J. King Saud Univ., Vol. 11, *Comp. & Info. Sci.*, pp. 45-67 (A.H. 1419/1999)

Associative Memories and Processors: The Exact Match Paradigm

Sateh M. Jalaleddine
Micro Linear Corporation, 2092 Concourse Drive
San Jose, California 95131, USA

(Received 10 October 1996; accepted for publication 06 May 1997)

Abstract. Associative or content addressable memories (CAM) are crucial in the implementation of high performance computing architectures for applications that require intensive data management or are cognitive in nature. The basic architecture of associative memories can be based on either the exact match or neural network models. This paper focuses on exact match associative memories. The milestone achievements in the field since the first associative memory implementation four decades ago are discussed. A classification of the diverse associative computing architectures is presented. It comprises of two levels of distinction, the associative memory organization and the processing capability which heavily depends on the application domain. Recent development in associative processing applications are also discussed which include fast routing in communication networks, memory management, database management, image processing, and artificial intelligence applications.

1. Introduction

The terms "associative" and "content addressable" have been synonymously used to identify a class of memories in which data are accessed on basis of content rather than data-location address. In contrast to conventional random access memories (RAM), a content addressable memory (CAM) can be acquired (searched) in a time only weakly dependent on the amount of stored information (e.g. words, patterns). In the past four decades, development and implementation of two fundamental associative memory architectures have evolved; namely the exact match and neural network models.

On one hand, the primitive operation of the "exact match CAM" involves finding in parallel all stored words that match an input. The masking of bit fields can be achieved at the input where only selected fields are allowed to participate in the match operation. Also another type of masking can be accomplished by having the bit representation of stored words ternary ("0", "1", and "don't care") instead of two-state

binary. Although all words are compared simultaneously, results are reached with *no* interaction between stored words.

On the other hand, a "neural inspired CAM" operation requires merely all the information in the memory to interact and collectively evolve to a decision (e.g. best match). In particular, Neural CAMs possess outstanding capabilities in retrieving patterns from noisy input data. For instance, a winner-take-all CAM dynamically finds the best match (in Hamming distance sense) pattern to an input in one cycle, while an exact match CAM requires several memory cycles to achieve the best match result [1]. From an architecture point of view, the "exact match CAM" involves employing a comparator for each data entity (e.g. bit, word, block) which merely involves calculating the exclusive - OR function, while a neural network CAM implements a biological neural inspired model. Due to the large amount of contributions in each area, the scope of this paper will be limited to the discussion of exact match associative memories. In the rest of this paper, the term "exact match" will be implicit whenever CAM or associative memories are mentioned unless otherwise stated.

The key feature of content addressing in associative memories makes such devices very efficient search engines. When an associative memory is accessed, all data elements (or selected bit fields of selected words) are compared in parallel against the input data. Thus associative memories are naturally parallel single-instruction multiple-data (SIMD) machines, whereby one instruction (match) is broadcast to multiple data elements to perform data-parallel search. The ability of an associative memory to retrieve and process data eliminates the classical Von Neumann bottleneck. Memory access and manipulation of pointers/indexes that constrain the performance of conventional computing architectures are avoided. Since stored data in associative memories are accessed by content, the need for data-location address is eliminated. This provides system level flexibility; namely scalability and fault tolerance. Scalability of memory size (from a system point of view) is facilitated since the number of data elements is not limited by an address space. Also with content-based data storage and deletion, faulty data elements can be easily bypassed (switched off) without the overhead of ensuring a full address space as is the case in RAM-based systems.

The excellent search capability of associative memories makes them cost-effective for applications that consume a significant amount of processing time in searching data structures. Many data structures can be supported by an associative memory such as arrays, tables, trees, and graphs. Tables and arrays are the most natural fit because of the two-dimensional physical organization of associative memories. Trees and graphs can be mapped to an associative memory by numbering the different levels of trees/graphs and appending these numbers (included as tag fields) to each stored node. One-dimensional data structures such as strings of characters can also be mapped into an associative memory where each string contains a linear array of characters.

For associative systems to be cost effective, their architecture and functionality has to be tailored for the targeted application. Several recent implementations of associative systems as application-specific accelerators have been reported and will be discussed in the paper. These associative architectures have been developed for a wide range of high-performance applications with various degrees of complexity ranging from virtual memory management subsystems to real time speech translation and image processing systems. Such systems typically comprise of a relatively small size associative memory and functional blocks that are optimized to execute application-specific operations.

This paper presents a unified framework for the diverse architectures of associative memories and processors. It also presents their applications where emphasis is placed on recent developments in the field. The following section gives a brief genealogy of associative systems. In this section, no attempt was made to change the wide range of terminology used in the literature (which is often confusing) to describe associative architectures. The third section presents a classification of associative computing architectures. This classification clearly identifies the salient differences among exact match associative systems at two levels of distinction. At the first level, the basic memory architecture can be either hardware implemented in a fully parallel fashion or emulated by RAM in a bit/word serial structure. The second level of distinction further classifies associative architecture according to their degree of functionality and data processing capability. Eventhough the classification at this level is heavily dependent on the application domain, the distinction is solely made on the basis of the fundamental processing capability of the architecture. The last section discusses the various associative computing applications using the terminology established in this paper.

2. Milestone Achievements

The speed and flexibility of accessing data by content for information retrieval was envisaged five decades ago. Inspired by the power of association in the human mind, Vannevar Bush [2] described a storage device (the "memex") that performs fast information retrieval based on association rather than indexing. In 1956, motivated by the developments in superconductive circuits technology (cryotron), Slade and McMahon [3] implemented the first associative memory (referred to as catalog memory). For a basic CAM architecture, the reader is referred to Fig. 2 of this paper. The constructed cryogenic memory consisted of 4-word by 5-bit fully parallel associative memory capable of executing match and write operations. As the versatility of the semiconductor technology was becoming evident in the early 1960s, proposals for fully parallel associative memory circuits have been reported [4]. The fabrication of the first integrated circuit associative memory which consisted of a 1-bit cell was presented

in 1966 [5]. In the decade following the introduction of the cryogenic associative memory, a large number of small scale fully parallel associative memories implementations involving a wide spectrum of technologies were reported and reviewed in [6].

Following the implementation of the catalog memory, the late 1950s and 1960s have witnessed the development and consolidation of concepts in the architecture, algorithms, applications, and implementation of associative systems. As an extension to the concept of the fully parallel catalog memory, an associative computing architecture based on the so called distributed logic memory was introduced in 1962 [7]. Its targeted application was information retrieval in which variable length character strings can be stored and retrieved. The architecture consists mainly of a linear array of intercommunicating cells. Each cell stores a character (or a set of characters) and posses the necessary logic to match in parallel incoming data and to send control signals to its neighboring cell. Many implementations based on the distributed logic memory concept have been proposed, however an early practical implementation was exemplified in the PEPE system developed for radar related problems [8]. Because of the large number of processing logic required in fully parallel architectures, bit-serial associative systems have emerged [9,10]. Instead of having logic associated with each bit, a processing element is allocated for each word (or a set of bits). Thus, the memory is processed in a bit-serial word-parallel fashion. Since the number of processing elements is considerably reduced especially for large size words, such architecture allows the practical realization of complex parallel operations besides the search function. A prominent implementation of an associative bit-serial architecture is the Goodyear's Staran system [11, 12]. For mass storage devices, an associative architecture based on the so called block-oriented associative memory was introduced in 1970 [13]. This architecture targeted database applications in which a processing element is associated with each block of data (logic-per-track) of a fixed-head storage disk.

