



A Scratchpad Memory with Advanced DMA for Data Transfer with Cache Assistance for Multi-bit Error Correction

Asapu Harika¹, Mr. A. Sai Kumar Goud², Mr. Pradeep Kumar Reddy³

¹PG Scholar, ²Associate Professor, ³Associate Professor,
Department of ECE, CMR Institute of Technology,
Kandlakoya, Hyderabad, Telangana, India

ABSTRACT

Modern embedded processors includes both scratchpad memory (SPM) and cache memory in their architectures. SPM's are prone to soft errors like Single event upsets (SEUs) and Single event multiple upsets (SEMUs). For correcting soft errors, we use Error Correcting Codes (ECC) like single-error correction double-error detection (SEC-DED) and SEC-DED double-adjacent error correction (SEC-DED-DAEC) and parity duplication approach. These approaches are used only error correction for less number of corrections and thus increases speed.

This paper proposes a duplication scheme, cache-assisted duplicated SPM (CADS) to correct SEUs and SEMUs in data SPM lines detected by a low-cost error detecting code. This duplication approach does error detection in SPM which is faster than ECC providing multibit error correction.

The extension is to include DMA with ring descriptors. The descriptor-based model provides the most flexibility in managing a system's DMA faster data transfers to or from Scratchpad memory.

Keywords: Cache memory, Scratchpad memory, error correcting codes, soft errors, replicas and DMA.

I. INTRODUCTION

A wide range of modern embedded processors includes both Scratchpad memory (SPM) and cache memory in their architectures to fulfill the application requirements of predictability, performance, and energy budget. Examples of such processors are ARM Cortex-R Series and SH7785, which are used in automotive, industry,

and medical applications. These applications are safety-critical requiring high reliable processors. Scratchpad memory (SPM) is widely used in modern embedded processors to overcome the limitations of cache memory.

Cache memory, also called CPU memory, is random access memory (RAM) that a computer microprocessor can access more quickly than it can access regular RAM. This memory is typically integrated directly with the CPU chip or placed on a separate chip that has a separate bus interconnect with the CPU. The basic purpose of cache memory is to store program instructions that are frequently re-referenced by software during operation. Fast access to these instructions increases the overall speed of the software program.

As the microprocessor processes data, it looks first in the cache memory; if it finds the instructions there, it does not have to do a more time-consuming reading of data from larger memory or other data storage devices.

Most programs use very few resources once they have been opened and operated for a time, mainly because frequently re-referenced instructions tend to be cached. This explains why measurements of system performance in computers with slower processors but larger caches tend to be faster than measurements of system performance in computers with faster processors but more limited cache space.

MEMORY ORGANIZATION:

Computer memories constitute a common unique system for execution of program. Memory devices in computers are utilized for putting away unique types of data, for example, information, programs, addresses, literary records and status data on the processor and other Computer devices. Data put away in memory devices can be separated into bits, bytes, words, pages, and other bigger information structures, which have their own identifiers. In primary memory, data is put away in memory cells or memory areas. Memory areas contain data to which an entrance can occur. To read or write data in a memory location, a single memory access task must be executed, which requires independent control signs to be provided to the memory.

In view of data addressing technique, memory devices can be classified into two:

- In which access to location is controlled by addresses.
- Access to locations is controlled by the memory content.

In the first type memories, accessible locations have a place in which each available a location has its address that can be utilized to choose a location in the memory and execute an operation. These are memories which are addressed, depend on hardware circuits, which do address decoding and select the required location for a memory access operation. The total addresses in a given memory are called an address space of this memory.

In the second type, the selection of a memory location is done by comparing with the contents of the memory. The positive result of comparison gives the readout of the remaining data in that location. Whereas, for write operation, basic data, which will be accessed in the future, an extra information is stored in each location, which will be used for searching the basic data using a comparative method. These memories do not have address decoders.

Data is put in a matrix of the information word locations. The framework can be spoken to as a two-dimensional arrangement of the bit locations. That is the reason this kind of memory is called at some point a 2D primary memory.

To the address decoder inputs, the address bus of a processor is connected. Through the control bus, read/write signal is given by the processor. A buffer register is connected to the external data bus of the

processor at the output of memory cell matrix. It stores the data read from the memory or which is to be written into the memory.

MEMORY HIERARCHY IN COMPUTERS:

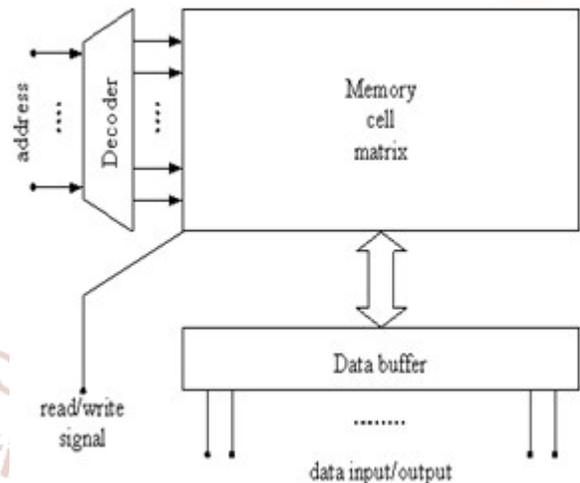


Fig : Memory module of Linear section

The name of this memory originates from a single address decoder utilized as a part of a memory module. Each output line of the decoder chooses (initiates) a word location. The primary addressable location is a series of bit cells corresponding to a word. Accordingly, in this sort of memory, we have a direct task of addresses and output of word locations from the decoder. To every bit cell with the same word, control lines initiate read or write in this area and an output line of the address decoder is provided when the control is given.

