L-CBF: A Low-Power, Fast Counting Bloom Filter Architecture

Elham Safi, *Student Member, IEEE*, Andreas Moshovos, *Senior Member, IEEE*, and Andreas Veneris, *Senior Member, IEEE*

***Abstract*—An increasing number of architectural techniques have relied on hardware counting bloom filters (CBFs) to improve upon the energy, delay, and complexity of various processor structures. CBFs improve the energy and speed of membership tests by maintaining an imprecise and compact representation of a large set to be searched. This paper studies the energy, delay, and area characteristics of two implementations for CBFs using full custom layouts in a commercial 0.13-μm fabrication technology. One implementation, S-CBF, uses an SRAM array of counts and a shared up/down counter. Our proposed implementation, L-CBF, utilizes an array of up/down linear feedback shift registers and local zero detectors. Circuit simulations show that for a 1 K-entry CBF with a 15-bit count per entry, L-CBF compared to S-CBF is 3.7× or 1.6× faster and requires 2.3× or 1.4× less energy depending on the operation. Additionally, this paper presents analytical energy and delay models for L-CBF. These models can estimate energy and delay of various CBF organizations during architectural level explorations when a physical level implementation is not available. Our results demonstrate that for a variety of L-CBF organizations, the estimations by analytical models are within 5% and 10% of Spectre simulation results for delay and energy, respectively.**

***Index Terms*—Computer architecture, counting bloom filters, implementation, low power, microprocessors.**

I. Introduction

An increasing number of architectural techniques have relied on hardware *counting bloom filters* (CBFs) to improve upon the power, delay, and complexity of various processor structures. For example, CBFs have been used to improve performance and power in snoop-coherent multiprocessor or multi-core systems [1], [2]. CBFs have been also utilized to improve the scalability of load/store scheduling queues [3] and to reduce instruction replays by assisting in early miss determination at the L1 data cache [4]. In these applications, CBFs help eliminate broadcasts over the interconnection network in multiprocessor systems [1]; CBFs also help reduce accesses to much larger and thus much slower and power-hungry content addressable memories [3], or cache tag arrays [1], [2], [4].

In all aforementioned hardware applications, CBFs improve the energy and speed of membership tests. Checking whether a memory block is currently cached is an example of a membership test in processors [4]. The CBF provides a definite answer for most, but not necessarily for all, membership tests. As such, the CBF does not replace entirely the underlying conventional mechanism (e.g., cache tags), but it dynamically bypasses the conventional mechanism, which can be slow and power hungry, as frequently as possible. Accordingly, the benefits obtained through the use of CBFs depend on two factors. The first factor is how frequently a CBF can be utilized. Architectural techniques and application behavior determine how many membership tests can be serviced by the CBF. The second factor is the energy and delay characteristics of the CBF. The more membership tests are serviced by the CBF "alone" and the more speed and energy efficient the CBF implementation is, the higher the benefits.

This work focuses exclusively on the second factor as it investigates implementations of a CBF that improve its energy and delay characteristics. A key contribution of this work is the introduction of L-CBF. L-CBF is an energy- and delay-efficient implementation that utilizes an array of up/down linear feedback shift registers (LFSRs) and local zero detectors. Previous work assumes a straightforward SRAM-based implementation that we will refer to it as S-CBF [2]. We investigate the energy, delay, and area characteristics of L-CBF and S-CBF implementations in a commercial 0.13-μm CMOS technology. We demonstrate that depending on the type of operation, L-CBF compared to S-CBF is 3.7× or 1.6× faster and requires 2.3× or 1.4× less energy. This work also presents analytical energy and delay models for L-CBF. These analytical models can estimate energy and delay of various CBF organizations early in the design stage during architectural level explorations. These explorations are performed well before the physical level implementation phase in a design flow. Comparisons show that the estimations by the models are within 5% and 10% of Spectre circuit simulation results for delay and energy, respectively.

The significant contributions of this work are as follows. 1) It proposes L-CBF, a novel, energy- and speed-efficient implementation for CBFs. 2) It compares the energy, delay, and area of two CBF implementations, L-CBF and S-CBF, using their circuit-level implementations and full-custom layouts in 0.13-μm fabrication technology. 3) It presents analytical delay and energy models for L-CBF and compares the model estimations against simulation results.

The rest of this paper is organized as follows. Section II reviews CBFs and their previously assumed implementation, S-CBF. Section III-B presents L-CBF, our novel implementation. Section IV discusses the analytical delay and energy

Manuscript received February 1, 2007; revised June 18, 2007. This work was supported in part by an NSERC Discovery Grant, by a Canada Foundation for Innovation Equipment Grant, and by funds from the University of Toronto. This paper appeared in part in the Proceedings of the 2006 International Symposium on Low Power Electronics Design, Tegernsee, Bavaria, Germany, October 4–6, 2006, pp. 250–255.

The authors are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4 Canada (e-mail: elham@eecg.toronto.edu; moshovos@eecg.toronto.edu; veneris@eecg.toronto.edu).

Digital Object Identifier 10.1109/TVLSI.2008.2000244

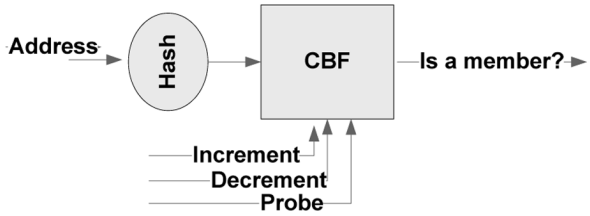

Fig. 1. CBF as a black box.

models of L-CBF. Section V presents the experimental results. Section VI summarizes our findings.

II. CBFs

This section reviews CBFs and their characteristics. Additionally, it discusses the previously assumed implementation for the CBFs, which has not been investigated at the physical level.

A. Introduction to CBFs

1) CBF as a Black Box: As shown in Fig. 1, a CBF is conceptually an array of counts indexed via a hash function of the element under membership test. A CBF has three operations: 1) increment count (INC); 2) decrement count (DEC); and 3) test if the count is zero (PROBE). The first two operations increment or decrement the corresponding count by one, and the third one checks if the count is zero and returns true or false (single-bit output). We will refer to the first two operations as *updates* and to the third one as a *probe*. A CBF is characterized by its number of entries and the width of the count per entry.

2) CBF Characteristics: Membership tests using CBFs are performed by probe operations. In response to a membership test, a CBF provides one of the following two answers: 1) "definite no," indicating that the element is definitely not a member of the large set and 2) "I don't know," implying that the CBF cannot assist in a membership test, and the large set must be searched.

The CBF is capable of producing the desired answer to a membership test much faster and saves power on two conditions. First, accessing the CBF is significantly faster and requires much less energy than accessing the large set. Second, most membership tests are serviced by the CBF. The later is investigated by studying the application behavior. For instance, when CBF is exploited as a miss predictor, previous work [4] shows that more than 95% of the accesses to the cache tag array are serviced by the CBF.

The CBF uses an imprecise representation of the large set to be searched. Ideally, in the CBF, a separate entry would exist for every element of the set. In this case, the CBF would be capable of precisely representing any set. However, this would require a prohibitively large array negating any benefits. In practice, the CBF is a small array and the element addresses are hashed onto this small array. Because of hashing, multiple addresses may map onto the same array entry. Hence, the CBF constitutes an imprecise representation of the content of the large set and keeps a superset of the existing elements. This impreciseness is the reason of the "I don't know" answers by the CBF. To reduce the frequency of such answers, and hence improving accuracy, multiple CBFs with different hash functions can be used.

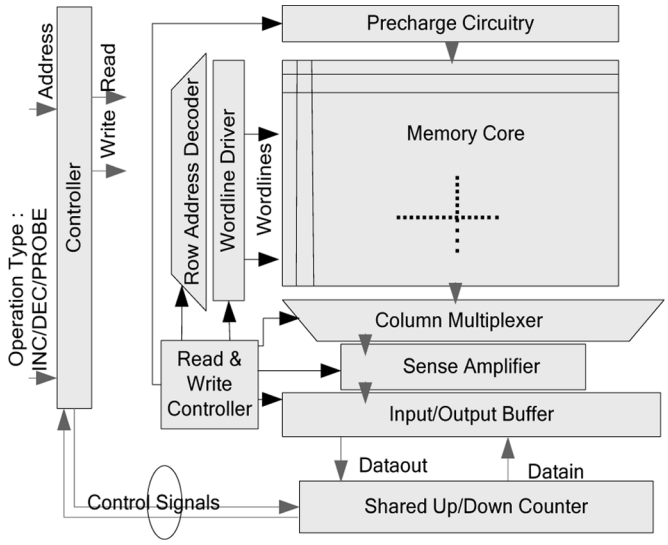

Fig. 2. S-CBF architecture: an SRAM holds the CBF counts; INC/DEC: read-modify-write sequences; PROBE: read-compare sequence.

