Algorithm and Architecture for a Low-Power Content-Addressable Memory Based on Sparse Clustered Networks

Hooman Jarollahi, *Student Member, IEEE*, Vincent Gripon, Naoya Onizawa, *Member, IEEE*, and Warren J. Gross, *Senior Member, IEEE*

***Abstract*— We propose 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. The proposed architecture is based on a recently developed sparse clustered network using binary connections that on-average eliminates most of the parallel comparisons performed during a search. Therefore, the dynamic energy consumption of the proposed design is significantly lower compared with that of a conventional low-power CAM design. 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. TSMC 65-nm CMOS technology was used for simulation purposes. Following a selection of design parameters, such as the number of CAM entries, the energy consumption and the search delay of the proposed design are 8%, and 26% of that of the conventional NAND architecture, respectively, with a 10% area overhead. A design methodology based on the silicon area and power budgets, and performance requirements is discussed.**

***Index Terms*— Associative memory, content-addressable memory (CAM), low-power computing, recurrent neural networks, sparse clustered networks (SCNs).**

I. 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], virus detection [8] Lempel–Ziv compression [9], and image coding [10].

Due to the frequent and parallel search operations, CAMs consume a significant amount of energy. CAM architectures typically use highly capacitive search lines (SLs) causing them not to be energy efficient when scaled. For example, this power inefficiency has constrained TLBs to be limited to no more than 512 entries in current processors. In Hitachi SH-3 and StrongARM embedded processors, the fully associative TLBs consume about 15% and 17% of the total chip power, respectively [11]–[13]. Consequently, the main research objective has been focused on reducing the energy consumption without compromising the throughput. Energy saving opportunities have been discovered by employing either circuit-level techniques [14], [15], architectural-level [16], [17] techniques, or the codesign of the two, [18], some of which have been surveyed in [19]. Although dynamic CMOS circuit techniques can result in low-power and low-cost CAMs, these designs can suffer from low noise margins, charge sharing, and other problems [16].

A new family of associative memories based on sparse clustered networks (SCNs) has been recently introduced [20], [21], and implemented using field-programmable gate arrays (FPGAs) [22]–[24]. Such memories make it possible to store many short messages instead of few long ones as in the conventional Hopfield networks [25] with significantly lower level of computational complexity. Furthermore, a significant improvement is achieved in terms of the number of information bits stored per memory bit (efficiency). In this paper, a variation of this approach and a corresponding architecture are introduced to construct a classifier that can be trained with the association between a small portion of the input tags and the corresponding addresses of the output data. The term CAM refers to binary CAM (BCAM) throughout this paper. Originally included in [26], preliminary results were introduced for an architecture with particular parameters conditioned on uniform distribution of the input patterns. In this paper, an extended version is presented that elaborates the effect of the design's degrees of freedom, and the effect of nonuniformity of the input patterns on energy consumption and the performance.

The proposed architecture (SCN-CAM) consists of an SCN-based classifier coupled to a CAM-array. The CAM-array

Manuscript received July 5, 2013; revised February 24, 2014; accepted April 8, 2014. Date of publication April 30, 2014; date of current version March 18, 2015.

H. Jarollahi and W. J. Gross are with the Department of Electrical and Computer Engineering, McGill University, Montreal, QC H3A 2A7, Canada (e-mail: hooman.jarollahi@mail.mcgill.ca; warren.gross@mcgill.ca).

V. Gripon is with the Department of Electronics, Télécom Bretagne, Brest 29238, France (e-mail: vincent.gripon@telecom-bretagne.eu).

N. Onizawa is with the Research Institute of Electrical Communication, Tohoku University, Sendai 980-8577, Japan (e-mail: naoya.onizawa@mail.mcgill.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2014.2316733

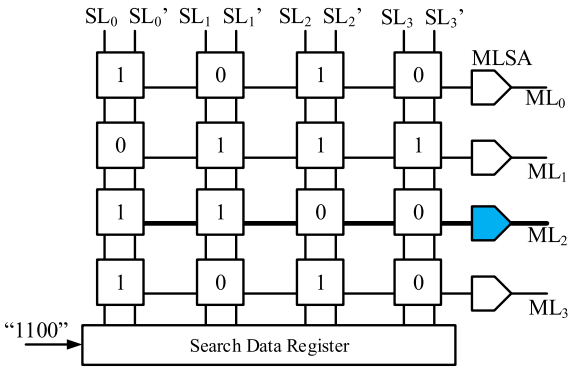

Fig. 1. Simple example of a 4×4 CAM array consisting of the CAM cells, MLs, sense amplifiers, and differential SLs.

is divided into several equally sized sub-blocks, which can be activated independently. For a previously trained network and given an input tag, the classifier only uses a small portion of the tag and predicts very few sub-blocks of the CAM to be activated. Once the sub-blocks are activated, the tag is compared against the few entries in them while keeping the rest deactivated and thus lowers the dynamic energy dissipation.

The rest of this paper is organized as follows. Section II describes basic operation of the CAM. In Section III, some of the recent research works related to this area are summarized. In Section IV, the proposed associativity algorithm is introduced. Section V describes the hardware architecture followed by Section VI with the simulation results. Circuit level simulations throughout this paper are obtained using HSPICE and TSMC 65-nm CMOS technology. Finally, conclusions are drawn in Section VII.

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 a data word in a static random access memory (SRAM) block. The actual data of interest are stored in the SRAM and a tag is simply a reference to it. Therefore, when it is required to search for the data in the SRAM, it suffices to search for its corresponding tag. Consequently, the tag may be shorter than the SRAM-data and would require fewer bit comparisons.

An example of a typical CAM array, consisting of four entries having 4 bits each, is shown in Fig. 1. A search data register is used to store the input bits. The register applies the search data on the differential SLs, which are shared among the entries. Then, the search data are compared against all of the CAM entries. Each CAM-word is attached to a common match line (ML) among its constituent bits, which indicates, whether or not, they match with the input bits. Since the MLs are highly capacitive, a sense amplifier is typically considered for each ML to increase the performance of the search operation.

As an example, in TLBs, the tag is the virtual page number (VPN), and the data are the corresponding physical page number (PPN). A virtual address generated by the CPU consists of the VPN, and a page offset. The page offset is later used along with PPN to form the physical address. Since most TLBs are fully associative, in order to find the corresponding PPN, a fully parallel search among VPNs is conducted for every generated virtual address [2].

A BCAM cell is typically the integration of a 6-transistor (6T) SRAM cell and comparator circuitry. The comparator circuitry is made out of either an XNOR or an XOR structure, leading to a NAND-type or a NOR-type operation, respectively. The selection of the comparing structure depends on the performance and the power requirements, as a NAND-type operation is slower and consumes less energy as opposed to that of a NOR type.

The schematic of two types of typical BCAM cells are shown in Fig. 2. In a NAND-type CAM, the MLs are precharged high during the precharge phase. During the evaluation phase, in the case of a match, the corresponding ML is pulled down though a series of transistors [M_5 in Fig. 2(b)] performing a login NAND in the comparison process. In a NOR-type CAM [Fig. 2(a)], the MLs are also precharged high during the precharge phase. However, during the evaluation phase, all of the MLs are pulled down unless there is a matched entry such that the pull-down paths $M_3 - M_4$ and $M_5 - M_6$ are disabled. Therefore, a NOR-type CAM has a higher switching activity compared with that of a NAND type since there are typically more mismatched entries than the matched ones.

Although a NAND-type CAM has the advantage of lower energy consumption compared with that of the NOR-type counterpart, it has two drawbacks: 1) a quadratic delay dependence on the number of cells due to the serial pull-down path and 2) a low noise margin.