MAIN MEMORY WITH TWO-DIMENSIONAL SECTION:

In this memory, the address of the memory is divided into row and column address and the given address can be found at the intersection of row and column address. In this memory module, two address decoders are utilized: one for the column address and another for the row address. In the process of decoding of the address, two lines originating from the decoders are initiated. Bits of all words at a similar position are put away in bit cells put in the same rectangular bit cell matrix called bit plane. The memory module contains the number of planes equal to the Number of bits in the memory word. To every cell in the plane, the lines that are associated are one line from the row decoder, one line from column decoder, read/write control lines. The lines from decoders are recreated the same number of times the same number of bit planes are in the memory module. The read or write in bit cells of a given word happens in the meantime

as the lines from decoder as many times as the bit planes in the memory modules are used.

Each output line from the row and column decoders chooses (enacts) a succession of bit cells that have a place with planes of back to back bits in a memory word. In such memory module, the full network of bit cells has three-dimensional structure.

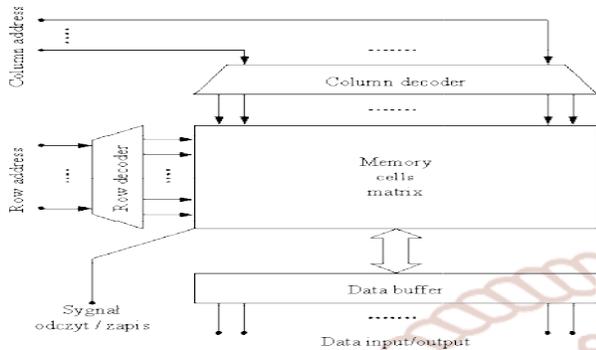


Fig Memory module with a 2D selection of memory locations

FAULT-TOLERANT SCHEMES:

Fault-tolerant schemes in ON-chip memories (SPM or cache) can be ordered into two methodologies. The principal approach is the utilization of error correcting codes (ECCs), e.g., single error correction double error detection (SEC-DED) and SEC-DED double adjacent error correction (SEC-DED-DAEC), to detect and correct errors. The greater part of the ON-chip memories can be secured utilizing this approach. Be that as it may, this approach has two difficult issues: 1) a constrained error correction capacity and 2) an altogether higher overhead, when ECCs are utilized to redress multiple bit errors, for example, SEMUs. The second way to detect and correct errors is a joint utilization of parity code and a duplication of memory sections; we call this approach parity duplication. The principle advantage of this approach is its ability to correct every single detected errors. This approach has been regularly connected to structures, for example, instruction cache, instruction SPM, and write through data cache. In these structures, a duplicate of all entries is intrinsically accessible in lower memory levels and the overheads of memory protection mechanism are as low as the parity code overheads. In any case, the parity duplication approach does not offer full assurance for data SPM and compose back write back data cache since a small amount of information hinders in these structures does not have any duplicate in the lower memory levels for error correction.

SPM Reliability Enhancement

A data duplication plot utilizes dead blocks of SPM to keep a duplicate of live blocks. Dead blocks are distinguished at compiler level utilizing an algorithm that analyzes the examples of getting to SPM lines. At runtime, the redundant blocks are made if a free block exists in SPM. This plan experiences two noteworthy constraints. To begin with, because of high usage of SPM lines and lack of dead blocks, a huge division of SPM blocks stay unprotected. Second, refreshing the imitation on each write operation to SPM will force high execution overhead.

An embedded RAID-ON-chip (E-RoC) scheme ensures ON-chip appropriated SPM modules in chip multiprocessor frameworks. In this component, SPM blocks are copied in the other SPM modules under the control of an E-RoC manager module. A high energy overhead is forced because of parallel access to SPMs and overseeing of the SPM contents. Forceful voltage downscaling is utilized to diminish this overhead, its outcome being the exponential increment in the vulnerability of SPMs to soft errors. Considering distributed SPM in multicore processors, an in-scratchpad memory replication (ISMR) plot copies the dirty SPM blocks into inactive SPM space. In this plan, a offline profiling is performed to examine the entrance examples of the SPM blocks. At runtime, the status of SPM squares is dictated by a label included to all SPM lines and a dirty SPM blocks are duplicated to latent SPM blocks with the guide of a reproduction administration unit.

A memory-mapped SPM (MM-SPM) conspire has been acquainted with secure guideline SPM and isn't appropriate to data SPM. Fault tolerant SPM scheme allotments SPM into three areas with various levels of soft error protection and maps data blocks onto SPM regions as indicated by the weakness of data blocks. An data recompilation calculation recoups the incorrect data block in SPM utilizing its essential data components by re-executing the directions delivering the data block.

Duplication Schemes in Cache

In-cache replication repeats a small amount of dirty lines of data cache into lines that have not been utilized for quite a while. Replication cache depends on keeping a redundant duplicate of dirty cache lines in a small embedded cache. Multicopy cache keeps numerous duplicates for cache lines to identify and correct process-variety actuated

blames in a forcefully voltage-scaled cache design. This plan can essentially lessen the energy utilization of the store in installed applications that their working set is significantly littler than the cache size.

