IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 6, Issue 6, Ver. III (Nov. - Dec. 2016), PP 21-28
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Content Addressable Memory Architecture based on Sparse Clustered Networks

Ramagoni Swapnika[1], Dr.Mamatha Samson[2]
[1]*(ECE, GRIET, INDIA)*
[2]*(ECE, GRIET, INDIA)*

Abstract: *The main aim of this project is to design a low-power Content-Addressable Memory (CAM) by applying an algorithm for associativity between the input tag and the address of the corresponding output data. This architecture is based on a recently developed sparse clustered networks using binary connections that on-average eliminates most of the parallel comparisons which are performed during a search. Therefore, the dynamic energy consumption of this design will be less compared to conventional low-power CAM design. Given an input tag, this architecture will compute a minimum number of possibilities for the location of the matched tag and it will perform the comparisons on them to locate a single valid match.*

Keywords: *CAM, SCN-CAM, Tag.*

I. Introduction

Content Addressable Memory (CAM) is a type of memory which can be accessed using its contents rather than the explicit address. To access a particular entry in such memories, comparison between the search data word and previously stored entries is performed in parallel to find a match. Each stored entry is associated with a tag which is used in the comparison process. Once the search data word is applied to the input of a CAM, the matched data word will be retrieved within a single clock cycle if it exists. This great feature makes CAM a promising candidate for applications which require frequent and fast look-up operations, such as in translation look-aside buffers, database accelerators, network routers, image processing, parametric curve extraction, Hough transformation, Huffman encoding/decoding, virus detection Lempel–Ziv compression, and image coding. Due to the frequent and parallel search operations, CAMs require a significant amount of energy. CAM architectures typically make use of highly capacitive search lines (SLs), not causing them to be energy efficient when scaled. For example, this power inefficiency has constrained TLBs to be limited to not more than 512 entries in current processors. The fully associative TLBs consume about 15% and 17% of the total chip power, in Hitachi SH-3 and Strong ARM embedded processors respectively. Consequently, the main research objective has been focused on minimizing the energy utilization without compromising the throughput. Energy saving opportunities have been discovered by applying either circuit-level techniques, architectural-level techniques, or the co-design of the two, some of which have been surveyed in [1].

A family of associative memories based on Sparse Clustered Networks has been recently introduced and is describes in [2]. These memories make it possible to store many short messages instead of few long messages as in the conventional Hopfield networks with significantly less computational complexity. Furthermore, a considerable amount of improvement is achieved in terms of efficiency (the number of information bits stored per memory bit). In this paper, a variation of this approach and the corresponding architecture are introduced to construct a classifier that will be trained with the association between a small portion of the input tags and the addresses of the corresponding output data. In this paper the term CAM refers to binary CAM (BCAM). The proposed architecture (SCN-CAM) consists of an SCN-based classifier coupled to an array of CAMs. The CAM-array is divided into several equally sized sub-blocks and each sub-block can be activated independently. Providing an input tag for a previously trained network, the classifier uses only a small portion of the tag (sub-tag) and predicts very few sub-blocks to be activated. Once the sub-blocks are activated, the tag is compared with few entries in them while keeping the rest deactivated and thus decreases the dynamic energy dissipation.

II. Cam Review

In a conventional CAM array, each entry consists of a tag that, if matched with the input, points to the location of the required data word in the static random access memory (SRAM) block. The actual data of interest is stored in the SRAM and a tag is just a reference to it. Therefore, when it is required to search for a data in the SRAM, it is sufficient to search for its corresponding tag. Consequently, the tag may be shorter than the SRAM-data and it requires fewer bit comparisons. An example of typical CAM array, having four entries of 4 bits each, is shown in Fig. 1. A search data register is used to store the input data bits, this register applies the search data on the differential search lines s (SLs), which are shared among the entries. Then, the search data is compared against all of the CAM entries. Every CAM-word is attached to a common match line (ML) among its constituent bits, which indicates, whether they match with the input bits or not. Since the MLs have high capacitance, a sense amplifier is typically considered for each ML to improve the performance of the search operation.

