Reliability Increasing Method Using a SEC-DED Hsiao Code to Cache Memories, Implemented with FPGA Circuits

NOVAC Ovidiu[1], SZTRIK Janos[2], VARI-KAKAS Stefan[3], KIM Che-Soong[4]

University of Oradea, Romania,
[1]Department of Decorative Arts and Design, Faculty of Visual Arts,
[3]Department of Electrical Engineering, Faculty of Electrical Engineering and Information Technology,
1, Universităţii Str., 410087 Oradea, Romania, E-Mail: ovnovac@uoradea.ro

[2]University of Debrecen, Hungary,
Department of Informatics Systems and Networks, Faculty of Informatics,
Egyetem tér 1., 4032 Debrecen, Hungary, E-Mail: jsztrik@inf.unideb.hu

[4]Sangji University, South Korea,
Department of Industrial Engineering,
Kangwon 220-7021, Wonju, South Korea, E-Mail: dowoo@sangji.ac.kr,

Abstract ***– In this paper we will apply a Hsiao code to the cache level of a memory hierarchy to increase the reliability of the memory. We have selected the Hsiao code from the category of SEC-DED (Single Error Correction Double Error Detection) codes. For correction of a single-bit error we use, a check bits generator circuit, a syndrome generator and a syndrome decoder. Implementation of SEC-DED code in the cache is made with FPGA Xilinx circuits.***

Keywords: ***SEC-DED; cache; FPGA circuits; HSIAO code;***

I. INTRODUCTION

In the design of the computer systems an issue that raises particular problems is the slowly increase of memory speed compared with the increase of processor speed [1],[2]. The processor allocates, during the execution time, an increasing fraction of time, waiting for data to be brought from main memory. To reduce the gap between processor speed and memory speed, current processors allocates most of their hardware resources, to the cache level. For example Intel Itanium 2 processor allocates 86% of its transistors to L3 cache level [3],[5]. Cache memory is the fastest storage buffer for the central processing unit of a PC.

In this paper we will use both names: cache and cache memory.

In the efficiency analysis methods to increase the dependability of a memory hierarchy, the most vulnerable part to turn to critical applications in terms of reliability is the cache memory.

A memory hierarchy is the solution to the need for programmers to have a large and fast memory. This hierarchy is organized on several levels, each with less storage capacity, higher speed and cost per bit higher than the previous level. The objective that we are looking, in a memory hierarchy, is to obtain a memory system that have a cost almost as low as the cheapest level of memory and speed almost as great as fastest level of hierarchy [6].

Memory hierarchy is based on several fundamental properties of information storage technology. Different storage technologies have different access times. Faster technologies have a higher cost per bit than slower technologies, but those have a greater capacity to store the information. Figure 1 represents such a memory hierarchy [7].

The hierarchy has at the bottom level, the slowest, most expensive and the highest capacity storage. As we move to the top of the pyramid, we have increasingly faster levels, greater cost per bit and with less storage capacity.

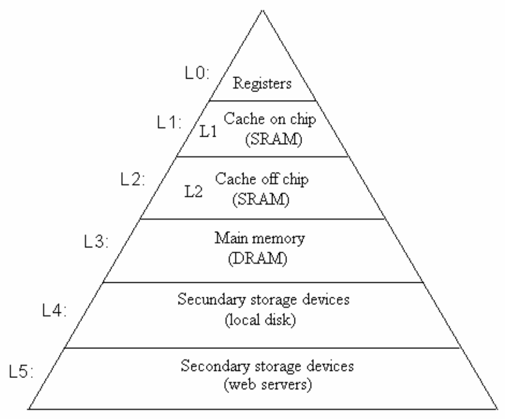

Figure 1. Memory hierarchy

In the level situated on the top of the pyramid (L0), we have a small number of CPU registers, with low access time, because they are accessed by the CPU in one clock cycle.

In the next, one or two levels, there is a medium size SRAM cache, which can be accessed in a few CPU clock cycles. On the next level there is a DRAM main memory, with large storage capacity. This memory can be accessed in tens or hundreds of clock cycles. Below are local disks, with very large dimensions and with the disadvantage that they are very slow. On the last level, some systems include an additional level disks or remote servers, which can be accessed via a network.

A possible solution to increase the reliability of the cache level is the use of fault tolerant approach in the design. Traditionally this is realized by the introduction of information redundancy based on data coding.

The widely used code for fault tolerant design is the Hamming code, based on multiple parity bit generation. We present and implement a more efficient design, with a modified version of this code. The resulted design was implemented and tested in an FPGA circuit.

