学術論文

パーソナルロボット用機能別並列計算機アーキテクチャ *ASPIRE* の RISC を用いた設計と実装

山崎信行[*1]　安西祐一郎[*2]

Design and Implementation of Function-Classified Parallel Computer Architecture for Personal Robots *ASPIRE* using RISC Processors

Nobuyuki Yamasaki[*1] *and* Yuichiro Anzai[*2]

We believe that personal robots like current personal computers will be used in an office and at home in the near future. And we have already designed function-classified parallel computer architecture for personal robots: *ASPIRE (ASynchronous, Parallel, Interrupt-based, and REsponsive architecture)*. In this paper, we design and implement the personal robot *ASPIRE-II* based on *ASPIRE* using RISC processors. Many researchers think that RISC processers are not suitable for embedded application due to the pipeline hazards, and the loads and stores of a large number of registers in case of interrupts. However we dare to apply RISC prosessers (SPARC family) to *ASPIRE* so as to have both high computing performance and good interrupt capability. Definitely, we apply register windows of SPARC architecture to *ASPIRE*. We also describe the efficiency of the architecture by evaluating *ASPIRE-II*.

Key Words: Personal Robot, Function-Classified Parallel Computer, RISC, *ASPIRE*

1. まえがき

近年，コンピュータは様々な分野で幅広く使用されており，家庭におけるパーソナルコンピュータの普及率も年々急激に上昇している．同様に，コンピュータネットワークも急速に普及しつつあり，マルチメディアの流行と共に動画像や音声をリアルタイムで扱うことも可能となってきている．さらに，バーチャルリアリティの研究によって，電子的に伝達できる情報ならば，現実のような臨場感を伴って伝達可能になりつつある．

しかしながら，コンピュータ単独では電子的な情報しか扱うことができない．我々は今後の計算機システムにおいては，従来の計算機システムが扱っていた「電子世界」だけでなく，我々が活動する「物理世界」もその処理対象に含める必要があると考えている．そこで我々は，電子世界と物理世界の両方を扱うことができるものとしてパーソナルロボット[7][14]に注目している．現在，我々はパーソナルロボットを用いることによって，電子世界と物理世界を統合し，従来の計算機科学の研究を物理世界へと拡張する試みを行っている[1]．

パーソナルロボットとは，現在のパーソナルコンピュータと同様に，個人で所有し簡単に機能の追加・変更やプログラミング等が可能な，パーソナルユースの知能ロボットのことである．用途としては，オフィスワーク支援ロボット，家事支援ロボット，アミューズメントロボット，福祉ロボット等が考えられる．

パーソナルロボットが人間と自然にインタラクションを行うためには，パーソナルロボットの機敏で正確な動作が必要である．そこで，我々はパーソナルロボット用機能別並列計算機アーキテクチャとして *ASPIRE* (*ASynchronous, Parallel, Interrupt-based and REsponsive architecture*) を提案している[15]．*ASPIRE* ではロボットを機能別モジュールに分割し，各機能別モジュールにプロセッサを配置し，イベントの伝達をすべて割り込みによって実現する．*ASPIRE* にしたがい，CISC 型プロセッサである Motorola 68000 系の Processing Unit (PU) を用いてプロトタイプのパーソナルロボット *ASPIRE-I* がすでに設計・実装されているが，その処理能力は十分とはいえない．そこで本研究では，従来までは組み込み用途には不向きであると考えられていた RISC 型プロセッサ (SPARC 系) を敢えて用い，高速演算性能とレスポンシブ性 (3.1 節参照) の両方を実現することを試みる．具体的には，SPARC のレジスタウィンドウを *ASPIRE* に応用した新しいアーキテクチャを考案する．そして，そのアーキテクチャに基づいてパーソナルロボット *ASPIRE-II* の機能別モジュールを設計・実装し，プロトタイプである *ASPIRE-I* との比較・検討を行う．そしてこの SPARC を応用した *ASPIRE* のアーキテクチャ的有効性を示す．

2. 背　景

近年，注目されているロボットアーキテクチャとして，サブサンプションアーキテクチャ[3][4]がある．サブサンプション

原稿受付　1995 年 6 月 9 日
[*1]電子技術総合研究所
[*2]慶應義塾大学大学院理工学研究科計算機科学専攻
[*1]Electrotechnical Laboratory
[*2]Department of Computer Science, Keio University

アーキテクチャでは行動型アプローチを採り，ボトムアップに学習していくので，周囲の環境への対応が速くなる．このアーキテクチャでは，外界からのセンサ情報を最優先し，既存のAIを用いたプランニングを排除することによって，反応性の高いロボットを作ることを可能にしている．逆にトップダウンに計画・行動することが困難で既存のAI技術が応用しづらいという欠点があり，パーソナルロボットのアーキテクチャには不向きであると考えられる．

サブサンプションアーキテクチャに既存の制御技術とAI技術を複合し改良したものとしてSSS [5] やATLANTIS [8] などがある．しかし，これらはナビゲーションのアーキテクチャであり，ハードウェアレベルの考慮は行われていない．

センサ情報を重視して設計された優秀な小型自律移動ロボットとして山彦 [16][17] がある．山彦は移動速度が速く，反応性も既存の自律移動ロボットと比較するとかなり良い．しかし，イベント伝達を高速に行うための機能が備わっていないので，リアクティブ性とリアルタイム性に限界があると考えられる．

我々はパーソナルロボット用機能別並列計算機アーキテクチャとして *ASPIRE*（*ASynchronous, Parallel, Interrupt-based and REsponsive architecture*）[15] を提案している．そして *ASPIRE* にしたがい，CISC型であるMotorola 68000系のプロセッサをPUに用いてパーソナルロボット *ASPIRE-I* を設計・実装している．*ASPIRE-I* はリアクティブ性とリアルタイム性を兼ね備えたレスポンシブ・システムとして設計され，雑踏のような環境でも正常に動作することができた．しかしながら，プロセッサパワーの大部分をロボット自身の制御に消費してしまうため，大規模なAI的手法（プランニング等）をオンラインで行ったり，画像処理などの計算量の大きいタスクをリアルタイムに行うことは困難であった．したがって，実用的なアプリケーションをパーソナルロボット上に構築するためには，*ASPIRE-I* と同程度の割り込み処理能力を持ち，さらに高速な演算能力を持つパーソナルロボットを設計・実装する必要がある．