III. Related Work

Energy reduction of CAMs employing circuit-level techniques are mostly based on the following strategies: 1) reducing the SL energy consumption by disabling the precharge process of SLs when not necessary [18], [27]–[30] and 2) reducing the ML precharging, for example, by segmenting the ML, selectively precharging the first few segments and then propagating the precharge process if and only if those first segments match [31]. This segmentation strategy increases the delay as the number of segments is increased. A hybrid-type CAM integrates the low-power feature of NAND type with the high-performance NOR type [12] while similar to selective precharging method [31], the ML is segmented into two portions. The high-speed CAM designed in 32-nm CMOS [1] achieves the cycle time of 290 ps using a swapped CAM cell that reduces the search delay while requiring a larger CAM cell (11-transistors) than a conventional CAM cell [9-transistors (9T)] used in SCN-CAM. A high-performance AND-type match-line scheme is proposed in [32], where multiple fan-in AND gates are used for low switching activity along with segmented-style match-line evaluation to reduce the energy consumption.

In the bank-selection architecture [33], [34], the CAM array is divided into B equally partitioned banks that are activated based on the value of added bits of length $\log_2(B)$ to the search data word. These extra bits are decoded to determine, which banks must be selected. This architecture was considered at

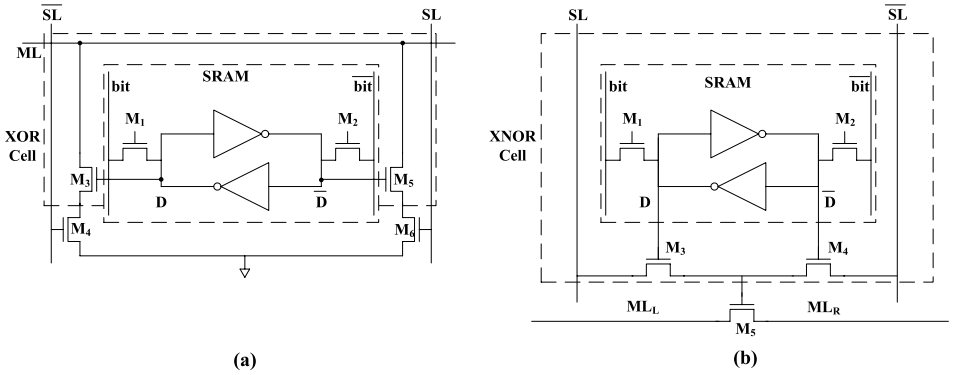

Fig. 2. Classical BCAM cell types. (a) 10T NOR. (b) 9T NAND.

first to reduce the silicon area by sharing the comparison circuitry between the blocks but was later considered for power reduction as well. The drawback of this architecture is that the banks can overflow since the length of the words remains the same for all the banks. For example, let us consider a 128k-entry CAM that incorporates 60-bit words and one additional bank-select bit such that two banks result with 64k entries each. Therefore, each bank can have 2^{60} possibilities causing an overflow probability that is higher compared with when not banked. This overflow would require extra circuitry that reduces the power saving opportunity since as a result multiple banks are activated concurrently [19].

The precomputation-based CAM (PB-CAM) architecture (also known as one's count) was introduced in [16]. PB-CAM divides the comparison process and the circuitry into two stages. First, it counts the number of ones in an input and then compares the result with that of the entries using an additional CAM circuit that has the number of ones in the CAM-data previously stored. This activates a few MLs and deactivates the others. In the second stage, a modified CAM hierarchy 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 $M \times \lceil \log(N+2) \rceil + (M \times N)/(N+1)$ bits, where M is the number of entries in the CAM and N is the number of bits per entry. In the proposed design, we demonstrate how it is possible to reduce the number of comparisons to only N bits. Furthermore, in PB-CAM, the increase of the tag length affects the energy consumption, the delay, and also complicates the precomputation stage.

In the asynchronous architecture proposed by [15], as the CAM assigns consecutive search data matched in different word blocks, it operates based on the delay of matching its first few bits instead of its full length as long as the consecutive subsearch words are different. However, the cycle time is drastically increased when the search-data patterns are correlated. For example, if we have correlations in the first 8 bits of the stored data, the cycle time is increased to 1.359 ns, which is 5.2 times that of the noncorrelated scenario. In the proposed design, the cycle time is independent of the correlation between the input patterns. Furthermore, the asynchronous architecture in [15] is more susceptible to process variations compared with its synchronous counterparts.

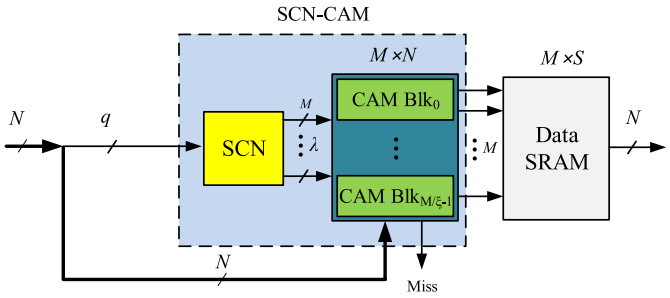

Fig. 3. Top level block diagram of SCN-CAM. The CAM array is divided into $M/\zeta - 1$ sub-blocks that can be independently activated for comparison. The compare-enable signals are generated by the SCN-based classifier.

IV. SCN-CAM Algorithm

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 SCN-based classifier is at first trained with the association between the tags and the address of the data to be later retrieved. The proposed CAM array is based on a typical architecture, but is divided into several sub-blocks that can be compare-enabled independently. Therefore, it is also possible to train the network with the association between the tag and each CAM sub-block if the number of desired sub-blocks is known. However, in this paper, we focus on a generic architecture that can be easily optimized for any number of CAM sub-blocks. Once an input tag is presented to the SCN-based classifier, it predicts which CAM sub-block(s) need to be compare-enabled and thus saves the dynamic power by disabling the rest. Disabling a CAM sub-block avoids charging its highly capacitive SLs, while applying the search data, and also turns the precharge path off for the MLs.

We show how it is possible, through the algorithmic reduction in hardware complexity, to reduce the number of comparisons to only one in average. SCN-CAM uses only a portion of the actual tag to create or recover the association

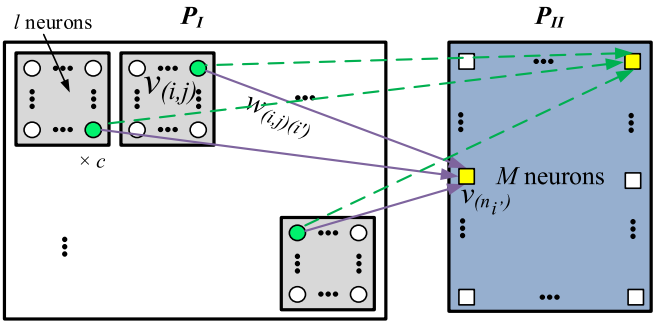

Fig. 4. Representation of the proposed SCN-CAM for consisting of M entries and a reduced-length tag of $c \times \log_2(l)$.

with the corresponding output. The operation of the CAM, on average, allows this reduction in the tag length. 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.