An "I don't know" answer to a membership test incurs power and delay penalty since in case of such an answer, the large set must be checked in addition to the CBF. The delay penalty occurs if the CBF and the large set accesses are serialized. This delay penalty can be avoided if we probe the CBF and the large set in parallel; in this case, power benefits will be possible only if the in-progress access to the large set can be terminated once the CBF provides a definite answer. These overheads do not concern us as often CBF can provide the definite answer. To verify this, the interested reader could refer to [1]–[4] for examples of CBF applications in computer architecture.

3) CBF Functionality: The CBF operates as follows. Initially, all counts are set to zero and the large set is empty. When an element is inserted into, or deleted from the large set, the corresponding CBF count is incremented, or decremented by one. To test whether an element currently exists in the large set, the corresponding CBF count is inspected. If the count is zero, the element is definitely not in the large set; otherwise, CBF cannot assist and the large set must be searched.

B. S-CBF: SRAM-Based CBF Implementation

Previous work assumes a CBF implementation consisting of an SRAM array of counts, a shared up/down counter, a zero-comparator, and a small controller [2]. We will refer to this implementation as S-CBF. The architecture of S-CBF is depicted in Fig. 2. Updates are implemented as read-modify-write sequences as follows: 1) the count is read from the SRAM; 2) it is adjusted using the counter; and 3) it is written back to the SRAM. The probe operation is implemented as a read from the SRAM, and a compare with zero using the zero-comparator. A small controller coordinates this sequence of actions.

An optimization was proposed to speedup probe operations and to reduce their power [2]. Specifically, an extra bit Z is added to each count. When the count is nonzero the Z is set to false and when the count is zero, the Z is set to true. Probes can now simply inspect Z. The Z bits can be implemented as a separate SRAM structure which is faster and requires much

less power. This type of optimization is compatible with both S-CBF and L-CBF architectures.

III. L-CBF: A Novel LFSR-Based CBF Implementation

Section V demonstrates quantitatively that much of the energy in S-CBF is consumed on the SRAM's bitlines and wordlines. Additionally, in S-CBF, both delay and energy suffer as updates require two SRAM accesses per operation. The shared counter may increase the energy and the delay further. We could avoid accesses over long bitlines by building an array of up/down counters with local zero detectors. In this way, CBF operations would be localized and there would be no need to read/write values over long bitlines. L-CBF is such a design. For the CBF, the actual count values are not important and we only care whether a count is "zero" or "nonzero." Hence, any counter that provides a deterministic up/down sequence can be a choice of counter for the CBF. L-CBF consists of an array of up/down LFSRs with embedded zero detectors. L-CBF employs up/down LFSRs that offer a better delay, power, and complexity tradeoff than other synchronous up/down counters with the same count sequence length (see Section III-A2).

As Section V demonstrates, L-CBF significantly reduces energy and delay compared to S-CBF at the cost of more area. The increase in area though is a minor concern in modern processor designs given the abundance of on-chip resources and the very small area of the CBF compared to most other processor structures (e.g., caches and branch predictors).

The rest of this section reviews up/down (reversible) LFSRs and discusses the architecture of L-CBF.

A. LFSRs

A *maximum-length* n-bit LFSR sequences through $2^n - 1$ states. It goes through all possible code permutations except one. The LFSR consists of a shift register and a few embedded XNOR gates fed by a feedback loop. Each LFSR has the following defining parameters:

- *width*, or *size*, of the LFSR (it is equal to the number of bits in the shift register);
- number and positions of *taps* (taps are special locations in the LFSR that have a connection with the feedback loop);
- initial state of the LFSR which can be any value except one (all ones for XNOR feedback).

Without the loss of generality, we restrict our attention to the Galois implementation of LFSRs [6]. State transitions proceed as follows. The non-tapped bits are shifted from the previous position. The tapped bits are XNORed with the feedback loop before being shifted to the next position. The combination of the taps and their locations can be represented by a polynomial (see Section I). Fig. 3 shows an 8-bit maximum-length Galois LFSR, its taps, and polynomial.

By appropriately selecting the tap locations it is always possible to build a maximum-length LFSR of any width with either two or four taps [1], [6]. Additionally, ignoring wire length delays and the fan-out of the feedback path, the delays of the maximum-length LFSR is independent of its width (size) [5], [6]. As Section V-B shows, delay increases only slightly with size, primarily due to increased capacitance on the control lines.

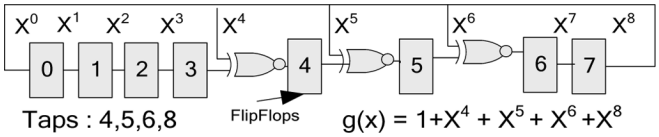

Fig. 3. Eight-bit maximum-length LFSR.

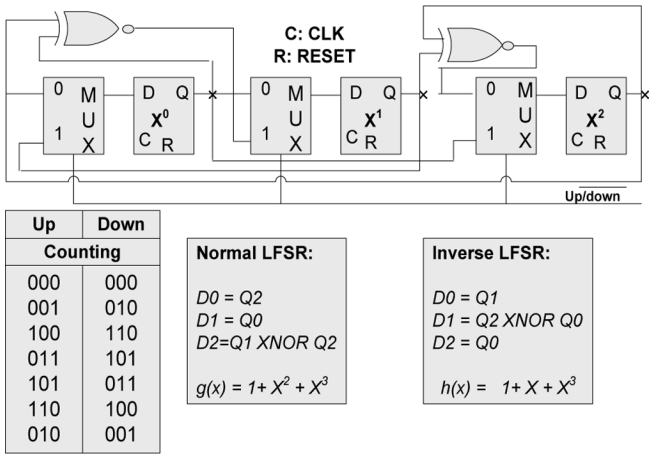

Fig. 4. Three-bit maximum-length up/down LFSR.

1) Up/Down LFSRs: The tap locations for a maximum-length, unidirectional n-bit LFSR can be represented by a primitive polynomial $g(x)$ as depicted in (1)

$$g(x) = \sum_{i=0}^{n} C_i \times X^i \quad (C_0 = C_n = 1). \tag{1}$$

In (1), X^i corresponds to the output of the i^{th} bit of the shift register and the constants C_i are either 0 (no tap) or 1 (tap). Given $g(x)$, a primitive polynomial $h(x)$ for an LFSR generates the reverse sequence as depicted in (2) [7]

$$h(x) = \sum_{i=0}^{n} C_i \times X^{n-i} \quad (C_0 = C_n = 1). \tag{2}$$

The superposition of the two LFSRs (the original and its reverse) forms a reversible "up/down" LFSR. The up/down LFSR consists of a shift register similar to the one used for the unidirectional LFSR; a 2-to-1 multiplexer per bit to control the shift direction; and twice as many XNOR gates as the unidirectional LFSR. Fig. 4 shows the construction of a 3-bit maximum-length up/down LFSR. It also depicts the polynomials and count sequence of both up and down directions. In general, it is possible to construct a maximum-length up/down LFSR of any width with two or six XNOR gates (i.e., four or eight taps) [6]. Reference [6] reports tap positions for n up to 168.

2) Comparison With Other Up/Down Counters: In this section, we compare LFSR counters with other synchronous up/down counters that could be a choice of counter for CBFs. We restrict our discussion to synchronous up/down counters of width n with a count sequence of at least $2^n - 1$ states.