II. APPLYING HSIAO CODE TO CACHE MEMORY

In modern computer systems, at the cache level of the memory hierarchy, we can successfully apply multiple error correction codes. This type of code for detection and correction of errors are added to memories to obtain a better reliability.

In high speed memories the most used codes are Single bit Error Correcting and Double bit Error Detection codes (SEC-DED) [7]. This codes can be implemented in parallel as linear codes for this type of memories.

We have chosen the Hsiao code, because of its properties. Hsiao code is a SEC-DED code preferred in computer technique due to its favourable recovery capacilty from multiple errors. The Hsiao code is a modified Hamming code, with an odd-weight-column, because every column contains an odd number of 1's [4],[9].

In the check matrix of the Hsiao code, another property is that no two columns are the same [8],[9],[10].

For the cache memory we use a (22,16,6), Hsiao code . For this code there are k = 6 control bits, u = 16 useful (data) bits and the total number of code bits is t = 22. In this case, for correcting a single bit error it is satisfied the condition $2^k > u+k+1$. Usually it is enough a number of k= 5 control bits, but we will use k = 6 control bits, in order to achieve a double bit error detection. Parity check matrix of the Hsiao code, is defined by matrix H presented below:

$$H = \begin{bmatrix} 1&0&0&0&0&0&1&1&1&1&0&1&1&0&0&0&1&1&1&1&1&0 \\ 0&1&0&0&0&0&0&0&0&1&1&1&1&1&0&0&1&1&1&1&0&1 \\ 0&0&1&0&0&0&1&0&0&0&1&0&1&1&1&0&1&1&1&0&1&1 \\ 0&0&0&1&0&0&1&1&0&0&0&1&0&1&1&1&1&1&0&1&1&1 \\ 0&0&0&0&1&0&0&1&1&0&0&0&0&0&1&1&1&0&1&1&1&1 \\ 0&0&0&0&0&1&0&0&1&1&1&0&0&0&0&1&0&1&1&1&1&1 \end{bmatrix} \quad (1)$$

We have generated the Hsiao matrix, so that the column vectors corresponding to useful information bits to be different one from other.

A typical codeword, of this matrix has the folowing form:

$$u=(c_0c_1c_2c_3c_4c_5u_0u_1u_2u_3u_4u_5u_6u_7u_8u_9u_{10}u_{11}u_{12}u_{13}u_{14}u_{15})$$

It has parities in positions 1,2,3,4,5, 6 and data bits from position 7 to 22.

The control bits are calculated with parity equations (2) **:**

$$\begin{aligned} c_0 &= u_0\oplus u_1\oplus u_2\oplus u_3\oplus u_5\oplus u_6\oplus u_{10}\oplus u_{11}\oplus u_{12}\oplus u_{13}\oplus u_{14} \\ c_1 &= u_3\oplus u_4\oplus u_5\oplus u_6\oplus u_7\oplus u_{10}\oplus u_{11}\oplus u_{12}\oplus u_{13}\oplus u_{15} \\ c_2 &= u_0\oplus u_4\oplus u_6\oplus u_7\oplus u_8\oplus u_{10}\oplus u_{11}\oplus u_{12}\oplus u_{14}\oplus u_{15} \\ c_3 &= u_0\oplus u_1\oplus u_5\oplus u_7\oplus u_8\oplus u_9\oplus u_{10}\oplus u_{11}\oplus u_{13}\oplus u_{14}\oplus u_{15} \quad (2) \\ c_4 &= u_1\oplus u_2\oplus u_8\oplus u_9\oplus u_{10}\oplus u_{12}\oplus u_{13}\oplus u_{14}\oplus u_{15} \\ c_5 &= u_2\oplus u_3\oplus u_4\oplus u_9\oplus u_{11}\oplus u_{12}\oplus u_{13}\oplus u_{14}\oplus u_{15} \end{aligned}$$

Decoding of a received vector uses the syndrome equations (3) :

$$\begin{aligned} s_0 &= c_0\oplus u_0\oplus u_1\oplus u_2\oplus u_3\oplus u_5\oplus u_6\oplus u_{10}\oplus u_{11}\oplus u_{12}\oplus u_{13}\oplus u_{14} \\ s_1 &= c_1\oplus u_3\oplus u_4\oplus u_5\oplus u_6\oplus u_7\oplus u_{10}\oplus u_{11}\oplus u_{12}\oplus u_{13}\oplus u_{15} \\ s_2 &= c_2\oplus u_0\oplus u_4\oplus u_6\oplus u_7\oplus u_8\oplus u_{10}\oplus u_{11}\oplus u_{12}\oplus u_{14}\oplus u_{15} \quad (3) \\ s_3 &= c_3\oplus u_0\oplus u_1\oplus u_5\oplus u_7\oplus u_8\oplus u_9\oplus u_{10}\oplus u_{11}\oplus u_{13}\oplus u_{14}\oplus u_{15} \\ s_4 &= c4\oplus u_1\oplus u_2\oplus u_8\oplus u_9\oplus u_{10}\oplus u_{12}\oplus u_{13}\oplus u_{14}\oplus u_{15} \\ s_5 &= c_5\oplus u_2\oplus u_3\oplus u_4\oplus u_9\oplus u_{11}\oplus u_{12}\oplus u_{13}\oplus u_{14}\oplus u_{15} \end{aligned}$$