A. SCN-Based Classifier

As shown in Fig. 4, an SCN-based classifier consists of two parts: 1) $\mathrm{P_I}$ and 2) $\mathrm{P_{II}}$. The neurons in $\mathrm{P_I}$ are binary, correspond to the input tags, and are grouped into c equally sized clusters with l neurons in each. Processing of an input tag in the SCN-based classifier is for either of the two situations: training or decoding. In this paper, either for training or decoding purposes, the input tag is reduced in length to q bits, and then divided into c equally sized partitions of length κ bits each. Each partition is then mapped to the index of a neuron in its corresponding cluster in $\mathrm{P_I}$, using a direct binary-to-integer mapping from the tag portion to the index of the neuron to be activated. Thus, $l = 2^{\kappa}$. If l is a given parameter, the number of clusters is calculated to be $c = q/\log_2(l)$. 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 $\mathrm{P_I}$. $\mathrm{P_{II}}$ is a single cluster consisting of M neurons, which is equal to the number of entries in the CAM. Each neuron in $\mathrm{P_{II}}$, $n_{i'}$, is connected to every neuron in $\mathrm{P_I}$ 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 $\mathrm{P_I}$, and the i'th neuron in $\mathrm{P_{II}}$.

1) Network Training: The binary values of the connections in the SCN-based classier indicate associations of the input tags and the corresponding outputs. The connection values are set during the training process, and are stored in a memory module such that they can later be used to retrieve the address of the target data. A connection has a value 1 when there exists an association between the corresponding neuron in $\mathrm{P_I}$ and a CAM entry, represented as a neuron in $\mathrm{P_{II}}$.

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 $\mathrm{P_I}$: 5 for 101 and 6 for 110. Finally, the connections from these neurons toward the target neuron, 4, in $\mathrm{P_{II}}$ are added. That is, $w_{(1,5)(4)}$, and $w_{(2,6)(4)}$ is equal to 1.

2) Network Update: When an update is requested in SCN-CAM, retraining the entire SCN-based classifier with the entries is not required. The reason lies in the fact that the output neurons of $\mathrm{P_{II}}$ are independent from each other. Therefore, by deleting the connections from a neuron $\mathrm{P_{II}}$ to the corresponding connections in $\mathrm{P_I}$, 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 $\mathrm{P_{II}}$, but to different neurons in $\mathrm{P_I}$, adds a new entry to the SCN-based classifier. The new entry can therefore be added by adding new connections while keeping the previous connections for other entries in the network.

3) Tag Decoding: Once the SCN-based classifier has been trained, the ultimate goal after receiving the tag is to determine which neuron(s) in $\mathrm{P_{II}}$ 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 can be selected within the tag bits in such a way to reduce the correlation.
2) *Local Decoding (LD):* A single neuron per cluster in $\mathrm{P_I}$ is activated 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 $\mathrm{P_{II}}$ must be activated based on the results from LD and the stored connection values. If there exists at least one active connection from each cluster in $\mathrm{P_I}$ toward a neuron in $\mathrm{P_{II}}$, that neuron is activated. GD can be expressed as

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

where $\bigvee$ and $\bigwedge$ represent logical OR and AND operations, respectively. $v_{(i,j)}$ is the binary value of the jth neuron in the ith cluster in $\mathrm{P_I}$, whereas $v_{n_{i'}}$ is the binary value of the i'th neuron in $\mathrm{P_{II}}$.
4) If more than one neuron are activated in $\mathrm{P_{II}}$, then, the same number of word comparisons are 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 $\mathrm{P_{II}}$ 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.

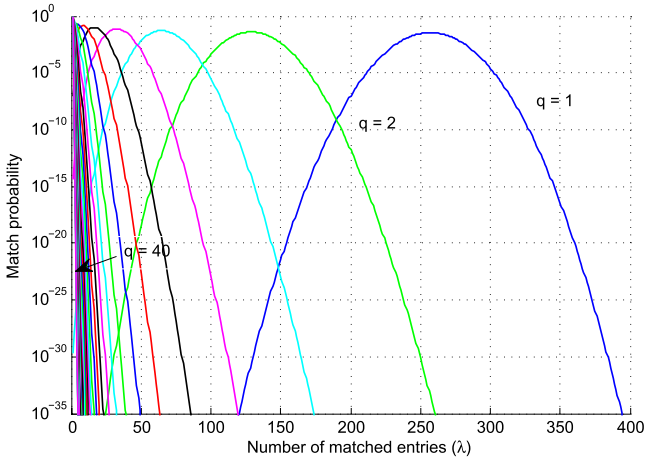

Fig. 5. Relationship between the length of the truncated tag (q), the number of matched entries in SCN-CAM (λ), and the estimated matching probability ($P(\lambda)$) for $M = 512$.

B. Tag-Length Reduction

Given the input tags, the number of bits in the reduced-length tag, q, determines the number of possible ambiguities in P_{II}. The generated ambiguities can be corrected with additional comparisons to find the exact match in the CAM. Therefore, no errors are produced in determining the matched result(s). On the other hand, no length reduction leads to the generation of no ambiguities, but a higher level of hardware complexity in the SCN-based classifier, since more neurons are required.

C. Data Distribution

The number of ambiguities, generated in P_{II} is dependent on the correlation factor of the tag pattern, that is the number of similar repeating bits in the subset of tags. A higher degree of similarities results in a higher number of ambiguous neurons. If the tag pattern is previously known, it is possible to select the reduced-length tag bits among those that have a lower level of similarity likelihood. Otherwise, SCN-CAM detects the valid match but with a higher cost of power consumption.

Assuming a random distribution for the CAM entries, the expected number of possible matches is the actual match plus the expected number of ambiguities. The number of ambiguities given a random and uniformly distributed input pattern can be estimated using (2), where λ is a random variable representing the number of ambiguities. P_λ is the probability that exactly λ ambiguities occur using q bits of the tag. Therefore, it follows a binomial distribution as shown in (2). Software simulations on numerical data samples verify the validity of (2). Therefore, according to the binomial law the expected value of λ, $E(\lambda)$, and its variance, $\mathrm{Var}(\lambda)$, can be calculated as shown in (3) and (4), respectively. If $q = \log_2(M)$, only one ambiguity is achieved on average leading to the activation of one extra neuron in P_{II} in addition to the actual match. Consequently, this value of q, and λ is used throughout this paper as a starting point to estimate the cycle time and the energy consumption of the proposed CAM design.

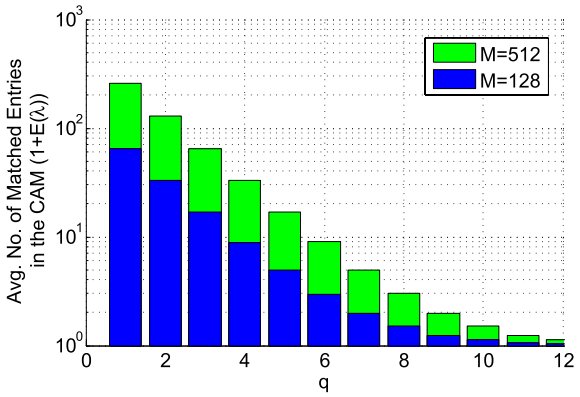

Fig. 6. Expected value of the number of required comparisons in SCN-CAM versus the number of bits in the reduced-length tag.