Arithmetic and relational search algorithms for associative memories that exploit the content search primitive have been developed. Falkoff [14] presented algorithms that can perform relational search functions such as maximum, minimum, greater than, less than, between limits, nearest to (in magnitude), and sorting. These operations were described for both fully parallel and bit-serial associative memory architectures. Estrin and Fuller [15] introduced the sequential-state-transformation technique for performing bit-serial word-parallel arithmetic operations. This technique requires only two primitive operations, the exact match and multiwrite operation. The multiwrite operation involves parallel write to selected bits of several selected words. An associative processor employing the sequential-state-transformation technique developed for image processing applications was reported in [16]. The processor content addressing was implemented using fully parallel associative memory organization. A similar processor architecture was proposed [17] with provision of using ternary associative memory cells [18] as

opposed to two-state binary cells. The ternary associative memory, called a functional memory, has been identified to be effective in image processing and tree/graph search applications [17]. The first integrated circuit implementation of a dynamic ternary associative memory cell was reported in 1972 [19].

The large number of contributions, as of early 1970s, discussing many aspects of associative processing was manifested in two survey papers [20, 21]. A later survey [22] also included such developments. The architecture and features of associative processors developed in the 1960s and early 1970s are presented in [23, 24, 25]. In [25] associative algorithms have been discussed and extended to cover many variants of arithmetic and logical functions. The availability of VLSI in the 1980s has led to the improvement of existing associative systems and the evolvement of new ones. A VLSI version of the Staran for airborne radar tracking was developed [26, 27]. Two bit-serial associative architectures have also been reported [28, 29]. Many implementations of associative architectures have been developed for various applications. These contemporary implementations will be discussed in the following sections.

3. Classification of Associative Architectures

Renewed interest in associative processing have been steadily gaining ground since the mid 1980s [30-34]. This can be attributed to (i) the technological advances in microelectronics which paved the way for efficient and cost-effective implementation of associative architectures, (ii) as conventional processor speeds are reaching their physical limitations, associative solutions are becoming increasingly important in search intensive applications, (iii) and the evolvement of new high speed applications (e.g. LAN routers) in which the employment of associative memories is imperative for maintaining high system performance. Many associative architectures with different degrees of functionality have been developed for various applications. This section presents an articulation of a classification for associative computing architectures. It provides a distinctive view of the various memory organization, fundamental architectural principles, and processing capability of associative systems.

Figure 1 shows a classification of associative computing architectures. At the highest level, the basic distinction made is based on the type of associative paradigm embodied in the architecture. The two fundamental associative models are the "exact match" and "neural network" models. As mentioned earlier, this paper is concerned with the "exact match" associative architectures. The first level of distinction in exact match architectures is the associative memory organization. The choice of memory organization is based on whether the match operation is hardware implemented in a fully parallel fashion or serialized in one of the two memory dimensions. Associative architectures are further classified according to their degree of functionality and data

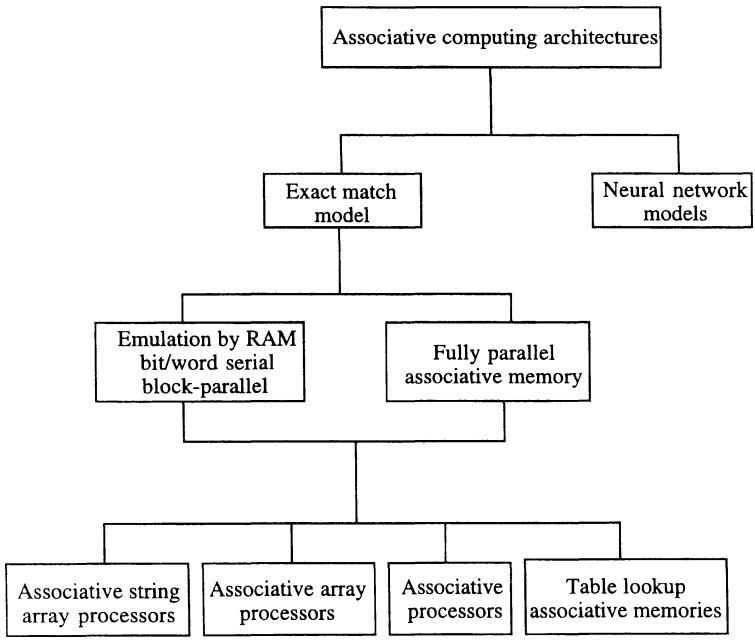

Fig. 1. Classification of associative computing architectures.

processing capability. This level of distinction is heavily dependent on the application domain. Four categories are identified; table lookup associative memories, associative processors, associative array processors, and associative string array processors. Each category features specific characteristics and unique architecture oriented towards improving the performance of certain data processing functions. In the following sub-sections, first the different associative memory organizations will be discussed which consist of either fully parallel or RAM emulated bit/word serial structures. Then as the functionality of the basic associative memory is extended, the four associative computing architectures are identified and presented.

3.1 Associative memory organization

The basic architecture of a CAM can be either hardware implemented in a fully parallel fashion or emulated by RAM. Fully parallel associative memory is capable of performing the match operation in one cycle at the expense of extra logic per storage element. Associative memories emulated by RAM require less logic per storage element while more than one cycle is needed to execute the match operation. In all CAM configurations, a SIMD control style is assumed where a control unit broadcasts an instruction to all memory elements.

3.1.a Fully parallel associative memory

All rows and columns in a fully parallel associative memory are compared simultaneously. Fig. 2 shows a typical n-word by b-bit fully parallel associative memory. Each bit location consists of a storage cell and a two-input Exclusive-OR. The Exclusive-OR inputs are the stored bit and the corresponding bit of the input word. Each row consists of a b-bit word and one match line. The match line is connected to ground (logic zero) through a switch at each bit location in the row. If any switch in the row is turned on, then the match line for that row is at logic zero. The switch at each bit location is controlled by the output of the Exclusive-OR. Thus whenever any bit in a word mismatches the corresponding input bit, the Exclusive-OR output would be high (logic one). Consequently the switch at that bit location turns on and the match line for that word is at logic zero. Thus all the words that do not match the input word will have their match line at logic zero.

The mask register shown in Fig. 2 provides the capability of having only selected bits to participate in the match operation. This is an important feature in associative memories since it allows the search of specific memory fields. The execution of the exact match operation is performed by first precharging the match lines to logic high by enabling the match line precharge signal shown in Fig. 2. The input data is then presented at the bit lines as determined by the information stored in the I/O buffers and the mask register. Only the words that exactly match the input will have their match line stay at logic high, otherwise the match line discharge to zero. This information is sensed by the sense amplifiers and stored in the word response registers.

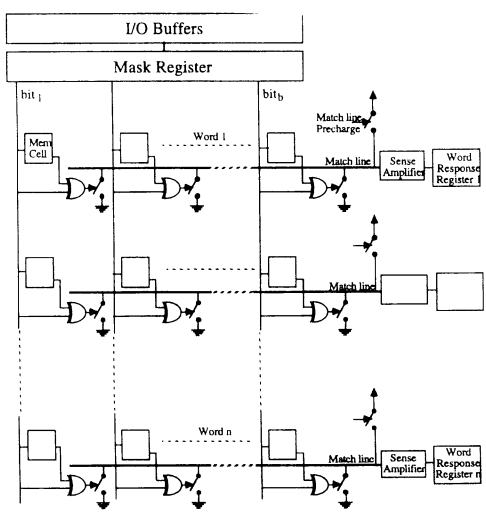

Fig. 2. n-word by b-bit fully parallel associative memory.

