Exploiting Fault Tolerance within Cache Memory Structures

Somak Das
Dept. of Computer Science & Engineering
University Institute of Technology,
The University of Burdwan
Burdwan, India
somakdas2@gmail.com

Sowvik Dey
Dept. of Computer Science & Technology
Regent Institute of Science & Technology
Barrackpur, India
sowvikdey@gmail.com

***Abstract*— Cache memories can work as buffer between processors and main memories. It enables rapid access of data for a processor in operation. Set-associativity provides optimality in mapping of cache memories and reduction of cache miss probability. Design of a high speed cache has always been a desirable criteria of hardware experts as it increases processor utilization. Exploiting fault tolerance within such a cache memory of higher throughput ensures reliable data transfer and is an open research problem in the domain of high-performance computing. This paper proposes a design of low-order interleaved set-associative cache memory with lesser response time and exploits a high degree of fault tolerance.**

Keywords*— *Cache memory; set-associativity; processor utilization; interleaved memory; fault tolerance; fine grain; field programmable gate array.

I. INTRODUCTION

Design of high speed computers has always remained a research challenge for computer engineers. High end gigahertz ranged processors have been successfully designed and marketed since last decade. The performance of a computer is necessarily dependent on the relatively slower storages its processor has to work with. Bulk amount of data gets transferred between the processor and the relatively slower hierarchy of memory. This causes delay in the overall response time of the processor. It is known as Von-Neumann bottleneck.

Several design steps were taken to eliminate the speed gap between processor and memory. Cache memory was introduced as a high speed intermediate storage between the primary memory and the processor [1]. Cache memories are widely used in von-neumann machines which has simple low speed scalar processors [1],[2] but became unimpressive in harvard computers that have high speed vector processors. Changes in storage technology to design high speed memory devices, increase of memory wordlength and width of databus [3] etc. can be used as alternative ways for increasing throughputs of memory. Again, these design changes had their own limitations due to the predictable saturation of moore's law and chances of inappropriate power dissipation in low power VLSI circuits. The most formidable alternative to these approaches is an interleaved storage system. Here the memory is segmented into equi-banks of storage modules which are connected in an interleaved fashion [4],[5]. Detailed study, analysis and verification of structure and speed of interleaved memories have been done in many previous research articles [6],[7],[8],[9]. The speed of interleaved memories depends on how the memory modules are addressed [8]. Two types of memory interleaving methods exists. In a high-order interleaved structure the most significant address bus lines are used to select banks of memory. Whereas, in low-order interleaved structure bank selection is done by least significant address bits. Data storing methods for interleaved memories adjacent to custom processors have already been surveyed [10]. Detailed analysis and study on the power and thermal efficiency of interleaved memories were also done [11].

Low and high-order interleaved memories have their own share of limitations. High-order interleaved memory is slower and thus, lower in bandwidth. Whereas, low-order interleaved memories are non-modular by nature, i.e. a fault in any of the banks adversely effects the whole address space [12]. Field programmable gate array (FPGA) based memory design are can be faulty and there exists several corrective methods of these faults [13]. FPGA based design of high-order interleaved fault tolerant memory are already established [14]. Efficient low-order interleaved fault tolerant memories has also been realised [15]. FPGA based grid computation methods [16] use fault-intolerant low-order interleaved memories.

In this paper, a fault tolerant design of low-order interleaved cache memory is proposed and the structure has been implemented on FPGA. One or more faulty cache lines can easily be bypassed in the reconfigurable platform with reallocation of the address space among the fault free cache lines. With such a line level fault tolerant scheme which is proposed in this paper, only the faulty cache lines are removed from the address space rather than a cache set as a whole. This new method also includes the speed advantage of low-order interleaving and brings in fault tolerance property within the cache memory by insignificantly reducing the addressable cache memory space. The proposed cache memory structure is not only unique but also advantageous over the already existing cache architectures.