The simplest type of synchronous counter is the binary modulo-2^n n-bit counter. For this counter, speed and area are conflicting qualities due to carry propagation. For example, the n-bit ripple-carry synchronous counter, one of the simplest

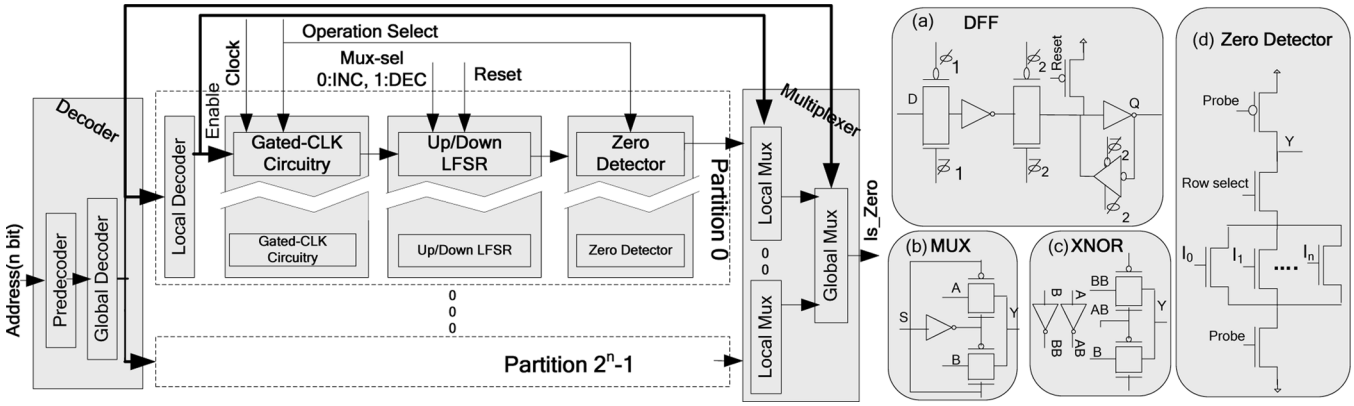

Fig. 5. Architecture of L-CBF; the basic cells of an up/down LFSR: (a) the two-phase flip-flop; (b) the 2-to-1 multiplexer; (c) XNOR gate; and (d) a bit-slice of the embedded zero detector.

counters, has a delay of $O(n)$ [5]. Counters with a Manchester carry-chain, carry-lookahead and binary tree carry propagation [8] have delay of $O(\log n)$ though at the cost of more energy and area. In applications where the count sequence is unimportant [e.g., pointers of circular first-inputs–first-outputs (FIFOs) and frequency dividers], an LFSR counter offers a speed-power-area efficient solution. The delay of an LFSR is nearly independent of its size. Specifically, the LFSR delay consists of a flip-flop delay, an XNOR gate delay, and a feedback loop delay. The feedback loop delay is the propagation delay of the last flip-flop output to the input of the furthest XNOR gate from the last flip-flop. Ignoring secondary effects on the feedback path, the delay of an n-bit maximum length LFSR is O(1) and independent of the counter size [5], [6]. These characteristics make LFSRs a suitable counter choice for CBFs.

B. L-CBF Implementation

Fig. 5 depicts the high-level organization of L-CBF. L-CBF includes a hierarchical decoder and a hierarchical output multiplexer. The core of the design is an array of up/down LFSRs and zero detectors. The L-CBF design is divided into several partitions where each row of a partition consists of an up/down LFSR and a zero detector.

L-CBF accepts three inputs and produces a single-bit output *is-zero*. The input *operation select* specifies the type of operation: INC, DEC, PROBE, and IDLE. The input *address* specifies the address in question and the input *reset* is used to initialize all LFSRs to the *zero* state. The LFSRs utilize two non-overlapping phase clocks generated internally from an external clock.

We use a hierarchical decoder for decoding the address to minimize the energy-delay product [9]. The decoder consists of a predecoding stage, a global decoder to select the appropriate partition, and a set of local decoders, one per partition. Each partition has a shared local *is-zero* output. A hierarchical multiplexer collects the local *is-zero* signals and provides the single-bit *is-zero* output.

Fig. 5 also depicts the basic cells of each up/down LFSR and zero decoder. Shown are the flip-flop used in the shift registers, the multiplexer that controls the direction of change ("up"/"down"), the XNOR gate, and a bit-slice of the zero decoder. Further details on L-CBF implementation are presented in Section IV.

1) Multi-Porting: Some applications require simultaneous operations from the CBF. In the simplest implementation, the CBF can be banked to support simultaneous accesses to different banks. This mirrors the organization of high-performance caches that are often banked to support multiple accesses instead of being truly multi-ported. True multi-porting is straightforward by selective resource replication in case of simultaneous accesses to different counts. For S-CBF, we need an SRAM with multiple read and write ports and multiple shared up/down counters. For L-CBF, we need to replicate the decoder, the zero detectors, and the output multiplexer.

When multiple accesses map to the same count, multi-porting is not straightforward. A simple solution detects such accesses and serializes them. Alternatively, circuitry can be added to determine the collective effect of all accesses. For example, for two simultaneous increment operations, the net effect is to increase the counter by two. For S-CBF, this circuitry can be embedded into the shared counter. For L-CBF, the capability of shifting by multiple cells in one cycle is required. This work does not consider these enhancements.

IV. Analytical Models

Analytical models help computer architects to estimate the energy and delay of various architectural alternatives under exploration. To the best of our knowledge, there are no such analytical models for CBFs. This section presents analytical models of the worst case delay and energy (dynamic and leakage) for the L-CBF implementation. These analytical models can be incorporated in architecture level power-performance simulators such as Wattch [20].

The models predict L-CBF's delay and energy as a function of entry count, entry width and the number of banks. The models were extrapolated starting from our L-CBF's full custom implementation in a 0.13-μm CMOS process (detailed in Section V). The utility of the analytical models is in estimating the energy and delay of L-CBF organizations without having a physical level implementation.

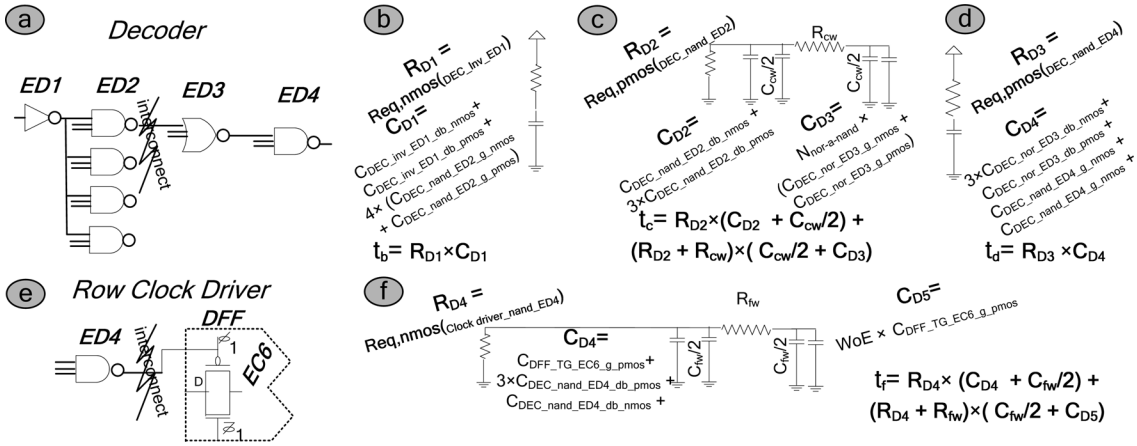

Fig. 6. *RC* circuit analysis along the critical path of L-CBF. (Decoder and row clock driver.)

TABLE I
ANALYTICAL MODEL INPUT PARAMETERS

Externally Visible Organizational Parameters	
NoE	Number of entries
WoE	Width of entry (count width)
Internal Organizational Parameters	
NoRP	Number of rows per partiton
Technology Parameters	
$\mathbf{C_w}$, $\mathbf{R_w}$	Per unit length capacitance and sheet resistance of metal layers
Other parameters as in [19], such as C_{gate}, $C_{ndiffarea}$, $C_{ndiffside}$, $C_{ndiffgate}$, $C_{pdiffarea}$, $C_{pdiffside}$, $C_{pdiffgate}$, $R_{eq.nmos}$, $R_{eq,pmos}$, V_{dd}.	

