Multifunction Content Addressable Memory for Parallel Speech Understanding

R. A. Cagle
212 W.16th Street
Sanford, Fl 32771

R. B. Holl
1216 S.Park Ave
Titusville, FL 32780

R. F. DeMara
Univ. of Central Florida, ECE Dept.
Orlando, Fl 32816-2450

1.0 Introduction

Content Addressable Memories (CAMs) allow considerably finer-grained parallelism than conventional shared or distributed memory multi-processors. This fine-grained "Processor-In-Memory" concept can be employed to a large degree during Semantic Network processing in support of Artificial Intelligence (AI) with specific applications in speech and natural language processing. A special-purpose CAM configuration is presented based on requirements for a nominally-sized 64K node semantic network with 8 bit-markers and 32 relationship types. Analysis for a target application shows that the extensive use of parallel Marker-Propagation and Set Theoretic Operations yields approximately 30-fold speedup over systems with standard Random Access Memories.

1.1 Background

Real-time Semantic Network Processing is an important area of AI research because of the need to close the "Semantic Gap" between natural language and digital computation. The predominate approach is to construct a small to medium-sized knowledge base which is a subset of a larger domain to allow the reasoning problem to become more tractable. Nonetheless, even relatively simple domains must contain tens of thousands of concepts, along with the many relationships between concepts, to be of practical value. This immense amount of knowledge and the marker-propagation operations which are performed on the knowledge base require a large amount of processing power. Clearly this application is well-suited for general-purpose parallel computers, but is best suited for array processors.

Specifically, a semantic network is a knowledge representation scheme that utilizes a series of *nodes* to represent concepts (each node denoting a unique concept). A pair of these conceptual nodes which are related to each other are connected by a single *link* that represents the particular type of relationship between the nodes. Additional information can be inferred through *inheritance* which allows subordinate concepts to inherit general properties from superior concepts, thus keeping the knowledge model simpler and more compact. However, there are further complications with this representation. Links need to be represented as directional, which force the number of unique representations of links to be increased. Another issue is that a single node can have many links from/to it to/from other nodes. A single search could very quickly bifurcate into hundreds of intermediate nodes, eventually reaching thousands of termir.al nodes. Generally speaking, the search operations required can be very complex search. Also, the conceptual association that the node belongs to needs to be represented. This representation is commonly called the node *color* and will be referenced during various reasoning activities which use the semantic network.[8]

1.2 Computational Requirements

All of this information needs to be represented within a formal model which can efficiently handle the large amount of data to be stored. Additional information that is utilized in the search routines gives rise to the need for associating *marker bits* with each node. This marker information is used for the following operations on the network:[4]

1) *Association*

2) *Set Theoretic Operations*
3) *Marker-Propagation*
4) *Arithmetic Operations*

For example, to perform a typical query on the network to determine a desired association between nodes, first the two nodes in question are assigned different markers. Then the association links are traversed to the conceptually adjoining node which are subsequently marked with the same marker code that was assigned to the originating node. This process is followed up through (or down through) all levels of association in the network. At the node that joins the two different marker paths, both markers are set. At the completion of the query, the node that has both markers set is identified as the desired result. Certainly, this is the most rudimentary example and the search is typically plagued by multiple common nodes. A solution to this problem is to encode even more information into the links such as probability weights which carry the strength of particular paths as floating-point numbers. This is referred to as *value passing*.

For the efficient application of parallel processing, the semantic domain can be partitioned into smaller segments distributed across each processor's local memory. This would then allow a parallelized search by each Processing Element (PE) with respect to the others (*intra-propagation parallelism*) and along the links between the partitions allotted between processors (*inter-propagation*).

