Increasing the hash-memory effectiveness by using reconfigurable Boolean hash functions generators

Dr. E.G BARDIS
Hellenic Ministry of Public Order, Athens,
4, P. Kanelopoulou
10177 Athens
GREECE

Dr. A.LEROS
Associate Professor
Department of Automation
Technological Education Institutes of Halkis
34400 Psahna, Halkis, Evia,

D.V. ANDRIKOU, MSc
Telecommunications and Computer Science Engineer
GREECE

Abstract: - In this paper a new approach for increasing the effectiveness of hash-memory using a hash-function generator is proposed. It has been shown that using an orthogonal Boolean functions system generator which is key-reconfigured allows decreasing the collision probability and increasing the effectiveness of probing and of hash-memory utilization. Key-generated hash-functions can be useful for data security in hash-memory. The hash-memory structure is designed based on hash-functions key-reconfigured generator. The implementation of the proposed functions generator is oriented in Field Programmable Gate Array (FPGA). The rating of the hash-memory effectiveness increase by the proposed approach is also presented.

Key-Words: - Hash memory, hash functions, reconfigurable Boolean hash functions generators

1 Introduction

Future development of computer technology and its utilization spanning all spheres of human activity are determined by the increase rate of the computers productivity.

In recent years a considerable rise of the computer systems productivity has been achieved due to the expansion of parallel computation.

The main part of research efforts has been concentrated on the organization or architecture of concurrent processors. The problem of parallel data computing in memory is more complicated. The most drastic way to solve this problem consists of using hardware Content Addressable Memory (CAM) devices. But the existent nowadays VLSI CAM have insufficient memory capacity and plentiful cost for the majority of practical applications.

At the same time, there are sufficiently many practically important tasks, the solution of speed which in a determining degree depends on the time needed for information search in memory.

For this class of tasks, the time spent on the searching and sorting operations in memory is determined by the memory's function of sequential nature.

The need for the parallel execution of operations in memory has stimulated recently the intense development of the CAM Devices.

However, the increase of the information volumes up to now, anticipates the rates of the improvement of the VLSI CAM characteristics, which makes their application justified only for solving a relatively small number of searching tasks.

Therefore the studies, directed towards increasing the effectiveness of the software method of the associative access to the memory, are important. This approach is called hash - addressing and is being utilized since the 60's.

The achieved progress of the processor, means productivity, and the increased possibilities of the integrated technology make it possible to approach the problem of the improvement of hash - addressing from new positions, using for this, fundamentally new possibilities.

It is known that the main disadvantage in the hash - addressing is the incomplete use of memory capacity, but the steady tendency of the increase in the capacities of VLSI RAM makes it possible to substantially decrease the significance of the indicated disadvantage for practical applications.

On the other side, the intensive development of the FPGA technology allows to realize more effective hash - algorithms.

2 Basic determinations and problem formulation

The essence of hash-addressing is that the address in which the key is stored in memory, depends on the key's code.

In the hash-memory, pairs <X,Y> are usually stored, where X is an n-bits length search key and Y is a q-bits length information code, connected to the search key X.

Let M be the total hash-memory addresses number, m is a real number of keys and V is the total hash-memory capacity. Then the load factor of hash-memory is defined as: $\alpha=m/M$ and the coefficient of memory efficiency w is defined as: $w=m\cdot n/V$. If n-bits length keys are stored in hash-memory then $w=\alpha$. The main time characteristics of hash-addressing are the average T_a and maximum T_m number of accesses in memory for key searching.
The hash-memory effectiveness is characterized by the dependence of the average T_a and maximum T_m number of accesses in memory for key searching from the load factor α of hash-memory [2].

Usually the complete n-bits length keys are stored in the hash-memory [2]. Before the key X is recorded into hash-memory, its h-bits length hash-address H(X) is calculated ($h=\lfloor \log_2 M \rfloor$). If the addressed by H(X) hash-memory cell is free, then in this cell the key X and the corresponding to X information Y will be recorded. Otherwise the key X and information Y will be recorded into the first after H(X) free hash-memory cell (linear probing).