In our implementation and models, the gates are sized to have equal rise and fall delays. The models do not account for the external loads as they are independent of the CBF implementation. While it is feasible to extend the models to predict delay and energy for other technologies, this extension is not a focus of this work.

The rest of this section is organized as follows. Section IV-A discusses the methodology used for developing analytical models and the input parameters of the models. Section IV-B and IV-C present the delay and energy models, respectively. Discussing the accuracy of the models is postponed until Section V-C, where we compare the model estimations with simulation results.

A. Methodology

To model delay and energy per operation, we decompose L-CBF into several equivalent resistance–capacitance (*RC*) circuits. We use the methodology of CACTI [19] to estimate equivalent "on" resistance and capacitance. References [14] and [15] detail how C_{gate} and $C_{\text{diffusion}}$, C_{ovelap}, $R_{\text{eq-nmos}}$, and $R_{\text{eq-pmos}}$ are estimated. Information such as transistor sizes and the length of interconnects, required for capacitance and resistance estimations, is extracted from our layout. Transistors are scaled to minimize the energy-delay product for larger CBFs.

Table I lists the input parameters of the analytical model that fall under three broad classes: externally visible organizational parameters, internal organizational parameters, and technology specific parameters. The externally visible L-CBF organization is defined by the total number of entries (NoE) and the width of each entry count (WoE). Internally, L-CBF can be partitioned into banks of NoRP rows to balance or improve power and delay.

B. Delay Model

This section presents an analytical model for the worst case delay of L-CBF. Figs. 6–8 depict the *RC* circuit analysis for the delay along the critical path. For clarity, we assign a label to each element in the path and use it as a subscript to identify the corresponding resistance and capacitance. The type of gates (e.g., inverter) and capacitors (e.g., drain: d; source: s; gate: g) are also denoted in the subscripts. We model the delay of CBF operations separately. The delay of an update operation consists of the decoder delay, the row clock driver delay, and the up/down LFSR delay. The delay of a probe operation is comprised of the decoder delay, the zero detector delay, and the output multiplexer delay. The following sections discuss the delay analysis for each component (e.g., decoder) focusing on resistance and capacitance estimation. Then, we present the analytical delay models of CBF operations.

1) Component Delay—Decoder: Fig. 6(a)–(f) show the simplified critical path of the decoder and the equivalent *RC* circuit. To estimate the *RC* delay, we determine the number and size of transistors and interconnects that appear along the critical path. These are a function of NoE and NoRP. The decoder utilizes a hierarchical architecture. In the predecode stage, each 3-to-8 decoder generates a 1-of-8 code for every three address bits. If the number of address bits is not divisible by three, a 2-to-4 decoder or an inverter is used. Each x-to-2^x decoder is implemented using 2^x NAND gates and x inverters to complement the address inputs. In the second stage, the predecode stage outputs are combined using NOR gates. When beneficial, an inverter chain is used at the predecode stage's output to reduce delay.

The decoder delay is the interval between the moment the address input passes the INV(ED1)' threshold voltage and the

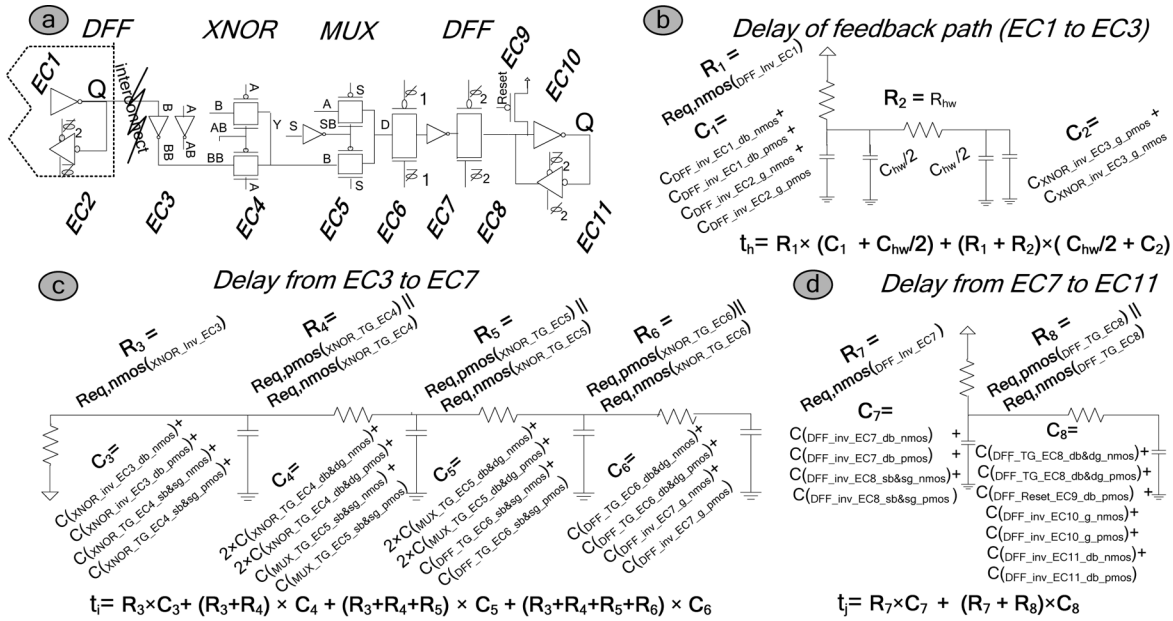

Fig. 7. *RC* circuit analysis along the critical path of L-CBF. (Up/down LFSR.)

moment the NOR (ED3)' output reaches the threshold voltage of the following NAND (ED4). Equations (3)–(11) calculate subsequently the number of address bits (N_{addr}), the number of 3-to-8 decoders $(N_{3\text{ to }8})$, the number NOR gates (N_{nor}), the fan-in of a NOR gate $(N_{\text{nor-input}})$ as a function of NoE. The formulas *Extra-2to4* and *Extra-inv* calculate whether an additional 2-to-4 decoder or an inverter is required when the number of address bits is not divisible by three. The formula $N_{nor-a-nand}$ calculates the number of NOR gates that are fed by a NAND gate. The wire length between the NOR gates and the corresponding resistance and capacitance are calculated by (10) and (11).

2) Component Delay—Row Clock Driver: Fig. 6(e) and (f) show the simplified critical path of the row clock driver and its equivalent *RC* circuit, respectively. The NAND gate (ED4) performs clock gating. Its inputs are the global clock, decoder output and operation select. If a row is selected and the operation is an INC or DEC, the clock signal is applied to the addressed up/down LFSR. The worst case delay occurs when the clock signal is delivered to the last DFF. The wire length between the row clock driver and the last DFF (L_{fw}) is proportional to the LFSR width. This is also true for the length of the LFSR feedback path (L_{hw}). Both L_{fw} and L_{hw} are calculated by (12). This wire length is used for estimating equivalent resistance and capacitance.

3) Component Delay—Up/Down LFSR: The delay of an up/down LFSR is comprised of a DFF delay, a 2-to-1 multiplexer delay, an XNOR gate delay, and a feedback path delay. Fig. 7(a)–(d) show the equivalent *RC* circuit for the up/down LFSR. The feedback path delay is the propagation delay of the last DFF's output to the furthest XNOR gate from it. As addressed in Section III-A, a maximum-length n-bit up/down LFSR requires at most six XNORs [6]. The length of feedback path for a maximum-length *WoE*-bit up/down LFSR is given by (12)

$$N_{\text{addr}} = \lceil \log_2(\text{NoE}) \rceil \tag{3}$$

$$N_{3\text{ to }8} = \left\lfloor \frac{1}{3} \times (N_{\text{addr}}) \right\rfloor \tag{4}$$

$$\text{Extra-2 to 4} = \left\lfloor \frac{1}{2} \times [(N_{\text{addr}}) - 3 \times (N_{3\text{ to }8})] \right\rfloor \tag{5}$$

$$\begin{aligned}\text{Extra-inv} = &((N_{\text{addr}}) - 3 \times N_{3\text{ to }8}) - 2 \\ &\times (\text{Extra-2-to-4-predecoder}))\end{aligned} \tag{6}$$