II. ARCHITECTURE OF SET ASSOCIATIVE CACHE

Figure.1 gives an example of a 4-way set-associative cache memory with 8 lines being mapped with a main memory with 4

978-1-4799-5958-7/14/$31.00 ©2014 IEEE

blocks. Each of the blocks of main memory contains 4 words of data. These words can be reside in the cache memory in such a style that words from block#0, block#2 can only be loaded in set#0, block#1 can only be loaded in set#1 of the cache. As a matter of fact, values 1,5 of block#0 gets loaded into the first two lines of set#0 in cache and values 3,7 of block#2 gets loaded into the last two lines of set#0 in cache. Similarly, values 2,6 of block#1 gets loaded into the first two lines of set#1 in cache and values 4,8 of block#3 gets loaded into the last two lines of set#1 in cache. So, it is evident that all words of a block of main memory cannot reside with a cache set at the same time and depending on the requirement of the processor these leftout words can be loaded into cache.

Figure.2 describes the post cache replacement senario where words valued 9 and 15 from block#0 and block#2 respectively gets loaded into the first two lines of set#0 based on first-in-first-out (FIFO) cache replacement scheme. Similarly, words valued 10 and 12 from block#1 and block#3 respectively gets loaded into the first two lines of set#1 based on FIFO cache replacement scheme.

III. PROPOSED FAULT TOLERANT INTERLEAVED CACHE

A. High-Order Interleaved Fault Tolerant Cache

A 4-way set associative cache memory as shown in fig. 3 can be interleaved in a high-order fashion to make it fault tolerant. 8-individual lines of the cache can be locally addresses using 3 address lines. Here, the cache is divided into 2-sets, two least significant address lines (A_1A_0) can be used to point lines within a set. Whereas, the most significant address line A_2 is used as a select line of a multiplexer which lets the output data lines from the cache-sets pass through it. Here, if any faulty condition occur in lines of set#0 then this set can be removed from the cache structure simply by changing the bit value of A_2. But, high-order interleaved cache is slower and the speed advantage of cache memory is lost.