Data storage and access in the fully parallel CAM are word-oriented. Recently, a flag-oriented CAM has been proposed [35, 36]. It is realized by introducing a simple masking scheme to the standard RAM address decoder. Depending on the input word, the maskable address decoder can access several memory cells at a time. The memory array is formed by a one bit column of a standard RAM. Each word is stored as a 1-bit flag by having the word value considered as the address to the 1-bit flag cell. If the content of a flag cell is "1", this implies that a word with content equal to the corresponding address of that bit cell is stored. The advantages of this architecture are the easy extensibility of the memory size and the ordered nature of the stored words. Cascading of the bit cells is simple since it is similar to that of a standard RAM. Since the words are stored in an ascending order, relational search operations are facilitated. A disadvantage of the flag-oriented CAM is that the size of the 1-bit flag vector is 2^n where n is the number of bits per word. Thus, the practicality of such architecture is limited to applications with small word length.

3.1.b Emulation by RAM

Two practical associative memory configurations can be emulated using a RAM which are bit-serial word-parallel (bit serial) and word-serial bit-parallel (word serial) the bit cell area. Instead of having a comparison logic for each bit as in the fully parallel case, a group of bits share a common comparison logic. Bit serial associative memory includes comparison logic for each bit column (bit slice) and thus each memory column is processed at a time. In the word serial case, each memory row or word has comparison logic which allows each word to be processed at a time. A variation of these two associative memory configurations is the block-parallel configuration. This entails dividing the memory into blocks of words while bit/word serial processing is performed within each block. for instance in a word-serial block-parallel configuration, all blocks are processed concurrently while each block is processed in a word-serial bit-parallel fashion.

The basic structure of a typical bit serial associative memory is shown in Fig. 3. The memory array comprises of conventional storage cells with bit column access capability. Data are stored horizontally and processed in a vertical fashion. The I/O buffer provides the external memory access for data read/write operations. Each memory row has a 1-bit comparison logic for bit serial processing. The comparison logic block comprises of one bit comparator and a register for storing intermediate match results. The final match results are stored in the word response registers. Bit-slice data movement to the comparison logic can be achieved by either having the memory array implemented as a block of circular shift registers or providing access capability to individual bit-slices. The bit-slice memory access scheme is widely used and adopted in the CAM shown in Fig. 3. One way to vertically access the memory is by having each word in the memory array connected to the comparison logic through a one bit line. Such connection provides the comparison logic with the content of the selected bit. The

bit-slice selector selects a memory bit column by connecting each memory cell in the selected bit-slice to its corresponding word line. The bit-slice selector is also responsible for providing the comparison logic with the input bit which corresponds to the selected bit column.

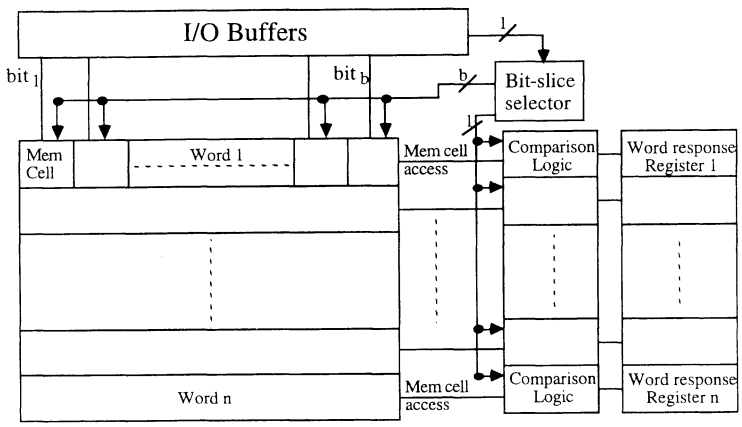

Fig. 3. A bit serial associative memory.

The operation of a bit serial CAM entails serial processing of memory bit columns. The bit-slice selector unit selects one bit column at a time for comparison with the corresponding input bits in the comparison logic block. After all bit columns have been processed, the final match results are stored in the corresponding word response registers. In such memory architecture masking of bit locations is achieved through the bit-slice selector unit in which only the desired bit columns are accessed and allowed to participate in the match operation. Bit serial associative memories are best suited for applications where the average number of bits to be accessed in match operations is relatively small compared to the word length. The most popular and early bit serial associative memory implementation is that of the Staran associative system in which the memory module consisted of 256-bit by 256-word. An extension to the bit serial architecture is the byte-serial associative memory in which a number of bit-slices (byte wide) are processed at a time instead of just one bit-column. Such associative memory is usually employed in database applications where character (byte) matching is desired. The degree of area-speed tradeoff is a design parameter which depends on the application and specific system requirements.

The operation of a word serial CAM is similar to that of a bit serial CAM except that words (memory rows) rather than bit columns are processed serially. In a word serial associative memory, all bits in a word are compared simultaneously while the

memory words are time multiplexed through the comparison logic. The memory access is needed only in the horizontal dimension for both read/write and comparison operations. The number of 1-bit comparison logic units needed for the word serial CAM is equal to the number of bits per word. For efficient use of the comparison logic, the word serial CAM is usually employed in applications where masked (through the input mask register) match operations are infrequently needed. Moreover, for a reasonable speed compromise the number of words should be kept relatively small.

3.2 Table lookup associative memories

This is the most basic category in which associative memories are employed as simple table lookup memories. Applications in this category include virtual memory management (translation-look-aside buffers) and address filtering in communication networks. The effectiveness of the associative memory solution in such applications has resulted in the introduction of the first commercial CAM [37]. This CAM is designed for real-time address filtering in LAN bridges.

Associative memory architectures, discussed in the previous section, produce a word response vector as the outcome of a match operation. In table lookup associative memories, the content of the word response vector is processed to arrive at a desired outcome tailored to the targeted application. Typical table lookup associative memory outputs are count of matched words, some/none match flag, and serial output of the content/address of matched words. The some/none match flag indicates whether a match is found. In case of a multiple match, the count signal provides a tally of the number of matched words. Moreover, for serial retrieval of matched words, a multiple response resolver is required. The multiple response resolver selects only one matched word at a time. The selection order is immaterial to the CAM operation. However, the most efficient implementation of a prioritizer is one that is based on physical word location in which the first memory location is assigned the highest priority. The prioritizer output can be either encoded to provide the address of the matched word or utilized to output the data stored at the match location.

Depending on the application, the output of an associative memory in response to a match instruction can be either the physical address or data content of the match location. The write instruction of a CAM can also be based on either physical location address or stored information content. The write instruction based on location address is identical to that of a RAM. A more popular write scheme is one that is based on the content of internal status registers. For instance, an empty/occupied flag bit could be allocated for each word to identify whether a word contains useful data. Such addressless write scheme allows the data to be stored anywhere in the memory regardless of location address. The CAM content can also be modified by using the results of a match instruction as the criterion to write certain memory locations. For example, a CAM may be instructed to modify the content of all the words that were responders to a previous

match instruction. This requires a multiwrite instruction where several words are written in parallel. In table lookup associative memories such multiwrite operation is usually used for memory testing purposes where all memory words are initialized to zero or one. Many techniques for increasing the word length and number of words in associative memories have been developed and summarized in [38].

3.3 Associative processors

Table lookup associative memories that encompass arithmetic or logical processing capabilities are referred to as associative processors. the key function of an associative processor is the parallel processing of match operation results. The basic architecture of such an associative processor is illustrated in Fig. 4. It consists of a set of processing elements (PEs) which are controlled by a control unit. Each PE comprises a CAM row (word) along with processing logic and storage registers. The processing logic typically implements Boolean functions for manipulating the match results. The registers are needed for storage of intermediary results in bit serial arithmetic and multi-instruction operations. This also allows the conditional execution of CAM instructions on selected rows based on previous instruction results.

