ISSN:0975-9646

Data Identification using L-CBF

V.Srilatha, Dr. Rudra Pratap Das, S. Rama Krishna

Dept. of E.C.E, Pydah College Of Engineering & Technology,
Visakhapatnam, A.P, India

Abstract— counting bloom filters (CBFs) are used 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 studies the energy, delay, and area characteristics of two implementations for CBFs using full custom layouts in a commercial 0.13- m 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 presents analytical energy and delay models for L-CBF. 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.

INTRODUCTION

Many 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. CBFs have been also utilized to improve the scalability of load/store scheduling queues and to reduce instruction replays by assisting in early miss determination at the L1 data cache. In these applications, CBFs help eliminate broadcasts over the interconnection network in multiprocessor systems, CBFs also help reduce accesses to much larger and thus much slower and power-hungry content addressable memories, or cache tag arrays. In all above mentioned 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.

The CBF provides a definite answer for 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. If the key distribution is not known, or too complicated to yield to analysis, then the use of a particular hash function may have adverse effects: it may magnify correlations among keys and fill the hash table non-uniformly. In universal hashing, one of several hash functions is chosen at random. Here, we see a different technique in Bloom filters, several hash functions are applied to each key. Again, this allows us to use simple hash functions while at the same time minimizing the effects of any particular hash function.

The main purpose of Bloom filters is to build a space data structure for set membership. Indeed, to maximize space efficiency, correctness is sacrificed: if a given key is not in the set, then a Bloom filter may give the wrong answer (this is called a false positive), but the probability of such a wrong answer can be made small. A typical application of Bloom filters is web caching. An ISP may keep several levels of carefully located caches to speed up the loading of commonly viewed web pages, in particular for large data objects, such as images and videos. If a client requests a particular URL, then the service needs to determine quickly if the requested page is in one of its caches. False positives, while undesirable, are acceptable: if it turns out that a page thought to be in a cache is not there, it will be loaded from its native URL, and the penalty is not much worse than not having the cache in the first place.

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-m fabrication technology. 3) It presents analytical delay and energy models for L-CBF and compares the model estimations against simulation results.

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.

Introduction to CBFs:

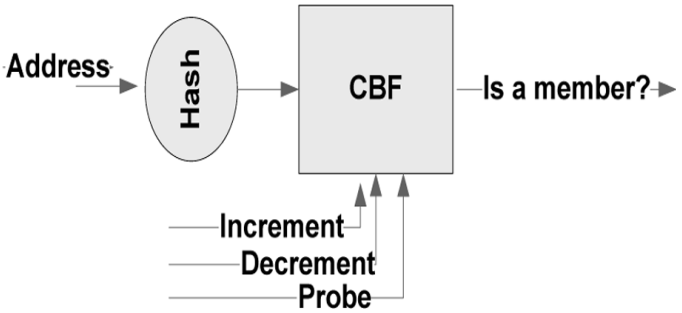


Figure 1: CBF as black box.

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. 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.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.

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;

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.

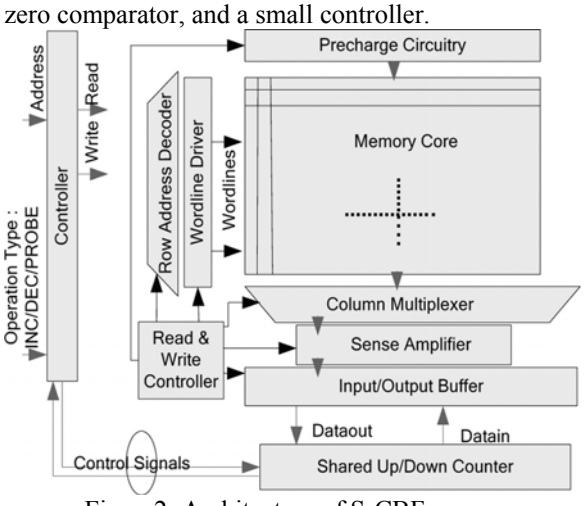


Figure2: Architecture of S-CBF

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) itis 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 speed up probe operations and to reduce their power. 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

L-CBF: LFSR-BASED CBF IMPLEMENTATION

More 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 needto 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. 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.

A. ***LFSRs*** A *maximum-length* -bit LFSR sequences through 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 bitsin 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).

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. Fig. 3 shows an 8-bit maximum-length Galois LFSR, its taps, and polynomial.

