログラム（ソフトウェア）の重要性や開発の大変さを考えて同じ命令体系で設計したというのは非常に先見の明があったと思います．

国産トランジスタ計算機の最初は ETL の Mark-III です．その頃トランジスタをちゃんと作ってくれるのは東通工（現在のソニー）と日立しかありませんでした．東通工は点接触型トランジスタを作りました．このトランジスタはものすごく敏感でちょっと多めに電流を流すとすぐに壊れてしまいました．私はトランジスタを受け取るために五反田の会社によく行ったものです．そこに江崎玲於奈博士がいらっしゃいました．あるとき，江崎さんが面白い物を見せてあげるよといって見せていただいたのが，あのノーベル賞をとったエサキダイオードのトンネル効果の特性でした．ダイオードに不純物を入れて行くと面白い負性抵抗特性が出てくるんだよとおっしゃるのですが，私にはどこが面白いのか全然分かりませんでした．この人は何やってるんだろうと思ったものです．それから，保坂衛先生の国鉄の MARS-1 があります．これは実時間性，信頼性を追及した計算機でした．

このような計算機ができ上がったのがだいたい 50 年代後半ですから，世界から 10 年くらい遅れていたわけです．この時代，アーキテクトがしたことは，ともかく開発をするのが前提ですから，まず基本回路というものを設計したのです．東京大学がパラメトロン，TAC（Todai Automatic Computer）は真空管のフリップフロップ，電気試験所はトランジスタ回路です．特に ETL Mark-IV は私の修論です．点接触は不安定なので接合型トランジスタを使って作るというものでした．このトランジスタを作ってくれたのがやはり東通工と日立でした．私の最初の課題はトランジスタを選べというものでした．何本かを選んで特性を測ってみますと，大信号特性は日立の方が揃っていてきれいなのに対し東通工のはバラバラでした．でもスピードは東通工の方が速かった．これはラジオ用の小信号のために作っていたからです．東通工の方がカットオフ周波数が数倍高く，こちらならおそらく 1 MHz で動いたでしょうが，あまりにも特性にばらつきがあったものですから私は日立の HJ 23 というトランジスタを選びました．これがなんと 1 本 5000 円でした．その頃の公務員の初任給が 9700 円ですから，いかにしてトランジスタの数を減らすかが課題でした．それで作ったのがトランジスタは 1 本しか使っていませんが，そのかわりトランスとコンデンサとバリコンが必要な複雑な回路です．こんな部品は今の LSI には絶対に入らないです．これを駆動するためにクロックを 3 つ作って分配します．これを 200 KHz で動かすのですが，これが

大変でした．これを動かすには回路がよく分かってないといけないし，アースのとり方も重要ですし，電磁気も分かってないといけない．私は慶應大学で先輩に，大学の勉強はオームの法則しか役に立たないと言われたものですから，学部時代は遊んでいました．ところが電気試験所に来てシマッタと思いました．電気試験所の人はものすごくよく基礎が分かっていて，様々な電気現象をちゃんと理論レベルで説明してくれるわけです．ですから，この回路を実際に大規模に動かそうとすると，電磁気とか回路の基礎知識がないと駄目だと気付きました．

Mark-IVの良いところは，これをプリント基板にしたために大変安定に動いたということです．これを普通にソケットで作っていたら駄目だったと思います．もっともプリント基板も電気試験所の専門家が作ってくれたのですが，電流が流れる線が太さ2mmくらいありました．その専門家の人がこれ以上は細くできないというので，ずいぶん大きなものになってしまいました．というわけで，この時代は回路の特徴を競った時代でした．

次に60年代から70年代の前半のお話をします．この時代になるとずいぶんアーキテクチャらしくなってきました．アーキテクチャという言葉はIBMのSTRETCHという計算機について書いた本（W. Buchholz (Ed.)：Planning a Computer System—Project STRETCH, McGraw-Hill (1962)）の中に初めて出てきます．またパイプラインといった言葉も出てきます．その後G. AmdahlがIBMのSystem 360でアーキテクチャをきちんと整理して体系化しました．またBurroughsのR. Bartonが，ディスクリプタという概念を発表します．これは命令やデータにタグをつけて特殊な動作をさせるものです．次いでマンチェスタ大学のT. Kilburnがワンレベルストレージという，今の仮想記憶の原型となるものを発表します．それからミニコンピュータPDP-8（1965年発表）と続きます．私は1966年にDEC本社に行きました．倉庫のようなところでしたがDECの社長は，粗末なところだが良い技術はこういうところから生まれるのだ，と言いながらPDP-8の説明をしてくれました．

もう1つ私が非常にショックを受けたことがあります．その頃，通産省が大型プロジェクトを始めて，時分割システムが注目を集めました．そのため日本工業振興協会がMITのJ. Soltzer助教授を日本に招き，MULTICSという時分割システムのOSについて連続講義をしていただきました．私にとってはこの講義が本当にショックでした．もちろん中身も斬新でしたが，とにかく講義がうまかったのです．その後も欧米から何人も呼んで講義をしていただきましたが，彼ほどうまい人はいませんでした．ユタ大学のE. Organick先生もかなり面白かったです．その後は，Basic Machine，並列コンピュータのILLIAC-IV，CDC 6600，CDC STAR，ロンドン大学のDistributed Processor, CRAY-1と次々新しい概念のコンピュータが出てきました．それから，データフローの概念，高級言語マシン，様々な分散処理，フォールトトレラント計算機構です．そしてインテルのマイクロコンピュータが開発されます．幕開けともいえる時代ですね．技術の方向としてはまずCISCの方に展開していき，この後RISCアーキテクチャの方にいくわけです．

