Design and Implementation of a Memory for Joint Improvement of Error Tolerance and Access Efficiency

Kavya Cheraukula, Chowdam Venkata Sudhakar

Abstract— The on-chip memory becomes increasingly exposed to the dual challenges of device-level reliability degradation and architecture-level performance gap. We propose to exploit the inherent memory soft redundancy for on-chip memory design. Due to the mismatch between fixed cache line size and runtime variations in memory spatial locality, many irrelevant data are fetched into the memory thereby wasting memory spaces. The proposed soft-redundancy allocated memory detects and utilizes these memory spaces for jointly achieving efficient memory access and effective error control. We design an CRC & ECC with error correction techniques by making use of standard Ethernet (004C11DB7H) polynomial and compare with each other, which will be implemented in FPGA proposed system design which take care of the Cache and memory by re-checking the cache when a miss is identified and help in effective functionality of the system and finally we compare which method will give good results in terms of cost and reliability.

Index Terms— cache memory, ECC, soft error, URL.

I. INTRODUCTION

Technology scaling enables a dramatic increase in chip density and speed, which in turn fuels the architecture-level innovations for performance improvement. With this trend to continue, microprocessor design has entered the multibillion-transistor-on-chip era[1],[2]. Accompanied with the growing chip complexity are the new challenges at both device and architecture levels.

Among all on-chip function units, memory is particularly exposed to the emerging issues related to technology scaling. As the performance of microprocessors is still struggling with the memory wall, reliability of memory circuits is also getting worse. In digital integrated systems, the bulk of the logic gates are typically spent on memory (e.g., caches). Memory circuits are built on minimum-geometry devices that are very sensitive to variations in process parameters, supply voltage and temperature [3], [4]. These inherently unreliable memory circuits also suffer from soft errors caused by particle strikes and timing failures [5], [6]. As a result, cache memory has become not only the bottleneck of performance improvement but also a major reliability concern in physical design. Robustness to errors is a key consideration in the light of excessive process variations and external disturbances that affect the reliability.

Manuscript Received June 10, 2012.

CH.kavya, pursuing MTECH,VLSI, ECE Department, Sree vidyanikethan Engineering college, Tirupathi, India.

C.Venkata sudhakar, Assistant professor, ECE Department, Sree vidyanikethan Engineering college, Tirupathi, India.

II. EXISTING SYSTEM

The techniques in improving memory error tolerance include radiation-hardened memory [7], [8], double or triple memory redundancy [9], [10], and code checking logic [11], [12]. These techniques in general incur performance overhead and thus are effective if this overhead is manageable, e.g., in non-critical and off-chip memory with long access latency and out-of-path error correction. Fault/error tolerance has also been addressed at the program and instruction levels [13]. However, these techniques do not target memory circuits specifically.

In computer science, a cache is a component that improves performance by transparently storing data such that future requests for that data can be served faster.

The data that is stored within a cache might be values that have been computed earlier or duplicates of original values that are stored elsewhere. If requested data is contained in the cache (cache hit), this request can be served by simply reading the cache, which is comparably faster. Otherwise (cache misses), the data has to be recomputed or fetched from its original storage location, which is comparably slower. Hence, the more requests can be served from the cache the better the overall system performance is.

As opposed to a buffer, which is managed explicitly by a client, a cache stores data transparently: This means that a client who is requesting data from a system is not aware that the cache exists? Therefore the name cache (from French "cacher", to conceal).

To be cost efficient and to enable an efficient lookup of data, caches are comparably small. Nevertheless, caches have proven extremely effective in many areas of computing because access patterns in typical computer applications have locality of reference. References exhibit temporal locality if data is requested again that has been recently requested already. References exhibit spatial locality if data is requested that is physically stored close to data that has been requested already. To adapt a varying locality of program some existing techniques dynamically adjust the block size [15] , [16] or replacing policy during the execution.

Operation: In

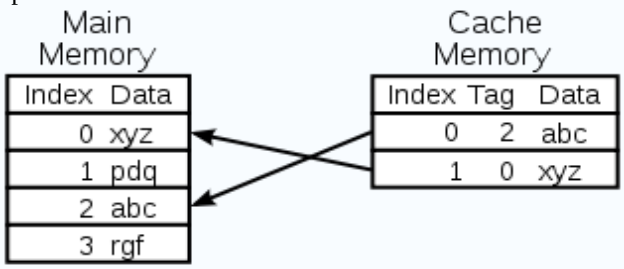


Diagram of a CPU memory cache

Hardware implements cache as a block of memory for temporary storage of data likely to be used again. CPUs and hard drives frequently use a cache, as do web browsers and web servers.

A cache is made up of a pool of entries. sub-blocked [14] cache is beneficial to bandwidth usage. Each entry has a datum (a nugget of data) - a copy of the same datum in some backing store. Each entry also has a tag, which specifies the identity of the datum in the backing store of which the entry is a copy.