The CAM can be configured using one of the two memory implementations: 1) SRL16E based CAM with a 16 clock cycle write operation and a one clock cycle search operation.2) Block RAM-based CAM with only two clock cycles for write operation and one clock cycle for search operation [3]. Any of these two techniques can be used depending upon the application.

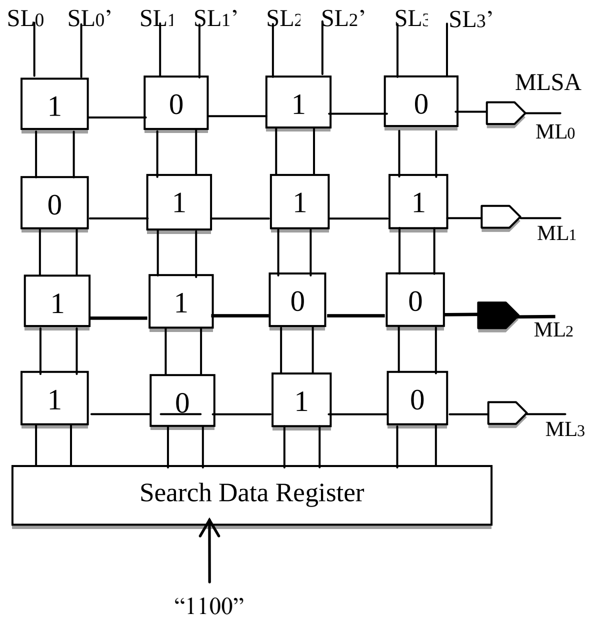

Figure 1. 4X4 CAM array

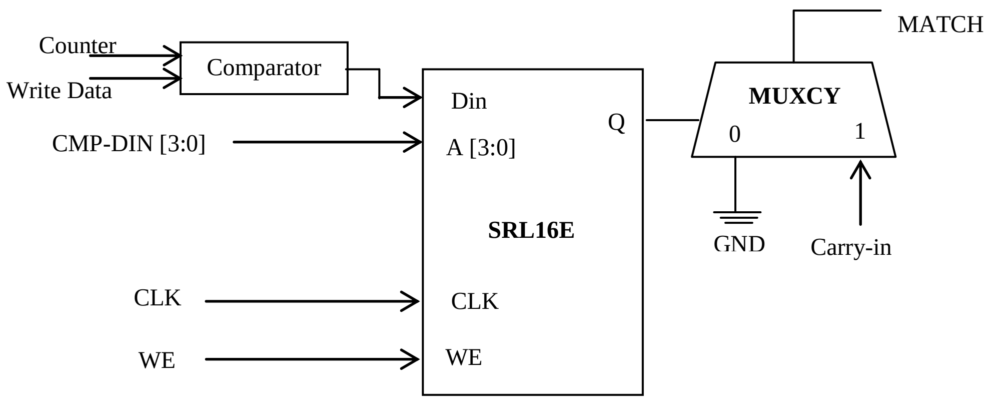

Figure 2. Example of 4x1 CAM using SRL16E

Implementing a CAM using SRL16s makes use of many components found within each FPGA slice. As an example, a 4 x 1 CAM which is built using an SRL16E primitive and the MUXCY is shown in Fig.2. Adding more SRL16E/MUXCY pairs allows for extension of the CAM width and depth.