私は今はアーキテクチャの研究はしておりませんが，皆さんの研究の題名を見ておりまして思うことは，最近はずいぶん議論が細かくなり，アイディアが出しにくい時代になったなと思います．それとスーパRISCのあたりにはまだ大きな研究テーマがあるのではないかと思っています．

3. 飯塚肇先生

私は，現在新しいマシンを構築する研究をしているわけではありません．ただ，昔やっていろいろと失敗もしていますので，少し思い出をお話しさせていただいて，それから今後のアーキテクチャへの期待なども話せたらいいなと思っています．

まずアーキテクチャというのはこれからも研究になり得るのかということですが，私は結論的にいえば，常に新しい研究テーマであり得ると思っています．コンピュータアーキテクチャというのは，これが絶対のアーキテクチャだというのはないわけでありまして，良いアーキテクチャというのは周辺技術との関係から決まります．要求にし

てもテクノロジーにしても，これから先も変わっていきますから，新しいアーキテクチャの研究を先を見通してやり続けるということは必要です．そういう意味でアーキテクチャの研究は永遠です．

ではこれから昔の反省をしながら少しお話しいたします．自分の身近な電総研のアーキテクチャの流れを見ながら考えてみましょう．電総研のアーキテクチャの研究というのは，50年代から60年代にかけては，まず実用的なコンピュータを作り，その後でより高速なコンピュータを作るということがテーマでした．そこで出てきた様々なアイデアとしては，先ほどの基本回路であるとか，相磯先生の先行制御（今のパイプライン制御），あるいはキャッシュのはしりであった命令バッファ，そういったものがいくつも見られました．その頃はとにかく動いて，当時の技術として一番バランスの良いアーキテクチャを考えるのが中心でした．私は修士卒業が64年ですが，その頃外から見ていて電総研というのは，いろいろ新しいことをやっていて面白そうだなと思い，電総研に入りました．

ところが60年代半ばになりますと，メインフレームを作るのはメーカでなければできないようになり，研究所で作るのは難しい時代になってきました．それと同時に60年代から70年代はソフトウェアが重要であるといわれて，ある意味で電総研はアーキテクチャの研究を止めてしまい，システムソフトやTSSのOSの研究をやったわけです．私自身も最初はハードをやろうと思っていたのですが，一番最初にファイルシステムをやらされました．

70年代に入りますと，ご存じのようにマイクロプロセッサが出てきて，いろいろなアーキテクチャ研究ができる時代になりました．そこで，マイクロプロセッサ，マルチプロセッサ，ファームウェアコーディングマシンなど，電総研のアーキテクチャ研究の立ち上げみたいなことをやりました．今思ってみると，マルチキャッシュや，分散共有メモリ，あるいはロードストア命令主体のマイクロプロセッサといった，現在にも関わるようなことを少し研究していました．

ところが80年頃になりまして電総研が筑波に行きました．私は筑波に行くのが嫌でしたので辞めまして(笑)，それは私のアーキテクチャ研究という意味では失敗でした．その後，島田さん，弓場さん，山口さんらがデータフローマシンを始めて，一生懸命立派な成果をあげられているわけです．

以上のことを題材にしましてアーキテクチャ研究に対するコメントをいくつか申し上げようと思います．まず1つは，アーキテクチャの研究というのは継続性が非常に重要であるということ．アーキテクチャの研究をいったん止めますと，もう一回やり直すというのは大変な労力が必要です．たとえば，電総研はアーキテクチャの研究を一時期止めてしまい，その後，相磯先生あたりが中心になってもう一回立ち上げようとしたのですが，なかなかうまくいきませんでした．現在は良い研究をされていますが，やはり一度止めてしまうと大変です．

もう1つは，粘り強さも重要です．実は相磯先生や私がやっていた研究も今思ってみますと，それなりに良いアイデアがあって，その発展形が現在実用になっていたりします．ですが，その研究当時ではもう1つ突っ込みが足りなかった．たとえば，マルチキャッシュがそうです．論文を1つ2つ書いたら止めてしまうというのではなく，出てきた芽を発展させ突っ込んでいくということが大切です．

それと，あまり欲張り過ぎも駄目です．私の場合，マルチプロセッサの研究をしていましたが，マルチプロセッサの構成をやるだけでなく，実はプロセッサ自身も新しいのを作りたかったわけで，どうしてもそちらもやってしまう．ところが人間が多くはいないので，だんだん手が回らなくなってしまう．たとえばKSR，nCUBEなど現在でも全部作っている会社もありますが，ちょっとチップ作りが遅れてしまっています．ポイントを絞り欲張り過ぎないでやるのがよいと思います．

もう1つ．過去何百台，何千台の試作機というのが作られ論文がそれぞれにおいて何件か出たわけですが，たいていはそれが出てしまうと，だんだんしぼんでしまいます．なぜ駄目だったのか，何が失敗だったのかということをデータをとり公開するという仕事が少し足りないという気がします．残念なことに，そういう駄目なデータ，たと

えばこのキャッシュのヒット率はこう悪かったという論文を学会誌に投稿しても，オリジナリティがないといわれて，多分採録されないでしょう．しかし，実際はどう悪かったのかということをきっちり究明しなければなりません．一番マズイのは，そのまま放ったらかすことです．

