計算機設計とマイクロプログラミング

相　磯　秀　夫†

1. まえがき

もともと，マイクロプログラミングは計算機における制御回路の理想的な設計手法として M. V. Wilkes によって提唱されたものである．しかしながら，今日におけるそれは制御回路の論理設計のみならず，ソフトウェアや将来の計算機のあり方にまで大きな影響を与えようとしている．マイクロプログラミングの意義は単に狭い視野からのみ評価すべきでないが，ここでは計算機設計を広義にとらえ，その観点からマイクロプログラミングの考え方に対して簡単に私見を述べる．

2. マイクロプログラミングの位置

計算機を論理構造という観点からとらえると，次のような階層が存在すると見ることができる．ただし，{ } は集合を表わすものと解釈する．

（1） {論理回路}，
（2） {サブユニット，ゲート，タイミング}，
（3） {基本操作}，
（4） {機械命令}，
（5） {アセンブラ}，
（6） {コンパイラ}，
（7） {ユーザ・プログラム}．

今までの計算機では，アーキテクチャが設定されると，先ず（4）が固定化され，それに伴って（1）～（3）の，いわゆるハードウェアが設計される．（3）は種々の機械命令に共通で，最も単純，かつ基本的な命令の集合であり，（2）は機械命令を論理回路で実現する際に用いる一般化した機能モジュール群とそれらの間のパスを時間関係を考慮しながら定義したものである．また，（4）～（6）をソフトウェアとしてユーザに公開する立場をとるのが普通である．これに対して，（2）または（3）をどの計算機にも共通になるように，機能的に一般性をもたせ，そのレベルでユーザ（または設計者）にハードウェアを公開する立場をととのえるのがマイクロプログラミングであると考えられる．したがって，（4）以下のレベルはユーザが自由に定義することができ，しかも階層を一つにする可能性を与えている．従来の配線論理（Wired logic）による計算機の（4）がそうであったように，マイクロプログラム計算機では（2）または（3）がハードウェアとソフトウェアのインタフェースになっている．特に（2）は水平型，（3）は垂直型マイクロプログラミングに相当するものと考えられる．

3. 設計から見た特徴

設計という意味を方式設計を含めて，広義に解釈し，マイクロプログラミングの特徴を列挙すれば次の通りである．先ず利点と考えられるものは，

（1） 詳細な方式設計が決まらなくても，論理設計を進めることができる．
（2） 複雑な制御論理に規則性が与えられ，設計・保守・調整が非常に容易になる．
（3） ハードウェア細部のプログラム制御が可能で，機能に融通性が出ると同時に，機能の変更・追加が容易に行える．
（4） 基本的なシステム構成が単純，かつ一般的で，計算機システムとしての寿命が長くなる．これは半導体などの技術進歩にシステム構成などが大きな影響を受けず，必要に応じてハードウェアの実装だけを変えればよいことを意味している．
（5） ハードウェア内部の構造が規格化され，標準化が行い易い．特に制御装置とそれ以外の装置（演算装置など）を論理的にも物理的にもきれいに分離でき，しかも LSI 化が困難な制御回路の大部分をマイクロプログラムの中に吸収できることを考えれば，計算機の LSI 化に適した方式といえる．また，入出力インタフェースをはじめ，種々の信号線の解釈をマイクロプログラムで適当に行わせることを考えれば，インタフェースの標準化を促進することも可能である．このように，マイクロプログラミングは計算機の標準化の有用な手段といえる．
（6） 診断機能を容易に盛込むことができる．故障診断のためのハード・コアが少くてすみ，しかも故障検出率が高くなる．

† 慶応義塾大学工学部電気工学科

(7) エミュレーションにより既存のプログラムの活用が容易になるため，新しいシステム開発もらくになる．

(8) ファームウェア化によりデータ処理の効率，価格性能比の改善が図られる．特に主記憶へのアクセスが少なく，複雑な深い論理操作を必要とする機能のファームウェア化は効果が大きい．

(9) ユーザの使用環境に適したシステム作りができるような配慮が払える．これは動的（ダイナミック）マイクロプログラミングの普及を意味する．

(10) 計算機，特にハードウェア設計の教育が容易になる．説明が難かしい制御装置が単純になるため，方式設計や論理設計に携わる者，保守員には大変都合がよい．また，マイクロプログラミングを介して，配線論理とマイクロプログラム論理の一対一対応，ハードウェアとソフトウェアの設計の同一性などが容易に理解できる．

ことなどであろう．一方，欠点としては，

(1) マイクロ命令（特に垂直型マイクロ命令）の処理が比較的逐次的で処理効率が悪くなる．

(2) 処理速度がマイクロプログラム記憶の動作速度待ちの状態であり，高速計算機に不向きである．したがって，大型機では高速処理が要求される一般的な基本命令は配線論理で実現され，深い論理操作を必要とする命令（変換・編集など）や診断機能などはマイクロプログラム論理によってファームウェア化されるものと予想される．むしろ，小型計算機やミニコンピュータの分野で先ずマイクロプログラミングの有用性が発揮されるものと考えられる．

(3) マイクロプログラミングを行うためには，若干ハードウェアの細部を理解することが必要で，高級言語レベルのユーザには困難がある．

ことなどであろう．

4. 今後の課題

マイクロプログラミングの実用化はここ数年来急速に進展し，それがもつ潜在的な能力も次第に明らかになっている．ここでは計算機設計という観点から今後に残された課題をいくつか挙げる．