Fig. 5 shows simulations results on how it is possible to reduce the estimated number of required comparisons by increasing q. It is interesting to note that the number of clusters in P_I does not affect the number of required comparisons

$$P_\lambda = \binom{M-1}{\lambda}\left(\frac{1}{2^q}\right)^\lambda\left(1-\frac{1}{2^q}\right)^{M-1-\lambda} \tag{2}$$

$$E(\lambda) = \sum_\lambda \lambda \cdot P_\lambda = (M-1)/(2^q) \tag{3}$$

$$\mathrm{Var}(\lambda) = (M-1)(1/2^q)(1-1/2^q). \tag{4}$$

If the input pattern is correlated in a sense that certain bits repeat their values among all of the CAM, (3) can be modified to the following:

$$E_c(\lambda) = \frac{M-1}{2^{q-k}} \tag{5}$$

where k is the number of similar bits in q, and E_c is the expected value of the number of matched entries. We can consider more complex models, such as when the reduced tags are obtained using a Bernoulli distribution with parameter α. We then partition reduced tags depending on the number of ones, i, they contain. In such a case, we obtain

$$E_G(\lambda) = (M-1)\sum_{i=0}^{q}\binom{q}{i}(\alpha^i(1-\alpha)^{q-i})^2. \tag{6}$$

In particular, we verify that for $\alpha = 1/2$, $E_G(\lambda)$ corresponds to the independent identically distributed uniform case. Fig. 6 shows simulation results based on one million random and uniformly distributed reduced-length tags and two different CAM sizes. It shows how the expected value of the number of possible matches ($E(\lambda)$) is decreased to only one by increasing the value of the number of bits in the reduced-length tag as shown in (2) and (3). The algorithm of SCN-CAM is similar to that of the precomputation-based CAM (PB-CAM) [16], [17]. A drawback of such methods, unlike SCN-CAM, is that as the length of the tags is increased, the cycle time and the circuit complexity of the precomputation stage are dramatically increased. Furthermore, we will show that unlike the PB-CAMs, SCN-CAM can potentially narrow down the search procedure to only one comparisons with a simple computational complexity that does not grow with the increase of the tag length.

TABLE I
REFERENCE DESIGN PARAMETERS

	Parameter	Value
SCN	M	512
	N	128
	n	10
	ζ	8
	β	64
	$E(\lambda)$	1
	q	9
	c	3
	l	8
CAM	CAM type	XOR
	ML Arch.	NOR
	Supply Voltage	1.0V
	Technology	65 nm

V. CIRCUIT IMPLEMENTATION

A top-level block diagram of the implementation of SCN-CAM is shown in Fig. 3. It shows how the SCN-based classifier is connected to a custom-designed CAM array shown in Fig. 9, where an example pertaining to the operation of a 4-bit CAM is demonstrated. A 10-transistor (10T) NOR-type CAM with NOR-type ML architecture was used. The conventional NAND and NOR-type CAM architectures were also implemented for comparison purposes.

In order to implement a circuit that can elaborate the benefit of the proposed algorithm, a set of design points were selected among 15 different parameter sets with the common goal of discovering the minimum energy consumption per search, while keeping the silicon-area overhead and the cycle time reasonable. The optimum design parameters 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, where n is the number of cells attached to a local match line (LML). Since $N = 128$, the last segment of the LML is considered to include eight cells attached to it.

In SCN-CAM, we use the NOR-type CAM structure, in order to take advantage of its better noise margin and the low latency, compared with the NAND-type counterpart. We will then show how it consumes lower energy per search compared with that of a conventional NAND-type architecture—one of the low-energy architectures of CAM.

A. SCN-CAM: Architecture of SCN-Based Classifier

The SCN-based classifier in SCN-CAM architecture generates the compare-enable signal(s) for the CAM sub-blocks attached to it. The architecture of the SCN-based classifier is shown in Fig. 7. It consists of c κ-to-l one-hot decoders, c SRAM modules of size $l \times M$ each, M $c-$input AND gates, M/ζ $\zeta-$input OR gates, and M/ζ 2-input NAND gates. Each row of an SRAM module stores the connections from one tag to its corresponding output neuron. Each reduced-length tag of length q is thus divided into c subtags of κ bits each, where each subtag creates the row address of each SRAM module.

1) Training: During the training process, the SRAM modules store the connection values between the input tags and their corresponding outputs, which are later used in the decoding process. The training process of the SCN-based classifier in hardware is similar to that of a conventional CAM in principle.

First, an input tag is reduced in length to q bits, and segmented into c parts. Each segment is then presented to the corresponding one-hot decoder, as shown in Fig. 7, to determine the row-address of the SRAM module corresponding to the segmented tag's cluster. The association between a tag and its corresponding output is written in the SRAM module by accessing one row per SRAM module (i.e., one neuron per cluster in $\mathrm{P_I}$), and writing a 1 into the ith bit of the M-bit SRAM row, corresponding to the ith neuron in $\mathrm{P_{II}}$. This process takes a single clock cycle per entry per SRAM module, M cycles in total if parallel writing in the SRAM modules is possible, and $M \times c$ cycles otherwise.

2) Decoding: The decoding process shown in (1) is implemented using the structure of the SRAM modules, and the c-input AND gates. The input tag is first reduced in length to q bits, and segmented into c equal-length parts. Each segment is presented to its corresponding one-hot decoder, as shown in Fig. 7, that determines which row of the SRAM is to be accessed. The location of this row corresponds to the index of an activated neuron in $\mathrm{P_I}$. In order to read the stored values in the SRAM, a sense amplifier is not required since the bit lines are only attached to few cells. Furthermore, a cell storing a 1 is the point of interest because it determines an active connection. Therefore, in order to reduce the switching activity of the successive logical stages, the complementary bit line (BL') is used to read the values of the SRAM cells since it is pulled down during the read in case of reading a 1.

In each SRAM module, the accessed row is the only row that can contain the information leading to the activation of a neuron in $\mathrm{P_{II}}$ and inherently eliminates unnecessary $w_{(i,j)(i')} \bigwedge v_{(i,j)}$ operations between the connection values and the neural values in (1). In other words, since there are never any ambiguities in $\mathrm{P_I}$, instead of calculating the neural values in $\mathrm{P_{II}}$ by computing all of the possible logical AND and OR operations in (1), only those connections coming from the activated neurons in $\mathrm{P_I}$ are used. This simplification is possible by integrating the one-hot decoders and the SRAM modules in a configuration shown in Fig. 7. Since there exists $c \times l$ SRAM modules in the SCN, the learning process, or dynamically updating the SCN-based classifier's connections only takes l clock cycles, and is not dependent on M. Therefore, the delay required by the SCN-based classifier to update its connections is by far less than what would be required to access and retrieve values from the processor that keeps track of the tags.

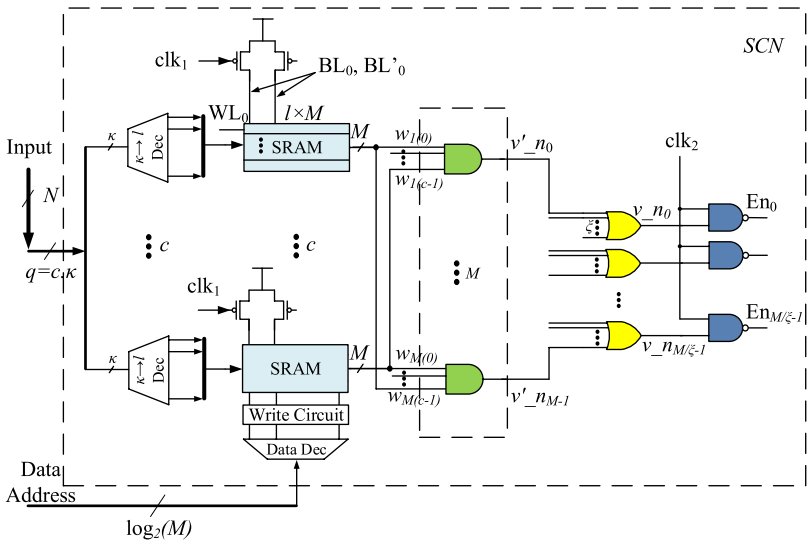

Fig. 7. Simplified schematic of the SCN-based classifier generating compare-enable signals for the CAM array.