3. パーソナルロボット用機能別並列計算機アーキテクチャ：*ASPIRE*

3.1 レスポンシブ・システム

パーソナルロボットでは，外部からの入力（センサ，ユーザ入力等）に対して速やかに反応し，その入力に対応する動作を行う必要がある．これらの外部入力（割り込み）は頻繁にかつ連続的に生じ得るので，それらをできるだけ短い反応時間（response time）かつ短い処理時間（processing time）で正確に処理する必要がある．また，各種デバイス（モータ，センサ等）の制御等は，ある時間制約を満たして実行する必要がある．

そこで我々はパーソナルロボットの動作における最も重要な性質として，「レスポンシブ（responsive）性」を導入する．レスポンシブ性とは「リアクティブ（reactive）性とリアルタイム（real-time）性とを兼ね備えた性質」である．そして，パーソナルロボットはレスポンシブ性を有したレスポンシブ・システム（responsive system）として設計されるべきであると我々は提案している．

ここでリアクティブ性とリアルタイム性とは，簡単には以下のような性質である．

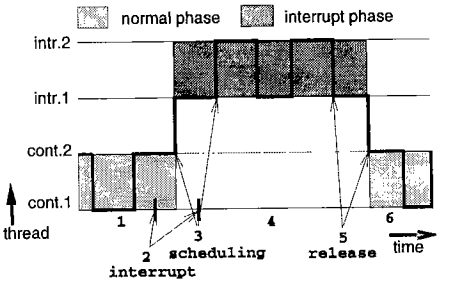


Fig. 1 Scheduling of *μ-PULSER*

リアクティブ性　外界からの刺激に対して反射的に動作を行う性質 [2][12]

リアルタイム性　システムの正確さが（計算あるいは行動等の）結果だけではなく，結果が導き出された時間にも依存する性質 [9]

レスポンシブ・システムでは，リアルタイム性が要求される低レベルなデバイス等の制御や，リアクティブ性が要求される緊急時の処理等を同時に満たすことが求められる．このレスポンシブ性はハードウェアとソフトウェア（オペレーティングシステム等）の両方を互いに考慮しながら適切に設計することによって初めて実現可能である．我々はすでにマルチスレッドベースのマルチタスク・リアクティブ・リアルタイム・オペレーティングシステム *μ-PULSER* [13] を開発している．*μ-PULSER* では，ハードウェア割り込みとソフトウェア割り込みをDirect Interrupt（DI）という概念で単なるイベントとして統一的に扱う．また，DIでは複数の割り込み処理を並行に実行することが可能となっている．従来のOSでは，ある割り込み処理中に別の割り込みがかかると，現在処理をしている割り込みレベルよりもレベルが高ければ多重割り込みとして処理され，低ければ処理を待たされてしまう．*μ-PULSER* では，これらの複数の割り込みを並行に処理できるため，パーソナルロボットのリアクティブな制御を可能としている（**Fig. 1** 参照）．

3.2 *ASPIRE* の設計方針

我々は，非人工的な環境でも自律的に正確に動作するパーソナルロボットを実現することを目標としている．「レスポンシブ性」をハードウェア的にサポートする基本的な戦略は，以下のように単純化できる．

- 発生したイベントに対応する反応時間の短縮
- 発生したイベントに対応する処理時間の短縮

これを実現するためには，我々は以下を実現すればよいと考える．

- イベントの即時的伝達
- 異種の複数イベントに対する処理の並列実行
- 同種の複数イベントに対する処理の並行実行

センサ等からイベントが発生すると，それを処理するためのモジュールに即時的にそのイベントを伝達する．異種のイベントに関しては処理するモジュールを機能分散し並列実行させることによって処理時間と反応時間の両方を短縮させる．同種の複

数イベントに関してはその処理を並行実行することによって反応時間を短縮させる．

雑踏のような実環境では，常に周囲の状況が変化していくので，「緊急（例外）処理」を迅速に行うことができなければならない．このような状況では，緊急処理が非常に頻繁に行われ，ロボットの動作は緊急処理の組み合わせのようになる．つまり，緊急処理があたかも通常処理のように行われる．したがって，緊急処理を通常処理のように扱う機構が必要である．

パーソナルロボットには「ロバスト性」も必要である．ここでいうロバスト性とは，ロボットのある一部分が故障したとしても，その故障した部分を切り離して動作し続ける性質のことである．

また，これらの性質を持たせるためには，パーソナルロボットの各モジュールやデバイスを効率よく「並列」に動作させる必要がある．ロボットの各動作は，本質的に並列動作しているので（例えば，センシングしながら移動する等），それらを逐次的に処理するよりも並列処理した方が効率がよく，システムのスループットを向上させることができる．

以上をまとめると，*ASPIRE*の設計方針は，以下の性質を実現することにある．

- レスポンシブ性
  - リアクティブ性
  - リアルタイム性
- 緊急（例外）処理の即時性
- ロバスト性
- 並列性

3.3 *ASPIRE*の設計

並列性を引き出すために，ロボットを機能別のモジュールに分割し，各モジュールごとにPUを置き，機能別並列計算機として設計する．この機能別並列計算機上に*μ-PULSER*が載り，機能分散されたタスク（スレッド）が各モジュールで並列かつ並行に実行される．また各モジュールごとにPUがあるので，各モジュールごとに単独で処理（判断）することが可能となる．その結果としてロバスト性も増大する．この機能別モジュールの分割は，各モジュールごとに必ずレスポンシブな処理を可能にするように，そのモジュールごとの計算量によって決定する．

*ASPIRE*では，機能別モジュールのひとつとしてメインモジュールを用意する．メインモジュールは，システム全体のプランニングやスケジューリングを行う．また，モジュール間の矛盾する行動を調停し，システム全体のコンシステンシを維持する．したがってシステム全体の制御は，基本的には分散制御であるが，システムとして不整合が生じた場合はメインモジュールがアービタとなるアービタ付分散制御を用いる．