We will apply this SEC-DED code to the design of the error control part of a 64K x 16 bit cache memory. When the information is retrieved from the cache, we read the useful data bits ($u_0\, u_1\, u_2\, u_3\, u_4\, u_5\, u_6\, u_7\, u_8\, u_9\, u_{10}\, u_{11}\, u_{12}\, u_{13}\, u_{14}\, u_{15}$) and the control bits ($c_0\, c_1\, c_2\, c_3\, c_4\, c_5$) too. We implement with XOR gates, the equations (2) and generate the control bits c_0' c_1' c_2' c_3' c_4' c_5' from data bits that we have read from the cache. For example, to generate the control bit c_0', we use equation (4):

$$c_0' = u_0\oplus u_1\oplus u_2\oplus u_3\oplus u_5\oplus u_6\oplus u_{10}\oplus u_{11}\oplus u_{12}\oplus u_{13}\oplus u_{14}, \quad (4)$$

In order to implement this equation we use 10 XOR gates with two inputs, situated on four levels, as presented in figure 2.

We do in the same mode to generate all control bits, c_1', c_2', c_3', c_4', c_5'. The generated control bits ($c_0'c_1'c_2'c_3'c_4'c_5'$) are compared with the control bits that we have read from the cache ($c_0\, c_1\, c_2\, c_3\, c_4\, c_5$), also with two input XOR gates, and we get as result syndrome equations: $s_0 = c_0 \oplus c_0'$, $s_1 = c_1 \oplus c_1'$, $s_2 = c_2 \oplus c_2'$, $s_3 = c_3 \oplus c_3'$, $s_4 = c_4 \oplus c_4'$, $s_5 = c_5 \oplus c_5'$. We connect one NOT gate on each syndrome line, and we construct with 16 AND gates with six inputs, the syndrome decoder. Equations (5) are used to build the syndrome decoder.

$$\begin{aligned} u_0' &= s_0 \cdot \overline{s_1} \cdot s_2 \cdot s_3 \cdot \overline{s_4} \cdot \overline{s_5} \\ u_1' &= s_0 \cdot \overline{s_1} \cdot \overline{s_2} \cdot s_3 \cdot s_4 \cdot \overline{s_5} \end{aligned}$$

$$
\begin{aligned}
u'_2 &= s_0 \cdot \overline{s_1} \cdot \overline{s_2} \cdot \overline{s_3} \cdot s_4 \cdot s_5 \\
u'_3 &= s_0 \cdot s_1 \cdot \overline{s_2} \cdot \overline{s_3} \cdot \overline{s_4} \cdot s_5 \\
u'_4 &= \overline{s_0} \cdot s_1 \cdot s_2 \cdot \overline{s_3} \cdot \overline{s_4} \cdot s_5 \\
u'_5 &= s_0 \cdot s_1 \cdot \overline{s_2} \cdot s_3 \cdot \overline{s_4} \cdot \overline{s_5} \\
u'_6 &= s_0 \cdot s_1 \cdot s_2 \cdot \overline{s_3} \cdot \overline{s_4} \cdot \overline{s_5} \\
u'_7 &= \overline{s_0} \cdot s_1 \cdot s_2 \cdot s_3 \cdot \overline{s_4} \cdot \overline{s_5} \\
u'_8 &= \overline{s_0} \cdot \overline{s_1} \cdot s_2 \cdot s_3 \cdot s_4 \cdot \overline{s_5} \\
u'_9 &= \overline{s_0} \cdot \overline{s_1} \cdot \overline{s_2} \cdot s_3 \cdot s_4 \cdot s_5 \\
u'_{10} &= s_0 \cdot s_1 \cdot s_2 \cdot s_3 \cdot s_4 \cdot \overline{s_5} \\
u'_{11} &= s_0 \cdot s_1 \cdot s_2 \cdot s_3 \cdot \overline{s_4} \cdot s_5 \\
u'_{12} &= s_0 \cdot s_1 \cdot s_2 \cdot \overline{s_3} \cdot s_4 \cdot s_5 \\
u'_{13} &= s_0 \cdot s_1 \cdot \overline{s_2} \cdot s_3 \cdot s_4 \cdot s_5 \\
u'_{14} &= s_0 \cdot \overline{s_1} \cdot s_2 \cdot s_3 \cdot s_4 \cdot s_5 \\
u'_{15} &= \overline{s_0} \cdot s_1 \cdot s_2 \cdot s_3 \cdot s_4 \cdot s_5
\end{aligned}
\qquad (5)
$$

We present in figure 2 a scheme used for single error correction.

