VLSI DESIGN AND IMPLEMENTATION OF CONTENT ADDRESSABLE MEMORY WITH RECONFIGURABLE MEMORY BANKS

[1]Ilavarasi S, [2]Karthika A, [3]Senthilkumar K K

[1]B.E, ECE, Prince Shri Venkateshwara Padmavathy Engineering College, TN, India
[2]B.E, ECE, Prince Shri Venkateshwara Padmavathy Engineering College, TN, India
[3]M.E, Ph.D., ECE, Prince Shri Venkateshwara Padmavathy Engineering College, TN, India

ABSTRACT

Content Addressable Memory (CAM) is a type of memory that can be accessed using its contents rather than an explicit address. In the existing design, a low-power content-addressable memory (CAM) employing a new algorithm for associativity between the input tag and the corresponding address of the output data was presented. The presented architecture was based on a recently developed sparse clustered network using binary connections that on-average eliminates most of the parallel comparisons performed during a search. Given an input tag, the proposed architecture computes a few possibilities for the location of the matched tag and performs the comparisons on them to locate a single valid match. CAM is popular parallel matching circuits. They provide the capability, in hardware, to search a table of data for a matching entry. This functionality is a high performance alternative to popular software-based searching schemes. CAMs are typically found in embedded circuitry where fast matching is essential. This method is to presents CAM implementation using run-time reconfiguration. This approach produces CAM circuits that are smaller, faster and more flexible than traditional approaches.

1. INTRODUCTION

A CONTENT-addressable memory (CAM) is a type of memory that can be accessed using its contents rather than an explicit address. In order to access a particular entry in such memories, a search data word is compared against previously stored entries in parallel to find a match. Each stored entry is associated with a tag that is used in the comparison process. Once a search data word is applied to the input of a CAM, the matching data word is retrieved within a single clock cycle if it exists. This prominent feature makes CAM a promising candidate for applications where frequent and fast look-up operations are required, such as in translation look-aside buffers (TLBs) [1], [2], network routers [3], [4], database accelerators, image processing, parametric curve extraction [5], Hough transformation [6], Huffman coding/decoding [7], Content Addressable Memories or CAMs are a class of parallel pattern matching circuits. In one mode, these circuits operate like standard memory circuits and may be used to store binary data. Unlike standard memory circuits, however, a powerful match mode is also available. This match mode permits all of the data in the CAM device to be searched in parallel.

While CAM hardware has been available for decades, its use has typically been in niche applications, embedded in custom designs. Perhaps the most popular application has been in cache controllers for central processing units. Here CAMs are often used to search cache tags in parallel to determine if a cache\hit" or \miss" has occurred. Clearly in this application performance is crucial and parallel search hardware such as a CAM can be used to good effect. A second and more recent use of CAM hardware is in the networking area. As data packets arrive into a network router, processing of these packets typically depends on the network destination address of the packet. Because of the large number of potential addresses, and the increasing performance demands, CAMs are beginning to become popular in processing network address information.

1.1THE RECONFIGURABLE CONTENT ADDRESSABLE MEMORY(RCAM)

The Reconfigurable Content Addressable Memory or RCAM makes use of runtime reconfiguration to efficiently implement a CAM circuit. Rather than using the FPGA flip flops to store the data to be matched, the RCAM uses the FPGA Look Up Tables or LUTs. Using LUTs rather than ip-ops results in a smaller, faster CAM. The approach uses the LUT to provide a small piece of CAM functionality. In Figure 1.2, a LUT is loaded with data which provides a \match 5" functionality. That is, whenever the binary encoded value \5" is sent to the four LUT inputs, a match signal is generated. Note that using a LUT to implement CAM functionality, or any functionality for that matter, is not unique. An N-input LUT can implement any arbitrary function of N inputs, including a CAM.

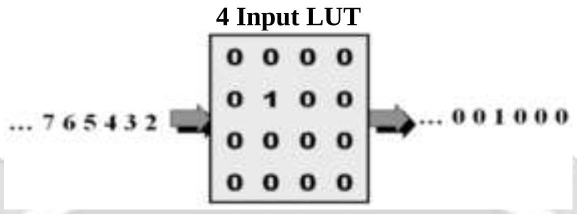


Figure 1.1 Using a LUT to match 5.

