

論文 / 著書情報
Article / Book Information

題目(和文)	マイクロコード化 L S I プロセッサの高性能化及び高機能化に関する研究
Title(English)	
著者(和文)	前島英雄
Author(English)	Hideo Maejima
出典(和文)	学位:工学博士, 学位授与機関:東京工業大学, 報告番号:乙第1621号, 授与年月日:1986年11月30日, 学位の種別:論文博士, 審査員:
Citation(English)	Degree:Doctor of Engineering, Conferring organization: Tokyo Institute of Technology, Report number:乙第1621号, Conferred date:1986/11/30, Degree Type:Thesis doctor, Examiner:
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

マイクロコード化LSIプロセッサの
高性能化及び高機能化に関する研究

昭和61年4月

前 島 英 雄

目 次

まえがき	1
第1章 序 論	3
1.1 まえがき	3
1.2 マイクロプロセッサの動向	4
1.2.1 L S I技術の動向	4
1.2.2 プロセッサの形態	6
1.3 マイクロコード化マイクロプロセッサ	10
1.3.1 構成法	10
1.3.2 技術課題	12
1.4 まとめ	17
1.5 第1章の参考文献	18
第2章 マルチチップ形L S Iプロセッサのチップ分割法	20
2.1 まえがき	20
2.2 制御用L S Iプロセッサの概要	21
2.2.1 位置づけ	21
2.2.2 要求仕様	23
2.3 チップ分割プロセッサ構成法	24
2.3.1 チップ分割法	25
2.3.2 並列処理動作	29
2.3.3 高速回路構成	32
2.4 R A S機能	35
2.4.1 エラーの種類	36
2.4.2 エラー処理方法	37
2.5 L S I構成と性能評価	39
2.6 まとめ	42
2.7 第2章の参考文献	43
第3章 高集積・高速マイクロコンピュータの構成法	45

3.1	まえがき	45
3.2	システムオンチップ形マイクロコンピュータの概要	45
3.2.1	利用形態から見たチップ構造	45
3.2.2	カーネル・プロセッサ構成上の必須事項	47
3.3	プロセッサの命令制御部構成	49
3.3.1	マイクロプログラム制御方式	49
3.3.2	命令解読手順とマイクロプログラミング	55
3.3.3	性能及びサイズの評価	60
3.4	マイクロプログラム圧縮法	62
3.4.1	2レベルマイクロプログラム方式	62
3.4.2	マイクロ命令合成方式	64
3.5	プロセッサのデータ処理部構成	69
3.5.1	バス指向のデータパス構成	69
3.5.2	マイクロレベルのパイプライン制御	71
3.6	LSI構成と機能	75
3.7	まとめ	78
3.8	第3章の参考文献	79
4章	アプリケーション適応形プロセッサの可変構造化構成法	80
4.1	まえがき	80
4.2	アプリケーション適応形プロセッサの概要	81
4.2.1	可変構造化の概念	81
4.2.2	可変構造化プロセッサ構成法	82
4.3	命令制御方式	85
4.3.1	命令制御部構成法	85
4.3.2	命令解読定義テーブルの構成法	88
4.4	レジスタ制御方式	90
4.4.1	レジスタ制御部構成法	90
4.4.2	レジスタ構成定義テーブルの構成法	94

4.5	LSI構成と機能	95
4.6	まとめ	98
4.7	第4章の参考文献	98
5章	専用プロセッサの機能分散化マルチプロセッサ構成法	99
5.1	まえがき	99
5.2	グラフィック処理専用プロセッサの概要	99
5.2.1	グラフィック・システム構成	99
5.2.2	要求仕様	101
5.3	マルチプロセッサ構成法	106
5.3.1	機能分散構成法	107
5.3.2	階層論理構造	113
5.4	LSIの構成と性能評価	114
5.5	まとめ	119
5.6	第5章の参考文献	119
第6章	結 論	122
謝 辞		124

まえがき

半導体技術に支えられたマイクロエレクトロニクス時代の中核を成すマイクロコンピュータも10年以上を経過し、その規模も4, 8, 16, 32ビットと急速な発展を遂げてきた。これらの中には、ミニコンピュータの性能領域を侵蝕するものもあり、システムへのインパクトは非常に大きい。このようなマイクロコンピュータの進歩は、特にMOS (Metal Oxide Semiconductor)技術の進歩によるところが大きい。MOSデバイスは、年率20%程度の早さでその寸法が微細化されてきており、各々の時点の集積度に見合ったマイクロコンピュータが開発されてきている。どの時代にあっても共通的なことは、その時代のシステム・ニーズがより高集積な半導体チップ、言い換えれば、高性能かつ高機能なマイクロコンピュータを要求していることである。この事は、その時代の半導体技術で実現し得る最高の性能を引き出し、より多くの機能をいかにマイクロコンピュータに盛り込むかが重要な課題となっている。即ち、マイクロコンピュータの内部構成(マイクロチップ・アーキテクチャ)に多大な努力が傾注されている。

本論文は、このようなマイクロコンピュータの設計・開発上で直面する高性能化及び高機能化に関してプロセッサ構成法(Implementation)の観点からの問題に検討を加え、その解決策を示したもので次の内容から成る。

第1章では、先ず、LSI (Large Scale Integration ; 大規模集積回路)技術の動向を示し、これが半導体チップ上に作られるプロセッサの形態へ与えるインパクトを述べる。次に、ミニコンピュータ、大型コンピュータを始めとして、プロセッサ構成法の主流となっているマイクロプログラム制御のマイクロプロセッサ(マイクロコード化LSIプロセッサ)の構成法とその技術課題を挙げる。

第2章では、制御用ミニコンピュータのLSI化を取り挙げ、ハードウェア量が多く、1チップ化の制限を越える場合に必要となる論理を複数のチップに分割する手段を示す。ここでは、複数チップ(マルチチップ)への分割によって問題となる性能低下を解決する分割法を提案した上で、回路構成を含めてプロセッサの高性能化を検討している。更に、制御用マイクロプロセッサとして重要な高信頼化についても示す。

第3章では、1チップ上にプロセッサだけでなく他の多くの機能を集積するシステムオンチップ形マイクロコンピュータを取り挙げ、このマイクロコンピュータの核となるプロ

セッサの構成上必須な高集積化(高機能・超小形)、高速化を達成する構成法を提案する。命令デコーダを省略したマイクロプログラム制御方式、マイクロプログラム圧縮方式、マイクロレベルのパイプライン制御方式を提案し、それらの効果について考察を加える。

第4章では、多様化するマイクロコンピュータ応用に対処するプロセッサ構成法の1つの解決策としてアプリケーションに適応し易い可変構造化設計法について述べる。ここでは、マイクロプロセッサの命令体系あるいは命令コードに束縛されない普遍的なマイクロチップ・アーキテクチャとして、命令制御部及びレジスタ制御部にアーキテクチャを定義するテーブルを設ける構成法を提案する。この方法を実際に適用したシステムオンチップ形マイクロコンピュータの構成と機能について述べる。

第5章では、アプリケーションに特化して性能・機能向上を図ることを目的とした特殊プロセッサの構成法を述べる。ここでは、グラフィック処理専用プロセッサをビークルとして、多機能の専用プロセッサ構成法として有効な機能分散化マルチプロセッサ構成法を提案し、これによって得られる性能及び機能の向上について述べる。

第6章では、本論文の結論を要約した上で、マイクロチップ・アーキテクチャの今後の展望を述べる。

第1章 序 論

第 1 章 序 論

1.1 まえがき

近年、広範な分野でエレクトロニクス化が進んできた。このDriving Forceとなったのは、メモリとマイクロプロセッサである。特に、マイクロプロセッサは種類が豊富で、アプリケーションに応じて選択され、活用されている。第1の応用は、パーソナル・コンピュータ、ワードプロセッサ、ワークステーション等のデータ処理プロセッサであり、第2の応用は、プリンタ、自動車エンジン、ロボット等の機器制御用のコントローラである。これらはいずれもその分野での目的に適したビット長、命令体系及び入出力インタフェースがとり入れられている。一方、マイクロプロセッサを研究・開発する側から考えると、その時点のLSI技術によって性能・機能に制限が加えられる。即ち、集積度、チップ・サイズ、半導体チップを実装するパッケージのピン数等の制約から1チップに集積し得る論理規模には制限が生じてしまう。こうした制限を意識しながらマイクロプロセッサの内部構成(マイクロチップ・アーキテクチャ)を決定してゆくため、LSI技術とマイクロプロセッサ構成技術との間には常に密接な関係が保たれている。

LSI技術の進歩に伴う集積度の向上は、次の3つの方向へ向けられている。第1の方向は、マイクロプロセッサのスループットの向上、即ち、4, 8, 16, 32ビットといったビット長の拡大に向けられる^{1)~7)}。この方向の発展過程では、命令機能向上やミニコンピュータ及び大型コンピュータで実証されているマイクロプログラム制御、パイプライン制御、仮想記憶制御等の高速化・高機能化のための手法⁸⁾も着実にとり入れられてきている。第2の方向は、プロセッサのみならずメモリやタイマ等の周辺機能を同一チップ上に集積するシステムオンチップ形マイクロコンピュータ⁹⁾に見ることができる。更に、第3の方向は、従来、主プロセッサによってソフトウェア処理していた特殊分野、例えばグラフィック制御^{10), 11)}、イメージ処理¹²⁾、通信制御¹³⁾等の機能をLSI化(ハードウェア化)した専用プロセッサに見ることができる。いずれの方向でも、高性能化及び高機能化という基本的な目的を達成するためのマイクロチップ・アーキテクチャがLSI構成上、重要な役割を果している。

本章では、マイクロプロセッサの動向とその中で主流の構成法となっているマイクロコールド化LSIプロセッサの構成法及び技術課題について述べる。

1.2 マイクロプロセッサの動向

1.2.1 LSI技術の動向

LSI技術の進展に伴うマイクロプロセッサの3つの形態への展開について前節に述べたが、その根元となっているLSI技術^{14), 15)}における集積度と性能の動向について述べる。

図1.1(a)にはマイクロプロセッサの集積度動向を、図1.1(b)にはメモリのそれを示してある。メモリの場合には、4~5年に4倍、即ち、16K、64K、256K、1Mビットといった一定の集積度向上が図られている。この向上は半導体の微細化技術の進展と完全に歩調を合せている。これはメモリが最新の半導体技術のDriving Forceとなっているからである。現在では1MビットのDRAM(Dynamic Random Access Memory)、256KビットのSRAM(Static RAM)の製品化段階であり、MOSトランジスタのゲート長は1.2~1.5 μm のレベルにある。研究段階ではサブミクロンのレベルである。

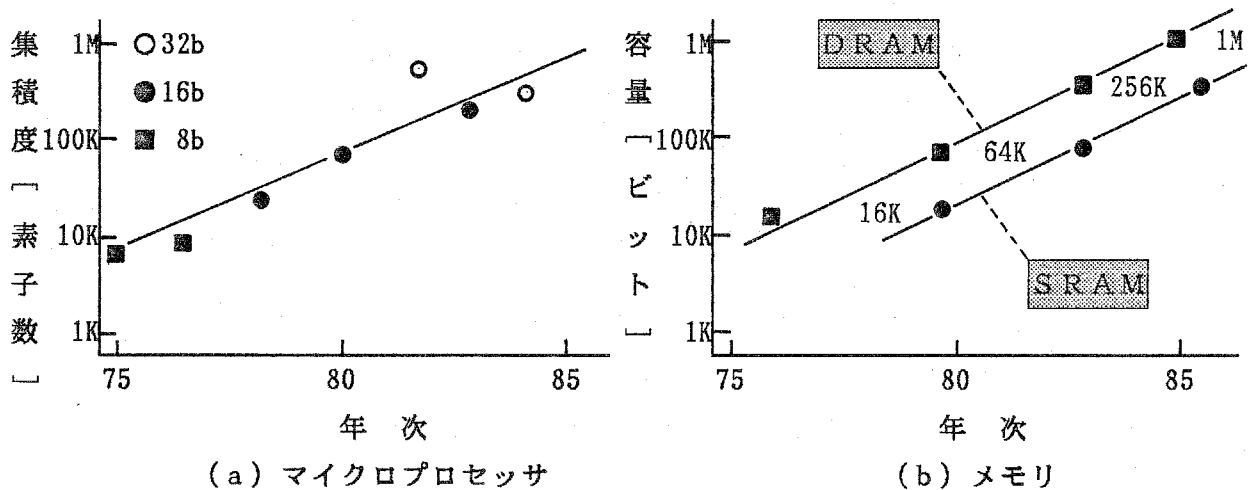


図1.1 LSI技術の動向

一方、マイクロプロセッサの場合、メモリに使われたLSI技術に対して2~3年遅れで追従しているのが一般的な傾向である。これは、マイクロプロセッサでは、

- (1) 演算回路を始めとして多くの要素回路が必要であること
- (2) 仕様がメモリのようには明確ではなく、論理の構成方法に自由度が大きいこと
- (3) 集積度の向上が性能や機能など論理を複雑化する方向に働くこと

等々の理由によって、設計に多大な時間を要することに帰因している。現在ではメモリの

1世代前の256KビットDRAMに使われた2 μ m技術が中心となっている。図1.1(a)に示したように、ほぼ1.5倍/年の集積度向上が図られていることになる。この傾向から判断すれば、1987~88年には100万以上の素子を集積するULSI(Ultra Large Scale Integration)マイクロプロセッサの出現が予測される。

性能について見ると、メモリの場合にはアクセス時間あるいはサイクル時間にその傾向をとらえることができる。DRAMは、コンピュータ・システムにおける主メモリへの用途が多いため集積度向上が中心であり、性能向上は顕著には見られない。SRAMは、コンピュータ・システムにおけるWCS(Writable Control Storage)、キャッシュ・メモリ、アドレス変換テーブル(TLB; Translation Lookaside Buffer)等の高速応用分野もあるため、高速性に焦点を合せた製品形態もある。アクセス時間では、DRAMで100~150ns¹⁶⁾、SRAMで35~50ns¹⁷⁾の製品が得られている。

これに対して、マイクロプロセッサの性能は、システムとしての総合性能という見方をしなければならないが、プロセッサ単体としては命令の実行性能で評価することが多い。よく使われる評価としては、システム管理プログラム(OS; Operating System)の実行性能を示すSystem Mix、プロセッサ単体の性能を示すEDNミックス、浮動小数点演算を含む科学技術計算の実行性能を示すGibson Mix、ミニコンピュータの性能評価に使われるWhetstone Benchmark等がある。

図1.2はマイクロプロセッサを動作周波数の観点からその動向を整理したものである。着実に性能向上が図られていることが判る。

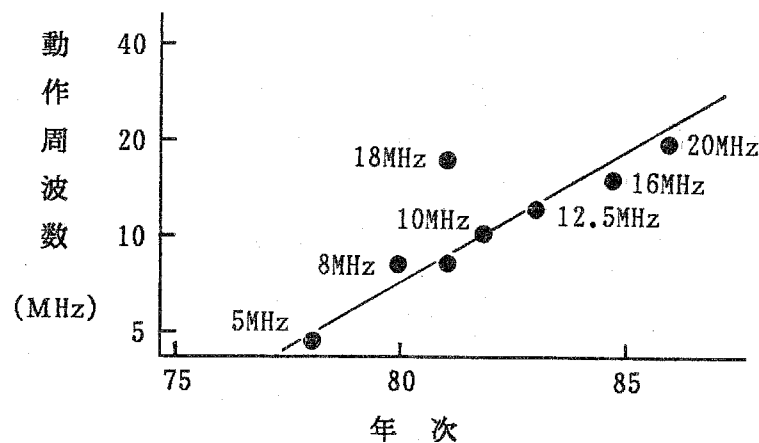


図1.2 マイクロプロセッサの動作周波数動向

よりハードウェアの性能を示すものとしてマシンサイクル時間がある。これは、動作周波数に比例するものであり、マイクロプロセッサの最小処理単位の実行時間である。マイクロプロセッサの高集積化と共に、動作周波数の向上も着実に図られていることがわかる。これだけで全ての性能を議論できないが、ハードウェアの基本的な実力を評価できるものである。

1.2.2 プロセッサの形態

マイクロプロセッサは、1チップ上に集積される機能の違いから大別すると図1.3に示すように4種の形態に分けられる。

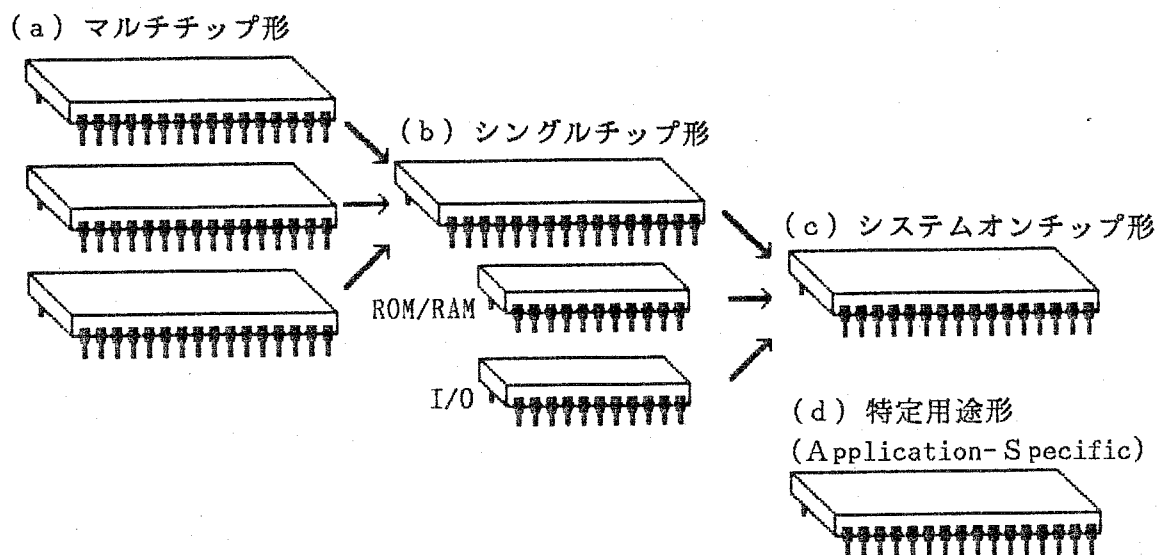


図1.3 マイクロプロセッサの形態

(1) マルチチップ形

1つのプロセッサを複数の半導体チップで構成する形態^{18), 19)}。マイクロコンピュータよりも機能的に高いミニコンピュータのローエンド機をLSIで実現する際に、その時点の集積度の制約から1チップ化できない場合、この形態をとることが多い。ミニコンピュータは汎用マイクロプロセッサの性能・機能向上によって、常に上位へと追いつけられている。従って、その論理規模は増加の一途をたどっており、汎用マイクロプロセッサと同じLSI技術では1チップ化は困難である。また、拡張性を重視することが多く、後述するマイクロプログラム用のメモリを外付けするため、積極的にマルチチップ構成をとる場合もある。

(2) シングルチップ形

1つのプロセッサを1つの半導体チップで構成する形態^{2)~8)}。この形態は最も一般的なものであり、主に汎用化を目指して高性能化と高機能化を追究してきている。この形態では、4ビットに始まり現在では32ビットに達している。集積度の向上がビット長の増加、即ち、システムのスループットの向上に向けられている。

(3) システムオンチップ形

1つの半導体チップ上にプロセッサと周辺I/O機能を同時に集積した形態^{9)~21)}。この形態は集積度の向上をシステムオンチップ(System Integration)に向けたものであり、パーソナル・コンピュータ等のシステムの実装規模を限界まで小形化しようとする意図がある。そのため、周辺I/O機能としてはプログラム・メモリやパターン・メモリとして用いるROM(Read Only Memory)、プログラム実行中のデータを一時記憶するRAM、計時を行うタイマ/カウンタ、プロセッサ間通信や入出力機器との通信を行うシリアル通信インタフェース、メモリ~メモリ間あるいはメモリ~I/O間の高速データ転送を実行するDMA(Direct Memory Access)コントローラ、A/D(Analog-to-Digital)変換器、クロック発振器など様々なものがある。これらの機能はマイクロコンピュータの用途に応じて選択され、プロセッサと共に1チップ上に集積される。従って、応用面から見ると、システムオンチップ形はシングルチップ形に比較して専用色が濃い。核となるマイクロプロセッサは、高速・高機能だけでなく小形化の必要性が高い。

(4) 特定用途形(Application-Specific)

1つの半導体チップ上に特定された機能を集積し、システムの主プロセッサの役割の一部を分担する形態^{10)~13)}。この形態は、シングルチップ形のマイクロプロセッサがソフトウェアによって処理するか、ビットスライス形のバイポーラ・マイクロプロセッサによる専用ハードウェアで構成していた部分、例えば、グラフィック、イメージ、通信等の各機能の一部あるいは全部をシングルチップ化したものである。マイクロプロセッサによって構築されるシステムが高性能化、高機能化した近年になって特に発展してきた形態である。高性能化が最大の課題である。

以上の4つの形態のマイクロプロセッサは、それぞれ目的とするところが異なり、それぞれに適合したプロセッサ構成が要求される。

一方、マイクロプロセッサの形態を論理方式の観点から見ると、ランダム論理方式と規

則論理方式に2分できる。

(1) ランダム論理方式

マイクロプロセッサのマイクロチップ・アーキテクチャは、半導体技術とコンピュータ技術の接点であり、LSIに向けた論理方式の必要性が高い。図1.4は一般的なマイクロプロセッサの構成を示したものである。プログラムを格納する主メモリ①から命令語が読み出され、これが命令レジスタ②に置数される。この内容は命令解読回路③によって解読され、命令制御回路④により命令実行回路⑤を動作させる。命令実行回路⑤は、加減算などの算術演算と、AND/ORなどの論理演算を実行する演算回路ALU(Arithmetic and Logic Unit)と演算レジスタ群から成る。集積度の低い段階ではチップ上に搭載し得る論理規模が小さくなるため、特に命令制御回路④には実装効率が高いとされるランダム論理方式(あるいは布線論理方式とも呼ぶ)がとられていた。ランダム論理方式は、各種のゲート回路を配線により接続して所望の論理を得る方式であり、むだのない論理を構成できると考えられていた。ところが、集積度の向上に伴ってマイクロプロセッサの機能向上が図られるようになった結果、論理が複雑化し、実装効率と設計効率の両面で問題がでてきた。

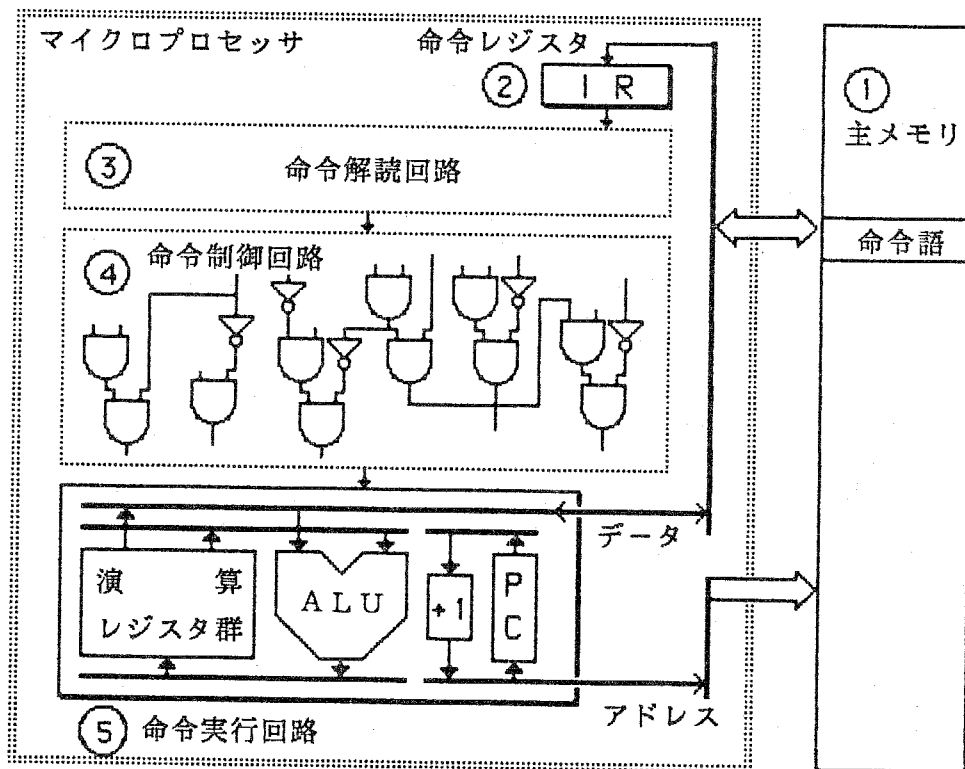


図1.4 マイクロプロセッサの構成(ランダム論理方式)

(a) 実装効率の低下

ランダム論理方式における論理回路の実装面積は、ゲート回路群の占めるゲート領域とこれらを配線する配線領域の和である。半導体の微細化により、ゲート回路は微細化係数(例えば、 $5\mu\text{m}$ から $3\mu\text{m}$ への縮小時では $3/5$)の2乗に比例して小さくなる。ところが、配線に関してはそれ程の効果はなく、微細化されるに従って配線領域の占める割合が増加してきている。

(b) 設計効率の低下

マイクロプロセッサの高機能化に伴い、図1.4の命令制御回路④が最も複雑化する。高機能かつ多数の命令を制御するため、莫大な論理式を書き出し、これをゲート回路と配線で実現しようとする、複雑さのためにもともとランダム論理方式の利点であった「むだのない論理」の特長が薄れてくる危険性もある。更に、論理変更あるいは修正のあった場合にはレイアウト設計に多大な労力を費やすことになる。このことは、マイクロプロセッサのバージョン・アップ(例えば、命令拡張など)の場合にも同じことがいえる。

一方、LSI設計を支援する自動レイアウトDA(Design Automation)や各種CAD(Computer Aided Design)が開発され、(b)設計効率の低下の問題の一部は解消されつつあるが、依然として論理設計における複雑さは残る。実装効率を犠牲にして設計効率を高める必要のある多種少量生産向きの相対的に集積度の低いゲートアレイLSIには良いが、多量生産を前提とし、LSI技術の極限を追及し高集積化をはからなければならないマイクロプロセッサには厳しい。この点はCADやDAの問題ではなく、マイクロプロセッサにおける論理方式が問題ということになる。

(2) 規則論理方式

前述したように、初期のマイクロプロセッサではランダム論理方式が一般的に使われていたが、半導体の微細化に伴い「規則論理方式」が採用されるようになってきた。この規則論理方式は図1.4における命令制御回路④の部分をマイクロコード化してメモリに置き換えたマイクロプログラム制御方式にその典型的な形態を見ることができる。マイクロプログラム制御方式は、ミニコンピュータや大型コンピュータにおいて古くから採用されている方式であり、大規模論理の設計し易さ、機能拡張性、仕様変更のし易さ等々の数多い利点をもつことから不可欠の設計手段として不動の位置を占めている。マイクロプロセッサにも全く同じことが言える^{2), 4), 9)}。ランダム論理方式で問題となってきた実装効率

の低下と設計効率の低下を一挙に解消できる。これはそれぞれ次の理由によるものである。

(a) 実装効率

マイクロプログラム制御方式における制御論理はマイクロコード化されてROMに格納されるため、微細化に伴ってDRAMが高密度になる場合と同等の効果が得られる。

(b) 設計効率

マイクロプログラム制御方式での設計手順は次のようなステップを踏む。

- (i) 実現しようとする命令機能をRTL(Register Transfer Language)によりプログラムする(マイクロプログラミング)。
- (ii) マイクロプログラミングの結果を基にマイクロ命令の形式(フォーマット)を設計し、各フィールドの機能及びコードの割付けを行う。
- (iii) RTLによるマイクロプログラムは変換ソフトウェアによって(ii)で設計したマイクロコードに変換する。
- (iv) 変換されたコードはROMの「目」、即ち、MOSトランジスタの有無という形でマスク・パターンに反映される。

以上のように、マイクロプログラム方式によれば、システムティックな設計が進められる上に各ステップでCADツールが使えるため設計効率は極めて向上する。

以上の結論として、本論文ではマルチチップ形、シングルチップ形を含めたシステムオンチップ形、特定用途形の各形態のマイクロプロセッサに対し、マイクロプログラム制御方式を採用したマイクロコード化LSIプロセッサに限定してその高性能化及び高機能化を達成する構成法に検討を加えることにする。

1.3 マイクロコード化マイクロプロセッサ

1.3.1 構成法

図1.5はマイクロプログラム制御方式のマイクロプロセッサの一般的な構成を示したものである。一つの命令処理は、命令語を主メモリ①から読み出す「命令フェッチ」、命令語を解釈する「命令デコード」、命令の内容に応じて演算等を行う「命令実行」の3つのサイクルを順次繰り返して行われる。マイクロコード化したマイクロプロセッサの外観仕様はランダム論理方式のそれと何ら変わらないが、それぞれのサイクルを実行するマイクロプロセッサ各構成要素の動作は大きく異なることになる。この点に着目してマイクロ

コード化LSIプロセッサの動作原理を以下に述べる。

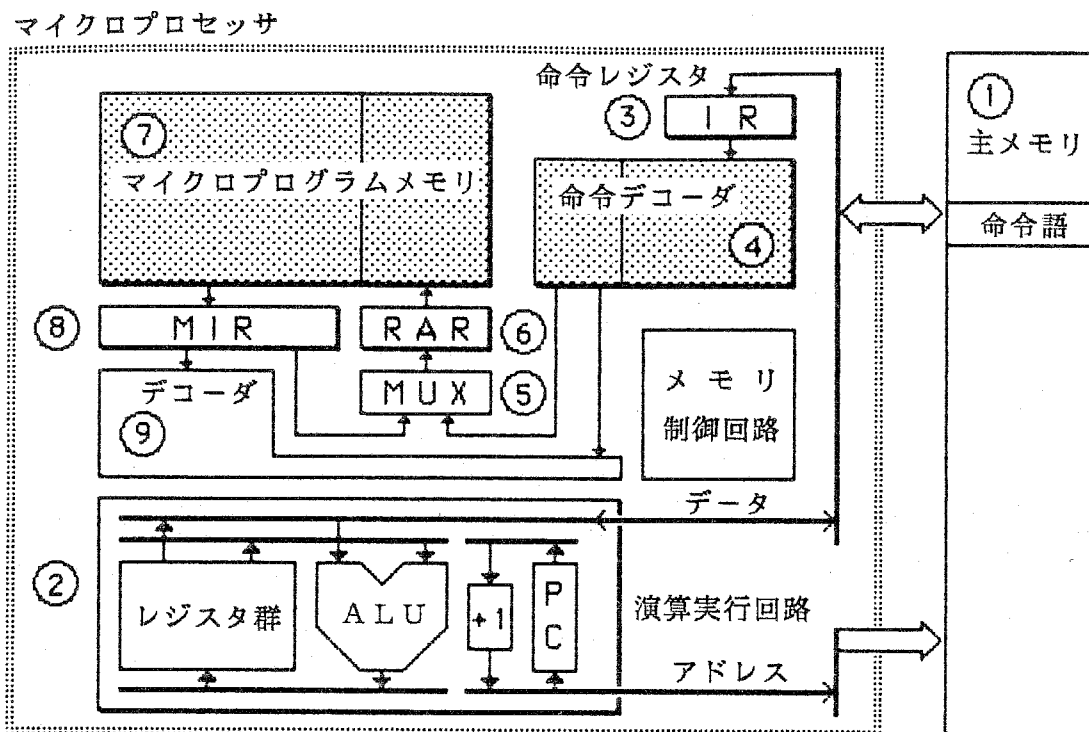


図1.5 マイクロコード化マイクロプロセッサの構成

(1) 命令フェッチ・サイクル

演算実行回路②内のプログラムカウンタPC (Program Counter)の内容をアドレスバスに出力し、これに従って主メモリ①に格納された命令語を読み出す。読み出された命令語はデータバスを介して命令レジスタ③に置数される。一方、PCは次の命令語の読み出しの準備のため更新(インクリメント)される。

(2) 命令デコード・サイクル

命令レジスタ③に置数された命令は命令デコーダ④によって解読され、命令に1対1に対応したマイクロプログラムの先頭アドレス(エントリ・アドレス)に変換される。また、命令語の一部は、オペランドとなるレジスタあるいは主メモリ①上にあるオペランドの位置を示すアドレスレジスタを指定しているため、命令デコーダ④の解読結果にはアドレスレジスタを含むレジスタ指定が含まれることがある。エントリ・アドレスは、マルチプレクサ⑤によって選択され、マイクロプログラムメモリ(μROM)⑦のアドレスレジスタRAR (ROM Address Register)⑥に置数される。この内容に従ってμROM⑦内のマ

マイクロ命令が読み出され、マイクロ命令レジスタMIR (Micro Instruction Register) ⑧に置数される。以後、マイクロ命令に含まれる次のマイクロ命令のアドレスがマルチプレクサ⑤で選択されて、RAR⑥に置数される動作を命令処理が終了するまで繰り返す。

(3) 命令実行サイクル

逐次MIR⑧に置数されるマイクロ命令はデコーダ⑨によって解釈され、演算実行回路②を操作する制御信号群に変換される。これらの制御信号により、演算実行回路②内の各種レジスタや演算回路ALU (Arithmetic and Logic Unit) が制御されて所望の命令処理が実行される。一方、マイクロ命令は演算実行回路②に密接に関係して決定されているのでその間の意味的なギャップは小さい。従って、デコーダ⑨は主メモリ①から読み出された命令語とは無関係に設計され、その規模も命令語の複雑さには直接影響されない。

以上述べたように、マイクロプログラム制御のマイクロプロセッサの構成では、命令処理手順はマイクロプログラムメモリ⑦にマイクロ命令の組合せとしてコード化(マイクロプログラム)されるため、命令の複雑さの多くはこのマイクロプログラムメモリに吸収されることになる。命令の種類や機能の違いはマイクロプログラム量と演算実行回路②内の構成要素に反映される。

1.3.2 技術課題

マイクロプログラム方式はMOSデバイスの微細化に伴って、マイクロプロセッサが高機能化するに従って設計効率のみならず実装効率の面でも有利となってきている。しかしながら、マイクロプロセッサの性能面を見ると必ずしも満足できるものではない場合もある。ここで、マイクロコード化マイクロプロセッサの構成法をLSIに向けた形で見直す必要があると考える。以下、マイクロコード化LSIプロセッサを多角的に見た場合の欠点を洗い出し技術課題を挙げる。

(1) 複数LSIチップによるプロセッサ構成

ミニコンピュータはマイクロプロセッサの高性能化、高機能化に伴ってその性能及び機能を向上している。即ち、常にマイクロプロセッサの数倍から一桁上の性能を保持しなければならないため、その論理規模も必然的に増大してきている。従って、ミニコンピュータの論理規模は、その時代のマイクロプロセッサの論理規模と比較して1桁前後大きくなってしまふことが多い。しかし、既存のミニコンピュータのロー・エンド機種は、システムの小形化、低価格化、高性能化、高信頼化といった数々の目的からニーズが高い。

以上の背景からミニコンピュータをMOS/LSI化する際にはプロセッサの論理を複数の半導体チップに分割しなければならないことが多い。複数の半導体チップによるプロセッサの構成は大型コンピュータにおいてゲートアレイLSIによって成されている。このゲートアレイLSIは主にECL(Emitter Coupled Logic)回路で構成されており、ゲート遅延時間はサブナノ秒であり、消費電力は数ワットである。小形化と低価格化を狙うミニコンピュータではECLゲートアレイの利用は電源、冷却装置の点で好ましくない。また、半導体チップ数も実装の面から高々数個に抑えねばならない。ところで、MOS/LSIでプロセッサを実現する場合、MOSデバイスの高集積性から数チップのLSIで構成することは可能であるが、次の問題点がある。

(a) LSI間の信号遅延が大きい

MOSデバイスはバイポーラ・デバイスのような電流駆動形ではないので、LSIの入出力性能が低い。そのため、命令処理の最小単位である1マシンサイクル中でのチップ間入出力回数を減らす構成が必要である。

(b) LSIパッケージのピン数に制限がある

1つのプロセッサを数チップのLSIに分割する場合、その分割方法によっては1つのLSIでの入出力ピン数が大きく変化する。LSIのパッケージとして標準パッケージのピン数(例えば、40ピン、48ピン、64ピン等)に抑えなければならない。

(c) MOSデバイスは高速性に欠ける

MOSデバイスはECL等のバイポーラ・デバイスに比較して低速である。MOSの高集積性を活かした高速回路構成が性能向上のため必要である。

以上の3つの問題点を解決して始めて高性能かつ高機能なLSIプロセッサを実現できる^{18), 19)}。

(2) シングルチップあるいはシステムオンチップ形のプロセッサ構成

プロセッサ全体が1チップで実現できる場合、特にシステムオンチップ形のマイクロコンピュータにおけるプロセッサでは他の周辺I/O機能を同一チップ上に集積するため、カーネルとなるプロセッサは小形化と高性能化が必須である。このため、この種のマイクロコード化マイクロプロセッサでは次の問題点がある。

(a) マイクロプログラム制御方式によって処理速度が低下する

マイクロプログラム制御方式のプロセッサの動作原理は既に述べたが、この方式ではラ

ランダム論理方式のプロセッサと比較して命令処理のマシンサイクル数が増える。図1.6は両方式の命令デコードから演算実行回路の制御信号発生までの信号変換の様子を示したものであるが、明らかにマイクロプログラム制御方式の方が命令実行までのオーバーヘッドが大きい。命令のデコードと、これに伴うマイクロプログラムメモリのアクセスを合わせると2マシンサイクルになるからである。プロセッサを高集積化できる場合には、パイプライン制御等の手法で解決されるものであるが、小形化を追究する機種では問題となる。

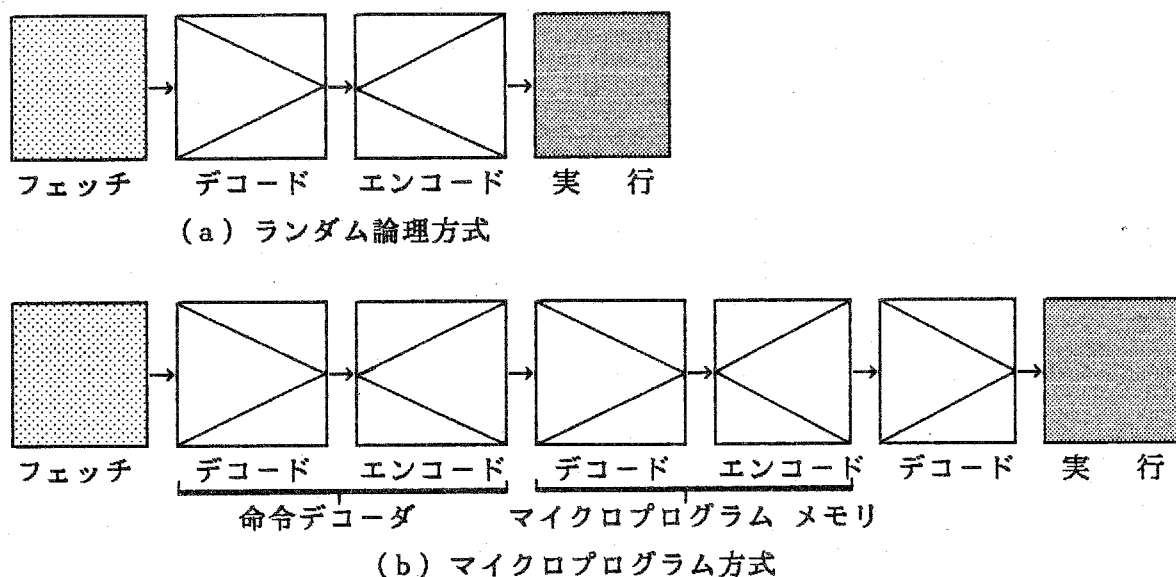


図1.6 ランダム、マイクロプログラム方式の信号変換

(b) マイクロプログラム読み出しと演算実行との間のサイクル時間に差がある

マイクロプログラム制御方式では、命令処理の実行はマイクロプログラムメモリから逐次読み出されるマイクロ命令を演算実行回路で実行することによって達成される。通常では、図1.7に示すようにマイクロ命令の読み出し(マイクロ命令フェッチ)と演算実行回路での実行がパイプライン制御される。従って、マイクロプログラムメモリと演算ユニットそれぞれのサイクル時間の大きい方に合わせてマシンサイクル時間が決定される。命令の機能が複雑な場合には、マイクロプログラムメモリが大容量化してこのアクセス時間が増加するし、データのビット長が長ければ、演算に要する時間が増加することになる。マイクロプログラムメモリのアクセス時間は回路技術により決定され、論理方式では対応できない。これに対し、論理方式を駆使できる演算実行回路のサイクル時間縮小が、高速な命令処理に重要である。

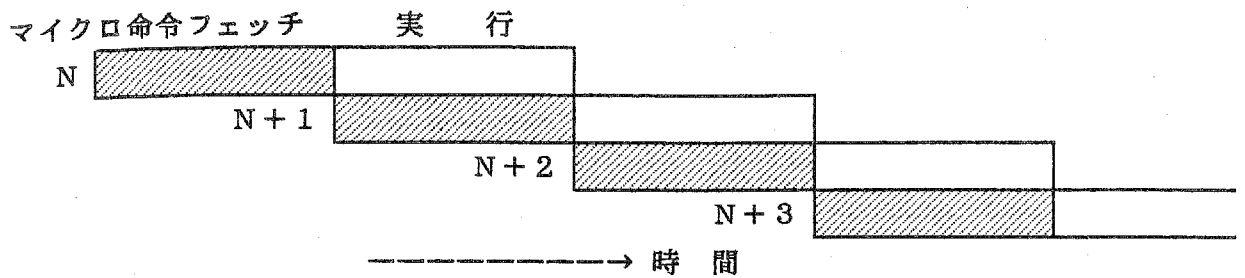


図1.7 マイクロ命令のパイプライン制御

(c) マイクロプログラム メモリのメモリ効率が低い

図1.8はマイクロ命令の種類として高性能化を意図した水平形と、高いメモリ効率を意図した垂直形と、この両者を隔合したナノプログラム方式を示したものである。ナノプログラム方式はマイクロプログラム メモリのメモリ効率を向上させる有効な方式であるが、マイクロプログラム方式より更に命令実行までのオーバーヘッドが問題となり、高性能化のためにはより多段のパイプライン制御が必要となる。そこで、マイクロプログラム方式の上でメモリ効率を上げる手段がプロセッサの小形化のため必要である。

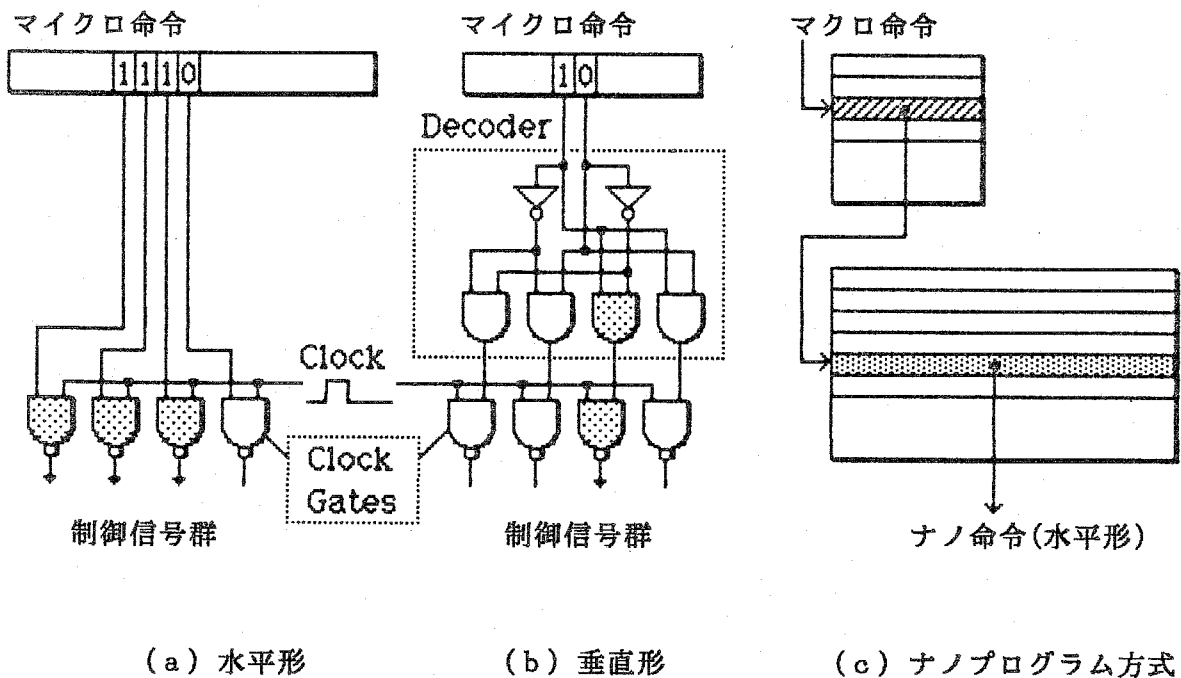


図1.8 マイクロ命令の種類

以上の3つの問題点を解決すれば、小形で高性能なマイクロプロセッサの実現が可能となる^{9), 19)~21)}。

以上の形態のマイクロプロセッサを全体的に考えると、これらは目的毎の構造を持ったマイクロプロセッサである。しかし、命令語を解説してこれを実行するという観点からは共通的な機能を果していることになる。次に挙げるプロセッサ構成法は、アプリケーションに適応し易い柔軟な構成を追求したものである。