モジュール間やモジュール内で何らかのイベントが生じたことを伝達するためには，必ず割り込みを用いる．割り込みは，例外処理，通常処理の区別なく，何らかのイベントが起こると必ず発生するようにする．そして，人間の神経と同様に割り込み信号線をイベントを起こすすべてのデバイス（I/O等）に張りめぐらせ，何らかのイベントが発生すると，この割り込み信号線を用いて即時的にイベントを伝達する．この機構によって，モジュール間およびモジュール内において，即時性と並列性を最大限に活用する．これらのイベントを通常処理や例外処理の区別なく*μ-PULSER*のDIによって，各モジュールごとに並列に，同じモジュール内で並行に処理することによって，レスポンシブな処理を行う．このように*ASPIRE*では基本的にポーリングをする必要がないので，*μ-PULSER*上ではイベント（DI）が起こるまでは他の複数のスレッドをタイムスライスしながら並列および並行実行することができ，各プロセッサのアイドリングを可能な限り抑えることができる．その結果，各モジュールのスループットが向上し，ロボット全体のスループットも向上する．これらの点が，*ASPIRE*の最大の特徴となっている．

すべてのイベントの伝達に割り込みを用いるので，互いに結合しているモジュール間では，双方向に割り込みをかける機構を設ける．

イベントを発生させた後，各モジュール間で何らかの通信をするために，モジュール間にコミュニケーションメモリを設ける．通信は割り込みを併用して非同期に行う．

4. *ASPIRE*のRISCを用いた設計

4.1 プロトタイプ*ASPIRE-I*の問題点

*ASPIRE*にしたがったプロトタイプのパーソナルロボット*ASPIRE-I*は，CISC型であるMotorola 68000系のプロセッサ（Motorola MC68030, Toshiba TMP68301）を用いVMEバスで結合した分散共有メモリを持つ機能別並列計算機として設計・実装を行った．このプロトタイプのロボットにより実験・評価を行った結果，割り込み処理能力は十分であり，すべてのイベントを割り込みにより伝達することによって，自律移動ロボットとして機敏に動作できることが分かった．しかしながら，プロセッサパワーの大部分をロボット自身の制御（モータの制御やセンサの制御等）に消費してしまうため，大規模なAI的手法（プランニング等）をオンラインで行うことは困難であることも分かった．

4.2 組み込み用途でのRISCの問題点

オンラインで大規模なプランニングを行ったり，リアルタイムで画像処理を可能にするためには，ロボットに搭載されているプロセッサが十分な処理速度を持つ必要がある．現在，高速なプロセッサの大部分がRISC型のものであるが，組み込み用途に用いる場合には大きな問題がいくつかあった．それらの問題のうち，特に重要なのは以下のふたつである．

- 割り込み時のパイプラインの乱れ
- 割り込み時のレジスタの退避

RISC型プロセッサは，一般にパイプラインとキャッシュで命令やデータを途切れなく処理することによって高速な処理速度を得ている．しかし，割り込みを多用する組み込み用途では，割り込み時にパイプラインが乱れたり，多くのレジスタをメモリに退避しなければならない．したがって，RISC型プロセッサは組み込み用途ではその高速性を発揮できないと考えられていたため，ロボット等の制御に用いられることはほとんどなかった．

*ASPIRE*ではすべてのイベントを割り込みによって伝達するので，割り込み処理が頻繁に行われる．さらに，*ASPIRE*上に

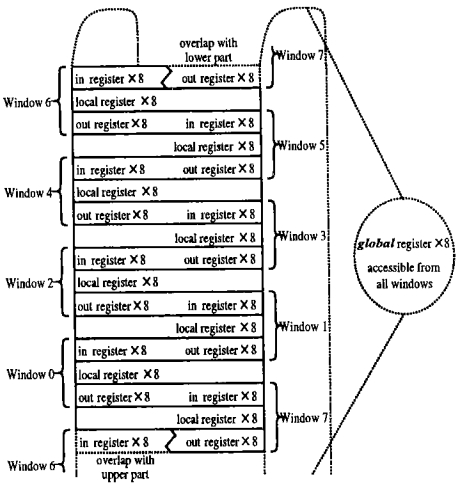


Fig. 2 Register windows (1/2)

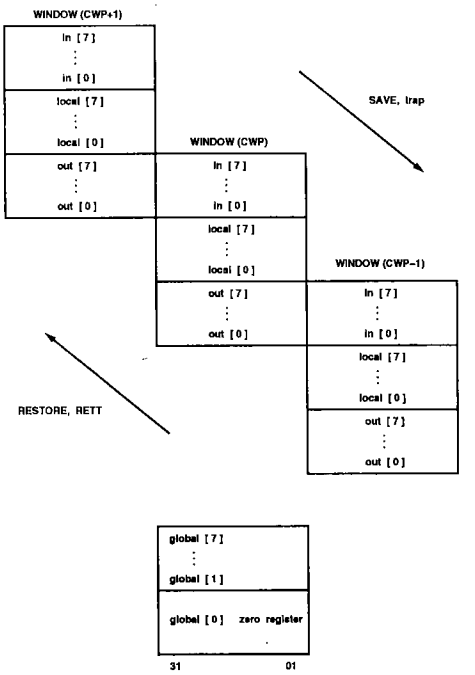


Fig. 3 Register windows (2/2)

実装されている*μ-PULSER*はスレッドベースのOSなので，スレッド間のコンテキストスイッチも頻繁に行われる．

これらの割り込み処理やコンテキストスイッチを頻繁に行うと，通常のRISC型プロセッサではレジスタの退避やパイプラインのストールに起因するオーバヘッドが大きく，十分な処理速度を得ることは困難である．特にRISC型プロセッサは退避すべきレジスタを多く持っているので，割り込み処理は大きなオーバヘッドを伴うと考えられる．

4.3 SPARCのレジスタウィンドウを用いた応用

我々は*ASPIRE*の設計にSPARC系プロセッサを用い，SPARCアーキテクチャのレジスタウィンドウを応用することによって前節で指摘した問題点を回避し，プロトタイプ(CISC型プロセッサでの設計)と比較して割り込み性能を落とすことなく，演算性能を飛躍的に向上させる試みを行う．

