pbCAM: probabilistically-banked Content Addressable Memory

Tolga Soyata, John Liobe
University of Rochester, Dept. of Electrical and Computer Engineering, Rochester, NY 14627
{soyata, liobe}@ece.rochester.edu

***Abstract*—Content Addressable Memories find wide use in network routers and certain image processing applications. However, their use is limited due to their high power demand resulting from their high activity factor. A banked CAM, on the other hand, partitions the entire CAM into smaller banks to cut down on excessive searches, while it may reduce the effective CAM size when the data entries are unevenly distributed. A new banked CAM design is introduced in this paper which achieves significant energy and power savings through the use of Bloom Filters by effectively decoupling data elements from their bank index. Simulation results show energy savings of nearly an order of magnitude.**

I. Introduction

A CAM facilitates search operations based on the *content*, rather than the *physical location* of the data. This allows for much faster database operations, such as insertion, deletion, and search. The speed improvement primarily comes from the ability to search multiple entries in parallel. Although the speed of the operation is improved, very large CAMs cannot be built due to the energy and power limitations. This is due to the 100% activity factor CAMs have during their search operation. At some point, it becomes technologically infeasible to dissipate the amount of power required for the CAM. For example, the state of the art CAM consumes 6fJ per search per bit as per 2006 data [1], and the straightforward scaling of this number based on ITRS projections [2] yields a 150W power consumption projection for a 256KB CAM by 2012, nearly at the limit of conventional cooling for any IC. Although sophisticated circuit design techniques, such as pipelining and banking can reduce this number to a smaller value, such as, 50W, it is still evident that, a CAM in the order of Gigabytes is not feasible using today's technology. Using ITRS [2] and Moore's law, one can predict that in 2025, a CAM will only have 1M entries for the same power consumption.

A variant of CAM, the ternary CAM (or TCAM) stores each element in two bits, allowing three-valued (i.e., '0', '1', 'x') logical operations. This style CAM enables additional operations at the expense of storage area and power penalty. Due to the size limitation, CAM/TCAMs have been confined into a limited application space, such as, network routers, to perform packet classification and routing. In network router applications, much smaller (T)CAMs are needed and the search speed is of primary concern, reducing the negative impact of the high power consumption.

There has been considerable work in the area of CAM design to reduce the power and/or energy consumption of (T)CAMs [3]–[5]. Although numerous techniques have been proposed to make CAMs more energy-efficient, the power savings have been limited, thereby significantly narrowing the application space of CAMs. In this paper, we propose an energy-efficient CAM that eliminates one of the most important hurdles in building large banked-CAMs: bank conflicts. Our proposed pbCAM design de-couples data values from their physical location, thereby permitting much larger CAMs to be constructed within the same power budget.

This paper is organized as follows. In Section II, background information is provided for CAMs and Bloom Filters. Section III and Section IV provide an overview and operational details pbCAM, respectively. Our concept is simulated and evaluated using Cadence design tools in Section V. Related work is provided in Section VI and conclusions and future work are given in Section VII.

II. Background and Motivation

A. Content Addressable Memory (CAM)

CAMs provide the payload associated with a key [1]. As an example, if one was to use a CAM to store zip codes and their associated city names, the *key* would be the city names, and the zip codes would be the *payload* (i.e., the result of the search). A traditional CAM is shown in Figure 1, where the term *tag* is used interchangeably with the *key*. A traditional N-entry CAM searches all N elements in a single cycle to match a single key, thus performing $N-1$ wasteful searches. Therefore, this approach is very energy inefficient.

One potential improvement is to use a banked CAM structure as shown in Figure 2, where the prefix of the data element (e.g., the MSB 4 bits) is used to pre-eliminate a large portion of the banks (e.g., 15). Only a single bank actually performs the query. This approach has a drawback: Since certain data elements can only reside in certain banks, the effective size of the CAM could be drastically reduced for non-uniformly distributed data elements (e.g., having a lot more city names that start with the letter M). This asymmetric distribution of the entries could eventually negate the savings from banking. We propose a new solution that eliminates these *bank conflicts*.

B. Bloom Filters

Bloom Filter (BF) was introduced in [6] to provide a tool for determining the existence of an entry in a dataset with a high probability. The most common BF has no false negatives, and a small false positive rate, p. The BF of a dataset is calculated

by computing a hash function, $H(D)$ on each data element D in the dataset and maintaining the aggretaed result in the BF as the logical-OR of all of the $H(D)$ values. By using k different hash functions (e.g., k=2), the false positive ratio can be decreased. As an example, assume that a data set contains three 16-bit entries, 6D34h, B3A5h, and 9A05h. Assume a hash function $H()$, which, when applied to these data entries, yields 24-bit values as follows: $H(6D34h)$=080000h, $H(B3A5h)$=004000h, $H(9A05h)$=100000h. The resulting BF for these three data elements can be calculated by the logical OR of these three hash values, i.e., $BF(D)$=184000h.