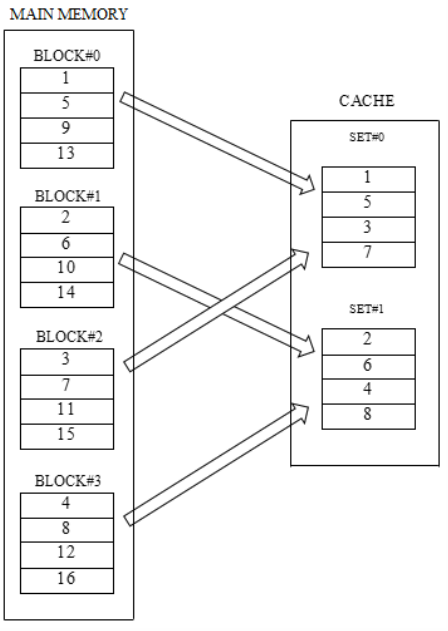

Fig. 1. 4-way Set-Associative Cache with 8 lines

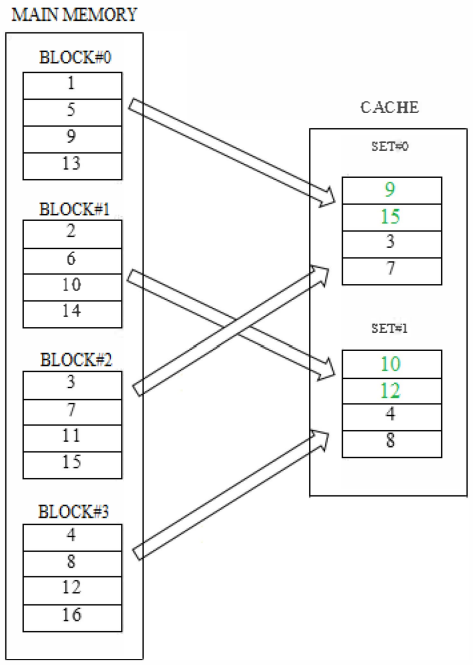

Fig. 2. Cache Replacement within 4-way Set-Associative Cache

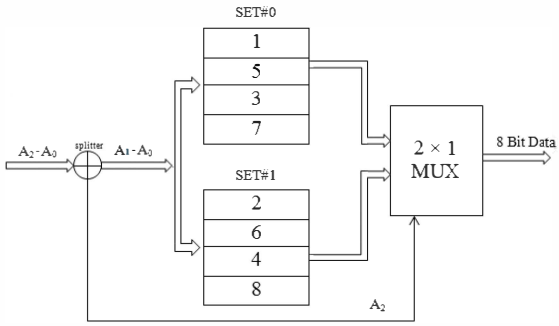

Fig. 3. High-Order Interleaved 4-way Set-Associative Cache

B. Low-Order Interleaved Fault Tolerant Cache

If the cache memory is low-order interleaved then it turns out to be a high speed one. But, low-order interleaved structures are not modular in nature and therefore, inherently fault-intolerant. Fig. 4 gives an idea of low-order interleaved cache structure. The cache is divided into 2-sets, two most significant address lines (A_2A_1) can be used to point lines within a set. Whereas, the least significant address line A_0 is used as a select line of a multiplexer which lets the output data lines from the cache-sets pass through it. Here, if any faulty condition occur in lines of set#0 then this set cannot be removed from the cache structure by changing the bit value of A_0 as it would jeoperdise the entire cache addressing. Therefore, a fine grained approach is needed to remove the faulty line(s) from the cache address space without disturbing the contiguity and set-associativity of the cache memory. Fig.5 and fig.6 depicts a clear picture of the proposed structure incase of single and multiple faults respectively within this low-order

interleaved cache structure. In fig.5 as fault lies in line with address 100, this cache line is bypassed and the line addresses are reallocated in such a pattern that it avoids any gap in address space. So, address 100 now points to the next non-faulty cache line which was previously addresses by 101, address 101 points to the next cache line (previous address 110) and so on. The last cache line address points to the faulty cache line and thus address 111 holds a bit sequence in high impedence state (ZZZZZZZZ).

In fig.6 cache lines with initial addresses of 010 and 101 are supposed to be faulty. Bypassing these cache lines and reallocation of the cache address space replaces old addresses of 011 with 010, 100 with 011, 110 with 100, 111 with 101. Faulty cache lines with initial address of 010 and 101 now hold addresses 110 and 111 and both hold bit sequences in high impedence states. Figure.7 shows the same cache lines to be faulty and the reallocation of the cache address space is also done in the same fashion as described above. The only difference with that of fig. 6 is the cache line values. Here (in fig.7) the values stored in cache lines reflect a post-replacement senario. The values 9, 11, 5 and 10, 4, 6 entered set#0 and set#1 respectively from the designated blocks of main memory after cache replacement.

Here, the fault avoidance scheme does not hamper the low-order interleaving structure, retains the speed advantage and prevents wastage of valuable non-faulty cache lines.