1.3 Previous Work

Massively Parallel Computers such as Connection Machine's CM-2 can be extremely well suited for this task for large Semantic Domains or Paradigms. The association and set intersection gathering operations are very fast due to Single Instruction Multiple Data (SIMD) level of parallelism available to the domain.[1] The CM-2 allows a potential ratio of one concept node per PE for most large domains. This was an extreme benefit when computational explosion is possible, but for domains with a reduced level of parallelism the CM-2 wastes a significant amount of processing time on communication. Due to the design of an overwhelming number of PEs (64K), the problem of large *marker fanout* is handled well, but marker propagation's are costly between the many processing elements because of the serial communication overheads.[5]

Another supercomputer that uses a different architecture is that of the Cray type Vector Computers. The Cray X-MP was tested with the same semantic domain that the CM-2 was tested under, and the results indicate the architectural difference. The Cray was forced to develop arrays to index the concept nodes which created a large vector to process. The marker propagation routines caused additional overhead of collection of the visited nodes and developing a parsed array before the next propagation. This method forces a slower overall problem than the CM-2 when one considers the many links that need to traversed.[5]

While neither of these machines were built specifically to process a semantic problem, other machines have been. The Semantic Network Array Processor (SNAP-1) was constructed to specifically support marker propagation.[2] The SNAP-1 machine utilized 144 PEs, clustered together into 32 multiprocessor groups. Multiport memory was utilized as a storage medium for the semantic domain to limit the amount of process memory contention. For increased speed during domain node traversing, there were 20 custom instructions included to operate on the network. These high-level instructions helped to bridge the semantic gap and increase the programmability for various cases of marker-propagation. An interesting finding of the research was that while marker-propagation statements comprised approximately 17% of the dynamic instruction count, nearly 65% of the processing time was spent on them. Additionally, for the tested domain of 32K nodes, around 100 propagation's would be performed in parallel implying the need for 100 parallel processors for quick problem resolution. These processors, however, would be used for short bursts of time.

Clearly, the more available parallelism which exists among the nodes in the domain the greater the potential performance improvement. This gives rise to enhanced architecture using CAMs. CAMs contain sufficient logic to allow each individual memory cell to determine if it matches a given query. These queries generally contain a series of mask (or don't care) bits to find similar data. CAM memories can perform in parallel *read, write, search,* and *logical operations*. This reflects a trend towards distributing the processing power globally over the semantic network by associating logic with each individual CAM word. The evolution further into VLSI logic allows this memory structure to be partially feasible in size, but still grossly under the volume of storage in DRAM memory and even SRAM memories.[6]

Even with the size limitation, a distributed semantic network can still be mapped into a CAM Architecture. With the utilization of CAM, the marker-propagation can be done very quickly because it can be done in parallel rather than sequentially. This parallelism allows the processing elements to be designed more economically but still with potential improvements in the propagation algorithm. The additional speedup of this is given by the brief inclusion of multiple PEs (contained within each CAM) to the network for specific operations.

A parallel associative processor was custom built to study the speedup potential using CAMs. Called the IXM2, this machine each with a CAM bank that allowed a distributed semantic network to be completely contained.[5][6] This machine, gave consistent performance times for both marker propagation and fanout propagation, unlike the CM-2. Another benefit was the global operations that could be performed in a SIMD fashion similar to the CM-2. The performance for the IXM2 was placed between the CM-2 and Cray Super computers because of the slower processing speed, but a drawback of the CAM memories slow access and operations times.[5] However, for certain domains, the IXM2 performed better than both the CM-2 and Cray computers.

2.0 Design

CAMs have the very desirable property of being able to search all of their memory contents for a specific value during one cycle. This allows for consistently fast memory searches that are needed to process a semantic network. The output of a positive search is the memory address where the matching data is located.

2.1 Design Objectives

When designing a special purpose processor that uses CAMs, two factors must be considered. The first is the size of the semantic domain. This will determine how many bits are needed to uniquely address each node. The second is how many bits wide each word is within the CAM. This determines what other information can be stored in the CAM. For our design, we have selected a domain size of 64K nodes and a CAM with 48 bits per word. With a little study of the domain, we can see that a domain size of 64K can represent different concepts. With this in mind, we specify that the 64K domain will have:

1) A maximum of 64K nodes requiring a 16 bit address field.
2) A maximum of approximately 64K relationships between nodes.