最初にも言いましたが，アーキテクチャというのはバランスが良くないと良いアーキテクチャにはなり得ません．ですからアーキテクチャというのは総合的な評価をしなければいけません．たとえ新規性があっても，自分の主張しているポイントだけが良くなるようなシミュレーションデータを出しているのは，良いアーキテクチャではないと私は思います．全体のバランスがとれたアーキテクチャを考えないといけない．ですから試作することに意味がある．論文の方もそういうことを十分考えて評価されなければいけない．単に新しいというだけで論文が評価されるのは，アーキテクチャの場合は特によろしくないと思います．

良いアーキテクチャはディスカッションから生まれます．私は大学に行ってつくづく思ったのですが，電総研にいた頃は，たくさんの優秀な研究者とのディスカッションによって，よりよいものを作れました．ところが今の私の場合，アーキテクチャの研究者が回りに少ない．するとどうしても抜けが出てきます．そういう意味で，良い指導者のもとでグループで研究していくことが必要だと思います．そして良い指導者に指導された人が，また次の指導者となっていく，ということがパイプライン的に続けば，良い研究が続いていくのでしょう．

さて，ここまでは反省でしたが，ちょっと期待の持てる話を申し上げます．これからのアーキテクチャ研究をどうするのかということです．

私はもともと電総研で研究をして，その後大学に行きました．それで新しいコンピュータを開発するための研究と，教育や論文のための研究とは少し違うという気がしています．開発の研究というのは，要するに次世代のコンピュータアーキテクチャにインパクトを与える，実用性のある意欲的な研究が，一番良い研究だと思います．その意味で日本で成功した研究というのは少ないです．たとえば相磯先生のやられたMark-IVはインパクトの大きかった開発の研究だと思います．また，今の最先端のマシンに使われている基本技法は，必ずしもアメリカのものばかりではなくて，よく見ると日本にも昔からあったと思うんです．ところが海外から言われないと，なかなか認識されない．たとえば京都大学の有名なQA-2は，VLIWアーキテクチャとしてはずいぶん昔からおやりになっているわけで，当然日本のメーカがもっとVLIWに注目していてもいいはずなんですが，それほどでもない．日本にも良い研究の芽というのがあるので，その芽をもっと大切にして欲しい，とメーカの方々にお願いしたい．

一方，大学では主に教育重視の研究をすべきです．しかし将来的に役立つアーキテクチャ教育の研究ができる大学は，日本ではそうたくさんはありません．では普通の大学でアーキテクチャの教育はしなくてもいいかというと，これは情報工学の基礎ですから，教えなければいけないと思います．アーキテクチャ教育では，口だけでなく実際にマシンやソフトを作ってみる必要があります．それによって優れたコンピュータアーキテクトが生まれるでしょう．大学の方は，論文を作るためだけの研究をやっていても駄目で，企業の技術者の教育も含めて，良いアーキテクトを作るための研究を進めていく必要があると思います．ただ，アーキテクチャの本格的な教育・研究を大学でやるのは，日本の場合は大変苦しいと思います．何人かのアーキテクチャの専門家と優秀な学生が，継続的に議論できる環境というのは難しい．特に大学の場合はもうちょっとポイントを絞って新しいコンセプトの研究を発展させて欲しい．そういう力を持っている大学の先生方には，ぜひお願いしたいと思います．それと，企業はもっと幅広く学校を支援して欲しいと思います．これは必ずしもお金を出すということではないんです．アメリカのように，シミュレーションするためのベンチマークやトレースしたデータを企業が公開したり，試作を引き受けたり．アーキテクチャ研究をスムーズに進めるための様々な方策があり得ると思います．

21世紀には，日本にも真のコンピュータアーキテクトがぜひ誕生して欲しいです．ここでいう真のコンピュータアーキテクトというのは，たとえばPattersonとかCrayとかいった1つの哲学を持って研究開発している人々です．天才は突然

生まれません．じっくり環境を整備していかないといけないと思います．

あまり面白い話になりませんでしたけれども，要は，アーキテクチャというのは今後とも大事ですから，どんどん研究しなければいけない．さらにいえば，アーキテクチャの研究や論文の評価は，もしかすると他の分野での評価とは少し違うかもしれません．この研究会の運営に携わっているようなアーキテクチャ研究のプロが査読して良いと評価した論文は，一般の査読者も含まれているような電子情報通信学会の論文の価値よりも高いというくらいになるべきじゃないかと思うんです．以上思い出話とこれからの期待を申し上げました．

4. 田中英彦先生

きょうお話ししたいことは3つあります．第五世代の話，並列の話，プロセッサアーキテクチャの話です．

アーキテクチャにはいろいろなシーズがあって，それらを組み合わせて，何か新しいアーキテクチャを考えるのだと思います．たとえば次のような条件，仕様が考えられます．まず単純であること．スケーラビリティがあること．それから1チップで2000万ゲート，1億トランジスタぐらいは使える．これをうまく使いこなしていくアーキテクチャが欲しい．クロックは500 MHzぐらい．かなりの容量の主記憶あるいはキャッシュが内蔵化できる．それからキャッシングはimplicitだけでなく，コンパイラが静的に計算して明示化する．

難しいのはデファクト標準の話です．今インテルが世界で一人勝ちをして70数%のプロセッサがインテル製です．日本のメーカはほとんどギブアップというのが現状だと思います．しかし，日本のアーキテクチャ研究は今までとても良い仕事をやってきたというのが私の感想です．1つはデータフローの研究で，とてもいい研究がなされたと思います．ただ，それをうまく取り上げるタイミングがまだ来ていないのかもしれません．先程，飯塚先生がおっしゃられたように，そこで粘り強くやっていく必要があるのではないかと思います．

