計算機アーキテクチャ 113－2
（1995．8．23）

# VLIWハードウェアスタックプロセッサ

中村浄重　日笠雄一郎　酒居敬一　阿江忠

広島大学 工学部
〒739 広島県東広島市鏡山 1-4-1

高速なネットワークを介して動画, 静止画, 音声, テキストなどのさまざまなデータがやり取りできるようになってきた。このマルチメディアデータをリアルタイムに処理するシステムとして、VLIWプロセッサを用いたマルチメディアネットワークシステムを提案する。主にOSIモデルのプレゼンテーション層の処理を担うフロントエンドプロセッサとしてVLIWアーキテクチャを使用、プロセッサエレメントにはハードウェアスタックプロセッサを採用する。本稿では、VLIWハードウェアスタックプロセッサの構成について述べ、その基本ユニットであるハードウェアスタックプロセッサのプロトタイプについて述べる。

# VLIW Hardware Stack Processor

Kiyoshige Nakamura　Yuichiro Hikasa　Keiichi Sakai　Tadashi Ae

Faculty of Enginecring,Hiroshima University
1-4-1 Kagamiyama,Higashi-hiroshima City,Hiroshima,739 Japan

A high-speed network enables us to transfer various data, such as dynamic pictures,static pictures,audio and text. We propose a multimedia network system using VLIW processor, that deals with the maltimedia data. The VLIW processor supports mainly the presentation layer of OSI protocol. We adopt a hardware stack processor as a processor element of VLIW architecture. In this paper,we describe the VLIW hardware stack processor and the prototype of hardware stack processor which is an element of VLIW processor.

# 1 はじめに

近年、動画, 静止画, 音声, テキストなどさまざまなデータが、ネットワークを介して行き来している。このようなマルチメディアデータをリアルタイムに転送できるようネットワークの高速化が図られている。この高速なネットワークによって膨大なデータをやり取りする事が可能になるが、従来のように転送されたデータを端末によりリアルタイムに処理しようとすると端末のかなりのプロセッサパワーを費やす事になり端末の他のプロセスにも大きな負担となる。そこで、端末とネットワークの間にマルチメディアデータを加工するマルチメディアネットワークプロセッサ (以下 MNP) を挿入し、端末の負担を減らす方法が有効となろう。

MNP は、次のような条件を満たす。

1. ATM(Asynchronous Transfer Mode) ネットワークのような高速なネットワークを介したマルチメディアデータをリアルタイムに処理する。
2. OSI モデルのプレゼンテーション層にあたる高レベルの処理をサポートする。

このように MNP には高度な処理をリアルタイムに実行する必要があるため、MNP は並列プロセッサである事が不可欠である。

並列プロセッサにはスーパースカラや VLIW などがあげられるが、スーパースカラは並列実行のスケジュールをハードウェアで行う。多くの命令を並列に実行できるように分岐予測や out-of-order などのさまざまな技術が導入されハードは非常に複雑になった。このことは動作周波数をあげる妨げになる。

VLIW はスケジューリングをコンパイラが行うためコンパイラの負担は大きいが、ハードウェアは非常に簡素に構成できる。ただし、命令がハードのアーキテクチャに大きくよるためソフトウェアの互換が採れない問題がある。

MNP は専用プロセッサのためソフトの互換性はそれほど問題にならない事、ハードウェアが簡素に構成できる事から、MNP には VLIW アーキテクチャを採用する。これは、従来のネットワークプロセッサの VLIW 化という方向を踏襲している。

