Available online at www.sciencedirect.com
ScienceDirect
Computers and Mathematics with Applications 55 (2008) 2005–2011
An International Journal
computers & mathematics with applications
www.elsevier.com/locate/camwa

Design of new XOR-based hash functions for cache memories☆

Sung-Jin Cho[a,*], Un-Sook Choi[b], Yoon-Hee Hwang[c], Han-Doo Kim[d]

[a] *Division of Mathematical Sciences, Pukyong National University, Busan 608-737, Republic of Korea*
[b] *Department of Multimedia Engineering, Tongmyong University, Busan 608-711, Republic of Korea*
[c] *Department of Information Security, Graduate School, Pukyong National University, Busan 608-737, Republic of Korea*
[d] *School of Computer Aided Science, Inje University, Gimhae, 621-749, Republic of Korea*

Received 12 April 2006; accepted 27 July 2007

Abstract

A hash function H is a computationally efficient function that maps bitstrings of arbitrary length to bitstrings of fixed length, called hash values. Hash functions have a variety of general computational uses. They are used in processors to augment the bandwidth of an interleaved multibank memory or to enhance the utilization of a prediction table or a cache. In this paper, we design new XOR-based hash functions, which compute each set index bit as XOR of a subset of the bits in the address by using the concepts of rank and null space. These are conflict-free hash functions which are of different types according to whether m is even or odd. To apply the constructed hash functions to the skewed-associative cache, we show that the degree of interbank dispersion between two hash functions is maximal.

Keywords: XOR-based hash functions; Conflict-free; 2-way skewed-associative caches; Interbank dispersion; Null space; Rank

1. Introduction

A hash function H is a computationally efficient function that maps bitstrings of arbitrary length to bitstrings of fixed length, called hash values. Hash functions have a variety of general computational uses. They are used in processors to augment the bandwidth of an interleaved multibank memory or to enhance the utilization of a prediction table or a cache [1–6]. The basic requirements for designing hash function are: the input can be of any length, the output has a fixed length, $H(x)$ is relatively easy to compute for any given x, $H(x)$ is 1-way, and $H(x)$ is conflict-free. Bank conflicts can severely reduce the bandwidth of an interleaved multibank memory and conflict misses increase the miss rate of a cache. Therefore it is important that a hash function has to succeed in spreading the most frequently occurring patterns over all indices. So it maps the elements of the frequently occurring patterns without conflicts.

Frailong et al. [7] suggested XOR-schemes that allowed parallel access to all previously mentioned data templates, which are of use in image processing and numerical analysis. Vandierendonck et al. [8] constructed XOR-based

☆ This work was supported by grant No. (R01-2006-000-10260-0) from the Basic Research Program of the Korea Science and Engineering Foundation.
* Corresponding author.
E-mail addresses: sjcho@pknu.ac.kr (S.-J. Cho), choies@tu.ac.kr (U.-S. Choi), yhhwang@pknu.ac.kr (Y.-H. Hwang), mathkhd@inje.ac.kr (H.-D. Kim).

0898-1221/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2007.07.008

hash functions which provided conflict-free mapping for a number of patterns for multibank memories and skewed-associative caches. They used fundamental notions of null space and column space for constructing these functions [8]. Their functions map $2m$ bits to $m(= 2k)$ bits which are conflict-free. But they did not construct hash functions which are conflict-free when m is odd. Also they constructed two XOR-based hash functions for skewed-associative caches [8–10]. But the degree of interbank dispersion between two hash functions is less than $2m$, which is the maximum degree between them. So they changed the basis of one hash function to overcome this problem.

In this paper, we design new XOR-based hash functions, which compute each set index bit as XOR of a subset of the bits in the address by using the concepts of rank and null space [11–14]. These are conflict-free hash functions which are of different types according to whether m is even or odd. To apply the constructed hash functions to the skewed-associative cache, we show that the degree of interbank dispersion between two hash functions is maximal. And thus we need not change the basis of the null space of the constructed hash functions.

2. Hash functions

In this section we introduce XOR-based hash functions in a general way and then we describe their representation by a vector–matrix product. Hash functions are used in processors to increase the bandwidth of an interleaved multibank memory or to improve the use of a cache. They map an address to a set index. Since all indices are used, the function should be surjective.

Definition 2.1. A hash function is a function from $\{0, \ldots, 2^n - 1\}(:= \mathbf{A})$ of n bit addresses to $\{0, \ldots, 2^m - 1\}(:= \mathbf{S})$ of m bit indices $(m < n)$.

Especially XOR-based hash functions can be modeled by means of a matrix representation and by using null spaces.

1. The matrix representation

An n bit address $\mathbf{a}$ is represented by a bit vector $a_1, \ldots, a_n$. A hash function mapping n bits to m bits is represented as a binary matrix H with n rows and m columns. Since H is surjective, the image of $\mathbf{A}$ is $\mathbf{S}$. Therefore $\operatorname{rank}(H) = \dim(\mathbf{S}) = m$, where $\dim(\mathbf{S})$ is the dimension of $\mathbf{S}$. The bit on the ith row and the jth column is 1 when address bit a_i is an input to the XOR-gate computing the jth set index bit. Consequently, the computation of the set index $\mathbf{s}$ can be expressed as the vector–matrix multiplication over $GF(2)$, where addition is computed as XOR and multiplication is computed as logical AND, denoted by $\mathbf{s} = \mathbf{a}H$.

2. Null spaces

Since matrices are considered as linear transformations, they can be characterized by their null spaces. The null space of a matrix H is the set of all addresses which map to index $\mathbf{0}$, namely $N(H) = \{\mathbf{x} : \mathbf{x}H = \mathbf{0}\}$.

The following theorem is well-known.

Theorem 2.2 (*See [15]*). *Let A be an $n \times m$ matrix. Then*

$$n = \operatorname{rank}(A) + \dim N(A).$$

Theorem 2.3. *Let A and B be $n \times m$ matrices. Then $N(A) \cap N(B) = N([A\ B])$, where $[A\ B]$ is the augmented matrix of A and B.*

Proof. The result is obtained by the following equivalence.

$$\mathbf{x} \in N(A) \cap N(B) \Leftrightarrow \mathbf{x}A = \mathbf{x}B = 0 \Leftrightarrow \mathbf{x} \in N([A\ B]). \quad \square$$

3. Design of efficient XOR-based hash functions

In this section we model efficient XOR-based hash functions when the number of address bits is $2m$, where m is even or odd. We propose the new model of XOR-based hash functions for two cases. We give XOR-based hash functions by the following.

Definition 3.1. We define XOR-based hash functions of four types as the following:

(i) $m = 2k - 1$ $(k \in \mathbf{N})$, where $\mathbf{N}$ is the set of all positive integers.