さて，第五世代プロジェクト，特にアーキテクチャについてはどういう意味があったのか．たとえばPIM（Parallel Inference Machine）を作りましたが，どう評価するのか．何人かのメーカの方がおっしゃるには，一生懸命PIMを作ったが同時に出てきたマイクロプロセッサの性能がやたら高いので，すぐにPIMからRISCに乗り替えたと．これが現状ではないかと思います．これでは「評価し次に備える」という重要な仕事が抜けていると思います．

並列処理という面から見たとき，たとえばCRAYなど商用のFORTRAN系のマシンは，逐次型言語からコンパイラで並列度を抽出して速く動かそうとしています．でも第五世代のマシンというのは一番極端に並列を出す言語からスタートしていて，そのままだとオーバヘッドが大きいので，うまく粒度を大きくして効率をあげようという方向です．オーバヘッドを減らすことと，いかに並列制御するかというのがポイントですから，FORTRAN系とは視点が違うわけです．どちらの方向が良いのかはまだ分かっていません．第五世代プロジェクトはせっかく細粒度のエンドから出発しているわけですから，どこまで行けるかという評価が出るまでやって欲しい．私どもの研究室でも小さなマシンを作ってそういう意味の研究を細々とやっております．

もう1つは並列の話ですが，ここにはたくさんの専門家の方がおられますのでやめまして，3つ目のプロセッサアーキテクチャの話に移りたいと思います．新しいプロセッサアーキテクチャを考えようという話です．プロセッサというのはデファクト標準が，ものすごい力を持っています．それに対して性能が1割高いとかいっても全然使ってもらえません．しかし，たとえば今のPentiumは元々86ベースです．それをずっと引きずってきています．ものすごい数のトランジスタを費やして結果として良い性能を出しています．が，私たちのなすべきことは，今Pentiumに勝つことではなくて，たとえば二千数年に良いプロセッサを出すことです．86ベースの古いアーキテクチャを引きずっていたのでは，これは重荷になると思います．そういう重荷なしにスタートしているきれいなアーキテクチャ，たとえばRISC，IBM系の601でも結構ですが，ああいう

ものと比べてみると，Pentiumには技術的に無理があるように思います．ですから，次のピークはPentiumの延長線上にはないと思います．次のピークを我々が作っていくことを，今からぜひ考えたいと思います．逆に言いますと，自由にアーキテクチャ研究ができるのは，今，日本が負けているからでして，気楽に勝手なアーキテクチャを考えてみたいと思います．

たとえばπを計算するという話から始めます．次のような非常に簡単な積分式でπが計算できます．

$$\pi=4\times\int_0^1\frac{1}{1+t^2}dt$$

まず最初に目的としてπが欲しいというのがあって，次にアルゴリズムを決定します．上の式を使おうと思ったのがアルゴリズムの決定の前半で，後半として，この積分のためにたとえばSimpsonの公式を使おうと決定します．そして上の式を展開すると2つのループができますが，その1つを機械語で書いてみるとこんな操作になります．

```
    h←0.5
    h←h/n
    x←h
    j←1
l1: x2←x*x
    x3←1+x2
    delta←4/x3
    sum←sum+delta
    j←j+1
    x←x+2h
    if(j≤n)goto l1
    halt
```

この計算で一番最初に出現するのは，変数jとパラメータhです．nは外から与えられています．計算のための新たな変数として$x2$や$x3$がありますが，これらはアルゴリズムを規定したときには不要なものでした．最終的には，和の値を求めるのがスペックです．しかしスペックを拡大して，$x2$，$x3$，*delta*も求めければならないという制約条件のもとに最適化をやってしまう．すると，どうしても最適化がグローバルに行われません．

まずアルゴリズムレベルの最適化に関して通常アルゴリズムを決めると並列度や速度も決まるという言い方をします．これに関しては，プログラム変換である程度アルゴリズムをいじることができるのではないかと思います．たとえば先程のループでも，N個の数を足せというのが最初のスペックですが，ループにしてしまった途端に順番に足すことになってしまう．そういう制約を入れてしまうところが問題なのです．最初に戻って，全部足すというスペックからプログラム変換をして機械語に置き換えれば，より良い最適化が図れるかもしれません．もう1つの最適化はループの融合です．先程の2つのループは実は融合すると断然性能が良くなります．そういう最適化がソフトウェア的に行われて，効率の良いプログラムを作るのがコンパイラの重要な役目です．

そういうことを考えますと，中間変数をできるだけ除きたい，アルゴリズムを自動的に変換したくなります．そういう要求を考え合わせたとき，これに向いた一番速い計算機はどんな構成かということで，理想的処理をデータフローグラフで書いてみます．これを一番速く計算するのは，恐らくデータフローグラフがそのままハードになったものでしょう．しかし本当は，そういう汎用的なハードを作るのは大変だから諦めて，従来方式では非常に汎用的な個々の機能を組み合わせて，この繰返しとして実現しています．では，従来のアプローチのオーバヘッドはどこから来るかといいますと，それはメモリの読み書きです．メモリに書いてまたメモリから読み出す，それをできるだけなくしたいわけです．間に高速なラッチを挟んだ演算器をたくさん並べればきっと速いでしょうから，こういうアーキテクチャを検討してみたいと思います．