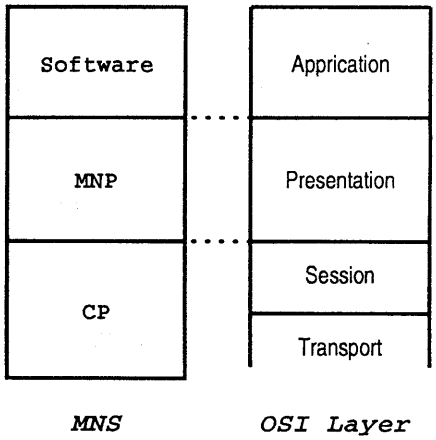


図 1: MNS と OSI モデル

VLIW のプロセッサエレメントには下記の理由からハードウェアスタックプロセッサ (以下 HSP) を用いる。

1. プログラムはプロシージャの集まりであり、例えば C 言語では約 20 命令に 1 つが call か return だと言われている [1]。他の言語についても同様の事が言える [2]。このプロシージャコールにおけるパラメータの受け渡し、レジスタ, フラグの保存、リターンアドレスの保持をスタックを使用して行うとすれば、そのスタックをハードウェア化する事でかなりの高速化が図れる。
2. ネットワークの専用プロセッサとして使用するため、コンテキストスイッチは頻繁に起こらないと考えられる。よってある程度のサイズのスタックを持たせても支障がない。
3. 演算のソース, デスティネーションをスタックトップとする事で、ハードウェアを簡略化する事ができる。また、コンパイラにおいてレジスタの割当を行う必要がないためコンパイラの負担を減らす事ができる。このことは、並列化の際の制約となり若干並列度が落ちるかも知れないがハードウェア, コンパイラの簡略化の方をとる。

本稿では、VLIW プロセッサの構成と HSU のプロトタイプの詳細について述べる。

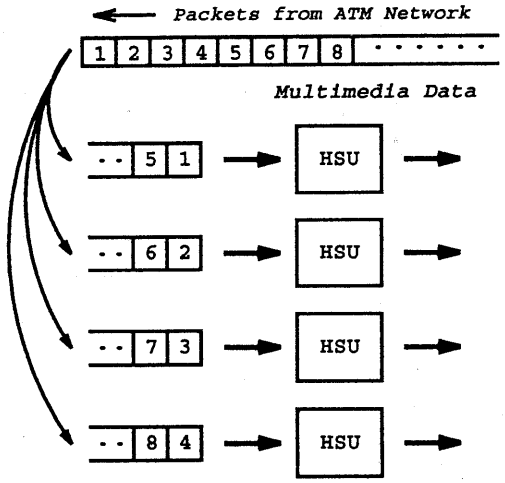


図 2: MNP の動作イメージ

## 2 マルチメディアネットワークシステム

マルチメディアネットワークシステムは2つのネットワークプロセッサで構成される。1つは、過去に当研究室で設計されたようなセッション層より下を受け持つネットワークプロセッサ(以下CP:Communication Processor)である[3]。これに関してはよいシミュレーション結果がでている[4] [5]。もう一つは、プレゼンテーション層を主に受け持つ今回のMNPである。各プロセッサとOSIモデルのネットワーク層との対応を図1に示す。ネットワークにATMを想定したMNPの動作のイメージは図2である。ATMネットワークはデータをパケットに分割して転送する。このパケットをキューを介して各HSUに入力し、HSUが順次データの処理を行う。

## 3 VLIWハードウェアスタックプロセッサ

MNPの構成を図3に示す。

MNPは4ないし8個のハードウェアスタック演算ユニット(HSU)と、コントロール部から成る。各HSUは、演算用データを保持するスタック、ALU、内部処理用レジスタ、バスインターフェースユニットを持つ。コントロール部にはリターンアドレスを保持するスタック、命令フェッチユニット、その他のコントロール回路からなる。

現時点で決定している仕様について述べる。

メモリの構成、アクセス形態はBulldog CompilerのELIモデル[8]にあるメモリ・バンク・アクセス方式を採る。これは、各プロセッサエレメント(MNPではHSU)にメモリを振り分け、メモリアドレスをa、PE数をbとする時、メモリのアクセスをa mod bのPEが処理する機構である。これにより最大bのメモリアクセスが同時に行える事になる。この方式はメモリアクセスを2種類に分けて処理する。MNPでは次のようになる。

1. コンパイル時にデータの所在がわかっている場合は、コンパイラは対応するHSUにローカルアドレスを与えデータフェッチを行い、必要に応じてHSU間でデータを転送する。(front door)

2. コンパイル時にデータの所在が分からない場合は、コンパイラはグローバルアドレスを使用しHSUにback doorを使用する命令を与える。命令実行時に対応するHSUによりデータフェッチ行い、データを転送する事になる。(back door)