(3) 可変構造化プロセッサ構成

命令語を主メモリあるいはこれに準じるメモリから読み出し、これを解説し実行する形式をとるマイクロプロセッサでは、命令語の構成に依存した構成となる。従って、命令語に依存せず、命令語に合わせて構造を自由に換えられるためには命令語を解説する部分が柔軟である必要がある。マイクロプロセッサの一般的な構成では、次の2点が特に大きな問題点となる。

(a) 命令語の構成(フォーマット)が機種によって異なる

命令語を構成する各フィールド、例えば、命令の種類を示すオペレーション・コード、命令実行に使用するオペランドの種類を示すオペランド・スペシファイア等の内容、ビット長、配置それぞれが全て異なる。どんな構成の命令語であっても処理可能とするためには、任意のフィールドの切り出し及びこの解説を自由に設定できる構造が必要である。

(b) レジスタ構成が機種により異なる

マイクロプロセッサを構成するレジスタは機種によりその数及び指定方法が異なる。前者は十分な数のレジスタを準備し、後者は(a)の命令語解説の柔軟性で対処することが考えられる。これらを実現するレジスタ構成法が必要となる²¹⁾。

(4) 特定用途のマイクロプロセッサ構成

グラフィック、イメージ、通信制御等の特定用途のマイクロプロセッサでは、汎用マイクロプロセッサの機能を特化するため、高機能化と高性能化の両立が最大の課題である。機能をファームウェア化して多機能を吸収する一方で、高機能を得るためには次の問題点がある。

(a) 機能が1つのプロセッサに集中すると性能低下を招く

多くの機能を1つのプロセッサで実行するようにすると、それらの機能はマイクロプログラミングで実現でき、設計効率が高くなる利点がある。しかし、汎用プロセッサによる

ソフトウェア処理との性能差があまり期待できなくなる。機能を分散して複数のプロセッサに分担する必要がある。

(b) 複数プロセッサ構成は設計効率が低い

(a)の問題を解決するために複数のプロセッサをオンチップ化すると設計工数が比例して増加する。これらのプロセッサの設計手順を統一化して設計効率を上げること、同系統の他の特定用途マイクロプロセッサへの展開を容易にすることによって等価的に設計工数を下げる配慮も必要である。

以上の相反する2つの問題を解決し、特定用途に向けた高性能かつ高機能のマイクロプロセッサを達成することができる^{10), 11)}。

以上に述べた4種類のプロセッサ構成上の問題点を、筆者らは、マイクロプロセッサにおける重要な課題として取り組んできた。

1.4 まとめ

マイクロプロセッサはLSI技術の進歩に並行して、その用途に応じた色々の機種が現われてきた。この中でプロセッサ構成法からみると、LSIに向けた論理方式、即ち、マイクロチップ・アーキテクチャがマイクロプロセッサの性能及び機能に重要な役割を果たしていることがわかった。マイクロプロセッサはより微細化の進む半導体技術のインパクトを受けて、更にその形を変えてゆくものと考えられる。この過程でマイクロチップ・アーキテクチャもコンピュータ・アーキテクチャに合わせて変化してゆくことが予想される。本論文では、多角的見地からマイクロプロセッサの高性能化及び高機能化を実現するためのハードウェア構成法を取りあげており、これらの技術は次世代チップへの発展の基本となるものと考えている。

筆者らは、本章で示したマイクロプロセッサ構成上の技術課題を、具体的なピークルを設定して研究を進めてきた。それらは、制御用ミニコンピュータのLSI化機種1件、周辺I/O機能を同一チップ上に集積したシステムオンチップ形マイクロコンピュータ2件、グラフィック制御用LSI1件である。これらは全てマイクロコード化したマイクロプロセッサであり、それぞれの目的を達している。ここで述べられた手法は、他のマイクロプロセッサにも広く適用されており、その基本的思想は今後共、色々な形で活用されていくものと考えている。

1.5 第1章の参考文献

- (1) Dave Bursky : "MOS/digital/Analog mix=low-cost, high-performance single chip μ Cs", Electronic Design, Vol.27, No.13, pp.43~48 (1979-6)
- (2) S.P.Morse et al. : "The Intel 8086 Micro Processor : A 16-bit Evolution of the 8080", Computer, Vol.11, pp.18~27 (1978-5)
- (3) M.Shima : "Two Versions of 16-bit chip span microprocessor, minicomputer needs", Electronics, Vol.51, No.26, pp.81~88 (1978-6)
- (4) E.Stritter et al. : "A Microprocessor Architecture for a Changing World: The Motorola 68000", Computer, Vol.12, No.2, pp.43~52 (1979-2)
- (5) J.A.Bayliss et al. : "The Instruction Decoding Unit for the VLSI 432 General Data Processor", IEEE Journal of Solid-State Circuits, Vol.SC-16, No.5, pp.531~537 (1981-10)
- (6) R.Mateosian : "System Consideration in the NS32032 Design", Proc. of 1984 NCC, pp.77~81 (1984-7)
- (7) D.MacGregor et al. : "The Motorola MC68020", IEEE Micro, Vol.6, No.8, pp.101~108 (1984-8)
- (8) D.MacGregor et al. : "Virtual Memory and the MC68010", IEEE Micro, Vol.2, No.4, pp.24~39 (1983-6)
- (9) H.Maejima et al. : "The VLSI Control Structure of a CMOS Microcomputer", IEEE Micro, Vol.3, No.6, pp.9~16 (1983-12)
- (10) H.Maejima et al. : "VLSI for High Performance Graphic Control Which Utilizes Multi-Processor Architecture", Proceedings of the International Conference on Computer Design, pp.586~591 (1984-10)
- (11) K.Katsura, H.Maejima et al. : "VLSI for High-Performance Graphic Control Utilizing Multiprocessor Architecture", IEEE Transaction on Electron Device, Vol.ED-32, No.11, pp.2232~2237 (1985-11)
- (12) H.Kurokawa et al. : "The Architecture and Performance of Image Pipeline Processor", Proceedings of VLSI '83 International

- Conference, pp.275~284 (1983-8)
- (13) A.Szczepanek et al. : "An I E E E 802.5 Compatible LAN Controller with On-Chip ROM", I S S C C Digest of Technical Papers, Vol.28, pp.188~189 (1985-2)
 - (14) 田丸外 : "超 L S I 技術とマイクロプロセッサ", 電子通信学会誌, Vol.62, No4, pp.455~461 (昭54-4)
 - (15) 向井 : "L S I メモリの概要", 電子通信学会誌, Vol.67, No11, pp.1146~1151 (昭59-11)
 - (16) K.Itoh et al. : "An Experimental 1Mb DRAM with On-Chip Voltage Limiter", I S S C C Digest of Technical Papers, Vol.27, pp.282~283 (1984-2)
 - (17) S.Yamamoto et al. : "A 256K CMOS SRAM with Variable-Impedance Loads", I S S C C Digest of Technical Papers, Vol.28, pp.58~59 (1985-2)
 - (18) 前島外 : "制御用16ビットマイクロコンピュータのアーキテクチャ", 情報処理学会論文集, Vol.21, No3, pp.208~215 (昭55-5)
 - (19) W.N.Johnson : "A V L S I Superminicomputer CPU", I S S C C Digest of Technical Papers, Vol.27, pp.174~175 (1984-2)
 - (20) H.Maejima et al. : "CMOS 8-Bit Single-Chip Microcomputer HD6301", Hitachi Review, Vol.30, No4, pp.171~176 (1981-8)
 - (21) H.Maejima et al. : "A CMOS Microprocessor with Instruction-Controlled Register File and ROM", I S S C C Digest of Technical Papers, Vol.28, pp12~13 (1985-2)
 - (22) 前島外 : "高集積マイクロコンピュータに適したマイクロプログラム制御方式", 情報処理学会論文集, Vol.23, No1, pp.16~24 (昭57-1)

第2章 マルチチップ形LSIプロセッサの チップ分割法

第2章 マルチチップ形LSIプロセッサの チップ分割法

2.1 まえがき

最近の計算機制御分野では機能を分散化した制御システムが一般化しており^{1)~5)}、その特長はホストに高性能スーパーミニコンピュータ、制御端末に多数のマイクロコンピュータを配置することによって大規模かつ拡張性に富むシステムが容易に構成できる点にある。このような大規模システムにおいて、ホスト・コンピュータには高いスループットが要求され、そのため大型コンピュータに匹敵する性能を備えていなければならなくなっている。これに対し、制御端末のコンピュータはサブシステム毎に分散制御するため、高集積のマイクロプロセッサによって構成されるようになってきた。この動きは汎用のマイクロプロセッサが8ビット系から16ビット系^{6)~8)}、更に32ビット系^{9)~12)}へと進行し、半導体の高集積化が急速に発展していることといった周囲状況から着実に進展している。

このような端末のマイクロプロセッサにはどのような機種が適しているのだろうか。汎用のマイクロプロセッサ又は上位のミニコンピュータと互換性のあるLSI化機種の2種類が考えられる。前者が不特定多数のアプリケーションを対象として汎用的なアーキテクチャをとっているのに対し、後者が上位ミニコンピュータのアーキテクチャを踏襲している点に根本的な思想の違いがあろう。従って、ミニコンのLSI化機種の場合、多くのアプリケーションをこなすことによって長い間に蓄積した豊富なソフトウェア財産や信頼性を含めた数々のノウハウを継承することができる。このため、システム固有の強みを集積したLSIプロセッサによって構成した超小形マイクロコンピュータがミニコンピュータのロー・エンド機として分散制御システムにおけるキー・コンポーネントとして有利となり易い。ところで、ミニコンピュータの機能は一般的に汎用マイクロプロセッサのそれよりも高いために、汎用のマイクロプロセッサのように1つの半導体チップで実現することは半導体の集積度からみて困難である。そこで、この場合の多くは複数の半導体チップで1つのプロセッサを構成する「マルチチップ形」^{13)・14)}をとる。

本章では、マルチチップ形LSIプロセッサのマイクロチップ・アーキテクチャに関し、特に制御用として重要な高性能化及び高信頼化を達成するプロセッサ構成法を論じる。高性能化に関しては命令実行中に次の命令フェッチを行うパイプライン制御方式を実現し得

るチップ分割法と、これにより実現される並列処理動作及びそのマシンサイクルを向上するための高速回路構成を示す。高信頼化に関してはプログラムの暴走を防止するシステム監視機能、プログラム破壊を防止する主メモリ保護機能のオンチップ化を提案する。

2.2 制御用LSIプロセッサの概要

2.2.1 位置づけ

制御用としてのLSIプロセッサの具体的応用を図2.1に示す。この図は分散制御システムにおける端末コンピュータ(TC: Terminal Computer)として使われた例を示したものであるが、この場合を例に制御用LSIプロセッサの位置づけを述べ、その設計を行う上で必須となる技術課題を合せて示す。

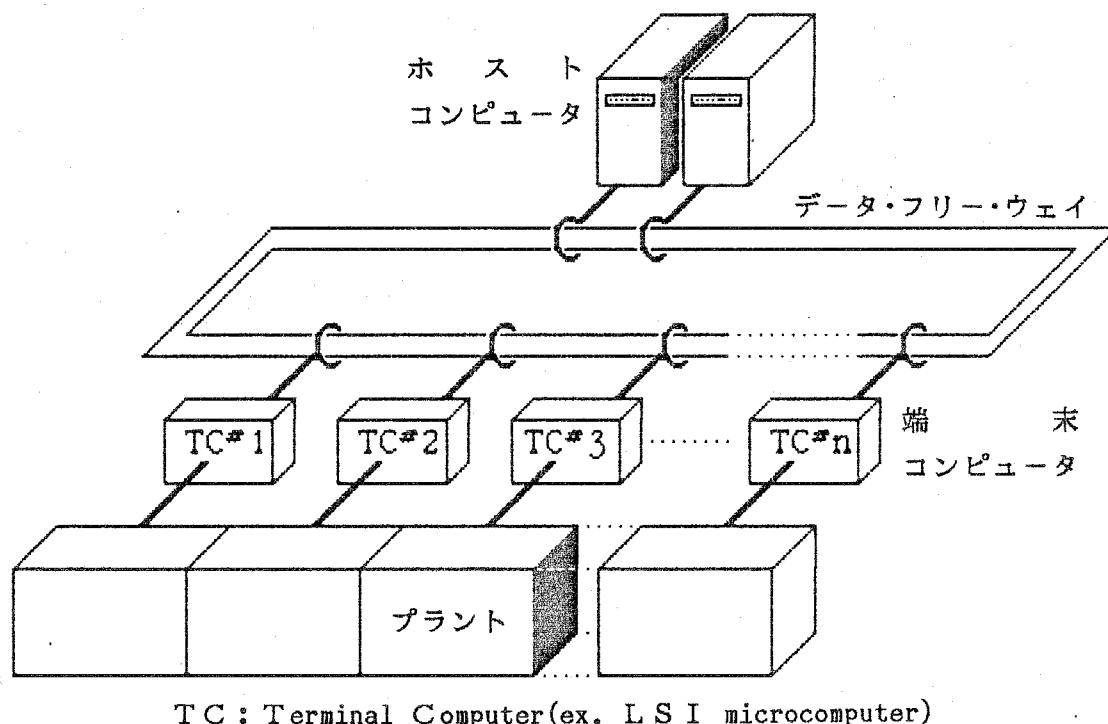


図2.1 分散制御システムの構成

(1) 超小形化

分散制御システムでは大規模なシステムを多数のサブシステムに分散して制御するため、各々のコンピュータは超小形であることがシステムの最適化(性能・価格比向上)に不可欠である。

ところで、マイクロプロセッサに用いられている半導体プロセスは主として次の2つである。第1はビットスライス型マイクロプロセッサに用いられているLS-TTL (Low-power Schottky Transistor Transistor Logic)に代表されるバイポーラ、第2はマイクロプロセッサに用いられているMOSである。一般的には集積度でMOS、デバイスの性能でLS-TTLが優位にあるが、MOSにおける微細加工技術の進歩には著しいものがある。その波及効果は極めて大きく、高密度かつ高速なLSIの実現に対し、将来にわたって基本的となるプロセスと判断できる。

本章で述べるLSIプロセッサも超小形マイクロプロセッサの実現に向けてNチャンネルMOS技術を用いて開発されたものである。

(2) 高速性

制御用の端末コンピュータは図2.1にみられるように制御対象を直接制御するため、本質的に実時間処理を行うものであり、高性能な命令処理を必要としている。特に圧延機制御のような機械と直結する部分でのDDC (Direct Digital Control)では機械の応答速度に見合った高性能が要求される。

前に述べたように、MOSデバイスの性能がLS-TTLのそれよりも劣っており、これを克服する鍵はLSIプロセッサのマイクロチップ・アーキテクチャと高速論理回路方式にある。前者は命令の並列処理性、後者はMOSデバイスに適した高速論理の追求が課題である。

(3) 高信頼性

制御用コンピュータの分野ではRAS (Reliability, Availability, Serviceability)は重要な機能である。小さな誤りがシステム・ダウンに波及する致命的な障害に至ることが多々起り得るからである。この防止には先ず種々の段階でプロセッサの誤動作を検出することから始まる。一般的には主メモリに係るデータとアドレスの誤り、即ち、パリティエラー、アドレスエラーの検出機能、プログラム領域(例えば、OSのプロシージャ)の破損を防止する主メモリ保護機能が最低限要求される。また、ハードウェアとソフトウェアの誤動作を一括して検知するシステム監視機能もシステム暴走の危険を回避する有力な手段となる。このようにマイクロプロセッサがデータ処理に留まらず、エラー検出・処理も行うことによって制御用コンピュータとしても特長を活かすことができるだけでなく、全体の部品数を低減する効果もある。

(4) 互換性

マイクロプロセッサが既存システムで開発したソフトウェア資産(コンパイラ, サブルーチン・ライブラリ等の各種ユティリティ)やハードウェア資産(主メモリ, 入出力装置等)を共用できる互換性はシステム構成, 運用の面で利点が多い。例えば, 分散制御システムにおいて, 同一命令によってホスト・コンピュータと端末コンピュータ相互間の入出力装置, ファイルの共用(データ管理)やホストから端末のプログラム起動/停止(タスク管理)といった「使い易さ」に魅力あるシステムが得られる。

2.2.2 要求仕様

表2.1は制御用16ビットマイクロプロセッサの仕様概要を示したもので, 既存のミニコンピュータとソフトウェア, 入出力装置共に完全な互換性を持っている。命令はS1, S2の1語命令(16ビット)とL1, L2の2語命令(32ビット)で構成されており, それぞれの形式を図2.2に示した。

表2.1 制御用16ビットマイクロプロセッサの仕様概要

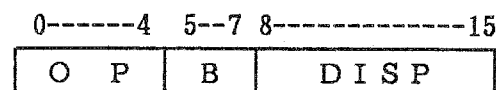
項 目	内 容
デ - タ 長	16, 32ビット
命 令 長	16, 32ビット
命 令 形 式	16ビット(S1, S2), 32ビット(L1, L2)
命 令 体 系	59命令(浮動小数点を含む)
主メモリ容量	64K語(16ビット/語)
メモリ・バス	データ(16ビット)+パリティ(1ビット) アドレス16ビット, DMA機能内蔵
割 込 み	3レベル(ベクタ方式)
レジスタ(16ビット)	命令レジスタ×2 ベースレジスタ×6(内, インデックスレジスタとして3本使用可能) 内部ワークレジスタ×5 プログラムカウンタ×1 ステータスレジスタ×1

本プロセッサの特長は, 高速の固定小数点乗除算, 浮動小数点演算を標準装備した強力な命令体系, 割込みのハードウェア・ベクタリングによる高応答性, システムの信頼性を

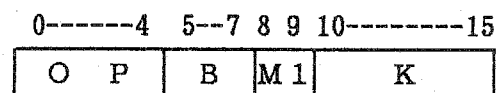
考慮したRAS機能の充実にあり、これらは制御用として欠くことのできない重要なものである。

また、マイクロプログラム制御方式の採用により、シーケンス制御、DDCといった特殊なアプリケーションに対し、シーケンス命令、平方根・不感帯・飽和等の特殊命令を容易に追加することができ、適応性の高いプロセッサでもある¹⁵⁾。

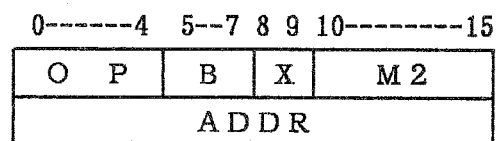
1. Format S1



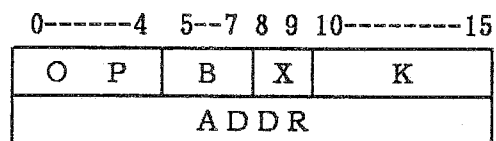
2. Format S2



3. Format L1



4. Format L2



Examples

(A) + (EA) → A
EA = (B) + DISP

Shift left (A), K-bit
M1 : OP modifier

(A) + (EA) → A
EA = (B) + (X) + ADDR
M2 : OP modifier

if condition K is true, then
EA → PC
otherwise, PC + 2 → PC

図 2.2 制御用16ビットマイクロプロセッサの命令形式

2.3 チップ分割プロセッサ構成法

MOSデバイスによりマイクロプロセッサを実現する場合、最大の課題はデバイス特性を見極め、いかに高速性を引き出すかにあるといえよう。このためには2段階の技術が必要であり、第1はシステムの並列処理性を高めるプロセッサ構造、第2はLSI内部における信号遅延を最小とする回路構造である。

以下、それぞれについて述べることにするが、特に第1の段階は、マイクロプロセッサの論理を複数の半導体チップへ分割する方法に大きく影響を受けることになる。先ず、このチップ分割法に関して述べる。

2.3.1 チップ分割法

本項では集積度の制約を3,000ゲート(8,000~9,000MOSトランジスタ),パッケージ・ピン数の制約を40~42ピンとして,これらの制約を解決するプロセッサの論理分割手法を示す。この制約は5 μ mのNチャンネルMOSプロセスを使用する場合の大体の目安である。

従来,集積度,パッケージ・ピン数の制約から逃れるために論理を複数のチップに分割した例は幾つか存在する。ここでは2チップへの分割に限定してそれらの方法に評価を加えることにする。これらの方法は図2.3に示したように,基本的にはプロセッサを構成する機能ブロック,即ちデータパス部(データ及びアドレス演算部),制御部(命令解読部及びマイクロプログラム制御部),インタフェース部(メモリ及びI/O制御部)を横割り或いは縦割りしたものである。それぞれの方法での優劣を見極めた上で最適な分割法を提案する。

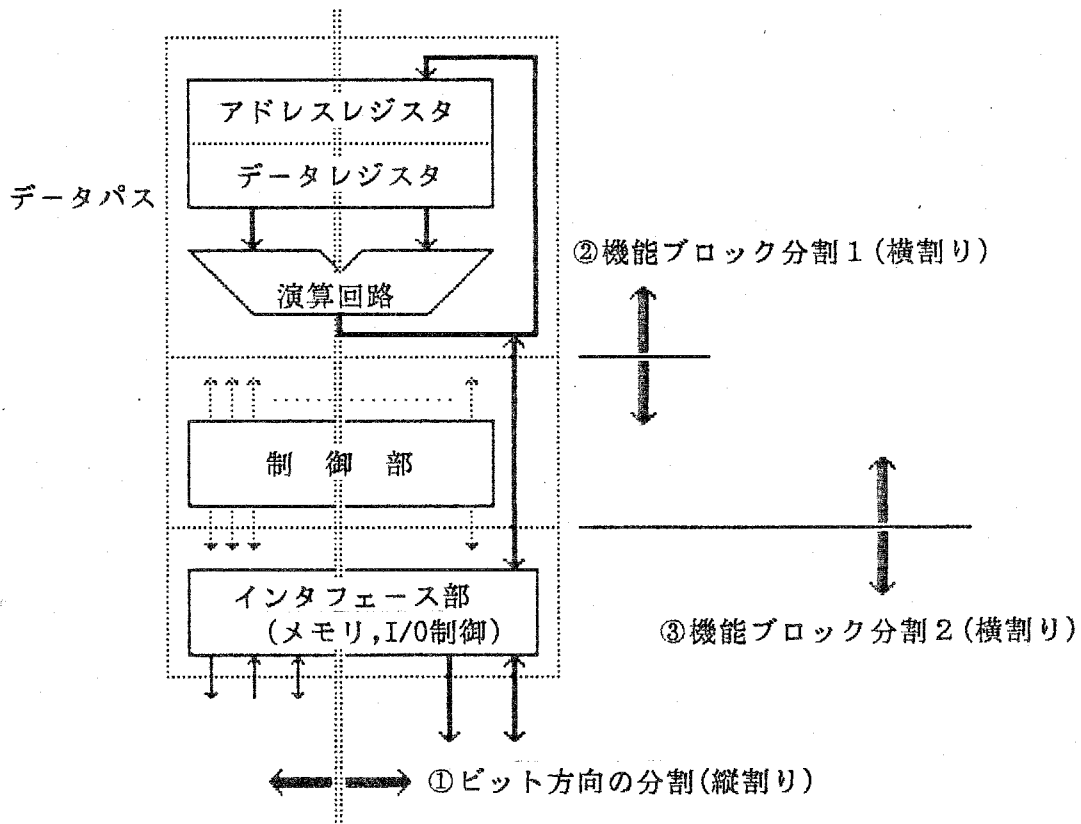


図2.3 従来のチップ分割法

(1) ビット方向の分割(縦割り)

16ビットの構造をビット方向への分割は,バイポーラ・ビットスライス形LSIで行

われている方法である。ここでは8ビット単位の分割を意味する。1種2チップのLSIで16ビット構成がとれるため、LSI開発品種が1個でよい反面、2チップ間のインタフェース信号が増加して実現が難しい(ピン・ネック)。

(2) 機能ブロック分割1(横割り)

プロセッサの機能ブロック中のデータ部を1チップ、制御部及びインタフェース部を1チップのLSIに分割する方式である。LSI構成上に問題点は少ないが、チップ間のインタフェース信号が増加し、このため並列処理性に欠ける。

(3) 機能ブロック分割2(横割り)

プロセッサの機能ブロック中のデータ部及び制御部を1チップ、インタフェース部を1チップのLSIに分割する方式である。前者のLSIに多くの機能が集中することにより集積度、ピン数の両面で非常に厳しい反面、論理動作がLSI内部で閉じるため比較的高速化を達成しやすい。しかし、この構成法もまた命令処理における各種動作(命令フェッチ、命令デコード、命令実行)の並列処理性の点で難しい。

以上に述べた3つのチップ分割法における機能的な流れを解析すると、一方のLSIが他方のLSIに従属的であり、両方が独立に動作することが少ない。これは1つの命令を処理する際、動作する論理領域が狭いことを意味する。高速化を達成するには2チップのLSIが可能な限り独立して(並列して)動作できることが重要である。

そこで、2チップのLSIにおける並列処理化を狙って、新しい論理分割法を提案する。プロセッサが命令を処理する場合の機能的な流れは、命令フェッチ、アドレス演算(データの格納されている主メモリのアドレス計算)、データ読み出し、データ演算であり、更にこれらの制御及び割込み制御機能が加わる。この流れは命令フェッチと命令実行の2つに分けられ、後者の機能は前者に従属的ではあるがそれぞれの機能は独立性の強いものとなる。従って、命令フェッチが終了して命令実行に制御が移った時点で命令フェッチ機能が開放されることから、これを次の命令フェッチに当てられる。このように命令フェッチ機能と命令実行機能とが2種のチップに独立していれば、それぞれの処理の並列化が可能となる。

次に、以上に述べた考え方を基本とするチップ分割の1つの例を図2.4に示す。2チップのLSI名称は、それぞれの機能分担からIFCU(Instruction Fetch Control Unit)とRALU(Registers and Arithmetic & Logic Unit)と呼び、それぞれの機

能の分担は次のとおりである。

(a) IFCU

主に、命令フェッチに係る諸機能を有する。

- (i) 命令語の読み出しと実効アドレス計算
- (ii) 命令語の解釈及びマイクロプログラム制御
- (iii) インタフェース制御(主メモリ, 入出力装置, 割込み制御)

(b) RALU

主に、命令実行に係る諸機能を有する。

- (i) 命令の実行(各種演算)
- (ii) テスト(上記各種演算で発生する状態のテスト, マイクロプログラム分岐制御)

以上の論理分割により得られる2種LSIそれぞれの集積度, パッケージ・ピン数は図2.4に示したとおりである。

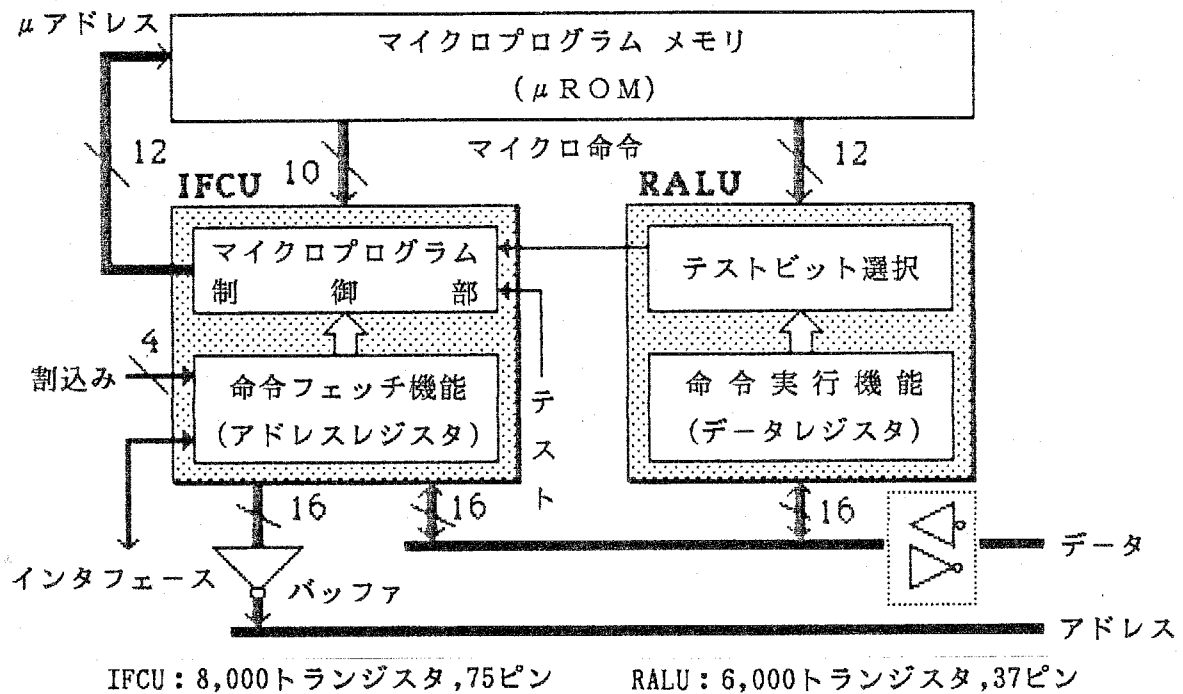


図2.4 並列処理化し易いチップ分割

図2.4の分割法は、集積度については問題無いがピン数の点で実現が難しい。即ち、IFCUチップのインタフェース信号が集中してしまう。この問題を解消するために、図2.5に示した方法を試みた。

(1) アドレス、データバスの時分割制御

アドレスバッファレジスタを外部に設けて、アドレス出力専用の16ピンを削減する(図2.5中の①)。外部レジスタの追加は図2.4の方式でのアドレスバス駆動用バッファに置き変わるだけであるので、LSI以外の外部ハードウェア増加は無い。

(2) マイクロ命令バスの時分割制御

マイクロプログラム用アドレスレジスタを外部に設けて、マイクロ命令とマイクロプログラム・アドレスのバスを共通化し、アドレス出力専用の12ピンを削減する(図2.5中の②)。本アドレスレジスタを外付けしたことにより、このレジスタの制御信号(分岐かインクリメントかのモード信号)とRALU内のテスト結果を高速にマイクロプログラム制御に反映させるため同LSI内にこの信号を発生する機能を持たせる(図2.5中の③)。

(3) 割込み回路

割込みは命令フェッチに係る。これをIFCUが管理するのが自然であるが、ピン・ネック及び割込みレベルの判定(テスト機能)の関係でRALUに持たせる(図2.5中の④)。

以上の方法で図2.5に示すようにピン・ネックが解消されている。これらの方法の実施によって図2.4の考え方を保つことができる。並列性が犠牲になることはない。

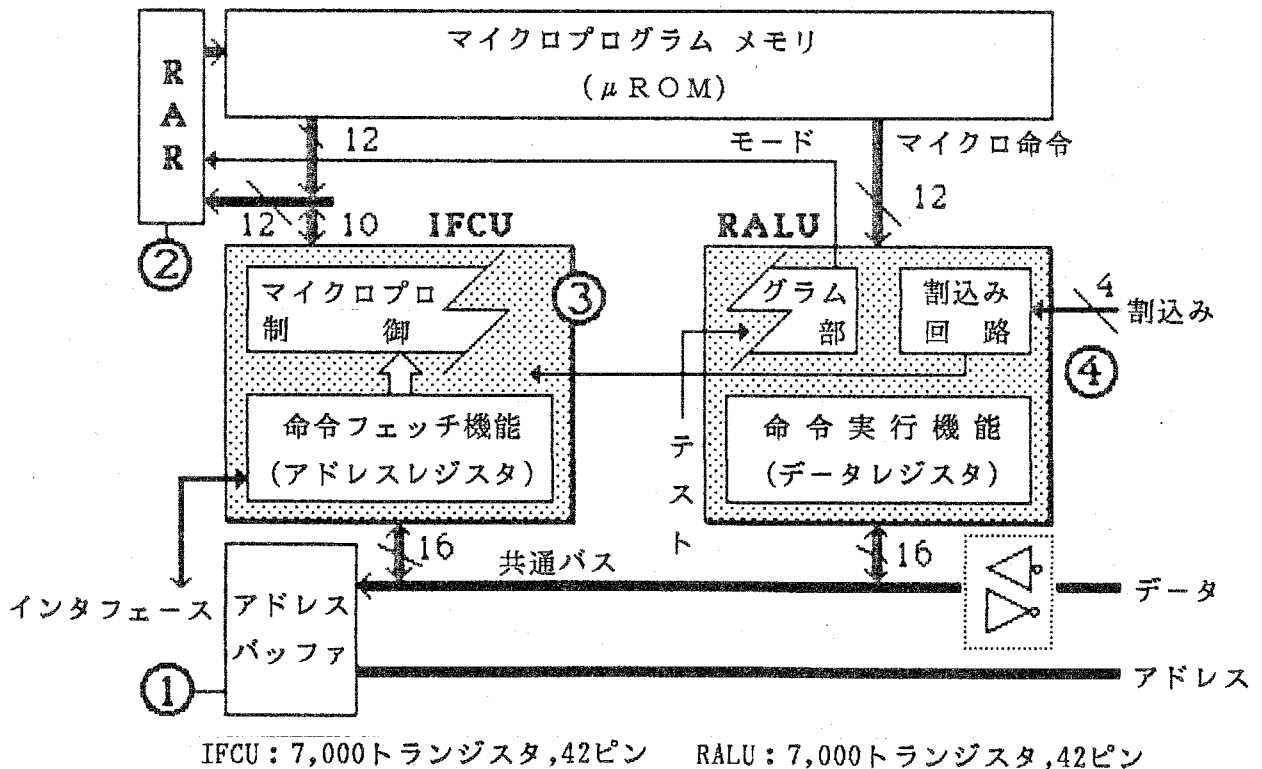
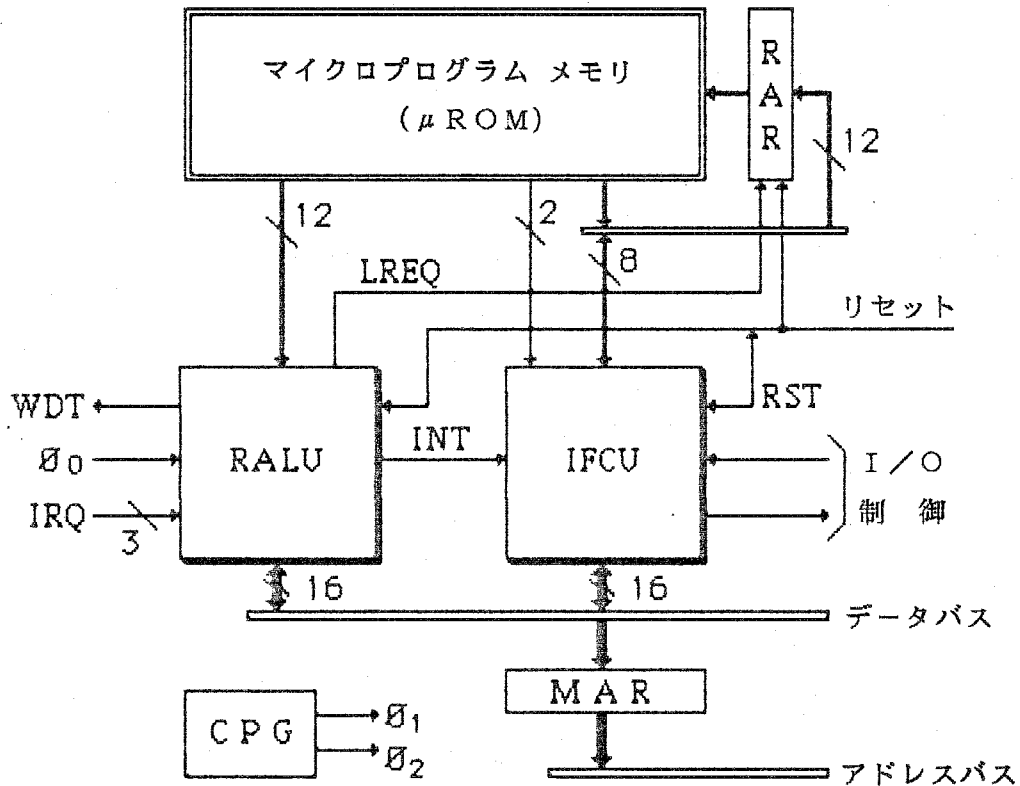


図2.5 新しいチップ分割法

2.3.2 並列処理動作

並列処理方式は「パイプライン制御」とか「先取り制御」とか呼ばれており、ミニコンピュータ以上の機種では常識的に用いられている技術である。しかし、マイクロプロセッサ、特にミニコンピュータの機能をLSI化しようとする時、その時点の集積度、パッケージのピン数に制限があり、その構成は難しい。

前項に示したプロセッサにおいて、そのハードウェアを機能面から考察し、命令フェッチを行うLSI(IFCU)と命令実行を行うLSI(RALU)の2種LSIに分割した構成法を提案したが、本項ではその有効性を示す。その構成を図2.6に示したが、これらの2種LSIを基本としてメモリアドレスレジスタ(MAR)、マイクロプログラムメモリ(μ ROM)、 μ ROMアドレスレジスタ(RAR)を基本構成要素としている。



- | | | | |
|------------|---|------|---------------------------|
| RAR | : ROM Address Register | LREQ | : Load Request |
| RST | : Reset | WDT | : Watchdog Timer |
| INT | : Interrupt | MAR | : Memory Address Register |
| RALU | : Registers and Arithmetic & Logic Unit | | |
| IFCU | : Instruction Fetch Control Unit | | |
| CPG | : Clock Pulse Generator | | |
| μ ADDR | : Micro Address | | |

図2.6 LSIプロセッサの構成

これをビットスライス形のバイポーラLSI^{16),17)}で構成したものと実装面で比較すると、プロセッサは約1/3に小形化される。この効果は、低消費電力化、低コスト化、更に、高信頼度化へ波及する。以下に本プロセッサ構成上の特長を示す。

(1) IFCUの役割(命令フェッチ)

命令語を主メモリから読み出し、これを解読してオペランドのアドレスを計算するところまでを行う。更に、命令実行を制御するマイクロプログラムメモリROM中のマイクロプログラム先頭アドレスを作成し、これをRARへ転送することによりRALUへ制御を渡す。以後、次の命令フェッチを開始できる。

以上の役割から、IFCUは主メモリインタフェース、命令デコーダ、アドレス計算に必要なアドレスレジスタ群と加算回路、マイクロプログラム・シーケンサを分割して有する。

(2) RALUの役割(命令実行)

前記RARに置数されたアドレスに対応してROMから読み出されるマイクロ命令を順次実行することによりIFCUにより読み出された命令の実行を行う。実行終了時に次の命令実行のためのマイクロプログラム先頭アドレスをIFCUに要求する。また、マイクロプログラムでの条件分岐制御も本LSIが管理する。

以上の役割から、RALUはマイクロ命令デコーダ、命令実行に必要なデータレジスタ群とALU、マイクロプログラム制御での条件分岐に必要なフラグ類を分割して有することになる。

(3) フェッチ制御方式

フェッチ動作は命令形式(S形式、L形式)、命令毎のアドレッシングの違いなど複雑な制御を必要としているため、マイクロ命令で制御するとその語長は長くなってしまふ。そこで図2.7に示すように、マイクロ命令制御はフェッチの起動のみとし、以後はLSI内部制御として同時に高速化も図っている。この結果、IFCU制御のマイクロ命令語長は10ビットとなり、S形式、L形式の命令フェッチ・サイクル数はそれぞれ2、3サイクルとなった。

(4) パイプラインの同期化

前記したように命令フェッチがIFCU内部で閉じた制御を行っているのに対し、命令実行はマイクロ命令制御となっている。従って、命令フェッチと実行の間には一義的な関

係は存在せず、2チップの間のパイプライン動作には「ずれ」が生ずることもある。この「ずれ」は図2.8に示すような方式で解決した。IFCUがマイクロプログラムの先頭アドレスをRARを介して μ ROMに提供し、 μ ROMから読み出されたマイクロ命令をRALUが受け取るという基本的立場から、IFCUはそのフェッチ動作が終了していない場合には特定のアドレスを提供するようにしている。この特定アドレスに対応するマイクロ命令は無実行かつ命令実行終了を指示しており、RALUはそのマイクロ命令の実行を繰り返すことによってIFCUのフェッチ終了を待つ。IFCUのフェッチ終了サイクルではフェッチした命令を解読した結果得られるマイクロプログラムの先頭アドレスを出力し、RALUでの命令実行に移る。このようにして両チップ間のパイプライン同期をとった。

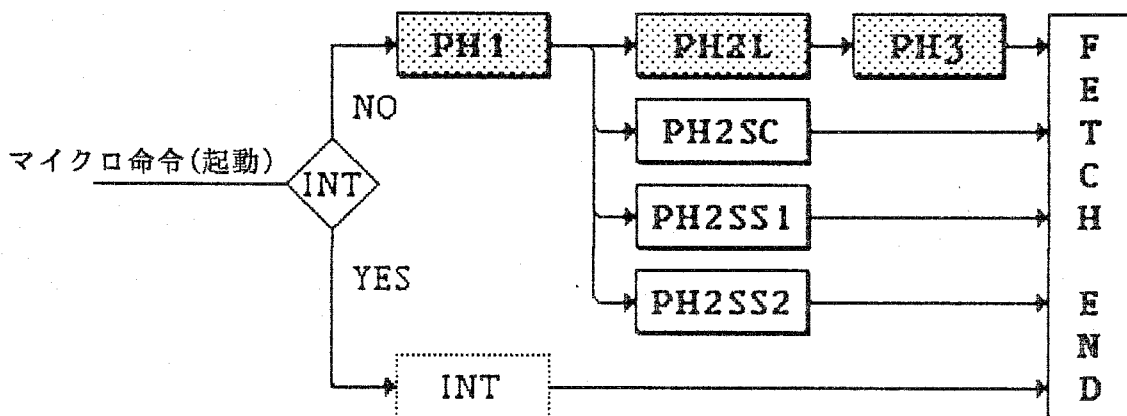


図2.7 フェッチ制御方式

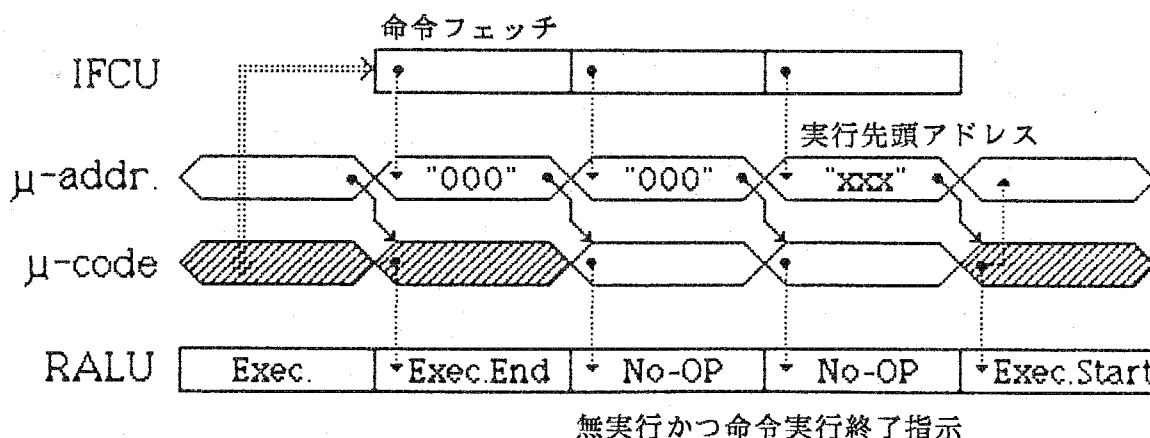


図2.8 IFCUとRALU間のパイプライン同期化

(5) 割込みの受付け

割込みは、先ずRALUに入力し、ここでレベル間の優先判定を行い、同時に割込みの有無をIFCUに知らせる。IFCUではフェッチ起動を指示するマイクロ命令を受けとった時にこの判定を行う。もし、割込みが有る場合にはフェッチ動作を抑制し、前命令実行終了時に割込み処理用マイクロプログラム先頭アドレスを提供する。