では，そういうアーキテクチャが，次の図のような構造で実現できるかということについて考えてみます（**図-1**）．ここで処理装置と書いてあるのはデータフローグラフです．つまりデータフローグラフを自由に構成できるようなハードウェアをここに置いて，上からデータを流して何かしらメモリに書き込む．メモリに書き込んだデータを次のサイクルでまた処理装置に流す．データフローグラフの構成を与える処理指示と，データをメモリのどこに入れるかという出力指示，メモリから出てきたものをどこに流すかという入力指示が

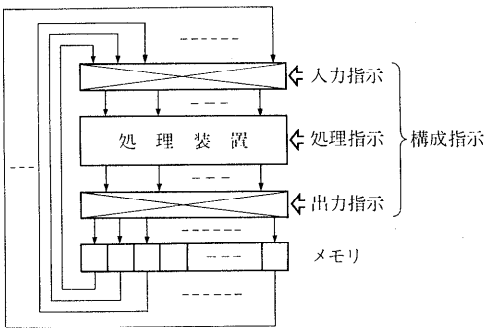


図-1　処理装置の全体構成

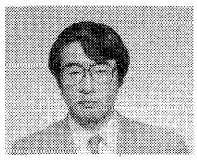
あります．これらを構成指示として与えながら回すというアーキテクチャです．現在の計算機とある意味で同じですが，違うところは，構成指示のオペレーションを完全に切り離して独立させるということと，処理装置の幅がやたら長いということです．たとえば2000万ゲートですと，おそらく100台以上の演算器が1チップに納まります．

データフローグラフの上をデータがワッと流れて，いろいろな場所からデータが出てきますが，考えるべきは，どの時間にどのノードからデータが現れるかということです．時間 T から $T+\Delta$ の間に動作するノードを計算してスケジュールします．スケジュールを立てるときの問題は，if文のために静的に決定できないことでしょう．しかし，そういうことを解決して，早く出力され次の操作で使われるデータは早いメモリに流して，もっと後で使われるデータは遅いメモリに流すようなスケジューリングをします．ですから，データフローグラフを用いてどこに流すかというのを予測を立てて計算しておく．メモリの方は，時間が来ると適切なデータを勝手に送り出して行くというスケジューリングをします．もう1つのポイントは構成指示です．構成指示は，何かしら条件判断が決まると，その結果をもとにデータフローグラフの開始可能時刻記憶を更新します．大部分はコンパイル時に作っておくんですが，更新することによって処理を動的にちょっと変更して次の構成指示を作ります．この指示を普通のALUをたくさん並べて，スイッチ網で結んだ処理装置に与えます．

このアーキテクチャは何も新しいことはありませんし，これが良いかどうかはまだ分からないのですが，これから我々が研究すべき1つの方向ではないかと思っています．なぜこういうアーキテクチャを言い出したかと言いますと，動機の1つは，スーパスケーラのように小さなプロセッサをたくさん集めて並列処理をするアプローチでは，処理の分割や同期など細かいところまできちんと制御しないといけないので，ソフトウェアに対する負担が非常に大きくなります．これをできるだけ小さくしたい．大きなデータフローグラフを流してゆっくり制御するという形のアーキテクチャだと，大規模並列になったとき，制御のためのソフトウェアが作りやすくなるのではないかと思っています．

素人の感想として，こんなアーキテクチャも考えられるということを述べさせていただきました．日本のコンピュータアーキテクチャのグループは今までとても良い仕事をしてきたと思います．繰返しになりますが飯塚先生がおっしゃったように，あと一歩の粘りが欲しいと思います．あと一歩の粘りをつけ加えて世界に冠たるマイクロプロセッサを出し，21世紀を支えるようなコンピュータを作って行っていただきたいと思っております．

5. 富田眞治先生



今日は，プロセッサアーキテクチャの話をしたいと思います．最近，部品を外国から買って集めてシステムを作って売ればよいという状況になっておりますが，こういうことをしていますと後進国がすぐ追いついてきますので，基盤的なところを押さえておかないといけないと思います．最近の日本の状況を見ると，田中先生のお話にもありましたが，メーカはまともにプロセッサを作らなくなりつつあります．唯一，スパコンのベクタユニットのところでは頑張っておりますが，それ以外のところは寂しい状況です．研究について見てみますと，昨年Hennessy and Pattersonの訳本を出版しましたが，その中の309件の参考文献のうち日本人の論文はただ1つ富士通のスパコンの論文だけでした．我々日本のアーキテクトはもう少し頑張らないといけないと思います．

将来を語るには過去を知らないといけないということで，ちょっと過去に溯ってみたいと思いま

す．アーキテクチャの研究は，その時代の技術上の制約の中でうまくシステムを作っていかないといけないわけです．そういう目で過去を振り返ってみますと，まず，50年代の技術上の制約は素子数なわけです．アーキテクチャはアキュムレータ方式が一般的でした．60年代の前半は，メモリ容量が制約で，それからスタックマシンが出てきます．1964年にIBMが360を出しました．互換性の問題，エミュレーションの問題などが出てきます．この頃はメモリ速度が問題でした．マイクロプログラムだけ速いメモリを使うということになってました．もう1つの制約はアセンブリ言語レベルでのプログラミングということです．

60年代の後半はIBM 360/91，CDC 6600といった命令パイプラインが出てきます．IBM 360/91の論文は今読んでも示唆に溢れていて，out-of-order実行などがありますが，製品としては割込みなどがうまく制御できなくて失敗しました．