各PEの演算器の出力及びスタックの内容はバスまたはクロスバーを通じて相互に転送できる。

命令フェッチはコントロール部の命令フェッチユニットがメモリアドレスを生成し、各PEが対応するメモリデバイスからの出力を取り込む事で行う。データアクセスは各PEによって生成されるアドレスにより対応するメモリデバイスをアクセスする。この際、上記の2種類のメモリアクセス方法を使い分ける。

## 4 ハードウェアスタックユニット

HSUを設計する前にハードウェアスタックプロセッサ(HSP)の動作及び性能の評価のため、ディスクリート素子にて簡単なHSPを試作した。以下、このHSPについて述べる。

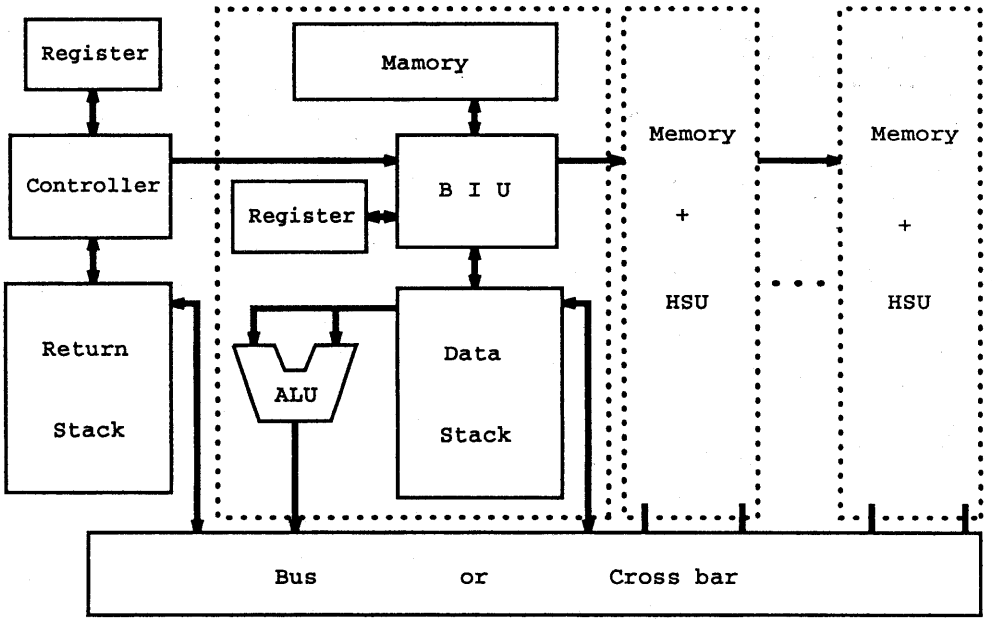


図 3: MNP のブロック図

### 4.1 ハードウェアスタック

#### 4.1.1 ハードウェアによるスタックの実現

スタックをハードウェアで実現する方法としてはスタックポインタと連続したメモリを使用する方法, 双方向のシフトレジスタを使用する方法があげられる。

#### 4.1.2 連続したメモリとスタックポインタを使用する方法

連続メモリには一般のメモリデバイスを用いる。この方法は、図 4 のように通常のメモリアクセスと同様の機構をスタックに使用する。欠点は、オペランドを同時読み出しが困難な事である。

#### 4.1.3 双方向のシフトレジスタを使用する方法

図 5 に示すように大きなサイズのシフトレジスタを用意するか、適当なサイズのシフトレジスタをチェーン状に連ねることでスタックを実現し、その一端をスタックトップとして利用する。この場合、ビット幅の数だけ同じものを並べる。この方法は、ポインタを用いないためそれを制御する機構もいらずスタックを制御が容易である。欠点としては、スタックの操作の際すべてのラッチが動作するため消費電力が大きくなる事があげられる。

#### 4.1.4 HSP に採用したハードウェアスタック

HSP をディスクリート素子で制作するため市販のシフトレジスタを用いれることと、スタックの制御が簡単なことから第 4.1.3 節の双方向シフトレジスタを用いてハードウェアスタックを構成する。