SPARC系プロセッサでは，プロシージャコールを高速に行うためにレジスタウィンドウ(サイクリックにオーバラップした汎用レジスタ群)が用意されている(**Fig. 2** 参照)[11]．SPARCアーキテクチャでは，同時にこれらすべてのレジスタ群にアクセスするのではなく，カレントウィンドウポインタ(CWP)で指されているカレントウィンドウのみがアクセスされる．カレントウィンドウは8個のインレジスタと8個のローカルレジスタおよび8個のアウトレジスタから構成される(**Fig. 3** 参照)．アウトレジスタは隣接するインレジスタ(CWP−1で指し示される)とオーバラップし，インレジスタは隣接するアウトレジスタ(CWP+1で指し示される)とオーバラップしている．プロシージャコールを行う際には，アウトレジスタに出力パラメータを代入してSAVE命令を実行するとCWPがデクリメントされ，カレントウィンドウが移動する．移動したカレントウィンドウのインレジスタには入力パラメータが渡されており利用できる．呼び出し元に戻るときには，インレジスタに復帰値を代入してRESTORE命令を実行することによりCWPがインクリメントされ，サブルーチンから呼び出し元に復帰する．このレジスタウィンドウを割り込み処理およびコンテキストスイッチに用いて割り込みやコンテキストスイッチ等に起因するペナルティを減少させる．

割り込みに関しては，まずレジスタウィンドウに空きウィンドウを少なくともひとつ以上確保しておく．SPARCでは割り込み(トラップ)が起きるとハードウェア的にCWPがデクリメントされ，割り込み処理用にウィンドウが与えられる(Fig. 3参照)．つまり，単にレジスタウィンドウを切り替えるという処理によって割り込み処理を行うことが可能であり，レジスタをスタックに退避するというオーバヘッドの伴う処理を基本的には行う必要がない．したがって，OSレベルでレジスタウィンドウのスケジューリングを適切に行うことにより，頻繁な割り込み処理を柔軟かつ高速に行うことを期待できる．

同様にコンテキストスイッチに関しても，レジスタウィンドウを切り替えるという処理によってオーバヘッドを最小にするように行うことが可能である．*μ-PULSER*はスレッドベースのリアクティブ・リアルタイムOSでありコンテキストスイッチが頻繁に行われるので，レジスタウィンドウの使用方法が非常に重要となる．

4.4 レジスタウィンドウのスケジューリングポリシ

汎用のマルチタスクシステム(UNIX等)では，そのシステムのスループットはプロシージャコール，コンテキストスイッチおよびユーザ・カーネルモード遷移に密接に依存すると考えられる．全体のスループットを向上させるには，コンテキストスイッチおよびプロシージャコールによって生じるオーバヘッドをできるだけ少なくすればよい．競合するプロセス間で共用されるレジスタウィンドウが多いほど，ウィンドウオーバフロートラップとアンダーフロートラップが起きにくくなるため，総プロシージャコール時間が減少する．逆に，コンテキストスイッチ時にセーブされるレジスタウィンドウの平均数が増えるので総コンテキストスイッチ時間は増大する．OSはこれらふ

たつのオーバヘッドの合計をできるだけ少なくするようにバランスをとる必要がある．

しかしながら，組み込み用途やリアルタイムシステムでは，以下の要素が総オーバヘッドより重要になると考えられる．

- 平均コンテキストスイッチ時間を最小にする
- 最悪コンテキストスイッチ時間を最小にする
- コンテキストスイッチ時間を一定にする
- 平均プロシージャコール時間を最小にする
- 最悪プロシージャコール時間を最小にする
- プロシージャコール時間を一定にする

これらを実現するために，我々は次のような方法を採る．

従来のOS（UNIX等）では，ある量子時間においてひとつのタスクがレジスタウィンドウのすべてを単純に線形に利用していた．これに対して*ASPIRE-II*＋*μ-PULSER*では，レジスタウィンドウを複数のウィンドウのグループに分割し，各グループのレジスタウィンドウを実行中のスレッドに割り当てるようにする．コンテキストスイッチ時は，コンテキストを移すスレッドのスタックポインタをロードし，そのスレッドのレジスタウィンドウに対してCWPをセットする．このとき，コンテキストスイッチ中にウィンドウレジスタの内容をロード・ストアする必要がないので，高速にコンテキストスイッチを行うことが可能となる．また，このスレッドのスケジューリング時に，必ずハードウェア割り込み用にレジスタウィンドウを最低ひとつ以上確保するようにする．このようにすることで，割り込み応答時間を最小にすることが可能となる．

ここで，SPARCでは専用レジスタとしてウィンドウ無効マスクレジスタ（WIM）が定義されており，このレジスタには各レジスタウィンドウに対するマスクビットを設定できる．CWPをインクリメントもしくはデクリメントする動作をするときにWIMビットが1であるウィンドウをCWPが指す場合，ウィンドウオーバフロートラップもしくはウィンドウアンダーフロートラップが発生する．この読み書き可能なWIMとCWPによって，オペレーティングシステムは任意のスケジューリングポリシでレジスタウィンドウのスケジューリングが可能となる．*μ-PULSER*では，これらの専用レジスタを用いて，上記のようなレジスタウィンドウのスケジューリングを行う．

以上の戦略にしたがい，SPARCアーキテクチャを*ASPIRE*に応用することによってレスポンシブ性を維持しながら，スループットを向上させる試みを行う．

5. *ASPIRE*のRISCを用いた実装

5.1 モジュール間結合

モジュールの分割は*ASPIRE*にしたがい，メインモジュール，モータ制御モジュール，センサ制御モジュール等の機能別モジュールに分割する．

これらのモジュール間結合としては，リンク結合，バス結合などが考えられる．今回の実装では，*ASPIRE-I*との互換性も考慮しモジュール間結合として汎用バスであるVME busを用いる．また，データ転送量の多いモジュール間では，VME bus以外に専用のローカルバスを設けてデータ転送を高速に行うことができるように設計する．