$$N_{\text{nor}} = \text{NoE} \tag{7}$$

$$\begin{aligned}N_{\text{nor-inputs}} = &(N_{3\text{ to }8} + \text{Extra-2-to-4-predecoder} \\ &+ \text{Extra-inverter})\end{aligned} \tag{8}$$

$$N_{\text{nor-a-nand}} = \frac{\text{NoE}}{8} \text{ if } (N_{\text{addr}} \text{ is divisible by3}) \tag{9}$$

$$\begin{aligned}L_{cw}(\mu\text{m}) = &\text{wire length between two NOR gates fed} \\ &\text{by the same NAND } (ED2) \text{ gate in the} \\ &\text{predecode stage. (extracted from the layout)}\end{aligned} \tag{10}$$

$$\begin{aligned}R_{cw}(\text{Ohm}) &= R_{\text{Ohm}/\square} \times \left(\frac{L_{\text{wire}}}{W_{\text{wire}}}\right) \\ C_{cw}(\text{Farad}) &= C\left(\frac{\text{Farad}}{\mu\text{m}}\right) \times L_{\text{wire}}(\mu\text{m})\end{aligned} \tag{11}$$

$$\begin{aligned}L_{\text{hw}}(\mu\text{m}) = &(\text{width of DFF} + \text{width of Mux}) \\ &\times (\text{WoE} - 6) \\ &+ (\text{width of DFF} + \text{width of XNOR} \\ &+ \text{width of MUX}) \times 6.\end{aligned} \tag{12}$$

4) Operation Delay: Increment and Decrement: The delay of the update operation consists of the decoder delay, the clock driver delay, and the up/down LFSR delay. All the gates are sized to have the same rise and fall delay. The delay of update operation is calculated by (13), where τ_b through τ_j are time constants that are given in Figs. 6 and 7, respectively

$$\text{Delay}_{\text{Update}} = 0.69 \times (\tau_b + \tau_c + \tau_d + \tau_f + \tau_h + \tau_i + \tau_j). \tag{13}$$

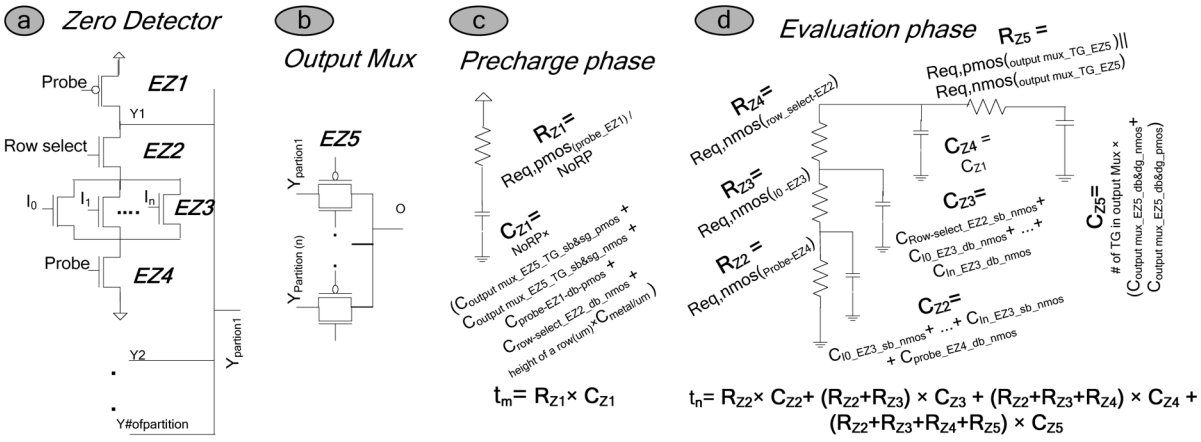

Fig. 8. *RC* circuit analysis along the critical path of L-CBF. (Zero detector and output multiplexer.)

5) Component Delay—Zero Detector and Output Multiplexer: The zero detectors of every set of *NoRP* rows in a partition have a shared output. This output is steered to the single-bit output, *is-zero*, through the output multiplexer.

A probe proceeds in three stages: 1) decode and precharge; 2) evaluation; and 3) transfer to the output. The decoding stage is the same for update and probe operations. The precharging stage is concurrent with the decoding stage. In the precharge stage, the shared output of a partition is charged to the supply voltage $V_{\rm dd}$. During the evaluation stage, based on the current value of the associated up/down LFSR, the partition output is discharged to zero or stays at $V_{\rm dd}$. The output of the selected partition is transferred to the *is-zero* output by the output multiplexer.

6) Operation Delay—Probe: Fig. 8(a)–(d) depict the equivalent *RC* circuits for the zero detector and the output multiplexer. The delay of the probe operation consists of the decoder delay, the zero detector delay, and the output multiplexer delay. The delay is calculated by (14), where τ_b to τ_n are time constants that are presented in Figs. 6 and 8

$$\text{Delay}_{\text{Probe}} = 0.69 \times (\tau_b + \tau_c + \tau_d + \tau_m + \tau_n). \quad (14)$$

C. Energy Model

There are four sources of the power dissipation in L-CBF. First is the dynamic switching power due to the charging and discharging circuit capacitances. Second is the leakage power from reverse-biased diodes and subthreshold conduction. Third is the short-circuit power because of finite signal rise/fall times. Fourth is the static biasing power found in some types of logic styles (i.e., pseudo-nMOS). For the given technology, circuit simulations suggest that the first two are the principal sources of energy consumption.

1) Dynamic Power: Dynamic power is the result of the gates' output transitions. Output transitions cause a capacitive load driven by the gate to be charged or discharged. To estimate the energy per operation, we add up the gate (e.g., NAND) and interconnect capacitances in the signal path for each component. The energy dissipated per transition (*0-to-1* or *1-to-0*) is given by (15), where C_L is the load capacitance, $V_{\rm dd}$ is the supply voltage, and ΔV is the voltage swing of the output

$$E_{\text{dynamic}} = 0.5 \times C_L \times V_{\rm dd} \times \Delta V. \quad (15)$$

The analytical energy models use the capacitance estimations of the delay *RC* analysis section. For instance, the decoder energy is calculated by (16)

$$E_{\text{decoder}} = 0.5 \times V_{\rm dd}^2 \times (C_{D1} + C_{D2} + C_{cw} + C_{D3}). \quad (16)$$

The same methodology is used for the remaining components.

D. Leakage Power

This section discusses the leakage power calculation methodology. To calculate the leakage current in a MOSFET, similar to [16], we use the model proposed by Zhang *et al.* [17] given by (17)

$$I_{lkg} = \mu_0 \cdot C_{\rm ox} \cdot \frac{w}{l} \cdot e^{b(V_{\rm dd} - V_{\rm dd0})} \cdot v_t^2 \cdot (1 - e^{-V_{\rm dd}/v_t}) \cdot e^{(-v_{\rm th} - v_{\rm off}/n v_t)}. \quad (17)$$

As shown in [16], for a given threshold voltage $(V_{\rm th})$ and temperature (T), all terms except the width (W) are constant for all the transistors in a given fabrication technology. Hence, (17) can be reduced to (18), where I_l is the leakage current of a unit width transistor at a given T and $V_{\rm th}$

$$I_{lkg} = W \times I_l(T, V_{\rm th}). \quad (18)$$

As in [17], we identify the distribution of the inputs for each component (e.g., single transistors or gates) based on the operation characteristics of L-CBF. Then, we derive $I_l(T, V_{\rm th})$ for each component at different input states by simulation and we consider the worst case. Finally, we sum the $I_l(T, V_{\rm th})s$ for all components.

As an example, we discuss the methodology of leakage current calculation for the decoder. The same methodology is used for the other components. In L-CBF, by activating the enable signal during the update and probe, the 3-to-8 predecoder's outputs are triggered (stage one), and the output of one of the NOR gates will take the logic value of one (stage two). We modeled

the worst case leakage current in these two stages as given by (19) and (20), respectively. The leakage current for the decoder is given by (21). Multiplying the $I_{\rm dec}$ by $V_{\rm dd}$ gives the leakage power estimation

$$I_{\rm stage1} = N_{3\ {\rm to}\ 8} \times (3 \times I_{\rm ED1} + 8 \times I_{\rm ED2}) \quad (19)$$

$$I_{\rm stage2} = {\rm NoE} \times (I_{\rm ED3}) \quad (20)$$

$$I_{\rm dec} = I_{\rm stage1} + I_{\rm stage2}. \quad (21)$$

V. Experimental Results

This section compares the energy, delay, and area of S-CBF and L-CBF. Moreover, this section compares the analytical model estimations against simulation results for L-CBF.