The hash-address for given key X is formed as a result of hash-transformation H(X) and it is called primary hash-address. The hash-addressing collision is that situation when two different keys X_1 and X_2 correspond to one hash-address: $H(X_1)=H(X_2)$, $X_1\neq X_2$. Usually, the collisions are solved by probing. Probing is a process of forming secondary hash-addresses $H_2(X)$, $H_3(X)$,…, $H_t(X)$ in case of collision. The probing process is finished if the hash-memory cell addressed by $H_t(X)$ is free.

There are in existence some probing types: linear, when $H_j(X) = H(X)+j-1$ or quadric: $H_j(X) = H(X)+(j-1)^2$. It is important that for all probing types there is a strong correlation between primary and secondary hash-addresses. This is the cause of that, if primary hash-addresses grouping is arising, then secondary hash-addresses grouping is arising, too. This in turn, causes the probing to increase and correspondently slows down the key searching rate. Thus, the current ways of probing are one of the most important factors that the average and especially the maximum search time is increasing.

The hash-memory effectiveness depends on the following factors:
- the hash-algorithm's quality, which is determined by its ability to generate random-evenly distributed hash-addresses on any distribution of keys. This ability ensures the minimum probabilities of primary collisions appearance;
- the time of hash-transformation implementation;
- the way that hash-addressing collisions are solved, which defines the probabilities of secondary collisions.
- the effectiveness of the memory utilization.
- the possibility of the data access control in hash-memory.

In case of using a hash-transformation, which ensures an "ideal" random distribution of hash-addresses, the average T_a number accesses in memory for key searching is determined by the formula [3]:

$$T_a = \frac{1}{1-\alpha} \qquad (1)$$

From the theoretical point of view, the "ideal" random distribution of hash-addresses may be achieved by using a system of orthogonal balanced Boolean functions $f_1(x_1,.., x_n),\ldots,f_h(x_1,\ldots, x_n)$ which are determined on sets of key bits.

The Boolean function $f(x_1,\ldots, x_n)$ is called balanced [1], if its truth table contains equal number of zeros and ones:

$$\sum_{X\in Z} f(X) = 2^{n-1} \qquad (2)$$

where the Z- set consists of all 2^n possible sets of values of n variables.

The system $f_1(x_1,.., x_n),\ldots,f_h(x_1,\ldots, x_n)$, $h\leq n$, of Boolean functions is called orthogonal, if the linear combination of any arbitrary subset of these functions is a balanced function [3].

The basic condition for the absence of the collisions occurrence in the secondary hash-addresses grouping is the use of such mechanisms for the formation of secondary hash-addresses, that would ensure the random distribution of the secondary hash-addresses at any distribution of the primary hash-addresses [3].

In other words, this means that the mechanism of probing must ensure the absence of correlation between the primary and secondary hash-addresses. This condition is not satisfied with linear probing.

In the work [1] it is shown that the effectiveness of the hash- memory utilization can be considerably increased due to the elimination of information

redundancy, which occurs during the storage of full-size keys in the hash- memory.

For this, it is proposed [1] at the hash - address H(X) to store not the full-size code of key X, but its convolution S(X), formed in such a way that the pair of the codes <H(X),S(X)> would unambiguously correspond to the code of key X. In this case, the coefficient w of hash - memory efficiency exceeds α and is:

$$w = \frac{\alpha}{1 - \frac{h - \lfloor \log_2 m \rfloor}{n+q}} \qquad (3)$$

To eliminate the information redundancy during the organization of hash- memory it is necessary that both the hash - algorithm and the convolutions formation algorithm would provide the condition of maximum total information entropy.