Using this Bloom filter, if one needs to perform a search for a certain data element Q which has a hash value of, say, $H(Q)$=000100h, since the BF is 184000h, this signals a guaranteed FALSE, which means that the data element does not exist in the dataset. This is determined by examining the individual bits of the $H(Q)$ and the corresponding bits of BF. For each 1 bit of the $H(Q)$, if the corresponding BF bit is 0, the element is guaranteed to be absent, otherwise, either the data element exists, or a false positive is being signaled by the BF. For an N-element dataset, a m-bit BF, the maximum false positive rate can be approximated by the following formula:

$$p = (1 - e^{\frac{-k(N+0.5)}{m-1}})^k \quad (1)$$

where k is the number of different hash functions [7], assuming a uniform distribution of the storage elements. As the BF length (m) increases, the false positive probability decreases. This structure will form the basis of our pbCAM design, where the existence of a data element will be tested using a Bloom filter before the query is performed.

III. pbCAM: Probabilistically Banked CAM

Although banked CAMs reduce unnecessary searches, their power and speed benefits can be completely nullified by bank conflicts. To solve this issue, we propose the probabilistically-banked CAM (pbCAM), which leverages Bloom Filters (BF) at the input of each bank. Figure 3 shows BFs recording the new elements inserted into its associated bank.

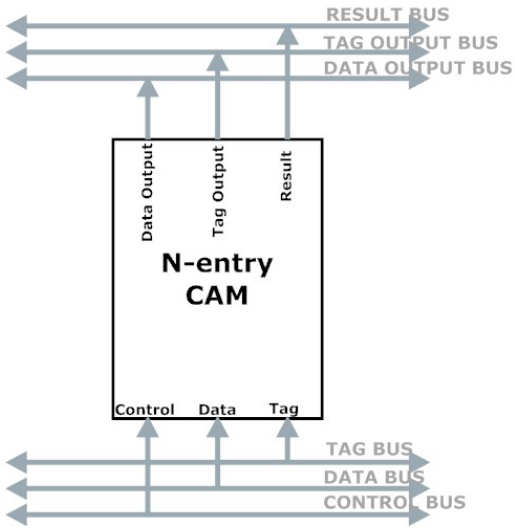

Fig. 1. Traditional CAM structure. N entries are searched in parallel.

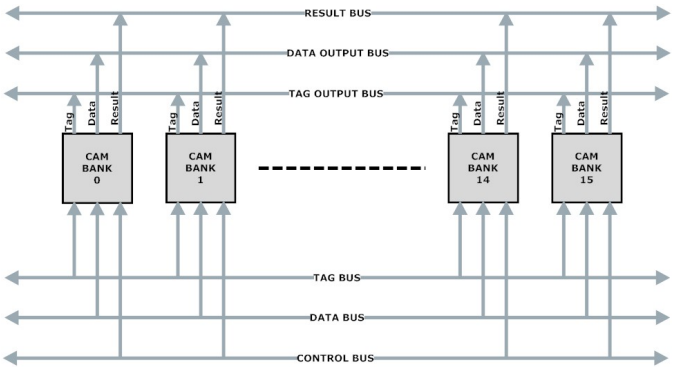

Fig. 2. Banked CAM structure to divide N elements into B banks.

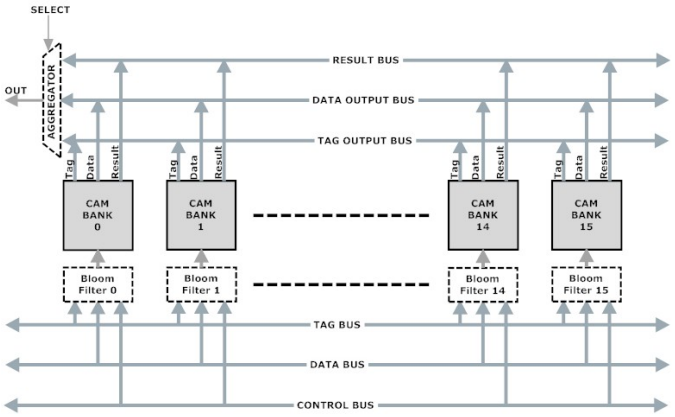

Fig. 3. Proposed pbCAM design. The Bloom filter attached to each bank estimates the likelihood of the existence of a key.