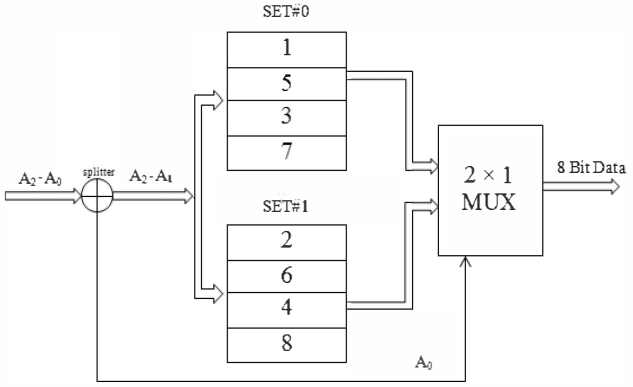

Fig. 4. Low-Order Interleaved 4-way Set-Associative Cache

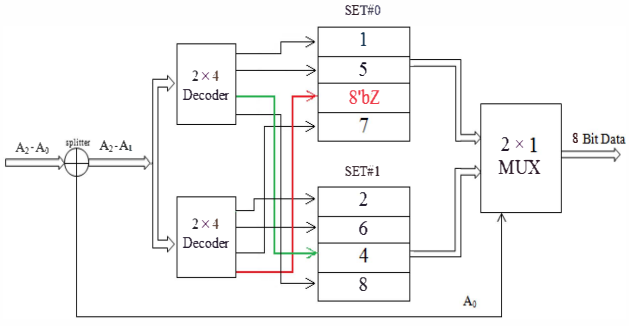

Fig. 5. Single line fault in a low-order interleaved cache

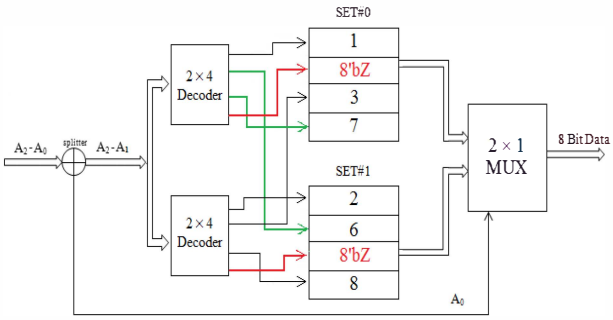

Fig. 6. Multiple line faults in a low-order interleaved cache

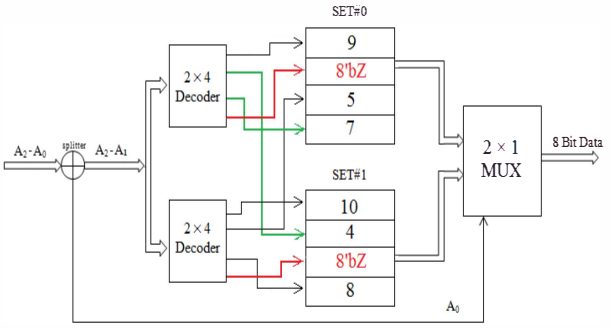

Fig. 7. Multiple line faults in a low-order interleaved cache after cache replacements

C. *Algorithms for Cache Address Space Reallocation*

Algorithm for the FIFO replacement and cache address space reallocation in occurrence of a single cache line fault is shown in fig. 8.

```
L1: SET COUNT<=0;
L2: IF data is not in cache
      THEN
        (CACHE_ADDR_BASE+COUNT) <= MEM (CONTENT);
        COUNT<=COUNT+1;
        IF (CACHE_ADDR_BASE+COUNT) is CACHE_ADDR_LAST
              THEN GOTO L1;
        ELSE GOTO L2;
    ELSE
        IF fault location have address CACHE_ADDR,
              THEN (CACHE_ADDR) <= CACHE_ADDR + 1;
                   (CACHE_ADDR+1) <= CACHE_ADDR + 2;
              REPEAT TILL
                   (CACHE_ADDR_Last-1) <= CACHE_ADDR_Last;
                   Content (CACHE_ADDR_Last) <= 'ZZZZZZZZ';
```

Fig.8. Algorithm of replacement and cache address space reallocation for single cache line fault