We compare S-CBF and L-CBF on a per operation basis. Both designs are implemented using the Cadence(R) tool set in a commercial 0.13-μm fabrication technology. We developed a transistor-level implementation and a full-custom layout for both designs that were optimized for the energy-delay product. We employed Spectre for circuit simulations. This is a vendor recommended simulator for design validation prior to manufacturing.

The rest of this section is organized as follows. We initially consider a 1 K-entry CBF with 15-bit counts as this configuration is representative of the CBFs used in previous proposals [2], [4]. Then, we present results for other CBF configurations. In Section V-A, we compare the energy, delay and area of the two designs for all CBF operations (updates and probes). In Section V-B, we study how energy and delay change as the number of entries and the width of the counters vary. In Section V-C, we discuss the accuracy of analytical models.

A. Delay and Energy Per Operation

We compare implementations of a 1 K-entry, 15-bit count per entry CBF. For S-CBF, an SRAM with a total capacity of 15 Kbits is used. The SRAM is partitioned to minimize the energy-delay product. For S-CBF, we do not consider the delay and energy overhead of the shared counter since our goal is to demonstrate that L-CBF consumes less energy and is also faster. To further reduce energy for probes in S-CBF, we introduce an extra bit per entry which is updated only when the count changes from, or to, zero as described in Section II-B (Z-bits). On a probe, we only read this bit. Furthermore, we apply a number of delay and power optimizations on S-CBF [9]–[12]. In detail, we implement the divided word line (DWL) technique which adopts a two-stage hierarchical row decoder structure. The DWL technique improves speed and power [10], [12]. Moreover, we reduce power further via pulse operation techniques for the word-lines, the periphery circuits and the sense amplifiers [12]. We also use multistage static CMOS decoding [9] and current-mode read and write operations to further reduce power [12]. For L-CBF, we utilize 16-bit LFSRs such that the LFSR can count at least 2^{15} values.

Table II shows the delay in picoseconds, the energy (static and dynamic) per operation in picojoules, and the area in square millimeters for both L-CBF and S-CBF. The last column reports the ratio of S-CBF over L-CBF per metric. The two rows per category report, respectively, measurements for the update and probe operations. For delay and energy, we report the worst case which is measured by selecting appropriate inputs. The delay and energy of the shared counter of S-CBF is not included; otherwise, the actual delay and energy of S-CBF would be higher.

TABLE II
Energy, Delay, and Area of S-CBF and L-CBF Implementations for a 1 K-Entry, 15-bit CBF

	Operation	L-CBF	S-CBF	S-CBF/ L-CBF
Delay (ps)	INC/DEC	447.26	1670	3.7
	PROBE	580.32	910.12	1.6
Energy (pj)	INC/DEC	38.73	88.98	2.3
	PROBE	30.36	41.02	1.4
Area (mm^2)		0.95	0.30	0.31

As observed from Table II, L-CBF is 3.7 and 1.6× faster than S-CBF during update and probe operations, respectively. In addition, L-CBF consumes 2.3 or 1.4× less energy than S-CBF for update and probe operations, respectively. These significant gains in speed and energy consumption come at the expense of more area. L-CBF requires about 3.2× more area than S-CBF. However, as discussed in Section III, area is less of a concern in modern microprocessor designs.

Disregarding the overhead (delay and energy) of the shared counter, the measurements for S-CBF are optimistic. An up/down 15-bit LFSR counter has a delay of 240 ps and energy per update of 25 FJ. If this LFSR was used as the shared counter for S-CBF, L-CBF would be 4.3 or 1.98× faster than S-CBF for updates and probes, respectively (relative energy remains virtually the same).

1) Per Component Energy Breakdown: Fig. 9 shows a per component breakdown of energy consumption for S-CBF and L-CBF. Most of the energy (79% and 74%, respectively, for updates and probes) in S-CBF is consumed by the memory core (worldlines, bitlines, and SRAM cells). The decoder and the sense-amplifiers consume considerably less energy. This is expected as we applied aggressive energy and delay optimizations to these components. For L-CBF, during probes, about 50% of the total energy is dissipated in inactive components, the LFSR array and row drivers. For L-CBF, during updates, 50% of the total energy is dissipated in non-active LFSRs, row drivers, zero detectors, and output multiplexer.

2) Per Component Delay Breakdown: Fig. 10 shows a per component breakdown of delay for both S-CBF and L-CBF for updates and probes. In S-CBF, the update operation delay consists of the decoder delay, the SRAM read access delay (excluding the decoder delay) and the SRAM write access delay (excluding the decoder delay). In detail, the update operation delay consists of the decoder delay, the read-wordline delay, the read-bitline delay, the read-sense amplifier delay, the read-output multiplexer delay, the write-write driver delay, the write-wordline delay, the write-bitline delay, and the precharge delay. The precharge delay is included as the update operation involves a read-modify-write sequence. In S-CBF, significant part of the delay belongs to the memory core, demonstrating that significant potential exists for improvements with L-CBF.

For L-CBF, the delay of the update operation consists of the decoder delay, the row clock driver delay, and the up/down LFSR delay. For L-CBF, the probe operation delay is comprised

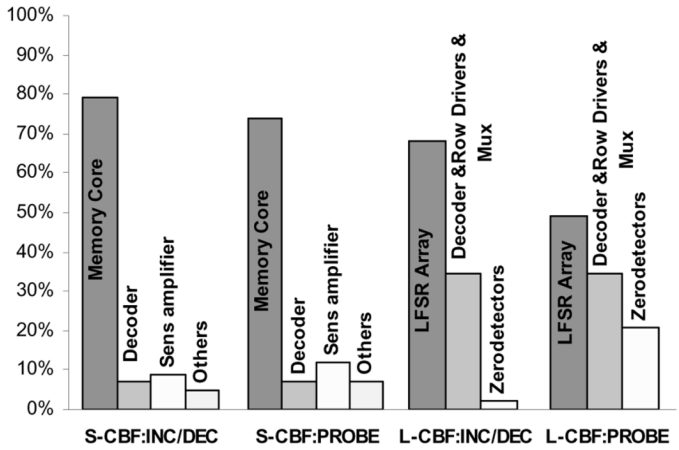

Fig. 9. Per component energy consumption for S-CBF and L-CBF. Breakdown for (INC/DEC) and probe (PROBE).

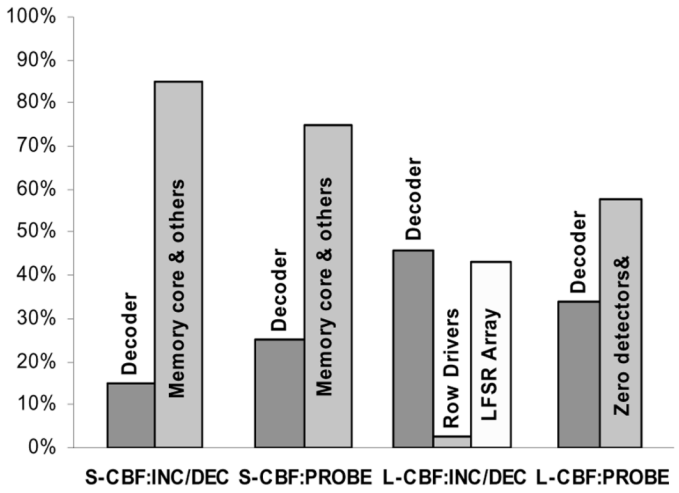

Fig. 10. Per component delay breakdown for S-CBF and L-CBF. Breakdown for (INC/DEC) and probe (PROBE).