Energy reduction of CAMs by applying circuit level techniques are mostly based on the following strategies: 1) Decreasing the energy consumption of SL by disabling the precharge process of SLs when not necessary [5], [5]–[8] and 2) Decreasing the ML precharging, for example, by segmenting the ML, selectively precharging the first few segments and then propagating the precharge process for the next segments if and only if those first segments match [9]. This segmentation technique increases the delay as the number of segments is increased. A hybrid-type CAM integrates the low-power feature and high-performance of NAND and NOR type [10] respectively. In the bank-selection architecture [11], [12], the CAM array is divided into *B* equally partitioned banks and are activated based on the value of extra bits of length $\log_2(B)$ which are added to the search data word. These extra bits are decoded in order to determine the banks which must be selected. The drawback of this design is that the banks may overflow since the length of the words remains same for all the banks. For example, a 128k-entry CAM that incorporates 60-bit words and one additional bank-selection bit such that two banks result with 64k entries each. Therefore, each bank can have 260 possibilities causing an overflow probability that is higher compared with when not banked. This overflow would require additional circuitry that reduces the power saving opportunity since as a result multiple banks are activated concurrently [1]. The precomputation-based CAM (PBCAM) architecture (also known as one's count) was introduced in [13]. PB-CAM divides the circuitry and the comparison process into two stages. First, it counts the number of ones in an input and then compares the result with that of entries using an additional CAM circuit that has the number of ones in the previously stored CAM-data. This activates a few MLs and deactivates the others. In the second stage, a modified hierarchy of CAM is used, which has reduced complexity, and has only one pull-down path instead of two compared with the conventional design. The modified architecture only considers 0 mismatches instead of full comparison since the 1s have already been compared. The number of comparisons can be reduced to M ×log (N+2)+(M × N)/(N +1)bits, where M is the number of entries in the CAM and N is the number of bits per entry. In the proposed design, the possibility of reducing the number of comparisons to only N bits is demonstrated. Furthermore, in PB-CAM, the increase of the tag length affects the energy consumption, the delay, and also complicates the precomputation stage.

III. SCN-CAM

As shown in Fig. 3, the proposed architecture (SCN-CAM) consists of an SCN-based classifier, which is connected to a special-purpose CAM array. The classifier is at first trained with the association between the tags and the address of the data to be retrieved later. The proposed CAM array is based on a typical architecture, but is divided into several sub-blocks which can be activated independently. Therefore, it is also possible to train the network with the associativity between the tag and each CAM sub-block, if the number of desired sub-blocks is known. However, in this paper, the focus is on a generic architecture that can be easily optimized for any number of CAM sub-blocks.

Once an input tag is given to the SCN-based classifier, it predicts which CAM sub-block(s) need to be activated and thus saves the dynamic power by disabling the remaining sub-blocks. In this paper the possibility of reducing the number of comparisons to only one in average is shown. SCN-CAM uses only a portion of the actual tag to create or recover the association with the corresponding output. The operation of the CAM, on average, allows this reduction in the tag length.

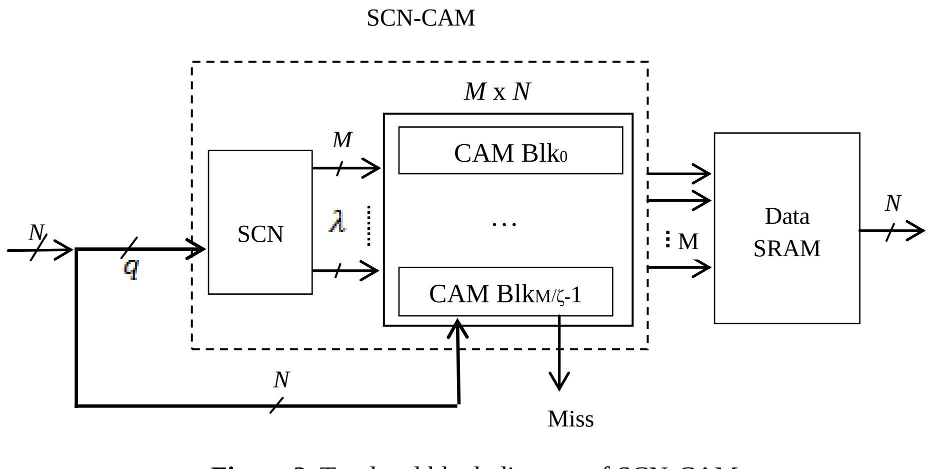

Figure 3. Top level block diagram of SCN-CAM

A large enough tag length permits SCN-CAM to always point to a single sub-block. However, the length of reduced-length tag affects the hardware complexity of the SCN-based classifier. The length of the reduced-length tag is not dependent on the length of the original tag but rather dependent on the number of CAM entries.

3.1. SCN-CAM ALGORITHM

As shown in Fig. 4, the SCN-based classifier consists of two parts: 1) PI and 2) PII. The neurons of PI are binary and correspond to the input tags. These neurons are grouped into *c* equally sized clusters of *l* neurons each. Processing of an input tag in the SCN-based classifier is either for training or decoding. In this paper, both for training and decoding purposes, the tag is reduced in length to *q* bits, then it is divided into *c* equally sized partitions of length *κ* bits each. Then each partition is mapped to the index of a neuron in its corresponding cluster in PI, using a technique called direct binary-to-integer mapping from the tag portion to the index of the neuron to be activated. Therefore, $l = 2^{\kappa}$. If l is the given parameter, the number of clusters is calculated as $c=q/\log_2(l)$.

