Content-Addressable memory (CAM) and its network applications

Midas Peng and Sherri Azgomi
Altera International Ltd.

Introduction
Most memory devices store and retrieve data by addressing specific memory locations. As a result, this path often becomes the limiting factor for systems that rely on fast memory accesses. The time required to find an item stored in memory can be reduced considerably if the item can be identified for access by its content rather than by its address. A memory that is accessed in this way is called content-addressable memory or CAM. CAM provides a performance advantage over other memory search algorithms, such as binary or tree-based searches or look-aside tag buffers, by comparing the desired information against the entire list of pre-stored entries simultaneously, often resulting in an order-of-magnitude reduction in the search time.

CAM is ideally suited for several functions, including Ethernet address lookup, data compression, pattern-recognition, cache tags, high-bandwidth address filtering, and fast lookup of routing, user privilege, security or encryption information on a packet-by-packet basis for high-performance data switches, firewalls, bridges and routers. This article discusses several of these applications as well as hardware options for using CAM.

Basics of CAM
Since CAM is an outgrowth of Random Access Memory (RAM) technology, in order to understand CAM, it helps to contrast it with RAM. A RAM is an integrated circuit that stores data temporarily. Data is stored in a RAM at a particular location, called an address. In a RAM, the user supplies the address, and gets back the data. The number of address line limits the depth of a memory using RAM, but the width of the memory can be extended as far as desired. With CAM, the user supplies the data and gets back the address. The CAM searches through the memory in one clock cycle and returns the address where the data is found. The CAM can be preloaded at device startup and also be rewritten during device operation. Because the CAM does not need address lines to find data, the depth of a memory system using CAM can be extended as far as desired, but the width is limited by the physical size of the memory.

CAM can be used to accelerate any application requiring fast searches of data-base, lists, or patterns, such as in image or voice recognition, or computer and communication designs. For this reason, CAM is used in applications where search time is very critical and must be very short. For example, the search key could be the IP address of a network user, and the associated information could be user’s access privileges and his location on the network. If the search key presented to the CAM is present in the CAM’s table, the CAM indicates a ‘match’ and returns the associated information, which is the user’s privileges. A CAM can thus operate as a data-parallel or Single Instruction/Multiple Data (SIMD) processor.

Applications
CAM can be used to accelerate any applications ranging from local-area networks, database management, file-storage management, pattern recognition, artificial intelligence, fully associative and processor-specific cache memories, and disk cache memories. Although CAM has many applications, it is particularly well suited to perform any kind of search operations. Following is a discussion of some of these applications.

Data Compression
Data compression removes redundancy that resides in a given piece of information, producing an equivalent but shorter message. CAM is well suited for data compression because the movement of packets through local- or wide-area networks require some form of address translation. Since a good portion of a compression algorithm’s time is spent searching and maintaining these data structures, replacing them with a hardware search engine can greatly increase the throughput of the algorithm.

In a data compression application, CAM lookup is performed after each word of the original data is presented (see Figure 1). If the code corresponding to the word bit pattern in the input register is found, then the appropriate symbol or token is output and input register is flushed. If the code is not found in the CAM, then another word is shifted in. A CAM will generate a result in a single transaction regardless of table size or length of search list. This makes CAM an ideal candidate for data compression schemes that use sparsely populated tables as part of their algorithm.
Network Switch
CAM is used in switch applications to extract and process the address information from incoming data packets. In order to switch the packet to the correct outgoing port, CAM compares the destination address with a table of addresses stored within. For example, a CAM might store Ethernet address and switch port numbers. The CAM compares the received data against the table that has been stored within, and if the comparison yields a match, then the port identification is given and routing control forwards the packet to the correct port or address (see figure 2).

IP Filter
An IP filter is a security feature that restricts unauthorized access to LAN resources or restricts traffic on a WAN link (IP traffic that goes through the router). IP filters can be used to restrict the types of internet traffic that are permitted to access a LAN, and LAN workstations can be restricted to specific internet-based applications (such as e-mail). CAM can be used to work as a filter which blocks all access except for those packets that are given explicit permissions according to the rules of the IP filter. In this application, CAM compares the packet being routed to the port against the IP filter rules. When a match is found, the packet is either permitted or denied, as shown in Figure 3.