以上のように、2チップ間の制御の受渡しをマイクロプログラムのアドレスで実現したことにより2チップの独立性(並列性)を高めると同時に、LSI実現の際、常に問題となるパッケージのピン数を42ピンに抑えることができた。

2.3.3 高速回路構成

プロセッサの高速処理を実現する方法は、プロセッサの構成法の他に処理の最小単位であるマイクロサイクル時間を短縮すること、即ち、1マイクロサイクル中に行う論理動作時間を最小にすることである。1マイクロサイクルでは乗除算における繰り返し演算(加減算とシフト)が最もクリティカルな論理動作である。このような論理動作において、前項に示したIFCUにおけるオペランドのアドレス計算及びRALUにおける命令実行過程での各種データ演算を行う加減算回路に最も時間的比重が大きい。加減算回路では桁上げ(キャリ)を発生し、上位桁にこれを伝搬する加算(あるいは減算)の速度が最も厳しい。本項では加算回路を例として、MOSに適した高速回路方式を提案する。

加算回路は、従来、リップル・キャリ方式、キャリ・ルックアヘッド方式が知られているが、いずれの方式をとっても1マイクロサイクルでの論理段数全体の40~50%を占めている。従って、この部分を高速化することが最も効果的な手段となる。16ビット加算回路の信号遅延時間の算出は、図2.9のモデルに従い、次式を用いて行った。

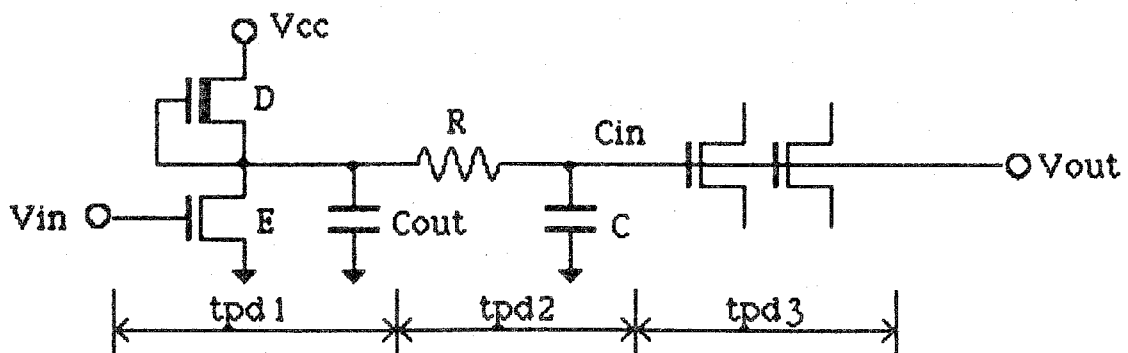


図2.9 MOS回路遅延モデル

$$t_{pd} = \sum_{i=1}^N (t_{pd1i} + t_{pd2i} + t_{pd3i}) \quad (1)$$

ここで、 N は加算回路の論理段数、 t_{pd1} は各論理ゲートの出力容量 C_{out} による遅延時間、 t_{pd2} は各論理ゲートの配線容量及び配線抵抗 R による遅延時間、 t_{pd3} は次段の論理ゲートの入力容量 C_{in} の総和、即ちファンアウトによる遅延時間を表わす。

MOS論理回路の信号遅延時間は論理段数 N と3種の遅延要素から決定されるが、それぞれの要素は互に独立でないため、各要素間の干渉をきめ細かく分析しなければならない。以下、その過程を述べることにする。

(1) 論理段数 N の短縮

論理段数 N だけに注目すると従来方式ではキャリ・ルックアヘッド方式が最も少なく、16ビットのデータ加算におけるクリティカルパスは図2.10に示したように10段程度となる($N=10$)。これ以外に論理段数を減少させる方法は、加算演算を処理する回路の並列性を増すことに尽きる。回路規模を極端に増加させない論理上の工夫が必要である。

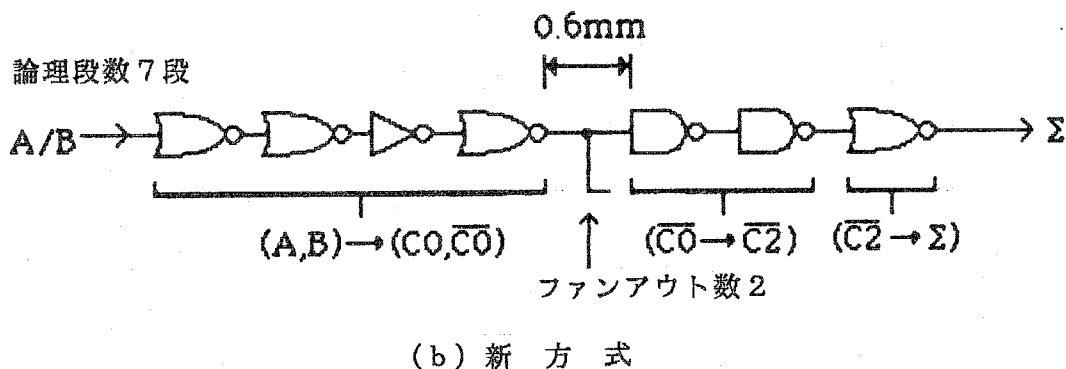
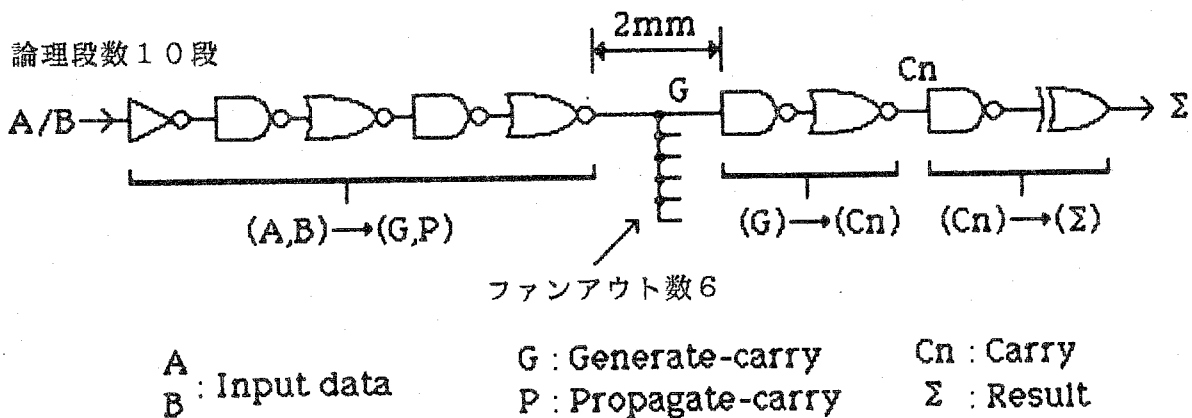
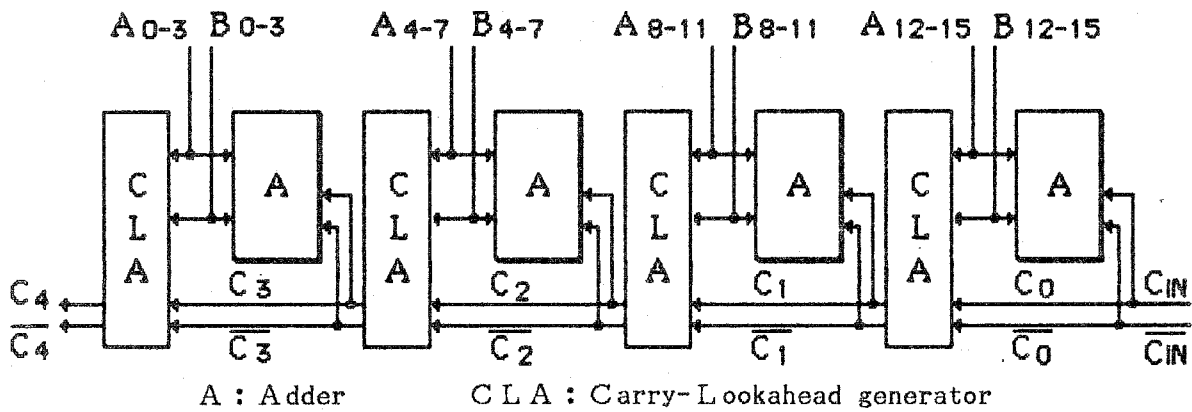


図2.10 加算回路におけるクリティカルパス

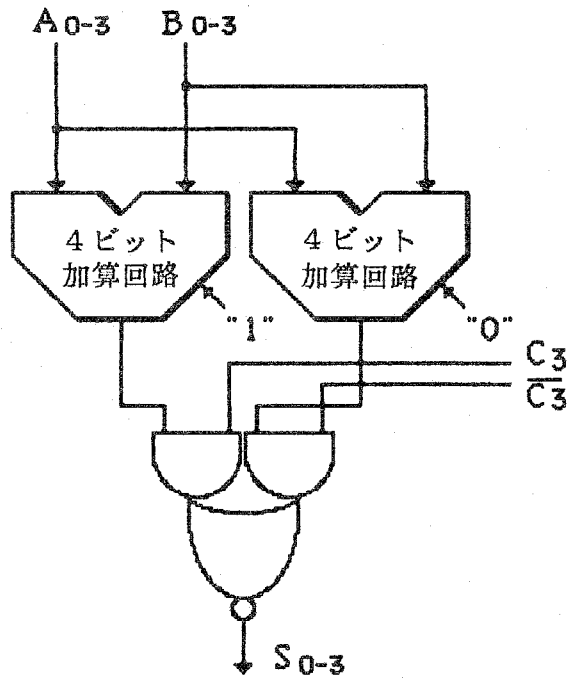
その具体的手段として次の3項目を提案する。

- (a) 加算部(A)と桁上げ発生部(CLA)を分離する(並列動作)
- (b) 下位からの桁上げ発生の起こる前に先行して2組の加算結果を準備する(加算部の2重化)
- (c) 正負両論理1対の桁上げ伝搬論理を設ける(桁上げ伝搬の高速化)

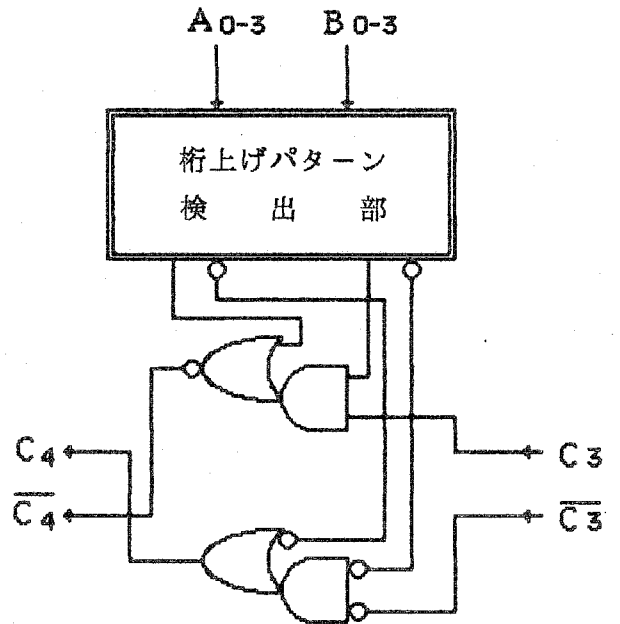
以上の3点を基本に構成した16ビット高速加算回路を図2.11に示す。



(a) 16ビット加算回路の構成



(b) A (Adder)



(c) CLA

図2.11 16ビット高速加算回路

本方式のクリティカルパスは図2.10及び図2.11に示したように7段となり、実用的な加算回路としては最小である。また、この回路方式は論理段数が少ないばかりでなく、配線領域をも少なくするのでLSIに向けた方式といえる。この点に関しては(3)項で詳しく述べる。

(2) t_{pd1} の短縮

各論理ゲートの出力容量 C_{out} は図2.12に示したように、MOSトランジスタのチャネル幅(W)とチャネル長(L)に依存し、 t_{pd1} はこの C_{out} 及び(W/L)値によって決まる。 (W/L) 値を大きくするとトランジスタの駆動能力が増加し、スイッチングが速くなるが、 C_{out} 及び入力容量 C_{in} は増加する。このため、最適な(W/L)値を選ばなければならない。本回路では回路全体のバランスを考え、これを0.5~2.0の範囲とした。

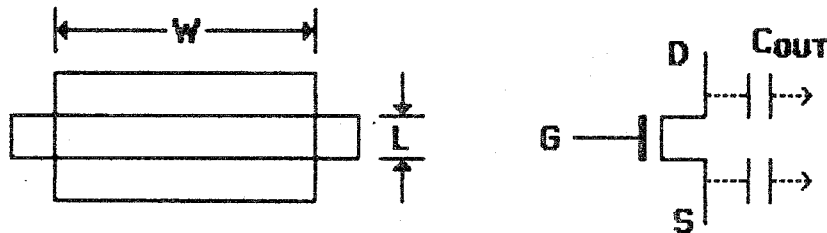


図2.12 MOSトランジスタの出力容量

(3) t_{pd2} , t_{pd3} の短縮

各論理ゲート間の配線長及びファンアウトは回路方式により大きく異なる。図2.10に示したように、キャリ・ルックアヘッド方式では最大配線長が約2mm、ファンアウト数が6となる大きな負荷が存在した。このため、配線による信号遅延、回路サイズへの影響が大きくなり、高速なMOS回路には適さない。これに対し、新しい方式では、4つのCLA回路によって回路を分散化した結果、最大配線長を0.6mm程度に抑え、かつファンアウト数を2とすることができた。従って、配線容量 C 、次段論理ゲートの入力容量 C_{in} の総和それぞれを共に1/3に減少し、 t_{pd2} , t_{pd3} の短縮を達成した。

以上の結果、計算機による回路解析では16ビットのデータ加算遅延時間を約76nsと推定し、これは実例データによっても確認された。一方、従来のキャリ・ルックアヘッド方式では150ns前後であり、2倍以上の高速化が図られたことになる。

2.4 RAS機能

ミニコンピュータに匹敵する性能を備え、制御用として用いられるマイクロプロセッサにとってR A S (Reliability, Availability, Serviceability)は重要な機能である。この機能がプロセッサを構成する半導体チップの中に集積されてゆくのは、丁度、DMA (Direct Memory Access)機能やシリアル通信インタフェース機能といった周辺I/O機能がプロセッサに付加され、システムオンチップ化してゆく方向と同様に進んでゆくものと思われる。本節では「制御用」を強く意識し、R A Sを高速処理性と同等の重要さでとらえている。

以下、R A S機能におけるプロセッサのエラー(誤動作)の検出法とその処理方法、特にパイプライン制御下での処理方法を述べる。

2.4.1 エラーの種類

プロセッサの誤動作を分析すると、その大半の原因は主メモリに直接あるいは間接的に関係している。従って、この誤りを検出し、処理することがR A Sの大きな役割である。本項で制御用マイクロプロセッサのR A S機能として最低限必要であると考えた誤り検出を挙げ、その処理方式を示す。

(1) パリティエラー(データ誤り)

主メモリに係るハードウェア障害が原因である。通常、8ビット毎に1ビットのパリティビットが付加されている。主メモリを構成するメモリL S I自身の障害(メモリのソフトエラーを含む)やプロセッサとの接続バス上のノイズ等多くの要因が考えられる。主メモリの制御回路で検出し、I F C Uに報告されて処理される。

最近ではE C C (Error Correcting Code)手法を用いて主メモリ側でのデータの信頼性を向上してきており、データ誤りの確率は低くなってきている。

(2) アドレスエラー(アドレス誤り)

ハードウェア或いはソフトウェア障害が原因であり、実装されていないメモリ空間をアクセスしたりする時に発生する。(1)の場合と同様に検出、処理される。

(3) インバリッドエラー(無定義命令誤り)

定義されていない(命令体系にない)命令コードをフェッチした時に発生する誤りであり、I F C Uにおいて命令解釈後に検出される。その結果はマイクロプログラム先頭アドレスに反映し、無定義のコードがあたかも1つの命令であるかのようにマイクロプログラム処理される。

(4) プロテクトエラー(主メモリ保護誤り)

主メモリの書き込み禁止領域(例えばOSのプロシージャ部)への書き込み指示を行った時に発生する。IFCU内のアドレス情報と保護すべき領域を定めたプロテクトレジスタの内容を比較・検出し、(1)と同様の手順で処理される。

(5) システム暴走

ソフトウェア、ハードウェアその他原因不明のエラーによるシステムの暴走を検出するにはプロセッサの動作を何らかの手段で常に監視しなければならない。そこで、本LSIプロセッサではウォッチドッグ・タイマと呼ぶハードウェア・タイマをRALU内に他の論理と切り放して配置した。本タイマは予め定めた一定時間内にソフトウェアによるイニシャライズ(初期化)が成されることを前提に使われ、もしハードウェアあるいはソフトウェアの誤りによってイニシャライズされない時にシステム異常とみなすものである。この結果は直ちにRALU外部に報告され、緊急停止等の処理が成される。

以上述べたものの他にマイクロプログラムメモリでのパリティエラー検出やマイクロプログラム動作における異常検出¹⁸⁾など数々の信頼性向上策がある。

2.4.2 エラー処理方法

IFCUとRALUが並列処理(パイプライン制御)している本LSIプロセッサでは前項におけるパリティ、アドレスエラーの発生時期が重要なポイントである。即ち、パイプライン状態か否かでエラー検出後の処理方法が分かれることになる。これは命令フェッチ中(パイプライン中)に発生するエラーによって実行中の前命令への外乱をなくすこととエラー発生時のLSI内部状態保存が目的である。内部状態保存はエラー発生要因をソフトウェア等により解明する際に必要である。

エラー検出後の2種の処理方法を図2.13に示し、以下にその内容を述べる。

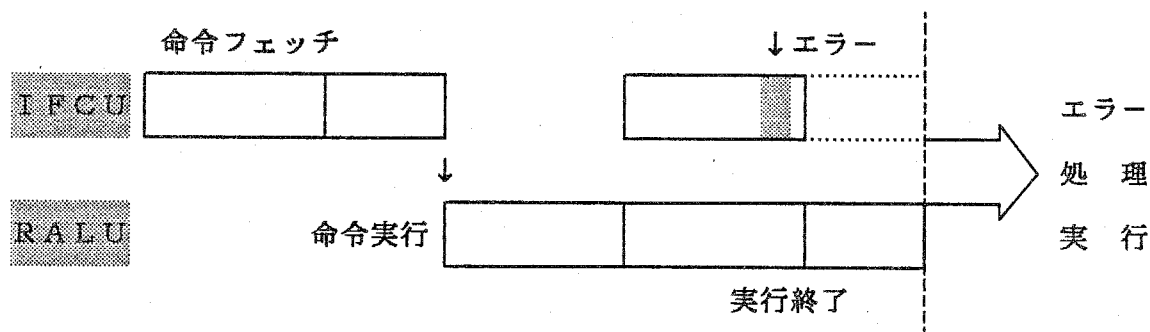
(1) パイプライン中のエラー

パイプライン中ではIFCUとRALUが共に動作しているわけであるから、メモリに関するエラーはIFCUが行なっている命令フェッチ中のエラーに限定される。この理由はRALUにおけるオペランドのフェッチがないことを前提にIFCUの動作を開始するからである。命令処理は、IFCUでの命令フェッチ後RALUへ制御が渡され、命令実行が開始される。従って、命令実行中に並行する次の命令フェッチでエラーが発生した場合、直ちにエラー処理へは行けず、実行中の前命令の実行終了を待って図2.6に示した

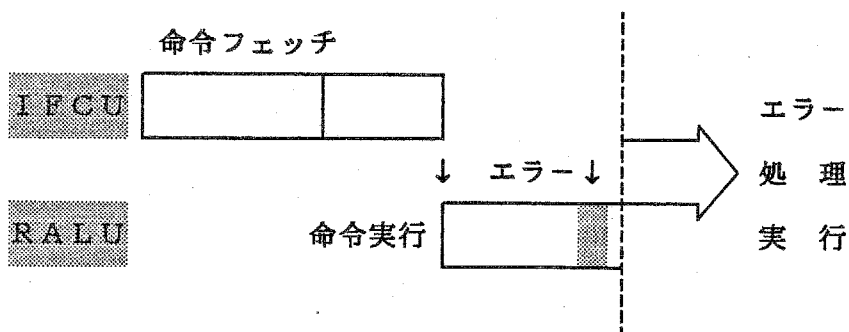
IFCUにおけるリセット端子(RESET)からシステム リセット信号を外部に送出する。これによりRARがリセットされ、0番地から開始するエラー処理用マイクロプログラムを起動し、エラーを処理する。エラー処理では、ソフトウェアによって要因などの解析が行なわれるが、LSIプロセッサの内部レジスタ群の内容を主メモリに掃き出し、エラー処理用ソフトウェアがこれを解析してエラーの原因を突き止める。従って、エラー発生時のLSIプロセッサ内部状態の凍結が重要なこととなる。本プロセッサでは、リセット端子に入力する信号がプロセッサの動作中か否(例えば、電源の投入時)かによって処理を分けられるように配慮している。

(2) 非パイプライン中のエラー

パイプラインが行なわれていない時にはIFCUは次の命令フェッチを開始していないわけであるから、エラーの原因はRALUの行なったオペランドのフェッチによるものである。従って、RALUでの命令実行終了を待つこと無しに、直ちにシステム リセットを行い、(1)と同様に処理する。



(a) パイプライン中のエラー



(b) 非パイプライン中のエラー

図2.13 パイプライン制御におけるエラー処理

2.5 L S I 構成と性能評価

(1) L S I 構成

パイプライン制御に加え R A S 機能を持つ I F C U 及び R A L U と呼ぶ 2 種の L S I の内部構成及びチップ写真をそれぞれ図 2.14, 図 2.15 に示す。これらの L S I は, 16 ビットマイクロプロセッサを構築する上で核の部分を作成するものである。初期の 8 ビットマイクロプロセッサの多くと同じ $5 \mu\text{m}$ NMOS プロセスによって製造されたものである。I F C U, R A L U それぞれの L S I 仕様は表 2.2 に示すとおりである。

単一 5 V 電源で消費電力が約 0.7 W, T T L コンパチブルの入出力特性をもっている。また, $0^{\circ}\text{C} \sim 70^{\circ}\text{C}$ の全領域で 3.3 MHz のクロック周波数(マシンサイクル 300 ns)で動作する。8 ビットマイクロプロセッサが同一の $5 \mu\text{m}$ プロセスで製造されており, この場合, 最高の機種でも 2 MHz であることから判断して, 本章で論じたチップ分割方式や高速回路技術がいかに効果があったかを示している。

表 2.2 プロセッサの L S I 仕様

項 目	I F C U / R A L U
プ ロ セ ス	$5 \mu\text{m}$ NチャンネルMOS
集 積 度	7,000 MOS トランジスタ
チ ッ プ ・ サ イ ズ	6 mm 角
動作周波数(マシンサイクル)	3.3 MHz (300 ns)
消 費 電 力	0.7 W
電 源	単一 +5 V
動 作 温 度	$0 \sim 70^{\circ}\text{C}$
パ ッ ケ ー ジ	42ピン セラミック

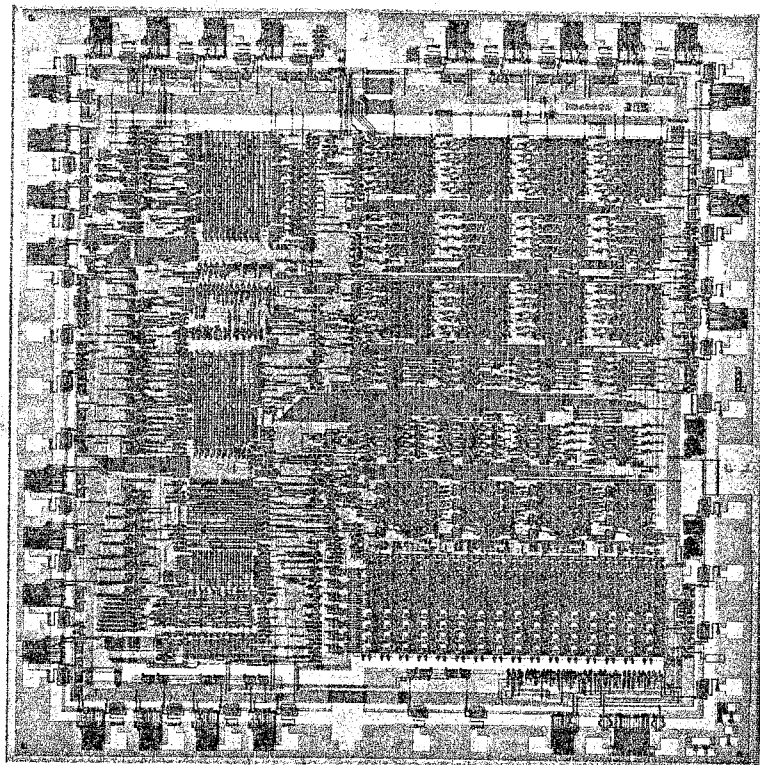
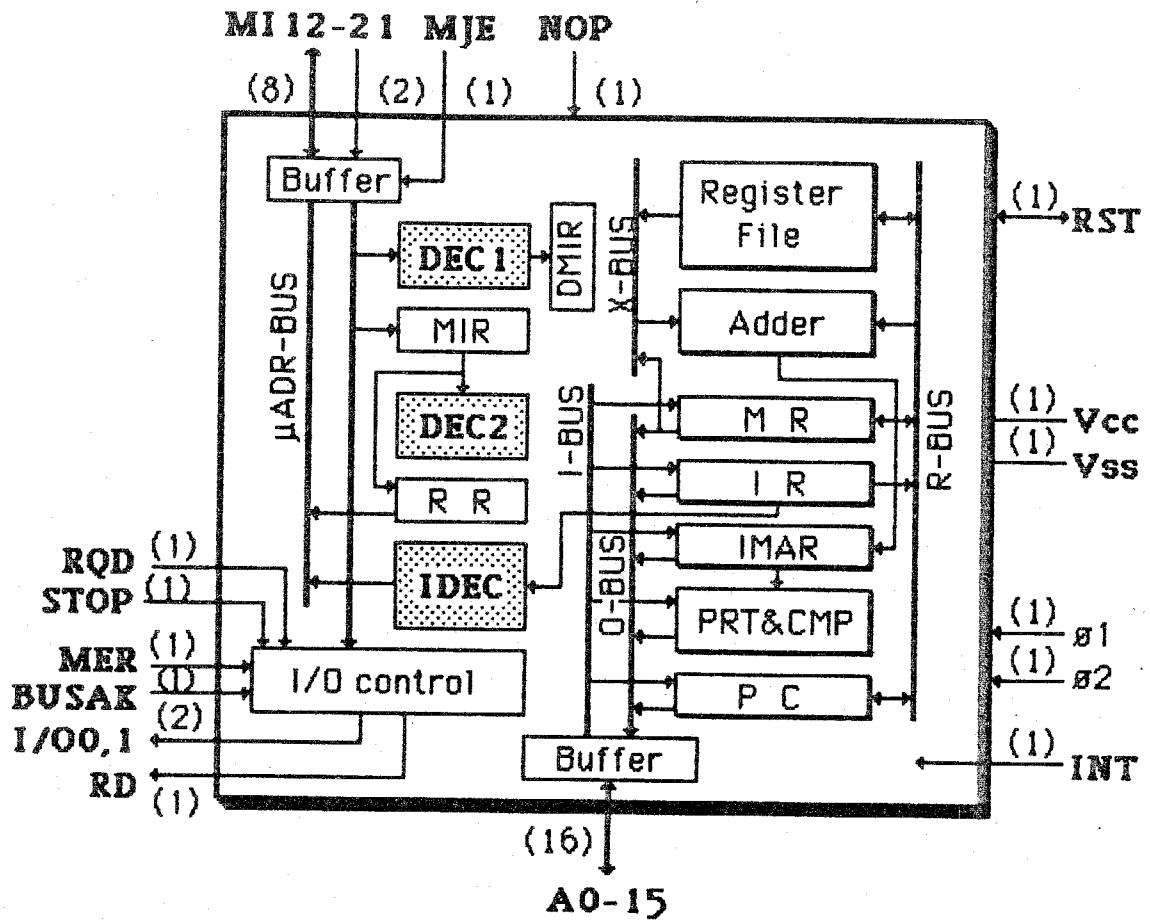


図 2.14 IFCUの構成及びチップ写真

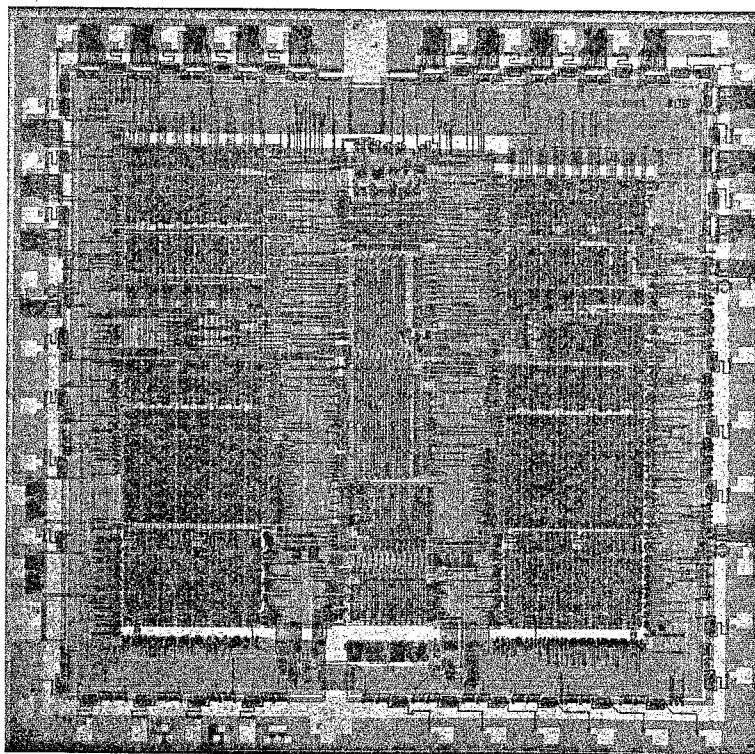
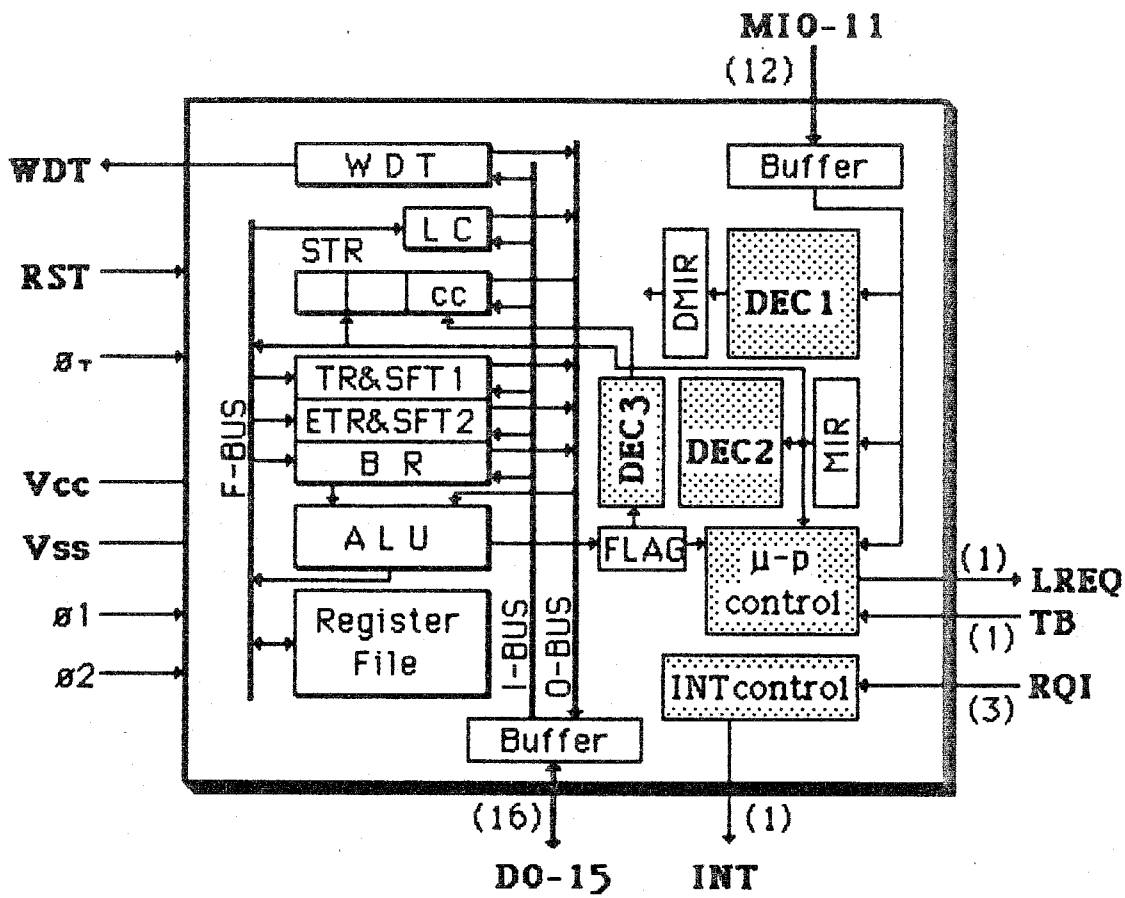


図 2.15 RALU の構成及びチップ写真

(2) 性能評価

命令実行時間はアクセス時間約525nsの主メモリを用いた場合、レジスターメモリ間の16ビット加算命令が2.3 μ s, その他の実行時間は表2.3に示したとおりである。これを総合的に評価すると、科学計算における平均命令実行時間(Scientific Gibson Mix)が6.8 μ s, OSにおける平均命令実行時間(System Mix)が3.2 μ sとなる。これらの数値は素子スピードがMOSのそれよりも数倍速いバイポーラ素子で構成したプロセッサの性能を20%向上している。

表2.3 命令実行時間(メモリアクセス525ns)

項 目	加算(μ s)	乗算(μ s)	除算(μ s)
固 定 小 数 点	2.3	11.4	15.3
浮 動 小 数 点	22.8	44.4	66.8
ギブソン・ミックス	6.8 μ s		
システム・ミックス	3.2 μ s		

以上のように、5 μ m程度のMOSプロセスであってもLSIプロセッサ構成法及び回路技術の追求によって、ミニコンピュータの性能・機能をLSIに集約することができる。このようなチップ分割法は微細化が進んでも有効な手段として位置づけられる。

2.6 まとめ

制御用16ビットマイクロプロセッサのプロセッサ構成法について述べた。マルチチップ形のLSIプロセッサ構成法はコンピュータ・アーキテクチャ(命令体系等)同様に、プロセッサの処理性能に与える影響が大きい。高機能を得るため、命令処理の流れに注目し、命令フェッチ用LSI(IFCU)と命令実行用LSI(RALU)の2種LSIに分割してパイプライン制御し、処理の並列性を増すことで実現した。また、LSI内部では加算回路において、キャリ的高速伝搬方式を示し、マイクロレベルでの高速化も図った。この結果、LSIプロセッサは300nsのマシンサイクルで動作し、16ビットのレジスターメモリ間加算2.3 μ s, ギブソン・ミックス6.8 μ sの高速性能を達成した。更に、制御用マイクロプロセッサとして重要な高信頼度を得る諸機能(RAS)のオンチップ化を果たした。

2種のLSIを含むマイクロプロセッサは、既存のミニコンピュータと同一の命令体系

を持ちながら1ボードで構成でき、高級言語、アセンブラ、各種ユーティリティのソフトウェア・サポート、入出力装置のレパートリも既に準備されている。更に、ファームウェア技術により標準のコンピュータとして動作中に問題向きプロセッサ(シーケンサ、DDC用コントローラ等)への切り換えも可能となり、多彩なアプリケーションに適応するものとなっている¹⁵⁾。

尚、本章で述べたLSIは、制御用16ビットマイクロプロセッサHIDIC08-LのカスタムLSIとして開発されたものである。

2.7 第2章の参考文献

- (1) 平子外：“制御用分散処理システム”，情報処理，Vol.20，No4，pp.346～349，(昭54)
- (2) 平井外：“制御用計算機における分散処理ネットワーク技術の動向”，日立評論，Vol.60，No7，pp.1～6 (昭53)
- (3) 桑原外：“制御用コンピュータのネットワーク・システム”，電気学会雑誌，Vol.98，No3，pp.199～203 (昭53)
- (4) E.D.Jensen：“The Honeywell Experimental Distributed Processor -An Overview”m Computer，Vol.11，No1，pp.28～38 (1978)
- (5) M.J.Sebern：“A minicomputer compatible micro-computer system，The DEC LSI-11”，Proceedings of IEEE，Vol.65，No6 (1976)
- (6) S.P.Morse et al.：“The Intel 8086 Micro Processor：A 16-bit Evolution of the 8080”，Computer，Vol.11，pp18～27 (1978-5)
- (7) M.Shima：“Two Versions of 16-bit chip span microprocessor，minicomputer needs”，Electronics，Vol.51，No26，pp.81～88 (1978-6)
- (8) E.Stritter et al.：“A Microprocessor Architecture for a Changing World：The Motorola 68000”，Computer，Vol.12，No2，pp.43～52 (1979-2)
- (9) J.A.Bayliss et al.：“The Instruction Decoding Unit for the VLSI 432 General Data Processor”，IEEE Journal of Solid-State Circuits，Vol.SC-16，No5，pp.531～537 (1981-10)
- (10) D.L.Budde et al.：“The Execution Unit for the VLSI 432 General

- Data Processor”, IEEE Journal of Solid-State Circuits, Vol.SC-16, No.5, pp.514~521 (1981-10)
- (11) R.Mateosian : “System Consideration in the NS 32032 Design”, Proceedings of 1984 NCC, pp.77~81 (1984-7)
 - (12) D.MacGregor et al. : “The Motorola MC68020”, IEEE Micro, Vol.6, No.8, pp.101~108 (1984-8)
 - (13) 前島外 : “制御用16ビットマイクロコンピュータのアーキテクチャ”, 情報処理学会論文集, Vol.21, No.3, pp.208~215 (昭55-5)
 - (14) W.N.Johnson : “A VLSI Superminicomputer CPU”, ISSCC Digest of Technical Papers, Vol.27, pp.174~175 (1984-2)
 - (15) 増田, 前島外 : “DDCへのファームウェアの応用と処理能力の評価”, 計測自動制御学会論文集, Vol.17, No.2, pp.57~62 (昭56-4)
 - (16) 今井, 前島外 : “制御用マイクロコンピュータHIDIC08”, 日立評論, Vol.59, No.5, pp.397~402 (昭52-5)
 - (17) 保田外 : “制御用マイクロコンピュータHIDIC08-E”, 日立評論, Vol.60, No.10, pp.65~70 (昭53-10)
 - (18) 前島外 : “高集積マイクロコンピュータに適したマイクロプログラム制御方式”, 情報処理学会論文集, Vol.23, No.1, pp.16~24 (昭57-1)

第3章 高集積・高速マイクロコンピュータの 構成法

第3章 高集積・高速マイクロコンピュータの構成法

3.1 まえがき

MOS技術の進歩に伴い、プロセッサに加え主メモリ用のROM、RAMや周辺I/O機能から成るシステムをわずか数ミリ角のシリコン・チップ上に集積した4~16ビットのシステムオンチップ形のマイクロコンピュータ(System Integration)^{1)~4)}が出現している。これらのLSIはほとんどが微細なMOSプロセスを用いており、数万から十数万素子を1チップ上に集積するVLSI(Very Large Scale Integration)の領域に達しているものも多い。

このような状況の下では、膨大な論理がゆえに設計作業はなかなか収束せず、高集積化に伴って開発期間がますます延びてしまう。設計作業の単純化あるいは機械化がVLSI設計技術として重要なファクタになりつつある。この段階ではもはやデバイスの問題ではなく、VLSIに向けたプロセッサ構成法の問題に帰着する。マイクロプロセッサのような論理LSIにおいてもメモリLSIと同様に規則性を持ち、単純かつ整然とした構造が要求される。

そこで本章ではシステムオンチップ形マイクロコンピュータにおいて核となるマイクロプロセッサを例として、規則論理構造を実現しながらも高速性を与える新しいマイクロプログラム制御方式、及びプロセッサの小形化に有効なマイクロプログラム圧縮法について示す。更に、バス指向のデータパス構成を高速性とより柔軟性を持たせる方法で構成する手法を述べる。

3.2 システムオンチップ形マイクロコンピュータの概要

3.2.1 利用形態から見たチップ構造

システムオンチップ形マイクロコンピュータはシステム規模の比較的小さい分野、例えばポータブルなパーソナル・コンピュータやワードプロセッサにおけるデータ処理や自動車エンジン、プリンタ等の機器制御に応用されて、システムの実装規模をよりコンパクトにする効果をもつものである。利用形態から見ると、図3.1に示したように2つの形態に大別できる。

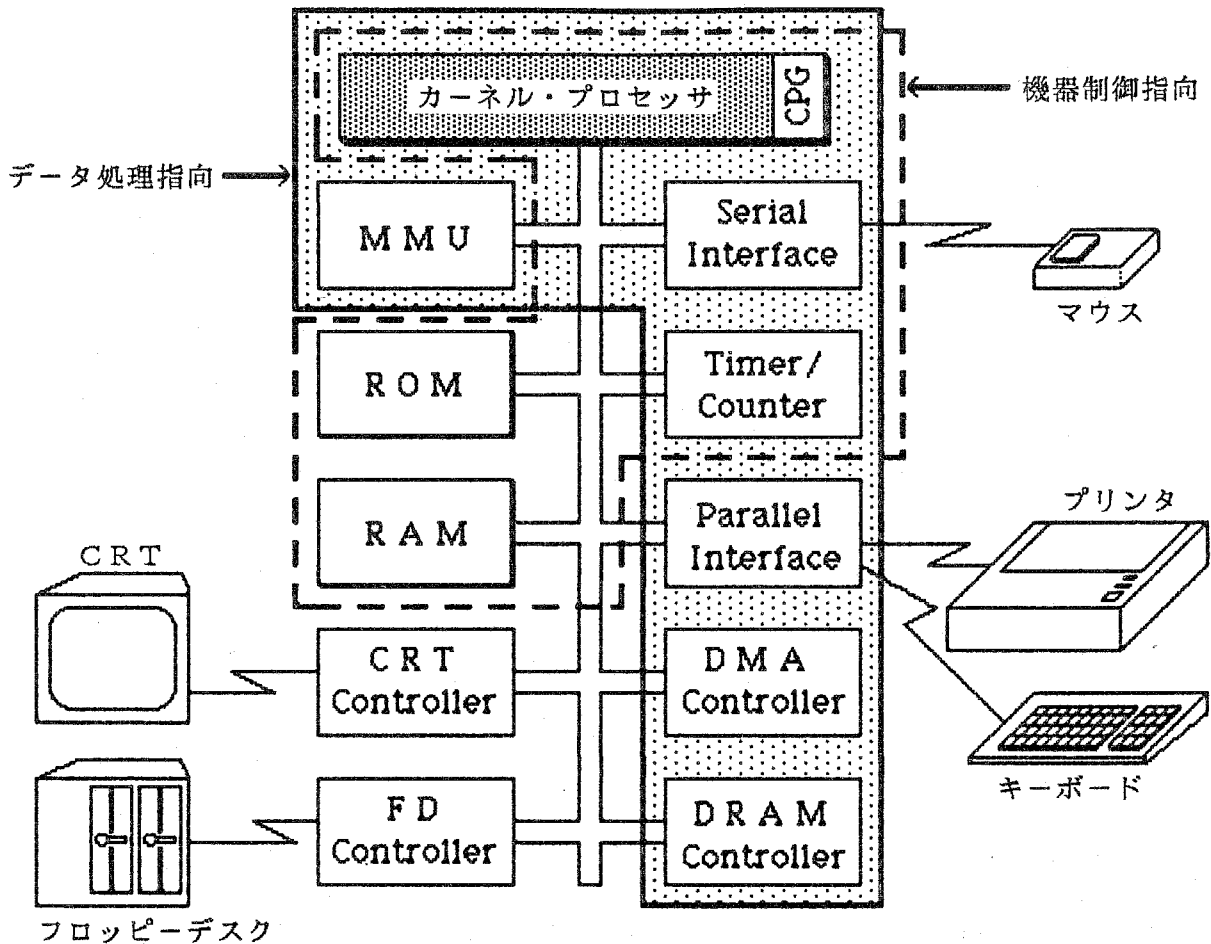


図 3.1 システムオンチップ形マイクロコンピュータの構成

(1) データ処理指向

パーソナル・コンピュータやワードプロセッサのように莫大なデータを扱うものでは、大容量の主メモリが必要である。そのため、主メモリとなるROMやRAMは内蔵せずオフチップとし、代りに制御に関する機構をオンチップ化する傾向にある。即ち、主メモリ空間を拡張するメモリ管理機構(MMU: Memory Management Unit)、DMAコントローラ、DRAMコントローラ等をオンチップ化する。シリアル・インタフェース、タイマ/カウンタはほとんどの用途に必須となっている。