Assume an example 16-bank (B=16) CAM using a 8192-bit BF for each bank (m=8192) and 2 hash functions (k=2). The false positive rate is p=5% from Equation 1. By sifting the queries using this BF before they reach the bank, one can ascertain if this bank contains the entry or not with a 0% false negative, and a 5% false positive ratio. Using this BF, one bank (the one actually containing the entry) will correctly accept the query and the other 15 banks will erronuously accept the query with a 5% probability, this yielding an average activity factor of, $\frac{1+p(B-1)}{B}$ (≈ 0.11 in this example), instead of the ideal $\frac{1}{B}$ (≈ 0.06). These spurious ($p(B-1)$) searches consume energy without benefit. However, reducing these extra queries requires a larger BF (i.e., higher m) according to Equation 1, thereby increasing the die area, and eventually increasing the energy consumption. This intricate relationship between m and p affords pbCAM a design freedom to trade-off die area against energy consumption, creating a unique opportunity that doesn't exist in traditional banked CAMs.

pbCAM has the ability to store a data element in any bank without restrictions using an appropriate hash function which decouples the D from the $H(D)$. Such hash functions are common, especially in cryptography (i.e., crypto-strength hash functions), where one of the important design parameters is the independence of the output of a hash function from its input. Achieving a uniform activity level of $\frac{1+p(B-1)}{B}$ in each pbCAM bank creates an opportunity to reduce the operation frequency of each bank by a factor of $\frac{1+p}{B}$. This frequency reduction, combined with a voltage reduction yields super-

linear energy savings in the pbCAM design.

IV. pbCAM Circuit Design and Operation

One key observation in the design of pbCAM is that, the hash function of a data element being searched, $H()$, needs to be calculated only once which can be used by every bank. Therefore, for a synchronous pipelined CAM IC that is processing s queries per second, s Hash calculations per second are necessary, which can be done at the main entry point of the CAM before the query reaches the pbCAM banks. $H(D)$ will be required by the Bloom Filter (BF) of *every bank* to determine whether there is a pre-match, but, the actual corresponding data D will only be required by the bank that contains the element, as well as the banks that issued a false positive match signal. This indicates that, the design of the $H()$ function calculator can be decoupled from the Bloom filter itself as well as the pre-match circuitry. These separate elements will be described in the following subsections.

A. Hash Calculator

Although there has been existing work to pre-eliminate the searches partially by using the initial few bits of the data [1], this type of pre-elimination using the actual data itself proves to be a weak filter due to the necessity of forcing certain data elements into pre-determined banks. The BF-based approach described in this paper places no restriction on where each data element can reside due to the hashing. Since, based on our design, each data element only needs to be hashed once, the hash function can be chosen with primary focus on its energy consumption. Furthermore, using multiple hash functions provides design alternatives to trade-off false positives against Bloom filter size based on Equation 1.

Figure 4 depicts the proposed reference pbCAM design with B=16. The entire CAM is designed to have only a single Hash calculator which derives $H(D)$ from D right at the synchronous input of the pbCAM. The pbCAM is assumed to work at a frequency of f_{CAM} which is the clock rate of the Hash calculator. Since CAM banks work at a significantly lower rate than f_{CAM}, with a theoretical best case of $\frac{f_{CAM}}{B}$ at $p=0$, and a theoretical worst case of f_{CAM} (at an average $p=1$), and an expected rate of $f_{CAM}\frac{1+p}{B}$, the data (D) and hash values ($H(D)$) can be multiplexed on the internal CAM data bus without disrupting the overall CAM throughput.

Multiplexing of the $H(D)$ and D values is achieved by sequentially placing the $H(D)$ values on the data bus followed by the D value for the banks that responded positive to the pre-Bloom filter match. Each bank's BF outputs a match/no match flag on their BMout through the control bus, requesting the search data D. Banks that respond FALSE to the BF match simply ignore this search entry, waiting for the next $H(D)$ value in the next cycle. Note that, accepting a search entry does not exempt a bank from listening to the next request. It is possible that, two or more entries in a row cause a *hit* in the same bank. In the worst case, B entries in a row cause a hit, forcing that bank to increase its frequency dynamically, towards f_{CAM}. Alternatively, in the best case, when every bank gets a hit uniformly, each bank can slow down to a frequency of $\frac{f_{CAM}(1+p)}{B}$, while still achieving a global CAM throughput of f_{CAM}.