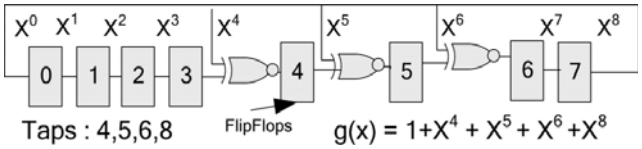
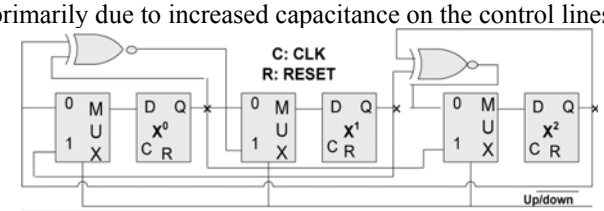


Figure : Eight bit maximum length LFSR.

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. 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). Delay increases only slightly with size, primarily due to increased capacitance on the control lines.



Up	Down
Counting	
000	000
001	010
100	110
011	101
101	011
110	100
010	001

Normal LFSR:

D0 = Q2
D1 = Q0
D2=Q1 XNOR Q2

$g(x) = 1+X^2+X^3$

Inverse LFSR:

D0 = Q1
D1 = Q2 XNOR Q0
D2 = Q0

$h(x) = 1+X+X^3$

Figure : Three-bit maximum-length up/down LFSR.

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

$g(x)=\sum C_i X^i \quad (C_0 = C_n = 1)$

in the above equation X^i corresponds to the output of the bit of the shift register and the constants are either 0 (no tap) or 1 (tap). Given , a primitive polynomial for an LFSR generates the reverse sequence as

$h(x)=\sum C_i X^{n-i} \quad (C_0 = C_n = 1)$

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).

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. 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 aspeed-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. These characteristics make LFSRs a suitable counter choice for CBFs.

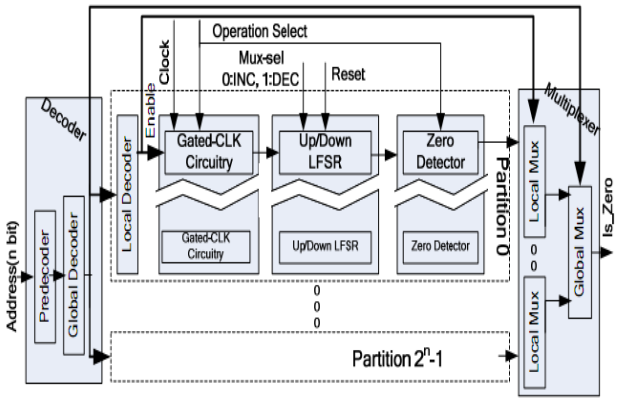
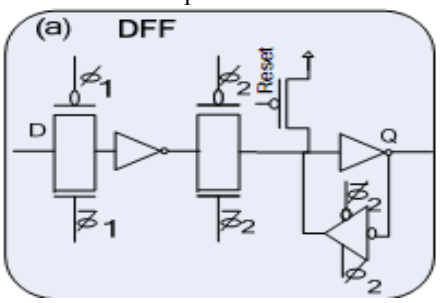
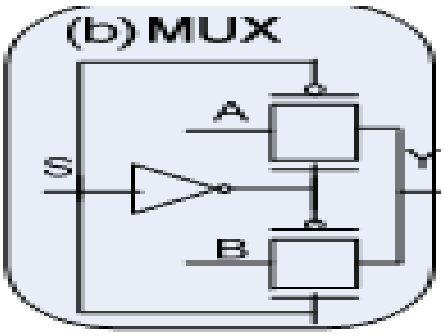


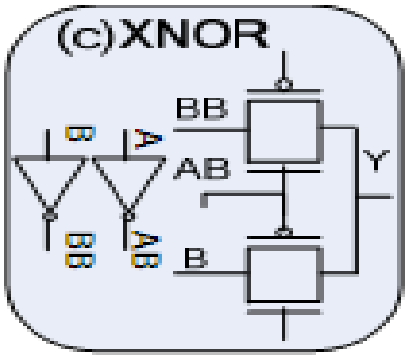
Fig. 5. Architecture of L-CBF; the basic cells of an up/down



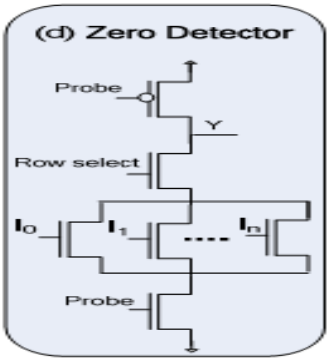
(a) Two-phase flip-flop;



(b) 2-to-1 multiplexer;



(c) XNOR gate;



(d) a bit-slice of the embedded zero detector

A.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. The decoder consists of a pre decoding 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.

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- m 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. Then, we present results for other CBF configurations.