So, the functions that formate the separate bits of the hash - addresses, considered as Boolean functions $f_1(x_1,\ldots, x_n)$, …, $f_h(x_1,\ldots,x_n)$ and the Boolean functions that generate the bits of the hash-convolutions $\varphi_{h+1}(x_1,\ldots, x_n),\ldots, \varphi_n(x_1,\ldots, x_n)$ must form an orthogonal system:

$$\begin{aligned} H(x_1,\ldots,x_n) &= \sum_{j=1}^{h} 2^{j-1} \cdot f_j(x_1,\ldots,x_n) \\ S(x_1,\ldots,x_n) &= \sum_{i=h+1}^{n} 2^{i-h-1} \cdot \varphi_i(x_1,\ldots,x_n) \end{aligned} \qquad (4)$$

The majority of the hash - algorithms used in practice are based on the operations of multiplication, of finding the remainder from the division, multiple summing of the fragments and others. The approach in [2] is oriented to the effective realization of the central processor and does not satisfy the requirements of maximum entropy condition.

This means that, firstly, they do not ensure the strictly random distribution of the formed hash - addresses (i.e. even distribution), realizing only quasi-random distribution.

Secondly, these hash-algorithms do not allow to realize the elimination of the inherent hash - memory information redundancy, and correspondingly, they do not allow to achieve the theoretically possible level of the effectiveness of the hash - memory utilization.

Thirdly, the existing hash - algorithms cannot ensure the data protection in the hash – memory [4]. Special means are additionally required for the realization of data protection functions.

Thus, the use of hash – transformations based on the arithmetic operations, which are effectively realized on the universal processor is the factor, which limits the total effectiveness of hash - memory.

In the work of [1] it is proposed for decreasing the collisions probability and eliminating the information redundancy to use a system of orthogonal linear Boolean functions.

This approach, although it makes it possible to increase the effectiveness of hash - addressing due to the decrease of collisions and elimination of information redundancy, leaves without solution a number of problems connected with the information security in the hash – memory and secondary hash-addressing.

For the realization of these important components of the hash-memory effectiveness additional special means are required.

The existing approaches to avoid the collisions via probing (linear or quadratic) do not ensure entirely the exception of the phenomenon of the secondary hash-addresses grouping [2].

It is obvious that for the effective solution of the collisions problem, the algorithm that calculates the secondary hash- address must be according to its properties identical to the primary (basic) hash-algorithm.

In some systems of hash- addressing [2], two different hash - algorithms are used (for the primary and secondary hash - transformation), however, this solution should not be recognized as acceptable due to the fact that with their use remains open the problem of the realization of tertiary, quaternary and so forth of hash - transformations.

The analysis performed on the problem of increasing the hash- addressing effectiveness showed that the majority of studies is directed towards a partial improvement in one or several of the effectiveness criteria given above.

Until now there hasn't been designed an approach to the selection of the hash - algorithm, which would make it possible to completely improve all criteria of the effectiveness of the hash - memory.

3 Structural organization of hash – memory, based on orthogonal nonlinear Boolean functions systems generators

The basic limiting factor at the selection of a hash – algorithm is the effectiveness of its software realization on the universal processor.

At the same time, the dynamical progress of FPGA technology allows to a considerable extent to remove the limitations, connected with the factor of

the effective software realization, and respectively, to be oriented to a wider circle of possible hash - transformation algorithms when designing the hash - memory.

The application of FPGA - structures, allows from the theoretical hash - memory point of view to directly realize optimal hash - transformsation algorithms in the form of orthogonal Boolean functions systems, whose software calculation on the central processor proves to be ineffective.

For the effective avoidance of collisions via probing, it is expedient to use a basic hash- transformer not only for the primary hash - address formation, but also for the secondary hash - addressing.

It is obvious that a more effective solution of the problem is the design of reconfigurable Boolean hash-transformers i.e., it is necessary to use as a hash - trasformer a generator of independent functions, each of which ensures the random distribution of the formed hash - addresses.

The use of an orthogonal Boolean functions systems generator allows besides the minimization of primary collisions, the elimination of the phenomenon of the secondary hash-addresses grouping and the effective use of memory capasity due to the elimination of the information redundancy in the hash- memory, and also ensures the data protection from an illegal access to hash-memory.

