Associative and Parallel Processors

KENNETH J. THURBER

Product Development Group, Sperry Univac Defense Systems, St. Paul, Minnesota 55165
Computer, Information and Control Sciences Department, University of Minnesota, Minneapolis, Minnesota 55455

LEON D. WALD

Test Equipment Engineering, Government and Aeronautical Products Division, Honeywell, Inc., Minneapolis, Minnesota

This paper is a tutorial survey of the area of parallel and associative processors. The paper covers the main design tradeoffs and major architectures of SIMD (Single Instruction Stream Multiple Data Stream) systems. Summaries of ILLIAC IV, STARAN, OMEN, and PEPE, the major SIMD processors, are included.

Keywords and Phrases: associative processor, parallel processor, OMEN, STARAN, PEPE, ILLIAC IV, architecture, large-scale systems, SIMD processors, array processors, ensemble.

CR Categories: 3.80, 4.20, 6.22

INTRODUCTION

Purpose and Scope

The purpose of this paper is to present a tutorial survey on the subject of parallel and associative processors. The paper covers the topics of system categorizations, applications, main tradeoff issues, historically important architectures, and the architectures of systems that are currently available.

Currently, the microprocessor/computer-on-a-chip revolution is providing the potential for production of very cost-effective high-performance computers through utilization of a large number of these processors in parallel or in a network. The parallel and associative processors provide a class of architectures which can be readily used to take immediate advantage of the microprocessor technology.

Surveys

A number of good surveys on the subject (or bordering on the subject) of parallel and associative processors have been published [1-10]. Some of these surveys are only of historical interest due to their age. Several conferences in this area may also be of interest [11-14].

Classification of Computer Architectures

Many approaches to classification of computer architectures are possible. Most techniques use global architecture properties and are thus valid only within limited ranges. The main classification techniques discussed below provide a framework within which to view associative and parallel processors.

Flynn [15] proposed a classification scheme that divides systems into categories based upon their procedure and data streams. The four categories are:

1) SISD (Single Instruction Stream Single Data Stream) – a uniprocessor such as a single processor IBM 360.
2) MISD (Multiple Instruction Stream Single Data Stream) – a pipeline processor such as CDC STAR.

CONTENTS

3) SIMD (Single Instruction Stream Multiple Data Stream) – an associative or parallel processor such as ILLIAC IV.
4) MIMD (Multiple Instruction Stream Multiple Data Stream) – a multiprocessor such as a Univac 1108 multiprocessor system.

Murtha and Beadles [16] proposed a classification technique based upon parallelism properties. Their categories are:

1) General-purpose network computer;
2) Special-purpose network with global parallelism; and
3) Non-global, semi-independent network with local parallelism – this category is a catchall for machines which do not fit into the first two categories.

Categories 1 and 2 have the following subcases:

1.a) General-purpose network with centralized common control;
1.b) General-purpose network with identical processors but independent instruction execution actions;
2.a) Pattern processors;
2.b) Associative processors.

The purpose of this classification technique was to differentiate between multiprocessors and highly parallel organizations.

Another possible classification view, suggested by Hobbs, et al.; [1] consists of the following categories:

1) Multiprocessor
2) Associative processor
3) Network or array processor, and
4) Functional machines.

Further, Hobbs, et al.; suggested that architectures could be classified based upon the amount of parallelism in:

1) Control
2) Processing units, and
3) Data streams.

However, it was noted that these parameters were present in all highly parallel machines and were therefore not adequate to define a machine architecture.

In his article against parallel processors, Shore [17] presents a unique classification technique which derives the machine descriptions from the description of a uniprocessor. The machine categories considered are summarized as follows:

1) Machine I – a uniprocessor.
2) Machine II – a bit-slice associative processor built from Machine I by adding bit-slice processing and access capability (e.g., STARAN).
3) Machine III – an orthogonal computer derived from Machine II by adding parallel word processing and access capability (e.g., OMEN).

4) Machine IV – a machine derived from Machine I by replicating the processing units (e.g., PEPE).
5) Machine V – a machine derived from Machine IV by adding interconnections between processors (e.g., ILLIAC IV).
6) Machine VI – a machine derived from Machine I by integrating processing logic into every memory element (e.g., Kautz's logic-in-memory computers [85, 86]).