ECC-Based Schemes in Cache:

A Per-Set Protected cache scheme misuses the parallel access of cache lines to apply ECCs in larger data granularity.

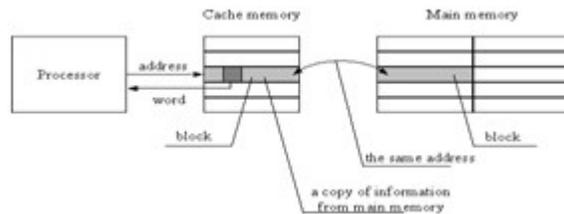


Fig : Implementation in cache memory on Hit

Utilizing an indistinguishable number of bits from SEC-DED, a SEC-DED-DAEC code gives a higher error correction ability. The objective of the matrix based ECC scheme is to give the capacity of remedying adjacent multiple errors. The coding plan proposed can rectify double random errors and also burst errors of length 3 and 4 b. The ECC scheme has the capacity of double adjacent error correction and also adjacent error detection. The objective of the plan is to maximize the probability of detecting double adjacent bit errors in SEC and triple adjacent bit errors in SEC-DED. A modified version of Hamming code gives the capacity to identify 2-and 3-b burst errors including correction of single bit errors.

We finish up this segment by featuring the primary contrasts between our proposed scheme and previous data duplication schemes. Every single past data SPM duplication plans keep the replicas in SPM, though we propose to keep the imitations in the cache. Keeping the imitation in SPM requires a complicated equipment module to deal with the replicas or complex application profiling and SPM administration change. Additionally, it forces a noteworthy performance overhead because of diminishment in the SPM usable space. Then again, because of structural contrast among SPM and cache memory, duplication plans for cache protection either are not applicable to SPM or force critical overheads to give replicas to all SPM lines.

PROPOSED SCHEME:

In cache-assisted duplicated SPM(CADS) to correct SEUs and SEMUs in data SPM lines detected by a low-cost error detecting code. This duplication approach does error detection in SPM which is faster than ECC providing multi-bit error correction.

The extension is to include DMA with ring descriptors. The descriptor-based model provides the most flexibility in managing a system's DMA faster data transfers to or from Scratchpad memory.

CACHE MEMORY

A cache memory is a quick arbitrary access memory where the computer stores data that is currently utilized by programs (data and instruction), stacked from the primary memory. The cache has shorter access time than the main memory due to the costly implementation technology. The cache has a restricted volume because of the properties of the applied technology. The data to the cache memory is utilized more than, the access time to it will be much shorter than for the situation if this data were retrieved in the primary memory and the program will executed at a higher speed.

Time efficiency of utilizing cache comes of the locality of access to data that is seen amid program execution. We see here time and space locality:

Time locality comprises to utilize similar instructions and data in programs amid neighbouring time interims as many times.

Space Locality is a tendency to store instructions and data utilized as a part of a program in short separations of time under neighbouring locations in the main memory.

Memory Updating Methods After Cache Modifications:

A cache memory contains duplicates of information in the main memory. When a difference in information in a cache happens, then there is a change in main memory and cache memory cells i.e., the data is changed. Following two data coherency strategies are applied resulting the change.

- **Write through:** In this strategy, new cache contents is written down to the main memory immediately after the write to the cache memory,
- **Write back:** In this strategy when the given block of data is replaced with new block fetched from main memory or upper level cache then the new cache contents are written down to the main memory immediately. Only state bits are changed in the modified block, indicating that the block has been modified (a dirty block) after data has been written into the cache. The

write back updating takes more time efficient, as the block cells modification is done many times while being in the cache and are updated in the main memory only once.

SCRATCHPAD MEMORY:

Scratchpad memory (SPM) is widely used in modern embedded processors to overcome the limitations of cache memory.

Scratchpad memory (SPM), also known as scratchpad, scratchpad RAM or local store in computer terminology, is a high-speed internal memory used for temporary storage of calculations, data, and other work in progress. In reference to a microprocessor ("CPU"), scratchpad refers to a special high-speed memory circuit used to hold small items of data for rapid retrieval.

In contrast to a system that uses caches, a system with scratchpads is a system with Non-Uniform Memory Access latencies, because the memory access latencies to the different scratchpads and the main memory vary. Another difference from a system that employs caches is that a scratchpad commonly does not contain a copy of data that is also stored in the main memory.

Scratchpads are employed for simplification of caching logic, and to guarantee a unit can work without main memory contention in a system employing multiple processors, especially in multiprocessor system-on-chip for embedded systems. They are mostly suited for storing temporary results (as it would be found in the CPU stack) that typically wouldn't need to always be committing to the main memory; however when fed by DMA, they can also be used in place of a cache for mirroring the state of slower main memory. The same issues of locality of reference apply in relation to efficiency of use; although some systems allow strided DMA to access rectangular data sets. Another difference is that scratchpads are explicitly manipulated by applications. They may be useful for real-time applications, where predictable timing is hindered by cache behavior.