3) Updating: Updating the network with a new entry is according the approach that was explained in Section IV-A2. First, to delete a previously trained entry, one row per SRAM-block, corresponding to the entry that is being deleted, is read into registers. Then, the ith bit in each read row is converted to a 0 to remove the connections from the ith neuron in P_{II} to the corresponding neuron in P_I. Finally, the modified row is written back into the SRAM-block. The deletion process thus takes two clock cycles. To update an entry after deletion, the new entry is added to the network using the same approach as in the training process.

B. SCN-CAM: CAM Architecture

In order to exploit the prominent feature of the SCN-based associative memory in the classification of the search data, a conventional CAM array is divided into sufficient number of compare-enabled sub-blocks such that: 1) the number of sub-blocks are not too many to expand the layout and to complicate the interconnections and 2) the number of sub-blocks should not be too few to be able to exploit to energy-saving opportunity with the SCN-based classifier. Consequently, the neurons in P_{II} are grouped and ORed as shown in Fig. 7 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 of entries of the CAM, and ζ is the number of CAM-rows per sub-block in the hierarchical arrangement as shown in Fig. 9, with schematic details in Fig. 10. The number of cells attached to an LML is denoted by n. 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

$$\psi = \left[1 - \left(1 - \frac{1}{\beta}\right)^{1+E(\lambda)}\right] \cdot \beta. \tag{7}$$

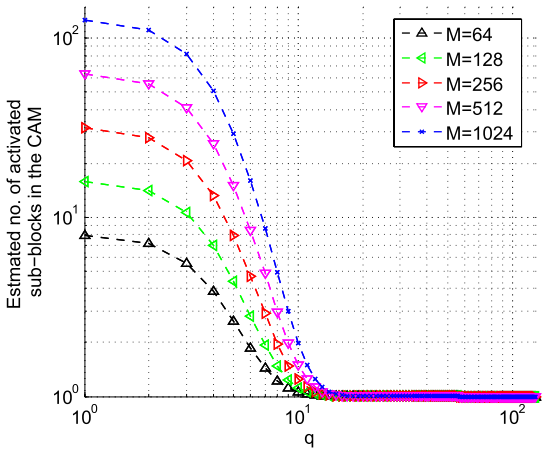

Fig. 8. Number of activated sub-blocks in the CAM (ψ) versus the number of bits in the reduced-length tag (q).

Fig. 8 shows the number of activated sub-blocks for various values of M, while sweeping q. It shows how it is possible to reduce β to only one sub-block by increasing the value of q sufficiently depending on the number of entries in the CAM array. Each sub-block has pass-gate devices attached to the SLs, which are controlled by the compare-enable signals from the SCN-based classifier. Furthermore, the precharging process of the MLs are also controlled by the SCN-based classifier. This way, the dynamic energy consumption due to both charging of the SLs and the MLs can be controlled using the same compare-enable signal(s) generated by the SCN-based classifier. Since very few sub-blocks are activated using SCN-CAM, it is possible to exploit the low-latency feature of the NOR-type architecture instead of the NAND-type counterpart for the CAM part of SCN-CAM. Because the energy saving opportunity is achieved using the architectural modification of the search procedure, we will still significantly reduce the energy consumption compared with that of the conventional NAND-type architecture while taking the advantage of the high-speed feature of NOR-type CAMs. For ultralow power applications, a NAND-type CAM-cell may also be used with the cost of speed.

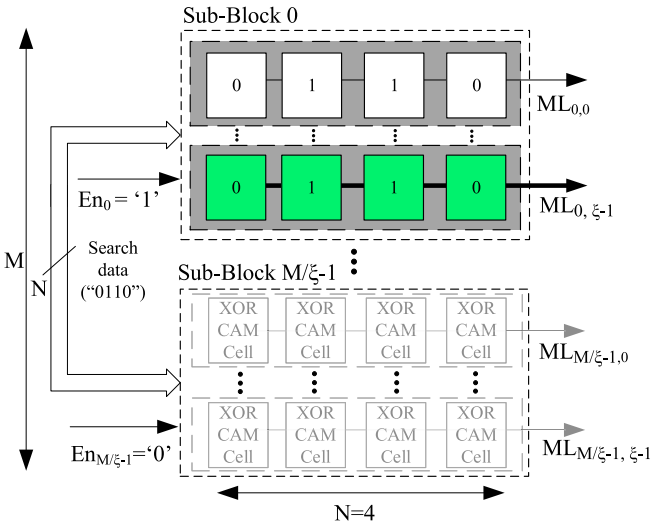

Fig. 9. Simplified array organization of the proposed CAM architecture showing an example when $N = 4$, search data word is 0110 and $En_0 = 1$. The sub-block compare-enable signals are generated by the SCN-based classifier.

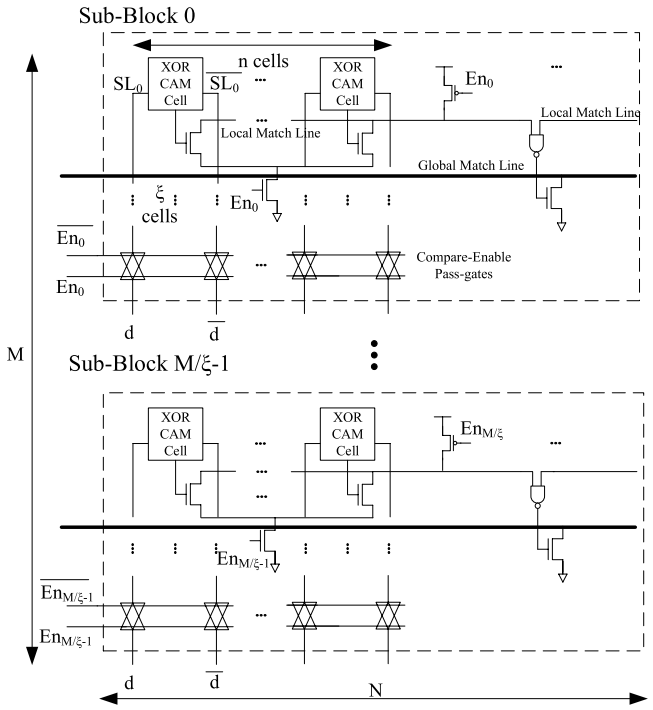

Fig. 10. Simplified schematic view of the proposed CAM array. The compare-enable signals for the CAM sub-blocks are generated by the SCN-based classifier.

The total number of sub-blocks can be selected depending on the silicon-area availability since each sub-block will slightly increase the silicon area. If the input data word is not uniformly distributed, more sub-blocks will be activated during a search consuming higher amounts of energy while the accuracy of the final output is not affected (Fig. 14). Therefore, a false-negative output is never generated. However, since the full length of the tag is not used in SCN-CAM, it is possible to select the reduced-length tag bits depending on the application and according to a pattern to reduce the tag correlation.

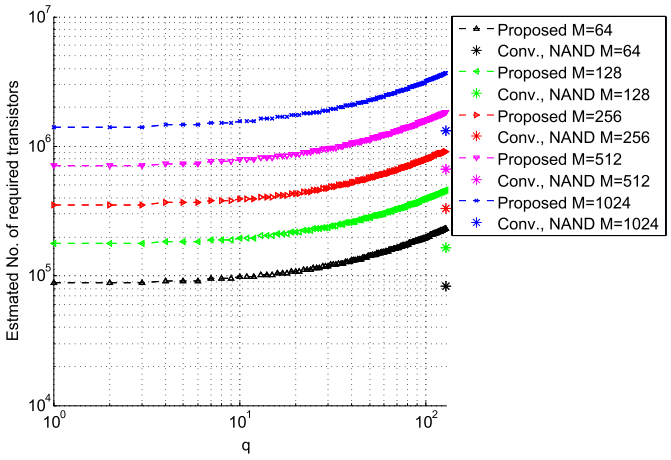

Fig. 11. Estimated number of required transistors for SCN-CAM designed for 128×40 and 512×40 CAMs, and comparing them with the conventional low-power NAND-type CAMs of the same size for various truncated tag bits (q).