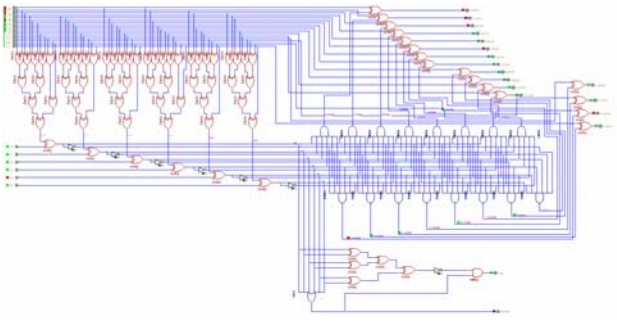
Figure 2. Design of Hsiao (22,16,6) code circuit with XILINX software tool

To correct the data bits we use 16 XOR gates with two inputs, following equations (6).

$$
\begin{aligned}
u_{0cor} &= u_0 \oplus u_0' \\
u_{1cor} &= u_1 \oplus u_1' \\
u_{2cor} &= u_2 \oplus u_2' \\
u_{3cor} &= u_3 \oplus u_3' \\
u_{4cor} &= u_4 \oplus u_4' \\
u_{5cor} &= u_5 \oplus u_5' \\
u_{6cor} &= u_6 \oplus u_6' \\
u_{7cor} &= u_7 \oplus u_7' \\
u_{8cor} &= u_8 \oplus u_8' \\
u_{9cor} &= u_9 \oplus u_9' \\
u_{10cor} &= u_{10} \oplus u_{10}' \\
u_{11cor} &= u_{11} \oplus u_{11}' \\
u_{10cor} &= u_{10} \oplus u_{10}' \\
u_{11cor} &= u_{11} \oplus u_{11}' \\
u_{12cor} &= u_{12} \oplus u_{12}' \\
u_{13cor} &= u_{13} \oplus u_{13}', \\
u_{14cor} &= u_{14} \oplus u_{14}' \\
u_{15cor} &= u_{15} \oplus u_{15}'
\end{aligned}
\qquad (6)
$$

In figure 2 we have designed an error correction scheme based on Hsiao (22,16,6) code, which can be implemented with FPGA Xilinx circuits.

III. IMPLEMENTATION OF HSIAO CODE TO THE CACHE MEMORY.USING FPGA XILINX CIRCUITS

The design process with FPGA Xilinx circuits is fast and efficient. The internal structure of an FPGA circuit contains a matrix composed from Configurable Logic Blocks (CLB) and Programable Switch Matrices (PSM), surrounded by I/O pins.

The programable internal structure includes two configurable elements: Configurable Logic Blocks, with functional elements that implements the designed logical structure and Input Output Blocks (IOB), wich realises the interface between internal signals and the outside of circuit, using pins.

The logical function realised by CLB is implemented by static configuration memory [7].

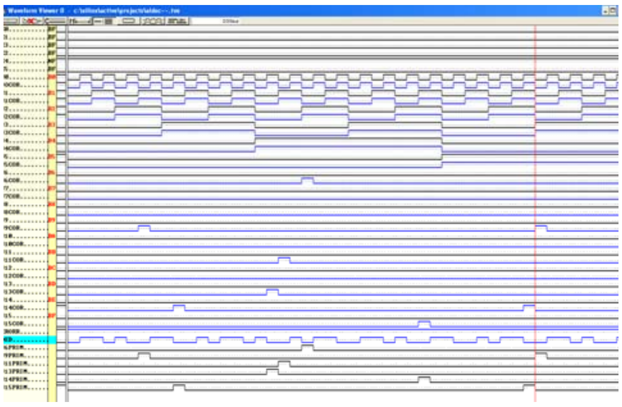
Figure 3. Simulation with Logic Simulator of XILINX

Figure 3 shows the simulation that we have made through the Logic Simulator module of XILINX program.