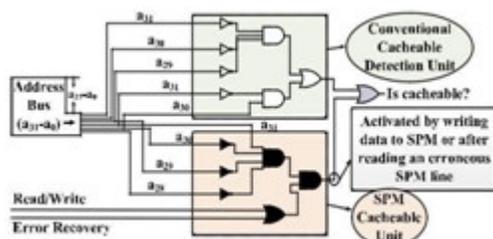


Fig. Proposed cacheable detection unit to support SPM write access caching.

To make the SPM write access as cacheable activities and to keep SPM read access as non-cacheable tasks, CADS upgrades the cacheable detection unit of the cache controller. The traditional cacheable detection unit chooses whether the address created by the processor is cacheable or not. In CADS, the read/write signal is additionally checked by the cacheable detection unit. The read/write signal demonstrates whether the write access and the address produced by the processor are in the SPM area, the cacheable detection unit enacts the cacheable signal.

It shows the proposed design and a theoretical perspective of the modified cacheable detection unit. To characterize SPM write access as cacheable tasks, a SPM cacheable unit (SCU) is added to the regular cacheable detection unit. The SCU initiates its yield if the address is inside the SPM area and the write signal is active. The yield of the ordinary cacheable detection unit and the SCU are ORed to deliver the final cacheable signal. The SCU has input signal, called error recovery signal, to influence SPM to write through as cacheable task in the SPM error recovery stage.

CADS:

The high vulnerability of SPM to soft errors, however, limits its usage in safety-critical applications. An efficient fault-tolerant scheme, called cache-assisted duplicated SPM (CADS), to protect SPM against soft errors. The main aim of CADS is to utilize cache memory to provide a replica for SPM lines. Using cache memory, CADS is able to guarantee full duplication of all SPM lines. We also further enhance the proposed scheme by presenting buffered CADS (BCADS) that significantly improves the CADS energy efficiency. BCADS is compared with two well-known duplication schemes as well as single-error correction scheme.

One major source of system failure in such applications is soft errors caused by radiation-induced particle strike into chips. Single-event upsets (SEUs) and single-event multiple upsets (SEMUs) are two types of soft errors in SPM and cache as ON-chip SRAM memories. Correcting soft errors in ON-chip memories (SPM or cache) can be categorized into two approaches. The first approach is the use of error-correcting codes (ECCs), e.g., single-error correction double-error detection (SEC-DED) and SEC-DED double-adjacent-error correction (SEC-DED-DAEC), to detect and correct

errors. All of the ON-chip memories can be protected using this approach.

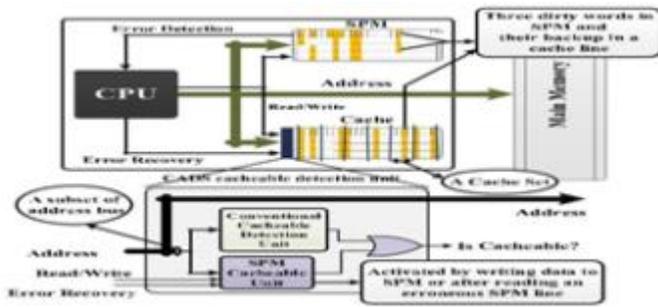


Fig. Proposed CADS architecture

However, this approach has two serious problems: 1) a limited error correction capability and 2) a significantly higher overhead, when ECCs are employed to correct multiple bit errors such as SEMUs. The second approach to detect and correct soft errors is a joint use of parity code and a duplication of memory entries; we call this approach parity duplication. The main advantage of this approach is its capability to correct all detected errors. This approach has been commonly applied to structures such as instruction cache, instruction SPM, and write-through data cache.

ERROR CORRECTING CODES:

Any error-correcting code can be used for error detection. A code with minimum Hamming distance, d , can detect up to $d - 1$ errors in a code word. Using minimum-distance-based error-correcting codes for error detection can be suitable if a strict limit on the minimum number of errors to be detected is desired.

Codes with minimum Hamming distance $d = 2$ are degenerate cases of error-correcting codes, and can be used to detect single errors. The parity bit is an example of a single-error-detecting code.

By proposing a duplication scheme, so-called cache-assisted duplicated SPM (CADS), to correct SEUs and SEMUs in data SPM lines detected by a low-cost error detecting code. The key idea in CADS to provide a replica for software-managed SPM is enforcing the hardware-managed cache to keep a copy of non-cacheable SPM lines. In particular, CADS duplicates all dirty SPM lines in cache memory considering the fact that clean SPM lines have inherently a copy in lower memory hierarchy. To this aim, we propose a cache controller circuitry that is capable of storing a copy of non-cacheable SPM lines in cache memory. To reduce the energy consumption overhead of CADS, we further propose a buffered CADS (BCADS) technique in which a mini buffer is inserted between SPM and cache to

minimize the extra cache accesses for updating the replicas.

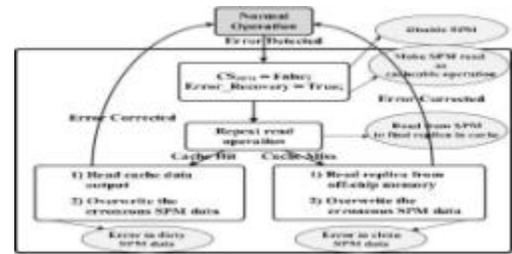


Fig. Error correction in CADS

Cacheable Table For System Addressable Area:

Address Range	Area	Cached
0x00000000-0x1FFFFFFF	PROM	Cacheable
0x20000000-0x3FFFFFFF	I/O	Non-Cacheable
0x40000000-0x7FFFFFFF	RAM	Cacheable
0x8E000000-0x8FFFFFFF	SPM	Non-Cacheable

This shows the point by point design of the modified cacheable detection unit for addressing areas of a regular processor with the end goal that its cacheable table is as indicated by Table I. As appeared in Table I, PROM and RAM address areas are cacheable, though I/O and SPM address regions are non-cacheable accesses.