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 delayand 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. 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. Moreover, we reduce power further via pulse operation techniques for the word-lines, the periphery circuits and the sense amplifiers. We also use multistage static CMOS decoding and current-mode read and write operations to further reduce power. For L-CBF, we utilize 16-bit LFSRs such that the LFSR can count at least 2^{15}values.

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

Table II shows the delay in picoseconds, the energy (static and dynamic) per operation in pico joules, 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. As observed from Table II, L-CBF is 3.7 and 1.6 x faster than S-CBF during update and probe operations, respectively. In addition, L-CBF consumes 2.3 or 1.4x less energy than S-CBFfor update and probe operations, respectively. These significant gains in speed and energy consumption come at the expense of more area. L-CBF requires about 3.2x more area than S-CBF.

RTL Schematic

RTL View is a Register Transfer Level graphical representation of the design. This representation (.ngr file produced by Xilinx Synthesis Technology (XST)) is generated by the synthesis tool at earlier stages of a synthesis process when technology mapping is not yet completed. The goal of this view is to be as close as possible to the original HDL code. In the RTL view, the design is represented in terms of macro blocks, such as adders, multipliers, and registers. Standard combinatorial logic is mapped onto logic gates, such as AND, NAND, and OR.

1.TopPartition

The top partition has five single bit inputs and an 8 bit input. It has eight single bit inputs and one 8 bit output. The 8 bit input data is the address of the item to be added/deleted or checked from a list. The outputs depend on the count of the item. There are eight individual partitions described in the top partition.

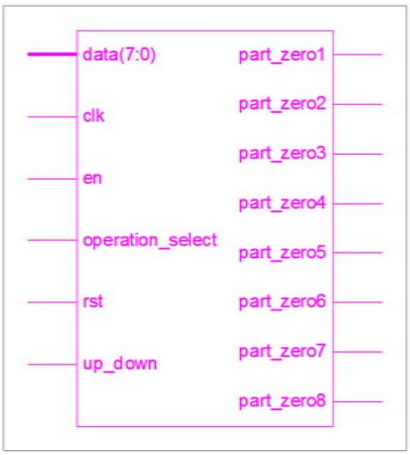


Figure : Top Partition

2. Hash function

The hash function is used to enter/delete an item into/from a list. When an item is checked weather it is in a particular list the answer may be a false positive. The false positiveness will be reduced by hash function. The input data is 8 bit and it has 8 single bit outputs.

3. Partitions

The addresses of the item to be entered into the list or deleted from the list or searched weather it is in the list or not is decoded and given to the 8 individual partitions. The each partition has a single bit output.

4. Gated clock

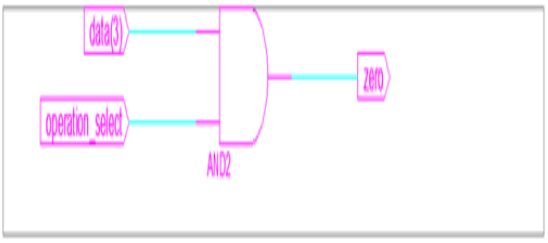


Figure: Gated Clock

The gated clock is an unit that has two inputs and output. One input is 4 bit input data and the other is the type of operation to be performed.

SIMULATION RESULTS

Simulation is a powerful and important tool because it provides a way in which alternative designs, plans and/or policies can be evaluated without having to experiment on a real system, which may be prohibitively costly, time consuming, or simply impractical to do.

1. Generating Hash function

The hash function is used to enter/delete an item into/from a list. When an item is checked weather it is in a particular list the answer may be a false positive. The false positiveness will be reduced by hash function. The input data is 8 bit and it has 8 single bit outputs.