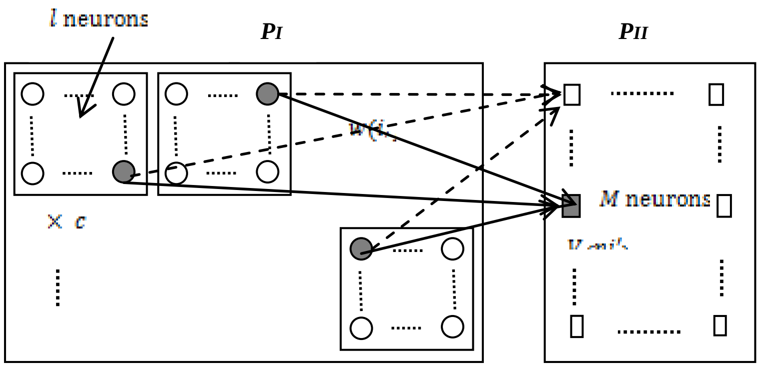

Figure 4. Representation of SCN-CAM with M entries and a reduced length tag of c x log_2 (l)

Therefore, for simplicity in hardware implementation, we can choose q to be a multiple of κ. It is important to note that there are no connections between the neurons in PI. PII is a single cluster consisting of M neurons, which is equal to the number of CAM entries. Every neuron in PII, ni', is connected to each neuron in PI via a connection whose binary value is $w(i,j)(i')$, and thus equal to either 0 or 1. The value of $w(i,j)(i')$ determines whether there exists an association between the jth neuron in the ith cluster in PI, and the i'th neuron in PII.

3.1.1. Network Training

The binary values of the connections in the SCN-based classifier indicate associativity between the input tags and the corresponding outputs. During the process of training, the connection values are set, and these values are stored in a memory module such that they can be used to retrieve the address of the target data whenever it is required. A connection has a value 1 if there is associativity between the corresponding neuron in PI and a CAM entry, represented as a neuron in PII. For example, let us assume $c = 2$ and $q = 6$. For a reduced-length input tag 101110 associated to the fourth entry in the CAM, first, we split this input into two parts: 101 and 110. Then, each part is associated with a neuron in the corresponding cluster in PI:5 for 101 and 6 for 110. Finally, the connections from these neurons toward the target neuron, 4, in PII are added. That is, $w(1,5)(4)$, and $w(2,6)(4)$ is equal to 1.

3.1.2. Network Update

When an update is requested in SCN-CAM, retraining the entire SCN-based classifier with all the entries is not required. The reason lies in the fact that the output neurons of PII are independent from each other. Therefore, by deleting the connections from a neuron PII to the corresponding connections in PI, a tag can be deleted. In other words, to delete an entry, c connections are removed, one for each cluster. Adding new connections to the same neuron in PII, but to different neurons in PI, adds a new entry to the SCN-based classifier. Therefore the new entry can be added by adding new connections while keeping the previous connections for other entries in the network.

3.1.3. Tag Decoding

Once the SCN-based classifier has been trained, the ultimate objective after receiving the tag is to find out which neuron(s) in PII should be activated based on the given q bits of the tag. This process is called decoding in which the connection values are recalled from the memory. The decoding process is divided into four steps.

1. An input tag is reduced in length to q bits and divided into c equally sized partitions. The q bits of the entire tag can be selected in such a way to reduce the correlation.
2. *Local Decoding (LD):* A single neuron per cluster in PI is activated by using a direct binary-to-integer mapping from the tag portion to the index of the neuron to be activated.
3. *Global Decoding (GD):* GD determines which neuron(s) in PII must be activated based on the results from LD and the stored connection values. If there is at least one active connection from each cluster in PI toward a neuron in PII, that neuron is activated. GD can be expressed as:

$$v_{ni'} = \bigwedge_{i=1}^{c} \bigvee_{j=1}^{l} \left(w_{(i,j)} \bigwedge v_{(i,j)} \right) \qquad (1)$$

Where $\wedge$ and $\vee$ represent logical OR and AND operations, respectively. *v(i, j)* is the value of the *j*th neuron in the *i*th cluster in PI, whereas *vni'* is the value of the *i'*th neuron in PII.