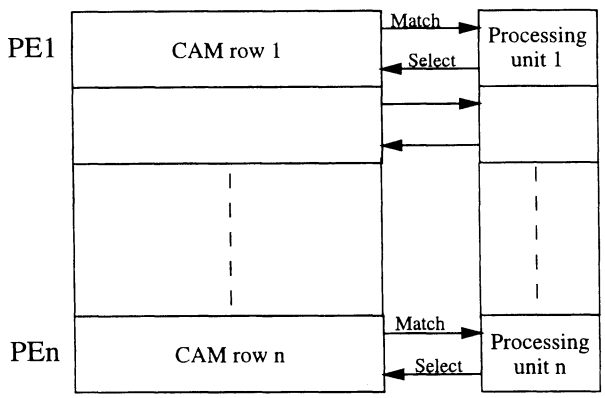

Fig. 4. Assiciative processor architecture.

The main difference between an associative processor and a traditional parallel processor is that the former uses the match results for data manipulation as opposed to direct data manipulation in the case of a RAM-based parallel processor. The processing unit in an associative processor uses the match result as the basis for selecting the corresponding CAM row for further processing (e.g. write, read). For instance, the realization of bit serial word parallel arithmetic operations using the sequential state transformation method [15] requires a series of match and write operations on selected

bit fields. The implementation of such a processor is exemplified in the associative processor presented in [39].

3.4 Associative array processors

The key characteristic of associative array processors is that the PEs are interconnected through an interprocessor communication network. Figure 5 shows a generalized associative array processor architecture which is similar to that of the associative processor except for the added feature of interprocessor communication network. The PEs are relatively simple and achieve parallel operation by spatial replication of instruction execution. However, the presence of the interprocessor communication network provides the opportunity of having the array partitioned into subsets of one or more PEs where distinct control signals could be applied to each subset. This provides the realization of a dynamically reconfigurable multiple-instruction multiple-data (MIMD) architecture.

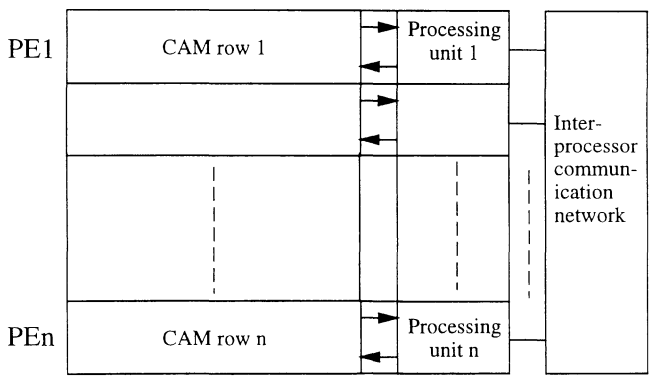

Fig. 5. Associative array processor.

The selection of interprocessor communication network topology depends on the requirements of the targeted application domain. For instance, the most popular intercommunication network for image processing applications is the two-dimensional mesh. Such topology entails arrangement of the PEs into a two-dimensional array in which each PE is interconnected to its neighboring PEs. Depending on the desired intercommunication complexity, each PE can be connected to its four or more neighboring PEs. Fig. 6 shows a two-dimensional PE arrangement where each PE is connected to its four nearest neighbors. The figure also shows the array partitioned into subsets of PEs in which the execution of multiassociative operations can be achieved. A notable implementation of an associative array processor is the lowest level (pixel) processor in a hierarchical image understanding architecture. An overview of associative array processors can be found in [40-42].

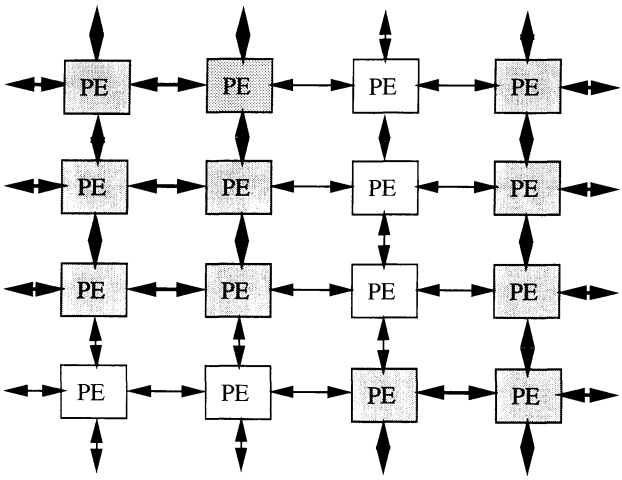

Fig. 6. Two-dimensional grid arrangement of PEs.

3.5 Associative string array processors

The key distinctive feature of associative string array processors is the local intercommunication control between PEs. Figure 7 shows a basic associative string array processor. The PEs are arranged in a linear array where neighboring PEs are interconnected by a local physical link. The main utility of such local link is to propagate the match result from one PE to its adjacent PE. This allows PE activation based on the match result of its neighboring PE. Communication between distant PEs can be accomplished through the intercommunication network. This provides the capability of forming a contiguous set of PEs that are physically apart. The application domain which motivated the development of such processors is pattern or character string matching. These processors have been referred to as distributed logic memory or cellular associative processors.

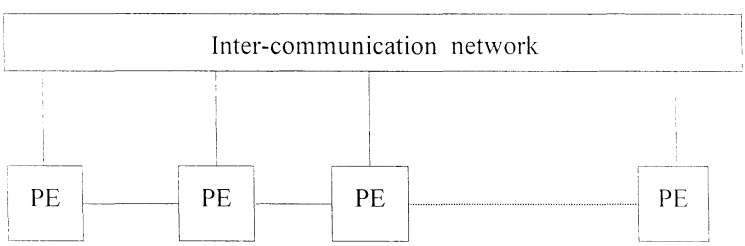

Fig. 7. Associative string processor.

The concept of associative string array processors was proposed by Lee [7]. The proposed architecture is based on local intercommunication between adjacent PEs. Recent architectures such as the one in [42] provided intercommunication of control signals among distant PEs through a global communication network as illustrated in Fig.7. Data in associative string array processors are initially distributed along the PEs for subsequent parallel processing. Functions are implemented as a sequence of instructions. Execution of instructions results in activation of substrings based on their content. Data within the activated substrings are then processed.

4. Applications

In high-performance applications where the speed of data search or pattern matching is the major impediment to the aggregate system performance, associative computing would be a cost-effective solution. Recently associative computing has been employed in a wide range of applications. In this section we discuss these applications which are fast routing in communication networks, memory management, database management, image processing, and AI applications.

4.1 Fast routing in communication networks

As switching speed and bandwidth of communication networks increase, packet routing in multiple interconnected networks has to be executed at high speed. The basic function of packet routing is address table lookup at the interface between interconnected networks. Thus, the use of an associative memory in such applications is indispensable. In LAN bridges and routers, CAMS are very crucial for achieving fast address filtering especially in high speed networks such as FDDI [43-46]. The clear advantage of CAMs in the acceleration of address filtering in LAN bridges has prompted their commercial availability. Since the introduction of the first commercial CAM in late 1988 by Advanced Micro Devices, several manufacaturers have introduced similar CAMs which are aimed at address filtering applications in LAN bridges [47]. These CAMs belong to the table lookup associative memories category described in Section 3.2.

The central function of a LAN bridge is to pass packets between different networks. A popular routing strategy is the non-source routing which is based on the destination address of the addressed node. In such systems the only routing information that is sent in the packet is the destination address. Thus the bridge must match each packet address against an address list of all the active nodes that reside on an attached network. This address matching has to be performed quickly so that network speed is not degraded. Table lookup CAMs have been very effective in handling such address filtering in high speed networks. CAMs have also been proposed for fast routing in telephone networks where hierarchical addresses are employed [48-50].