Because a LUT can be used to implement any function of N variables, it is also possible to provide more flexible matching schemes than the simple match described in the circuit in Figure 1.2. In Figure1.3, the LUT is loaded with values which produce a match on any value but binary \4". This circuit demonstrates the ability to embed a mask in the configuration of a LUT, permitting arbitrary disjoint sets of values to be matched, within the LUT. This function is important in many matching applications, particularly networking.

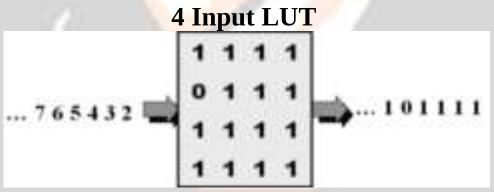


Figure 1.2 Using a LUT to match all inputs except 4.

This approach can be used to provide matching circuits such as match all or match none or any combination of possible LUT values. Note again, that this arbitrary masking only applies to a single LUT. When combining LUTs to make larger CAMs, the ability to perform such masking becomes more restricted.While using LUTs to perform matching is a powerful approach, it is some what limited when used with traditional design tools. With schematics and HDLs, the LUT contents may be specified, albeit with some difficulty. And once specified, modifying these LUTs is difficult or impossible. However, modification of FPGA circuitry at run-time is possible using a run-time reconfiguration tool such as JBits. JBits permits LUT values, as well as other parts of the FPGA circuit, to be modified arbitrarily at run time and in-system.

An Application Program Interface (API) into the FPGA configuration permits LUTs, for instance, to be modified with a single function call. This, combined with the partial reconfiguration capabilities of new FPGA devices such as Virtex (tm)permit the LUTs used to build the RCAM to be easily modified under software control, without disturbing the rest of the circuit. Finally, using run-time reconfiguration software such as JBits, RCAM circuits may be dynamically sized, even at run-time. This opens the possibility of not only changing the contents of the RCAM during operation, but actually changing the size and shape of the RCAM circuit itself. This results in a situation analogous to dynamic memory allocation in RAM. It is possible to \allocate" and \free" CAM resources as needed by the application.

1.2 An RCAM Example

One currently popular use for CAMs is in networking. Here data must be processed under demanding real-time constraints. As packets arrive, their routing information must be processed. In particular, destination addresses, typically in the form of 32-bit Internet Protocol (IP) addresses must be classified. This typically involves some type of search. Current software based approaches rely on standard search schemes such as hashing. While effective, this approach requires a powerful processor to keep up with the real-time demands of the network. Offloading the computationally demanding matching portion of the algorithms to external hardware permits less
powerful processors to be used in the system. This results in savings not only in the cost of the processor itself, but in other areas such as power consumption and overall system cost.

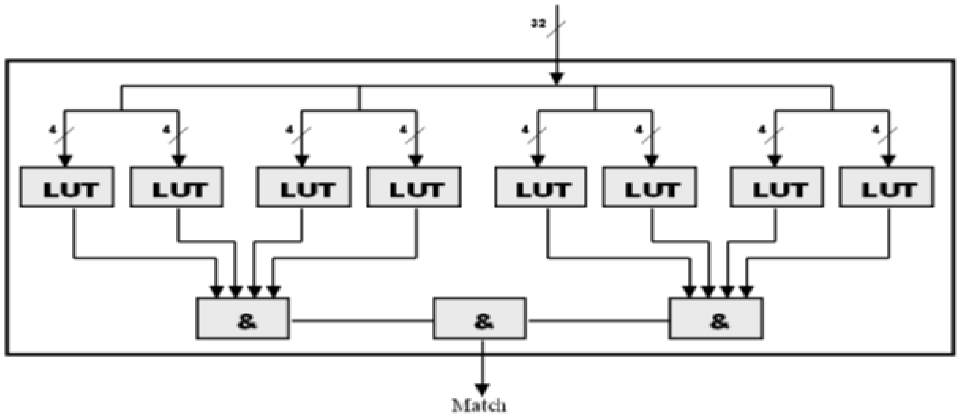


Figure 1.3 Matching a 32-bit IP header.

In addition, an external CAM provides networking hardware with the ability to achieve packet processing in essentially constant time. Provided all elements to be matched it in the CAM circuit, the time taken to match is independent of the number of items being matched. This provides not only good scalability properties, but also permits better real-time analysis. Other software based matching schemes such as hashing are data-dependent and may not meet real time constraints depending on complex interactions between the hashing algorithm and the data being processed. CAMs suffer no such limitations and permit easy analysis and verification. Figure 1.4 shows an example of an IP Match circuit constructed using the RCAM approach.