Design Optimization:

To reproduce all dirty SPM lines in the cache, each write access to the SPM needs an extra access to the store. Additional cache access are forced in light of refreshing the replicas in the cache or dispensing new cache lines for the allocation of SPM lines on the off chance that they are not as of now accessible. This prompts a critical dynamic energy overhead in CADS. The energy overhead of additional access memory for allocating and updating the copies exists for all past duplication plans. Our assessments demonstrate that the dynamic energy overheads for updating the replica in the plan in and CADS are 52% and 43%, separately.

To beat this overhead and decrease the cache accessing because of SPM line duplication, we improve the CADS architecture by inserting a buffer between the SPM and the cache. This improved architecture is named BCADS. This

architecture lessens the cache access by putting away most as of late updated SPM line. The span of the support is viewed as the same as the extent of a solitary cache line, e.g., 32 B in our arrangement. Utilizing this buffer, we can keep the replica of eight 32-b SPM words dwelling in a similar cache line.

For every write operation to the SPM, the write address is compared with the address of data line stored in the buffer. If the replica is already in the buffer, which can be interpreted as a buffer hit, the buffer entry will be updated. Otherwise, on a buffer miss, after writing back the buffer entry to the cache, the buffer will be allocated to the new data written to the SPM. For the write-back data line from the buffer to the cache, if the replica line is already allocated in the cache, the line will be rewritten. Otherwise, a new cache line is selected for replication before writing back the buffer entry.

BCADS can significantly reduce the extra cache accesses by utilizing the locality of references in SPM access. The SPM lines are mainly allocated to data arrays accessed in loops, which have a highly localized access pattern. Therefore, it is highly probable that consecutive requests for updating the replicas refer to the same cache line. By taking advantage of such a locality, the buffer will be able to catch the majority of replica update requests.

DMA:

A direct memory access (DMA) is an operation in which data is copied (transported) from one resource to another resource in a computer system without the involvement of the CPU. The task of a DMA-controller (DMAC) is to execute the copy operation of data from one resource location to another. The copy of data can be performed from: - I/O-device to memory - memory to I/O-device - memory to memory - I/O-device to I/O-device A DMAC is an independent (from CPU) resource of a computer system added for the concurrent execution of DMA-operations. The first two operation modes are 'read from' and 'write to' transfers of an I/O-device to the main memory, which are the common operation of a DMA-controller. The othertwo operations are slightly more difficult to implement and most DMA-controllers do not implement device to device transfers.

The DMAC replaces the CPU for the transfer task of data from the I/O-device to the main memory (or vice versa) which otherwise would have been executed by the CPU using the programmed input output (PIO) mode. PIO is realized by a small instruction sequence executed by the processor to copy data. The 'memcpy' function supplied by the system is such a PIO

operation. The DMAC is a master/slave resource on the system bus, because it must supply the addresses for the resources being involved in a DMA transfer. It requests the bus whenever a data value is available for transport, which is signaled from the device by the REQ signal. The functional unit DMAC may be integrated into other functional units in a computer system, e.g. the memory controller, the south bridge, or directly into an I/O-device. simplified logical structure of a system with DMA Arbiter CPU Memory DMA controller I/O device.

SIMULATION AND SYNTHESIS RESULTS:

In this section, we show the simulation results of cache memory, cache controller, Scratchpad memory, DMA. Those simulations can be shown in below figures.

SIMULATION OF CACHE MEMORY:

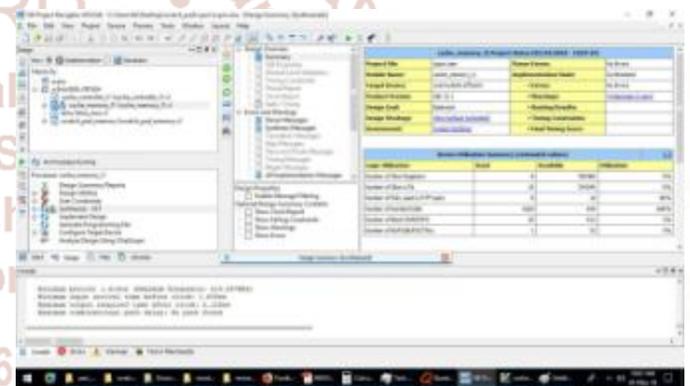


Fig : Simulation of Cache Memory

The Above figure shows that the number of elements that are to used in the process of simulation of Cache memory.