(2) 機器制御指向

自動車エンジンやプリンタのようにメカを伴うものでは、プログラムやデータの容量がシステム設計時点で決まり、この容量が小さければROM、RAMをオンチップ化するとシステムの小型化に効果的である。この場合、マスクROMあるいはユーザ・プログラム

が可能なEPROM(Erasable Programmable ROM)を用いた機種も考えられる。このような用途でもシリアル・インタフェースやタイマ/カウンタは必須となる。更に、パラレル・インタフェースも重要となってくる。

以上のいずれの形態でも、システムオンチップ形マイクロコンピュータの核となるマイクロプロセッサ(カーネル・プロセッサ)は、システムの性能、機能及び規模を決める重要な位置にある。次項ではカーネル・プロセッサに要求される事項を明確にする。

3.2.2 カーネル・プロセッサ構成上の必須事項

(1) 高速性

プロセッサの性能は、デバイスの性能と論理構造(マイクロチップ・アーキテクチャ)に依存する。ここではマイクロプログラム制御機構及びデータパス機構を高速化する必要性を示す。

(a) マイクロプログラム制御機構の高速化

図3.2は従来のマイクロプログラム制御機構を示したもので、先ず、高速性から見た問題点を明らかにする。

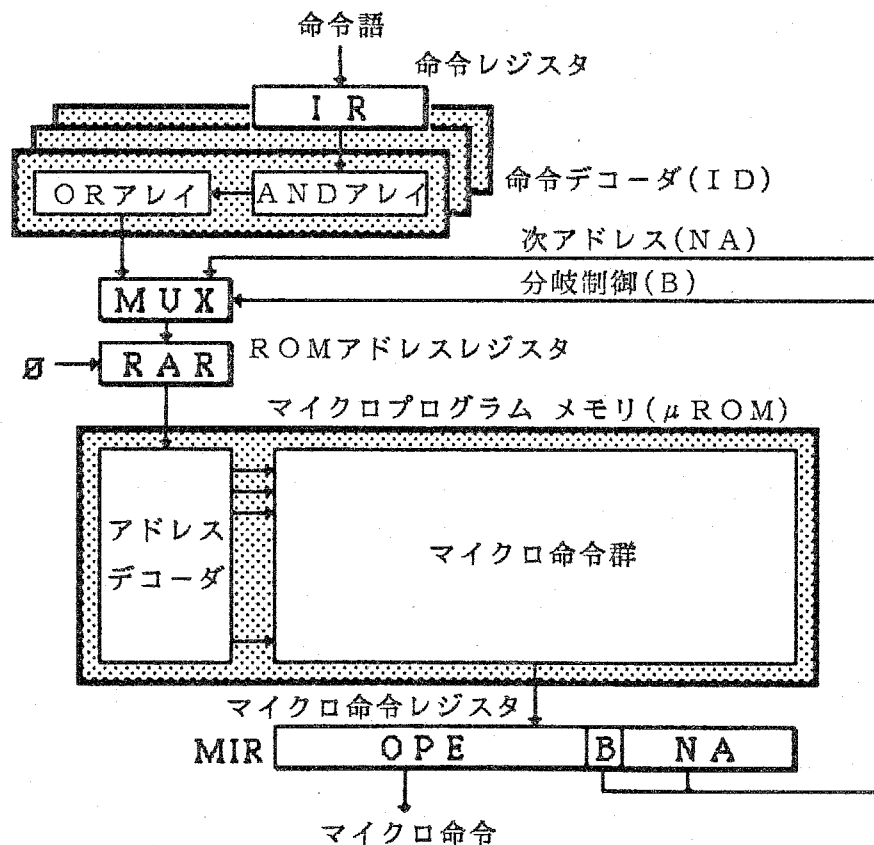


図3.2 従来のマイクロプログラム制御機構

この構成では、まず、命令語が命令レジスタ (IR) に読み出され、命令デコーダ (ID) によって解釈される。この命令デコーダを必要とするのは、マイクロプロセッサでは命令のコード効率を高めるために相当に命令コードを圧縮しているため、これを分析する必要があるからである。解釈結果はマイクロプログラムメモリ (μ ROM) 中に格納される命令処理手順を示すマイクロプログラムの先頭アドレス (エントリ・アドレス) として得られる。これを ROM アドレスレジスタ (RAR) に格納して命令処理の先頭のマイクロ命令の 1 つ (エントリ・マイクロ命令) をマイクロ命令レジスタ (MIR) に読み出す。このマイクロ・シーケンスのタイムチャートを図 3.3 に示したが、これから明らかなように、一連の命令実行に対して、命令語の解釈期間 (ID の信号遅延時間) が生じてしまう。これは命令実行に要するマイクロサイクル数を常に 1 サイクル増加する。従って、マイクロサイクル数の少ない命令の多いマイクロプロセッサにとっては不都合なものとなり、プロセッサの高速性に大きな影響を与える。マイクロサイクル数の実行的な短縮はパイプライン制御により可能である。しかし、コンパクトなマイクロプロセッサの実現には、自由度の高い回路を駆使できる LSI の特長を活かすことが論理設計の単純化の見地からも理想である。こうした発想から命令解釈機能を省略して、この機能を ROM 構造の中に一体化して内蔵した μ ROM 構成法を提案する。

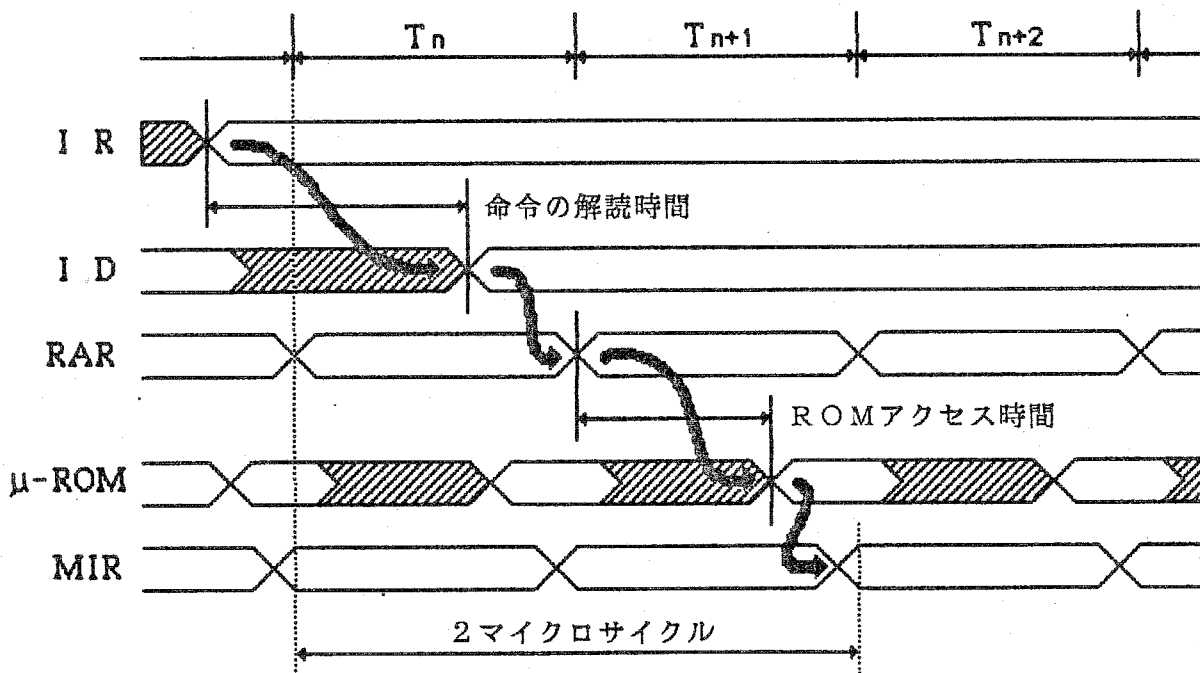


図 3.3 従来のマイクロプログラム制御機構のタイムチャート

(b) データパス機構の高速化

マイクロプロセッサのマイクロサイクル時間は、マイクロプログラム制御機構における μ ROMのサイクル時間に合せ、データパス機構における処理内容をそれに合わせるようにする。1マイクロサイクルでの演算内容を深くし、マイクロプログラムのステップ数を減らし高速化しようとする、この動作によりマイクロサイクル時間が大きくなってしまふ。また、1マイクロサイクルでの演算内容を浅くし、マイクロサイクル時間を縮めようとする、マイクロプログラムのステップ数が増加して性能を落とすことになる。この相反する項目を解決するため、データパス機構に対しては1マイクロサイクルでの演算内容を深くしながらもマイクロサイクル時間の短縮を図る必要がある。そこでマイクロ命令実行におけるパイプライン(マイクロ・パイプライン)制御技術を提案する。

(2) 小形化

マイクロコード化プロセッサで常に議論されることであるが、マイクロプログラムメモリのメモリ効率の問題がある。このメモリ効率を高めるためナノプログラム方式が使われている例がある⁵⁾。しかし、この方式は制御機構を複雑にするだけでなく、命令フェッチからマイクロ命令読み出しまでの応答時間がナノプログラムメモリのアクセス分遅くになってしまう欠点もある。簡素かつ小形のマイクロプロセッサ実現のためにはマイクロプログラムメモリ上でのメモリ効率を向上する方法が必要となる。そこで、長短2レベルのマイクロプログラム方式と、複数のマイクロ命令から1つのマイクロ命令を合成して作成する方式によって冗長なマイクロコードを排除する手段を提案する。

以上の必須事項とその解決策を次節以降に詳細に述べる。

3.3 プロセッサの命令制御部構成

3.3.1 マイクロプログラム制御方式

図3.2で示したマイクロプログラム制御構造における命令デコーダ(ID)は、命令語を実行するマイクロプログラムのエントリ・アドレスに変換する機能をもつものである。即ち、命令語の内容に応じて、マイクロプログラムメモリのいずれかアドレスに「マッピング」するテーブル(分岐テーブル)である。ところで、命令語の読み出し(フェッチ)からマイクロ命令の読み出し(マイクロ命令フェッチ)までの一連の流れに注目すると、2回の情報変換がなされていることが分かる。即ち、第1は命令語からマイクロプログラムの

先頭アドレスへの変換であり、第2は先頭アドレスからマイクロ命令への変換である。これを命令語からマイクロ命令への直接変換に置き換えれば、図3.2における命令デコーダ(ID)を省略できる。このような命令からマイクロプログラムへの直接マッピングを行うプロセッサ⁶⁾は既にあるが、本章で提案するマッピング方式の特徴はLSI上の回路構成の自由度を活かした μ ROM構造⁷⁾にある。即ち、LSIでは限られたチップ面積の中に所望の機能を集積しなければならない反面、目的に合せて回路設計が行えるという大きな利点がある。そこで、PLA(Programmable Logic Array)とROMを一体化し、アドレスとこれに対応するマイクロコードを同時設計することで効率のよい μ ROM構造の実現を図るものである。

以下、直接マッピング方式のマイクロプログラム制御構造及びその原理を1つの命令形式を例にとって述べる。

(1) 命令形式

図3.4に8ビットのマイクロプロセッサの命令形式を一例として示す。第1語目は各種演算モードを指定するOP(Operation)と、演算レジスタを指定するR(Register)と、各種アドレッシング・モードを指定するMOD(Modifier)の3つのフィールド、第2語目はアドレスあるいはイミディエイト・データとなる“m”フィールドから成る。この命令形式を解釈しマイクロ動作の制御を行うマイクロプログラム制御構造を示す。

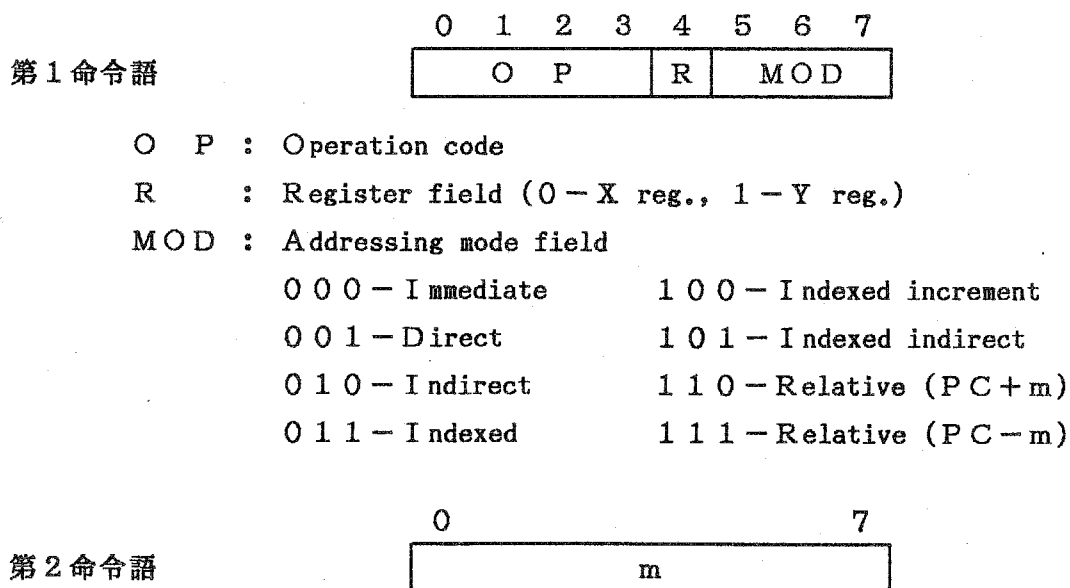


図3.4 8ビットマイクロプロセッサの命令形式

(2) マイクロプログラム制御機構の構成

前記したように、ここで提案する制御機構における μ ROMでは命令の解釈機能を含むため、図3.5に示す2つの部分に分けられる。第1の部分は命令解釈を行うデコード領域であり、第2の部分は通常のマイクロ動作を行うワーク領域である。後者については従来の構造(図3.2)と全く同じ意味をもつ。そこでここでは省略し、前者についてのみ述べることにする。

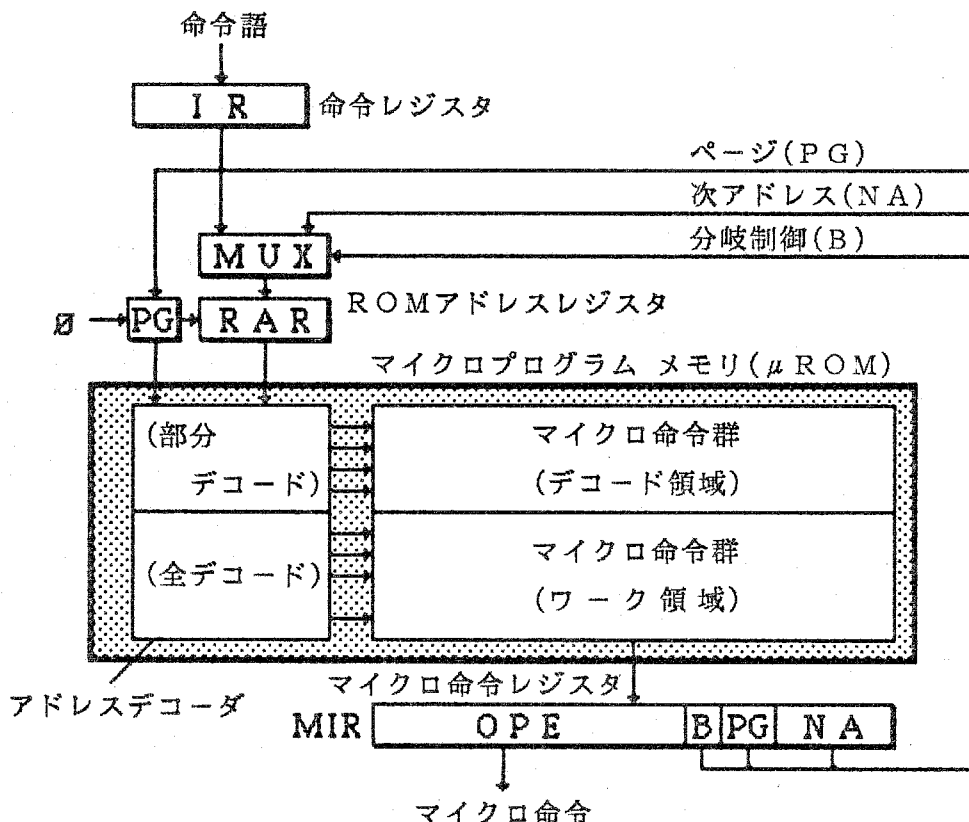


図3.5 新しいマイクロプログラム制御機構

本構成におけるデコード領域とワーク領域との違いを先ず述べる。

(a) デコード領域

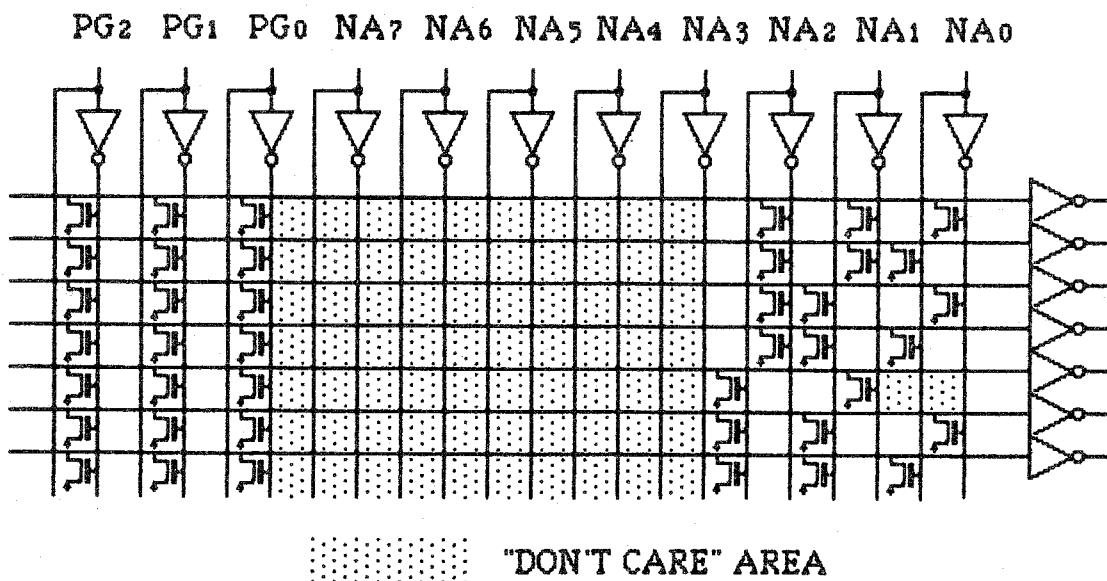
デコード領域は、各種命令を実行するためのマイクロプログラムの先頭のマイクロ命令(エントリ・マイクロ命令)を格納する部分である。従って、この部分のアドレスデコーダは、命令コード中のデコードに必要な部分だけを切り出して他をマスクしなくてはならない。そのため、命令レジスタからマルチプレクサ(MUX)とROMアドレスレジスタを経由して入力する命令コードの一部だけを解釈する部分デコード機能を持っている。この機能は丁度PLAのANDアレイと同じ機能である。また、マイクロ命令群の格納されて

いる部分はPLAのORアレイに相当するものである。図3.4に示した命令形式を解釈する際には、OP, R, MODの各フィールドが互いに独立しているため、命令語のコードから1つのフィールドをだけをデコードするようにしている。即ち、命令の内容に応じてエントリのマイクロ命令が決定されるデコード・サイクルにおいては、PLAで構成したデコード領域に配置したマイクロ命令の1つを選択して読み出す。図3.6にMODフィールドのデコード例を示す。

ページ	命令コード	マイクロ動作
2 1 0	7 6 5 4 3 : 2 1 0	
0 0 0	x x x x x 0 0 0	No Operation
0 0 0	x x x x x 0 0 1	MAR ← MDR
0 0 0	x x x x x 0 1 0	MAR ← MDR
0 0 0	x x x x x 0 1 1	MAR ← I X R + MDR
0 0 0	x x x x x 1 0 x	MAR ← I X R + MDR
0 0 0	x x x x x 1 1 0	MAR ← PC + MDR
0 0 0	x x x x x 1 1 1	MAR ← PC - MDR

don't care部 MODフィールド

(a) デコード例



(b) デコード回路

図3.6 MODフィールドのデコード

このデコードには、OP、Rフィールド(ビット0~4)は不要であり、その部分はアドレスデコーダにおいて無視する“don't care”ビットとしてハード的にマスクをかけている。その実際の回路イメージも同図に示してある。

(b) ワーク領域

ワーク領域は、前記したデコード領域から読み出されたエントリ・マイクロ命令に続くマイクロ命令群を配置する部分である。この領域中にあるマイクロ命令を選びだすのは、まず、エントリのマイクロ命令であり、その指定はマイクロ命令中の次アドレス・フィールドである。従って、この部分でのアドレスデコーダでは、ROMアドレスレジスタから入力するコードを全てデコードする全デコードでよい。即ち、通常のROMでよい。但し、再び命令コードによる分岐を必要とした時はデコード領域へ戻る。

これら2種類の領域での μ ROMアクセス方法の組合せによりマイクロ動作が実行される。ここで、命令コードによるアクセスの際の命令中のデコードすべきフィールドの違いは、 μ ROMのデコード領域を論理的に分割することで達成される。また、マイクロ命令によるアクセスは、ワーク領域を各デコード領域と論理的に分けることで円滑に行われる。これらの μ ROMの論理的分割は、「ページ」の概念で実現される。

以上のように、 μ ROMとしてデコード領域にPLA構造を、ワーク領域にROM構造を用い、これらを1つのROMとして合成することで命令解読機能を μ ROM中に埋設することができる。図3.7に本マイクロプログラム制御構造での動作タイミングを示す。

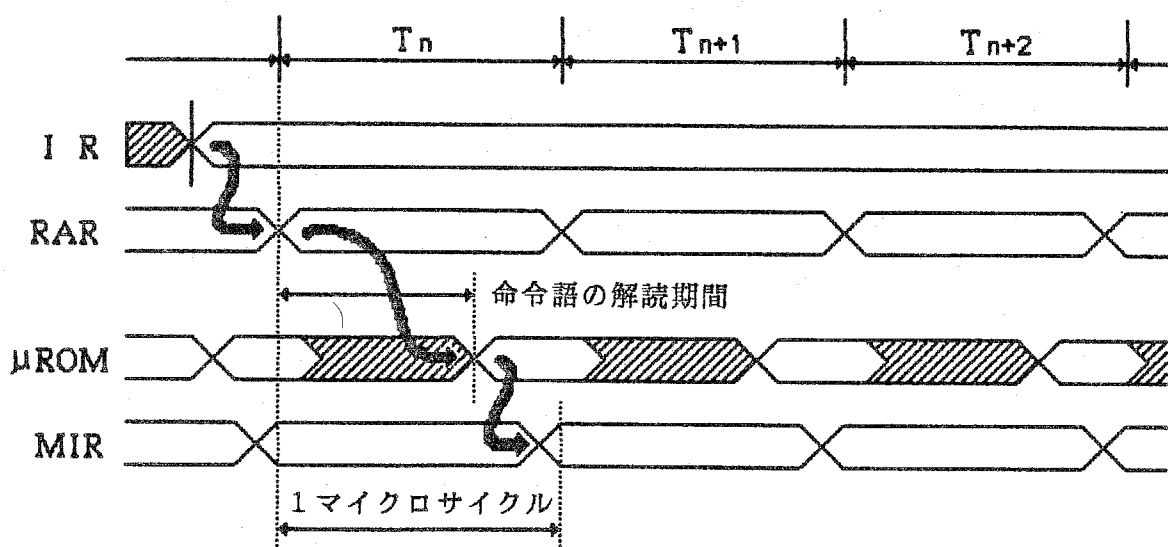


図3.7 新マイクロプログラム制御方式でのタイムチャート

これから明らかなように、図3.3に含まれていた命令デコードに要する1マイクロサイクル分がなくなる。このことは命令語が命令レジスタ(IR)に取り込まれ、1マイクロサイクル後にはマイクロ命令が実行できることを意味する。

以上に示したとおり、我々の提案する新しいマイクロプログラム制御方式は従来の方式における命令デコーダをマイクロプログラムメモリ中のアドレスデコーダに埋設したものである。図3.8は μ ROMにおけるデコード・ページ概念を明確に示したものであるが、新しい構造では従来構造における命令デコーダを構成するANDアレイを取り出してこれを μ ROMのアドレスデコーダの一部に置き換えたものとみることができる。一般に、命令処理では1つの命令コードの中に含まれる各種の情報(命令の種類、アドレッシングのモード等)を必要とするため、その都度命令をデコードしなければならない。マイクロプログラムメモリにおけるページ概念はこうした要求を満足するのに最適である。即ち、命令中でデコードを必要とする部分だけをデコードする図3.6の回路やデコードの意味によってページを分ける図3.8の構造が効果を発揮するのである。

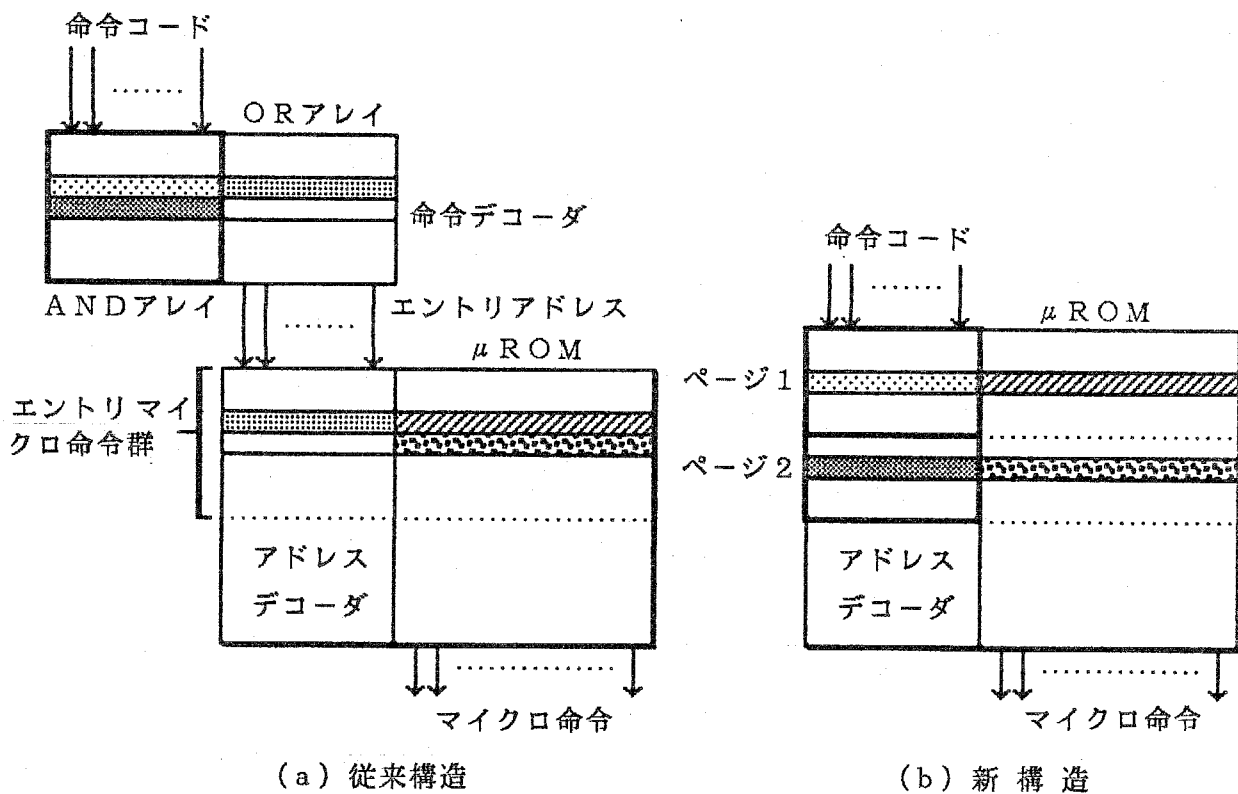


図3.8 μ ROMデコード・ページ概念

3.3.2 命令解説手順とマイクロプログラミング

(1) マイクロ命令仕様

図3.9はマイクロ命令形式を示したもので、全体で32ビットである。ここでは命令解説機能一体形の μ ROM構成におけるマイクロ・シーケンス制御に関するフィールドについてのみ述べる。

(a) B (Branch type) [ビット21]

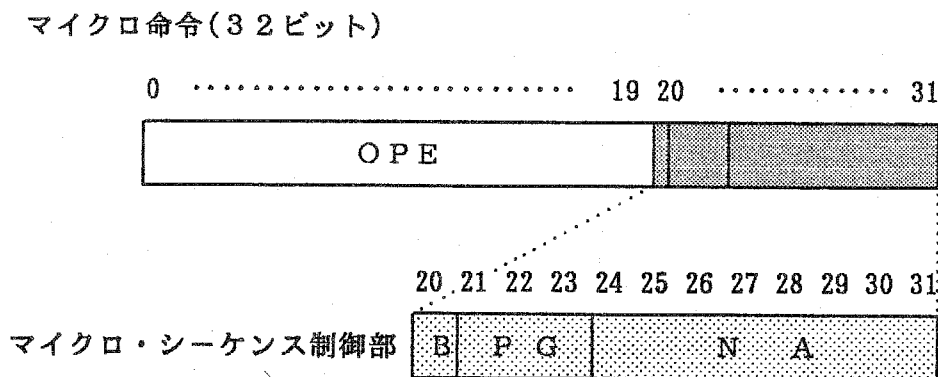
μ ROMの2つのアドレス源のいずれかを選択するフィールドで、命令コード(8ビット)あるいはマイクロ命令の後述するアドレス・フィールド(8ビット)の一方が選ばれる。

(b) PG (Page) [ビット22~23]

μ ROMにおける各種マイクロ命令の存在するページを示すフィールドである。マイクロ命令が次のマイクロ命令へ分岐して行く際に、どのような命令デコード(あるいはアドレスデコード)を行うかを指示する。図3.4の命令形式では、OP, R, MODの3つの命令デコード・ページと1つのワーク・ページが必要である。この場合、 μ ROMは4つのページに分離すればよい。

(c) NA (Next Address) [ビット24~31]

μ ROMのワーク・ページ内アドレスを示すフィールドであり、最大256語である。

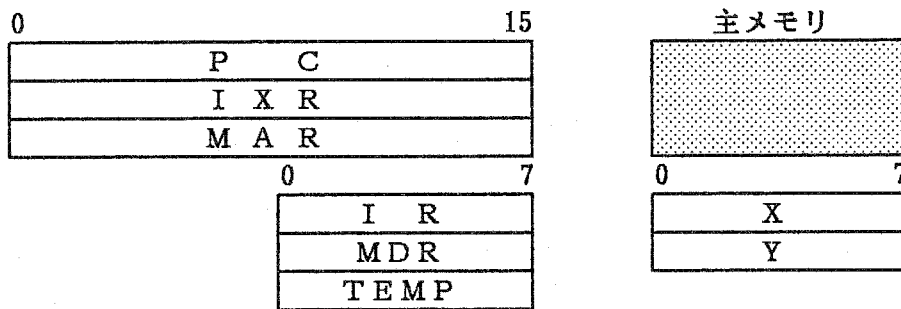


O P E : Operation control
B : Branch type selection
P G : Page control
N A : Next Address

図3.9 マイクロ命令形式

(2) ページ制御

図3.4の命令を、図3.5のマイクロプログラム制御構造及び図3.10に示すレジスタ構成のプロセッサで実行する場合を想定して、マイクロ命令のページ制御原理とそのフローを示す。この場合のマイクロ動作を含むマイクロ・シーケンスのフローは図3.11に示してある。



- | | | | |
|-----|---------------------------|------|------------------------|
| PC | : Program Counter | IXR | : Index Register |
| MAR | : Memory Address Register | IR | : Instruction Register |
| MDR | : Memory Data Register | TEMP | : Temporary register |
| X | : X register | Y | : Y register |

図3.10 8ビットマイクロプロセッサのレジスタ構成

(a) 命令フェッチ・サイクル

プログラムカウンタ(PC)の内容に従い、主メモリから命令の2語目までを読み出す。命令フェッチは全ての命令で共通であるため、命令語の内容に無関係である。従って、次に示す2つのマイクロ命令は μ ROMのワーク・ページ(ページ0)に配置される。

- (i) $MAR \leftarrow PC, PC \leftarrow PC + 1, M(R), IR \leftarrow (M),$
PAGE $\leftarrow 0$ (第1命令フェッチ)
- (ii) $MAR \leftarrow PC, PC \leftarrow PC + 1, M(R), MDR \leftarrow (m),$
PAGE $\leftarrow 1$ (第2命令フェッチ)

第2のマイクロ命令でページを1に換えるのは、次のオペランド・フェッチサイクルが命令に依存するからである。

(b) オペランド・フェッチサイクル

図3.4に示したように、第1の命令語中のMODフィールドでは8種類のオペランド・

フェッチ形式が指定され、その実行は μ ROM のページ 1 に格納されたマイクロ命令によって成される。ページ 1 のアドレスデコーダはビット 5 ~ 7 の 3 ビットだけをデコードする (図 3.6 参照)。間接アドレッシング (第 2 語目の指す主メモリ内データがアドレス) の場合でのオペランド・フェッチ処理を次に示す。

- (iii) $MAR \leftarrow MDR, M(R), MDR \leftarrow (M),$
 $PAGE \leftarrow 0$ (オペランド・アドレスのフェッチ)
- (iv) $MAR \leftarrow MDR, M(R), MDR \leftarrow (M),$
 $PAGE \leftarrow 2$ (オペランド・フェッチ)

第 3 マイクロ命令でのページ 0 指定は、次のマイクロ命令が NA フィールドによって決まるためである。また、第 4 マイクロ命令でページ 2 を指定するのは次のレジスタ読み出しサイクルに移るためである。

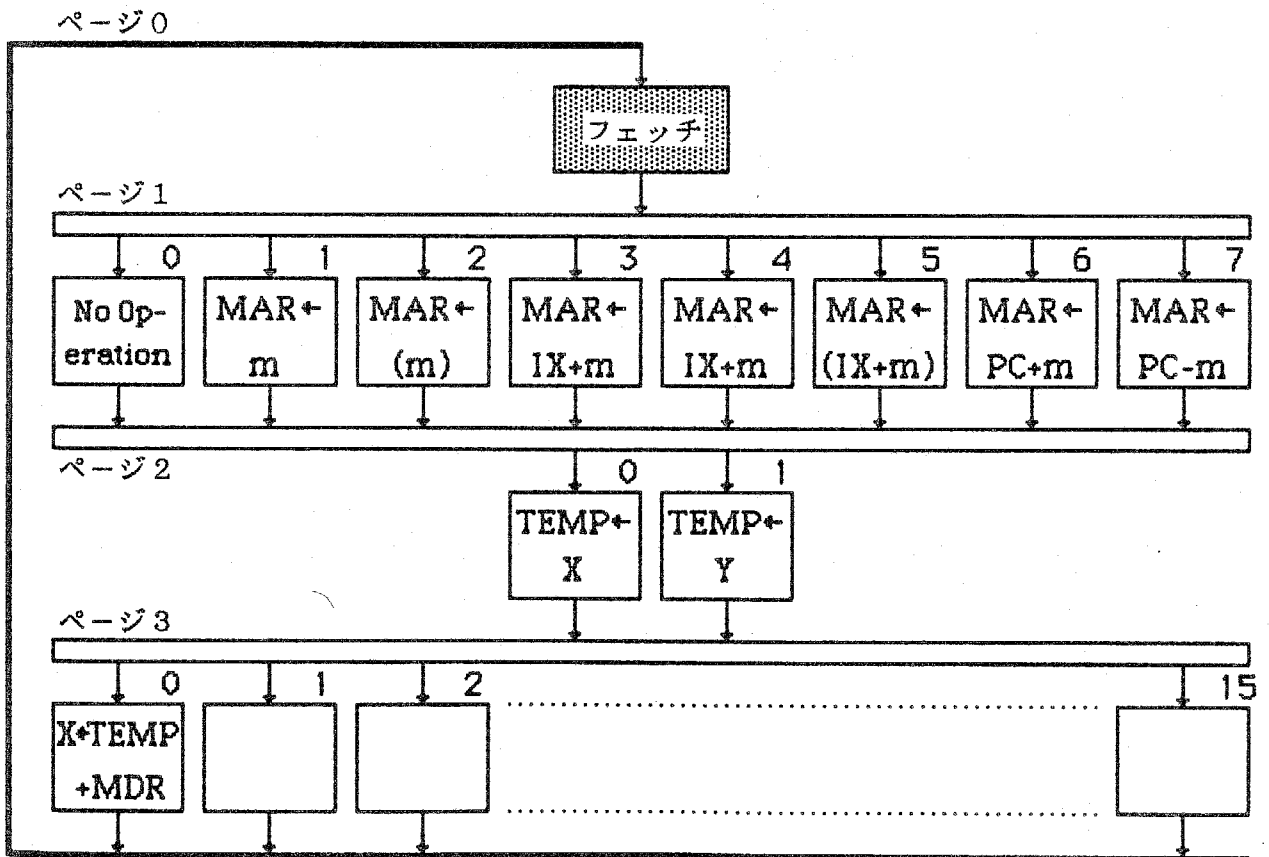


図 3.11 マイクロ・シーケンス・フロー

(c) レジスタ読み出しサイクル

第1命令語中のRフィールドでは、X、Yいずれかのレジスタを指定する。従って、 μ ROMのページ2ではビット4の1ビットだけをデコードする。そのマイクロ命令は次のとおりであり、次に続くOPフィールドのデコードのためにページ3を指定する。

(v) $TEMP \leftarrow X,$

PAGE \leftarrow 3 (レジスタXの読み出し)

(d) 命令実行サイクル

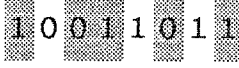
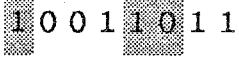
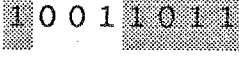
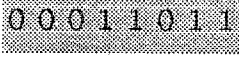
レジスタの読み出しが終るとオペランドは全てそろそろ。最後に、OPフィールドで指定する演算を行い、命令処理は完了する。以後、再び命令フェッチサイクル(ページ0)へ分岐し、命令処理が繰り返される。加算の場合のマイクロ命令を示す。

(vi) $X \leftarrow TEMP + MDR,$

PAGE \leftarrow 0 (加算実行)

図3.4の命令形式は理想的にフィールドが区分されているため、命令デコードが非常に簡素であった。しかし、現実の命令形式には複雑な構成のものも少なくない。表3.1に、実際に筆者らが開発したCMOS 8ビットマイクロコンピュータHD6301⁹⁾におけるマイクロプログラミングと命令のデコード方法の一例を示す。これからも分かるように、命令のデコード位置は必ずしも一定のフィールド(図3.4におけるOP、R、MODの各フィールド)である必要はなく、命令機能が複雑に組み込まれたものであってもよい。命令処理全体から各ページでの命令デコードの種類を取り決めた上で、ページ内での命令間のデコードで競合が起らないようにしさえすればよい。表3.1の例でも明らかなように、1つの命令中で同一のビットが何回もデコードに用いられている。この理由は新しいマイクロプログラム制御方式のピークルとなったHD6301での命令コードが巧妙に圧縮されて作られているからである。しかし、どんなに巧妙な命令コードであっても、命令コードはある一定の規則の基づいて作られている。従って、同一の機能、例えばアドレッシング・モード等は、命令の違いはあっても、また命令コード中のビット位置が疎らであっても同一のパターンをしていることが多い。このことは、本方式を用いた場合にも従来の命令デコーダ方式と同じように、同一の機能を同一のエントリ・マイクロ命令へ分岐させることができることを意味する。提案した新しいマイクロプログラム方式の場合でも、従来方式と全く同じ方法でプログラミングできる。

表3.1 HD6301のマイクロプログラミング

マイクロ サイクル	マイクロ命令	ページ	命令コード 及び デコード・ビット
i	1 PC → MAR, PC + 1 → PC	00	
	2 M(R)		
	3 PC → MAR, PC + 1 → PC DB → IR (Load ADD命令)		
	4 M(R)		
ii	5 DB → MAR	01	
	6 M(R)	00	
iii	7 PC → MAR, PC + 1 → PC DB → DBR	10	
iv	8 M(R), ACCA + DBR → ACCA	11	
v	9 第3番目のサイクルに同じ (Load AEA命令)	00	
	10 第4番目のサイクルに同じ		
vi	11 PC → MAR, PC + 1 → PC	01	
	12 M(R), ACCA + ACCB → ACCA		

PC = Program Counter

MAR = Memory Address Register

M(R) = Memory Read


DB = Data Bus

IR = Instruction Register

DBR = Data Buffer Register

ACCA = Accumulator A

ACCB = Accumulator B

 : デコード部

3.3.3 性能及びサイズの評価

命令解説機能一体形の μ ROM方式をとり入れたHD6301における命令実行性能と μ ROMサイズ(容量)を従来のマイクロプログラム方式のマイクロプロセッサのそれらと比較評価する。

(1) 性能

図3.12はGibson Mixにおける浮動小数点命令を除いた各種命令の実行時間と平均命令実行時間を2つの方式について比較したものである。HD6301では、命令語を読み出してから対応するマイクロプログラムの第1語目(エントリ・マイクロ命令)を読み出すまでの時間が従来方式のものと比較して1マイクロサイクル短縮される。各命令に適当な頻度(疑似Gibson Mix)を仮定して平均命令実行時間を算出すると、2MHzのマイクロサイクルの場合、従来方式では3.38 μ s、新方式では2.66 μ sとなり、21.3%の高速化が得られる。8ビット程度のマイクロプロセッサでは命令が比較的単純なものが多く、平均のマイクロサイクル数が小さいため、提案する新しいマイクロプログラム制御方式の「命令からマイクロ命令実行までの即応性」が性能向上の決め手と成りうるのである。

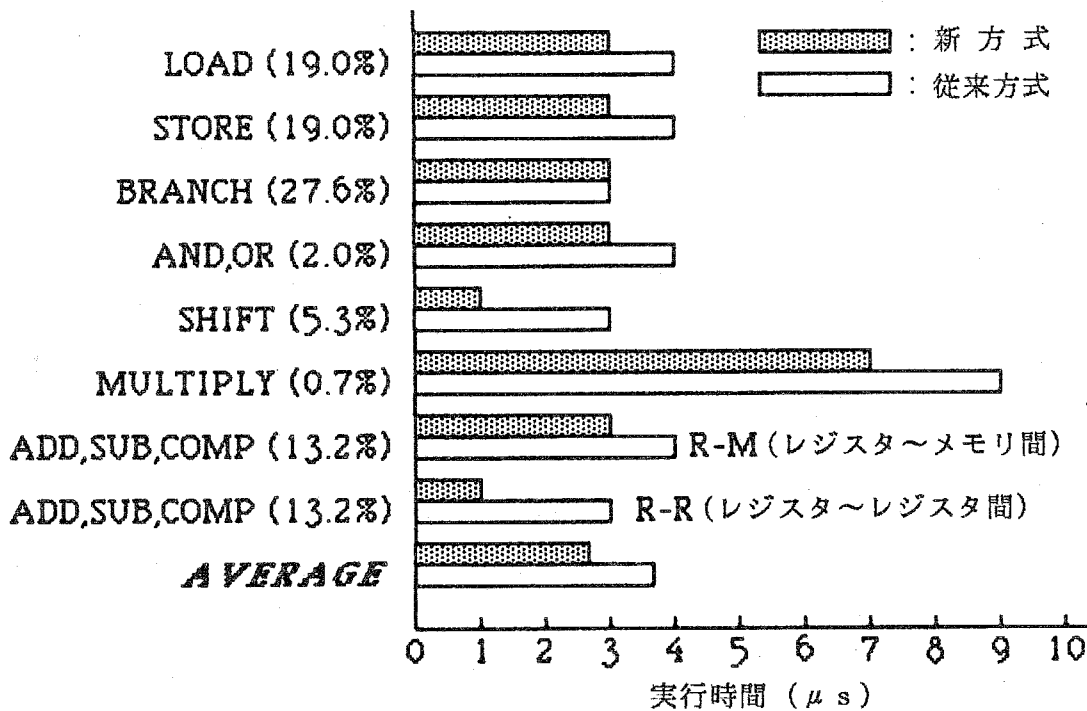


図3.12 疑似ギブソン・ミックス評価

(2) サイズ

32ビット長のマイクロ命令を用いてHD6301の全命令をマイクロプログラミングした結果、表3.2に示すように、従来方式では271語、新方式では320語となった。また、これらのマイクロプログラミングにおけるエントリ・マイクロ命令(各マイクロプログラムの先頭)は155語となるため、従来方式における命令デコーダは8入力(命令コード)、155ターム、9出力(エントリ・アドレス)のPLAが必要である。両方式を総合的に評価するため、ROMあるいはPLAの配線(ワードまたはターム線とビット線)格子点の数を算出すると、従来方式が17,425、新方式が16,640となった。即ち、マイクロプログラム制御部のサイズでは新方式が従来方式に比較して4.5%の縮小を果している。更に、従来方式では命令デコーダの制御論理や配線領域が必要である。従って、これらを考慮すると、新方式は面積で評価すると約15%の縮小を達成している。