In fig.1. the hash - memory structure based on the orthogonal Boolean functions systems generator is given.

The generator of the orthogonal Boolean functions systems has a control of r-bit input ($r \leq n$) and an n-bit input for the supply of the key- argument of search - X.

For every code of the control key K, the generator forms h Boolean functions of hash- address $H_K(X)$ and n-h functions of hash - convolutions $S_K(X)$, that form an orthogonal system.

The key K of the reconfiguration of the hash - memory function generator is known to legal users.

When writing the information into the hash - memory, the hash - address $H_K(X)$ is calculated according to the key X, if the addressed hash - memory cell is free, then the formed by the generator code hash - convolutions $S_K(X)$ and the information Y, connected to the key X is written to the cell addressed by $H_K(X)$.

If the cell is occupied, then using the probing counter the control key K is incremented and, respectively, the secondary codes of hash - address $H_{K+1}(X)$ and hash - convolution $S_{K+1}(X)$ are calculated.

If the cell, addressed by $H_{K+1}(X)$ is free, then the secondary hash - convolution $S_{K+1}(X)$ and the information Y associated to the key X is written in it.

Even with the secondary hash – addressing, if the cell is occupied, then by using the described mechanism a tertiary $<H_{K+2}(X), S_{K+2}(X)>$, quaternary $<H_{K+3}(X), S_{K+3}(X)>$ and so forth pairs of functions are generated until a free hash - memory cell is found at the j-th step of probing, addressed by $H_{K+j}(X)$.

In this cell the hash- convolution $S_{K+j}(X)$ and the information Y, linked to the key X, will be written.

At the search operation, the legal users of hash – memory, install the initial reconfiguration key K.

By the given search argument X, the calculation of hash - address $H_K(X)$ and hash - convolution $S_K(X)$ is carried out.

From the memory cell, addressed by $H_K(X)$, the code of convolution $H_K(X)$ is read and compared to the calculated code $S_K(X)$.

In the case when the codes are equal, the search is finished and from this cell the information Y, linked to the key X is read.

Otherwise, by the probing counter the increment of the reconfigured key K is performed, and the secondary functions $H_{K+1}(X)$ and $S_{K+1}(X)$ are calculated. From the cell, addressed by $H_{K+1}(X)$ the code $S[H_{K+1}(X)]$ is read, which is then compared to $S_{K+1}(X)$.

The described process continues for j times until either a cell is found for which $S[H_{K+j}(X)]=S_{K+j}(X)$ or until the cell, addressed by $H_{K+j}(X)$ is empty. In this case there is no information in the memory, linked to the given key X - search argument.

In the described structure of hash – memory, the utilized generator at any key K, forms an orthogonal system of Boolean functions $H_K(X)$ and $S_K(X)$.

This ensures the random distribution of the formed hash - addresses in both the process of primary hash - addressing and the process of probing.

Thus in the proposed structure, the possibility of the negative phenomenon of the secondary group of writtings is principally removed.

Since in both the primary hash - addressing and in the probing process the uniqueness of the correspondence of the key X to the search pair $<H_K(X), S_K(X)>$ is ensured, this makes it possible to noticeably increase the coefficient of w of the hash - memory's eficiency in comparison with both the classical hash - memory [2] and the structures of hash - memory with a constant hash - transformer based on orthogonal linear functions [1]:

$$w = \frac{\alpha}{1 - \frac{h}{n+q}} \qquad (5)$$

The functions of protection from the unsanctioned access to hash - memory are ensured due to the fact that the key for the hash - functions generator reconfiguration is known only to licenced users.

The large key length and the arbitrary nature of its selection ensure from one side the high level security (the required sorting volume for the key selection it composes is 2^n), and from the other the convenience in the key selection.

For the realization of the hash - memory access protection, the Boolean functions, formed by the generator must have high nonlinearity and correspond to SAC for any key reconfiguration K.

One additional important advantage of the proposed hash-memory structure with the reconfigurated hash-functions is the possibility of the multiple-access to the hash-memory by different users.