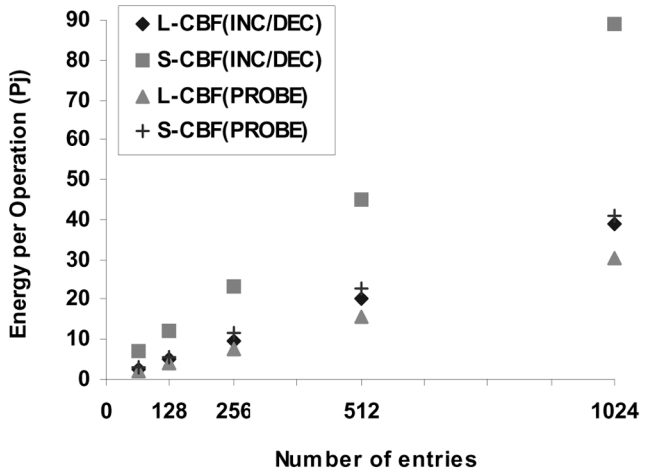

Fig. 11. Energy per operation as a function of the number of entries for L-CBF and S-CBF with 15-bit counts.

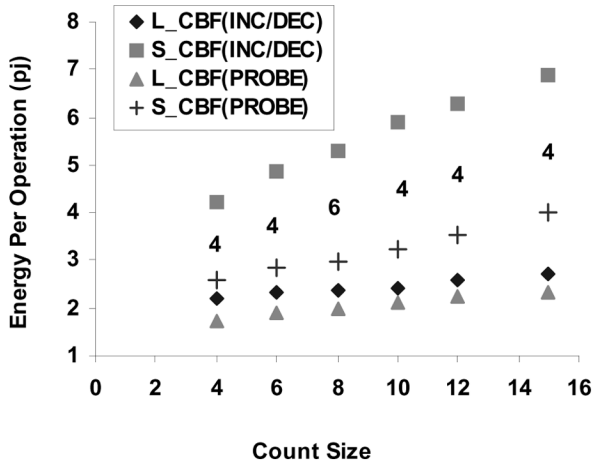

Fig. 12. Energy per operation as a function of count width for L-CBF and S-CBF for a 64-entry CBF.

of the decoder delay, the zero detector delay, and the output multiplexer delay. In L-CBF, the delay is balanced across the LFSR core and the decoder demonstrating that the L-CBF successfully reduces delay compared to S-CBF.

B. Sensitivity Analysis

This section investigates delay and energy variation as a function of the number of entries and count width for both L-CBF and S-CBF.

1) Energy Per Operation: Fig. 11 reports the energy per operation for CBFs as a function of entry count for 64 through 1-K entries in power of two steps. We observe that L-CBF consistently consumes less energy than S-CBF and the relative difference increases slightly for larger entry counts.

Fig. 12 reports the energy per operation as a function of count width in the range of four to 16 bits for a 64-entry CBF. Along L-CBF measurements, we also report the number of taps needed by each count width (either four or eight). We observe that the energy of L-CBF scales better than that of S-CBF. Communication in L-CBF is primarily between adjacent cells. For this reason, increasing the number of cells does not impact the overall energy significantly. The energy of S-CBF increases at a greater rate because additional bitlines and sense amplifiers are introduced and the wordlines become longer. Fig. 12 shows that changing the number of taps from four to eight in LFSRs does not significantly impact energy.

2) Delay: Fig. 13 reports the delay for CBFs of 64 through 1-K entries in power of two steps. As the number of entries increases, the size and the delay of the decoder increase and so does the size and delay of the output multiplexer. L-CBF is consistently faster than S-CBF. The difference in speed increases slightly with the number of entries.

Fig. 14 reports the delay as a function of LFSR width in the range of four to 16 bits for a 64-entry CBF. We observe a negligible increase in the update operation as the width increases. For larger LFSR widths there are three potential sources of increased delay: the row clock driver, the LFSR feedback loop and the embedded zero detector. Increasing the LFSR width elongates the clock driver wire for each row and consequently the clock driver's load. By resizing the row driver or by adding a buffer chain it is possible to avoid any significant increase in delay at the cost of more energy. As the counter width increases, so does

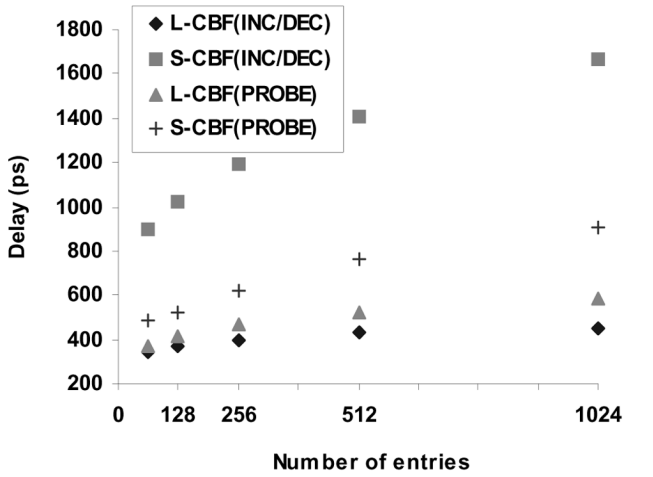

Fig. 13. Delay as a function of number of entries for L-CBF and S-CBF with15-bit counts.

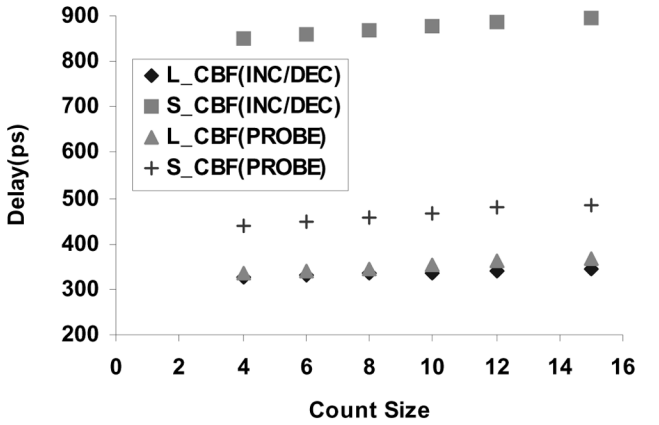

Fig. 14. Delay as a function of count width for L-CBF and S-CBF for a 64-entry CBF.

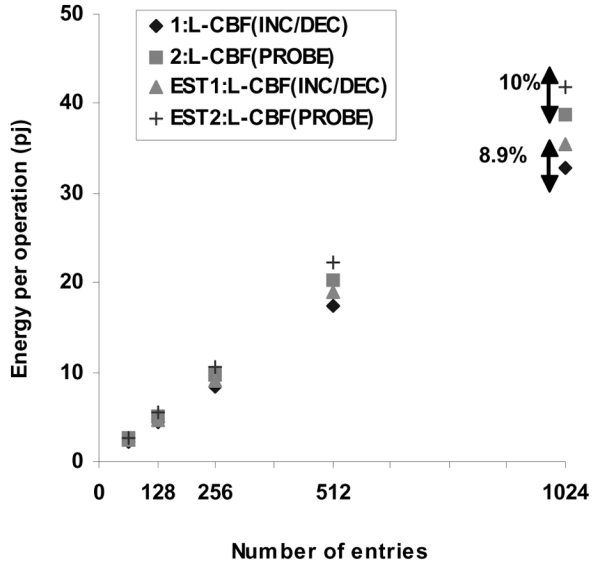

Fig. 15. Energy per operation as a function of number of entries for L-CBF with 15-bit counts: simulation results and model estimations.

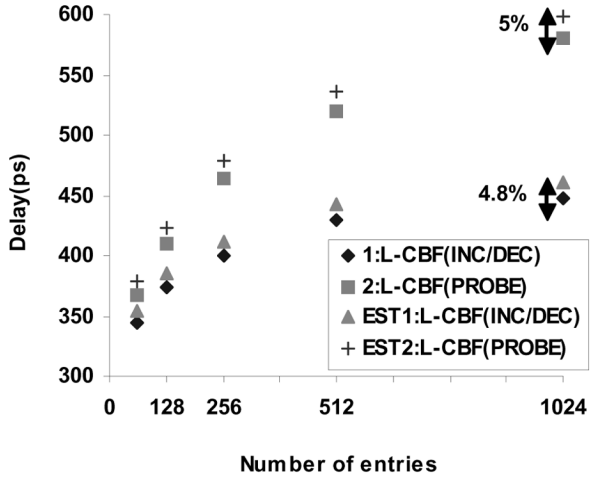

Fig. 16. Delay as a function of number of entries for L-CBF with 15-bit counts: simulation results and model estimations.

the length of the feedback loop and the delay of the LFSR. As discussed earlier, in practice, this increase is negligible for the widths considered in this study. Increasing the LFSR width increases the number of the inputs for zero detector, and hence the delay of it. We observe that the delay of L-CBF increases slightly for wider counts compared to S-CBF.