4.2 Memory management

A widely known application of CAMs is the implementation of memory mapping hardware used in virtual memory systems. In virtual memory computer architectures the CPU issues a virtual address which must be converted to physical address before the local RAM can be accessed. A fully associative cache known as translation lookaside buffer (TLB) is often used to perform the virtual-to-physical address translation. This associative hardware solution is crucial since the TLB is located in the critical timing path between the CPU and memory. Each word in the TLB corresponds to a block or page of words stored in the RAM. The TLB stores the most frequently used, or at least the most recently used, virtual page addresses. When the CPU issues a virtual address, the TLB searches for a match. If a match occurs, the TLB produces the physical address of the corresponding page stored in the RAM. Discussions on TLB design issues can be found in [51-53].

CAMs have also been utilized for memory reliability enhancement. As the size of memory ICs increases, yield and reliability issues become important. CAMs have been employed in improving memory yield by isolating hard errors (defects) through redundancy. For memory system reliability improvement, CAM-based redundancy approach has been reported to be more efficient than error-correction coding and page-swapping techniques [54]. CAM-based redundancy is accomplished by incorporating a CAM into the memory system (multiple standard memory ICs) to provide replacement for defective memory locations. Thus CAM locations are used to map into the address space in place of faulty locations. When a faulty memory location is detected during testing, its address along with the corresponding data are stored in the CAM. Once a faulty location is accessed, the CAM matches the address and the corresponding stored data in the CAM is accessed. The CAM access is in parallel with the memory access which avoids increase in memory access time. A CAM-based redundancy approach has also been used a improving the reliability of Flash IC memories by integrating a small CAM on chip [55].

Another application of CAMs is in the implementation of the matching memory for data driven processors. These parallel computing processors coordinate instruction execution based on the availability of data operands. Instructions wait for the availability of their operands before being executed. Thus, high speed data matching is very crucial for such computing architectures. Several table lookup associative memory implementations for data-flow processors have been reported [56-59].

4.3 Database management

The parallel search capability of associative memories makes such devices well suited for database applications. Many associative memory systems have been reported for relational database management [60-66] and text processing [67-72]. Relational database machines are dedicated computing engines which accelerate relational

operations such as Selection, Projection, Join, and Set operations. Text processing systems are dedicated processors which accelerate text information retrieval operations such as character string search.

The relative high cost per bit of associative memories prohibits their use for database mass memory storage. Thus, different approaches have been adopted to overcome such limitation. One approach is to build associative processing capabilities into the read/write mechanism of the mass storage device. This approach represents early attempts for implementation of database accelerators which is known as logic-per-track concept [13]. It involves adding an associative processor to each read/write head of a fixed-head disk. Stored data in each track are searched and only relevant data is transferred from the disk. Placing a head for every track is not an economical approach in modern disks [73]. In any case, attempting to modify the secondary storage device for database acceleration is quite costly since specific design is needed for each storage device type. Another approach to associative database acceleration is to implement an associative memory as the database cache. However, such approach is costly since the size of the associative memory needed to implement the database main memory is quite large. Many early designs have adopted this approach [60-62] where the database machine is built around an associative main memory.

Instead of attempting to modify the secondary storage device or main memory of the database computer, recent approaches [63-72] involve implementation of associative data filtering at the interface between main memory and secondary storage. Such approach can be implemented using relatively small size associative memory and it can be easily incorporated into existing secondary storage device controllers. The rationale behind database filtering is to reduce the amount of data transfer from the mass memory media to main memory (processor staging memory). All data read from secondary storage are compared against some search conditions in order to determine the qualified data that need to be transferred to the host computer. Besides data selection, data filters often implement more complex database operations. For instance, relational database filtering usually involves operations such as Projection, Join and Set operations which are frequently used and require large data movement. Efficient associative memory hardware implementation which is suited for relational operations is presented in [74].

4.4 Image processing

In real time image processing applications large amount of data arrays has to be processed, using identical processing steps, at high speed. The inherent SIMD processing and array oriented data structure in associative computing architectures make them attractive for such real time applications. A wide range of image processing applications have been implemented using associative architectures with varying degree of functionality; computer vision [75-78], image processing tasks [79-88], pattern matching [89], particle track finding [90], data compression [91-93], and filtering [94].

In [85-94] table lookup associative memories are employed where the main operation in these implementations is matching of an input vector against stored data. In image processing applications, associative array processors are employed where image pixels are spatially mapped into corresponding processing elements (PEs) in the array. PEs are basically simple bit-serial associative processors which are connected through a communication network. Recent implementations [75, 95] have employed dynamically reconfigurable networks to support effective implementation of a wide range of applications.

Typically, computer vision involves three levels of abstraction. At the lowest level, binary image pixels are mapped into an array of PEs where processing such as filtering and local feature (e.g. region, edge, texture, etc.) extraction are applied. The intermediate level involves relating the extracted features to image objects. Functions implemented at this level include operations such as grouping of regions and statistical measures of extracted information. At the highest level, knowledge-based interpretation of the intermediate level results is performed. A representative implementation of such an architecture is the image understanding architecture (IUA) [75]. It consists of three tightly coupled heterogeneous parallel processors with a pyramid structure. Associative processing is extensively utilized at the lowest level where a 512 x 512 associative array processor is employed [96]. Each PE is a bit-serial associative processor with 320 bits of local memory. The PEs are connected in a two-dimensional mesh, however, the network can be dynamically partitioned into several independent segments. This allows multiassociative processing where each segment is able to operate on locally broadcast values to locally compute its own responder summary in parallel with all other segments.

4.5 AI Applications

Production systems and logic programming are widely used in many artificial intelligence applications. Such declarative programming approaches use some form of If-Then rule-based programming. Pattern or symbolic strings matching is extensively utilized in such programming environments. This makes associative computing very attractive for accelerating the execution of these programming paradigms.

Many associative string array processor implementations have been reported for artificial intelligence applications [97-110]. In [97-108] various approaches for symbolic processing and especially for Prolog execution accelerations have been presented. These implementations have focused mainly on the acceleration of functions such as clause filtering, unification, and backtracking and dereferencing stack management. Recently implementation of associative processing in AI real time speech translation [109] and genetic algorithm-based machine learning [110] have been reported.

5. Conclusions

This paper has presented a unified classification of associative architectures. It clearly identifies the salient architectural differences at both levels of distinction; the memory organization and the functional levels. At the memory organization level, associative architectures can be either fully parallel or emulated by RAM. At the functional level, four associative architectures have been identified which are table lookup associative memories, associative processors, associative array processors, and associative string array processors. The paper also discusses the recent associative applications which span a wide spectrum of areas which include fast routing in communication networks, memory management, database management, image processing, and AI applications.

The data search feature of associative memories will always make their level of integration and cost per bit inferior to those of RAMs. However in high-performance applications that require frequent data search or pattern matching, associative memories have been shown to be cost-effective and in some cases the only solution for maintaining the required high system performance. As the relative cost of high performance chip fabrication is decreasing, the implementation of associative solutions in new applications is becoming increasingly attractive. It is our hope that a unified view of associative architectures would aid in expediting the developments in the field. As the first commercial associative memory was introduced only few years ago, the full potential of associative computing in all its aspects is yet to be seen.

References

[1] Jalaleddine, S.M. and Johnson, L.G. "Integrated Circuit Associative Memory Based on Neural Mutual Inhibition." *IEE Proc., Pt.G*, 139, No. 4 (Aug. 1992), 445-449.

[2] Bush, V. "As We May Think." *Atlantic Monthly*, 176, No. 1 (July 1945), 101-108.