Note that this example assumes a basic 4-input LUT structure for simplicity. Other optimizations, including using special-purpose hardware such as carry chains are possible and may result in substantial circuit area savings and clock speed increases. This circuit requires one LUT input per matched bit. In the case of a 32 bit IP address, this circuit requires 8 LUTs to provide the matching, and three additional 4-input LUTs to provide the AND for the MATCH signal. An array of this basic 32-bit matching block may be replicated in an array to produce the CAM circuit. Again, note that other non-LUT implementations for generating the MATCH circuit are possible. Since the LUTs can be used to mask the matching data, it is possible to put in\match all" conditions by setting the LUTs to all ones. Other more complicated masking is possible, but typically only using groups of four inputs. While this does not provide for the most general case, it appears to cover the popular modes of matching.

2. PROPOSED SYSTEM

The CAM core cells are arranged into four horizontal words, each five bits long. Core cells contain both storage and comparison circuitry. The search lines run vertically in the figure and broadcast the search data to the CAM cells. The matchlines run horizontally across the array and indicate whether the search data matches the row's word. An activated matchline indicates a match and a deactivated matchline indicates a non-match, called a mismatch in the CAM literature. The matchlines are inputs to an encoder that generates the address corresponding to the match location. The Table 3.1 shows the simplified routing table, how the search address location matches the output port in the RAM. It is explained with an example in the Figure 3.2.

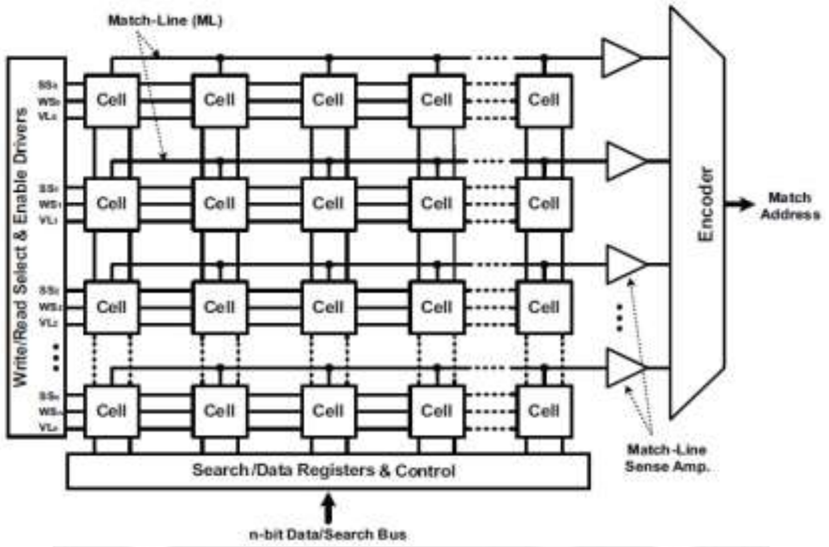
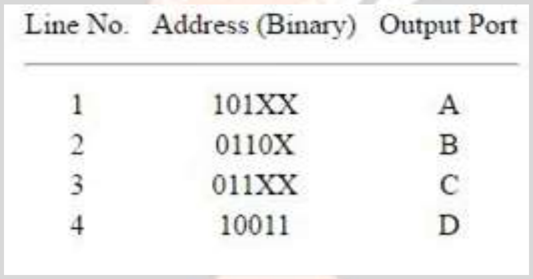


Figure 2.1 CAM Architecture

Table 2.1 Simplified routing table



Line No.	Address (Binary)	Output Port
1	101XX	A
2	0110X	B
3	011XX	C
4	10011	D

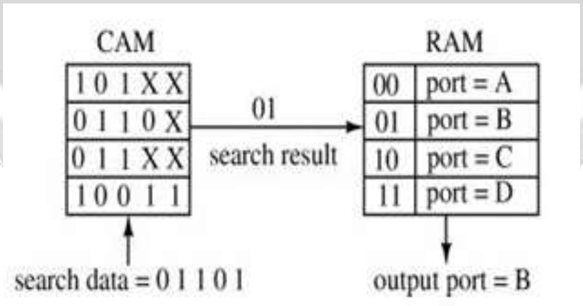


Figure 2.2 Address lookup with CAM/RAM

2.1 Block diagram explantaion

In the proposed system, a high speed reconfigurable CAM is designed. Content address memory is nothing but the normal memory architecture in which the logical modules enable the user to retrieve the information stored in the memory within few nano seconds (few clock cycles) through a systematic search algorithm based on memory index data. The data stored in the memory under goes a process of encryption before storing takes place as shown in the Figure 3.3. The data is purely packed in the encryptor end where the data un-packed at the decryptor end as well.