## 5 HSP の設計

### 5.1 基本仕様

HSP には基本的な機能を持たせるということで以下のような基本的な仕様を決定する。

- ○ バス幅は、8 ビットではアクセスできるメモリが小さく, 32 ビットにするとディスクリート素子の数や配線数が大きくなるため内部, 外部のバス幅はすべて 16 ビットにする。データアクセスは 16 ビットを 1 ワードとする単位でワード境界に整合させて行なう。

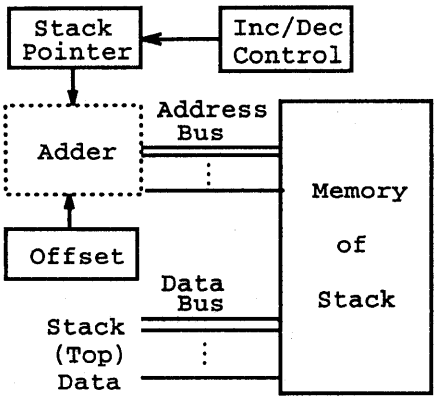


図 4: メモリを使用したスタック

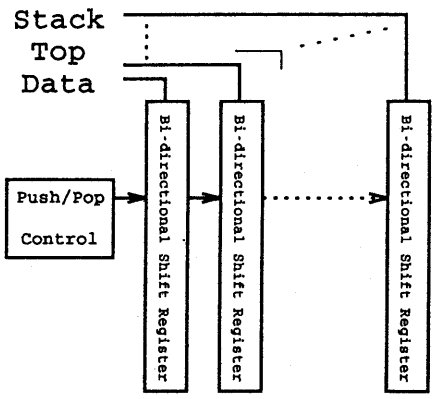


図 5: 双方向シフトレジスタを使用したスタック

○ ハードウェアスタックはローカル変数、リターンアドレスなどのいくつかの用途に使用するため、複数持っているのが望ましい。しかし、制御回路、規模の問題から 1 つだけ持たせることにする。スタックの大きさであるが、幅はバスに合わせて 16 ビットとし、深さは深い方が望ましいが、これも規模の問題から 32 ワードとする。

○ 内部状態とアドレスを保持するため次の 5 つのレジスタを用意する。

| | |
|---|---|
| **Instruction Pointer** | 命令のアドレスを保持。 |
| **Data Pointer** | データアドレスを保持。 |
| **Frame Pointer** | データアドレスを保持。 |
| **I/O Pointer** | I/O ポートアドレスを保持。 |
| **Flag Register** | フラグを保持。 |

○ FR は下位 8 ビットを使用し、上位 8 ビットは常に 0 とする。

○ 周辺回路のインターフェースのためにハードウェア割り込みをサポートする。NMI,INT0,INT1 の 3 つを用意し、固定されたエントリーポイントへのジャンプによって割り込みの処理に移るものとする。

## 5.2 命令

命令は基本的に 0 オペランド方式を採用する。演算はすべてスタックトップの要素に対して行う。プロトタイプでは演算命令、フラグ操作命令、レジスタ操作命令、データ転送命令、割り込み命令を用意する。表 1 に命令のリストを示す。ジャンプは IP にアドレスを POP することで実現する。乗算、除算は回路が複雑になるため HSP には用意しない。レジスタ操作命令は、データアクセスのためのアドレスを操作したり、フラグレジスタを読み書きするものである。データ転送命令は、メモリ及び I/O ポートのデータの読み書きするものである。スタックに即値を書