The number of relationships will determine how many CAM chips are used, where each relationship equals one cam word.

After the domain size and CAM size have been fixed then other design parameters can be determined. These include:

1: Number of unique types of relationships in the semantic network.
2: Number of processors to be addressed.
3: Number of bit-markers required.
4: Word length for weighting values.
5: Distribution of individual data items between CAM and regular memory.
6: Field assignments and bit positions within the CAM for the above values?

2.2 CAM Bit Allocation

The 48-bit word we have to work with dwindles quickly. As shown in Figure 1, we must assign 16 bits for the source node ("NODE 1") and 16 bits for the destination node ("NODE 2") which only leaves 16 bits for all other information in each word of the CAM. Out of this last 16 bits, two marker bits

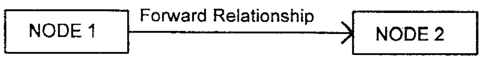

Figure 1.

are needed for the ANDing operation to determine the intersection node. This leaves 14 bits for the remaining information. Rather than using one large CAM, the CAM hierarchy is distributed over the different processors in the system. This allows a semantic network to be mapped in subsections to the different processors. If a node along a related path is mapped such that the relationship goes from a source node in one processor to a destination node in another processor, the processor number for the destination node must be known. This implies that for a *p* processor system, $\log_2 p$ bits need to be allocated for the destination processor number. The 3-bit field used below supports up to 8 processors which is sufficient for the 7 processor system we are designing for. This leaves a total of 13 bits remaining.

In a domain with 64K relationships, we must assign each relationship a unique binary number, requiring 16 bits. Since only 13 bits are available, we assign relationships on the basis of type. If we assign 5 bits to relationship types then we can have up to 32 types in the domain. This leaves the last 8 bits for markers. The mapping of a CAM word is shown in Figure 2.

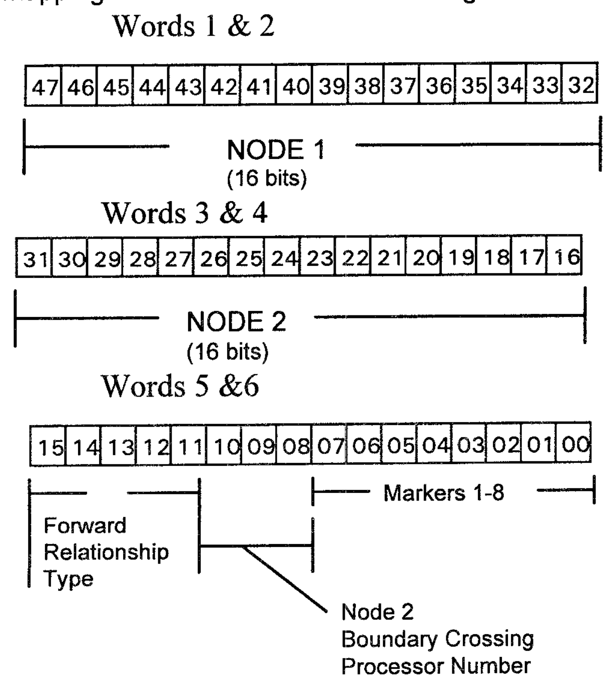

Figure 2.

Another field that is needed for the domain is the weight of the relationship. This weight is a floating-point number which consumes more bits than can be allowed for in the CAM. Therefore the weight must be stored in RAM. To do this, a special mapping must be developed to ensure that a certain weight corresponds to the correct relationship in the cam.