The encrypted information is stored in the CAM. The data can be accessed using the index bits for fast retrieval. The size of the memory banks can be varied with respect to the low power reconfiguration bits available through a Switch outside the FPGA. Hence some of low power techniques such as clock gating and power gating is being used to improve the architecture performance in terms of time. The Figure 3.4 shows the working of CAM with blocks of address counter, data packing and search content.

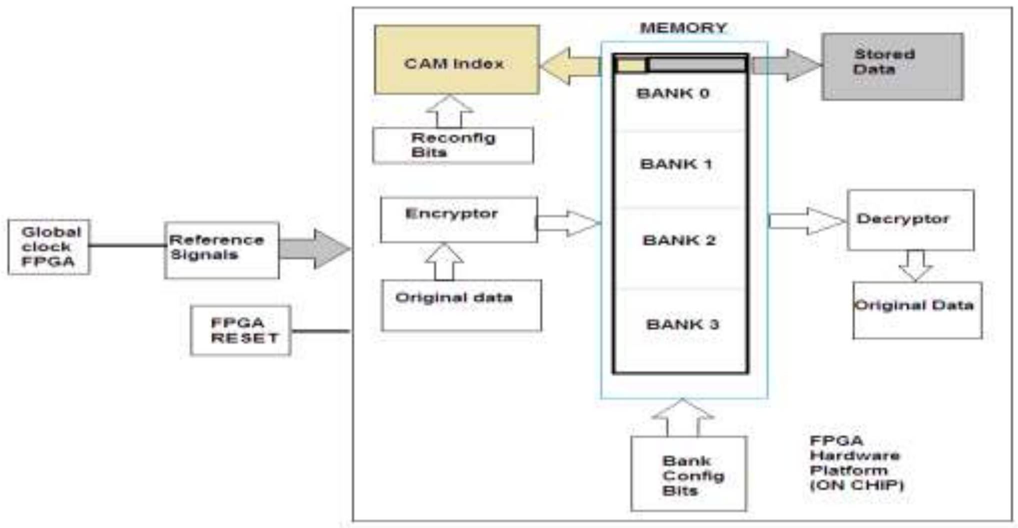


Figure 2.3 Block diagram of CAM

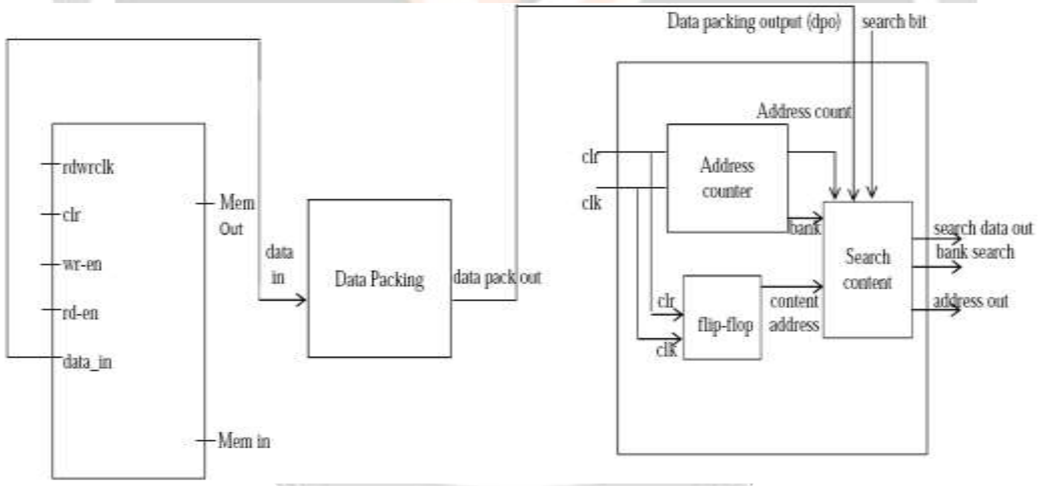


Figure 2.4 working of CAM

1. Data packing

Data packing will contains loads of information (data) and stores it with in the pack were the data can be compressed to reduce its file size.Thus with the stored in the memory the compare operation of CAM can be performed against the search data.