表 1: 命令リスト

| | | | |
|---|---|---|---|
| calculate | | | |
| ADD | ADC | SUB | SBB |
| AND | XOR | OR | |
| TEST | CMP | XCHG | |
| shift,rotate | | | |
| SAL | SAR | SHL | SHR |
| RCL | RCR | ROL | ROR |
| select | | | |
| WS | WP | WC | WO |
| WZ | WB | WA | WI |
| WNS | WNP | WNC | WN0 |
| WNZ | WNB | WNA | WNI |
| incliment | | | |
| INC i | INC DP i | INC FP i | |
| flag operate | | | |
| SETS | SETP | SETC | SETO |
| SETZ | SETB | SETA | SETI |
| CLRS | CLRP | CLRC | CLRO |
| CLRZ | CLRB | CLRA | CLRI |
| CPLS | CPLP | CPLC | CPLO |
| CPLZ | CPLB | CPLA | CPLI |
| register operate | | | |
| PUSH IP | PUSH DP | PUSH FP | |
| FUSH IOP | PUSHF | | |
| POP IP | POP DP | POP FP | |
| POP IOP | POPF | | |
| data transfar | | | |
| PUSH [DP] | PUSH [FP] | PUSH [IOP] | PUSH imm |
| POP [DP] | POP [FP] | POP [IOP] | |
| interupt,non-operation | | | |
| RESET | NMI | INT0 | INT1 |
| INT2 | INT3 | INT4 | INT5 |
| NOP | HALT | | |

き込むために1つオペランドをとるPUSH imm命令を用意する。

命令コードは、命令の種類や数から考え、8ビットとする。基本仕様より、命令フェッチは1ワードずつ行なう。命令の区切りをワード境界に整合させるため，適宜NOP命令を挿入させる。従って、命令フェッチはほぼ2命令に1度行うことになる。

### 5.3 ディスクリート素子による評価用ボードの実現

回路は、ライブラリの種類がある程度あり、入手し安いことから74HCシリーズのC-MOSロジックICを用いて構成した。また、74HCシリーズのライブラリにないものや複雑なロジックはLattice社のGALを用いて実現した。

機能をディスクリート素子で実現して行く上で、さらに以下のような仕様を加える。

- ○ ハードウェアスタックは双方向シフトレジスタ74HC299を16 × 4並べることで実現する。ただし、スタックトップは演算,転送命令でのデータフローを考えレジスタにし、スタックトップの上に2項演算時データを保持するテンポラリレジスタ(TR)を設ける。
- ○ ALUには74HC181を用い、シフト,ローテートはGALで実現し、命令に応じて選択することにする。ALUを通した後フラグを作成し、フラグレジスタに書き込む。
- ○ フェッチした2つ(1ワード)の命令を保持し、1命令ずつ選択して実行できるようインストラクションレジスタ(IR)を設ける。
- ○ データアクセス時の読み込み,書き込みデータを保持するためにリードデータレジスタ(RDR),ライトデータレジスタ(WDR)を設ける。
- ○ レジスタのデータ転送を行うために1本のバスを設ける。
- ○ メモリアクセスタイム,各部分の処理時間を考慮し、最大3クロックにわけて命令を実行する。また、命令フェッチは3クロックかけて、上位バイトの命令が外部バスを使用しない場合は命令の実行と同時に行ない、外部バスを使用する場合はその命令の実行後フェッチを行う。従って、命令フェッチを含めると

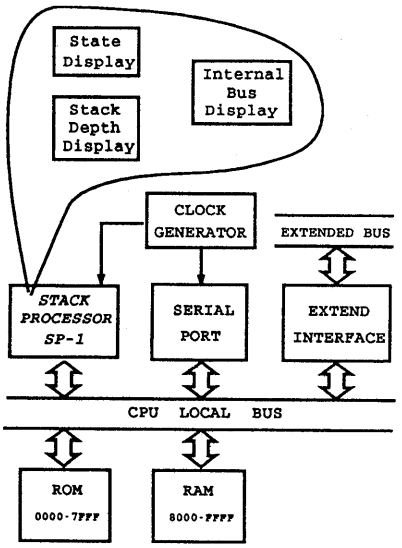


図 6: HSPのテスト用回路のブロック図

下位バイトの命令実行は1-3クロック要し、上位バイトの命令実行は3-6クロック要することになる。

以上の仕様を念頭に置き、回路の設計を行った。設計したHSPのブロック図を図7に示す。

## 6 HSUの評価用回路の設計

HSUを評価するため周辺回路を設計し同じ基板に載せる。以下のような周辺回路を用意する。

- ステップ入力が可能なクロックジェネレータ
- ROMとRAM
- 入出力装置としてRS232Cインターフェース
- 拡張インターフェース
- 内部状態を表示するディスプレイ