C. On the Accuracy of the Analytical Models

This section discusses the accuracy of the analytical models. In this analysis, the relative estimation error is calculated by (22)

$$\%\ \text{Error} = \frac{\text{Analytical} - \text{Simulation}}{\text{Simulation}} \times 100. \qquad (22)$$

Figs. 15 and 16 compare circuit measurements with analytical model estimations for energy and delay as a function of L-CBF's entry count. The circuit measurements are reproduced from Figs. 11 and 13, respectively. The worst case relative error per operations is also depicted. The worst case relative error for energy and delay is respectively within 10% and 5% of the Spectre simulation results. As observed, the error is monotonic and the estimations are in agreement with the simulation results in predicting the trend of delay and energy per operation variations.

Analytical model estimations may differ from simulation results because of several factors. Comparisons of the model estimated and layout extracted capacitances show that about 5% of the error is due to capacitance estimation inaccuracy. The formulas used to calculate gate and diffusion capacitances are over-simplified and the capacitances are assumed to be voltage independent. The energy model exhibits a worst case error of about 10%. The leakage power model accounts for 4.5% of this error. Leakage current largely depends on the state of the circuit Hence, it is difficult to quantify the leakage power accurately without circuit simulations.

VI. Conclusion

In this paper, we investigate physical level implementations of CBFs and we propose L-CBF. L-CBF is a novel implementation consisting of an array of up/down LFSRs and zero detectors. We compare L-CBF with S-CBF. S-CBF is the previously assumed implementation consisting of an SRAM array

of counts and a shared counter. We evaluate the energy, delay, and area of L-CBF and S-CBF in a commercial fabrication technology. L-CBF is superior to S-CBF in both delay and speed at the expense of more area. Additionally, we present analytical delay and energy models for L-CBF. These models facilitate estimation of the delay and energy variation for CBFs during architectural level investigations when physical level implementation is not yet available. Comparisons demonstrate that the estimations provided by the models are in satisfying agreement with the simulation results.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers of this paper and the reviewers of its earlier conference version for their helpful comments. The authors would also like to thank M. Haji Rostam and N. Azizi for their help in physical-level design and simulation.

REFERENCES

[1] A. Moshovos, "RegionScout: Exploiting coarse-grain sharing in snoop-coherence," in *Proc. Ann. Int. Symp. Comput. Arch.*, Jun. 2005, pp. 234–245.

[2] A. Moshovos, G. Memik, B. Falsafi, and A. Choudhary, "Jetty: Filtering snoops for reduced energy consumption in smp servers," in *Proc. Ann. Int. Conf. High-Performance Comput. Arch.*, Feb. 2001, pp. 85–96.

[3] S. Sethumadhavan, R. Desikan, D. Burger, C. R. Moore, and S. W. Keckler, "Scalable hardware memory disambiguation for high-ILP processors," *IEEE Micro*, vol. 24, no. 6, pp. 118–127, Nov. 2004.

[4] J. K. Peir, S. C. Lai, S. L. Lu, J. Stark, and K. Lai, "Bloom filtering cache misses for accurate data speculation and prefetching," in *Proc. Ann. Int. Conf. Supercomput.*, Jun. 2002, pp. 189–198.

[5] M. R. Stan, "Synchronous up/down counter with clock period independent of counter size," in *Proc. Ann. Symp. Comput. Arithmetic*, Jul. 1997, pp. 274–281.

[6] P. Alfke, "Efficient shift registers, LFSR counters, and long pseudo-random sequence generators," Xilinx, San Jose, CA, Appl. Note 052, Jul. 1996.

[7] P. H. Bardell, W. H. McAnney, and J. Savir, *Built-in test for VLSI: Pseudorandom techniques*. New York: Wiley, 1987.

[8] M. R. Stan, A. F. Tenca, and M. D. Ercegovac, "Long and fast up/down counters," *IEEE Trans. Comput.*, vol. 47, no. 7, pp. 722–735, Jul. 1998.

[9] B. S. Amrutur and M. A. Horowitz, "Fast low-power decoders for RAMs," *IEEE J. Solid-State Circuits*, vol. 36, no. 10, pp. 1506–1515, Oct. 2001.

[10] B. S. Amrutur, "Design and analysis of fast low power SRAMs," Ph.D. dissertation, Elect. Eng. Dept., Stanford Univ., Stanford, CA, 1999.

[11] B. S. Amrutur and M. A. Horowitz, "Speed and power scaling of SRAM's," *IEEE J. Solid-State Circuits*, vol. 35, no. 2, pp. 175–185, Feb. 2000.

[12] M. Margala, "Low-power SRAM circuit design," in *Proc. IEEE Workshop Memory Technol., Design Test.*, Aug. 1999, pp. 115–122.

[13] D. Burger and T. Austin, The Simplescalar Tool Set v2.0 Comput. Sci. Dept., Univ. Wisconsin-Madison, Madison, Tech. Rep. UW-CS-97-1342, 1997.

[14] H. E. W. Neil and D. Harris, *Principles of CMOS VLSI Design*, 3rd ed. Reading, MA: Addison Wesley, 2004.

[15] D. A. Hodges, H. G. Jackson, and R. A. Saleh, *Analysis and Design of Digital Integrated Circuits*, 3rd ed. New York: McGraw-Hill, 2004.

[16] M. Mamidipaka, K. Khouri, N. Dutt, and M. Abadir, "Analytical models for leakage power estimation of memory array structures," in *Proc. Int. Conf. Hardw./Softw. Co-Design Syst. Synth.*, Sep. 2004, pp. 146–151.

[17] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan, "Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects," Univ. Virginia, Charlottesville, Tech. Rep. CS-2003-05, 2003.

[18] X. N. Chen and L. S. Peh, "Leakage power modeling and optimization of interconnection network," in *Proc. Int. Symp. Low Power Electron. Des.*, Aug. 2003, pp. 90–95.

[19] S. Wilton and N. Jouppi, "An enhanced access and cycle time model for on-chip caches," 1994.

[20] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural level power analysis and optimizations," in *Proc. Ann. Int. Symp. Comput. Arch.*, Jun. 2000, pp. 83–94.

[21] E. Safi, A. Moshovos, and A. Veneris, "L-CBF: A fast, low-power counting bloom filter architecture," in *Proc. Ann. Int. Symp. Low Power Electron. Des.*, Oct. 2006, pp. 250–255.

Elham Safi (S'05) received the B.Sc. and M.Sc. degrees in computer hardware engineering and computer architecture from the University of Tehran, Iran. She is currently pursuing the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada.

Her research interests include computer architecture with emphasis on hardware design and implementation.

Andreas Moshovos (S'96–M'99–SM'05) received the Ptyhion degree and the M.Sc. degree in computer science from the University of Crete, Hellas, Greece, and the Ph.D. degree in computer science from the University of Wisconsin-Madison, Madison.

He is an Associate Professor with the Department of Electrical and Computer Engineering, University of Toronto. His research interests include microarchitectural optimizations for high-performance processors and systems.

He is a member of the Association for Computing Machinery (ACM).

Andreas Veneris (S'96–M'99–SM'05) received the Diploma in computer engineering and informatics from the University of Patras, Patras, Greece, the M.S. degree in computer science from the University of Southern California, Los Angeles, and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign (UIUC), Urbana.

He is currently an Associate Professor, cross-appointed with the Department of Electrical and Computer Engineering and Department of Computer Science, University of Toronto, Toronto, ON, Canada. His research interests include computer-aided design for the debugging, verification, synthesis and test of digital circuits and systems as well as data structures and combinatorics.

He is a member of the Association for Computing Machinery (ACM), the American Association for the Advancement of Science (AAAS), the Technical Chamber of Greece, and the Planetary Society.