(1) マイクロ命令

新しい計算機の開発に際して，マイクロ命令が備えるべき機能の設定は重要な問題である．この設定は開発する計算機の目的によって若干異なってくる．特定な機械命令をもっている場合は，それに見合った機能さえ準備すればよいが，エミュレーションを主目的とした計算機（MLP-900, QM-1 など）では，機能的な完全性 (functional completeness) を配慮しなければならない．ところが，論理回路レベルよりもマクロな機能における完全性と効率的な機能との関係は必ずしも明確ではない．しかも，これらはハードウェアの基本構成にも密接な関連があり，今後の経験に待つところが多い．

マイクロ命令の型式を論理設計のし易さから見れば，垂直型が望ましいが，処理速度という立場からは並列操作が可能な水平型がよい．論理設計の専門家には後者が好まれようが，マイクロプログラム記憶容量，ビット利用率などの点からは問題が多い．現状では，垂直型またはそれに近いマイクロ命令を設定することが普通になりつつある．処理速度の改善は半導体技術の進歩に期待している．結局のところ，経済的にどこまで水平型に近づけられるかが一つの技術的な課題といえる．

大型高速計算機における並列処理，パイプライン処理，ならびに連想処理制御を行うためにどのようなマイクロ操作を準備すべきかもこれからの重要な課題である．

(2) 動的マイクロプログラミング

当然のことながら，半導体技術の進歩に伴って，動的（ユーザ）マイクロプログラミング方式が常識化するものと考えられるが，この場合はハードウェア技術の他に，マイクロプログラミングをユーザに公開する方法が問題になる．先ず第一に，一般のユーザが容易にファームウェア作りができるように，マイクロプログラム・アセンブリ言語や B-1700 に見られるようなMIL (Microprogram Implementation Language), SDL (System Development Language), およびそれらのシミュレータなどを開発することが急務になる．このことはユーザ・プログラムの処理効率を考えてもいえることで，将来は今までのような機械命令レベルの中間言語をもたずに，マクロな命令から直接マイクロ命令に変換されることが多くなると思われる．このような意味でもマイクロプログラムの記述言語の開発は重要なものである．

ユーザ・マイクロプログラミングは既存のオペレイティング・システムにも密接な関係があるが，両者の

間をいかに容易に関連づけるかが現実的な問題であり，これらの問題が解決されるにつれて，ユーザ自身の応用に適したシステム作りが可能になるものと思われる．

（3） ハードウェア構成

IBM System/360 に見られるように，マイクロプログラム計算機ではレジスタ群を共通バスを介して接続する構成をとることが多い．これはサブユニットの標準化，LSI 化につながり，しかも機能の拡張，実装などが行い易い特徴を生む．また，見通しのよいマイクロ命令の設定にも役立ち，故障診断，保守が容易になる利点が出てくる．したがって，今後共バス構造方式の採用が常識になると考えられるが，データ・パスが長くなり，処理速度が低下する欠点と，LSI 化に際して分割した場合に接続点が増える心配が起ってくる．

一般に，マイクロプログラム制御の効果を最大限に発揮させるためには，可能な限り主記憶をアクセスしないことが肝要であるといわれる．このような意味ではレジスタを多数備えることが望ましい．これは特にエミュレーションを配慮する場合には重要なことである．

しかしながら，最近の半導体記憶の著るしい進歩に伴って，主記憶とマイクロプログラム記憶との動作速度に大きな差がなくなる傾向にあり，前述のような処理効果は期待できなくなる傾向もある．したがって，高速性が要求される計算機では配線論理の併用やマイクロプログラム記憶に小容量高速バッファ記憶を設け高速化を図ることも検討されると思われる．

（4） 入出力制御

中央処理装置の制御回路にはマイクロプログラミング方式が完全に適用され，その有用性も理解されているが，マイクロプログラミングのよさを入出力制御装置に活かすこともこれからの課題であろう．今まで入出力制御装置に積極的に活用されなかった理由は，入出力制御方式が中央処理装置における制御方式ほど機能的に標準化されておらず，用途別に特別な制御装置を開発していたからであろう．しかしながら，マイクロプログラミングがもつ汎用的な機能から推察しても，今までの方法は必ずしも得策でなく，今後は恐らく汎用マイクロプログラム入出力制御装置（一種の汎用小型入出力計算機）が活用されるものと考えられる．この傾向は既に IBM System/370/125 などにも表われている．

入出力インタフェースの標準化についても全く同じことがいえるように思われる．すなわち，適当な信号線を準備し，信号の意味をマイクロプログラムで解釈し，多様性に富む入出力機器の制御を統一的に行うことが考えられる．この考え方は入出力装置のみならず各種のインタフェースの標準化に有用であろう．

（5） 適応機能

マイクロプログラミングの大きな特長の一つはハードウェアの機能に見かけ上可変構造性を与えることである．したがって，計算機の使用環境に応じて，動的に最適性能を発揮するような計算機の開発も不可能ではない．現に，一部の故障の回避をマイクロプログラムの reprogramming によって行なったり，計算機複合体における故障計算機の切離し，再構成などもマイクロプログラミングの助をかりて行う試みもある．また，多種類のプログラミング言語において，それぞれの処理効率がよりよくなるように計算機自身の機能が動的に変わるようなシステムの開発も期待できる．

いずれの場合も，既存の大型計算機よりも小規模な計算機システムにおいて研究成果が実るものと思われるが，その成果の影響は測り知れないものがある．仮りに，計算機が適応性や学習機能をもつようになったとすれば，その実現には動的マイクロプログラミング方式が大役を果しているものと予想される．

5. あとがき

マイクロプログラミングについて，計算機の設計という広い観点から眺めてみた．マイクロプログラミングはやっと実用の域に入り，種々の問題点が指摘されている．この分野は今後急速に進展し，マイクロプログラミングの潜在的な能力が発揮されるものと思われる．マイクロプログラミングをとりまく興味ある話題も数多く提供されるものと期待される．

（昭和 48 年 4 月 18 日受付）