70年代の前半は半導体メモリの高速化と大容量化が進み，IBM 370で仮想記憶が登場します．この頃私も計算機アーキテクチャの研究を始めました．そしてハンダ付けできるということで，助手候補として坂井利之先生から萩原宏先生に紹介していただきました．萩原先生にお会いすると先生はハンダの冷やし方，息のかけ方が大事だとかそんな話ばかりをして，研究の話は全然ありませんでした．ずいぶんのんびりした時代でしたが，だからこそ良い研究ができたともいえます．

1973年以降，Writable Control Storage (WCS) が出てきました．我々の研究における神様的なマシンはBurroughs B 1700とNanodataのQM-1でした．B 1700はCOBOL，FORTRAN，システム記述言語に向いた命令セットを持ち，それをマイクロプログラムで解釈実行するようになっていました．このようなWCSを使って高級言語マシンを作るというのが盛んになってきました．間接実行型の高級言語計算機もありました．この頃にバイブルともいうべき，MyersのComputer Architectureという本が出ます．当時の流行語は「セマンティックギャップを埋める」でした．コンパイラ不要とか，高級言語レベルでデバッグ可能とかいうことです．その頃ソフトウェア危機という言葉も出てきまして，ハードウェアで支援できることは支援するという雰囲気でした．ただ，こういう話をしますと京大の大野豊先生に，そういう発想ではソフトウェア危機は回避できない，ソフトウェアの危機にはもっと深い理由があるよと言われたことを思い出します．

この頃の研究は，メモリの高速化と，集積度の向上ということがあまり理解できていなかったように思います．当時は，マイクロプログラムの速度とメモリの速度のギャップを，高級言語マシンが間に入ることでうまくバランスしていたわけですが，キャッシュがチップに入ってマイクロプログラムと同じ速度で動くようになると，システム構成の方法が変わってくるわけでして，そのあたりの展望が十分にできていなかったように思います．

この頃，我々はVLIWの提案を行いました．1号機のQA-1は160ビット長のマイクロ命令で，ICはTTL（SSIとMSI）で5000個くらいです．作ったのは現在京都工芸繊維大学の柴山潔先生，東芝の小柳滋さんと私とで1974年から開発していました．この上でグラフィックスの実時間処理をしていました．さらに高速化を目指すためQA-2を作りました．マイクロ命令長256ビット，ECLとTTLで22000個くらいのマシンです．全部で20人くらいの学生が参加しました（主要メンバは，現京都工芸繊維大学の柴山潔氏，富士通北村俊明氏，河村武司氏，京大中島浩氏，NEC中田登志之氏，NTT山下博之氏，日立栗山和則氏，釜田栄樹氏，九大村上和彰氏など）．

このような研究の中で感じましたことは，まず概念の飛躍をしないといけないということです．当時，萩原先生はマイクロプログラムの権威でした．QA-1，QA-2を作るときに悩んだことは，採用した命令形式がマイクロ命令か機械語命令かということでした．マイクロ命令にしてはフィールド構成がきれいで，しかもそのマイクロ命令が制御記憶に入っているのですが，その容量が1K語しかないということで，主記憶にバックアップして仮想制御記憶を採用していました．そこまでやっていながら，これは機械命令だと言い切れなかったわけです．そこが限界だったわけです．機械命令であるとしていれば，違った研究ができたと思います．

もう1つはネーミングです．QAというのは

Quadruple ALUsから来ているのですが，もっと概念に密着した名前をつけるべきでした．イエール大学のFisherがVLIWという名前を付けたのですが，これは概念と名前がうまく合っているのです．名前をちゃんと付けられないというのは概念がまだ明確になっていないということです．

人材についても感じたことがあります．アクティブな人材を育てるには，細かい研究をしていては駄目です．大きなプロジェクトを組織して育てて行った方がよいと思います．QA-2の研究のとき，修士の学生で朝から晩までラッピングをしていた人が，不思議に偉くなってます．メモリ回りのラッピングばかりしていたある学生は，仮想記憶の研究という40ページの修士論文を書いて，これが本当の「仮想」記憶だと笑っていました．

コンパイラも問題です．これをあまりやりませんでした．当時はソフトウェア危機ということもあって，ハードウェアの機能をあげることをやっていましたから，コンパイラなんて要らないという誤った発想を持っていました．ソフトウェアパイプラインなどの発想もありましたが，うまく発展させられませんでした．やはり面白いなと思ったらとことんやってみるべきだったと思います．

1970年代の後半になるとベクタプロセッサが出てきまして，80年代になるとRISCが出てきます．RISCの台頭は，プロセッサとメモリが同じチップの中に入ることで速度が同じになってきたことが大きなポイントだと思います．

現時点では，1チップのベクトルプロセッサとか，64ビットのスーパスカラで4多重の命令実行，CISCの巻き返し，マルチプロセッサ対応などが出ています．現状は，スーパスカラとVLIWがほとんどで，命令の乱発行はほとんど入っていなくて，順発行で多重処理というのが一般的です．

これから近未来はどうなるかという話ですが，まずスーパスカラでは，乱発行，投機的な発行が出てくると思います．Power PC 604では乱発行が入っているようです．さらに速度をあげようとしますと，マルチスレッドにせざるを得ないと思います．そして，キャッシュの容量をあまり大きくしないで，ワーキングセットを広げないでうまくスレッドを切り換えるといった方向になると思います．