B. pbCAM Bank Design

The micro-architecture of the proposed pbCAM bank is shown in Figure 5. Insertion, deletion, and update requests are queued up in the TAG FIFO and Data FIFO, only if the BF responds positive. Otherwise they are discarded by this bank. This design allows the CAM bank to dynamically adjust its frequency between $\frac{f_{CAM}(1+p)}{B}$ and f_{CAM}. The pipeline control can take advantage of Dynamic Voltage and Frequency Scaling (DVFS) to translate lower frequencies into a super-linear energy advantage by reducing the frequency and the voltage of the pipeline simultaneously. The output of the results are queued up at the TAG OUT and DATA OUT registers for the aggregator to combine them.

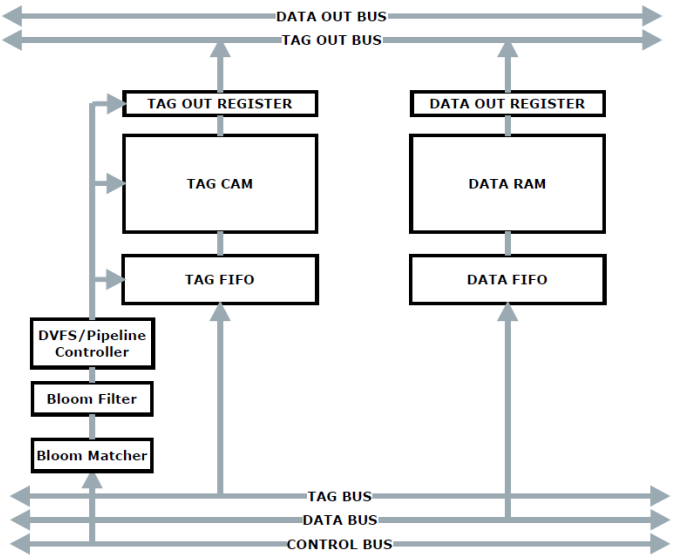

Fig. 5. pbCAM bank microarchitecture. DVFS is utilized to save energy on banks with lower search load.

The total energy that is expanded in a naive design is $E_{CB}\times B$ per cycle, where E_{CB} is the search energy of each CAM bank and B is the number of banks. The total energy per cycle in the proposed pbCAM is approximated by

$$E_{pbCAM} = E_{CB}(1+p(B-1))\times R_F^{\alpha} + B\times E_{BF} \quad (2)$$

where R_F is the reduction in frequency, and α is a value between 1 and 2, signifying the super-linear energy savings due to simultaneous frequency and voltage reduction. E_{BF} is the energy consumed by each bank's BF, and p is the false positive probability. The pbCAM achieves lower energy consumption per cycle when the following constraints are met:

$$\begin{aligned} E_{CB}\times B &> E_{CB}(1+p(B-1))\times R_F^{\alpha} + B\times E_{BF} \\ E_{CB}(B-(1+p(B-1))\times R_F^{\alpha}) &> B\times E_{BF} \\ E_{CB}(1-\frac{(1+p(B-1))R_F^{\alpha}}{B}) &> E_{BF} \end{aligned} \quad (3)$$