[3] Slade, A.E. and McMahon, H.O. "A Cryotron Catalog Memory System." *Proc. Eastern Joint Computer Conference*, New York, , Dec. 10-12 (1956), 115-120.

[4] Lee, E.S. "Semiconductor Circuits in Associative Memories." *Proc. IEEE Pacific Computer Conference*, (Mar. 1963), 96-108.

[5] Igarashi, R., Kurosawa, T. and Yaita, T. "A 150-Nanosecond Associative Memory Using Integrated MOS Transistors." *Proc. International Solid-state Circuits Conference - Digest of Technical Papers*, (Feb. 1966), 104-105.

[6] Hanlon, A.G. "Content-Addressable and Associative Memory Systems." *IEEE Trans. Electronic Computers*, EC-15, No. 4 (Aug. 1966), 509-521.

[7] Lee, C.Y. "Intercommunicating Cells, Basic for a Distributed Logic Computer." *Proc. AFIPS 1962 Fall Joint Computer Conference*, Philadelphia, PA (Dec. 4-6, 1962), 130-136.

[8] Berg, R. O., Schmitz, H.G. and Nuspl, S.J. "PEPE-An Overview of Architecture, Operation and Implementation." *Proc. IEEE National Electronics Conference* (1972),312-317.

[9] Kaplan, A. "A Search Memory Subsystem for a General-purpose Computer." *Proc. AFIPS 1963 Fall Joint Computer Conference*, 24 (Nov. 1963), 193-200.

[10] Ewing, R.G. and Davies, P.M. "An Associative Processor." *Proc. AFIPS 1964 Fall Joint Computer Conference*, 25 (1964), 147-158.
[11] Rudolph, J.A. "A Production Implementation of an Associative Array Processor: STARAN." *Proc. AFIPS 1972 Fall Joint Computer Conference*, 41, Pt. 1, Anaheim, CA (Dec. 5-7, 1972), 229-241.
[12] Batcher, K.E. "Flexible Parallel Processing and STARAN." *Proc. 1972 WESCON Technical Papers*, Session 1 (Sept. 1972), 1/5-1 - 1/5-3.
[13] Slotnick, D.L. "Logic Per track Devices." In: J. Tou, (Ed.), *Advances in Computers*, New York, NY: Academic Press, 10 (1970), 291-297.
[14] Falkoff, A.D. "Algorithms for Parallel-Search Memories." *Journal of the ACM*, 9, No. 4 (Oct. 1962), 488-511.
[15] Estrin, G. and Fuller, R. H. "Algorithms for Content-addressable Memories." *Proc. IEEE Pacific Computer Conference* (Mar. 1963), 118-130.
[16] Fuller, R.H. and Bird, R.M. "An Associative Parallel Processor with Application to Picture Processing." *Proc. AFIPS 1965 Fall Joint Computer Conference*, 27 (1965), 105-116.
[17] Wesley, M.A., Chang, S. K. and Mommens, J.H. "A Design for an Auxiliary Associative Parallel Processor." *Proc. AFIPS 1972 Fall Joint Computer Conference*, 41, Pt. 1, Anaheim, CA (Dec. 5-7, 1972), 461-472.
[18] McKeeveer, B.T. "The Associative Memory Structure." *Proc. AFIPS 1965 Fall Joint Computer Conference*, 27 (1965), 371-388.
[19] Mundy, J.L. *et al.* "Low-cost Associative Memory." *IEEE J. Solid-State Circuits*, Vol. SC-7, No. 5, pp. 364-369, Oct. 1972.
[20] Minker, J. "An Overview of Associative or Content-addressable Memory Systems and a KWIC Index to the Literature: 1956-1970." *Computing Reviews*, 12, No. 10 (Oct. 1971), 453-504.
[21] Parhami, B. "Associative Memories and Processors: An Overview and Selected Bibliography." *Proceedings of the IEEE*, 61, No. 6 (June 1973), 722-730.
[22] Kohonen, T. *Content-addressable Memories*. 1st ed., Berlin: Springer-Verlag, 1980.
[23] Thurber, K.J. and Wald, L.D. "Associative and Parallel Processors." *Computing Surveys*, 7, No. 4 (Dec. 1975), 215-255.
[24] Yau, S.S. and Fung, H.S. "Associative Processor Architecture - A Survey." *Computing Surveys*, 9, No.1 (Mar. 1977), 3-27.
[25] Foster, C.C. *Content Addressable Parallel Processors*. New York, NY: Van Nostrand Reinhold, 1976.
[26] Batcher, K.E. "Bit-serial Parallel Processing Systems." *IEEE Trans. Computer*, C-31, No. 5 (1982), 377-383.
[27] Davis, E.W. "Application of the Massively Parallel Processor to Database Management Systems." *Proc. 1983 National Computer Conference*, (1983), 299-307.
[28] Stuttgen, H.J. *A Hierarchical Associative Processing System*, Berlin: Springer-Verlag, 1985.
[29] Fernstrom, C., Kruzela, I. and Svensson, B. *LUCAS: Associative Array Processor-design, Programming and Application Studies*. Berlin: Springer-Verlag, 1986.
[30] Waldschmidt, K. "Associative Processors and Memories - Overview and Current Status." *Proc. International Conference on Computer Technology, Systems, and Applications*. Germany: Hamburg, (May 1987), 19-26.
[31] Chisvin, L. and Duckworth, R.J. "Content-addressable and Associative Memory: Alternatives to the Ubiquitous RAM." *Computer Magazine*, 22, No. 7 (July 1989), 51-64.
[32] Special issue, "Associative Processes and Memories." *IEE Proceedings, Pt. E.*, 136, No. 5(Sept. 1989).
[33] Special issue, "Associative Memories and Processors." *IEEE Micro*, Parts 1 and 2, 12, Nos. 3 & 6 (June and Dec. 1992).
[34] Special issue, "Associative Processing and Processors." *Computer Magazine*, 27, No 11(Nov. 1994).
[35] Tavangarian, D. "Flag-algebra: A New Concept for the Realization of Fully Parallel Associative Architectures." *IEE Proceedings*, Pt. E, 136, No. 5 (Sept. 1989), 357-365.
[36] Tavangarian, D. "Flag-oriented Parallel Associative Architectures and Applications." *Computer Magazine*, 27, No. 11 (Nov. 1994), 41-52.
[37] Bursky, D. "Content-addressable Memory Does Fast Matching." *Electronic Design*, 36, No. 27 (Dec. 8, 1988), 119-121.

[38] Moors, T. and Cantoni, A. "Cascading Content-addressable Memories." *IEEE Micro*, 12, No. 3 (June 1992), 56-66.

[39] Stormon, C. *et al.* "A General-purpose CMOS Associative Processor IC and System." *IEEE Micro*, 12, No. 6 (Dec. 1992), 68-78.

[40] Weems, C.C. "Architectural Requirements of Image Understanding with Respect to Parallel Processing." *Proceedings of the IEEE*, 79, No. 4 (April 1991), 537-547.

[41] Potter, J.L. and Meilander, W.C. "Array Processor Supercomputers." *Proceedings of the IEEE*, 77, No. 12 (Dec. 1989), 1896-1914.

[42] Lea, R.M. and Jalowiecki, I.P. "Associative Passively Parallel Computers." *Proceedings of the IEEE*, 79, No. 4 (April 1991), 469-479.

[43] Yamada, H. *et al.* "Real-time String Search Engine LSI for 800-Mbit/sec LANs." *Proc. Custom Integrated Circuits Conference*, (May 1988), 21.6.1-21.6.4.

[44] Amitai, Z. "Address Filtering in FDDI LAN Bridges: the CAM Solution." *Proc. Wescon/89 Conference*, San Francisco, (Nov. 14-15, 1989), 235-239.