1チップベクトルプロセッサ，CISCプロセッサも面白いと思います．また，ユーザ特化プロセッサもマルチメディアへの対応ということで出てくると思います．並列処理の対応は我々も頑張ってやっておりますが，まだあまりニーズがありません．昨日，中澤先生（筑波大）から並列の次にどのような概念を構想すべきかというお話がありましたが，私は次は自己成長の機能を持つシステムであるような気がします．遺伝的アルゴリズムをやっている方々もそのようなことを言っています．FPGAのようなものを使うと回路が自動的に組めますので，そのあたりを発展的に研究して行ってもよいと思います．並列計算機の場合ですと，何十日もかかるような長大ジョブに対してはロボットが結線を変えるという昔のアナログコンピュータのような話もあります．そのような自律的な機構が欲しいと思っています．

最後に若い方への提言ですが，独創的な研究がないという点では概念形成とネーミング能力をもっと磨かないといけないと思います．また欧米の研究を無批判に受け入れているようにも思います．それから最近はシミュレーション評価つきの小さい研究が多いのですが，もっと革新的で評価データなどなしというような研究を行う必要がある．たとえばCRAYやBSPといった計算機では論文などはほとんどありません．良いアイデアやスマートなアーキテクチャの論文を，査読でもっと通して行ってもよいと思います．

それから，国際的な場での発表と日本人論文の相互引用を活発化しないといけないと思います．欧米の人間から日本の研究がよく見えていないと思います．また，実プロセッサ，商用プロセッサを作って行くべきです．今，基盤技術としてのアーキテクチャを育てないと将来禍根を残すことになると思います．

6. 島田俊夫先生

私の話は一言でいうと，もう少し並列計算機が使われるようになるにはどうしたらよいかという話です．

技術の流れを見てみますと，今はマイクロプロセッサを利用するという流れが一番大きいと思います．RISC並列ですと

か，コヒーレントキャッシュといったものが盛んです．一方ではデータフロー並列技術があって，最近はマルチスレッドに移っているという状況です．それから，ネットワークの研究が盛んで，昔では考えられなかったようなネットワークが実現されております．ICPP の過去3年間の論文の分類表を作ってみたのですが，ネットワークに関する論文が一番多かったです．それからソフトウェアも，新言語，モデル，関数型，論理型，オブジェクト指向などが出てきています．コンパイラ最適化の論文，スケジューリング関係ではデータ分割，分散といった話もたくさんあります．ここで言いたいことは，並列計算機もたくさん出ているし，並列計算機に関する技術的な論文もたくさん出ています．したがって計算機アーキテクチャ研究会に出ているような人は，並列計算機を使って仕事をしているはずです．ところが実際はそうではないわけです．現実を見ますと，商用の並列計算機は多種類になって安くなってきています．しかし少なくとも，どこかの機械が何万台も売れたという話は決して聞きません．

実は私の近くでも並列計算機が使えまして，来年になるともう2セットほど入ってきます．そうした計算機のユーザの方と話をしてみました．そうした中で，なぜ並列計算機が使われないかということを私なりに考えてみました．実はこれが本当のタイトルで「Empirical comments on parallel processing—なぜ並列計算機は使われないのか」です．まず私の近くにある並列計算機を見ていますと，それは並列研究用の計算機なんです．だれが使っているかというと，並列計算機屋さんが，ベンチマークプログラムを持ってきて数回実行して評価をして，もう使わない，というのが非常に多いのです．もちろん，ごく少人数のエキスパートは使ってはいます．どうして日常的に使うユーザがつかないのかと問うと，プログラムの開発が困難というのと，実行性能が実はあまりよくないという返事が返ってきます．それで私はこの話が本当だろうかと試してみようと思いました．

最初は実行性能の話です．Livermore loop の1番，7番，流体方程式の計算の性能を見てみますと，ほとんどのプログラムで1台より性能が悪いわけです．その理由はこれから明らかにしていきます．Livermore loop の8番ですと16台で4倍よりは下です．3番では16台で0.2倍です．あえて並列計算機に乗り換えるほどの性能ではないでしょう．次に，Livermore よりはもう少し実際的なプログラムとして，NAS パラレルベンチマークの多次元 FFT をやってみました．これはデータも大きいですし，プログラムも実用的な程度に大きいものです．これで見ますと7倍程度になっていますが，問題規模が小さいと16台で2倍程度になってしまいます．このような結果では多分普通の数値計算屋さんが並列計算機に乗り換えるメリットはないわけです．しかも，この程度の性能を出すにも，すごいプログラムを書かないといけないわけです．

私は並列計算機屋ですので，どうしたらよいだろうと考えてみました．まず通信時間を調べてみますと，1バイトずつ転送するときは40マイクロ秒です．この計算機は世の中の標準から見るとそんなに通信が遅いマシンではないのです．速い方の例としては，富士通の AP 1000 で17マイクロ秒，電総研の EM-4 が9マイクロ秒です．もし通信時間が非常に高速であれば計算速度が飛躍的にあがるかというと，そんなことはないわけです．今の計算が遅い原因は，簡単にいえば計算と通信のバランスがとれていないからです．今の計算機の演算速度は非常に速くて40マイクロ秒で転送している間にどれだけの計算ができるかということを考えると，これらの問題では明らかに転送しないほうが得です．これが主たる原因です．