5.2 プロセッシングユニット

本研究では，2種類のプロセッサを用いてPUを実装する．メインモジュールと画像処理モジュール等の演算性能重視型のモジュールのPUにはTI製μ-SPARC（TMX390S10GF）を用い，それ以外の機能別モジュールのPUにはFUJITSU製SPARClite（MB86931）を用いる．

SPARCliteは，SPARC V.7に上位互換性を持つコアにオンチップキャッシュ，割り込みコントローラ，タイマ等を集積したプロセッサである．SPARCliteは割り込みコントローラを内蔵しハードウェア的に割り込みを高速に行う工夫がされており，組み込み用途やリアルタイムシステムを意識した構造になっている．したがって，高速な割り込み性能が期待できるので，機能別モジュールに採用する．しかしながら，このチップはMMUやFPUを内蔵していないため，高速な演算能力およびメモリ管理能力を必要とするメインモジュールには適当ではない．

μ-SPARCは，SPARC V.8のコアにオンチップキャッシュ，MMU，FPU，SBusコントローラ等を集積したプロセッサである．μ-SPARCは元々Sunのワークステーション用に開発されたチップであり，バスインタフェースに直接SBusを採用している．したがって割り込みはSBusコントローラを介してプロセッサコアに伝達されるので割り込み処理能力はSPARCliteにかなり劣るが，演算能力やメモリ管理能力等が優れており，メインモジュールや画像処理モジュール等の演算性能重視型のモジュールに採用する．

このように，このふたつのプロセッサはそれぞれ長所と短所を持っており，どちらか一方のみを用いて設計・実装するとシステムのレスポンシブ性を維持することは困難である．*ASPIRE*にしたがい，このふたつのプロセッサを組み合わせて設計・実装することによって互いの短所を補い，前述の高速なコンテキストスイッチを可能にするレジスタウィンドウのスケジューリングを行うことによって，システム全体としてレスポンシブ性を維持することを実現する．

5.3 機能別モジュール

本研究では，以下のような機能別モジュールを実装した（**Fig. 4**参照）．

- メインモジュール（μ-SPARC）：イーサネット，SCSI-2, Keyboard IF, Mouse IF, SIO, PIO, SBus IF
- モータモジュール（SPARClite）：DCモータ（PWM制御），SIO
- センサモジュール（SPARClite）：超音波センサ，赤外線センサ，タッチセンサ，光ファイバジャイロ，SIO
- *Chaser*モジュール（SPARClite）[10]: 音源方向定位センサ，熱源方向定位センサ，SIO
- 汎用I/Oモジュール（SPARClite）：Local bus, SIO
- 画像処理モジュール（μ-SPARC）：640 × 480, 24 bit color, イーサネット，SCSI-2, Keyboard IF, Mouse IF, SIO, PIO
- 音声認識・合成モジュール（SPARClite）：MN1901611BAT1, VCS11A-CIF, SIO

5.4 ブロックロボット

*ASPIRE*では，ユーザは自分が用いたいモジュール群をロボットに挿せば自分の好みの構成のパーソナルロボットを実現

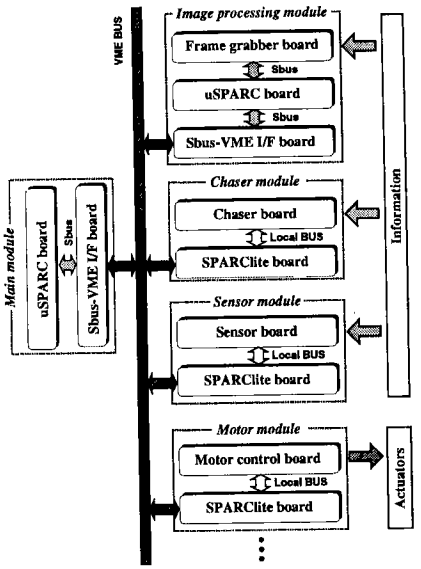


Fig. 4 Function-classified modules

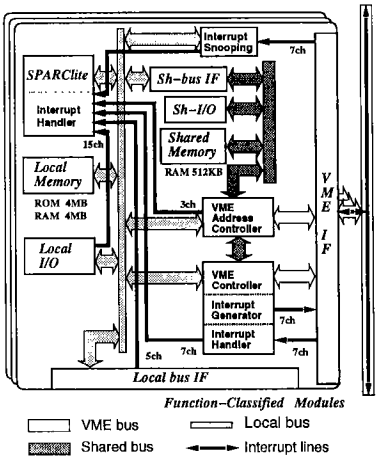


Fig. 5 Diagram of SPARClite modules

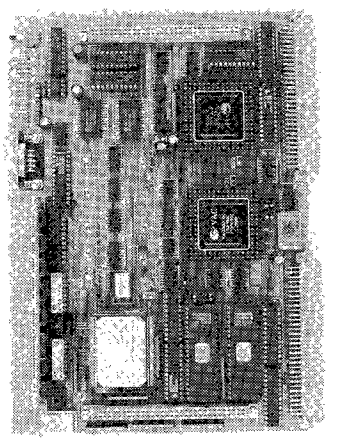


Fig. 6 SPARClite board

できるように設計を行う．つまり，各種モジュールをブロックのように組み合わせて自分専用のパーソナルロボット（ブロックロボット）を構築できるように設計を行う．

ASPIRE はモジュール自体も容易に追加できるように拡張性を考慮して設計を行う．現在の設計では，すべてのモジュールに VME bus のインタフェース部は必ず必要である．しかし，この VME bus のインタフェース部は，基板面積のかなりの部分を占めるので，他の部品の配置を圧迫してしまう．また，新しい機能を実現するごとに新たにひとつずつモジュールを設計・実装していたのではコストが高くなってしまう．そこで，VME bus インタフェースとローカル I/O 用および共有 I/O 用の専用バスインタフェースを持ったマザーボードを作っておけば，研究者が新しい機能を持ったモジュールを設計したい場合，その専用バスインタフェースを持ったドータボードを作るだけで，新しい機能別モジュールを作成することが可能となる．SPARClite ボードでは，その専用バスインタフェースは我々が独自に設計したローカルバスインタフェース（**Fig. 5, 6** 参照）を持ち，μ-SPARC ボードでは SBus（**Fig. 8, 10** 参照）を持つ．この設計の特徴は，以下のようになる．