When the cache client (a CPU, web browser, operating system) needs to access a datum presumed to exist in the backing store, it first checks the cache. If an entry can be found with a tag matching that of the desired datum, the datum in the entry is used instead. This situation is known as a cache hit. So, for example, a web browser program might check its local cache on disk to see if it has a local copy of the contents of a web page at a particular URL. In this example, the URL is the tag, and the content of the web page is the datum. The percentage of accesses that result in cache hits is known as the hit rate or hit ratio of the cache.

The alternative situation, when the cache is consulted and found not to contain a datum with the desired tag, has become known as a cache miss. The previously uncached datum fetched from the backing store during miss handling is usually copied into the cache, ready for the next access.

III. PROPOSED SYSTEM

The diagram of one of the field programmable elements that facilitate this function in the array. The data-path can be configured so that the output will XOR the two inputs, or simply output *Input 1*. The control-path contains a configuration register which selects the data-path function and is programmed via the *Config Data* input when the *Config Enable* input is set high. The generic architecture of the proposed soft redundancy allocated memory. The numeric values of bit-width used.

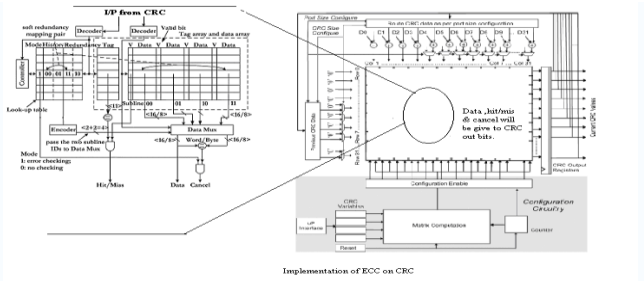


Fig.(1) Cache architecture

In addition, the proposed technique does not increase memory size (e.g., no extra memory cells for redundancy as in -modular redundancy) but exploits the inherent transient redundancy for error protection. Our technique is suitable to a large body of general purpose computing applications. Note that for specific applications, such as multimedia processing, some techniques different from traditional ECC or –modular redundancy for general scenarios have also been developed. One example is reconstruction of the missing macro blocks for error concealment. Different from these techniques, our approach focuses specifically on memory systems (e.g., on-chip cache) which are vulnerable to soft errors. Soft redundancy was initially proposed for on-chip memory design in our past work, where some preliminary results demonstrated joint improvement of error tolerance and access efficiency. In this paper, we extend our past work by making the following contributions. First, a new soft-redundancy allocation mechanism is developed to allow more effective utilization of soft-redundant memory resources. This is achieved by distinguishing two types of cache misses: tag mismatches and subline misses, thereby enabling more accurate judgment on the status of cache lines to detect soft redundancy. Second, a new runtime optimization method is proposed to adaptively utilize the soft-redundant memory for time-varying workloads. This makes the proposed technique more flexible, maintainable, and scalable. Third, a comprehensive suite of simulations is performed with new results covering design optimization, error tolerance, access performance, bandwidth usage, and scalability of the proposed technique. These results demonstrate the significant improvements in error tolerance as well as the reduction in memory miss rate and bandwidth usage over the conventional memory techniques.

Fig. 1 shows the cache architecture allocating soft redundancy for memory access. The numerical values of bit-widths are used in an example described below for the purpose of illustration. In this example, the physical memory address consists of 24 bits. The total cache size is 32 KB with set-associativity of four, and each cache line contains 32 B. Only one way is shown in Fig. 1 as all the four ways are identical. In order to identify and utilize soft redundancy, each cache line is divided into multiple sub lines (e.g., four sub lines in Fig. 1 for the purpose of illustration). Note that the subline size is a key parameter that affects the distribution of soft redundancy,

the coverage of error control, the miss rate, and the bandwidth usage of memory access.

IV. SIMULATION RESULTS

Xilinx edition (MXE) tool will be used for simulation and results will be verified. Fig.6.Xilinx Synthesis technology (XST) will be used for FPGA synthesis. Fig.7. Timing analysis will be carried out to predict the maximum achievable clock speeds for chosen Xilinx Spartan 3E FPGA device. The xpower analysis for estimation of power in the new proposed circuit.