4. If more than one neuron is activated in PII, then, the same number of word comparisons is required to detect the correct match. A single activated neuron means no further comparisons are required. Because we may not afford (in terms of the silicon area) to implement only one independently controlled CAM-row per neuron, the neurons in PII are grouped into ζ-neurons. Each group of neurons generates a single activation signal to enable parallel comparison operations in its corresponding CAM sub-block. A logical OR operation is thus performed on the value of each group of neurons resulting generation of M/ζ bits, which is also equal to the number of CAM sub-blocks.

The number of compare-enabled sub-blocks (ψ) can be estimated by multiplying the probability that a sub-block can be enabled by the total number of sub-blocks:

$$\left[\psi = 1 - \left(1 - \frac{1}{\beta}\right)^{1+E(\lambda)}\right].\beta \qquad (2)$$

Below table describes all the Design parameters which are used to implement the CAM architecture.

Table 1: Reference Design Parameters

Parameter	Value
M	512
N	128
ζ	8
β	64
$E(\lambda)$	1
q	9
c	3
l	8

Where M is the number of CAM entries, N is the number of input bits, the neurons in PII are grouped into ζ- neurons, β represents the number of CAM subblocks. $E(\lambda)$ is the expected value of λ where λ is the random variable. *q* represents the reduced length tag, *c* is the number of clusters in PI and *l* represents the the number of neurons in each cluster.

3.2. SCN-CAM Architecture

In order to exploit the major feature of the SCN-based associative memory in the classification of the search data, a CAM array is divided into adequate number of compare-enabled sub-blocks such that, the number of subblocks are not too many to complicate the interconnections and are not too few to exploit to energy-saving opportunity of the SCN-based classifier.

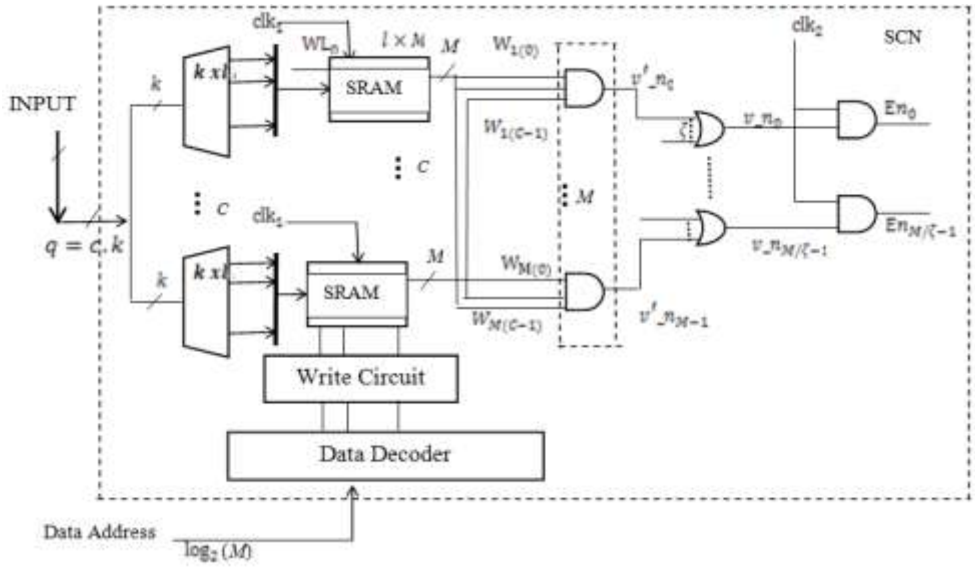

Figure 5. SCN Classifier

Consequently, the neurons in PII are grouped and ORed as shown in Fig. 5 to construct the compare-enable signal(s) for the CAM array. Even the conventional CAM arrays need to be divided into multiple sub-blocks since long bit lines and SLs can slow down the read, write, and search operations due to the presence of drain, gate, and wire capacitances. The number of sub-blocks, β, is equal to M/ζ, where M is the total number CAM entries, and ζ is the number of CAM-rows per sub-block.

