Extending The LANCAM® Comparand

Content-addressable memories (CAMs) have found a ready home in a variety of applications, from computer and communication networks to image processing, from data compression to data cacheing. CAMs' rapid acceptance is due to their unique architecture vis-a-vis normal random access memories (RAMs). It might be helpful to review the differences between RAM and CAM. Data is stored in a RAM in specifically addressed locations. The user retrieves the data stored in the RAM by supplying a specific address, and the RAM outputs the data previously stored. Then, to find data located somewhere in a RAM requires supplying an address, examining the output to see if it's the desired data, and if not, incrementing the address and examining the new output, and so on, until the desired data is found. Although various algorithms have been developed, out of necessity, to speed up RAM-based searches, finding data in a large RAM system can be a lengthy process because of the RAM's basic architecture.

On the other hand, data can be stored in the LANCAM, using a Move to Next Free Address instruction, whereby the data is copied from the input Comparand register to the first empty location anywhere in the CAM system. The LAMCAM cascading feature means that the system Next Free address will always be found in the first device with an empty location, no matter how many devices are in the string. Data can thus be stored in the LAMCAM in a rather random fashion, without the user having to maintain location information in a separate pointer RAM. When it comes time to find the data, a specific address is not required as in a RAM, because the data itself is input to the CAM's Comparand register, and the CAM automatically performs a compare against every location in memory simultaneously. Each 64-bit wide memory location has its own match line, and if any bit in the stored entry is a mismatch with the same bit in the Comparand register, the match line is negated. All 1024 match lines feed a priority encoder that determines which entry in memory is the highest-priority match, asserts the Match flag to indicate the presence of the desired data in memory, and stores the location of the highest-priority match in the Status register. In the case of multiple matches, the Multiple Match bit in the Status register is set.

In many applications, only the presence or absence of a match is needed as an input to some decision making algorithm. In other applications, additional data associated with the stored data is also needed, as in a translation table for example. In this case, the LAMCAM can be configured into CAM and RAM partitions, so that each entry stored in the CAM has Associated data stored in RAM at the same memory location. If additional Associated data is needed, the location address of the highest-priority match can be read out and used to point to external RAM of any desired width where additional data is stored.

Extending the depth of a database stored in the LAMCAM is done easily, using the Match flag and Full Flag cascading systems. Each device has a Match In and Full In signal that allows match and full information from the previous device to affect the Match Flag output and Full Flag output, resulting in a System Match flag and System Full flag. These cascading systems also provide the pointers to the specific device with the system highest-priority match and the system Next Free address, for efficient read-out, updating, or storage based on the compare results.

Extending the width of an entry and comparand is done in a different fashion due to the impossibilities of connecting 1024 match lines between devices. Several methods have been developed that illustrate the relative ease of extending the LAMCAM word width, bringing the CAM's advantage of a rapid database search to applications that require a comparand wider than 64 bits, while still providing associated data.

Method 1

In Method 1, half the comparand is in one CAM and the other half is in a second CAM, as shown in Figure 1. After initialization, both CAMs are empty, and as entries are stored, both halves are stored at the same location in both CAMs, either using an absolute address command, such as SPD M@aaaH,V, which addresses a specific memory location for data write cycles; or a relative address command, such as SPD M@[AR],V, which allows a DMA type data transfer, using the incrementing feature of the address register (AR); or by copying data from the Comparand register to the first empty location in the CAMs using a MOV NF,CR,V command. When a compare is later performed between the unknown data in the Comparand registers and the contents of both CAMs, if only one CAM reports a match, the unknown data is not present in the

CAMs. If both report a match, both CAMs must have a match at the same memory location to be valid. This can be found by examining the Status registers of both CAMs to see if the match addresses are the same. If not, the Multiple Match flag in the upper portion of the Status registers can be examined to see if there are other matches to be examined, and in which CAM they are located. Then the match in the CAM with the lowest match address is set to skip using a VBC HM,S command, and a CMP V command issued, which will find the location of the next higher match. Then the status register of that CAM can be examined to see if the match address agrees with the match address in the other CAM. After all matches have been examined, or the valid match found, all the skipped entries can be returned to valid using a CMP S through an opaque mask followed by a VBC ALM,V instruction. This method is extendable to more than two CAMs by continuing the Status register checking process.

Associated data is easily accommodated in Method 1 by setting the device configuration into CAM and RAM partitions, because only the CAM portions will enter into comparison with the Comparand register. After a valid match has been found between the two CAMs, the associated data can be read out from both CAMs using an SPS M@HM command and performing data reads from memory.