With the bits selected to represent the different functions they must be mapped into a CAM word. The CAM we selected as a reference uses 16 bits for data I/O so that the placement of the fields in CAM must accommodate the 16 bit borders. Since there are 16 bits each for the nodes then these can be positioned for two of the three 16-bit sub-words. In the last 16 bits the remaining information can be mapped in any order. As shown in Figure 2, the CAM word is mapped out where node 1 is the upper 16 bits, node 2 is the middle 16 bits, and the remaining information is contained in the last 16 bits. The following calculations were performed to determine the number of CAM memories needed for a workable system:

NUMBER OF CAMS REQUIRED

Given: 7 PROCESSORS
256 WORDS / CAM
64K WORDS = 65535 WORDS / DOMAIN

CALCULATIONS

$$\frac{\text{65536 WORDS / DOMAIN}}{\text{256 WORDS / CAM}} = \text{256 CAMS / DOMAIN}$$

NUMBER OF CAMS PER PROCESSOR

$$\frac{\text{256 CAMS/DOMAIN}}{\text{7 PROCESSORS/DOMAIN}} = \text{36.6 CAMS/PROCESSORS}$$

NEED EVEN NUMBER OF CAMS PER PROCESSOR

∴ 37 CAMS / PROCESSOR

WORDS OF DOMAIN PER PROCESSOR

37 CAMS/DOMAIN • 256 WORDS/CAM
= 6912 WORDS/PROCESSOR

ACTUAL TOTAL DOMAIN

37 CAMS/PROCESSOR • 7 PROCESSORS/DOMAIN
= 259 CAMS/DOMAIN

and

37 CAMS/PROCESSOR • 7 PROCESSORS/DOMAIN
• 256 WORDS/CAM
= 66304 WORDS/DOMAIN

3.0 Performance Analysis

A question arises about the validity of using a CAM network versus a RAM network to contain the domain data. The reason for this is because of the speeds of the chips available. For example, the speed of AM99C10 Advanced Micro Devices CAM is 100ns and their AM99C1341/AM99C1441 dual-port RAM is 35ns.

3.1 Access Schemes

In comparing the two types of memory, a basic guidelines and a target application must be used. To obtain meaningful results, one of the most time consuming operations in using a semantic domain was selected: the ANDing of the markers to find the intersecting node. This node will have two or more markers set by the propagation through the domain in response to a query. In the worst case application of this search, all addresses must be searched in memory.

In making a comparison between the two types of memory, some constraints must be assumed. The C99 is 256 words by 48 bits and the C1341/C1441 is 4096 words by 8 bits. To match a RAM bit pattern to the CAM bit pattern, 6 RAMs must be used in parallel to equal a 48 bit word. The comparison of the RAM is only done on the first 256 words. This is because one cam module is 256 words long. Since the loading of the CAM must be done 16 bits at a time, the ram data path must be considered to be 16 bits wide also. With a 16-bit data path we will be able to select a 2-module 16-bit bank of RAM for data while a CAM operated in 16-bit mode we can select the bit positions in the word to read from.

The searching routine under consideration will obtain the same result but differ vastly in approach. In the CAM this consists of:

1) 1 cycle to load the bit pattern.
2) 1 cycle to perform matching operation.
3) 2 cycle wait period for status completion.
4) 1 cycle to retrieve match address.
5) 1 cycle to retrieve target information.

Thus, the number of cycles needed = 6 cycles
since t_{cam} = 100ns
then total CAM time = $T_c = 6\, t_{cam}$.

The RAM search varies in the number of cycles required, as given by the following procedure:

1) 1 cycle to select bank of memory.
2) 1 cycle to retrieve word.
3) 1 cycle to compare word.
4) At maximum, steps 2 and 3 will be done 256 times.
5) 1 cycle to select bank of memory.
6) 1 cycle to get target information.

Thus, the number of cycles = 1 + 2*256 + 1 + 1
= 515 cycles

since t_{ram} = 35ns
then total RAM time = $T_r^{max} = 515\, t_{ram}$.