- 新しいモジュールを作成するのが容易になる
- 新しいモジュールを作成する際のコストが下がる
- I/O 専用に広い基板面積をとることができる

したがって *ASPIRE-II* 上では，

- VME bus
- SBus
- Local bus

のバスインタフェースを持ったボードが利用可能であり．拡張性と柔軟性を有している．VME bus および SBus を用いた市販のボードを利用することも可能である．

5.4.1 SPARClite を用いたモジュール

SPARClite ボードは大部分の機能別モジュールのマザーボードとなり，VME bus IF と Local bus IF を持つ（Fig. 5 参照）．Fig. 5 より，*ASPIRE* にしたがいすべての I/O が割り込み線で接続されていることが分かる．

モータモジュールやセンサモジュール等は SPARClite ボード上のドータボードとして設計・実装されている．例えば，SPARClite ボード（Fig. 6）にモータ用ドータボードをアドオンするとモータモジュール（**Fig. 7**）になる．

5.4.2 μ-SPARC を用いたモジュール

μ-SPARC ボードはメインモジュールや画像処理モジュール等のマザーボードとなり，VME bus IF と SBus IF を持つ（Fig. 8 参照）．

実際には SBus-VME IF ボード（**Fig. 9** 参照）にふたつの SBus IF を持つ μ-SPARC ボードを挿すことによって，メインモジュール（Fig. 10 参照）となる．また，画像処理モジュールは，残りひとつの SBus IF にフレームグラバボードを挿すことによって実現する．

5.5 コミュニケーションメモリ

ASPIRE では，モジュール間通信をするためにモジュールとモジュールの間にコミュニケーションメモリを配置する．

バス結合型の並列計算機では，一般に各モジュールが共有メモリや共有 I/O に頻繁にアクセスすると共有バスがボトルネッ

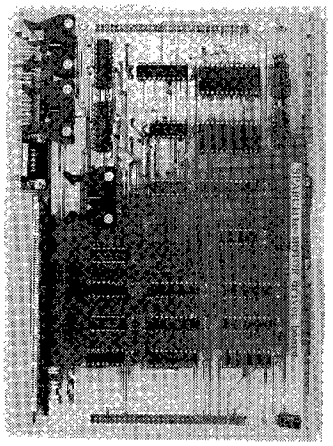


Fig. 7 Motor module (SPARClite)

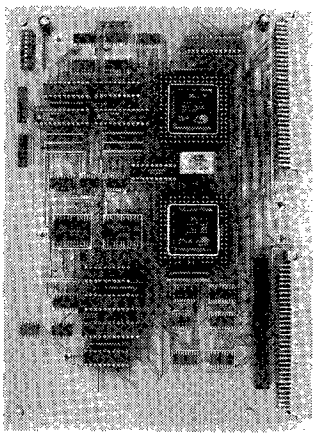


Fig. 9 SBus–VME interface board

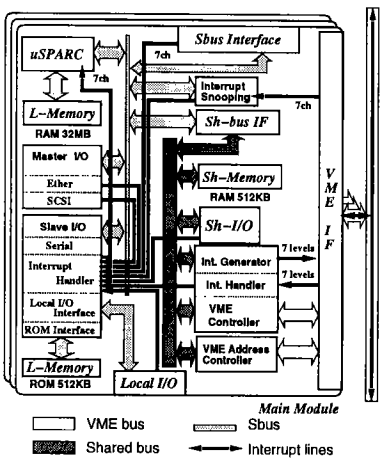


Fig. 8 Diagram of μ-SPARC modules

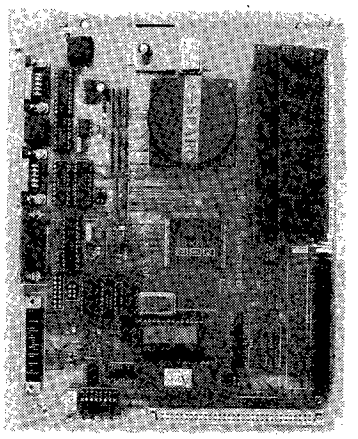


Fig. 10 Main module (μ-SPARC)

クとなりシステム全体のスループットが低下する．*ASPIRE*ではこのボトルネックを避けるために分散共有メモリを採用している（Fig. 5, 8 参照）．モジュール上の分散共有メモリをアクセスするときは共有バスを介さないので，ローカルな分散共有メモリに頻繁にアクセスをしても共有バスには負荷が加わらない．

例えばセンサモジュールでは新しいセンサデータが得られるごとにセンサ情報をモジュール上の分散共有メモリに書き込み，他モジュールからの参照を可能にしている．また，各モジュール間の通信は，自モジュール上あるいは通信先モジュール上の分散共有メモリ上に情報を書き込み，割り込みを用いて相手モジュールに知らせることによって行う．この機構によって，メイルボックスアーキテクチャ[6]をハードウェア的にサポートし，通信にレスポンシブ性を持たせることが可能となる．

また，これらの分散共有メモリを使用して各モジュール間で平等にデータ通信等を行うために，すべてのモジュールが平等にマスタになれるように設計を行う．

5.6 割り込み

*ASPIRE*にしたがい，すべてのI/Oに割り込み信号線を張り巡らせる（Fig. 5, 8 参照）．

VME busの割り込みは，7レベルの優先度付き割り込みになっている．この7レベルの割り込みをプログラマブルに変更でき，かつモジュール間で自由にかけたりかけられたりすることができるように設計を行う．同様に，VME busに対して7レベルの優先度付き割り込みをソフトウェア的にかける機構を設ける．これらはCYPRESS製のインテリジェントコントローラ VIC068A, VAC068Aによって制御を行う．

*ASPIRE*で提供するこれらの柔軟なハードウェア割り込み機構とオペレーティングシステム *μ-PULSER* でのDI機構によって各種のイベントをレスポンシブに処理することを可能にしている．

イベントが起こる可能性のあるものすべてに徹底的に割り込み線を張り巡らせている結果，イベント伝達の即時性およびプロセッサのアイドリングを可能な限り低く抑えることを実現する．