A.Final results:

Release 8.1i - xst I.24
Copyright (c) 1995-2005 Xilinx, Inc. All rights reserved.
--> Parameter TMPDIR set to ./xst/projnav.tmp
CPU : 0.00 / 3.41 s | Elapsed : 0.00 / 3.00 s

--> Reading design: ECC.prj

TABLE OF CONTENTS
1) Synthesis Options Summary
2) HDL Compilation
3) HDL Analysis
4) HDL Synthesis
   4.1) HDL Synthesis Report
5) Advanced HDL Synthesis
   5.1) Advanced HDL Synthesis Report
6) Low Level Synthesis
7) Final Report
   7.1) Device utilization summary
   7.2) TIMING REPORT

Synthesis Options Summary
==========
---- Source Parameters
Input File Name : "ECC.prj"
Input Format : mixed
Ignore Synthesis Constraint File : NO

---- Target Parameters
Output File Name : "ECC"
Output Format : NGC
Target Device : xc3s100e-4-vq100

---- Source Options
Top Module Name : ECC
Automatic Register Balancing : No

---- Target Options
Add IO Buffers : YES
Global Maximum Fanout : 500
Add Generic Clock Buffer(BUFG): 8
Register Duplication : YES

---- General Options
Optimization Goal : Speed
Optimization Effort : 1
Keep Hierarchy : NO
RTL Output : Yes
Global Optimization : AllClockNets
Write Timing Constraints : NO
Hierarchy Separator : /
Bus Delimiter : <>
Case Specifier : maintain
Slice Utilization Ratio : 100
Slice Utilization Ratio Delta : 5

---- Other Options
lso : ECC.lso
Read Cores : YES
cross_clock_analysis : NO
verilog2001 : YES
safe_implementation : No
Optimize Instantiated Primitives : NO
use_clock_enable : Yes
use_sync_set : Yes
use_sync_reset : Yes
Final Report
==========
Final Results
RTL Top Level Output File Name : CRC_top.ngr
Top Level Output File Name : CRC_top
Output Format : NGC
Optimization Goal : Speed
Keep Hierarchy : NO
Design Statistics
IOs : 99

Cell Usage :
BELS : 197
INV : 2
LUT2 : 195
FlipFlops/Latches : 580
FD : 1
FDC : 341
FDP : 238
Clock Buffers : 2
BUFGP : 2
IO Buffers : 49
IBUF : 17
OBUF : 32
==========
Device utilization summary:
==========
Selected Device: 3s400fg320-4

Number of Slices	333 out of	3584	9%
Number of Slice Flip Flops:	531 out of	7168	7%
Number of 4 input LUTs:	195 out of	7168	2%
Number of bonded IOBs:	51 out of	221	23%
IOB Flip Flops: 49			
Number of GCLKs:	2 out of	8	25%

==========
TIMING REPORT
GENERATED AFTER PLACE-and-ROUTE.
Clock Information:

Clock Signal	Clock buffer(FF name)	Load
CLK2	BUFGP	305
CLK1	BUFGP	275

Timing Summary:

Speed Grade: -4

Minimum period: No path found
Minimum input arrival time before clock: 1.825ns
Maximum output required

time after clock: 7.165ns
Maximum combinational path delay: No path found

CPU :13.73/17.88 s|Elapsed : 13.00/17.00 s

-->
Total memory usage is 117044 kilobytes

Number of errors	:	0	(0 filtered)
Number of warnings	:	301	(0 filtered)
Number of infos	:	13	(0 filtered)

B. Simulation of proposed method

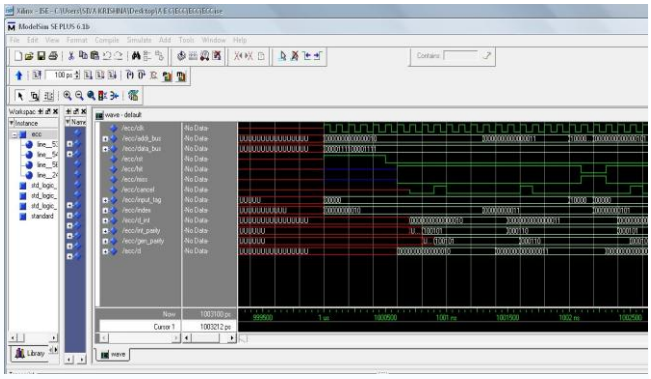
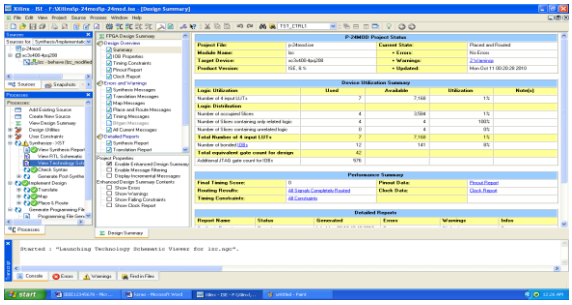


Fig. 3. Simulation waveform.