Method 2

Another approach with multiple CAMs allows random storage of entries by putting the most random portion of the entry in the first CAM, and storing the remaining portions in the upper segments of succeeding CAMs along with the address of the random portion in the first CAM in the lowest segment. In the case of a match in the first CAM, the match location can be read out of its Status register and inserted into segment 0 of the Comparand register of the other CAMs, and a comparison on all segments performed that includes the match address. If all CAMs report a match, the valid match has been found. This method will support up to 64,000 entries of any desired width, while using only one 16-bit segment of the succeeding CAMs for storage of the entry location in the first CAM.

Method 3

Method 3 extends the comparand width as long as desired useing only one CAM, as shown in Figure 2, but concatenates adjacent entries using validity bits to identify successive entries. In this method, the first entry (e.g., Entry 1A) in a long comparand is stored as Valid, but the succeeding entries in that comparand (e.g., Entry 1B and Entry 1C) are stored as RAM-only. If a match is found, the Status register is examined to find the match address and whether multiple matches exist. Then the contents of the memory at the match location + 1 are read out, and an external compare is performed against the next portion of the unknown. If a match is found, the contents of the memory at the match location +2 can be read out, and another compare executed. If a match is not found in these

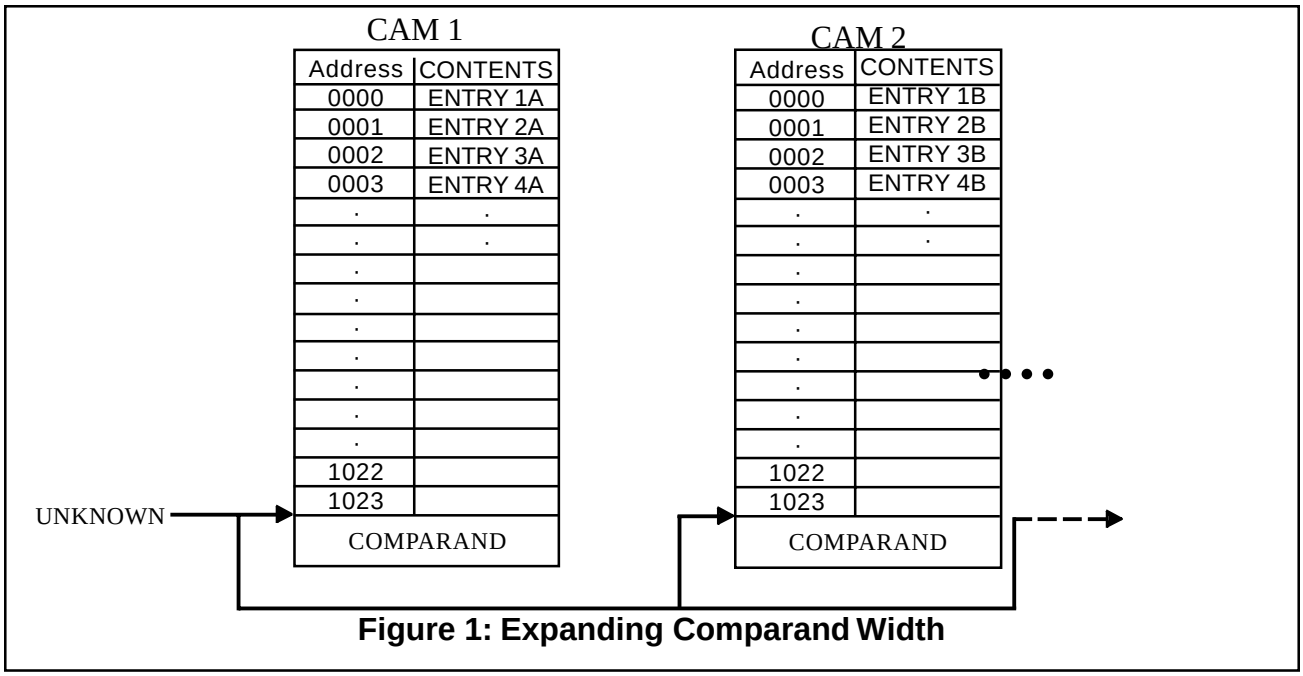

Figure 1: Expanding Comparand Width

external compares, the match location (e.g., Entry 1A) in the CAM can be set to "SKIP" and a CMP V command issued to the CAM to find the next matching location, and the external compare process continued until a match is found with the entire unknown. After completion, the "SKIPPED" entries are returned to "VALID" as shown in Method 1.