各周辺回路の詳細を決める。

- ○ クロックジェネレータはRS232Cインターフェースのそれも兼ねる。発生周波数はハードウェアスタックプロセッサ，メモリ等の動作速度を吟味し確実に動く1.2288MHz,2.4576MHzとする。このクロックとステップ入力によるクロックを選択しプロセッサに加える。また、RS232Cインターフェース用には307.2kHz, 153.6kHz,

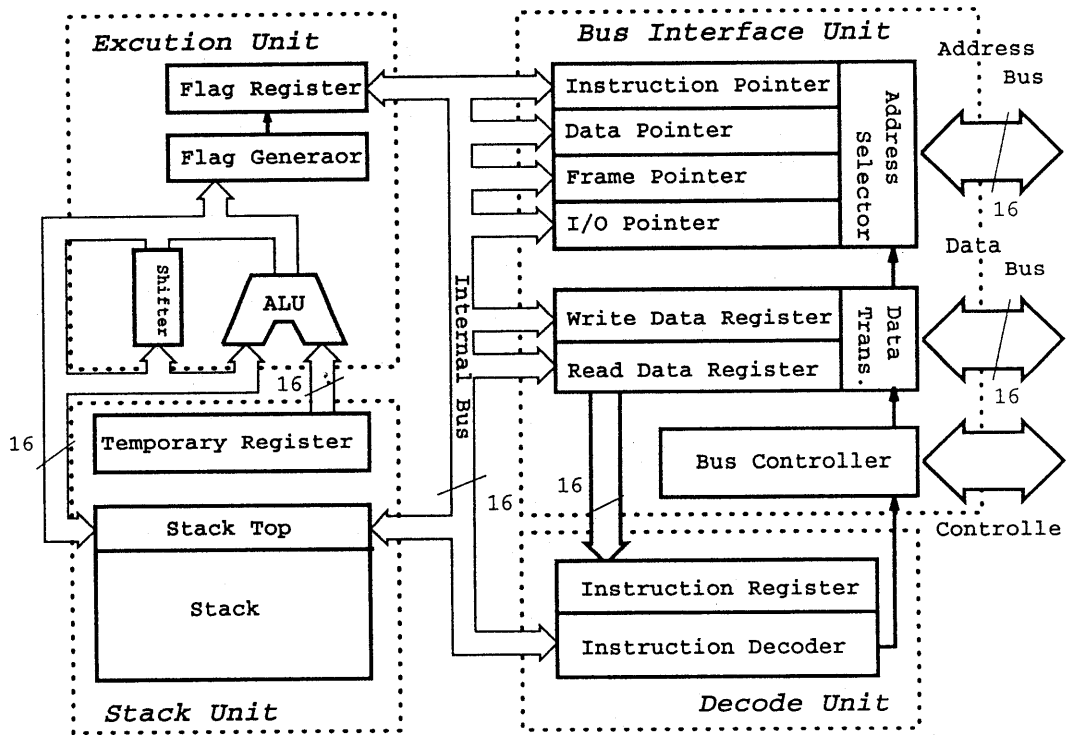


図 7: HSP ブロック図

76.8kHz, 38.4kHz のクロックを発生する。

- ○ ROM,RAM はプロセッサが RESET で 0 番地へジャンプするので ROM を 0 番地から配置し、その後ろに RAM を置く。サイズはプロセッサがサポートする 64k ワードをすべて使い半分ずつの ROM32k ワード,RAM32k ワードとする。
- ○ RS232C インターフェースにはインテル 8251 互換の$\mu PD71051C$を使用する。クロックジェネレータで発生させたクロックを用い、19.6Kbaud, 9600baud, 4800baud, 2400baud の 4 つの通信速度をサポートする。
- ○ 外部への拡張インターフェースはバッファを介して、コントロール信号, 電源を含めて外に出すことにする。
- ○ ディスプレイは 3 種類用意する。プロセッサ内部のバス上のデータを 4 桁の 16 進数で表示するもの、スタックの使用状況を把握するために現在使用しているスタックの深さを 2 桁の 10 進数で表示するもの、プロセッサの動作状態を知るために現在の実行ステートを LED で表示するものを設ける。