C. Synthesis summary report;



V. CONCLUSION

This paper presents an error-tolerant technique referred to as the *soft redundancy* for jointly improving memory access efficiency and error tolerance. A resource allocation mechanism is proposed to efficiently utilize the soft redundancy under various memory activities. Design optimization and runtime reconfiguration ensure the flexibility and scalability of the proposed technique under a variety of workloads and program behaviours. Simulation results demonstrate 74.8% average error-control coverage ratio on the SPEC CPU2000 benchmarks with average of 59.5% and 41.3% reduction in memory miss rate and bandwidth usage, respectively, in a 4-way set-associative cache as compared to the existing techniques. Future work is being directed towards exploiting soft redundancy at other architectural levels, e.g, OS, multithreading, and multi-core. When multiple processes are running at the same time, the current scheme may become inadequate to detect and utilize the soft redundancy. In addition, the processes may have different memory requirements, which makes it difficult for the current runtime reconfiguration to satisfy every process.

REFERENCES

1. The International Technology Roadmap for Semiconductors, 2003 at http://public.itrs.net/Files/2003ITRS/Home2003.htm.
2. R. W. Keyes, "Fundamental limits of silicon technology", *Proceedings of the IEEE*, vol. 89, no. 3, pp. 227-239, 2001.
3. A. Mupid, M. Mutyam, N. ijaykrishnan, Y. Xie, and M. J. Irwin,"Variation analysis of CAM cells," *International Symposium on Quality Electronic Design*, pp. 333-338, 2007.
4. A. Agarwal, B. C. Paul, S. Mukhopadhyay, and K. Roy, "Process variation in embedded memories: failure analysis and variation aware architecture,"*IEEE Journal of Solid-State Circuits*, vol. 40, no. 9, pp. 1804-1814, 2005.
5. R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*,vol. 5, no. 3, pp. 305-316, 2005.
6. S. S. Mukherjee, J. Emer, and S. K. Reinhardt, "The soft error problem: an architectural perspective," *International Symposium on High-Performance Computer Architecture*, pp. 243-247, 2005.
7. H. L. Hughes and J. M. Benedetto, "Radiation effects and hardening of MOS technology: devices and circuits," *IEEE Trans. on Nuclear Science*,vol. 50, pp. 500-521, June 2003.
8. Q. Zhou and K. Mohanram, "Gate Sizing to Radiation Harden combinational Logic," *IEEE Trans. on computer-aided design of integrated circuitsand systems*, vol. 25, pp. 155-166, Jan 2006.
9. K. Chakraborty, S. Kulkami, M. Bhattacharya, P. Mazumder, and A. Gupta, "A physical design tool for built-in self-repairable RAMs,"*IEEETrans. on VLSI*, vol. 9, pp. 352-364, April 2001.
10. M. Horiguchi, "Redundancy techniques for high-density DRAMS," *IEEE Innovative Systems Silicon*, pp. 22-29, Oct. 1997.
11. M. Biberstein and T. Etzion, "Optimal codes for single-error correction,double-adjacent-error detection," *IEEE Trans. on Information Theory*, vol.46, pp. 2188-2193, Sept. 2000.
12. T. Suzuki, Y. Yamagami, I. Hatanaka, A. Shibayama, H. Akamatsu, and H. Yamauchi, "A Sub-0.5-V Operating Embedded SRAM Featuring a Multi-Bit-Error-Immune Hidden-ECC Scheme," *IEEE Journal of Solid- State Circuits*, vol. 41, pp. 152-160, Jan. 2006.
13. S. K. Reinhardt and S. S. Mukherjee, "Transient fault detection via simultaneous multithreading," *Proc. International Symposium on Computer Architecture*, pp. 25-36, 2000.
14. R. D. Fellman, R. T. Kaneshiro, and K. Konstantinides, "Design and evaluation of an architecture for a digital signal processor for instrumentation applications," *IEEE Transaction on Acoustics, Speech, and Signal Processing*, vol. 38, no. 3, pp. 537-546, 1990.
15. A. V. Veidenbaum, W. Tang, R. Gupta, A. Nicolau, and X. Ji, "Adapting cache line size to application behavior," *International Conference on Supercomputing*, pp. 145-154, 1999.
16. P. Pujara and A. Aggarwal, "Increasing the cache efficiency by eliminating noise," *International Symposium on High Performance Computer Architecture*, pp. 145-154, 2006.

AUTHORS PROFILE

Ms. CH. Kavya, Student, is currently Pursuing her M.Tech VLSI., in ECE department of Sree Vidyanikethan Engineering College, Tirupati. She has completed graduation in Electronics and Communication Engineering, from Padmavathi mahila university, Tirupathi. Her research areas are VLSI, Digital IC Design, and ASIC Design.

Mr. C. Venkata sudhakar, M.Tech.,is currently working as an Assistant Professor in ECE department of Sree Vidyanikethan Engineering College, Tirupati. He has completed M.Tech in VLSI Design, in JNTUCEH, Hyderabad. His research areas are VLSI Design, &Digital design