5.7 インタラプト・スヌーピング

他モジュール間のVME割り込みを監視するために，VME割り込みをスヌーピングする機構を設ける．μ-SPARCで実装するモジュール，SPARCliteで実装するモジュール共に，常に最新のVME間割り込みをハードウェア的に保持するように設計を行う（Fig. 5, 8 参照）．このインタラプト・スヌーピング

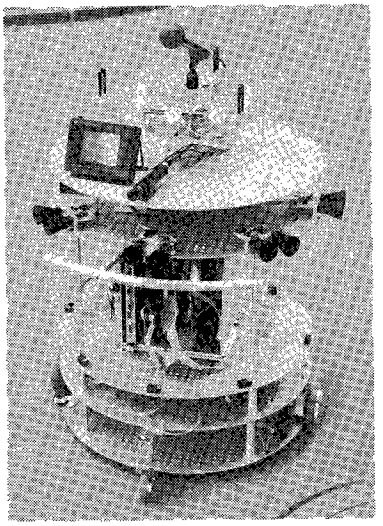


Fig. 11 Personal robot *ASPIRE-II*

機構によって，他モジュール間の協調動作を知ることが可能となり，モジュールごとの自律性および並列性を増すことが可能となる．その結果，モジュールごとのプランニングが容易になり，さらにシステム全体の分散制御の確実性を増すことも可能となる．

5.8 筐体

今回実装したパーソナルロボット *ASPIRE-II* の筐体を含めた概観を **Fig. 11** に示す．*ASPIRE-II* は，左右に駆動輪を持ち前後に補助輪を持つ構成をとっている．

6. 評価および考察

ASPIRE-II のハードウェアの基本性能を測定し評価を行う．そして，今回設計・実装した *ASPIRE-II* とプロトタイプの *ASPIRE-I* を比較・検討する．

6.1 演算性能

ここでは，演算性能の評価・考察を行う．プロトタイプの *ASPIRE-I* の各モジュールと今回実装した *ASPIRE-II* の各モジュールの演算性能の比較を **Table 1** に示す．TMP68301 を用いたモジュールと SPARClite を用いたモジュールには標準では FPU がないので，演算性能は Dhryston VAX MIPS による整数演算性能によって比較・評価を行った（各 10 回の平均値）．この際，コンパイラには同じバージョンの gcc を用いた．

Table 1 より，*ASPIRE-I* と *ASPIRE-II* で使用されている機能別モジュール同士で比較すると，

SPARClite / TMP68301 = 15.6 倍

となり，整数演算性能が約 16 倍向上したことが分かる．同様に，メインモジュール同士で比較すると，

μ-SPARC / MC68030 = 7.9 倍

となり，整数演算性能が約 8 倍向上したことが分かる．

このように，*ASPIRE-I* での CISC 型プロセッサ（68000 系）による設計・実装と比較すると *ASPIRE-II* での RISC 型プロセッサ（SPARC 系）による設計・実装の方が飛躍的に演算性能が向上したことが分かる．これによって，オンラインでの大規模なプランニングやリアルタイム画像処理が可能になると考えられる．また，モータモジュールやセンサモジュールに関してはレスポンシブな処理を行ったとしてもプロセッサパワーが余る可能性があり，余ったプロセッサパワーは他のタスクに割り当てることが可能になる．これによって，現在の *μ-PULSER* ではサポートしていないモジュール間でのタスク（スレッド）のマイグレーションが実現可能になり，ロボットシステム全体のスループットをさらに改善できると考えられる．

Table 1 Dhryston VAX MIPS

Processor	MIPS
TMP68301 (16 [MHz])	3.0
MC68030 (25 [MHz])	7.2
SPARClite (40 [MHz])	46.7
μ-SPARC (50 [MHz])	57.2

Table 2 Interrupt time (μsec)

Processor	Interrupt time
TMP68301 (16 [MHz])	24.01
MC68030 (25 [MHz])	16.47
SPARClite (40 [MHz])	12.20
μ-SPARC (50 [MHz])	50.66

6.2 割り込み

ここでは，割り込み処理の評価・考察を行う．割り込み処理が高速に行えることによって初めてリアクティブに緊急停止を行ったりリアルタイムにセンサ情報を得ることが可能となり，レスポンシブ性を獲得できる．

各モジュールの外部割り込みポートに接触して割り込みを起こし，PU がローカル割り込み処理に要する時間を測定した．測定には，デジタルストレージオシロスコープを用い，外部割り込みポートが接触された瞬間から，その外部割り込みがアクノリッジされる瞬間までの時間間隔を測定した（**Table 2** 参照）．

Table 2 より，*ASPIRE-I* と *ASPIRE-II* で使用されている機能別モジュール同士を逆数（1/t）で比較すると，

SPARClite / TMP68301 = 1.97 倍

となり，約 2 倍割り込み処理が高速になったことが分かる．同様にメインモジュール同士で比較すると，

μ-SPARC / MC68030 = 0.33 倍

となり，割り込み性能は約 1/3 になったことが分かる．

機能別モジュールの割り込み性能に関しては，RISC 型プロセッサ（SPARClite）による設計・実装の方が，CISC 型プロセッサ（TMP68301）による設計・実装に比較して悪くなるのではなく，反対に約 2 倍割り込み性能が良くなった．これは，SPARClite が割り込みコントローラを内蔵し組み込み用途を意識して設計されていること，およびプロセッサのクロックが 2 倍以上速いことに起因すると考えられ，*ASPIRE-I* 以上の割り込み処理性能を得ることができた．

メインモジュールの割り込み性能に関しては，RISC 型プロセッサ（μ-SPARC）による設計・実装は CISC 型プロセッサ（MC68030）に比較して約 1/3 に低下した．これは，μ-SPARC は本来 WS 用のチップであること，μ-SPARC のバスインタ

フェースがSBusであることなどが考えられる．しかしながら，μ-SPARCを用いるメインモジュールや画像処理モジュール等はそれぞれプランニングや画像処理といった演算が主であり，割り込みは他の機能別モジュールに比較すると非常に低い割合でしかかからない．したがって，演算性能と割り込み性能のトレードオフを考えると，十分有効であると考えられる．