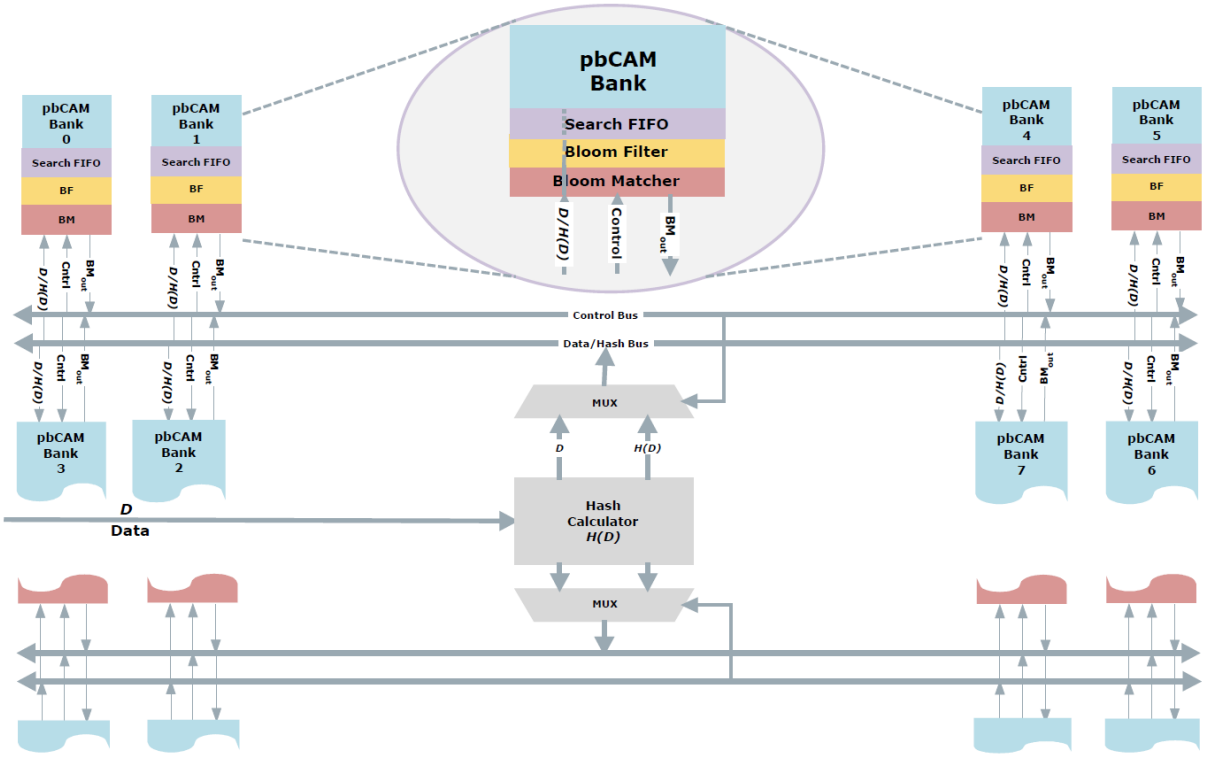

Fig. 4. pbCAM bank layout structure. Using only a single hash calculator for the entire CAM results in significant energy savings.

Substituting practical numbers, $p=0.05$, $\alpha = 1.5$, $R_F \approx 0.11$, and $B=16$, we obtain nearly an order-of-magnitude energy savings for pbCAM as compared to the traditional CAM, which will also be demonstrated through our simulations.This analysis ignores the bus traffic, which can be integrated into the formula. However, it is clear that this design offers a methodology to trade-off BF energy for CAM bank energy.

C. pbCAM Operations: Insertion, Deletion, Search

a) Insertion: This function is performed by calculating the hash value $H(D)$ of a data element D and inserting it into the selected pbCAM bank and updating the valid bit of the corresponding entry in that bank. After the insertion into the bank, the BF of the selected bank is updated as follows:

$$BF^{t+1}[n] = BF^{t}[n] + H(D) \tag{4}$$

where $+$ is the logical OR operation, and $BF[n]_t$ and $BF[n]_{t+1}$ are the stored Bloom Filter values of bank n at the synchronous clock edge t and $t+1$, i.e., before and after the insertion, respectively. Note the saturation nature of the Bloom filter, where at each update, the selectivity of the Bloom filter decays exponentially according to Equation 1. During the insertion of a data element, D into the pbCAM can be performed simply inserting it into the first available bank in a Round-Robin fashion, and updating the BF of the bank according to Equation 4, this approach does not take advantage of any of the positive features of pbCAM: 1) Since pbCAM decouples data elements from the bank index, this feature can be exploited by an intelligent centralized controller to uniformly distribute the data elements into pbCAM banks, resulting in an effectively larger BF, 2) the same intelligent central control can also be used to pre-calculate the resulting Bloom Filter for all available B banks and choose the bank that causes the lowest bit-flips, thereby lowering the probability of Bloom Filter saturation, again, effectively, increasing m, 3) insertion can be performed into the least *polluted* bank, thereby improving the effectiveness of the BFs. The *pollution* concept will be explained later when deletions are described. Note that, the vast amount of options enabled by different *insertion policies* present many exciting research opportunities which are far beyond the scope of this paper.

b) Deletion: Despite its significant energy savings, the Bloom filter presents significant challenges in performing deletions. Since the logical OR function used to update the insertions into the BFs is a one-way function, deletions are not possible without significant ramifications in pbCAM operation. Simply flipping the 1's into 0's in the BF after a deleted element will introduce *false negatives*, which contradicts many of the assumptions that allowed the pbCAM to work efficiently. This feature of BFs has been the focus of significant research [7]–[9]. There are multiple ways to handle deletions with different trade-offs. They are: 1) Doing nothing, which will increase the effective false positive rate of the associated BF, albeit with zero impact on correctness of the pbCAM, 2) Maintaining a *dirty counter*, and not changing the BF. The increase in the false positive rate could be overcome by eventually flushing the BF and reconstructing it.

c) Search: This operation is performed by comparing the hash value of a data element $H(D)$ to the Bloom filter of the bank being searched. For k hash functions, there are only k logic 1's in the hash value of $H(D)$. This presents interesting energy-savings alternatives for designing the search circuitry.