表3.2 マイクロプログラム制御部のサイズ評価

項 目		従来方式	新方式
マイクロ命令	語 長	32ビット	32ビット
	語 数	271語	320語
μROM	コード部	8,672ビット	10,240ビット
	アドレス部	4,878ビット	6,400ビット
命令デコーダ (PLA)		入 力: 8 ターム: 155 出 力: 9 トータル: 3,875ビット	
構 成			
総ビット数		17,725 (100%)	16,640 (95.5%)
面 積		1	0.85

3.4 マイクロプログラム圧縮法

マイクロコード化マイクロプロセッサでは、命令処理の多くがマイクロプログラムメモリに集約されるため、その容量は命令機能の向上に伴って増大する。従って、マイクロプログラムメモリでのマイクロ命令コード効率が向上すれば半導体チップの面積減少に貢献する。本節では、マイクロプログラムを圧縮してマイクロプログラム制御部の面積を縮小する2つの方法を提案する^{9) 10)}。いずれの場合でも、半導体チップ内に作り込まれるマイクロプログラムROMを前提として考えたもので、LSIにおける回路設計の自由度を積極的に活用したものである。

3.4.1 2レベルマイクロプログラム方式

(1) マイクロ命令の分析

2レベルマイクロプログラム方式は、一般的にはナノプログラム方式を指す場合が多いが、ここで提案する方式は長短2語長を持つマイクロ命令によるものである。筆者らが開発したCMOS 8ビットマイクロコンピュータHD6301でのマイクロ命令は既に述べたように、32ビット/語で、総語数は320語となった。ここで、各種のマイクロ命令での仕様頻度を調べてみると、上位3種のマイクロ命令は命令フェッチに関する処理であり、104語を占めた。これらには、①命令のパイプライン処理を行なうためマイクロプログラムの途中に挿入されたもの、②2バイト目の命令フェッチを行なうもの等があり、次マイクロ命令のアドレスを指定するNA(Next Address)フィールドの8ビット、PG(Page)フィールドの2ビット、B(Branch type)フィールドの1ビットの計11ビットの内容が異なるだけで、OPE(Operation)フィールドの21ビットは全く同一のパターンとなっている。これら共通のマイクロ命令にはマイクロレベルのサブルーチン化ができるものもあるが、命令コードにより分岐する先のものも多いため簡単には減少しない。これら頻度の高いマイクロ命令をより短い語長で表現することによってメモリ効率を向上しようと考え、これを次のように実現した。

(2) マイクロ命令の構成

図3.13は2レベルのマイクロプログラム制御方式での(a)長短の2語長のマイクロ命令及び(b)μROMの構成を示したものである。上記した3種のマイクロ命令のパターンを2ビットで表現し、これにB、PG、NAの各フィールド11ビットを加えた13ビット長のマイクロ命令を4つのマットのうちの1つに格納する。この13ビットの短語長

マイクロ命令は、 μ ROMからの読み出し時にパターンを示す2ビットのENC(Encode)フィールドをデコード回路によって解読し、21ビットのOPEフィールドを生成する。デコード回路は3種類の21ビットのパターンをENCフィールドのパターンに基づいて選択・出力するだけの簡単なものである。生成後は32ビットの長語長マイクロ命令と同一に機能する。

2レベルマイクロプログラム方式は、頻度の高いマイクロ命令中の同一パターンのフィールドをエンコードして短語長化するものである。従って、多くの種類のマイクロ命令を短語長化しようとするとき、

(a)エンコード後のビット数(ENCフィールド)が増加すること

(b)エンコード部(ENCフィールド)のデコーダ回路規模が増大すること

の問題が生じる。即ち、長短2語長のマイクロ命令で構成する2レベルマイクロプログラム方式の極限はナノプログラム方式に近づく。マイクロとナノの2階層構成のために、命令フェッチからナノ命令実行までの時間が増加するため性能が低下する。この性能低下をパイプラインを深くして解消することもできるが、チップ面積縮小の目的からは不合理である。短語長化するマイクロ命令の種類を適当に抑えることが必要である。

以上のように μ ROMのメモリ効率向上を図った結果、 μ ROMの面積は元のものと比較して約14%減少することができた。

(3) μ ROM構成

長短2語のマイクロ命令は、図のように μ ROMをレイアウト上4つのマットに分割してこれに配置する。4つのマットのそれぞれの意味は次のとおりである。

(a) SM(Short Microcodes)領域

13ビットの短語長のマイクロ命令を配置する領域。語数は80語である。104語の3種のマイクロ命令は80語に限定してここに配置した。

(b) LM(Long Microcodes)領域

32ビットの長語長のマイクロ命令を配置する領域。語数は前記したSM領域と同様に80語である。

(c) LM(Even, Odd)領域

奇数と偶数のアドレスの32ビットマイクロ命令ペアを並列に配置する領域。マイクロプログラム・レベルの条件分岐の際に並列に読み出すことができる。

【 長語長マイクロ命令(32ビット) 】

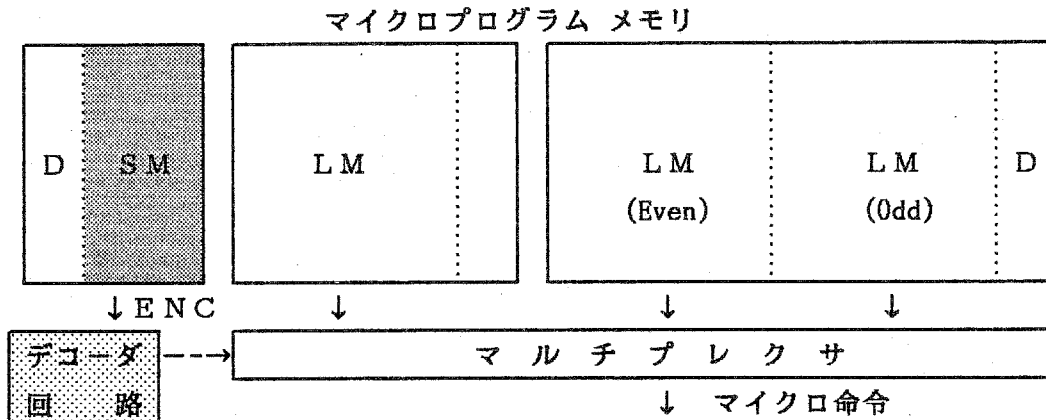


圧縮

【 短語長マイクロ命令(13ビット) 】



(a) マイクロ命令の構成



D : Address Decoder B : Branch type
 SM : Short Microcodes PG : Page
 LM : Long Microcodes

(b) μ ROMの構成

図3.13 2レベルマイクロプログラム方式

3.4.2 マイクロ命令合成方式

前記した2レベルマイクロプログラム方式は、図3.13に示したように μ ROMの構成に制約が生じる。そこで、ここでは μ ROMの構成(形状)を変えずにマイクロプログラム語数だけを減少する別の圧縮方式も提案する。

本方式の基本的な考え方は、1つのマイクロコード(Physical Microcode)を複数の論理的あるいは仮想的マイクロコード(Logical Microcodes or Virtual Microcodes)の合成によって表現しようとするものである。このため、本方式を仮想マイクロプログラミング技法(Virtual Microprogramming Technique)と呼ぶ。即ち、1つのアドレスによって μ ROM中の複数の論理的マイクロコードをアクセスし、これらを μ ROMのビット線上で“OR”をとることで所望の物理的マイクロコードを得るものである。図3.14はその概念を示したものである。

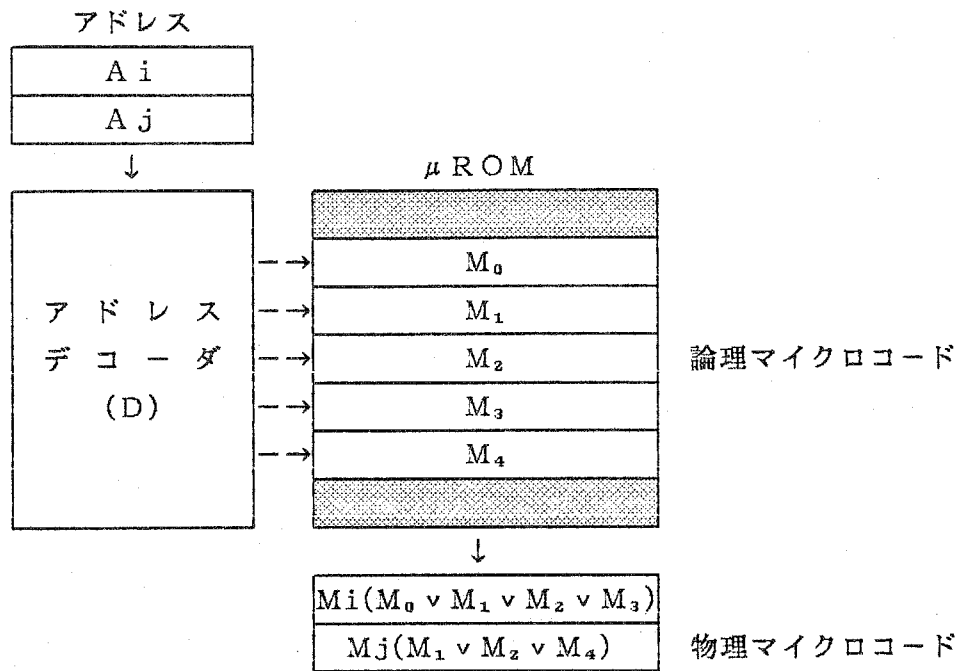


図 3.14 仮想マイクロプログラミング技法の概念

この図の例では、μROMのアドレスデコーダ(D)に入力する1つのアドレス(A_i)に対し、4つのマイクロコード(M₀, M₁, M₂, M₃)がアクセスされ、他のアドレス(A_j)に対しては3つのマイクロコード(M₁, M₂, M₄)がアクセスされる。従って、それぞれのアドレスに対する所望のマイクロコード(M_i), (M_j)は次の式で表わされる。

$$M_i = (M_0) \vee (M_1) \vee (M_2) \vee (M_3) \quad (1)$$

$$M_j = (M_1) \vee (M_2) \vee (M_4) \quad (2)$$

このような関係が、μROM中のマイクロ命令間で、簡単にしかもあらゆるところで成り立つことは難しい。そこで、マイクロプログラミングにおけるアドレス割り付けの際、このような関係を持ち易いようにする必要がある。図3.15はその手順を示したものである。

(1) マイクロ命令のグループ化

マイクロプログラミング後、マイクロ命令中の次のマイクロ命令を指定するNAフィールドを除いてコード化し、この中から類似した(上記(1),(2)式を作成し易い)コードを持つマイクロ命令群を集め、いくつかのグループに編成する。図3.15(a)は共通コードA, Bを持つそれぞれ4つのマイクロ命令群をグループ化したものである。それぞれのグ

ループは、共通でない部分のコードが (a_0, a_1, a_2, a_3) と (b_0, b_1, b_2, b_3) となっている。

(2) マイクロ命令アドレス割り付け

グループ化されたそれぞれのマイクロ命令群に対して、図3.15(b)に示すような連続したアドレスを割り付ける。連続したアドレスを割り付ける理由は、共通コードとなっているA、Bをどのアドレスからもアクセスできるようにするためである。共通コードAを持つグループでは、アドレス(C000~C011)、Bを持つグループではアドレス(C100~C111)が割り付けられる。

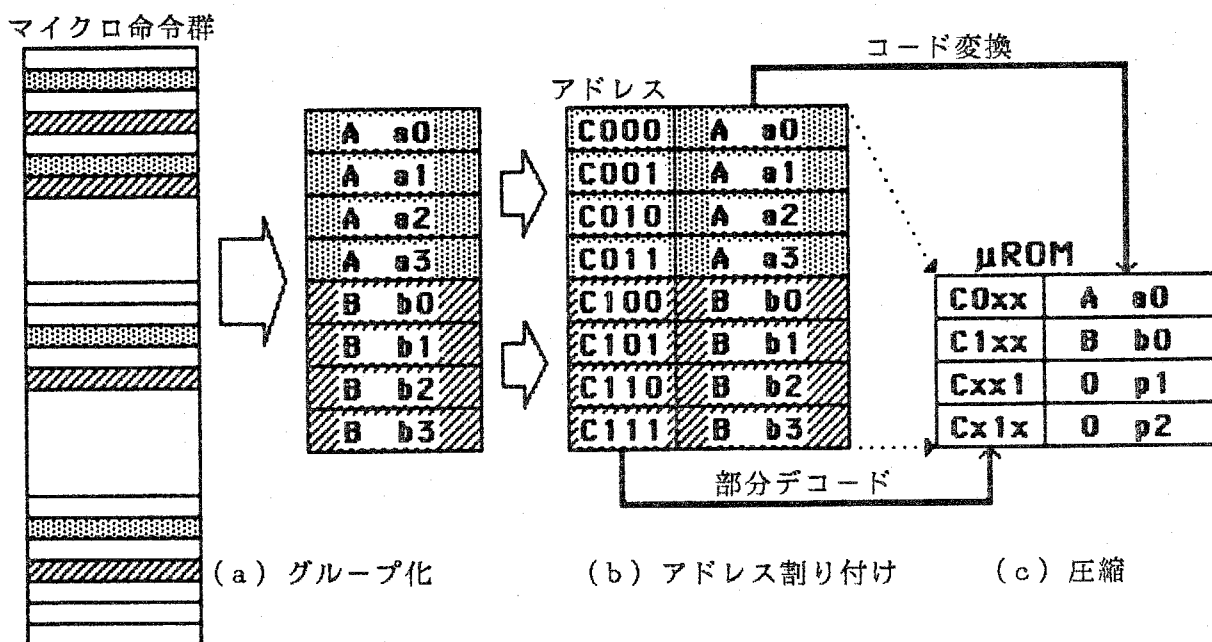


図3.15 仮想マイクロプログラミングの手順

(3) 物理マイクロ命令の論理マイクロ命令への分解

2つのグループにおける計8つの物理マイクロ命令は、図3.15(c)に示すように、次式(3)~(8)の関係を保って4つの論理マイクロ命令に分解する。これらの式が成立し易い状況は、前項におけるマイクロ命令のアドレス・フィールド以外のフィールドが共通のもの、ある機能のマイクロ命令(共通パターン)に他の簡単な機能が付加されたもの等、いくつかの場合が考えられる。これらの色々の場合をマイクロプログラミングした全体に渡って調べることにより本方式を適用することができる。具体的な例としては、レジスタ部が同一で演算モードが加算或いは減算といったものが挙げられる。

$$a_1 = a_0 \vee p_1 \quad (3)$$

$$a_2 = a_0 \vee p_2 \quad (4)$$

$$a_3 = a_0 \vee p_1 \vee p_2 \quad (5)$$

$$b_1 = b_0 \vee p_1 \quad (6)$$

$$b_2 = b_0 \vee p_2 \quad (7)$$

$$b_3 = b_0 \vee p_1 \vee p_2 \quad (8)$$

この式からも明らかなように、共通コードA、B以外のコード中からも共通的なコード a_0 、 b_0 が抽出されている。これは、原理説明の簡単化のためであり、必ずしもその必要はない。(3)~(8)式中の p_1 、 p_2 は論理マイクロ命令から物理マイクロ命令を合成して生成するために必要なコードである。

一方、アドレスデコーダ(D)では、1つのアドレスに対して(3)~(8)式の関係を作り立たせる論理マイクロ命令をアクセスできるように、言い換えれば、1つの論理マイクロ命令を複数のアドレスでアクセスできるように冗長性を持たせ、部分的なデコードができるようにしている。図3.15(c)のような μ ROM構成において、例えば、物理マイクロ命令(A、 a_3)を得るために、アドレス(C011)が入力した場合、アドレスデコーダ(D)では(C0xx)、(Cxx1)、(Cx1x)の3つの論理マイクロ命令を選択することになる。この結果、共通コード部は(A \vee 0)、それ以外のコード部では($a_0 \vee p_1 \vee p_2$)の内容が出力され、(5)式から(A、 a_3)のコードとなることが分かる。ここで、“0”は全て0のコード(Null code)である。

以上の手順に従って論理マイクロ命令を構成することで、マイクロ命令の語数減少が図れる。本方式によれば、同一のマイクロコードを持つマイクロ命令が 2^n 語の場合、($n+1$)語に、またこの組み合わせ(グループ)が m 種類ある場合には、($2^n \times m$ 語)が($n+m$)語に縮退される。図3.15の場合、 $n=2$ 、 $m=2$ であり、8語が半分の4語に縮退された例である。

以上の方式は、筆者らが開発したシステムオンチップ形のCMOS8ビットマイクロコンピュータHD64180に適用された。マイクロプログラム圧縮手順をマニュアルで実施した結果、約13%のコード圧縮が図られた。この試行では容易に本方式を導入できる部分についてのみ実施されたため、この程度の効果に終わった。従って、本方式をCAD化することによってより高いコード圧縮が可能である。また、PLAの最適化手法も色々と

行われているが、この方法はPLAのコードを機械的に冗長性を排除してコード圧縮を行うものである。本方式のように、元々のコード(アドレス・コード)まで立ち入るものではないが、両方式の併用は効果的である。

図3.16にHD64180で実際に本方式を適用した場合のマイクロコード列を示す。これから分かるように、元のマイクロプログラムのコード中には共通パターンが多く存在しており、16語の物理的マイクロ命令が6語の論理的マイクロ命令に縮退されている。このようなROM圧縮を数箇所を実施し、上記のような効果が得られた。

		物理マイクロ命令									
		Address	N	A	M	m	c	Register	ALU	cc	RT
P0	:	10111010	10100001	000000	0000100	1	01111	11000	111010		
P1	:	10111011	10100001	00	000	0000100	1	01111	11000	111010	
P2	:	10110010	10100001	00	000	0000100	1	00101	11000	111010	
P3	:	10110011	10100001	00	000	0000100	1	00101	11000	111010	
P4	:	10111001	10100001	01	000	0000100	1	01111	10010	111010	
P5	:	10110001	10100001	01	000	0000100	1	00101	10010	111010	
P6	:	10111000	10100001	00	000	0000100	1	01111	10110	111010	
P7	:	10110000	10100001	00	000	0000100	1	00101	10110	111010	
P8	:	10101001	10100001	01	000	0000100	1	01111	00000	111010	
P9	:	10100001	10100001	01	000	0000100	1	00101	00000	111010	
P10	:	10100000	10100001	00	000	0000100	1	00101	00000	111010	
P11	:	10100010	10100001	00	000	0000100	1	00101	00000	111010	
P12	:	10100011	10100001	00	000	0000100	1	00101	00000	111010	
P13	:	10101000	10100001	00	000	0000100	1	01111	00000	111010	
P14	:	10101010	10100001	00	000	0000100	1	01111	00000	111010	
P15	:	10101011	10100001	00	000	0000100	1	01111	00000	111010	
論理的マイクロ命令 (x : Don't care bit)											
L0	:	1011x01x	10100001	00	000	0000100	1	00000	11000	111010	
L1	:	101x00xx	10100001	00	000	0000100	1	00101	00000	111010	
L2	:	101x10xx	10100001	00	000	0000100	1	01111	00000	111010	
L3	:	101xx001	00000000	01	000	00000000	1	00000	00000	000000	
L4	:	1011x000	00000000	00	000	00000000	1	00000	10110	000000	
L5	:	1011x001	00000000	00	000	00000000	1	00000	10010	000000	

図3.16 ROM圧縮例

3.5 プロセッサのデータ処理部構成

3.5.1 バス指向のデータパス構成

ここでは高性能なデータ処理を得るためのデータパス構成を8ビットマイクロプロセッサ(HD6301のカーネル・プロセッサ)を例に述べる⁹⁾。

図3.17はデータパス構成を示したものであるが、各要素を規則的に配列した「バス指向」の構造を持っている。この構成は正にメモリLSIの構成を指向した高集積向きのものである。即ち、各種データバスはメモリLSIのビット線に、各種制御信号はワード線に、レジスタやALUの1ビット分はメモリLSIの記憶セルに対応する。

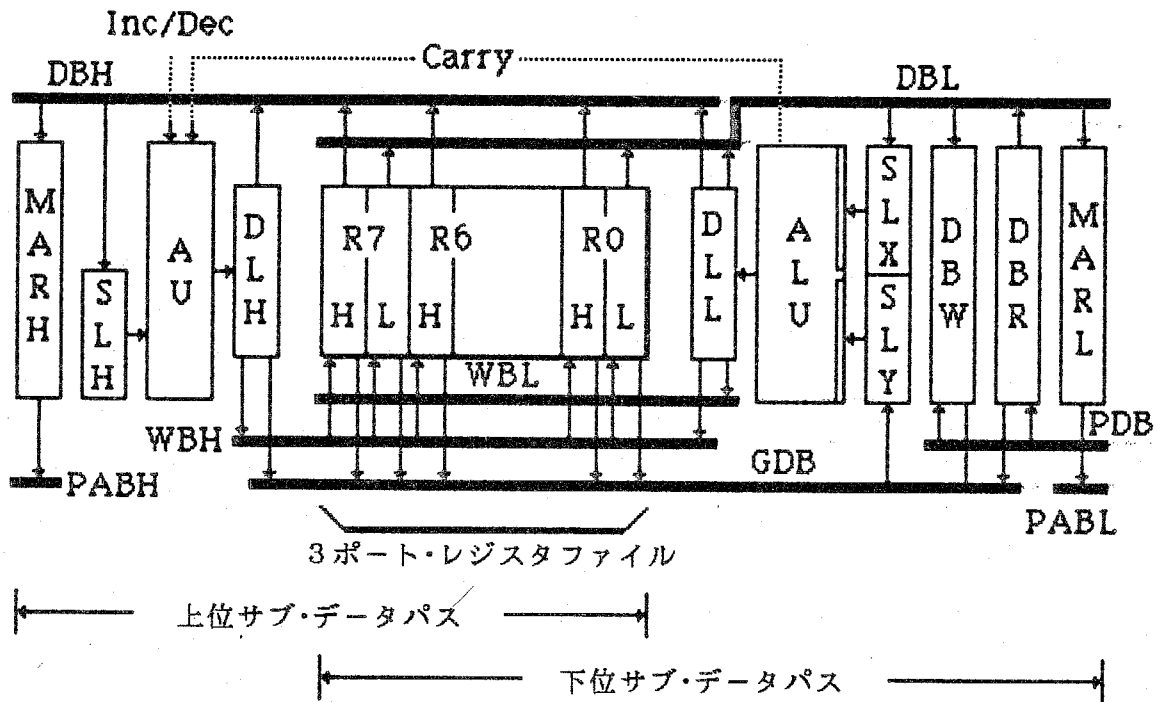


図3.17 データパス構成

以下に、本データパスの構成上の特徴を示す。

(1) デュアル8ビットのサブ・データバス

マイクロプロセッサのデータ処理では、8ビット(1バイト)のデータ処理を基本としながらもワード(2バイト)処理も存在する。これらはメモリアドレス計算やワード・レジスタのインクリメント、デクリメント等である。従って、バイト、ワード処理を効率的に行えるように、下位バイトと上位バイトに分離したデュアル・データパス構成がよい。

(a) バイト処理:

下位、上位バイトに属するレジスタ群は全て下位データパスのALUで処理。

(b) ワード処理：

上位、下位バイトのデータパスはそれぞれ独立に動作し、下位のALUから生じる桁上げ(キャリ)だけが上位に伝搬される。

(2) 多重内部データバス

図3.17のデータパスでは5種の内部データバスを持つ。これらは全て8ビット幅であり、DBH(read Data Bus High)、DBL(read Data Bus Low)、WBH(Write data Bus High)、WBL(Write data Bus Low)、そしてGDB(General Data Bus)がある。これらのバスは次のように用いられる。

(a) バイト処理：

2つのバイト・オペランドを演算する場合、第1のオペランドを格納するレジスタの内容がGDBに、第2のオペランドを格納するレジスタが上位あるいは下位のサブ・データバスのいずれに属するかにより、その内容をDBHあるいはDBLに載せる。これら2つのオペランドはそれぞれのバスを経由して、ソースラッチSLX及びSLYに同時に格納される。2つのオペランドはALUにより演算され、その結果をディスティネーションラッチDLLに格納する。DLLに格納されたデータはWBLあるいはWBHを経由して再びレジスタの1つに書き込まれる。

(b) ワード処理：

プログラムカウンタ(PC)はワード・レジスタであり、このインクリメントの場合には4種のバスが用いられる。プログラムカウンタの上位と下位バイトのデータは、それぞれDBH及びDBLを経由してソースラッチSLH、SLXに格納される。それぞれのデータはAU(Adder Unit)及びALUにより独立にインクリメントが行われる。その結果はそれぞれディスティネーションラッチDLH及びDLLに格納される。DLLの内容は、WBLに、DLHの内容は下位バイトからの桁上げ(ALUからのキャリ)が発生した場合に限ってWBHに載せられる。下位バイトからの桁上げが発生しなかった場合、SLHの内容がDLHにかわってWBHに載せられ、プログラムカウンタの内容が更新される。

(3) 3ポート・レジスタファイル

内部データバスの構成上、3ポート(2-read, 1-write)のレジスタファイルが必要

である。それぞれ8個のバイト・レジスタが上位と下位のサブ・データパスに独立に配置され、これらはマイクロ命令により指定される。従って、レジスタの割り当てはほぼ任意となり、異なるアーキテクチャのマイクロプロセッサに適合し易い。これによって、データパスの融通性が高まり、拡張性もでてくる。VLSI設計にとって非常に有効な構造となり得る。3ポート・レジスタファイルの必要性は「マイクロレベル」のパイプライン制御に不可欠なものであり、この詳細については後述する。

一方、3ポート・レジスタファイルは大きなチップ領域を必要とするように見える。しかし、ビット方向の大きさはむしろALUやAUによって決められるので、これに合わせてレジスタを構成するメモリセルのレイアウト設計を注意深く行うことによって小さな面積でこれを実現可能である。高速性に対する利点の方が大きい。

3.5.2 マイクロレベルのパイプライン制御

通常、高性能化のためマイクロプロセッサでも命令の先取り等の簡単なマクロレベルのパイプライン制御を行うことが多い。このためには、命令キュー等のハードウェアを必要とする場合があるが、マイクロプログラミングによってもパイプライン制御可能である。図3.18はその一例を示したもので、ABA命令（レジスタA, B間の加算命令）において最終マイクロ命令を命令フェッチから2マイクロサイクル後に配置することで次の命令フェッチを開始できる。

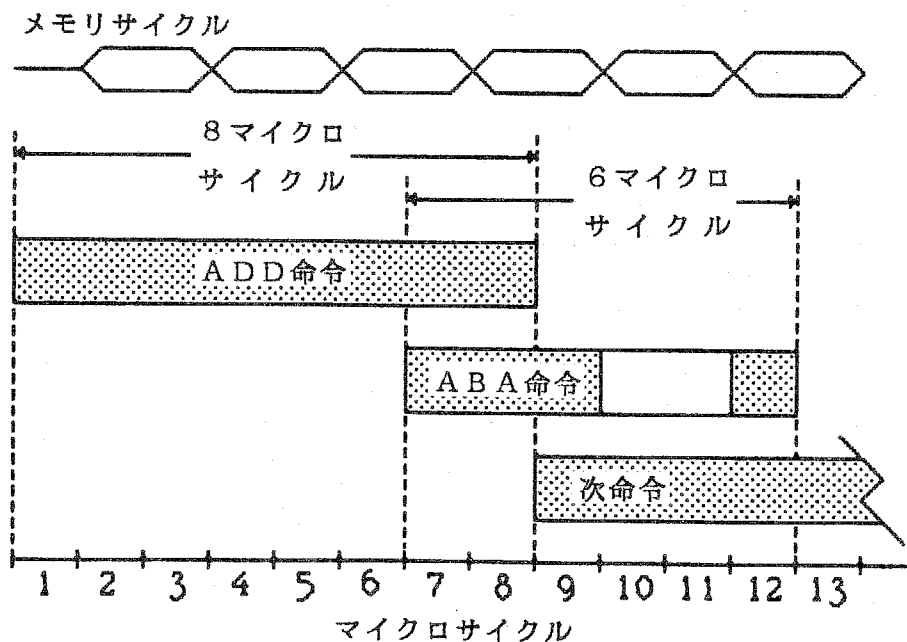


図3.18 マクロパイプライン制御

この結果4マイクロサイクル必要なABA命令を実際には6マイクロサイクルかけて、等価的に2マイクロサイクルに短縮している。

以上の処理においてもう1つの高速化手段は1マイクロサイクル時間の短縮がある。その具体的手段としてマイクロレベルのパイプライン制御を提案する。

通常のマикроプログラム制御では、図3.19(a)に示すように、マイクロ命令の実行と次のマイクロ命令の読み出しとを並列に行うだけのものが多い(パイプライン1段)。

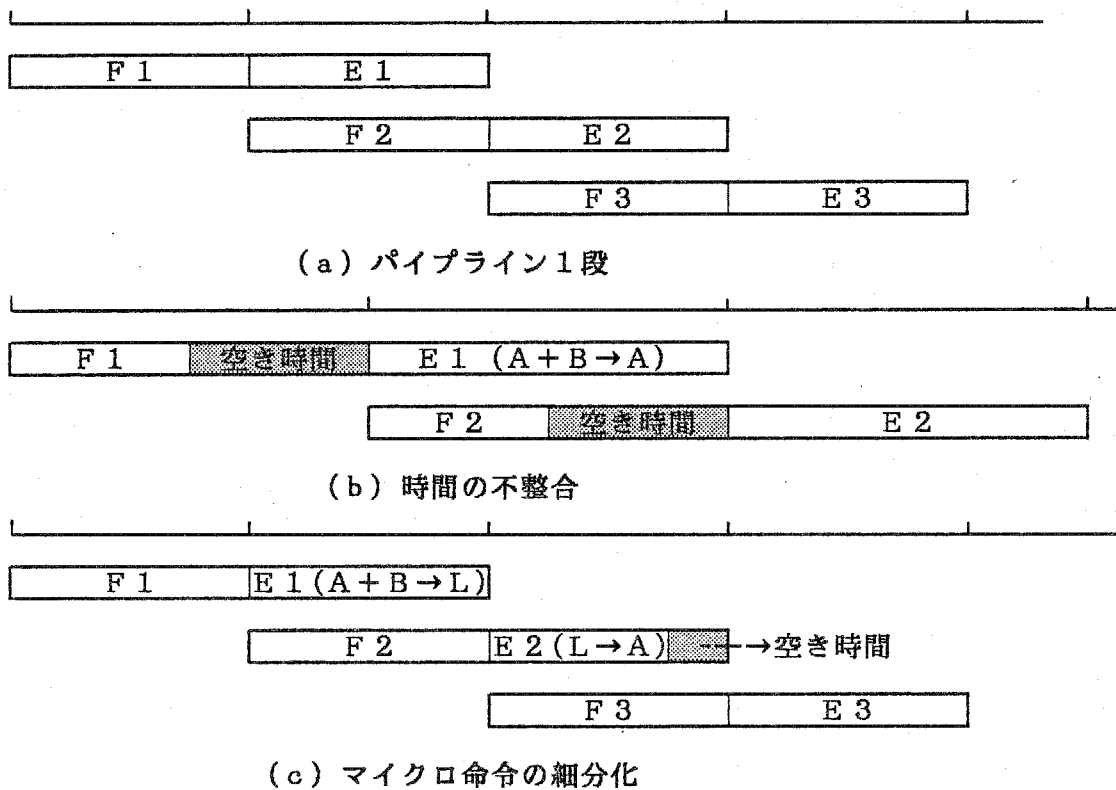


図3.19 マイクロパイプラインにおける問題点

この場合、次のような問題が発生する。

(1) マイクロ命令の読み出し時間と実行時間の不整合

レジスタ群の中の2語の読み出し、これらを演算して再びレジスタの1つに書き込むレジスタ間演算では、1マイクロ命令の実行時間が長くなる。これに対し、マイクロ命令の読み出し時間(μ ROMのアクセス時間)は語数、ビット長によって異なるが、実行時間の約1/2である(3 μ mCMOSの場合)。従って、マイクロサイクルはマイクロ命令実行

に要する時間から一義的に決定される。この場合、図3.19(b)に示したように最小のMOSサイズでは2MHz(マイクロサイクル時間500ns)程度となってしまふ。

(2) マイクロ命令処理の細分化によるマイクロプログラム語数の増加

レジスタ間演算のように論理の深い(論理段数の多い)処理はマイクロ命令処理を細分化し、その実行時間をμROMサイクルに合せれば(1)の問題は解決する。この場合、1つのマイクロ命令をレジスタの読み出しと演算・書き込み(あるいはレジスタの読み出し・演算と書き込み)の2つのマイクロ命令処理に分ける必要がある。これによって、マイクロプログラム語数が約2倍に増加すると共に、2つのマイクロ命令処理を均等な時間に分割する事はできない。図3.19(c)に示すように、マイクロサイクルはいずれかの実行時間の長い方に合わせる事になり、効率が悪い。

以上に述べた従来方式の欠点を解決するため、次の方法を提案する。(1)の問題に対してはレジスタ間演算を2つのマイクロサイクルに分ける事、これによって生ずる(2)の問題に対しては2つに分けたマイクロサイクルを1つのマイクロ命令で実現する。以下にその詳細を述べる。

(a) 3段のマイクロパイプライン制御

図3.20に示すように、マイクロ命令読み出し、マイクロ命令実行(レジスタの読み出し及び演算の一部)、マイクロ命令実行(演算の残り及びレジスタへの書き込み)をそれぞれ1マイクロサイクルで処理し、1マイクロサイクル分ずらす3段のパイプライン制御を行う。

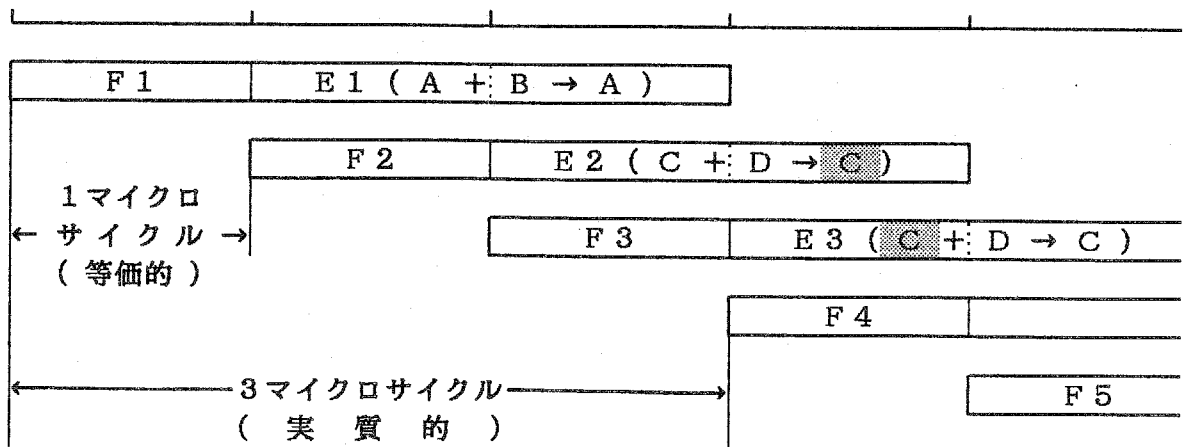


図3.20 3段のマイクロパイプライン制御

これによって等価的に1マイクロサイクルの実行とする。この場合1つのマイクロ命令実行におけるレジスタへの書き込みと次のマイクロ命令実行におけるレジスタの読み出しが並列に実行される。従って、レジスタの書き込み系バスと読み出し系バスは独立していなければならない。図3.17のデータパス構成はこの点を考慮して設計されているものである。ところが、同一のレジスタに対して書き込みと読み出しが同マイクロサイクルで発生すると競合の問題が生じる。即ち、マイクロ命令実行であるレジスタに書き込みが行われ、その内容が十分に確定しないうちに次のマイクロ命令実行でそのレジスタを読み出してしまふ問題である。このような競合が行っても問題なくするためにはレジスタへの書き込みと読み出しとの間に十分なマージンをとることが必要である。そのような方法ではマイクロ命令実行時間が増大し、パイプライン制御を行った意味が無くなる。

(b) レジスタ競合の回避

読み出しと書き込みレジスタのアドレスを常にみる一致検出回路を設け、アドレスの一致か否かによって図3.21に示すように2種類のデータのパスを作る。

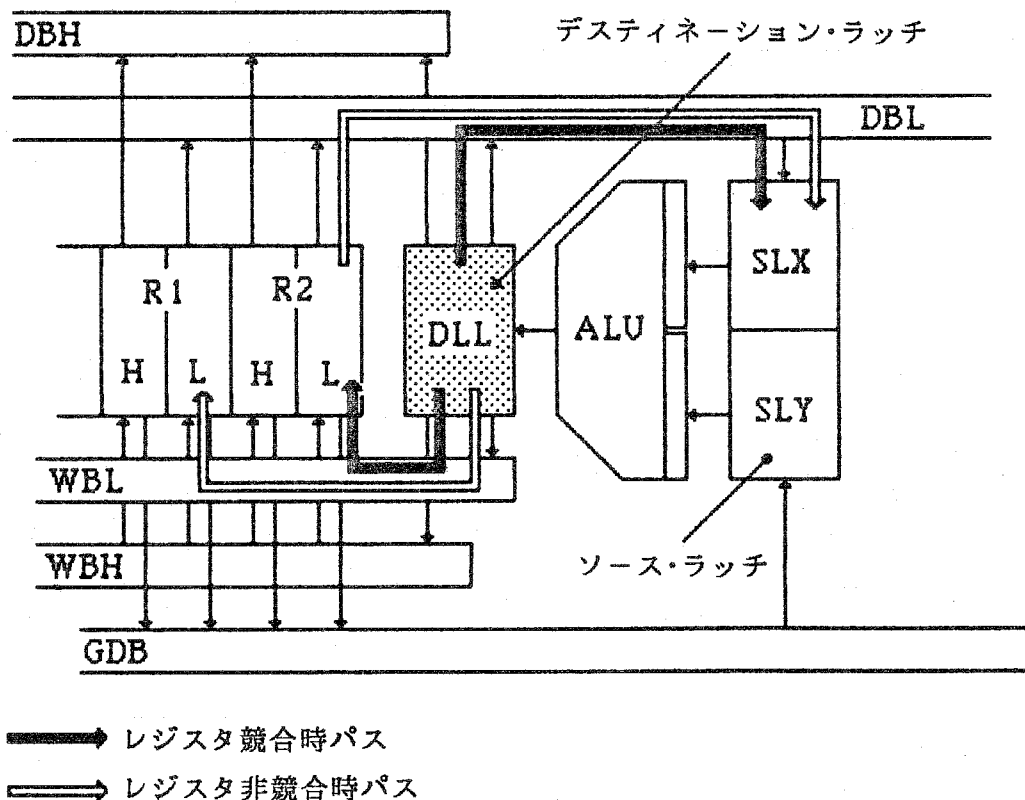


図3.21 レジスタ競合の回避

(i) 競合のない場合：ALUにおける演算結果を格納した出力ラッチDLLの内容をバスWBLに載せ、これを介してレジスタR2へ書き込む。一方、レジスタR1の内容はバスDBLを介してALUの入力ラッチSLXに格納される。

(ii) 競合のある場合：出力ラッチDLLの内容はバスWBLを介してレジスタR2に書き込まれる。一方、レジスタR2の読み出し(バスDBLへの送出)は禁止され、代りに出力ラッチDLLの内容をバスDBLに載せ、これを介して入力ラッチSLXに格納する。

以上の方法により、マイクロプログラミング時にレジスタ競合を意識する必要はない。

(c) 効果

1マイクロ命令の実行を2つのマイクロサイクルに分けたことにより、マイクロサイクル時間は従来方式の約1/2になる。この結果、図3.12に示した平均命令実行時間を1.33 μ sにすることができる。この数値は全く従来のプロセッサ構成法で実現した場合の3.38 μ sに対し、2.54倍の性能向上を示している。新しい命令解読機能一体形のマイクロプログラム方式とマイクロレベルのパイプライン制御により、これだけの性能向上が図られることは、プロセッサ構成(マイクロチップ・アーキテクチャ)の意義がいかに大きいかを示している。

3.6 LSI構成と機能

(1) LSI構成

システムオンチップ形CMOS8ビットマイクロコンピュータHD6301⁹⁾のチップ全体及びプロセッサ部のブロック図を図3.22に、チップ写真を図3.23に示す。このマイクロコンピュータは3 μ mCMOSプロセスによって製造されており、LSI仕様は表3.3に示すとおりである。8MHzの入力クロック(マイクロサイクル4MHz)の場合、レジスタ間加算命令を0.5 μ sで実行し、約60mWの低消費電力を実現している。また、4KバイトROM、128バイトRAMを含め約82,000のMOSトランジスタから構成されている。

表 3.3 L S I 仕様

項 目	仕 様
プ ロ セ ス	3 μ m C M O S
集 積 度	8 2, 0 0 0 M O S ト ラ ン ジ ス タ
チ ッ プ サ イ ズ	6. 8 2 \times 7. 8 9 m m
消 費 電 力	3 0 m W (4 M H z 動 作 時)
動 作 周 波 数	最 大 8 M H z
マ イ ク ロ サ イ ク ル	4 M H z (2 5 0 n s)

(2) 機能

表 3.4 はマイクロコンピュータとしての仕様を示したものである。カーネル・プロセッサは 80 の命令と 6 種類のアドレッシング・モードを有しており、次の特徴も合せ持っている。

(a) スリープ機能

プロセッサの状態を保持したまま停止、消費電力を通常動作時の 1 / 10 まで下げる低消費電力モードであり、CMOS の低消費電力性を活かしたものである。スリープ命令の実行によってこのモードに入り、割込み信号あるいはリセット信号で解除される。

(b) 命令コードエラー割込み

未定義の命令コードを命令語として読み込んだ場合、これを検出し割込みを発生する。ソフトウェアエラーやバス上でのノイズによる信号レベルの反転等によるエラーに対して、大きな効果を発揮する。

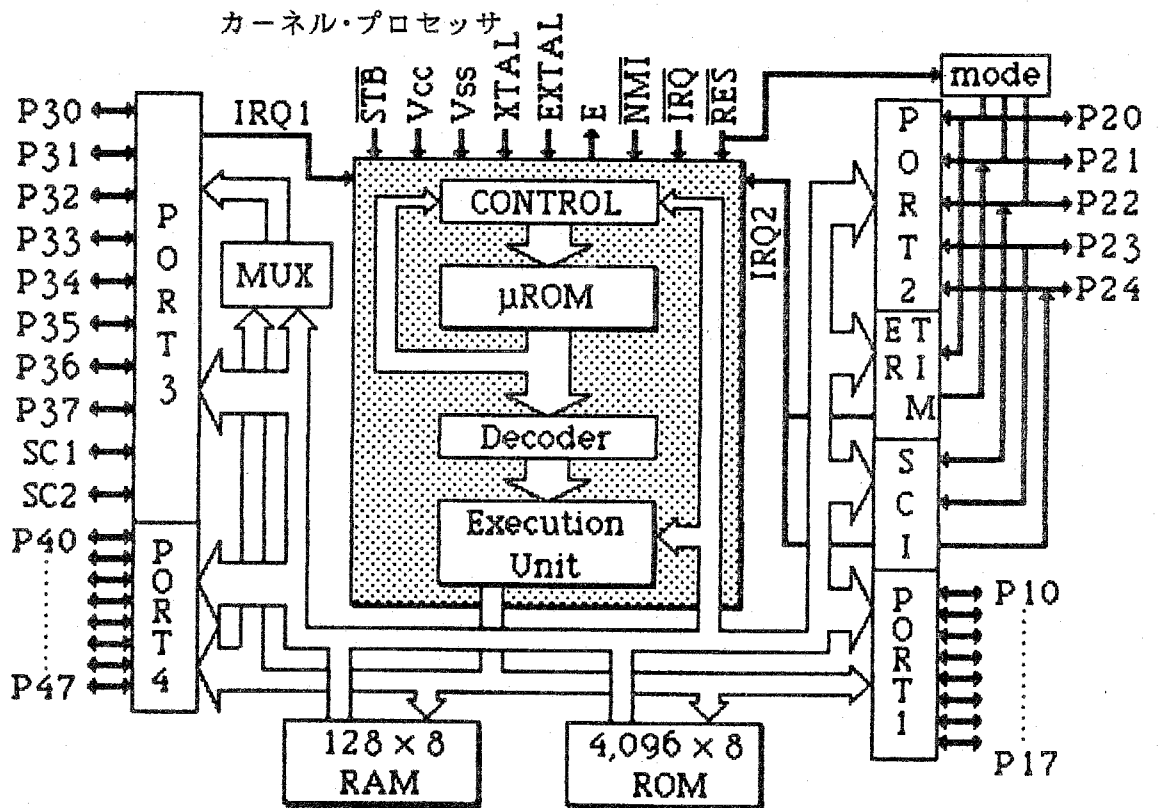
(c) アドレスエラー割込み

プログラム領域として使用できないメモリ空間(オンチップ I / O 空間等)から命令語を読み出そうとした場合、これを検出して割込みを発生する。この機能も各種のエラーに対して効果が大きい。