Inputs And Outputs For Synthesis Of Cache Memory:

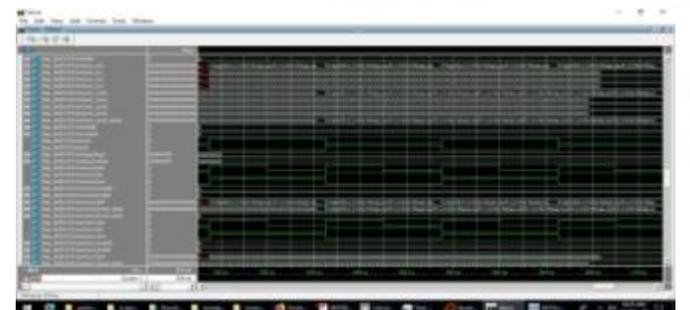


Fig: Synthesis of Cache Memory

The figure indicates the number of pins are used as inputs and output pins are as follows, clk,rst,enable,wr_rdb,addr,way_in0,way_in1,way_in2, way_in3, raddr,waddr .

The output pins are: way_out0,way_out1,way_out2,way_out3.

Simulation of Cache Controller:

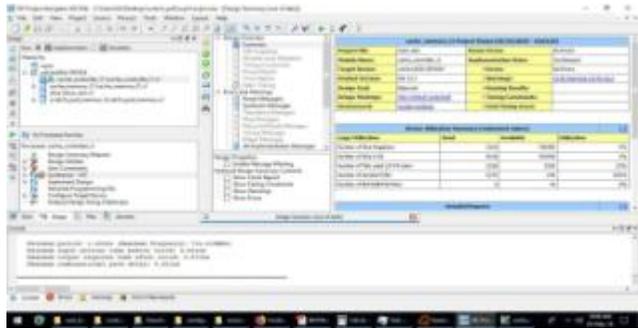


Fig ,.:Simulation of Cache Controller

The above figure shows that the design summary for the cache controller which are we used in the process.

Number of Slice Registers:	2160
Number of Slice LUTs:	9318
Number of fully used LUT-FF pairs:	2158
Number of Bonded IOBs:	5375

Synthesis of Cache Controller:

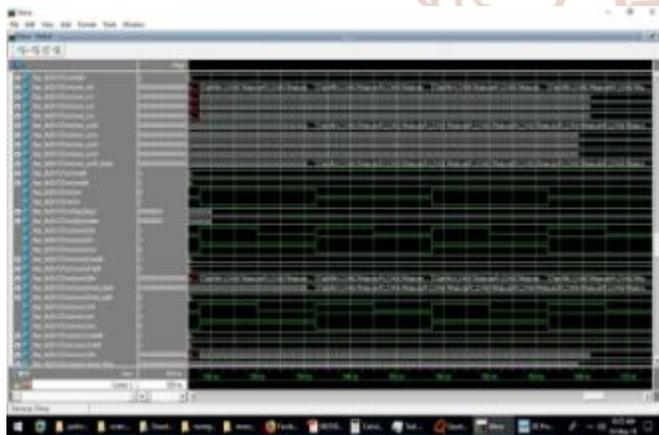


Fig : Synthesis of Cache Controller

The input pins are:

clk,rst,wr_rdb_from_higher_level,en_from_higher_level,wdata_from_higher_level, addr_from_higher_level,rdata_from_lower_level,rd ata_valid_from_lower_level

rdata_to_higher_level,request_done_to_higher_level,hit,miss,cache_rdata_valid.

The output pins are: wr_rdb_to_lower_level,en_to_lower_level,wdata_to_lower_level,addr_to_lower_level,cache_wr_rdb,cache_en,cache_addr,cache_wdata0,cache_wdata1,cache_wdata2,cache_wdata3,cache_rdata0,cache_rdata1,cache_rdata2,cache_rdata3.

These are the parameters cache_size,cache_line_size,num_ways,num_cache_lines,num_sets>tag_bits, index_bits,lru_bits,offset_bits, memory_line_size.

In idle state,no read or write operation is done, else read or write operation is done.If en_from_higher_level is done,then read operation is accessed.

The address from higher level should match with the request from the processor. If it is matched,it is a hit,or else miss.

For read operation, the address from the request should match with the memory address.For write operation, the address request should match with the cache address, so that the address is copied.

Simulation of Scratchpad memory:



Fig :Simulation of Scratchpad

The input pins are clk,en,wr_rdb,addr,wdata. The output pins are rdata,error_present.We use parameters like address bits,data width,parity bits,memory depth.

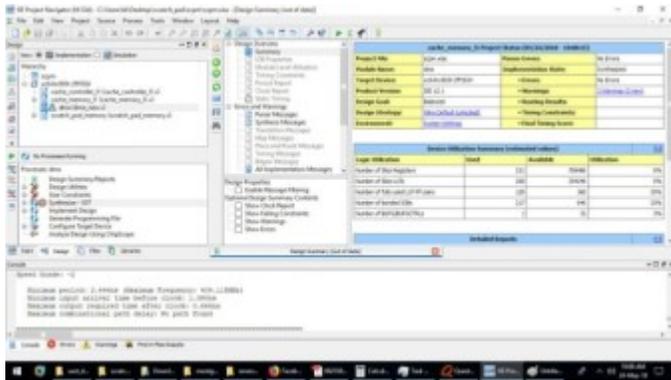
Simulation of DMA:

Fig:Simulation of DMA

IN this DMA simulation,we use

Number of Slice registers:	231
Number of Slice LUTs:	258
Number of fully LUT_FF pairs:	129
Number of Bonded IOBs:	217

Synthesis of DMA:

Fig : Synthesis of DMA

The Above Figure Shows that Synthesis of DMA controller .