The search operation is performed as follows:

$$Match'[n] = H'(D).BF[n] \quad (5)$$

where . is the bitwise logical OR operation, $H'(D)$ is the bitwise logical inverse of $H(D)$ and $BF[n]$ is the Bloom filter value of bank n. If the result is TRUE (i.e., $Match[n]$=non-zero), either the bank contains the element, or the answer is a false positive.

D. Bloom Matcher and Bloom Filter

Equation 5 states that, for each logic 1 bit of the Bloom filter, if even a single corresponding $H(D)$ bit is zero, this bank does not contain the entry. A dynamic-NAND gate can be designed as shown in Figure 6 which determines the $Match'$ value by pre-charging the match line and letting the hash value to pull it down based on the BF value. The Bloom Filter is constructed simply by using an SRAM array of m bits. These $BF[n]$ bits are applied to the top transistor of the matching circuit to speed up the match process. Since the only transistor that suffers from source degeneration is the top transistor, by applying the steady BF value to the top transistor, the overall match speed is improved by almost completely eliminating the charge/discharge time of the top transistor. The BF value is updated only after an insertion which potentially changes certain BF bits, immediately updating the state of the top transistor, thereby nearly doubling the response time of the match circuit.

The sense amplifier (SA) included in the BF is a typical current-race implementation [1]. Although a more energy efficient SA can be designed for this specific application, the SA utilized for the CAM design described below is re-used here. Looking ahead, for a real implementation, the BF layout will be constrained by the CAM layout and, therefore, it behooves the designer to re-use components whereever feasible.

V. pbCAM Performance Evaluation

A. Experimental Setup

In order to demonstrate the energy reduction potential of the pbCAM architecture, we designed a 64-entry x 80-bit CAM schematic and simulated in SPICE using PTM (Predictive Technology Model) files [10]. Cadence tools are utilized; specifically, Virtuoso for the schematic realization and AMS and Spectre for the simulations. We targeted four technology nodes: 65 nm, 45 nm, 32 nm, and 22 nm. Although extracted-view-based simulations and a larger CAM size would be ideal and the most accurate, they are not conducive for the efficient vetting of the proposed pbCAM concept. However, RC-based interconnect models are included in our CAM design and the RC interconnect resistance and capacitance values specified by ITRS [2] are used in our simulations.

B. Experimental Results

The primary objective of the simulations is to extract E_{CB} (CAM bank energy per cycle) and E_{BF} (i.e., BF energy per cycle). At each technology node, we investigated three different power supplies: a high-performance value (HV), a nominal value (NV), and an 80% of nominal value (LV). The clock period is consistent throughout the simulations and is defined by the limiting case or from the LV power supply at the 65 nm technology node. E_{CB} is the sum of the precharge energy and the evaluation energy. These values can be further parsed into the bit-line and sense-amplifier (SA) energies. For the scope of this paper, only the resulting total energy is reported. From the discussion in Section IV-D, the worst-case CAM energy is consumed when the activity factor is 100%, i.e., when every bank is queried every clock cycle. For this instance, the HV supply value is requisite. On the other hand, if the queries are being properly filtered via the BF/BM circuity, the LV power supply may be utilized. Table I details a juxtaposition of a typical CAM energy consumption versus that of the proposed Bloom Filter-driven, pbCAM energy consumption. The 64-entry x 80-bit CAM is evaluated at each technology node and at each aforementioned supply voltage. The BF circuit shown in Figure 6 is also simulated in the same fashion. The results of this investigation are posted in the first four rows of the table. As one would expect, the energy consumption scales with both technology and power supply voltage. The BF, in many cases, is over two orders of magnitude more energy efficient than the CAM bank. Using the example given in Section IV-B, where B=16 and p=0.05, the rest of the table is populated. E_{CAM} is $B \times E_{CB}(@HV)$ and is considered the worst-case energy consumption of a standard CAM architecture. However, the worst-case implementation of the pbCAM is $B \times E_{BF} + 1.75 \times E_{CB}(@HV)$. The inclusion of the BF reduces the worst-case CAM activity factor from $\frac{16}{16}$ to $\approx \frac{1.75}{16}$, thereby translating to significant energy gains. A more realistic case is that given by E_{pbCAM} column. For this deployment and given an expected false positive rate of 5%, from the discussion in Section III, one would expect CAM banks to operate at the NV 10% of the time; at the HV 5% of the time; and at the LV 60% of the time, resulting in an aggregate energy consumption reported in the *nom* row on Table I. The best-case results (i.e. the last row of Table I) are only marginally better than for the expected nominal case. In both the *nom* and *BC* cases in the last two rows of Table I, most of the pbCAM activity is performed at the LV supply, which dictates this energy characteristic. This analysis shows that by filtering the data queries appropriately, an order-of-magnitude energy savings may be realized at each technology node.