The algorithm for the FIFO replacement and cache address space reallocation in occurrence of multiple cache line faults is shown in fig. 9.

```
L1: SET COUNT1<=0;
L2: IF data is not in cache
      THEN
         (CACHE_ADDR_BASE+COUNT1) <= MEM (CONTENT);
         COUNT1<=COUNT1+1;
          IF (CACHE_ADDR_BASE+COUNT1) is
             CACHE_ADDR_LAST
             THEN GOTO L1;
          ELSE GOTO L2;
   ELSE
      SET COUNT2<=0;
      L3: IF fault location is (CACHE_ADDR+COUNT2)
          THEN (COUNT2) <=COUNT2+1;
                CACHE_ADDR<=CACHE_ADDR+COUNT2;
                GOTO L3;
          ELSE
                (CACHE_ADDR)<=CACHE_ADDR+COUNT2;
      L4: IF fault location is (CACHE_ADDR+COUNT2+1)
          THEN (COUNT2) <= COUNT2+1;
                (CACHE_ADDR+1)<=CACHE_ADDR+COUNT2+1;
                GOTO L4;
          ELSE
            (CACHE_ADDR+1)<=CACHE_ADDR+COUNT2+1;
            REPEAT TILL
          (CACHE_ADDR_Last-COUNT2) <= CACHE_ADDR_LAST;

          Content (CACHE_ADDR_Last-COUNT2) <= 'ZZZZZZZZ';
        Content (CACHE_ADDR_Last-COUNT2+1) < = 'ZZZZZZZZ';
          REPEAT TILL
          Content (CACHE_ADDR Last) <= 'ZZZZZZZZ';
```

Fig.9. Algorithm of replacement and cache address space reallocation for multiple cache line faults

IV. SIMULATION RESULTS OF FAULTS OCCURING WITHIN THE PROPOSED LOW-ORDER INTERLEAVED CACHE

Figure. 10 gives the simulation result for occurance of faults within the proposed fault tolerant low-order interleaved cache as shown in fig.7. Contents of the main memory blocks are shown by 'memout'. In every positive edge of clock pulse 'cp', the content of cache line for given 'address' is shown in 'c00'. Signal 'f_on' is used to inject the faulty lines into the system. The system is designed as it can bypass the data of the faulty lines and display the data of next cache line and so on. Here, fault is injected in the 2nd line containing value 11 and 5th line containing value 6. The simulation bypasses this physical lines from the cache set system and allocates its address to the next physical cache line. So, the simulation result shows the value 4 in the 2nd line in place of 11 and value 7 in the 5th line in place of 6. It clearly shows a valuatic left shift of all cache lines succeeding the faulty lines allocating high impedence values for the last two cache lines.

V. DESIGN IMPLEMENTATION ON XILINX FPGA

The design of the proposed low-order interleaved fault tolerant cache has been implemented on Xilinx SpartanII FPGA (device: xc2s200-5pq208). Figure.11 shows the technology schematic of the fault tolerant cache and green and yellow colored regions of fig.12 gives the positioning of the design within Xilinx SpartanII FPGA (xc2s200-5pq208). There has been 9% utilization in the number of slices and 4-input LUTs.

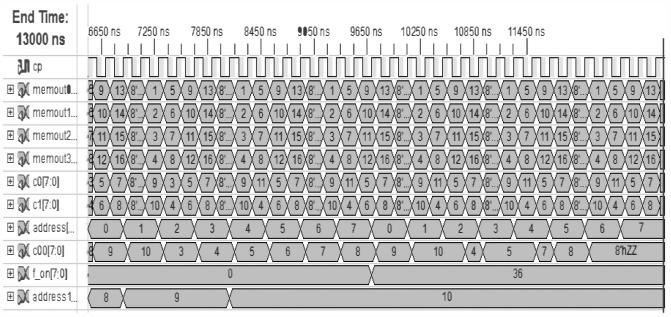

Fig.10. Simulation result of address reallocation for line faults within low-order interleaved cache structure [Xilinx ISE8.2i simulator]

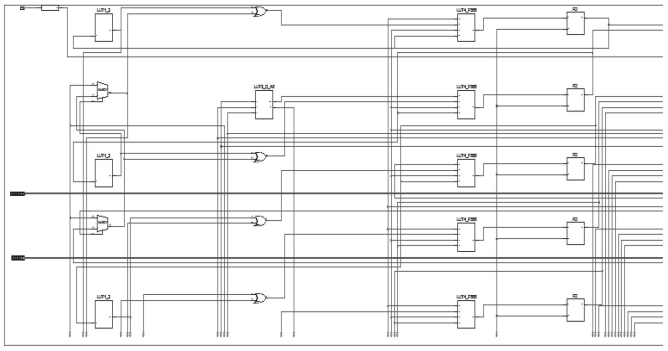

Fig.11 Technology Schematic for the fault tolerant low-order interleaved cache design in SpartanII FPGA xc2s200-5pq208