The input pins are:

clk,rst,program_dma,reg_addr,reg_data,mem_read,mem_write,mem_addr,mem_wdata,mem_rdata,mem_rdata_valid,

The output pins are:

dma_done,spm_write,spm_addr,spm_read,spm_wdata,spm_rdata.

For example, source address is 32 bit data and destination address is 32 bit data,transfer length is 8 bits.

If source address and destination address completely matched,then read or write operation is done.If both addresses matches,dma_done signal is enabled.

Advantages:

The Cache assisted duplicated SPM is the best method to resolve soft errors in SPMs with very little area over head and performance overheads and no change in SPM access policies.

Applications:

In embedded processors for use in applications in IOT and in DSP processors where processing happens on large arrays of data.

Future scope:

Implement a Robust ECC scheme using less number of redundant bits to detect more number of errors.

References:

- 1) (2015). *The ARM Corporation*. [Online]. Available:<http://www.arm.com/products/processors/cortex-r>
- 2) (2015). *The Renesas Electronics Corporation*. [Online]. Available:<http://www.renesas.com/products/mpumcu/superh/sh7780/sh7785/index.jsp>
- 3) W. Zhang, S. Gurumurthi, M. Kandemir, and A. Sivasubramaniam, "ICR: In-cache replication for enhancing data cache reliability," in *Proc. Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2003, pp. 291–300.
- 4) M. Manoochchri, M. Annavaram, and M. Dubois, "CPPC: Correctable parity protected cache," in *Proc. 38th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2011, pp. 223–234.
- 5) L. A. D. Bathen and N. D. Dutt, "Embedded RAID-on-chip for busbased chip-multiprocessors," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 4, pp. 83:1–83:36, 2014.
- 6) A. Dixit and A. Wood, "The impact of new technology on soft error rates," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Apr. 2011, pp. 5B.4.1–5B.4.7.

- 7) N. N. Sadler and D. J. Sorin, "Choosing an error protection scheme for a microprocessor's L1 data cache," in *Proc. Int. Conf. Comput. Design (ICCD)*, Oct. 2007, pp. 499–505.
- 8) J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe, "Multi-bit error tolerant caches using two-dimensional error coding," in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Dec. 2007, pp. 197–209.
- 9) H. Farbeh and S. G. Miremadi, "PSP-cache: A low-cost fault-tolerant cache memory architecture," in *Proc. Design, Autom. Test Eur. (DATE)*, Mar. 2014, pp. 1–4.
- 10) A. Neale and M. Sachdev, "A new SEC-DED error correction code subclass for adjacent MBU tolerance in embedded memory," *IEEE Trans. Device Mater. Rel.*, vol. 13, no. 1, pp. 223–230, Mar. 2013.
- 11) A. Sánchez-Macián, P. Reviriego, and J. A. Maestro, "Enhanced detection of double and triple adjacent errors in Hamming codes through selective bit placement," *IEEE Trans. Device Mater. Rel.*, vol. 12, no. 2, pp. 357–362, Jun. 2012.
- 12) F. Li, G. Chen, M. Kandemir, and I. Kolcu, "Improving scratch-pad memory reliability through compiler-guided data block duplication," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2005, pp. 1002–1005.
- 13) W. Zhang, "Replication cache: A small fully associative cache to improve data cache reliability," *IEEE Trans. Comput.*, vol. 54, no. 12, pp. 1547–1555, Dec. 2005.
- 14) A. M. H. Monazzah, H. Farbeh, S. G. Miremadi, M. Fazeli, and H. Asadi, "FTSPM: A fault-tolerant scratchpad memory," in *Proc. 43rd Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2013 pp. 1–10.
- 15) A. Dutta and N. A. Toubia, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in *Proc. 25th IEEE VLSI Test Symp. (VTS)*, May 2007, pp. 349–354.
- 16) J. Hong, J. Kim, and S. Kim, "Exploiting same tag bits to improve the reliability of the cache memories," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 2, pp. 254–265, Feb. 2015.
- 17) S. Wang, J. Hu, and S. G. Ziavras, "Replicating tag entries for reliability enhancement in cache tag arrays," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 4, pp. 643–654, Apr. 2012.
- 18) J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, 2006.
- 19) M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *Proc. IEEE Int. Workshop Workload Characterization*, Dec. 2001, pp. 3–14.
- 20) H. Farbeh, M. Fazeli, F. Khosravi, and S. G. Miremadi, "Memory mapped SPM: Protecting instruction scratchpad memory in embedded systems against soft errors," in *Proc. 9th Eur. Dependable Comput. Conf. (EDCC)*, May 2012, pp. 218–226.
- 21) H. Sayadi, H. Farbeh, A. M. H. Monazzah, and S. G. Miremadi, "A data recomputation approach for reliability improvement of scratchpad memory in embedded systems," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2014, pp. 228–233.
- 22) P. R. Panda, N. D. Dutt, and A. Nicolau, "Efficient utilization of scratchpad memory in embedded processor applications," in *Proc. Eur. Design Test Conf. (ED&TC)*, Mar. 1997, pp. 7–11.
- 23) *LEON2 Processor User's Manual, Version 1.0.30*, XST ed., Aeroflex Gaisler, Gothenburg, Sweden, Jul. 2005.
- 24) J. Lee *et al.*, "FaCSim: A fast and cycle-accurate architecture simulator for embedded systems," in *Proc. Conf. Lang., Compil., Tools Embedded Syst. (LCTES)*, Jun. 2008, pp. 89–100.
- 25) C. A. Argyrides, P. Reviriego, D. K. Pradhan, and J. A. Maestro, "Matrix-based codes for adjacent error correction," *IEEE Trans. Nucl. Sci.*, vol. 57, no. 4, pp. 2106–2111, Aug. 2010.
- 26) N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A tool to model large caches," HP Inc., Palo Alto, CA, USA, Tec. Rep. HPL-2009-85, 2009.
- 27) L. Li, V. Degalahal, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Soft error and energy consumption interactions: A data cache perspective," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2004, pp. 132–137.