3.2 Upper Bound Speedup

With this information, relative performance increase of CAM over RAM usage can be developed:

$$T_r^{max} / T_c = 515 t_{ram} / 6 t_{cam} = 85.8\, t_{ram} / t_{cam}$$

Using the timing data for the selected memories, an approximate value for maximum speedup is obtained:

$$\text{speedup}^{max} = 85.8 * (35 \text{ ns} / 100 \text{ ns}) = 30.03$$

Thus when doing an exhaustive search, the CAM application is approximately 30 times faster than the RAM approach for the same data.

3.3 Lower Bound Speedup

A lower bound on speedup is obtained when only 1 word of the RAM must be searched so that steps 2) and 3) are performed only once. In this case,

$$T_r^{min} = 5\, t_{ram} \quad \text{thus,}$$
$$speedup^{min} = (5/6) * (35 \text{ ns} / 100 \text{ ns}) = .833 * .35 = 0.292$$

Since $speedup^{min}$ is fractional, a performance degradation will occur. Fortunately, even a small knowledge base typically has several hundred or more activation's in parallel so that this degradation is not encountered in practice. We are currently applying data gathered from different applications to determine average or typical values of speedup that can be expected between the two extremes.

4.0 Conclusion

The need for fast marker traffic and propagation is evident in Natural Language Processing application which activate approximately 100 nodes per propagation step for a vocabulary of a few thousand words. The serial node evaluation process is very costly in terms of machine cycles with standard memory, but a CAM appears well-suited for this. With global searches and marker analysis, all semantic nodes can be processed simultaneously to yield speedups which justify the additional cost of the CAMs for real-time processing and large knowledge bases.

An implementation drawback of the CAM approach is that the logic replicated at each word occupies a portion of the available area on the chip, thus resulting in a lower density storage per chip. Additional requirements of large word lengths to represent the semantic domain are constrained to small number of memory addresses which exist on a single chip. An extensive vendor search revealed that the largest chip available was 256 words by 48 bits. For a realistic semantic domain, a network of at least 64K nodes must exist to be practical.

A lower chip count could be obtained using 96 bits/word and 1k to 4k words per chip. To increase the parallelism of the system, this CAM would also be enhanced with several embedded ALU operations. This allows the CAM to perform mundane operations while allowing the coarse-grained PE to handle more complex operations.

References

[1] S. H. Chung and D. I. Moldovan, "Modeling Semantic Networks on the Connection Machine," *Journal of Parallel and Distributed Computing,* Feb. 1993.

[2] R. F. DeMara and D. I. Moldovan, "The SNAP-1 Parallel AI Prototype," IEEE Transactions on Parallel and Distributed Systems, September 1993.

[3] M. Evett, J. Hendler, L. Spector, "PARKA: Parallel Knowledge Representation on the Connection Machine," Univ.of Maryland Tech. Rep. UMIACS-TR-90-22, February 1990.

[4] Tatsumi Furuya, Tetsuya Higuchi, Hirohiko Kusumoto, Ken'ichi Handa, Akio Kokubu, "Architectural Evaluation of a Semantic Network Machine," in Database Machines and Knowledge Base Machines, pp. 544 - 556

[5] Testuya Higuchi, Kenichi Handa, Tatsumi Furuya, Naoto Takahashi, Akio Kokubu, "IXM2: A Parallel Associative Processor," CMU Tech. Rep., January 1991

[6] Testuya Higuchi, Kenichi Handa, Tatsumi Furuya, Hiroyuki Kusumoto, Akio Kokubu, "The Prototype of a Semantic Network Machine IXM," in *Proceedings of 1989 International Conference on Parallel Processing*, Vol. I, pg. 217 - 224

[7] Kai Hwang, Faye' A. Briggs, Computer Architecture and Parallel Processing, McGraw-Hill Publishing Company, 1984

[8] Dan I. Moldovan, Parallel Processing from Applications to Systems, Morgan Kaufmann Publishers, 1993