In this case, each of the users has his own private key for the reconfiguration of the hash-addresses generator of the general resource hash-memory, which secures his data from other users unsanctioned access.

The proposed concept of the multiple-access to hash-memory allows to improve the statistical characteristics of hash-memory and to partially remove the limitation on the number of records allowed to each user (while the total number of records of all users remains unchanged).

Utilization of the reconfigurable by key generator of orthogonal Boolean functions systems in hash-memory provides the possibility for an effective procedure of the hash-algorithms selection, which do not generate collisions for constant arrays of keys (static hash-addressing).

Thus, utilizing in the hash-memory, the reconfigurable by key generator of orthogonal Boolean functions system makes it possible to substantially increase its effectiveness, practically on all criteria.

4 Construction and realization of the orthogonal functions generator

The key point for the practical use of the proposed approach to increasing the hash-memory effectiveness due to utilization of the reconfigurable by key generator of hash-conversions, is obtaining a system of n Boolean functions $\phi_1(x_1,\ldots, x_n, k_1,\ldots, k_n),\ldots,\phi_n(x_1,\ldots, x_n, k_1,\ldots, k_n)$, determined on the set of bits of the key $X=\{x_1,\ldots,x_n\}$ - search argument and the reconfiguration key's set of bits $K=\{k_1,\ldots, k_n\}$, so that at any value of K, this system would be transformed into a system of orthogonal Boolean nonlinear functions $f_1(x_1,\ldots, x_n)$, …, $f_h(x_1,\ldots, x_n)$, $\varphi_{h+1}(x_1,\ldots, x_n),\ldots, \varphi_n(x_1,\ldots, x_n)$.

To ensure the data security in hash-memory each of the these functions must have high nonlinearity and satisfy SAC.

The given requirements for the hash-memory function generator in the majority of points coincide with the requirements, presented to the Boolean functions system generator, utilized in cryptographical algorithms [2].

Therefore, for the realization of the proposed hash-memory structure, the existing orthogonal Boolean functions system generator for the data security algorithms can be used.

In the simplest case, the cipher blocks of DES or Rijndael algorithms can be used as such a generator.

They have inputs, which can be used as search argument inputs in the hash-memory and key inputs, which can be used as inputs for the generator's reconfiguration.

At the cipher blocks, the output of an orthogonal system of nonlinear Boolean functions is formed. This approach to the generator's construction is justified at utilization of high speed specialized microcircuits for the cipher blocks realization.

In a different version, the synthesis of the hash-functions generator can be brought to the task of the synthesis of an orthogonal Boolean functions system, which have high nonlinearity and satisfy SAC.

Actually, it is known [2] that if functions $f_j(x_1,\ldots, x_n)$, $j=1,\ldots,n$ form a system of orthogonal Boolean functions, then the Boolean functions system $f_j(X\oplus A)=f_j(x_1\oplus\alpha_1,x_2\oplus\alpha_2,\ldots,x_1\oplus\alpha_2)$ will also be orthogonal at any binary vector $A=(\alpha_1,\alpha_2,\ldots,\alpha_n)$, $\alpha_i \in \{0,1\}$, $i=1,\ldots,n$.

Hence it follows that, the generator reconfigurable by a key can be realized in the form of Boolean functions system $f_1(x_1\oplus k_1,\ldots,x_n\oplus k_n),\ldots, f_h(x_1\oplus k_1,\ldots,x_n\oplus k_n),\varphi_{h+1}(x_1\oplus k_1,\ldots,x_n\oplus k_n),\ldots,\varphi_n(x_1\oplus k_1,\ldots,x_n\oplus k_n)$.

The design methods for such orthogonal systems of Boolean functions is proposed in the work [2].

For the practical realization of calculating the designed system of Boolean functions, it is most expedient to use programmable computational structures, which are very efficient for the realization of Boolean functions systems.

The preliminary analysis of the Boolean functions systems realization versions showed that

the hash-address formation time attainable using FPGA- structures will be substantially less in comparison with the time needed for the realization of the calculating procedures for hash-algorithms on the central processor.