2. Flip-flop

Flip-flops are used as data storage elements. A flip-flop stores a single bit (binary digit) of data; one of its two stages represent a “one” and the other represents a “zero”. Flip-flop can be either simple(transparent or opaque) or clocked(synchronous or edge triggered).

3. Address counter

A counter which increments an initial memory address as a block of data is being transferred into the memory locations indicated by the counter. The increasing order of the counter depends upon the rising and falling edge of the clock.

4. Search content

Search content will contains the index of the content which was stored in the memory, these index will helps in finding the matched content during the compare operation of the CAM.

The bank search, address out and search data will provide the output indicating that the address location and in which bank the searched data was found.

The memory in (write) and memory out (read) represents the read and write operation of the memory.

5. Memory Bank

As shown in Figure 3.5 the memory bank is a logical unit of storage in electronics, which is hardware-dependent.

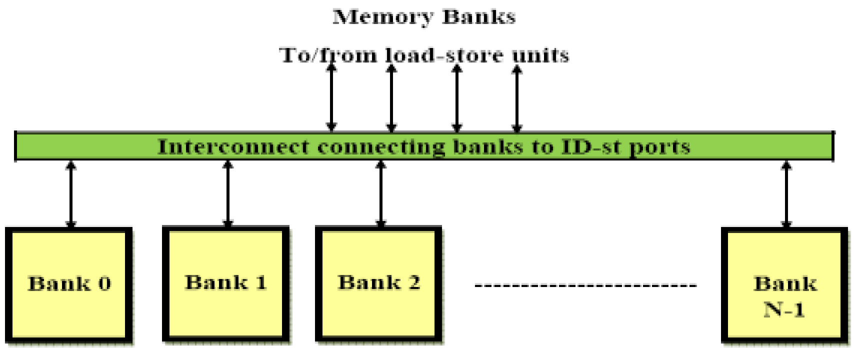


Figure 2.5 Memory banks

The memory bank may be determined by the memory controller along with physical organization of the hardware memory slots. In a typical synchronous dynamic random-access memory (SDRAM) or double data rate synchronous dynamic random-access memory (DDR SDRAM), a bank consists of multiple rows and columns of storage units, and is usually spread out across several chips. In a single read or write operation, only one bank is accessed, therefore the number of bits in a column or a row, per bank and per chip, equals the memory bus width in bits (single channel). The size of a bank is further determined by the number of bits in a column and a row, per chip, multiplied by the number of chips in a bank. Some computers have several identical memory banks of RAM, and use bank switching to switch between them. Harvard architecture computers have (at least) two very different banks of memory, one for program storage and other for data storage.

Reconfigurable computing is a computer architecture combining some of the flexibility of software with the high performance of hardware by processing with very flexible high speed computing fabrics like field-programmable gate arrays (FPGAs). The principal difference when compared to using ordinary microprocessors is the ability to make substantial changes to the datapath itself in addition to the control flow. On the other hand, the main difference with custom hardware, i.e. Application-Specific Integrated Circuits (ASICs) is the possibility to adapt the hardware during runtime by "loading" a new circuit on the reconfigurable fabric.

3. RESULTS AND DISCUSSION

VERILOG HDL is a hardware description language (HDL). A hardware description language is a language used to describe a digital system, for example, a computer or a component of a computer. One may describe a digital system at several levels. For example, an HDL might describe the layout of the wires, resistors, and transistors on an integrated circuits (IC) chip, i.e., the gate level. An even high level describes the register and the transfer of vector of information between the registers. This is called as Register Transfer Level (RTL). VERILOG supports at all these levels. However, this handout focuses on only the portion of VERILOG support the RTL level.