VI. Circuit Evaluation

A complete circuit for SCN-CAM was implemented and simulated using HSPICE and TSMC 65-nm CMOS technology according to Table I parameters, including full dimensions of CAM arrays, SRAM arrays, logical gates, and extracted parasitics from the wires in the physical layout.

A wave-pipelining approach has been followed for clk_1 and clk_2 signals in Fig. 7 to integrate the operation SCN-based classifier and the CAM sub-blocks. This approach is verified in terms of reliability and the latency under worst-case process variations [slow–slow (SS) corner for latency and fast-fast corner for reliability]. Other methods, such as registered pipelining [35], are also possible, where $M/\zeta - 1$ registers are placed after the OR gates in Fig. 7. This way, the frequency of operation is determined by taking the minimum reliable frequency between clk_1 and clk_2.

A. Energy Consumption Model

To investigate the energy consumption of SCN-CAM for various design parameters, such as q, c, and the effect of nonuniform input distributions, the energy consumption of SCN-CAM can be modeled as

$$\begin{aligned} E_{\text{Total}} &= E_{\text{SCN}} + E_{\text{CAM}} \\ E_{\text{SCN}} &= c \cdot E_{\text{Dec}} + M \cdot c \cdot E_{\text{SRAM}_{\text{acc}}} + (l-1) \cdot M \\ &\quad \cdot c \cdot E_{\text{SRAM}_{\text{idle}}} + E_{\lambda} \cdot E_{\text{AND}_c} + \psi \cdot E_{\text{OR}_{\zeta}} \\ E_{\text{CAM}} &= (\psi \cdot \zeta - 1) \cdot N \cdot E_{\text{CAM}_{\text{mismatch}}} + N \cdot E_{\text{CAM}_{\text{match}}} \\ &\quad + (M/\zeta - \psi) \cdot \zeta \cdot N \cdot E_{\text{CAM}_{\text{stat}}} \end{aligned} \tag{8}$$

where the total energy consumption is divided into the energy consumption in the SCN-based classifier (E_{SCN}), and the CAM sub-blocks (E_{CAM}). The SCN-based classifier's contribution to the energy consumption includes decoders, E_{Dec}, SRAMs (accessed, $E_{\text{SRAM}_{\text{acc}}}$, and idle, $E_{\text{SRAM}_{\text{idle}}}$), and the logical gates to perform the GD and to generate the compare-enable signals for the CAM array. For every search operation, one row in each SRAM is accessed and thus, the rest of the rows is in idle states, in which there exists a switching

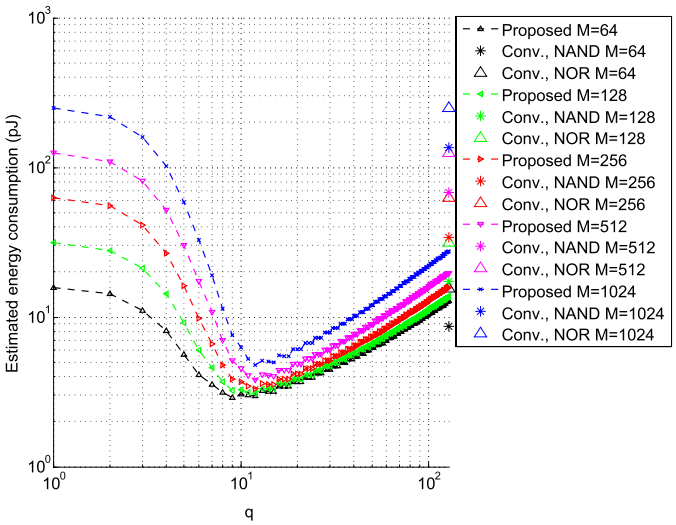

Fig. 12. Total estimated dynamic energy consumption per search for SCN-CAM, and comparing them with the conventional low-power NAND-type CAM design for various values of the reduced-length tag (q).

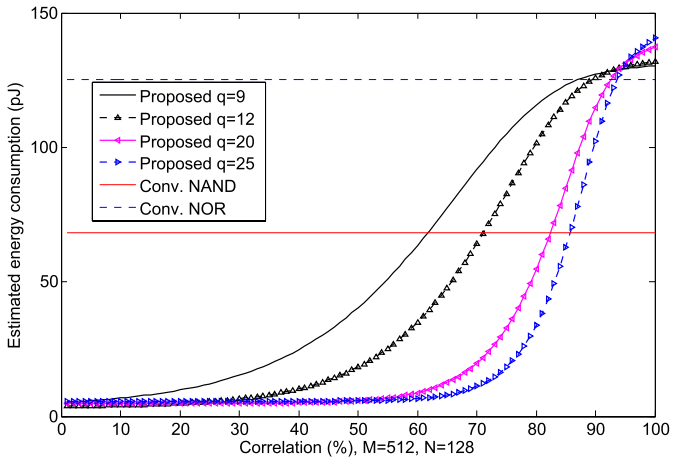

Fig. 14. Total estimated energy consumption per search and match for SCN-CAM, and comparing them with the conventional low-power NAND-type CAM design for various truncated tag bits (q).

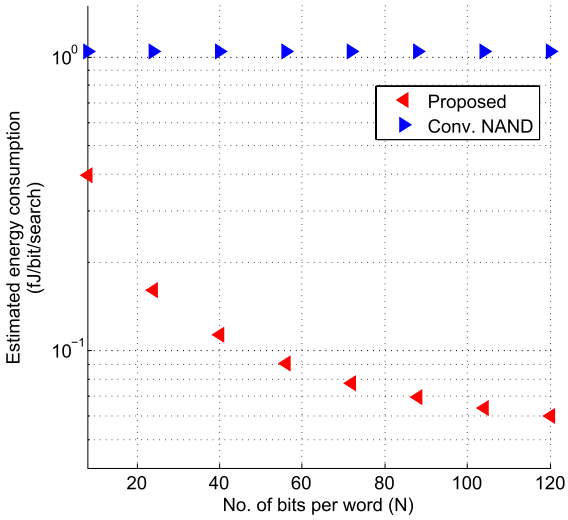

Fig. 13. Estimated energy consumption per bit per search of the proposed NOR architecture, and the conventional NAND-type CAM for various values of word lengths (N).

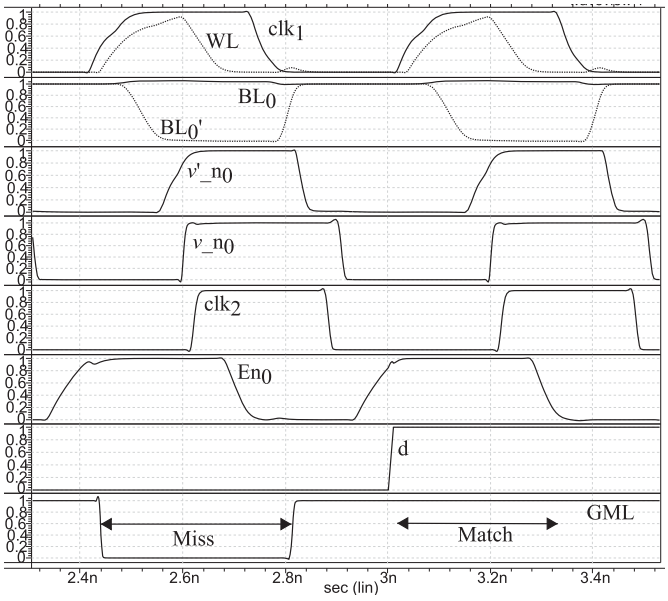

Fig. 15. Simulation results for SCN-CAM based on reference design parameters in Table I.

activity in their bit lines. Therefore, we have divided the energy consumption of the SRAMs into the corresponding states.