TABLE I
COMPARISON OF CAM VS. PBCAM ENERGY AT DIFFERENT TECHNOLOGY NODES AND DIFFERENT SUPPLY VOLTAGES (HV, NV, LV).

Technology Node		**65 nm** (fJ)	**45 nm** (fJ)	**32 nm** (fJ)	**22 nm** (fJ)
E_{CB}	@ LV	19,996	13,675	4,557	2,389
	@ NV	37,274	31,755	11,756	6,170
	@ HV	62,048	49,543	22,481	12,338
E_{BF}		347	182	162	104
E_{CAM}	@ LV	319,936	218,800	72,912	22,224
	@ NV	596,384	508,080	188,096	98,720
	@ HV	992,768	792,688	359,696	197,400
E_{pbCAM}	WC	114,138	89,610	41,938	23,249
	nom	44,378	30,443	12,187	6,714
	BC	40,548	26,841	10,571	5,838

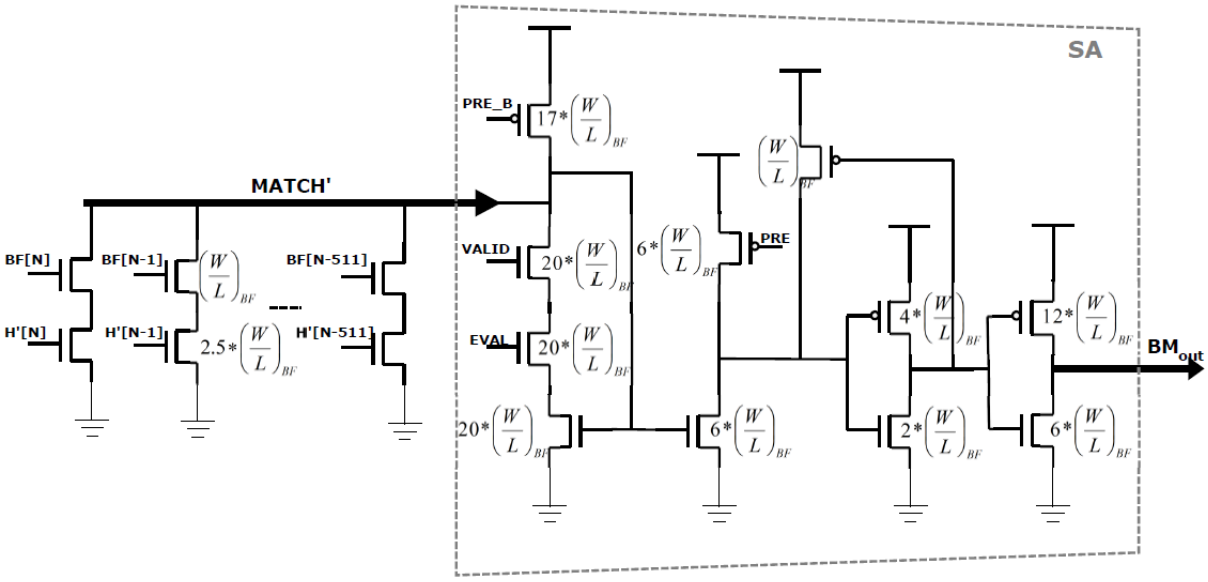

Fig. 6. Design of the Bloom Match circuit. Negative impact of the source degeneration is alleviated by placing the bloom filter on the top transistors.

VI. RELATED WORK

Bloom filters have been broadly studied in network routers, where IP lookups are sped up using multiple hash functions [11]. A thorough survey of Bloom filters is provided in [9]. A survey of Content Addressable Memories is provided in [1]. While CAMs have been widely studied in network routers [3] based on CMOS technology, recent work also focuses on TCAMs on non-volatile devices [4]. Other applications include synchronous performance improvement in Cloud Computing [12]–[16].