以上の他、表 3.4 に示すように多くのオンチップ I / O 機能を有しており、システムの完全シングルチップ化によるコスト低減を最大の目標とするアプリケーションに向けたマイクロコンピュータといえる。

表 3.4 マイクロコンピュータ仕様

項 目	内 容
命 令 数	89 (含む乗算, スリープ命令)
最小命令実行時間	0.5 μ s (8 MHz時)
	0.67 μ s (6 MHz時)
	1.0 μ s (4 MHz時)
オンチップ機能	メモリ ROM 4Kバイト RAM 128バイト
	I / O パラレル・ポート 29 シリアル・ポート 入出力各1
	タイマ 16ビット
	発振器 最大8 MHz



IRQ : Interrupt Request SCI : Serial Communication Interface
 NMI : Non-maskable IRQ RES : Reset
 EXTAL : E - Crystal XTAL : Crystal
 E : Enable STBY : Stand-by
 MUX : Multiplexer

図 3.22 CMOS 8ビットマイクロコンピュータHD6301の構成

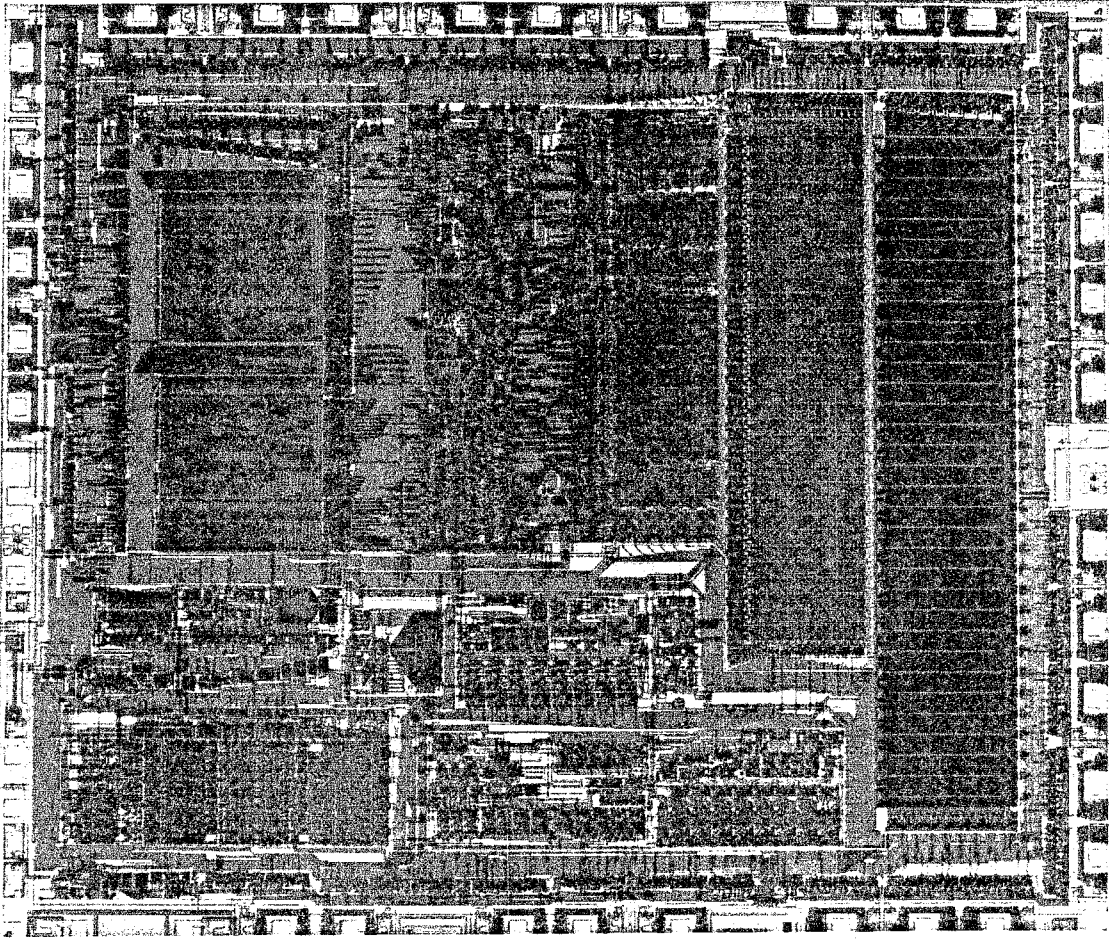


図 3.23 HD6301のチップ写真

3.7 まとめ

システムオンチップ形マイクロコンピュータにおけるカーネル・プロセッサの構成法について述べた。システムオンチップ形では、カーネル・プロセッサは多くの周辺I/O機能と同時に同一チップ上に集積されるため、コンパクトでありながら高速性を兼ね備えなければならない。従って、少ないハードウェアで高性能を得るマイクロチップ・アーキテクチャが必要である。小形化と高性能化を同時に実現する方式として、命令解読機能一体形の μ ROMを持つ新しいマイクロプログラム制御方式を示し、従来方式のものに比較して21.3%の高速化と15%の面積縮小を達成した。また、データパスにおける高速化手段として、マイクロレベルのパイプライン制御方式を示し、従来方式の2倍の動作周波数で動作し得る構成を実現した。更に、プロセッサの小形化手段として、2種類のマイクロプログラム圧縮方式を示し、13%~14%のマイクロコード圧縮を果たした。

以上で得た各プロセッサ構成法は、1チップのシリコン上に形成されるマイクロコード

化マイクロプロセッサに対して共通的に適用されるものである。全てLSIにおける回路構成の自由度(目的に合わせて回路構成を自由に選べること)を活かして実現した方式である。

3.8 第3章の参考文献

- (1) Dave Bursky: "MOS/digital/Analog mix=low-cost, high-performance single chip μ Cs", *Electronic Design*, Vol.27, No.13, pp.43~48 (1979-6)
- (2) S.P.Morse et al.: "The Intel 8086 Micro Processor: A 16-bit Evolution of the 8080", *Computer*, Vol.11, pp.18~27 (1978-5)
- (3) M.Shima: "Two Versions of 16-bit chip span microprocessor, minicomputer needs", *Electronics*, Vol.51, No.26, pp.81~88 (1978-6)
- (4) E.Stritter et al.: "A Microprocessor Architecture for a Changing World: The Motorola 68000", *Computer*, Vol.12, No.2, pp.43~52 (1979-2)
- (5) D.MacGregor et al.: "The Motorola MC68020", *IEEE Micro*, Vol.6, No.8, pp.101~108 (1984-8)
- (6) E.R.Fiala: "The Maxc System", *Computer*, Vol.11, No.5, pp.57~67 (1978-5)
- (7) 前島外: "高集積マイクロコンピュータに適したマイクロプログラム制御方式", *情報処理学会論文集*, Vol.23, No.1, pp.16~24 (昭57-1)
- (8) H.Maejima et al.: "The VLSI Control Structure of a CMOS Microcomputer", *IEEE Micro*, Vol.3, No.6, pp.9~16 (1983-12)
- (9) H.Maejima et al.: "A CMOS Microprocessor with Instruction-Controlled Register File and ROM", *ISSCC Digest of Technical Papers*, Vol.28, pp.12~13 (1985-2)
- (10) H.Maejima et al.: "CMOS 8-Bit Single-Chip Microcomputer HD6301", *Hitachi Review*, Vol.30, No.4, pp.171~176 (1981-8)

第4章 アプリケーション適応形プロセッサの 可変構造化構成法

第4章 アプリケーション適応形プロセッサ の可変構造化構成法

4.1 まえがき

マイクロプロセッサの応用分野は拡大の一途をたどっている。オフィスのエレクトロニクス化の波に乗って普及したパーソナル・コンピュータやワードプロセッサを始めとして、VTR (Video Tape Recorder)、エアコン等の家電品、自動車のエンジンやパネル制御装置、ゲーム・マシン等が挙げられる。これら各々の分野で応用されるマイクロプロセッサに対する要求は、その目的に応じて流通ソフトウェアが使えるもの、多くの周辺I/O機能を備えたもの、低消費電力のもの、高性能のもの、低価格のものなど数えあげれば切りが無い。これらの要求に応えるため、それぞれの分野に最適な命令体系を備えたマイクロプロセッサを1つ1つ開発したのでは膨大な設計パワーを必要とし、事実上不可能なことである。目的に合ったマイクロプロセッサを少ない開発パワーで短期間に仕上げる方法としては、ゲートアレイLSI¹⁾とStandard Cell方式のカスタムLSI²⁾の2つの方式があり得る。ゲートアレイLSIの場合、通常の論理ゲート(AND, NORなど)のみの組み合わせしか使えないため、集積度、性能、価格の問題で汎用マイクロプロセッサには向かない。また、Standard Cell方式のカスタムLSIの場合、マイクロセル・ライブラリを充実すれば、ある程度まで目的を達することができるかもしれない。このアプローチは後章で述べる階層論理構造化設計と類似するところがあるが、汎用化したマクロセルを組み合わせるため集積度、性能の点でこれも汎用マイクロプロセッサへの適用は難しい。専用プロセッサ³⁾が限界であろう。そこで、特に命令体系に対して、一部のマスクのみの変更で応用に適合し得るマイクロプロセッサを構成できると都合がよい。

本章ではシステムオンチップ形のマイクロコンピュータにおけるカーネル・プロセッサがアプリケーションに適応して形を変えられるように、これを可変構造化⁴⁾する方法について提案する。まず、可変構造化の概念を述べ、命令語の分析に基づいたプロセッサの構成法を示す。次に、このプロセッサ構成における命令制御方式(マイクロプログラム制御方式)とレジスタ制御方式(RAMアクセス方式)を示し、それらの具体的実現法を示す。最後に、可変構造化プロセッサを適用したシステムオンチップ形マイクロプロセッサの概要を示す。

4.2 アプリケーション適応形プロセッサの概要

4.2.1 可変構造化の概念

図4.1は可変構造化プロセッサの概念を示したものである。命令形式と命令コードなど命令体系に、依存しない部分と依存する部分に分けると次の2つの部分に大別できる。

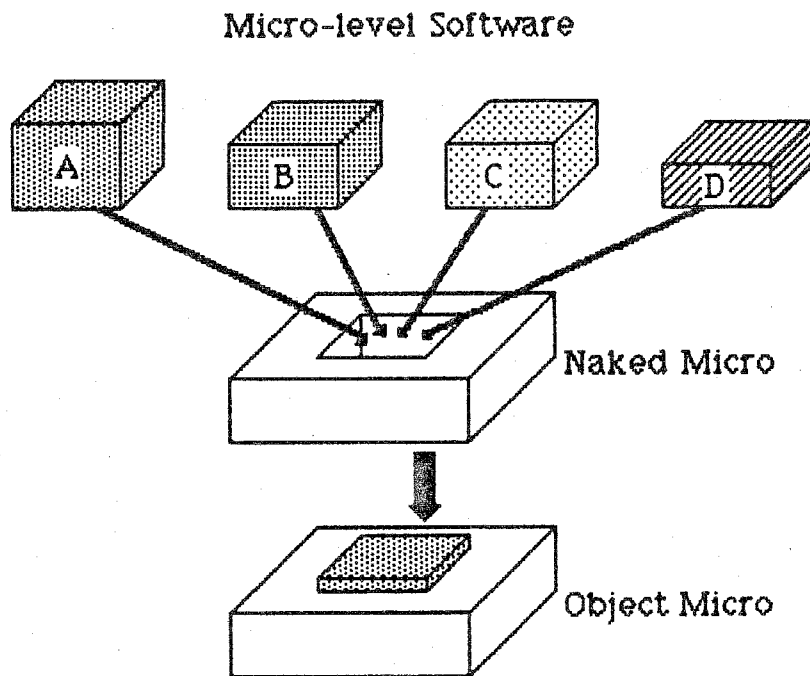


図4.1 可変構造化プロセッサの概念

- (1) 命令体系に依存しない部分：各種レジスタ群と演算回路から成る命令実行の基本部
(Naked Micro)
- (2) 命令体系に依存する部分：マイクロプログラムなど可変構造化のための命令制御部
(Micro-level Software)

この2つの要素によって目的とする命令体系のマイクロプロセッサを構築する。即ち、上記Naked Microに目的のアーキテクチャを閉じ込めたMicro-level Softwareを付加することで1つの命令体系を実行できるマイクロプロセッサ(Object Micro)が得られる。

可変構造化の考えはダイナミック・マイクロプログラミング⁵⁾に代表される手法で古くから研究されてきた。しかし、これらは主に中大型のコンピュータ上で実現しようとするものが多く、ハードウェアは固定的である。ソフトウェア又はファームウェア(マイクロプログラム)によるエミュレーションが主体である。そのため、エミュレーション時の性能

はあまり上がらないのが普通である。本章で提案する「可変構造」は、こうした固定的なハードウェアではなく、半導体チップ上で実現することを前提としている。即ち、1つの統一したプロセッサ構造の中でROM, PLA, 配線などのマスクを変えるだけのマスク・プログラムによって種々のアーキテクチャ(命令体系)を実現しようとするものである。図4.2に示すように、1つのシリコン・チップ上で可変にする部分を予め設定しておく。マイクロプログラムROMやデコーダPLAなどの斜線部分がMicro-level Softwareであり、その他の部分がNaked Microである。

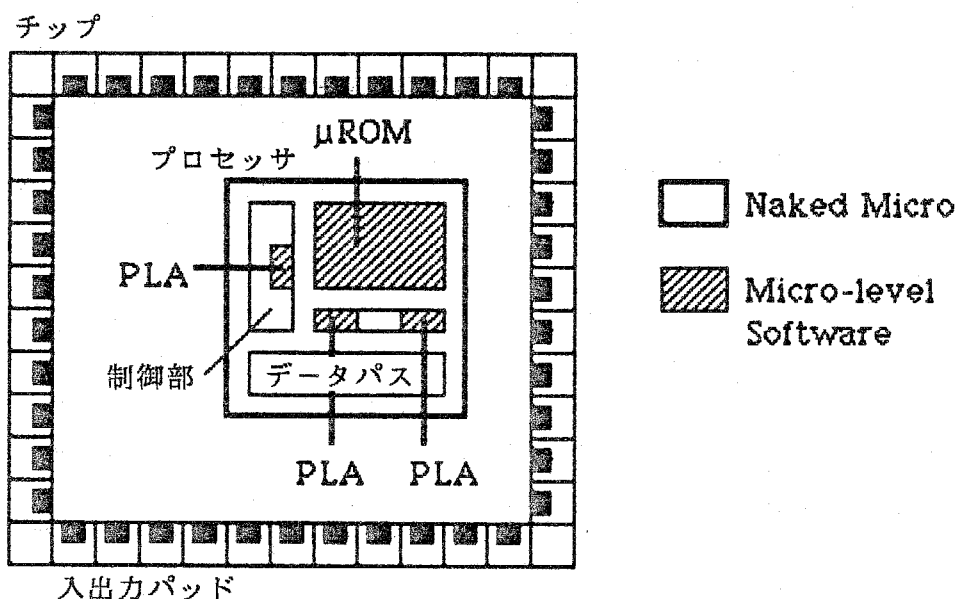


図4.2 可変構造化プロセッサの構成

4.2.2 可変構造化プロセッサ構成法

(1) 従来のプロセッサ構成法

図4.3は従来のマイクロコード化マイクロプロセッサの構成を示したものである。命令体系に最適な「専用構造」を採っており、次の2点が可変構造化のネックとなる。

(a) 命令デコーダ

命令を解釈し、マイクロプログラムメモリ(μROM)に格納される対応マイクロプログラムの先頭アドレスを作成する。複雑な命令形式をもつマイクロプロセッサでは、この命令デコーダが非常に複雑になり、そのサイズも大きくなる。命令デコーダの代わりに、命令の各機能を指定するフィールドを切り出してこれをμROMの入力にする方法もある。

この場合もどこを切り出すかを決めるための命令形式判定回路が必要となり、命令デコーダと同様の回路となる。アーキテクチャによって変化の大きい部分である。

(b) レジスタ

ソフトウェアから見える、即ち命令語から指定されるレジスタは、アーキテクチャによってその数が異なる。また、命令語中のレジスタ指定フィールドはレジスタの数によって長さ(ビット長)及びその位置がアーキテクチャあるいは命令形式によって異なる。この問題はレジスタ自身とその選択回路(レジスタデコーダ)にある。

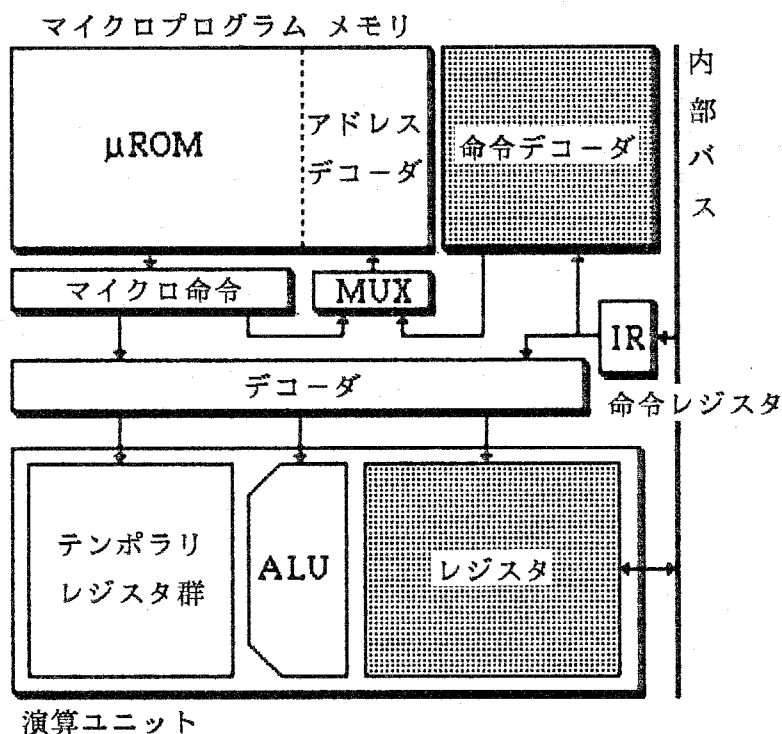


図4.3 従来のマイクロコード化マイクロプロセッサの構成

以上述べた問題点を解決するためには、命令デコーダ、レジスタ、レジスタデコーダを命令体系の制約から切り離す必要がある。

(2) 可変構造化プロセッサ構成法

プロセッサ中で命令体系に依存する部分である命令デコーダ、レジスタ、レジスタデコーダを図4.4に示した方法で構成する。

(a) 命令デコーダ

命令語の最小構成単位(8/16/32ビット)を入力として、マイクロプログラムの先

頭アドレスのみを得るテーブル構成とする。単なるテーブルであれば、従来の命令デコーダと何ら変るところがないが、命令語からマイクロ命令選択までの手順をいかに行うかが課題である。

(b) レジスタ

命令体系によってレジスタ数が異なる以上、十分な数のレジスタをもつ必要がある。図4.3におけるレジスタ群は、演算ユニット中に演算回路(ALU)と共に同一バス上に配置される。従って、ALUの1ビット分の大きさに左右され、あまり小さく構成できない。そこで、図4.4に示すように、レジスタを1つのRAMに集中させて演算ユニットから切り離す。RAM1ビットのサイズは演算ユニット内に納められるレジスタ1ビットのその1/4程度となるから、従来の4倍程度のレジスタ数を確保したとしてもその大きさは変わらない。

(c) レジスタデコーダ

命令デコーダと同様に命令語の最小構成単位(8/16/32ビット)を入力として、命令語中のレジスタ指定フィールド(論理的レジスタアドレス)を切り出し、RAMの1語を選択するアドレス(物理的レジスタアドレス)に変換する、いわばアドレス変換テーブル構成をとる。この場合も命令語からRAM中の1語を選択する手順が重要である。

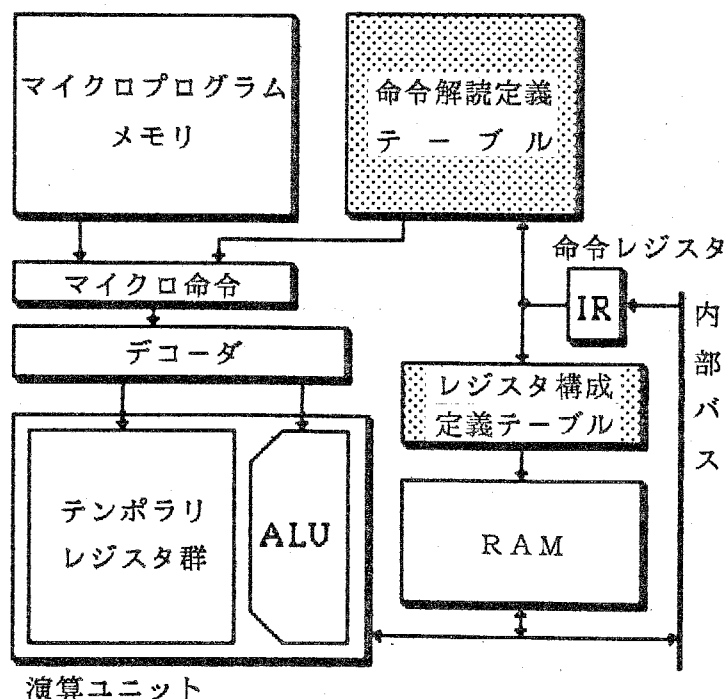


図4.4 可変構造化プロセッサの構成

以下、命令解釈及びレジスタ構成を定義するテーブルの構成法を中心に述べる。

4.3 命令制御方式

4.3.1 命令制御部構成

命令解釈定義テーブルの実現方法として、第3.3節に示した命令解釈機能一体形の μ ROM方式が有力な解となる。この方式は既に詳細に述べたが、図3.5及び図3.8に示したように、 μ ROMのアドレスデコーダ上に命令解釈機構を置く。図4.5は、命令解釈定義テーブルと通常の μ ROMとの接続関係を示したものであり、それぞれ次の働きを行う。

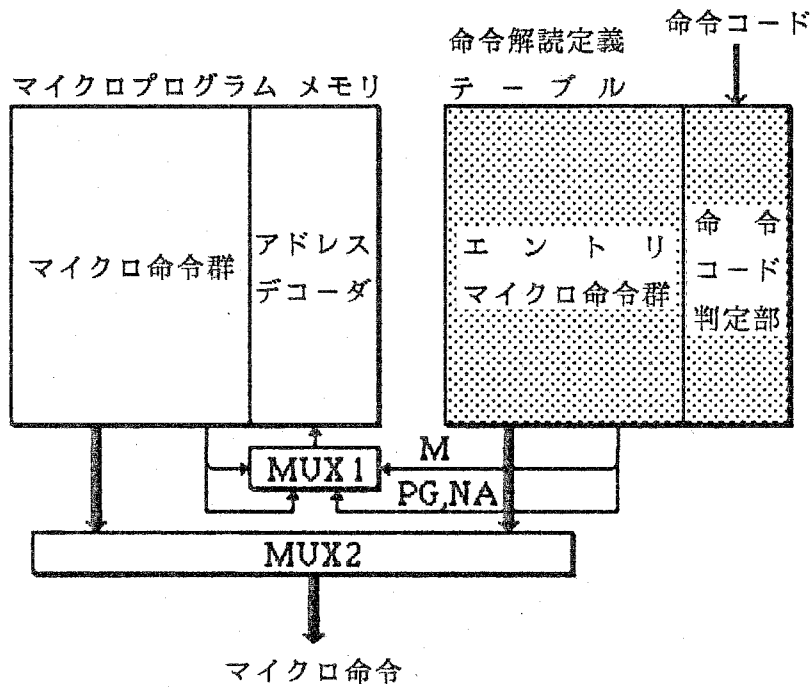


図4.5 命令制御部構成

(1) 命令解釈定義テーブル

命令コードを入力として、これを実行するためのマイクロプログラムの先頭のマイクロ命令(エントリ・マイクロ命令)を出力するテーブルである。但し、同一の命令コードであっても、オペランドのアドレッシング・モードを指定する部分(命令中のEAフィールド)や演算を指定する部分(命令中のOPフィールド)は異なるので、これらを分離してテーブルに書き込んでおかなければならない。そのためにテーブル記述の分離に必要な「ページ

番号」をマイクロ命令中のPG (Page)フィールドに記述してある。

(2) μ ROM

命令解読定義テーブルから読み出されるマイクロ命令中に含まれるPGフィールド及びNA (Next Address)フィールドを入力として、これに対応する次のマイクロ命令を読み出す。テーブルと μ ROMそれぞれから読み出されるマイクロ命令のいずれを使用するかは、更に、マイクロ命令中のM (Mode)フィールド1ビットが0か1かで分ける。Mフィールドが“0”の場合にテーブルを、“1”の場合に μ ROMを選ぶ。また、マイクロ命令中のPGフィールドを μ ROMのアドレスデコーダに入力している理由は、定義テーブルと μ ROMを通して「ページ番号」を自由に設定できるようにするためである。

以上示したように、命令解読定義テーブルと μ ROMとを独立に配置すると命令制御部の構成が複雑になり、更にマルチプレクサ(MUX₁, MUX₂)や配線領域が増加して命令制御部全体のサイズ増加とマイクロサイクル時間の低下に及ぶ。そこで、定義テーブルと μ ROMとの合体を図り、無駄な論理を省くことにより高速化を得る必要がある。

(3) 命令解読定義テーブル内蔵 μ ROM

図4.6に命令解読定義テーブルを μ ROMに内蔵した命令制御部の構成を示した。

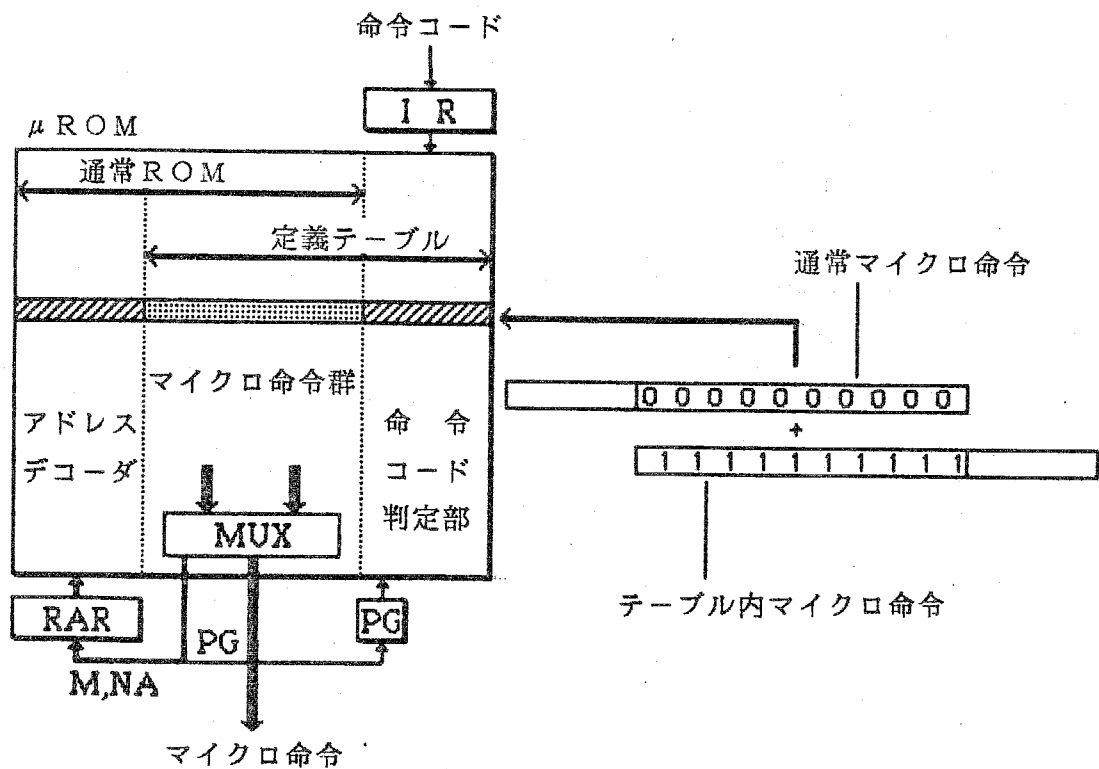


図4.6 命令解読定義テーブル内蔵 μ ROM

定義テーブルを命令コード判定部(アドレスデコーダに相当)と μ ROMのアドレスデコーダをマイクロ命令記憶部の両側に配置し、1つのワード線をそれぞれのアドレスデコーダで共有するようにする。従って、1つのワード線には定義テーブル及び μ ROMそれぞれ1つのマイクロ命令コードを互い違いに配置し、マイクロ命令中のMフィールドの内容により、 μ ROM中に内蔵したマルチプレクサ(MUX)でいずれか一方のマイクロ命令を選択するようにする。このような構成によって、図4.5と全く同じ機能を果たす上に、命令制御部のサイズ縮小とマイクロサイクル時間の向上を達成することができる。

以上のようにして、命令解読定義テーブルと μ ROMとを合体した μ ROMにおける2つのアドレッシング・モードを図4.7に示す。

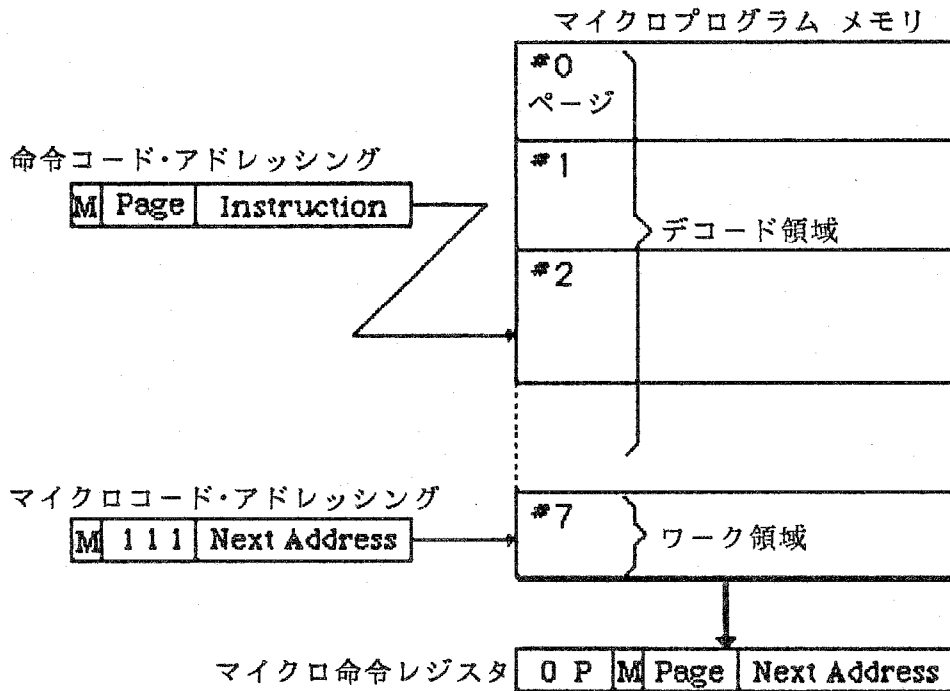


図4.7 命令解読定義テーブル内蔵 μ ROMのアドレッシング・モード

(1) マイクロコード・アドレッシング

通常のマイクロプログラム制御方式と同様に、マイクロ命令中のNA(Next Address)フィールドの内容で次のマイクロ命令をアクセスするモードである。このモードにするために、前記した同マイクロ命令中のMフィールドを“1”にする。図4.7の例では、本

アドレッシング・モードの場合、常にPGフィールドを“111”とする。勿論、他のページをマイクロコード・アドレッシング領域(ワーク領域)とすることもできる。

(2) 命令コード・アドレッシング

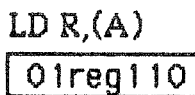
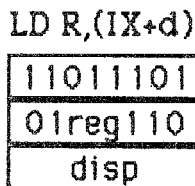
命令語を格納する命令レジスタの内容にマイクロ命令中のPGフィールドを連結させて次のマイクロ命令をアクセスするモードである。マイクロ命令中のMフィールドが“0”の時、本アドレッシング・モードとなる。

このように、命令コードに従ってマイクロ命令を読み出す必要のある場合には命令コード・アドレッシング・モードを選択してμROM中の命令解読定義テーブルから所望のマイクロ命令をアクセスする。また、命令コードに無関係な処理ではマイクロコード・アドレッシングを選ぶ。

4.3.2 命令解読定義テーブルの構成法

図4.8は、8ビットマイクロプロセッサの命令語、処理内容とこれに対するμROM中の命令解読定義テーブルの構成を示したものである。

(a) 命令コード



(b) μROM構成

命令解読定義テーブル		
①	000	11011101 2nd OP-code fetch, → L1
⑤	010	01XXX110 Operand fetch, → L100
⑨	000	01XXX110 Operand fetch, → L2
③	001	01XXX110 3rd OP-code fetch, → L10
ワーク領域		
②	111	L1 PC→MAR,PC+1, → PG(010)
④	111	L10 IX+d→MAR, → PG(011)
⑥	111	L100 PC→MAR,PC+1, → L101
⑦	111	L101 OP fetch,ORD→R, → L102
⑧	111	L102 PC→MAR,PC+1, → PG(001)
ページ	アドレス	マイクロ命令

図4.8 8ビットマイクロプロセッサにおける命令解読定義テーブルの構成

(1) 命令コードの意味

図4.8(a)は2つの命令語のコードを示している。第1は3バイトからなるインデックス・アドレッシング・モードのロード命令, $LD R, (IX+d)$, 第2は1バイトからなるアドレスレジスタ間接アドレッシング・モードのロード命令 $LD R, (A)$ である。それぞれのコードには同図に示したようなアドレッシング・モード, 命令の種類, レジスタ指定, 変位コード(displacement)の有無を示すコードが含まれる。

(2) 処理内容と命令解読定義テーブル構成

$LD R, (IX+d)$, $LD R, (A)$ の2命令が連続した場合の命令処理とこれに対応した命令解読定義テーブルとワーク領域の μROM 構成を図4.8(b)に示す。

- ① 命令レジスタに命令が格納されるとマイクロ命令でページ“000”の命令コード・アドレッシングを指定して, この部分がアクセスされる。ここでのマイクロ命令は第2バイト目の命令フェッチを指示すると共に, マイクロコード・アドレッシングでL1番地へ分岐する。
- ② ここではPC(Program Counter)のMAR(Memory Address Register)へのセットと更新を行い, ページ“001”の命令コード・アドレッシングで命令解読定義テーブルへ分岐する。
- ③ 第3バイトの目のdispの読み出しが指示され, マイクロコード・アドレッシングでL10番地へ分岐する。
- ④ インデックス計算($IX+displacement$)とこの結果のMARへのセットを行い, ページ“010”の命令コード・アドレッシングで再び命令解読定義テーブルへ分岐する。
- ⑤ 前サイクルでのインデックス計算の内容に従ってオペランドのフェッチを行い, マイクロコード・アドレッシングでL100番地へ分岐する。
- ⑥ 次の命令の先取りのためPCのMARへのセットと更新を行い, マイクロコード・アドレッシングでL101番地へ分岐する。
- ⑦ 次の命令の第1バイト目のフェッチを開始し, ⑤サイクルで行ったオペランド・フェッチの結果をレジスタへ格納する。このレジスタ指定方法は次節で述べるレジスタ制御方式で詳細を述べる。マイクロコード・アドレッシングでL102番地へ分岐する。

⑧ 更に次の命令(第1バイト目または⑦でフェッチした命令の第2バイト目)フェッチの準備を行い、命令コード・アドレッシングでページ“000”の命令解読定義テーブルへ分岐する。

⑨ ここからLD R,(A)命令の処理が開始される。

以上示したように、命令コード・アドレッシングとマイクロコード・アドレッシングを交互に繰り返すことで命令処理を行う。また、LD R,(IX+d)命令の第2バイト目のコードとLD R,(A)命令の第1バイト目とは同一コードであるが、命令解読定義テーブルのページを分けることでマイクロプログラムの分岐処理が異なる。

4.4 レジスタ制御方式

4.4.1 レジスタ制御部構成

マイクロプロセッサにおける各種レジスタ類は、実装密度の高いRAMに集中して収納する(RAMベース・アーキテクチャ)。図4.9はこのRAM周辺の構成を示したものである。RAM、RAM内の記憶部を各種レジスタに定義するレジスタ構成定義テーブル、このテーブルとマイクロコードからのレジスタ指定のいずれか一方を選定するマルチプレクサ(MUX)から成る。

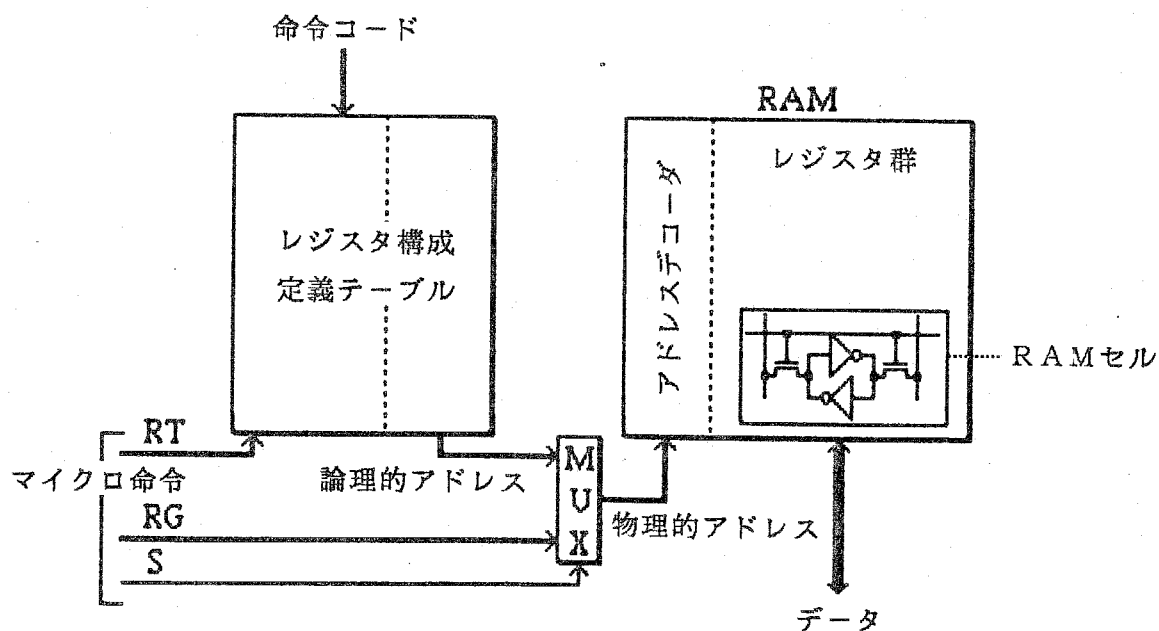


図4.9 レジスタ制御部構成

それぞれの働きは次のとおりである。

(1) RAM

命令実行過程に必要なワークレジスタを含めたレジスタ群を収納する。アドレスデコーダと記憶部から成る。

(2) レジスタ構成定義テーブル

命令コードを入力として、この中のレジスタ指定フィールドの内容(論理的レジスタアドレス)を判定してRAMの対応するアドレス(物理的レジスタアドレス)を出力するテーブルである。本テーブルの機能は前記した命令解読定義テーブルと同様の考え方である。1つの命令コードに2つのレジスタ指定がある場合にいずれのレジスタ指定フィールドを選択するかを決定するため、マイクロ命令には前記したPGフィールドに相当するRT(Register Type)フィールドが前述してある。また、本テーブルの出力とマイクロ命令中のレジスタ指定フィールド(RGフィールド)のいずれかを選択するマルチプレクサ(MUX)の制御は前記したMフィールドに相当するS(Select)フィールドを設けている。

以上示したように、レジスタ構成定義テーブルとマイクロ命令中のRGフィールドのパスを分けて構成すると、特にレジスタ構成定義テーブルを経由するパスが厳しくなり、マイクロサイクル時間を増加させる。これも丁度、命令解読定義テーブルの場合と同じである。また、命令コード中のレジスタ指定フィールド(論理的レジスタアドレス)からRAMアドレス(物理的レジスタアドレス)への変換部とRAMのアドレスデコーダの2種類のデコーダを必要とし、RAMまわりのサイズ増加を招く。そこで、命令制御部と同様に、レジスタ構成定義テーブルとRAMを一体化した構成を考える。

(3) レジスタ構成定義テーブル内蔵RAM

図4.10はレジスタ構成定義テーブルとRAMを一体化したレジスタ制御部の構成を示したものである。命令から直接アクセスするためのレジスタ構成定義テーブルとマイクロ命令からのアクセスを行なう普通のデコーダから成るアドレスデコーダと、レジスタ部を収容する記憶部から成る。この構成は、通常のスタティックRAMと同一の構成となるので、図4.9に示した構成での性能低下はなくなる筈である。以下に、それぞれの構成要素の役割を述べる。

(a) アドレスデコーダ

アドレスデコーダは図4.9におけるマルチプレクサ(MUX)を制御するマイクロ命令

中のSフィールド及びRGフィールドを同マイクロ命令中のRTフィールドに含めた上で、レジスタ構成定義テーブルのコード判定部とマイクロ命令中のRTフィールドのデコーダを1つのアドレスデコーダとして合成する。このような構成をとることにより、図4.9に示した構成と全く同様の機能を果たす上に、レジスタ制御部のサイズ縮小と高速RAMアクセスが可能となる。

(b) 記憶部

記憶部は図4.9のRAM構成からアドレスデコーダを除いたものである。前記したアドレスデコーダから出力されるレジスタ選択信号を駆動するドライバを備えている。各ワード毎にレジスタが配置される。

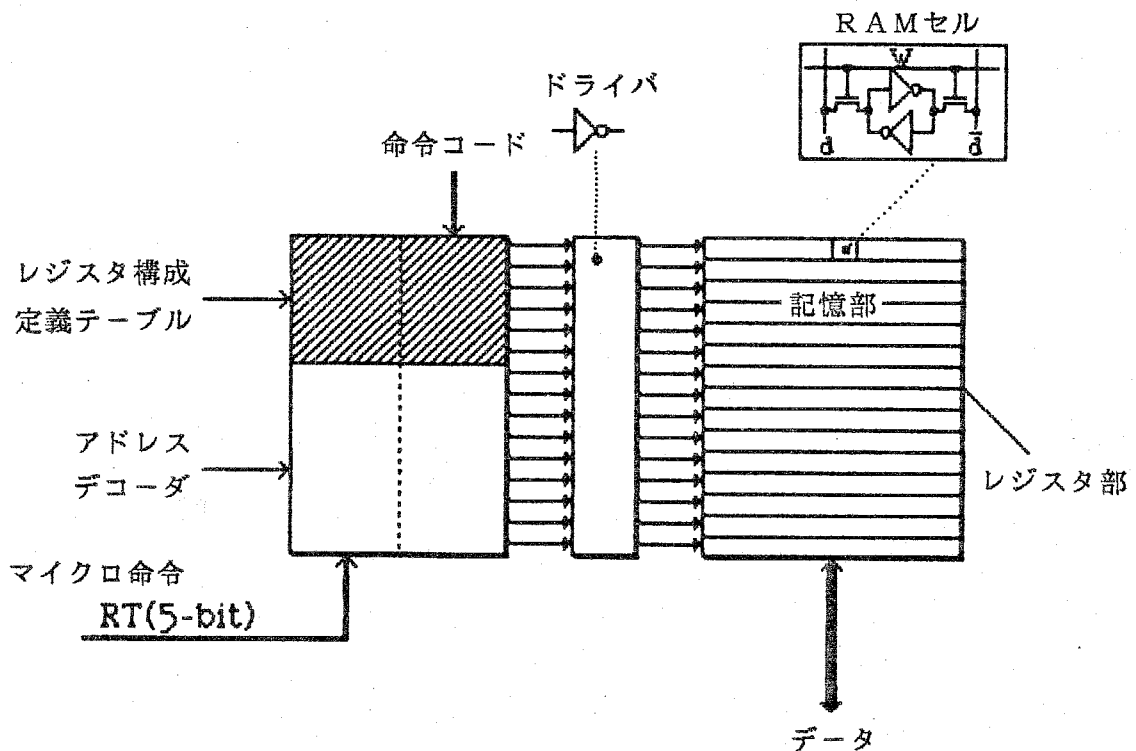


図4.10 レジスタ構成定義テーブル内蔵RAM

以上のようにして、レジスタ構成定義テーブルとRAMを一体化した構成でのRAMアドレッシング・モードを図4.11に示す。

(1) マイクロコード・アドレッシング

マイクロ命令中のRTフィールドは5ビットで構成され、上位2ビットが“00”以外に限って下位3ビットとの組み合わせで最大24個のレジスタを指定することができる。本アドレッシング・モードは基本的にはワークレジスタを指定するのに用いる。