- 28) Design compiler user guide, Ver. E-2010.12, Synopsys Inc., Mountain View, CA, USA, Dec. 2010.
- 29) J. Suh, M. M. Annavaram, and M. Dubois, "MACAU: A Markov model for reliability evaluations of caches under single-bit and multi-bit upsets," in *Proc. IEEE Int. Sym. High Perform. Comput. Archit. (HPCA)*, Feb. 2012, pp. 1–12.
- 30) D. H. Yoon and M. Erez, "Memory mapped ECC: Low-cost error protection for last level caches," in *Proc. Int. Symp. Comput. Archit. (ISCA)*, 2009, pp. 116–127.
- 31) D. H. Yoon and M. Erez, "Flexible cache error protection using an ECC FIFO," in *Proc. IEEE/ACM Int. Conf. High Perform. Comput. Netw., Storage, Anal. (SC)*, Nov. 2009, pp. 1–12. 65.
- 32) W. Ma, X. Cui, and C.-L. Lee, "Enhanced error correction against multiple-bit-upset based on BCH code for SRAM," in *Proc. IEEE 10th Int. Conf. ASIC (ASICON)*, Oct. 2013, pp. 1–4.
- 33) L.-J. Saiz-Adalid, P. Gil, J.-C. Baraza-Calvo, J.-C. Ruiz, D. Gil-Tomás, and J. Gracia-Morán, "Modified Hamming codes to enhance short burst error detection in semiconductor memories (short paper)," in *Proc. 10th Eur. Dependable Comput. Conf. (EDCC)*, May 2014, pp. 62–65.
- 34) L. Delshadtehrani, H. Farbeh, and S. G. Miremadi, "In-scratchpad memory replication: Protecting scratchpad memories in multicore embedded systems against soft errors," *ACM Trans. Design Autom. Electron. Syst.*, vol. 20, no. 4, pp. 61:1–61:28, 2015.
- 35) A. Chakraborty, H. Homayoun, A. Khajeh, N. Dutt, A. Eltawil, and F. Kurdahi, "E < MC2: Less energy through multi-copy cache," in *Proc. Int. Conf. Compil., Archit. Synth. Embedded Syst. (CASES)*, Oct. 2010, pp. 237–246.
- 36) S. Kang and A. G. Dean, "Leveraging both data cache and scratchpad memory through synergetic data allocation," in *Proc. IEEE 18th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2012, pp. 119–128.
- 37) G. Wang, L. Ju, Z. Jia, and X. Li, "Data allocation for embedded systems with hybrid on-chip scratchpad and caches," in *Proc. IEEE 10th Int. Conf. High Perform. Comput. Commun. (HPCC)*, Nov. 2013, pp. 366–373.
- 38) L. Wu, Y. Ding, and W. Zhang, "Characterizing energy consumption of real-time and media benchmarks on hybrid SPM-caches," in *Proc. IEEE 11th Int. Conf. High Perform. Comput. Commun. (HPCC)*, Aug. 2014, pp. 526–533.
- 39) W. Zhang and L. Wu, "Exploiting hybrid SPM-cache architectures to reduce energy consumption for embedded computing," in *Proc. IEEE Int. Conf. High Perform. Comput. Commun. (HPCC)*, Aug. 2014, pp. 340–347.
- 40) W. Zhang and Y. Ding, "Hybrid SPM-cache architectures to achieve high time predictability and performance," in *Proc. IEEE 24th Int. Conf. Appl.-Specific Syst., Archit. Process. (ASAP)*, Jun. 2013, pp. 297–304.

Authors profile:



ASAPU HARIKA Received his bachelor's degree in 2015 in electronics and communication engineering from Stanley College of Engineering and Technology for Women, Abids, Hyderabad, India, Which is affiliated to Osmania University, India. Her areas of interest include VLSI design. She is pursuing his M-Tech in VLSI SYSTEM DESIGN from CMR Institute of Technology.



Mr. A Sai Kumar Goud is working as Associate professor in ECE department of CMRIT. He has an academic experience of 16 years In Teaching as well as research. He worked in various Reputed Engineering colleges as Assistant professor, Associate professor HOD, of ECE. He has qualified for UGC-NET. He is ratified as Associate professor by JNTUH. His areas of research and interest are VISI, Communications & signal processing. He is a life member of ISTE and published several National and international journals.



Mr. S Pradeep Kumar Reddy B.E, M.Tech (Ph.D), Working as Associate Professor in CMR institute of technology. And has 15 Years of Experience in teaching field. His area of interest is Image processing.