ATM Switch
CAMs can be used in Asynchronous Transfer Mode (ATM) switching network components as a translation table. Since ATM networks are connection-oriented, virtual circuits need to be set up across them prior to any data transfer. There are two kinds of ATM virtual circuits: Virtual Path (identified by a virtual path identifier or VPI) and Channel Path (identified by a channel path identifier or VCI). VCI/VPI values are localized; each segment of the total connection has unique VPI/VCI combinations. Whenever an ATM cell travels through a switch, its VPI/VCI value has to be changed into the value used for the next segment of connection. This process is called VPI/VCI translation. Since speed is an important factor in ATM network, the speed at which this translation occurs forms a critical part of the network’s overall performance.

CAM can act as an address translator in an ATM switch and perform the VPI/VCI translation very quickly. During the translation process, the CAM takes incoming VPI/VCI values in ATM cell headers and generates addresses that access data in an external RAM (since standard CAM architectures cannot support the required capacity, a CAM/RAM combination enables the realization of multi-megabit translation tables with fully-parallel search capability). VPI/VCI fields from the ATM cell header are compared against a list of current connections stored in the CAM array. As a result of the comparison, CAM generates an address that is used to access an external RAM where VPI/VCI mapping data and other connection information is stored. The ATM controller modifies the cell header using the VPI/VCI data from the RAM, and the cell is sent to the switch. This application is shown in figure 4.

Memory Mapping
In a system that utilizes a dynamic memory map, CAM can be used to store memory addresses for quicker access. For example, in a PCI system, a single PCI device may contain up to six memory spaces allocated in system memory. The exact location of these spaces is determined on power-up, and their starting locations are written into the PCI interface’s six Base Address Registers (BAR). When a master interface requests access to a PCI device’s memory or I/O location, a CAM can be used to quickly match the request to the address in memory, as shown in Figure 5.
Hardware Solutions for CAM

Two types of hardware solutions exits for CAM: individual (or discrete) components and on-chip implementations. Individual CAM components have been available since the late 1980s as relatively specialized memory devices. Typically, whenever CAM was required, a designer would add the component to the PCB board. This approach allows a designer to take advantage of the features of the CAM, but takes additional board space. Furthermore, the overall system performance is reduced by on/off-chip delays of the CAM and the system device. Performance is also affected by board delays.

On-chip implementations of CAM in the form of embedded functions have only recently been made available. One such example is the CAM capability offered in the latest high-density programmable logic devices (PLDs). The advantage of this type of CAM is not only the board space efficiency, but also it is the increase in performance. This performance increase is due to elimination of on/off-chip delay along with the smaller process on on-chip CAMs. In these CAM-capable PLDs, an on-board structure called the Embedded System Block (ESB) can implement CAM. Figure 6 shows the CAM block diagram in one such PLD.

![Figure 6: CAM Block Diagram in programmable logic](image)

Each ESB in this device supports a 1K-bit CAM (32 words of 32 bits each). Wider or deeper CAM can be implemented by combining multiple CAMs using device logic resources. The associated design development software, automatically performs this operation of combining ESBs automatically if the user desires larger CAMs. The number of ESBs in a PLD ranges from 26 to 264. In order to make a deeper or wider CAM in one of these PLDs, the output of multiple ESBs can be cascaded together (we can cascade as many ESB as exist in a device to create a bigger CAM). Both the encoded and unencoded output implementations of ESB can be used to create a deeper and wider CAM.

The CAM can be pre-loaded with data during configuration of the PLD, or it can be written during system operation. In most cases, two clock cycles are required to write each word into CAM. This PLD CAM supports writing “don’t-care” bits into words of the memory. The don’t-care bit can be used as a mask for CAM comparisons; any bit set to don’t-care has no effect on matches. When don’t-care bits are used, a third clock cycle is required to write words into the CAM. The output of the CAM can be encoded or unencoded. When encoded, the ESB outputs an encoded address of the data location. Encoded output is better suited for designs that ensure no duplicate output. If duplicate data is written into multiple locations, unencoded output is required. ESB uses its 16 outputs and reads the outputs in two clock cycles. For system designs that incorporate CAMs, the PLD on-chip CAM generally offers better system performance than discrete CAM devices because of the elimination of on-chip and off-chip delays.

Conclusion

For any application that requires a fast memory search, CAM can provide a solution. There are many manufacturers that provide discrete CAM components, such as Music Semiconductor, Kawasaki LSI, Netlogic Microsystems Inc and Lara Technology. As CAM becomes more popular and system-on-a-chip design becomes more common, intellectual property and embedded versions of CAM such as that offered in the most advanced PLDs have also become available. In the coming months, CAM will enjoy increased popularity, particularly with the growing number of communications applications that can utilize its capabilities.

Author’s contact details

Midas Peng
Altera International Ltd.
Phone: 886-3-564-3846
Fax: 886-3-564-3847
Email: mpeng@altera.com