5 Conclusion

The proposed approach to increasing the hash-memory effectiveness due to the utilization of a reconfigurable generator of functions, formating hash-addresses, makes it possible to improve the majority of the effectiveness criteria and to enlarge the sphere of the hash-memory use.

The utilization of a generator of orthogonal system of Boolean functions for the hash-address and hash-convolusions allows to completely eliminate the inherent information redundancy in hash-memory, and due to this, also allows to increase the effectiveness of the accumulator's volume usage in comparison with the hash-addressing with a constant hashing function.

It was shown that the reconfigurable hash-functions generator makes it possible to effectively realize the data security functions in hash-memory, and also creates the prerequisites for constructing hash-memory systems with multichannel access.

The statistical simulation of the proposed hash-memory structure was presented along with the software realization of the orthogonal Boolean functions systems generator.

The simulation results confirmed the effectiveness of this approach for the total decrease of collisions, due to reduction in the probability of primary collisions, and also confirmed the absence of the phenomena of the second group of records in the hash-memory.

In particular, the use of the same mechanism for the formation of both the primary and secondary hash-address allows the decrease of the frequency of collisions by 10-20%.

References:

[1] Bardis E.G., Bardis N.G., Markovskyy A.P., Spyropoulos A.K., "High Storage Utilization of Hash Memory by Reducing of Information Redundancy for Hashing". *IMACS/IEEE CSCC'99, "Software and Hardware Engineering for the 21th Century*", ISBN: 960-8052-06-8, pp. 272-276, 1999.

[2] Kohonen T. Content-Addressable Memories. *Springer-Verlag. Berlin*.-1987,- 272 p.

[3] Polymenopoulos A., Bardis E.G., Bardis N.G., Markovskaja N.A., "Perfect Hashing Using Linear Boolean Functions", *WSEAS Press - Problem in Applied Mathematics and Computational Intelligence,* ISBN: 960-8052-30-0, 2001, pp. 5-11.

[4] Wayner P.C. Content-addressable search engines and DES-like systems. // *Advances in Cryptology –Crypto'92 Proceeding, Lecture Notes in Computer Science 740*-1992-P.575-586.

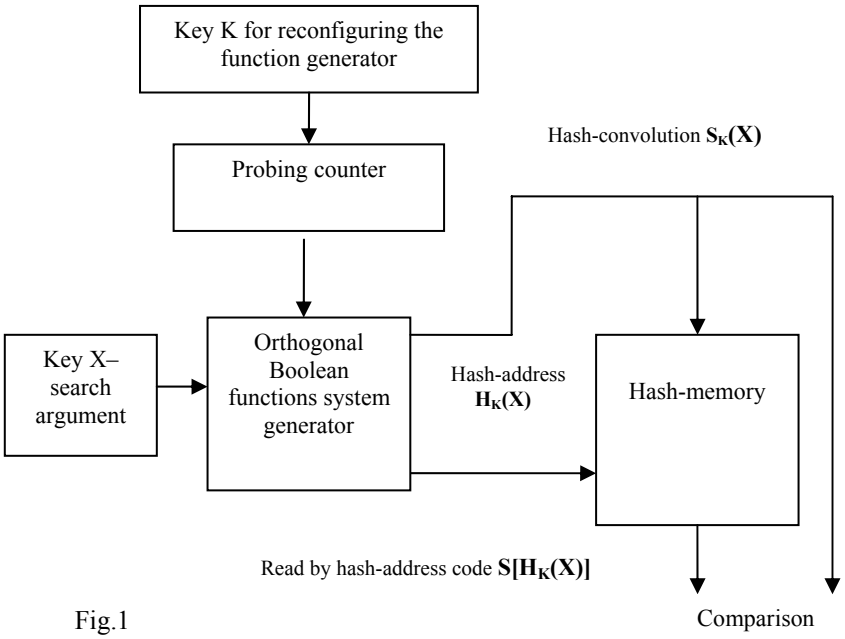

Fig.1