Higbie [67] classifies computers using Flynn's four basic categories. However, he expands the SIMD category into the following four subcategories:

1) Array Processor – a processor that processes data in parallel and addresses the data by address instead of by tag or value.
2) Associative Memory Processor – a processor that operates on data addressed by tag or value rather than by address (Note: this definition does not require parallel operation; however, the definition does allow for machines that operate on data in parallel).
3) Associative Array Processor – a processor that is associative and also operates on arrays of data (typically, the operations are on a bit-slice basis, i.e., a single bit of many words).
4) Orthogonal Processor – a processor with two subsystems – an associative array processor subsystem and a serial (SISD) processor subsystem – which share a common memory array.

Higbie's categories provide for the identification of the ability to perform parallel, associative, and serial processing. Higbie defines a parallel processor to be any computer that contains multiple arithmetic units and operates on multiple data streams. Clearly, all of Higbie's four subcategories of SIMD processors fit his definition of a parallel processor.

Although none of the classification schemes presented here are mathematically precise, they are necessary to place associative and parallel processors in perspective. In the next section the terminology of associative and parallel processors will be defined for use in the remainder of this paper.

Definitions

The definitions used for the remainder of this paper are based upon Flynn [15], and are consistent with those used by Thurber [10]. The definitions are:

1) SIMD machine: any computer with a single global control unit which drives multiple processing units, all of which either execute or ignore the current instruction.
2) Associative Processor: any SIMD machine in which the processing units (or processing memory) are addressed by a property of the data contents rather than by address (i.e., multiply A and B together in all elements where $A < B$).
3) Parallel Processor: any SIMD machine in which the processing elements are the order of complexity of current day small computers and which has, typically, high level of interconnectivity between processing elements.
4) Ensemble a parallel processor in which the interconnection level between processing elements is very low or nonexistent.

Clearly, the intersection of these definitions will not be null due to the overlap in classifying machine architectures.

Reasons for Use of SIMD Processors

There are many reasons for the use of SIMD architectures. Some of the most important are:

1) Functional
   a) Large problems such as weather data processing.
   b) Problems with inherent data structure and parallelism.
   c) Reliability and graceful degradation.
   d) System complexity.
   e) Computational load.
2) Hardware
   a) Better use of hardware on problems with large amounts of parallelism.
   b) Advent of LSI microprocessors.
   c) Economy of duplicate structures.
   d) Lower nonrecurring and recurring costs.
3) Software
   a) Simpler than for multiprocessor.
   b) Easier to construct large systems.
   c) Less executive function requirements.

However, the reader is cautioned to remember that these are special-purpose machines and any attempt to apply them to an incorrectly

sized, or designed, problem is an exercise in futility.

Applications

Numerous applications have been proposed for associative and parallel processors. Whether an application is ever implemented is based upon the economics of the problem. It is not enough to be able to describe a problem solution. To date, considerable energy has been expended searching for cost-effective parallel solutions, but little has been done to actually implement them. Examples of such applications are: matrix manipulation, differential equations, and linear programming.

Several areas of application appear quite well suited to associative and parallel processing. In some cases, SIMD processors provide cost-effective system augmentation (e.g., air traffic control and associative head-per-track disks). In others, SIMD processors are very close to functional analogs of the physical system (e.g., data compression, concentration, and multiplexing).

The use of associative and parallel processors appears promising in the elimination of critical bottlenecks in current general-purpose computer systems; however, only very small associative memories are required, and cost is really not a critical factor in the solution of these problems. Such problems may be encountered in the management of computer resources, and might involve such items as protection mechanisms, resource allocation, and memory management.

The application trend for associative processors is well defined. Due to cost factors, applications will be limited (in the near future) to problems such as resource management, virtual memory mechanisms, and some augmentation of current systems, rather than to data-base management or large file searching and processing. The application trend for parallel processors and ensembles seems to be in the area of large data computation problems such as weather data processing, nuclear data processing, and ballistic missile defense. Researchers must concentrate on systems aspects of the problem, not on developing solutions to problems which are ill-defined or nonexistent. The only truly useful data-processing applications of parallel and associative processors have been developed through extensive work at the systems level, followed by parallel system design, rather than vice versa.