7. むすび

本論文では，レスポンシブ・システムとして提案されているパーソナルロボット用機能別並列計算機アーキテクチャ*ASPIRE*（*ASynchronous, Parallel, Interrupt-based and REsponsive*）に基づき，従来までは組み込み用途等には不向きであると考えられていたRISC型プロセッサ（SPARC系）を用いてパーソナルロボット*ASPIRE-II*を設計・実装した．*ASPIRE-II*は，SPARCアーキテクチャを応用し，*ASPIRE*にしたがいすべてのI/Oに割り込み線を張り，VME busで結合し，分散共有メモリを持つブロック型の機能別並列計算機である．

RISC型プロセッサの高速な演算性能を活かしながら，SPARCアーキテクチャのレジスタウィンドウの新規なスケジューリングを行い，*ASPIRE*にしたがい割り込み処理を重視し，性質の異なる2種類のSPARC系プロセッサを用いて互いの短所を補うように設計・実装を行った．その結果，各モジュールの整数演算性能は約8〜16倍，割り込み性能でも約0.3〜2倍となり，システム全体としてレスポンシブ性を維持しながら高速演算性能を獲得することができた．これによって，パーソナルロボット上での大規模なプランニングや画像処理が可能になる．従来のロボットのように，大規模な演算を行うときに基地局のワークステーション等に頼る必要がなくなったため，真の意味での自立が達成できた．また，*ASPIRE-II*はバックエンドがない純粋な自立ロボットとしては，現時点で世界で最も高速な演算性能を有しているロボットのひとつであると考えられる．

この，*ASPIRE*にSPARCアーキテクチャを応用してレスポンシブ・システムを構築するアーキテクチャは，従来のアーキテクチャよりも優れていると考える．

参考文献

[1] Yuichiro Anzai: "Towards a New Paradigm of Human-Robot-Computer Interaction," Proceedings of IEEE International Workshop on Robot and Human Communication, pp.11–17, 1992.

[2] A. Benveniste, P. L. Guernic, Y. Sorel and M. Sorine: "A Denotational Theory of Synchronous Reactive Systems," Information and Computation, vol.99, pp.192–230, 1992.

[3] Rodney A. Brooks: "A Robust Layered Control System For A Mobile Robot," IEEE Journal of Robotics and Automation, RA–2, no.1, pp.14–23, 1986.

[4] Rodney A. Brooks and Anita M. Flynn: "FAST, CHEAP AND OUT OF CONTROL: A ROBOT INVASION OF THE SOLAR SYSTEM," The British Interplanetary Society, vol.42, pp.478–485, 1989.

[5] Jonathan H. Connell: "SSS: A Hybrid Architecture Applied to Robot Navigation," Proceedings of IEEE International Conference on Robotics and Automation, pp.2719–2724, 1992.

[6] A. Faro and O. Mirabella and L. Vita: "A Multi-microcomputer-based Structure for Computer Networking," IEEE Micro, vol.5, no.2, pp.53–66, 1985.

[7] J. Engelberger: ROBOTICS in Service, "Kőgan Page," p.218, 1989.

[8] David P. Miller and Erann Gat: "Exploiting Known Topologies to Navigate with Low-Computation Sensing," SPIE Sensor Fusion III: 3-D Perception and Recognition, vol.1383, pp.425–435, 1990.

[9] John A. Stankovic: "Misconceptions About Real-Time Computing," IEEE Computer, pp.2–10, 1988.

[10] Nobuyuki Yamasaki and Yuichiro Anzai: "*Active Interface* for Human-Robot Interaction," Proceedings of IEEE International Conference on Robotics and Automation, vol.3, pp.3103–3109, 1995.

[11] 富士通株式会社，SPARClite MB86930 ユーザーズマニュアル，1990.

[12] 矢向高弘，菅原智義，安西祐一郎："PULSER: リアクティブシステムの構築に適したオペレーティングシステム"，コンピュータ・ソフトウェア，vol.11, no.1, pp.24–35, 1994.

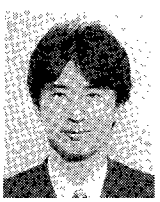
[13] 矢向高弘，菅原智義，安西祐一郎："μ-PULSER: パーソナルロボットを構築するためのオペレーティングシステム"，電子情報通信学会論文誌，J77–D–I, pp.207–214, 1994.

[14] 山崎信行，安西祐一郎："パーソナルロボットのためのアーキテクチャの提案"，日本機械学会ロボティクス・メカトロニクス講演会'92講演論文集，vol.A, pp.51–56, 1992.

[15] 山崎信行，安西祐一郎："パーソナルロボット用ハードウェアアーキテクチャ*ASPIRE*の設計と実装"，情報処理学会第48回全国大会講演論文集，vol.6, pp.91–92, 1994.

[16] 油田信一："自立移動ロボット"，機械の研究，vol.43, no.1, pp.137–144, 1991.

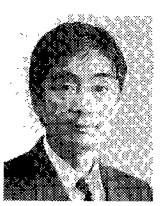
[17] 油田信一，飯島純一："自律移動ロボットの制御系のアーキテクチャ"，第8回日本ロボット学会学術講演会予稿集，pp.967–970, 1990.



山崎信行（Nobuyuki Yamasaki）

1966年5月1日生．1991年慶應義塾大学理工学部物理学科卒業，1993年同大学大学院理工学研究科計算機科学専攻修士課程修了．1996年同専攻博士課程修了．同年電子技術総合研究所入所．自律移動ロボット，並列計算機，OS，コンピュータアーキテクチャなどに興味を持つ．博士（工学）．

（日本ロボット学会正会員）



安西祐一郎（Yuichiro Anzai）

1946年8月29日生．1974年慶應義塾大学大学院博士課程修了．1988年より慶應義塾大学理工学部教授．1989年より同大学大学院計算機科学専攻教授兼任．この間，1981〜82年カーネギーメロン大学客員助教授．計算機科学，認知の情報処理過程の研究に従事．工学博士．（日本ロボット学会正会員）