Method 4: Another single CAM approach tags each succeeding stored entry in a long Comparand with its sequence number. The first entry is tagged "1", the second entry is tagged "2", the third entry is tagged "3", and so forth. After the first match is found, the second portion of the unknown is entered into the Comparand register along with the tag for "2", and if a match is found, the Status register is examined to see if the match address is equal to the previous match address + 1. If it is, then the third portion of the unknown can be loaded into the Comparand with the tag for "3", and so forth. If the match address is not an increment from the previous address, that means the match has failed, and the first matching location should be set to "SKIP" and a CMP V issued to find the next higher match, as discussed above.

Methods 3 and 4 also allow for Associated data with long comparands comprising a fixed number of CAM-able entries and a fixed number of Associated data entries, by storing the Associated data in memory after the CAM-able entries, and setting them as RAM entries so they don't enter into comparisons, as shown in Figure 3. Then after a valid match has been found, the desired number of RAM entries can be read out from the memory starting at the last match address + 1.

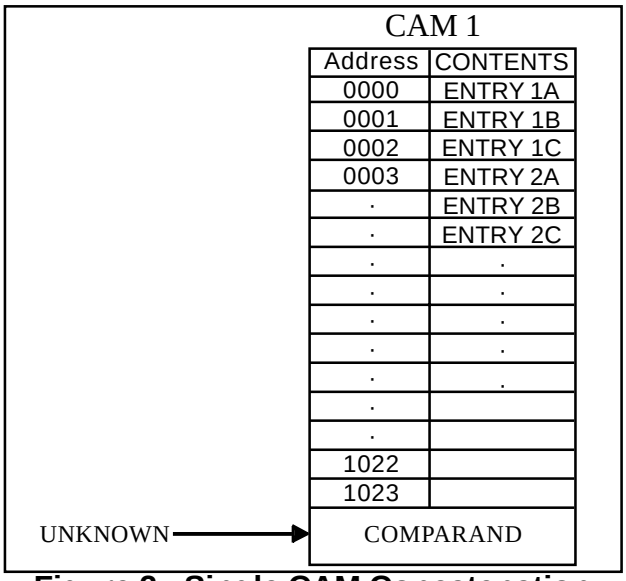

Figure 2: Single CAM Concatenation

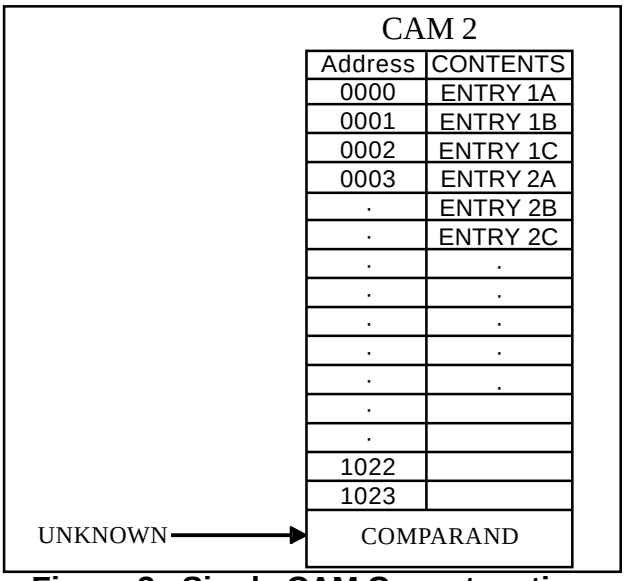

Figure 3: Single CAM Concatenation with Associated Data

NOTES

MUSIC Semiconductors Agent or Distributor:

MUSIC SEMICONDUCTORS

http://www.music-ic.com
email: info@music-ic.com

USA Headquarters
MUSIC Semiconductors
254 B Mountain Avenue
Hackettstown, New Jersey 07840
USA
Tel: 908/979-1010
Fax: 908/979-1035
USA Only: 800/933-1550 Tech. Support
888/226-6874 Product Info.

Asian Headquarters
MUSIC Semiconductors
Special Export Processing Zone 1
Carmelray Industrial Park
Canlubang, Calamba, Laguna
Philippines
Tel: +63 49 549 1480
Fax: +63 49 549 1023
Sales Tel/Fax: +632 723 62 15

European Headquarters
MUSIC Semiconductors
Torenstraat 28
6471 JX Eygelshoven
Netherlands
Tel: +31 45 5462177
Fax: +31 45 5463663