Furthermore, the CAM portion of the energy model consists of match ($E_{\mathrm{CAM_{match}}}$) and mismatch ($E_{\mathrm{CAM_{mismatch}}}$) portions, whose values depend on the number of ambiguities discussed in Section IV-B. Static energy consumption of the idle CAMs ($E_{\mathrm{CAM_{stat}}}$) has also been included due to the presence of leakage current occurring in advanced technologies. The estimation of the number of required transistors follows a similar model.

B. Area Estimation and Simulation Results

Fig. 11 shows the estimated overhead of the number of transistors in SCN-CAM for various number of entries of the CAM in comparison with the conventional design. For design selections in Table I, this overhead is only 3.4% compared with that of the conventional CAM. The silicon area of the SCN-based classifier can be estimated considering the area of the decoders, the SRAM arrays, the precharge devices, the read and write circuits, the interconnections, and the standard cells. Similarly, the area of the CAM array can be estimated by considering the gaps between the CAM sub-blocks, pass-gate transistors, read and write circuits. The area overhead is estimated to be 10.1% higher than that of the conventional CAM design, for design selections in Table I.

In the simulations for measuring the energy consumption and the cycle time (/bit/search), on average half of the data bits were assumed to mismatch in case of a word mismatch. In Fig. 12, the relationship between the dynamic energy consumption of SCN-CAM and the tag length is depicted for various number of entries of the CAM in comparison with the conventional CAMs. The estimated energy consumption is obtained based on (6), and the extracted values for energy consumption using HSPICE simulations. As the value of q is increased, the energy consumption is decreased as well

TABLE II
RESULT COMPARISONS

	PB* [16]	PF-CDPD* [33]	Hybrid [12]	STOS** [15]	HS-WA [1]	Ref. NAND	Ref. NOR	Proposed
Configuration	128 × 30	256×128	128×32	256×144	128×128	512×128	512×128	512×128
CAM type	BCAM	BCAM	BCAM	BCAM	BCAM	BCAM	BCAM	BCAM
Cell type	NOR	NAND	NAND-NOR	NAND	NAND-NOR	NAND	NOR	NOR
Technology	0.35 μm	0.18 μm	0.18 μm	90 nm	32 nm	65 nm	65 nm	65 nm
Cycle time [ns]	10	2.10	0.60	1.359	0.145	2.1	0.5	0.60
Scaled cycle time [ns]	0.563	1.365	0.39	0.982	0.295	2.10	0.50	0.60
Energy [fJ/bit/search]	86	2.33	1.30	0.162	1.070	1.040	1.910	0.078
Scaled energy [fJ/bit/search]	2.112	0.256	0.145	0.117	2.173	1.04	1.91	0.078

* Measurement results (without pads).
** The cycle time of this CAM, unlike SCN-CAM, is affected by 5.2× in a non-uniform distribution scenario of the input patterns.

since the number of comparisons is reduced but up to a point until the energy consumption of the SCN-based classifier itself would dominate that of the CAM array. Therefore, the energy consumption of the SCN-based classifier is not dependent on the original tag length, and rather on the number of entries in the CAM array.

Fig. 13 shows the effect of the word length on the energy consumption in comparison with the conventional design. The original tag length (N) does not change the architecture and the energy consumption of the SCN-based classifier. Furthermore, due to the small size of the sub-blocks, the search power of SCN-CAM is much smaller compared with that of the conventional. Consequently, as N is increased, the energy consumption per-bit-per-search is decreased in SCN-CAM while it stays constant in the conventional CAM. It also implies an advantage of SCN-CAM over PB-CAMs, such as in [16] and [17], where longer tag lengths increase the energy consumption as well as the precomputation delay. This is because longer tags will increase the complexity of the adders and the number of comparisons. Fig. 14 shows the effect of the correlation in the entries of the CAM on the energy consumption for various lengths of the reduced-length tag. The expected value of the number of sub-blocks has been calculated according to (5). The correlation effect is applied on the inputs by creating similarities within their contents. For example, a 10% correlation means that 10% of the bit values are similar in all input patterns. It is therefore observed that larger correlation costs energy consumption although it does not affect the performance as in [15]. Fig. 15 shows simulation results for measuring the cycle time of SCN-CAM for the selected design parameters shown in Table I. It shows the worst-case cycle time, where the last input of the AND/OR gates are pulled up and under SS corner. It also shows the wave-pipelining method of clk_1 and clk_2 signals as shown in Fig. 7. clk_2 is simply a delayed version of clk_1.

The cycle time is measured by the maximum reliable frequency of operation in the worst-case cycle time (SS) scenario. Table II summarizes the comparisons of the cycle time and the energy consumption between SCN-CAM and a collection of other related work including our own implementation of the conventional NAND-type and NOR-type CAMs. The technology-scaled version (to 65-nm CMOS) of these results are evaluated according to the method described in [18]. Unless otherwise indicated, the reported results are based on simulations. The energy consumption of SCN-CAM is 4.08%, 8.02%, 3.59%, 66.7%, 53.89%, 30.5%, and 3.69% of that of the referenced NOR-based, referenced NAND-based, [1], [12], [15], [16], and [32], respectively. On the other hand, the cycle time of SCN-CAM is 132%, 31.4%, 224%, 67.2%, 169%, 48.4%, and 107% of that of the referenced NOR, referenced NAND, [1], [12], [15], [16], and [32], respectively. Although the energy consumption of the CAM presented in [15] is small compared with most of the others, the cycle time is significantly increased (5.2×) when the search-data patterns are correlated, whereas the cycle time of SCN-CAM remains unchanged in a similar situation.

The required silicon area of SCN-CAM is estimated to be 10.1% larger than that of the conventional NAND-type counterpart mainly due to the existence of the gaps between the SRAM blocks of the SCN-based classifier. Consequently, the silicon area can be reduced if fewer sub-blocks are used with the cost of energy consumption.

VII. CONCLUSION

In this paper, the algorithm and the architecture of a low-power CAM are introduced. The proposed architecture (SCN-CAM) employs a novel associativity mechanism based on a recently developed family of associative memories based on SCNs.

SCN-CAM is suitable for low-power applications, where frequent and parallel look-up operations are required. SCN-CAM employs an SCN-based classifier, which is connected to several independently compare-enabled CAM sub-blocks, some of which are enabled 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 achieved without retraining the network entirely. With optimized lengths of the reduced-length tags, SCN-CAM eliminates most of the comparison operations

given a uniform distribution of the reduced-length inputs. Depending on the application, nonuniform inputs may result in higher power consumptions, but does not affect the accuracy of the final result. In other words, a few false-positives may be generated by the SCN-based classifier, which are then filtered by the enabled CAM sub-blocks. Therefore, no false-negatives are ever generated.

Conventional NAND-type and NOR-type architectures were also implemented in the same process technology to compare SCN-CAM against, along with other recently developed CAM architectures. It has been estimated that for a case study design parameter, the energy consumption and the cycle time of SCN-CAM are 8.02%, and 28.6% of that of the conventional NAND-type architecture, respectively, with a 10.1% area overhead. Future work includes investigating sparse compression techniques for the matrix storing the connections in order to further reduce the area overhead.

REFERENCES

[1] A. Agarwal *et al.*, "A 128×128 b high-speed wide-and match-line content addressable memory in 32 nm CMOS," in *Proc. ESSCIRC*, Sep. 2011, pp. 83–86.