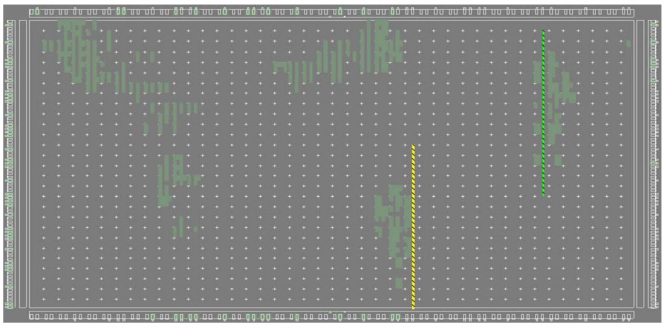

Fig.12 Placement of the fault tolerant low-order interleaved cache design in SpartanII FPGA xc2s200-5pq208

The number of bonded IOBs has also been utilized 50% and time delay for mapping the total data path is in approximately 13ns.

VI. EXPERIMENTAL CASE STUDY

An application of this low-order interleaved fault tolerant cache is in conjunction with an image processor. The proposed cache structure can be used for faster and reliable retrieval of image data. Faults in one or more cache lines do not pose any problem in regeneration of the stored image as the cache memory is highly fault tolerant. Fault correction mechanisms have also been exploited so that the regenerated image resembles the original one. Again the access rate of the newly

designed low-order cache is much higher than a non-interleaved or high-order interleaved architecture. An experimental case study has been performed to verify the efficiency of this fault tolerant cache taking example of real image data.

An image is nothing but a 2-dimensional matrix having pixel values stored in X and Y co-ordinates. If there are four memory blocks within a main memory structure, then the pixel values corresponding to X and Y co-ordinates can be stored in block#0 (if X_{LSB}=even, Y_{LSB}=even), block#1 (if X_{LSB}=even, Y_{LSB}=odd), block#2 (if X_{LSB}=odd, Y_{LSB}=even) or block#3 (if X_{LSB}=odd, Y_{LSB}=odd). Considering the image of fig.13, its pixel information is shown in fig.14. Suppose a block of pixel information is selected (fig.15), divided into four sub blocks (fig.16) and pixel values from each sub blocks are stored in different blocks of main memory in the above stated style. Let this pixel data from the main memory be loaded (as in fig.17 & fig.18) into the cache and then only be accessed by image processor. Now, if the cache develops faults in some of its lines (say the last line of set#0 and first line of set#1) then the image information stored within those faulty lines gets lost (fig.19). Trying to regenerate the image from the stored pixel data would have given us a coarse grained approximated image as fig.20 if the cache is not a fault tolerant one. But, if the pixel data (fig.14) of the original image is loaded from the main memory to the proposed fault tolerant low-order interleaved cache (fig.21 and fig.22), then the regenerated image (fig.23) is visually identical to the original image as the pixel information loss is taken care of and most of the desired pixel information (fig.24) can be retrieved.

Fig.13 Original Image

126	132	127	117
123	129	134	117
137	107	124	155
116	113	148	162
107	128	156	140
117	143	137	112
141	139	110	110
152	115	97	135
104	154	152	101
109	130	158	99

90	88	78	74
91	89	83	77
157	137	117	98
153	130	110	93
143	123	104	84
132	111	93	83
84	76	69	73
87	74	59	73

76	71	66	65
83	71	65	63
52	53	55	52
55	57	57	53
60	59	59	55
67	64	59	55
71	66	67	63
76	70	68	64
39	37	36	36
37	37	36	37