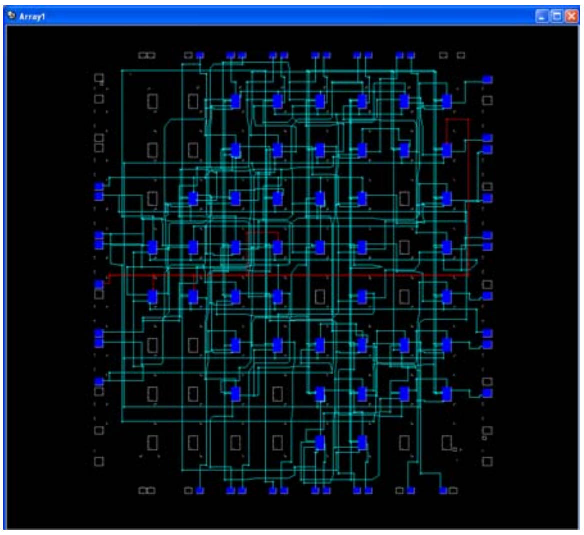

Figure 4. Implementation of Hsiao matrix (1) with FPGA XILINX, XC4000XL circuits

To follow up the simulation we have introduced: input signals (u_0-u_{15}), control signals (c_0-c_5), output signals (u_{0cor}-u_{15cor}) and signals ERORR and DED (double error detected). We have injected errors and checked the bahaviour of the design.

Figure 4 presents the implementation of HSIAO matrix (1), with FPGA XILINX, XC4000XL circuits. Analyzing the file Map Report we can conclude that only 44 CLB circuits have been used from a total of 64, meaning 68% of the total CLB circuits.

IV. CONCLUSIONS

We have subsequently applied this Hsiao code, to error detection and correction in the cache and we have implemented this code in a cache memory using FPGA programmable Xilinx circuits. We have determined the overhead due the additional circuits for error correction. This Hsiao code has the minimum number of 1's in the matrix, which makes the hardware and the speed of the encoding/decoding circuit optimal. The Hsiao code (22,16,6) that we have used to the cache level of a memory hierarchy permits single error correction and double error detection. This code was implemented to a cache memory, with this implementation we have reduced the size of the syndrome generator and the cost of error correcting scheme compared to the traditional Hamming code based solution. Another advantage is that if we increase the number of data bits, the proportion of overhead is decreasing. This solution using a SEC-DED Hsiao code, increases reliability through fault tolerance, leading to low cost and low memory chip dimension, because this method solves the problem of faults by testing and correcting errors inside the chip.

Results of this research were supported by Domus Hungarica.

REFERENCES

[1] John L. Hennessy, David A. Patterson, "Computer Architecture. A Quantitative Approach", Morgan Kaufmann Publishers, Inc. 1990-1996.

[2] John L. Hennessy, David. A. Patterson, "Computer Arhitecture. A Quantitative Approach", 3rd Edition, San Mateo, CA, Morgan-Kaufmann Publishing Co., 2003.

[3] J. Chang, Şt. Rusu, J. Shoemaker, S. Tam, M. Huang, M. Haque, et al., "A 130-nm Trimple-Vt 9-MB Third-Level On-Die Cache for the 1.7-GHz Itanium 2 Processor", Journal on Solid State Circuits, vol. 40, no. 1, pp. 195 – 203, 2006.

[4] T.R.N. Rao, E. Fujiwara,"Error-Control Coding for Computer Systems ", Prentice Hall International Inc., Englewood Cliffs, New Jersey, USA, 1989.

[5] L. D. Hung, "Soft Error Tolerant Cache Architectures", PhD Thesis, Department of Information Science and Technology, University of Tokyo, December 2006.

[6] H. R. Zarandi, S. G. Miremadi, "A Highly Fault Detectable Cache Architecture for Dependable Computing", M. Heisel et al. (Eds.), SAFECOMP 2004, LNCS 3219, pp. 45 – 59, 2004.

[7] Ovidiu Novac, "Cercetări ale eficienţei metodelor de creştere a dependabilităţii la treapta cache a unei ierarhii de memorii", PhD Thesis, ISBN: 978-973-625-593-9, Editura Politehnica, Timişoara, 2008.

[8] P. L. Howard, "The Design Book: Techniques and Solutions for Digital Computer Systems", Prentice-Hall Inc., Englewood Cliffs, N. J. 1990.

[9] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing", IEEE Transactions on Dependable and Secure Computing, Vol.1, No.1, pp 11 - 33, January-March 2004.

[10] W. Huffman, V. Pless, Fundamentals of error-correcting codes, Cambridge University Press, ISBN 9780521782807, 2003.