(2) 命令コード・アドレッシング

命令語を格納するレジスタの内容とマイクロ命令中のRTフィールドとでRAM中の1語をアクセスするモードである。RTフィールドの上位2ビットが“00”の場合、本アドレッシング・モードとなる。

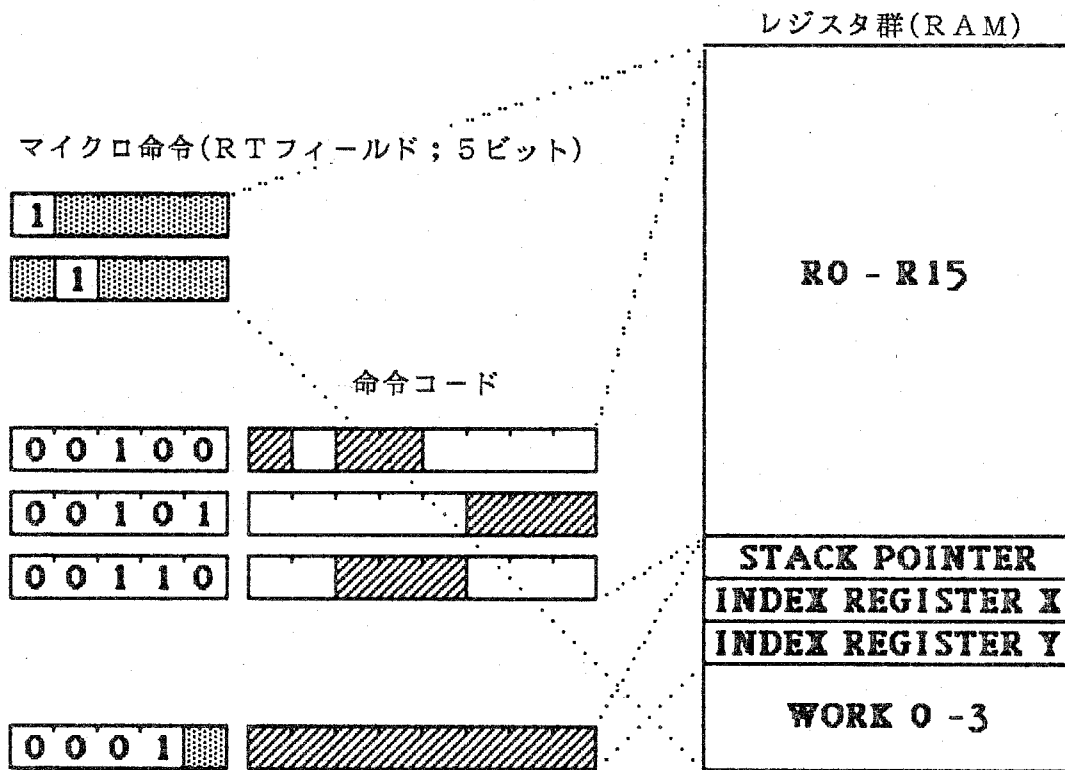


図4.11 RAMアドレッシング・モード

以上示したように、命令コードに含まれるレジスタ指定コードに従ってRAMをアクセスする場合には命令コード・アドレッシング・モードを、マイクロプログラム中で命令コードには無関係なワークレジスタなどをアクセスする場合にはマイクロコード・アドレッシング・モードを選ぶ。RAM中のレジスタには2つのアドレッシング・モードでアクセスできるものがある。これは、特殊なレジスタ指定では命令コード・アドレッシングだけでなく、マイクロ命令による直接指定を持っている方が効率のよいことがあるからである。

4.4.2 レジスタ構成定義テーブルの構成法

図4.12は、8ビットマイクロプロセッサの命令語とこの中のレジスタ指定フィールドを解釈する、レジスタ構成定義テーブルの構成を示したものである。レジスタ指定フィールドで指定されたレジスタは、それ以外の部分のコードによってその使い方が決められており、これに関しては前節に詳しく述べた。ここでは、レジスタ部分に限定して述べることにする。レジスタ間転送命令TFR Ri,Rj, スタック操作命令PUSH, POPを例題にレジスタ構成定義テーブルの構成を示す。

(1) レジスタRiの選択

図4.12に示した命令語は8ビットであり、上位2ビットがレジスタ転送を示すコードであり、下位6ビットがレジスタRi, Rjの指定フィールドとなっている。まず、転送のソースとなるRiの選択は、マイクロ命令中のRTフィールドのコードを“00110”としておく。これによってレジスタ構成定義テーブル内のコード判定部にRiのフィールドのみ8種類(R₀~R₇)のレジスタ選択信号を発生させる。

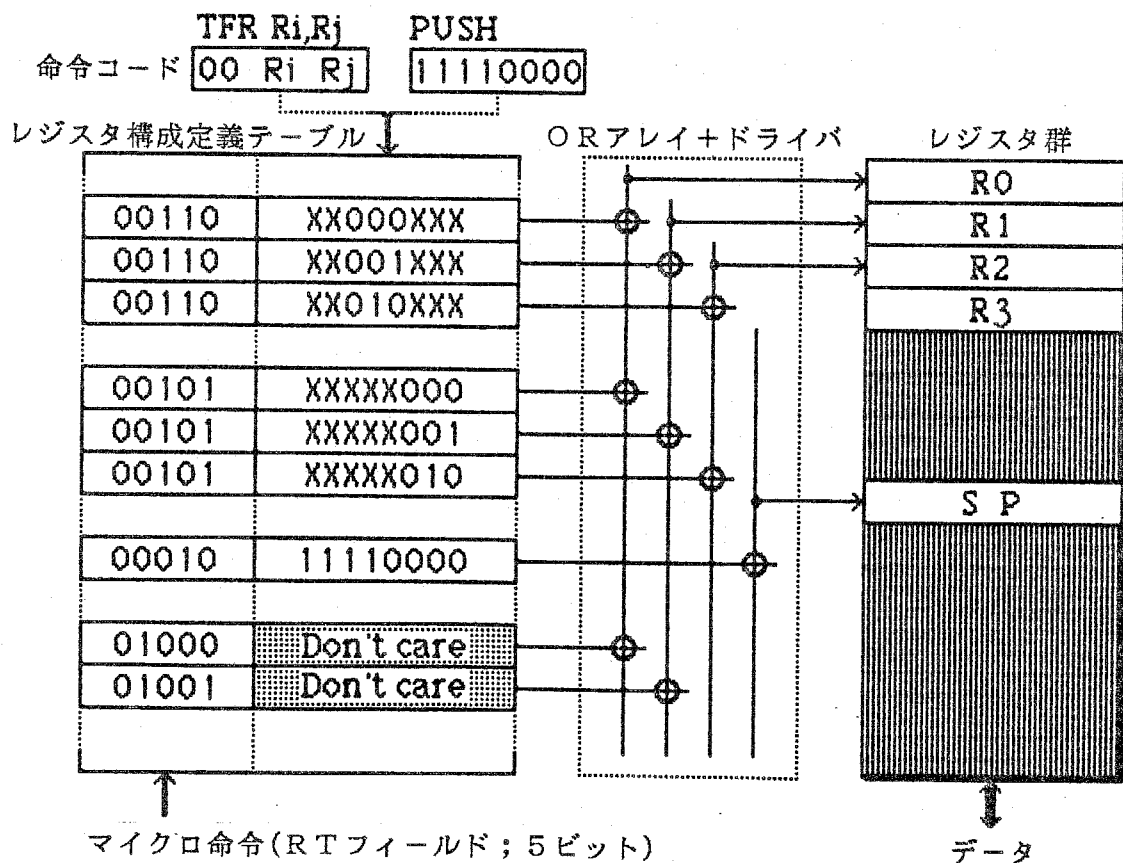


図4.12 8ビットマイクロプロセッサにおけるレジスタ構成定義テーブル

(2) レジスタ R_jの選択

レジスタ R_jと同様にマイクロ命令中の RT フィールドのコードを“00101”として、レジスタ構成定義テーブル内のコード判定部に R_jのフィールドのみ8種類(R₀~R₇)のレジスタ選択信号を発生させる。

以上の2つのレジスタ指定フィールド R_i, R_jの解釈結果のレジスタ選択信号は2重にあるため、これを図4.12に示したORアレイによって1組のレジスタ選択信号に絞り込む。

(3) インプライドのレジスタ指定

8ビットの命令コード全体でレジスタを指定する命令、例えばスタックポインタ(SP)を使うPUSH, POP命令、インデックスレジスタ(IX)を用いるINCX, DECX命令などではマイクロ命令中のRTフィールドのコードを“00010”に指定して行う。

以上は命令コード・アドレッシングの場合であるが、マイクロコード・アドレッシングの場合は、前記したようにマイクロ命令中に含まれるRTフィールドのコード上位2ビットを“00”以外のコードにして下位3ビットと組み合わせてレジスタを選択するようにする。命令コード・アドレッシングでのレジスタ選択と一致するものがあり得るが、これは前記したORアレイによって1本に絞り込む。従って、レジスタ構成定義テーブルは、命令コード及びマイクロコードによって指定され、レジスタ選択信号を生成するコード判定部(ANDアレイ)と、この結果を統合してRAMの記憶部のワード線に絞り込むORアレイから成る。

4.5 LSI構成と機能

(1) LSI構成

可変構造化プロセッサ構成法を実際に適用したシステムオンチップ形CMOS8ビットマイクロコンピュータHD64180のチップ全体構成及びチップ写真をそれぞれ図4.13, 図4.14に示す。このマイクロコンピュータは、2 μ mCMOSプロセスによって製造され、全体で45,000MOSトランジスタを集積している。また、5.98 \times 5.90mmと極めてコンパクトなチップサイズとなっている。そのLSI仕様は表4.1に示すとおりである。20MHzの入力クロック(マイクロサイクル10MHz)の場合、最小命令実行時間0.3 μ sである。

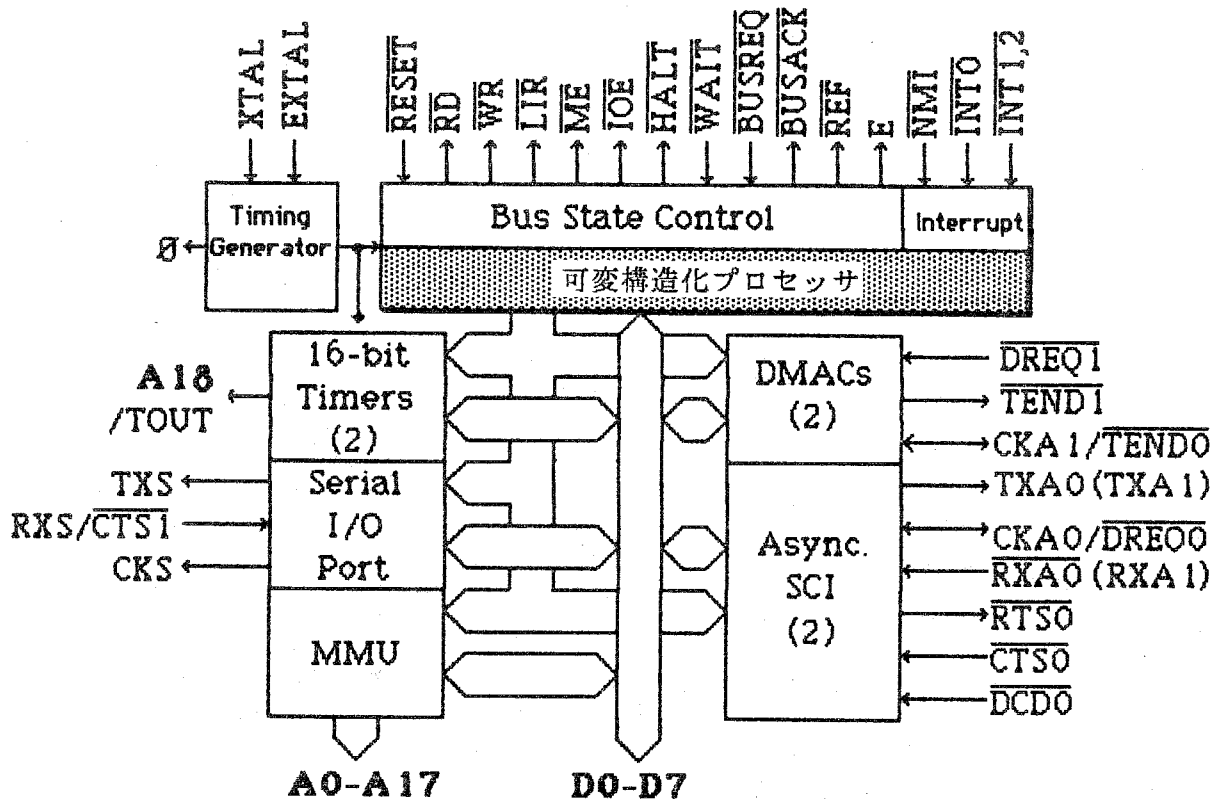


図4.13 CMOS 8ビットマイクロコンピュータHD64180の構成

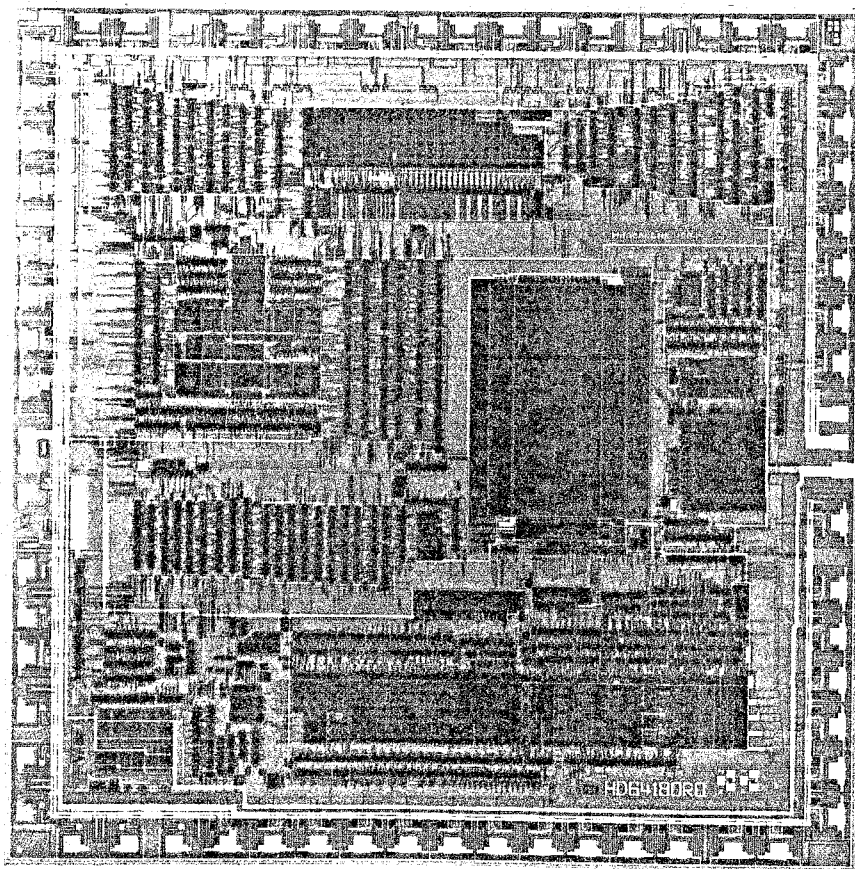


図4.14 HD64180のチップ写真

(2) 機能

表4.2はマイクロコンピュータ仕様を示したものであり、本章で述べた可変構造化プロセッサは165の命令を有する高機能なマイクロプロセッサに適用された。また、同表に示すように、多くの周辺I/O機能をプロセッサと共に同一チップ上に集積しており、システムの性能/価格比を極めて向上する。

表4.1 HD64180のLSI仕様

項目	内容
プロセッサ	2 μ m CMOS
集積度	45,000MOSトランジスタ
チップサイズ	5.98 × 5.90 mm
消費電力	50mW
動作周波数	20MHz
マイクロサイクル	10MHz (100ns)

表4.2 マイクロコンピュータ仕様

項目	内容
命令数	165
最小命令実行時間	0.3 μ s
メモリ空間	512Kバイト
周辺I/O機能	メモリ管理機構 (MMU) DMAコントローラ (2ch.) シリアル・インタフェース 非同期 : 2ch. 同期 : 1ch. 16ビット・タイマ 2本 DRAMレフレッシュ機能 クロックパルスジェネレータ

4.6 まとめ

システムオンチップ形マイクロコンピュータにおけるカーネル・プロセッサの可変構造化構成法について述べた。多様なマイクロコンピュータ応用に対し、目的に適合する命令体系を1つのプロセッサの中のマイクロコードを含む命令解読定義テーブル、レジスタ構成定義テーブルの内容をマスクで書き換えることによって実現する可変構造化構成法を示した。また、各可変要素の構成に当り、可変構造がゆえに性能低下及びサイズ増大する部分をLSI上での回路構成の自由度を活かしながら性能を低下することなく、小さなサイズで達成できた。

以上で得た可変構造化マイクロプロセッサ構成は、定義テーブルにCAM(Content Addressable Memory), μ ROMにRAMを用いれば、動的にアーキテクチャを切り換え得る高度なものとなる。また、EPROMによっても大きな効果が得られる。更に、本プロセッサ構成法は、マイクロプログラミングの効果を一層高めるシステムティックな設計法としても意義が大きいと考えている。

4.7 第4章の参考文献

- (1) T. Itoh et al.: "A 6000-Gate CMOS Gate Array", ISSCC Digest of Technical Papers, Vol.25, pp.176-177 (1982-2)
- (2) K. Takeda et al.: "A Single Chip 80b Floating Point Processor", ISSCC Digest of Technical Papers, Vol.28, pp.16-17, (1985-2)
- (3) H. Maejima et al.: "VLSI for High Performance Graphic Control Which Utilizes Multi-Processor Architecture", Proceedings of the International Conference on Computer Design, pp.586-591, (1983-10)
- (4) H. Maejima et al.: "A CMOS Microprocessor with Instruction-Controlled Register File and ROM", ISSCC Digest of Technical Papers, Vol.28, pp.12-13 (1985-2)
- (5) W.T. Wilner: "Design of the Burroughs B1700", Proceedings of FJCC, pp.579-586 (1972)

第5章 専用プロセッサの機能分散化 マルチプロセッサ構成法

第5章 専用プロセッサの機能分散化マルチ プロセッサ構成法

5.1 まえがき

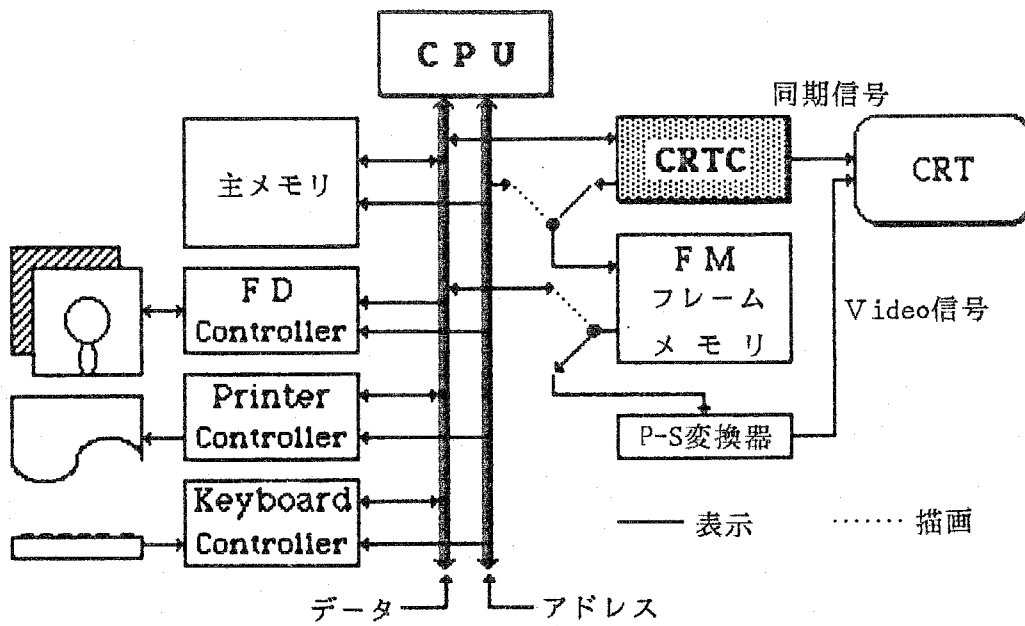
最近のマイクロプロセッサの動きで特徴的な方向は、目的に合わせて専用化した特定用途形マイクロプロセッサの出現である。グラフィック処理^{1)~3)}、イメージ処理⁴⁾、通信制御⁵⁾、ディスク⁶⁾等のファイル制御がその例である。これまで、これらの分野の多くは汎用マイクロプロセッサを用いてソフトウェア処理によって実現していたが、システムの機能向上に伴う性能低下が問題になる応用では、ビットスライス形のバイポーラ・マイクロプロセッサ等を用いて高性能化を図っている。この場合、システムを構成するにはLSIの他にMSI (Medium Scale Integration)やSSI (Small Scale Integration)を必要とするため、システム全体の部品数が多くなること、これに伴い消費電力が増大することからシステム実装規模の増大と、システム価格の上昇が大きな問題となる。このため、高性能化はワークステーション・クラス以上のシステムにしか望めない状況にある。ここで要求されるのはシングルチップ化した特定用途向きの専用マイクロプロセッサである。専用マイクロプロセッサは本来の目的通り、性能と機能の向上をコンパクトな形で実現するものであり、シングルチップ化には多くの難しい問題をかかえている。

本章ではグラフィック処理用の専用マイクロプロセッサ^{1) 2)}を例に挙げ、先ずグラフィック・システム構成とそこに要求される仕様を示す。この仕様を満足し、効率の良い設計を行えるオンチップの機能分散化マルチプロセッサ構成法と階層論理設計法を提案する。最後に、実際に筆者らによって開発されたグラフィック処理専用プロセッサでの方式の効果进行を述べる。

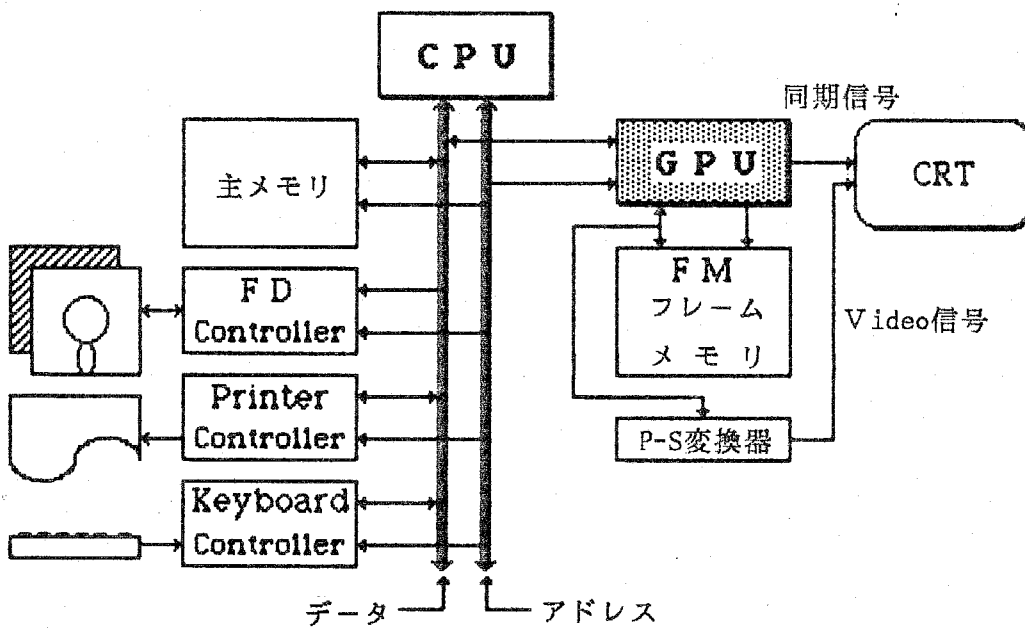
5.2 グラフィック処理専用プロセッサの概要

5.2.1 グラフィック・システム構成

図5.1はパーソナル・コンピュータにおけるグラフィック処理を行うためのシステム構成例を示したものである。グラフィック処理を行う主体が主プロセッサか専用プロセッサかで2形態に分けられる。いずれの場合でも、グラフィックを表示するCRT装置を制御するコントローラは必須である。



(a) 主プロセッサ(CPU)によるソフトウェア処理



(b) 専用プロセッサ(GPU)によるハードウェア処理

図5.1 グラフィック・システム構成例

(1) 主プロセッサによるソフトウェア処理

図5.1(a)は主メモリ空間に位置して、図形データを格納しこれを表示するためのフ

フレームメモリ(FM: Frame Memory)上に主プロセッサ(CPU: Central Processing Unit)がソフトウェアによって図形を描画するものである。FM上に描画された図形は、CRTC(CRT Controller)⁷⁾によりCRT上に表示される。このシステムでは、汎用マイクロプロセッサによって構成する主プロセッサ(CPU)が、その命令の組み合わせによって直線や円等の図形を構成する画素(ドットまたはピクセル)の位置を逐次計算(DDA: Differential Digital Analyzer)し、これを主メモリ空間にあるFM上に描画する。CRTへの表示の際、FMは主メモリ空間から切り離され、CRTCによって制御される。FMより読み出された図形データは並列・直列変換されビデオ信号としてCRTへ送出される。従って、CPUによるFMへの描画実行中ではCPUが他の処理を進められないこと、CPU命令自体がグラフィック専用命令でないために効率が低いこと等の理由で高速描画が望めない。グラフィック性能の向上のためには、CPU処理の中からグラフィック処理の一部の機能を分散化する必要がある。

(2) 専用プロセッサによるハードウェア処理

図5.1(b)は主メモリ空間から分離したフレームメモリ(FM)上にグラフィック処理専用プロセッサ(GPU: Graphic Processing Unit)がCPUとは独立して描画及び表示を実行するものである。このシステムでは、主メモリ上のセグメントバッファと呼ぶ部分にCPUによって生成された直線や円等のマクロなグラフィック・コマンド/パラメータ群をCPUが逐次読み出し(あるいはCPUがGPUへ転送し)、これを実行することによってFM上に図形を描画する。FM上の図形データはGPUによって主メモリとは独立に読み出されてCRTに表示される。従って、このシステムではCPUによるグラフィック・コマンド群の生成を含む処理と、GPUによる図形描画、表示処理が並列に行えるため高速描画及びシステム全体の高性能化が図れる。

以下、本システム構成に必要なグラフィック処理専用プロセッサ(GPU)に要求される仕様を示してその実現上の問題点を洗い出す。

5.2.2 要求仕様

図5.1(b)に示されたGPUは、ラスタ走査形CRTにグラフィック及びキャラクタを表示するための専用プロセッサである。フル・ビットマップ方式による高速グラフィック描画機能とCRTへの表示制御機能を備えたシングルチップのマイクロプロセッサである。

(1) 位置づけ

GPUはグラフィック・システムにおけるキー・デバイスであり、図5.2に示したようにキャラクタのみの簡単な表示機器から大規模なフル・グラフィック・システムまでの広範なアプリケーションに応用し得ることを目標とする。そのため、GPUには次の性格が要求される。

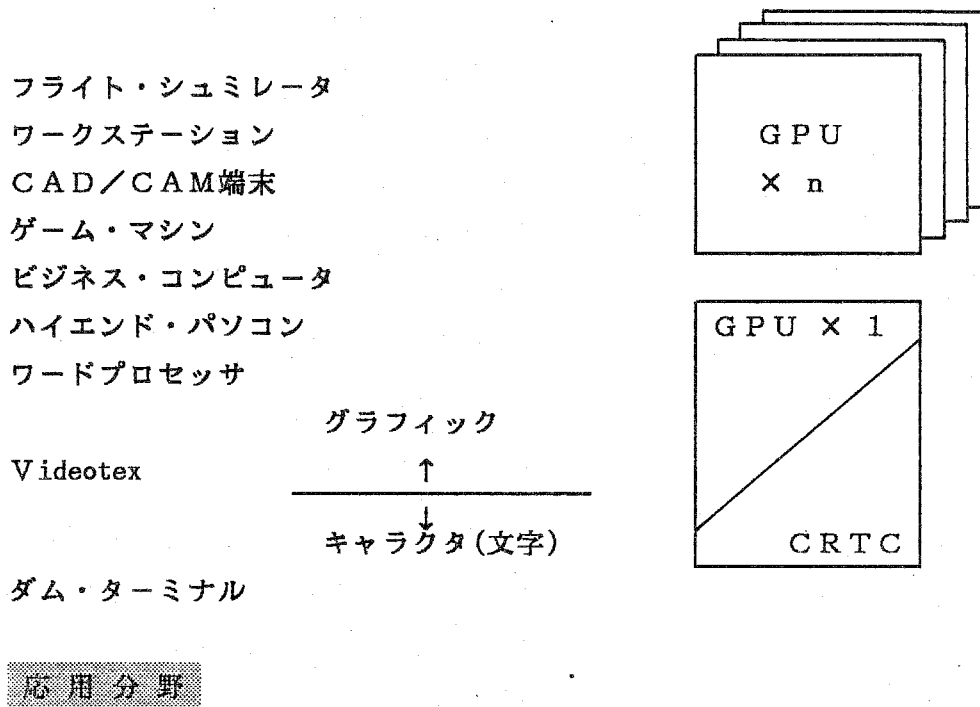


図5.2 GPUの応用範囲

(a) 高レベルなソフトウェア・インタフェース

最近ではVDI (Virtual Device Interface)あるいはCGI (Computer Graphic Interface)やGKS (Graphic Kernel System)⁸⁾などグラフィックスにおける標準化の方向がある。このようなマクロなレベルとの親和性をとっておく必要がある。現段階のLSI技術では、それらを直接サポートすることは不可能である。また、汎用性向上も含めてGPUではprimitiveな描画コマンドを高速に実行することで対応する。

(b) プログラマビリティ

表示機器は、システムの個性が最も顕著に現われるところである。そのため、種々の機能が固定的な押し着せであってはならない。自由にシステムを構築し得るように、LSIとしてのプログラマビリティが必須である。

(c) 同期動作

高度なアプリケーションでは1つのGPUの能力だけでは機能的、性能的に不十分な場合が起り得る。例えば、大画面、多色多階調、多画面分割などへの応用では複数台のGPUを用いて対処する必要があるだろう。ここでの同期動作が可能なこと、また、GPUは文字多重など一般のテレビに内蔵されて使用されるニーズも大きい。この場合、テレビの同期信号にも同期して動作する能力が必要である。

(d) 将来への拡張性

グラフィック専用といってもシステムに応じて要求される機能の内容が異なる。LSIとしてのプログラマビリティだけでは対応できないものもある。LSIチップ構成レベルからの「プログラマビリティ」が望まれる。

(2) 仕様

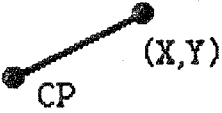


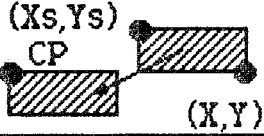

(a) 高度な描画コマンド

図5.3はGPUの代表的な描画コマンドの例を示したものである。コマンドは1ワード(16ビット)のオペレーション・コードとそれに続く数ワードのパラメータから構成される。描画点の座標系は2次元のX-Y座標を用いることにし、座標系の原点に対する絶対値指定と現在の描画点(CP: Current Point)に対する相対値指定を使い分けることができる。直線、円などの線描画のほか、塗りつぶしやコピーなどの2次元コマンドを持たせる。線描画における線種情報や面描画における模様パターンは、32バイトから成るパターンRAMを用いて自由に定義できる。CPUの分担するグラフィック・ソフトウェアでは、線種、模様パターン、カラーなどの属性記述を図形の記述から切り離すことができるため、その生産性を高めるとともに、ハードウェアに依存しない移植性の高いソフトウェア記述が可能となる。

LINEコマンドは、CPを始点とし、パラメータで指定される終点までの直線を発生する。POLYGONコマンドは、CPからパラメータで与えられる点を順次結んで多角形を描画する。CIRCLEコマンドは、パラメータで半径Rを指定するだけで、CPを中心とする円を発生する。楕円、円弧も発生できる。PAINTコマンドは、任意の閉領域内を模様パターン(パターンRAM内のデータ)を用いて高速に塗りつぶすことができる。これにより、地図の塗り分けや円グラフのハッチングなども容易に実現することができる。COPYコマンドは、フレームメモリ内の矩形領域を他の領域に高速転送する。90°単

位の回転やミラー反転も可能で、種々の図形編集に応用できる。

以上の描画はモノクロ、カラーを問わず最高400ns/ドットで高速に実行される。

ニモニック	内 容	描画図形	コマンド
ALINE X,Y	Absolute Line		OP X Y
RRCT DX,DY	Relative Rectangle		OP DX DY
CRCL R	Circle		OP R
AGCP Xs,Ys, X,Y	Absolute Graphic Copy		OP Xs Ys X Y
PAINT	Paint		OP

CP : Current Point

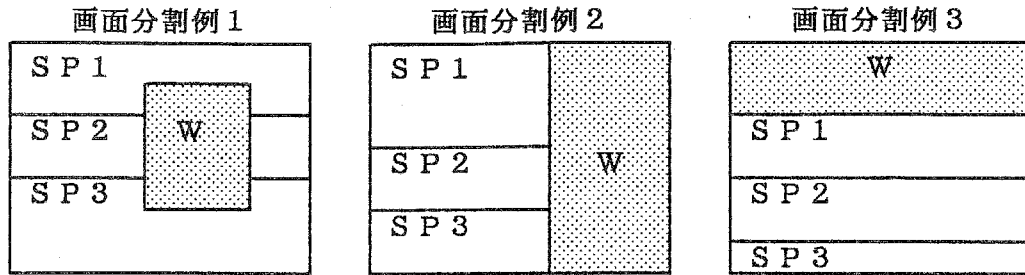
図5.3 GPUの代表的な描画コマンド

(b) 大容量フレームメモリ空間

フレームメモリは主メモリと分離して存在するため、CPUのメモリ空間を侵蝕することなく、グラフィック用に最大2Mバイト、キャラクタ用に最大128Kバイトを持てる。グラフィック用フレームメモリの最大容量は、モノクロ表示で4096×4096ドットの超高精細CRTに対応するものである。また、256色(8ビット/画素)のカラー表示で2048×1024ドットの高精細CRTを1つのGPUでサポートできる。

(c) 画面分割表示

図5.4に示すように、表示画面を最大で水平3分割、更に1つのウィンドウ画面を設定することができる。これらの画面分割の位置、ウィンドウ画面の大きさや位置は任意にプログラムできるものとし、多様な画面構成が可能である。



W : Window
 SP 1 : Split Screen # 1
 SP 2 : Split Screen # 2
 SP 3 : Split Screen # 3

図 5.4 画面分割表示

(d) 画面の重ね合せ (Superimpose)

互いに独立な 2 つの画面を重ね合せて表示する機能 (Superimpose) を有する。これは、1 表示サイクルに第 1, 第 2 の画面の表示データをフレームメモリから交互に読み出すことにより実現する。

(e) スムース・スクロール

水平・垂直方向独立にドット単位の滑らかな画面移動を可能とする。また、文字 (水平方向)、行 (垂直方向) 単位の画面移動 (スクロール) も可能とする。

(f) 拡大表示

図 5.5 に示すように、分割画面単位に、1 から 16 までの整数倍で水平・垂直方向独立に拡大表示が可能。表示画面のみの操作であり、フレームメモリの内容は不変である。

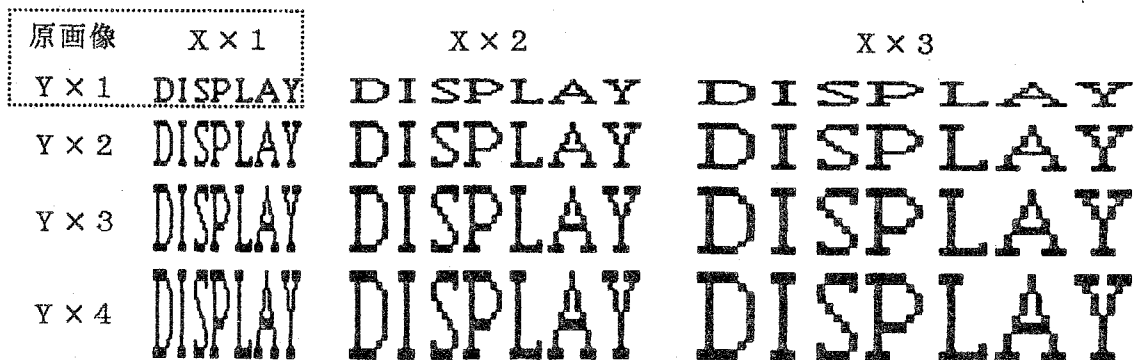


図 5.5 拡大表示

(g) 外部同期

GPUはマスタ/スレイブいずれかにプログラムできる。マスタに設定された場合、スレイブ機器に対して同期信号を出力する。スレイブの場合、マスタのGPUあるいはテレビ等の外部機器からの同期信号を受け、これらとの同期動作が可能。

(h) プログラマブル・カーソル

2つのカーソルを制御し、それらの形状、ブリンクなどをプログラマブルに設定することができる。更に、クロスヘアなどグラフィック・カーソルもサポートする。

(i) スキャン・モード

CRT制御としてノン・インタレース、インタレース、インタレース・シンク&ビデオの3種類のスキャン・モードの1つを選択可能とする。

(j) CPUへの割込み

ライトペンからのストローブ信号、描画コマンドの実行終了などCPUへの割込みを発生する機能を有する。これらはマスク可能で、システムでの選択に委ねる。

(k) DMA要求

主メモリとのデータ転送をDMAによって高速に行える機能を有する。データのほか主メモリ中のセグメントバッファ内の描画コマンド/パラメータの自動フェッチにも使用できる。

以上に示した主な機能を1つの半導体チップに集積するに当たり、性能及びチップ構成の自由度の観点から、機能に注目していくつかのプロセッサに分ける必要がある。以下の節にはその構成法を示す。

5.3 マルチプロセッサ構成法

コンピュータ・システムにおける一般的なマルチプロセッサは⁹⁾、図5.6に示すように、ローカルメモリを含む複数のプロセッサが1つのグローバルメモリを共有して並列動作を行う。この場合、各プロセッサの構成は同一であるが、ソフトウェアの違いにより異なる機能を実行する。LSI上のマルチプロセッサでは、プロセッサを機能に合わせて最適化しておくことがチップサイズ、性能などの観点から必要である。グラフィック・プロセッサの場合には、CPUあるいは主メモリ(セグメントバッファ)をオンチップのプロセッサで共有し、フレームメモリ側で各機能に応じたインターフェースを持つようにする。

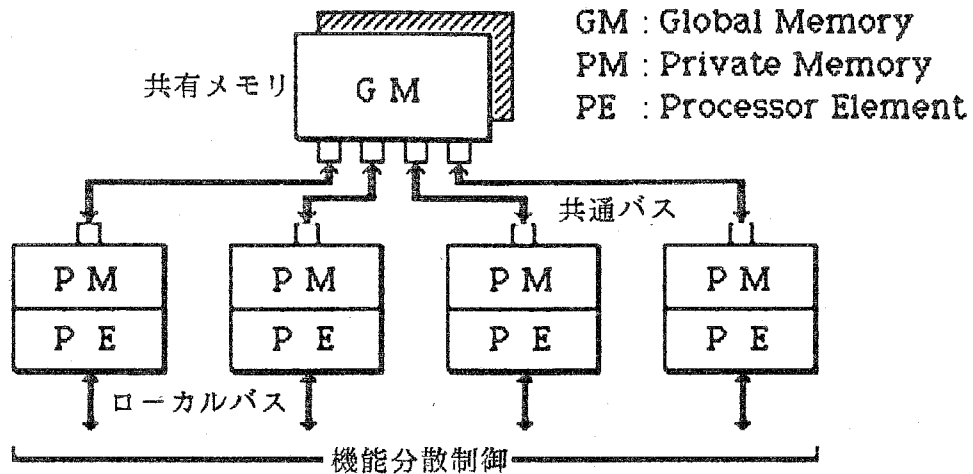


図5.6 一般的なマルチプロセッサ構成

5.3.1 機能分散構成法

前節に述べたGPUの諸機能を分析すると、GPUは図5.7に示した5つの機能モジュールから構成し得ることがわかる。

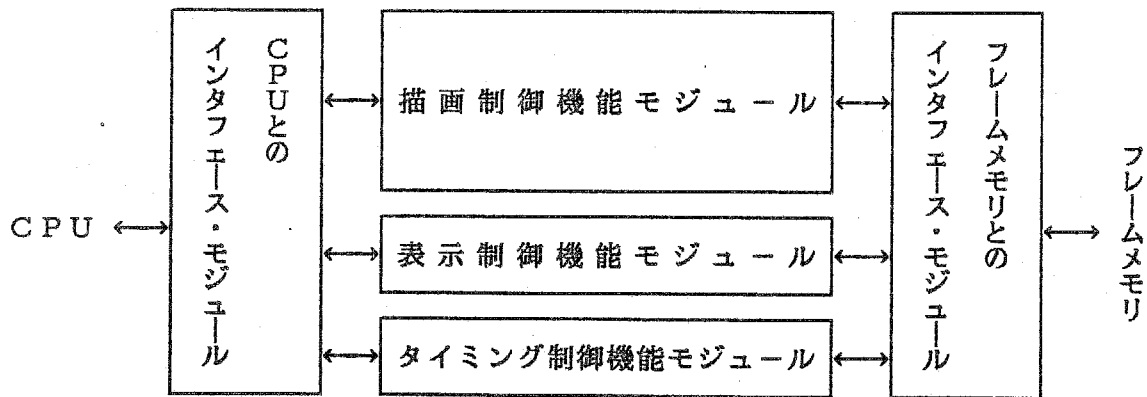


図5.7 GPUの機能分散構成

(1) 描画制御機能モジュール

図5.8に示すように、CPUから与えられる描画コマンドを解釈し、これを実行する機能モジュールである。実行過程では、描画コマンドの内容に従って描画すべき図形の1ドット毎にフレームメモリ上のアドレスを計算し、これによりフレームメモリから描画ドットを含む1ワードを読み出してカラー処理を加え、再び元の位置に戻す。以上の操作を

描画すべき図形を構成する全ドットに対して行う。描画点のアドレス計算は、図形の形状に応じた図形発生アルゴリズム¹⁰⁾をマイクロコード化し、これを逐次実行することで実現する。GPUは、既に述べた通り、CPUから与えられる描画コマンドのパラメータをX-Y座標系としているので、描画実行部は次の3つの実行部(エグゼキュータ)に機能分割すると性能向上が図れる。

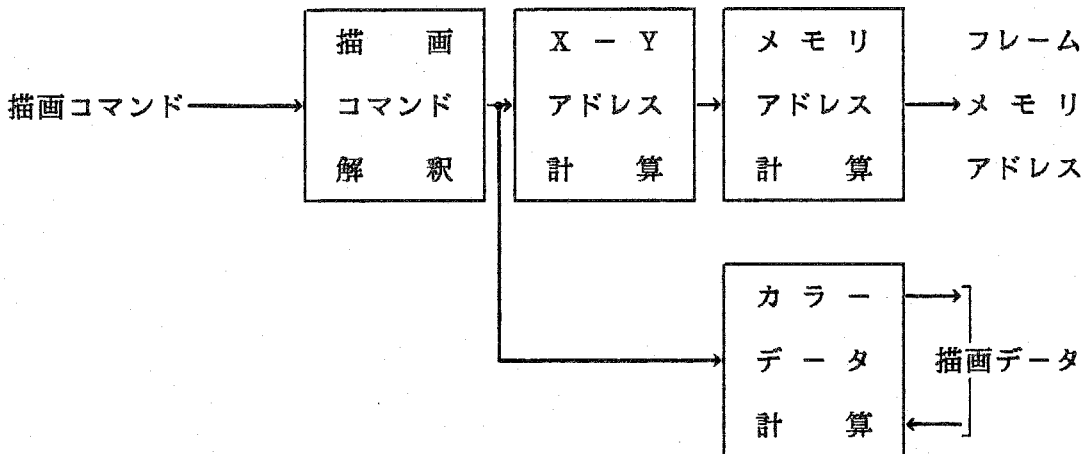


図5.8 描画制御機能モジュールの役割

(a) EX1: X-Yアドレス・エグゼキュータ

描画コマンドに続いて転送される座標パラメータを受け取り、各コマンドの処理アルゴリズムに従ってX-Y座標の計算を行う。現在の描画点の座標を示すCPレジスタを始めとして、各種レジスタや演算回路から成る。