- **Loading operation**

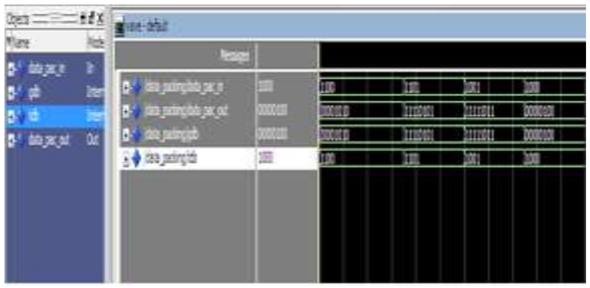


Figure 3.1 Data packing

The Figure 3.1 explains about how the data gets packed or loaded into the memory with the help of counter for each data to be packed for storage in order to perform the compare operation

- **Memory operation**

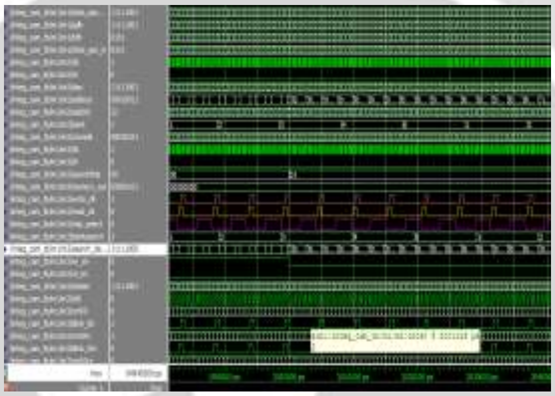


Figure 3.2 Memory operation

The Figure 3.2 explains about how the data gets stored into the memory with help of read and write operation .The read operation helps to retrieve the data from the memory where it was stored and write operation helps to write the data into the memory

- **Search operation**

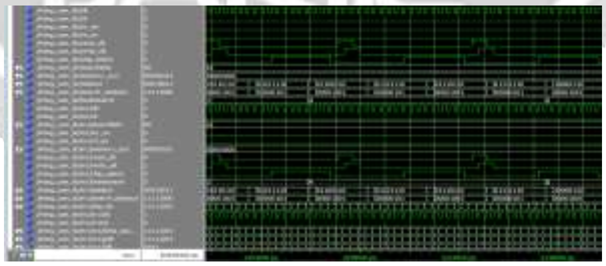


Figure 3.3 Integration operation

The Figure 3.3 gives the integration result of data packing and memory module and this is considered as overall simulation and it shows how the search operation and compare operation is done in CAM.

4. CONCLUSION

A FPGA architecture is presented for computing operation. The components required for performing the operation are being reduced in the proposed system CAM is suitable for low-power applications, where frequent and parallel look-up operations are required, which is connected to several independently compare-enabled CAM sub-blocks, some of which are enabled once a tag is present Simple and fast updates can be achieved without retraining the network entirely. With optimized lengths of the reduced-length tags, CAM eliminates most of the comparison operations. In future the proposed method will enhance for large memory size with N number of memory banks

.

6. REFERENCES

1. C.-C. Wang, C.-J. Cheng, T.-F. Chen, and J.-S. Wang, “An adaptively dividable dual-port BiT CAM for virus-detection processors in mobiledevices,” IEEE J. Solid-State Circuits, May 2009.
2. H. Noda et al., “A cost-efficient high-performance dynamic TCAMwith pipelined hierarchical searching and shift redundancy architecture,”IEEE J. Solid-State Circuits, Jan. 2005.
3. P.-T. Huang and W. Hwang, “A 65 nm 0.165 fJ/Bit/Search 256 × 144 TCAM macro design for IPv6 lookup tables,” IEEE J. Solid-State
   Circuits, Feb. 2011.
4. S.-H. Yang, Y.-J. Huang, and J.-F. Li, “A low-power ternary content addressable memory with Pai-Sigma matchlines,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., Oct. 2012.
5. S.-J. Ruan, C.-Y. Wu, and J.-Y. Hsieh, “Low power design of precomputation-based content-addressable memory,” IEEE Trans. Very
   Large Scale Integr. (VLSI) Syst., Mar. 2008.
6. Y.-J. Chang and M.-F. Lan, “Two new techniques integrated for energy efficient TLB design,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,Jan. 2007.
7. Y.-J. Chang and Y.-H. Liao, “Hybrid-type CAM design for both power and performance efficiency,” IEEE Trans. Very Large Scale Integer. (VLSI) Syst., Aug. 2008.