IV. Circuit implementation

A top-level block diagram of the implementation of SCN-CAM is shown in Fig. 3. The optimum design parameters of any design depend on the speed, energy consumption, and area requirements. If the area budget is limited, smaller values of ζ is preferred with the cost of higher number of comparisons and thus the energy consumption. If the energy consumption is a critical design parameter, and the budget for the silicon area is more relaxed, a balance between a large enough q and a small ζ needs to be considered. A preferred set of design choices based on the experimental simulations on a 512-entry CAM is summarized in Table I. In SCN-CAM, SRL16E based CAM structure is used. A complete circuit for SCN-CAM was implemented and simulated using Xilinx ISE simulator according to Table I parameters, including full dimensions of CAM arrays, SRAM arrays, logic gates. Two signals clk1 and clk2 are used in Fig. 5 to integrate the operation of SCN-based classifier and the CAM sub-blocks.

V. Result analysis

Simulation results of SCN-Classifier, and SCN-CAM are provided below.

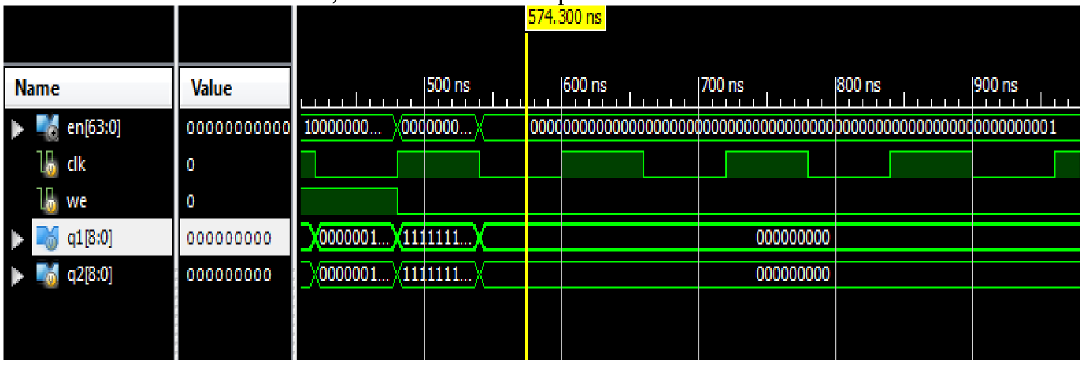

Figure 6. Simulation result of SCN classifier

Here inputs are q1[8:0], q2[8:0], clk, we, and outputs are en[63:0]. When clk is high, only one en signal is high depending upon the input signal, so that only one sub block can be compare enabled disabling the rest, thus the comparison process becomes simpler. In Fig.6, LSB of en[63:0] is "1" which represents that the subblock-1 must be compare enabled and all other subbolcks must be disabled for comparison process.

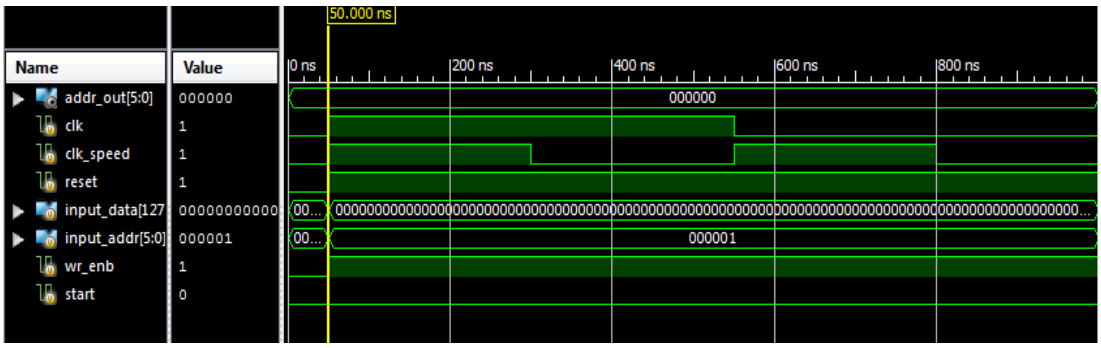

Figure 7. Simulation result of SCN CAM

Here inputs are clk, clk_speed, input_data[127_0], input_addr[5:0] and output is addr_out[5:0]. For a given input tag, address of the corresponding subblock is generated. In Fig.7, addr_out is "000000", represents that the subblock-1 is active. Here two clock signals are used which are clk and clk_speed. clk is the delayed signal of clk_speed.