HSP の評価用ボードのブロック図を図 6 に示す。

また、評価用 HSP ボードの写真を図 に示す。

## 7 おわりに

現時点では、HSP の動作確認が済んだ段階である。今後 HSP にてハードウェアスタックの効果について評価を行った後、MNP の設計、評価を行う予定である。

HSP の評価用にはディスクリート素子にて構成したが、MNP は回路の規模, 速度, 実現のしやすさから FPGA を用いて制作する。

VLIW ハードウェアスタックプロセッサを我々はマルチメディアネットワークシステムのために設計したが、他の応用も可能である。

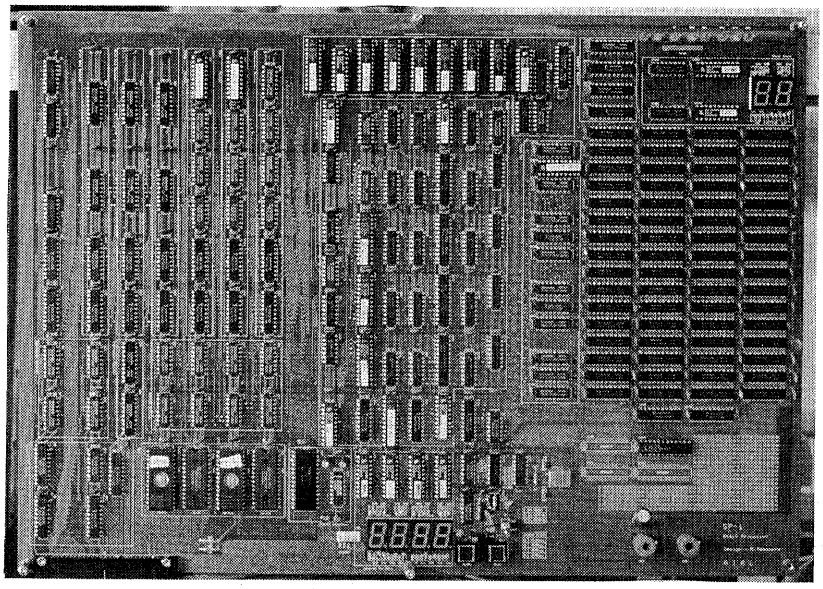

図 8: HSP の写真

ハードウェアの設計の他に、VLIW コンパイラ,MNP 上のソフトウェアについても研究していく必要があるが、その場合ネットワーク応用のみならず、広く応用できる事を考慮する必要があろう。

## 参考文献


[1] D.R.Ditzel and H.R.McLellan: *Register Allocation for Free:The C Machine Stack Cache*, SIGPLAN Notices,April 1982.

[2] R.P.Weicker. *Dhrystone:A Synthetic Systems Programming Benchmark.*, Communications of the ACM,27(10):pp.1013-1030.October 1984.

[3] Yoshihisa Sunada,Ken-ichi Nitta,Reiji Aibara, Masafumi Yamashita and Tadashi AE: *Design and Evaluation of a High-Speed Communication Processor for LANs*, Proc. of International Computer Symposium,Taipei,pp.204-209,1988.

[4] Ken-ishi Nitta,Reiji Aibara and Tadashi Ae: *On the Software of High-Speed Local Area Networks*, Proc.of 1987 IEEE Workshop on Languages for Automation, Vienna,Austria,pp.145-148,Aug.1987.

[5] Tadashi Ae and Ken-ichi Nitta, *A Design of Network Controller*, Proc.of GLOBECOM 87,Tokyo,pp.2095-2099,Nov.1987.

[6] 阿江 忠:*VLSI* コンピュータ, 電子情報通信学会,1988.

[7] Tadashi Ae,et al.: *Real-Time Multimedia Network System using VLIW Hardware Stack Processor*, Proc. IEEE Workshop on Parallel and Distributed Real-Time Systems, pp.84-89,April 1995.

[8] John R.Ellis,: *Bulldog:A Compiler for VLIW Architectures*,
The MIT Press,Masschusetts,1986.

[9] Philip J.Koopman,Jr (田中 清臣監訳/藤井 敬雄訳): スタックコンピュータ, 共立出版,1994.