次はプログラミングの問題です．Livermore loop の3番のプログラムは3行程度だったのですが，これがいかに大きくなってしまうのかが分かると思います．それでもこれなら，まぁなんとかなるという程度です．そこで，さっきの NAS パラレルベンチマークの3次元 FFT プログラムをどうやって作ったかと申しますと，まず X 方向に FFT をかけ，次に同じプログラムを使うために X，Y，Z を転置しまして，また1次元の FFT をしています．きわめて初歩的なやり方で，この方式では実行中にデータを移動させるプログラムを書かないといけません．このときプロセッサを意識して書かなければなりません．それでも転置くらいでしたらライブラリがあるかもしれませんが，この例ですと下が実部で上が虚部ですから，それを考えて書かないといけない．つまり転

置するだけでかなり長いプログラムで，しかもプロセッサを意識したプログラムを数値計算屋さんに書いてもらわなければならないわけです．しかし，結果としての性能は先程お見せした程度なのです．

これは，たまたま私の近くにある計算機のソフトが発展途上にあってこれからもっと良くなるという意見があるかもしれません．そこで，VPP 500 とか AP 1000 などのソフトがもう少し良いのではないかと思って調べてみました．VPP FORTRAN というのは，High performance FORTRAN とか，FORTRAN D の機能を取り込んでいまして，少しは行数が減ります．しかし，プログラムを最適化するためにはディレクティブが必要でして，これが簡単かというとそうは思えません．富士通の名誉のために申し上げますと，結果は劇的でしてリニアでスピードアップしますが，やはりこのプログラミングは大変です．数値計算機屋さんに自分のプログラムを全部このように書けというと使ってはくれません．学内の並列計算機を選ぶ委員会で，並列計算機屋さんは絶対ベクトル計算機を買うべきだと主張しています．私は並列計算機が売れないのには，少なくとも現状ではちゃんとした理由があると思っています．

まとめると当たり前のことばかりですが，まずプログラムの開発が困難であるということです．並列部分の抽出，同期と通信の挿入，通信時間に応じて粒度を変えてプログラムしなければならない．特に最後の点はマシンに強く依存してしまいます．それからデータ分散の問題です．最初のデータの分配は何かライブラリがありますが，途中でデータを動かすのは大変です．特にデータが均一でないと，よく考えないといけません．実際にやってみて，頑張れば良い結果が出るプログラムもありますが，それは一般的ではないように思われます．良い結果を出すためには，プロセス数，通信遅延，粒度などを考えなければならないのですが，それを一般ユーザに要求するのは難しいと感じました．

ではどうしたらよいかということを考えてみましょう．私が研究してきた関数型言語 DFC で並列プログラムを書けば，同期や通信はユーザが書く必要はありません．文法は C 言語に準拠しているので，逐次型の C プログラムを書くように記述できます．もう 1 つ良かった点は，データ分散をコンパイラで静的に行い，後の通信はシステムの高速な通信に任せたことです．こうすればプログラミングは楽になります．今後はソフトウェアで先行的にデータ移動を行い，それを計算とオーバラップさせるという方法も使えるでしょう．

このような自動並列処理言語をデータフロー計算機 SIGMA-1 で実行したときのプロセッサ使用効率は，大体 17%くらいでした．その後開発された EM-4 のようなマルチスレッド型計算機ではパイプラインの使用効率が高いためもっと良い性能が出せると思います．

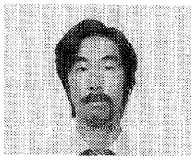
今までの並列計算機の研究は性能が優先されていて，リニアな性能向上のためには，多少のプログラミングの苦労は我慢していました．そういう限界を追求する研究ももちろんよいのですが，システムとして見ますと，容易に使えてベストではないがある程度の性能が出せるようなシステムが，これからは必要だと思います．そのためには，プログラミングは手続き型言語からの移植がきわめて簡単，場合によっては自動的に変換できるような言語で，並列化は完全に自動化しないと駄目だと思います．

データ分散/分割は共有分散メモリモデルにしてユーザが意識しないようにしなければなりません．物理的な実現方法は別ですが，ユーザからはそう見えなければなりません．もう 1 つは，粒度の問題です．実は私のところで，他にもいくつかプログラムを書いてみました．特に音声認識グループと交流をしながら，音声認識系のプログラムをいかに速く走らせるかという研究を開始しました．しかし，このプログラムは台数効果が 1 を超えることがないという性能でして，このために挫折しかかっています．現在使用している商用の並列計算機は通信が遅いので，通信がちょっとでも入るとすぐに 1 台の計算機に負けてしまう状況です．性能を出そうとすると，ループを 10 万回程度は回す必要があります．私たちの身の回りのアプリケーションでは，そういう粒度の大きな計算はほとんど見当たりません．したがって，もっと細粒度並列を効率よく実現しないと応用範囲が狭い，計算する問題がないということになります．

こういう方向で行けば並列計算機の将来は多少

明るくなるのではないかと思います．

7. おわりに（中島浩先生）



以上紹介したように，研究会設立以前も含めたアーキテクチャ研究の歴史から，今後進むべき方向に至るまで，「昨日，今日，明日」について様々な視点からの講演であった．特に印象的であったのは，過去の研究に対する反省も含めた率直な評価に基づき，未来へ向けての様々な提言がなされたことである．これらの提言には，これまで日本の研究をリードしてきた立場からの重みがあり，会場を埋めた若い研究者達にとって得るものが大きかったのではないかと感じた．

150回あるいは200回は次世紀に迎えることとなるだろうが，それまでに日本のアーキテクチャ研究がさらに発展し，新しい「昨日，今日，明日」が再び語られることを期待したい．

（平成7年1月19日受付）