VII. CONCLUSIONS AND FUTURE WORK

We demonstrated via both theoretical analysis and simulations a new Content Addressable Memory (CAM) design with unique energy savings advantages. Our design, pbCAM, eliminates bank conflicts by decoupling the data elements from their corresponding bank index, thereby enabling new energy savings opportunities: 1) Each bank of the pbCAM can be operated at a lower frequency than the actual pbCAM itself, thereby achieving super-linear energy savings, 2) Larger CAMs can be constructed due to the reduced power consumption, 3) faster CAM throughputs can be realized by operating the banks slower, which is typically the limiting factor in determining the overall CAM speed. These improvements are achieved via a Bloom-filter-based probabilistic pre-elimination of the search entries. A Bloom filter used in each pbCAM bank requires two orders-of-magnitude less energy than the pbCAM bank itself and elimminates all but a small percentage of the unnecessary activity. As demonstrated by our simulations, this advantage is used to lower the frequency and the voltage of pbCAM banks, thereby yielding significant energy savings.

REFERENCES

[1] Kostas Pagiamtzis and Ali Sheikholeslami, "Content-Addressable Memory (CAM) Circuits and Architectures : A Tutorial and Survey," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, Mar. 2006.

[2] ITRS, "International Technology Roadmap for Semiconductors: 2011," 2011, http://www.itrs.net/models.html/.

[3] Banit Agrawal and Timothy Sherwood, "Ternary cam power and delay model: Extensions and uses," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 5, pp. 554–564, May 2008.

[4] Mourad El Baraji, Virgile Javerliac, and Guillaume Prenat, "Towards an ultra-low power, high density and non-volatile ternary cam," in *Non-Volatile Memory Technology Symposium (NVMTS)*, Pacific Grove, CA, Nov. 2008, pp. 1–7.

[5] Banit Agrawal and Timothy Sherwood, "Modeling tcam power for next generation network devices," in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, March 2006, pp. 1–10.

[6] B Bloom, "Space/time tradeoffs in hash coding with allowable errors," in *Communications of the ACM*, 1970, pp. 422–426.

[7] Adam Kirsch and Michael Mitzenmacher, "Building a Better Bloom Filter," Technical report, Department of Computer Science, 2005.

[8] Flavio Bonomi, Michael Mitzenmacher, Rina Panigrahy, Sushil Singh, and George Varghese, "An improved construction for counting bloom filters," in *14th Annual European Symposium, ESA*, 2006, pp. 684–695.

[9] Andrei Broder and Michael Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[10] Wei Zhao and Yu Cao, "Predictive technology model for nano-cmos design exploration," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 3, no. 1, April 2007.

[11] Andrei Broder and Michael Mitzenmacher, "Using multiple hash functions to improve ip lookups," in *IEEE Infocom*, Anchorage, Alaska, April 2001, pp. 1454–1463.

[12] Tolga Soyata and Eby G. Friedman, "Synchronous performance and reliability improvements in pipelined asics," in *Proceedings of the IEEE ASIC Conference*, Sep 1994, pp. 383–390.

[13] Tolga Soyata, Eby G. Friedman, and J. H. Mulligan, "Monotonicity constraints on path delays for efficient retiming with localized clock skew and variable register delay," in *Proceedings of the International Symposium on Circuits and Systems*, May 1995, pp. 1748–1751.

[14] Tolga Soyata, R. Muraleedharan, S. Ames, J. H. Langdon, C. Funai, M. Kwon, and W. B. Heinzelman, "Combat: mobile cloud-based compute/communications infrastructure for battlefield applications," in *SPIE Defense, Security, and Sensing 2009. Modeling and Simulation for Defense Systems and Applications VII*, April 2012, vol. 8403-20.

[15] Tolga Soyata, Rajani Muraleedharan, Colin Funai, Minseok Kwon, and Wendi Heinzelman, "Cloud-Vision: Real-Time face recognition using a Mobile-Cloudlet-Cloud acceleration architecture," in *17th IEEE Symposium on Computers and Communications (IEEE ISCC 2012)*, Cappadocia, Turkey, July 2012.

[16] Tolga Soyata, Rajani Muraleedharan, Colin Funai, Minseok Kwon, and Wendi Heinzelman, "SOLARCAP: super capacitor buffering of solar energy for self-sustainable field systems," in *25th IEEE International System-on-Chip Conference*, Niagara Falls, NY, Sept. 2012.