(b) EX2: メモリアドレス・エグゼキュータ

X-Y座標に対応するフレームメモリアドレスを算出する。フレームメモリのアドレスポインタや演算回路から成る。種々のカラー・ビット数(例えば、16色では4ビット、256色では8ビットなど)に対応したアドレス計算を高速に実行する。描画コマンドの1つであるORIGINコマンドによってX-Y座標の原点に対応するフレームメモリのアドレスが定義されると、以後の描画では常にカレントポインタCPの移動に伴ってメモリアドレス・ポインタも移動する。即ち、EX1とEX2の2つのエグゼキュータの協調によって、パイプラインでX-Y座標からフレームメモリのアドレスへの高速アドレス変換が実行される。

(c) EX3: カラーデータ・エグゼキュータ

パターンRAMを参照して、フレームメモリから読み出されたデータに対し画素単位の描画演算を実行する。各種カラー・レジスタやカラー同時処理を行う専用演算回路から成る。

以上述べた点と図5.8から明らかのように、本機能モジュールはCPUと同じような構成となり、DDAの実行を中心に、そのアルゴリズムを内蔵する。そして、フレームメモリに対して描画を専門に機能分散されているため描画プロセッサと呼ぶ。描画プロセッサでは、グラフィック・コマンドを入力しこれを実行する点でCPUと同じような動作をするが、処理する対象がグラフィック・データである点異なる。このため、図5.8に示したように、X-Yアドレス計算、メモリアドレス計算、カラーデータ計算といったグラフィック専用の演算回路を持つことになる。

(2) 表示制御機能モジュール

図5.9はCRTにフレームメモリ上の図形データを表示する際の表示アドレス制御の様子を示したものである。前節に示したCRT表示制御の各項目に対して次のようなアドレス制御を行う。

(a) 画面分割

CRT表示画面での水平3分割では、第1画面(上側)、第2画面(中側)、第3画面(下側)のそれぞれのスタート・アドレス(SA_1 , SA_2 , SA_3)からの表示データ読み出し、ウィンドウ画面でのスタート・アドレス(SA_w)からの表示データの読み出しを後述するタイミング制御との連携で実行する。

(b) 画面の重ね合せ(Superimpose)

2つの表示画面を重ねて表示する場合、フレームメモリから2つのアドレスで交互にアクセスする。

(c) スムース・スクロール

N回の画面表示毎にスタート・アドレス(SA)を更新し、CRT上に表示される画面領域を変える。スクロール速度は毎画面でスタート・アドレス(SA)を変えれば($N=1$)、最も速い。Nを大きくするに従って遅くなる。

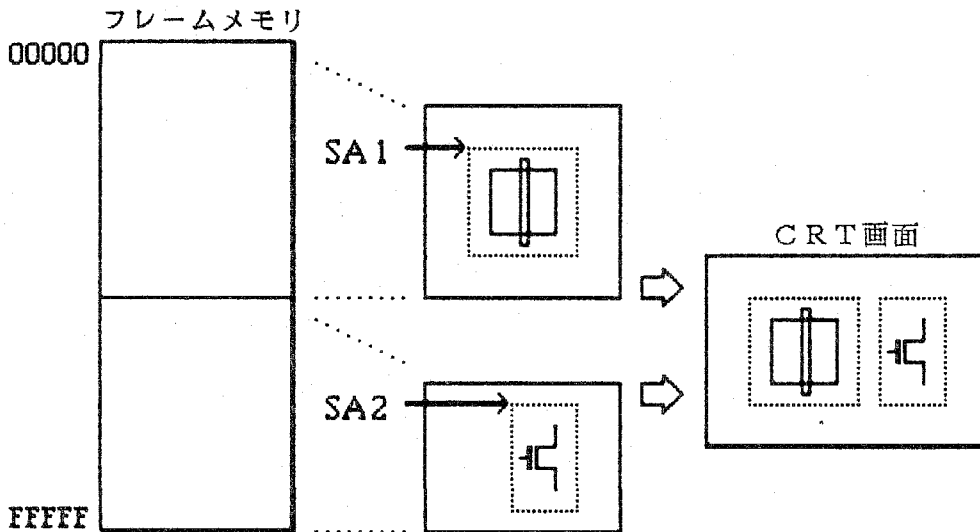
(d) 拡大表示

Y方向の拡大の場合、その方向の拡大係数(L_y)分だけ水平走査でのスタート・アドレ

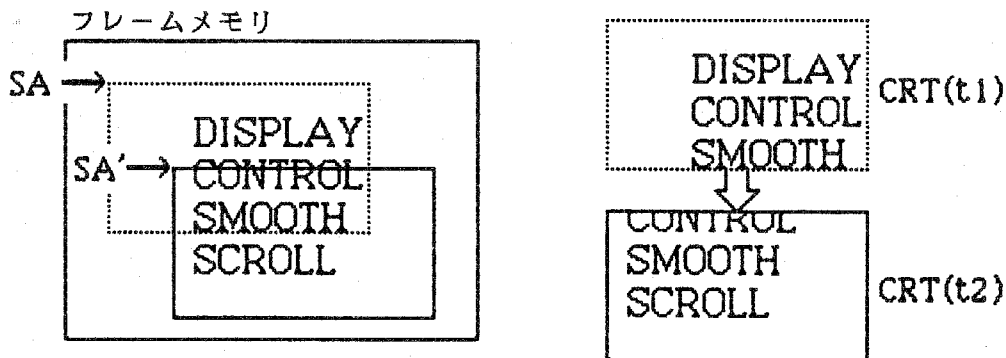
ス(SA)の更新を抑える。

(e) スキャン・モード

ノン・インタレース・モードの場合、CRT表示のスタート・アドレス(SA)はフレームメモリ上の表示領域を1回水平走査する毎にY方向へ1ドット分ずれたスタート・アドレスに変更してこの動作を繰り返す。一方、インタレース・モードの場合、スタート・アドレス(SA)はフレームメモリ上の表示領域を1回目の垂直走査では偶数番目の水平走査に相当する部分を常に指し、2回目の垂直走査では奇数番目の水平走査に相当する部分を指すように更新される。即ち、1画面の走査のスタート・アドレスと次の画面の走査のスタート・アドレスはY方向に1ドットずれたものとなる。また、各水平走査での次の水平走査のスタート・アドレスは2ドットずれたものとなる。



(a) 画面分割



(b) スムース・スクロール

図5.9 表示アドレス制御

以上に示したように、本モジュールは主にフレームメモリのアドレス管理が主体である。描画プロセッサ同様、CPUでのアドレス計算機構のような働きをし、マイクロコード化プロセッサの構造をとる。表示プロセッサと呼ぶ。

(3) タイミング制御機能モジュール

図5.10に示すように、CRTの表示領域を定義する水平・垂直同期信号、画面分割における各画面の表示スタート・タイミングを始め、各機能モジュールに対する基準タイミングや同期タイミングを発生するのが、本モジュールである。これまでの設計法では、この部分はランダム論理の典型的なものであったが、LSIのプログラマビリティと高集積化のための論理の規則性を高めるためにマイクロプログラム制御のプロセッサ構造とする。タイミング・プロセッサと呼ぶ。

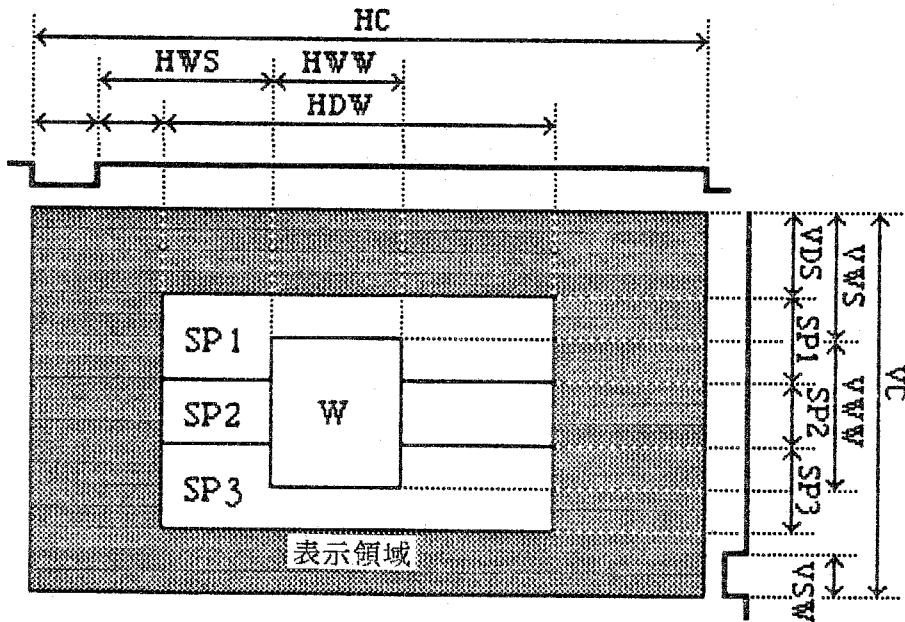


図5.10 CRTの表示領域定義

(4) CPUとのインタフェース・モジュール

CPUから描画プロセッサへの描画コマンド/パラメータのWrite, 描画, 表示, タイミング・プロセッサへの制御語のRead/Write, 主に描画プロセッサ中の状態語のRead, フレームメモリ上のデータと主メモリとの転送などのインタフェース。

(5) フレームメモリとのインタフェース・モジュール

フレームメモリへの描画データのRead/Write, 同表示データのRead, CRTへの同期信号, ライトペン・ストローク信号などのインタフェース。

以上述べたとおり，グラフィック処理専用プロセッサ(GPU)は，図5.11に示すように2レベルのオンチップ・マルチプロセッサ・アーキテクチャをとることによって各機能の並列処理が可能となる。第1のレベルは，描画，表示，タイミングの3つのマイクロコード化プロセッサで構成され，第2のレベルは，描画プロセッサに見られるように，X-Yアドレス，メモリアドレス，カラーデータの各エグゼキュータから構成される。即ち，オンチップの機能分散形マルチプロセッサ(PM: Processor Module)による並列処理と，マルチエグゼキュータ(EX: Executor)におけるパイプライン処理によってグラフィック処理の高速化が図られる。これは汎用プロセッサと周辺LSI数個分に相当し，システムの小型化にも大きく貢献することになる。

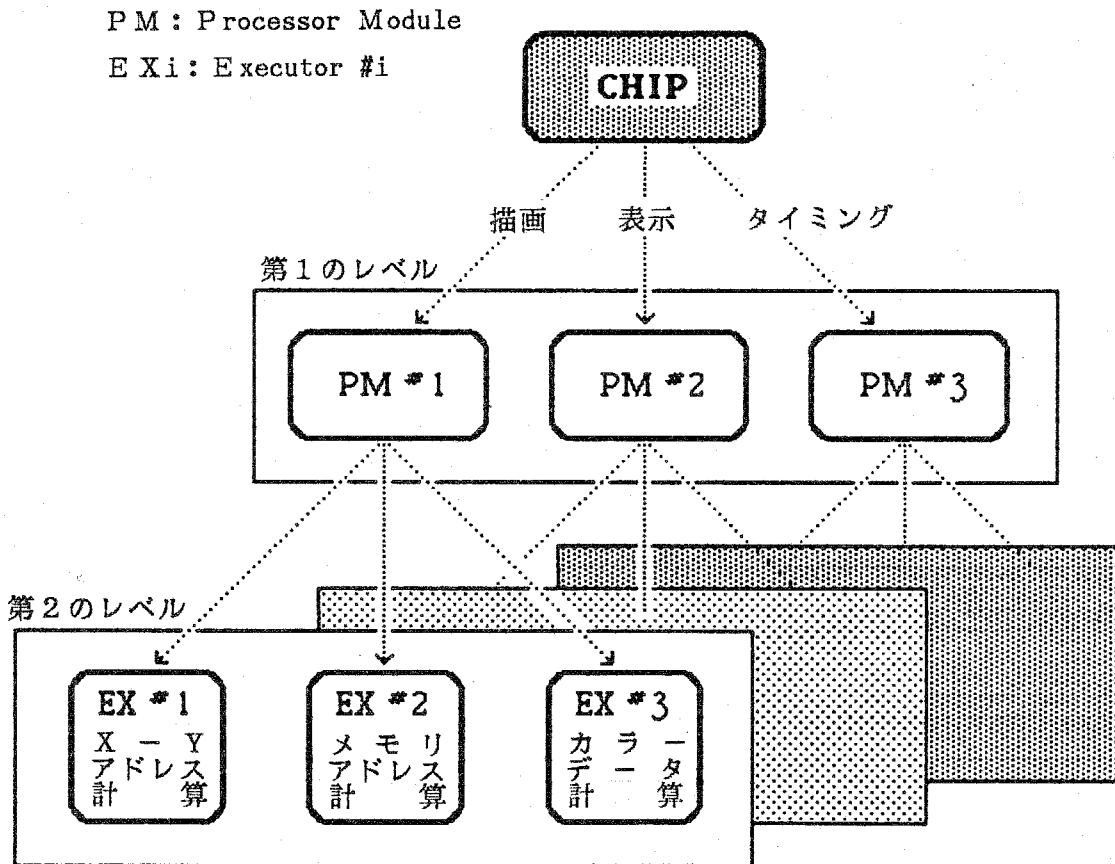


図5.11 2レベルのオンチップ・マルチプロセッサ・アーキテクチャ

5.3.2 階層論理構造

図5.11に示したように階層構成は、グラフィック処理における機能分散による高性能のみならず、設計の階層化も可能にする。各モジュール(プロセッサ)は互いの独立性が強まり、疎結合することで大規模論理の複雑さを分散し、設計工程での並列化ができ、各モジュールのプロセッサ構造化による規則性の向上と共に、設計工数低減と期間短縮を図れるものである。あらかじめモジュール間のインタフェースを定義しておけば、各モジュールを設計する際に他のモジュールの詳細を知る必要がないからである。図5.12は、図4.11に示したグラフィック処理専用プロセッサ(GPU)の2レベル・マルチプロセッサ・アーキテクチャをレイアウト設計の観点から表現したものである。チップは3階層に分け、論理設計とレイアウト設計共にTop-down(一部Bottom-up)で設計していく。従って、論理及びレイアウト設計を各階層で閉じて進めることができる。設計効率を高めるために、各階層では次の点に留意しなければならない。

(1) チップ

機能単位で分割したモジュールの組み合わせで構成し、各モジュール間は標準的なインタフェースで接続する。即ち、各モジュールの独立性を強め、モジュール間を疎結合することで大規模論理の複雑さを分散させる。一部の特殊回路についてはセル或いはマクロセルの組み合わせによって構成する。

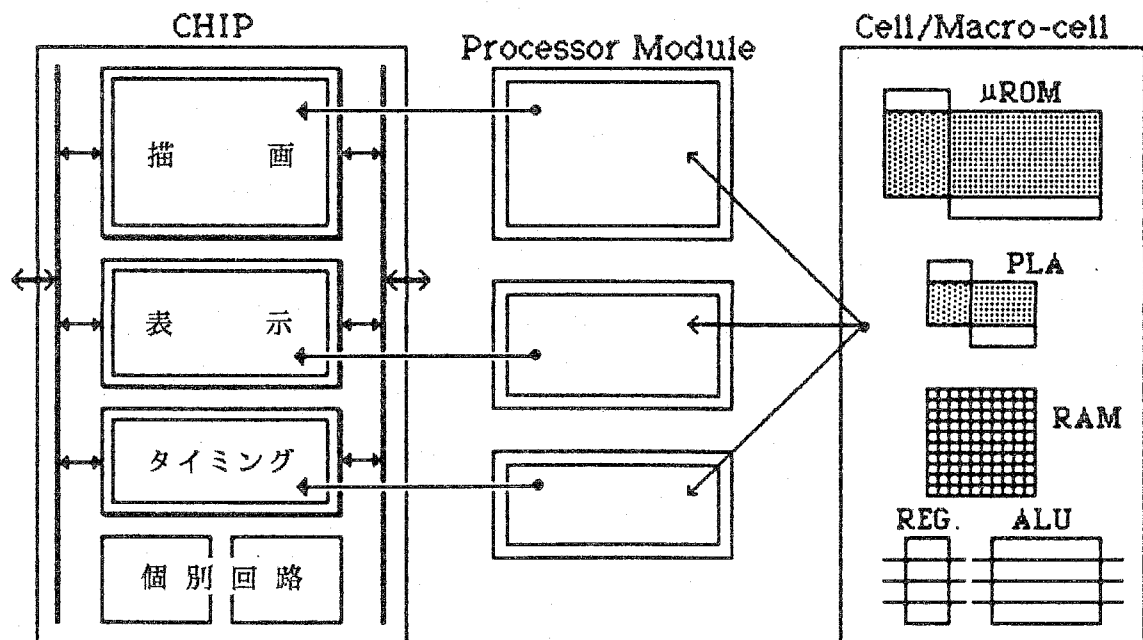


図5.12 階層論理構造

(2) モジュール

基本的にはプロセッサ構造として構成し、各モジュール間の構成要素の統一化を図る。複数に分けられた機能に対して共通的な構造を抽出し、汎用性(融通性)の高いマイクロチップ・アーキテクチャで構成する。マイクロコード化プロセッサが最有力である。モジュールの構成は規則的な配列をもつ要素を基本とし、セルまたはマクロセルの繰り返しによって設計すべきセル数の低減を図ると共に、チップ面積の極小化も合わせて追求すべきである。

(3) セルまたはマクロセル

モジュールを構成する最小単位の要素である。規則的な配列を実現するためのセルまたはマクロセルは、繰り返しの方向や他のセル(またはマクロセル)との接続が円滑にできる物理的インタフェースを持つ必要がある。ROM, RAM, PLAのように規則性の強い要素をできる限り使った論理構造が設計セル数を低減する効果的手段である。

以上に示した階層論理構造は、Standard Cell方式¹¹⁾のようなアプローチにも合った設計法であり、将来の拡張性も備えている。

5.4 L S I の構成と性能評価

(1) L S I チップ構成

グラフィック処理専用プロセッサ(GPU)のチップ写真を図5.13に示す。GPUは、描画、表示、タイミングの3つのマイクロコード化プロセッサと、CPU及びフレームメモリ、CRTとそれぞれ独立のインタフェースをもつ。CPUとのインタフェースで主メモリ上のセグメントバッファからのグラフィック・コマンドを受け取りながら、フレームメモリとのインタフェースで、フレームメモリに対して描画を実行できる。汎用の8, 16, 32ビットのマイクロプロセッサと容易にインタフェースがとれる。

このVLSIは2 μ mCMOSプロセスによって製造されており、全体で117,000のMOSトランジスタで構成されている。この中で、最も規模の大きいのが描画プロセッサである。チップサイズは8.3 \times 9.26mmで、64ピンのパッケージに収納されている。また、動作周波数は10MHz、マイクロサイクル時間は100ns、フレームメモリのアクセス最小時間は200nsである。最小のコマンド実行時間は、フレームメモリに対して1回のRead/Modify/Writeを必要とする直線描画で、400nsである。

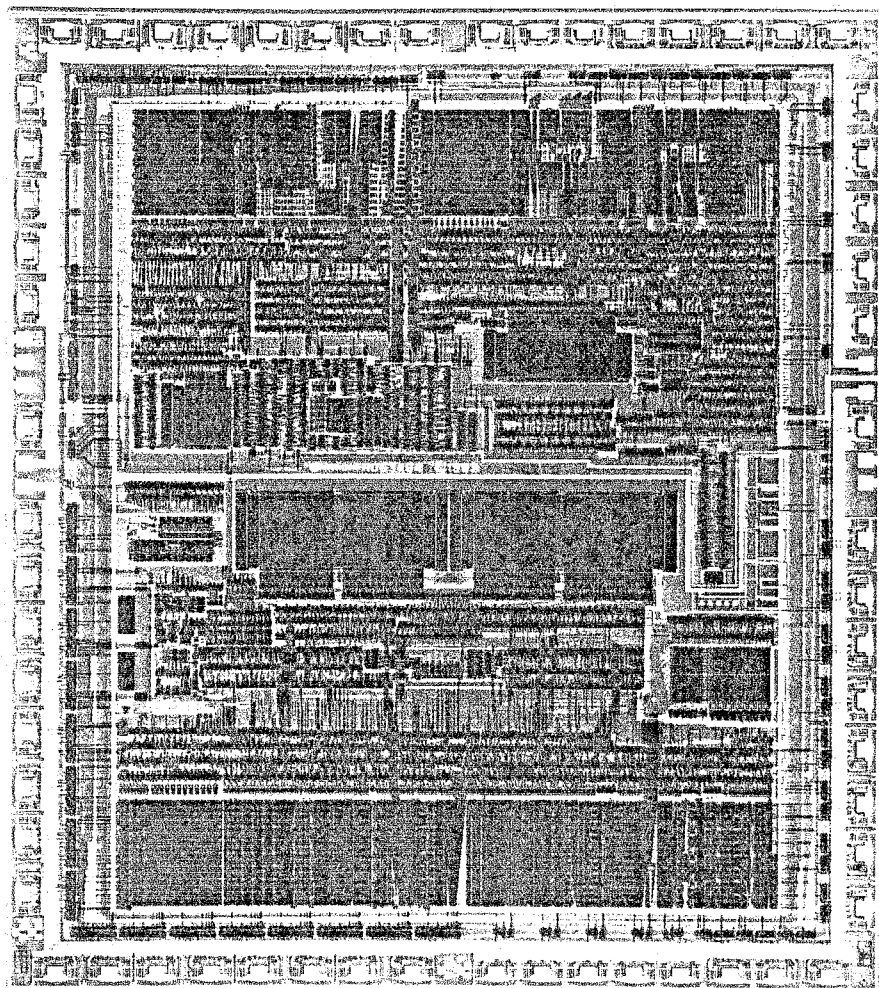


図5.13 GPUのチップ写真

(2) 描画プロセッサ構成

図5.14はグラフィック処理の中心をなす描画プロセッサの構成を示したものである。描画コマンド、パラメータ、図形データ等の入出力バッファ・メモリとなっている8段のFIFO(First-In First-Out)メモリ、線種や塗りつぶしのパターンを格納する16×16ビット(32バイト)のパターンRAM、各種描画アルゴリズムを納めた48ビット×600語のマイクロプログラムメモリ(μROM)、X-Yアドレス、メモリアドレス、カラーデータ演算用の3つの16ビットエグゼキュータから成る。描画プロセッサは、これだけで約60,000のMOSトランジスタから構成されており、汎用の16ビットマイクロプロセッサ以上の集積度をもつ^{12)~14)}。

また、3つのエグゼキュータは図5.15に示すようにパイプライン制御され、直線描画の場合、8マイクロサイクル必要な処理を半分の4マイクロサイクル(Read-Modify-

Writeの2メモリサイクル)で実行できる。

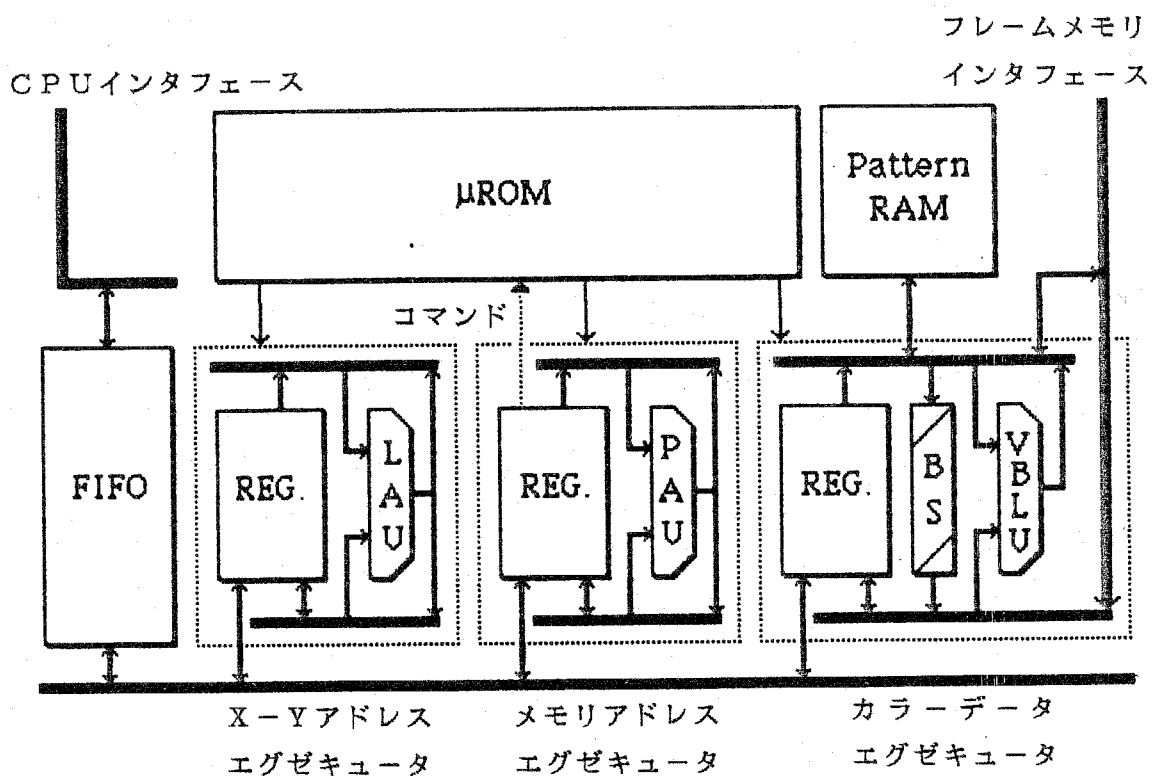


図5.14 描画プロセッサの構成

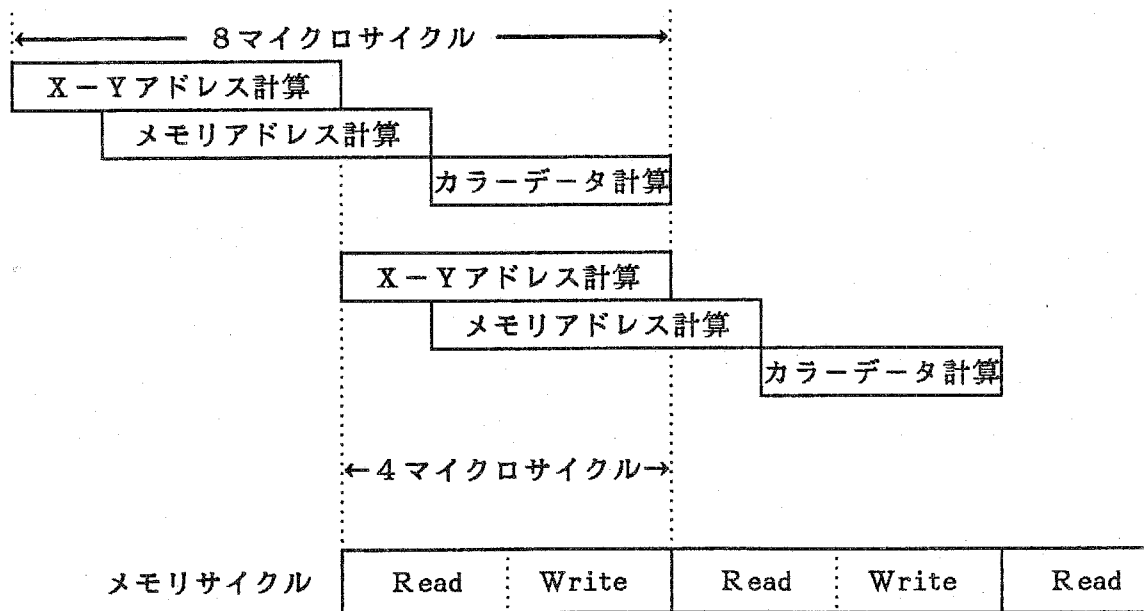


図5.15 描画プロセッサにおけるパイプライン制御

(3) 性能評価

図5.16は16色カラーの場合の直線描画時間を比較した結果を示したものである。横軸は描画ドット数、縦軸は描画時間である。下記に示した3つについて比較している。

- ①汎用16ビットマイクロプロセッサによるソフトウェア処理
- ②ドット展開処理だけをハードウェア化したLSIを用いるハードウェア/ソフトウェア混在処理
- ③GPUによる準ハードウェア処理

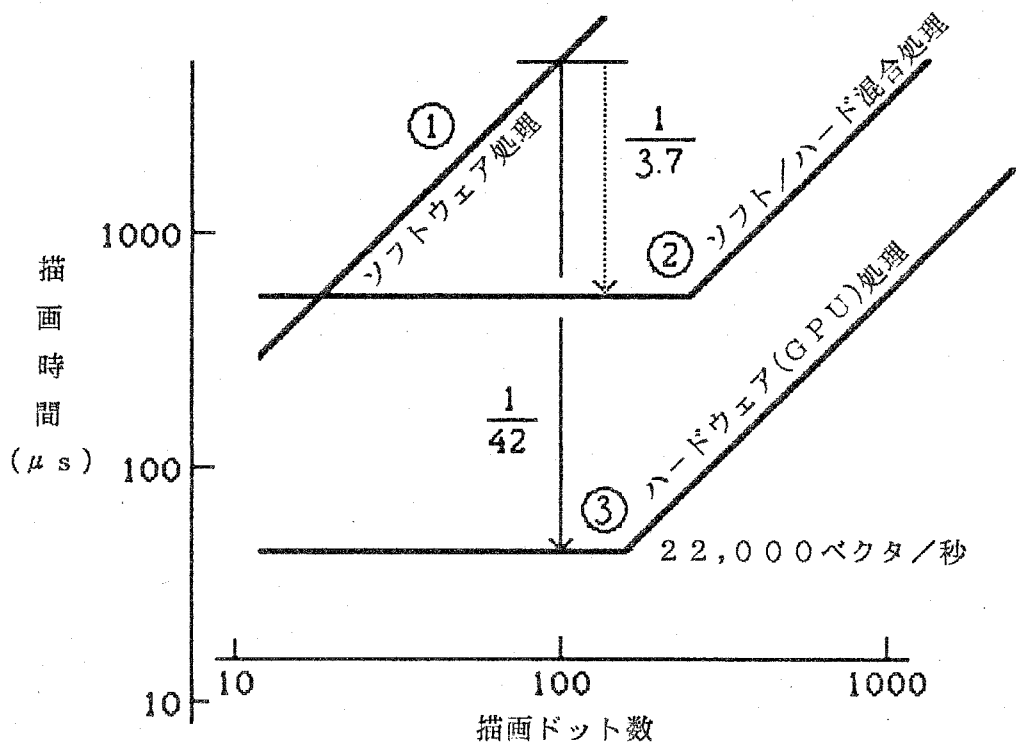


図5.16 直線描画時間比較(16色)

評価の結果、次のことがわかる。

(a) カラー・ビット数に対する描画速度

GPUは独特のカラー同時処理方式によって、1~65536色(1~16ビット/ドット)の範囲で1ドット当りの描画時間は同一であるが、①及び②の場合の描画時間はカラー・ビット数に応じて変動する。

(b) 描画ドット数に対する描画時間

①ソフトウェア処理の場合、描画ドット数が増大するに従って、描画時間は単調増加す

る。②及び③の場合、横軸に平行な部分(描画ドット数に無関係)と単調増加する部分とから成る。描画ドット数の少ない領域では、CPUの処理時間で性能が決まる。描画ドット数の多い領域では、ドット展開処理するLSIあるいはGPUの処理速度が支配的となり、描画時間はドット数に比例する。

一般に、グラフィック処理では100ドット以内の比較的短い直線(ベクタ)が多く用いられる。描画ドット数が100ドットの場合で比較すると、ドット展開だけをハードウェア処理する方式(②)は、ソフトウェア処理(①)に比べて3.7倍に性能向上する。ドット数が20~30以下の場合にはソフトウェア処理の方が早い。ドット展開のハードウェアへのコマンド転送に要する時間の方が大きいためである。これに対して、GPUはX-Y座標系に基づく高レベルの描画コマンドによってソフトウェアの負担を大幅に軽減しながらも、描画性能の向上を図っている。100ドットの直線描画の場合で、ソフトウェア処理の4.2倍に性能向上する。1秒間に22,000本の直線を発生できる能力をもつ。また、他の描画コマンドにおけるソフトウェア処理とGPU処理との描画性能比較を図5.17に示すが、極めて高速な結果が得られた。

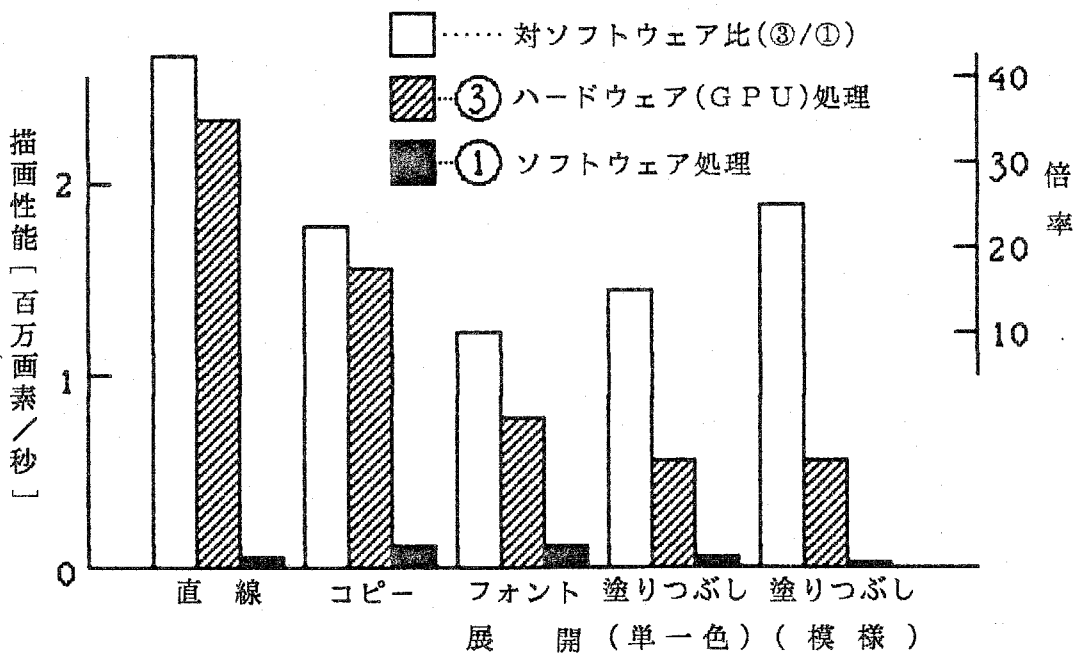


図5.17 描画コマンドの実行時間比較

5.5 まとめ

グラフィック処理専用プロセッサを例題として、マルチプロセッサ形のマイクロプロセッサの機能分散化構成法について述べた。専用プロセッサでは高性能化が最重点課題であるが、集積度に見合った高機能化も重要である。高機能化に伴う論理の大規模化は、論理設計、レイアウト設計に多大な時間を要する。そこで、本章ではこれらの課題を一挙に解決する手段として、2レベルのオンチップ・マルチプロセッサ・アーキテクチャを提案してその詳細を示した。まず、第1のレベルでは機能分散を行い、描画、表示、タイミングの3つのマイクロコード化マイクロプロセッサに分け、それらの並列処理性を高めた。第2のレベルでは、描画プロセッサの機能からX-Yアドレス、(フレーム)メモリアドレス、カラーデータ演算を行う3つのエグゼキュータに分け、それらのパイプライン処理を実現した。この結果、最高10MHz(マイクロサイクル時間100ns)動作時、描画プロセッサは直線描画で、モノクロ、カラーを問わず400ns/ドットの高速性を達成した。また、描画コマンドにはX-Y座標系のパラメータを採用したことで、グラフィック・ソフトウェアの生産性を向上させると共に、CPUの負担を軽減した。以上を総合的に評価すると、CPUのソフトウェア処理に対して1桁以上の高速性が得られることがわかった。

尚、本章で述べたGPUは、筆者らが汎用16ビットマイクロプロセッサの周辺LSIとして開発したACRTC(Advanced CRT Controller)と呼んでいるHD63484である^{1), 2), 15)~18)}。

5.6 第5章の参考文献

- (1) H.Maejima et al.: "VLSI for High Performance Graphic Control Which Utilizes Multi-Processor Architecture", Proceedings of the International Conference on Computer Design, pp.586~591 (1984-10)
- (2) K.Katsura, H.Maejima et al.: "VLSI for High-Performance Graphic Control Utilizing Multiprocessor Architecture", IEEE Transaction on Electron Device, Vol.ED-32, No.11, pp.2232~2237 (1985-11)
- (3) T.Oguchi et al.: "A Single-Chip Display Controller", ISSCC Digest of Technical Papers, Vol.24, pp170~171 (1981-2)
- (4) T.Fukushima et al.: "An Image Signal Processor", ISSCC Digest

- of Technical Papers, Vol.26, pp.258~259 (1983-2)
- (5) A.Szczepanek et al.: "An IEEE 802.5 Compatible LAN Controller with On-Chip ROM", ISSCC Digest of Technical Papers, Vol.28, pp.188~189 (1985-2)
 - (6) T.Funabashi et al.: "CMOS Hard Disk Controller with Data Buffer and Error Correction", Hitachi Review, Vol.33, No5, pp.251~254 (1984-10)
 - (7) S.Kuboki et al.: "Single-Chip Controller for Raster-Scan CRT Displays", Proceeding of the S.I.D., Vol.19, No2 (1978)
 - (8) Graphic Standards Planning Committee: "Computer Graphics, A Quarterly Report of SIGGRAPH-ACM", ACM, Vol.13, No3 (1979)
 - (9) 坂東, 前島外: "制御用マルチコンピュータシステムにおける共有メモリの設計と解析", 情報処理学会論文集, Vol.9, No9, pp.810~816 (昭53-9)
 - (10) W.M.Newman et al.: "Principles of Interactive Computer Graphics (2nd Ed.)", McGraw-Hill, Inc. (1979)
 - (11) K.Takeda et al.: "A Single Chip 80b Floating Point Processor", ISSCC Digest of Technical Papers, Vol.28, pp.16~17 (1985-2)
 - (12) S.P.Morse et al.: "The Intel 8086 Micro Processor: A 16-bit Evolution of the 8080", Computer, Vol.11, No.11, pp.18~27 (1978-5)
 - (13) M.Shima: "Two Versions of 16-bit chip span microprocessor, minicomputer needs", Electronics, Vol.51, No26, pp.81~88 (1978-6)
 - (14) E.Stritter et al.: "A Microprocessor Architecture for a Changing World: The Motorola 68000", Computer, Vol.12, No2, pp.43~52 (1979-2)
 - (15) 御法川, 前島外: "座標で描画位置を指定でき, 塗りつぶしやコピーなど豊富なコマンドを持つCRTコントローラ", 日経エレクトロニクス, No343, pp.221~254 (昭59-5)
 - (16) K.Katsura, H.Maejima et al.: "ACRTC Controll Greatly Improving Graphic Processing Performance", Proceeding of IMAC'84, pp.279~286 (1984-5)

- (17) K.Katsura, H.Maejima et al.: "Graphic Display Processor to Integrate Drawing Algorithms and Display Controls", Proceeding of Wescon'84, No.2313 (1984-11)
- (18) K.Katsura, H.Maejima et al.: "Advanced CRT Controller for Graphic Display", Hitachi Review, Vol.33, No.5, pp.247~255 (1984-10)

第6章 結 論

第6章 結論

半導体技術とりわけMOS技術を活かしたマイクロプロセッサは、いくつかの形態をとってエレクトロニクス分野を始めとして、あらゆる分野に浸透してきた。筆者らは、このような多方面に渡るマイクロプロセッサをその創成期から、開発に取り組み、その中でシリコン・チップ上に構築する最適なプロセッサ構造(マイクロチップ・アーキテクチャ)を中心に研究を進めてきた。

第1に、開発時点のLSI技術では1チップ化し得ないミニコンピュータのLSI化機種に対する複数チップ(マルチチップ形)構成法として、

- (1) 命令フェッチと命令実行を行う2種のLSIチップへの論理分割による並列処理
- (2) MOSデバイスの弱点である低速性を補う高速加算回路
- (3) 信頼性を向上するRAS(Reliability, Availability, Serviceability)機能のオンチップ化法

を取りあげ、これを実現する手段を示した。これにより、分散制御システムのキー・デバイスとなる制御用16ビットマイクロプロセッサの高性能化と高機能化を可能にした。

第2に、マイクロプロセッサ及び周辺I/O機能を同一チップ上に集積するシステムオンチップ形のマイクロコンピュータの核となるマイクロプロセッサに対するプロセッサ構成法として、

- (1) 命令解読機能をマイクロプログラムメモリ(μ ROM)中に一体化した新しいマイクロプログラム方式
- (2) マイクロプログラムメモリ(μ ROM)の圧縮法
- (3) マイクロ命令の実行をパイプライン制御し得るデータパス構成法

を取りあげ、LSI上に自由に展開し得る回路構成の特徴を活かして高速且つ小形のマイクロプロセッサの開発に成功した。その効果は、従来の設計法に対して、性能で2.54倍を保ちながら、マイクロプログラム制御部で15%程度のサイズ縮小を実現した。

第3に、目的に合わせて構成を変えられるプロセッサ構成法を命令体系に対する可変構成化に絞って取りあげ、

- (1) マイクロプログラムメモリ(μ ROM)に一体化した命令解読定義テーブル
- (2) レジスタをRAMに格納し、この中に内蔵したレジスタ構成定義テーブル

の2つのテーブルを目的に合わせてプログラムする方法で達成した。結果として、可変構造がゆえに性能を落とすことなく、小形のプロセッサの開発を可能にした。

最後に、グラフィック処理など専用プロセッサに対して高性能を実現しつつ設計工数の増大を抑える構成法として、

(1) 多くの機能を複数のプロセッサに分け、機能を分散化するオンチップ・マルチプロセッサ・アーキテクチャ

(2) 各プロセッサを実現する上での階層論理構造化設計手法

を示した。その結果、X-Y座標系を扱う高機能な描画コマンドによるグラフィック・ソフトウェアの生産性向上を図りながらもソフトウェア処理に対して1桁から2桁の性能向上を達成するグラフィック処理専用マイクロプロセッサを開発できた。

以上のように、各種目的に応じたマイクロチップ・アーキテクチャを提案し、これを具体化したところ、それぞれ好結果が得られた。ところで、コンピュータ・アーキテクチャの世界でも日進月歩の進歩が続いているが、最近になってVLSI(ハードウェア)を意識したアーキテクチャが出現してきている。マイクロチップ・アーキテクチャは、このような最新のコンピュータ技術のインパクトを受けながら次第に変貌してゆくものと考えられる。コンピュータ・システム全体のバランスを考え、コンパイラ、OSを中心としたソフトウェアと、VLSI(或いはULSI)を基本としたハードウェアの有機的な結合を考慮してゆかなければならないであろう。本論文で示したマイクロコード化マイクロプロセッサの構成法は、こうした動きに対する基本技術として位置づけられるものとする。

謝 辭

謝辞

本報告をまとめるにあたり、懇切に御指導いただいた東京工業大学工学部・情報工学科・当麻喜弘教授に心からお礼申し上げます。

本研究の推進に対しては、日立製作所日立研究所，同武蔵工場，同大みか工場など多くの事業所の多くの人々に御指導並びに御助力いただいた。特に，日立製作所研究開発部 高砂常義(元日立研究所所長)，宇都宮大学 奥田健三(元日立研究所部長)の両工学博士からは本研究の動機づけを，日立製作所日立研究所所長 川本幸雄工学博士からは本研究に対する大局的な御指導を，更に西原元久，増田郁朗両工学博士からは日頃のより具体的な御指導をいただいている。また，日立製作所日立研究所 桂晃洋，木田博之両研究員の御協力をいただいた。

以上の方々に対し，深甚の謝意を表する。