SIMD processors have been proposed and appear well suited for a number of applications these are listed here along with their primary references. Factors to be considered in applying SIMD processors have been discussed elsewhere [18]. To be useful for solving the listed problems, the highly parallel processors must become more cost-effective. The suggested application areas are:

1) Applications in which associative processors appear to be cost-effective:
   a) Virtual memory mechanisms [19]
   b) Resource allocation [20]
   c) Hardware executives [21]
   d) Interrupt processing [22,23]
   e) Protection mechanisms [19]
   f) Scheduling [20]
2) Applications in which parallel processors appear cost-effective and in which associative processors may be cost-effective:
   a) Bulk filtering [24]
   b) Tracking [25]
   c) Air traffic control [26]
   d) Data compression [27]
   e) Communications multiplexing [28,29]
   f) Signal processing [30]
3) Applications in which associative processors would be useful if they were more cost-effective and in which parallel processors are probably cost-effective are:
   a) Information retrieval [31]
   b) Sorting [32]
   c) Symbol manipulation [33]
   d) Pattern recognition [34]
   e) Picture processing [35]
   f) Sonar processing [36]
   g) Sea surveillance [37]
   h) Graph processing [38]
   i) Dynamic programming [39]
   j) Differential equations [40]
   k) Eigenvector [41]
   l) Matrix operations [42]
   m) Network flow analysis [43]
   n) Document retrieval [44]
   o) Data file manipulation [45]
   p) Machine document translation [46]
   q) Data file searching [47]
   r) Compilation [48]
   s) Formatted files [49]
   t) Automatic abstracting [50,51]

u) Dictionary-look-up translations [52]
v) Data management [53]
w) Theorem proving [54]
x) Computer graphics [55]
y) Weather data processing [56]

Hollander's paper [2] presents many experts' opinions about the applicability of associative and parallel processors. Slotnick [57] and Fuller [58] have compared associative and parallel processors. They concluded that parallel processors appear more useful than associative processors, but the machines would always be special purpose. Shore [17] has presented a very eloquent case against SIMD machines and parallel processors.

Matrix Multiplication on a Parallel Processor

Matrix manipulations can be used to illustrate the potential of a parallel processor. One of the major computations in many of the possible applications cited previously is the matrix multiply. Assume that a parallel processor such as that shown in Figure 1 is used to perform this operation.

Each processing element will be assumed to have three registers, AREG, BREG, and CREG.

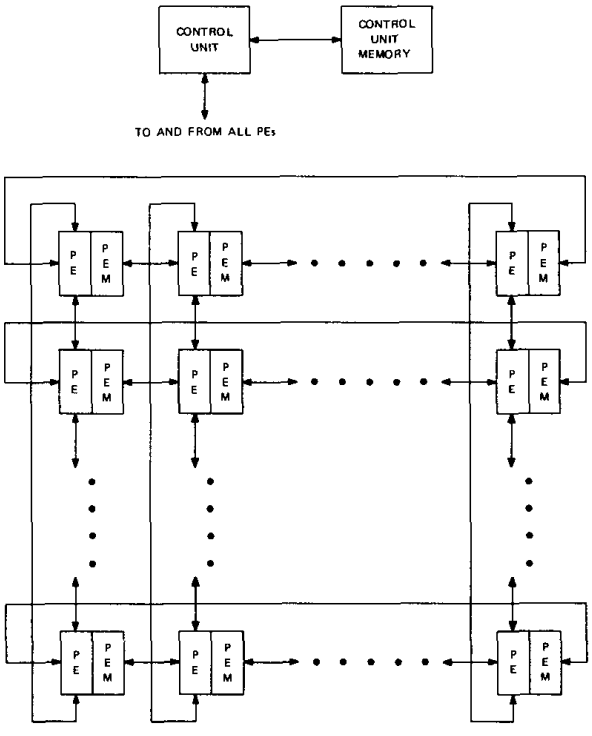

Figure 1. Parallel Processor with PE connections to the Four Nearest Neighbors.

Each cell is interconnected to its four nearest neighbors. Cannon [59] derived the following algorithm to multiply two n x n matrices together in n stages using this processor.

Algorithm:

Set:	CREG = 0 BREG [$PE_{I,J}$] = B [I,J] for all I,J $\leqslant$N AREG [$PE_{I,J}$] = [I,J] for all I,J $\leqslant$N
Shift:	Ith row of A left I-1 columns for all I $\leqslant$N Jth column of B up J-1 rows for all J $\leqslant$N
Multiply:	(TREG = AREG times BREG) In Parallel in all PEs
Add:	(CREG = CREG + TREG) In Parallel in all PEs
Shift:	AREG right one row BREG down one column
Jump:	If not N^{th} pass, Jump to Multiply:

As an example let:

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}$$

$$B = \begin{bmatrix} b_1 & b_4 & b_7 \\ b_2 & b_5 & b_8 \\ b_3 & b_6 & b_9 \end{bmatrix}$$

and

$C = A \times B$,

After initialization, the memory map for CREG is:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

For AREG, the memory map is:

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_5 & a_6 & a_4 \\ a_9 & a_7 & a_8 \end{bmatrix}$$

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.