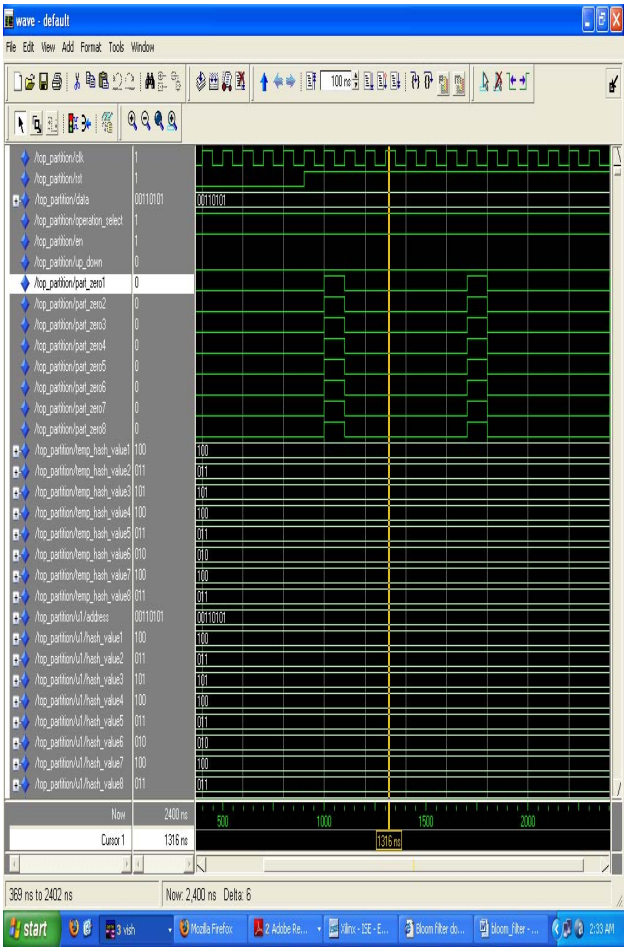


Figure : Generating hash function

2. Individual Partition

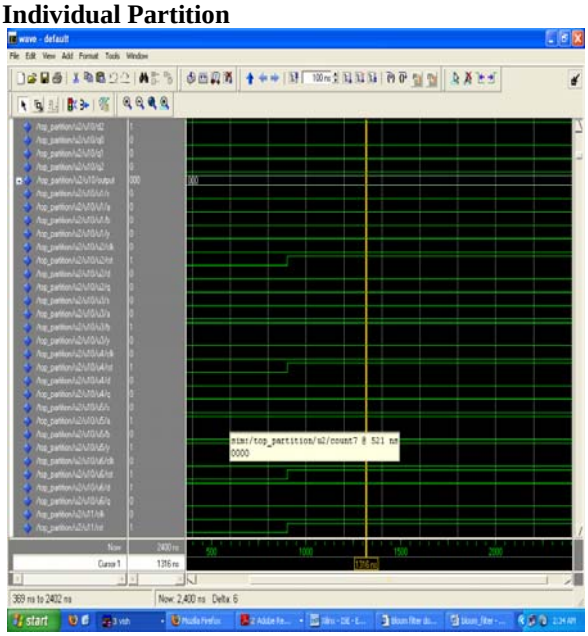
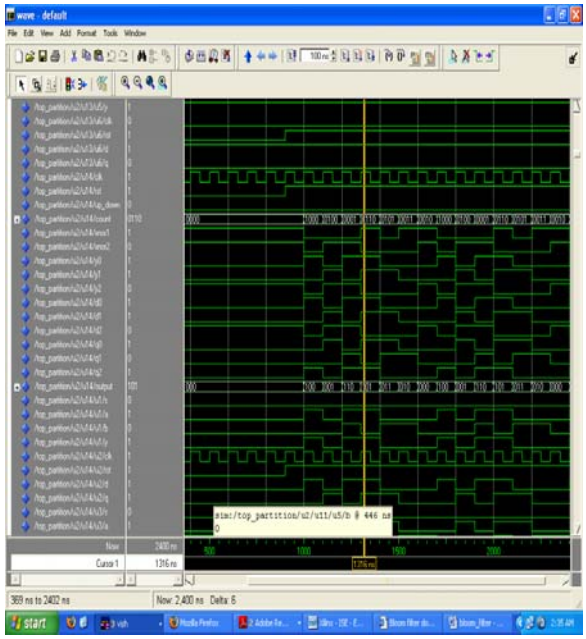


Figure : Individual partition result

The address of the item to be enter into the list or deleted from the list or searched weather it is in the list or not is decoded and given to the 8 individual partitions. The each partition has a single bit output. The output of the partition is either 1 or 0 based on whether the particular item is in the list or not.

3. LFSR Count



The LFSR's are used in up/down counters. When an item enters into a list its count increased by 1.if an item deleted from a set then the count will be decreased by 1. An LFSR is a shift register that, when clocked, advances the signal through the register from one bit to the next most-significant bit . Some of the outputs are combined in exclusive OR configuration to form a feedback mechanism. A linear feedback shift register can be formed by performing exclusive OR on the outputs of two or more of the flip-flops together and feeding those outputs back into the input of one of the flip-flops.

CONCLUSION

In this thesis the investigation of physical level implementations of CBFs is done and proposed LCBF. LCBF is a novel implementation consisting of an array of up/down LFSRs and zero detectors. Compare LCBF with SCBF is made. SCBF is the previously assumed implementation consisting of an SRAM array of counts and a shared counter. LCBF is superior to SCBF in both delay and speed at the expense of more area. The proposed LCBF is a novel implementation consisting of an array of up/down LFSRs and zero detectors. It will test the membership of the set by Increment, Decrement and probe operations in LFSR .It will produce the single out "is zero". Comparisons demonstrate that the estimations provided by the models are in satisfying agreement with the simulation results.

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-PerformanceComput. 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.