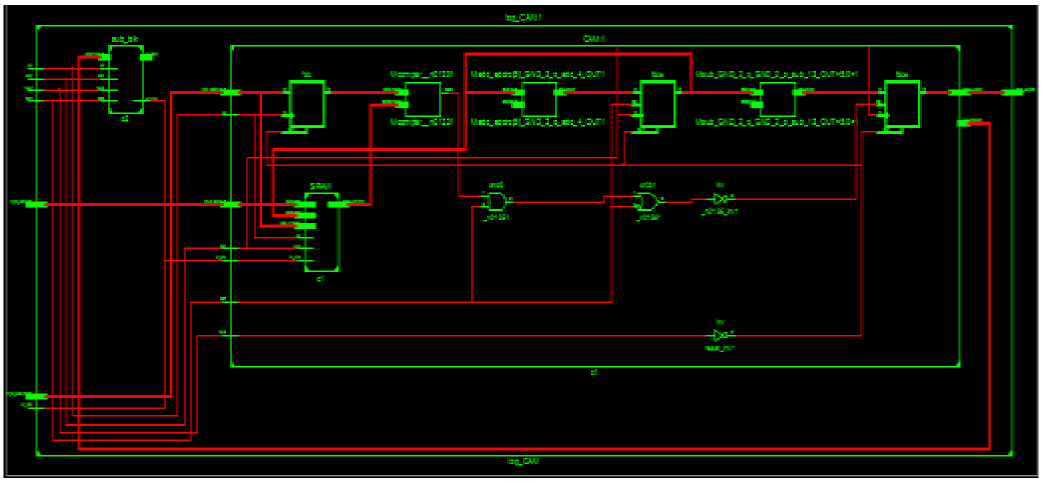

Figure 8. RTL Schematic of SCN CAM

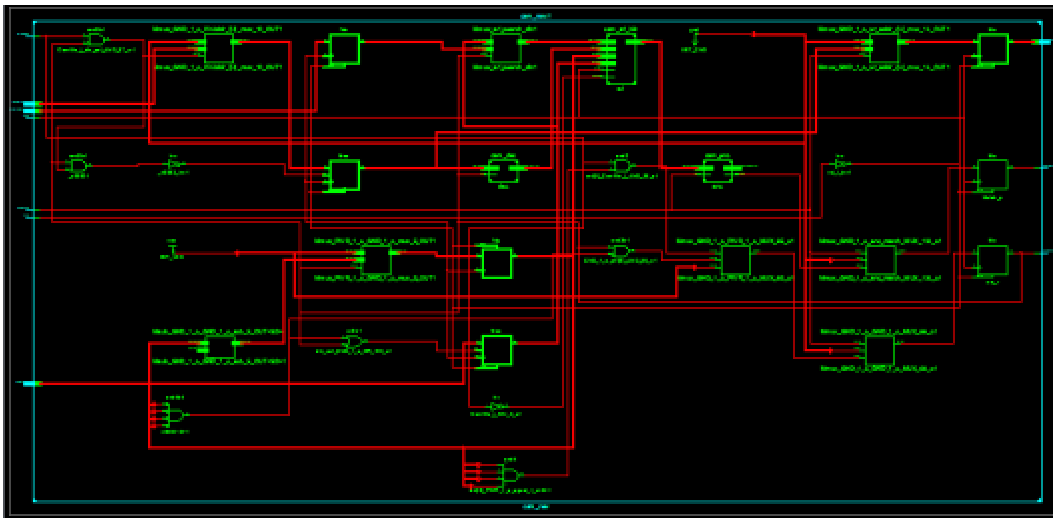

Figure 9. RTL Schematic of CAM using SRL16E

Table 2: Synthesis reports of CAM and SCN-CAM

S.No	Parameter	CAM	SCN CAM
1.	Delay	10.923ns	2.617ns
2.	Memory	275124Kb	258096Kb
3.	No. of slice LUTs	569	69
4.	No. of fully used LUT FF pairs	79	24