Fig.14 Pixel data of original image

157	137	117	98
153	130	110	93
143	123	104	84
132	111	93	83

Fig.15 Selected block of pixels

157	137
153	130

117	98
110	93

143	123
132	111

104	84
93	83

Fig.16 Four sub blocks from the selected block (fig. 15) of pixels

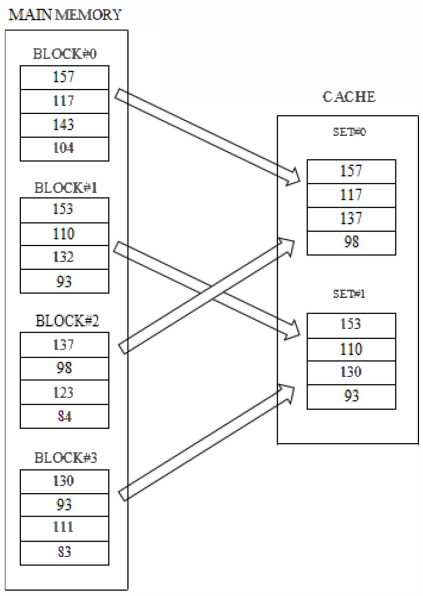

Fig.17 initial loading of pixel data from main memory to cache-sets

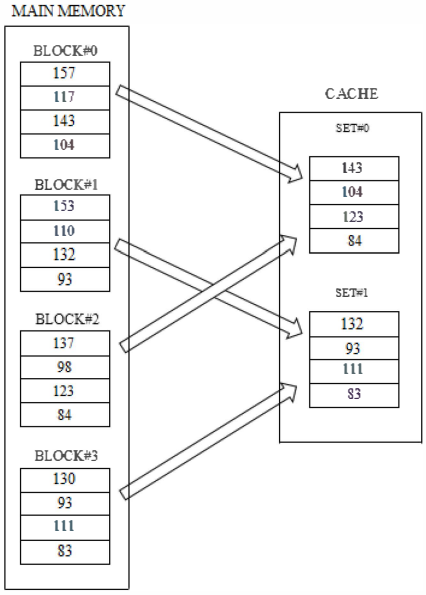

Fig.17 subsequent loading or replacement of pixel data from main memory to cache-sets

126	132	127	117
123	129	134	117
137	107	124	155
116	113	148	162
107	128	156	140
117	143	137	112
104	154	152	101
109	130	158	99

90	88	78	74
91	89	83	77
157	137	117	8b'Z
8b'Z	130	110	93
143	123	104	8b'Z
8b'Z	111	93	83
84	76	69	73
87	74	59	73

76	71	66	65
83	71	65	63
52	53	55	52
55	57	57	53
60	59	59	55
67	64	59	55
39	37	36	36
37	37	36	37

Fig.19 Pixel data retrieved from faulty memory

Fig.20 Faulty image

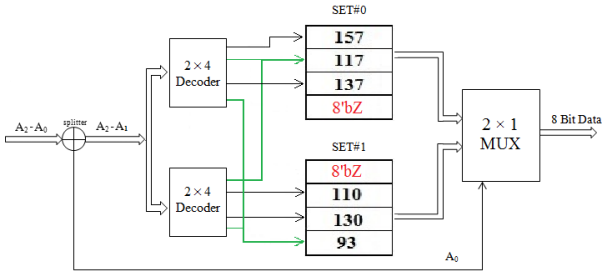

Fig. 21 the first phase of fault tolerance of the two caches sets

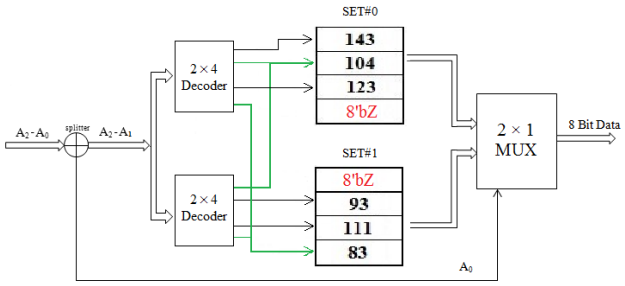

Fig. 22 the second phase of fault tolerance of the two caches sets

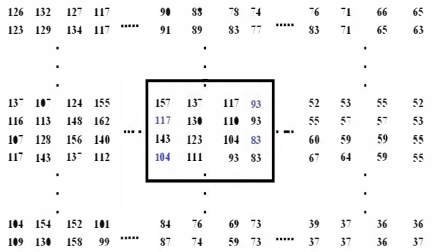

Fig.23 Regenerated image Fig.24 data from proposed cache (fig.21 & fig.22)