[45] Wyland, D.C. and Amitai, Z. "VLSI CAM Applications - an Overview." *Proc. Wescon/89 Conference*, San Francisco, (Nov. 14-15, 1989), 226-227.

[46] Wilnai, D. "A General-purpose, Expandable CAM Board." *Proc. Wescon 89 Conference*, San Francisco, (Nov. 14-15, 1989), 228-230.

[47] Gallant, J. "FDDI Routers and Bridges Create Niche for Memories." *EDN*, 37, No. 8 (April 9, 1992), 61-68.

[48] McAuley, A.J. and Francis, P. "Fast Routing Table Lookup Using CAMs." *Proc. 12th Annual Joint Conference of the IEEE Computer and Communications Societies*, San Francisco, 28 (Mar.- 1 Apr., 1993), 1382-1391.

[49] McAuley, A.J. and Cotton, C.J. "A Self-testing Reconfigurable CAM." *IEEE Journal of Solid-State Circuits*, 26, No. 3 (Mar. 1991), 257-261.

[50] Djemal, R., Mazare, G. and Michel, G. "Towards Reconfigurable Associative Architecture for High Speed Communication Operators." *Proc. IEEE Symposium and Workshop on Engineering of Computer-Based Systems*, Germany: Friedrichshafen, (Mar. 11-15, 1996), 74-79.

[51] Tamura, L.R. *et al.* "A 4-ns BiCMOS Translation-lookaside Buffer." *IEEE Journal of Solid-state Circuit*, 25, No. 5 (Oct. 1990), 1093-1101.

[52] Goksel, A.K. *et al.* "A Content Addressable Memory Management Unit with On-Chip Data Cache." *IEEE Journal of Solid-State Circuit*, 24, No. 3 (June 1989), 592-596.

[53] Venkataraman, M., Pai, M. and Canaga, S. "A 7ns Cycle, High Speed Dual Compare CAM in 0.6mm GaAs." *Proc. IEEE GaAs IC Symposium*, San Diego, CA, (Oct.29- Nov. 1, 1995), 315-318.

[54] Lucente, M.A., Harris, C.H. and Muir, R.M. "Memory System Reliability Improvement Through Associative Cache Redundancy." *IEEE Journal of Solid-State Circuits*, 26, No.3 (Mar. 1991), 404-409.

[55] Jex, J. and Baker, A. "Content Addressable Memory for Flash Redundancy." Proc. *IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing*, (May 9-10, 1991), 741-744.

[56] Kadota, H. *et al.* "An 8-kit Content-Addressable and Reentrant Memory." *IEEE Journal of Solid-State Circuits*, SC-20, No. 5 (Oct. 1985), 951-957.

[57] Uvieghara, G.A. *et al.* "An On-chip Smart Memory for a Data-flow CPU." *IEEE Journal of Solid-State Circuits*, 25, No. 1 (Feb. 1990), 84-94.

[58] Takata., H. *et al.* "A 100-Mega-access per Second Matching Memory for a Data-driven Microprocessor." *IEEE Journal of Solid-State Circuits*, 25, No. 1 (Feb. 1990), 95-99.

[59] Bergh, H., Eneland, J. and Lundstrom, L. E. "A Fault-tolerant Associative Memory with High-speed Operation." *IEEE Journal of Solid-State Circuits*, 25, No. 4 (Aug. 1990), 912-919.

[60] Su, S.Y.W. *Database Computers*, New York, NY: McGraw-Hill, 1988.

[61] Berra, P.B. and Oliver, E. "The Role of Associative Array Processors in Database Machine Architecture." *Computer Magazine*, 12, No. 3 (Mar. 1979), 53-61.

[62] Hurson, A.R. *et al.* "Parallel Architectures for Database Systems." *In: M.C. Yovits,* (Ed.), *Advances in Computers*, Boston, MA: Academic Press, Vol. 28 (1989), 107-151.

[63] Zeidler, H.Ch. "Content-addressable Mass Memories." *IEE Proceedings, Pt. E.* 136, No.5 (Sept. 1989), 351-356.

[64] Wade, J.P. and Sodini, C.G. "A Ternary Content Addressable Search Engine." *IEEE Journal of Solid-State Circuits*, 24, No. 4 (Aug. 1989), 1003-1013.

[65] Faudemay, P. and Mhiri, M. "An Associataive Accelerator for Large Databases." *IEEE Micro*, 11, No. 6 (Dec. 1991), 22-34.

[66] Mishra, P. and Eich, M.H. "Join Processing in Relational Databases." *ACM Computing Surveys*, 24, No. 1 (Mar. 1992), 63-113.

[67] Yamada, H. *et al.* "A High-speed String-search Engine." *IEEE Journal of Solid-State Circuits*, SC-22, No. 5 (Oct. 1987), 829-834.

[68] Hirata, M. *et al.* "A Versatile Data String-search VLSI." *IEEE Journal of Solid-State Circuits*, 23, No. 2 (Apr. 1988), 329-335.

[69] Takahashi, K., Yamada, H. and Hirata, M. "A String Search Processor LSI." *J. Information Processing*, 13, No. 2 (1990), 183-189.

[70] Motomura, M. *et al.* "A 1.2-Million Transistor, 33-Mhz, 20-B Dictionary Search Processor (DISP) ULSI with A 160-Kb CAM." *IEEE Journal of Solid-State Circuits*, 25, No. 5 (Oct. 1990), 1158-1165.

[71] Lee, D.L. and Lochovsky, F.L. "HYTREM - A Bybrid Text-Retrieval Machine for Large Databases." *IEEE Trans. Computers*, 39, No. 1 (Jan. 1990), 111-123.

[72] Lipovski, G. J. "A Four Megabit Dynamic Systolic Associative Memory Chip." *Journal of VLSI Signal Processing*, Vol. 4 (1992), 37-51.

[73] Katz, R.H., Gibson, G.A. and Patterson, D.A. "Disk System Architectures for High Performance Computing." *Proceedings of the IEEE*, 77, No. 12 (Dec. 1989), 1842-1858.

[74] Jalaleddine, S.M. and Johnson, L.G. "Associative IC Memories with Relational Search and Nearest-match Capabilities." *IEEE Journal of Solid-State Circuits*, 27, No. 6 (June 1992), 892-900.

[75] Weems, C.C. *et al.* "The Image Understanding Architecture." *Internat. J. Comput. Vision*, 2,No. 3 (1989), 251-282.

[76] Weems, C.C. *et al.* "The DARPA Image Understanding Benchmark for Parallel Computers." *Journal of Parallel and Distributed Computing*, 11, No. 1 (Jan. 1991), 1-24.

[77] Krikels, A. "Computer Vision Applications with the Associative String Processor." *Journal of Parallel and Distributed Computing*, 13, No. 2 (Oct. 1991), 170-184.

[78] Storer, R. *et al.* "An Associative Processing Module for a Heterogeneous Vision Architecture." *IEEE Micro*, 12, No. 3 (June 1992), 42-55.

[79] Duller, A.W.G. *et al.* "An Associative Processor Array for Image Processing." *Image and Vision Computing*, 7, No. 2 (May 1989), 151-158.

[80] Herrmann, F.P. and Sodini, C.G. "A Dynamic Associative Processor for Machine Vision Applications." *IEEE Micro*, 12, No. 3 (June 1992), 31-41.

[81] Grosspietsch, K.E. and Reetz, R. "The Associative Processor System CAPRA: Architecture and Applications." *IEEE Micro*, 12, No. 6 (Dec. 1992), 58-67.

[82] Lea, R.M. "SCAPE: A Single-chip Array Processing Element for Signal and Image Processing." *IEE Proceedings, Pt.E.*, 133, No. 3 (1986), 145-151.