VI. Conclusion

In this paper, the algorithm and the architecture of a low-power CAM are introduced. The proposed architecture (SCN-CAM) employs an associativity mechanism based on a recently developed family of associative memories based on SCNs. SCN-CAM is suitable for low-power applications, which require frequent and parallel look-up operations. SCN-CAM employs an SCN-based classifier, which is connected to several independently compare-enabled CAM sub-blocks, some of which are activated once a tag is presented to the SCN-based classifier. By using independent nodes in the output part of SCN-CAM's training network, simple and fast updates can be generated without retraining the network entirely. With optimized lengths of the reduced-length tags, SCN-CAM removes most of the comparison operations given a uniform distribution of the reduced-length inputs.

References

[1]. K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," *IEEE J. Solid-State Circuits, vol. 41, no. 3, pp. 712–727, Mar. 2006.*

[2]. H.Jarollahi, Vincent Gripon, Naoya Onizawa, "Algorithm and Architecture for a Low-Power Content Addressable Memory based on Sparse Clustered Networks", in *IEEE Trans., on VLSI, vol 23, No. 4,April 2015.*

[3]. Kyle Locke "Parameterizable Content Addressable Memory" in *XAPP1151, vol1.0, March. 2011.*

[4]. 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-StateCircuits, vol. 46, no. 2, pp. 507–519, Feb. 2011*

[5]. H. Noda et al., "A cost-efficient high-performance dynamic TCAM with pipelined hierarchical searching and shift redundancy architecture," *IEEE J. Solid-State Circuits, vol. 40, no. 1, pp. 245–253, Jan. 2005.*

[6]. K. Pagiamtzis and A. Sheikholeslami, "Pipelined match-lines and hierarchical search-lines for low-power content-addressable memories," in *Proc. IEEE Custom Integr. Circuits Conf., Sep. 2003, pp. 383–386.*

[7]. K. Pagiamtzis and A. Sheikholeslami, "A low-power content addressable memory (CAM) using pipelined hierarchical search scheme," *IEEE J. Solid-State Circuits, vol. 39, no. 9, pp. 1512–1519, Sep. 2004.*

[8]. H. Noda et al., "A 143 MHz 1.1 W 4.5 Mb dynamic TCAM with hierarchical searching and shift redundancy architecture," in *Proc. IEEEISSCC, vol. 1. Feb. 2004, pp. 208–523.*

[9]. C. Zukowski and S.-Y. Wang, "Use of selective precharge for low power on the match lines of content-addressable memories," in *Proc.Int. Workshop Memory Technol., Des. Test., Aug. 1997, pp. 64–68.*

[10]. Y.-J. Chang and Y.-H. Liao, "Hybrid-type CAM design for both power and performance efficiency," *IEEE Trans. Very Large Scale Integr. (VLSI)Syst., vol. 16, no. 8, pp. 965–974, Aug. 2008.*

[11]. M. Motomura, J. Toyoura, K. Hirata, H. Ooka, H. Yamada, and T. Enomoto, "A 1.2-million transistor, 33-MHz, 20-b dictionary search processor (DISP) ULSI with a 160-kb CAM," *IEEE J. Solid-StateCircuits, vol. 25, no. 5, pp. 1158–1165, Oct. 1990.*

[12]. K. Schultz and P. Gulak, "Fully parallel integrated CAM/RAM using preclassification to enable large capacities," *IEEE J. Solid-State Circuits, vol. 31, no. 5, pp. 689–699, May 1996.*

[13]. C.-S. Lin, J.-C.Chang, and B.-D. Liu, "A low-power precomputation based fully parallel content-addressable memory," *IEEE J. Solid-StateCircuits, vol. 38, no. 4, pp. 654–662, Apr. 2003.*

[14]. H. Jarollahi, V. Gripon, N. Onizawa, and W. J. Gross, "A low-power content-addressable memory based on clustered-sparse networks," in *Proc. 24th IEEE Int. Conf. ASAP, Jun. 2013, pp. 305–308.*