VII. CONCLUSIONS

In this paper, fault tolerance has been incorporated within a low-order interleaved cache with a set-associative structure. The design of the proposed cache is not only fault tolerant but also of high data access rate in comparison to existing cache architectures. Here lies the novelty of the new design. This newly designed cache memory is experimentally found to be useful in adjacency of image processors. Again in FPGA based grid computations [16] where grid points are to be stored in memory banks and to be retrieved by the processor via a cache this proposed cache architecture can really prove beneficial for high speed reliable grid point access. Fine grained cache-line level fault tolerance demonstrated in this work also reduces the wastage of valuable storage space of cache. Implementing this fault tolerant cache architecture for more advanced cache replacement schemes remains as future scope of research.

REFERENCES

[1] "Cache memories", ACM Comput. Surveys, vol. 14, pp. 473-530, 1982.

[2] J. E. Smith and J. R. Goodman, "A study of instruction cache organizations and replacement policies", Proc. 10th Annu. Symp. Comput. Archifecture, pp. 117-123, 1983.

[3] Nikitas A Alexandridis, "Design of Microprocessors".

[4] K. Hwang and F. A. Briggs, "Computer Architecture and Parallel Processing", McGraw-Hill, New York, 1984.

[5] P. M. Kogge, "The Architecture of Pipelined Computers", McGraw-Hill, New York, 1981.

[6] D.P.Bhandarkar, "Analysis of memory interference in multiprocessors", IEEE Trans. Comput., vol. C-24, pp.897-908, 1975.

[7] G. Burnett and E.G. Coffman, "A Study of interleaved memory systems", proc. AFIPS 1970 Spring Joint Comput.Conf.,pp.467-474, 1970.

[8] B.R.Rau, "Program behavior and performance of interleaved memories", IEEE Trans. Comput., vol. C-28, pp.191-199, 1979.

[9] F.W.Terman, "A study of interleaved memory systems by trace driven simulation", Proc. Symp. Simulation Comput. Syst., pp. 3-9, 1976.

[10] A.Postula, S.Chen, L.Jozwiak and D.Abramson, "Automated Synthesis of Interleaved Memory Systems for Custom Computing Machines", 1089-6503/98, IEEE, 1998.

[11] M. E. Tolentino, "Managing Memory for Power, Performance and Thermal Efficiency", PhD dissertation, Faculty of Virginia Polytechnic Institute and State University, 2009.

[12] D.K. Pradhan, "Fault Tolerant Computing: theory and Techniques", Englewood Cliffs, NJ: Prentice-Hall, 1986.

[13] Yubo Li, Brent Nelson and Michael Wirthlin, "Reliability Models for SEC/DED Memory with Scrubbing in FPGA-Based Designs", IEEE Trans. On Nuclear Science, vol. 60, No. 4, August 2013.

[14] Somak Das and Sowvik Dey, "FPGA based Design of a Fine-Grained Fault Tolerant Interleaved Memory", Proc. of IEEE Int. Conf. on ACCCT, pp. 565-568, 2014.

[15] Somak Das and Sowvik Dey, "Design of Fault Tolerant Low-Order Interleaved Memory Based on the Concept of Bubble-Stack An Image Storage Perspective", Proc. of ISVDAT, 2014.

[16] Tom VanCourt and Martin Herbordt, "Application specific memory interleaving for FPGA based grid computation : a general design technique", Proc. of IEEE Conf. on FPL, pp. 1-6, 2006.