[2] Y.-J. Chang and M.-F. Lan, "Two new techniques integrated for energy-efficient TLB design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 1, pp. 13–23, Jan. 2007.

[3] H. Chao, "Next generation routers," *Proc. IEEE*, vol. 90, no. 9, pp. 1518–1558, Sep. 2002.

[4] N.-F. Huang, W.-E. Chen, J.-Y. Luo, and J.-M. Chen, "Design of multi-field IPv6 packet classifiers using ternary CAMs," in *Proc. IEEE Global Telecommun. Conf.*, vol. 3. 2001, pp. 1877–1881.

[5] M. Meribout, T. Ogura, and M. Nakanishi, "On using the CAM concept for parametric curve extraction," *IEEE Trans. Image Process.*, vol. 9, no. 12, pp. 2126–2130, Dec. 2000.

[6] M. Nakanishi and T. Ogura, "A real-time CAM-based Hough transform algorithm and its performance evaluation," in *Proc. 13th Int. Conf. Pattern Recognit.*, vol. 2. Aug. 1996, pp. 516–521.

[7] L.-Y. Liu, J.-F. Wang, R.-J. Wang, and J.-Y. Lee, "CAM-based VLSI architectures for dynamic Huffman coding," *IEEE Trans. Consum. Electron.*, vol. 40, no. 3, pp. 282–289, Aug. 1994.

[8] C.-C. Wang, C.-J. Cheng, T.-F. Chen, and J.-S. Wang, "An adaptively dividable dual-port BiTCAM for virus-detection processors in mobile devices," *IEEE J. Solid-State Circuits*, vol. 44, no. 5, pp. 1571–1581, May 2009.

[9] B. Wei, R. Tarver, J.-S. Kim, and K. Ng, "A single chip Lempel–Ziv data compressor," in *Proc. IEEE ISCAS*, May 1993, pp. 1953–1955.

[10] S. Panchanathan and M. Goldberg, "A content-addressable memory architecture for image coding using vector quantization," *IEEE Trans. Signal Process.*, vol. 39, no. 9, pp. 2066–2078, Sep. 1991.

[11] T. Juan, T. Lang, and J. Navarro, "Reducing TLB power requirements," in *Proc. Int. Symp. Low Power Electron. Des.*, Aug. 1997, pp. 196–201.

[12] 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.

[13] Z. Lei, H. Xu, D. Ikebuchi, H. Amano, T. Sunata, and M. Namiki, "Reducing instruction TLB's leakage power consumption for embedded processors," in *Proc. Int. Green Comput. Conf.*, Aug. 2010, pp. 477–484.

[14] 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.*, vol. 20, no. 10, pp. 1909–1913, Oct. 2012.

[15] N. Onizawa, S. Matsunaga, V. C. Gaudet, and T. Hanyu, "High-throughput low-energy content-addressable memory based on self-timed overlapped search mechanism," in *Proc. Int. Symp. Asynchron. Circuits Syst.*, May 2012, pp. 41–48.

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

[17] 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.*, vol. 16, no. 3, pp. 331–335, Mar. 2008.

[18] 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*, vol. 46, no. 2, pp. 507–519, Feb. 2011.

[19] 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.

[20] V. Gripon and C. Berrou, "Sparse neural networks with large learning diversity," *IEEE Trans. Neural Netw.*, vol. 22, no. 7, pp. 1087–1096, Jul. 2011.

[21] V. Gripon and C. Berrou, "Nearly-optimal associative memories based on distributed constant weight codes," in *Proc. ITA Workshop*, Feb. 2012, pp. 269–273.

[22] H. Jarollahi, N. Onizawa, V. Gripon, and W. J. Gross, "Architecture and implementation of an associative memory using sparse clustered networks," in *Proc. IEEE ISCAS*, Seoul, South Korea, May 2012, pp. 2901–2904.

[23] H. Jarollahi, N. Onizawa, V. Gripon, and W. J. Gross, "Reduced-complexity binary-weight-coded associative memories," in *Proc. IEEE ICASSP*, May 2013, pp. 2523–2527.

[24] H. Jarollahi, N. Onizawa, and W. J. Gross, "Selective decoding in associative memories based on sparse-clustered networks," in *Proc. IEEE Global Conf. Signal Inf. Process.*, Dec. 2013, pp. 1270–1273.

[25] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

[31] 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.

[32] J.-S. Wang, H.-Y. Li, C.-C. Chen, and C. Yeh, "An AND-type match-line scheme for energy-efficient content addressable memories," in *IEEE ISSCC Dig. Tech. Papers*, vol. 1. Feb. 2005, pp. 464–610.

[33] 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-State Circuits*, vol. 25, no. 5, pp. 1158–1165, Oct. 1990.

[34] 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.

[35] 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.

Hooman Jarollahi (S'09) received the B.A.Sc. and M.A.Sc. degrees in electronics engineering from Simon Fraser University, Burnaby, BC, Canada, in 2008 and 2010, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada.

He was a Visiting Scholar with the Research Institute of Electrical Communication, Tohoku University, Sendai, Japan, from 2012 to 2013. His current research interests include design and hardware implementation of energy-efficient and application-specific VLSI systems, such as associative memories and content-addressable memories.

Mr. Jarollahi was a recipient of the Teledyne DALSA Award in 2010, for which he presented a patented architecture of a power and area-efficient SRAM.

Vincent Gripon received the M.S. degree from École Normale Supérieure of Cachan, Cachan, France, and the Ph.D. degree from Télécom Bretagne, Brest, France.

He is a Permanent Researcher with Institut Mines-Télécom, Télécom Bretagne. His intent is to propose models of neural networks inspired by information theory principles, what could be called informational neurosciences. He is also the Co-Creator and Organizer of an online programming contest named TaupIC, which targets the French top undergraduate students. His current research interests include information theory, neuroscience, and theoretical and applied computer science.

Naoya Onizawa (M'09) received the B.E., M.E., and D.E. degrees in electrical and communication engineering from Tohoku University, Sendai, Japan, in 2004, 2006, and 2009, respectively.

He was a Post-Doctoral Fellow with Tohoku University from 2009 to 2011, the University of Waterloo, Waterloo, ON, Canada, in 2011, and the McGill University, Montreal, QC, Canada, from 2011 to 2013. He is currently an Assistant Professor with the Frontier Research Institute for Interdisciplinary Sciences, Tohoku University. His current research interests include the energy-efficient VLSI design based on asynchronous circuits and multiple-valued circuits, and their applications, such as LDPC decoders, associative memories, and network-on-chips.

Dr. Onizawa was a recipient of the Best Paper Award at the IEEE Computer Society Annual Symposium on VLSI in 2010.

Warren J. Gross (SM'10) received the B.A.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, and the M.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, ON, in 1996, 1999, and 2003, respectively.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, McGill University, Montral, QC, Canada. His current research interests include the design and implementation of signal processing systems and custom computer architectures.

Dr. Gross is currently the Chair of the IEEE Signal Processing Society Technical Committee on Design and Implementation of Signal Processing Systems. He served as a Technical Program Co-Chair of the IEEE Workshop on Signal Processing Systems in 2012, and as the Chair of the IEEE ICC 2012 Workshop on Emerging Data Storage Technologies. He served as an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING. He has served on the Program Committees of the IEEE Workshop on Signal Processing Systems, the IEEE Symposium on Field-Programmable Custom Computing Machines, and the International Conference on Field-Programmable Logic and Applications, and has served as the General Chair of the 6th Annual Analog Decoding Workshop. He is a licensed Professional Engineer in the Province of Ontario.