[83] Jones, S.R. *et al.* "A 9-Kbit Associative Memory for High-Speed Parallel Processing Applications." *IEEE Journal of Solid-State Circuits*, 23, No. 2 (Apr. 1988), 543-548.

[84] Duller, A.W.G. *et al.* "Design of an Associative Processor Array." *IEE Proceedings, Pt.E.*, 136, No. 5 (Sept. 1989), 374-382.

[85] Shin, Y.C. *et al.* "A Special-purpose Content Addressable Memory Chip for Real-time Image Processing." *IEEE Journal of Solid-State Circuits*, 27, No. 5 (May 1992), 737-744.

[86] Hariyama, M. and Kameeyama, M. "Design of a CAM-based Collision Detection VLSI Processor for Robotics." *IEICE Trans. Electron.*, E77-C, No. 7 (July 1994), 1108-1115.

[87] Hanyu, T. *et al.* "2-Transistor-cell 4-valued Universal-literal CAM for a Cellular Logic Image Processor." *Digest IEEE International Solid-State Circuits Conferance*, San Francisco, CA (Feb. 6-8, 1997), 46-47.

[88] Parhami, B. "Performance Analysis and Optimization of Search and Selection Algorithms for Highly Parallel Associative Memories." *Proc. Fourth International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, San Jose, CA (Feb. 1-3, 1996), 217-221.

[89] Chae, S.I. *et al.* "Content-addressable Memory for VLSI Pattern Inspection." *IEEE Journal of Solid-State Circuits*, 23, No. 1 (Feb. 1988), 74-78.
[90] Doughty, D. C. *et al.* "The Use of Content-addressable Memories in the Level 2 Trigger for the CLAS Detector at CEBAF." *IEEE Trans. on Nuclear Science*, 43, No. 1 (Feb. 1996), 111-117.
[91] Panchanathan, S. and Goldberg, M. "A Content-addressable Memory Architecture for Image Coding Using Vector Quantization." *IEEE Trans. on Signal Processing*, 39, No. 9 (Sept. 1991), 2066-2078.
[92] Jones, S. "100 Mbit/s Adaptive Data Compressor Design Using Selectively Shiftable Content-addressable Memory." *IEE Proceedings, Pt. G*, 139, No. 4 (Aug. 1992), 498-502.
[93] Lee, C.Y. and Yang, R.Y. "High-throughput Data Compressor Designs Using Content-addressable Memory " *IEE Proceedings - Circuits Devices Syst.*, 142, No. 1 (Feb. 1995), 69-73.
[94] Lee, C. Y., Hsieh, P.W. and Tsai, J.M. "High Speed Median Filter Designs Using Shiftable Content-addressable Memory." *IEEE Trans. on Circuits and Systems for Video Technology*, 4, No. 6 (Dec. 1994), 544-549.
[95] Lea, R.M. "ASP: A Cost-effective Parallel Microcomputer." *IEEE Micro*, 8, No. 5 (Oct. 1988), 10-29.
[96] Shu, D. *et al.* "A Content-addressable, Bit-Serial Associative Processor." In: R.W. Broadersen, and H.S. Moscovitz, (Eds.), *VLSI signal processing, III*, New York, NY: IEEE Press, (1988), 120-128.
[97] Delgado-Frias, J.G. and Moore, W.R. (Eds.), *VLSI for Artificial Intelligence*, Boston, MA: Kluwer Academic Publishers, (1989), 93-129.
[98] Oldfield, J.V. "Logic Programs and an Experimental Architecture for Their Execution." *IEE Proceedings, Pt. E.*, 133, No. 3 (May 1986), 163-167.
[99] Ribeiro, J.C.D.F. *et al.* "Content-addressable Memories Applied to Execution of Logic Programs." *IEE Proceedings, Pt. E.*, 136, No. 5 (Sept. 1989), 383-388.
[100] Naganuma, J. *et al.* "High-speed CAM-based Architecture for a Prolog Machine (ASCA)." *IEEE Trans. on Computers*, 37, No. 11 (Nov. 1988), 1375-1383.
[101] Ogura, T. *et al.* "A 20-Kbit Associative Memory LSI for Artificial Intelligence Machines." *IEEE Journal of Solid-State Circuits*, 24, No. 4 (Aug. 1989), 1014-1020.
[102] Shankar, S. "A Hierarchical Associative Memory Architecture for Logic Programming Unification." *Proc. 5th International Conference and Symposium on Logic Programming*, (Aug. 1988), 1428-1447.
[103] Ali-Yahia, T. and Dana, M. "High-performance CAM-based Prolog Execution Scheme." *Proc. Applications of Artificial Intelligence IX -SPIE, 1468*, Orlando, Florida, (Apr. 2-4, 1991), 950-959.
[104] Dou, C. and Wu, S.M. "An Efficient Pattern Match Architecture for Production Systems Using Content-addressable Memory." *Proc. 1991 IEEE International Conference on Computer Design: VLSI in Computer and Processors*, Cambridge, MA, (Oct. 14-16, 1991), 374-378.
[105] Potter, J.L. *Associative Computing*, New York, NY: Plenum Press, 1992.
[106] Correa, N. *et al.* "An ASIC CAM Design for Associative Set Processors." *Proc. 4th Annual IEEE International ASIC Conference and Exhibit*, Rochester, NY, (Sept. 23-27, 1991), P18-3.1 - P18-3.4.
[107] Robinson, I.N. "Pattern-addressable Memory." *IEEE Micro*, 12, No. 3 (June 1992), 20-30.
[108] Kasabov, N. K. *et al.* "Model for Exploiting Associative Matching in AI Production Systems." *Knowledge-Based Systems*, 8, No. 1 (Feb. 1995), 14-20.
[109] Higuchi, T. *et al.* "The IXM2 Parallel Associative Processor for AI." *Computer Magazine*, 27, No. 11, (Nov. 1994), 53-63.
[110] Twardowski, K. "An Associative Architecture for Genetic Algorithm-based Machine Learning." *Computer Magazine*, 27, No. 11 (Nov. 1994), 27-38.

الذاكرة والحاسب الآلي الوصفي: نموذج التطابق التام

س. م. جلال الدين
مايكرو لاينير كاربوريشن، ٢٠٩٢ كنكورس درائيو
سان جوس، كاليفورنيا ٩٥١٣١، الولايات المتحدة الأمريكية

(قدّم للنشر في ١٩٩٦/١٠/١٠م؛ وقبل للنشر في ١٩٩٧/٥/٦م)

ملخص البحث . الذاكرة الوصفية تلعب دوراً هاماً في بناء هياكل الحاسب الآلي ذات الأداء العالي. إن هذه الهياكل ضرورية لتسريع عملية البحث عن المعلومات المخزونة في الحاسب الآلي. التركيب الأساسي لهذه الهياكل ذات الذاكرة الوصفية ترتكز على نموذج التطابق التام أو نموذج شبكات عصبية.

هذا البحث يركز على الذاكرة الوصفية ذات التطابق التام ويشمل استعراض أهم التطورات منذ انشاء فكرة الذاكرة الوصفية منذ أربعة عقود. يحتوي هذا البحث أيضاً على تصنيف دقيق وشامل لهذه الهياكل. وهذا التصنيف يميز بني تركيب الذاكرة الوصفية والقدرات الحسابية والتي تعتمد اعتماداً كلياً على مجال التطبيقات. ويناقش هذا البحث أخيراً أحدث التطورات والتطبيقات في علم الذاكرة الوصفية والتي تشمل: "توزيع سريع للمعلومات في شبكة الاتصالات"، "إدارة موسوعة المعلومات"، "الذكاء الاصطناعي"، "معالجة الأشكال".